KOCAELİ ÜNİVERSİTESİ * FEN BİLİMLERİ ENSTİTÜSÜ

VERİ ŞİFRELEMESİNDE SİMETRİK VE ASİMETRİK ANAHTARLAMA ALGORİTMALARININ UYGULANMASI (HYBRID ŞİFRELEME)

YÜKSEK LİSANS TEZİ

Kerim YILDIRIM

Anabilim Dalı: Bilgisayar Mühendisliği

Danışman:H.Engin DEMİRAY

KOCAELİ, 2006

KOCAELİ ÜNİVERSİTESİ * FEN BİLİMLERİ ENSTİTÜSÜ

VERİ ŞİFRELEMESİNDE SİMETRİK VE ASİMETRİK ANAHTARLAMA ALGORİTMALARININ UYGULANMASI (HYBRID ŞİFRELEME)

YÜKSEK LİSANS TEZİ

Kerim YILDIRIM

Tezin Enstitüye Verildiği Tarih: 06 Haziran 2006
Tezin Savunulduğu Tarih: 06 Temmuz 2006

Tez Danışmanı	Üye	Üye
Yrd.Doç.Dr. H.Engin DEMİRAY	Prof.Dr.A.Coşkun SÖNMEZ	Doç.Dr. Adnan KAVAK
(.....)	(.....)	(.....)

KOCAELİ, 2006

ÖNSÖZ ve TEŞEKKÜR

Oğlum Deniz'e…

Teknolojinin giderek hayatımızın bir parçası olduğu günümüzde, mahremiyet kavramı da farklı bir boyuta büründü. Kişisel bilgilerin, haberleşmenin, veri iletişiminin, bilgi paylaşımının gizliliği en önemli konular haline geldi. Tüm bunları sağlamak için kullanılan şifreleme (kriptoloji), gelişen teknoloji sayesinde uygulanması kolay bir hale dönüştü.

Veri güvenliğini sağlamak için kullanılan şifreleme yöntemleri Sezar'dan bu yana çol gelişti ve hayatımızın her alanına girdi. Artık, özel bilgilerin güvenliğini sağlamak için veriler değişik şekillerde şifrelenmekte ve bu şekilde karşı tarafa iletilmektedir. İhtiyaçlar doğrultusunda yeni sistemler sürekli geliştirilmekte ve insanların kullanımına sunulmaktadır. Burada anlatılan çalışmada da veri güvenliği ile ilgili yeni bir sistem geliştirilmiş ve kullanıma sunulmuştur.

Bana tez konusu seçiminde esin kaynağı olan ve tez çalışmam süresince fikir ve yapıcı eleştiriyle desteğini benden esirgemeyen, ihtiyacım olduğunda her türlü teknik desteği sağlayan, grup çalışmasının güzelliklerini gösteren, değerli hocam Sayın Yrd.Doç.Dr. H.Engin DEMİRAY'a teşekkür eder, şükranlarımı sunarım. Ayrıca çalışmam boyunca gerek bilgi gerekse teknik konularda yönlendirmeleriyle bana destek olan Sayın Doç.Dr. Adnan KAVAK ve Sayın Doç.Dr. Yaşar BECERİKLİ'ye de minnetlerimi sunarım.

Yüksek lisans süresince maddi ve manevi desteğini benden esirgemeyen değerli eşim Özlem'e de ayrıca teşekkür ederim.

İÇİNDEKİLER

ŞEKİLLER DİZİNİ

TABLOLAR DİZİNİ

SEMBOLLER

$string$:Herhangi bir string

φ_{sym} :*String*'den random anahtarın elde edilmesini sağlayacak olan algoritma. Algoritmaya aynı string birden fazla kez girilse bile farklı bir anahtar üretecek yapıya sahiptir.

k_{sym} :Alıcıya gönderilecek olan random anahtar. Bu anahtar, simetrik şifreleme algoritmasında kullanılacak olan simetrik anahtarın (k_{group}) üretilmesinde de kullanılmaktadır.

δ_{cons} :Simetrik anahtarı (k_{group}) üretecek olan algoritma.

k_{group} :Simetrik şifrelemede kullanılacak olan anahtar. Bu anahtarın ismini group olarak adlandırdık, çünkü simetrik anahtar olarak bir veya 3DES gibi birkaç anahtar kullanılabilir. Algoritma her ikisine de uyumlu olarak çalışabilmektedir.

m :Şifrelenecek metin, orijinal metin.

c :Simetrik anahtarla şifrelenmiş metin

E^{sym} :Simetrik şifreleme algoritması [10]

H :Hash fonksiyonu.

h :Şifrelenmemiş metnin (plain text) hash değeri.

DQ^{asym} :Asimetrik deşifreleme algoritması.

x :Dijital imza.

y :Anahtar kapsülü (KEM). Dijital imza içine gömülü olarak gönderilen random anahtar k_{sym}, mesajın yolda değişmediğinin (mesaj bütünlüğünün) kontrolünde kullanılan h ve gönderen kişinin kimlik tespitinde kullanılan pk_{sender} bu kapsülün içinde yer almaktadır.

EQ^{asym} :Asimetrik şifreleme algoritması

EW :Karıştırma (Scrabble) algoritması. Şifrelenmiş metin (DEM) ile anahtar kapsülünü (KEM) karıştırıp şifreleyen algoritma

ω :Alıcıya gönderilecek olan nihai mesaj

DW :KEM ile DEM birbirinden ayıran algoritma

Ω :KEM ve DEM'i karıştırma işleminde kullanılacak olan random anahtar

λ :Anahtar kapsülü. En son gönderilecek metne eklenecek olan değer.

β :Orijinal mesajın karıştırılmasından elde edilen metin.

ψ KEM-DEM'in karıştırılmasından elde edilen metin.

Φ :Sayaç. Gönderilen mesajın kaçıncı mesaj olduğunu gösteren bir sayı.

Alt indisler

sk_{sender} :Mesajı gönderen kişinin gizli anahtarı.

pk_{sender} :Mesajı gönderen kişinin açık anahtarı

$pk_{receiver}$:Alıcının açık anahtarı

$sk_{receiver}$:Alıcının gizli anahtarı

Kısaltmalar

AES : Advanced Encryption Standart
C : Şifrelenmiş bilgi ya da metin
D : Şifreyi çözme bloğu
DEM : Data Encapsulation Mechanism
DES : Data Encryption Stantard
E : Şifreleme bloğu ya da şifreleme tekniği
ECC : Eliptic Curve Crpytosystem
K : Anahtar, şifrelenmemiş bilgiye eklenir.
KEM : Key Encapsulation Mechanism
P : Şifrelenmemiş bilgi ya da metin.
PKI : Public Key Infrastructure

VERİ ŞİFRELEMESİNDE SİMETRİK VE ASİMETRİK ANAHTARLAMA ALGORİTMALARININ UYGULANMASI
(HYBRID ŞİFRLEME)

Kerim YILDIRIM

Anahtar Kelimeler: Hybrid şifreleme, KEM-DEM yapısı, veri bütünlüğü, sayısal imza, tüm iletişimin kontrolü, geliştirilmiş oturum anahtarı.

Özet: Şifrelemede kullanılan simetrik ve asimetrik anahtarlı algoritmaların birbirlerine göre avantajları ve dezavantajları bulunmaktadır. Her ikisinin de avantajlarını birleştiren sistem Hybrid şifreleme olarak adlandırılmaktadır. Hybird şifrelemede kullanılan değişik yöntemler mevcut olup burada anlatılan çalışmada yeni key encapsulation mechanism (KEM) ve data encapsulation mechanism (DEM) yapıları sunulmuştur.

Klasik KEM ve DEM yapısında, bu yapıların herhangi birinde bir zayıflık varsa, herhangi bir kişi her ikisine de aynı anda saldırarak şifreyi kırabilir. Yapılan uygulamada bu tarz bir atağı önlemek için klasik yapıya bir karıştırma algoritması eklenmiştir. Bu karıştırma algoritmasında KEM ve DEM öncelikle birleştirilmekte, daha sonra byte'ları permütasyon işlemine tabi tutulmakta ve simetrik anahtar kullanılarak şifrelenmektedir. Burada amaç ikili olan yapıyı tek blok haline dönüştürmektir. Ayrıca uzun süreli olarak kurulacak bir iletişimde kullanılmak üzere yeni bir metot önerilmektedir: Combined KEM-DEM. Combined KEM-DEM metodu mesaj bütünlüğü ve kimlik kontrolünü geliştirilmiş oturum anahtarı ile sağlamaktadır. Bu metotta, her bir mesaj ayrı bir anahtar ile şifrelenmekte ve tüm iletişim trafiği kontrol edilmektedir. RSA algoritmasıyla değişik anahtar uzunlukları (256, 512, ve 1024 bit) kullanılarak yapılan simülasyon göstermiştir ki combined KEM-DEM metodu, Tag-KEM/DEM [14] ve Fujisaki-Okamoto'nun KEM-DEM [15] metodlarıyla karşılaştırıldığında, şifreleme ve deşifreleme zamanını %40 oranında azaltmaktadır.

Scrabbled KEM-DEM metodunun simülasyon programı CD'de simulasyon klasörü altında verilmiştir.

APLICATION OF SYMMETRIC AND ASYMMETRIC ENCRYPTION ALGORITHMS USED IN DATA SECURITY (HYBRID ENCRYPTION)

Kerim YILDIRIM

Keywords: Hybrid encryption, KEM-DEM structure, message integrity, digital signature, whole communication process control, integrated session key.

Abstract: Symmetric and asymmetric algorithms which are used for encyption have advantages compared to each other. Hybrid encryption combines advantages of both symmetric and asymmetric encryption algorithms. There are differant methods used in hybird encryption and in this thesis new key encapsulation mechanisms (KEM) and data encapsulation mechanisms (DEM) are proposed.

In conventional KEM and DEM structure, it is possible that anyone may attack both KEM and DEM and crack one of them, if any weakness exists in these algorithms. In order to prevent such kind of attacks, we propose a scrabble algorithm in which KEM and DEM are combined and permuted, then encrypted with symmetric key driving into one block. In addition, for a long term communication between two nodes, we propose a new scheme combined KEM-DEM which provides message integrity and sender's identity control via an integrated session key. In this scheme, each message is encrypted with different symmetric key and whole communication traffic is controlled. Simulations made using RSA with different key lengths (256, 512, and 1024 bits) show that by employing the combined KEM-DEM scheme, computation times of KEM for the encryption and decryption are reduced by 40% when compared to Tag-KEM/DEM [14] and Fujisaki-Okamoto's KEM-DEM [15] schemes.

The simulation program for scrabbled KEM-DEM scheme is given in the simulasyon folder in the CD.

BÖLÜM 1: GİRİŞ

İnsanlar birbiri ile haberleşirken, haberleştikleri konunun gizliliğini isterler. İşte bu yüzden kriptoloji bilimi ortaya çıkmıştır. Aslında bilgileri gizlemek ya da şifreli haberleşmek daha çok askeri servislerin bilgilerini düşman servislerinden saklamak için kullandıkları bir yöntemdi. Fakat günümüzde hemen hemen bütün durumlarda artık insanlar gizli şekilde haberleşmek isterler. Eğer kriptolojinin geçmişine bakarsak, bilinen en eski şifreleme yaklaşık 4000 yıl önce Mısır'daki firavunların mezarlarından çıkan tabletteki yazılar olarak alınır. Daha sonra ise 17.yüzyılda Phoitas isimli bir bilim adamı bir metin şifrelemiştir ve bu metin hala çözülememiştir.

2000 yıl önce ise Julius Sezar, basit bir kaydırma yöntemi ile şifreleme mantığı oluşturmuştur ve bu şifreleme tekniği Sezar şifresi olarak bilinir.1200'lü yıllarda Robert Bacon bir kaç değişik yöntem ortaya koymuştur. 1460'lı yıllarda ise Leon Alberti çevirme mantığına dayanan bir şifreleme tekniği ortaya çıkarmıştır ve bu frekans analizini olarak bilinir.1585 yılında ise Blaise de Vigenere kriptoloji hakkında bir kitap yayınlayarak, kaydırma yöntemindeki şifreleme mantığını geliştirmiştir. Bu yöntem özellikle savaş alanlarında uygulanmıştır. Fakat teknolojinin gelişmesi ile yeni şifreleme teknikleri ortaya çıkmıştır. Artık bilgilerin daha gelişmiş şifreleme teknikleri ile şifrelenmesine ihtiyaç duyulmuştur.

1.1. Elektronik Güvenlik

E-güvenlik; tüm sistemi ağ yapısı, kullanıcıları ve yazılımları ile bir arada bir bütün olarak düşünmeyi gerektirir. Güvenli bir uygulamanın minimum gereksinimleri şunlardır:

- Gizlilik: Mesaj sadece gönderici ve alıcı kişiler tarafından görülebilmeli.
- Bütünlük: Mesaj gönderici dışında hiç kimse tarafından değiştirilememeli.

- Doğrulama: Mesajı gönderen kişinin kimliği doğrulanmalı, böylece yetkili kişiler dışında hiç kimsenin mesaja erişimine izin verilmemeli.
- İnkar Edememe: Mesajı gönderen kişi mesajı gönderdiğini inkar edememeli. [1]

Bu gereksinimleri karşılamak için kriptolojiden yararlanılır. Özellikle 1990'lı yıllardan başlayarak yaşanan hızlı teknolojik gelişmeler ve internetin yaygınlaşmasının bir sonucu olarak bilişim güvenliği son yıllarda giderek önem kazanan bir konu haline gelmiştir. Konunun önümüzdeki dönemde kurumların öncelik listesinde giderek artan bir öneme sahip olacağı ve kurumların bilişim güvenliği alanına gereken önemi vermeye başladıkları, ilgili önlemleri alma çabası içine girdikleri bilinmektedir. Ancak, bilişim güvenliğinin sadece teknolojik önlemlerle sağlanabileceği gibi genel bir yanılsamanın olduğu da gözlenmektedir. [2]

1.2. Bilişim Güvenliği

1.2.1. Bilişim güvenliği nedir?

1990'lı yıllarda yaşanan hızlı teknolojik gelişmelerin bir sonucu olarak bilgisayarlar, modern hayatın her alanına girmiş ve vazgeçilmez bir biçimde kullanılmaya başlanmıştır. Hayatımızın birçok alanında bilgisayar ve bilgisayar ağı teknolojileri "olmazsa olmaz" bir şekilde yer almaktadır. İletişim, para transferleri, kamu hizmetleri, askeri sistemler, elektronik bankacılık, savunma sistemleri, bu alanlardan sadece birkaçıdır. Teknolojideki bu gelişmeler, bilgisayar ağlarını ve sistemlerini, aynı zamanda, bir saldırı aracı haline, kullandığımız sistemleri de açık birer hedef haline getirmiştir.

Bilişim sistemlerine ve bu sistemler tarafından işlenen verilere yönelik güvenlik ihlalleri inanılmaz bir hızla artmaktadır. Bilişim sistemlerine olan bireysel ve toplumsal bağımlılığımız arttıkça bu sistemlerde meydana gelebilecek arıza ve saldırılara karşı duyarlılığımız da o denli artacaktır. Bu duyarlılık arttıkça da bilgisayar sistemlerine ve ağlarına yönelik olarak gerçekleştirilecek olan saldırıların sonucunda; para, zaman, prestij ve değerli bilgi kaybı da artacaktır. Bu saldırıların hastane bilişim sistemleri gibi doğrudan yaşamı etkileyen sistemlere yönelmesi durumunda ise kaybedilen insan hayatı bile olabilir.

Bilgisayar Güvenliği Enstitüsü (Computer Security Institute - CSI) ve Federal Araştırma Bürosu (FBI) tarafından geleneksel olarak gerçekleştirilen "Bilgisayar Suçları ve Güvenlik Araştırması"nın 2001 yılı raporuna göre bilişim suçları 1997-2001 yılları arasında her yıl neredeyse ikiye katlanacak biçimde artmıştır. Aynı araştırma, gizli bilgilerin çalınması ve finansal kayıtlarda yapılan yasadışı değişikliklerin, en çok maddi zarara neden olan iki saldırı biçimi olduğunu göstermektedir.

1.2.2. Güvenlik prensipleri

Bilişim güvenliğinin birçok boyutu olmasına karşın, temel olarak şekil 1.1'de de gösterilene üç prensipten söz edilebilir: Gizlilik, Veri Bütünlüğü ve Süreklilik.

1.2.2.1. Gizlilik (Confidentiality)

Bilginin yetkisiz kişilerin eline geçmesinin engellenmesidir. Gizlilik, hem kalıcı ortamlarda (disk, tape, vb.) saklı bulunan veriler hem de ağ üzerinde bir göndericiden bir alıcıya gönderilen veriler için söz konusudur. Saldırganlar, yetkileri olmayan verilere birçok yolla erişebilirler: Parola dosyalarının çalınması, sosyal mühendislik, bilgisayar başında çalışan bir kullanıcının, ona fark ettirmeden özel bir bilgisini ele geçirme (parolasını girerken gözetleme gibi). Bunun yanında trafik analizinin, yani hangi gönderici ile hangi alıcı arası haberleşmenin olduğunun belirlenmesine karşı alınan önlemler de gizlilik hizmeti çerçevesinde değerlendirilir.

Şekil 1.1: Temel güvenlik prensipleri

1.2.2.2. Veri bütünlüğü (Data integrity)

Bu hizmetin amacı, veriyi göndericiden çıktığı haliyle alıcısına ulaştırmaktır. Bu durumda veri, haberleşme sırasında izlediği yollarda değiştirilmemiş, araya yeni veriler eklenmemiş, belli bir kısmı ya da tamamı tekrar edilmemiş ve sırası değiştirilmemiş şekilde alıcısına ulaşır. Bu hizmeti, geri dönüşümü olan ve olmayan şekilde verebiliriz. Şöyle ki; alıcıda iki tür bütünlük sınaması yapılabilir: Bozulma Sınaması ya da Düzeltme Sınaması. Bozulma Sınaması ile verinin göndericiden alıcıya ulaştırılması sırasında değiştirilip değiştirilmediğinin sezilmesi hedeflenmiştir. Düzeltme Sınaması'nda ise, Bozulma Sınaması'na ek olarak eğer veride değişiklik sezildiyse bunu göndericiden çıktığı haline döndürmek hedeflenmektedir.

1.2.2.3. Süreklilik (Availability)

Bilişim sistemleri, kendilerinden beklenen işleri gerçekleştirirken, hedeflenen bir başarım (performance) vardır. Bu başarım sayesinde müşteri memnuniyeti artar, elektronik işe geçiş süreci hızlanır. Süreklilik hizmeti, bilişim sistemlerini, kurum içinden ve dışından gelebilecek başarım düşürücü tehditlere karşı korumayı hedefler. Süreklilik hizmeti sayesinde, kullanıcılar, erişim yetkileri dahilinde olan verilere, veri tazeliğini yitirmeden, zamanında ve güvenilir bir şekilde ulaşabilirler.

Sistem sürekliliği, yalnızca kötü amaçlı bir "hacker"ın, sistem başarımını düşürmeye yönelik bir saldırısı sonucu zedelenmez. Bilgisayar yazılımlarındaki hatalar, sistemin yanlış, bilinçsiz ve eğitimsiz personel tarafından kullanılması, ortam şartlarındaki değişimler (nem, ısı, yıldırım düşmesi, topraklama eksikliği) gibi faktörler de sistem sürekliliğini etkileyebilir. Aşağıda, yukarıdaki üç temel prensibe ek olarak ikinci planda değerlendirilebilecek izlenebilirlik, kimlik sınaması, güvenilirlik ve inkâr edememe prensiplerinden bahsedilmiştir.

1.2.2.4. İzlenebilirlik ya da kayıt tutma (Accountability)

Bu hizmetin hedefi sistemde gerçekleşen olayları, daha sonra analiz edilmek üzere kayıt altına almaktır. Burada olay dendiğinde, bilgisayar sistemi ya da ağı üzerinde

olan herhangi bir faaliyeti anlayabiliriz. Bir sistemde olabilecek olaylara, kullanıcının parolasını yazarak sisteme girmesi, bir web sayfasına bağlanmak, e-posta almak göndermek ya da icq ile mesaj yollamak gibi örnekler verilebilir. Toplanan olay kayıtları üzerinde yapılacak analiz sonucunda, bilinen saldırı türlerinin örüntülerine rastlanırsa ya da bulanık mantık kullanılarak daha önce rastlanmayan ve saldırı olasılığı yüksek bir aktivite tespit edilirse alarm mesajları üretilerek sistem yöneticileri uyarılır.

1.2.2.5. Kimlik sınaması (Authentication)

Ağ güvenliği açısından kimlik sınaması; alıcının, göndericinin iddia ettiği kişi olduğundan emin olmasıdır. Bunun yanında, bir bilgisayar programını kullanırken bir parola girmek de kimlik sınaması çerçevesinde değerlendirilebilir. Günümüzde kimlik sınaması, sadece bilgisayar ağları ve sistemleri için değil, fiziksel sistemler için de çok önemli bir hizmet haline gelmiştir. Akıllı karta ya da biyometrik teknolojilere dayalı kimlik sınama sistemleri yaygın olarak kullanmaya başlanmıştır.

1.2.2.6. Güvenilirlik (Reliability - Consistency)

Sistemin beklenen davranışı ile elde edilen sonuçlar arasındaki tutarlılık durumudur. Başka bir deyiş ile güvenilirlik, sistemden ne yapmasını bekliyorsak, sistemin de eksiksiz ve fazlasız olarak bunu yapması ve her çalıştırıldığında da aynı şekilde davranması olarak tanımlanabilir.

1.2.2.7. İnkâr edememe (Non-repudiation)

Bu hizmet sayesinde, ne gönderici alıcıya bir mesajı gönderdiğini ne de alıcı göndericiden bir mesajı aldığını inkâr edebilir. Bu hizmet, özellikle gerçek zamanlı işlem gerektiren finansal sistemlerde kullanım alanı bulmaktadır ve gönderici ile alıcı arasında ortaya çıkabilecek anlaşmazlıkların en aza indirilmesini sağlamaya yardımcı olmaktadır. Bu hizmetler, zaman içinde bilgisayar sistemlerine karşı ortaya çıkmış tehditler ve yaşanmış olaylar sonucunda ortaya konmuştur. Yani her bir hizmet, belli bir grup potansiyel tehdide karşı sistemi korumaya yöneliktir, denilebilir. [2]

1.3. Kriptoloji Terminolojisi

Kriptoloji: Gizli karakterleri yaratma, çözme ve bu karakterlerin iletilmesini de içeren bilimsel çalışma olarak isimlendirilir. Matematiksel bir bilimdir.
Kriptografi: Gizlilik sistemlerinin tasarımı ve uygulaması ile ilgilenen kısımdır.
Kriptoanaliz: Gizliliğin kırılmasına ilişkin metodları ve sistemleri inceler.
Key: Şifre araçlarına verilen isimdir.
Plaintext: Orjinal metin
Ciphertext: Şifrelenmiş veri ya da bilgi
Encryption (Şifreleme): Plaintext'i Ciphertext'e dönüştürme işlemidir.
Decryption (Çözümleme): Ciphertext'i plaintext'e geri dönüştürme işlemidir.
Cryptoanalist (Attacker): Plaintext'i geri elde etmeye iz bulma (pattern finding) çalışan kimse.

1.3.1. Kriptoloji

Veriyi şifrelemek ve çözmek için kullanılan matematiksel bir bilimdir. Böylece hassas bilgilerin saklanması ya da güvensiz ağlar üzerinden güvenli bir şekilde aktarılması sağlanır. Bir bilim olarak ortaya çıkışından beri güvenli iletişimi analiz etme ve kırma bilimi olarak ortaya çıkmıştır. Klasik kriptoanaliz, analitik sebeblerin-matematiksel uygulamaların, iz bulma işlemi sabrın ve şansın bir birleşimidir. Kriptoanalistler attacker olarak da adlandırılırlar. Kriptoloji, kriptografi ve kriptoanalizin bir birleşimidir.

Kriptografinin güçlülüğü, şifrelenmiş verinin orijinal metine dönüştürülmesi için geçen süre ve kaynak miktarı ile ölçülebilir. Güçlü kriptografi anlayışında şifrelenmiş text'in uygun decode aracı olmadan deşifre edilmesi zordur. Fakat günümüzde bilgisayarların işlem gücü bunu imkansız kılmamaktadır. Hiç kimse ben mükemmel bir şifreleme algoritması geliştirdim diyemez.

1.3.2. Kriptografi

Kriptografik algoritma şifreleme ve şifre çözme işlemini gerçekleştirmek için kullanılan matematiksel bir fonksiyondur. Kripto algoritmasının çalışması için bir anahtar (key) ve şifrelenecek bir ifade gerekir. Farklı anahtarlar kullanıldığında aynı şifrelenmemiş metinden farklı şifrelenmiş metinler üretilebilir. Şifrelenmiş bilginin güvenliği iki şeye bağlıdır;

- Algoritmanın Güçlülüğüne
- Anahtarın Gizliliğine

Kriptografik algoritma tüm olası anahtarların ve tüm protokollerin toplamıdır. Kriptografi, veriyi yalnızca okuması istenen şahısların okuyabileceği bir şekilde saklamak ve göndermek amacıyla kullanılan bir teknolojidir. Kriptografi'de veri, matematiksel yöntemler kullanılarak kodlanır ve başkalarının okuyamayacağı hale getirilir. Bu matematiksel kodlamaya "kripto algoritması" adı verilir.

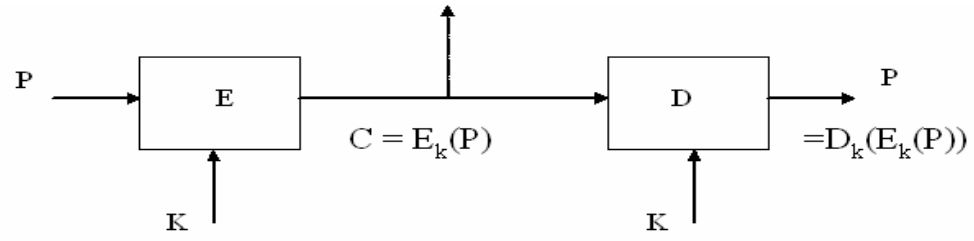
Şifrelenmemiş bir bilgiye "açık metin" (clear text veya plain text) denir. Açık metin, bir insanın okuyabileceği bir yazı ya da bir bilgisayarın anlayabileceği çalıştırılabilir (.exe, .com) bir program ya da bir veri dosyası (.txt) olabilir. Bir kripto algoritması kullanılarak, herkesin okuyamayacağı bir şekilde kodlanmış bilgiye ise "şifreli metin" (cipher text) denir. Açık metinden şifreli metne geçme işlemi "şifreleme", şifreli metinden açık metne geçme işlemi ise "şifre çözme" olarak adlandırılır. Şifreleme ve şifre çözme yapan bir sistem de "kriptosistem" olarak adlandırılır. Bir kriptosistemin, şifreleme ve şifre çözme yapan hem donanım hem de yazılım bileşenleri olabilir. [2]

Algoritmalar, açık metin üzerinde yapılan karmaşık işlemlerden oluşan matematiksel formüllerdir. Bir algoritma, hem yazılımla hem de donanım bileşenleri ile gerçekleştirilebilir. Birçok algoritma, şifreleme ve şifre çözme işlemini gerçekleştirmek amacıyla, açık metinden başka, "anahtar" denen bir değer de kullanır. Anahtar "0" ve "1" lerden oluşan uzun bir bit dizisidir. Her algoritmanın kullandığı anahtar boyları farklıdır. Genellikle anahtar boyu arttıkça, olası anahtar

sayısı arttığından, saldırganın bu şifreyi çözmesi güçleşir, ama aynı zamanda da şifreleme ve şifre çözme hızı yavaşlar. Bir algoritmanın olası tüm anahtarlar olasılıklarının oluşturduğu topluluğa "anahtar uzayı" denir. [2]

Kriptoloji şifreleme ile çözme işleminin bir arada yapılabildiği bir çalışma alanıdır. Burada iki önemli unsur vardır. Bunlar;

- Şifreyi yaratanlar
- Şifreyi çözmeye çalışanlar



Şekil 1.2: Şifreleme ve şifre çözme işleminin blok şeması

- P: Şifrelenmemiş bilgi ya da metin.
- K: Anahtar, şifrelenmemiş bilgiye eklenir.
- C: Şifrelenmiş bilgi ya da metin
- E: Şifreleme bloğu ya da şifreleme tekniği
- D: Şifreyi çözme bloğu

Burada zorla sisteme girmeye çalışan kimsenin amacı tabii ki şifrelenmiş bilgileri elde etmek ve onları çözmek olacaktır. Şekil 1.2'deki sisteme göre de şifrelenmiş bütün bilgileri alıp okuyabilir. Fakat şifreleme tekniğini bilmiyor ise sadece şifrelenmiş bilgileri okuyabilir. Bu yüzden şifreleme tekniğiniz çok önemlidir. Çünkü uygun çözücü algoritmanız olmadan şifrelenen bilgiyi anlamanız oldukça zordur.

