T.C.
TRAKYA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

ŞİFRELEME TEKNİKLERİ VE GÜNCEL UYGULAMA OLANAKLARI

Tarık YERLİKAYA
Yüksek Lisans Tezi
Bilgisayar Mühendisliği Anabilim Dalı

Danışman: Yrd. Doç. Dr. Ercan BULUŞ

EDİRNE-2002

T.C.
TRAKYA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

ŞİFRELEME TEKNİKLERİ VE GÜNCEL UYGULAMA OLANAKLARI

Tarık YERLİKAYA

YÜKSEK LİSANS TEZİ

BİLGİSAYAR MÜHENSDİSLİĞİ ANA BİLİM DALI

Bu tez 20/08/2002 tarihinde aşağıdaki jüri tarafından kabul edilmiştir.

Yrd. Doç. Dr. Ercan BULUŞ (Danışman)

Yrd. Doç. Dr. Erdem UÇAR

Yrd. Doç. Dr. Şaban AKTAŞ

ÖZET

Günümüzde, ülkeler arasında, teknoloji ve güvenli bir hayatın sağlanması açısından rekabetin artması nedeniyle, bilginin aktarılmasının gizlilik ve güvenirliği önem kazanmıştır. Geliştirilen hızlı bilgisayarlar ve güçlü sistemler sayesinde veriler, oldukça hızlı ve iyi şifrelenerek iletilebilmektedir. Verilerin hızlı bir şekilde iletilmesi kadar, kullanılan şifreleme algoritmalarının güvenirliğinin test edilmesi de çok büyük önem taşımaktadır.

Şifrelenmiş mesajların kırılması ve şifreleme algoritmalarının test edilmesi için bir çok kullanılır. Şifrelemenin hızlı ve güçlü olmasını sağlayan, ileri teknoloji ve üstün hesaplama gücüne sahip bilgisayarlar, şifrelerin kırılması işlemlerini de kolaylaştırırlar. Elde edilen bilgi miktarına, anahtarın geçerlilik süresine ve şifrelemede kullanılan algoritmaya göre, şifrelerin kırılması için kullanılan yöntem değişir.

Tezin tamamı 4 bölümden oluşmaktadır.İlk bölümde şifreleme algoritmaları üzerine yapılan ilk çalışmalar ve şifrelemenin tarihçesi açıklanmaktadır.

İkinci bölümde günümüzde kullanılan şifreleme algoritmaları ve sayısal örnekleri anlatılmıştır.

Üçüncü bölümde bu algoritmaların Delphi 6.0' da yazılmış programların açıklanmaları yer almaktadır.

Son bölümde sonuçlar yer almaktadır.

ABSTRACT

Today, rivalry among countries in providing techonology and a secure life is increasing. So secrecy and reliability in transmission of information have become more important. Data can be transmitted very quickly and well encripted thanks to high speed computers and powerful systems. To test the reliability of encription algorithms is as crucial as quick data tranmission.

Many methods are used to decript the encripted data and to test encription algorithms. Computers that have high technology and a high computing capability not only provide quick and powerful encription but also they make it easier to break the code. The method used for breaking the code changes depending on the amount of the received information, the validity period of the key and the algorithm used for encription.

The dissertation consists of four chapters. In the first chapter, the firt studies on cryptographic algorithms and the history of cryptography are explained.

The second chapter describes the cryptographic algorithms and numeric examples that are used today.

The third chapter is devoted to the explanation of the programmes of these algorithms that are written in Delphi 6.0

The final chapter involves the conclusions obtained as a result of this study

TEŞEKKÜR

Bu çalışmanın hazırlanmasında bana yol gösteren, destek ve yardımlarını benden esirgemeyen danışman hocam Yrd. Doç. Dr. Ercan BULUŞ'a, tüm çalışmalarımda bana yardımcı olan değerli hocalarıma, çalışma arkadaşlarıma ve aileme teşekkürlerimi sunarım.

1. GİRİŞ

Kriptoloji, Yunanca krypto's (saklı) ve lo'gos (kelime) kelimelerinin birleştirilmesinden oluşturulmuştur ve iletişimde gizlilik bilimi olarak değerlendirilmektedir.

Ticari ilişkilerde, devlet işlerinde, askeri işlerde ve personel ilişkilerinde güvenli iş çalışması yapmak büyük bir sorundur.

Sistemler arası bağlantılarda ya da herhangi iki nokta arasındaki haberleşmede verinin güvenli bir şekilde gittiğinden emin olmak gerekir. Bunun sağlanması ise gönderilen verinin şifrelenmesi ile olur. Böylece açık haberleşme kanalları kullanılarak verinin güvenli bir şekilde ulaştırılması sağlanır. İletişimde, açık bir haberleşme kanalı kullanılıyorsa gizli tutulmak istenen bilginin yetkisiz bir kişi tarafından dinlenebileceği veya haberleşme kanalına girip (araya girme) veriyi bozabileceği ya da değiştirebileceği (yanlış verinin gönderilmesi) düşüncesi her zaman için önemli bir problem oluşturur.

Kriptoloji esas olarak iki bölüme ayrılır : Kriptografi (şifreleme) ve kriptoanaliz (şifre çözme). Gönderilmek istenen orijinal mesaj açık mesaj (plain text) ve bu mesajın şifrelenmiş hali şifreli mesaj (cipher text-cryptograph) olarak adlandırılır. Şifreleme, askeri ve diplomatik iletişimde (haberleşmede) güvenliği sağlamak için bin yıldır kullanılmaktadır. Ancak bugün artık özel sektörde de gereksinim duyulmaktadır. Sağlık hizmetleri, finansal işler (örneğin: kredi oranları) gibi konularda bilgisayarlar arasındaki haberleşmede açık kanallar kullanılarak yapılmaktadır. Bu açık kanalların kullanılması sırasında yukarıda sayılan işlerin güvenli ve gizli bir şekilde yapılabilmesi için şifrelemeye gerek duyulmaktadır.

1.1 Kriptoloji Tarihine Genel Bir Bakış

Kriptoloji tarihi uzmanı *Kahn*'a göre çağdaş kriptoloji geçmişteki Kriptolojiden üç temel nitelikle ayrılır. Bunlar sırasıyla:

- Sistemlerdeki karmaşıklık artışı,
- Kullanım yaygınlığı,
- İşlemlerdeki niceselleşme artışı veya matematikleşmedir.

Tarihsel sürecin incelenmesi; sistemlerin giderek karmaşıklaşmasını ortaya koyan en ideal yaklaşım olması açısından, kısa kronolojik gelişmeler aşağıda sunulmuştur.

Tarihi insanlık tarihi ile eşyaşlı olan kriptoloji biliminde devrimsel nitelikli ilk değişim; öncelikle elektrik ve daha sonra giderek elektronik ve bilgisayarların bu alanda kullanımıyla ortaya çıkıştır. Anılan tüm bu gelişmelerin 1939-1940 yıllarından itibaren olması nedeniyle, II. Dünya savaşı başlangıcı temel kilometre taşlarından ilki olarak seçilmiş ve 1940 öncesi yıllar bilimsellikten uzak "kağıt-kalem" kriptolojisi yılları olarak adandırılmıştır.

1.1.1 1940 Öncesi Kriptoloji Dönemi

Bu dönem; sırasıyla eski Yunan ve Romalılar dönemi, Orta Çağ Dönemi, Rönesans Dönemi ve 20. Yüzyıl ilk yarısı olarak dört ana başlık altında irdelenebilir.

Eski Yunan ve Romalılar döneminde M.Ö. 400 yıllarında Spartalıların kullandığı *Scytale* şifresi ve M.Ö. 50 yıllarında Sezar' ın geliştirdiği Yerinekoymalı Sezar şifresi en temel şifreleri oluşturmuştur.

Orta Çağda Parma'lı *Gabriele de Lavinde* (1300 'lü yıllar) şifreler üzerine ilk el kitabını yayımlamıştır.

Rönesans; diğer sanat ve bilim dallarında olduğu gibi Kriptolojide de yeni yaklaşımlara yol açmış ve bu dönemde *Polialfabetik* şifreler ortaya çıkmıştır. Kriptoloji alanında "Polygraphia" adlı ilk kitap bu dönemde yayımlanmış olup, *Johannes Trithemius*'a (1462-1516) aittir.

20. Yüzyılın ilk yarısında dikkati çeken temel olaylar; 1917 yılında Amerika Birleşik Devletlerinin I. Dünya savaşına girmesine yol açan *Zimmermann* telgraf şifresi olayı ve 1926' da *Vernam* tarafından yayımlanan , gerçek anlamda kırılmayacak tek simetrik şifre olan Tek Kullanımlık Şifre (one-time-pad) algoritmasıdır.

1.1.2 1940 Sonrası Çağdaş Kriptoloji

Bugünkü anlamlarıyla çağdaş kripto cihazları, elektriğin bu cihazlara uygulanmasıyla başlamıştır. Bu alandaki en özgün örnekler; Polonyalılar tarafından tasarlanıp, Almanlarca geliştirilen *Enigma* ve yine Almanlarca yapılan *Geheimschreiber* cihazlarıdır. II. Dünya savaşı yılları boyunca Almanya tarafından yoğun bir şekilde kullanılan *Enigma* , İngilizlerce yapılan ilk bilgisayarlardan biri olarak adlandırılabilecek Colossus tarafından kırılabilmiştir.

II. Dünya savaşı yılları Atlantik Okyanusu'nun öteki yakasındaki, Amerika Birleşik Devletleri'nde, çağdaş kriptanalizin doğuşuna şahit olmuştur. Bugün tüm simetrik kripto sistemler için kullanılan kriptanalitik yöntemler bu yıllar boyunca *William F. Friedman* (1891-1969) ve grubu tarafından geliştirilmiş, savaş yılları boyunca Japonlar tarafından kullanılan *Purple* şifresi yine *Friedman*'ın kriptanalitik çalışmaları sonucunda kırılmıştır. Bugün şifre kırma çalışmalarının tamamını içine alan Kriptanaliz teriminin babası da yine *Friedman*' dır.

1948 ve 49 yılları, kriptografinin matematiksel olarak formalize edildiği yıllar olup; Bilgi Kuramı'nın yaratıcısı olan *Shannon*'ın bu alanda yayımladığı ilk bildiri, bugün de çağdaş kriptografinin temeli olarak kabul edilir.

Bilgisayar bilim ve teknolojisinin kendisini bütünüyle kabul ettirmesi kriptolojinin "kağıt-kalem"' den yongalara geçmesini sonuçlamıştır. 20.yüzyılın son çeyreği içinde bu alandaki temel gelişmeler şöylece sıralanabilir:

1971 yılında Yerine Koymalı ve Yer Değiştirmeli şifreleme tekniklerini olağanüstü karmaşık biçimde kullanan, 128 bit uzunluğunda anahtarla çalışan *Lucifer* şifresi IBM' ce geliştirilmiştir. Aynı algoritma 1977 yılında, 56 bit anahtar uzunluğuyla Veri Şifreleme Standart algoritması (data encryption standart)-DES- adıyla Amerika Birleşik Devletler Standart Enstitüsü' nce standartlaştırılmıştır.

1976 yılı, Kriptografi ve kriptanaliz çalışmalarını bundan sonraki uzun yıllar boyunca etkileyecek , radikal değişimlerin ortaya konduğu yıl olarak tanımlanabilir. Bu yılda *Diffie* ve *Hellman* şifreleme ve deşifreleme işlemlerinde birbirinden bağımsız anahtarların kullanıldığı

asimetrik kripto sistemleri ortaya koymuştur. Temel teorisi 1976 yılında böylece atılan asimetrik kripto sistemler; 1978 yılında tam anlamıyla formalize edilerek, Halk Anahtarlama Kripto Sistemler başlığı altında, yaratıcıların baş isimlerinden oluşan RSA(Rivest-Shamir-Adleman) algoritması tanımıyla kullanıma girmiştir.

Gücünü Kompleksite Kuramından alan RSA algoritması, her ne kadar yüzyıllarca kırılmayacak bir kriptosistem olarak algılanmışsa da , 1994 baharında ,129 hane uzunluğunda bir anahtara sahip RSA şifresinin kırıldığı duyurulmuştur.

Kriptoloji sistemlerindeki karmaşanın giderek artması, aslında bilim ve bunun doğal uzantısı olan teknolojideki gelişmelerin bir sonucudur. Hemen her alanda yapılan bilimsel çalışmalar; güvenlik gereksinimi nedeniyle Kriptolojiye uygulanmakta olup, Kuantum fiziğinden yola çıkılarak geliştirilen Kuantum Kriptografi ve genetik biliminden yararlanılarak ortaya konan genetik algoritmalar ve Genetik Kriptografi bu çabaların günceldeki en somut örnekleridir.

Öte yandan bilgisayar devriminin oluşturduğu sayısal toplum, güvenlik ve gizlilik gereksinimini her zamankinden daha çok duymakta, bu da Kriptoloji bilim ve teknolojisinin hızla gelişerek yaygınlaşmasını sonuçlanmaktadır.

İşlemlerdeki nicelik artışı ve matematikleşme, yine *Kahn*'a göre olayın felsefi boyutunu temsil etmekte olup; Kriptolojinin dini ve/veya mistik öğretilerden ayrılarak, özellikle 20. Yüzyılda bütünüyle matematik formalizasyona geçişini temsil etmektedir.Aslında; bilimselleşme süreci olarak da tanımlanabilecek bu dönüşüm, Rönesansla başlamış, 1700'lerin sonuna doğru ilk matematiksel Kkriptoloji çalışmalarıyla sürmüştür. Bütünüyle matemetik-istatistik olan kriptanaliz, Friedman'nın 1920 yılında yayımladığı Tesadüflük İndeksi çalışmasına kadar nicel bir anlamda incelenmemiş, ancak bundan sonra frekans dağlımlarından gidilerek metinleri yeniden oluşturulabileceği fikri uygulamaya başlamıştır.

20. Yüzyılın temel belirteci olan pozitif bilim ve teknoloji, Kriptolojiyi sadece matematik bir temele oturtmakla kalmamış, güncel yaşamın temel gereksinimleri olan güvenlik ve gizliliğin sağlanmasında en vazgeçilmez unsur kılmıştır.

1.2 Kripto Algoritmalarına Giriş

Şifreleme işlevinin güvenli bir şekilde gerçekleştirilmesi kriptolama sırasında kullanılan tüm yöntem ve bilgilerin gizliliğine dayanır. Ancak herhangi bir nedenle kripto işlevlerinin açığa çıkabileceği düşünülerek iletişim güvenliği ,kripto anahtarı denen ek bilgi ile arttırılmıştır. Bu durumda kriptolama işlemi sırasında açık mesaj ,kripto anahtarı aracılığı ile şifrelenir. Bir başka deyişle açık mesaj ile kriptolanmış mesaj arasındaki geçişler kripto algoritmasına bağlı olduğu kadar kullanılan anahtar bilgisine de bağlıdır,

Günümüzde kullanılan kriptografik algoritmalar ikiye ayrılır. Bunlar, kullandıkları anahtar biçimine göre simetrik veya asimetrik olarak adlandırılırlar. Simetrik algoritmalarda verinin kriptolanmasında kullanılan anahtar bilgisi ile kriptolanmış verinin kriptosunun çözülmesinde kullanılan anahtar aynıdır. Bu sebeple açık kullanılan anahtarın üçüncü kişilerden gizlenmesi gerekmektedir. Bu algoritmalara örnek olarak DES'i verebiliriz. Asimetrik algoritmalarda ise her iki kullanıcı ver iyi farklı anahtar bilgisi ile kriptolar ve çözerler. Bu amaçla her bir kullanıcı biri gizli diğeri açık iki anahtar kullanır. Açık anahtar bilgisi alıcı tarafa herhangi bir koruma yapılmadan iletilir. Açık anahtarı alan taraf bu bilgiden kriptoyu çözmek amacı ile kullanacağı gizli anahtarı üretir. Açık anahtarın üçüncü bir kişinin eline geçmesi tek başına hiç bir şey ifade etmez. Her ne kadar bu yöntemin, anahtarların alıcı tarafa iletilmesi işlemini basitleştirdiği düşünülse de asimetrik algoritmaların işlem süresinin yüksek oluşu, veri kriptolamada yaygın olarak kullanılmalarını engellemektedir. Hem simetrik hem de asimetrik algoritmaların temel özelliği yapılarının açık olarak bilinmesidir. Böylece kripto algoritmalarının tasarımında, kriptolanmış verinin anahtar bilgisi olmadan şifrenin çözülememesi ilkesi göz önüne alındığından, iletişim güvenliği, kripto anahtarlarının güvenliği problemine indirgenmiştir.

Kriptografik anahtarların üretilmelerinden dağıtılmalarına ve kullanılmalarına kadar olan sürecin sıkı bir denetim altına alınması kriptografik işlemlerin güvenilirliği açısından büyük öneme sahiptir. Bu işlemlerin tümüne anahtar yönetim denir. Anahtar yönetiminin en zayıf halkası, dış müdahalelere tümüyle açık olan anahtarların güvenli bir şekilde dağıtılması işlevidir. Geleneksel olarak kriptografiden yararlanan, askeri ve diplomatik servisler, bu

problemi, anahtarları, güvenilir kuryeler aracılığı ile ileterek çözmüşlerdir. Günümüzde bu yöntem hala kullanılmaktadır. Ancak bu oldukça pahalı ve yavaş bir yöntemdir.

Günümüzde kişilerin veya kuruluşların gereksindiği bilgi miktarı geçmiş on yılla dahi karşılaştırıldığında hızla yükselmiştir. Bunun sonucu olarak iletişim olanakları ile birlikte iletişim ortamları da değişmiştir. Bugün artık sistem tasarımcıları çağın gerektirdiği bilgiyi işlemek üzere bilgisayarları geliştirmişler ve verilerin hızlı bir şekilde iletilebilmesini sağlamak amacı ile veri ağlarını tasarlamışlardır. Veri işlenmesinde kullanılan bilgisayar ağları, veri işleyen ana bilgisayar birimlerinden, kontrol birimlerinden ve terminallerden oluşur. Tüm bu yapılar birbirlerine veri bağları ile bağlıdırlar. Veri bağları uydu ve mikrodalga bağlantıları olabileceği gibi anahtarlanmış veya anahtarlanmamış iletişim hatları da olabilirler. Veri ağı birbirine veri bağları ile bağlı düğümlerden (ana bilgisayarlar) ve veri ağının yalnızca ana bilgisayarlar üzerinden kullanabilen terminallerden oluşmaktadır. Veri ağının temel işlevi bir kullanıcının (kişi veya programın) diğer kullanıcı ile haberleşmesi için gerekli erişim yolunu sağlamaktır. Veri ağlarının bu yapısı isteyen kullanıcının bu sistemleri kullanmasını sağlamakla birlikte ortaya iletişim sırasında kullanılan gizli mesajların korunması problemini de çıkarmaktadır. Her ne kadar mesaj güvenliği kriptografik işlemlerle sağlansa da kripto anahtarlarının güvenli bir biçimde üretilmesi ve kullanıcılara dağıtılması problem olmaya devam etmektedir.

Kriptografik anahtar yönetimini ana bilgisayar ve terminallerden oluşan veri ağlarında gerçekleştirebilmek için anahtarların üretilmesinden saklanmasına ve kullanım amacı i!e dağıtılmasına kadar olan sürecin denetlenmesi gerekmektedir. Güvenli iletişimin gerçekleştirileceği ortamın askeri, diplomatik veya sivil olması, kullanılacak anahtar yönetim sistemini de belirlemektedir. Askeri uygulamalarda anahtar üretimi genellikle, Merkezi Denetim Sistemi (MDS) denilen tek bir merkezde yapılmakta ve çağrı kurmak isteyen kullanıcılar bağ!ı bulundukları düğüm aracılığı ile bu merkeze çağrı isteklerini iletmektedirler. Bu istek üzerine üretilen anahtarlar şifrelenmiş biçimde oturum anahtarı bilgisini çağrıyı kurmak isteyen terminale iletir!er. Bu yöntemin avantajı, anahtar yönetimini tek bir merkezden belirlediği için güvenlik denetiminin tek bir ana bilgisayar üzerinde olmasının yeterli oluşudur. Askeri uygulamalar için oldukça uygun olan bu yapının birinci dezavantajı, güvenli iletişim başlamadan önce tüm kullanıcıların tek bir merkezden oturum anahtarı almak zorunluluğunda oluşlarından dolayı MDS üzerindeki işlem yükünün veri ağına bağlı kullanıcı sayısına ve çağrı istek yoğunluğuna bağlı olarak artması, do!ayısıyla çağrı

kurulma süresinin uzamasıdır. İkinci dezavantajı ise MDS ile oluşabilecek herhangi bir iletişim kopukluğunun güvenli iletişimi tamamen durdurma riskidir.

Bir diğer yöntem MDS'nin işlevlerinin ana bilgisayarlara yüklendiği, tek bir merkez yerine veri ağını oluşturan tüm ana bilgisayarlara dağıtılmış anahtar yönetim sistemidir. Bu yöntemin ana ilkeleri IBM tarafından konmuştur. Bu sistemin amacı özellikle sivil uygulamalarda kullanılabilecek, tek bir MDS'nin getirdiği risklerden arınmış, yeni bir anahtar yönetimi ortaya koymaktır.

Bu ortamda herhangi bir A terminaline bağlı kullanıcı, güvenli bir haberleşme gerçekleştirmek istediği zaman çağrı isteğini bağlı bulunduğu ana bilgisayar A'ya iletir. Ana bilgisayar, gelen oturum başlatma isteği mesajını aldığında iletişim sırasında kullanılacak olan KS oturum anahtarını üretir ve bunu A terminaline, ilgili terminalin K_{MTa} master anahtarını kullanarak kriptolayıp $K_{KMTa}(KS)$ biçiminde iletir. Aynı anahtar ve oturum çağrı isteği, karşı terminalin bağlı bulunduğu ana bilgisayar B'ye, ana bilgisayarlar arasında kutlanılan tek yönlü bir anahtar olan KNCab eşliğinde, $E_{KNCab}(KS)$ biçiminde iletilir. Çağrı isteğini alan ana bilgisayar, gerekli oturum anahtarını, çağrılan terminalin master anahtarını kullanarak ilgili terminale $E_{KMTb}(KS)$ biçiminde iletir. Bu işlemler sırasında her iki terminal birbirlerinin kimlik verilerini karşılıklı olarak sorgularlar. Bu sorgulamanın amacı, iletişimi gerçekleştirecek olan her iki kullanıcının, karşı tarafın, A veya B terminali yerine, bir düşman terminal olmasından şüphe etmesidir. Bu işlemin tamamlanmasından sonra kullanıcılardan her biri, M açık mesajını $E_{KS}(M)$ biçiminde kriptolayıp diğer kullanıcı kriptolanmış mesajı $D_{KS}(E_{KS}(M))$ biçiminde çözerek haberleşebileceklerdir.

Oturum anahtarının bu şekilde dağıtılması, haberleşme sırasında kullanıcının dışarıdan bir anahtar yükleme gereksinimini ortadan kaldıracaktır. Her terminal için özel olan ve terminallere manüel olarak yüklenen terminal master anahtarlarının değişme gereksinimi sık olmadığından güvenli haberleşme, ana bilgisayarlar tarafından üretilen oturum anahtarları ile gerçekleşir. Bunun sonucunda, veri anahtarlarının haberleşme ortamının dışında ayrı bir güvenli kanaldan iletim gereksinimi ortadan kalkmaktadır. Bu çalışmada IBM PC ortamında çalışabilecek simetrik bir kripto algoritması örneği olarak DES kullanılmıştır. Ayrıca DES algoritmasını kullanan, bir yan rassal anahtar üretim fonksiyonu, tek yönlü hash fonksiyonu standardı olan MD4 kullanılarak gerçekleştirilmiştir. Anahtar yönetim fonksiyonları, terminal ve ana bilgisayar uygulamaları olarak ayrı ayrı tasarlanmıştır. Bu fonksiyonlar ana bilgisayar

ve terminallerden oluşan bir ağ topolojisinde kullanılacak örnek bir anahtar yönetim protokolünde kullanılmışlardır.

1.2.1 Kriptoloji Temelleri

Kriptoloji gizli sistem oluşturma veya çözme bilimidir. Bunun tasarım veya gizliliği oluşturma kısmına kriptografi, çözme veya gizliliğin ortadan kaldırılma kısmına ise kriptoanaliz denir. .

Bir mesajın anlaşılabilirliğini gizlemek amacı ile iki yöntem uygulanabilir. Bunlardan birincisi mesajın belirli bir yöntem uyarınca kısaltılması, diğeri ise belirli bir teknikle mesajın anlaşılamaz bir biçime dönüştürülmesidir. Bunlardan ilkine stenografi, diğerine ise kriptografi denir.

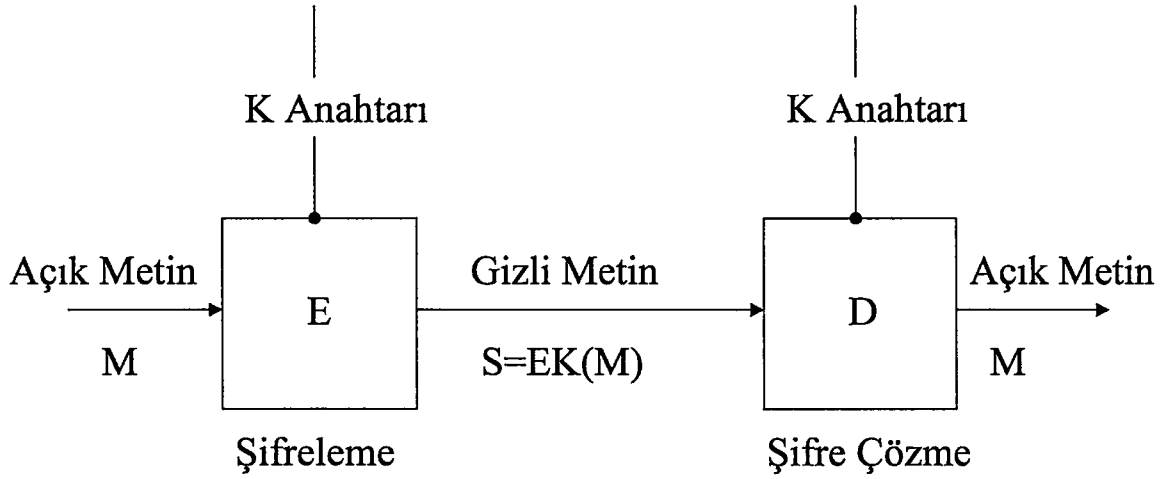
Verinin orijinal biçimindeki haline açık metin denir. Dönüştürülmüş biçimine ise şifrelenmiş metin veya gizli metin denir. Dönüştürme işlemine şifreleme veya kriptolama diyoruz. Dönüştürülmüş metnin ters işlem sonucu açık metin halinin elde edilmesine ise şifre veya kripto çözme denir. M açık metni, E ve D simgeleri de kriptolama ve kripto çözme işlemlerini temsil ettiğinde, gizli metin S,

$S = E_K(M)$

olarak gösterilir. Gizli metinden açık metnin elde edilmesi amacı ile kripto çözme işlevinde kullanılması,

$M = E_K(S)$

olarak gösterilir. Her iki ifadede kullanılan _K, kripto anahtarını temsil etmektedir. Şekil 1. bu iki işlemi göstermektedir.



Şekil 1: Şifreleme ve şifre çözme işlemleri

Burada K anahtarını kullanmanın amacı, eğer M açık metni sadece E kripto işlevi ile şifrelenirse gizliliği sağlayacak olan tek parametrenin kripto algoritması olması ve iletişim güvenliğini arttırmak için alıcı tarafın her yeni kripto işlevi için farklı bir kripto algoritmasını bilmesinin gerekmesidir. Eğer mesajın şifrelendiği E işlevi kaybolacak olursa yeni bir algoritma tasarlanmak durumundadır. Bu durumda zaman kaybı da göz önüne alınacak olursa bu oldukça pahalı bir yöntemdir. Bunun yanı sıra K anahtarı mesajın kripto algoritması bilinse de ikinci bir güvenlik parametresi olduğundan iletişim güvenliğini arttırmaktadır

.

Şifrenin üçüncü kişiler tarafından çözülebilmesine kripto analitik saldırı denir. Kripto analitik saldırı üç biçimde gerçekleşebilir, bunlar gizli metin saldırısı, bilinen açık metin saldırısı ve seçili açık metin saldırısıdır. Adından da anlaşılabileceği gibi gizli metin saldırısı kripto analizcinin yalnızca gizli metni elde etmesi ile gerçekleşir. Sadece şifrelenmiş metinden hareket edilerek çözülebilen kripto işlevine sahip sistemler hiç bir şekilde kullanılmamalıdır. Bilinen açık metin saldırısında, kripto analizci, elinde bulunan açık metin ile gizli metin çifti arasında ilişkiler bularak sistemi çözer. En güçlü saldın yöntemi seçili açık metin saldırısıdır. Bu yöntemde kripto analizci kendisinin belirlediği bir açık metine karşılık düşen gizli metin yardımı ile kullanılan anahtarı belirlemeye çalışır. Bu yöntemde eğer algoritma bazı açık metinler karşısında anahtar bilgisini dışarıya sızdırıyor ise istenen amaç elde edilmiş olur.

