T.C.
SAKARYA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

ŞİFRELEME ALGORİTMALARININ PERFORMANS ANALİZİ

YÜKSEK LİSANS TEZİ

Bilg.Müh. Ümit GÜNDEN

Enstitü Anabilim Dalı	**:**	**BİLGİSAYAR VE BİLİŞİM MÜHENDİSLİĞİ**
Tez Danışmanı	**:**	**Doç. Dr. Nejat Yumuşak**

Eylül 2010

T.C.
SAKARYA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

ŞİFRELEME ALGORİTMALARININ PERFORMANS ANALİZİ

YÜKSEK LİSANS TEZİ

Bilg.Müh. Ümit GÜNDEN

Enstitü Anabilim Dalı : BİLGİSAYAR VE BİLİŞİM MÜHENDİSLİĞİ

Bu tez 13 /09 /2010 tarihinde aşağıdaki jüri tarafından Oybirliği ile kabul edilmiştir.

Doç.Dr. Nejat YUMUŞAK	Yrd.Doç.Dr. Ali GÜLBAĞ	Yrd.Doç.Dr. Fahri VATANSEVER
Jüri Başkanı	Üye	Üye

TEŞEKKÜR

Tezimi hazırlık aşamamda ve tüm hayatım boyunca gerek maddi gerek manevi hiçbir desteğini esirgemeyen anneme, babama ve kardeşime teşekkürlerimi sunarım.

Bu çalışmayı hazırlamam aşamasında yardım ve desteklerini esirgemeyen değerli danışmanım Doç. Dr. Nejat Yumuşak'a teşekkürlerimi bir borç bilirim.

Ümit GÜNDEN

İÇİNDEKİLER

SİMGELER VE KISALTMALAR LİSTESİ

AES	: Gelişmiş Şifreleme Standardı (Advanced Encryption Standard)
CBC	: Cipher Block Chaining Mode
CFB	: Cipher Feedback Mode
CPU	: Merkezi İşlem Birimi (Central Processing Unit)
DES	: Veri Şifreleme Standardı (Data Encryption Standard)
DSA	: Dijital İşaret Algoritması (Digital Signature Algorithm)
FIPS	: Federal Bilgi İşleme Standartları (Federal Information Processing Standarts)
HMAC	: Keyed-Hashing for Message Authentication
IBM	: Uluslararası İş Makineleri (International Business Machines)
IDEA	: Uluslararası Veri Şifreleme Algoritması (International Data Encryption Algorithm)
ISO	: Uluslararası Standartlık Örgütü (International Organization for Standardization)
MAC	: Mesaj Doğrulama Kodu (Message Authentication Code)
MB	: Mega Byte
MD5	: Mesaj Özet 5 (Message Digest 5)
MDC	: Modification Detection Code
MOSS	: Object Security Services
NIST	: Ulusal Standart ve Teknoloji Enstitüsü (National Institute of Standarts and Technology)
NSA	: Ulusal Güvenlik Teşkilatı (National Security Agency)
OBEB	: Ortak bölenlerin En Büyüğü
PEM	: Privacy Enhanced Mail
PGP	: Pretty Good Privacy
RAM	: Rastgele Erişimli Hafıza (Random Access Memory)

RC2	: Ron’s Code 2
RSA	: Rivest-Shamir-Adleman
SHA	: Güvenli Hash Algoritması (Secure Hash Algorithm)
SSL	: Güvenli Yuva Katmanı (Secure Socket Layer)
TEA	: Ufak Şifreleme Algoritması (Tiny Encryption Algorithm)

ŞEKİLLER LİSTESİ

TABLOLAR LİSTESİ

ÖZET

Anahtar kelimeler: Şifreleme algoritmaları, performans analizi, kriptoloji, bilgi güvenliği

Bilgi güveliği kapsamında değerlendirilen bütün kavramlar bu güvenliği sağlayacak şifreleme tekniklerini kullanırlar. Bilişim teknolojilerindeki gelişmelere paralel olarak bilgi güvenliği disiplinler arası bir konu olduğundan gün geçtikçe önemi artmaktadır. Bu çalışmada bilgi güvenliğini sağlama amaçlı olarak kullanılan simetrik ve asimetrik algoritmalardan en sık kullanılanları zaman karmaşıklığı, işlemci karmaşıklığı ve hafıza karmaşıklığı bakımından incelenmiş ve uygulamaları gerçekleştirilmiştir. Şifreleme algoritmaları performansları bakımından karşılaştırılmış ve performans sıralamaları yapılmıştır. Simetrik şifreleme algoritmalarından Blowfish, Twofish, IDEA, TEA, DES, AES, 3DES, RC2 şifreleme algoritmaları ve asimetrik şifreleme algoritmalarından RSA algoritması incelenmiştir.

Performans değerlerinin elde edilmesi amacı ile C#.Net 2008 kullanılarak bir uygulama tasarlanmış ve farklı donanım ve yazılım özelliklerine sahip bilgisayarlar üzerinde çalıştırılmıştır.

PERFORMANCE ANALYSIS OF ENCRYPTION ALGORITHMS

SUMMARY

Key Words: Encryption algorithms, performance analysis, cryptology, data security

The all concepts that can be valueable in the information security fields ,provide high safety by using encryption algorithms. In parallel with advance of information tecnologies, information security gets more importance day by day because of Its positon is an issue ,in the branch of information knowledge. This study would provide a fieldwork and implemetations of frequently used symmetrical and asymmetrical algoritms which are used to ensure a secure environment in information security by testing their processor complexity,time complexity and memory complexity. Encryption algorithms compared by their performances and put in order from high performance to low performance. Blowfish, Twofish, IDEA, TEA, DES, AES, 3DES, RC2 algorithms analyzed as symmetrical encryption algorithms and also RSA as asymmetrical encryption algorithm.

To obtain all perfomance values an application developed using C#.net 2008 programming language and tested on many computers that have different hardware and sofware specifications.

BÖLÜM 1. GİRİŞ

1.1. Giriş

Teknolojinin geliştiği ve hızlı bir şekilde gelişmeye devam ettiği günümüzde, internet ve bilgisayarlar hayatımızın vazgeçilmez birer unsurları haline gelmiştir. Bu gelişmeye paralel olarak ortaya çıkan güvenlik açıklarıda bir o kadar önem taşımaktadır. İnternet üzerinden yapılan satış (e-ticaret), bankacılık işlemleri, kredi kartı işlemleri gibi yüksek güvenlik gerektiren uygulamalarda en gelişmiş şifreleme metotlarını kullanmak zorunlu hale gelmiştir. Verilerin güvenilir bir biçimde iletilmesi ve elde edilmesi için kriptografi bilimi aracılığıyla çeşitli şifreleme, anahtarlama ve çözümleme algoritmaları sunulmaktadır.

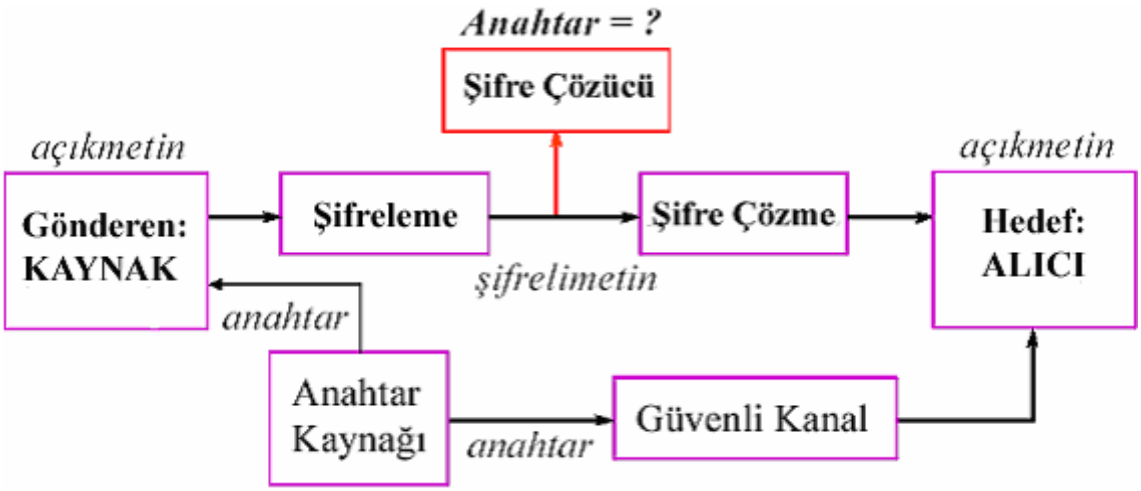
Şifreleme işlevinin en geleneksel kullanımı bilgilerin belirli kişilerden saklanması amacını güden kullanımdır. Bu fikir ister istemez, rakip ya da düşman (belirli kişiler ile kast edilen her ne ise) kavramına da referansta bulunur. Bu bağlamda bilginin gizlenmesi genellikle bir mesajın şifrelenmesini yani rakibin eline geçse (mesajı zorla ele geçirme, bir kopyasına ulaşma, mesajın gönderildiği kanalı dinleme) dahi rakip tarafından kolayca anlaşılamayacak, çözülemeyecek bir şekile dönüştürülmesi anlamına gelir. Gizleme yani dönüştürme işleminin sahip olması gereken bir diğer önemli özellik de, mesajı alması beklenen yetkili kişinin şifrelenmiş mesajı kolayca ve kısa sürede çözebilmesidir.

Şifrelemenin modern kullanımlarından biri de mesajı gönderenin doğrulanmasıdır (authentication). Yani mesajı alan kişi belli bir algoritmayı kullanarak, aldığı mesajın, düşündüğü kişi tarafından gönderildiğinden, mesajın kendisine gelirken değiştirilmediğinden emin olabilmelidir. Bir mesajın değiştirilip değiştirilmediğini anlama problemi çok önemli bir problemdir ve bu veri bütünlüğü (data integrity)

problemi olarak da bilinir. Söz konusu rakibin ortam gürültüsü olduğu durumda basit bir kontrol yeterli olabilirken daha tehlikeli ve riskli iletişim ortamlarında mesajla oynanıp oynanmadığını anlamak için daha karmaşık algoritmalar gerekmektedir.

1.2. Şifreleme

Bir ileti, düz metinden oluşmakta ve bazen açık metin de denmektedir. Bir iletinin içeriğini saklamak üzere yapılan gizleme işlemi de şifreleme olarak tanımlanmaktadır. Şifrelenmiş bir ileti şifreli metindir. Şifreli metni düz metne geri çevirme işlemi şifre çözmedir. Bu işlem Şekil 1.1'de gösterilmektedir.



Şekil 1.1. Düz metnin şifrelenmesi ve çözülmesi işlemi [1]

Sıradan şifreleme yöntemlerinde mesajı gönderen kişi ve alıcı kişi tek bir gizli anahtara sahiptir. Gönderen kişi bu anahtarla mesajı şifrelemekte, alıcı kişide aynı anahtarla şifrelenmiş mesajı açmaktadır. Bu yöntem gizli anahtar şifrelemesi ya da simetrik şifreleme sistemi olarak bilinmektedir. Burada en önemli problem gönderen kişinin ve alıcı kişinin bir gizli anahtar üzerinde anlaşmasını sağlamaktır. Eğer birbirinden farklı fiziksel ortamlarda bulunuyorlarsa, bir kuryeye ya da telefona güvenmek zorundadırlar. Fakat anahtarı herhangi bir şekilde elde eden bir kişi, o anahtar ile şifrelenmiş tüm mesajları okuyabilmektedir. Anahtarların üretimi,

aktarımı ve saklanması işlemine anahtar yönetimi adı verilir ve bu tüm şifreleme sistemlerinin dikkate alması gereken noktalardan biridir.

Elektronik iletişim, günümüzde kağıt üzerinde yazı yazarak yapılan her türlü iletişimin yerine geçmeye adaydır.Kişi/kuruluş/toplumların,özel/kamusal/resmi haberleşmelerini elektronik iletişim ağları üzerinden yapabilmeleri, açık ağlar üzerinden iletilen bilginin güvenliği ve güvenilirliğiyle yakından ilgilidir. Açık ağlardan gönderilen iletiler üçüncü şahıslar tarafından dinlenme ve değiştirilme tehdidi altındadırlar [1].

İleti güvenliğini sağlama bilimi şifre analizidir. Matematiğin hem şifrebilimi hem de şifreanalizini kapsayan dalı şifrelemedir ve şifrebilimciler tarafından icra edilirler [2].

Düz metin, herhangi bir metin dosyası, resim, ses ve görüntü dosyası yani herşey olabilir. Bilgisayar için düz metin sadece ikili bir veridir. Düz metin iletim veya depolama amacıyla tasarlanabilir. Her iki durumda da, düz metin şifrelenecek bir iletidir.

Şifreli metin de ikili bir veridir, bazen düz metin ile aynı boyutta, bazen de daha büyük olabilmektedir. Şifreleme işlemi şifreli metni üretmek üzere düz metin üzerinde gerçekleştirilmektedir.

Şifre çözme işlemi, özgün düz metni oluşturmak üzere şifreli metin üzerinden gerçekleştirilmektedir. Bir iletinin önce şifrelenmesi ardından da çözülmesi, özgün düz metnin eski haline getirilmesi işlemidir.

Şifreleme, okunur bir formattaki bilginin başkalarının okuyamayacağı bir formata dönüştürülmesi bilimidir, diğer bir deyişle de bilgi güvenliğini inceleyen bilim dalıdır. Güvenilirlik, veri bütünlüğü, kimlik doğrulama gibi bilgi güvenliği konularıyla ilgilenen ve matematiksel yöntemler üzerine yapılan çalışmalar olarak da tanımlanabilir. Şifreleme kelimesi, yunanca gizli anlamına gelen kryptos kelimesinden türemiştir. Bu süreçte, bilgi hedeflenen alıcı dışında bir başkasının

okuyamayacağı veya değiştiremeyeceği bir şekilde kodlanmaktadır. Mesajı orjinal şekline çevirme işlemine şifre çözme adı verilir. Bu işlemde bir şifre çözme anahtarı ile gerçekleştirilmektedir. Şifreleme ve şifre çözme işlemi, verinin okunabilir ve kodlanmış formatlara dönüştürülmesini sağlayan bir matematiksel formül veya algoritma ile bir anahtar gerektirmektedir. Anahtar, şifreli mesajı üretmek için kullanılan ham metin ile birleştirilmiş özel bir sayıdır. Şifreli mesaj iletim sırasında alıkonulsa bile mesajı çözme kabiliyetine sahip olmayan bir kişi tarafından kesinlikle okunamamaktadır [1].

Bir kriptosistemin gücü genellikle en zayıf noktasına eşittir. Dolayısıyla, algoritma seçiminden, anahtar dağıtımına ve kullanım koşullarına kadar hiç bir şey göz ardı edilmemelidir.

Modern kriptografinin ilgilendiği ana konuları şu şekilde sıralamak mümkündür;

Gizlilik - Bilgi istenmeyen kişiler tarafından anlaşılamaz.
Bütünlük - Bilgi saklanması veya iletilmesi sırasında, farkına varılmadan değiştirilemez.
Reddedilemezlik - Bilgiyi oluşturan ya da gönderen, daha sonra bilgiyi kendisinin oluşturduğunu veya gönderdiğini inkar edemez.
Kimlik belirleme - Gönderen ve alıcı, birbirlerinin kimliklerini doğrulayabilirler.

Gizlilik için bir kriptolama sisteminin sağlaması gereken koşullan şu şekilde sıralamak mümkündür;

- Şifre-kıncı şifrelenmiş veriden asıl veriye ulaşamamalıdır.
- Şifre-kıncı şifrelenmiş verinin bildiği bölümlerinden şifre çözme ile ilgili bilgiye ulaşamamalıdır.

Kimlik kontrolü ve mesaj koruması için ise bir kriptolama sisteminin sağlaması gereken koşullar şu şekilde sıralanabilir;

- Mesajı alanın mesajın kaynağım sorgulama olanağı olmalıdır.
- Mesajı alanın, mesajın iletim sırasında değiştirilmediğini kontrol edebilme olanağı

olmalıdır.

- Mesajı gönderen daha sonra mesajı gönderdiğini inkar edememelidir.
- Mesajı alan, aldığı mesajın bir tekrar mesajı olduğunu sezebilmelidir [3].

Bilginin güvenliği, başkası tarafından dinlenme, bilginin değiştirilmesi, kimlik taklidi gibi tehditlerin ortadan kaldırılması ile sağlanmaktadır ve bu amaçla kullanılan temel araç şifre analizidir.

Bilgi toplumlarına bakıldığında, teknolojinin de yardımıyla, milyonlarca insanın gözetiminin hükümetler tarafından yapılmakta olduğunu görmekteyiz. Şifre analizi sayısal dünyadaki kişilere bu özelliği sağlayan başlıca araçtır. Şifre analizi, artık sadece resmi özelliğe sahip olan veya uzak durulması gereken askeri bir araç değildir. Şifre analizini öğrenmek ve onun modern toplumlara sağladığı avantajlardan yararlanmak hem kişisel gizliliğimiz hem de elektronik dünyadaki güvenliğimiz için kaçınılmazdır.

Şifrelemenin çok kullanılan bazı tipleri özel donanımlar gerektirmektedir. Bununla beraber geçmiş yıllarda şifreleme tekniklerinde bir artış görülmüştür. Ordu ve aşırı önemli ticari uygulamalar ise şifrelemede donanım tekniklerini tercih etmişlerdir.

Donanım tekniklerinin seçilmesi için 3 ana sebep vardır;

- Hız
- Güvenlik
- Kurulum kolaylığı

Özel yapılandırılmış donanımlar her türlü yazılım tekniğinden daha hızlıdır. Bir bilgisayarda, yazılım şifreleme tekniğinin fiziksel güvenliği yoktur. Bu yüzden fark edilmeden algoritma ile oynamak mümkündür. Fakat donanım teknikleri bu probleme karşı daha iyi koruma sağlamaktadır. Açılması zor olan kutular saldırganların uzak tutulmasında donanım kullanmanın bir diğer yoludur. Ayrıca yongalar, algoritma anahtarlarını okuyacak saldırganlara karşı koruma sağlayacak şekilde tasarlanabilirler.

3 basit tipte şifreleme donanımı mevcuttur;
- Kendi başına çalışan şifreleme modülleri
- Haberleşme bağlantıları için adanılan şifreleme kutuları
- Kişisel bilgisayarlara takılan kartlar

Kendi başına çalışan şifreleme modülleri, tipik olarak şifre onaylama ve anahtar yönetimi gibi alanlarda banka ve benzeri kuruluşlar tarafından kullanılmaktadır. Adanmış şifreleme kutuları, genelde noktadan noktaya iki site arasındaki bilgi transferini güvenli hale getirmek için kullanılmaktadır. Kişisel bilgisayarlarda kullanılan kart şifreleyiciler, normalde harddisk`e gönderilen her şeyi şifrelemektedirler ve disket sürücü gibi diğer depolama ünitelerine gönderilen bilgileri de şifrelemek için ayarlanabilirler.

Yazılım teknikleriyle, genelde dosyalar şifrelenir ve şifreleri çözülür.. Bu teknikler şifreleme anahtarları ile çalışırlar. Bu yüzden şifreleme anahtarları, disk gibi kolay bulunabilecekleri yerde değil güvenli yerlerde saklanmalıdır [1].

1.3. Şifreleme Algoritmalarının Tarihçesi

Kriptoloji insanlığın yaratılışından çeşitli evreler geçirerek günümüze ulaşmıştır. Kısacası insanlık ne zaman var olmuşsa Kriptolojide o zaman var olmuştur. İlk başlarda insanlar sadece gizlilik kavramını gerçekleştirmek için bu bilime ihtiyaç duydu devir değiştikçe Kriptoloji bölümünü alanları da değişti, özellikle web teknolojisinin gelişmesiyle Kriptoloji' ye daha çok ihtiyaç duyuldu.