1.4. Genel Şifreleme Sistemi

Şifreleme teknikleri genel anlamda 2'ye ayrılır. [3,4]

- Simetrik Anahtarlı Sistemler
- Asimetrik Anahtarlı Sistemler

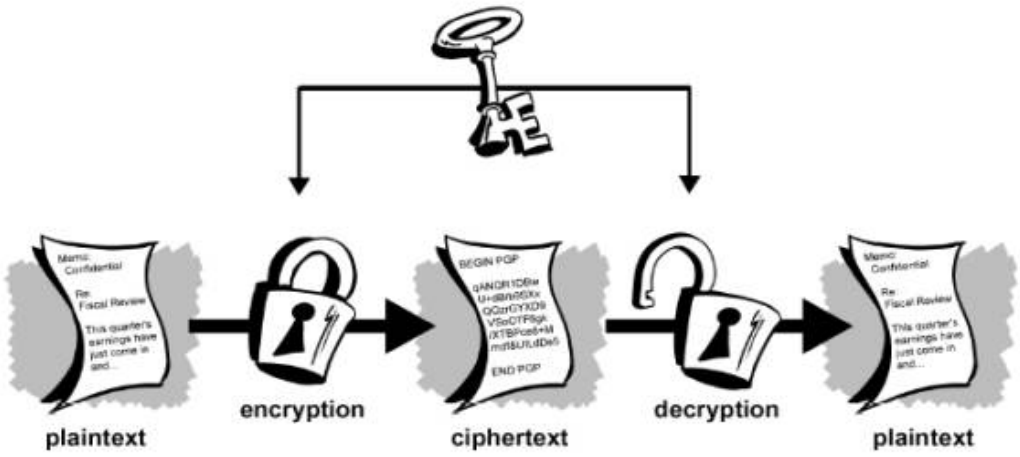
Simetrik anahtarlı ve asimetrik anahtarlı şifreleme algoritmalarını kapsayan iki ana başlık gibi düşünülebilir.

1.4.1. Simetrik anahtarlı (Convential-anlaşmalı) sistemler

Genel yapısı şekil 1.3'de gösterilen simetrik anahtarlı algoritmalarda şifreleme ve şifre çözme için aynı anahtar kullanılır. Bu anahtara gizli anahtar (secret key) denir. Bu gizli anahtar iki tarafça da (gönderici ve alıcı) bilinir.

Simetrik algoritmalar asimetrik algoritmalara nazaran daha hızlı çalışırlar. Bununla beraber, asimetrik algoritmalara nazaran saldırıya karşı daha az dirençlidirler. Simetrik algoritmalara örnek olarak AES, DES, 3DES, Blowfish, IDEA ve RC4 algoritmaları verilebilir. [3-5]

Şifreleme ve çözücü algoritmasının anahtarları haberleşen kişiler tarafından bilinmesi gerekmektedir. Eğer şifreleme anahtarı biliniyor ise bilginin elde edilmesi oldukça kolaydır. 1970'den önceki bütün kripto sistemleri simetrikti. Bu tür sistemlere örnek verecek olursak; DES (Data Encryption Stantard) [6] ve AES [7] (Advanced Encryption Standart). Burada önemli olan sorun anahtarın kimseye ulaşmadan alıcıya ulaşmasıdır.



Şekil 1.3: Simetrik anahtar şifrelemesi

Simetrik anahtarlama sistemlerinin en basit uygulamalarından biri olan yerine koyma algoritması olarak da bilinen Sezar şifresidir. Bu algoritmada bir parça bilgi, diğerinin yerine koyulur. Bu tür algoritmalar alfabe kaydırma işlemi esas alınır. Fakat günümüzde oldukça zayıf kalan bir algoritmadır. Simetrik anahtarlı sistemlerin avantaj ve dezavantajlarını şöyle sıralayabiliriz;
Avantajları;

- Çok hızlıdır. Şifrelenmiş verinin bir yere gönderilmediği durumlarda kullanışlıdır.
- Şifrelenmiş verinin gönderilmesi sırasında şifrenin gizli tutulması maliyeti artıran bir uygulamadır.
- Bu tasarımda gönderici ve alıcı bir anahtar üzerinde anlaşmalı ve bu anahtar aralarında gizli kalmalıdır.

Dezavantajları;

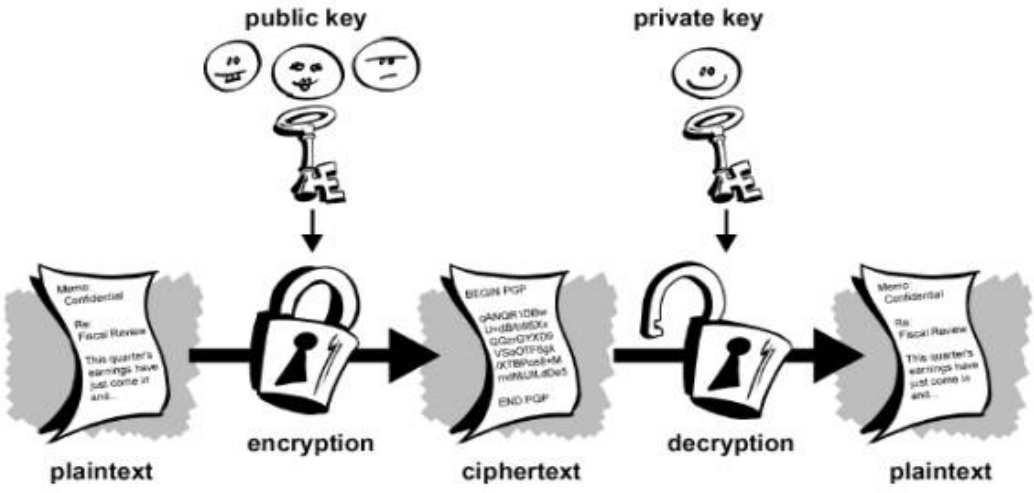
- Eğer kullanıcılar farklı yerlerde ise gizli anahtarın gönderilmesi sırasında hatların güvenliğinden emin olunmalıdır.
- Herhangi birisi anahtar değerini dinleyebilir ya da değiştirebilir.
- Burada ana problem anahtar dağıtma problemidir.

1.4.2. Asimetrik anahtarlı sistemler

Şifreleme ve şifre çözme için ayrı anahtarlar kullanılır. Bu anahtarlardan birine açık anahtar (public key), diğerine özel anahtar (private key) denir ve asimetrik anahtarlı sistemlerin genel yapısı şekil 1.4'te gösterilmiştir. Kullanılacak bu iki anahtar birlikte üretilirler. Bununla birlikte bu anahtarlardan herhangi birine sahip olan bir şahıs, diğer anahtarı üretemez, bu matematiksel olarak imkansız denebilecek derecede zordur.

Asimetrik algoritmalar, simetrik algoritmalara göre daha güvenli ve kırılması zor algoritmalardır. Bununla birlikte, başarımları (performans) simetrik algoritmalara göre oldukça düşüktür. Asimetrik algoritmalarda her şahsın bir anahtar çifti vardır. Bir şahsın özel anahtarı, yalnızca kendi kullanımı içindir ve başkalarının eline geçmemesi gerekir. Bu şahsın açık anahtarı ise, bu şahsa mesaj göndermek isteyen herhangi biri tarafından kullanılabilir. Gönderici mesajı, alıcının açık anahtarı ile şifreler. Alıcı, gelen mesajı kendi özel anahtarı ile açar.

Mesaj gönderebileceğimiz kullanıcıların sayısı arttıkça, elde etmemiz gereken açık anahtar sayısı da artacaktır. Sistemde 100 kullanıcı varsa, her bir kullanıcının ayrı bir açık anahtarı olacağından, tüm bu açık anahtarlar, erişilebilir olmalıdır. Bu problem de sayısal sertifikalar teknolojisi yardımı ile çözülebilmektedir. [2]



Şekil 1.4: Asimetrik anahtarlı sistem

Simetrik sistemlerde özellikle geniş ağ yönetiminde anahtar kullanımı ve dağıtımı oldukça zordur. Bu sistem ile anahtar kullanımı ve anahtar yönetimi problemleri ortadan kaldırılmıştır. [5]

Özel anahtarınızı gizli tutarak açık anahtarınızı tüm dünyaya yayabilirsiniz. Açık anahtara sahip bir kişi bilgiyi sadece şifreleyebilir fakat çözemez. Yalnızca özel anahtara sahip olan kişi bilgiyi okuyabilir.

Asimetrik Anahtar sisteminde gönderici ve alıcının gizli anahtarları paylaşmaları gereksinimi ortadan kalkmıştır. Tüm iletişimler sadece açık anahtar üzerinden gerçekleştirilir. Özel anahtarınız hiç bir şekilde paylaşılmaz ya da gönderilmez. Bazı açık anahtar kripto sistemleri örnekleri Elgamal, RSA, ECC, Diffie-Hellman ve DSA'dır.

Görüldüğü gibi simetrik ve asimetrik algoritmaların birbirlerine göre bir takım üstünlükleri ve zayıf yönleri vardır. Her iki algoritma grubunun üstünlüklerinden faydalanarak zayıf yönlerini bir kenara bırakmak amacıyla "hibrid kripto sistemler" kullanılmaktadır. Bu tür sistemlerde hem simetrik hem de asimetrik algoritmalarla hem başarımı hem de güvenliği yüksek şifreleme yapılabilmektedir. [2]

1.4.3. Hybrid şifreleme

Bu yöntem simetrik ve asimetrik sistem algoritmalarının iyi yönlerini birleştirmiştir. [8] Bu tasarımın en güzel örneği PGP'dir.

PGP her iki algoritmanın iyi özelliklerini birleştiren bir yöntemdir. Bu yöntemde kullanıcı veriyi PGP algoritması ile şifrelediğinde, PGP öncelikle veriyi sıkıştırır. Veriyi sıkıştırma işlemi modem iletim zamanı ve disk alanı bakımından kar sağlar aynı zamanda güvenliği artırır. Birçok şifre çözme metodu, şifrelenmemiş veri üzerindeki exploit şablonların bulunması ve şifrenin çözülmesi esasına dayanır. Sıkıştırma işlemi şifrelenmemiş veri üzerinde bu bölgeleri azalttığı için saldırganlara karşı direnç kazandırılmış olur. Daha sonra session key oluşturulur. Bu anahtar rasgele bir numaradır ve kullanıcının rasgele mouse hareketlerinden ve bastığı

tuşlardan oluşturulur. Session key çok güvenli ve hızlı simetrik anahtar algoritması ile açık metni şifreler. Bu şifrelemeden sonra bir de açık anahtar kullanılarak şifrelenir. Şifrelenmiş verinin çözülme işlemi de ters şekilde çalışır. Öncelikle özel anahtar ile sesion key çözülür. Daha sonra da şifrelenmiş veri çözülür. Bu algoritma açık anahtar algoritmasından 1000 kez daha hızlı çalışır. Anahtar paylaşımı ve verinin iletim problemlerine çözüm getirir. Güvenli bir şekilde uygulanabilir.

1.5. Anahtar (Key)

Anahtar bir sayıdır ve kriptografik algoritma ile çalışır. Anahtarlar temel olarak çok büyük sayılardır. Örneğin 2048 bit gibi. Asimetrik anahtar sistemlerinde büyük anahtar kullanılması daha güvenli şifrelenmiş bilgi oluşturulmasını sağlar. Bir simetrik 80 bit anahtar, 1024 bit açık anahtar gücüne sahiptir.

Anahtarları doğru büyüklükte seçmek çok önemlidir. Büyük anahtarlar sağlam güvenlik sağlarken, küçük anahtarlar işlem zamanından tasarruf sağlarlar. Büyük anahtarların kırılması uzun zaman gerektirir. Şifrelenmiş verilerinizin yıllarca saklanmasını düşünüyorsanız büyük anahtar kullanabilirsiniz. Tabii gelecekte bilgisayarların ne kadar güçlü ve etkili olabileceklerini kim bilebilir ki. Anahtarlar da şifrelenmiş olarak saklanırlar.

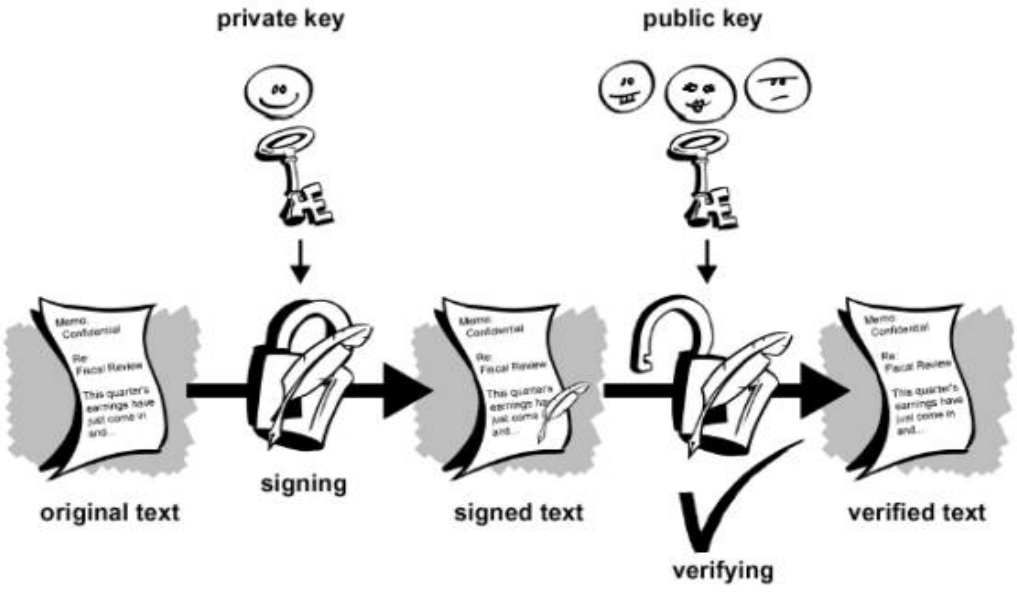
1.6. Özetleme Fonksiyonları

Bir özetleme fonksiyonu, herhangi bir uzunluktaki metni, giriş değeri olarak alır ve sonuç olarak sabit uzunluklu bir değer üretir. Bu değere mesaj özeti (message digest) adı verilir. Burada üretilen özet, fonksiyona giren metnin karakterini taşımaktadır denilebilir. Giriş metninde yapılacak tek bir karakter değişikliği bile üretilecek özette büyük değişikliklere yol açar. Ayrıca, özetleme fonksiyonu tek yönlü olduğundan, özetten asıl metne geri dönüş yoktur. Özetleme fonksiyonları, uzun metinlerin, asimetrik bir algoritma ile şifrelenmeleri sırasında, asimetrik algoritmanın başarım dezavantajını ortadan kaldırmak amacıyla kullanılırlar. Tüm mesaj metni değil de yalnızca mesajın özeti alınarak asimetrik algoritmayla şifrelenir. Özetleme algoritmalarına örnek olarak SHA-1, DSS, MD2, MD4 ve MD5 algoritmaları

verilebilir. [2, 9]

1.7. Sayısal İmzalar

Sayısal imzalar alıcının bilgilerini doğrulamak için kullanılır ve bu yöntem kullanılarak verinin iletim esnasında değiştirilmediğinden emin olunur. Böylece sayısal imzalar, authentication ve veri bütünlüğü sağlarlar. Bir sayısal imza aynı zamanda, göndericinin gerçekten veriyi göndermediği halde gönderdiğini iddia etmesine engel olur.



Şekil 1.5: Sayısal imza oluşumu

Sayısal imzalar el imzaları ile aynı amacı taşımaktadır. El imzalarını taklit etmek kolaydır. Sayısal imzalar el imzalarından daha üstündürler taklit edilmeleri neredeyse imkansızdır. Sayısal imzanın genel gösterimi şekil 1.5'te olduğu gibidir. [10]

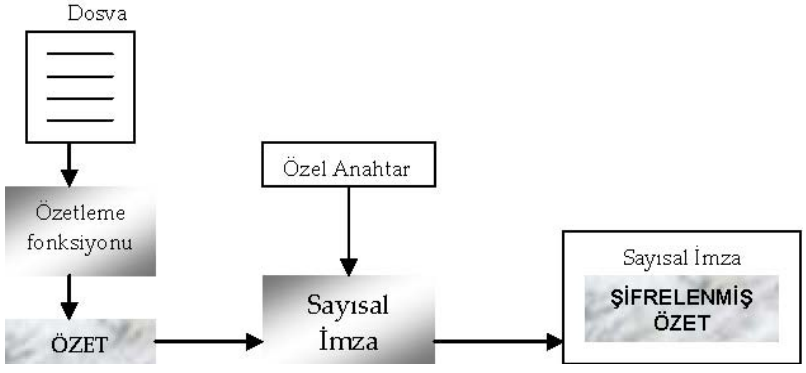
Bir sayısal imza, şifrelenmiş bir özet (hash) değeridir. Sayısal imzalar yardımıyla, alıcı taraf göndericinin kimliğinin sınamasını yapar ve göndericinin kim olduğundan tam olarak emin olur. Bunun yanında, sayısal imza teknolojisi, gönderilen verilerin bütünlük sınamasında da kullanılabilir. Buna göre sayısal imza teknolojisi, daha önce bahsedilen Kimlik Sınaması ve Veri Bütünlüğü prensiplerinin gerçekleştirilmesinde kullanılırlar. [2]

Sayısal imzalar, gerçek hayatta kullanılan ve elle atılan imzanın (ıslak imzanın) bilişim dünyasındaki karşılığı olarak görülebilir. Bir sayısal imza, imzaladığı içeriğin, imzalandığı andan itibaren değişmediğinin kanıtlanmasında kullanılabilir.
Sayısal imzalama, asimetrik kripto algoritmaları yardımı ile yapılır. Sayısal imzalama, mesaj bir mektup zarfına konulduğunda üzerinin mühürlenmesi gibi düşünülebilir. Sayısal imzalar, bilgisayar ağları yoluyla yapılan finansal işlemlerin güvenli bir şekilde yapılması ve veritabanı bütünlüğünün kontrolü gibi kullanım alanları bulmuşlardır.

Sayısal imzalar, Açık Anahtar Altyapısı (Public Key Infrastructure – PKI) teknolojisinin de belkemiğini oluşturur. PKI, çok geniş bir coğrafi alana yayılmış kullanıcılar arasında, güvenli bir haberleşme altyapısı kurmayı hedefleyen bir teknolojidir. PKI'yı oluşturan elemanlardan başlıcaları şunlardır [2].

1.8. Sayısal Sertifikalar

Bir sayısal sertifika, gerçek hayatta kullanılan bir kimlik kartının, bilişim güvenliğindeki karşılığıdır. Bir sertifikanın içinde; sahibinin kimlik bilgileri, yetki derecesi, sertifikanın son kullanma tarihi, sahibinin kripto anahtarı bilgisi yer alır. Bir sayısal sertifika, bir kullanıcının bir sisteme girerken kimlik sınamasının yapılmasında ya da kriptolu e-posta mesajlarının gönderilmesinde kullanılabilir. Sayısal sertifikalar için ISO tarafından X.509 standardı yayınlanmıştır ve bu standart yaygın olarak kullanılmaktadır. Sayısal imzanın nasıl oluşturulduğu şekil 1.6'da gösterilmiştir.



Şekil 1.6: Bir mesajın sayısal imzasının oluşturulması

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 şifreleyerek 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. Ancak web sitesinin gerçekten dürüst bir satıcı mı, yoksa sahte bir site mi olduğundan emin olamayacaktır. Bunun da çözümü SERTİFİKA yöntemiyle sağlanmaktadır. [2]

Açık anahtarlı kriptosistemler doğru kişinin anahtarının şifrelendiğinden emin olmak için sürekli tetikte bulunmalıdır. Anahtarların sunucular yolu ile paylaşıldığı bu çevrelerde "man-in-the-middle" saldırıları önemli bir tehdit oluşturmaktadır. Bu tip saldırılarda herhangi birisi alıcının adı ve User ID'si ile birlikte sahte bir şifre gönderebilir. Artık veri başkasının elinde olan sahte anahtar ile şifrelenmiştir. Açık anahtar ortamında verinin; gönderilmek istenen kişinin açık anahtarı ile şifrelenmesi hayati önem taşır. Fiziksel olarak elinizde olan anahtarları kullanarak kolayca şifreleme yapabilirsiniz. Fakat hiç karşılaşmadığınız bir insanla bilgi değiş tokuşu yaptığını düşünelim. Elimizde gerçek anahtarın olduğundan nasıl emin olabiliriz ki. Sayısal sertifikalar görevi basitçe anahtarın gerçek kişiye ait olup olmadığının kontrolüdür. Bu sertifika kimlik kartı gibidir. Örneğin bazı sertifikalarda kimliğinizi doğrulamanız gerekmektedir. Bu yüzden bunları kaybetmemelisiniz. Aksi takdirde bir başkası sizin yerinize geçebilir. Sayısal sertifikaların fonksiyonları da fiziksel sertifikalar ile aynıdır.

Sayısal Sertifikalar şu bilgileri içerirler;

- Açık anahtar
- Sertifika Bilgisi
- Bir ya da daha fazla sayısal imza

Sertifikasyon Otoriteleri: Bir PKI sistemine dahil olan kullanıcılar için sayısal sertifika üretim ve saklama merkezleridir. Bir kullanıcıya mesaj gönderilirken ya da

gelen bir mesajdaki sayısal imzanın doğruluğunun sınanırken, o alıcının açık anahtarına ihtiyaç duyulur. Bu açık anahtarı elde etmek için, sertifikasyon otoritesinden, kullanıcının kimliği yardımı ile kullanıcının sayısal sertifikası elde edilir. Kullanıcının açık anahtarı, bu sertifika içerisinden alınarak kullanılır. [2]

1.9. Kriptonun Avantajları

Öncelikle kriptonun ne için gerekli olduğunu ve buradaki terminolojiden bahsettik. Şimdi de kriptolojinin sağladığı avantajlardan ve kullanıldığı uygulamalardan bahsedelim. Kriptonun bize sağladığı avantajlar;

- Güvenlik
- Doğruluk
- Güvenirlilik
- Özdeşlik

1.10. Kriptonun Uygulama Alanları

Günümüzde kriptoloji her tarafta kullanılmaktadır. Özellikle bilgilerinizin korunmasını ve bu bilgilerin korunarak iletilmesini isterseniz kriptoya ihtiyaç vardır. Bu yüzden kullanım alanı oldukça geniştir. Aşağıda gösterilen uygulamalarda kripto kullanılmaktadır.

- Haberleşme
- Dosya ve bilgi güvenliğinde
- Elektronik Ticaret
- Sayısal imza
- Elektronik Posta
- Güvenlik protokolleri ve uygulamalarında

1.11. Bazı Şifreleme Teknikleri

1.11.1. Shift cipher

Shift cipher en eski şifreleme ve klasik yöntemlerden biridir ve Sezar şifrelemesi olarak da bilinir. Çünkü ilk kullanan Julius Caesar'dır.

- P=C=K Є Z $0 \leq k \leq 25$ (kєK)
- ek(x)=(x+k) mod 26 (x,y) Є Z (İngiliz harf kümesi)
- dk(x)=(y-k) mod 26
- Burada her harfe bir sayı karşılık getireceğiz.

A B C .....X Y Z
0 1 2 ..... 23 24 25
Burada k harfleri ne kadar kaydırdığımızı göstermektedir. Bir kaç örnek verecek olursak;

Örnek=We will meet at midnight (k=11)
W=(22+11) mod 26 = 7
M=(12+11) mod 26 = 23

Açık Metin = KERIM YILDIRIM VERI GUVENLIGI (k=8)
Şifreli Metin =SMZQU HQTLQZQU DMZQ OCDMVTQOQ

Açık Metin = EN BUYUK FENERBAHCE (k=15)
Şifreli Metin =TC QJNJZ UTCTHQPWRT

Burada sadece kaydırma yapılmıştır ve bir başka harf yerine verilen k değeri kadar alfabe kaydırılarak bulunan harf yerleştirilmiştir. Şifre Çözme işlemi de tersi yapılarak bulunur.

1.11.2. Affin cipher

- Bir x uzayını y uzayına çeviren şifrelemedir. Lineer bir dönüşüm fonksiyonudur.
- Y=ax+b, P=C ε Z
- K{(a,b)εZ*Z: ebob(a,26)=1}
- Z={1,2,3.....,m-1} m(26) bölündüğünde kalan kümesi
- k=(a,b)εK
- ek(x)=(ax+b) mod 26
- dk(y)=a^{-1}(y-b) mod 26

Örnek verecek olursak;

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

Açık Metin = FIRE AT NOON

Burada a=3 ve b=8 alınırsa;

Alfabedeki numara karşılıkları= 5 8 17 4 0 19 13 14 14 13

$f(5) = (3(5) + 8) \bmod 26$

$= 23 \bmod 26.$

$f(8) = (3(8) + 8) \bmod 26$

$= 6 \bmod 26.$

Yukarıdaki gibi bütün harfler dönüştürülürse oluşan numara dizisi aşağıdaki gibi olur;

23 6 7 20 8 13 21 24 24 21

Oluşan cümle ise; XGHU IN VYYV'dir.

Bunun şifre çözme işlemi ise;

23=(3*(?)+8) Mod 26ise

?=(23-8)/3(mod 26)'dan 5 çıkar.

6=(3*(?)+8)Mod 26 ise ?=8 çıkar.

Buradan şifre çözme işlemi de çözülmüş olur.

1.11.3. Substitution cipher

- Alfabeyi herhangi bir sırada sıralıyoruz ve yalnızca bir tane permütasyonu alıyoruz.
- Alfabe 26! şeklinde sıralanabilir.
- $\pi \varepsilon K$ olmak üzere
- $e(x)= \pi(x)$
- $d(\pi)= \pi\text{-}1(y)$
- $\pi^* \pi\text{-}1=I$

Burada alfabeyi değişik şekilde sıraladıktan sonra şifreleyeceğiniz metindeki harfleri, sıraladığınız alfabedeki harflere göre yerine koyuyorsunuz.