Kriptografik sistemlerin güvenli olabilmesi iki şekilde mümkündür. Birinci durumda kripto analizi gerçekleştirmeye çalışan kişi, elinde tüm olasılıkları denemek için ne kadar işlem gücü olursa olsun, hiç bir zaman, gizli mesajdan açık mesaja dönüşümü sağlayacak yeterli veriyi elde edemez. Bu durumda sisteme koşulsuz olarak güvenli denir. İkinci durumda ise kripto analizi gerçekleştiren kişinin elinde yeterli veri olsa da kripto analitik problemin çözüme ulaşabilmesini sağlayacak yeterli işlem gücüne sahip değildir. Bu tür sistemlere işlemsel olarak güvenli sistemler denir.

1.2.2 Kripto Algoritmaları

Gelişmiş, klasik kripto algoritmaları, güvenliği istatistiksel olarak belirsizliklerinden kaynaklanmaktadır. Bugün birçok klasik kripto sistemi yeterince güvenli bulunmadığından dolayı kullanılmamaktadır. Diğer yandan koşulsuz güvenli kripto algoritmaları sadece bu sınıfta yer almaktadır.

1.2.2.1 Sezar Şifresi

En eski kripto algoritmalarından biri olan Sezar şifresi adını Julius Caesar'dan almaktadır. Orijinal olarak kripto algoritması şu şekilde ifade edilebilir:

$S=M+3 \pmod{26}$

Burada S açık metin harfini M gizli metin harfini göstermektedir. Daha sonra bu algoritma genelleştirilerek şu biçimi almıştır,

$S = M + i \pmod{26}, 0 <= i <= 25$

Burada i öteleme katsayısı genel anlamda bir kriptografik anahtara karşılık düşmektedir. Burada kullanılan i anahtarının tanım aralığının 25 sayısı ile kısıtlı olması algoritmanın güvenilirliğinin düşük olmasına yol açmaktadır.

1.2.2.2 Tekli Alfabetik Yer Değiştirme

Daha yüksek güvenlikli kripto sistemleri düşünüldüğünde alfabede bulunan her harfin yerine başka bir harfin, belirli bir tablo uyarınca yerleştirildiği yapılar ortaya çıkmıştır. Bu tekniğe tekli alfabetik yer değiştirme adı verilir . Bu teknik özel bir alt uygulaması olarak Sezar şifresini de içermektedir. Olası uygulama sayısının yüksekliği göz önüne alındığında tekli alfabetik yer değiştirme güçlü bir algoritma olarak düşünülebilir. Ancak şifrelemeden sonra ortaya çıkan gizli metinde bulunan harflerin dağılım sıklığı, kullanılan diller göz önüne alındığında bazı istatistiksel ipuçları verdiğinden dolayı, bu tür şifrelemenin de çözülebilmesi oldukça basittir.

1.2.2.3 Çoklu Alfabetik Yer Değiştirme

Çoklu alfabetik yer değiştirme tekniğinde, alfabede bulunan her bir harf üzerine n periyotlu bir dizi yer değiştirme uygulanır. Bu yöntem, dilin istatistiksel olarak tahmin edilebilen yapısından kaynaklanan birebirliği bozmayı amaçlamaktadır.

Çoklu alfabetik yer değiştirme yönteminin bir uygulaması olan Vigenere şifresi periyodik olarak belirli bir anahtar değerinde yer değiştirme uygular. Bu uygulamada K anahtarı bir dizi harf olar ak belirlenir. Bu durumda $K = k_1, k_2, ... k_d$ olarak gösterilen ifadede kullanılan k; değeri i=l,2,...,d olmak üzere, alfabede yapılması gereken i'nci ötelemeyi gösterdiğinde, şifreleme sonucu elde edilen harf,

$f(a) = a + k_i \pmod n \ (n = 26)$

biçiminde gösterilir.

Yer değiştirme şifrelerinde sistemin güvenilirliği anahtar uzunluğuna bağlı olarak değişir. Eğer anahtar uzunluğu mesaj uzunluğunda olur ise bu duruma, çalışan anahtar şifresi adı verilir.

1.2.2.4 Tek Kullanımlı Şerit Yöntemi

Analitik olarak koşulsuz güvenli tek şifreleme yöntemidir. Bunu gerçekleştirmek amacı ile tek bir kez kullanılmak üzere, mesaj uzunluğuna eşit veya daha uzun, tümüyle rassal bir anahtar seçilir . Anahtar, ikili sayı düzeninde düşünülen mesaj ile dış veya'lanır. Ancak bu sistem bir çok uygulama için oldukça kullanışsızdır. Çünkü her bir kullanım için en az mesaj uzunluğunda olan anahtarın, haberleşme öncesi her iki tarafa da ulaştırılması gerekmektedir.

1.2.2.5 Dönüşüm Şifreleri

Yer değiştirme şifreleri açık metinde bulunan tüm sembollerin yerini değiştirmezken sadece bu sembollerin yerine şifreleme amacı ile kullanılacak olan yeni sembolleri belirler. Ancak dönüşüm şifrelerinde açık metnin harfleri de yer değiştirir. Bu yöntem ilk olarak Eski Yunanda kullanılmıştır. Uzun dar bir şerit üzerine yazılan açık metnin, çapı anahtar bilgisi olan bir silindire spiral olarak dolandırılması ile elde edilen gizli metnin çözülmesi ancak aynı çaplı bir silindire gizli mesajın sarılması ile mümkün oluyordu.

Bu sistemde, kullanılan harflerin dağılım sıklığı korunuyordu. Ancak dilin yapısında olan ikili, üçtü ve daha yukarı seviyede harflerin bir araya gelmesi olasılığı bozulduğundan, bu yöntem, dil üzerinde basit yer değiştirmeye dayalı kripto yöntemlerine göre daha güvenli olarak kabul edilir.

1.2.3 Kriptografik Algoritmaların Gücü

İyi kriptografik sistemler öyle tasarlanmalıdırlar ki bunları kırmak mümkün olduğu kadar zor olsun. Uygulamada kırılamayacak sistemler yapmak olasıdır (ancak genelde bu ispatlanamaz). Bir sistem tasarımcısı için sistemi kırılabilir bırakmanın hiçbir özrü yoktur. Güvenliği aşmak için kullanılabilecek her mekanizma açığa çıkarılmalı, belgelenmeli ve son kullanıcının dikkatine sunulmalıdır. Teoride, herhangi bir anahtarlı kriptografik metot olası tüm anahtarların denenmesi ile kırılabilir.

Eğer tüm anahtarların denendiği kaba kuvvet (brute force) kullanımı tek yolsa, gerekli hesaplama gücü anahtarın uzunluğu ile üstel olarak artar. 32 bitlik bir anahtar (yani 4,294,967,296) adım alır. Bu herhangi bir ev bilgisayarı ile yapılabilecek bir şeydir. 40 bitlik anahtarlar adım alır- bu tür bir hesaplama (kullanılan algoritmanın etkinliğine bağlı olarak) modern bir ev bilgisayarında bir hafta gibi bir zaman gerektirir. 56 bit anahtarlı bir sistem (DES gibi) esaslı bir zahmet gerektirir (çok sayıda ev bilgisayarının güç paylaşımı ile bunu kırmak birkaç ay alır), ama özel donanımlarla kolayca kırılabilir. Özel donanımların maliyetleri de doğal olarak yüksektir, ama organize suç örgütleri, büyük hükümet ve şirketler bunları alabilirler. 64 bitli anahtarlarda şimdiden kırılabilir durumdadırlar. 80 bitli anahtarlar bir kaç yıl sonra kırılabilecekken ve 128 bitli anahtarlar kaba kuvvet ile muhtemelen kırılamayacaklardır. Ancak anahtar uzunluğu tek önemli konu değildir. Pek çok cipher olası tüm anahtarlar denenmeden de kırılabilir. Genelde, diğer metotların daha da etkili kullanımı ile bile kırılamayacak cipherlar tasarlamak çok zordur. Kendi cipherlarınızı tasarlamak eğlenceli olabilir, ancak gerçek uygulamalarda bunu yapmanız tavsiye edilmemektedir. Çoğunlukla, algoritmanın gizliliğine dayanan algoritmalar güvenli değillerdir. Özellikle yazılım yoluyla, herhangi bir kimse algoritmayı tersine çevirip çalıştırabilecek (disassamble) birini kiralayabilir.

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

1.2.4 Ülkelerin Kriptografi Politikalarındaki Farklılıklar

Dünya üzerindeki değişik ülkelerin kriptografi politikaları oldukça önemli farklılıklar göstermektedir. ABD, kriptografik ürünlerin dış satımına kısıtlamalar koymakta, örneğin çok yaygın olarak kullanılan bir açık(çift) anahtarlı şifreleme yöntemi olan RSA algoritmasının anahtarı olan n sayısının 512 ikili'yi aşmasına izin vermemektedir. Ülke içinde 1024 ikili

uygulamaları yasaklamadığı halde, bu durumdan endişe duymakta ve sade vatandaşla birlikte, yasadışı örgütlerin de çok önemli bir gizli iletişim gücü ele geçirmesini sakıncalı bulmaktadır. ABD'nin bu soruna bulduğu çözüm, güvenilir üçüncü kuruluşlar, yani GÜK'ler yardımıyla, herkesin gizli (özel) anahtarına, yasalar gerektirdiği zaman ulaşılmasını sağlayacak yöntemler geliştirilmesidir. Fransa, İsrail, Belçika, Çin gibi ülkeler de, kriptografik ürünlerin dışalımını kısıtlamış, ve gizli anahtarların GÜK'ler tarafından saklanmasını zorlayıcı önlemler almışlardır. İngiltere'deki kriptografi politikaları da benzer endişelerle planlanmaktadır. En uç önlem ise bir Güneydoğu Asya ülkesinden gelmiş ve Birmanya, 1996'nın Eylül ayında Internet bağlantılarını yasaklamıştır.

Öte yandan birçok Avrupa ülkesi ve Avustralya, Japonya gibi ülkeler, kısıtlamalar ve yasaklamalardan yana değildirler. Özellikle Avrupa Birliği ülkeleri, ikili gizliliği tehdit eden her türlü önlemin, insan haklarına aykırı olduğunu ve elektronik ticaretin serbest gelişimini engelleyeceğini ileri sürmektedirler. Japonya, kriptografiye öncelikle ekonomiyi canlandırması açısından bakmakta, ulusal güvenlik yönünden kriptografiyi bir tehdit olarak algılamamaktadır. Bu ülkelerden, GÜK'ler konusundaki politikası belli olmayan Japonya dışındakiler, GÜK'lerin gerekliliğine inanmamakta, üçüncü kuruluşlara güvenilmesi gereksinimi yaratan bir toplumsal düzenlemenin, sonunda GÜK'lerin güvenilirliğini zorlayacak, sarsacak yöntemler gelişmesine yol açabileceğini düşünmektedirler. Bu durum ise yasalara uyan vatandaşların haklarını zedeleyecek, sayısal imzaların taklit edilebilmesine, gizli mesajların istenmeyen kişiler tarafından okunmasına, ekonominin zarar görmesine yol açabilecek ve E-ticaretin gelişmesini engelleyebilecektir.

GÜK'leri gerekli veya gereksiz bulan iki ayrı görüş, onay kurumlarının (OK) gerekliliği konusunda ise birleşmektedir. Onay kurumlarını GÜK'lerden ayıran en önemli özellik, gizli anahtarları saklamaması ve anahtar çiftini üretip kullanıcıya verse bile kendisindeki gizli anantar kopyasını imha etmesidir.

1.2.4.1 Türkiye'deki Durum

Yurdumuzda bilgisayar ağları altyapısının geliştirilmesi yönünde önemli çabalar vardır. Türk Telekom A.Ş.'nin Internet'in sağlıklı gelişimi ve Türkiye'de Internet gereklerinin saptanması için yürütmekte olduğu çalışmalar hızla devam etmektedir. Türk Telekom'un Internet stratejisi oluşturulurken, diğer ülkelerdeki telekomünikasyon operatörlerinin bu

alandaki rolleri incelenmiş ve bu deneyimlerin sonuçları Türkiye uygulamasına aktarılmaya çalışılmıştır. Son yıllarda gerek Internet dünyasında, gerek Türkiye Internet şebekesinde önemli değişiklikler yaşanmıştır. Bu bağlamda, Türkiye'yi içine alan yeni teknoloji temelli ve en son uygulamalara açık bir Internet altyapı ağının kurulmasının gerekliliği konusunda görüş birliğine varılmıştır. 1995 yılı sonunda hızlı Internet hizmetleri için TURNET ihalesine çıkılmış ve 1996'da TURNET şebekesi kurulmuştur. Ancak, telefon konuşmaları için ortalama bağlantı süresi 1-2 dakika olarak düşünülen telefon şebekesi üzerinden yapılan veri bağlantıları, ortalama 15-20 dakikaya kadar çıktığı için, Internet trafiğini kaldıramayan şebekede teknik sorunlar çıkmıştır. Yurtdışı çıkış kapasitesi de oldukça yetersiz kalan TURNET ağı yerine 1998 yılı sonuna kadar, 3 x 34 megabit/saniyelik yurtdışı çıkışlı TTNET ağının kurulması planlanmaktadır. Üç metropolitanı (Ankara, İstanbul, İzmir) saniyede 155 megabit hızla birbirine bağlayacak olan bu ağ, 20 büyük ilde 34, geri kalan illerde de 2'şer megabit/saniyelik bağlantılar sağlayacaktır.

Türkiye'de birçok kamu kuruluşu, bilgisayar donanım ve yazılım altyapısını geliştirerek, yapmakla yükümlü olduğu işlerin otomasyonu, böylelikle insani hatalardan arındırılması, veri bankalarında toplanan bilgilerin erişim kolaylığı ve çabukluğundan yararlanılması gibi amaçlarla projeler yürütmektedir. İçişleri Bakanlığı Merkezi Nüfus İşleri Sistemi (MERNİS), Sağlık Bakanlığı Temel Sağlık İstatistikleri Bilgi Sistemi, Gümrük Bakanlığı Gümrük Sistemleri Otomasyonu gibi projeler bunların başlıcalarıdır.

Türk bankacılık sektörünün de oldukça gelişmiş bir bilgisayar altyapısı vardır. Birçok büyük banka, merkez ve şubeleri arasında gerçek zamanda bilgisayar iletişimini sağlamıştır . Merkez bankası ve diğer bankalar arasındaki elektronik fon transferleri (EFT) yine gerçek zamanda ve 'RTGS-Real Time Gross Settlement' sistemi kullanılarak yapılmaktadır, ve Türk bankalarının yüzde 99'u bu sisteme bağlıdır. RTGS kullanımında Türkiye, dünyadaki öncü 5-6 ülke içerisindedir. 1992 yılından beri çalışmakta olan EFT-1 projesinden sonra, çok daha kapsamlı ve yetenekleri arttırılmış olan EFT-2 ve EMKT (Elektronik Menkul Kıymet Transferi) projeleri kapsamında çalışmalara devam edilmektedir.

Elektronik ticarete başlangıç olarak düşünülebilecek bazı girişimler de vardır. Bilgisayar ürünleri ve kitap satan birkaç firma, ve büyük bir süpermarket, Internet üzerinde hazırladıkları 'web siteleri' yani tül sayfaları ile kullanıcıya ulaşmakta, zengin ürün çeşitleri sergileyebilmektedirler. Fakat bu uygulamaların hiçbirinde açık anahtarlı kriptografi kullanılmadığı için, firmaları kötü niyetli kullanıcıların aldatmasına karşı koruyabilecek bir önlem de yoktur. Diğer bir deyişle, kullanıcının iddia ettiği kimliğin kanıtlanması, kredi kartı numarasının kendisine ait olduğunun belirlenmesi ve mesajın yolda bozulmadığının gösterilmesine yarayacak sayısal imzanın olmaması, firmayı E-ticaret uygulamasında bir risk altına sokmaktadır.

Benzer eksiklikler kamu kuruluşlarının EDI projelerinde de sorun yaratmakta, sayısal imza için gereken teknik ve hukuksal altyapının Türkiye'de hazır olmaması, sistemi kullanmak için başvuran yurttaşların 'ıslak imzalarının' bulunduğu belgelerinin, görevliler tarafından, bilgisayar ekranındaki belgeyle karşılaştırılarak kontrolü insani hataları gündeme getirebilecektir.

2. GÜNÜMÜZDE KULLANILAN ŞİFRELEME ALGORİTMALARI

Günümüzde kullanılan kriptografik algoritmalar ikiye ayrılır. Bunlar, kullandıkları anahtar biçimine göre simetrik veya asimetrik olarak adlandırılırlar. Simetrik algoritmalarda verinin kriptolanmasında kullanılan anahtar bilgisi ile kriptolanmış verinin kriptosunun çözülmesinde kullanılan anahtar aynıdır. Bu sebeple açık kullanılan anahtarın üçüncü kişilerden gizlenmesi gerekmektedir. Bu algoritmalara örnek olarak DES'i verebiliriz. Asimetrik algoritmalarda ise her iki kullanıcı ver iyi farklı anahtar bilgisi ile kriptolar ve çözerler. Bu amaçla her bir kullanıcı biri gizli diğeri açık iki anahtar kullanır. Açık anahtar bilgisi alıcı tarafa herhangi bir koruma yapılmadan iletilir. Açık anahtarı alan taraf bu bilgiden kriptoyu çözmek amacı ile kullanacağı gizli anahtarı üretir. Açık anahtarın üçüncü bir kişinin eline geçmesi tek başına hiç bir şey ifade etmez. Her ne kadar bu yöntemin, anahtarların alıcı tarafa iletilmesi işlemini basitleştirdiği düşünülse de asimetrik algoritmaların işlem süresinin yüksek oluşu, veri kriptolamada yaygın olarak kullanılmalarını engellemektedir. Hem simetrik hem de asimetrik algoritmaların temel özelliği yapılarının açık olarak bilinmesidir. Böylece kripto algoritmalarının tasarımında, kriptolanmış verinin anahtar bilgisi olmadan şifrenin çözülememesi ilkesi göz önüne alındığından, iletişim güvenliği, kripto anahtarlarının güvenliği problemine indirgenmiştir.

Kriptografik anahtarların üretilmelerinden dağıtılmalarına ve kullanılmalarına kadar olan sürecin sıkı bir denetim altına alınması kriptografik işlemlerin güvenilirliği açısından büyük öneme sahiptir. Bu işlemlerin tümüne anahtar yönetim denir. Anahtar yönetiminin en zayıf halkası, dış müdahalelere tümüyle açık olan anahtarların güvenli bir şekilde dağıtılması işlevidir. Geleneksel olarak kriptografiden yararlanan, askeri ve diplomatik servisler, bu problemi, anahtarları, güvenilir kuryeler aracılığı ile ileterek çözmüşlerdir. Günümüzde bu yöntem hala kullanılmaktadır. Ancak bu oldukça pahalı ve yavaş bir yöntemdir.

Günümüzde kişilerin veya kuruluşların gereksindiği bilgi miktarı geçmiş on yılla dahi karşılaştırıldığında hızla yükselmiştir. Bunun sonucu olarak iletişim olanakları ile birlikte iletişim ortamları da değişmiştir. Bugün artık sistem tasarımcıları çağın gerektirdiği bilgiyi

işlemek üzere bilgisayarları geliştirmişler ve verilerin hızlı bir şekilde iletilebilmesini sağlamak amacı ile veri ağlarını tasarlamışlardır. Veri işlenmesinde kullanılan bilgisayar ağları, veri işleyen ana bilgisayar birimlerinden, kontrol birimlerinden ve terminallerden oluşur. Tüm bu yapılar birbirlerine veri bağları ile bağlıdırlar. Veri bağları uydu ve mikrodalga bağlantıları olabileceği gibi anahtarlanmış veya anahtarlanmamış iletişim hatları da olabilirler. Veri ağı birbirine veri bağları ile bağlı düğümlerden (ana bilgisayarlar) ve veri ağının yalnızca ana bilgisayarlar üzerinden kullanabilen terminallerden oluşmaktadır. Veri ağının temel işlevi bir kullanıcının (kişi veya programın) diğer kullanıcı ile haberleşmesi için gerekli erişim yolunu sağlamaktır. Veri ağlarının bu yapısı isteyen kullanıcının bu sistemleri kullanmasını sağlamakla birlikte ortaya iletişim sırasında kullanılan gizli mesajların korunması problemini de çıkarmaktadır. Her ne kadar mesaj güvenliği kriptografik işlemlerle sağlansa da kripto anahtarlarının güvenli bir biçimde üretilmesi ve kullanıcılara dağıtılması problem olmaya devam etmektedir.

2.1 Günümüzde Kullanılan Simetrik Algoritmalar

Günümüzde kullanılan birçok şifreleme yöntemi esas olarak yer değiştirme ve dönüştürme yöntemlerinin bir arada kullanıldığı yapılardan oluşur. Bu algoritmaların güvenilirliği, çözülmeleri için gerekli işlem miktarının yüksek oluşuna bağlıdır. Burada tanıtılan tüm yöntemlerin güvenilirliği en doğru biçimde, seçilmiş açık bir metin ile ondan elde edilmiş gizli metin arasındaki ilişkinin, istatistiksel olarak değerlendirilmesiyle mümkündür. Açık metin dizisi üzerinde her bir karakter yerine kullanılan bit dizisi üzerinde yapılan kriptolama işine dizi kriptolama denir. Diğer bir yöntem ise metnin bloklara ayrılması ve bunlar üzerinde şifrelemenin gerçekleştirildiği blok kriptolamadır.

2.1.1 Dizi Kriptolama (STREAM CHIPHERS)

Geçmişte kullanılan yer değiştirme algoritmalarının günümüzde kullanılan biçimidir. Dizi kriptolama yönteminde, tek kullanımlı şerit yönteminde kullanıldığı gibi, uzun anahtar bilgisine ihtiyaç vardır. Bu sebepten, yarı rassal nitelikte bir anahtar üretmek amacıyla, geri beslemeli öteleme kütüklerinden yararlanılır. Üretilen anahtar ile açık metin dış veya'lanarak gizli metin elde edilir. Şifrenin çözülebilmesi için üretilen anahtarın alıcı tarafta da üretilmesi

gerekir. Bunun sonucunda gizli metin ile dış veya'lanan anahtar bilgisi açık metni verecektir. Ancak anahtar üretecinde doğrusal olmayan bir yöntem kullanılmayacak olursa bu şifreleme yöntemi bilinen metin saldırısına açık olacaktır.

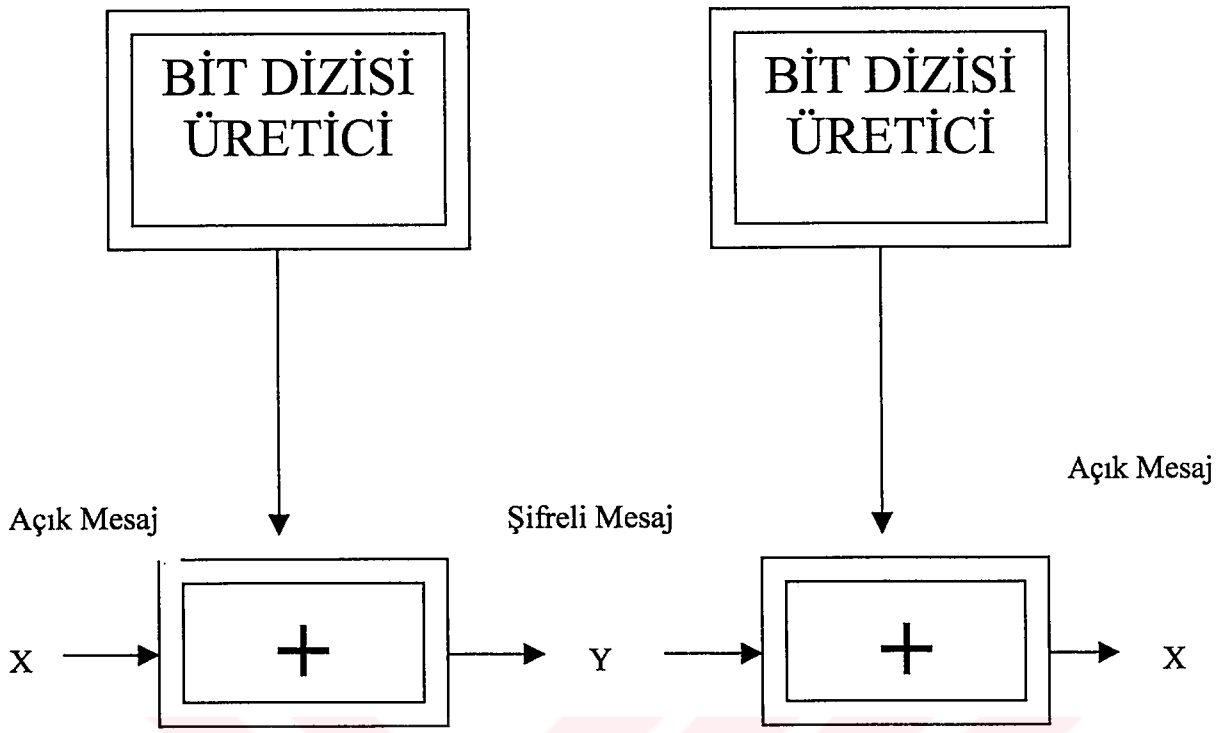
Dizi şifreleyiciler, gizli anahtarlı şifreleme sistemler içinde yer alırlar. Bugün en çok kullanılan dizi şifreleyiciler iki toplayıcı dizi şifreleyiciler. Böyle bir şifreleyicide K bitlik gizli anahtar (Z) bir kayan anahtar üretici kontrol etmek için kullanılır. Kayan Anahtar Üretici Z_1, Z_2,.....,Zn ikili dizisini üretir. Bu dizi kayan anahtar olarak adlandırılır. Genellikle N>>K dir.Şifreli mesaj dizileri basit olarak söylenirse açık mesajın modulo 2'ye göre toplanmasından oluşur.

$Y_n=X_n+Zn$ n=1,2,.....,N dir. (1.1)

Şifreleme ve şifre çözme aynı araçlarla yapılır. Açık mesajın bir tek bitit, şifreli mesajın bir tek bitini etkilerse bu, kötü yayılma özelliğidir. Fakat anahtar biti şifreli mesajın bir çok bitini etkilerse bu da, iyi yayılma özelliğidir.

İkili toplayıcılı dizi şifreleyicilerle bir tek ikili one-time-pad (tek kullanımlı anahtar) arasında kesin bir benzerlik vardır. Eğer $Z=Z_n$ ise yani gizli anahtar kayan anahtar olarak kullanılırsa toplayıcılı dizi şifreleyici one-time-pad'in aynısı olur.

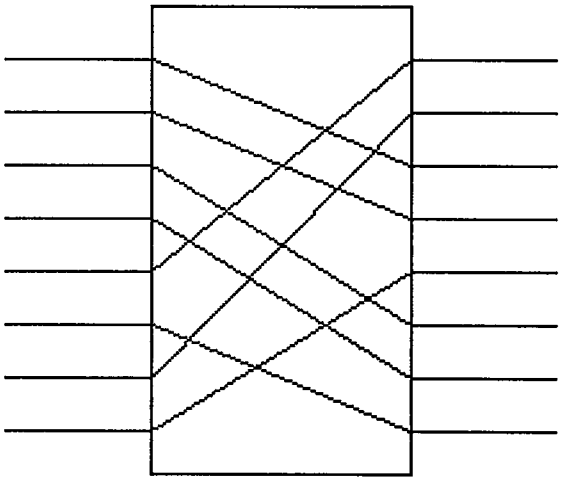
Dizi şifreleyiciler, blok şifreleyicilere göre daha güvenilirdirler fakat burada senkronizasyon problemi vardır. Çünkü açık mesajın şifrelendiği anahtarn şifre çözümü sırasında oluşması için kesin senkronizasyon yapılmış olması gerekir. Burada anahtar zamanla değişen bir fonksiyondur.