Askeri haberleşmelerinde kriptografi kullanan ilk ulus Ispartalılardır. MÖ 5. yüzyılda kendi geliştirdikleri bir cihazı tarihin ilk yer değiştirme sistemini uygulamak için kullanıyorlardı. Şifre anlamına gelen İngilizce "cipher" ve Fransızca "chiffre" kelimeleri bu dillere Arapçadan (cifr ya da cifir) geçmiştir. Avrupa'da şifre sistemlerinin ilk yaygın kullanım yeri Rönesans'a muhalefet eden Kilise'ydi.

MÖ.1900 dolaylarında bir Mısırlı katip yazdığı kitabelerde standart dışı hiyeroglif işaretleri kullandı.

MÖ.60-50 Julius Caesar (MÖ 100-44) normal alfabedeki harflerin yerini değiştirerek oluşturduğu şifreleme yöntemini devlet haberleşmesinde kullandı. Bu yöntem açık metindeki her harfin alfabede kendisinden 3 harf sonraki harfle değiştirilmesine dayanıyordu.

725-790 Abu Abd al-Rahman al-Khalil ibn Ahmad ibn Amr ibn Tammam al Farahidi al-Zadi al Yahmadi, kriptografi hakkında bir kitap yazdı. (Bu kitap kayıp durumdadır). Kitabı yazmasına ilham kaynağı olan, Bizans imparatoru için Yunanca yazılmış bir şifreli metni çözmesidir. Abu Abd al-Rahman, bu metni çözmek için ele geçirdiği şifreli mesajın başındaki açık metni tahmin etme yöntemini kullanmıştır.

1000 – 1200 Gazneliler den günümüze kalan bazı dokümanlarda şifreli metinlere rastlanmıştır. Bir tarihçinin dönemle ilgili yazdıklarına göre yüksek makamlardaki devlet görevlilerine yeni görev yerlerine giderken şahsa özel şifreleme bilgileri (belki şifreleme anahtarları) veriliyordu.

1586 Blaise de Vigenère(1523-1596) şifreleme hakkında bir kitap yazdı. İlk kez bu kitapta açık metin ve şifreli metin için otomatik anahtarlama yönteminden bahsedildi. Günümüzde bu yöntem hala DES CBC ve CFB kiplerinde kullanılmaktadır.

1623 Sir Francis Bacon, 5-bit ikili kodlamayla karakter tipi değişikliğine dayanan stenografı buldu.

1790 Thomas Jefferson, Strip Cipher makinesini geliştirdi. Bu makineyi temel alan M- 138-A, ABD donanmasının 2. Dünya savaşında da kullandı.

1917 Joseph Mauborgne ve Gilbert Vernam mükemmel şifreleme sistemi olan “one-time padi’i buldular.

1920 ve 1930'larda FBI içki kaçakçılarının haberleşmesini çözebilmek bir araştırma ofisi kurdu.

William Frederick Friedman, Riverbank Laboratuarlarını kurdu, ABD için kriptoanaliz yaptı, 2. Dünya savaşında Japonlar'ın Purple Machine şifreleme sistemini çözdü.

2. Dünya savaşında Almanlar Arthur Scherbius tarafından icat edilmiş olan Enigma makinesini kullandılar. Bu makine Alan Turing ve ekibi tarafından çözüldü.

1952'de ABD'de Resmi olarak Ulusal Güvenlik Teşkilatı (NSA) kuruldu. Ayrıca bilgi toplamak için internet, telefon görüşmeleri ve e-postaları da izlerler. Dünyadaki en büyük telefon görüşmeleri arşivine bu teşkilat sahiptir. İllegal olarak sivillerin telefon görüşmelerini kaydettikleri (ellerinde görüşmeleri kaydetmek için hiç bir yasal yetki, mahkeme emri olmaksızın) ve telekomünikasyon şirketlerinden telefon kayıtlarını istedikleri ortaya çıkmıştır)

1970 Horst Feistel (IBM) DES'in temelini oluşturan Lucifer algoritmasını geliştirdi.

1976 DES (Data Encryption Standard), ABD tarafından FIPS 46(Federal Information Processing Standard) standardı olarak açıklandı.

1976 Whitfield Diffie ve Martin Hellman Açık Anahtar sistemini anlattıkları makaleyi yayınladılar.

1978 Ronald L. Rivest, Adi Shamir ve Leonard M. Adleman: RSA algoritmasını buldular.

1985 Neal Koblitz ve Victor S.Miller ayrı yaptıkları çalışmalarda eliptik eğri kriptografik (ECC) sistemlerini tarif ettiler.

1990 Xuejia Lai ve James Massey: IDEA algoritmasını buldular.

1991 Phil Zimmerman: PGP sistemini geliştirdi ve yayınladı.

1995 SHA-1 (Secure Hash Algorithm) özet algortiması NIST tarafından standart olarak yayınlandı.

1997 ABD'nin NIST (National Institute of Standards and Technology) kurumu DES'in yerini alacak bir simetrik algoritma için yarışma açtı.

2001 NIST'in yarışmasını kazanan Belçikalı Joan Daemen ve Vincent Rijmen'e ait Rijndael algoritması, AES (Advanced Encryption Standard) adıyla standart haline getirildi.

2005 Çin'li bir ekip tarafından SHA-1 algoritmasının kırıldığı duyurulmuştu. Buna göre 2^80 gücündeki algoritma, 2^63′e kadar indirilmişti. Bunun üzerine Amerikan Hükümeti ve Microsoft, Sun gibi birçok büyük firma artık kullanmayacaklarını açıklamıştı.

2007 Artık her programcı algoritma geliştirebiliyor ve bu algoritmayı programlayabiliyor ve kırılması zor diyor lakin her seferin de kırılıyor.

2009 Kriptografi konusunda dünyanın ilk olimpiyatları Belçika'nın Katholieke Üniversitesinde 25-28 Şubat tarihleri arasında yapılacak olan ön eleme ile başladı. Amaç SH1 lerin kırılmasından sonra yeni bir güvenli algoritma oluşturmak.

Amerikan hükümeti Kriptoloji biliminin ne kadar önemli olduğunu farkında olduğundan dolayı NSA yı kurmuştur. Günümüz Türkiye'sinde ise Aselsan ve Tübitak çalışmaları başlı başına kripto çalışmalarıdır [4].

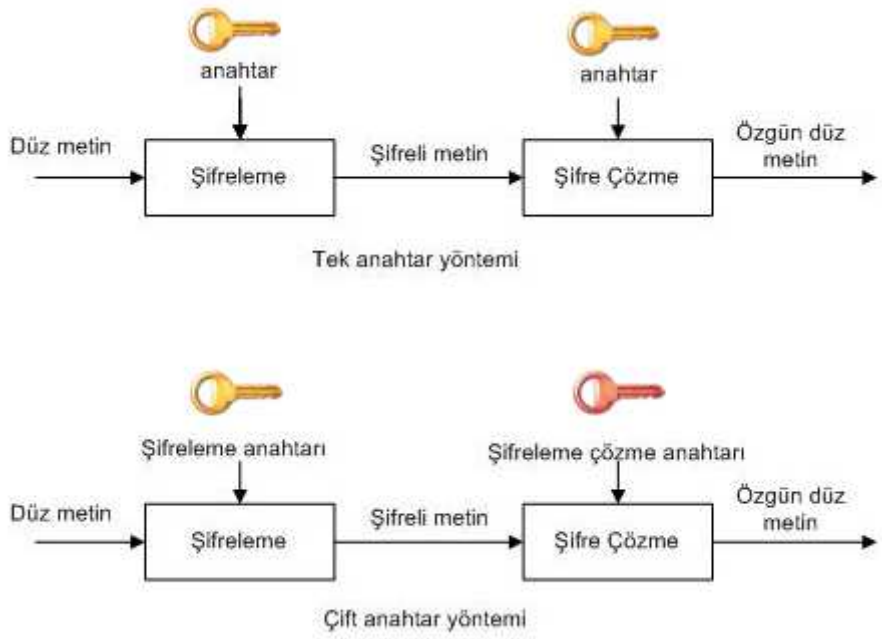
1.4. Algoritmalar ve Anahtarlar

Algoritmalar ve anahtarlar şifreleme ve şifre çözme için kullanılan matematiksel işlemlerdir. Eğer bir algoritmanın güvenliği bu algoritmanın çalışma biçimini gizlemeye dayalı ise, bu bir sınırlandırılmış algoritmadır. Sınırlandırılmış algoritmalar günümüzün şartlarına pek uymamaktadırlar ve bir gruba ait kullanıcılar bunları kullanamamaktadırlar. Çünkü gruptan bir kullanıcının her çıkışında geri kalan herkesin başka bir algoritmaya geçmesi gerekmektedir. İçlerinden birisi yanlışlıkla gizlenen anahtarı açığa vurduğunda, diğer herkesin algoritmalarını değiştirmeleri gerekmektedir. Daha da kötüsü, sınırlandırılmış algoritmalar kalite kontrolüne ve standartlaşmasına olanak tanımamaktadır. Her bir grup kullanıcının kendisine ait bir algoritması olmalıdır. Bu tür bir grup rafta hazır satılan şifre çözüm anahtarının yazılım veya donanım ürünlerini kullanamazlar çünkü davetsiz bir misafir aynı ürünü alıp algoritmayı öğrenebilmektedir. Kendi algoritmalarını ve gerçekleştirimlerini kendileri yazmaları gerekmektedir [1,5].

Bu ciddi zorluklara rağmen, sınırlandırılmış algoritmalar düşük güvenlik gerektiren uygulamalarda yoğun olarak kullanılmaktadır. Kullanıcılar sistemlerinde bulunan güvenlik sorunlarının ya farkında değildir ya da bunları önemsememektedirler.

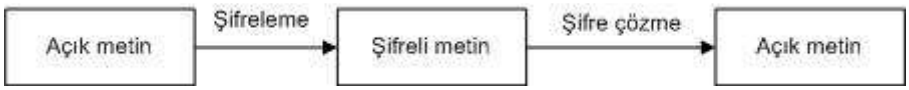
Günümüz şifre analizi, bu sorunu bir anahtar ile çözmektedir. Bu anahtar çok çeşitli değerler alabilen herhangi bir anahtar olabilir. Anahtarın alabileceği olası değerler genişliğine anahtar uzayı denir. Hem şifreleme hemde şifre çözme işlemleri bu anahtar uzayını kullanmaktadırlar.

Kimi algoritmalar farklı bir şifreleme ve şifre çözme anahtarı kullanırlar. Yani, şifreleme anahtarı buna karşılık gelen şifre çözme anahtarından farklıdır. Şekil 1.2’te tek anahtar ve çift anahtar kullanımları görülmektedir [2].



Şekil 1.2. Tek anahtar ve iki farklı anahtar ile şifreleme ve şifre çözme

Bu algoritmalardaki bütün güvenlik işlemleri, anahtar veya anahtarlara dayalıdır ve hiçbiri algoritmanın ayrıntılarında yer almamaktadır. Bu, algoritmanın yayınlanabildiği ve incelenebildiği anlamına gelmektedir. Bir davetsiz misafirin sizin algoritmanızı bilmesi önemli değildir, sizin özel anahtarınızı bilmedikçe iletilerinizi okuyamamaktadır. Geleneksel şifreleme yöntemi aşağıdaki şekilde gösterildiği gibidir.

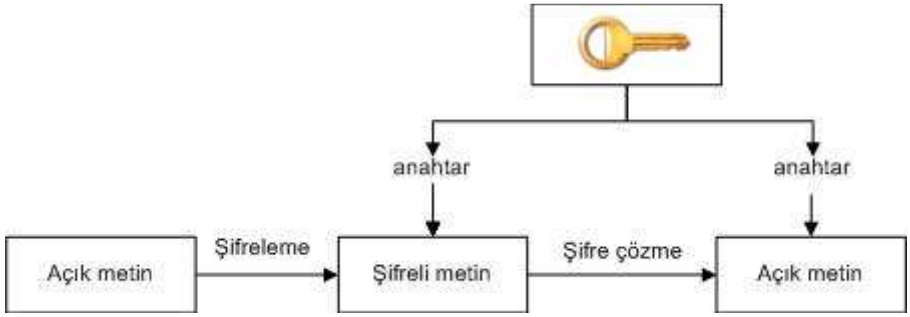


Şekil 1.3. Geleneksel şifreleme

Bir şifre sistemi, algoritmalardan, olası bütün düz metinlerden, şifreli metinlerden ve anahtarlardan oluşmaktadır. Anahtar yapısı bakımından, şifreleme algoritmaları iki grupta incelenirler.

1.4.1. Simetrik algoritmalar

Gizli anahtarlı şifreleme, simetrik şifreleme ya da tek anahtarlı şifreleme olarak da adlandırılır. Tek bir anahtarın hem şifreleme hem de şifre çözme amacıyla kullanıldığı daha geleneksel bir yöntemdir. Kimi zaman geleneksel algoritmalar olarak da adlandırılan simetrik algoritmaların çoğunda şifreleme anahtarı ile şifre çözme anahtarı aynıdır. Tek anahtarlı, gizli anahtarlı veya özel anahtarlı algoritmalar da denilen bu algoritmalar ile güvenli bir şekilde iletişim kurmadan önce gönderici ve alıcı ve kişilerin özel anahtar olarak adlandırılan bir anahtar üzerinde uzlaşmaları gerekmektedir. Bir simetrik algoritmanın güvenliği anahtarda yatmaktadır. Anahtarın ilan edilmesini isteyen herkesin iletileri şifreyebileceği ve şifreleri çözebileceği anlamına gelmektedir. İletişimin gizli kalması için anahtarın da gizli kalması gereklidir.Simetrik şifreleme algoritmaları aşağıdaki şekilde ki gibidir [1,6].



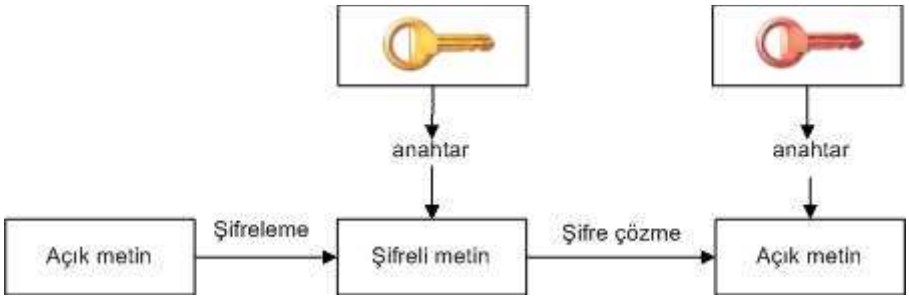
Şekil 1.4. Simetrik Şifreleme

Gizli anahtarlı şifrelemede temel problem, gönderici ve alıcı kişilerin, anahtarın üçüncü bir kişinin eline geçmesini engelleyerek ortak bir anahtar üzerinde anlaşmalarıdır. Bu durum, iki tarafın dinlenme korkusu duymadan iletişim kurmasını sağlayacak bir yöntem gerektirmektedir. Ancak, gizli anahtarlı yapıların avantajı, açık anahtarlı yapılara göre daha hızlı olmalarıdır. Asimetrik algoritmalar farklı bir şifreleme ve şifre çözme anahtarı kullanırlar. Yani, şifreleme anahtarı buna karşılık gelen şifre çözme anahtarından farklıdır. Şekil 1.2'de tek anahtar ve çift anahtar kullanımları görülmektedir [2].

Simetrik algoritmalar iki kategoriye bölünebilir. Kimileri düz metin üzerinden zaman başına belli bir anda tek bir bit veya bazen sekiz bit olarak çalışırlar bunlara akıcı algoritmalar denir. Diğerleri düz metin üzerinden bit grupları halinde çalışırlar. Bit gruplarına blok denir, algoritmalara da blok algoritmalar denir. Modern bilgisayar algoritmaları için, tipik bir blok büyüklüğü 64 bittir ve incelemelere engel olacak kadar büyük ve çalışabilecek kadar ufak bir büyüklüktür [2].

1.4.2. Asimetrik algoritmalar

Gizli anahtarlı yapılarda, genelde güvenli anahtar yönetiminin sağlanmasında problemler yaşanmaktadır. Anahtar yönetimi problemini çözmek amacıyla, Whitfield Diffie ve Martin Hellman, 1976 yılında açık anahtarlı yapıları geliştirmişlerdir. Açık anahtarlı şifrelemenin iki temel kullanımı vardır bunlar şifreleme ve sayısal imzadır. Bu sistemde her kişi biri açık, diğeri gizli olmak üzere bir çift anahtar edinmektedir. Açık anahtar herkesin erişimine açıktır, gizli anahtar ise sadece sahibinin erişebileceği şekilde saklanmaktadır. Burada gönderen ve alıcı kişilerin aynı gizli bilgiyi tutma durumları ortadan kalkmıştır. Bütün iletişim sadece açık anahtarları gerektirmektedir ve gizli anahtarlar ne iletilmekte ne de paylaşılmaktadır. Bu sistemde, kaygı duyulacak tek nokta açık anahtar ve anahtar sahibinin güvenilir ve doğru şekilde örneğin, güvenilir bir dizinde eşleştirilmesidir. Açık anahtarı kullanarak herhangi bir kişi şifreli mesaj gönderebilmektedir, ancak gönderilen şifreli mesajı sadece kullanılan açık anahtarın eşi olan gizli anahtar açabilmektedir. Bundan başka, açık anahtarlı şifreleme sadece gizliliği sağlamak amacı ile değil, kimlik denetimi diğer bir deyişle sayısal imza ve daha birçok teknik için kullanılmaktadır. Asimetrik şifreleme Şekil 1.5’te gösterildiği gibidir.



Şekil 1.5. Asimetrik şifreleme

Açık anahtarlı sistemlerde, her zaman gizli anahtarın açık anahtarla matematiksel bir bağlantısı vardır. Bu sebeple açık anahtarlı bir sisteme, açık anahtarı kullanarak saldırmak her zaman mümkündür. Buna karşı, açık anahtardan gizli anahtarı elde etme işlemi mümkün olduğu kadar zorlaştırılmaktadır. Bu anahtarları oluşturmak için çözülememiş matematik problemler kullanıldığından, açık anahtarı kullanarak gizli anahtarı elde etme işlemininde mümkün olmadığı kabul edilmektedir [2].

1.4.3. Simetrik ve asimetrik şifrelemenin birbirlerine göre avantaj ve dezavantajları

Açık anahtarlı şifrelemenin öncelikli avantajı, gizli anahtarın herhangi bir şekilde taşınması gibi bir durum söz konusu olmadığından, daha fazla güvenlik sağlamasıdır. Bunun aksine gizli anahtarlı yapılarda, şifrelemede ve şifre çözmede kullanılan aynı anahtar olduğundan, gizli anahtarın el ile ya da iletişim kanalları üzerinden iletilmesi söz konusudur. Bu da gizli anahtarın istenmeyen kişilerce elde edilme olasılığını doğurmaktadır.

Açık anahtarlı yapıların diğer bir önemli avantajı reddedilemez sayısal imzalar oluşturabilmesidir. Gizli anahtarlı yapılar kullanılarak yapılan kimlik denetiminde gizli bir bilginin paylaşılması ve bazı durumlarda üçüncü bir kişiye güven duyulması gerekliliği vardır. Bu durumda taraflardan biri, anahtarın diğerlerince kötü niyetle kullanıldığını iddia edebilmektedir. Ancak açık anahtarlı yapılarda herkes kendi

anahtarından sorumlu olduğu için böyle bir durum söz konusu değildir. Bu özelliğe reddedilemezlik denmektedir.

Kimi algoritmalar farklı bir şifreleme ve şifre çözme anahtarı kullanırlar. Yani, şifreleme anahtarı buna karşılık gelen şifre çözme anahtarından farklıdır. Şekil 1.4'te tek anahtar ve çift anahtar kullanımları görülmektedir [2].

Açık anahtarlı yapıları kullanmanın bir dezavantajı da şifreleme hızıdır. Çoğu gizli anahtarlı yapı açık anahtarlı yapılara göre daha hızlıdır. En güvenli ve en hızlı yöntem iki yapıyı birlikte kullanmaktır.