Örneğin ;
Açık Metin = KERIM YILDIRIM
Seçilen alfabe= QFSHMRGUTDIJEPYCNWZXLBVAOK
Şifreli Metin = IMNTE OTJHTNTE

Açık Metin = BILGISAYAR AGIMIZA VIRUS SALDIRISI OLACAKTIR
Seçilen Alfabe= FDGZECOWKLNQTJIRXSYAMUPHVB
Şifreli Metin = DKQOKYFVFS FOKTKBF UKSMY YFQZKSKYK IQFGFNAKS

Bu şifrenin çözülmesi içinde hangi sıralamayı kullandığımızın karşı taraf tarafından bilinmesi gerekmektedir.

1.11.4. Viganure cipher

- $m \varepsilon N$
- P=K=C=(Zm)
- Zm= (Z içinde m'li gruplar)
- k=(k1,k2,k3.....km)
- e=(x1,x2,.....xm)=(x1+k1,x2+k2,....xm+km)(mod26)
- d=(y1,y2,.....ym)=(y1+k1,y2+k2,....ym+km)(mod26)

Örnek;

Kelime=cipher

m=6

k=(2,8,15,7,4,17)

c i p h e r

p='thiscryptosystemisnotsecure'

19 7 8 18 2 17 24 15 19 14 18 24 18 19...

2 8 15 7 4 17 2 8 15 7 4 17.....

X1+k1=21(mod26)=

21 15 26 25 6 8 15

V P X Z G I P

Örnek = Açık Metin=BU SENE FENERBAHCE SAMPIYON OLACAK

Kelime = GALATASARAY

Şifreli Metin = HUDEGEXEEEPHASCXSSMGWUNZLTCSK

1.11.5. Hill cipher

- $P=C=K=Z^m$
- K={(m*m) tekil olmayan matris (mod26))
- $e(x)=x$, $d(y)=y*k^{-1}$
- (y1,y2,...ym)=(x1,x2,...xm)*

$$\begin{pmatrix} C_k \\ C_{k+1} \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} P_k \\ P_{k+1} \end{pmatrix} \pmod{26}$$

$$C_1 \equiv aP_1 + bP_2 \pmod{26}$$
$$C_2 \equiv cP_1 + dP_2 \pmod{26}$$

şeklinde şifrelenebilir. Şifre çözülme olayı ise ;

$$\begin{pmatrix} P_1 \\ P_2 \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix}^{-1} \begin{pmatrix} C_1 \\ C_2 \end{pmatrix} \pmod{26}$$

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix}^{-1} = \frac{1}{ad-bc} \begin{pmatrix} d & -b \\ -c & a \end{pmatrix}$$

şeklinde de şifre çözülebilir.

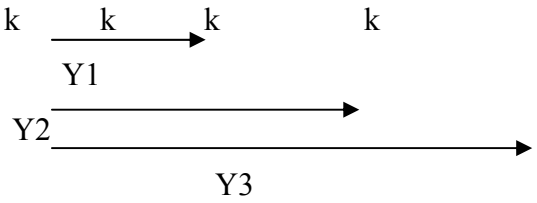
Örnek: AG VE VERI GUVENLIGI cümlesini , $\begin{bmatrix} 11 & 5 \\ 8 & 13 \end{bmatrix}$'e göre şifrelersek;

çıkan cümle "TUGGG GIAZQ GGFHD GKT" olacaktır.

1.11.6. Stream cipher

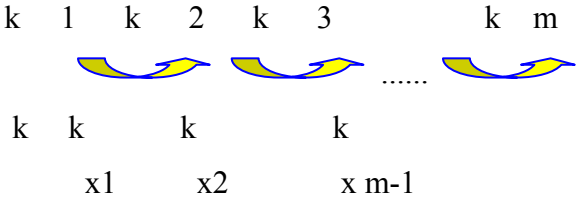
Bloklama ile tek tek şifreleme arasında çok fark yoktur. Anahtarı her seferinde değiştirirsek, dönüşümde anahtar bir önceki açık metne uygulanacaktır. Başlangıçtaki anahtar gönderici tarafından belirlenebileceği gibi belli bir algoritmaya da bağlı olabilir veya bir önceki açık metne göre de yeni anahtar belirlenebilir.

$Y = Y1, Y2, \ldots, Ym = e(x1) * e(x2) * e(x3) * \ldots * e(xm)$



$ki = f(k, x, x, \ldots, x)$ keystream i. Elemanı x'e bağlıdır.
1 2 m-1 m-1

$Y = e(x)*e(x)*e(x) * \ldots * e(x)$



örnek:

k=8

p='rendezvous'

17 4 13 3 4 25 21 14 20 18 1- plaintext

x x x x x x x x x x

1 2 3 4 5 6 7 8 9 10

8 17 4 13 3 4 25 21 14 20 2- text with key

k x x x x x x x x x

1 1 2 3 4 5 6 7 8 9

17 4 13 3 4 25 21 14 20 18 3- (text+plain) mod26

x x x x x x x x x x

1 2 3 4 5 6 7 8 9 10

17 4 13 3 4 25 21 14 20 18 3- (text+plain) mod26

x x x x x x x x x x

1 2 3 4 5 6 7 8 9 10

↓ ↓

(x1+x2)mod26 (x10+x9) mod26 4- gönderirken

25mod26 38mod26

e (x): 25 21 17 16 7 3 20 9 8 12

z

Z V R Q H D U J I M

ALİCE ---------- >>>>>>>> BOB 5- alırken

X = d (25)= (25-8)mod26= 17 R

1 8

X = d (21)= (21-7)mod26=4 E

2 17

……. N 6- tekrar dönüştürüldü

……. D

……. E

……. Z

……. V

……. O

X = d (8)= (8-14) mod26 = 20 U

9 14

X = d (12)=(12-20)mod26=18 S

10 20

BÖLÜM 2: ASİMETRİK ANAHTARLI ŞİFRELEME SİSTEMLERİ

Şifreleme bir gereksinim olarak ortaya çıkmasından 20. yüzyılın son çeyreğine kadar hep tek bir anahtar kullanılarak gerçekleşti. Yani veriyi şifrelerken ve şifreli veriyi çözerken kullanılan anahtar aynıydı. Bu yöntem simetrik anahtarlı şifreleme, gizli anahtarlı şifreleme veya geleneksel şifreleme olarak adlandırılır. Simetrik anahtarlı şifreleme sistemleri temel olarak çeşitli yer değiştirme ve permütasyon işlemlerine dayanmaktadır. Bu tip şifreleme yöntemlerinin en basitini Julius Casear'ın savaş alanına komutanları ile haberleşmek için kullandığı alfabedeki harflerin belirli bir sayı kadar kaydırılıp verinin bu yeni alfabe ile kodlandığı şifreleme yöntemidir. Simetrik anahtarlı şifreleme sistemlerinin en bilineni ise IBM tarafından geliştirilen DES (Data Encryption Standart) yöntemidir. Bu yöntem 1997 yılında yüzlerce işlemciye sahip bir süper bilgisayar ile kırılabilmiştir. Daha sonra geliştirilen 3-DES ise hala güvenliğini korumaktadır.

Açık anahtarlı (Asimetrik) şifrelemenin veri şifrelemede yeni bir çağ açtığı söylenebilir. Bu tip yöntemlerde simetrik şifreleme tekniklerinin aksine veriyi şifrelemek ve şifreyi çözmek için farklı anahtarlar kullanılmaktadır [3-5]. Böylece simetrik anahtarlı şifreleme sistemlerinin en büyük sorunlarından birine çözüm getirilmiştir. Her iki şifreleme sistemi de günümüzde kullanılmaya devam etmektedir. Bu sistemlerden birinin diğerine göre üstün olduğunu söylemek zordur. Uygulamaya göre sistem gereksinimleri göz önüne alınarak kullanılacak şifreleme yöntemi seçilmelidir.

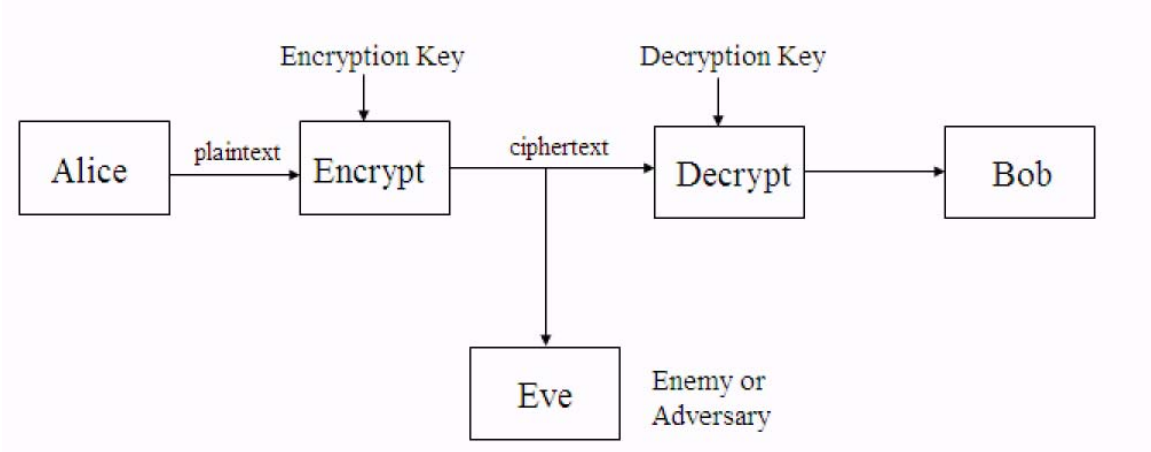
Öte yandan, şifreleme ve deşifreleme dönüşüm fonksiyonlarının kullandıkları anahtarlar birbirinden ayrılarak anahtar güvenliği sorunu kesin biçimde çözülebilir. Anılan çözüm, anahtarların farklılığı nedeniyle Asimetrik Kriptosistem olarak bilinen ve ilk kez 1976'da Diffie ve Hellman (EL-GAMAL şifreleme algoritması) tarafından belirlenen yeni bir dönüşüm tekniğiyle elde edilmektedir. 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 şifreleme algoritmaları aynı zamanda Halk Anahtarlı Kriptosistemler (Public Key Cryptosystem-PKS) olarak da bilinir.[3-5, 12]

Bu bölümde açık anahtarlı şifreleme sistemleri hakkında genel bilgiler verildikten sonra bu tip şifreleme sistemlerinin en bilineni olan RSA şifreleme tekniğinin teorisi anlaşılırlık açısından bir örnek üzerinden anlatılacaktır.

2.1. Açık Anahtarlı Şifreleme Sistemleri

Temel bir haberleşme senaryosu Şekil 2.1' de verilmektedir. Alice ve Bob birbirleri ile güvensiz bir kanal üzerinden güvenli bir haberleşmeye gerçekleştirmeye çalışmaktadır. Alice, Bob' a göndereceği orijinal veriyi (plaintext) bir şifreleme anahtarı (encryption key) kullanarak şifreler ve şifreli veriyi (ciphertext) güvensiz kanal üzerinden Bob' a yollar.



Şekil 2.1: Temel haberleşme senaryosu

Şifreli veriyi alan Bob, bir şifre çözme anahtarı (decryption key) kullanarak orijinal veriyi elde eder. Eve ise haberleşme kanalını dinlemektedir. Eve'nin amaçları şöyle sıralanabilir;

- Mesajı okumak
- Alice'in anahtarını elde etmek ve o anahtarla şifrelenmiş bütün mesajları okumak

- Mesajın içeriğini bir şekilde değiştirerek Bob'un değiştirilmiş mesajı Alice'den geliyormuş sanmasını sağlamak
- Alice'i taklit etmek ve Bob'la Alice'miş gibi iletişim kurmak.

Bu şifreli haberleşme sisteminde şifreleme anahtarı ile şifre çözme anahtarı aynı ise bu şifreleme sistemi simetrik anahtarlı şifreleme, bu anahtarlar farklı ise asimetrik anahtarlı şifreleme veya açık anahtarlı şifreleme sistemi olarak adlandırılır.

Açık anahtarlı bir şifreleme tekniği ile haberleşen kişi veya kurumların şifreleme anahtarları herkes tarafından bilindiğinden veya bilinebileceğinden bu kişi veya kurumlara isteyen herkes şifreli bir mesaj gönderebilir. Örneğin bir kişi veya kuruma zarar vermek isteyen biri yanlış veya yanıltıcı bir bilgiyi şifreli biçimde geçerek çeşitli sorunlara yol açabilir.

Açık anahtarlı şifrelemede bunun bir çözümü bulunmaktadır. Bu çözüm bir çeşit sayısal imza olarak değerlendirilebilir. Şifreli mesaj göndermek isteyen kişi öncelikle kısa bir metin seçer ve bu metne şifreli bir metin gibi davranarak kendi şifre çözme anahtarı ile bu metnin şifresini çözer. Sonra bu kısa mesajı göndermek istediği orijinal metne ekleyerek bütün metni göndermek istediği kişinin şifreleme anahtarı ile şifreler.

Şifreli metni şifre çözme anahtarı ile çözen kişi örneğin aşağıdaki gibi bir metin ile karşılaşır : "Şu marka ürünün satış fiyatını şu günden itibaren %20 artırıyoruz. Bütün birimlerinizi bu konuda bilgilendirmeniz gerekmektedir. ABCDEF'yi YHFVAS olarak şifreledim".

Bu metni alan kişi mesajı okuduktan sonra mesajın gerçekten tanıdığı bir kişiden geldiğini anlamak için YHFVAS metninin mesajı gönderenin açık anahtarı ile şifreler (aslında şifresini çözmüş oluyor). Eğer bu şifreleme işlemi sonrası ABCDEF'yi elde ediyorsa mesajı gönderen kişi gerçekten tanıdığı kişidir. Bu sertifikasyon metnini sürekli olarak değiştirmek doğru olacaktır.

2.2. RSA Şifreleme Sistemi

RSA şifreleme sistemi [11], açık anahtarlı şifreleme sistemlerinin en bilinenlerindendir. Ron Rivest, Adi Shamir ve Len Adleman tarafından 1977 yılında geliştirilmiştir ve geliştiricilerinin soyadlarının baş harfleri olan RSA olarak anılmaktadır. Bu yöntem basit bir matematiksel gerçeğe dayanarak geliştirilmiştir.

Sayıları çarpmak matematiksel olarak kolay bir işlemdir. Özellikle gelişen bilgisayar teknoloji sayesinde herhangi iki sayıyı çarpmak oldukça kolaydır. Ancak herhangi bir sayıyı çarpanlara ayırmak zor olabilir. Örneğin bilgisayarlar 1459160519 sayısının çarpanlarını, olası bütün kombinasyonları deneyerek bulabilir. Denenmesi gereken kombinasyon sayısı, ilgili sayının karekökü kadardır. Yani √1459160519 = 38000 olası kombinasyon denemelidir. Bu bilgisayar için çok da zor olmayan bir görevdir.

RSA şifreleme algoritmasının çalışması şekil 2.2'de 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.

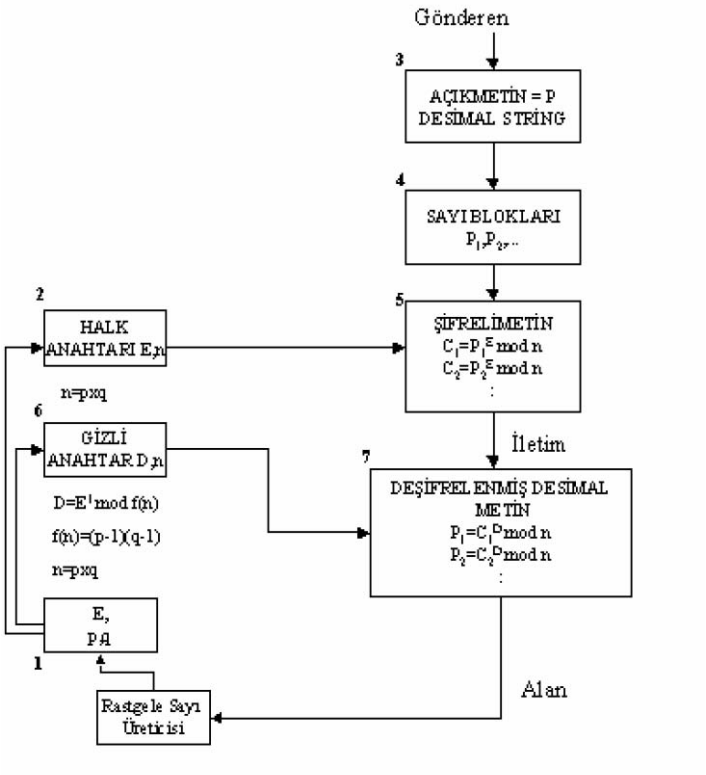
RSA şifreleme algoritmasında şifrelenecek olan açık metni öncelikle [0, n-1] arasındaki pozitif tamsayı bloklar haline dönüştürülür. Şekil 2.2'de ayrıntılı olarak matematiksel işlemler gösterilmektedir.

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 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 $m(i)=C(i)^D \bmod N$ olur. [12]



Şekil 2.2: RSA şifreleme algoritmasının yapısı

Örnek: RSA şifreleme sistemi kullanılarak STOP sözcüğünün şifrelenmesi:

p=43, q=59 ve E=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:

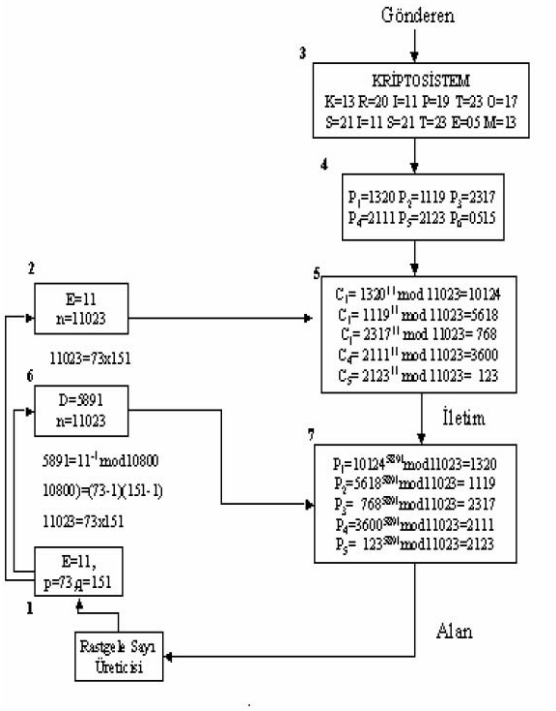
C1=181913 mod 2537 =2081 C2=141513 mod 2537 =2182

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



Şekil 2.3: RSA şifreleme algoritması için sayısal bir örnek

2.3. RSA'nın Güvenliği Hakkında

Şifreleme için seçilen p ve q asal sayıların çarpımlarından oluşan N sayısının boyutu, RSA algoritmasında anahtar boyu (key size) olarak anılır. Anahtarın boyutu büyüdükçe ilgili anahtarla şifrelenmiş metnin şifre çözme anahtarına sahip olmayan kişiler tarafından çözülmesi de zorlaşır. Ağustos 1977'de Martin Gardner Scientific American dergisinde 129 haneli bir sayı (426-bit) kullanarak RSA ile şifrelediği aşağıdaki mesajı yayınladı.

N = 114, 381, 625, 757, 383, 867, 669, 235, 779, 976, 146, 612, 010, 218, 296, 721, 242, 362, 562, 561, 842, 935, 706, 935, 245, 733, 897, 830, 597, 123, 563, 958, 705, 058, 989, 075,147, 599,290, 026, 879, 543, 541

Bu Mesaj Nisan 1996'da 600 gönüllüden oluşan bir ekibin çalışması sonucu sekiz ayda çözüldü. Şifresi çözülen metin aşağıdaki gibidir.

Plaintext = "the magic words are squeamish ossifrage" .

RSA Çarpanlara Ayırma Problemi RSA Security Şirketi tarafından Mart 1991'de başlatılmıştır. En etkileyici sonuç RSA-155 (155 haneli key) ile alınmıştır. RSA-155 çağrısı duyurulduktan yedi ay sonra Ağustos 1999'da bir grup araştırmacı tarafından 300 iş istasyonu ve PC'ler kullanılarak bu görevi tamamladı. 512-bit olan bu şifrenin çözülmesi önemliydi çünkü o yıllarda internet üzerinden yapılan e-ticaret uygulamalarında 512-bit'lik şifreleme kullanılıyordu. Çarpanlarına ayrılan sayı ve çarpanlar aşağıda verilmiştir.

109417386415705274215097073226403576120037329454492059909138421314763499842889347847179972578912673324976257528997818337970765372440271467435315933543338897 =

10263959282974110577205419657399167590071656780803806680334193352179071130777 9 *

10660348838016845482092722036001287867920795857598929152227060823719306280864 3.

512-bit şifre ancak oldukça yoğun çalışmalar sonucu kırılabilmesine karşın RSA Security şirketi 768-bit şifrelemeyi önerdi. Tablo 2.1'de RSA ile farklı anahtar uzunluklarında şifrelenmiş metinlerin hangi yıllarda çözülebildiğini verilmiştir.

Tablo 2.1: RSA çarpanlara ayırma problemi

Year	Number of decimal digits	Number of bits	MIPS years
1984	71	236	0,1
1988	106	352	140
1993	120	399	825
1994	129	429	5000
1995	119	395	250
1996	130	432	750
1999	140	466	2000
1999	155	512	8000

RSA-155 saniyede l milyon komut işleyebilen bir bilgisayar ile bütün olasılıklar denenerek 30000 yılda çözebilecekken iken 3.5 ayda yüzlerce bilgisayar kullanılarak çözülmüştür (Toplam işlem zamanı 8000 MIPS- MIPS : Millions of Instructions Per Second).

1995 yılında 512-bit şifrenin l milyon $'dan az bir yatırımla sekiz ayda kırılabileceği ileri sürülürken RSA-155 çerçevesinde bu şifre 1999 yılında yedi ayda kırılabilmiştir. Buradan 512-bit'lik şifrelerin kısa süreli olarak güvenli olduğu söylenebilir. Anahtar uzunluğunun seçimi tamamen uygulamaya bağlıdır. RSA Security şireketi 2000 yılında 1024-bit şifreleme önermekle birlikte daha yüksek gevenlik gereksinimi olan uygulamalar için 2048-bit şifreleme önermektedir.

Tablo 2.2'de çeşitli anahtar uzunlukları için RSA ile şifrelenmiş bir metnin bütün olasılıklar denenerek çözülmesi için gereken işlem yükü verilmektedir.

Tablo 2.2: RSA anahtar uzunluğunun çözülmesi için gereken işlem miktarı

Size of integer to be foctored (in bits)	MIPS years
512	$3x10^4$
768	$2x10^8$
1024	$3x10^{11}$
1280	$1x10^{14}$
1536	$3x10^{16}$
2048	$3x10^{20}$

Bu tablodan görüldüğü üzere anahtar uzunluğu arttıkça şifreli metnin elde edilmesi için gereken işlem sayısı da artmaktadır. Tablo 2.3'de çeşitli asimetrik ve simetrik şifreleme tekniklerinin anahtar uzunluğuna göre dayanıklılığı karşılaştırılmalı olarak verilmektedir.

Tablo 2.3: Farklı tekniklerin anahtar uzunluğuna göre dayanıklılığının karşılaştırılması

Symmetric	56	80	112	128	192	256
RSA n	512	1024	2048	3072	7680	15360
DSA p	512	1024	2048	3072	7680	15360
DSA q	112	160	224	256	384	512
ECC n	112	161	224	256	384	512

Bu tablodan görüldüğü üzere simetrik bir şifreleme tekniği (örneğin DES) ile 56-bitlik bir anahtar kullanılarak şifrelenmiş metnin kırılması ile RSA tekniği ile 512-bitlik bir anahtar kullanılarak şifrelenmiş metnin kırılması aynı işlem gücünü gerektirmektedir. Yani simetrik şifreleme tekniklerinde daha kısa anahtar uzunlukları ile asimetrik şifrelemeyle aynı güvenlik sağlanabileceği söylenebilir.

Ayrıca bu tablodan asimetrik anahtarlı başka bir şifreleme yöntemi olan ECC'nin (Eliptic Curve Crpytosystem) daha kısa anahtar uzunluğu ile RSA ile aynı güvenliği saylayabildiği açıkça görülmektedir.

2001 yılında Illinois Üniversitesinden Daniel J. Bernstein çarpanlara ayırma sorununda kullanılan "Number Field Sieve" atağını daha da geliştirmiştir. Bu noktada 1024-bit RSA ile şifrelemenin güvensiz olduğu hatta l milyon $ yatırımla dakikalar içinde kırabileceği iddiaları ileri sürülmüştür. RSA Security şireketi bunun Bernstein'ın çalışmasının yanlış yorumlanması sonucu olduğunu öne sürmüştür.

RSA'in tasarımcılarından Adi Shamir'in Haziran 2003 yayınladığı "On the Cost of Factoring RSA-1024" başlıklı yazısında 1024-bit RSA'in 15-20 yıl daha güvenli olduğunu ancak bazı varsayımlar altında geliştirilecek özel bir donanım ile 10 milyon $'lık bir yatırımla l yılda kırabileceğini fikrini ileri sürmüştür.

RSA şifreleme sistemi geliştirilen ilk asimetrik anahtarlı şifreleme sistemlerinden biri olmasına karşın hala güvenirliliğini korumaktadır. Ancak simetrik şifreleme sistemlerine göre işlem yükü oldukça fazladır ve simetrik anahtarlı şifreleme sistemleriyle aynı güvenliği sağlamak için daha uzun anahtarlara gereksinim duyar.

2.4. Diffie-Hellman

Anahtar değiş-tokuşu için yaygın olarak kullanılan bir protokoldür. Pek çok kriptografik protokolde iki taraf aralarında bir iletişim başlatmak isterler. Başlangıçta aralarında herhangi bir ortak gizliliğe sahip olmayan taraflar gizli anahtar kriptosistemlerini kullanabilirler. Bu durum için, Diffie-Hellman protokolü tarafından sağlanan anahtar değiş-tokuşu, güvenli olmayan kanallar üzerinden ortak bir gizli anahtar iletiminin sağlanmasına bir çare bulmuştur. Diffie-Hellman problemi olarak adlandırılan bu yöntem, kesikli logaritmalarla ilgili bir problem üzerine kurulmuştur. Bu problemin çok zor olduğu ve bazı durumlarda kesikli logaritma problemi kadar zor olduğu düşünülmektedir.

Diffie-Hellman protokolünün, uygun bir matematiksel grup kullanıldığında genelde güvenli olduğu düşünülmektedir. Özel olarak, üslü ifadelerde kullanılan üretici eleman geniş bir peryoda (sıraya) sahip olmalıdır. Kesikli logaritma algoritmaları Diffie-Hellman'a saldırmak için kullanılabilir ve -parametrelerin doğru olarak seçildiğini kabul edersek- şu anda yapılabileceklerin en iyisi pasif saldırılardır. Eğer alışıldık bir aritmetik modülo asıl sayı kullanılarak Diffie-Hellman uygulanılırsa, yeterince geniş bir asal seçmek ve üretici elemanın seçiminde özen göstermek yeterli olacaktır. Güç algılanan problemler, üretecin kötü seçimlerinden kaynaklanıyor olabilir. [1]

2.5. Eliptik Eğri Kriptosistemleri (Elliptic Curve Cryptography)

Kriptografideki eliptik eğriler temel olarak, p karakteristiğinin (p>3 olmalıdır) sonlu alanında düşünüldüğünde, $y^2 = x^3 + ax + b$ denklemini sağlayan noktaların bir kümesidir. p = 2 ve p = 3 karakteristikleri için biraz farklı bir denklem gerekmektedir.