Şekil 2.1. Dizi Şifreleyici Blok Diyagramı

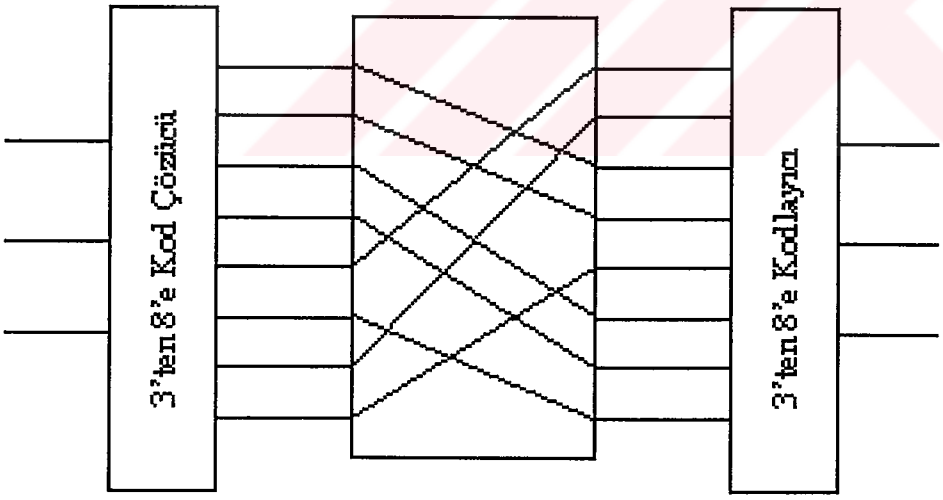
2.1.2 Blok Kriptolama (BLOCK CHIPHERS)

Günümüzde kullanılan blok şifreleme metodu temel olarak basit yer değiştirme ve dönüşüm şifreleme yöntemlerine dayanır. Her bir blok 32'den 128'e kadar değişen oktet uzunluklarından oluşur. Anahtar uzunluğu blok uzunluğuna eşittir.



Şekil 2.2 Permitasyon Kutusu

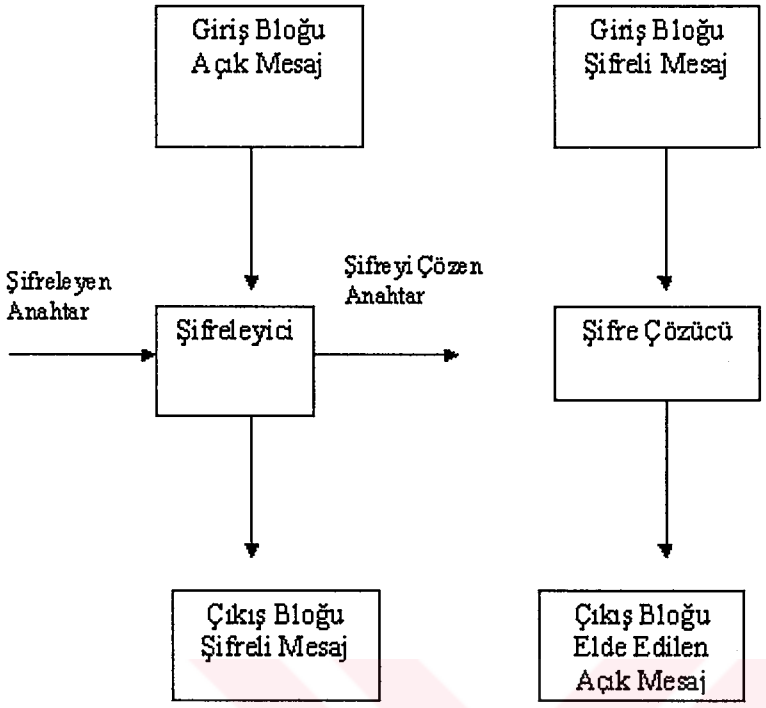
Blok şifrelemeyi oluşturan yapılardan biri P (Permütasyon), diğeri S (Yer Değiştirme) kutusudur. Şekil 2.2.`de bir P kutusu gösterilmektedir. P kutusunun amacı, girişinde kullanılan bit dizisinin sıralamasını belirli bir permütasyon uyarınca değiştirmektir. S kutusu Şekil 2.3.`de gösterildiği gibi üç aşamadan oluşmaktadır. Bunlar dan birincisi kod çözücü, ikincisi bir P kutusu, üçüncüsü ise bir kodlayıcıdır. P ve S kutularının bir araya getirilmesi ile güçlü şifre algoritmaları oluşturulabilir. Yaygın bir şekilde kullanılan Veri Kriptolama Standardı (DES) bu şekilde tasarlanmıştır.



Şekil 2.3 Yerdeğiştirme Kutusu

Bir blok şifreleyici sabit uzunluktaki bir dizi giriş bitini sabit uzunluktaki bir dizi çıkış bitine dönüştürür. Şifreleme ve şifre çözme fonksiyonları öyle tasarlanmışlardır ki çıkış bloğundaki her bir bit giriş bloğundaki ve anahtardaki her bir bite bağımlıdır. Bir şifrenin blok uzunluğu (blok içindeki bitlerin sayısı) kriptografik olarak sağlamlık düşüncesi ile belirlenir,

dolayısı ile basit, şifreli mesaj çözme saldırılarını azaltabilmek için yeterli miktarda büyük olmalıdırlar.



Şekil 2.4 Blok Şifreleme

2.1.3 Blok Şifreleme ile Dizi Şifreleme Arasındaki Temel Farklar

Blok şifreler, bir anda bir tek veri bloğunu şifreler. Kriptografik sağlamlık düşüncesi ile belirlenen minimum blok büyüklüğü blok şifreler için yeterlidir. Dizi şifrelerin minimum blok büyüklükleri yoktur. (Dizi şifreler minimum blok büyüklüğüne gerek duymazlar) En uç noktada 1 bit'e 1 bit temeline dayalı şifreleme kullanırlar.

Blok şifrelemede, her şifreli bit (şifrelenmiş bit) uygun giriş bloğundaki her açık mesaj bitinin karmaşık bir fonksiyonudur. Dizi şifrelemede, her şifreli bit y(i), onun açık mesaj biti x(i)'ye y(i)=x(i)+r(i) ilişkisi ile bağlıdır.

Blok şifreleyiciler bir başlangıç vektörüne ihtiyaç duyabilirler yada duymayabilirler. Açık mesaj bilgisi ve uygun mesaj dizi şifrelemede olduğu gibi aynı yolla bilgiyi açığa çıkarmaz. Kriptografik olarak güçlü dizi şifreleyici tekrar üretilemez (reorginate) ve bu yüzden bir başlatma vektörü gerektirir. Bir başlangıç vektörü blok şifreleyici için her zaman bir ihtiyaç olmamasına rağmen, blok zincirlemede kullanılırlar. Yeniden üretilen (reoriginate)

blok şifreleyicilerde, başlangıç vektörü (initializing vector) sınırlı zaman periyodunda kullanılabilir.

Blok ve Dizi şifreleyicilerin her ikisi de iltişimde ve veri işleme sistemlerinde (Data Processing System) kullanılabilir. Bir blok şifreleyci ile, data şifrelenir ve bloklardaki şifre çözülür. Bu işlemlere tabi tutulan uzunluk önceden algoritma tasarımcısı tarafından belirlenmiştir. Bir dizi şifreleyicisinde ise şifrelenen ve şifresi çözülen datanın uzunluğu kullanıcı tarafındna belirlenir. Bu durum dizi şifrelemeye esneklik getirir. Dizi şifreleme, algoritma ve anahtara ek olarak başlama vektörü (initializing vector) olarak tanımlanan diğer bir parametreyide kullanır. Blok ve Dizi Şifreleme'de geri besleme modları elde edilebilir (burada geçmiş bilgiye bağlılık) kullanılır. Zincirleme, şifrelemeyi yanlızca güçlendirmez aynı zamanda öncelik gerektirmediği durumlarda datayı doğrulamak içinde kullanılabilir.

Bir kriptografik sistemin güçlülüğü, zincirleme metodu kullanılarak zenginleştirilebilir.

Zincirleme (chaining) şifreleme işlemi sırasında kullanılabilen bir prosedürdür. Buna göre bir çıkış bloğu sadece o andaki giriş bloğu ve anahtara değil fakat aynı zamanda daha önceki giriş yada çıkışa da bağlıdır.

Kriptografide (şifrelemede) temel problem, açık mesajı şifreli mesaja dönüştüren kriptoanalize dayanıklı prosedürler oluşturmaktır. Çünkü kriptoanaliz, şifreli iletişimin içine nüfuz etmek (girmek) ve kaynak bilgiyi elde etmek için değişik teknikler kullanır.

Böyle sistemleri kullanan prosedürler ya kod sistemi yada şifre (şifreli) sistem içerirler. Kod sistemler bir kod kitaba yada sözlüğe ihtiyaç duyarlar. Bu kitap yada sözlükler kelimeleri, deyimleri ve açık mesaj cümlelerini buna eşdeğer şifreli kod grubuna dönüştürürler. Bununla birlikte dönüştürülebilen açık mesaj gruplarının sayısı kod kitabının büyüklüğüne bağlıdır. Ayrıca her mesaj kodlanamayabilir ve bu kod sisteminin yeteneği sınırlıdır.

Diğer taraftan, şifreleme sistemleri yeteneklidir. İki temel elemana ihtiyaç duyarla: Bir kriptografik Algoritma, bir prosedür yada doğada değişmez olan bir kurallar yada adımlar

kümesi; ve kriptografik anahtar değişkenlerinin bir kümesi. Anahtar; kullanıcı tarafından seçilen, nispeten kısa gizli karakter yada sayı dizisidir.

Geleneksel Algoritmalar (DES gibi) ve Public-Key Algoritmaları (RSA algoritması ve trapdoor knapsack algoritması gibi) blok şifreleyici algoritmalar arasında gösterilebilirler.

Blok ve Dizi şifreleyicilerin her ikisi de iltişimde ve veri işleme sistemlerinde (Data Processing System) kullanılabilir. Bir blok şifreleyci ile, data şifrelenir ve bloklardaki şifre çözülür. Bu işlemlere tabi tutulan uzunluk önceden algoritma tasarımcısı tarafından belirlenmiştir. Bir dizi şifreleyicisinde ise şifrelenen ve şifresi çözülen datanın uzunluğu kullanıcı tarafindna belirlenir. Bu durum dizi şifrelemeye esneklik getirir. Dizi şifreleme, algoritma ve anahtara ek olarak başlama vektörü (initializing vector) olarak tanımlanan diğer bir parametreyide kullanır. Blok ve Dizi Şifreleme'de geri besleme modları elde edilebilir (burada geçmiş bilgiye bağlılık) kullanılır. Zincirleme, şifrelemeyi yanlızca güçlendirmez aynı zamanda öncelik gerektirmediği durumlarda datayı doğrulamak içinde kullanılabilir.

2.1.4 Veri Şifreleme Standardı – Data Encryption Standard (DES)

En çok kullanılan şifreleme tekniği 1977'de şimdiki adı Ulusal Standart ve Teknolojiler Enstitüsü olan Ulusal Standartlar Bürosunda ortaya atılan Veri Şifreleme Standardıdır (DES). DES' de veri, 56-bitlik bir anahtar kullanılarak 64-bitlik bloklar halinde şifrelenir. Algoritma, 64-bitlik bir girişi bazı aşamalar sonucu 64-bitlik bir çıktı oluşturacak şekilde dönüştürür. Şifrelemeyi geri almak için aynı adımlar, aynı anahtar kullanılarak işlenir.

DES' in çok geniş bir kullanım sahası vardır. Güvenilirlik derecesi ise tartışıla gelen bir konu olmuştur. DES' in tarihçesi şu şekildedir:

1960'ların sonunda IBM , Horst Feistel'in liderliğinde bilgisayar şifrelemeleri için bir proje araştırması başlattı. Proje, 1971'de LUCIFER adı verilen bir algoritmanın geliştirilmesiyle sonuçlandı. LUCIFER, 64-bitlik blokları , 128-bitlik bir anahtarla şifreleyen bir yapıdaydı. LUCIFER'i kazanç haline dönüştürmek isteyen IBM, tek bir çip halinde bir şifreleyici ürün geliştirme peşine düştü. Bu görevi yürütülmesi Walter Tuchman Carl Meyer başkanlığında, sadece IBM araştırmacılarına değil ayrıca diğer bilirkişi ve teknik

danışmanlarca yapıldı. Bu çalışmanın ürünü LUCIFER'in tasfiye edilmiş ve anahtar büyüklüğünün azaltılarak 56-bite düşürülmüş tek bir çipe sığdırılabilen bir versiyonudur.

Bu arada, Ulusal Standartlar Bürosu 1973'te ulusal bir şifreleme standardı arayışı içindeydi. IBM; Tuchman-Meyer projesinin sonuçlarını bildirdi. Bu, şimdiye kadar oluşturulmuş en iyi algoritmaydı ve 1977'de Data Encryption Standard (DES) olarak kabul edildi.

2.1.4.1 DES Şifreleme Algoritması

DES şifrelemesinin tümüyle akışı Şekil 2.5'de gösterilmiştir. Diğer tüm şifreleme yöntemleri gibi bunda da iki girdi vardır: şifrelenecek düzyazı ve anahtar. Burada düzyazı 64-bit, anahtar ise 56-bit uzunluğunda olmalıdır.

Şeklin sol yanına bakarsak, düzyazının üç safhada işlendiğini görürüz. Önce, 64-bitlik düzyazı bir başlangıç permütasyonundan geçerek, bitlerinin sırası değiştirilmiş bir hali elde edilir. Bunu hem permütasyon hem de yerinde koyma fonksiyonlarını içeren bu fonksiyonunun 16 kez tekrar edilmesi izler. Sonuncu (16.) tekrar, giriş değerleri olan düzyazı ve anahtarın 64-bitlik bir sentezidir. Bu çıktı sol ve sağ olmak üzere iki parçaya ayrılır ve ön çıktı adını alır. Son olarak, ön çıktı başlangıç permütasyon fonksiyonunun tersi olan (IP^{-1}) işlemine tabi tutulur ve 64-bitlik şifreli metin oluşur.

Şeklin sağ bölümünde ise 56-bitlik anahtarın işlenmesi gösteriliyor. Başlangıç olarak anahtar, bir permütasyon fonksiyonuna gönderilir. Sonra her bir 16 tekrarı için, sol kaydırma döngüsü ve permütasyonu içeren işlem sonucu bir alt anahtar (K_i) üretilir. Permütasyon fonksiyonu her tekrar için aynıdır ancak anahtar bitlerinin tekrarlı kayması sonucu her seferinde farklı bir alt anahtar üretilir.

2.1.4.2 Başlangıç Permütasyonu

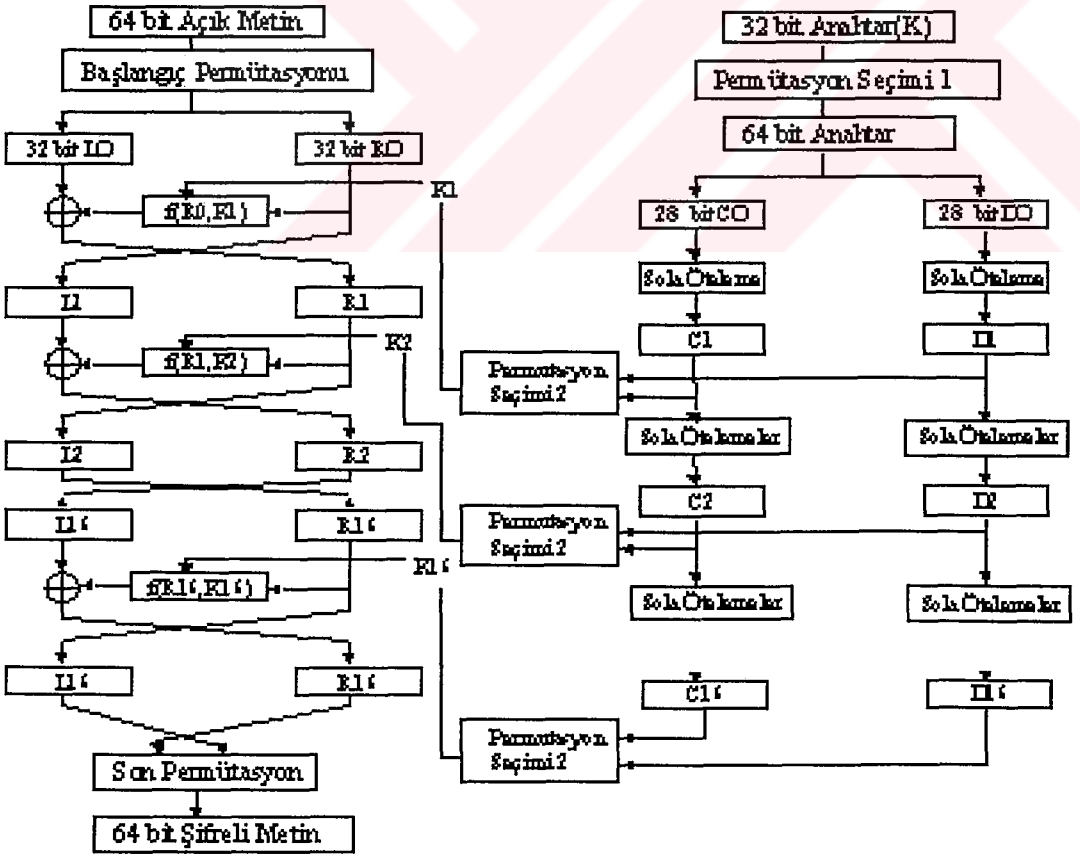
Başlangıç permütasyonu ve tersi Tablo 2.5a ve 2.5b'de sırasıyla verilmiştir. Bu iki permütasyon fonksiyonunun birbirinin tersi olduğunu görmek açısından şöyle bir 64-bitlik M'yi giriş olarak alalım:

M_1 M_2 M_3 M_4 M_5 M_6 M_7 M_8 M_9 M_{10} M_{11} M_{12} M_{13} M_{14} M_{15} M_{16}
M_{17} M_{18} M_{19} M_{20} M_{21} M_{22} M_{23} M_{24} M_{25} M_{26} M_{27} M_{28} M_{29} M_{30} M_{31} M_{32}
M_{33} M_{34} M_{35} M_{36} M_{37} M_{38} M_{39} M_{40} M_{41} M_{42} M_{43} M_{44} M_{45} M_{46} M_{47} M_{48}
M_{49} M_{50} M_{51} M_{52} M_{53} M_{54} M_{55} M_{56} M_{57} M_{58} M_{59} M_{60} M_{61} M_{62} M_{63} M_{64}

Burada M_i'ler ikili basamaklardır. O halde permütasyon X = IP(M) şu şekilde olur:

M_{58} M_{50} M_{42} M_{34} M_{26} M_{18} M_{10} M_2 M_{60} M_{52} M_{44} M_{36} M_{28} M_{20} M_{12} M_4
M_{62} M_{54} M_{46} M_{38} M_{30} M_{22} M_{14} M_6 M_{64} M_{56} M_{48} M_{40} M_{32} M_{24} M_{16} M_8
M_{57} M_{49} M_{41} M_{33} M_{25} M_{17} M_9 M_1 M_{59} M_{51} M_{43} M_{35} M_{27} M_{19} M_{11} M_3
M_{61} M_{53} M_{45} M_{37} M_{29} M_{21} M_{13} M_5 M_{63} M_{55} M_{47} M_{39} M_{31} M_{23} M_{15} M_7

Bu permütasyonun tersini alırsak yani $Y = IP^{-1}(IP(M))$ işlemi sonucu orijinal sıralamanın elde edildiğini görebiliriz.



Şekil 2.5. DES Algoritmasının Genel Bir Gösterimi

TABLO 2.5(a) **DES için Kullanılan Permütasyon Tabloları**

(a) Başlangıç Permütasyonu (IP)

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

TABLO 2.5(b) Başlangıç Permütasyonunun Tersi (IP^{-1})

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

(b) Genişleme Permütasyonu (E)

Çıktı biti	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Girdi bitinden	32	1	2	3	4	5	4	5	6	7	8	9	8	9	10	11
Çıktı biti	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
Girdi bitinden	12	13	12	13	14	15	16	17	16	17	18	19	20	21	20	21
Çıktı biti	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48
Girdi bitinden	22	23	24	25	24	25	26	27	28	29	28	29	30	31	32	1

(c)Permütasyon Fonksiyonu (P)

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

2.1.4.3 Bir Tek Döngünün Ayrıntıları

Şimdi, Şekil 1.9'da gösterilen bir döngünün daha yakından incelemesini yapalım. Yine şeklin sol tarafını ele alarak başlayalım.

Esas olarak, yer değiştirmiş 64-bitlik giriş, 16 tekrardan geçerek her tekrar sonucu 64-bitlik orta seviyeye sahip bir değer üretir. Her 64-bitlik orta seviyeli değer, sol ve sağ olarak ayrılır ve 32-bitlik bu parçalar L(Left) ve R(Right) olarak adlandırılır.

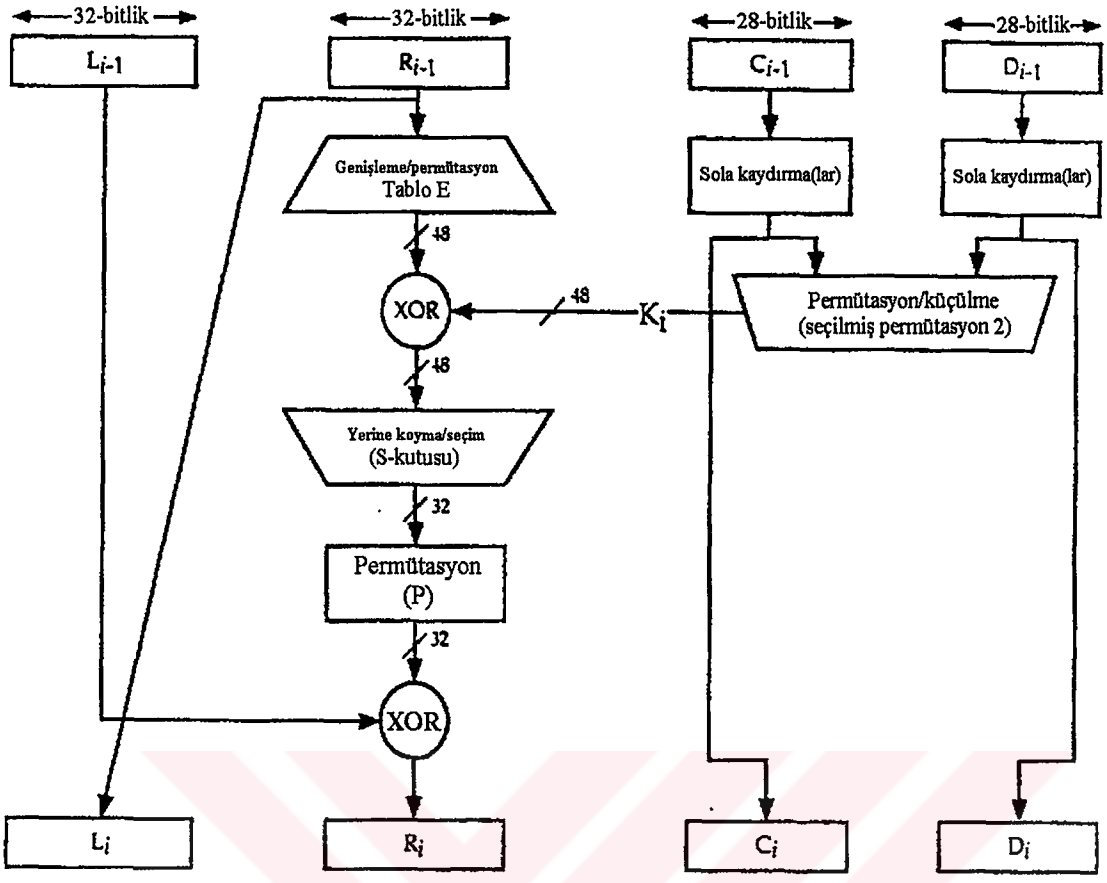
Her bir tekrarın tüm işlemleri şu şekilde şu formüllerle özetlenebilir:

$$L_i = R_{i-1}$$

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

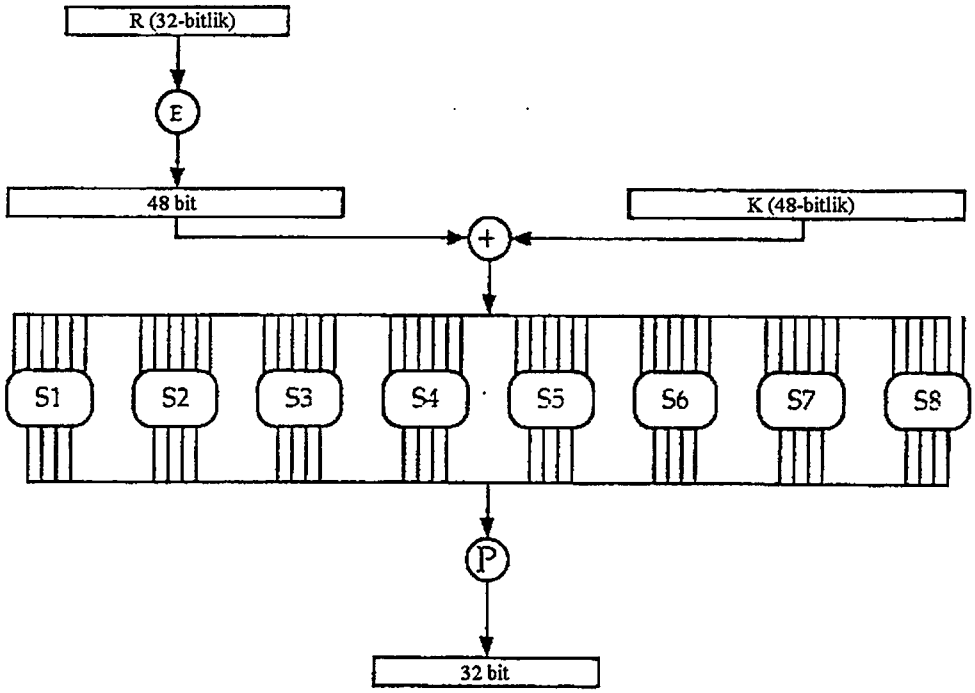
Burada $\oplus$, bit işlemi XOR dur.

Buna bakarak, bir tekrarın sol çıktısının (L_i), basitçe sağ girdi (R_{i-1}) olduğu söylenebilir. Sağ çıktı (R_i) ise, L_{i-1} ve karmaşık f fonksiyonunun özel veyalanması sonucu elde edilir. f fonksiyonu Şekil 2.7'da canlandırılmıştır. Tekrar anahtarı K_i , 48 bittir. R girişi ise 32 bittir. Bu R girişi önce Tablo 2.5c'de verilen işlem sonucu 48 bite genişletilir. Bu genişletme işlemi, bir permütasyon ve R bitlerinin 16 tanesinin tekrarı ile genişletme işlemlerini kapsar. Sonuçta oluşan 48 bit, R_i ile özel veyalanır. Bu 48-bitlik sonuç, 32-bitlik bir çıkış üreten bir yerine koyma fonksiyonuna iletilir. Bu fonksiyon Tablo 2.5d'de verilmiştir.



Şekil 2.6 DES Algoritmasının Bir Turu

Yerine koyma işlemi, giriş olarak 6 bit alan ve çıkışta 4 bit üreten 8 adet S-kutusunun bir kümesini içerir. Bu dönüşümler Tablo 2.9'da gösterilmiştir ve şu şekilde ifade edilebilir: S_i kutusuna girdi olan değerin ilk ve son basamakları 2-bitlik ikili bir sayı oluşturur ve bu sayı S_i kutusunun içindeki tabloda hangi satırın alınacağını belirler.



ŞEKİL 2.7 f(R,K)'nın Hesaplanması

Geriye kalan ortadaki 4 bit ise sütun numarasını oluşturur. Bu satır ve sütunun kesişimindeki hücrede yer alan onluk değer, çıkış oluşturmak için 4-bitlik gösterimine dönüştürülür. Örneğin S_1 için girdi 011011 olduğu taktirde, satır 01 (yani 1.satır)'dır. Sütun ise 1101 (yani 13. sütun)'dur. 1. satır ve 13. sütunun kesişimindeki değer 5'tir ve bunun dönüştürülmüş hali olan çıktı 0101 olur.

Şekil 2.8, S-kutusu işleminin ayrıntılarını gösteriyor. Birinci ve sonuncu bitlerin oluşturduğu sayı satırlarca tanımlanmış dört permütasyondan birini seçiyor. Şekil, S_1 kutusundaki sıfırıncı satırın permütasyonunu göstermektedir.