Açık anahtarlı şifrelemede, onay kurumuna yapılan başarılı bir saldırı sonucu, herhangi bir kullanıcının açık anahtarı yerine istenilen açık anahtar koyularak, bu kullanıcıya gönderilen mesajlar elde edilebilmekte, mesajlar değiştirilerek kullanıcıya, kullanıcının kendi açık anahtarıyla şifrelenerek gönderilebilmektedir.

Bazı durumlarda açık anahtarlı yapılar gereksizdir, gizli anahtarlı şifreleme tek başına yeterlidir. Örneğin gönderici ve alıcı kişiler yüz yüze görüşerek anahtar üzerinde anlaşabilirler. Bütün anahtarları bilen ve yöneten bir otorite bulunduğu durumlarda, açık anahtarlı şifreleme önemini yitirmektedir. Ancak kullanıcı sayısı arttığında bu da problem olabilmektedir.

Tek kullanıcının bulunduğu bir ortamda açık anahtarlı yapılar çok anlamlı değildir. Örneğin kişisel dosyalarınızı şifreli saklamak isterseniz, istediğiniz herhangi bir gizli anahtar algoritmasıyla kendi kişisel şifrenizi anahtar olarak kullanarak şifreleme yapabilirsiniz. Genel olarak açık anahtarlı yapılar, çok kullanıcılı açık ortamlar için idealdir.

Açık anahtarlı yapılar, gizli anahtarlı yapıların yerine geçmeye aday değildirler, daha çok onları daha güvenli hale getirecek tamamlayıcı unsurlardır. Örneğin, gizli anahtarları açık ağlar üzerinden taşımak için açık anahtarlı şifreleme kullanılır [1].

1.5. Şifreleme Algoritmalarının Performans Kriterleri

Bir şifreleme algoritmasının performansı;

- Kırılabilme süresinin uzunluğu.
- Şifreleme ve çözme işlemlerine harcanan zaman (Zaman Karmaşıklığı).
- Şifreleme ve çözme işleminde ihtiyaç duyulan bellek miktarı (Bellek Karmaşıklığı).
- Bu algoritmaya dayalı şifreleme uygulamalarının esnekliği.
- Bu uygulamaların dağıtımındaki kolaylık ya da algoritmaların standart hale getirilebilmesi.
- Algoritmanın kurulacak sisteme uygunluğu

kriterlerine göre belirlenir [7,8].

Bu çalışmada şifreleme algoritmaları performansları bakımından karşılaştırılmış ve performans sıralamaları yapılmıştır. Şifreleme algoritmaları karşılaştırılırken şifreleme ve şifre çözme işlemlerine harcanan zaman, hafıza ve işlemci kullanımları kriter olarak alınmıştır. Konu ile ilgili olarak bir uygulama tasarlanmış ve test sonuçlarının elde edilmesinde bu uygulamadan faydalanılmıştır. Uygulama farklı donanım ve yazılım özelliklerine sahip bilgisayarlar üzerinde çalıştırılmıştır.

1.6. Konu ile ilgili çalışmalar

Bu bölümde şifreleme algoritmaları ve analizleri konusunda yapılan çalışmalar incelenmiştir. Bu çalışmalar çalışmamızda bir başlangıç noktası oluşturmuştur.

ERHAN, 1993 yılında İstanbul Teknik Üniversitesi'nde yaptığı "RSA Algoritmasını Kullanan Şifreleme / Deşifreleme Yazılımının Tasarımı" adlı çalışmasında, RSA algoritmasının yapısını incelemiştir ve bu algoritmayı kullanarak kişisel bilgisayarlarda dosya güvenliğini sağlayan bir yazılım tasarlamıştır [9].

YILDIRIM, 1995 yılında İTÜ'de yaptığı "DES ve Benzer Şifreleme Sistemlerinin Diferansiyel Kripto Analizi" adlı çalışmasında, DES gibi yinelemeli sistemlerin diferansiyel kripto analizlerini incelemiştir. İnceleme yapılmadan önce DES ve benzeri algoritmaların yapısını anlatmıştır. Ardından bu algoritmalar üzerinde diferansiyel kripto analiz yönteminin uygulanmasını bir yazılım ile göstermiştir. Çalışmada, diferansiyel kripto analiz yönteminin tam. uygulanabilmesi için bol miktarda veri olması gerektiği anlaşılrnıştır [10].

ÜLGER, 1999 yılında Marmara Üniversitesi'nde yaptığı "Holotransformasyon Metodu ile Veri Şifreleme" adlı çalışmasında ilk olarak şifrelemenin tarihçesini ve şifreleme yöntemlerini incelemiştir. Ardından Holotransformasyon metodunu inceleyerek zayıf yönlerini ortaya koymaya çalışmıştır. Bu zayıf yönlerini gidermek amacıyla yeni teknikler geliştirmiştir. Bu yeni teknikleri bir yazılım uygulaması ile göstermiştir. Çalışmanın sonunda geliştirilen program kullanılarak; çeşitli dosya türleri şifrelenerek elde edilen iyileşmeyle eski yöntem karşılaştırılmıştır [11].

ANDİÇ, 2002 yılında Marmara Üniversitesi'nde yaptığı "Bilgisayar Haberleşmesinde Şifreleme Yazılımıyla Güvenliğin Sağlanması" adlı çalışmasında, ilk olarak şifrelemenin tarihçesini ve şifreleme algoritmaları çeşitlerini ele almıştır. Ardından AES algoritmasının yapısını anlatmıştır. Çalışmanın sonunda harici modemlerle kullanılabilen bir şifreli iletişim protokolü yazmıştır. Protokolün çalışmasını bir test programıyla göstermiştir [12].

ÇALIŞKAN, 2004 yılında Marmara Üniversitesi'nde yaptığı "Şifreleme Algoritmalarının Performans-Kripto Analizleri ve Eğitimde Kullanılması" adlı çalışmasında, şifreleme algoritmalarının sınıflandırılması ve analizleri yapılmıştır ve şifreleme algoritmalarını eğitiminde kullanılmak üzere bir eğitim simülatörü tasarlanmıştır [13].

ERKOÇ, 2004 yılında Sakarya Üniversitesi'nde yaptığı "Kriptoloji ve Bilgi Güvenliği" adlı çalışmasında, açık anahtarlı RSA (Rivest-Shamir-Adleman) algoritmasının yazılım uygulaması geliştirilmiştir [3].

DERELİOĞLU, 2005 yılında Yıldız Teknik Üniversitesi’nde yaptığı "Değişken Uzunluklu RSA Şifreleme Sistemlerinin Tasarımı ve Gerçeklenmesi" adlı çalışmasında, RSA sisteminin alternatif matematiksel yöntemleri ve bu yöntemlerin literatür çalışmaları doğrultusunda en az donanımsal yapıyla nasıl gerçeklenebilecekleri araştırılmıştır [14].

ORDU, 2006 yılında İstanbul Teknik Üniversitesi’nde yaptığı "AES Algoritmasının FPGA Üzerinde Gerçeklenmesi ve Yan Kanal Analizi Saldırılarına Karşı Güçlendirilmesi" adlı çalışmasında, AES simetrik şifreleme algoritmasının yapısını incelemiştir [15].

KAYIŞ, 2006 yılında İstanbul Teknik Üniversitesi’nde yaptığı "AES Uygulamas’nın FPGA Gerçeklemelerine Karşı Güç Analizi Saldırısı" adlı çalışmasında, AES simetrik şifreleme algoritmasının yapısını incelemiştir [16].

YERLİKAYA, 2006 yılında Trakya Üniversitesi’nde yaptığı "Şifreleme Algoritmalarının Analizi" adlı çalışmasında, simetrik ve asimetrik şifreleme algoritmalarının yapıları incelenmiş ve karşılaştırmaları yapılmıştır [7].

BULUŞ, 2006 yılında Trakya Üniversitesi’nde yaptığı "Temel Şifreleme Algoritmaları ve Kriptanalizlerinin İncelenmesi" adlı çalışmasında, şifreleme algoritmalarının kriptanalizleri incelenmiştir [17].

GÜVENOĞLU, 2006 yılında Trakya Üniversitesi’nde yaptığı "Görüntü Şifreleme Algoritmaları ve Performans Analizleri" adlı çalışmasında, mevcut olan resim şifreleme algoritmaları, algoritmaların genel yapıları ve performansları hakkında bilgi verilmektedir [1].

BÖLÜM 2. ASİMETRİK ALGORİTMALAR

2.1. Rivest-Shamir-Adleman (RSA) Kriptosistemi

RSA (Rivest-Shamir-Adleman) asimetrik şifreleme algoritması 1977 yılında R.Rivest, A.Shamir ve L.Adleman tarafından bulunmuş ve daha sonra asimetrik şifreleme algoritmalarına (genel anahtar şifrelemesi) uygun biçimde geliştirilmiştir. Bu algoritma, açık anahtarlı şifreleme sistemlerini ve sayısal imza işlemlerini işlemlerinde güvenli bir şekilde kullanılır [7,22].

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

RSA Algoritması açık anahtarlı şifreleme yönteminin temel bir uygulamasıdır. Bu şifreleme yönteminin önemli bir özelliği ise önceden aralarında hiçbir görüşme yapmamış olan alıcı ve vericinin kendi aralarındaki iletişimin güvenli bir ortamda (güvenli gönderim ve mesajların doğrulanması) yapılmasıdır.

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

Daha önce hiç karşılaşmamış, birbirini tanımayan kişiler bile birbirlerine gizli mesajlar gönderebilir. Örneğin Internet'ten alışveriş yapan birisi, kendisini hiçbir şekilde tanımayan bir web sitesine giderek, sitenin kamuya açık anahtarını alır, kart

numarasını bu anahtarla 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. Ama, acaba Web sitesi gerçekten dürüst bir satıcı mı, yoksa sahte bir site mi? Bundan emin olamayacaktır, ancak bunun da çözümü SERTİFİKA yöntemiyle sağlanmaktadır [7,23].

İnternetin şu anki durumu güvenliği tam olmayan kanallar üzerinden işler. Bu durumda asimetrik şifreleme tekniğiyle internette güvenlik sağlanmıştır.

RSA şifreleme algoritmasının çalışması Şekil 2.1'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 [7,24].

2.1.1. RSA algoritmasının yapısı

RSA şifreleme algoritmasında şifrelenecek olan açık metni öncelikle [0, n-1] arasındaki pozitif tamsayı bloklar haline dönüştürülür [7,25].
Bundan sonraki işlem 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 ikilik olarak $2^k < N$ olacak şekilde k bitlik kısımlara ayrılır.

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

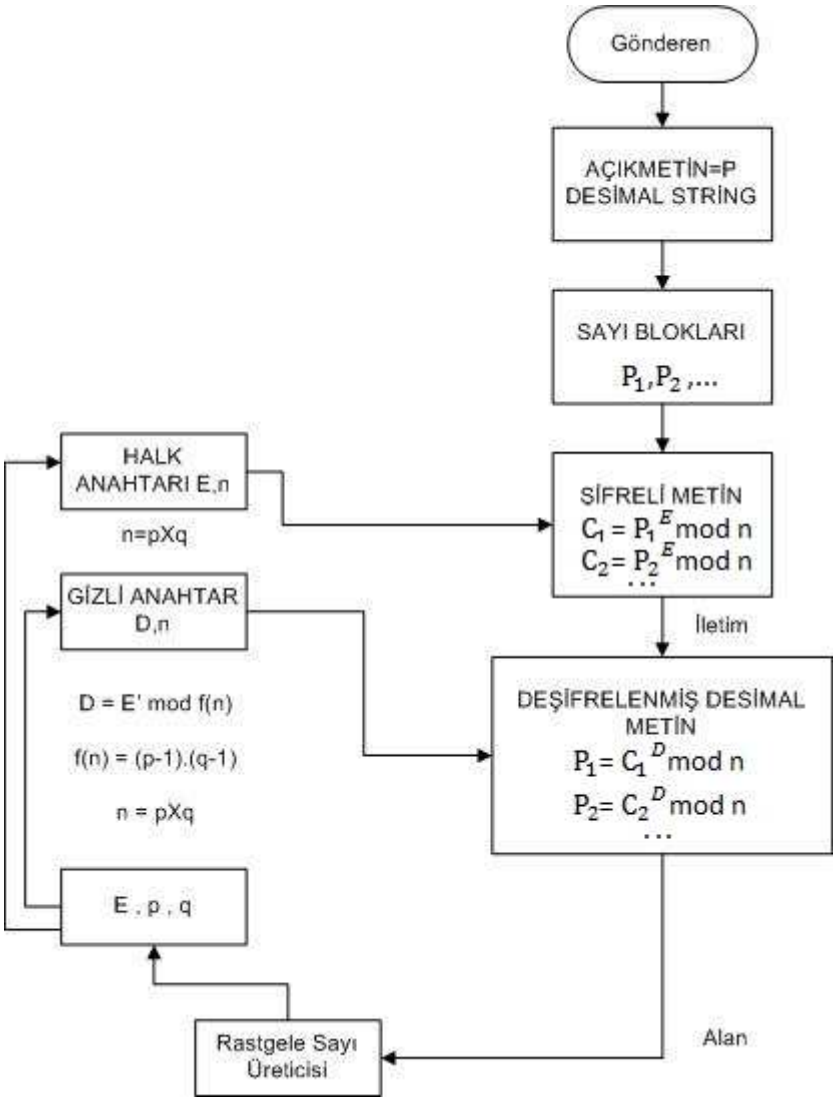
Daha sonra şifreleme için her bir kısma $C(i)=m(i)^E \bmod N$ işlemi uygulanır.

Böylece şifreleme işlemi bitmiş olur. Girişte kullanılan açık metin m şifrelenmiş olarak C şeklinde elde edilir.

Deşifreleme

Belirlenen D gizli anahtarı ile elimizde bulunan şifrelenmiş C metnini çözülmesi gerekmektedir. Bunun içinde şifrelemek için kullanılan bir matematiksel işlem kullanılır [7].

Gizli anahtar {D,n} kullanılarak şifre çözümü: $m(i)=C(i)^D \bmod N$



Şekil 2.1. RSA algoritması [7]

2.1.2. RSA sisteminin güvenirliği

RSA sisteminin "kırılması" birkaç değişik şekilde yorumlanabilir. Sisteme en çok zarar verecek saldırı bir kriptanalistin belli bir açık anahtara karşı gelen gizli anahtarı bulmasıdır. Bunu başarabilen bir “hasım” hem şifrelenen bütün masajları okuyabilir hem de imzaları taklit edebilir. Bunu yapmanın en akla gelen yolu n’nin asal çarpanlara ayrılması, yani p ve q’nun hesaplanmasıdır. p, q ve açık üs e kullanılarak d kolaylıkla hesaplanabilir. Ancak buradaki zorluk n modülünün çarpanlarına ayrılmasıdır. RSA sisteminin güvenliği çok büyük sayıların asal çarpanlarına

ayrılmasının zorluğu varsayımına dayanır. Büyük sayıların çarpanlarına ayrılmasının zorluğu ispatlanmış değildir. Son üç yüzyıl içerisinde Fermat ve Legendre gibi ünlü matematikçiler bu konuda çalışmalar yapmışlardır [7,26].

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

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

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

Bir başka saldırı da n'i çarpanlarına ayırmadan ϕ>(n)'nin hesaplanmasıdır. Eğer bir kriptanalist ϕ >(n)'i hesaplayabilirse e'nin ϕ >(n) modülüne göre çarpımsal tersini hesaplayarak d'yi bulabilir. Ancak bu n'nin çarpanlarına ayrılmasından daha kolay değildir çünkü bu yolla kriptanalist ϕ >(n)'i kullanarak n'i kolaylıkla çarpanlarına ayırabilir. Bunun için ϕ >(n) = n -(p + q) + 1 eşitliğinden n bilindiği için. (p + q) hesaplanır.

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

olduğundan (p -q) bulunur. Bu iki eşitlikten p ve q hesaplanabilir. Ayrıca d'nin hesaplanmasının n'in çarpanlarına ayrılmasından daha kolay olamayacağı iddia edilmektedir. Çünkü eğer d bilinirse n kolaylıkla çarpanlara ayrılabilir [7,27,28,29].

2.2. Dsa Algoritması

2.2.1. DSA algoritmasının yapısı

DSA (Digital Signature Algorithm), NIST (National Institute of Standards and Technology) tarafından sayısal imza standardı olarak yayınlanmıştır [3,30].

DSA ayrık logaritma problemine dayanır ve Schnorr ve ElGamal tarafından geliştirilen algoritmalarla benzer yapıdadır. DSA algoritması da RSA gibi açık anahtarlı bir kriptografik algoritmadır. RSA'dan farkı sadece imzalama amaçlı kullanılması, şifreleme yapılamamasıdır [3,30].

2.2.2. Dijital imzalar

İmzalar uzun zamandır yetkinin bir göstergesi yada en azından doküman içeriğinin kabul edildiğinin bir göstergesi olarak kullanılmaktadır. İmzalar hakkındaki genel kabulleri şu şekilde sıralamak mümkündür.

- İmzalar tekrar kullanılamazlar.
- İmzalı dokümanlar değiştirilemezler.
- İmzalar reddedilemezler inkar edilemezler.
- İmzalar taklit edilemezler.
- İmzalar mükemmel bir yetkilendirme sağlar.

Bununla birlikte, günlük hayatta elle attığımız imzalar için bu ifadelerin hiçbiri tamamıyla doğru değildir. İmzalar taklit edilebilir, dokümanlardan çıkartılabilir ve bu şekilde doküman içeriği imzalandıktan sonra bile değiştirilebilir.

Bu tür işlemler bilgisayar ortamında yapılmaya kalkıldığında, daha farklı problemlerle karşılaşılır. Öncelikle bilgisayar ortamında, dosyalar kolaylıkla kopyalanabilir. Yani bir dokümandan geçerli bir imzayı (kişinin imzasının grafiksel gösterimi) alıp diğerine yapıştırmak oldukça kolay olacaktır. Bunun yanında, dosyalar herhangi bir kanıt olmaksızın imzalandıktan sonra bile çok kolay şekilde değiştirilebilirler.

Dijital imzalar da bu tür problemleri aşmak amacıyla geliştirilmiştir. Elle atılan imza sadece imzalayana ait kimlik bilgisini gösterirken, dijital imza kimlik bilgisinin yanı sıra mesaj içeriğini de bildirmektedir. Bu yönüyle de dijital imza, el imzasından üstündür. Güvenli özet (haslı) fonksiyonları kullanıldığı sürece, kişinin imzasını mesajın içinden elde etmek ve başka bir mesaja eklemek veya imzalı bir mesajın, içeriğini değiştirmek hiçbir şekilde mümkün değildir. İmzalı bir mesajın içeriğindeki en küçük bir değişiklik dijital imza doğrulama işleminde hataya sebep olacaktır, eğer bir dijital imza doğrulanamazsa bunun sebebinin bir taklit girişimi yada bir iletim hatası olup olmadığına karar vermek zordur.

2.2.2.1. Dijital imzaların çalışması

Bir dokümanı dijital olarak imzalamak amacıyla açık anahtarlı kriptoloji kullanılabilir. Bazı açık anahtarlı algoritmalarda şifreleme ve şifre çözme işleminin simetrisine bağlı olarak, şifreleme işlemi için açık anahtar veya gizli anahtar kullanılabilir. Bu algoritmalar dijital olarak imzalanan dokümanlarda kullanılmaktadır. Bu algoritmalar ilk olarak Diffie ve Helmann tarafından geliştirilmiştir [3,31]. Böyle bir sistemde kişinin kendi gizli anahtarını kullanarak mesaj içeriğini imzalamasıyla güvenli bir dijital imza elde edilir. DSA gibi sistemlerde ise dijital imzalar için şifreleme algoritmasından farklı başka bir algoritma kullanılır. Protokol basit olarak şu şekilde çalışır [3]:

- Gönderici mesajı imzalamak suretiyle, kendi gizli anahtarı ile mesaj içeriğini şifreler.

- Gönderici imzalı mesajı alıcıya gönderir.
- Alıcı, dijital imzayı doğrulamak için gönderilen mesajı göndericinin açık anahtarı ile çözer.