Eliptik eğriler üzerindeki noktalar bir araya toplanabilir ve bunlar grup adı verilen bir yapı oluştururlar (aslında bir Abel Grubu). Bu, şunu söylemenin farklı bir yoludur; aynı tamsayılarla yaptığımız gibi bunlar ile de sadece toplama ve çıkarma kullanarak aritmetik yapabiliriz.

Bunların bazı teorik faydaları vardır ve ayrıca çok pratiktirler. Hiperbolik eğriler, bir sonlu alan veya diğer bazı pek çok gruptaki kesikli logaritmalarda olduğunun aksine, eliptik eğrilerin kesikli logaritma problemlerini hesaplamak için bilinen bir subexponential algoritma yoktur. Eliptik eğriler için hızlı kesikli logaritma hesaplamasının olmamasının bir faydası anahtar genişliğinin, üretilen dijital imzaların ve şifrelenen mesajların küçük olmasıdır. Gerçekte, anahtar genişliği için güvenlik düzeyinin hesaplanmasının kolay bir yolu, bir anahtar genişliğini, bir gizli anahtar kriptosistemine bitler halinde almak ve sonra bunu 2 ile çarpmaktır.

Eliptik eğriler donanım ve yazılım ile çok etkin bir biçimde uygulanabilirler ve hız bazında RSA ve DSS gibi kriptosistemler ile yarışabilirler. Eliptik eğri

kriptosistemlerini standartlaştırmak için pek çok girişimde bulunulmaktadır (örneğin, ANSI tarafından ECDSA). Şu anda eliptik eğriler yaygın olarak bilinmektedirler, ama pratikte pek kullanılmamaktadırlar. Özel örneklere karşı saldırılarda gelişmeler kaydedilmiş olmasına rağmen, eliptik eğri kriptosistemlerinin güvenliği yıllardan beri oldukça sağlamdır.

Lenstra ve Verheul tarafından geliştirilen XTR algoritması eliptik eğrilere karşı güçlü bir rakip haline gelebilir. Ancak, eliptik eğriler performans açısından bir miktar iyi görünmekte ve anahtar genişliğinde kesinlikle daha iyidirler. [1]

BÖLÜM 3: SİMETRİK ANAHTARLI ŞİFRELEME SİSTEMLERİ

İçinde bulunduğumuz çağda teknolojinin akıl almaz bir hızda gelişiyor olması bir anlamda bizim açımızdan büyük bir tehlike arz etmekte. Çünkü teknolojinin gelişme hızına bağlı olarak verilerimizin güvenliği de aynı hızda tehlike altına girmiş oluyor. Bir zamanlar asla kırılamaz diye nitelendirilen DES algoritmasının, 250.000 dolara yakın bir maliyetle 3–4 gün içerisinde kırılması teknolojinin ne kadar büyük bir güç olduğunu göstermiştir. Ama teknoloji geliştikçe bilgi güvenliği sağlama konusunda çalışmalar da aynı paralellikte sürecektir. Bununla birlikte DES kırılmıştır ama yerine daha güçlü bir algoritma gelmiştir; Triple DES. Bu çalışmada DES algoritmasının yapısı açıklanarak, bu algoritmanın gerçeklendiği C programı da pratik bir uygulama olarak gerçeklenmiştir.

Bu bölümde blok şifreleme ve özellikleri, AES algoritması [7], DES sistemi [6] daha sonra da Triple DES sisteminin DES algoritmasından farkları açıklanacaktır.

Kriptografide blok şifreleme ve akış (stream) şifreleme olmak üzere iki temel simetrik algoritma tipi vardır. Bunlardan blok şifreleme, orijinal metni veya şifreli metni bloklara bölerek şifreleme/deşifreleme işlemini yapar. Akış şifrelemede ise bir bit veya byte üzerinde şifreleme ve deşifreleme işlemleri yapılır.

Blok şifreleme algoritmalarına, SPN (Substitution-Permutation Network), DES (Data Encryption Standard), AES (Advanced Encryption Standard), FEAL örnek verilebilir. Blok şifreleme algoritmalarının gücü söz konusu olduğunda algoritmada kullanılan S kutuları, döngü sayısı, anahtarların XOR işlemine sokulması, blok uzunluğu, anahtarın uzunluğu ve özelliği büyük önem taşımaktadır. Shannon şifreleme algoritmasının gücü için "blok uzunluğunun en azından anahtar uzunluğuna eşit olması gerekir" demiştir. Ayrıca kullanılacak anahtarın rastlantısal olması da gerekir. Diğer yandan algoritmaya yapılan saldırılara karşı dayanıklılıkta günümüz algoritmalarının gücünün ölçülmesinde bir kıstas olmuştur. Bu saldırılara örnek olarak lineer kriptanaliz, diferansiyel kriptanaliz, ilişkili anahtar saldırısı

verilebilir. Ayrıca bu saldırılar bazı koşullar altında geliştirilerek daha farklı saldırı türleri de ortaya çıkmıştır. Bunlara örnek olarak imkansız diferansiyel kriptanaliz verilebilir. Genel olarak bakıldığında bir saldırı için üç parametre önemlidir. Bunlar veri, veri alanı ve zamandır. Bu parametreler blok şifreler için üç temel saldırıyı ortaya çıkarır. Bunlar sözlük saldırısı (dictionary attack) kodkitabı saldırısı (codebook attack) ve tüm anahtarların denenerek doğru anahtarın arandığı geniş anahtar arama saldırısı (exhaustive key search) olarak özetlenebilir.

Burada, günümüzde modern şifreleme algoritmalarındaki güç incelenmiştir. Bu algoritmalara kuvvetli denebilmesi için gerekli olan kıstasların neler olabileceği ve günümüzde önemli bir şifreleme algoritması olan AES algoritmasının tasarlanırken ne gibi aşamalardan geçtiği de araştırılmıştır. [13]

3.1. Blok Şifreleme

Blok şifreler, Shannon'un önerdiği karıştırma (confusion) ve yayılma (diffusion) tekniklerine dayanır. Karıştırma şifreli metin ve açık metin arasındaki ilişkiyi gizlemeyi amaçlarken, yayılma açık metindeki izlerin şifreli metinde sezilmemesini sağlamak için kullanılır. Karıştırma ve yayılma, sırasıyla yer değiştirme ve lineer transformasyon işlemleri ile gerçeklenir. Feistel ağları ve Yer değiştirme-Permütasyon ağları olmak üzere iki ana blok şifreleme mimarisi vardır. Her ikisi de yer değiştirme ve lineer transformasyonu kullanır. Ayrıca her iki mimari ürün şifrelerinin örneklerindendir. Yani birden fazla şifreleme işleminin birleşmesi ile oluşturulurlar. Tekrarlanan şifreler yine ürün şifreleridir ve aynı şifreleme adımının tekrarlanan uygulamasını içerir ve her şifreleme adımına döngü denir. Bir döngü birden fazla şifreleme adımı içerebilir. Genellikle her döngüde farklı anahtar materyali kullanılır. [13]

3.2. Blok Şifreleme Algoritmalarının Önemli Özellikleri

3.2.1. Anahtar

Blok şifreleme algoritmalarında anahtarın uzunluğu ya da bit sayısı, en temel saldırı olan geniş anahtar arama saldırısına karşın güçlü olmalıdır. Örneğin DES algoritması 56-bit anahtar kullanırken AES, algoritması DES'in bu zaafını örter niteliktedir ve 128, 192, 256 bit anahtar seçenekleri mevcuttur. Ayrıca anahtarın rastlantısal olması gerekmektedir.

3.2.2. Döngü sayısı

Blok şifreleme algoritmalarında döngü sayısı iyi seçilmek zorundadır. Çünkü lineer transformasyon ve yer değiştirmelerin bu seçilen değerle algoritmaya yeterli gücü vermesi gerekmektedir. Ayrıca yapılan saldırıların başarısız olması için en önemli şartlardan biridir. Bu sayı için herhangi bir teorik hesaplama olmamasına rağmen Lars Knudsen'e göre kabaca döngü sayısı r >= dn/w formülü ile hesaplanır. Burada r döngü sayısını, d yer değiştirme durumuna bir word'ü almak için gerekli maksimum döngü sayısını, n blok genişliğini, w ise tüm şifrede yer değiştirme durumuna giriş olan minimum word genişliğini temsil etmektedir. Yukarıdaki formülde yayılma tekniği ihmal edilmiştir. Tablo 3.1 Lars Knudsen'e göre bazı algoritmaların döngü sayılarının neler olması gerektiğini göstermektedir.

Tablo 3.1: Bazı şifreleme algoritmaları için döngü sayıları

Algoritma	Döngü sayısı	Olması gereken döngü sayısı
DES	16	21
IDEA	8	8
Blowfish	16	16
AES(Rijndael)	10	16

3.2.3. S kutuları

S kutuları bir blok şifreleme algoritmasının en önemli ana elemanıdır. Çünkü algoritmadaki tek non-lineer yapıdır ve dolayısıyla algoritmaya gücünü vermektedir. S kutuları için üç önemli nokta vardır. Bunların belirlenmesinde lineer kriptanaliz, diferansiyel kriptanaliz, Davies saldırıları etkili olmuştur. Bunlar; SAC (Strict Avalance Criteria); 1 bit giriş değişimi sonucunda her çıkış bitinin değişme olasılığı ½ olur. S kutularının genişliği; Kriptanaliz saldırıları düşünüldüğünde büyük bir kutu küçüğüne oranla daha iyi olacaktır. Ayrıca diferansiyel saldırılardan korunmak için büyük sayıda çıkış bitleri ve lineer saldırılardan korunmak için büyük sayıda giriş bitleri gereklidir. S kutusu gereksinimleri; Çıkışların dağılımları Davies saldırısına karşın kontrol edilmeli, çıkışlar girişe göre lineer olmamalı, S kutusunun her sırasındaki değerler tek olmalıdır. Daha güçlü S kutuları yaratmak için çeşitli çalışmalar da yapılmıştır.

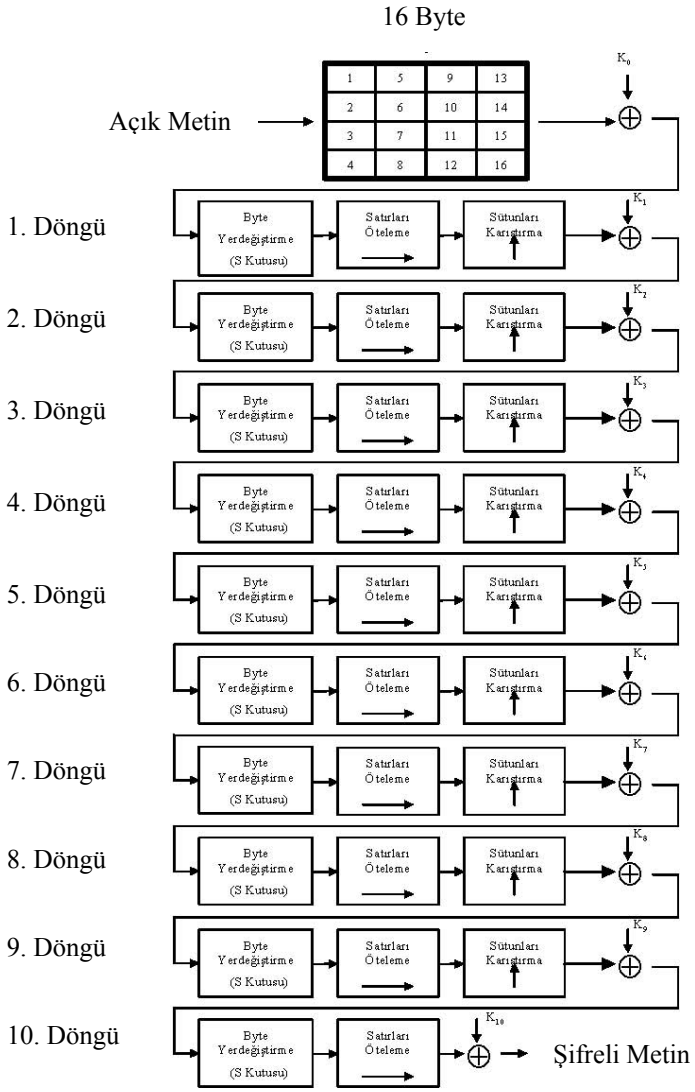
3.3. AES Algoritması

AES [7] (Rijndael) algoritması şu ana kadar bilinen algoritmalar içerisinde en güçlülerinden biridir. Bu gücü nerden aldığını incelemekte fayda vardır. DES (Data Encryption Standard) 1970'li yıllarda IBM ve NSA ile birlikte öne sürüldüğünde 1990'lı yıllara kadar kendini başarı ile savundu. Ancak onun en büyük zaafı 56 bit anahtara sahip olması ayrıca anahtarın bir şekilde daha büyük uzunlukta kullanılabilecek bir yönteminin olmamasıydı. Daha sonraları paralel işlemcili bilgisayarların kullanılması ve geniş anahtar arama saldırısı ile kısa bir zaman içinde kırılması mümkün hale gelmişti. Ayrıca güçlü kriptanaliz saldırılarına karşı da yetersiz kalmaya başlamıştı. DES algoritması Feistel mimarisine sahipti. Yani veri bloğu iki parçaya bölünerek şifreleme işlemi yapılıyordu. Heys, 1994 yılındaki çalışmasında, bir SPN algoritması kullanarak diferansiyel ve lineer kriptanalize karşı güçlü DES algoritmasına eşit güçte bir algoritmayı 8 döngüde sağladı. Bu algoritma 64 bit blok uzunluğunda, 64 bit anahtar ve 8 bit girişli - 8 bit çıkışlı random S kutuları kullanmaktaydı. Bunun yanında S kutularının güçlü hale getirilmesi için bazı çalışmalar da gerçekleştirildi. Random S kutuları, S kutularının sırasını değiştirme, anahtar bağımlı S kutularının sırasını değiştirme, anahtar bağımlı S kutularının

transformasyonu ve matematiksel bağlamda saldırılara karşı güçlü S kutuları tasarlanmaya çalışıldı. Nyberg S kutularının tasarlanmasında alan terslerinin (field inverse) kullanılmasını önerdi. Bu çalışmaların yardımıyla da tasarlanan AES algoritmasında feistel mimarisinden yerdeğiştirme-permütasyon mimarisine geçilmiş oldu.

AES (Rijndael) algoritması 128 bit veri bloklarını 128, 192, 256 bit anahtar seçenekleri ile şifreleyen bir algoritmadır. SPN algoritmasının geniş bir çeşididir. Square ve Crypton şifreleri AES benzeri SPN şifrelerdir. Döngü sayısı anahtar genişliğine göre değişmektedir. 128 bit anahtar için 10 döngüde şifreleme yapılırken 192 ve 256 bit anahtarlar için sırasıyla 12 ve 14 döngüde şifreleme yapılmaktadır. AES algoritmasında her döngü dört katmandan oluşur. İlk olarak 128 bit veri 4×4 byte matrisine dönüştürülür. Daha sonra her döngüde sırasıyla byte'ların yer değiştirmesi, satırları öteleme, sütunları karıştırma ve anahtar planlamadan gelen o döngü için belirlenen anahtar ile XOR'lama işlemleri yapılır. Byte'ların yer değiştirilmesinde 16 byte değerinin her biri 8 bit girişli ve 8 bit çıkışlı S kutusuna sokulur. S kutusu değerleri, Galois alanı'nda (Galois Field - GF) $GF(2^8)$, 8 bitlik polinom için ters alındıktan sonra lineer bir transformasyona sokularak elde edilmiştir. Satırların ötelenmesi işleminde 4×4 byte matrisinde satırlar ötelenmiş ve sütunların karıştırılması işleminde herhangi bir sütun için o sütundaki değerler karıştırılmıştır. Sütun karıştırma işleminde Galois alanında iki sayının çarpım kavramı kullanılmıştır. Döngünün son katmanında ise o döngüye ait anahtar ile XOR'lama yapılmıştır.

Algoritmadaki sütunların karıştırılması bir SPN algoritmasına bakıldığında ek bir lineer transformasyon işlemidir. Şekil 3.1, 10 döngülük AES algoritmasını göstermektedir. AES Algoritmasında S kutularının tasarımında sonlu alanda ters alma işlemi, lineer kriptanaliz için kullanılan lineer yaklaşım tablolarına ve diferansiyel kriptanaliz için kullanılan fark (difference) dağılım tablolarına girişlerin olabildiğince uniforma yakın olmasını sağlarken (diferansiyel ve lineer kriptanalize karşı etkin olması demek), lineer transformasyon işlemi saldırılarda az sayıda aktif S kutusu (lineer ve diferansiyel kriptanalizde az sayıda olması daha az açık metin/şifreli metin kullanılması demek) kullanmayı imkansız hale getirir.



Şekil 3.1: AES algoritması (128 bit anahtar için)

AES algoritmasına, imkansız diferansiyel saldırısı gibi çeşitli saldırlar yapılmıştır. Ancak bu saldırılar azaltılmış döngü sayısına sahip AES algoritmalarına karşı gerçekleştirilmiştir. [13]

3.4. DES Algoritması ve İşleyiş Mantığı

DES (Data Encryption Standard) [6], 1970'lerin ortalarında geliştirilmiş bir simetrik algoritma türüdür (şifreleme ve çözme işlemi aynı anahtarla yapılır). ABD Ulusal Teknoloji ve Standartları Enstitüsü (NIST) tarafından standart haline getirilmiştir.

DES 64 bitlik mesaj gruplarıyla çalışır. Yani mesaj 64 bitten az ise onu eklediği 0 larla 64 bite tamamlar. Eğer mesaj 64 bitten fazla ise mesaj girdisini 64 bitlik bloklara ayırır ve her birine şifreleme işlemi uygular. 64 bit 16'lık sayı tabanında 16 sayıya denk gelmektedir. Örnek olarak:

$(0110\ 1110\ 0010\ 1100\ 0000\ 1101\ 1111\ 0011\ 0001\ 0110\ 1001\ 1111\ 0010\ 0000\ 0010\ 0111)_2$

$(\ 6\quad E\quad 2\quad C\quad 0\quad D\quad F\quad 3\quad 1\quad 6\quad 9\quad F\quad 2\quad 0\quad 2\quad 7\ )_{16}$

sayısına denk gelmektedir.

DES şifreleme yapabilmek için kullanıcı tarafından girilen 64 bitlik "private key" yani özel bir anahtar kullanır. Fakat bu anahtarda her 8. bit göz ardı edilir. Dolayısıyla etkin anahtar uzunluğu 56 bittir ancak her durumda 64 bitlik gruplar esastır ve DES'in temelini oluşturur.

3.4.1. DES algoritmasına detaylı bir bakış

Şifreleme yapacağımız mesajımız "burdurlu" olarak seçilsin (64 bitlik bir mesaj olması dolayısıyla kolay incelenebilir olduğundan mesaj olarak bu kelime seçilmiştir).

Şifrelemede kullanacağımız anahtarımız hexadecimal olarak 6F67757A68616E75 olsun. Buna şifre kelimesinin baş harfini verelim:

Ş= 6F67757A68616E75 .

Şimdi bu Ş'nin ikili tabandaki karşılığını yazalım:

Ş=0110 1111 0110 0111 0111 0101 0111 1010 0111 1000 0110 0001 0110 1110 0111 0101

Soldan sağa okuyoruz. Böylece en soldaki bit 1.bit (0), en sağdaki bit 64. bit (1) oluyor. DES işlemi 64 bitlik bloklar içinde 56 bitlik anahtarlar ile gerçekleşir (yukarıda etkin anahtar uzunluğunun 56 bit olduğundan söz edilmişti).

Ş anahtarımızı ikili olarak yazdıktan sonra bir K (“key”in baş harfi olarak düşünülebilir) anahtarı oluşturacağız. Bu anahtarı Ş anahtarımızın her 8. bitini 1 yaparak elde edelim:

K=0110 1111 0110 0111 0111 0101 0111 1011 0111 1001 0110 0001 0110 1111 0111 0101

K anahtarı oluşturulduktan sonra izlenecek ilk adım 48 bit uzunluğunda 16 tane anahtar oluşturmak olacaktır.

3.4.2. Adım-1: 48 bitlik 16 anahtar oluşturma

64 bitlik K anahtarımızı Tablo 3.2 deki sıralamaya göre permüte ediyoruz (bu tabloya PC-1 adı verilsin).Yani yeni bir sıralama oluşturmuş oluyoruz. Tabloya bakılacak olursa, yapılacak işlemde ilk bit yerine 57. bit geçecek, ikinci bit yerine 49. bit, üçüncü biti 41. bit oluşturacak v.s.Bu işlem sonucunda oluşan yeni anahtara K+ adı verilecek olursa:

K+ = 0000 0000 1111 1111 1111 1111 1001 0100 1011 1100 0111 0101 1001 1100

Görüldüğü gibi yeni anahtarımız 56 bitliktir. Çünkü tabloya dikkat edilecek olursa 8. bitler yer almamaktadır.

Tablo 3.2: PC-1, Permütasyon seçim tablosu

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 aşamadan sonraki işlem permüte edilmiş anahtarı sağ ve sol olmak üzere 28 bitlik iki parçaya bölmek. Böleceğimiz bloklardan sağ tarafta olanına C_0, sol tarafta olanına D_0 adını vereceğiz. Buna göre:

C_0 = 0000 0000 1111 1111 1111 1111 1001

D_0 = 0100 1011 1100 0111 0101 1001 1100 olur.

Bu tanımladığımız C_0, D_0 ile C_n, D_n şeklinde 16 blok oluşturabiliriz.(1<=n<=16). Her C_n, D_n çiftini sola doğru kaydırma işlemi yapılmış bir önceki çiftten meydana getireceğiz. Sola doğru kaydırma yapmak için birinci bit hariç her biti bir veya iki solundaki basamağa, birinci biti de en sona geçireceğiz. Buna ait kaydırma tablosu aşağıda verilmiştir:

Tablo 3.3: Kaydırma tablosu

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

Tablo 3.3'den anlamamız gereken C_3 ve D_3, C_2 ve D_2 iki kez sola kaydırılarak elde edilmiştir ve her n için kaydırma değeri değişmektedir.

NOT: Yazının bu aşamasından sonraki ikili sayılar gerçek değerleri ile verilmeyecektir. Burada amaç sadece DES şifreleme algoritmasının nasıl işlediğini gösterebilmektir.

Kaydırma tablosundan C_n, D_n değerlerini elde edelim:

C_0 = 0000 0000 1111 1111 1111 1111 1001

D_0 = 0100 1011 1100 0111 0101 1001 1100

C_1 = 0000 0001 1111 1111 1111 1111 0000

D_1 = 1101 0111 1000 1110 1011 1011 1000

C_2 = 0000 0101 1111 1111 1111 1110 1100

D_2 = 0100 1011 1100 0111 0101 1001 1100

C_3 = 0000 0011 1111 1111 1111 1100 1001

D_3 = 0101 0011 1000 0111 0101 1001 1100

C_4 = 0000 0000 1001 1011 1111 1001 0000

D_4 = 0100 1011 1100 0101 0101 1001 1100

C_5 = 0000 0000 1001 1101 1111 1111 1001

D_5 = 0100 1011 1100 0111 0101 1001 1100

C_6 = 0010 1100 1101 1111 1111 1111 1001

D_6 = 0100 1011 1100 0111 0101 1001 1100

C_7 = 0010 0000 1111 1111 1111 1111 1001

D_7 = 0100 1011 1000 0111 0101 1111 1110

C_8 = 0010 1001 1111 1111 1110 0100 0000

D_8 = 0111 1011 0100 0111 0111 1001 0100

C_9 = 0000 0000 1001 1101 0011 1011 1001

D_9 = 0100 1011 1110 0111 0111 1101 1100

C_{10} = 0110 1000 1111 1001 0111 1111 1001

D_{10}= 0100 1011 1110 0111 0101 1001 1100

C_{11} = 0000 0000 1111 1001 1111 1111 1001

D_{11}= 0101 1011 1100 0111 0101 1101 1101

C_{12} = 0000 0000 1111 1111 1001 1111 1001
D_{12} = 0100 1011 1100 0111 0101 1001 1111

C_{13} = 0000 0100 0111 1111 1101 1111 1001
D_{13} = 0101 0011 1100 0101 0101 1001 1110

C_{14} = 1111 1111 1111 1000 0000 0000 1011
D_{14} = 0100 1011 1101 0111 0101 1101 1111

C_{15} = 0000 0000 0000 1111 1111 1111 1010
D_{15} = 0100 1111 1110 0111 0101 1001 1100

C_{16} = 110 0000 1110 1111 1011 1101 1001
D_{16} = 1110 1111 1100 0111 0101 1001 1100

Bu aşamadan sonra Adım-1 de amaçladığımız 48 bitlik anahtarları oluşturabiliriz. Buna göre C_n D_n çiftlerini birleştirip yeni bir tabloya göre permüte edeceğiz. Anahtarlarımızın 48 bit olabilmeleri için yeni tablomuz 48 bitlik olmalı. Bu 56 bitlik olan C_n D_n çiftlerini 8. bitlerini göz ardı ederek 48 bite indirecek bir tablo olacağı anlamına geliyor. Bu tabloya PC-2 adı verilsin. Tablo 3.4 aşağıda görülmektedir:

Tablo 3.4: PC-2, Permütasyon seçim tablosu

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

Tablolara göre permüte etme işlemi hep aynı mantıkla yapılmakta. Yani sayıların yerlerini, sıralarını değiştirme şeklinde.

Tablodaki yer değiştirmeler sonucunda birleştirmiş olduğumuz her bir C_n D_n çiftine K_n adı verilecek olursa elde edeceğimiz 48 bitlik anahtarlar aşağıdaki gibi olur:

K_1= 1110 0000 1011 1110 0110 0110 0000 0001 0110 0110 0001 1011
K_2= 0111 1001 1010 1110 1101 1001 1101 1011 1100 1001 1110 0101
K_3= 0101 0101 1111 1100 1000 1010 0100 0010 1100 1111 1001 1001
K_4= 0111 0010 1010 1101 1101 0110 1101 1011 0011 0101 0001 1101
K_5= 0111 1100 1110 1100 0000 0000 1111 1111 0101 0011 1010 1000
K_6= 0110 0011 1010 0101 0010 0111 1100 1000 0111 1011 0010 1111
K_7= 1110 1100 1000 0100 1001 0110 1111 1110 0001 1000 1011 1100
K_8= 1111 0111 1000 1010 0000 0111 0101 1001 0011 1011 1111 1011
K_9= 1110 0000 1101 1011 1111 1101 0111 1101 1110 0111 1000 0001
K_{10}=1011 0001 1111 0011 0110 1000 1111 0110 0100 0110 0100 1111
K_{11}=0010 0001 0101 1111 1101 0011 1101 1110 1101 0011 1000 0110
K_{12}=0111 0101 0111 0001 1111 0101 1001 0100 0110 0111 1110 1001
K_{13}=0101 0101 0110 1110 1110 0000 0000 1110 1111 0110 1100 1110
K_{14}=0001 0001 0110 0000 0000 0011 1101 1110 1001 1001 1100 1001
K_{15}=0111 1111 1001 0011 1001 0101 0101 0100 0101 1110 0000 0010
K_{16}=0111 0100 1010 0101 1101 0110 1101 1011 0000 1100 1101 0110

Şimdi ilk adımda hedeflediğimiz amacımıza ulaştığımıza göre ikinci adıma geçebiliriz. İkinci adımımız her 64 bitlik bilginin kodlanmasıdır.