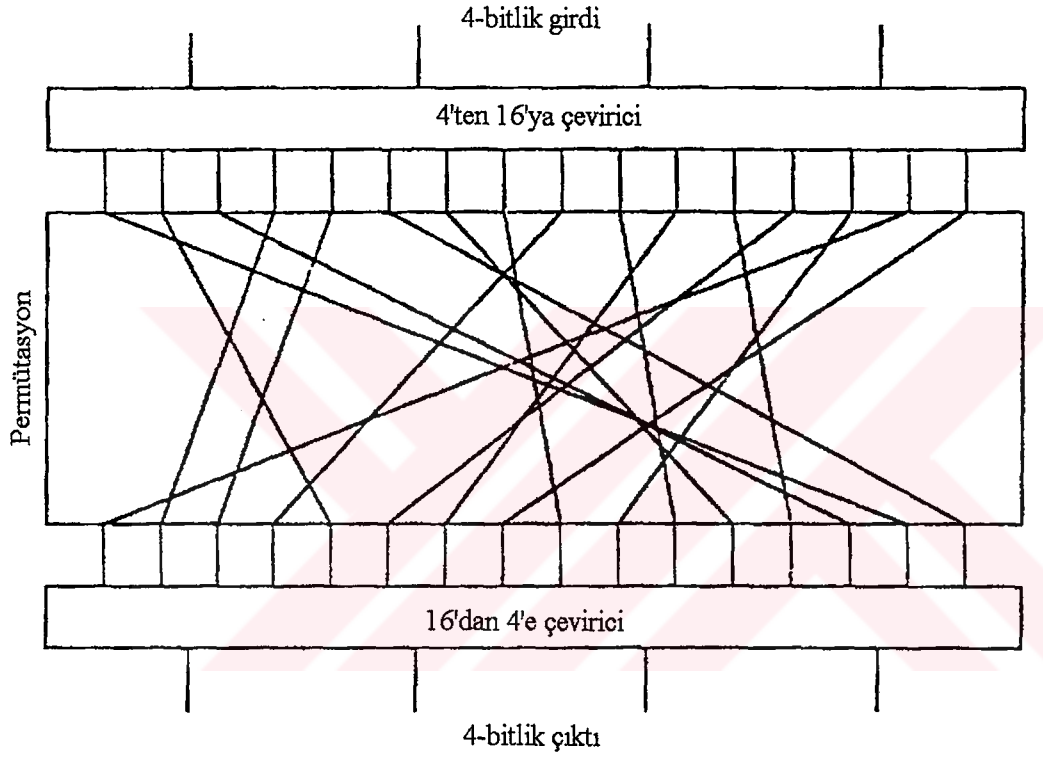
S-kutularının yapısı daha dikkatle incelenmeye değer. K_i'nin elde edilmesini bir yana bırakalım. Şimdi genişleme tablosu incelenirse 32-bitlik girdinin 4-bitlik gruplara bölündüğü ve sonra iki ardışık gruptan dıştaki bitleri alarak 6 bitlik gruplara dönüştüğü görülür. Örneğin giriş kelimesinin bir bölümü şuysa:

.... e f g h i j k l m n o p .....

Dönüşümden sonra şöyle olur:

... d e f g h i h i j k l m l m n p o q .....

Her grubun dıştaki iki biti, dört olası yerine koyma tablosundan birini seçer. Sonra 4-bitlik giriş için (ortada yer alan 4 giriş biti) , 4-bitlik bir çıktı değeri oluşturulur. Sekiz S-kutusundan çıkan 32-bit, permütasyona tabi tutulur öyle ki, bir sonraki tekrarda S-kutularının çıktısı mümkün olan en çok etkiyi sağlasın.



ŞEKİL 2.8. S-kutusunun Ayrıntıları (S_1'in 0. satırı)

2.1.4.4 Anahtarın Oluşturulması

Şekil 2.3'a tekrar bakarsak, algoritmaya girdi olarak kullanılacak 56-bitlik anahtar öncelikle "Seçilmiş Permütasyon Bir" isimli Tablo 2.10a'da tanımlanmış permütasyona tabi tutulur. Buradan elde edilen 56-bitlik anahtar iki tane 28-bitlik parça olarak ayrılır ve C_0, D_0 olarak adlandırılır. Her bir fonksiyon tekrarında, C ve D ayrık olarak Tablo 2.10c'den faydalanılarak döngülü sola kaydırma veya çevrim işlemine tabi tutulur ve 1 veya 2 bit kaydırılır.

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

TABLO 2.9 DES' teki S-kutularının Tanımları

Bu kaydırılmış değerler bir sonraki tur için girdi oluşturur. Bunlar ayrıca, "Seçilmiş Permütasyon İki" (Tablo 2.10b) için de girdi olurlar. Bu ikinci permütasyon sonucu $f(R_{i-1},K_i)$ Fonksiyonuna girdi olarak alınarak 48-bitlik bir çıktı oluşturulur.

2.1.4.5 DES Deşifrelemesi

DES' in deşifre edilme işlemi, özünde şifreleme işlemi ile aynıdır. Kural şöyle ifade edilebilir: Şifreli metin, DES algoritmasına girdi olarak kullanılır, ancak K_i anahtarları ters sırada kullanılır. Yani ilk turda K_{16} , ikinci turda K_{15} kullanılır. Son tura gelindiğinde ise kullanılacak olan anahtar, K_1 olur.

Anahtarların ters sırada kullanılmasıyla aynı algoritmanın doğru sonucu verdiğini göstermek açısından; solda yukarıdan-aşağıya şifreleme işleminin, sağda ise aşağıdan-yukarıya doğru deşifreleme işleminin yer aldığı Şekil 2.11'yi inceleyelim.

TABLO 2.10 DES Anahtarı Hesaplanmasında Kullanılan Tablolar

Seçilmiş Permütasyon Bir (PC-1)

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

Seçilmiş Permütasyon İki (PC-2)

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

Sola Kaydırma Tablosu

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

Şekil gösteriyor ki, her bir seviyede deşifrelemenin orta seviye değeri, şifrelemenin ilgili değerinin iki yarısının yer değiştirilmiş haline eşittir. Bunu bir başka şekilde göstermek istersek, i . şifrelemenin sonucunun $L_i \| R_i$ (L_i ve R_i'nin oluşturduğu değer) olduğunu düşünelim. O zaman, (16 – i) . deşifreleme seviyesinin ilgili girdi değeri $R_i \| L_i$ olur.

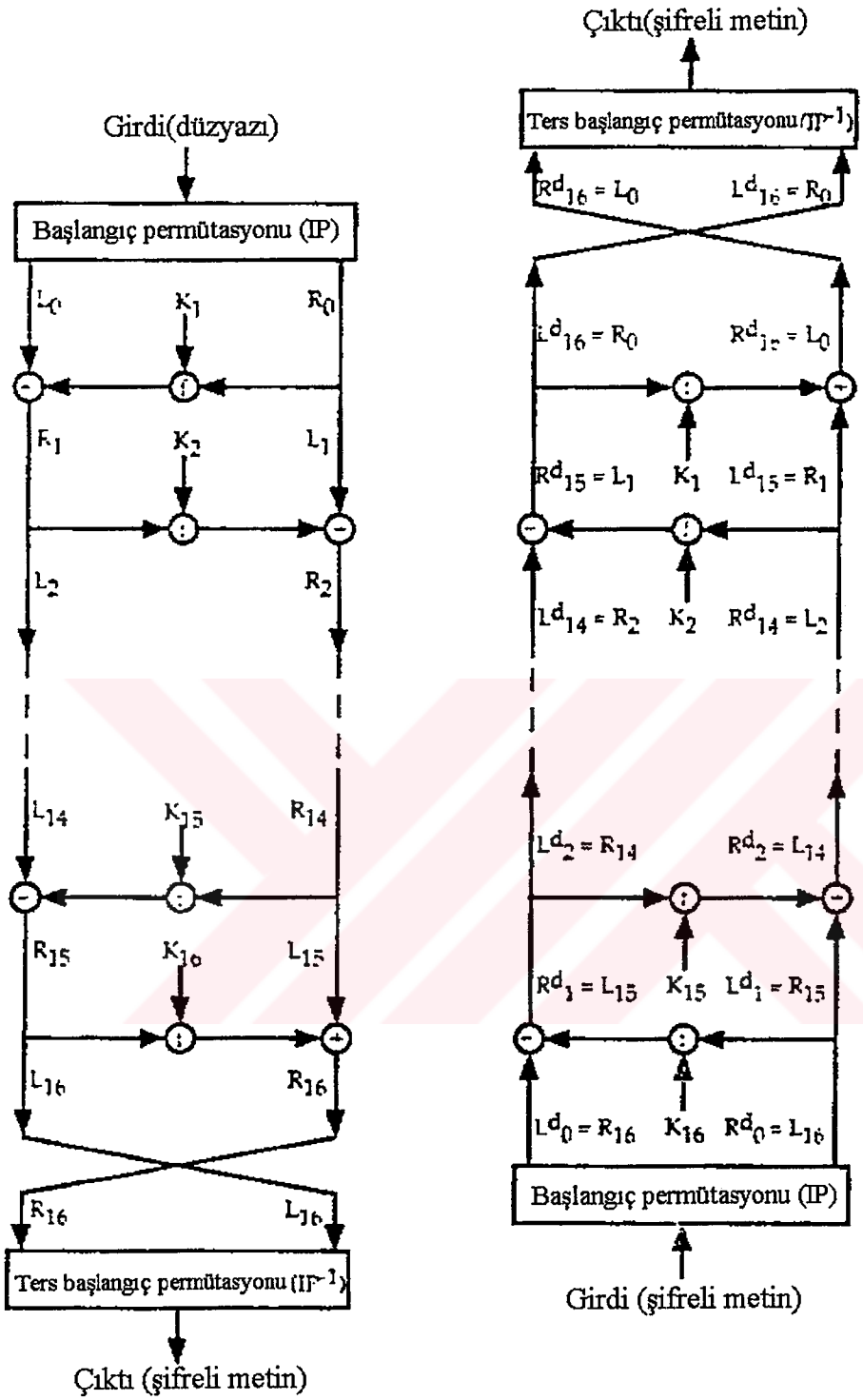
Bu saptamaların doğruluğunu göstermek için Şekil 2.11'yi inceleyelim. Şifreleme işleminin son basamağından sonra oluşan değerin iki yarısı yer değiştirilir öyle ki son IP^{-1} seviyesine girdi olarak $R_{16} \| L_{16}$ alınır. Bu adımın sonucu şifreli metindir. Şimdi şifreli metni DES algoritmasına girdi olarak alalım. İlk adım 64-bitlik bir değer olan $L^d_0 \| R^d_0$ 'ı üretmek için kullanılan IP'den (initial permütation) geçirmektir. Fakat biz zaten biliyoruz ki IP,IP^{-1}'in ters işlemidir. Bundan dolayı:

$L^d_0 \| R^d_0 = IP$ (şifreli metin)

Şifreli metin $= IP^{-1} (R_{16} \| L_{16})$

$L^d_0 \| R^d_0 = IP (IP^{-1} (R_{16} \| L_{16})) = R_{16} \| L_{16}$

Yani deşifreleme işleminin ilk adımına girdi olarak kullanılan değer, şifreleme işleminin 16 . adımında üretilen 32-bitlik yer değiştirmiş halidir.



Şekil 2.11 DES ile Şifreleme ve Deşifreleme

Şimdi de deşifreleme işleminin ilk adımının çıktısının, şifreleme işleminin 16 . adımına girdi olan değerin 32-bitlik yer değiştirmiş (takas edilmiş) hali olduğunu gösterelim. İlk olarak şifreleme işlemini düşünürsek:

$L_{16} = R_{15}$

$R_{16} = L_{15} \oplus f(R_{15}, K_{16})$ olduğunu görürüz.

Diğer taraftan deşifreleme işleminde:

$L^d_1 = R^d_0 = L_{16} = R_{15}$

$R^d_1 = L^d_0 \oplus f(R^d_0, K_{16})$

$= R_{16} \oplus f(R_{15}, K_{16})$

$= [L_{15} \oplus f(R_{15}, K_{16})] \oplus f(R_{15}, K_{16})$

eşitlikleri yazılabilir.

XOR, şu özelliklere sahiptir:

$[A \oplus B] \oplus C = A \oplus [B \oplus C]$

$D \oplus D = 0$

$E \oplus 0 = E$

Yani, $L^d_1 = R_{15}$ ve $R^d_1 = L_{15}$ eşitlikleri geçerlidir. Bundan dolayı, deşifreleme işleminin ilk adımının çıktısı, şifreleme işleminde 32-bit takas edilerek, 16 . adımda girdi olarak alınan $L_{15} \| R_{15}$ 'dir. Bu benzerlik, her 16 adım için de geçerlidir ve şekilde gösterilmiştir. Bu işlemi genel bir eşitlik olarak yazarsak, şifreleme algoritmasının i . adımı için :

$L_i = R_{i-1}$

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

Eşitliği düzenlersek:

$R_{i-1} = L_i$

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

olur.

Bununla i . adımın girdilerini, çıktıların bir fonksiyonu olarak tanımladık ve bu eşitlikler Şekil 2.11'nin sağ tarafındaki atamaların doğruluğunu gösterdi.

Son olarak, deşifreleme işleminin son adımının çıktısının $R_0 \parallel L_0$ olduğu görülmektedir. Bu, 32-bitlik bir takasa tabi tutulur (her iki yarısı yer değiştirir) ve IP^{-1} adımına $L_0 \parallel R_0$ olarak girer.

Ancak,

$$IP^{-1}(L_0 \parallel R_0) = IP^{-1}(IP(\text{düzyazı})) = \text{düzyazı}$$

olduğundan orijinal düzyazı elde edilir ve DES deşifreleme işleminin geçerliliği ispatlanmış olur.

2.1.4.6 56-Bitlik Anahtarları Kullanılması

56-bit uzunluğunda bir anahtar kullanılarak 2^{56} olası anahtar (yaklaşık 7.2×10^{16}) ortaya çıkabilir. Bu sebepten, bir kaba-kuvvet saldırısı pratikte kullanışlı değildir. Anahtarların yarısının deneneceği düşünülürse, bir DES şifrelemesini bir mikrosaniye de yapan tek bir makinenin şifreyi çözmesi 1,000 yıldan fazla sürer (Tablo 2.12).

Bununla birlikte, bir mikrosaniye de bir şifreleme yapılması kabulü pek doğru değildir. 1977 yılı itibariyle, Diffie ve Hellman teknolojinin gelişeceğini ve her birinin bir mikrosaniye de bir şifreleme yapabilen ve 1 milyon şifreleme cihazı içeren bir paralel makinenin inşa edilebileceğini düşünmüştür. Maliyetinin ise 1977'de, 20 milyon dolar civarında olacağını tahmin etmişlerdir.

Problemin en güncel analizi Wiener tarafından "bilinen düzyazı saldırısı" temel alınarak yapılmıştır. Bu analizde, saldırganın en azından birer tane (düzyazı,şifreli metin) çiftine sahip olduğu kabul edilir. Wiener, tasarımının ayrıntılarını şöyle ifade etmiştir:
"Şimdiye kadar DES anahtarlarının kırılmasının hızlı bir yöntemini bulmak için doğrulanamayan bir çok iddia ortaya atılmıştır. Böylesi iddialara bir yenisini eklememek için, bir anahtar arama makinesinin ayrıntıları eklerde verilmiştir. Bu ayrıntılı araştırma daha çok, böyle bir makinenin maliyeti ve DES anahtarını kırmak için gereken süre hakkında bir yaklaşımda bulunmaktadır. Böyle bir makinenin inşası için, hiçbir plan mevcut değildir."

Wiener, saniyede 50 milyon anahtar deneyebilen bir çipin tasarımındaki hesaplamaları yapmıştır. 1993 fiyatlarına göre 5,760 çipten oluşan ve maliyeti 100,000$ olan bir modül tasarlanmıştır. Bu tasarımın sonuçları şöyledir :

Anahtar Tarama Makinesi Birim Fiyatı	Tahmini Tarama Zamanı
100,000$	35 saat
1,000,000$	3.5 saat
10,000,000$	21 dakika

Bununla birlikte bir anahtar taramak için bilinen tek saldırı tüm olası anahtarları denemekle bitmez. Bilinen düzyazı mevcut değilse, analiz eden kişi düzyazıyı düzyazı olarak ayırt edebilmelidir. Eğer mesaj sadece düzyazı halinde İngilizce bir metin ise, İngilizce'yi tanıyabilen bir otomasyon yapılması gerekliliği ile birlikte sonuçlar oldukça olumlu bir yönde değişir. Eğer metin, şifrelenmeden önce sıkıştırılmışsa, tanınması çok daha zor olur. Hatta mesaj, bir metin değil de, nümerik bir dosya veya genel türde bir veri dosyası ise ve üstüne üstlük sıkıştırılmışsa otomasyon iyice zorlaşır. Bu sebeplerden, kaba-kuvvet saldırısı yaklaşımının gerçeklenmesi için, beklenen düzyazı hakkında bir parça ön bilgiye ihtiyaç duyulacaktır. Wiener'in tasarımı, DES' in güvenilirliği konusundaki yıllar boyu sürecek olan tartışmaların bir dönüş noktası ve belki de erken bir sonucudur. Bu yazı yazıldığı zamanlarda bile, kişisel ve ticari alandaki uygulamalarda DES'e güvenmek hala mantıklı gözükmektedir. Ancak, geleneksel şifrelemeye alternatif olacak çözümler arama zamanı gelmiştir. DES'in yerini alabilecek iki ümit verici aday, Üçlü(triple) DES ve IDEA olarak gösterilebilir.

TABLO 2.12 Eksiksiz Anahtar Tarama İçin Gereken Zaman

Anahtar Büyüklüğü	Alternatif Anahtar sayısı	mikrosaniye de bir şifreleme	mikrosaniye de 10^6 şifreleme
32 bit	$2^{32} = 4.3 \times 10^9$	$2^{31}\mu s = 35.8$ dakika	2.15 ms
56 bit	$2^{56} = 7.2 \times 10^{16}$	$2^{55}\mu s = 1142$ sene	10.01 saat
128 bit	$2^{128} = 3.4 \times 10^{38}$	$2^{127}\mu s = 5.4 \times 10^{24}$ sene	5.4×10^{18} sene
26 karakter (permütasyon)	$26! = 4.03 \times 10^{26}$	$2 \times 10^{26}\mu s = 6.4 \times 10^{12}$ sene	6.4×10^6 sene

2.1.4.7 DES Operasyon Modları

DES algoritması, veri güvenliği sağlamada kullanılan temel bir yapıdır. Ancak DES'i çeşitli uygulamalarda kullanmak üzere dört farklı işlem modu tanımlanmıştır Bu dört mod DES'in kullanılabileceği her türlü uygulamayı sanal olarak içermektedir. Bu modlar Tablo 2.13'da özetlenmiştir ve bölümün geri kalanında ayrıntılarıyla incelenecektir.

TABLO 2.13 DES İşlem Modları

Mod	Açıklama	Tipik Uygulamaları
Elektronik Kod Kitabı(EBC)	64-bitlik düzyazı blokları aynı anahtar kullanılarak bağımsız olarak şifrelenir.	Tek değerin güvenli olarak iletilmesi(örneğin şifreleme anahtarı)
Şifreli Blok Zincirleme(CBC)	Şifreleme algoritmasının girdisi, sonraki 64-bitlik düzyazı ile önceki 64-bitlik şifreli metnin XOR lanması ile elde edilen değerdir.	Genel amaçlı blok odaklı iletim Doğrulama
Şifre Geri Besleme(CFB)	Girdi,bir seferde J tane bit alınarak işlenir. Önceki şifreli metin,girdi olarak alınır ve geçici rasgele çıktı üretilir. Bu da ,düzyazıyla XOR lanarak sonraki şifreli metin birimi üretilir.	Genel amaçlı akış odaklı iletim Doğrulama
Çıktı Geri Besleme(OFB)	Şifreleme algoritmasına girdi olarak önceki DES çıktısını alması dışında CFB ile aynı.	Gürültülü kanal boyunca akış odaklı iletim(örneğin,uydu haberleşmesi)

Elektronik Kod Kitabı Modu (ECB)

Düzyazının 64 bit parçalar halinde işleme alındığı ve her birinin aynı anahtarla şifrelendiği bu mod, en basit DES modudur (Şekil 2.14). "Kod Kitabı" ifadesinin kullanılmasının nedeni, verilen bir anahtara karşılık her bir 64-bitlik düzyazı bloğu için tek bir şifreli metnin var olmasıdır. Yani her olası 64-bitlik düzyazı kalıbına karşılık bir kayıt barındıran devasa bir kod kitabı hayal edilebilir.

64 bitten uzun bir mesaj için yapılan işlem mesajı 64-bitlik bloklara ayırmak ve gerekirse son bloğu bölmektir. Deşifreleme işlemi, bir seferde bir blok alınarak ve hep aynı anahtar kullanılarak yapılır. Şekilde düzyazı 64-bitlik blokların dizilişinden (P_1, P_2,....., P_n) oluşur. İlgili şifreli metin blokları da (C_1, C_2,....., C_n) aynı şekilde dizilmiştir.

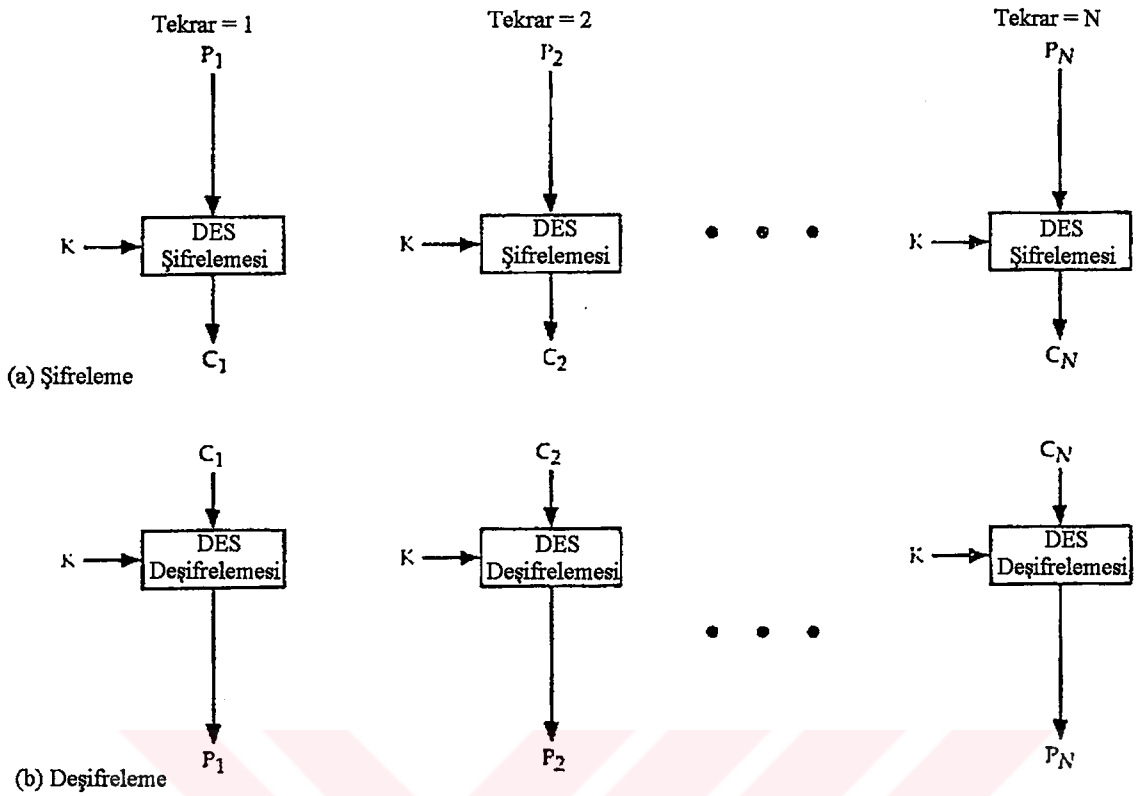
ECB metodu, az sayıdaki veriler için idealdir. Örneğin şifreleme anahtarı gibi. Demek ki bir DES anahtarını güvenli bir şekilde iletmek istersek, ECB kullanılacak en iyi moddur.

ECB modunun en göze çarpan özelliği aynı 64-bitlik düzyazı bloğunun birden fazla defa mesajda yer almasıyla, şifreli metinde de aynı şekilde ortaya çıkacağıdır.

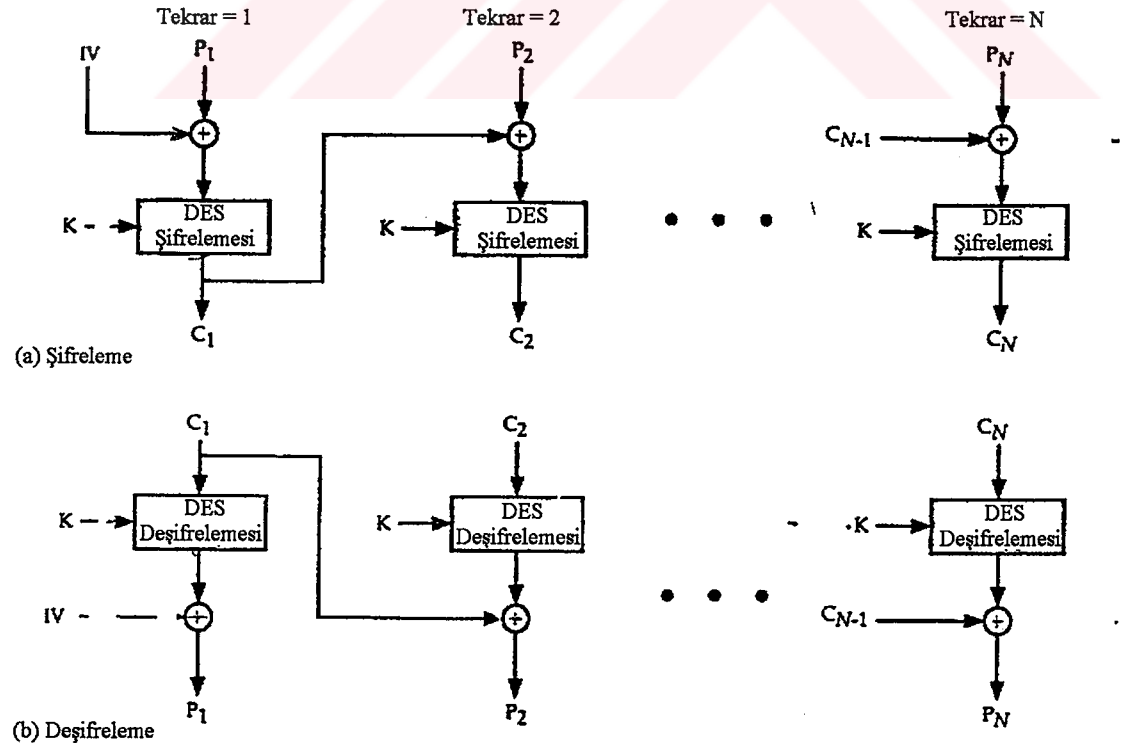
Uzun mesajlar için ECB modu güvenli olmayabilir. Eğer mesaj yüksek derecede yapısalsa, bir şifre analizcinin bu düzenlilikleri ortaya çıkarması olasıdır. Örneğin, eğer mesajın hep aynı önceden belirlenmiş kalıplarla başladığı biliniyorsa o zaman şifre analizci üzerinde çalışabileceği bir çok düzyazı/şifreli metin çiftine sahip olabilir. Bu da ECB modunu, uzun mesajlar için güvensiz yapar.

Şifreli Blok Zincirleme Modu (CBC)

ECB'deki güvenlik eksiklerini giderebilmek açısından, aynı düzyazı bloklarına karşılık farklı şifreli metin blokları üretebilen bir teknik gerekmektedir. Bunu sağlayan en basit yollardan biri CBC modudur (Şekil 2.15). Bu teknikte, şifreleme algoritmasının girdisini,o an ki düzyazı bloğu ile önceki şifreli metin bloğunun özel veya lanması oluşturur. Her bir blok için aynı anahtar kullanılır. Ardışık düzyazı blokları sanki birbirine zincirlenmiş gibidir. Şifreleme fonksiyonunun girdisi her bir düzyazı bloğu için, düzyazı bloğuyla ilişkili değildir. Bu sayede, 64 bitlik tekrarlı kalıplar kendini ele vermez.



Şekil 2.14 Elektronik Kod Kitabı (ECB) Modu



Şekil 2.15 Şifreli Blok Zincirleme (CBC) Modu

Deşifre etmek için, her bir şifreli blok deşifreleme algoritmasından geçirilir. Sonuç, önceki şifreli metin bloğu ile XORlanır ve düzyazı bloğunu oluşturur. Bunun doğruluğunu görmek için şunu yazabiliriz:

$C_n = E_k[C_{n-1} \oplus P_n]$

O zaman;

$$Dk[C_n] = Dk[E_k(C_{n-1} \oplus P_n)]$$

$$Dk[C_n] = C_{n-1} \oplus P_n$$

$$C_{n-1} \oplus Dk[C_n] = C_{n-1} \oplus C_{n-1} \oplus P_n = P_n$$

İlk şifreli metin bloğunu üretmesi için, bir başlama vektörü (IV) , düzyazının ilk bloğu ile XORlanır. Deşifreleme de; IV, deşifreleme algoritmasının çıktısı ile XORlanır ve düzyazının ilk bloğu geri alınır.