Eğer alıcı üçüncü adımı gerçekleştiremiyorsa, dijital imza geçerli değildir denilmektedir.

Bu protokol ayrıca ideal bir imzada olması gereken aşağıdaki özellikleri de sağlamaktadır [3].

- İmza gerçektir, eğer alıcı göndericinin açık anahtarı ile mesajı doğrulayabiliyorsa mesajın gönderici tarafından imzalandığını bilir.
- Gönderici gizli anahtarını sadece kendisi bildiği için imza taklit edilemez.
- İmza tekrar kullanılamaz, imza değeri mesaj içeriğinin bir fonksiyonu olacağı için bu imza diğer mesajlar için kullanılamaz.
- İmzalı doküman değiştirilemez, eğer mesaj içeriğinde herhangi bir değişiklik olursa imza gönderenin açık anahtarı ile doğrulanamaz.
- İmza reddedilemez, alıcı gönderenin yardımına ihtiyaç duymadan imzayı doğrulayabilir

Pratik uygulamalarda, büyük boyuttaki dokümanlar için açık anahtarlı algoritmaları kullanmak çok uygun değildir. Zaman açısından, dijital imza protokolleri tek yönlü özet (hash) fonksiyonları kullanılarak uygulanmaktadır. Gönderici dokümanın tamamını imzalamak yerine mesajın bir özeti olan özet (hash) değerini imzalar. Bu protokolde, tek yönlü özet (hash) fonksiyonu ve dijital imza algoritması önceden anlaşılır.belirlenir. İşlem akışı şu şekilde gerçekleşmektedir:

- Gönderici, dokümandan imzalanıp gönderilecek olan mesaj özetini çıkarır.
- Gönderici dokümanı imzalamak için çıkardığı mesai özetini kendi gizli anahtarı ile şifreler.
- Gönderici mesajı ve şifrelenmiş mesaj özetini alıcıya gönderir.

- Alıcı aldığı mesajın mesaj özetini çıkartır. Daha sonra, aldığı şifreli mesaj özetini göndericinin açık anahtarı ile çözer. Eğer çözülmüş olan mesaj özeti ile kendi çıkarmış olduğu mesaj özeti ile uyuşuyorsa, dijital imza geçerli bir imzadır.

Tek yönlü özet (hash) fonksiyonları kullanmanın ekstra faydalan bulunmaktadır. İlk olarak imzalama hızını arttırır. İkinci olarak imza mesajdan ayrı tutulabilir. Üçüncü olarak da alıcının doküman ve imza için ayırması gereken saklama alanı oldukça küçülür.

Genelde, imzalama ve doğrulama işlemleri, kullanılan algoritmanın detaylarından bağımsızdır. Bir mesaj d gizli anahtarı ile imzalanırken Sd(T) ilgili açık anahtar değeri olan e ile de Ve(T) imza doğrulanmaktadır.

İmzalandıktan (dökümanın gizli anahtar ile şifrelenmiş mesaj özeti) sonra dökümana eklenen bit dizisine dijital imza denir. Alıcıyı mesajın göndericisi ve mesaj içeriğini öğrenmesini sağlayan protokol yetkilendirme olarak adlandırılır.

BÖLÜM 3. SİMETRİK ALGORİTMALAR

3.1. DES Algoritması

DES (Data Encryption Standart) dünya üzerinde en çok kullanılan modern blok metodlu şifredir. Algoritma 64 bitlik bir veri blokunun 56 bitlik bir anahtarın kontrolü altında şifrelenmesi ve deşifre edilmesi için geliştirilmiştir. DES ayrıca 64 bit veri blokuyla çalışan bir Feistel şifresidir. Amerikan Ulusal Standartlar Bürosu NBS'in, değişik sistemler tarafından kullanılabilecek, çok gizli olmayan kamu bilgilerini veya özel sektöre ait hassas ticari bilgileri koruma altına alacak herkese açık bir algoritma geliştirme çabalarının ürünüdür. DES algoritması 1976 yılında onaylanmış ve 1977 yılında "Data Encryption Standard" FIPS PUB 43 numarasıyla yayınlanmıştır. DES ile ilgili son düzenlemelerin olduğu standart 1999 yılında yayınlanan FIPS PUB 43-3 tür. 1986 yılında DES algoritması ISO tarafından uluslararası standart olarak kabul edilmiştir. DES in en büyük kullanıcı gruplarından biri bunu EFT ve EFTPOS işlemlerinde kullanan banka sektörüdür [12].

3.1.1. DES algoritmasının yapısı

DES algoritması, 64 bit açık metni, 56 bit anahtar altında, 64 bit şifreli metne çeviren blok şifredir. DES tanımında, bit sıralaması soldan sağa 1’den 64'e kadar yapılmıştır. Başka bir deyişle, bir numaralı bit birinci sekizlinin (byte) en yüksek anlamlı bitine ve 64 numaralı bit sekizinci sekizlinin en düşük anlamlı bitine karşılık gelir. IBM tarafından bazı hassas verilerin güvenliğini sağlamak için geliştirilmiştir ve 1976'dan bu yana kullanılmaktadır.

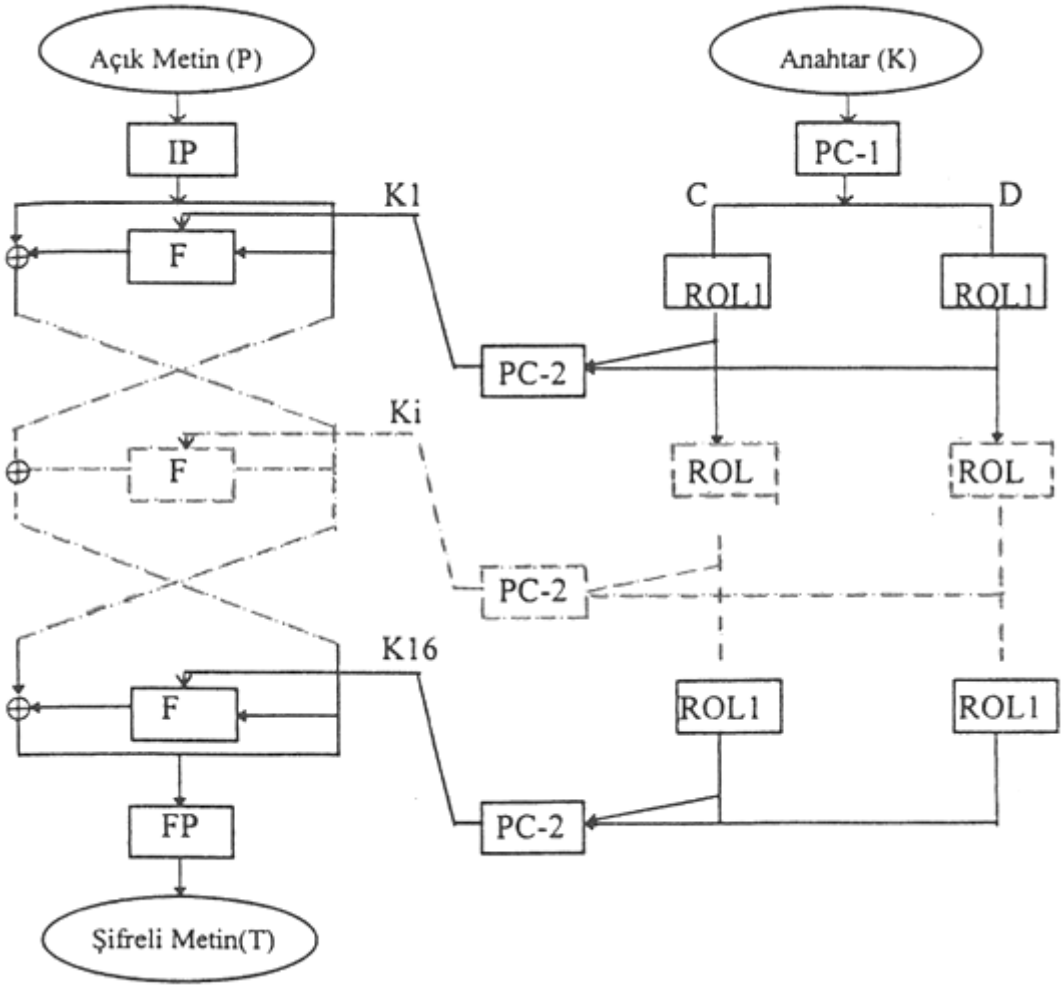
Algoritmanın ilk kısmında açık metnin bit sıralamasının değiştirildiği giriş permütasyonu (Initial Permutation, IP), sonunda da şifreli metnin bit sıralamasının

değiştirildiği ve giriş permütasyonunun tersi olan çıkış permütasyonu (Final Permutation, FP) bulunur. Bu permütasyonlar diferansiyel kripto analizde göz önüne alınmaz. İki permütasyon arasında yinelemeli çevrimlerden (round) oluşan, algoritmanın ana gövdesi yer alır. Algoritmanın ana gövdesi, veriyi 32 bitlik sağ ve sol yan veri olarak ikiye ayırır. Algoritmanın temel işlemi "çevrim" olarak adlandırılır. Her çevrimde, sağ ve sol yan veriler ile anahtar düzenleme algoritmasından elde edilen 48 bit anahtarlar kullanılarak yeni sağ ve sol yan veriler hesaplanır. Çeşitli sayılarda çevrime sahip olan versiyonlan bulunmakla beraber DES 16 çevrimden oluşur. Her çevrimde verinin sağ yansı ve her çevrim için farklı değere sahip olan 48 bit anahtar kullanılarak F fonksiyonu hesaplanır. Verinin sol yarısı bu F fonksiyonunun çıkışı ile XOR'lanır. İki çevrim arasında verinin iki yarısı yer değiştirilir (bu işlem birinci çevrimden önce ve son çevrimden sonra yapılmaz). DES'in yapısı Şekil 3.1'de, giriş permütasyonu Tablo 3.1'de gösterilmiştir [10].

Tablo 3.1 DES giriş permütasyonu [10]

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

F fonksiyonu, verinin 32 bitlik sağ yansını Tablo 3.2'de gösterilen E genişletmesini (Expansion) kullanarak 48 bite genişletir ve bu sonuç 48 bit anahtar ile XOR'lanir. Buradan elde edilen değer, herbiri kendi özel tablolarını kullanan, 6 biti 4 bite karşı düşüren S1, S2, ..., S8 S-kutularına verilir. Ömek olması amacıyla S1 kutusu Tablo 3.4'de verilmiştir. Diğer S-kutulan ve bunlara ait fark dağılım tabloları sırasıyla Ek A ve Ek B'de verilmiştir. Bu S kutularının çıkışları sıralanır ve Tablo 3.3'de gösterilen P permütasyonu kullanılarak çıkış bitlerinin sırası yer değiştirilir. Buradan elde edilen sonuç F fonksiyonunun çıkışını oluşturur. F fonksiyonu Şekil 3.2'de gösterilmiştir [10].



Şekil 3.1. DES ve anahtar düzenleme algoritması [10]

DES'in S-kutuları, 6 bitten 4 bite düzenleme tablolarıdır. Her bir S kutusu 64 mümkün giriş değerini (6-bit), 16 çıkış değerine (4-bit) karşı düşürür. DES'in standart tanımlamasında S-kutulan, 0,1,2,.....,15 değerlerinin dört ayrı permütasyonu olarak tanımlanır. 6 giriş bitinin ortadaki 4 tanesi sıralanacak değeri, kalan 2 tanesi de (bit 1 ve bit 6) permütasyon seçimini gösterir. Örneğin; S1 kutusunda 101001 giriş değeri 4 çıkış değerine karşı düşer [10].

Tablo 3.2. E genişletmesi [10]

32	1	2	8	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

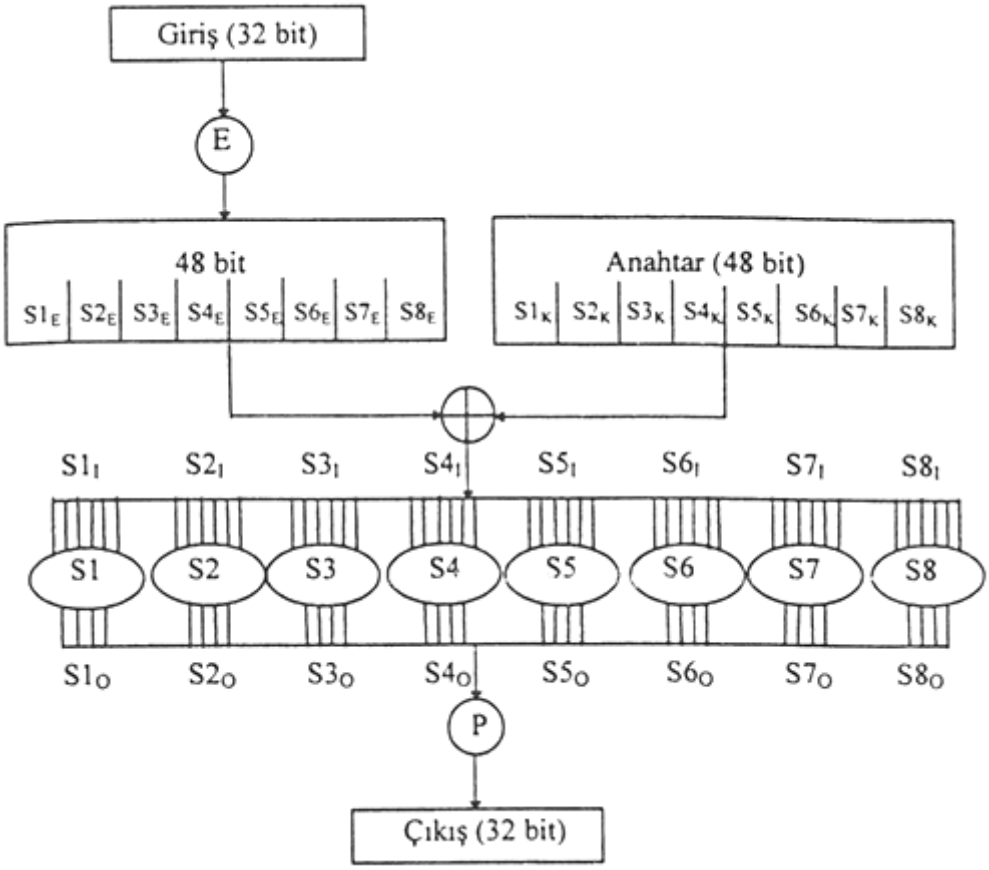
Tablo 3.3. P permütasyonu [10]

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

Tablo 3.4. S1 kutusu [10]

E	4	D	1	2	F	B	8	3	A	6	C	5	9	0	7
0	F	7	4	E	2	D	1	A	6	C	B	9	5	3	8
4	1	E	8	D	6	2	B	F	C	9	7	3	A	5	0
F	C	8	2	4	9	1	7	5	B	3	E	A	0	6	D

Anahtar düzenleme algoritması, 56 bit anahtardan 16 tane 48 bit K1, K2, ..., K16 anahtar değerlerini hesaplar. Bu K1, K2, ..., K16 anahtarları DES'in çevrimlerinde F fonksiyonu girişi olarak kullanılırlar. Başlangıçtaki anahtar bitleri, bit numarası sekize bölünenler (8,16,...) eşlik biti olacak şekilde l'den 64'e kadar numaralandırılır. Elde edilen anahtar bitleri Tablo 3.6'da görülen PC-1 permütasyonu kullanılarak yeniden sıralanır ve 28 bitlik C ve D kaydedicilerine yüklenir. C kaydedicisinin bitleri anahtarın 57, 49, ..., 36. bitleri, D kaydedicisinin bitleri de anahtarın 63, 55, ..., 4. bitleridir. Her bir çevrimde C ve D kaydedicileri Tablo 4.5'de gösterildiği gibi bir ya da iki bit sola kaydırılır. Elde edilen C ve D kaydedicisi değerleri birleştirilip l'den 56'ya kadar numaralandırılır ve Tablo 3.7'de gösterilen PC-2 permütasyonu kullanılarak 48 bit anahtar elde edilir. Anahtar düzenleme algoritması Şekil 3.2'de gösterilmiştir [10].



Şekil 3.2. DES'in F fonksiyonu [10]

Tablo 3.5. Anahtar düzenleme algoritmasında çevrim numarasına göre sola kaydırma sayıları [10]

Çevrim Numarası :	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Kaydırma Miktarı :	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

Tablo 3.6. PC-1 permütasyonu [10]

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

Tablo 3.7. PC-2 permütasyonu [10]

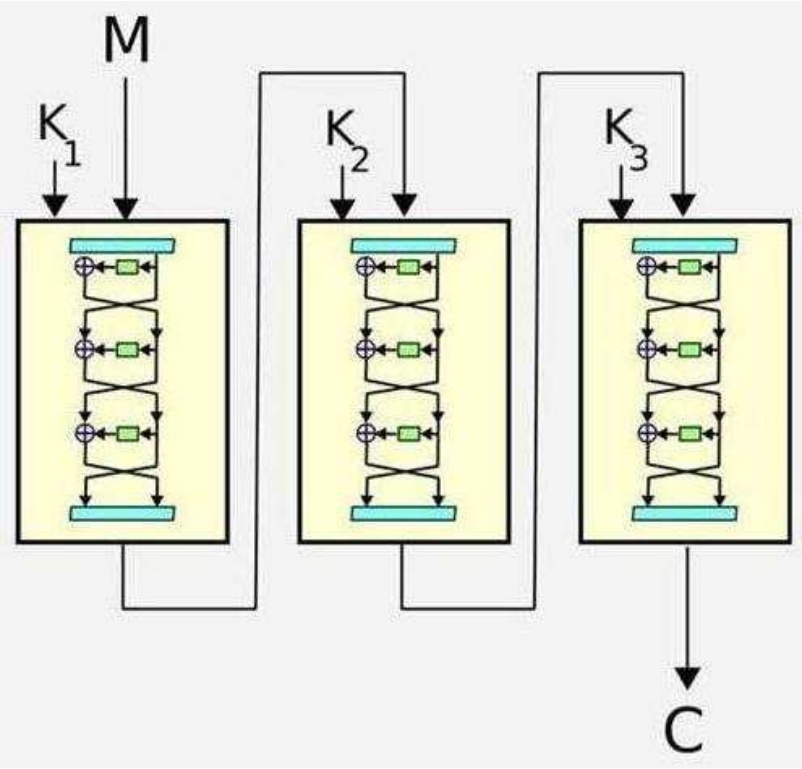
14	17	11	24	1	5
3	28	15	6	21	10
23	19	12	4	26	8
16	7	27	20	13	2
41	52	31	37	47	35
30	40	51	45	33	48
44	49	39	56	34	53
46	42	50	36	29	32

3.1.2. 3DES algoritması

3DES (Üçlü DES), 1978 yılında IBM tarafından geliştirilmiş olan bir şifreleme algoritmasıdır. Brute Force saldırılara karşı koymakta zorlanan DES (Data Encryption Standard-Veri Şifreleme Standardı) algoritmasının üzerine geliştirilmiştir.

Özellikleri:

- Çift yönlü çalışır. Şifrelenmiş veri geri çözülebilir.
- DES şifrelemesinin 3 kere art arda yapılması şeklinde çalışır.
- DES şifreleme yöntemine göre 3 kat daha yavaş çalışır.
- Şifreleme yapmak için uzunluğu 24 bayt olan bir anahtar kullanılır. Her bayt için 1 eşlik biti vardır. Dolayısıyla anahtarın uzunluğu 168 bittir.
- Veri, 3DES anahtarının ilk 8 baytı ile şifrelenir. Sonra veri anahtarın ortadaki 8 baytı ile çözülür. Son olarak anahtarın son 8 baytı ile şifrelenerek 8 bayt bir blok elde edilir.
- Algoritmanın akış diyagramı Şekil 3.3. ‘teki gibidir [32].