3.4.3. Adım-2: her 64 bitlik bilginin kodlanması

DES işleminin ikinci temel elemanı olan ilk permütasyon sayısına IP diyelim (initial permutation). Bu sayı 64 bitlik orijinal mesaj blokunun IP tablosuna göre permüte edilmesiyle elde edilir. İlk (başlangıç) permütasyon tablosu aşağıda görülmektedir:

Tablo 3.5: Başlangıç permütasyon tablosu

IP							
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

Yukarıdaki tablo 3.5'e göre permüte edilen mesaj aşağıdaki gibi olacaktır:

IP=1100 1100 0000 0000 1100 1100 1111 1111 1111 0000 1010 1010 1111 0000 1010 1010

Şimdi elde ettiğimiz bu sayıyı 32 bitlik sağ ve sol kısma ayıralım ve bunlara L(eft) ve R(ight) harflerini atayalım. Buna göre:

L_0= 1100 1100 0000 0000 1100 1100 1111 1111
R_0= 1111 0000 1010 1010 1111 0000 1010 1010

Buraya kadar her şey normal. İşin bu kısmından sonra işlemler karışmaya başlıyor. Şöyle ki:

Burada 1 den 16 ya kadar olan L_n R_n çiftlerini hesaplayacağız. Bunu yaparken de karmaşık bir işlem yürütecek olan *f* fonksiyonunu kullanacağız. *f* fonksiyonunun yapacağı iş yüzeysel olarak anlatılacak olursa 32 bitlik mesaj bloğu ile 48 bitlik bir anahtarı (K_n) işleme sokmaktır. Bu fonksiyonun yapacağı işlemi açıklamadan önce kullanacağımız formülü yazmakta yarar var:

$$L_n = R_{n-1} \quad (3.1)$$

$$R_n = L_{n-1} + f(R_{n-1}, K_n) \quad (3.2)$$

Formüle bakıldığında L_1'in R_0'a eşit olduğu hemen farkedilebilir.

İşin bu kısmına gelmişken *f* fonksiyonunu inceleyelim. *f* fonksiyonunu hesaplamak için R_0 sayısını 32 bitten 48 bite çıkarmalıyız. Bunu yapmak için de E adı verilen bir tablodan yararlanmak durumundayız. Bu tablonun tek yaptığı şey bazı bitleri tekrardan kullanarak giren bit sayısını 32 den 48 bite çıkarmaktır. Bu işlem oldukça basittir.

Tablo 3.6: E Tablosu

E BIT-SELECTION TABLE					
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 R_0'dan $E(R_0)$'i hesaplayabiliriz.

R_0 = 1111 0000 1010 1010 1111 0000 1010 1010
$E(R_0)$= 0111 1010 0001 0101 0101 0101 0111 1010 0001 0101 0101 0101

Şimdi artık elimizde 48 bit var. *f* i hesaplamaya devam edelim. $E(R_{n-1})$ bloğunu K_n anahtarıyla toplama işlemine sokalım:

$E(R_0)$= 0111 1010 0001 0101 0101 0101 0111 1010 0001 0101 0101 0101
K_1 = 1110 0000 1011 1110 0110 0110 0000 0001 0110 0110 0001 1011
$E(R_0)+K_1$= 011000 010001 011110 111010 100001 100110 010100 100111

Henüz *f* fonksiyonunu hesaplamayı bitirmedik. Şu ana kadar yaptığımız E tablosuna göre R_0'ı genişlettik ve K_1 anahtarı ile topladık.

Elimizde 48 bit var. Yani 8 tane 6 bitlik grup (yukarıda 4 bitlik yerine 6 bitlik ayırma yapmamın nedeni budur). Bu 6'lı grupları S adı verilen tablolarda adres olarak kullanacağız. Her 6'lı grup bize farklı S tablolarındaki adresleri verecek. Bu adreslerde 4 bitlik bir sayı yer alacak ve bu 4 bitlik sayılar 6 bitlik sayıların yerine geçerek 8 tane 4 bit yani 32 bitlik bloğu oluşturacak. Elimizdeki en son sonucu alalım;

$E(R_0)+K_1$= 011000 010001 011110 111010 100001 100110 010100 100111

6'lı her gruba B harfi vererek bunu farklı biçimde yazalım:

$E(R_0)+K_1=B_1B_2B_3B_4B_5B_6B_7B_8$ ve bahsettiğimiz işleme sokalım, yani S tablolarına

Denklemleştirecek olursak yapacağımız işlem:

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

Tablo 3.7: S Tabloları

S_1

Satır No \ Sütun No	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	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
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_2

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

S_3

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

S_4

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

S_5

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

S_6

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

S_7

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

S_8

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

B harfleriyle adlandırmış olduğumuz 6 bitlik grupların ilk ve son biti ikisi birden yan yana düşünüldüğünde 0'dan 3'e kadar 4 sayı değeri alabilir. Bu sayı değerine x diyelim. Ortada kalan 4 bitlik kısım da 0 ila 15 değerleri arasında bir karşılığa sahiptir. Buradan çıkacak olan sayıya da y diyelim. Tablomuzda x ve y nin kesiştiği yerdeki sayı 4 bit değerinde bir sayıdır ve S tablosu işleminin bir sonucudur.

Örneğin B_2'yi ele alalım. Yani denklem olarak ifade edecek olursak $S_2(B_2)$'yi bulacağız.

B_2=010001

İlk ve son bit alındığında $x = (01)_2 = 1$;

Ortadaki 4 bit alındığında $y=(1000)_2=8$ olarak bulunur. Yani S_2 tablosunda 1. satır 8. sütundaki sayı bizim işlemimizin cevabıdır. İşlem sonucu = $12 = (1100)_2$ dir.

Bu şekilde işleme sokmamız gereken 7 tane altılı grup ve 7 tane S tablosu daha vardır.Bunları işleme soktuğumuzda çıkan sonuç:

$S_1(B_1)S_2(B_2)S_3(B_3)S_4(B_4)S_5(B_5)S_6(B_6)S_7(B_7)S_8(B_8)$=01011100100000101011010110010111

bulunur. Bundan sonra *f* fonksiyonunu bulmak için tek yapmamız gereken elimizdeki sonucu son bir permütasyon tablosuna göre işleme sokmaktır.

Bu permütasyon tablosunun adı P Tablosu olarak geçmektedir. Bu tablo 32 bitlik bloğu işleme sokar ve yerlerini permüte edip 32 bitlik olarak tekrar çıkarır.

Tablo 3.8: 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=0010 0011 0100 1010 1010 1001 1011 1011 bulunur.

Bu tablo işleminden sonra *f* fonksiyonunun tam olarak ne tür bir işlem yaptığını anlamış olduk. Denklemsel olarak ifade edecek olursak:

$f=P(S_1(B_1)S_2(B_2)..... S_8(B_8))$ (3.3)

f fonksiyonunu bulduğumuza göre (3.2) de verdiğimiz formülü hatırlayacak olursak: $R_1= L_0 + f(R_0,K_1)$ olacaktır. Bu işleme göre tek yapmamız gereken L_0 ile bulduğumuz *f* fonksiyonunun sonucunu toplamak olacaktır. Buna göre;

L_0 = 1100 1100 0000 0000 1100 1100 1111 1111

\+ $f(R_0,K_1)$= 0010 0011 0100 1010 1010 1001 1011 1011

R_1= 1110 1111 0100 1010 0110 0101 0100 0100 bulunur.

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

L_{16}= 0100 0011 0100 0010 0011 0010 0011 0100
R_{16}= 0000 1010 0100 1100 1101 1001 1001 0101

Bu ulaşılan sayı bloğunu ters çevirdikten sonra yani R_{16} L_{16} şekline getirdikten sonra son bir tablo işlemine sokalım. Bu son tabloya bitiş permütasyonu (final permütation) denir. Aslında bu son permütasyon tablosu yukarıda (Tablo 3.5) vermiş olduğumuz başlangıç permütasyonunun inversi yani tersidir. Bu yüzden adlandırırken IP^{-1} olarak adlandırabiliriz.

Tablo 3.9: Başlangıç permütasyonunun tersi

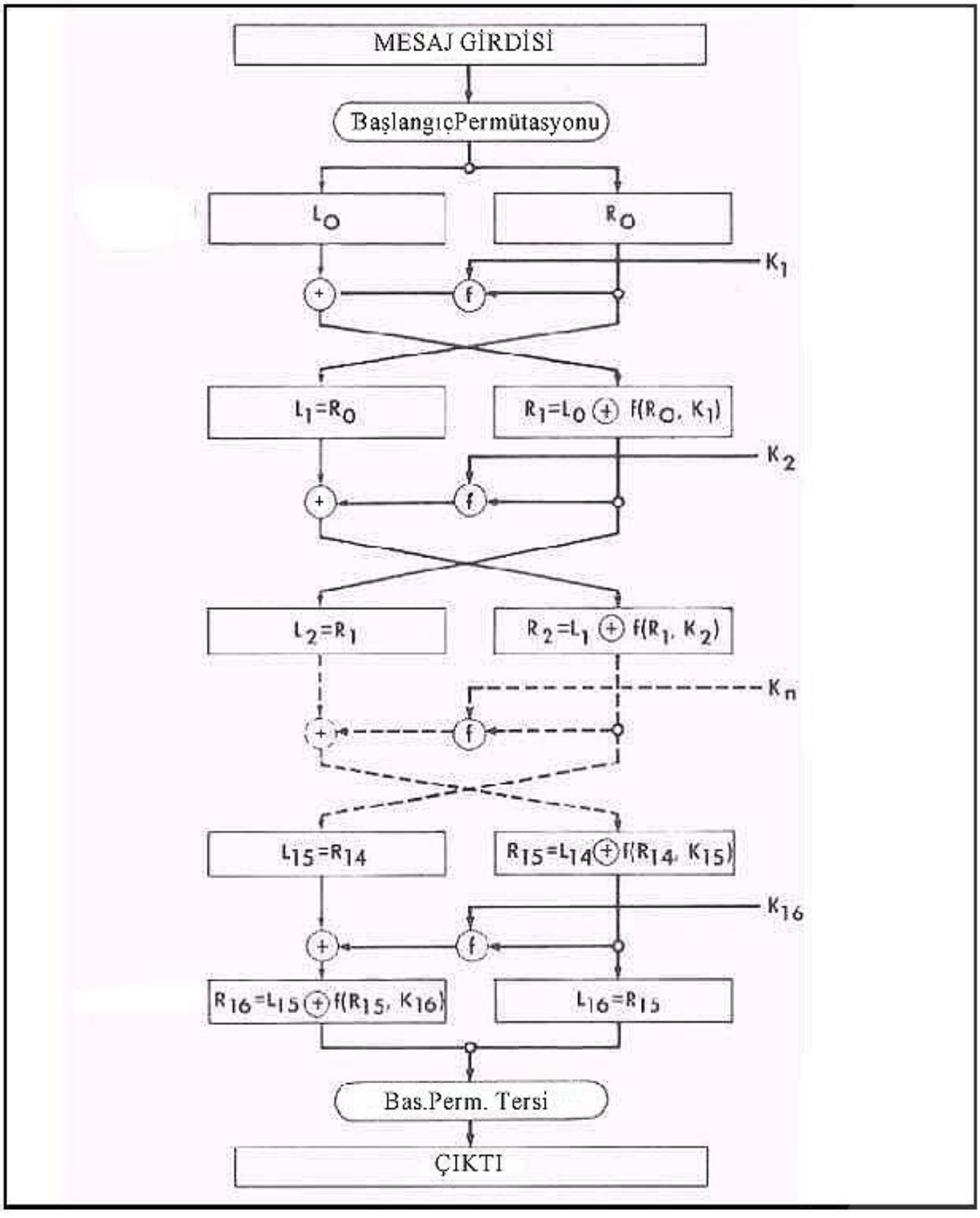
IP –1							
40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	9	49	17	57	25

IP^{-1}= 1000 0101 1110 1000 0001 0011 0101 0100 0001 0101 0110 1000 0000 0101

Şifrelenmiş hal = 85E813540F0AB405

Bu bulduğumuz sonuç da mesajımızın şifrelenmiş halidir.

3.4.4. DES algoritmasının şema üzerinde gösterimi



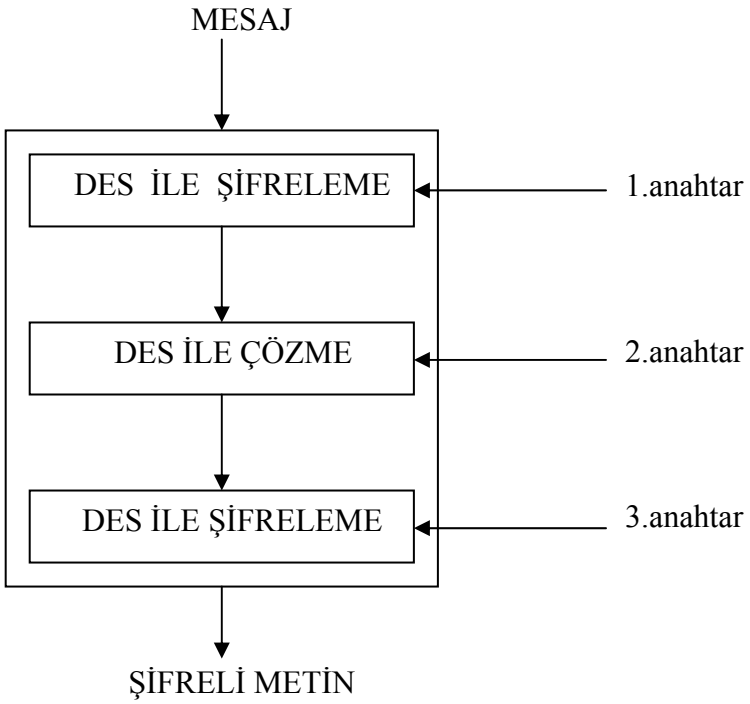
Şekil 3.2: DES algoritmasının yapısı

3.5. Triple-DES (3-DES) ve İşleyiş Mantığı

Triple-DES, DES`in daha çok güvenlik sağlayan bir çeşididir. Bu metod kriptolama anahtarındaki bitleri 3 katına çıkaran, DES`in 3 kez kullanımına dayalıdır. Yani DES'in toplam anahtar uzunluğu 64 bit iken Triple-DES'in ki 192 bit yani 24 byte (24 karakter)dir. Triple-DES kullanımı DES kullanımına göre 2 kat daha fazla güvenlik sağladığına inanılmaktadır. Bu 112 bitlik koda sahip olunması demektir. Ayrıca kodlama süresini de doğru orantılı olarak arttırmaktadır. Anahtar alanı 2^{112} veya 2^{168} sayısına ulaşınca bugün için veya tahmin edilebilir bir gelecekte çözülmesi mümkün olmayan bir kod olmaktadır. Buna "strong crypto" denilir.

Triple-DES üç kat daha fazla iş yaptığı için DES'e göre üç kat daha yavaştır. Fakat yavaş olduğu oranda da güvenlidir. Bu şifreleme algoritmasında dikkat edilmesi gereken tek husus üç tane anahtar bloğunun veya üçünden ikisinin (bire iki veya ikiye üç) aynı olmasından kaçınmaktır. Çünkü eğer anahtarlar aynı olursa işlem DES işleminden farksız ve bir o kadar da ağır yapılır. DES işleminden farksız olması da günün teknolojisiyle 10 saate kalmadan kodun çözülmesi anlamına gelir.

Triple-DES tarafından kullanılan teknik EDE (encrypt-decrypt-encrypt) olarak bilinmektedir. Düzyazı Triple-DES anahtarının ilk 8 byte`ı ile kriptolanır. Şifreleme mantığı DES in aynısıdır. Sonra mesaj anahtarın ortadaki 8 byte`ı ile dekriptolanır. Ve son olarak anahtarın son 8 byte`ı ile kriptolanarak 8 byte`lık bir blok elde edilir. Eğer her anahtar aynı ise işlem DES kullanmakla aynıdır. Eğer her anahtar birbirinden farklı ise anahtarın ortasındaki 8 byte ile dekriptolama işlemi mesajı iyice karmaşık hale getirir.3-DES'in işleyişi şekil 3.3'de gösterilmiştir.



Şekil 3.3: Triple-DES'in işleyiş şeması

DES'in ilk çıktığı sıralarda başka bir kripto algoritmasının yazarları olan Martin Hellman ve Whitman Diffie DES'e inanmadıklarını söylüyorlardı. Hellman NBS'ye (Amerikan Ulusal Standartlar Teşkilatı) yazdığı mektupta, DES'in sanıldığı kadar güvenli olmadığını ve akıllı bir organizasyonun DES'i kırabileceğini iddia etti.

Bundan sonra Diffie-Hellman ikilisi DES'e karşı bir "Brute Force" (deneme yanılma yolu yani 2^{56} ihtimali tek tek denemek) saldırı başlattı. Diffie-Hellman bu özel amaç için birbirine bağlı çalışan bir milyon çipi olan ve saniyede bir milyon anahtar deneyebilen bir bilgisayar kullandılar ve böyle bir makinenin maliyeti 20 Milyon dolardı.

1998'de FBI'dan John Gilmore liderliğinde bir ekip 220.000 dolar harcadı ve bütün olası DES anahtarlarını 4.5 günde deneyebilen bir sistem kurdu. 17 Temmuz 1998'de bir DES mesajını 56 saatte açabildiklerini açıkladılar. Bunu yaparken "Deep Crack" adını verdikleri ve her biri 64 işlemciden oluşan 27 kartı kullanan ve saniyede 90 milyar anahtar test edebilen bir bilgisayarı kullandılar. Bunun üzerine FBI, DES'i kırabilme gücüne sahip olduğunu açıklamıştır.

BÖLÜM 4: SCRABBLED, CASCADED VE COMBINED KEM-DEM

Simetrik ve asimetrik şifreleme sistemleri [8] veri iletiminde mesajın gizliliğinin korunması ve başkaları tarafından okunamamasını sağlamak amacıyla kullanılmaktadır. Hybrid şifreleme sistemleri de simetrik ve asimetrik şifreleme algoritmalarının iyi yönlerini birleştirmektedir. Hybrid şifreleme sistemlerinde genel olarak kullanılan yapıda, veri öncelikle simetrik anahtarla şifrelenmekte ve verinin şifrelenmesinde kullanılan simetrik anahtar da asimetrik şifreleme algoritması ile şifrelenmektedir.

Hybrid şifreleme sistemlerinde yapılan son çalışmalarda [14, 15] Key Encapsulation Mechanism (KEM) ve Data Encapsulation Mechanism (DEM) yapılarının geliştirilmesine odaklanılmıştır. Bu model hybird şifreleme sistemini iki parçaya ayırmaktadır: KEM ve DEM. Hybrid şifrelemede kullanılan sistemlerden bir tanesi Tag-KEM/DEM [14] sistemidir. Bu sistemde KEM'in oluşturulması için öncelikle random bir anahtar üretilmekte, sonrasında bu anahtar dışarıdan verilen bir tag ile şifrelenmektedir. Diğer bir hybird şifreleme sistemi olan Fujisaki-Okamoto KEM-DEM [15] sistemi zayıf bir yapıya sahip olan simetrik ve asimetrik şifreleme sistemlerini güçlü bir yapıya dönüştürmektedir.

Kriptolojide en çok dikkat edilecek noktalardan biri, veri bütünlüğü (message integrity), kimlik kontrolü ve anahtar güvenliğini sağlarken harcanan zamandır. Tag-KEM/DEM [14] ve Fujisaki-Okamoto KEM-DEM [15] sistemlerinin ikisi de bu özellikleri sağlamaktadır. Ancak her iki sistemde bizim burada önerdiğimiz combined KEM-DEM sisteminden daha yavaş çalışmaktadır.

Bu bölümde 3 farklı algoritma anlatılmıştır. Üç algoritmada da Hybrid şifrelemede kullanılan KEM-DEM [8, 14 ,15] yapısı temel olarak alınmış ve bu yapı kullanılarak scrabbled KEM-DEM, cascaded KEM-DEM ve combined KEM-DEM sistemleri geliştirilmiştir. Mevcut KEM-DEM yapısı [8, 14 ,15] aynı anda yapılabilecek iki

yönlü bir saldırıya açıktır. Saldırgan aynı anda hem KEM üzerinde hem de DEM üzerinde çalışabilir. Yani saldırganın elinde şifrelenen metin ile ilgili, üzerinde çalışabileceği iki tane veri bulunmaktadır. Eğer KEM-DEM yapılarının birisinde bir zayıflık varsa tüm şifreleme süreci başarısız olacaktır. Saldırgan böylelikle tüm bilgileri elde edebilir. Bizim burada sunduğumuz üç sistemde de bu zayıflığı ortadan kaldıran bir KEM-DEM yapısına sahiptir. Bunu sağlamak için klasik KEM-DEM yapısının sonuna (4.8) numaralı eşitlik ile gösterilen karıştırma (scrabble) algoritması (EW) eklenmiştir.

Scrabbled KEM-DEM sisteminde KEM ve DEM birbirine alıcının açık anahtarı kullanılarak karıştırma algoritması (EW) ile karıştırılmaktadır. Karıştırma algoritmasında KEM ve DEM öncelikle permütasyona tabi tutularak karıştırılmakta, sonrasında da karıştırılan KEM-DEM simetrik anahtarla şifrelenmektedir. Bu karıştırma işlemini sadece doğru anahtara sahip olan alıcı çözebilir.

Bazı network yapılarında, kullanılan iletişim protokolüne bağlı olarak, alıcı kimliğinin tespit edilmesi çok zor olmayabilir. Bu networklarda hybrid şifrelemede kullanıldığımız son adım karıştırma algoritmasını daha etkin yapmak için, karıştırma işlemini alıcının açık anahtarı yerine, random bir anahtarla yapmak daha kullanışlı olacaktır. Daha sonra bu anahtar alıcının açık anahtarı ile şifrelenerek [9-11,16], karıştırılan KEM-DEM'e eklenmekte ve alıcıya gönderilmektedir. Böylelikle alıcının kimliği biliniyor olsa bile, hem KEM-DEM'in birbirinden ayrılması hem de tüm mesajın sadece alıcı tarafından çözülmesi garanti altına alınmış olur. Çünkü açık anahtar ile şifrelenen mesajı çözebilecek olan gizli anahtara sadece alıcı sahiptir. Bu şifreleme işlemi için kullanılan sistem cascaded KEM-DEM sistemidir.

Üçüncü sistem; combined KEM-DEM, uzun süreli kurulan bir iletişimde, tüm iletişimin bütünlüğünün kontrolünü sağlamakla kalmaz, aynı zamanda hybrid şifreleme süresini de kısaltır. Yapılan simülasyon ile combined KEM-DEM sisteminin Tag-KEM/DEM [14] ve Fujisaki-Okamoto KEM-DEM [15] sistemlerinden daha hızlı olduğu ispatlanmış ve sonuçlar tablo 4.1'de gösterilmiştir. Combined KEM-DEM sistemi, verinin hybrid olarak şifrelenme/deşifrelenme süresi kısaltmış olmakla birlikte güvenlikle ilgili tüm beklentilere de (alıcının kimlik tespiti,

verinin yolda değişip değişmediğinin kontrolü, anahtar güvenliği gibi) cevap vermeye devam eder. Klasik oturum anahtarı daha da (session key) geliştirilmiş ve aynı oturum içinde bile gönderilen her mesaj için farklı bir anahtarın kullanılması sağlanmıştır.

4.1. Scrabbled KEM-DEM Sistemi

Veri şifrelemesinde kullanılan simetrik ve asimetrik anahtarlar, hybrid şifrelemede bir arada kullanılarak birbirlerinin eksik yönlerini tamamlamaktadırlar. Bu yapıya kimlik kontrolü (digital signature) ile veri bütünlüğünün kontrolü de (message integrity) eklenerek tam bir veri güvenliği sağlanmış olacaktır.

Klasik KEM-DEM yapısında, metnin şifrelenmesinde kullanılan simetrik anahtar, asimetrik anahtarla şifrelenmekte ve simetrik anahtar ile şifrelenen metne eklenmektedir [8,14,15,17,18]. Ancak bu yapı iki yönlü saldırıya açıktır. Saldırgan aynı anda hem KEM üzerinde hem de DEM üzerinde ayrı ayrı çalışarak, mesajın şifresini çözebilir veya KEM içine gömülü olan anahtarı elde edebilir. Yani saldırgan için üzerinde çalışabileceği iki adet veri bulunmaktadır. Eğer KEM veya DEM yapılarından birisinde bir zayıflık varsa ve saldırgan KEM ve DEM'den birisini çözmeyi başarırsa tüm metni ele geçirmiş olacaktır.

Bu noktada bizim önerdiğimiz sistem devreye girmektedir. Bu ikili yapıyı tek bir blok haline dönüştürerek iki koldan yapılabilecek saldırı engellenmiş olacaktır. Yani KEM ve DEM, (4.8) numaralı denklem ile gösterilen karıştırma algoritması (*EW*) yardımıyla karıştırılacak ve tek bir blok haline dönüştürülecektir. Tek blok haline gelen yeni yapıda, sadece doğru anahtara sahip olan kişi KEM-DEM'i birbirinden ayırabilecektir. Böylelikle hem anahtarın güvenliği hem de verinin güvenliği sağlanmış olacaktır.

Sistem KEM ve DEM'in elde edilmesi olmak üzere iki kısımdan oluşmakta ve en son karıştırma işlemi ile tek blok haline dönüştürülmektedir. Karıştırma işleminde kullanılacak olan anahtar ise alıcının açık anahtarıdır (public key). Dolayısıyla

karıştırılan KEM ve DEM sadece doğru alıcı tarafından birbirinden ayrılabilecektir ve orijinal mesaj elde edilebilecektir.

Sistemde ilk olarak orijinal metnin (plain text) şifrelenmesinde kullanılacak olan simetrik anahtar oluşturulmakta, oluşturulan bu simetrik anahtarla orijinal metin şifrelenerek DEM elde edilmektedir. Daha sonra, metnin şifrelenmesinde kullanılan simetrik anahtarın elde edilmesinde kullanılan random anahtar k_{sym}, mesajı gönderen kişinin özel anahtarı ile (4.6) numaralı eşitlik kullanılarak şifrelenmektedir. Bu işleme dijital imza denilmektedir. Dijital imzaya orijinal metnin hash özeti ve gönderen kişinin açık anahtarı da eklenerek asimetrik şifreleme algoritması ile alıcının açık anahtarı kullanılarak şifrelenmekte ve KEM elde edilmektedir.

En son adımda, karıştırma algoritması (*EW*) ile KEM ve DEM karıştırılmakta ve bu ikili yapı tek bir bloğa dönüştürülmektedir. Burada kullanılan karıştırma algoritmasında öncelikle KEM ve DEM permütasyona tabi tutulur ve sonrada şifrelenir. Permütasyon işleminin nasıl yapıldığının bir önemi yoktur. Bu işlemde KEM ve DEM'in byte'larına permütasyon işlemi uygulanmaktadır. Permütasyon işleminden sonra yapılan şifreleme işlemi de simetrik şifrelemedir ve anahtar olarak alıcının açık anahtarı kullanılmaktadır. Bütün bu işlemlerden sonra elde edilen çıktı alıcıya gönderilir.