IV vektörü, hem gönderen hem de alıcı tarafından bilinmelidir. Üst düzey güvenlik için, IV vektörü en az anahtar kadar iyi korunmalıdır. Bu, IV vektörünü ECB şifrelemesini kullanarak göndermekle sağlanabilir. IV vektörünün korunmasının bir nedeni şudur: Eğer bir rakip, alıcıyı kandırarak farklı bir IV değeri kullandırırsa o zaman bu rakip düzyazının ilk bloğundaki seçtiği bitleri çevirebilir. Bunu görebilmek için, şunu düşünelim:

$$C_1 = E_k(IV \oplus P_1)$$

$$P_1 = IV \oplus Dk(C_1)$$

Şimdi X[i] gösterilişinin 64-bitlik X'in i . bitini gösterdiğini düşünelim. O halde,

$$P_1[i] = IV[i] \oplus Dk(C_1)[i]$$

O zaman, XORun özelliklerini kullanarak denilebilir ki;

$$P_1[i]' = IV[i]' \oplus Dk(C_1)[i]$$

Bu demektir ki eğer bir rakip IV içindeki bitleri bilerek değiştirebilirse alınan P_1 değerinin ilgili bitleri değiştirilebilir.

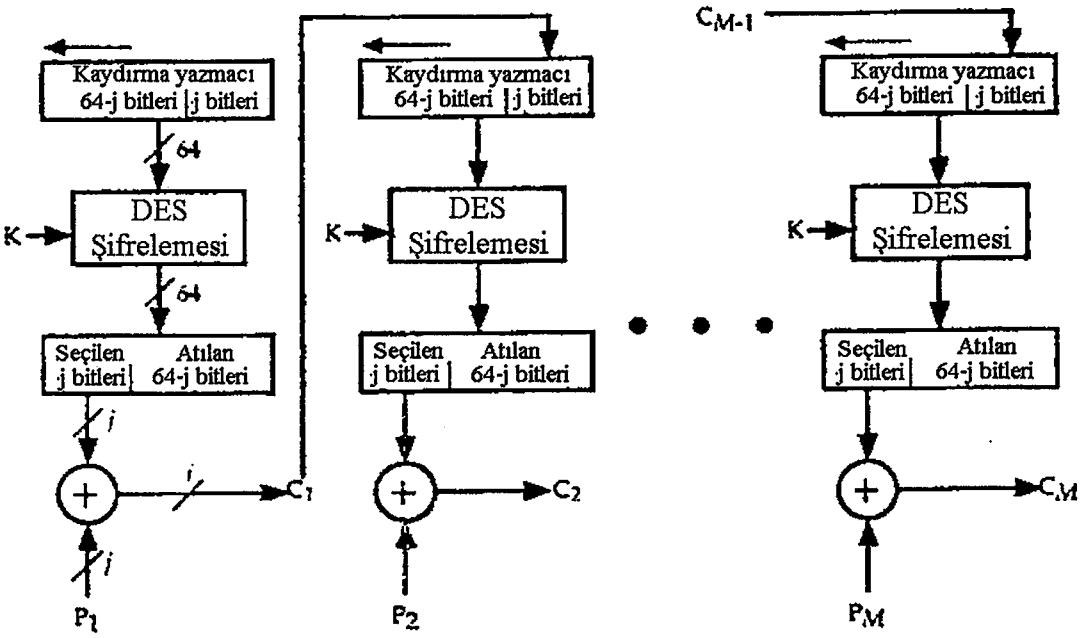
Sonuç olarak CBC'nin zincirleme mekanizmasından ötürü, 64 bitten uzun olan mesajları şifrelemede kullanılabilecek bir moddur.

Şifre Geri Besleme Modu (CFB)

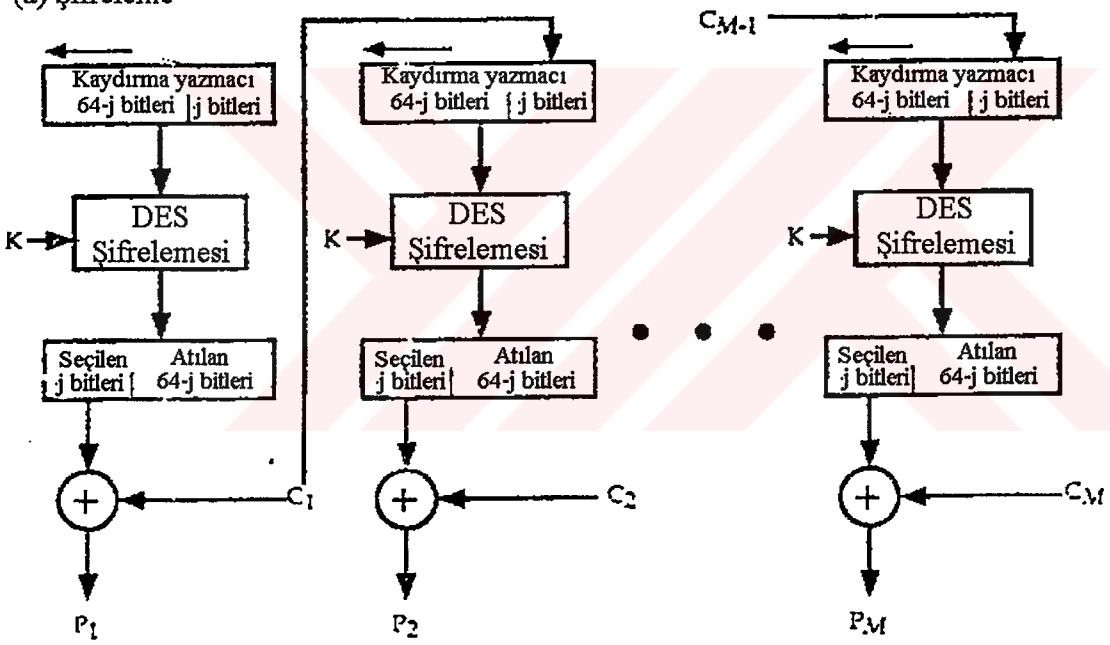
DES şifrelemesi özünde 64-bitlik blokları kullanan bir blok şifreleme tekniğidir. Bununla birlikte, DES'i bir sürekli şifrelemeye dönüştürebiliriz. Bu iki şekilde mümkündür: şifreli geri besleme ve çıktı geri beslemeli modlar. Bir sürekli şifreleme, mesajı belli bir sayıda bloğun katları uzunlukta olması için düzenleme gerekliliğini ortadan kaldırır. Ayrıca gerçek zamanlı olarak işlem yapılabilir. Yani, eğer bir karakter şeridi yollanıyorsa, her bir karakter anında karakter temelli bir şifreleme ile şifrelenerek yollanabilir.

Bir süreğen şifrelemenin ihtiyacı olan bir özellik şifreli metnin, düzyazıyla aynı uzunlukta olmasıdır. O zaman 8-bitlik karakterler gönderiliyorsa her bir karakter 8 bit kullanılarak şifrelenmelidir. Eğer 8 bitten fazlası kullanılırsa, gönderme kapasitesi ziyan edilmiş olur. Şekil 2.16, CFB tekniğini şemasal olarak göstermektedir. Şekilde, gönderilen birim verinin j bir olduğu kabul edilmiştir; ortak bir değer olarak j = 8 dir. CBC'deki gibi düzyazının birim parçaları birbirine zincirlenir öyle ki herhangi bir düzyazı biriminin şifreli metni o ana kadar olan tüm düzyazının bir fonksiyonudur.

Öncelikle şifreleme işlemini düşünelim. Şifreleme fonksiyonunun girdisi, bir başlama vektörüne(IV) atanan 64-bitlik kaydırılmış bir yazmaçtır. Çıktının en değerli j bitleri, düzyazı olan P_1'in ilk birimiyle XORlanır ve şifreli metin C_1'in ilk birimini üretir. Bu da, iletilir. Buna ek olarak, kaydırma yazmacının içeriği j bitleri ile sola kaydırılır ve C_1, kaydırma yazmacının en değersiz j bitlerine yerleştirilir. Bu işlem, tüm düzyazı birimleri şifreleninceye kadar sürdürülür.



(a) Şifreleme



(b) Deşifreleme

Şekil 2.16 J-Bitlik Şifre Geri Besleme (CFB) Modu

Deşifre etmek için aynı teknik kullanılır ancak farklı olarak; alınan şifreli metin birimi, şifreleme fonksiyonunun çıktısı ile XORlanarak düzyazı birimi üretilir. İlginç bir olay, deşifreleme değil de şifreleme fonksiyonunun kullanılmasıdır. Bu, kolayca açıklanabilir. X'in en değerli j bitlerini $S_j(X)$ olarak gösterirsek:

$$C_1 = P_1 \oplus S_j E(IV)$$

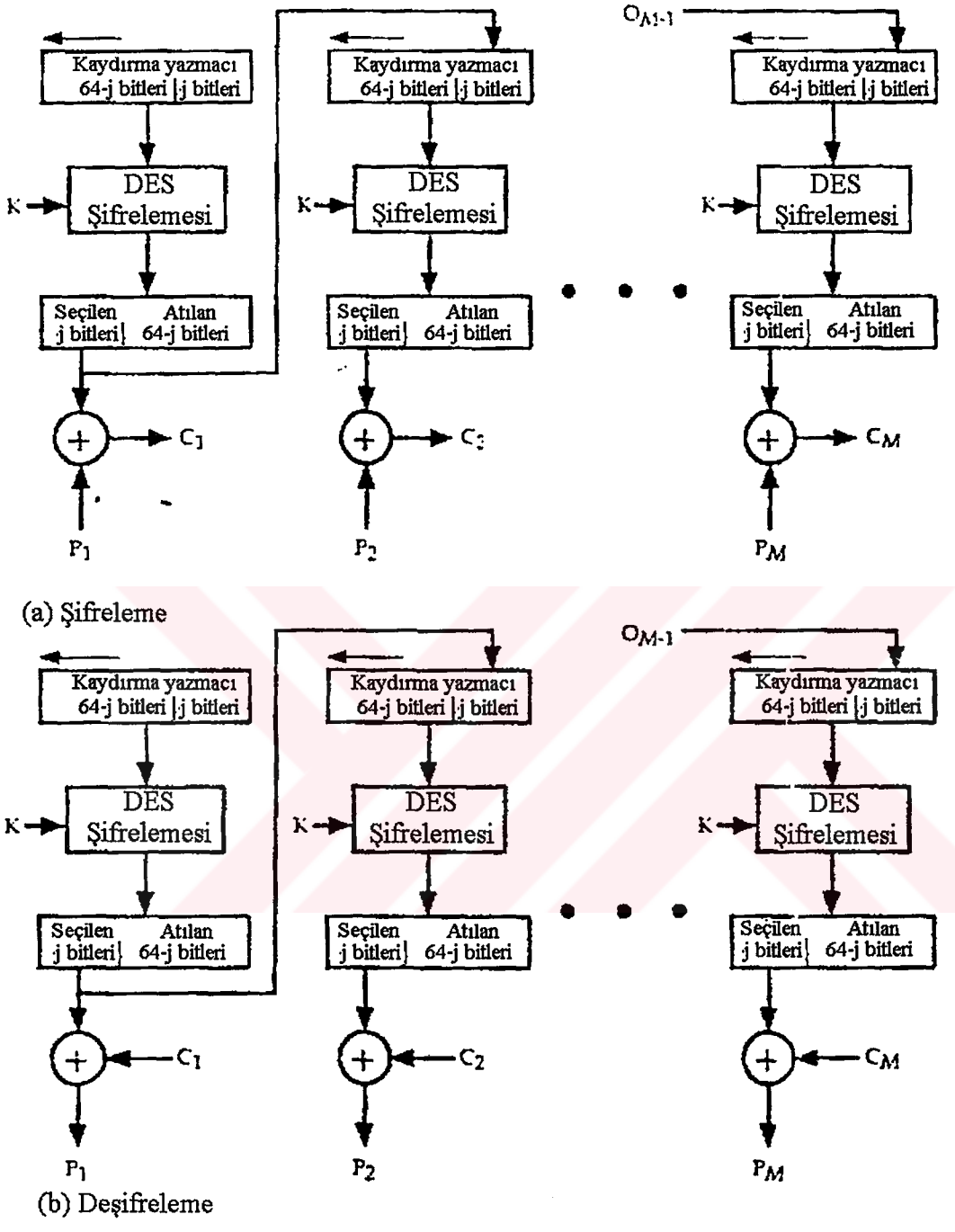
O halde,

$$P_1 = C_1 \oplus S_j E(IV)$$

Aynı mantık işlemin takip eden adımları için de geçerlidir.

Çıktı Geri Beslemeli Mod (OFB)

Çıktı geri beslemeli mod yapı olarak CFB'ye benzerdir ve şekil 2.17'de gösterilmiştir. Görüldüğü üzere, OFB'deki kaydırma yazmacına geri beslenen değer şifreleme fonksiyonunun çıktısı iken, CFB'de şifreli metin birimi, kaydırma yazmacına geri besleme yapıyordu.



Şekil 2.17 J-bitlik Çıktı Geri Beslemeli (OFB) Mod

CFB metodunun bir avantajı, iletilmiş bit hatalarının büyümemesidir. Örneğin eğer C_1'de bir bit hatası olursa sadece geri kazanılan P_1 değeri bundan etkilenecek ve geri kalan düzyazı birimleri bozulmayacaktır. CFB ile C_1 aynı zamanda kaydırma yazmacına girdi olarak da hizmet vererek ek bozulmaları da önlemiş olur.

OFB'nin dezavantajı ise bir mesaj-akış modifiyesi saldırısına CFB'den daha az dayanıklı olmasıdır. Düşünün ki şifreli metindeki bir biti değiştirmek geri kazanılan düzyazıdaki ilgili biti de değiştirmektedir. O halde, geri kazanılan düzyazı içerisinde kontrollü değişiklikler yapılabilir. Bu bir rakibin mesajdaki önemli bir kısımda gerekli değişiklikleri yaparak, şifreli metinde herhangi bir hata düzeltici kodun algılayamayacağı değişiklikleri yapmasına izin verir.

2.1.4.8 Des' in S-kutularına Yaklaşımlar

S-kutularının sahip olduğu gösterilen karakteristiklerinin rastgele (random) seçilmediği artık kabul edilmektedir. Helman ve arkadaşları DES'in S-kutuları üzerine araştırmalar yapmışlardır ve bazı şaşırtıcı sonuçlara varmışlardır. Bu sonuçlar ayrıca Standford Üniversitesinde bir rapor olarak yayınlanmıştır. Bulunan çoğu tuzaklar S4 kutusunu ilgilendirmektedir. S4 kutusunun %75'inin işe yaramaz (redundant) olduğu konusunda bazı duyumlar vardır. DES'in S-kutularındaki eylem 4 permütasyonla ifade edilir.; bir permütasyon S-kutusunun iki giriş bitinin değerine göre seçilir. S-kutusunun 4 çıkış biti, S-kutusunun iki giriş bitinin değerine göre seçilir. S-kutusunun 4 çıkış biti, S-kutusunun geri kalan girişlerinin üzerinde bu özel permütasyon hareketi ile seçilir.

Hellman ve arkadaşları, S4 kutusunun tanımında kullanılan permütasyonlardan üçünün, ilki tarafından kolaylıkla ifade edilebileceğini ortaya çıkarmışlardır. Diğer bir ilginç sonuç S-kutularının affine yaklaşıklarını içermektedir.

Bir affine dönüşüm bir lineer dönüşümden daha karmaşık olacaktır. Fakat bu karmaşıklık, bir kriptoanalistin ilgisini çekmeye yetecek kadar basittir. S4'ün affine yaklaşığı kötü değildir.

Keşfedilen diğer özellikler, S-kutusunun giriş bitlerinden biri ters çevrildiğinde en az iki çıkış bitinin ters çevrilmesidir. Bu özellik diferansiyel kriptoanalize karşı alınmış bir önlem olarak bilinmektedir.

Diğer bir aktif araştırma alanı ise (bugün de devam eden) S-kutularını tasarım kriterlerine erişmekle kolayca şifre çözmeye izin veren ve oldukça büyük bir ustalıkla gizlenmiş bazı tuzakların DES'de lma olasılıklarının araştırılmasıdır.

Shamir , S-kutularını az çok dengesiz olduğunu göstermiştir. Fakat aynı zamanda da böyle bir özelliğin kriptoanalitik saldırıya yada bir tuzağın gerçeklenmesine nasıl önayak olabileceğinin açık olmadığını da belirtmiştir.

Yıllardır biriken olaylardan anlaşılmaktadır ki çeşitli kritik kararlar ya S-kutularının tasarımı sırasında yada NSA tarafından gözden geçirilirken alınmıştır.

Hellman ve arkadaşlarının ulaştığı sonuç aşağıdaki paragrafla başlar;
"DES'te bilinen tipte saldırılara karşı sistemi güçlendirmek için yerleştirilmiş yapılar bulundu. Ayrıca sistemin zayıflığını gösteren yapılar da bulundu."

Brickell, Moore ve Purtill daha sonra açıkça bilinen tasarım kriterlerine göre S-kutularını üreterek, DES içindeki S-kutularının görünen karakteristiklerinin, tasarım kriterlerinin seçimine bağlı olabileceğini belirttiler.

DES için yapılan çalışmakar 1980'lere kadar devam etti. Davio ve arkadaşları DES'in kriptoanalizini deneyen farklı yaklaşımları gözden geçiren bir makale yayınladılar. Bu metodlar S-kutusu zayıflıklarını ele almaktadır. DES için eşdeğer formüller bularak ve hatta S-kutularının eylemi için resmi bir ifade bularak, 16 çevrimli DES boolean ifadelerin bir kümesinin analiine indiremiştir. Bu ilk önce Hellman ve arkadaşları tarafından desteklenmiş sonra Schaumller-Bichl tarafından da praik olmadığı gerekesi ile reddedilmiştir.

DES bütün bu ilk saldırılara dayanabilmiştir. Bütün bunların sonucu olarak ilginçtir ki bir çok araştırıcı Lineer Kriptoanaliz ortaya atıldıktan sonra Lineer Kriptoanalizin DES için en iyi saldırı yöntemi olduğuna karar vermişlerdir. Coppersmith'in raporunda DES'in tasarımcılarının diferansiyel kriptoanalize karşı uyanık olduklarını belirtmiştir. DES tasarımcılarının bu adımları diferansiyel bir saldırıyı geciktirmiştir. Bu adımlardan bazıları S-kutularını daha önce dikkate alınmayan özelliklerinden ve f fonksiyon çevriminde kullanılan permütasyondan kaynaklanmaktadır.

Lineer Kriptoanaliz yöntemi yaklaşık 2 yıl önce ilk 2^{43} açık mesaj –şifereli mesaj çifti biliniyorsa (aynı anahtar kullanılarak) 2^{30} anahtar uzayı taranarak DES algoritması kırılmıştır. Burada problem 2^{43} açık mesaj –şifreli mesaj çiftini (aynı anahtarla şifrelenmiş) elde edecek bilgisayarın bulunmasıdır. Bu yöntem den daha pratik bir yöntem ayrıntılı tarama (exhaustive search) dır. Çünkü bu yöntemde bir veya iki açık mesaj –şifreli mesaj çifti için 2^{56}'lık anahtar uzayını taramak yeterli olacaktır.

1976'da, National Security Agency (NSA) S-kutularının aşağıdaki özelliklerini tasarım ölçütleri olarak kabul etmiştir [11]

- Her S kutusunun her bir satırı 0'dan 15'e kadar tamsayıların bir permütasyonudur.
- Hiçbir S-kutusu girişlerinin lineer yada affine bir fonksiyonu değildir.
- Bir S – kutusunun bir giriş bitinin değişmesi en az iki çıkış bitinin değişmesine neden olur.
- Herhangi bir S-kutusu için herhangi bir x girişi S(x) ve S(x + 001100) en az iki bitte farklıdır. (Burada x, 6 bit uzunluğunda bir bit dizisidir). S kutularının aşağıdaki iki özelliği (tasarım ölçütlerini yaratan) NSA tarafından tasarlanmıştır.
- Herhangi bir S-kutusu, herhangi bir x girişi ve e, f $\in$ (0,1) için, $S(x) \neq S(x + 11ef00)$.
- Herhangi bir S kutusu için eğer herhangi bir giriş biti sabit tutulursa, 0'a eşit olan çıkış bitlerinin sayısı, 1'e eşit olan çıkış bitlerinin sayısına yakındır. Eğer birinci veya altıncı giriş biti sabit tutulursa 16 çıkış biti sıfır ve 16 tane çıkış biti bir olur. İkinci bitten beşinci bite kadar olan giriş bitleri için bu doğru değildir fakat dağılım eşit olmaya (uniform) yakındır. Daha net söylemek gerekirse herhangi bir S-kutusu için eğer herhangi bir giriş biti sabit tutulursa, 13 ile 19 arasında çıkış sıfır /bir değerini alır.

DES'in anahtar uzayı büyüklüğü 2^{56} dır. Bu güvenilirlik için çok çok küçüktür. Çeşitli özel makineler, bir ilinen açık mesaj (known plaintext) saldırısı için önerilmişlerdir. Bu makineler anahtar uzayında ayrıntılı tarama (exhustive search) gerçekleştirmek için yapılmışlardır. Yani verilen bir 64 bit açık mesaj X ve ona uygun şifreli mesaj Y için, her olası aday anahtar bir K anahtarı bulunana kadar test edilir. $E_k(X)=Y$ koşulunu sağlayan belki birden fazla K anahtarı bulunabilir.

1977'lerin başlarında Diffie ve Helmann, saniyede 10^6 anahtarı test edebilen bir VLSI chip yapılabilirse bütün anahtar uzayının 1 günde girilebileceğini ve böyle bir makinannın o zamanlar 20 milyon $'a inşaa edilebileceini öngörmüşlerdi.

CRYPTO'93 Ramp session'ında Micheal Wiener, bir anahtar tarama (key search) makinesinin ayrıntılı tasarımını vermiştir. İş hattı yapısına sahip makine, 16 şifrelemeyi eş zamanlı olarak yapabilmektedir. Bu chip saniyede $5*10^7$ anahtarı test edebilmekte ve bugünkü teknoloji ile chip başına maliyet 10.50$ olmaktadır. 5760 chip'i içeren bir çerçeve 100.000$'a inşaa edilebilir. Bu bir DES anahtarının ortalama 1.5 günde bulunmasını sağlar. 10 çerçeve kullanan bir makine 1 milyon $'a malolur fakat ortalama tarama zamanını üçbuçuk saate indirmektedir.

Bu gün donanım (hardware) gerçeklemeleri oldukça hızlı şifreleme yapabilmektedirler. Digital Equipment Corporation, CRYPTO 92'de, 250 Mhz saat hızı (clock rate) ile saniyede 1 Gbit hızında 50K transistörle şifreleme yapılabildiğini belirtmştir. Bu chip'in maliyeti 300$'dır. 1991'de DES'in 45 tane donanım ve firmware gerçeklemesi yapılmıştır ve bütün bunlar National Bureau of Standards tarafından geçerli sayılmıştır.

Bankalar DES'in önemli uygulama alanlarından biridir (Amerikan Bankacılar Birliği tarafından geliştirilen standardlar kullanılarak). DES personel kimlik numarasını şifrelemek (PIN) için kullanılır ve ATM (Automated Teller Machines) de kullanılan hesap işlerinde, hafta başına $1.5 *10^{12}$ $'ın üzerinde olan doğrulama işlemleri (authenticate transactions) için Clearing House Interbank Payment System (CHIP) de ve aynı zamanda oldukça geniş bir şekilde devlet organizasynlarında kullanılmaktadır (Amerika'da Enerji Departmanı, Adliye Departmanı ve Federal Depolama Sistemi gibi).

Diferansiyel kripto-analiz yönetimi, yinelemeli (iterated) kripto sistemleri için uygundur. DES ve DES-benzeri (FEAL, Khafre, REDOC-II, LOCI ve Lucifer) kripto-sistemleri ile Hash fonksiyonlarının analizi için kullanılabilir. Bu çalışmada sadece DES'in kripto-analizi DES'in analizine benzediğinden bir fikir vermesi amacıyla bu sistemlerin kısaca açıklanmasında yarar görülmüştür.

2.1.4.9 Des'e Ayrıntılı Bir Örnek

DES mesajları 64 bitlik bloklara parçalama (blok cipher), şeklinde işleme sokar ve çıktıyla, yani şifrelenmiş mesajla aynı uzunlukta olur. Bunun sonucu olarak ortaya 2^{64} gibi bir sayı çıkar ki bu, 64 bitin olası bütün aranjmanlarına eşittir. Her 64 bitlik blok 32'şer bitlik sağ ve sol yarıya ayrılır.

Mesajımız düz yazı ile yazılmış "KARAKUTU" olsun. Bunun ASCII tablosundaki hexadecimal karşılığı "4B4152414B555455" sayısına eşittir. Buna mesaj kelimesinin baş harfi olan **M** adını verelim ve **M** sayısını ikilik tabanda yazalım:

M=0100 1011 0100 0001 0101 0010 0100 0001 0100 1011 0101 0101 0101 0100 0101 0101

Anahtar olarak da "PORTAKAL" olsun.Bunu 16 lık düzende yazarsak

A= 504F5254414B414C

A= 0101 0000 0100 1111 0101 0010 0101 0100 0100 0001 0100 1011 0100 0001 0100 1100

ADIM 1: Her biri 48 Bit Uzunluğunda 16 Alt Anahtar Yaratma

64 bitlik anahtarımız şu tabloya göre permüte edilir (**PS-1**). Tablodaki ilk elemanın 57 olduğu görülüyor. Bunun anlamı orijinal **A** anahtarının 57. bitinin permüte edilmiş anahtarın (buna da **A+** diyelim) 1. biti olacağıdır. Orijinal anahtarın 49. biti de permüte edilmiş anahtarın 2. biti olacaktır ve bu şekilde devam eder. Orijinal anahtarın sadece 56 bitinin işleme girdiğini ve işleme giren bitlerin sekizin katı olmadığına dikkat edelim.

Değiştirme tablosu -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

Bu işlem ile orijinal anahtar oluşmuş anahtar A+ ile gösteriyor. şu şekilde olur ile gösteriyoruz.

A+ = 0000 0000 1111 1111 0000 0000 0000 0010 0110 1000 1010 1010 0010 1101

olur.

ADIM 2:Anahtarın ikiye bölünmesi(Cn ve Dn çiftlerinin oluşturulması)

Şimdiki işlem bu permüte edilmiş anahtarı iki parçaya bölmek. Bunlara C0 ve D0 adını vereceğiz:

C0 = 0000 0000 1111 1111 0000 0000 0000

D0 = 0010 0110 1000 1010 1010 0010 1101

ADIM 3:Alt anahtarların oluşturulması

Bu tanımladığımız C_0 ve D_0 ile C_n ve D_n şeklinde 16 blok yaratabiliriz (1<=n=<16). Her (C_n D_n) çiftini "sola doğru kaydırma" işlemi yapılmış bir önceki çiftten (C_{n-1} D_{n-1}) meydana getireceğiz. "Sola doğru kaydırma" yapmak için birinci bit hariç her biti bir solundaki basamağa , birinci biti de en sona geçirelim.

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

Bunun anlamı, örneğin; C_3 ve D_3 , C_2 ve D_2 iki kez sola kaydırılarak elde edilmiştir ve her n için kaydırma değerleri değişmektedir. Yukarıdaki listede bu değerler verilmiştir.

C1: 0000 0001 1111 1110 0000 0000 0000

D1: 0100 1101 0001 0101 0100 0101 1010

C2: 0000 0011 1111 1100 0000 0000 0000

D2: 1001 1010 0010 1010 1000 1011 0100

C3: 0000 1111 1111 0000 0000 0000 0000

D3: 0110 1000 1010 1010 0010 1101 0010

C4: 0011 1111 1100 0000 0000 0000 0000

D4: 1010 0010 1010 1000 1011 0100 1001

C5: 1111 1111 0000 0000 0000 0000 0000

D5: 1000 1010 1010 0010 1101 0010 0110

C6: 1111 1100 0000 0000 0000 0000 0011

D6: 0010 1010 1000 1011 0100 1001 1010

C7: 1111 0000 0000 0000 0000 0000 1111

D7: 1010 1010 0010 1101 0010 0110 1000

C8: 1100 0000 0000 0000 0000 0011 1111

D8: 1010 1000 1011 0100 1001 1010 0010

C9: 1000 0000 0000 0000 0000 0111 1111

D9: 0101 0001 0110 1001 0011 0100 0101

C10: 0000 0000 0000 0000 0001 1111 1110

D10: 0100 0101 1010 0100 1101 0001 0101

C11: 0000 0000 0000 0000 0111 1111 1000

D11: 0001 0110 1001 0011 0100 0101 010

C12: 0000 0000 0000 0001 1111 1110 0000

D12: 0101 1010 0100 1101 0001 0101 0100

C13: 0000 0000 0000 0111 1111 1000 0000

D13: 0110 1001 0011 0100 0101 0101 0001

C14: 0000 0000 0001 1111 1110 0000 0000

D14: 1010 0100 1101 0001 0101 0100 0101

C15: 0000 0000 0111 1111 1000 0000 0000

D15: 1001 0011 0100 0101 0101 0001 0110

C16: 0000 0000 1111 1111 0000 0000 0000

D16: 0010 0110 1000 1010 1010 0010 1101

elde edilir.