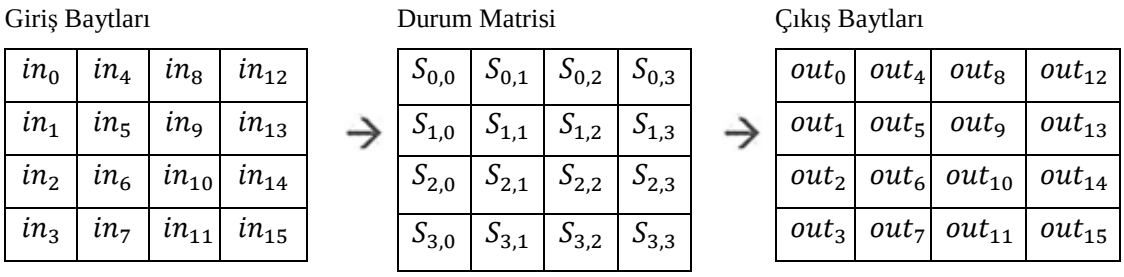
Şekil 3.3. 3DES algoritmasının akış diyagramı [32]

3.2. AES Algoritması

AES Amerika Birleşik Devletleri tarafından kabul edilmiş bir şifreleme standardıdır. Rijndael olarak da bilinen standart bir blok şifreleyicidir. DES algoritmasının gelişen teknoloji karşısında zayıflaması ve güvenilirliğini yitirmesi ile birlikte, yeni bir şifreleme standardının belirlenebilmesi amacıyla NIST (National Institute of Standards and Technology) organizasyonunun düzenlemiş olduğu yarışmanın sonucunda iki Belçikalı araştırmacı Joan Daemen ve Vincent Rijmen tarafından geliştirilen algoritma yeni standart olarak belirlenmiştir. Uzun süren bir standardizasyon ve doğrulama sürecinin ardından 26 Kasım 2001 tarihinde AES FIPS 197 standardı olarak NIST tarafından yayımlanmıştır. DES algoritmasına oranla AES algoritması daha yüksek güvenilirlik sağlamakla birlikte, kolay gerçeklenebilir olması açısından da avantajlara sahiptir. Algoritmanın farklı anahtar ve veri bloğu uzunluklarını destekliyor olmasına karşın standart, sabit 128-bit'lik veri bloğu ile 128-bit, 192-bit veya 256-bit'lik anahtar uzunluklarını kapsamaktadır. AES içerisinde 128-bit'lik veri blokları her biri 32-bit'ten oluşan 4 kelime olarak düşünülmektedir. AES ile şifreleme işlemine başlanırken 128-bit yani 4 kelimeden oluşan veri bloğu durum dizisi içerisine yazılır ve algoritma sırasındaki gerekli işlemlerin tümü bu dizi kullanılarak gerçeklenir. Şifreleme için gerekli en son işlemin de bitimiyle birlikte durum dizisinin son hali çıkış dizisine yazılır [33].

Örneğin; in_0, in_1, ..., in_{15} den oluşan veri bloğu durum dizisine yazılır ve gerekli işlemlerin tümü bu dizi üzerinde gerçeklenir. İşlemlerin tamamlanması ile birlikte şifrelenmiş veri çıkışa out_0, out_1, ..., out_{15} bayt dizisi olarak verilir.

AES algoritması genel olarak iki bloktan oluşur, ilk blok tur dönüşüm ikinci blok ise anahtar üretim bloğudur. Algoritma tekrarlı bir yapıya sahiptir, anahtar bloğunun uzunluğuna bağlı olarak 128-bit, 192-bit veya 256-bit, tur dönüşüm



Şekil 3.4. AES durum dizisi [33]

Tablo 3.8. Tur S-sayısının anahtar uzunluğuna göre değişimi [33]

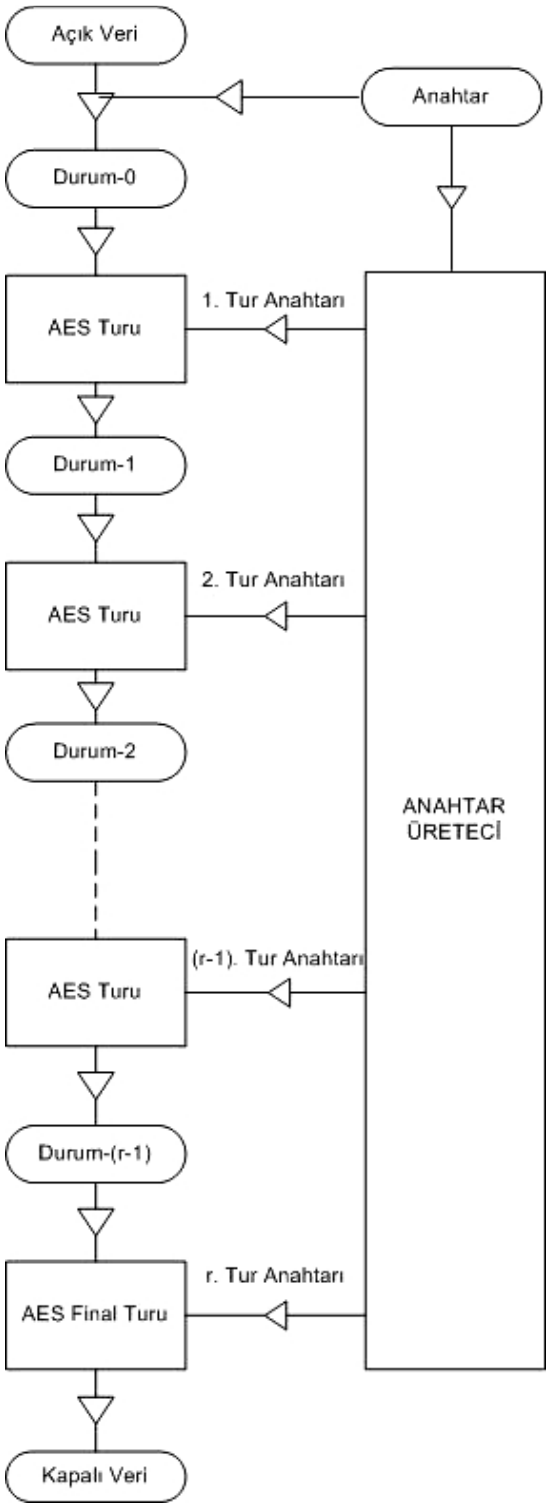
AES Tipi	Anahtar Uzunluğu (***Nk*** Kelime)	Tur Sayısı (N_r)
AES-128	4	10
AES-192	6	12
AES-256	8	14

işlemi sırası ile 10, 12 veya 14 kez tekrarlanır. Tekrarlanma sayıları anahtar bloğunun uzunluğuna bağlı olarak Tablo 3.8'de verilmektedir.

Şifreleme başlangıcında şifrelenecek olan blok, Şekil 3.4 uyarınca durum dizisine yazılır. Durum dizisinin giriş anahtarı ile toplanması işlemi ile birlikte şifreleme işlemine başlanmış olur. Anahtar bloğunun uzunluğuna bağlı olarak tur dönüşüm işlemi 10, 12 veya 14 kez tekrarlanır. Algoritmanın sözlü kod ifadesi, Nr tur sayısını ifade etmek üzere aşağıda verilmektedir, ayrıca algoritmaya ait blok şema Şekil 3.5'te verilmektedir.

```
AESşifreleme(GirişDurumu, Anahtar)
{
        T_Anahtarı[Nr : 1] = AnahtarÜreteci(Anahtar);
        Durum[0]= İlkAnahtarToplaması(GirişDurumu, Anahtar);
        for(i = 1;i < Nr ; i++)
                Tur(Durum[i-1] , T_Anahtarı [i] ) FinalTuru(Durum[Nr-1],
        T_Anahtarı[Nr])
}
```

Tur dönüşüm işlemi sırasında durum dizisi üzerinde Bayt Değiştirme, Satırları Kaydırma, Sütunları Karıştırma ve Tur Anahtarını toplama alt işlemleri uygulanmaktadır. Tur dönüşüm işleminin çıktısı olarak elde edilen 128-bitlik veri



Şekil 3.5. AES blok şema

anahtar üretim işlemi sonucunda üretilen anahtar verisiyle toplanmaktadır. En son tur işleminin gerçeklenip anahtar bloğu ile toplanması sonucu şifrelenmiş blok elde edilir. Son turda gerçeklenen işlemler önceki turlarda gerçeklenen işlemlerden farklılık gösterir. Son turda Sütunları Karıştırma işlemi yapılmamakta Satırları Kaydırma işleminin çıktısı tur anahtarı ile toplanmaktadır, sözlü kod ifadesi aşağıdaki algoritmada verilmektedir.

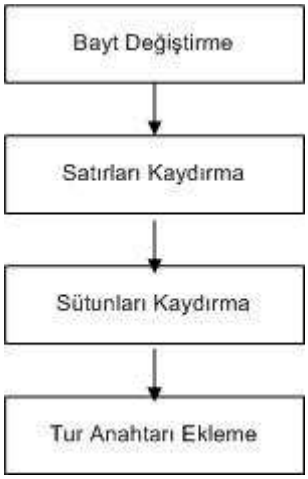
```
Tur(GirişDurumu, T_Anahtarı[i] )
{
        Duruml = BaytDeğişimi(GirişDurumu);
        Durum2 = SatırKaydırma(Duruml);
        Durum3 = SütunKarıştırma(Durum2);
        ÇıkışDurumu = TurAnahtarmıTopla (Durum3 , T_Anahtarı [i] ) ;
}
SonTur(GirişDurumu, T_Anahtarı[Nr])
{
        Duruml = BaytDeğişimi(GirişDurumu);
        Durum2 = SatırKaydırma(Duruml);
        ÇıkışDurumu = TurAnahtarmıTopla (Durum2 , T_Anahtarı [Nr] );
}
```

3.2.1. Tur dönüşüm işlemleri

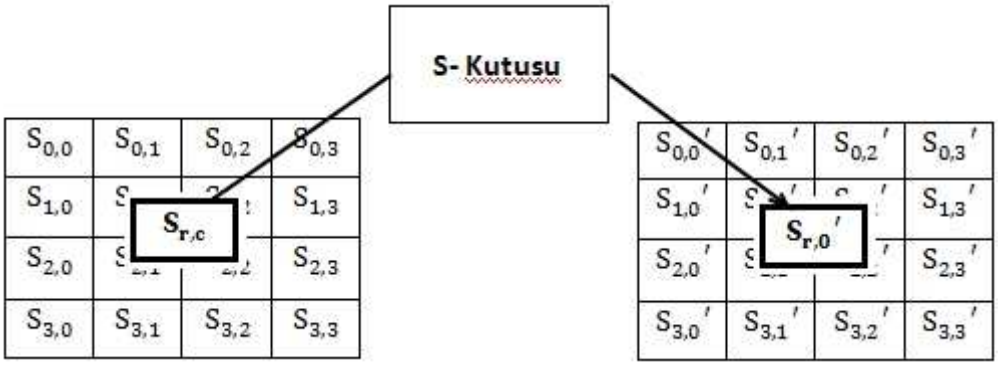
AES algoritması önceden de belirtildiği gibi tekrarlı bir yapıdan oluşur. Tur dönüşüm işlemi anahtar uzunluğuna bağlı olarak çok defa tekrarlanır. Tur dönüşüm işlemi içerisinde Bayt Değiştirme, Satırları Kaydırma ve Sütunları Karıştırma ve Tur Anahtarını Toplama işlemleri gerçekleştirilir. Tur dönüşüm işlemine ait blok diyagram Şekil 3.6'da verilmektedir.

3.2.2. Bayt değiştirme

Bayt değiştirme işlemi 8-bitlik veriler üzerinde yani bir bayt üzerinde gerçeklenen lineer olmayan bir işlemdir. Bu işlem durum matrisinin her bir baytı için bağımsız olarak Şekil 3.6'da verildiği gibi gerçeklenir. Her bir bayt için karşılık düşen bayt



Şekil 3.6. Tur dönüşüm blok şema



Şekil 3.7. Bayt değiştirme dönüşümü

birbirinden farklıdır. Bu dönüşüm iki aşamada gerçekleşir. Bu aşamalardan ilki çarpmaya göre ters alma işlemidir. Çarpmaya göre ters alma işlemi GF(2^8)'de gerçeklenir. Evrik alma işlemi gerçeklenirken kullanılan indirgeme polinomu P(x) = $x^8 + x^4 + x^2$ + x + 1 'dir. İkinci aşama ise evrik çikışm geçiş matrisi ile GF(2^8)'de çarpılması ve sabit bir matris ile toplanmasıyla gerçekleştirilir. Bu işlem Denklem

3.1'de verilmektedir. Bayt Değiştirme işlemini farklı yöntemlerle gerçeklemek mümkündür.

$$\begin{bmatrix} b_0' \\ b_1' \\ b_2' \\ b_3' \\ b_4' \\ b_5' \\ b_6' \\ b_7' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} \quad (3.1)$$

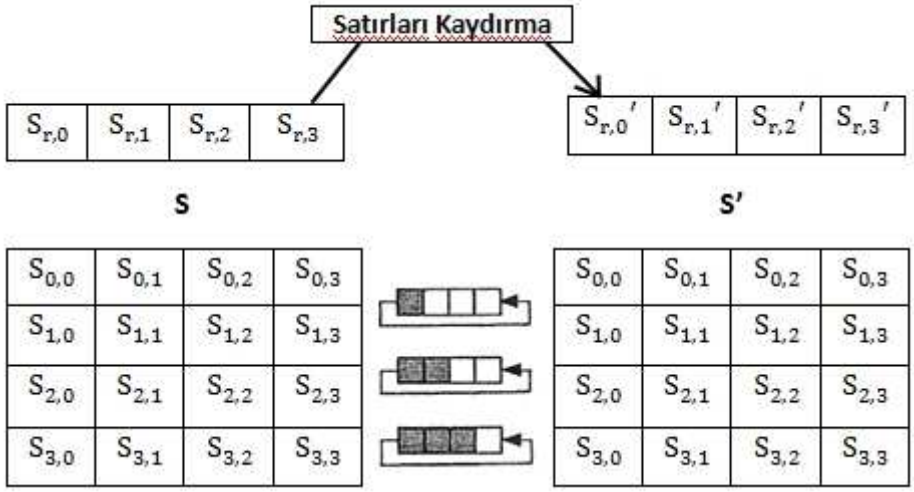
Bayt değiştirme işleminde her bir bayta karşılık düşen değer 16'lık tabanda Tablo 3.9'da verilmektedir.

Tablo 3.9. S-Kutusu çıkışları [33]

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	63	7c	77	7b	f2	6b	6f	C5	30	01	67	2b	fe	d7	ab	76
1	ca	82	c9	7d	fa	59	47	fO	ad	d4	a2	af	9c	a4	72	cO
2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	fl	71	d8	31	15
3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
4	09	83	2c	la	lb	6e	5a	aO	52	3b	d6	b3	29	e3	2f	84
5	53	dl	00	ed	20	fc	bl	5b	6a	cb	be	39	4a	4c	58	Cf
6	dO	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
8	cd	Oc	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	Ob	Db
A	eO	32	3a	Oa	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
B	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
C	ba	78	25	2e	lc	a6	b4	c6	e8	dd	74	lf	4b	bd	8b	8a
D	70	3e	b5	66	48	03	f6	Oe	61	35	57	b9	86	cl	İd	9e
E	el	f8	98	11	69	d9	8e	94	9b	le	87	e9	ce	55	28	Df
F	8c	al	89	Od	bf	e6	42	68	41	99	2d	Of	bO	54	bb	16

3.2.3. Satırları kaydırma

Satırları kaydırma işleminde Durum matrisini ilk satırı hariç diğer satırları kaydırma işlemine tabii tutulur. Bu kaydırma işlemi sırasında 2. satır bir bayt kaydırılırken, 3.satır iki bayt, son satır ise üç bayt kaydırılır. Bayt Kaydırma işlemine ait blok şema Şekil 3.8'de verilmektedir.



Şekil 3.8 Satırları kaydırma işlemi

3.2.4. Sütunları karıştırma

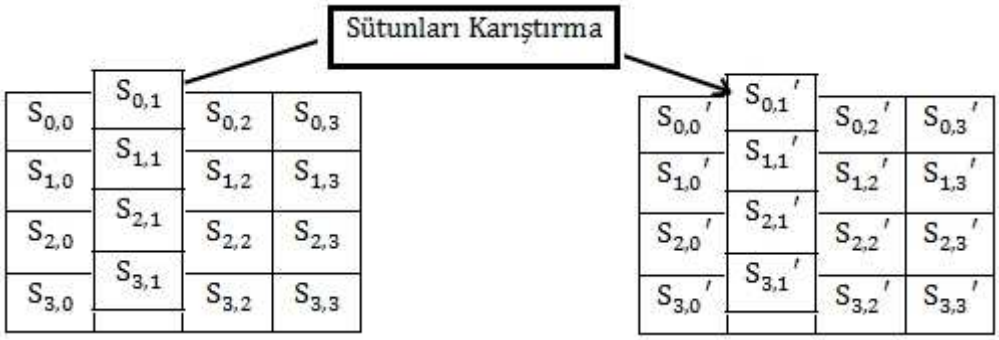
Sütunları Karıştırma işlemi Durum matrisindeki her bir sütun üzerinde bağımsız olarak gerçeklenir. Bu işlem gerçeklenirken her bir satır GF(2^8) 'de bir polinom olarak düşünülür. Her bir satır üzerinde a(x) polinomu ile modülo x4 + l'de çarpma işlemi gerçekleştirilir. Sabit a(x) polinomu ve satırları karıştırma işlemi Denklem 3.2 ve Denklem 3.3'de verilmektedir.

$$a(x) = \{03\}x^3 + \{01\}x^2 + \{01\}x + \{02\} \quad (3.2)$$

$$s(x) = a(x) * s(x) \quad (3.3)$$

Durum matrisi üzerinde gerçeklenen satırları karıştırma işlemi ve bu işlemin matris çarpımı olarak ifadesi Şekil 3.9 ve Denklem 3.4'de verilmektedir .

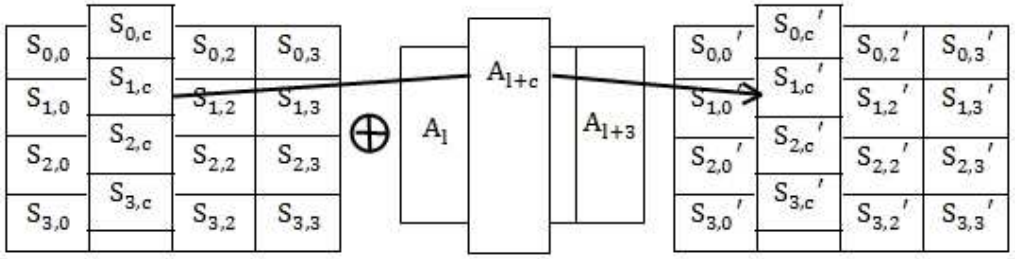
$$\begin{bmatrix} S_{0,c}' \\ S_{1,c}' \\ S_{2,c}' \\ S_{3,c}' \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} S_{0,c} \\ S_{1,c} \\ S_{2,c} \\ S_{3,c} \end{bmatrix} \quad (3.4)$$



Şekil 3.9 Satırları karıştırma işlemi

3.2.5. Tur anahtarı ekleme

Tur Anahtarının Ekleme işleminde ise önceden anahtar üretim bloğu tarafından üretilen 128-bitlik anahtar dizisi Durum matrisi ile e-xor'lanır Anahtar üretim bloğu ilerideki bölümde ayrıntılı olarak incelenecektir. Anahtar ekleme işlemine ait genel şema Şekil 3.10'da verilmektedir.