Bu sistemdeki önemli noktalardan bir tanesi de simetrik şifrelemede kullanılan anahtar ile KEM içine gömülen anahtarın aynı olmamasıdır. KEM içine gömülen anahtar (k_{sym}) orijinal mesajın şifrelenmesinde kullanılacak olan anahtarın (k_{group}) elde edilmesinde kullanılacaktır. Yani simetrik şifrelemede kullanılacak olan anahtarı elde edebilmek için ikinci bir anahtar üreteci kullanılmaktadır. Böylelikle herhangi bir şekilde üçüncü kişiler tarafından KEM içine gömülü olan anahtar ele geçirilmiş bile olsa, anahtar üreteci olmadan bu anahtar işlerine yaramayacaktır. Bu yapı anahtar güvenliğini daha da arttırmakta ve dışarıdan gelebilecek saldırılara karşı güvenliği attırmaktadır.

Aşağıda ayrıntılı olarak anlatılan sistemde; gönderenin kimlik kontrolü, mesajın bütünlük kontrolü, anahtarın güvenliği, mesajın simetrik anahtar ile şifrelenmesi ve en son KEM ve DEM'in karıştırılması anlatılmıştır.

4.1.1. Tanımlar

$string$: Herhangi bir string

φ_{sym} : *String*'den random anahtarın elde edilmesini sağlayacak olan algoritma. Algoritmaya aynı string birden fazla kez girilse bile farklı bir anahtar üretecek yapıya sahiptir.

k_{sym} : Alıcıya gönderilecek olan random anahtar. Bu anahtar, simetrik şifreleme algoritmasında kullanılacak olan simetrik anahtarın (k_{group}) üretilmesinde de kullanılmaktadır.

δ_{cons} : Simetrik anahtarı (k_{group}) üretecek olan algoritma.

k_{group} : Simetrik şifrelemede kullanılacak olan anahtar. Bu anahtarın ismini group olarak adlandırdık, çünkü simetrik anahtar olarak bir veya 3DES gibi birkaç anahtar kullanılabilir. Algoritma her ikisine de uyumlu olarak çalışabilmektedir.

m : Şifrelenecek metin, orijinal metin.

c : Simetrik anahtarla şifrelenmiş metin

E^{sym} : Simetrik şifreleme algoritması [7]

H : Hash fonksiyonu.

h : Şifrelenmemiş metnin (plain text) hash değeri.

sk_{sender} : Mesajı gönderen kişinin gizli anahtarı.

DQ^{asym} : Asimetrik deşifreleme algoritması.

x : Dijital imza.

pk_{sender} : Mesajı gönderen kişinin açık anahtarı

y : Anahtar kapsülü (KEM). Dijital imza içine gömülü olarak gönderilen random anahtar k_{sym}, mesajın yolda değişmediğinin (mesaj bütünlüğünün) kontrolünde kullanılan h ve gönderen kişinin kimlik tespitinde kullanılan pk_{sender} bu kapsülün içinde yer almaktadır.

EQ^{asym} : Asimetrik şifreleme algoritması

$pk_{receiver}$: Alıcının açık anahtarı

EW : Karıştırma (Scrabble) algoritması. Şifrelenmiş metin (DEM) ile anahtar kapsülünü (KEM) karıştırıp şifreleyen algoritma

ω : Alıcıya gönderilecek olan nihai mesaj

DW : KEM ile DEM birbirinden ayıran algoritma

$sk_{receiver}$: Alıcının gizli anahtarı

4.1.2. Şifreleme (Encription)

Scrabbled KEM-DEM sistemindeki şifreleme işlemi (4.1) numaralı eşitlikte verilmiştir.. Sistemin ayrıntıları ve içinde kullanılan algoritmalar da (4.2)'den (4.8)'e kadar olan eşitlikler ile verilmiştir.

$$\omega \leftarrow EW_{pk_{receiver}}(EQ^{asym}_{pk_{receiver}}(EQ^{asym}_{sk_{sender}}(k_{sym}), H(m), pk_{sender}), E^{sym}_{k_{group}}(m)) \tag{4.1}$$

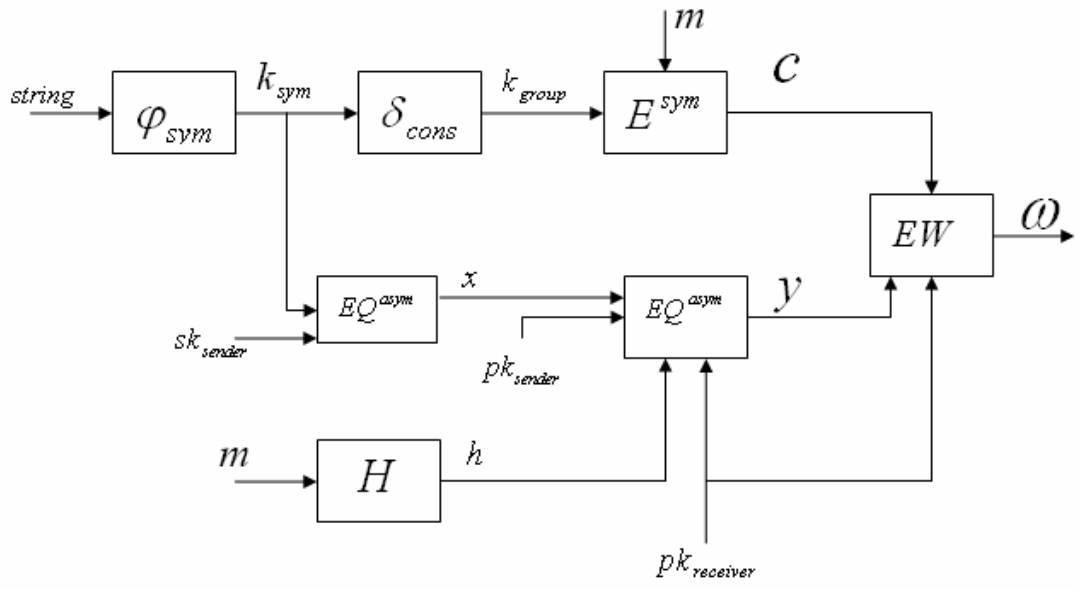
Scrabbled KEM-DEM sistemi şifreleme işlem basamakları:

Anahtar üretimi:

$$k_{sym} \leftarrow \varphi_{sym}(string) \tag{4.2}$$
$$k_{group} \leftarrow \delta_{cons}(k_{sym}) \tag{4.3}$$

Şifreleme:

$$c \leftarrow E^{sym}_{k_{group}}(m) \tag{4.4}$$
$$h \leftarrow H(m) \tag{4.5}$$
$$x \leftarrow EQ^{asym}_{sk_{sender}}(k_{sym}) \tag{4.6}$$
$$y \leftarrow EQ^{asym}_{pk_{receiver}}(x, h, pk_{sender}) \tag{4.7}$$
$$\omega \leftarrow EW_{pk_{receiver}}(c, y) \tag{4.8}$$



Şekil 4.1: Scrabbled KEM-DEM sistemi şifreleme blok diyagramı

Simetrik anahtarın Oluşturulması: Sistemde ilk olarak metnin şifrelenmesinde kullanılacak olan simetrik anahtar oluşturulmaktadır. Anahtarın üretimi iki aşamalıdır. İlk aşamada, simetrik anahtarın üretiminde kullanılacak olan random anahtar k_{sym} elde edilmektedir (4.2). φ_{sym} algoritması ile elde edilen k_{sym} değeri random bir değerdir. Algoritmaya (φ_{sym}) aynı *string* değer girilse bile her seferinde farklı bir sonuç üretecek yapıya sahiptir. Elde edilen k_{sym} değeri aynı zamanda KEM içinde alıcıya gönderilecek olan değerdir. δ_{cons} algoritması, random anahtardan (k_{sym}), simetrik anahtarın (k_{group}) elde edilmesinde kullanılmaktadır ve aynı değer girildiğinde aynı çıktıyı üreten bir yapıya sahiptir. Buradan elde edilen simetrik anahtar k_{group} ile orijinal metin şifrelenmektedir.

Mesajın Şifrelenmesi (Encryption): Burada orijinal metin simetrik anahtar ile şifrelenmektedir (4.4). Şifreleme işleminde kullanılan simetrik anahtar k_{group} tur. Elde edilen *c* değeri DEM [8,14,15] olarak adlandırılmaktadır. Veri bütünlüğünü (message integrity) kontrol edebilmek için mesajın hash özeti (4.5) numaralı eşitlik ile elde edilmekte ve elde edilen hash özeti KEM içine eklenmektedir. Mesaj alındığında, bütünlük kontrolü yapabilmek için, KEM içindeki h değeri ile metnin şifresinin çözülmesinden sonra elde edilen h değeri karşılaştırılmaktadır. Eğer değerler aynı değilse, mesajın iletiminde bir problem olduğu veya mesaja müdahale edildiği ortaya çıkacaktır. KEM'in içine gönderen kişinin kimlik doğrulamasının yapılabilmesi için, dijital imza eklenmektedir (4.6). Kimlik kontrolü için random anahtar (k_{sym}) gönderenin gizli anahtarı (private key) ile şifrelenmektedir. Gizli anahtar sadece gönderen kişinin kendisinde bulunduğu için kimlik kontrolünde kullanılabilecek en uygun araçtır. Alıcı bu anahtarı tekrar elde edebilmek için; mesajı gönderen kişinin açık anahtarını (public key) kullanarak asimetrik deşifreleme algoritması ile şifresini çözüyor. Eğer başlangıçta anahtar doğru kişi tarafından şifrelenmiş ise bu işlem sonucunda elde edilen anahtar doğru anahtardır ve bu anahtar kullanılarak metnin şifrelenmesinde kullanılan simetrik anahtar doğru olarak elde edilebilir.

(4.7) numaralı eşitlik KEM) olarak adlandırılmaktadır ve random anahtar üzerinde kimlik kontrol bilgisini, orijinal mesajın hash özetini ve mesajı gönderen kişinin açık anahtar bilgilerini içerir. Gönderenin kimliğini tespit edebilmek ve kimlik doğrulamasında kullanabilmek için pk_{sender} bilgisi, veri bütünlüğünü, yolda meydana gelebilecek değişiklikleri kontrol etmek için mesajın Hash özeti (h) ve gönderen kişinin sayısal imzası (x) KEM'e eklenmektedir. Burada gönderilen mesajdan mesajı gönderen kişinin kimliğinin tespit edilmesinin mümkün olmadığı kabul edilmekte ve bu nedenle pk_{sender} bilgisi KEM'e eklenmektedir. Anahtara sadece alıcının ulaşabilmesi için x, h ve pk_{sender} değerleri $pk_{receiver}$ ile asimetrik algoritma ile şifrelenmektedir.

Şifrelenmiş mesaj (c) (DEM) ile anahtar kapsülü (y) (KEM), (4.8) numaralı eşitlik karıştırma algoritması ile birbirine karıştırılmaktadır. Bu karıştırma işlemini sadece doğru açık anahtara sahip alıcı çözebilir. Burada gönderilen mesaj üzerinden alıcının kimliğinin tespit edilmesinin mümkün olmadığı kabul edilmiştir.

4.1.3. Şifreyi Çözme (Decryption)

Scrabbled KEM-DEM sistemi şifreyi çözme işlem basamakları:

$$(\hat{c}, \hat{y}) \leftarrow DW_{pk_{receiver}}(\omega) \tag{4.9}$$

$$(\hat{x}, \hat{h}, \hat{pk}_{sender}) \leftarrow DQ^{asym}_{sk_{receiver}}(\hat{y}) \tag{4.10}$$

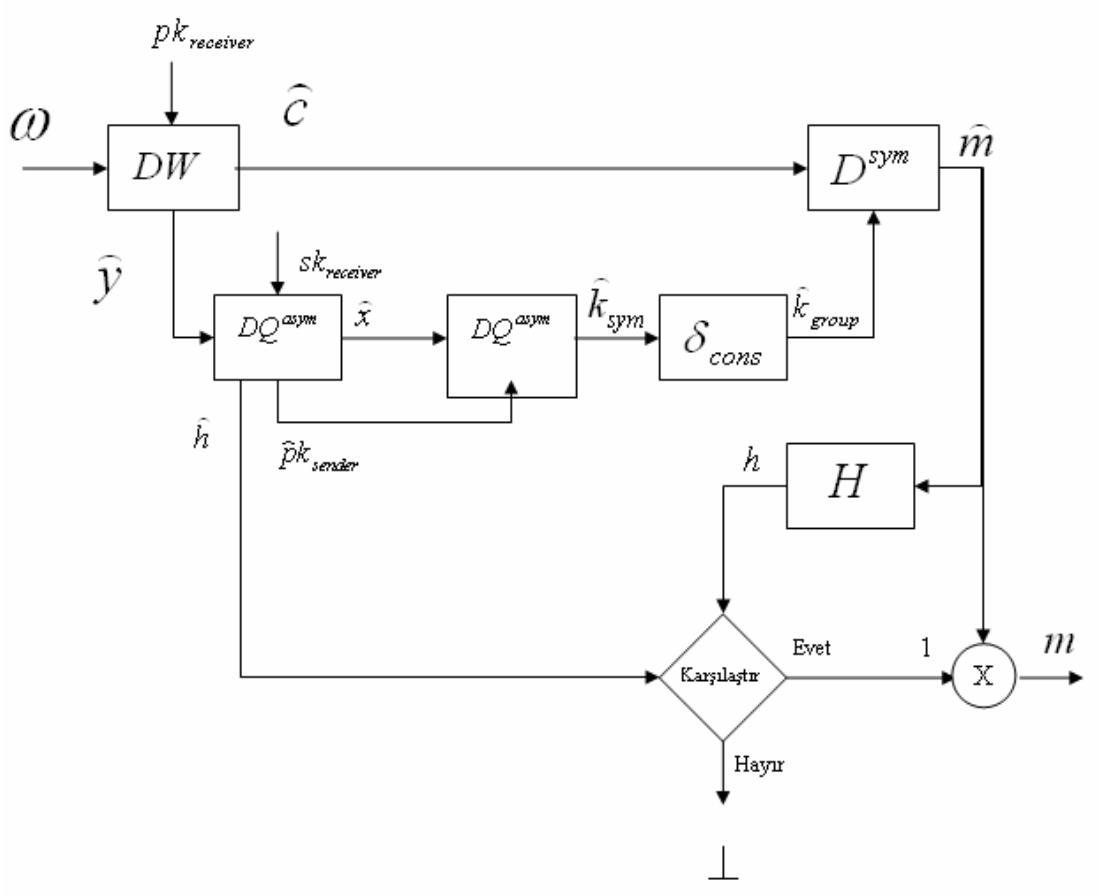
$$\hat{k}_{sym} \leftarrow EQ^{asym}_{\hat{pk}_{sender}}(\hat{x}) \tag{4.11}$$

$$(\hat{k}_{group}) \leftarrow \delta_{cons}(\hat{k}_{sym}) \tag{4.12}$$

$$\hat{m} \leftarrow D^{sym}_{\hat{k}_{group}}(\hat{c}) \tag{4.13}$$

$$h \leftarrow H(\hat{m}) \tag{4.14}$$

Eğer $h = \hat{h}$ ise sonuç $= m$ değilse sonuç $= \perp$



Şekil 4.2: Scrabbled KEM-DEM sistemi şifreyi çözme blok diyagramı

Şifreli bir mesaj alındığında yapılması gereken ilk işlem KEM ve DEM'in birbirinden ayrılmasıdır. Verinin şifrelenmesi aşamasında karıştırma algoritması ile yapılan karıştırma işlemini sadece doğru anahtara sahip olan alıcı çözülebilir. (4.9) numaralı eşitlik ile bu değerler birbirlerinden ayrıldıktan sonra sıra KEM'in içindeki anahtarı elde etmektedir. Alıcı kendi özel anahtarını ($sk_{receiver}$) kullanarak anahtar kapsülünden ($\hat{y}$), kimlik kontrol değerini ($\hat{x}$), hash özetini ($\hat{h}$) ve mesajı gönderen kişinin açık anahtarını ($\hat{pk}_{sender}$) elde eder (4.10).

Elde edilen kimlik kontrol değeri ($\hat{x}$) mesajı gönderen kişinin açık anahtarı ($\hat{pk}_{sender}$) kullanılarak çözülür. Eğer mesaj doğru kişi tarafından gönderilmiş ise, simetrik şifre çözme algoritmasında kullanılacak olan anahtar doğru şekilde elde edilebilir.

Şifreleme işleminde x değerinin elde edilmesi için, mesajı gönderen kişi kendi özel anahtarını (sk_{sender}) kullanmış ve k_{sym}'yi şifrelemişti. Bu anahtar sadece kendisinde bulunmaktadır. Anahtarın geri elde edilmesinde işlem tersine çevrilir ve pk_{sender} ile şifresi çözülürse k_{sym}'nin orijinal hali elde edilir. Elde edilen k_{sym} doğru ise DEM'in şifresi çözüldüğünde elde edilecek hash özeti ile KEM'den elde edilen h değerleri aynı olacaktır. Böylelikle aynı anda hem veri bütünlüğü kontrolü hem de gönderen kişinin kimlik kontrolü yapılmaktadır.

(4.12) numaralı eşitlik ile şifreli mesajın (DEM'in) şifresinin çözülmesinde kullanılacak olan simetrik anahtar (4.11) numaralı eşitlikten elde edilen $\widehat{k}_{sym}$ kullanılarak üretilir. Simetrik anahtar kullanılarak DEM'in şifresi çözülür ve mesajın orijinal hali (plain text) elde edilir (4.13).

Veri bütünlüğünün ve mesajın doğruluğunun kontrol edilmesi için $\widehat{m}$'nin Hash özeti elde edilir. Eğer anahtar kapsülünden (KEM'den) elde edilen hash özeti $\widehat{h}$ ile $\widehat{m}$'den elde edilen hash özeti h aynı ise ($h = \widehat{h}$) mesaj doğru olarak çözülmüştür ve sonuç=$\widehat{m}$ olur. Değilse sonuç = $\perp$ olur.

4.2. Cascaded KEM-DEM Algoritması

Scrabbled KEM-DEM sisteminde, mesajın şifrelenmesi işleminin son basamağında kullanılan $\omega \leftarrow EW_{pk_{receiver}}(c, y)$ hesaplaması ile KEM ve DEM'in birleştirilmesi ve birbirinden ayrılması, alıcının açık anahtarına bağlanmıştır. Şifreli mesajı alacak kişinin kim olduğu bilinmediği sürece, KEM ve DEM'i birbirinden ayrılması ve metnin şifresinin çözülmesi mümkün olmayacaktır.

Ancak bazı sistemlerde kullanılan iletişim protokolüne bağlı olarak haberleşme sırasında alıcı kimliğinin tespit edilmesi çok da zor olmayabilir. Eğer alıcının kimliği tespit edilirse, herhangi bir kişi, alıcının açık anahtar bilgisini kullanarak (4.8) numaralı eşitlik ile yapılan karıştırmayı çözebilir. Dolayısıyla alıcı kimliğinin kolay

tespit edilebildiği sistemlerde bu işlem basamağının kullanılmasının bir anlamı olmayacaktır.

Alıcı kimliğinin kolaylıkla tespit edilebildiği sistemlerde, bu işlem basamağını daha etkin hale getirmek için, karıştırma işleminde $pk_{receiver}$'ın kullanımı yerine, her seferinde değişen bir anahtarın kullanılması daha güvenli olacaktır. Bu anahtar alıcının açık anahtarı ile şifrelenecek [9-11,16] ve nihai metne eklenecektir. Böylelikle alıcının kim olduğu bilinse bile gizli anahtara sahip olan kişi –ki bu kişi de alıcının kendisidir– ancak bu anahtarı elde edebilecektir.

Bu sistemde kısaca; KEM ve DEM'i karıştırmada kullanılacak olan anahtar Ω üretilecek, bu değer (Ω) alıcının açık anahtarı ($pk_{receiver}$) ile şifrelenecek ve λ değeri olarak hesaplanacak, KEM ve DEM bu anahtar kullanılarak karıştırılacak, en son karıştırılan KEM-DEM'e şifrelenmiş olan anahtar (λ) eklenecek ve alıcıya gönderilecektir. Sistem ve açıklamaları aşağıda verilmiştir.

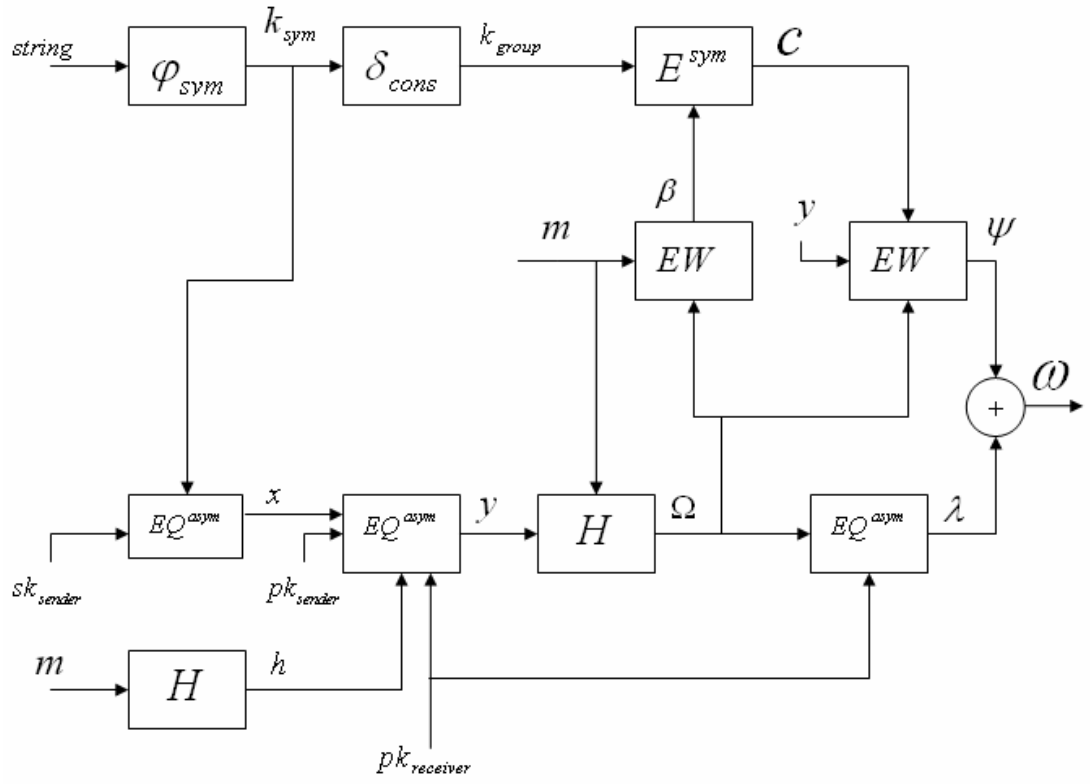
4.2.1. Tanımlar

Ω : KEM ve DEM'i karıştırma işleminde kullanılacak olan random anahtar

λ : Anahtar kapsülü. En son gönderilecek metne eklenecek olan değer.

β *:* Orijinal mesajın karıştırılmasından elde edilen metin.

ψ *:* KEM-DEM'in karıştırılmasından elde edilen metin.



Şekil 4.3: Cascaded KEM-DEM sistemi şifreleme blok diyagramı

4.2.2. Şifreleme İşlemi

Anahtar üretiminde kullanılan (4.16) ve (4.17) numaralı eşitlikler ile Scrabbled KEM-DEM sistemindeki (4.2) ve (4.3) numaralı eşitlikler, y değerinin elde edilmesinde kullanılan (4.18), (4.19) ve (4.20) numaralı eşitlikler ile (4.5), (4.6) ve (4.7) numaralı eşitlikler aynıdır. Cascaded KEM-DEM sistemi (4.15) numaralı denklem ile gösterilmektedir.

$$\omega \leftarrow (EQ^{asym}_{pk_{receiver}}(H(m,y)) \parallel EW_{\Omega}(E^{sym}_{k_{group}}(m), EQ^{asym}_{pk_{receiver}}(x,h,pk_{sender}))) \tag{4.15}$$

Anahtar üretimi ve şifreleme basamakları (4.16)'dan (4.26)'ya kadar olan eşitliklerle verilmiştir.

Cascaded KEM-DEM scheme şifreleme işlem basamakları:

Anahtar Üretimi:

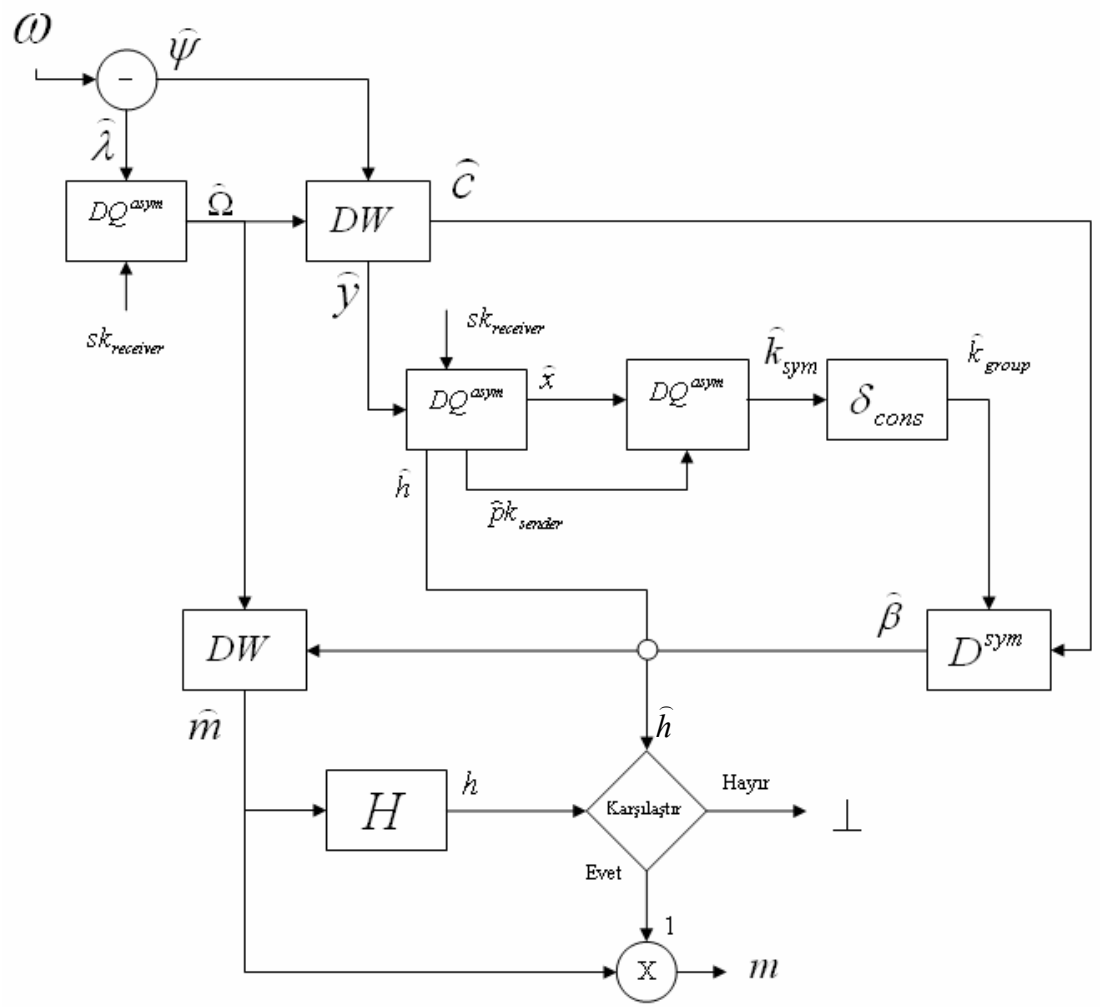
$$k_{sym} \leftarrow \varphi_{sym}(string) \tag{4.16}$$
$$k_{group} \leftarrow \delta_{cons}(k_{sym}) \tag{4.17}$$

Şifreleme:

$$h \leftarrow H(m) \tag{4.18}$$
$$x \leftarrow EQ^{asym}_{sk_{sender}}(k_{sym}) \tag{4.19}$$
$$y \leftarrow EQ^{asym}_{pk_{receiver}}(x, h, pk_{sender}) \tag{4.20}$$
$$\Omega \leftarrow H(m, y) \tag{4.21}$$
$$\lambda \leftarrow EQ^{asym}_{pk_{receiver}}(\Omega) \tag{4.22}$$
$$\beta \leftarrow EW_{\Omega}(m) \tag{4.23}$$
$$c \leftarrow E^{sym}_{k_{group}}(\beta) \tag{4.24}$$
$$\psi \leftarrow EW_{\Omega}(c, y) \tag{4.25}$$
$$\omega \leftarrow (\lambda \parallel \psi) \tag{4.26}$$

Elde edilen Ω anahtarı, orijinal mesajın (m) ve y değerlerinin hash özetidir (4.21). Burada Ω anahtarı mesajla ilişkilendirilmiştir. Eğer istenilirse Ω anahtarı olarak random bir değer de seçilebilir. Karıştırma işleminde kullanılacak olan Ω anahtarı, yalnızca alıcı tarafından elde edilebilmesi için, alıcının açık anahtarı ile şifrelenir (4.22). Elde edilen λ değeri karşı tarafa gönderilecektir. Orijinal mesaj da karıştırma algoritması ile karıştırılarak, DEM'in saldırılara karşı güvenliği arttırılmıştır. (4.23) numaralı eşitlik isteğe bağlı olarak kullanılabilir. Karıştırma algoritması ile karıştırılan orijinal mesaj simetrik anahtar ile şifrelenerek DEM elde edilir (4.24). Daha sonra KEM ve DEM karıştırma algoritması (*EW*) ile karıştırılır (4.25). En son şifrelenmiş karıştırma anahtarı (λ) ve tek blok haline gelen KEM ve DEM (ψ) birleştirilip alıcıya gönderilir.