Bundan sonra amacımız olan alt anahtarları (A_n) oluşturacağız. Bunun için C_n D_n çiftlerini başlangıçtakine benzer şekilde aşağıdaki tabloya göre permüte edeceğiz. Yine dikkat edin $C_n D_n$ çiftleri 56 bittir ancak permütasyon tablomuz bunun 48 tanesini kullanır.

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

A1 = 1 0 1 0 0 0 0 0 1 0 0 1 0 0 1 0 0 1 0 0 0 1 0 0 0 1 1 1 0 0 1 0 0 0 1 1 1 1 1 1 0 0

Yani C_1D_1 çiftinin 14. biti A_1'in ilk biti olacaktır. Aynı şekilde A_1'in ikinci biti C_1D_1'in 17. bitine eşit olur.

A2 = 1 0 1 0 0 0 0 0 0 0 0 1 0 0 1 0 0 1 0 1 0 0 1 0 1 1 0 0 0 0 0 0 1 1 1 0 0 1 1 0 0 0 0 0 0 1 1

A3 = 0 0 1 0 0 1 0 0 0 1 0 1 0 0 1 0 0 1 0 1 0 0 0 0 1 1 1 1 1 1 1 0 0 0 1 0 0 1 1 0 0 0 0 1 0 0 0

A4 = 0 0 0 0 0 1 1 0 0 1 0 1 0 0 0 1 0 1 0 1 0 0 0 0 1 0 1 1 1 0 0 0 0 1 0 1 0 0 1 1 0 1 0 0 1 0 1 0

A5 = 0 0 0 0 1 1 1 0 0 1 0 0 0 0 0 1 0 1 0 1 0 0 0 1 0 0 0 1 0 1 0 0 0 1 1 1 1 0 0 1 0 0 0 1 0 0 0 1 0

A6 = 0 0 0 0 1 1 1 1 0 1 0 0 0 0 0 1 0 0 0 0 1 0 0 1 1 1 1 0 1 0 0 0 0 1 0 0 1 1 0 0 0 1 1 0 0 0 0 0

A7 = 0 0 0 0 1 0 1 1 0 0 0 0 0 0 0 1 1 0 0 0 1 0 0 1 1 0 1 0 1 0 0 0 1 0 1 0 1 0 1 0 1 0 1 0 1 1 0 1 0

A8 = 0 0 0 1 1 0 0 1 0 0 0 0 1 0 0 0 1 0 0 0 1 0 0 1 0 0 1 1 0 1 0 1 1 1 1 0 1 1 0 0 0 0 1 0 0 1 0

A9 = 0 0 0 1 1 0 0 1 0 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 1 0 0 0 0 1 1 0 1 1 0 0 0 0 1 1 0 1

A10 = 0 0 0 1 0 0 0 0 0 0 1 0 1 0 0 0 1 0 0 0 1 1 0 0 0 1 0 1 0 0 1 0 0 1 0 1 0 0 1 1 1 0 1 1 0 1 0 0

A11 = 0 0 0 1 0 0 0 0 0 0 1 0 1 1 0 0 0 0 0 0 1 0 0 0 0 1 0 1 0 0 0 1 0 0 0 0 1 0 0 1 1 0 1 0 1 0 0 1

A12 = 0 1 0 0 0 0 0 0 0 1 0 1 1 0 0 0 0 1 0 0 1 0 0 1 1 0 0 0 0 1 0 0 1 1 0 0 0 0 1 1 0 0 1

A13 = 1 1 0 0 0 0 0 0 1 0 1 0 0 1 0 0 0 0 1 0 0 1 0 1 0 0 0 1 1 0 0 0 1 1 0 0 1 1 0 0 1 1 1 1 0 0

A14 = 1 1 0 0 0 0 0 0 1 0 0 0 0 1 1 0 0 0 1 0 0 0 1 0 0 0 1 1 0 0 0 1 0 0 0 1 1 0 0 1 0 1 0 1 0

A15 = 1 1 1 0 0 0 0 0 1 0 0 1 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 1 0 0 0 0 1 0 0 0 0 1 1 0 1 1 1

A16 = 1 0 1 0 0 0 0 0 1 0 0 1 0 0 1 0 0 0 1 0 0 0 1 0 1 1 1 0 0 0 1 1 0 0 0 0 1 0 1 1 0 0 0 0 0

ADIM 4: Her 64 Bitlik Mesajın Kodlanması

DES işleminin ikinci temel elemanı olan ilk permütasyon sayısına **(İP)** adını verelim. Bu sayı orijinal mesaj bloğunun ilk permütasyon tablosuna göre permüte edilmesiyle elde edilir. İlk permütasyon tablosu aşağıda görülmektedir.

İlk Permütasyon (İP)

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

M=0100 1011 0100 0001 0101 0010 0100 0001 0100 1011 0101 0101 0101 0100 0101 0101

İlk permütasyonu 64 bitlik **M** mesajı üzerinde uyguladığımızda;

İP=1111 1111 1110 0100 1110 0000 1011 1011 0000 0000 0000 0000 0001 0001 0001 0101

sayısını elde ederiz.

Şimdi bu elde ettiğimiz **İP** sayısını 32'şer bitlik sağ ve sol olmak üzere iki parçaya ayıralım, bunlara da $\mathbf{R_n}$ ve $\mathbf{L_n}$ diyelim. (**L**eft ve **R**ight)

$\mathbf{L_0}$ = 1111 1111 1110 0100 1110 0000 1011 1011
$\mathbf{R_0}$ = 0000 0000 0000 0000 0001 0001 0001 0101

ADIM 5:F fonksiyonu

Şimdiki işlemimiz oldukça komplike, 1'den 16'ya kadar $\mathbf{L_nR_n}$ çiftlerini hesaplayacağız. Bunu yaparken *f* fonksiyonunu kullanacağız. *f* fonksiyonu, 32 bitlik bir bilgi ve 48 bitlik bir anahtarı ($\mathbf{A_n}$) işleme sokar. Bu fonksiyonu açıklamadan önce formülümüzü gözden geçirelim. Formüldeki + bildiğimiz toplama işlemini gösteriyor.

$\mathbf{L_n = R_{n-1}}$
$\mathbf{R_n = L_{n-1} + f(R_{n-1},K_n)}$

Örneğin L_1' i hesaplayalım. Formülümüzden L_1'in R_0'a eşit olduğunu kolayca görebiliriz. Peki ya R_1. Şimdi *f* fonksiyonunu incelemenin tam vakti. *f* fonksiyonunu hesaplamak için R_0 sayısını 32 bitten 48 bite çıkarmalıyız.

Bunu yapmak için E adı verdiğimiz bir tablo işleminden faydalanırız. Bu tablonun tek yaptığı şey 32 bitlik bloktaki bazı bitleri tekrar kullanmaktır. Böylece E işleminin giren bit sayısı 32, işlemden çıkan bit sayısı 48 olacaktır.

Aşağıda görülen E tablosu oldukça basittir.

E-TABLOSU

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

Böylece $\mathbf{R_0}$ dan $\mathbf{E(R_0)}$ ı hesaplayabiliriz.

$\mathbf{R_0}$ = 0000 0000 0000 0000 0001 0001 0001 0101

$\mathbf{E(R_0)}$ = 1000 0000 0000 0000 0000 0000 0000 1010 0010 1000 1010 1010

Şimdi elimizde 48 bit var. ***f*** 'i hesaplamaya devam edelim. $\mathbf{E(R_{n-1})}$ bloğunu $\mathbf{A_n}$ anahtarıyla toplama işlemine sokal

A1 = 1010 0000 1001 0010 0100 0010 0100 0111 0010 1000 1111 1100

$\mathbf{A1 + E(R_0)}$ = 0010 0000 1001 0010 0100 0010 0100 1101 0000 0000 0101 0110

Henüz ***f*** fonksiyonunu hesaplamayı bitirmedik, bu noktaya kadar sadece **E** tablosuna göre $\mathbf{R_0}$'ı genişlettik ve $\mathbf{A_1}$ anahtarıyla topladık. Şimdi gerçekten çok ilgi çekici bir işlem yapacağız. (bu algoritmayı öğrendiğimde beni hayrete düşürmüştü) Elimizde 48 bit var yani 8 tane 6 bitlik grup. Bu 6'lı grupları **S** adını verdiğimiz tablolarda adresler olarak kullanacağız. Her 6'lı grup bize farklı **S** tablolarındaki adresleri verecek. Bu adreslerde 4 bit sayı yer alacak ve bu 4 bitlik sayılar 6 bitlik sayıların yerine geçerek 8 tane 4 bit yani 32 bitlik bloğu oluşturacak.

Ve 6'lı gruplara B harfini vererek bunu farklı bir biçimde yazalım

$\mathbf{A_1 + E(R_0) = B_1B_2B_3B_4B_5B_6B_7B_8}$ Ve bahsettiğimiz işleme sokalım, yani **S** tablolarına

$\mathbf{S_1(B_1)S_2(B_2)S_3(B_3)S_4(B_4)S_5(B_5)S_6(B_6)S_7(B_7)S_8(B_8)}$

Burada **Sn(Bn)** , n. **S** tablosu işleminden çıkan anlamındadır.

S_1 Tablosu;

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

S tablosu işlemi şu şekilde gerçekleşir. 6 bitlik grubumuzdaki ilk ve son 2 bit 0'dan 3'e kadar bir sayı olabilir. Bu sayıya x diyelim. 6'lık grubumuzda ortada kalan 4 bit ise 0 ile 15 aralığında olan bir sayıdır. Buna da y diyelim. Tablomuzda x ve y'nin kesiştiği yerdeki sayı 4 bit değerinde bir sayıdır ve **S** tablosu işleminin sonucudur.

Örneğin; $\mathbf{S_1(B_1)}$' i bulalım. $\mathbf{B_1}$ 001000 sayı grubuna eşittir. İlk ve son bitleri yani 00 sayısıdır. Bu bize tablodaki satır sayımızı verir (0). Ortadaki 4 bit yani 0100 bize sütun sayımızı verir (4). Tabloda 0. satır ve 4. sütunun kesişiminde görülen sayı "2"dir. Yani işlemimizin sonucu 0010 sayısıdır.

$\mathbf{S_1}(001000) = 0010$

Bu şekilde işleme sokmamız gereken 7 tane 6'lı grup ve 7 tane **S** tablosu daha vardır. Bunu 7 kere daha tekrar edersek;

S_1

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

S_2

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

S_3

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

S_6

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

$S_1(B_1)S_2(B_2)S_3(B_3)S_4(B_4)S_5(B_5)S_6(B_6)S_7(B_7)S_8(B_8) =$ 0010 1111 0011 1101 0000 0000 1101 1110 0000 0000 0000 0000 bloğuna ulaşırız. Bundan sonra ***f*** 'yi bulmak için tek yapmamız gereken elimizdeki sonucu son bir permütasyon tablosuna göre işleme sokmaktır. **P** tablosu 32 bitlik bloğu işleme sokar ve işleme giren bloğun bitlerini permüte ederek yine 32 bitlik bir blok oluşturur. $f = P(S_1(B_1)S_2(B_2)...S_8(B_8))$

P TABLOSU

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

f = 1100 1110 0001 1010 0101 0010 0111 0101

ADIM 7: İterasyonla Rn'lerin oluşturulması

Sonunda **f** ' yi bulduk şimdi tekrar başlangıca dönebiliriz. Formülümüzü hatırlayalım $\mathbf{R_1 = L_0 + f(R_0 , K_1)}$. O zaman toplama işlemini de yaparak R_1' i yani sağ taraftaki 32 biti bulabiliriz.

$\mathbf{R_1 = L_0 + f(R_0 , K_1)}$

= 1111 1111 1110 0100 1110 0000 1011 1011 $(\mathbf{L_0})$

(+) 1100 1110 0001 1010 0101 0010 0111 0101 $(\mathbf{f(R_0 , K_1)})$

= 0011 0001 1111 1110 1011 0010 1100 1110

Böylece sağ taraftaki 1 indisli 32 biti de bulmuş olduk. Bütün bu işlemleri 1 den 16 ya kadar tekrarlarsak $\mathbf{L_{16}R_{16}}$ çiftine ulaşırız.

L_{16} = 1101 1110 0000 0011 1100 0100 0000 1000
R_{16} = 0110 0111 1000 0010 0111 0000 0110 0100

Bu ulaştığımız sayı bloğunu ters çevirdikten sonra **($R_{16}L_{16}$)** son bir tablo işlemine sokalım. Bu son tablomuza **ST** adını verelim.

ADIM 8 : Şifrelenmiş mesajın elde edilmesi

$R_{16}L_{16}$ = 0110 0111 1000 0010 0111 0000 0110 0100 1101 1110 0000 0011 1100 0100 0000 1000

ST Tablosu

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

ST = 0 1 1 0 0 0 0 0 1 1 1 1 0 0 0 0 1 1 0 0 1 0 0 1 1 0 0 0 0 0 1 0 1 0 0 0 0 1 0 0 0 1 0 0 0 0 1 0 0 0 1 1 0 0 1 1 0 1 1 0 0 1 1 0 0 0

Mesajımızın şifrelenmesi bitti. Bunu hexadecimal sistemde yazalım.

M= "KARAKUTU" mesajımızın kript edilmiş hali

Ş = 60F0C9828445CD98

2.1.4.10 Des Programı

```
#include <stdio.h>

static int keyout[17][48];

void des_init(),lshift(),cypher(),des_encrypt(),des_descrypt();

void des_init(unsigned char *key){
 unsigned char c[28],d[28];
 static int pc1[56] = {57,49,41,33,25,17,9,
                       01,58,50,42,34,26,18,
                       10,02,59,51,43,35,27,
                       19,11,03,60,52,44,36,
                       63,55,47,39,31,23,15,
                       07,62,54,46,38,30,22,
                       14,06,61,53,45,37,29,
                       21,13,05,28,20,12,04};
 static int pc2[48] = {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};
 static int nls[17] = {
  0,1,1,2,2,2,2,2,2,1,2,2,2,2,2,2,1};

 static int cd[56],keyb[64];
 static int cnt,n=0;
 register int i,j;

 for(i=0;i<8;i++) /*Read in key*/
  for(j=0;j<8;j++) keyb[n++]=(key[i]>>j&0x01);

 for(i=0;i<56;i++) /*Permuted choice 1*/
  cd[i]=keyb[pc1[1]-1];
 for(i=0;i<28;i++){
  c[i]=cd[i];
  d[i]=cd[i+28];
 }
 for(cnt=1;cnt<=16;cnt++){
  for(i=0;i<nls[cnt];i++){
   lshift(c); lshift(d);
  }
  for(i=0;i<28;i++){
   cd[i]=c[i];
```

```
  cd[i+28]=d[i];
 }
 for(i=0;i<48;i++) /*Permuted Choice 2*/
  keyout[cnt][i]=cd[pc2[i]-1];
}
}

static void lshift(unsigned char shft[]){
 register int temp,i;

 temp=shft[0];
 for(i=0;i<27;i++) shft[i]=shft[i+1];
 shft[27]=temp;
}

static void cypher(int *r, int cnt, int *fout){
 static int expand[48],b[8][6],sout[8],pin[48];
 register int i,j;
 static int n,row,col,scnt;
 static int p[32]={
         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};

 static int   e[48] = {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};

 static char s[8][64] = {
         14,4,13,1,2,15,11,8,3,10,6,12,5,9,0,7, /*s1*/
          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,
         15,1,8,14,6,11,3,4,9,7,2,13,12,0,5,10, /*s2*/
          3,13,4,7,15,2,8,14,12,0,1,10,6,9,11,5,
          0,14,7,11,10,4,13,1,5,8,12,6,9,3,2,15,
         13,8,10,1,3,15,4,2,11,6,7,12,0,5,14,9,
         10,0,9,14,6,3,15,5,1,13,12,7,11,4,2,8, /*s3*/
         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,
          7,13,14,3,0,6,9,10,1,2,8,5,11,12,4,15,/*s4*/
         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,
```

```
        2,12,4,1,7,10,11,6,8,5,3,15,13,0,14,9,/*s5*/
       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,
       12,1,10,15,9,2,6,8,0,13,3,4,14,7,5,11, /*s6*/
       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,
        4,11,2,14,15,0,8,13,3,12,9,7,5,10,6,1,/*s7*/
       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,
       13,2,8,4,6,15,11,1,10,9,3,14,5,0,12,7, /*s8*/
        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
};

for(i=0;i<48;i++) expand[i]=r[e[i]-1]; /*Expansion Function*/
for(i=n=0;i<8;i++) {
 for(j=0;j<6;j++,n++) b[i][j]=expand[n]^keyout[cnt][n];
}

/*Selection functions*/

 for(scnt=n=0;scnt<8;scnt++){
  row=(b[scnt][0]<<1)+b[scnt][5];
  col=(b[scnt][1]<<3)+(b[scnt][2]<<2)+(b[scnt][3]<<1)+b[scnt][4];
  sout[scnt]=s[scnt][(row<<4)+col];
  for(i=3;i>=0;i--){
   pin[n]=sout[scnt]>>i;
   sout[scnt]=sout[scnt]-(pin[n++]<<i);
  }
 }
 for(i=0;i<32;i++) fout[i]=pin[p[i]-1]; /*Permutation Function*/
}

static int p[64] = {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};

static int invp[64]={
 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};

void des_encrypt(unsigned char *input){
 static unsigned char out[64];
 static int inputb[64],lr[64],l[32],r[32];
 static int fn[32];
 static int cnt,n;
 register int i,j;

 for(i=n=0;i<8;i++)
  for(j=0;j<8;j++) inputb[n++]=(input[i]>>j&0x01);

 for(i=0;i<64;i++){ /*Initial Permutation*/
  lr[i]=inputb[p[i]-1];
  if(i<32) l[i]=lr[i];
  else r[i-32]=lr[i];
 }
 for(cnt=1;cnt<=16;cnt++){ /*Main encryption loop*/
  cypher(r,cnt,fn);
  for(i=0;i<32;i++){
   j=r[i];
   r[i]=l[i]^fn[i];
   l[i]=j;
  }
 }
 for(i=0;i<32;i++){
  lr[i]=r[i];
  lr[i+32]=l[i];
 }
 for(i=0;i<64;i++) out[i]=lr[invp[i]-1]; /*Inverse IP*/

 for(i=1;i<=8;i++)
  for(j=1;j<=8;j++) input[i-1]=(input[i-1]<<1)|out[i*8-j];
}

void des_decrypt(unsigned char *input){
 static unsigned char out[64];
 static int inputb[64],lr[64],l[32],r[32];
 static int fn[32];
 static int cnt,rtemp,n;
 register int i,j;

 for(i=n=0;i<8;i++)
  for(j=0;j<8;j++) inputb[n++]=(input[i]>>j&0x01);
 for(i=0;i<64;i++){ /*Initial Permutation*/
  lr[i]=inputb[p[i]-1];
  if(i<32) l[i]=lr[i];
  else r[i-32]=lr[i];
 }
 for(cnt=16;cnt>0;cnt--){ /*Main decryption loop*/
```

```
 cypher(r,cnt,fn);
 for(i=0;i<32;i++){
  rtemp=r[i];
  if(l[i]==1 && fn[i]==1) r[i]=0;
  else r[i]=(l[i] || fn[i]);
  l[i]=rtemp;
 }
}
for(i=0;i<32;i++){
 lr[i]=r[i];
 lr[i+32]=l[i];
}
for(i=0;i<64;i++) out[i]=lr[invp[i]-1]; /*Inverse IP*/

for(i=1;i<=8;i++)
 for(j=1;j<=8;j++) input[i-1]=(input[i-1]<<1) | out[i*8-j];
}
int main(int argc, char *argv[]){
unsigned char *key;
unsigned char data[8];
int n;
FILE *in;
FILE *out;

if (argc!=4) {
 printf("\r\nUsage:  des [e][d] <source file> <destination file>\r\n");
 return 1;
}

key=(unsigned char*)getpass("Enter Key:");
des_init(key);

if((in=fopen(argv[2],"rb"))==NULL){
 fprintf(stderr,"\r\nCould not open input file: %s",argv[2]);
 return 2;
}

if((out=fopen(argv[3],"wb"))==NULL){
 fprintf(stderr,"\r\nCould not open output file: %s",argv[3]);
 return 3;
}

if(argv[1][0]=='e'){
 while ((n=fread(data,1,8,in)) >0){
  des_encrypt(data);
  printf("data enctyted");
  if(fwrite(data,1,8,out) < 8){
   fprintf(stderr,"\r\nError writing to output file\r\n");
   return(3);
  }
```

```
 }
}

if(argv[1][0]=='d'){
 while ((n=fread(data,1,8,in)) >0){
  des_decrypt(data);
  if(fwrite(data,1,8,out) < 8){
   fprintf(stderr,"\r\nError writing to output file\r\n");
   return(3);
  }
 }
}

fclose(in); fclose(out);
// return (0);
}
fprintf(stderr,"\r\nError writing to output file\r\n");
   return(3);
  }
 }
}
fclose(in); fclose(out);
return 0;
}
```

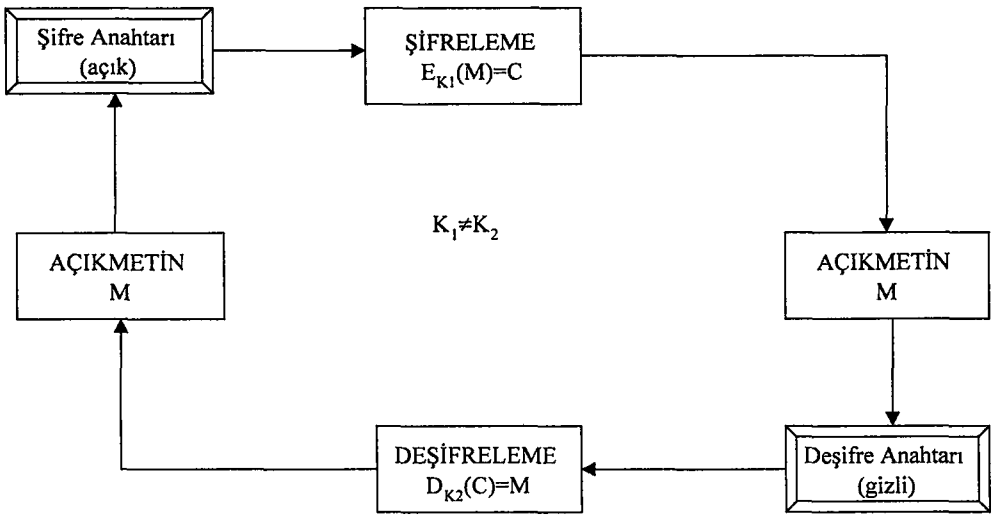
2.2 Günümüzde Kullanılan Asimetrik Algoritmalar

Şifreleme ve deşifreleme dönüşüm fonksiyonlarının tek ve aynı anahtar kullandığı simetrik kriptosistemler, hızlı ve birçok açıdan etkin olmalarına mutlak ve koşulsuz güvenli kriptosistemler sunabilmelerine karşın, tüm sistem güvenliğinin kullanılan anahtarla belirlenmesi bu sistemlerin en zayıf yanını oluşturmaktadır.

Bilgisayar bilim ve teknolojisinin eriştiği yüksek düzey göz önüne alındığında, simetrik kriptosistemlerin mutlak biçimde korumak zorunda oldukları anahtarların koruma ve dağıtım maliyetinin ne kadar yüksek ve koruma işleminin ne kadar zor olduğu kolayca görülebilir. Sırf bu nedenden ötürü, karşılıklı haberleşme içinde olan iki tarafın güvenli dağıtım kanalları oluşturması özellikle güncel bankacılık sisteminde , yaygın görülen bir örnektir.

Öte yandan, şifreleme ve deşifreleme dönüşüm fonksiyonlarının kullandıkları anahtarlar birbirinden ayrılarak anahtar güvenliği sorunu kesin biçimde çözülebilir. Anılan çözüm, anahtarların farklılığı nedeniyle Asimetrik Kriptosistem olarak bilinen ve ilk kez 1976'da Diffie ve Hellmarı tarafından belirlenen yeni bir dönüşüm tekniğiyle elde edilmektedir. Simetrik Kripto sistemlerin şematik görünümü Şekil 2.18'te sunulmuştur.
Şifreleme ve deşifreleme dönüşüm fonksiyonlarının birbirinden farklı anahtarlar kullanması, şifreleme anahtarının herkes tarafından bilinen açık bir anahtar olmasını sonuçlarken, deşifre anahtarı sadece yetkili alıcı tarafından bilinen gizli anahtar niteliğini yaratmıştır. Şifre anahtarı halka açık tutulduğu için, Asimetrik Kriptosistemler aynı zamanda Halk Anahtarlı Kriptosistemler (public key cryptosystem-PKS) olarak da bilinir.

Asimetrik Kriptosistemlerin tüm güvenliği deşifre anahtarının yalnız ve yalnız yetkili alıcı tarafından bilinmesinde yatar. Öte yandan, her ne kadar her iki anahtar birbirinden farklıysa da şifreleme anahtarından gidilerek deşifre anahtarını oluşturmak, teorik olarak olası, ancak pratikte çözümsüz bir problemdir.



Şekil 2.18 Asimetrik Kriptosistem

Bu açıdan incelendiğinde, Asimetrik Kriptosistemlerin en karakteristik özelliği; açık olan halk anahtarının ve ilişik kriptogranın, herkese açık ve dolayısıyla güvensiz bir kanaldan iletilebilmesidir. Asimetrik Kriptosistertılerin genel özelikleri aşağıda verildiği gibidir:

- Açık anahtar (Kp, p:public) ve gizli anahtar (K_s, s:secret) çiftinin oluşturulması basit olmalı ve alıcı tarafından polinomiyal zaman içinde yapılabilmelidir.
- Şifreleme işlemi :

  $C = E_{Kp}(M)$ olup, gönderici tarafından polinomiyal zamanda yapılabilmelidir.
- Alıcı tarafından yapılan deşifre işlemi, yine yalnız alıcı tarafından bilinen gizli anahtarla (K_s):$M = D_{Ks}(C)$ olarak ve polinomiyal zamanda gerçekleştirilebilmelidir.
- Düşman kriptanalisti açık anahtardan (Kp) giderek, gizli anahtarı (K_s) oluşturmaya kalktığında çözümsüz bir sorunla karşı karşıya gelmelidir.
- Düşman kriptanalist, açık anahtar ve kriptogram (Kp, C) ikilisinden hareketle açıkmetni (M) bulmaya çalıştığında, yine çözümsüz bir sorunla karşılaşmalıdır.

2.2.1 Rivest-Shamir-Adleman -RSA- Kriptosistemi

RSA public-key algoritması 1977 yılında R.Rivest, A.Shamir ve L.Adleman tarafından bulunmuş ve daha sonra public-key cryptography (genel anahtar şifrelemesi)ye uygun biçimde geliştirilmiştir. Bu algoritma, public-key encryption ve digital signature schemes işlemlerinde güvenli bir şekilde kullanılır.

S/MIME,PEM,MOSS ve PGP gibi gizli haberleşme protokolleri temel olarak RSA kullanır, ve bazı SSL,PCT gibi protokollerde kullanılır.

RSA Algoritması public-key şifreleme yönteminin temel bir uygulamasıdır. Bu şifreleme yönteminin diğer bir iyi tarafı ise önceden aralarında hiçbir görüşme yapmamış olan alıcı ve vericinin kendi aralarındaki iletişimin güvenli bir ortamda(güvenli gönderim ve mesajların doğrulanması) yapılmasıdır.

Görünüşte son derece basit matematiksel ilişkilerle çalışan bu yöntem de iki ayrı anahtar bulunmaktadır. Anahtarlardan birisi kamuya açık, birisi de gizlidir. Herkes açık anahtarını yayınlar ve kendisine şifreli bir mesaj göndermek isteyen birisi bu anahtarı kullanarak mesajı şifreler ve gönderir. Ancak mesajı sadece gizli anahtar kimde ise o çözebilir. Gizli anahtar da sadece sahibinde bulunur. Böylece, herkes çözüm için gerekli anahtarı bilmeden, güçlü bir şifreyle mesajları gizleyebilir.