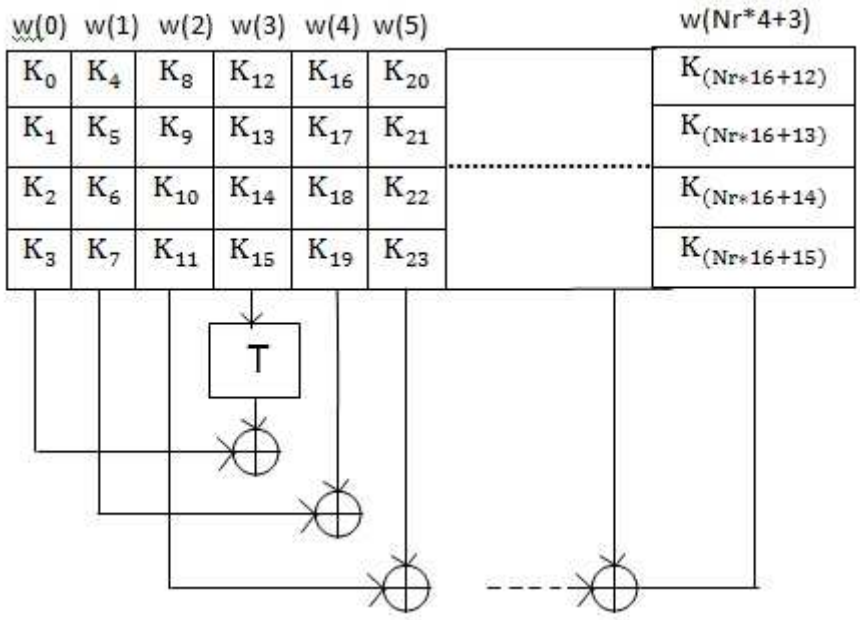


Şekil 3.10 Anahtar ekleme işlemi

3.2.6. Anahtar üretim işlemleri

AES algoritması K anahtar dizisini alıp her bir tur için gerekli anahtar bloklarını üretir. AES algoritması farklı anahtar uzunlukları ile çalışabilmektedir, fakat veri bloğunun uzunluğu 128-bit olduğu için üretilen anahtar uzunluğu 128-bittir. Her bir tur dönüşüm işleminin son kısmı olan Tur Anahtarını Ekleme işlemi sırasında

üretilen 128-bitlik anahtar kullanılır. Anahtar üretim bloğu ilk olarak girişten alınan 128 bitlik anahtar bloğunu elemanları baytlar olan 4x4'lük matrisler halinde kaydeder. Daha sonra üretilen diğer 128-bitlik anahtar blokları aynı şekilde matrisin genişletilmesi için kullanılır. Böylece Nr tur sayısı olmak üzere 4* (Nr + 1) boyutunda bir matris elde edilir. Anahtar üretim işlemleri bu matrisin sütunları üzerinde Şekil 3.11'de verildiği gibi gerçeklenir. Oluşturulacak olan sütun



Şekil 3.11. 128 bitlik giriş anahtarı için anahtar üreteci

girişte kullanılan anahtar bloğunun kelime sayısının (Nb) katı değilse (128 bitlik anahtar bloğu için Nb = 4) kendinden bir önceki ve Nb önceki sütunların karşılıklı e-xor'lanması sonucu bulunur. Oluşturulacak olan sütun kelime sayısının katı olduğu durumda ise kendinden bir önceki sütun dönüştürme işlemine sokulur, ardından Nb önceki sütunla karşılıklı e-xor'lama işlemi gerçekleştirilir. Dönüşüm işlemi üç aşamada gerçekleştirilir. İlk aşamada 4 baytlık veriden oluşan sütun üzerinde kaydırma işlemi gerçeklenir. Kaydırma işleminden sonra her bir bayt, bayt değiştirme işlemine tabii tutulur. Bayt değiştirme işleminin ardından tur vektörü ile e-xor'lama işlemi gerçekleştirilir. Tur vektörünün ilk baytı kaçıncı turda olunduğuna bağlı olarak değişir, diğer baytları ise sürekli sıfır değerindedir [33].

3.3. RC2 Algoritması

RC2, Ron Rivest tarafından RSA güvenliği için 1987 yılında tasarlanmıştır. Resmi olarak RC harflerinin “Rivest Cipher” simgelediği tanımlanmış olsa bile bir çok kişi “Ron’s Code” ’u simgelediğini düşünür.

RC 2 gizli anahtarlı şifreleme yapan klasik blok şifreleme algoritmalarından biridir. DES Algoritmasının yerini alması amacıyla geliştirilmiştir. Giriş ve çıkış blok boyutları , her biri için 64 bittir. Anahtar 1 ile 128 byte arasında değişken uzunluktadır. Ancak mevcut uygulama da 8 byte olarak kullanmaktadır. Anahtara 40 ile 88 bit arası uzunluklarda değişen “salt” adında ayrı ek bir anahtar daha kullanılabilmektedir.

Algoritma 16 bit mikroişlemcilere uygulanması amacıyla geliştirilmiştir. IBM AT’ye göre şifreleme işlemi DES algoritmasına gore iki kat daha hızlı çalışmaktadır. (uygun anahtar atandığında)

Algoritma ile ilgili detaylı bilgiler başlangıçta gizli tutulmaktaydı. Fakat 1996 yılında kaynak kodu bilinmeyen kişiler tarafından internete sızdırılmıştır [34,35].

Kaynak kodu aşağıdaki gibidir [36];

```
/*********************************************************************\
* To commemorate the 1996 RSA Data Security Conference, the following  *
* code is released into the public domain by its author.  Prost!       *
*                                                                      *
* This cipher uses 16-bit words and little-endian byte ordering.       *
* I wonder which processor it was optimized for?                       *
*                                                                      *
* Thanks to CodeView, SoftIce, and D86 for helping bring this code to  *
* the public.                                                          *
\*********************************************************************/

#include <string.h>
#include <assert.h>

/*********************************************************************\
* Expand a variable-length user key (between 1 and 128 bytes) to a     *
* 64-short working rc2 key, of at most "bits" effective key bits.      *
* The effective key bits parameter looks like an export control hack.  *
* For normal use, it should always be set to 1024.  For convenience,   *
```

```
* zero is accepted as an alias for 1024.                          *
\*****************************************************************/

void rc2_keyschedule( unsigned short xkey[64],
                      const unsigned char *key,
                      unsigned len,
                      unsigned bits )
    {
    unsigned char x;
    unsigned i;
    /* 256-entry permutation table, probably derived somehow from pi */
    static const unsigned char permute[256] = {
        217,120,249,196, 25,221,181,237, 40,233,253,121, 74,160,216,157,
        198,126, 55,131, 43,118, 83,142, 98, 76,100,136, 68,139,251,162,
         23,154, 89,245,135,179, 79, 19, 97, 69,109,141,  9,129,125, 50,
        189,143, 64,235,134,183,123, 11,240,149, 33, 34, 92,107, 78,130,
         84,214,101,147,206, 96,178, 28,115, 86,192, 20,167,140,241,220,
         18,117,202, 31, 59,190,228,209, 66, 61,212, 48,163, 60,182, 38,
        111,191, 14,218, 70,105,  7, 87, 39,242, 29,155,188,148, 67,  3,
        248, 17,199,246,144,239, 62,231,  6,195,213, 47,200,102, 30,215,
          8,232,234,222,128, 82,238,247,132,170,114,172, 53, 77,106, 42,
        150, 26,210,113, 90, 21, 73,116, 75,159,208, 94,  4, 24,164,236,
        194,224, 65,110, 15, 81,203,204, 36,145,175, 80,161,244,112, 57,
        153,124, 58,133, 35,184,180,122,252,  2, 54, 91, 37, 85,151, 49,
         45, 93,250,152,227,138,146,174,  5,223, 41, 16,103,108,186,201,
        211,  0,230,207,225,158,168, 44, 99, 22,  1, 63, 88,226,137,169,
         13, 56, 52, 27,171, 51,255,176,187, 72, 12, 95,185,177,205, 46,
        197,243,219, 71,229,165,156,119, 10,166, 32,104,254,127,193,173
    };

    assert(len > 0 && len <= 128);
    assert(bits <= 1024);
    if (!bits)
        bits = 1024;

    memcpy(xkey, key, len);

    /* Phase 1: Expand input key to 128 bytes */
    if (len < 128) {
        i = 0;
        x = ((unsigned char *)xkey)[len-1];
        do {
            x = permute[(x + ((unsigned char *)xkey)[i++]) & 255];
            ((unsigned char *)xkey)[len++] = x;
        } while (len < 128);
    }

    /* Phase 2 - reduce effective key size to "bits" */
    len = (bits+7) >> 3;
    i = 128-len;
    x = permute[((unsigned char *)xkey)[i] & (255 >> (7 & -bits))];
    ((unsigned char *)xkey)[i] = x;

    while (i--) {
        x = permute[ x ^ ((unsigned char *)xkey)[i+len] ];
        ((unsigned char *)xkey)[i] = x;
    }

    /* Phase 3 - copy to xkey in little-endian order */
    i = 63;
    do {
        xkey[i] =  ((unsigned char *)xkey)[2*i] +
                (((unsigned char *)xkey)[2*i+1] << 8);
```

```
        } while (i--);
        }

/************************************************************************\
* Encrypt an 8-byte block of plaintext using the given key.             *
\************************************************************************/

void rc2_encrypt( const unsigned short xkey[64],
                  const unsigned char *plain,
                  unsigned char *cipher )
        {
        unsigned x76, x54, x32, x10, i;

        x76 = (plain[7] << 8) + plain[6];
        x54 = (plain[5] << 8) + plain[4];
        x32 = (plain[3] << 8) + plain[2];
        x10 = (plain[1] << 8) + plain[0];

        for (i = 0; i < 16; i++) {
                x10 += (x32 & ~x76) + (x54 & x76) + xkey[4*i+0];
                x10 = (x10 << 1) + (x10 >> 15 & 1);

                x32 += (x54 & ~x10) + (x76 & x10) + xkey[4*i+1];
                x32 = (x32 << 2) + (x32 >> 14 & 3);

                x54 += (x76 & ~x32) + (x10 & x32) + xkey[4*i+2];
                x54 = (x54 << 3) + (x54 >> 13 & 7);

                x76 += (x10 & ~x54) + (x32 & x54) + xkey[4*i+3];
                x76 = (x76 << 5) + (x76 >> 11 & 31);

                if (i == 4 || i == 10) {
                        x10 += xkey[x76 & 63];
                        x32 += xkey[x10 & 63];
                        x54 += xkey[x32 & 63];
                        x76 += xkey[x54 & 63];
                }
        }

        cipher[0] = (unsigned char)x10;
        cipher[1] = (unsigned char)(x10 >> 8);
        cipher[2] = (unsigned char)x32;
        cipher[3] = (unsigned char)(x32 >> 8);
        cipher[4] = (unsigned char)x54;
        cipher[5] = (unsigned char)(x54 >> 8);
        cipher[6] = (unsigned char)x76;
        cipher[7] = (unsigned char)(x76 >> 8);
        }

/************************************************************************\
* Decrypt an 8-byte block of ciphertext using the given key.            *
\************************************************************************/

void rc2_decrypt( const unsigned short xkey[64],
                  unsigned char *plain,
                  const unsigned char *cipher )
        {
        unsigned x76, x54, x32, x10, i;

        x76 = (cipher[7] << 8) + cipher[6];
        x54 = (cipher[5] << 8) + cipher[4];
```

```
x32 = (cipher[3] << 8) + cipher[2];
x10 = (cipher[1] << 8) + cipher[0];

i = 15;
do {
	x76 &= 65535;
	x76 = (x76 << 11) + (x76 >> 5);
	x76 -= (x10 & ~x54) + (x32 & x54) + xkey[4*i+3];

	x54 &= 65535;
	x54 = (x54 << 13) + (x54 >> 3);
	x54 -= (x76 & ~x32) + (x10 & x32) + xkey[4*i+2];

	x32 &= 65535;
	x32 = (x32 << 14) + (x32 >> 2);
	x32 -= (x54 & ~x10) + (x76 & x10) + xkey[4*i+1];

	x10 &= 65535;
	x10 = (x10 << 15) + (x10 >> 1);
	x10 -= (x32 & ~x76) + (x54 & x76) + xkey[4*i+0];

	if (i == 5 || i == 11) {
		x76 -= xkey[x54 & 63];
		x54 -= xkey[x32 & 63];
		x32 -= xkey[x10 & 63];
		x10 -= xkey[x76 & 63];
	}
} while (i--);

plain[0] = (unsigned char)x10;
plain[1] = (unsigned char)(x10 >> 8);
plain[2] = (unsigned char)x32;
plain[3] = (unsigned char)(x32 >> 8);
plain[4] = (unsigned char)x54;
plain[5] = (unsigned char)(x54 >> 8);
plain[6] = (unsigned char)x76;
plain[7] = (unsigned char)(x76 >> 8);
}
```

3.4. Blowfish Algoritması

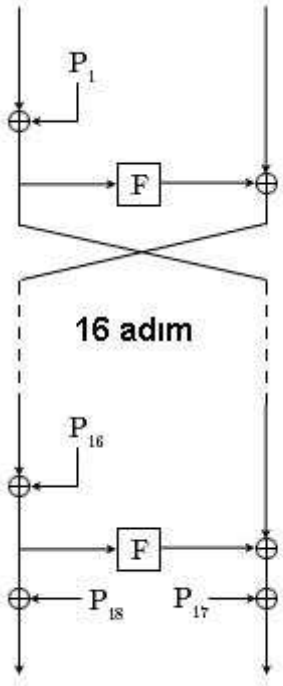
Feistel ağı kullanan bir blok şifreleme yöntemidir. Blowfish, Bruce Schneier tarafından 1993 yılında tasarlanmış, çok sayıda şifreleyici ve şifreleme ürününe dahil olan; anahtarlanmış, simetrik bir Block Cipher (öbek şifreleyici)dir. Blowfish ile ilgili olarak şu ana kadar etkin bir şifre çözme analizi var olmasa da, artık AES ya da Twofish gibi daha büyük ebatlı öbek şifreleyicilerine daha fazla önem verilmektedir. Schneier; Blowfish'i bir genel kullanım algoritması olarak, eskiyen DES'ın yerini alması için ve diğer algoritmalarla yaşanan sorunlara çözüm olarak tasarlamıştır. O zamanlarda, birçok diğer tasarım lisanslı, patentle korunmakta ya da devlet sırrı olarak saklanmaktaydı.

Bruce Schneier, bunu şu şekilde ortaya koymaktadır :

“ Blowfish, patentsizdir ve tüm ülkelerde bu şekilde yer alacaktır. Algoritma genel kamusal alanda bulunmakta olup, herkes tarafından özgürce kullanılabilir.“

Tasarımın belirgin özellikleri anahtar-bağımlı S-boxes ve oldukça karmaşık anahtar çizelgesini içerir [39].

Blowfish şifrelemesinde 16 adımdan oluşan feistel ağı kullanılmaktadır. Bu ağdaki mesaj boyutu 64bit ve anahtar boyutu 32 ile 448 bit arasında değişkendir.



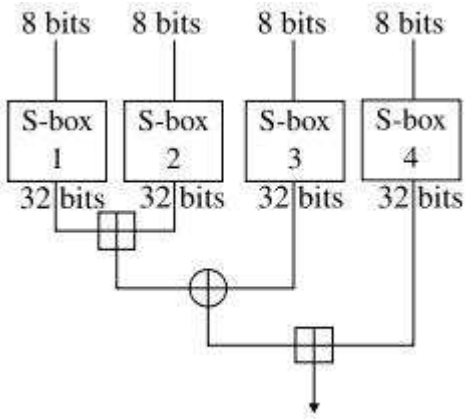
Şekil 3.12. Blowfish algoritması şeması [38]

Şekil 3.12'de toplam 16 adımdan oluşan blowfish algoritmasının şeması verilmiştir. Buna göre her adımda 32 bitlik işlem yapılmaktadır. Algoritma iki alt anahtar sırası (subkey array) tutar: 18-girişli P-sırası ve dört 256-girişli S-boxes bulunmaktadır.. S-boxes 8-bit girdi kabul eder ve 32-bit çıktı oluşturur. P-sırasının bir girişi her turda kullanılır ve son turdan sonra veri öbeğinin her bir yarısı geri kalan kullanılmamış iki P-girişinden biri tarafından XOR'lanır. [39]

Her adımda yer değiştirme dizilerinden birtanesi kullanılmaktadır. Son adımdan sonra veri bloğunun her iki yarısının yahutu (özel veyası (XOR)) alınmaktadır ve bu işlem sırasında arta kalan iki yerine koyma dizisi de kullanılmaktadır. (16 adımın her birisi için birer yerine koyma dizisi ve son adımda da 2 yerine koyma dizisi kullanıldığı için toplam 18 adet dizi bulunur)

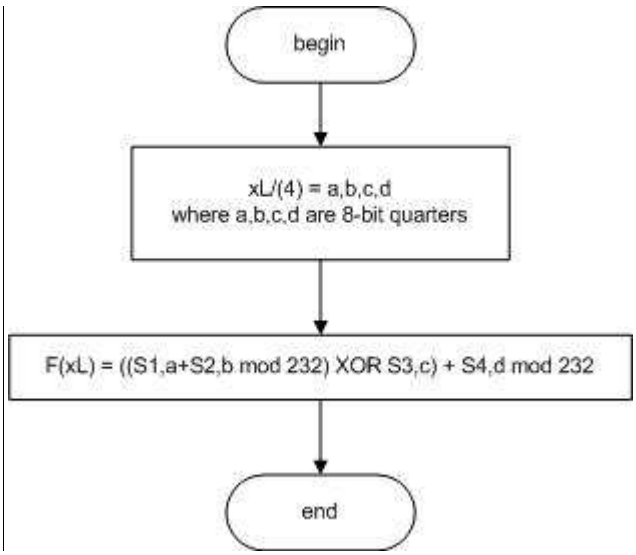
Şekil 3.13'de her F-Fonksiyonu için kullanılan yöntem verilmiştir. Buna göre mesajın yarısı olan 32 bit uzunluğundaki veri 4 adet 8bitlik parçaya bölünerek

aşağıdaki S-kutularına yerleştirilmekte ve her kutudan çıkan sonuç aşağıdaki şekilde işlenmektedir.



Şekil 3.13. Blowfish algoritması S-box şeması [38]

Sonuçlar 232 değerinde modulo işlemi alınarak yapılmakta ve 32 bitlik sonucu üretmek için özel veya (XOR) işlemine tâbî tutulmaktadır. Şifreleme yönteminin açılması için 17. ve 18. adımlarda kullanılan permütasyonun terse çevrilmesi ve her adımda bulunan permütasyonun sırasıyla geri gidilmesi yeterlidir.Algoritmada kullanılan F fonksiyonu Şekil 3.14'teki gibidir.



Şekil 3.14. Blowfish algoritması F fonksiyonu [39]

Bu yöntemde kullanılan permütasyon dizileri ve yerine koyma kutuları Π pi sayısından elde edilen sayılar ile oluşturulmaktadır. Bilindiği kadarıyla Π pi sayısının tekrar etmeyen yapısından dolayı yöntemin güvenli olduğu düşünülebilir [38].

2006 itibariyle Blowfish'in genel olarak bilinen etkin bir kriptanalizi mevcut değildir. 64 bit öbek büyüklüğü günümüzde çok kısa olarak düşünülse de; 232 den fazla veri öbeğini şifrelemek doğumgünü saldırıları nedeniyle plaintext hakkında bilgi sızmasını başlatabilir. Buna rağmen, Blowfish şu ana kadar güvenli görünmektedir. Kısa öbek büyüklüğü, e-posta gibi rutin kullanıcı uygulamaları konusunda ciddi endişeler yaratmasa da, Blowfish veri arşivleme gibi büyük şifresiz metinler konusunda uygun olmayabilir.

1996'da Serge Vaudenay kırmak için 28r + 1 şifresiz metin gerektiren (r turların sayısını ifade eder) bir bilinen-şifresiz metin saldırısı keşfetti. Daha da fazlası, yalnızca 24r + 1 bilinen-şifresiz metinlerle aynı saldırılarla kırılabilecek bir weak key (zayıf anahtar) sınıfı buldu. Bu saldırı tam 16-tur Blowfish 'e karşı kullanılamadığından; Vaudenay Blowfish'in indirgenmiş-turlu bir türevini kullandı. Vincent Rijmen, doktora tezinde, dört turdan fazlasını kıramayan ikincil-sıra diferansiyel saldırıyı sundu. Halen, tam 16-tur'u kırabilecek, brute-force search dışında bir yol bilinmemektedir.