4.2.3. Şifrenin Çözülmesi



Şekil 4.4: Cascaded KEM-DEM sistemi şifreyi çözme blok diyagramı

Scrabbled KEM-DEM sistemi şifreyi çözme:

$$(\hat{\lambda},\hat{\psi}) \leftarrow \omega \tag{4.27}$$

$$\hat{\Omega} \leftarrow DQ^{asym}_{sk_{receiver}}(\hat{\lambda}) \tag{4.28}$$

$$(\hat{c},\hat{y}) \leftarrow DW_{\hat{\Omega}}(\hat{\psi}) \tag{4.29}$$

$$(\hat{x},\hat{h},\hat{pk}_{sender}) \leftarrow DQ^{asym}_{sk_{receiver}}(\hat{y}) \tag{4.30}$$

$$\hat{k}_{sym} \leftarrow EQ^{asym}_{\hat{pk}_{sender}}(\hat{x}) \tag{4.31}$$

$$(\hat{k}_{group}) \leftarrow \delta_{cons}(\hat{k}_{sym}) \tag{4.32}$$

$$\hat{\beta} \leftarrow D^{sym}_{\hat{k}_{group}}(\hat{c}) \tag{4.33}$$

$$\hat{m} \leftarrow DW_{\hat{\Omega}}(\hat{\beta}) \tag{4.34}$$

$$h \leftarrow H(\hat{m}) \tag{4.35}$$

Eğer $h = \hat{h}$ ise sonuç = m değilse sonuç = $\perp$

Şifreli bir mesaj alındığında yapılması gereken ilk işlem λ ve ψ değerlerinin elde edilmesidir. Bu işlem basamağı çok basittir, çünkü λ ve ψ değerleri yan yana getirilerek elde edilmiştir.

$\hat{\lambda}$ ve $\hat{\psi}$ değerleri (4.27) numaralı eşitlik ile birbirlerinden ayrılır ve $\hat{\lambda}$ değerinden $\hat{\Omega}$ değeri elde edilir (4.28). Ω anahtarı şifreleme işleminde alıcının açık anahtarı ile şifrelenmişti. Tekrar geri elde edebilmek için alıcının gizli anahtarı ile şifre çözülerek $\hat{\Omega}$ anahtarı elde edilir. Elde edilen $\hat{\Omega}$ anahtarı ile şifreli mesaj ($\hat{c}$) (DEM) ile anahtar kapsülü ($\hat{y}$) (KEM) birbirinden ayrılır (4.29). Sadece $\hat{\Omega}$ anahtarını doğru olarak elde edebilen kişi karıştırılmış olan $\hat{c}$ ile $\hat{y}$'yi birbirinden kolayca ayırabilir.

KEM'in içeriğini elde etmek için (4.30), alıcının özel anahtarı ($sk_{receiver}$) kullanılarak, anahtar kapsülü ($\hat{y}$) deşifre edilir. Buradan kimlik kontrol değeri ($\hat{x}$), hash özeti ($\hat{h}$) ve mesajı gönderen kişinin açık anahtarı ($\hat{p}k_{sender}$) elde edilir.

Eğer mesaj doğru kişi tarafından gönderilmiş ise, simetrik şifre çözme algoritmasında kullanılacak olan random anahtar (k_{sym}) doğru şekilde elde edilebilir (4.31). Şifreleme işleminde x değerinin elde edilmesi için, mesajı gönderen kişi kendi özel anahtarını (sk_{sender}) kullanmış ve k_{sym}'yi şifrelemişti. Anahtarın geri elde edilmesinde işlem tersine çevrilir ve pk_{sender} ile deşifre edilirse k_{sym}'nin orijinal hali elde edilir. Elde edilen k_{sym} doğru ise DEM'in şifresi çözüldüğünde elde edilecek hash özeti ile KEM'den elde edilen h değerleri aynı olacaktır. Böylelikle aynı anda hem veri bütünlüğü kontrolü hem de gönderen kişinin kimlik kontrolü yapılmış olacaktır.

(4.32) numaralı eşitlik ile şifreli mesajın (DEM'in) şifresinin çözülmesinde kullanılacak olan simetrik anahtar elde edilir ve bu simetrik anahtar ile $\hat{c}$ deşifre edilerek karıştırılmış mesaj ($\hat{\beta}$) elde edilir (4.33). Orijinal mesajı elde etmek için de

karıştırılmış olan mesajdan ($\widehat{\beta}$) (4.34) numaralı eşitlik kullanılarak mesajın orijinal hali $\widehat{m}$ elde edilir.

Veri bütünlüğü ve mesajın doğruluğunun kontrolünün yapılabilmesi için $\widehat{m}$'nin Hash özeti elde edilir (4.35). Eğer anahtar kapsülünden (KEM'den) elde edilen hash özeti $\widehat{h}$ ile $\widehat{m}$'den elde edilen hash özeti h aynı ise mesaj doğru olarak çözülmüştür ve sonuç = $\widehat{m}$ olur. Değilse sonuç = $\perp$ olur. Burada aynı anda mesajı gönderen kişinin kimlik kontrolü de dolaylı olarak yapılmış olmaktadır.

4.3. Combined KEM-DEM Sistemi

Verinin şifrelenmesi ile ilgili şu ana kadar anlatmış olduğumuz tüm sistemler, tek seferlik haberleşmede veya çok az süren bir veri iletişiminde etkinlikle kullanılabilir. Ancak aynı sistemleri, iki düğüm arasında, uzun süreli kurulacak olan bir iletişimde kullanmak, sürenin etkin ve efektif olarak kullanılmasında zorluklara neden olacaktır.

Scrabbled KEM-DEM, cascaded KEM-DEM, Tag-KEM/DEM [14] ve Fujisaki-Okamoto's KEM-DEM [15] sistemlerinde KEM'in hesaplanması için harcanan süre DEM'in hesaplanması için harcanan süreden çok daha fazladır. Ayrıca KEM'e dijital imzanın ve veri bütünlüğü kontrol değerlerinin eklenmesi hesaplanma süresini daha da arttıracaktır. Bu artış kısa süreli olarak kurulan iletişimlerde çok önemli olmayabilir. Ancak iletişim zamanı uzadıkça bu sistemlerdeki KEM'in hesaplanma süresi sistemlerin zayıf yönü durumuna gelecektir.

Burada sunulan sistem: combined KEM-DEM; anahtarı gizlemek için gereken zamanı kayda değer oranda azaltmakla kalmamakta, tüm mesaj trafiğinin bütünlüğünün kontrolünü de sağlamakta ve garanti altına almaktadır. Ayrıca oturum anahtarı (session key) kullanımını daha da geliştirip, aynı oturum içinde dahi her mesajın farklı bir anahtar ile şifrelenmesini sağlanmaktadır. Bu işlemler esnasında güvenlikten de en ufak bir taviz verilmemekte ve kimlik kontrolü de dahil tüm kontroller yapılmaktadır.

Bu yöntemde daha önce anlatılan Scrabbled KEM-DEM sistemi temel alınmıştır. Bu sisteme ait özellikler korunmakla birlikte KEM'in hesaplanma süresinin kısaltılması amaçlanmıştır. Kısaca bu yöntemde; KEM'in (y) ve gönderilecek olan mesajın orijinal halinin hash değeri hesaplanacak, bu değer alıcının açık anahtarı ile şifrelenecek, karşı tarafa simetrik anahtarla şifrelenmiş mesajla birlikte bu değer gönderilecektir.

4.3.1. Tanımlar

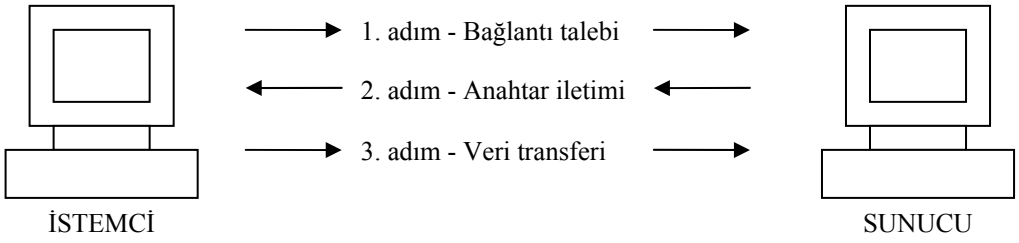
Ω : Şifrelenmiş mesajın karıştırılmasında kullanılan anahtar. Bu anahtar aynı zamanda, mesajı alan düğüm tarafından gönderilecek mesajın şifrelenmesinde kullanacak olan simetrik anahtarın elde edilmesinde de kullanılmaktadır.

Φ : Sayaç. Gönderilen mesajın kaçıncı mesaj olduğunu gösteren bir sayı.

λ : Ω ve Φ değerlerinin alıcının açık anahtarı ile şifrelenmiş hali

ψ : Simetrik anahtarla şifrelenmiş olan metnin karıştırma algoritması ile karıştırılmış hali

4.3.2. Algoritmanın İşleyişi ve Yapısı



Şekil 4.5: Combined KEM-DEM sisteminde iletişime başlama adımları

Yukarıdaki şekil 4.5'de görüldüğü gibi veri transferine üç adımda başlanılmaktadır.

Birinci adımda istemci tarafından sunucuya bağlantı talebi iletilmektedir. Yani bağlantı talebi sunucuya ω_1 değeri ile iletilmektedir. Algoritma aşağıdaki gibidir.

$$\omega_1 \leftarrow EW_{pk_{receiver}}(EQ^{asym}_{pk_{receiver}}(EQ^{asym}_{sk_{sender}}(k_{sym}), H(m_1), pk_{sender}, \Phi), E^{sym}_{k_{group}}(m_1)) \quad (4.36)$$

Bağlantı talebi adımında yapılan işlemler ayrıntılı olarak (4.37-4.43) numaralı eşitlikler ile verilmiştir.

Combined-KEM-DEM sistemi, 1. adım: Bağlantı talebi:

Anahtar üretimi:

$$k_{sym} \leftarrow \varphi_{sym}(string) \quad (4.37)$$
$$k_{group} \leftarrow \delta_{cons}(k_{sym}) \quad (4.38)$$

Şifreleme:

$$c \leftarrow E^{sym}_{k_{group}}(m_1) \quad (4.39)$$
$$h \leftarrow H(m_1) \quad (4.40)$$
$$x \leftarrow EQ^{asym}_{sk_{sender}}(k_{sym}) \quad (4.41)$$
$$y_1 \leftarrow EQ^{asym}_{pk_{receiver}}(x, h, pk_{sender}, \Phi) \quad (4.42)$$
$$\omega_1 \leftarrow EW_{pk_{receiver}}(c, y_1) \quad (4.43)$$

Burada şifrelenecek metin olarak kullanılacak m_1 değerinin içeriğinin ve büyüklüğünün bir önemi yoktur. Sadece istemcinin kimlik bilgisinin doğruluğunun kontrolünde kullanılacak olan bir parametredir. Bu sistemin çalışma mantığı daha önce anlatılan Scrabbled KEM-DEM sistemi ile birebir aynıdır. KEM'e (y_1) eklenen Φ değeri basit bir sayaçtır. İletilen mesajın sırasını, yani kaçıncı mesaj olduğunu gösteren bir değerdir. Şifreleme/şifreyi çözme çözülmesi işleminde hangi sistemin kullanılacağına Φ değerine göre karar verilmektedir. Eğer değeri 1 veya 2 ise kullanılacak olan şifreleme sistemi yukarıdaki gibidir. Şifreyi çözme sistemi de daha önce anlatılan scrabbled KEM-DEM sistemininkiyle aynıdır.

Anahtar dağıtımının yapıldığı ikinci adımda, iki nokta arasındaki veri transferinde kullanılacak olan oturum başlangıç anahtarı üzerinde anlaşılması gerekmektedir. Oturum başlangıç anahtarı istemciye sunucu tarafından m_2 değeri olarak bildirilmektedir. Yani m_2 değeri kullanılacak olan oturum başlangıç anahtarıdır. Bu

adımda ayrıca istemci, sunucunun kimlik doğrulamasını da yapmaktadır. Bu adımda da kullanılan algoritma (4.44-4.51) numaralı eşitlikler ile verilmiştir.

Combined-KEM-DEM sistemi, 2. adım: Oturum açma anahtarı dağıtımı:

Anahtar üretimi:

$$k_{sym} \leftarrow \varphi_{sym}(string) \tag{4.44}$$

$$k_{group} \leftarrow \delta_{cons}(k_{sym}) \tag{4.45}$$

Şifreleme:

$$c \leftarrow E^{sym}_{k_{group}}(m_2) \tag{4.46}$$

$$h \leftarrow H(m_2) \tag{4.47}$$

$$x \leftarrow EQ^{asym}_{sk_{sender}}(k_{sym}) \tag{4.48}$$

$$y_2 \leftarrow EQ^{asym}_{pk_{receiver}}(x, h, pk_{sender}, \Phi) \tag{4.49}$$

$$\omega_2 \leftarrow EW_{pk_{receiver}}(c, y_2) \tag{4.50}$$

Veri transferi adımı olan üçüncü adımda istemci, gelen mesajı kontrol etmiş ve karşıdaki kişinin kimliğinden emin olmuştur. Oturum başlangıç anahtarı bilgisi üzerinde anlaşıldığı için veri transferine başlanabilir.

Bu adımda kullanılacak olan sistem ilk iki adımdakinden farklı fakat onlardan bağımsız değildir. Veri transfer süresinin kısaltmasında kullanılacak olan Ω değeri, veriyi transfer eden noktanın en son gönderdiği mesajdaki y değerinden hesaplanacaktır. Burada amaç KEM'in büyüklüğünün ve hesaplanma süresinin kısaltılması ve tüm mesajları birbirlerine bağlayarak tüm trafiğin bütünlüğünün/kontrolünün sağlanmasıdır. Üçüncü adımda kullanılacak olan sistem (4.51) numaralı eşitlik ile verilmiştir.

$$\omega_n \leftarrow (EQ^{asym}_{pk_{receiver}}(H(m_n, y_{n-2}) \,\|\, \Phi_n) \;\|\; EW_{\Omega_n}(E^{sym}_{\delta_{cons}(\Omega_{n-1})}(m_n))) \tag{4.51}$$

Üçüncü adım ve daha sonrasında kullanılacak olan eşitlikler aşağıda verilmiştir.

Combined-KEM-DEM sistemi: Şifreleme, n. adım: Veri transferi:

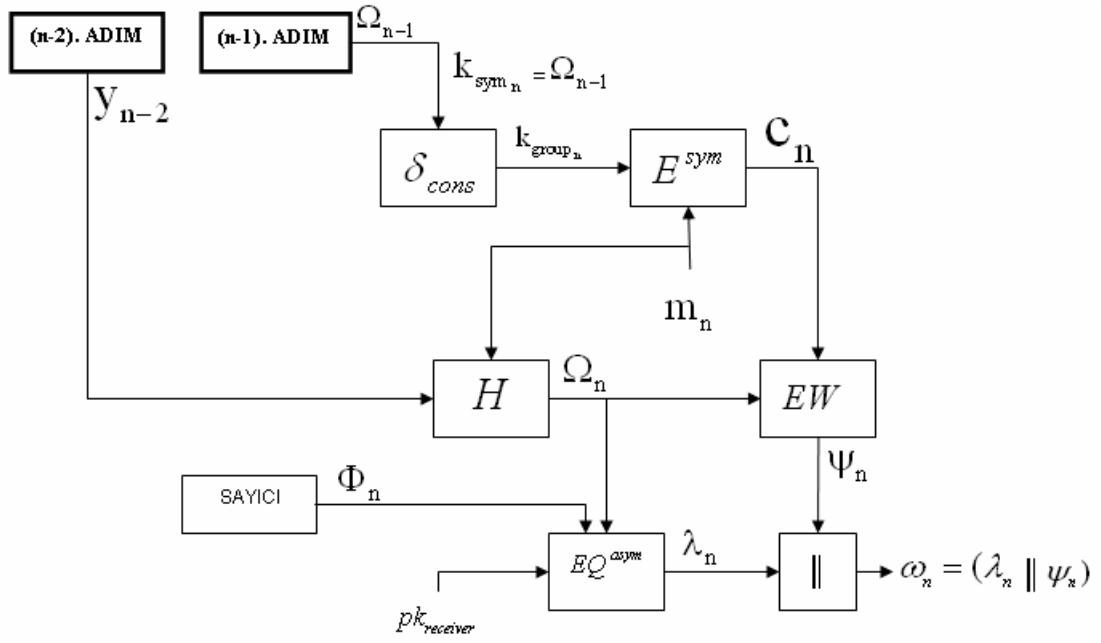
Anahtar üretimi:

$$\Omega_2 = m_2 \tag{4.52}$$
$$k_{sym_n} = \Omega_{n-1} \tag{4.53}$$
$$k_{group_n} \leftarrow \delta_{cons}(k_{sym_n}) \tag{4.54}$$

Şifreleme:

$$c_n \leftarrow E^{sym}_{k_{group_n}}(m_n) \tag{4.55}$$
$$\Omega_n \leftarrow H(m_n, y_{n-2}) \tag{4.56}$$
$$\psi_n \leftarrow EW_{\Omega_n}(c_n) \tag{4.57}$$
$$y_n \leftarrow (\Omega_n \parallel \Phi_n) \tag{4.58}$$
$$\lambda_n \leftarrow EQ^{asym}_{pk_{receiver}}(y_n) \tag{4.59}$$
$$\omega_n \leftarrow (\lambda_n \parallel \psi_n) \tag{4.60}$$

(4.52) numaralı eşitlik Ω_n için başlangıç değeridir ve sadece bir kez üçüncü adımda (n=3) kullanılmakta ve devam edem basamaklardaki veri iletimlerinde göz ardı edilmektedir. Daha sonraki veri transferi basamaklarında (n>3) kullanılacak Ω_{n-1} değeri, otomatik olarak bir sonraki veri transferini yapacak olan düğümün kullanacağı k_{sym_n} değerini belirtmektedir. (4.53) numaralı eşitlik gönderilen her mesajın ayrı bir anahtarla şifrelenmesini sağlamaktadır. (4.58) numaralı eşitlik ise üçüncü adımdan sonra (n>3), y değerinin hesaplanmasında kullanılmaktadır ve bu y değeri üçüncü adımdan sonra Ω'nin hesaplanmasında kullanılacaktır.



Şekil 4.6: Combined KEM-DEM sistemi şifreleme blok diyagramı: n. adım

Üçüncü adımdan sonra kullanılacak olan şifreleme/şifreyi çözme sistemleri, kendisinden önceki iki mesajla bağlantılıdır. n'nci mesajı gönderecek olan düğüm Ω_{n-1} değerini (n-1) nci mesajdan alır (Ω_{n-1} değeri en son gelen mesajdan elde edilmektedir) ve mesajını şifrelemek için simetrik anahtar k_{group_n}'i üretir. Üretilen bu simetrik anahtar ile orijinal metin (plain text) şifrelenir. Şifrelenen metin daha sonra Ω_n anahtarı kullanılarak karıştırma algoritması (EW) ile karıştırılır. Ω_n değeri m_n ve y_{n-2} değerleri kullanılarak üretilmekte, y_{n-2} değeri de gönderilen en son mesajdan alınmaktadır. Karıştırma işleminden elde edilen çıktı DEM olarak adlandırılmaktadır.

KEM (y_n)'i elde etmek için Ω_n değeri ile Φ_n değerleri birbirlerine eklenir. Birleştirme sonucunda elde edilen y_n değeri alıcının açık anahtarı ile şifrelenir. En son adımda, elde edilen KEM ve DEM birbirine eklenerek alıcıya gönderilir. Mesajı gönderen düğüm, Ω_n değeri kullanılarak üretilmiş simetrik anahtarla şifrelenmiş bir mesaj beklemeye başlar. Mesajı alan düğüm de kendi mesajını karşı tarafa göndermek için aynı işlem basamaklarını takip eder.

Eğer mesajı alan düğüm gerçek alıcı değilse, mesajı deşifre edemeyecek ve Ω değerini elde edemeyecektir. Çünkü KEM alıcının açık anahtarı ile şifrelendi ve sadece gizli anahtara sahip olan kişi KEM'in şifresini çözebilir. Dolayısıyla orijinal metnin şifrelenmesinde kullanılan simetrik anahtar k_{group}'u üretemez.

Bu sistemin sağladığı bir fayda da; karşı tarafa gönderilen mesajın içinde, orijinal metnin şifrelemesinde kullanılan anahtarın yer almamasıdır. Yani şifreli mesajı ele geçiren üçüncü kişi istese de metnin şifrelenmesinde kullanılan anahtara ulaşamayacaktır. Hatırlarsanız Ω_n değeri şifreli metne alıcı için eklenmişti. Alıcı Ω_n değerini alarak simetrik anahtarı üretir ve bu simetrik anahtarla mesajını şifreler. Alıcı da gönderdiği şifreli mesaja Ω_{n+1} değerini ekler. Ω_n ile Ω_{n+1} değerleri birbirlerinden farklıdırlar. Aslında tüm Ω değerleri birbirinden farklıdır. Böylece bu işlem bize her mesajın farklı bir simetrik anahtarla şifrelenmesini sağlar.

4.3.3. Şifreyi Çözme

Üçüncü adımdan sonra (n>3) kullanılacak olan deşifreleme işlem basamakları (4.61-4.67) eşitlikleri ile verilmiştir.

Combined-KEM-DEM sistemi: Deşifreleme algoritması: n. adım:

$$(\hat{\lambda}_n, \hat{\psi}_n) \leftarrow \omega_n \tag{4.61}$$

$$(\hat{\Omega}_n, \hat{\Phi}_n) \leftarrow DQ^{asym}_{sk_{receiver}}(\hat{\lambda}_n) \tag{4.62}$$

$$\hat{c}_n \leftarrow DW_{\hat{\Omega}_n}(\hat{\psi}_n) \tag{4.63}$$

$$k_{sym_n} = \Omega_{n-1} \tag{4.64}$$

$$k_{group_n} \leftarrow \delta_{cons}(k_{sym_n}) \tag{4.65}$$

$$\hat{m}_n \leftarrow D^{sym}_{k_{group_n}}(\hat{c}_n) \tag{4.66}$$

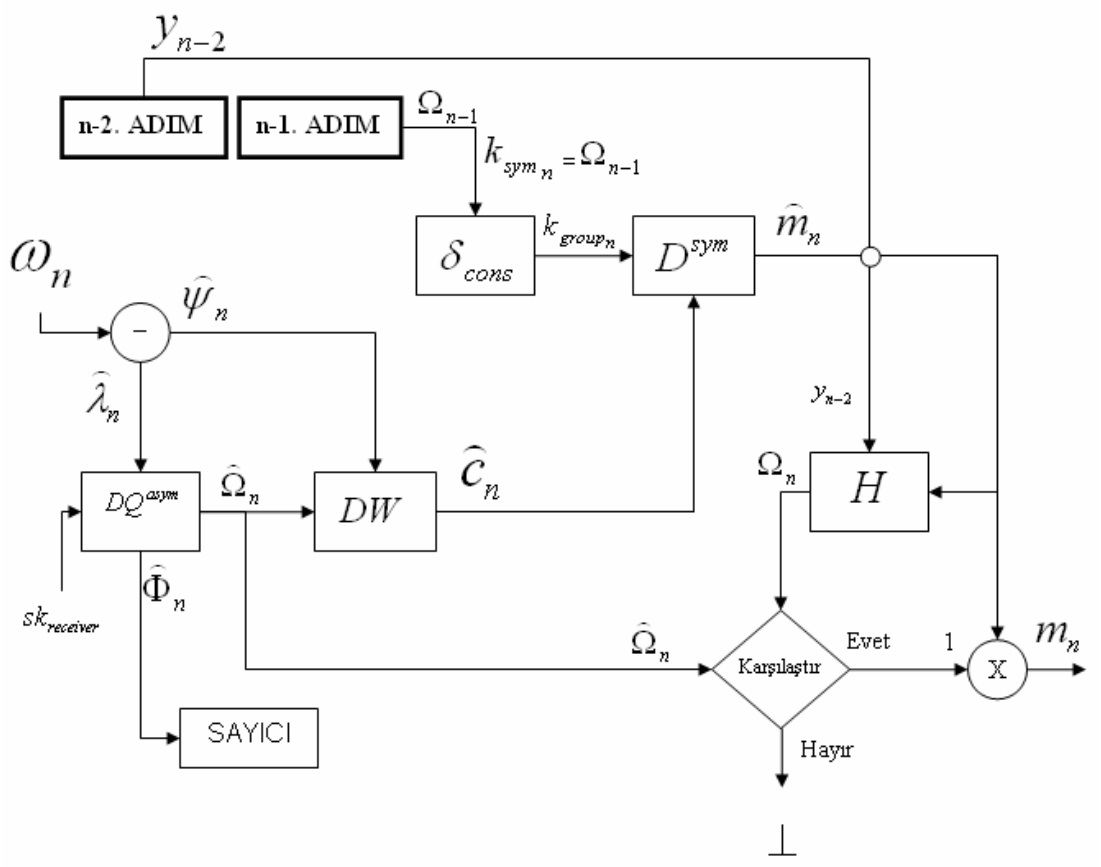
$$\Omega_n \leftarrow H(\hat{m}_n, y_{n-2}) \tag{4.67}$$

Eğer $\Omega_n = \hat{\Omega}_n$ ise $m_n = \hat{m}_n$ ve sonuç = $\hat{m}_n$ değilse sonuç = $\perp$

Düğüm tarafından n'nci mesaj alındığında yapılması gereken ilk işlem λ_n ve ψ_n değerlerinin elde edilmesidir. Bu işlem basamağı çok basittir, çünkü λ_n ve ψ_n

değerleri yan yana getirilerek elde edilmiştir. Ω_n ve Φ_n değerleri alıcının açık anahtarı kullanılarak şifrelenmişti. $\widehat{\Omega}_n$ ve $\widehat{\Phi}_n$ değerlerini tekrar elde etmek için $\widehat{\lambda}_n$ değeri alıcının gizli anahtarı ile deşifre edilir. $\widehat{\Phi}_n$ değeri iletişimde kaçıncı adımda olunduğunu göstermektedir.