Daha önce hiç karşılaşmamış, birbirini tanımayan kişiler bile birbirlerine gizli mesajlar gönderebilir. Örneğin Internet'ten alışveriş yapan birisi, kendisini hiçbir şekilde tanımayan bir web sitesine giderek, sitenin kamuya açık anahtarını alır, kart numarasını bu anahtarla şifrelereyek gönderir. Şifreli bilgiyi gönderen dahil hiç kimse çözemez, sadece web sitesinde bulunan gizli anahtarla gelen kart numarasını web sitesi çözebilir. Böylece kart hamili kart numarasının başkası tarafından okunmayacağından emin olacaktır. Ama, acaba Web sitesi gerçekten dürüst bir satıcı mı, yoksa sahte bir site mi ? Bundan emin olamayacaktır, ancak bunun da çözümü SERTİFİKA yöntemiyle sağlanmaktadır.

İnternetin şu anki durumu güvenliği tam olmayan kanallar üzerinden işler. Bu durumda public-key tekniğiyle internette güvenlik sağlanmıştır.

RSA şifreleme algoritmasının çalışması sekil 2.19'da gösterilmiştir.Şekilde gösterildiği gibi öncelikle p ve q olmak üzere iki tane asal sayı üretilir. Bunların birbirleriyle çarpılmasıyla n=p*q 'dan n elde edilir. Bundan sonra n sayısından küçük ve (p-1)*(q-1) sayısıyla 1 dışında herhangi bir ortak böleni bulunmayan bir e sayısı seçilir.

Daha sonra (E*D=1) sayısının (p-1)*(q-1) çarpımına tam olarak bölünmesini sağlayan bir D sayısı bulunur.

E ve D değerleri, sırasıyla, açık ve gizli anahtar olarak adlandırılırlar. Açık anahtarı (n,E) çifti, gizli anahtarı ise (n,D) çifti oluşturur. p ve q sayıları ya yok edilmeli ya da gizli anahtar ile birlikte saklanmalıdır.

Gizli anahtar olan D sayısının (n,E) sayılarından elde edilmesi zor bir işlemdir. Eğer bir kişi n sayısını çarpanlarına ayırarak p ve q sayılarını elde edebilirse gizli anahtarı da kolaylıkla bulabilir. Bu sebeple RSA sisteminin güvenliği çarpanlarına ayırma probleminin zorluğu temeline dayanır. Çarpanlarına ayırma işleminin kolay bir yönteminin bulunması, RSA algoritmasının kırılması anlamına gelir.

2.2.1.1 RSA Algoritmasının Yapısı

RSA şifreleme algoritmasında şifrelenecek olan açık metni öncelikle [0, n-1] arasındaki pozitif tamsayı bloklar haline dönüştürülür.Şekil 2.19** de ayrıntılı olarak matematiksel işlemler gösterilmektedir.

Bundan sonraki işlemimiz gizli anahtar ve açık anahtar çiftlerini elde etmektir.Bunun için p ve q şeklinde çok büyük iki tane birbirinden farklı iki asal sayı bulunur.

$$n = p*q \text{ ve } Z=(p-1)*(q-1)$$

hesaplanır. Z ile ortak böleni 1 olacak şekilde bir E sayısı bulunur. Açık anahtar (Public key) {E,n} olarak belirlenir.

$$D=E^{-1} \bmod Z$$

olacak şekilde bir D sayısı bulunur. Gizli anahtar (Private key){D,n} olarak belirlenir.

Şifrelenecek mesajı *m* kabul edersek bu mesaj binary olarak $2^k<N$ olacak şekilde k bitlik kısımlara ayrılır.

$$m=m(1)+m(2)+m(3)+...+m(n)$$

Daha sonra şifreleme için her bir kısma $C(i)=m(i)^E \bmod N$ işlemi uygulanır.

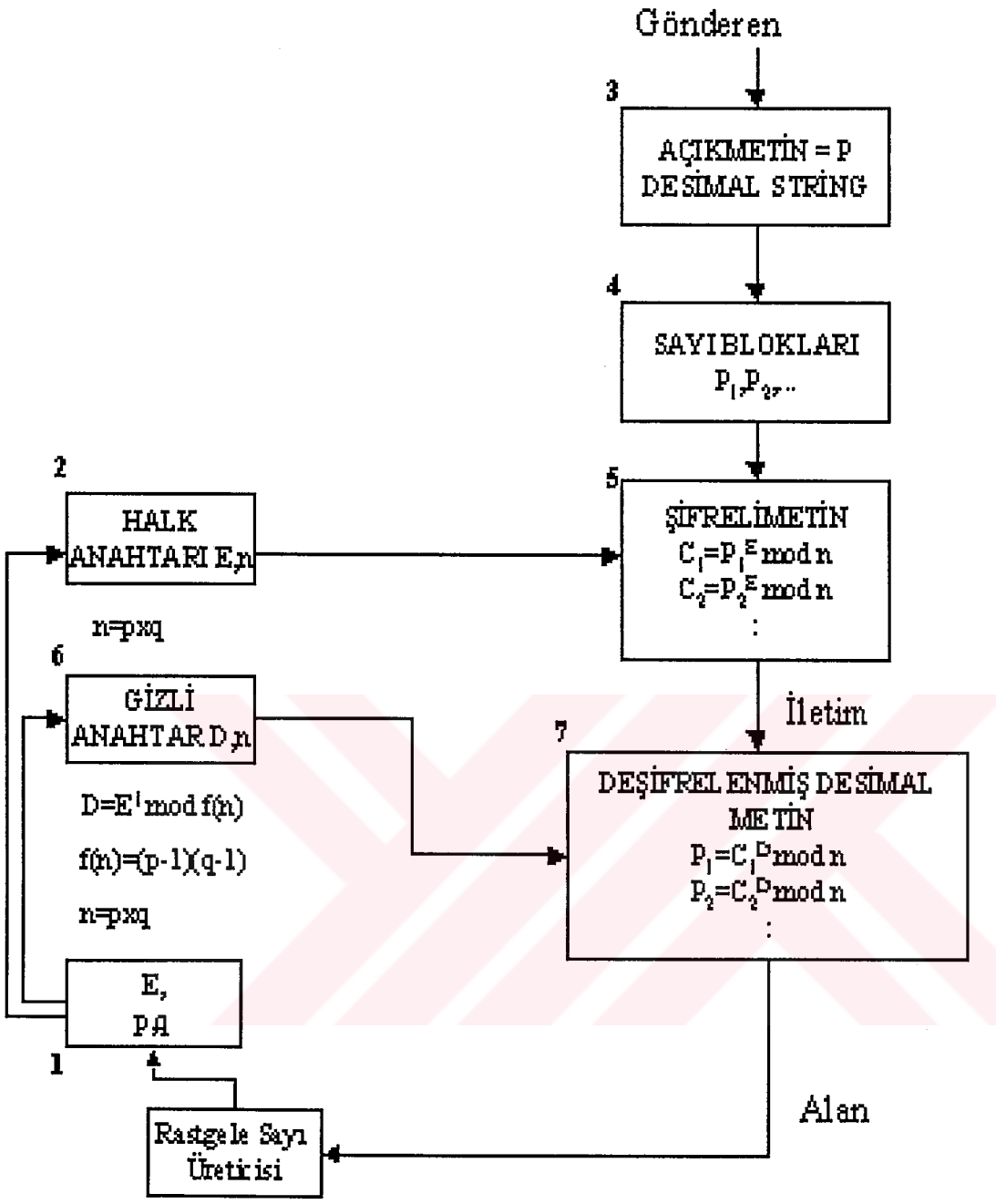
Böylece şifreleme işlemimizi bitirmiş oluruz.Girişte kullandığımız açık metin **m** şifrelenmiş olarak C şeklinde elde ederiz.

Deşifreleme

Belirlediğimiz D gizli anahtarı ile elimizde bulunan şifrelenmiş C metnini çözmemiz gerekiyor.Bunun içinde şifrelemek için kullandığımız bir matematiksel işlem kullanırız.

Gizli anahtar {D,n} kullanılarak şifre çözümü:

$$m(i)=C(i)^D \bmod N$$



Şekil 2.19 RSA Algoritması

2.2.1.2 RSA Algoritmasına Sayısal Örnekler

Örnek1:

RSA şifreleme sistemi kullanılarak **STOP** sözcüğünün şifrelenmesi.

p=43, q=59 ve n=13 için **STOP** sözcüğünü RSA kullanarak şifreleyiniz.

n=p*q=43*59=2537

gcd(e,(p-1)(q-1))=gcd(13,42*58)=1

Ortak bölenlerin en büyüğü=1 Kendi aralarında asal...

STOP sözcüğünü ikili bloklar halinde organize edersek:

STOP= 1819 1415

Harfleri rakama çevirmek için şöyle bir algoritma uygulanır:

A=01 ,B=03,C=04,D=05 .....

Her blok aşağıdaki formüle göre şifrelendirilir:

C1=181913 mod 2537 =2081

C2=141513 mod 2537 =2182

Şifrelenmiş bilginin deşifre edilmesi:

C^d=P (mod p*q) formülünü kullanarak:

C^1=2081,

C^2=2182,

d*13=1 (mod(42*58)), d={.....937.....} (biz 937 yi seçtik)

2081937=P (mod(43*59)) ->P1=1819

2182937=P (mod(43*59)) ->P2=1415

Örnek 2:

Şifreleme formülü,

$C=M^e$ MOD(n)

Çözme formülü $M = C^d$ MOD(n) buradaki MOD işlevi bir bölme işlemi sonunda kalan anlamına gelir. Örneğin 5/2 işlemi Bölme = 2, Kalan 1'dir. Yani $5 = 4 + 1 = 2 \times 2 + 1$ şeklinde ifade edildiğinde tam bölünemediği için son bölünebilen miktar 2, artan ya da kalan 1'dir. Örneğin

$C = 12^3$ MOD 17 = 1728

MOD(17)=(101x17+11)

MOD(17)=(1717+11)

MOD(17)=11

RSA, algoritması şöyle çalışır. Önce iki adet asal sayı p ve q bulunur. Gerçek uygulamada bu sayıların çok büyük olması, örneğin 100 ila 200 hane olması gerekir. Basit örneğimizde ise

p=11 , q=17 olursa

n=p*q=11x17= 187

m=(p-1)*(q-1)=10x16=160

bu aşama da uygun bir şifreleme anahtarı **e** seçmemiz gerekir. Bu herhangi bir asal sayı (3,7,11,..43...vs) olabilir. Tek dikkat etmeniz gereken **m** sayısına görece asal olmalıdır. Yani **e** sayısı **m** sayısının bir çarpanı olmamalıdır. Örneğin 5 olamaz, çünkü 5 . 32 = 160'dır. Ama 3 olabilir. 160, 3'e tam bölünemez. Böylece **e** = 3 sayısını bulduktan sonra çözüm anahtarı **d** şu formülle bulunur.

d = 1/e mod m

d = 1/3 mod 160= **107**

Bu aşamada artık **p** ve **q** sayılarını yok edebiliriz. Kamuya şifreleme anahtarımızı e,n olarak açıklarız. Açık Anahtarımız = { 3, 187) Gizli anahtarımız ise **d = 107** gizlenir ve kimseye söylenmez.

Şimdi süper web sitemizde kredi kartlarını kabul etmeye hazırız, ve ilk müşterimiz kart numarasını
(4123 4569 7890 1236) şu şekilde şifreler.

$C = 41^3 \bmod 187 | 23^3 \bmod 187 | 45^3 \bmod 187 | 69^3 \bmod 187 | 78^3 \bmod 187 | 90^3 \bmod 187 |$

$12^3 \bmod 187 | 36^3 \bmod 187 = 105 | 12 | 56 | 377 | 133 | 74 | 45 | 93$ ve bize

105:012:056:377:133:074:045:093 olarak kart numarasını gönderir.

Gördüğünüz gibi **4123456978901236** numaralı kartımız

105012056377133074045093 olmuştur.

Süper web sitemiz, süper gizli 107 şifre anahtarını çıkarır ve gelen kart numarasını çözer.

$M = 105^{107} \bmod 187 | 12^{107} \bmod 187 | 56^{107} \bmod 187 | 377^{107} \bmod 187 | 133^{107} \bmod 187 |$

$74^{107} \bmod 187 | 45^{107} \bmod 187 | 93^{107} \bmod 187 =$ **41 | 23 | 45 | 69 | 78 | 90 | 12 | 36**

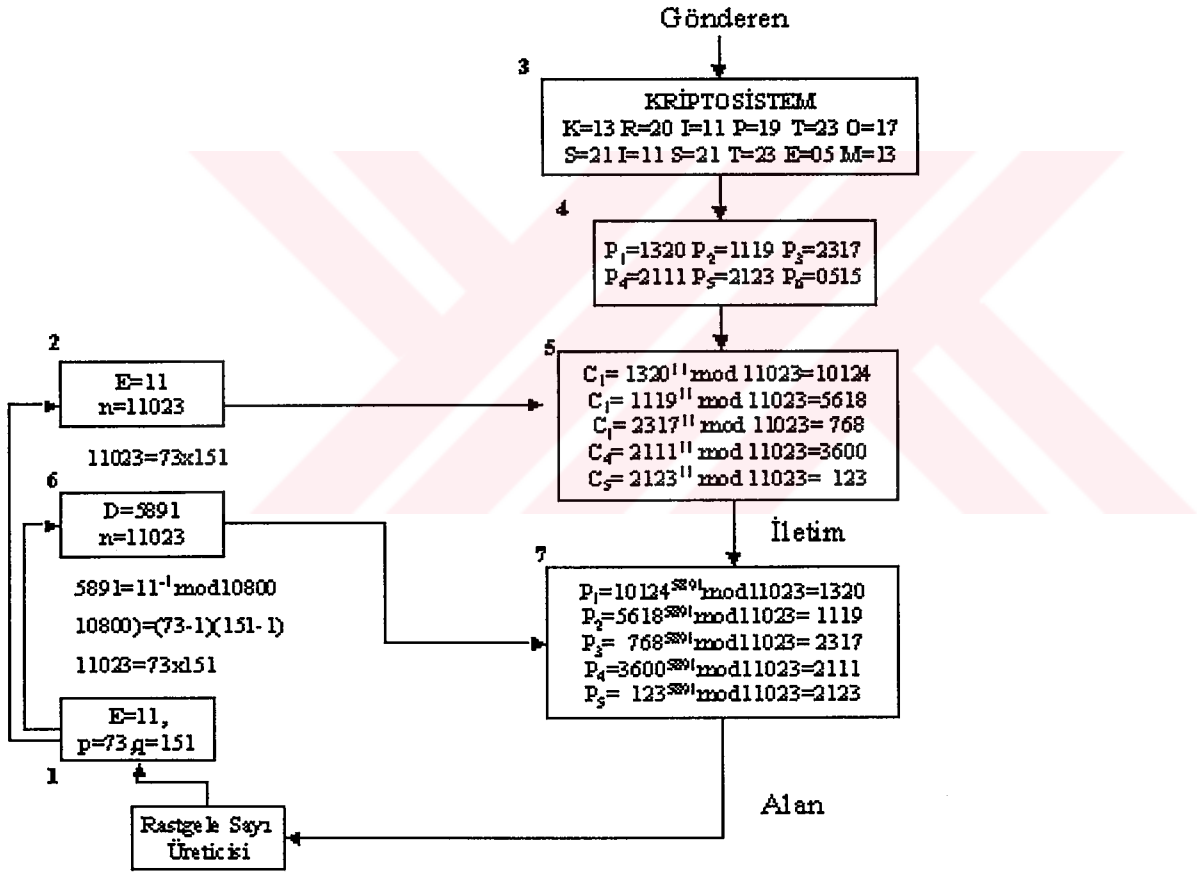
Tabii buna inanmanız için önce 105^{107} sayısını hesaplayabilmeniz ve bu sayıyı 187'ye bölerek kalanını bulabilmeniz gerekir. Bunu yaptığınızda yukarıdaki dönüşümün gerçekleştiğini göreceksiniz. Bu örnek çok basit bir örnektir, gerçek hayatta ise p ve q sayılarının 100 haneden fazla, n sayısının da 10000 haneden fazla olması gerektiğini bilmeniz gerekir.

RSA algoritmasının gücü iki çok büyük asal sayının çarpanlarının (faktörleri) bulunmasının zorluğudur. Bu örneğimizdeki 187 sayısının çarpanlarını çok basit bir algoritma ile bile, örneğin 1'den 187'ye kadar olası tüm sayıları tek, tek çarparak bulmamız mümkündür. Bu işlem bir saniye bile sürmeyecektir. Yani 187'nin çarpanları 11 ve 17 bulunduğu anda aynı formüllerle RSA şifresi kırılabilecektir. Ancak çok büyük sayılarda çarpanları bulmak imkansızlaşmaktadır.

Bugün itibariyle gelinen nokta 512 bitlik RSA anahtarları kırılabilmektedir. Bunun için 3000 adet hızlı bilgisayarın, 3 ay çalışması gerekmektedir. Bu nedenle çok gizli işler için 1024, askeri sırlar için ise 2048 bit gerekmektedir.

Örnek 3:

Bu örneğimizi şekil 2.19 de verdiğimiz algoritma üzerinde gösterilecektir.Burada harflerimizi belirli sayıların eşitliğini sağladık.Daha sonra sayı blokları halinde oluşturarak açık metin P'leri elde ettik.Simdi RSA algoritmasının matematiksel formüllerini kullanarak şifreleme işlemimizi yaptık.Şekil 2.20 de ayrıntılı olarak gösterilmiştir.



Şekil 2.20 RSA Algoritması Sayısal Örneği

2.2.1.3 RSA Sisteminin Güvenirliği

RSA sisteminin "kırılması" birkaç değişik şekilde yorumlanabilir. Sisteme en çok zarar verecek saldırı bir kriptanalistin belli bir açık anahtara karşı gelen gizli anahtarı bulmasıdır. Bunu başarabilen bir "hasım" hem şifrelenen bütün masajları okuyabilir: hem de imzaları taklit edebilir. Bunu yapmanın en akla gelen yolu n'nin asal çarpanlara ayrılması yani p ve q'nun hesaplanmasıdır. p q ve açık üs e kullanılarak d kolaylıkla hesaplanabilir. Ancak buradaki zorluk n modülünün çarpanlarına ayrılmasıdır. RSA sisteminin güvenliği çok büyük sayıların asal çarpanlarına ayrılmasının zorluğu varsayımına dayanır. Büyük sayıların çarpanlarına ayrılmasının zorluğu ispatlanmış değildir. Son üç yüzyıl içerisinde Fermat ve Legendre gibi ünlü matematikçiler bu konuda çalışmalar yapmışlardır.

Ancak n yeterince büyük seçilirse günümüzün teknolojisiyle n'nin çarpanlarına ayrılması "yeterince" uzun süreceği için bu yöntemle n'nin hesaplanması hesaplama açısından verimsiz olacaktır.

RSA'yı kırmanın bir başka yolu mod n'e göre e'inci köklerin hesaplanmasıdır . $c=m^e$ olduğuna göre c'nin e'inci kökü m'dir. Böyle bir saldırı, gizli anahtar bilinmese dahi şifrelenmiş mesajların deşifre edilmesini ya da imzaların taklit edilmesini sağlayabilir. Böyle bir saldırının n'nin çarpanlarına ayrılmasına eşdeğer olup olmadığı bilinmemektedir. Şu ana kadar RSA yöntemini bu yolla kırmaya çalışan bir metoda rastlanmamıştır.

Tabii bir kriptanalistin doğru olanı bulana kadar mümkün olan tüm d'leri denemesi mümkündür. Ancak böyle bir brute-force saldırı n'in çarpanlarına ayrılmasından daha verimsizdir, bir başka yöntem ise (p -1)(q -1)'in değerinin tahmin edilmesi olabilir . Bu da aynı şekilde n'in çarpanlarına ayrılmasından daha kolay değildir.

Yayınlanan makalede bahsedilen bir başka saldırı da n'i çarpanlarına ayırmadan ϕ>(n)'nin hesaplanmasıdır. Eğer bir kriptanalist ϕ >(n)'i hesaplayabilirse e'nin ϕ >(n) modülüne göre çarpımsal tersini hesaplayarak d'yi bulabilir. Ancak bu n'nin çarpanlarına ayrılmasından daha kolay değildir çünkü bu yolla kriptanalist ϕ >(n)'i kullanarak n'i

kolaylıkla çarpanlarına ayırabilir. Bunun için $\phi(n) = n - (p + q) + 1$ eşitliğinden n bilindiği için. (p + q) hesaplanır.

$$(p - q) = \sqrt{[(p + q)^2 - 4n]}$$

olduğundan (p -q) bulunur. Bu iki eşitlikten p ve q hesaplanabilir. Yine orijinal makalede d'nin hesaplanmasının n'in çarpanlarına ayrılmasından daha kolay olamayacağı iddia edilmektedir. Çünkü eğer d bilinirse n kolaylıkla çarpanlara ayrılabilir.

2.2.2 El Gamal Kripto sistemi

Temelde Ayrık Logaritma problemine dayanan bu sistem,

$Y = g^x \pmod p$; $p \in Z^+$, p asal sayı,

koşuluyla verilen denklemde, x sayısının bulunmasına dayanır. El Gamal tafindan 1985 yılında formalize edilen bu kripto sistemin ayrıntıları aşağıdaki gibidir :

A ve B kullanıcıları ortak bir anahtar oluşturmak istediklerinde:

X_a ve X_b , A ve B kullanıcılarınca bilinen gizli ifadeler

p, büyük asal bir sayı ve

a, (mod p)'nin ilkel elementi olsun.

Bu durumda :

$y_A = a^{X_A} \pmod p$ eşitliği A tarafından ve

$y_B = a^{X_B} \pmod p$ eşitliği B tarafından bulunarak,

değiş tokuş edilir.

Bu aşamada ortak gizli anahtar:

$$K_{AB} = a^{X_A X_B} \pmod p$$
$$= y_A^{X_B} \pmod p$$
$$= y_B^{X_A} \pmod p$$

olarak saptanır. Şimdi A'dan B'ye şifreli bir mesaj gönderilecekse:

A, 0 ile (p-1) arasında bir k sayısı seçerek,

$K = y_B^k \pmod p$ denklemiyle anahtarı oluşturur.

$y_B = a^{Xb} \pmod p$, B'nini açık şifreleme anahtarı olup,

B tarafından A'ya daha önceden gönderilmiştir.

Bu iki ifade kullanılarak şifreli metin c1 ve c2 ikilisi halinde aşağıdaki gibi oluşturulur.

$c1 = a^k \pmod p$ ve $c2 = Km \pmod p$

Burada dikkat edilmesi gereken nokta, kriptogram uzunluğunun, açık metin uzunluğunun iki katına çıkmış olmasıdır.

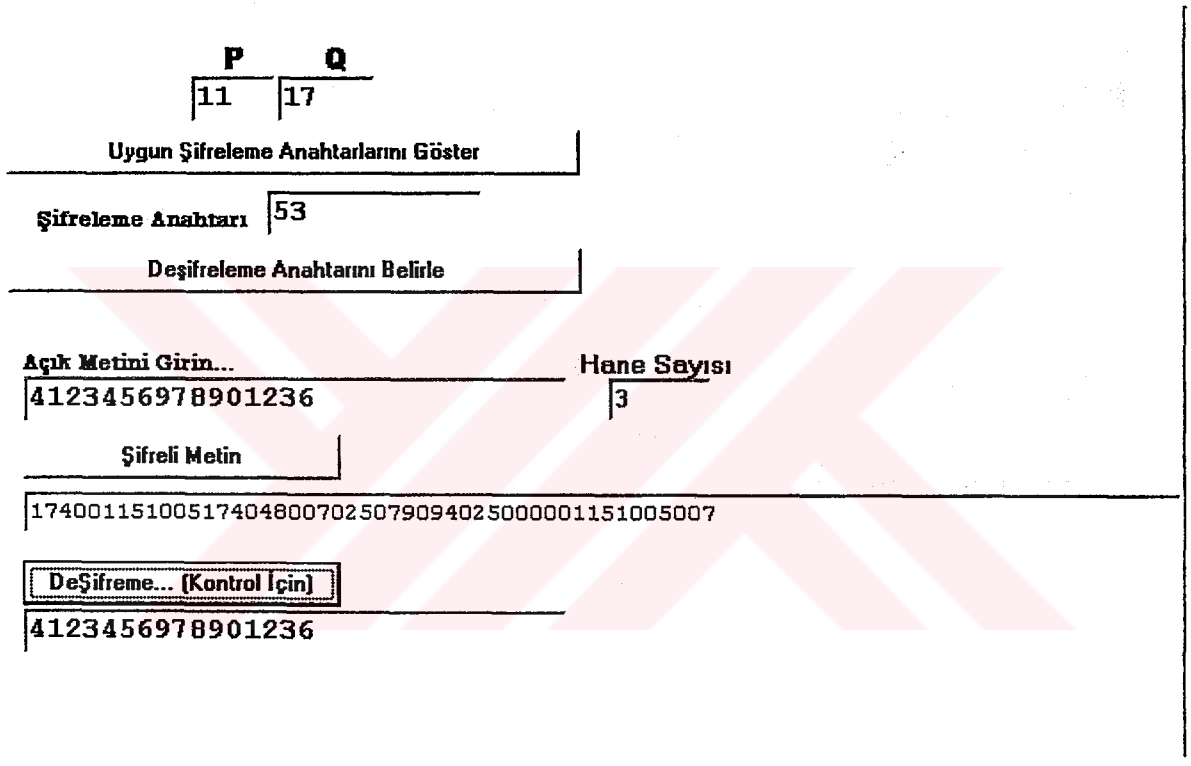
B'ye varan mesaj burada iki aşamada deşifre edilir:

- Birinci aşama anahtar -K- eldesi olup;
  $K = (a^k)^{X_B} = c^{1\,X_B} \pmod p$ ile gerçekleştirir.
- İkinci aşamada mesaj
  $m = (c2 / K)$ ile elde edilir.

3. RSA VE EL GAMAL ŞİFRELEME ALGORİTMALARININ PROGRAMLARI

3.1 RSA Programının Çalışması

RSA algoritmasının programını delphi 6.0 da gerçekleştirdik. Bu programda şifreleme ve deşifreleme işlemlerinin nasıl yapıldığını adım adım inceleyelim.



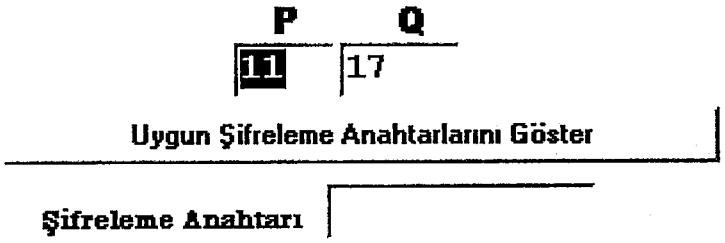
1.Şifreleme Anahtatarının elde edilmesi:

Programımıza bakıldığında öncelikle RSA algoritması temelini oluşturan p ve q sayılarını girmemiz gerekiyor. Bu sayılar çok büyük asal sayılardır.Biz programımızda 10.000 seviyelerindeki asal sayıları kullanacağız.

Programımızda,

$Z = (p-1)*(q-1)$ hesaplanır;

Ve Z ile ortak böleni 1 olacak şekilde bir şifreleme anahtarı bulunur. P ve q sayıları 10000 seviyelerinde olduğunu düşünürse şifreleme anahtarı olabilecek sayılar çok fazladır. Program içinde görüntülenen bu sayılardan bir tanesini şifreleme anahtarı olarak seçebiliriz.

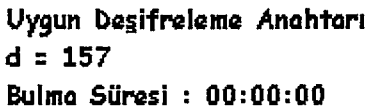


Şifreleme anahtarının bulunması p ve q 10000 seviyelerinde olduğunda 3-4 saniye sürmektedir.

2.Deşifreleme anahtarının bulunması:

Deşifreleme anahtarı bulurken seçtiğimiz şifreleme anahtarını kullanacağız.

$D = 1*E \bmod n \qquad n = p*q;$



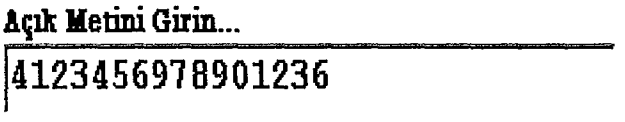
İşlemini yaparak D deşifreleme anahtarını buluruz.

Burada önemli olan bir nokta seçtiğimiz şifreleme anahtarına karşılık sadece bir tane deşifreleme anahtarının olmasıdır. Buda RSA algoritmasının en önemli özelliğidir.

Programda şifreleme anahtarı 5000 seviyelerinde seçtiğimizde yaklaşık olarak 3-4 dakika sürmektedir.