2005'te Dieter Schmidt, Blowfish anahtar çizelgesini araştırdı ve üçüncü ve dördüncü turlar için alt anahtarların ilk 64 bitlik kullanıcı anahtarından bağımsız olduğunu ortaya koydu [39].

Blowfish, anahtar değiştirme dışında; geniş kullanımdaki en hızlı öbek şifreleyecilerden biridir. Her yeni anahtar metnin yaklaşık 4 KB'ını şifrelemek için ön-işleme (pre-processing) eşdeğeri gerektirir ki bu diğer öbek şifreleyicilerine kıyasla çok yavaştır. Bu, bazı uygulamalarda kullanımı engeller. Bir uygulamada, aslında bu bir faydadır: OpenBSD'de kullanılan parola-bozma (password-hashing) yöntemi Blowfish'ten türetilen yavaş anahtarlı bir algoritma kullanır; burada düşünce, gereken fazladan hesaba dayalı çabanın dictionary attack (sözlük saldırılarına) karşı koruma sağlamasıdır.

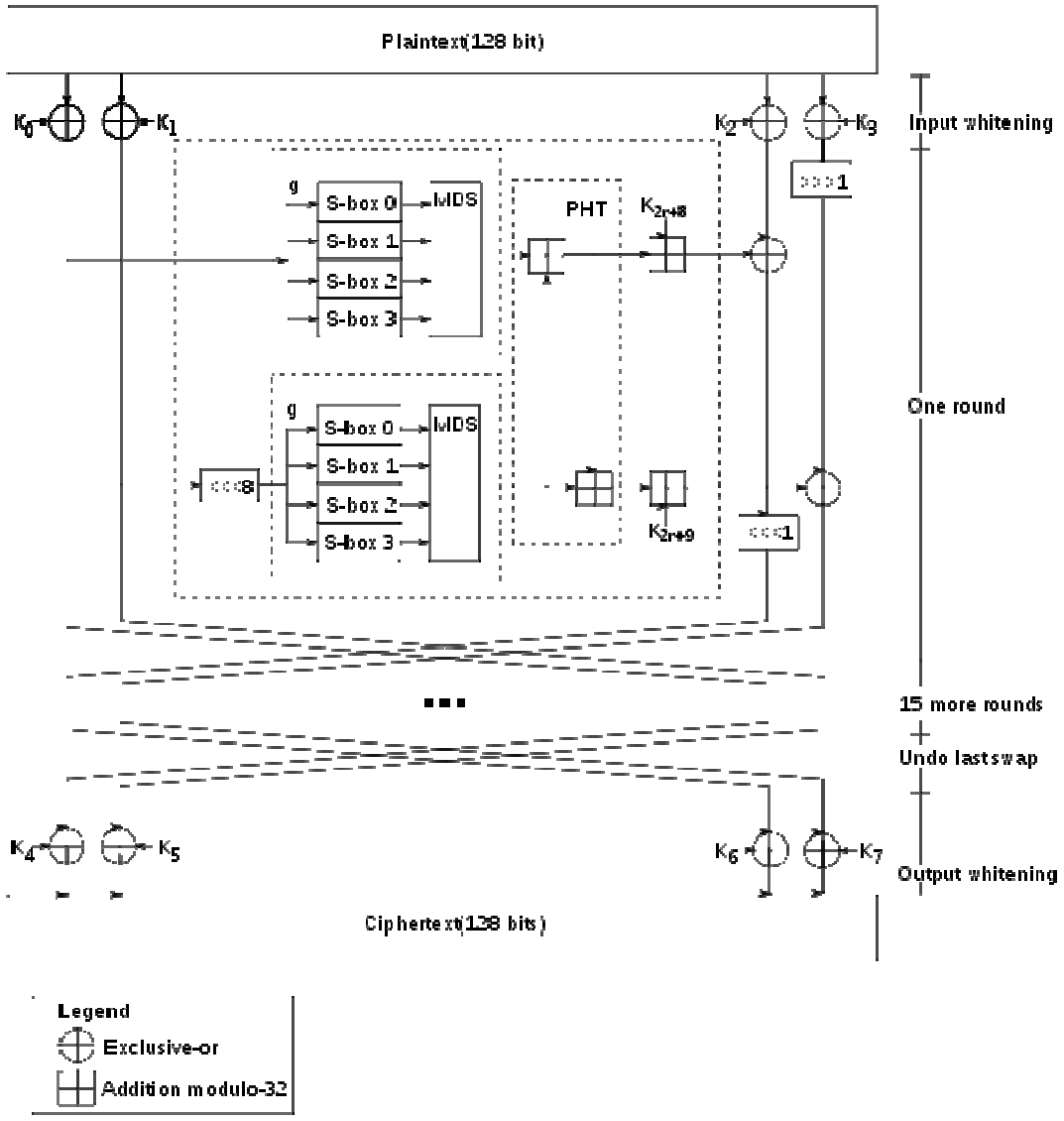
Bazı uygulamalarda, Blowfish, 4 kilobyte RAM dan biraz fazla, göreceli olarak büyük hafıza ayakizine sahiptir. Bu, eski daha küçük masa üstü ve diz üstü bilgisayarlar için bile bir sorun değildir, ancak, ilk smartcard (akıllı kartlar) gibi en küçük gömülü sistemlerde kullanımı engeller [39].

3.5. Twofish Algoritması

1998 yılında yayınlanan bu algoritma Bruce Schneier - John Kelsey -Doug Whiting - David Wagner - Chris Hall -Niels Ferguson tarafından yaratılmış ve analiz edilmiştir [40].

Simetrik blok şifreleme algoritmalarından ve AES finalistlerinden biridir ayrıca AES kadar hızlıdır. Aynı DES gibi Feistel yapısını kullanır. DES'den farklarından biri anahtar kullanılarak yaratılan değişken S-box (Substitution box – Değiştirme kutuları)'lara sahip olmasıdır. Ayrıca 128 bitlik düz metni 32 bitlik parçalara ayırarak işlemlerin çoğunu 32 bitlik değerler üzerinde gerçekleştirir.128,192 veya 256 bit lik değişken anahtar uzunluğuna sahiptir.AES'den farklı olarak eklenen 2 adet 1 bitlik rotasyon, şifreleme ve deşifreleme algoritmalarını birbirinden farklı yapmış, bu ise uygulama maliyetini arttırmış, aynı zamanda yazılım uygulamalarını %5 yavaşlatmıştır [41].

Algoritmanın çalışma şeması Şekil 3.15.'te verildiği gibidir.



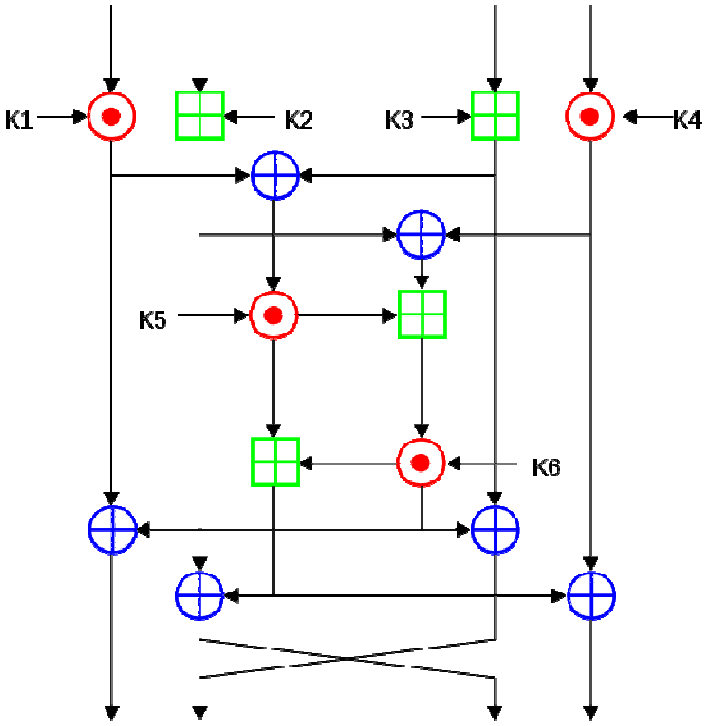
Şekil 3.15. Twofish algoritması çalışma şeması [41]

3.6. IDEA Algoritması

Açılımı “International Data Encryption Algorithm” olan IDEA 1990′da Xuejia Lai ve James Massey tarafından bulunmuştur. 1991 yılında tasarlanmış bir blok şifreleme algoritmasıdır. Bu algoritmaya DES yerine üretilen PES’in geliştirilmiş hali denebilir. Aynı zamanda Ascom tech adlı firmanın tescilli algoritmasıdır. IDEA algoritması PGP adlı program ile kullanılarak popüler bir hal almıştır. Bilinen en güçlü algoritmalardandır. İlk öncelerini PES adıyla anılan bu algoritma geçirdiği revizyonlar ertesinde şu andaki halini almıştır. İsviçre, Zürih’de geliştirilmiştir. Patenti Ascom Systec Ltd isimli bir firmaya aittir. Ancak ticari olmayan amaçlar için lisans ücreti talep edilmemektedir. PGP’nin temelini oluşturan 2 ana algoritmadan birisidir.

128 bitlik bir anahtar ile 52 adet 16 bitlik alt anahtarlar kullanılarak şifreleme işlemi gerçekleştirilmekdir. Şifrelenecek olan 64 bitlik metin 16 bitlik 4 eş parçaya bölünür .8 adımda yardımcı anahtarlar kulanılarak çarpma, toplama ve xor gibi matematiksel işlemler yardımıyla şifreleme işlemi gerçekleştirilir.

64-bit bloklar üzerinde 128-bit anahtar kullanan bir simetrik anahtar metodudur. IDEA patentli bir algoritmadır. RSA Laboratories tarafından geliştirilen RC5 algoritması değişken-uzunluklu anahtarlar ile çalışır. İhracat kısıtlamaları yüzünden farklı anahtar uzunluklarına ihtiyaç duyan uygulamalar için oldukça uygundur. 52 adet 16 bitlik yardımcı anahtar bulunmaktadır ve IDEA algoritması çeşitli matematiksel işlemlerin karışımından oluşmaktadır. Gerçekleşen işlemler mod 65536 da ve işaretsiz bildirimde kullanılmaktadır.Algoritmanın çalışma şeması Şekil 3.16.’da gösterildiği gibidir [43].



Şekil 3.16. IDEA algoritması çalışma şeması [42]

Yardımcı anahtar 52 tane anahtarın oluşturulmasında kullanılan 8 adetlik anahtarın kaydırılarak oluşturulması bir dezavantaj olarak yorumlanmaktadır. Buna rağmen matematiksel olarak herhangi bir eksiklik görülmemiştir. Bu algoritmanın avantaj ve dezavantajlarını aşağıdaki gibi listelenebilir;

Avantajları

- Algoritmalar hızlıdır
- Algoritmaların donanımla gerçeklenmesi kolaydır
- “Gizlilik” güvenlik hizmetini yerine getirir

Dezavantajları

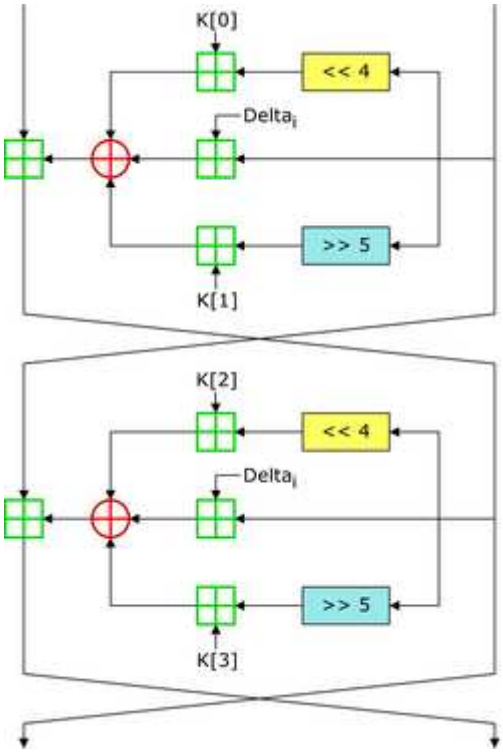
- Ölçeklenebilir değil
- Emniyetli anahtar dağıtımı zor
- “Bütünlük” ve “Kimlik Doğrulama” güvenlik hizmetlerini gerçeklemek zordur [42].

3.7. TEA Algoritması

Açılımı Tiny Encryption Algorithm' dir. Tiny Encryption Algorithm(TEA), Cambridge Bilgisayar Laboratuar'ında David Wheeler ve Roger Needham tarafından geliştirilmiştir. İlk olarak 1994 yılında Leuven'de Fast Software Algorithm (Hızlı Yazılım Şifreleme) atölyesinde sunulmuştur. Tiny şifreleme algoritması blok şifrelemeyi kullanır. TEA, "XOR, ADD ve SHIFT" gruplarını içeren karışık cebirsel işlemleri ve Fiestel ağını kullanan şifrelemedir. Basitliği ve çoğu şifreleme algoritmalarından daha kısa satırdan oluşan uygulamasıyla dikkate değerdir. Çok kısa olan kod boyutu ve basit algoritması sayesinde özellikle kod boyutunun oldukça sınırlı olduğu gömülü sistemlerde oldukça popüler olan bir şifreleme algoritmasıdır. TEA var olan en hızlı ve en etkili algoritmalardan birisidir. TEA, hafızada kaplanan yeri minimize etmek ve hızı maksimize etmek için geliştirilmiş bir algoritmadır. 64 bitlik bloklar kullanır. Bu 64 veri bloğunu 128 bitlik anahtar ile şifreler. 128 bitlik K anahtarı 32 bitlik bloklara bölünür. K = (K[0], K[1], K[2], K[3]) Değişebilmesine rağmen 64 adet Fiestel turu – 32 döngü tavsiye edilmektedir. (2 Fiestel turu = 1 döngü) Terminolojide 2′li yani çift tura Cycle = Döngü denmektedir. TEA, Shannon'un önerdiği ve güvenli bir blok şifreleme için gerekli olan karıştırma (confusion) ve yayılma (diffusion) özelliklerini sağlayan önemli bir şifreleme yöntemidir. Karıştırma(confusion) şifreli metin ve açık metin arasındaki ilişkiyi gizlemeyi amaçlarken, yayılma(diffusion) açık metindeki izlerin şifreli metinde sezilmemesini sağlamak için kullanılır. (Blok şifreler, Shannon'un önerdiği karıştırma ve yayılma tekniklerine dayanır.) Tam bir yayılma sağlar. Plain text, metin'de yapılan tek bir bit-in değişikliği cipher text, şifreli mesajda 32 bitlik değişikliğe sebep olur. Modern bir bilgisayardaki ya da çalışma alanındaki performansı etkileyicidir.

Kod çözme işlemi de temel olarak şifreleme süreci ile aynıdır. Decode sırasında şifreli mesaj, cipher text algoritmanın girdisi olarak kullanılır ama alt anahtarlar(sub keys) K[i] ters sırada kullanılır [44].

TEA için "2 Fiestel turu – 1 çevrim" yapısı aşağıdaki Şekil 3.17.'de gösterilmiştir.



Şekil 3.17. TEA algoritması çalışma şeması [44]

TEA en güvenli algoritmalardan biridir. Massey ve Xuejia Lai tarafından tasarlanan IDEA algoritması kadar güvenli olduğu söylenebilir. IDEA' da kullanılan aynı karışık cebirsel grupları kullanmasına rağmen daha basit ve de daha hızlıdır. Ayrıca TEA hiçbir kuruma ait olmamasına rağmen IDEA, İsviçre' de bulunan Ascom-Tech AG tarafından patentlenmiştir. TEA algoritmasının birkaç zayıf noktası vardır. Bunlardan en dikkat çekeni ise "equivalent key" durumudur. Bir şifreleme sisteminde, bir metnin K ve K* anahtarlarıyla şifrelenmesi sonucu aynı şifreli mesaj oluşuyorsa bu anahtarlara "equivalent key" denir. İyi dizayn edilmiş bir şifrelemede, equivalance – eşitlik sınıfına dahil olan her anahtarın plain text - cipher text dönüşümünü gösteren kesin bir haritası olması ve eşit, denk anahtarların bulunmaması gerekir. Böyle bir durumda 2^k eşitlik sınıfı bulunması beklenir. TEA yönteminde 128 bitlik anahtarlar vardır. Bu durumda 2^128 eşitlik sınıfı bulunması beklenirken araştırmalar 2^126 sınıf olduğunu göstermiştir. Bu zayıflığın sebebi ise tur fonksiyonudur. Yani etkili anahtar boyutu 126 bit olmaktadır. Yani aslında TEA 128 bitlik anahtarlar kullansa da 126 bitin sağlayacağı güvenliği sağlamaktadır. Bu

durum TEA şifrelemenin, blok şifrelemeye dayalı hash fonksiyon içinde kullanılmasını uygunsuz kılmaktadır. TEA ayrıca ilişkili anahtar saldırılarına karşı duyarlıdır. Bu zafiyetinden dolayı da TEA algoritmasının birkaç uyarlaması tasarlandı [44].

David Wheeler, 1994′de Cambridge Üniversitesi bilgisayar laboratuarında TEA algoritmasının şifreleme şablonunu, programını geliştirmiştir. Aşağıda C dilindeki şifreleme yapısı gösterilmiştir [45].

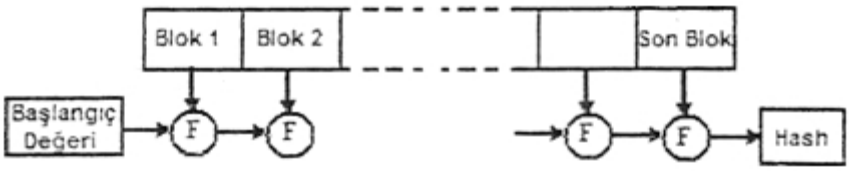
```
#include <stdint.h>

void encrypt (uint32_t* v, uint32_t* k) {
    uint32_t v0=v[0], v1=v[1], sum=0, i;           /* set up */
    uint32_t delta=0x9e3779b9;                     /* a key schedule
constant */
    uint32_t k0=k[0], k1=k[1], k2=k[2], k3=k[3];   /* cache key */
    for (i=0; i < 32; i++) {                       /* basic cycle
start */
        sum += delta;
        v0 += ((v1<<4) + k0) ^ (v1 + sum) ^ ((v1>>5) + k1);
        v1 += ((v0<<4) + k2) ^ (v0 + sum) ^ ((v0>>5) + k3);
    }                                              /* end cycle */
    v[0]=v0; v[1]=v1;
}

void decrypt (uint32_t* v, uint32_t* k) {
    uint32_t v0=v[0], v1=v[1], sum=0xC6EF3720, i;  /* set up */
    uint32_t delta=0x9e3779b9;                     /* a key schedule
constant */
    uint32_t k0=k[0], k1=k[1], k2=k[2], k3=k[3];   /* cache key */
    for (i=0; i<32; i++) {                         /* basic cycle
start */
        v1 -= ((v0<<4) + k2) ^ (v0 + sum) ^ ((v0>>5) + k3);
        v0 -= ((v1<<4) + k0) ^ (v1 + sum) ^ ((v1>>5) + k1);
        sum -= delta;
    }                                              /* end cycle */
    v[0]=v0; v[1]=v1;
}
```

3.8. Hash Algoritmaları

Hash fonksiyonu giriş olarak aldığı mesajlardan mesaj özeti ya da parmak izi olarak adlandırılan bir değer üretir. Açık metni önce bloklara ayırır. Bir başlangıç değerinden başlayarak bu başlangıç değerini her blok ile bir işleme tabi tutar. Bu başlangıç değerinin son hali mesaj özeti olur [13,37] (Şekil 3.18)



Şekil 3.18. Hash fonksiyonu

Bu özelliğinin haricinde kriptografıde hash fonksiyonlarını kullanabilmek için şu özelliklere sahip olmalıdır.