Alıcı Ω_{n-1} değerine -karşı tarafa kendisi gönderdiği için- zaten sahiptir. Bu nedenle sadece doğru düğüm gelen mesajı çözüp Ω_{n-1} değerini elde edebilir ve bu değeri kullanarak mesajını şifreleyeceği simetrik anahtarı üretebilir. Aksi takdirde her hangi bir düğümün doğru simetrik anahtarı kullanarak mesajını şifrelemesi mümkün olmayacaktır. Eğer KEM'den elde edilen $\widehat{\Omega}_n$ ile $\widehat{m}_n$ ve y_{n-2} değerlerinden tekrar hesaplanan Ω_n değeri aynı ise, karşı tarafın doğru düğüm olduğunu, şifreleme/deşifreleme işleminin başarıyla yapıldığını kolaylıkla söyleyebiliriz. Ayrıca Ω değerinin hesaplanmasında orijinal mesajın hash özeti kullanıldığı için veri bütünlüğünün de sağlandığını söyleyebiliriz.



Şekil 4.7: Combined KEM-DEM sistemi şifreyi çözme blok diyagramı: n. adım

4.4. Tag-KEM/DEM Sistemi

Bu sistemde Tag-KEM, giriş olarak ekstradan bir tag değeri alır. Bir kişinin Tag-KEM kullanması DEM'in pasif ataklara karşı güvenli olması için yeterlidir. Tag-KEM/DEM yapısı klasik KEM-DEM yapılarına göre daha fazla güvenlik sunmaktadır. Aynı zamanda Tag-KEM/DEM yapısı diğer sistemlere de adapte edilebilmektedir.

KEM'in oluşturulmasında şifreleme fonksiyonu iki parçalı olarak çalışmaktadır. Birinci kısımda random bir anahtar seçilmekte, ikinci kısımda ise bu anahtar verilen bir tag değeri ile şifrelenmektedir. Bu nedenle sistem Tag-KEM/DEM olarak adlandırılmaktadır. KEM'in yapısı (4.68-4.71) eşitlikleri ile verilmiştir.

Tag-KEM/DEM sistemi: KEM'in yapısı:

$$(pk, sk) \leftarrow TKEM.Gen(1^{\lambda}) \tag{4.68}$$
$$(w, dk) \leftarrow TKEM.Key(pk) \tag{4.69}$$
$$\psi \leftarrow TKEM.Enc(w, \tau) \tag{4.70}$$
$$dk \leftarrow TKEM.Dec_{sk}(\psi, \tau) \tag{4.71}$$

Burada kullanılan τ değeri rasgele bir değerdir ve *TKEM.Dec*'e doğrudan verilmektedir. τ değerinin mesajı şifreleyenin kimlik bilgilerini taşıdığı kabul edilmektedir.

DEM simetrik şifreleme algoritmasından (*DEM.Enc, DEM.Dec*) oluşmaktadır. DEM'in yapısı (4.72 ve 4.73) eşitlikleri ile verilmiştir.

Tag-KEM/DEM sistemi: DEM'in yapısı:

$$\chi \leftarrow DEM.Enc_{dk}(m) \tag{4.72}$$
$$m \leftarrow DEM.Dec_{dk}(\chi) \tag{4.73}$$

KEM ve DEM'in genel yapısı anlatıldıktan sonra hybird yapı (4.74-4.81) eşitlikleri ile, blok diyagramı da şekil 4.8'de verilmiştir.

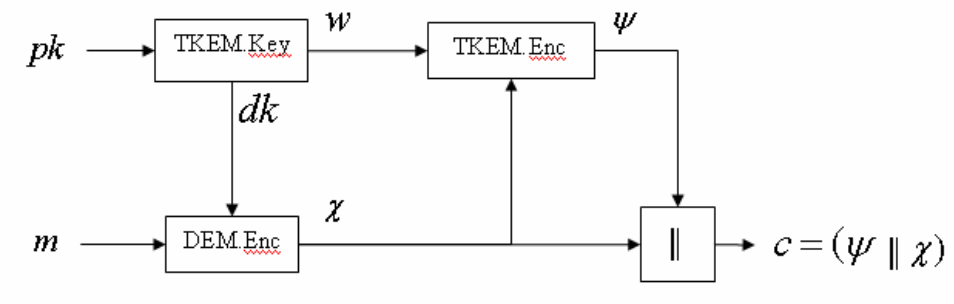
Tag-KEM/DEM sistemi: Hybrid yapı:

Şifreleme:

$$TKEM.Enc(w, \tau) \tag{4.74}$$
$$(pk, dk) \leftarrow w \tag{4.75}$$
$$\tau' = H(\tau) \tag{4.76}$$
$$\psi = PKE.Enc_{pk}(dk \parallel \tau') \tag{4.77}$$
Çıktı ψ

Deşifreleme:

$$TKEM.Dec_{sk}(\psi, \tau) \tag{4.78}$$
$$dk \parallel \tau' \leftarrow PKE.Dec(sk, \psi) \tag{4.79}$$
Eğer $\tau' = H(\tau)$ ise sonuç=dk (4.80)
Değilse sonuç= $\perp$ (4.81)



Şekil 4.8: Tag-KEM/DEM sistemi şifreleme blok diyagramı

4.5. Fujisaki-Okamoto's KEM-DEM Sistemi

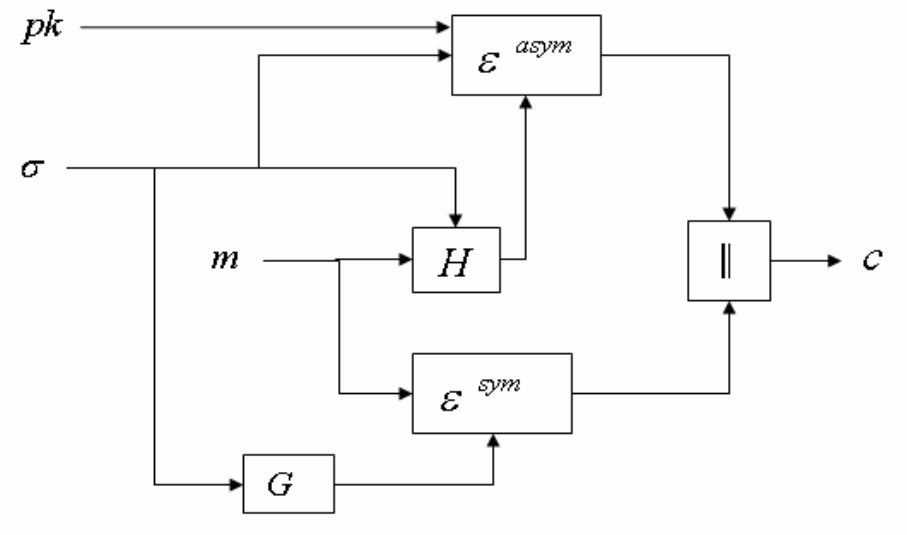
Bu sistemde zayıf bir yapıya sahip olan KEM-DEM sistemi ele alınmış ve bu yapı daha güçlü hale getirilmiştir. Burada asimetrik şifreleme sisteminin zayıf bir güvenliğe sahip olduğu ve üçüncü kişilerin de random bir metnin şifresini tamamen deşifre edemediği kabul edilmiştir. Simetrik şifrelemede de tüm olası mesajlar için, m1 ve m2, belirtilen bir mesaj uzayında, üçüncü kişilerin m1'in şifrelenmiş halini m2'nin şifrelenmiş halinden ayırt edemedikleri kabul edilmiştir (Burada üçüncü kişilere istenilen stringleri şifreleme veya deşifreleme hakkı yoktur). Bu bilgiler doğrultusunda yeni bir hybird algoritma geliştirilmiştir.

Fujisaki-Okamoto's KEM-DEM Sistemi ile herhangi bir metnin (m) şifrelenmesi (4.82) numaralı eşitlik ile, blok diyagramı şekil 4.8'de verilmiştir.

$$\varepsilon_{pk}^{hy}(m) = \varepsilon_{pk}^{asym}(\sigma; H(\sigma, m)) \,\|\, \varepsilon_{G(\sigma)}^{sym}(m) \tag{4.82}$$

Burada;

- σ Random bir değer
- $\varepsilon_a^{asym}(m, coin)$ Asimetrik şifreleme algoritması. Random bitlerden oluşan bir coin kullanılarak mesajın şifrelenmektedir.
- $\varepsilon_a^{sym}(m)$ Simetrik şifreleme algoritması. Şifreleme işleminde a string değeri kullanılarak mesaj şifrelenmektedir.
- G ve H Bu değerler hash fonksiyonlarını göstermektedir.

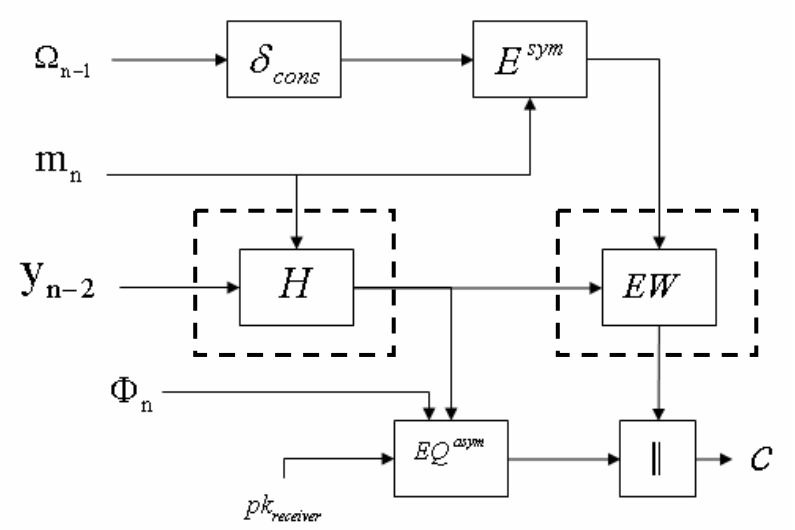


Şekil 4.9: Fujisaki-Okamoto's KEM-DEM sistemi şifreleme blok diyagramı

4.6. Performans Analizi

Konu 4.4 ve 4.5'te anlatılan iki metot, performans analizinde combined KEM-DEM sistemi ile karşılaştırmada kullanılmıştır. Karşılaştırma kolaylığı için combined KEM-DEM sisteminin blok diyagramı basit olarak şekil 4.9'da verilmiştir. Karşılaştırma yapılan metotlarla combined KEM-DEM arasındaki temel fark noktaları şekil üzerinde kesik çizgilerle gösterilmiştir. Hash fonksiyonu diğer metotlarda da bulunmaktadır, ancak girdi olarak kullanılan değerlerin kullanım amacı combined KEM-DEM metodunda biraz farklıdır. Hash fonksiyonu KEM'in hesaplanma süresinin kısaltılmasında kullanılan kilit noktadır. Blok diyagramın ayrıntılı anlatımı konu 4.3'de bulunmaktadır.

Combined KEM-DEM sisteminin Tag-KEM/DEM [14] ve Fujisaki-Okamoto's KEM-DEM [15] sistemleri ile yapılan performans analizi karşılaştırmasında, Dave Shapiro'nun JavaScript kullanarak hazırladığı RSA [19] kullanılmıştır. Simülasyon Pentium IV 2 Ghz, 512 MB RAM, Windows XP işletim sistemi olan bir PC'de çalıştırılmıştır. Kullanılan anahtar uzunlukları 256, 512 ve 1024 bit, asimetrik şifrelemede (KEM'in hesaplanmasında) kullanılacak olan simetrik anahtar ve hash özeti 128, 256 ve 512 bit olarak alınmıştır. KEM'in hesaplanma süreleri tablo 4.1'da gösterilmiştir.



Şekil 4.10: Combined KEM-DEM sistemi şifreleme blok diyagramı

Combined KEM-DEM sisteminin Fujisaki-Okamoto's KEM-DEM ve Tag-KEM/DEM sistemleri ile karşılaştırıldığında en büyük avantajı asimetrik şifreleme algoritmasına giren veri boyunun %50 oranında kısaltılmış olmasıdır. Asimetrik şifreleme algoritmasına giren verinin boyunun kısaltılmış olması :KEM'in hesaplanma süresinin de kısaltılmasını sağlamakta ve hazırlamış olduğumuz sistemin etkinliğini arttırmaktadır. Yapmış olduğumuz simülasyonun sonuçları, combined KEM-DEM sistemindeki KEM'in hesaplanma süresinin Fujisaki-Okamoto's KEM-DEM ve Tag-KEM/DEM sistemlerindekinden, özellikle küçük boyuttaki anahtar uzunluklarında, daha hızlı olduğunu göstermektedir. Anahtar uzunlukları arttıkça sistemlerin performansları benzerlik göstermeye başlamaktadır.

Tablo 4.1: Farklı RSA anahtar uzunlukları kullanılarak combined KEM-DEM, Tag-KEM/DEM ve Fujisaki-Okamoto's KEM-DEM sistemlerinde KEM'in hesaplanma süreleri

Simetrik Anahtar ve Hash Özeti Uzunlukları	**Hybrid Sistem**	**RSA Anahtar Uzunluğu**					
		256 bit		**512 bit**		**1024 bit**	
		Enc (ms)	**Dec (ms)**	**Enc (ms)**	**Dec (ms)**	**Enc (ms)**	**Dec (ms)**
128 bit	Fujisaki-Okamoto	250	6531	500	26102	969	92422
	Tag-KEM/DEM	250	6531	500	26102	969	92422
	Combined KEM-DEM	172	4078	282	13619	890	91560
256 bit	Fujisaki-Okamoto	453	10102	781	38975	1063	95123
	Tag-KEM/DEM	453	10102	781	38975	1063	95123
	Combined KEM-DEM	265	6540	515	26644	984	94875
512 bit	Fujisaki-Okamoto	843	19097	1421	60250	2093	189734
	Tag-KEM/DEM	843	19097	1421	60250	2093	189734
	Combined KEM-DEM	469	9856	859	37718	1781	189016

BÖLÜM 5: SONUÇLAR VE ÖNERİLER

Simetrik ve asimetrik şifreleme sistemleri [8] veri iletiminde mesajın gizliliğinin korunması ve başkaları tarafından okunamamasını sağlamak amacıyla kullanılmaktadır. Hybrid şifreleme sistemleri de simetrik ve asimetrik şifreleme algoritmalarının iyi yönlerini birleştirmektedir. Hybrid şifreleme sistemlerinde genel olarak kullanılan yapıda, veri öncelikle simetrik anahtarla şifrelenmekte ve verinin şifrelenmesinde kullanılan simetrik anahtar da asimetrik şifreleme algoritması ile şifrelenmektedir.

Hybrid şifreleme sistemlerinde yapılan son çalışmalarda [14, 15] Key Encapsulation Mechanism (KEM) ve Data Encapsulation Mechanism (DEM) yapılarının geliştirilmesine odaklanılmıştır. Bu model hybird şifreleme sistemini iki parçaya ayırmaktadır: KEM ve DEM. Hybrid şifrelemede kullanılan sistemlerden bir tanesi Tag-KEM/DEM [14] sistemidir. Bu sistemde KEM'in oluşturulması için öncelikle random bir anahtar üretilmekte, sonrasında bu anahtar dışarıdan verilen bir tag ile şifrelenmektedir. Diğer bir hybird şifreleme sistemi olan Fujisaki-Okamoto KEM-DEM [15] sistemi zayıf bir yapıya sahip olan simetrik ve asimetrik şifreleme sistemlerini güçlü bir yapıya dönüştürmektedir.

Kriptolojide en çok dikkat edilecek noktalardan biri, veri bütünlüğü (message integrity), kimlik kontrolü ve anahtar güvenliğini sağlarken harcanan zamandır. Tag-KEM/DEM [14] ve Fujisaki-Okamoto KEM-DEM [15] sistemlerinin ikisi de bu özellikleri sağlamaktadır. Ancak her iki sistemde bizim burada önerdiğimiz combined KEM-DEM sisteminden daha yavaş çalışmaktadır.

Bu çalışmada 3 yeni KEM-DEM metodu önerilmiştir. Üç metodda da Hybrid şifrelemede kullanılan KEM-DEM [8, 14 ,15] yapısı temel olarak alınmış ve bu yapı kullanılarak scrabbled KEM-DEM, cascaded KEM-DEM ve combined KEM-DEM sistemleri geliştirilmiştir. Mevcut KEM-DEM yapısı [8, 14 ,15] aynı anda

yapılabilecek iki yönlü bir saldırıya açıktır. Saldırgan aynı anda hem KEM üzerinde hem de DEM üzerinde çalışabilir. Yani saldırganın elinde şifrelenen metin ile ilgili, üzerinde çalışabileceği iki tane veri bulunmaktadır. Eğer KEM-DEM yapılarının birisinde bir zayıflık varsa tüm şifreleme süreci başarısız olacaktır. Saldırgan böylelikle tüm bilgileri elde edebilir. Bizim burada sunduğumuz üç sistemde de bu zayıflığı ortadan kaldıran bir KEM-DEM yapısına sahiptir. Bunu sağlamak için klasik KEM-DEM yapısının sonuna (4.8) numaralı eşitlik ile gösterilen karıştırma (scrabble) algoritması (EW) eklenmiştir.

Scrabbled KEM-DEM sisteminde KEM ve DEM birbirine alıcının açık anahtarı kullanılarak karıştırma algoritması (EW) ile karıştırılmaktadır. Karıştırma algoritmasında KEM ve DEM öncelikle permütasyona tabi tutularak karıştırılmakta, sonrasında da karıştırılan KEM-DEM simetrik anahtarla şifrelenmektedir. Bu karıştırma işlemini sadece doğru anahtara sahip olan alıcı çözebilir.

Bazı network yapılarında, kullanılan iletişim protokolüne bağlı olarak, alıcı kimliğinin tespit edilmesi çok zor olmayabilir. Bu networklarda hybrid şifrelemede kullanıldığımız son adım karıştırma algoritmasını daha etkin yapmak için, karıştırma işlemini alıcının açık anahtarı yerine, random bir anahtarla yapmak daha kullanışlı olacaktır. Daha sonra bu anahtar alıcının açık anahtarı ile şifrelenerek [9-11,16], karıştırılan KEM-DEM'e eklenmekte ve alıcıya gönderilmektedir. Böylelikle alıcının kimliği biliniyor olsa bile, hem KEM-DEM'in birbirinden ayrılması hem de tüm mesajın sadece alıcı tarafından çözülmesi garanti altına alınmış olur. Çünkü açık anahtar ile şifrelenen mesajı çözebilecek olan gizli anahtara sadece alıcı sahiptir. Bu şifreleme işlemi için kullanılan sistem cascaded KEM-DEM sistemidir.

Üçüncü sistem; combined KEM-DEM, uzun süreli kurulan bir iletişimde, tüm iletişimin bütünlüğünün kontrolünü sağlamakla kalmaz, aynı zamanda hybrid şifreleme süresini de kısaltır. Yapılan simülasyon ile combined KEM-DEM sisteminin Tag-KEM/DEM [14] ve Fujisaki-Okamoto KEM-DEM [15] sistemlerinden daha hızlı olduğu ispatlanmış ve sonuçlar tablo 4.1'de gösterilmiştir. Combined KEM-DEM sistemi, verinin hybrid olarak şifrelenme/deşifrelenme süresi kısaltmış olmakla birlikte güvenlikle ilgili tüm beklentilere de (alıcının kimlik tespiti,

verinin yolda değişip değişmediğinin kontrolü, anahtar güvenliği gibi) cevap vermeye devam eder. Klasik oturum anahtarı daha da (session key) geliştirilmiş ve aynı oturum içinde bile gönderilen her mesaj için farklı bir anahtarın kullanılması sağlanmıştır.

Combined KEM-DEM sistemi geliştirilmiş bir başlangıç oturum anahtarı kullanmakta ve her mesajı ayrı bir anahtarla şifrelemektedir. Ayrıca scrabbled KEM-DEM, cascaded KEM-DEM, Tag-KEM/DEM [14] ve Fujisaki-Okamoto KEM-DEM [15] sistemleri kadar güvenlidir ve mesajı gönderenin/alanın kimlik kontrolünü, mesaj bütünlük kontrolünü yapar ve anahtar güvenliğini sağlar. Combined KEM-DEM sisteminin temel avantajı şifreleme/şifreyi çözme zamanını kısaltmasıdır. RSA ile yapılan simülasyon da göstermiştir ki KEM'in hesaplanması için gereken zaman, Tag-KEM/DEM ve Fujisaki-Okamoto KEM-DEM sistemleri ile karşılaştırıldığında yaklaşık olarak %40 oranında azaltılmıştır.

Combined KEM-DEM metodu kolaylıkla diğer KEM-DEM metotlarına adapte olabilicek bir yapıya sahiptir. Bu adaptasyon diğer KEM-DEM algoritmalarının da hızlanmasını sağlayacaktır.

Gelecekte şifreleme sistemlerinin yapıları teknolojinin gelişmesine paralel olarak daha da karmaşık ve kompleks olurken kullanımları da tam tersi çok kolay olacak. İhtiyaçlar ve kullanıcı sayısı arttıkça yeni metotların geliştilmesi zorunlu olurken güvenlik için sadece bir metodun kullanımı yeterli olmayacak birkaç metodun veya tamamının birlikte kullanıldığı sistemlere doğru gidilecek. Belki de metodlar modüler yapıya kavuşacak ve kullanıcılar isteklerine göre bu metotları birleştirip kullanabilecekler.

Büyük sayıların çarpanlara ayrılmasının zorluğundan faydalanılarak geliştirilen asimetrik şifreleme algoritmaları günümüz için yeterli güvenliği sağlamaktadır. Ancak, büyük sayıların çarpanlara ayrılması aslında gelecekte hiç de zor olmayacak bir problem olarak görülebilir. Yeni yöntem ve sistem arayışlarına devam edilmeli ve teknolojiye bağımlı olmayan sistemler üzerinde araştırmalar yapılmalı, sistemler geliştirilmelidir.

KAYNAKLAR

[1] Bozkurt F., *Elektronik Güvenlik, Şifreleme Teknikleri ve Algoritması açık olan Şifreleme Teknikleri* [online], Dokuz Eylül Üniversitesi http://courses.cs.deu.edu.tr/cse428/assignment1/Ferhat Bozkurt_2001510051.doc **(Ziyaret tarihi: 15.02.2006)**

[2] *Bilişim Güvenliği* [online], Pro-G ve Oracle http://www.pro-g.com.tr/whitepapers/bilisim-guvenligi-v1.pdf **(Ziyaret tarihi: 15.02.2006)**

[3] Schneier, B., "Applied Cryptology, Second Edition: Protocols, Algorithms, and Source Code in C", ***Wiley Publishing***, (1996)

[4] Stalings, W., 1997, "Crytography and Network Security", Second Edition, Prentice Hall

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

[6] Brown, L. P., "Analysis of DES and the Design of the LOKI Encryption Scheme, for Doctor of Philosophy", Department of Computer Science, University College, ***University of new South Wales***, Australia, April 1991.

[7] Daemen, J., Rijmen, V., "The Design of Rijndael AES-The Advanced Encryption Standard Series: Information Security and Cryptography", ***Springer Verlag 2002***.

[8] Dent, A.W., *Hybrid Cryptology* [online] http://eprint.iacr.org/2004/210 **(Ziyaret tarihi: 15.07.2005)**

[9] Zheng, Y., *Authenticated Public Key Encryption Schemes using Universal Hashing* [online] http://grouper.ieee.org/groups/1363/P1363a/contributions/aes-uhf3.pdf **(Ziyaret tarihi: 15.02.2006)**

[10] Halevi, S., and Krawczyk, H., "Public-Key Cryptography and Password protocols". ***In 5th ACM Conference on Computer and Communication Security***. Nov 2000

[11] Bellare, M., and Rogaway, P., "Optimal Asymmetric Encryption-How to encrypt with RSA". ***Advances in Cryptology***-CRYPTO'94

[12] Yerlikaya, T., Buluş, E., "AES Finalistlerinin Karşılaştırılması - RSA Şifreleme Algoritmasının İncelenmesi ve Kriptanalizi", ***20. Türkiye Bilişim Kurultayı***, İstanbul-TÜRKİYE, 2003.

[13] Şahin, A., Buluş, E., Sakallı, M.T., "Modern Blok Şifreleme Algoritmalarının Gücünün İncelenmesi", ***II. Mühendislik Bilimleri Genç Araştırmacılar Kongresi***-MBGAK'2005, İstanbul-Türkiye, 2005.

[14] Abe, M., Gennaro, R., Kurusawa, K., "Tag-KEM/DEM: A New Framework for Hybrid Encryption". In R.Cramer, editor, ***Advance in Cryptology-Eurocrypt 2003, volume 3494 of Lecture Notes in Computer Science***. Springer-Verlag, 2003

[15] Fujisaka, E., Okamoto, T., "Secure Integration of Asymmetric and Symmetric Encryption Schemes". In W.Wiener, editor, ***Advances in Cryptology-CRYPTO'99, volume 1666 of Lecture Notes in Computer Science***.Springer-Verlag, 2001

[16] Bellare, M., Desai, A., Poitcheval, D., and Rogaway, P., "Relations of Security for Public-Key Encryption Schemes". ***Advances in Cryptology***-CRYPTO'98

[17] Galindo, D., Martin, S., Morillo, P., Villar, J. L., "Fujisaki-Okamoto IND-CCA Hybird Encryption Revisited"

[18] Bellare, M., Boldyreva, A., Palacio, A., "An Uninstantiable Random-Oracle-Model Scheme for a Hybrid-Encryption Problem". ***Advances in Cryptology-EUROCRYPT'04. Lecture Notes in Computer Science Vol***. Springer-Verlag, 2004.

[19] http://www.ohdave.com/rsa/ **(Ziyaret tarihi: 15.02.2006)**

ÖZGEÇMİŞ

1976 yılında Balıkesir'de doğdu. İlk, orta ve lise öğrenimini Balıkesir'de tamamladı. 1993 yılında girdiği Marmara Üniversitesi TEF Elektronik-Bilgisayar Öğretmenliği Kontrol ABD'dan 1998 yılında mezun oldu. Halen bir kamu kuruluşunda görev yapmakta olup evli ve bir çocuk babasıdır.