3.Açık metnin şifrelenmesi:

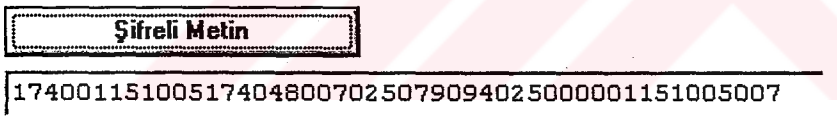
Şifreleyeceğimiz sayıları text'e giriyoruz.



Burada;

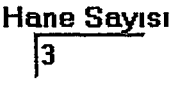
$$C = M^e \bmod n ;$$

İşlemini uygulayarak şifrelenmiş metni elde ediyoruz.



Programda daha büyük sayıları şifreleyebilmek için şifrelenecek sayının her bir rakamını ele alarak ayrı ayrı şifreliyoruz. Bu bize daha büyük sayılarla işlem yapmayı ve daha hızlı şifreleme olanağı sağlıyor.

Programda birde hücre sayısı bölümü var. Bu işlem bize şifrelenmiş sayıların her birinin aynı bit uzunluğunda olmasını sağlar.



4.Deşifreleme işlemi:

Deşifreleme yapmak için şu işlem uygulanır;

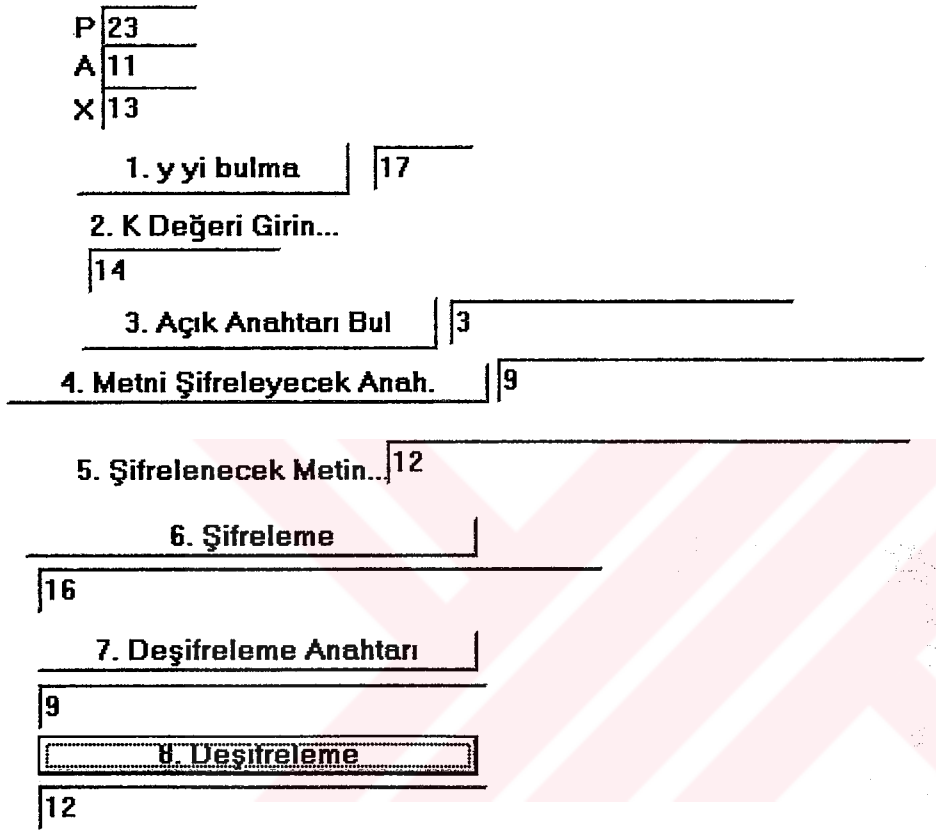
$$M = C^d \bmod n$$

Bu işlem şifrelenmiş metni hücre sayısı kadar bloklara ayırır ve deşifreleme işlemini yapar.

Deşifreleme anahtarı 10 milyon seviyelerinde alındığında deşifreleme işlemi yaklaşık olarak 8 dakika sürmektedir. Burada önemli olan noktalardan biri şifrelenecek sayının artmasıyla deşifreleme işlemi uzar.

3.2 EL GAMAL Programının Çalışması

El Gamal algoritmasının programını delphi 6.0 da gerçekleştirdik. Bu programda şifreleme ve deşifreleme işlemlerinin nasıl yapıldığını adım adım inceleyelim

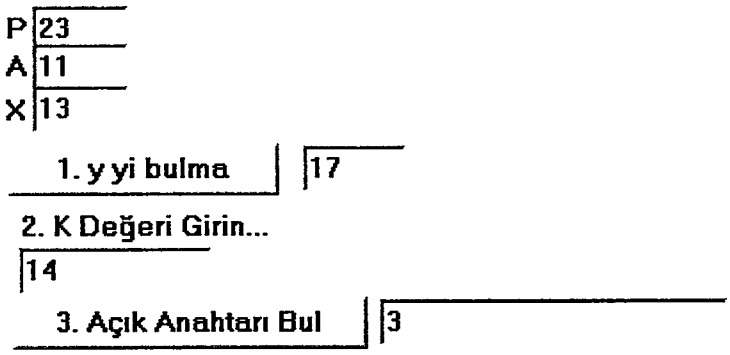


1.Verilerin Girilmesi ve Şifreleme işlemi:

Bu programda öncelikle verilerimizin yani şifreleme işlemini gerçekleştirecek sayılarımızı giriyoruz.

Büyük bir asal sayı olan p sayımızı giriyoruz.İkinci asal sayımız olan a sayımızı giriyoruz.Daha sonra x (asal olması zorunlu değil) gizli anahtarımızı giriyoruz.

$Y = a^x \bmod p$;



İşlemi ile y değerini buluruz. Bu Y değeri daha sonraki aşamada şifreleme anahtarını bulmaya yaracak.

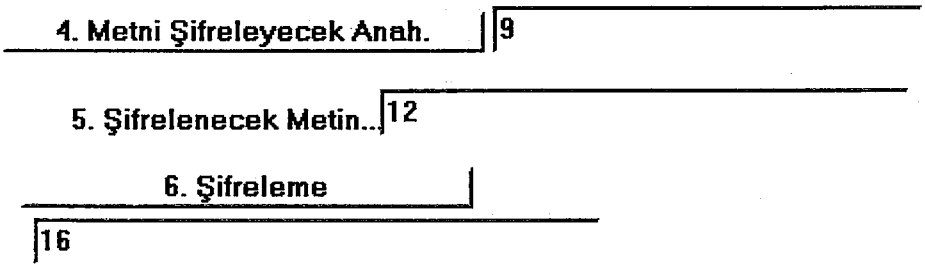
Rastgele bir k (asal olması zorunlu değil) sayısı girilir.

a^k mod p işleminin değeri bulunur.Bu sayı ortama verilir.Deşifreleme anahtarının bulunması için kullanılır.

$$S = Y^k \bmod p$$

işleminin değeri bulunur.Bu sayı şifreleme anahtarı olarak kullanılır.

Şifreleme işlemini gerçekleştirmek için bütün verilerimizi elde ettik. Şimdi şifreleme işlemini gerçekleştirebiliriz. Şifrelenecek metni girebiliriz.

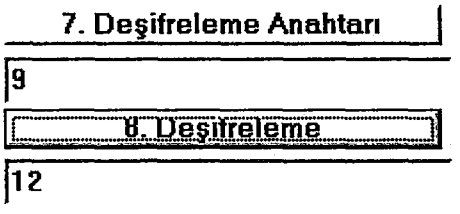


C = M * S mod p işlemi ile şifreleme işlemini gerçekleştiririz.

2.Deşifreleme İşlemi:

Deşifleme işleminde ortama verilen şifrelememiş metini ve açık anahtarı kullanıyoruz.

$D = (a^k \bmod p)^x \bmod p$ işlemini uygulayarak deşifreleme anahtarını kullnıyoruz.
$C = M * D \bmod p$ işlemi ile deşifreleme işlemini gerçekleştiriyoruz.



3.3 RSA – EL GAMAL Karsılaştırılması

Asimetrik kripto sistemlerde biri açık diğeri gizli anahtar olmak üzere şifreleme ve deşifreleme yapmak için iki farklı anahtar kullanmaktadır. Bu anahtarlar için önemli olan nokta aralarında çözülmesi zor bir matematiksel bağlantı olmasıdır.

El Gamal şifreleme algoritması Diffe-Hellman çalışmalarından sonra El Gamal tarafından geliştirilen sayısal imzalama ve şifreleme algoritmasıdır. RSA Rivest-Shamir-Adleman tarafından 1977 yılında geliştirilen ve günümüzde de kullanılabilen şifreleme algoritmasıdır.

Bu algoritmaların her ikisinde de modüler aritmetik mantığı kullanılmaktadır. Bunun yanında El Gamal şifrelemesinde Ayrık Logaritma problemi de kullanılmaktadır.

El Gamal ve RSA algoritmaları bir yazılı metni şifrelemekten çok, sayıları şifrelemek ve deşifrelemek üzere tasarlanmıştır .Bu da çok önemli bir noktadır.Bu bankacılık ve internet sisteminde kart numaralarının, kart şifrelerinin saklanmasında ve bu kartların kimlere ait olduğunun bilinmesi için imzalama sistemi önemlidir. El Gamal ve RSA bu istediklerimizi sağlayabilen şifreleme algoritmalarıdır.

Bu her iki şifreleme algoritmanın da programını yaptık ve bu algoritmalar arasında güvenlik, şifreleme ve deşifreleme hızları arasında avantaj ve dezavantajları bulunmaktadır. Öncelikle şunu söyleyebiliriz ki; RSA, El Gamal şifreleme algoritmasına göre daha güvenilir bir sistemdir ve RSA günümüzde kullanılan birçok şifreleme protokollerin alt yapısını oluşturmaktadır.

Şifreleme algoritmalarını göz önüne alırsak her iki algoritmada da şifreleme ve deşifreleme işlemi yapılırken farlı sayıda veriler girilmektedir. Her iki algoritmada da iki farklı asal sayı kullanıldığını söylemiştik. EL Gamal şifreleme metodunda bu iki asal sayıdan biri (p:programda görülüyor) hem şifreleme de hem de deşifreleme kullanılır. Diğer asal sayı(a:programda görülüyor) sadece şifreleme yaparken kullanılmaktadır. Yani a asal sayısı şifreleme anahtarının bulunmasında kullanılır. p asal sayısı ise hem şifreleme hem de deşifreleme de modüler işleminin seviyesini belirler. RSA da ise kullanılan asal sayılar (p ve q) hem şifreleme hem de deşifrelemede kullanılır. Her iki asal sayıda şifreleme ve deşifreleme anahtarının bulunmasında kullanılır.

El Gamal şifreleme metodunda iki asal sayıdan farklı olarak x ve k olarak asal sayı olmasına gerek olmayan iki tane rastgele sayı giriyoruz. X şifreleme anahtarının bulunmasında kullanılırken k açık metnin şifrelenmesinde kullanılmaktadır. Ayrıca x deşifreleme anahtarının bulunmasında da kullanılmaktadır. RSA algoritmasında p ve q asal sayılarından başka randım başka sayı girilmesin gerekmemektedir.

El Gamal şifreleme sisteminin en büyük dezavatajı girdiğimiz p asal sayısı derecesinde bir şifrelenecek sayıyı şifreleye bilmesidir. Buradan da bu şifreleme algoritmasının en büyük asal sayı kadar büyülükteki bir sayıyı şifreleyebileceği sonucu çıkmaktadır. RSA algoritmasında ise böyle bir problem yoktur.İstediğimiz uzunluktaki bir sayıyı şifreleyebiliriz.

Şifrelemede önemli olan noktalardan biri şifreleme ve deşifreleme anahtarlarının uzunluğudur. Bu anahtarları uzunluğu şifreleme sisteminin güvenliğini belirlemede önemli bir noktadır. Bu anahtarlar ne kadar uzun olursa sistem o kadar güvenli olur. EL Gamal şifreleme sisteminde şifreleme anahtarı ve deşifreleme anahtarı girdiğimiz p asal sayısından fazla olamıyor. Bu da sistemin güvenirliğini azaltıyor. RSA şifreleme sisteminde ise şifreleme ve deşifreleme anahtarlarının uzunluğu p*q seviyesindedir.Bizim yaptığımız programlar da EL

Gamal şifreleme sisteminin anahtar uzunluğu 10.000 seviyelerinde, RSA şifreleme sisteminde 40 milyon sevilerine çıkartılabiliyor.

Şifreleme ve deşifreme anahtarlarının uzunluğu şifreleme hızını ve deşifreleme hızın etkiler. EL Gamal sisteminde anahtarlar uzun olmadığı için şifreleme ve deşifreleme hızları yaptığımız programda 1 saniyenin altındadır. RSA sisteminde ise anahtar uzunluğu artıkça şifreleme ve deşifreleme hızı uzunluk oranında artmaktadır. RSA sisteminin farklı anahtar uzunlularındaki şifreleme ve deşifreleme hızları aşağıdaki tabloda verilmiştir.

Sonuç olarak El Gamal şifreleme sistemi şifreme anahtarlarını ve deşifreleme anahtarlarını bulurken daha hızlı fakat bunun nedeni daha küçük asal sayılarda işlem yaptığımızdan dolayı. RSA sistemi büyük asal sayılarla işlem yapmamızı ve daha büyük sayıları şifrelememize olanak sağlamaktadır. RSA sisteminde büyük sayılarla işlem yapıldığında deşifreleme anahtarının bulunması uzun sürebilir fakat bu algoritmanın güvenliğini arttırmaktadır. Ayrıca RSA bugünkü sistemlerin alt yapısını oluşturması en büyük avantajıdır. Bununla birlikte RSA sisteminin El Gamal sistemine göre daha az rastgele sayı kullanılmasından dolayı kullanım açısından daha iyi bir avantaja sahiptir. El Gamal sistemi ayrıca sayısal imzalamada kullanılması artı bir yönüdür.

P	Q	E	D	Deşifreleme anahtarının bulunma süresi	Şifreleme süresi (4 digit)	Deşifreleme süresi
137	149	127	6815	1 sn.'nin altında.	1 sn.'nin altında	1 sn.'nin altında.
503	509	523	212107	8 sn	1 sn.'nin altında	1 sn.'nin altında.
887	907	911	423827	14 sn	1 sn.'nin altında	1 sn.'nin altında.
1559	1567	1549	2312245	1 dk	1 sn.'nin altında	4 sn
2767	2777	2789	3055949	1.30 dk	1 sn.'nin altında	10 sn
3533	3457	3511	9818119	6.48 dk	1 sn.'nin altında	34 sn
4483	4597	4649	20368885	15.52 dk	1 sn.'nin altında	1.12 dk
5221	5197	5153	25965137	19.55 dk	1 sn.'nin altında	2.50 dk
6389	6359	6421	27411221	22.38 dk	1 sn.'nin altında	2.85 dk
7451	7457	7393	41594657	27.53 dk	1 sn.'nin altında	3.15 dk
8117	8221	8147	47933207	35 dk	1 sn.'nin altında	3.90 dk
9851	9923	9859	54640139	41.46 dk	1 sn.'nin altında	4.21 dk

3.4 RSA ve EL GAMAL Program Kodları

3.4.1 RSA Program Kodları

```
unit Unit1;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls;

type
  TForm1 = class(TForm)
    EditQ: TEdit;
    EditP: TEdit;
    Memo1: TMemo;
    Button1: TButton;
    Edite: TEdit;
    Button2: TButton;
    Edit1: TEdit;
    Edit2: TEdit;
    Button3: TButton;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Edit3: TEdit;
    Button4: TButton;
    Edit4: TEdit;
    Label5: TLabel;
    procedure AsalSayilar(sinir : integer);
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure Button3Click(Sender: TObject);
    procedure Button4Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;
  p,q,n,z,d,e,counter,m,c: Int64;
  plaintext, sifrelimetin : string;
```

```
implementation

uses Unit2;

{$R *.dfm}
procedure TForm1.AsalSayilar(sinir : integer);
var i,x: integer;
begin
///xxx
if sinir > 10000 then sinir := 10000;

 for i:= 2 to sinir do begin
  counter := 0;
   for x:= 2 to i do begin
   if i mod x = 0 then inc(counter); //asal sayilar
   if sinir mod i = 0 then inc(counter); // Kural 1 : z değerine bölündüğünde 0 vermeyecek
   end;

  if (counter = 1) and (i<10000) then Memo1.Lines.Add(inttostr(i));
  if (i=10000) then Memo1.Lines.Add('İşlem Devam Ediyor.');
  // eğer counter 1 çıkarsa demek ki sayı sadece kendisine bölünmüş ve kural 1 e uygun
  end;//for2

end;

procedure TForm1.Button1Click(Sender: TObject);
begin
q := strtoint(editP.text);
p := strtoint(editQ.text);

n := p*q;
z := (p-1)*(q-1);

Memo1.Lines.Clear;
//Memo1.Lines.Add('n := '+inttostr(n));
//Memo1.Lines.Add('z := '+inttostr(z));
Memo1.Lines.Add('Uygun Şifreleme Anahtarları');

AsalSayilar(z);

Memo1.Lines.Add('İşlem Tamamlandı.');
end;

procedure TForm1.Button2Click(Sender: TObject);
var i: longint;
begin
Memo1.Lines.Clear;
Memo1.Lines.Add('Uygun Deşifreleme Anahtarı');
Bekleyin.ProgressBar1.Max := z;
Bekleyin.ProgressBar1.Min := 0;
```

```
Bekleyin.ProgressBar1.Position := 0;
Bekleyin.Show;

baslangiczamani := Now;

e := strtoint(Edite.text);

for i := 1 to z do begin
Bekleyin.ProgressBar1.Position := i;
 if e*i mod z = 1 then begin
  d:=i;
  form1.Memo1.Lines.Add('d = '+inttostr(i));
  break;
end;
end;
//desifreleme anahtarının bulunması
Bekleyin.Close;
end;

function UsAlma(Sayi,Us,ModDeger: Int64): Int64;
var i,x: longint;
begin
x:=sayi;
for i := 1 to Us-1 do
 x := (x * sayi) mod ModDeger;
Result := x;
end;


procedure TForm1.Button3Click(Sender: TObject);
var i,x: Longint;
begin
edit2.Text:='';
plaintext:=edit1.text;
i:=1;
while (i <= strLen(PChar(plaintext))) do begin
m := strtoint(plaintext[i]);
c := UsAlma(m,e,n);

for x := strlen(PChar(inttostr(c))) to strtoint(edit4.Text)-1  do
edit2.Text := edit2.Text +'0';
edit2.Text := edit2.Text +inttostr(c);
i := i + 1;
end;

end;

procedure TForm1.Button4Click(Sender: TObject);
var i,x: Longint;
temp : string;
```

```
begin
edit3.Text:='';
sifrelimetin:=edit2.text;

Bekleyin.ProgressBar1.Max := strLen(PChar(sifrelimetin));
Bekleyin.ProgressBar1.Min := 0;
Bekleyin.ProgressBar1.Position := 0;
Bekleyin.Show;

baslangiczamani := Now;
i:=1;

while (i <= strLen(PChar(sifrelimetin))) do begin
Bekleyin.ProgressBar1.Position := i;
temp := EmptyStr;
for x := 1 to strtoint(edit4.Text) do begin
temp := temp + sifrelimetin[i];
inc(i);
end;
c := strtoint(temp);
m := UsAlma(c,d,n);
edit3.Text := edit3.Text +inttostr(m);
Memo1.Lines.Clear;
Memo1.Lines.Add('RSA');
end;

Bekleyin.Close;
end;

end.
```

3.4.2 El Gamal Program Kodları

```
unit Unit1;


interface


uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls;


type
  TForm1 = class(TForm)
    Label1: TLabel;
```

```
  Label2: TLabel;
  EditP: TEdit;
  EditA: TEdit;
  EditX: TEdit;
  Label3: TLabel;
  Button1: TButton;
  EditY: TEdit;
  EditK: TEdit;
  Label4: TLabel;
  Button2: TButton;
  EditACIK: TEdit;
  Label5: TLabel;
  Button3: TButton;
  EditSifre: TEdit;
  EditMetinS: TEdit;
  Button4: TButton;
  EditMetinSifreli: TEdit;
  Button5: TButton;
  EditADSifre: TEdit;
  Button6: TButton;
  EditDMetin: TEdit;
  procedure Button1Click(Sender: TObject);
  procedure Button2Click(Sender: TObject);
  procedure Button3Click(Sender: TObject);
  procedure Button4Click(Sender: TObject);
  procedure Button5Click(Sender: TObject);
  procedure Button6Click(Sender: TObject);
 private
  { Private declarations }
 public
  { Public declarations }
 end;

var
```

```
 Form1: TForm1;

implementation

{$R *.dfm}

function UsAlma(Sayi,Us,ModDeger: Integer): Longint;
var i,x: longint;
begin
x:=sayi;
for i := 1 to Us-1 do
 x := (x * sayi) mod ModDeger;
Result := x;
end;

procedure TForm1.Button1Click(Sender: TObject);
begin
EditY.Text :=
inttostr(UsAlma(strtoint(editA.Text),strtoint(editX.Text),strtoint(editP.Text)));
end;

procedure TForm1.Button2Click(Sender: TObject);
begin
EditACIK.Text :=
inttostr(UsAlma(strtoint(editA.Text),strtoint(editK.Text),strtoint(editP.Text)));
end;

procedure TForm1.Button3Click(Sender: TObject);
begin
EditSifre.Text :=
inttostr(UsAlma(strtoint(EditY.Text),strtoint(editK.Text),strtoint(editP.Text)));
end;

procedure TForm1.Button4Click(Sender: TObject);
```

```
begin
EditMetinSifreli.Text :=
inttostr(strtoint(EditMetinS.Text)*strtoint(EditSifre.Text) mod strtoint(EditP.Text));
end;

procedure TForm1.Button5Click(Sender: TObject);
begin
EditADSifre.Text :=
inttostr(UsAlma(strtoint(EditACIK.Text),strtoint(editX.Text),strtoint(editP.Text)));
end;

procedure TForm1.Button6Click(Sender: TObject);
var m: integer;
bitti : boolean;
begin
bitti := false;
m:=1;
while bitti = false do begin
 if strtoint(EditMetinSifreli.Text) =
 m * strtoint(EditADSifre.Text) mod strtoint(EditP.Text) then begin
 bitti := true;
 EditDMetin.Text := inttostr(m);
 end;
 inc(m);
end;

end;

end.
```

4. SONUÇLAR VE TARTIŞMA

Hayatın her alanında bilgisayarların yaygın kullanımı ve özellikle bilgisayar ağlarının gelişmesi, dünyanın dört bir yanındaki bilgiye erişmeyi kolaylaştırırken, güvenlik sorunlarını beraberinde getirmektedir. Bilginin korunması ve güvenli bir şekilde taşınması çok önemli bir sorun haline gelmiştir.

Şifreleme, bilgiyi matematiksel işlemleri kullanarak veya bilgiyi belli bir algoritmaya göre yer değiştirme işlemi yaparak karmaşık hale getirerek gerçekleştirilir. Uygulanan yöntemlere baktığımızda bu iki işlemi gerçekleştiren farlı şifreleme algoritmaları bulunmaktadır. Bugün bankacılık alanında kullanılan RSA şifreleme algoritması matematiksel işlemleri kullanarak, DES yer değiştirme işlemi uygulayarak şifreleme yapmaktadırlar.

Şifrelemede önemli bir noktada şifreleme anahtarıdır. İlk şifreleme algoritmaları bilginin hem şifrelemede hem de deşifrelemede tek anahtarı kullanmaktadır. Böylece şifrelenmiş bilgiyi alıcıya gönderirken şifreleme anahtarının da gönderilmesi gerekmektedir. Bu da önemli bir sorun olmuştur. Buna DES'i örnek verebiliriz. Sonraki çalışmalar şifrelemede ve deşifreleme farklı anahtar kullanılabilecek şifreleme algoritmaları geliştirmek üzere olmuştur. RSA bu çalışmaların en önemlisidir. Bu çalışmaların sonucunda deşifreleme anahtarının taşınmasına gerek kalmamıştır. Şu da bir gerçektir ki şifreleme anahtarının büyüklüğü şifreleme algoritmasının güçlülüğünü göstermektedir. Şifreleme anahtarının büyüklüğü bir avantaj olmasının yanında dez avantaja da sahiptir. Şifreleme anahtarı büyüdükçe şifreleme ve deşifreleme işlemlerini yapma hızı ve bu işlemleri gerçekleştirme zamanı artmaktadır. Burada önemli olan bilginin ne derece önemli olduğudur. Bilginin kısa süre saklanmasını ve hızlı bir şekilde şifrelenmesini ve deşifrelenmesini istiyorsak şifreleme anahtarını daha küçük alabiliriz. Bilginin çok önemli ve uzun süre saklanmasının ve zamanın önemi yoksa yüksek sayılarda bir şifreleme anahtarı kullanarak çözüme ulaşabiliriz.

Bu çalışmada, RSA ve El Gamal şifreleme algoritmalarının Delph 6.0'da programlarını gerçekleştirdik. Program 256 MB belleğe sahip bir Pentium III 933 MHz işlemcide gerçekleştirilmiştir. RSA ve El Gamal şifreleme algoritmalarında matematiksel işlemler üzerine olduğu için işlemci hızı şifreleme ve deşifreleme hızı hızını etkilemektedir.

İşlemci hızı düştükçe şifreleme ve deşifreleme hızı azalmakta, işlemci hızı artıkça şifreleme ve deşifreleme hızı artmaktadır. Bunun sonucu şifreleme ve deşifreleme işlemlerinin hızlarını artırmak için büyük işlemcili bilgisayarlar yada sadece bu işte kullanılan makinalar geliştirilmektedir. RSA ve El Gamal şifreleme algoritmalarının programlarının ayrıntılı karşılaştırılması üçüncü bölümde anlatılmıştır.

Şifrelemede önemli noktalardan biride şifreleme algoritmamızın kırılmasıdır. Bu konu üzerine yoğun çalışmalar vardır. Ve büyük işlemcili makinalar ile her şifreleme algoritmasının kırılabileceğine inanılır. Tabi ki zaman burada önemli bir yer tutar. Diferansiyal kripto-analiz şifreleme algoritmalarının kırılması kullanılan önemli bir yöntemdir. Ayrıca, diğer bir analiz yöntemi olan Lineer kripto-analiz yöntemi, diferansiyal kripto-analiz yöntemiyle birleştirilerek bu konuda daha geniş incelemeler yapılabilir.

Sonuç olarak günümüz teknolojisi kullanılarak daha güçlü şifreleme algoritmaları geliştirilebilir. Bunun yanında farlı matematiksel işlemleri ele alarak şifreleme algoritmasını güçlendirebiliriz. Önemli olan güçlü, kırılması zor, şifreleme ve deşifreleme işlemlerini hızlı yapabilecek ve her ortamda kullanılabilecek algoritmaya ulaşmaktır. Bu konu üstünde günümüzde çok yoğun çalışmalar vardır.

KAYNAKLAR

[1] **Bruce Schneider**, 1996. Applied Cryptography, Second Edition, John Wiley & Sons, Inc. New York, Ny

[2] **William Stalings,** 1997. Cryptography and Network Security, Second Edition, Prentice Hall

[3] **Hellman, M. E.**, 1978. An Overview of Puplic Key Cryptography,IEEE Communications Society Magazine

[4] **Koblitz, N.,** 1994. A Course in Number Teory and Cryptography, Springer-Verlag, New York

[5] **Salomaa,A.,** 1990. Public-Key Cryptography, Springer-Verlag, New York

[6] **Shamir A., Biham E.,** 1993. Differrential Cryptanalysis of the data Encription Standart, Springer-Verlag, New York

[7] **Trappe W., Washington L.,**2002. Introduction to Cryptography with Coding Theory, 0-13-061814-4 (Hardback)

[8] **Barr T.**, 2002. Invitation to Cryptology, 0-13-088976-8 (Hardback)

[9] **William Stalings,** 2000. Network Security Essentials *Applications and Standards*

[10] **Richard Smith,** 1997. Internet Cryptography,

ÖZGEÇMİŞ

19.12.1977 tarihinde Edirne'de doğdum. İlk öğrenimimi Keşan/Edirne'de, orta ve lise eğitimimi Edirne Anadolu Lisesi'nde tamamladım. 1995 -1999 yılları arasında Yıldız Teknik Üniversitesi Elektronik ve Haberleşme Mühendisliği Bölümü'nde öğrenim gördüm. 1999 yılında Trakya Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Anabilim Dalı'nda Yüksek Lisans'a başladım. Aynı yıl Trakya Üniversitesi Mühendislik- Mimarlık Fakültesi Bilgisayar Mühendisliği Bölümü'nde Araştırma Görevlisi olarak çalışmaya başladım. Halen aynı görevi sürdürmekteyim.