- Giriş olarak alman mesaj farklı uzunluklarda olabilir.
- Hash fonksiyonunun çıkışı sabit uzunlukta olmalıdır.
- Mesaj özetini hızlı bir şekilde hesaplayabilmelidir.
- Mesaj özetinden mesajı tekrar elde etmek imkansız olmalı bu özellikten dolayı hash fonksiyonlarına tek yol(one way) fonksiyonları adı verilir.
- Hash fonksiyonları kriptografide inkâr edememeyi sağlayan sayısal imzalar, doğrulama, mesaj bütünlüğü, etkilendirme uygulamalarında kullanılmaktadır
- Hash fonksiyonlarının çoğu sıkıştırma fonksiyonu adı verilen yinelemeli bir yapıya sahiptir. Mesaj önce blok zincirine dönüştürülür. Bir başlangıç değeri mesaj bloğu zincirinin her elemanı ile bir işleme tabi tutulur. Bu işlem bütün bloklar bitene kadar tekrarlanır. Elde edilen son değer mesajın özetidir.

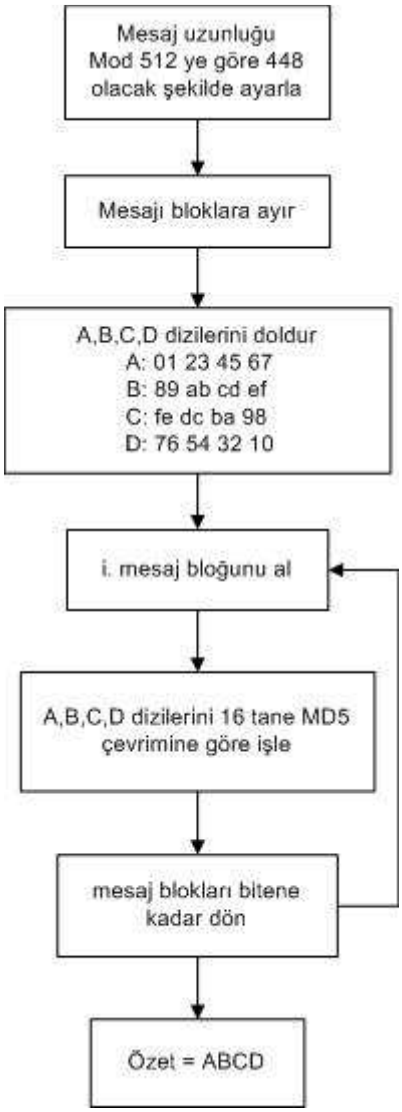
Hash fonksiyonlarının anahtarsız ve anahtarlı olmak üzere iki tipi bulunmaktadır.

3.8.1. Anahtarsız hash fonksiyonları

Anahtarsız hash fonksiyonlarında mesaj özeti sadece mesajdan elde edilir. MDC (değişiklik tarama kodu) adı verilen fonksiyonlar kriptografide yaygın biçimde kullanılır. MDC fonksiyonlarının temel amacı ise mesaj bütünlüğünü sağlamak için kullanılan bazı mekanizmalarda kullanılmak üzere bir mesajı temsil eden özet üretmektir. MD ailesi fonksiyonlar MD2, MD4, MD5 ve SHA-1 bunlardan bazılarıdır.

MD5 Hash Algoritması

MD5 algoritması b gibi keyfi uzunlukta bir mesajdan n bitlik bir mesaj özeti hesaplar (Şekil 3.13.) b nin sekizin katlan olması gerekmez. Algoritma şu adımlardan oluşur [36].



Şekil 3.19. MD5 algoritması

1.Adım: mesaj uzunluğunu ayarlama

Mesaj uzunluğu mod 512 ye göre 448 olacak şekilde ayarlanır. Doldurma işlemi bir 1 bir 0 ekleyerek uygun uzunluğa gelinceye kadar devam ettirilir.

Mesajın ekleme yapılmadan önceki uzunluğunun 64-bit gösterimi ekleme yapıldıktan sonraki haline eklenir. Bu 64 bitlik uzunluk değeri 32 bitlik kelimeler halinde eklenir. Low kısmı önce high kısmı sonra eklenir. Mesaj uzunluğu 2^64 ten fazla olması halinde uzunluk değerinin 64 bitlik low kısmı alınır. Bu eklemelerin

sonucunda mesaj 512 nin katı olan bir uzunluğa ulaşır. Bu da aynı zamanda 32 bitin katıdır. Mesaj şu şekilde 32 bitlik bloklara ayrılır.

M[0 ... N-l] (N 16 nm katıdır.)

2.Adım: MD tamponunun başlatılması

Mesaj özetini hesaplamak için A B C D isimlerinde 4 tane tampon kullanılır. Her biri 32 bitlik kaydedicilerdir. Bu kaydediciler low kısmı önce olmak üzere aşağıdaki değerlerle doldurulur.

word A: 01 23 45 67
word B: 89 ab cd ef
word C: fe dc ba 98
word D: 76 54 32 10

3. Adım: Mesajı 16 tane 32 bitlik bloklar halinde işleme

Mesaj blokları işlenirken aşağıdaki 4 yardımcı fonksiyon kullanılacaktır.

F(X,Y,Z) = X and Y or not(X)and Z
G(X,Y,Z) = X and Z or Y and not(Z)
H(X,Y,Z) = X xor Y xor Z
I(X,Y,Z) = Y xor (X or not(Z))
Bu adım 64 elemanlı T[1..64] tablosunu kullanılır. T[i]= 4294967296 * abs(sin(i)) nin tam kısmıdır.

Mesaj blokları şu algoritmaya göre işlenir.
For i = 0 to N/16-1 do
/* i. 16hk blok grubu X e kopyalanır. */
For j = 0 to 15 do
Set X[j] to M[i* 16+j].
end

/* A AA ya, B, BB ye , C CCye ve D DDye aktarılır. */

AA = A

BB = B

CC = C

DD = D

/* aşağıdaki 4 çevrim işletilir*/

/* 1. Çevrim */

/* [abed k s i] gösterimi aşağıdaki işlemi belirttiğini farz edelim.

a = b + ((a + F(b,c,d) + X[k] + T[i]) <<< s). */

/* Şu 16 işlem gerçekleştirilir. */

/* + sembolü mod.2^32 ye göre toplama ve <<< sembolü döngüsel kaydırmadır.*/

[ABCD 0 7 1] [DABC 1 12 2] [CDAB 2 17 3] [BCDA 3 22 4]

[ABCD 4 7 5] [DABC 5 12 6] [CDAB 6 17 7] [BCDA 7 22 8]

[ABCD 8 7 9] [DABC 9 12.10] [CDAB 10 17 11] [BCDA 11 22 12]

[ABCD 12 7 13] [DABC 13 12 14] [CDAB 14 17 15] [BCDA 15 22 16]

/* 2. Çevrim */

/* [abed k s i] gösterimi aşağıdaki işlemi belirttiğini farz edelim.

a = b + ((a + G(b,c,d) + X[k] + T[i]) <<< s). */

/* Şu 16 işlem gerçekleştirilir. */

[ABCD 1 5 17] [DABC 6 9 18] [CDAB 11 14 19] [BCDA 0 20 20]

[ABCD 5 5 21] [DABC 10 9 22] [CDAB 15 14 23] [BCDA 4 20 24]

[ABCD 9 5 25] [DABC 14 9 26] [CDAB 3 14 27] [BCDA 8 20 281

[ABCD 13 5 29] [DABC 2 9 30] [CDAB 7 14 31] [BCDA 12 20 32]

/* 3. Çevrim */

/* [abcd k s t] gösterimi aşağıdaki işlemi belirttiğini farz edelim.

a = b + ((a + H(b,c,d) + X[k] + T[i]) <<< s). */

/* Şu 16 işlem gerçekleştirilir. */

[ABCD 5 4 33] [DABC 8 11 34] [CDAB 11 16 35] [BCDA 14 23 36]

[ABCD 1 4 37] [DABC 4 1138] [CDAB 7 16 39] [BCDA 10 23 40]

[ABCD 13 4 41] [DABC 0 11 42] [CDAB 3 16 43] [BCDA 6 23 44]

[ABCD 9 4 45] [DABC 12 11 46] [CDAB 15 16 47] [BCDA 2 23 48]

/* 4. Çevrim */

/* [abed k s t] gösterimi aşağıdaki işlemi belirttiğini farz edelim.

a = b + ((a + I(b,c,d) + X[k] + T[i]) <<< s). */

/* Şu 16 işlem gerçekleştirilir. */

[ABCD 0 6 49] [DABC 7 10 50] [CDAB 14 15 51] [BCDA 5 21 52]

[ABCD 12 6 53] [DABC 3 10 54] [CDAB 10 15 55] [BCDA 1 21 56]

[ABCD 8 6 57] [DABC 15 10 58] [CDAB 6 15 59] [BCDA 13 21 60]

[ABCD 4 6 61] [DABC 11 10 62] [CDAB 2 15 63] [BCDA 9 21 64]

/* A, B, C, D kaydedicilerinin eski değerleri şu anki değerlerine eklenir */

A = A + AA

B = B + BB

C=C+CC

D = D + DD

end /* döngü sonu */

4. Adım: Çıkış Değeri

Mesaj özeti, A mesaj özetinin low kısmında, D ise high kısmında yer almak , A,B,C,D kaydedicilerinin son değerlerinin birleşiminden elde edilir [13].

3.8.2. Anahtarlı hash fonksiyonları

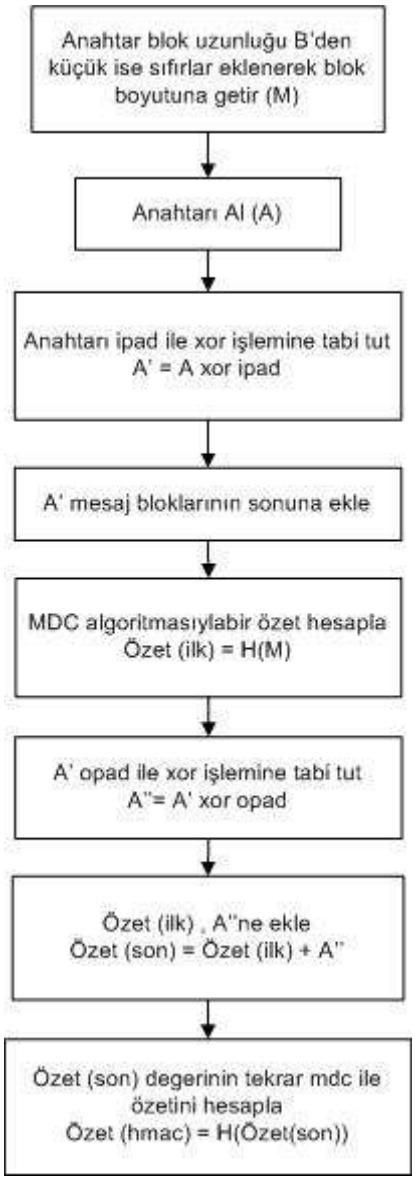
Anahtarlı hash fonksiyonlarında mesaj özeti hem mesaja hem de bir anahtara göre hesaplanır. Yaygın olarak kullanılanları MAC - mesaj doğrulama kodu - adı verilen anahtarlı hash fonksiyonlarıdır. MAC fonksiyonları, ek bir mekanizma kullanmadan bir mesajın bütünlüğünü sağlamak için kullanılırlar. MD5-HMAC ve SHA-1 HMAC bu algoritmalardan bazılarıdır.

HMAC

HMAC tipi algoritmalar MDC'ye dayalı MAC algoritmasıdır. Bir gizli anahtar, anahtarsız MDC hash algoritması ve ipad ile opad adı verilen özel sabitler kullanır. Mesaj özetinin uzunluğu kullanılan MDC algoritmasına bağlıdır. Anahtar ise 64 bayttan büyük olmaz, ipad ve opad sabitleri anahtar ile XORlanarak MDC hash fonksiyonunu hesaplar. MDC olarak genelde MD5 ve SHA-1 kullanılır.

HMAC algoritması H ile temsil edilen bir MDC tipi hash algoritması ve bir anahtar gerektirir. Kullanılan MDC algoritmasına göre, mesaj özetinin büyüklüğü L kadardır ve mesaj B ile temsil edilen bloklara aynhr(örneğin MD5 için L=16, B=64). Kullanılan anahtar B ile L arasında herhangi bir uzunlukta olabilir. Ayrıca ipad ve opad adı verilen iki özel sabit, algoritmada kullanılmaktadır [36].

ipad, 0x36 bayrının B defa tekrarıdır, opad ise 0x5C baytmm B defa tekrarıdır.
HMAC algoritmasının yapısı Şekil 3.14'teki gibidir.



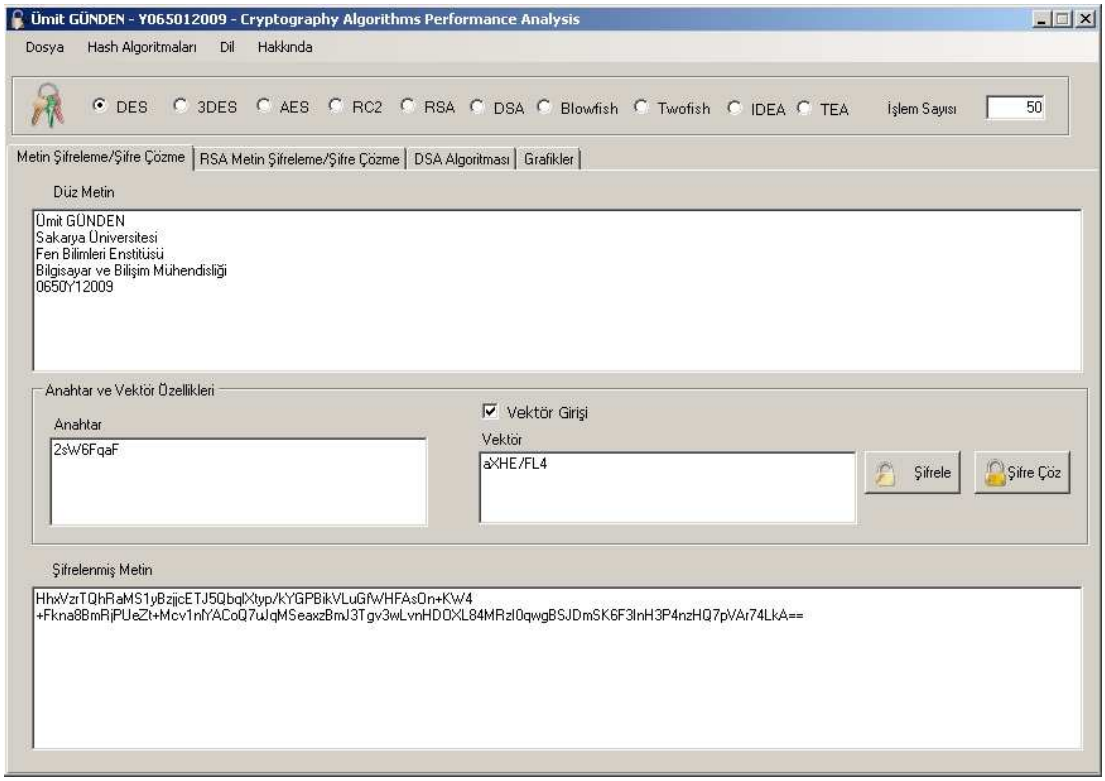
Şekil 3.20. HMAC algoritması

1. Eğer anahtar K'mn blok uzunluğu B den küçük ise sıfırlar eklenerek blok boyutuna getirilir.
2. Anahtar ipad ile xor işlemine tabi tutulur.
3. Adımın sonucu mesaj bloklarını sonuna eklenir.
4. Kullanılan MDC algoritmasıyla bir özet hesaplanır.
5. Adımdaki hesaplanan değer opad ile xor işlemine tabi tutulur.
6. Adımdaki hash özeti 5. adımda hesaplanan değere eklenir.
7. 6 adımda bulunan değerden mdc algoritmasıyla tekrar bir özet hesaplanarak hmac çıkışı olarak kullanılır.

BÖLÜM 4. ŞİFRELEME ALGORİTMALARININ PERFORMANS ANALİZİ

Gerek kişisel bilgilerin gerekse kurumsal bilgilerin güvenliği bilginin internet gibi araçlarla çokça paylaşıldığı bir çağda artan bir ihtiyaç olmuştur. Bu gereksinimle beraber kriptografı eğitimi dünyanın her yerinde hızla büyüyen bir sektör haline geldi.

Bu bölümde şifreleme algoritmalarını çalışmalarım incelenmesi için gerçekleştirilen programın yapısı ve kullanımı açıklanacaktır. Şekil 4.1 'de programın genel görünümü verilmiştir. Programın hazırlanmasında Visual C#.Net 2008 kullanılmıştır.



Şekil 4.1. Programın genel görünümü

4.1. Programın Amacı

Geliştirilen uygulamanın (Şekil 4.l) temel olarak şu kullanım amaçları vardır.

- Bir metnin istenilen tipteki algoritmayı kullanarak nasıl şifrelendiğini veya çözüldüğünü göstermek.
- Seçilen şifreleme algoritmasının metni şifrelerken ya da çözerken ne kadar zaman harcadığını, işlemci ve hafıza kullandığını hesaplamak. Diğer bir deyişle algoritmaların performanslarını ölçmek.
- Kullanılan algoritmaları birbirleriyle işlem hızı bakımından karşılaştırarak performans sıralamasını yapmak.

4.2. Kullanılan Algoritmalar

Programda kullanılan algoritmalar DES, AES, 3DES, RC2, DSA, Blowfish, Twofish, IDEA, TEA ve RSA algoritmalarıdır. Ayrıca Hash fonksiyonlarının genel olarak gösterimi ile ilgili bir bölümde tasarlanmıştır.

4.3. Tasarlanan Uygulama

Şifreleme algoritmalarının performans analizi için tasarlanmış olan uygulama 8 bölümden oluşmaktadır. (Şekil 4.1.)

- Dosya
- Metin Şifreleme / Şifre Çözme
- RSA Metin Şifreleme / Şifre Çözme
- DSA Algoritması
- Grafikler
- Hash Algoritmaları
- Dil
- Hakkında

Dosya bölümünde; daha önce yapılmış olan performans testlerinin sonuçlarının tekrar yüklenerek , programdan analizi yapılabilmektedir. Metin şifreleme ve şifre çözme bölümününde; simetrik şifreleme algoritmaları için şifreleme ve şifre çözme için gerekli işlemler yapılmaktadır. RSA metin şifreleme ve şifre çözme bölümünde; RSA asimetrik şifreleme algoritması kullanılarak anahtar üretimi, şifreleme ve şifre çözme işlemleri yapılmaktadır. DSA Algoritması bölümünde; algoritmanın dijital sertifika işlemlerinde kullanımı ile ilgili olarak, seçili dosya için imza oluşturma ve imza onay işlemleri yapılmaktdır. Grafikler bölümünde; yapılan performans testlerinin karşılaştırmalı sonuçları analiz edilebilmektedir. Hash algoritmaları bölümünde; hash algoritmalarının genel kullanımı ile ilgili kısım yer almaktadır. Dil bölümünden program dili değiştirilebilmektedir. Hakkında bölümünde ise programı hazırlayan kişi ile ilgili bilgiler yer almaktadır. Alt bölümlerde bu kısımlar daha ayrıntılı olarak anlatılmıştır.

4.3.1. Dosya bölümü

Tasarlanan programda her algoritma için şifreleme yada şifre çözme işlemi yapıldığında , o algoritma için elde edilmiş sonuç değerleri, programın bulunduğu bölümde oluşturulmuş “Result” klasöründe bir dosya oluşturularak o dosyaya kayıt edilmektedir. Kayıt edilen bu dosyalar , performans analiz sonuçlarının elde edilmesinde kullanılmaktadır. Analiz sonuçları program aracılığı ile daha sonra tekrar görüntülenmek istenildiğinde “Dosya” bölümünden faydalanılmaktadır. (Şekil 4.2.)

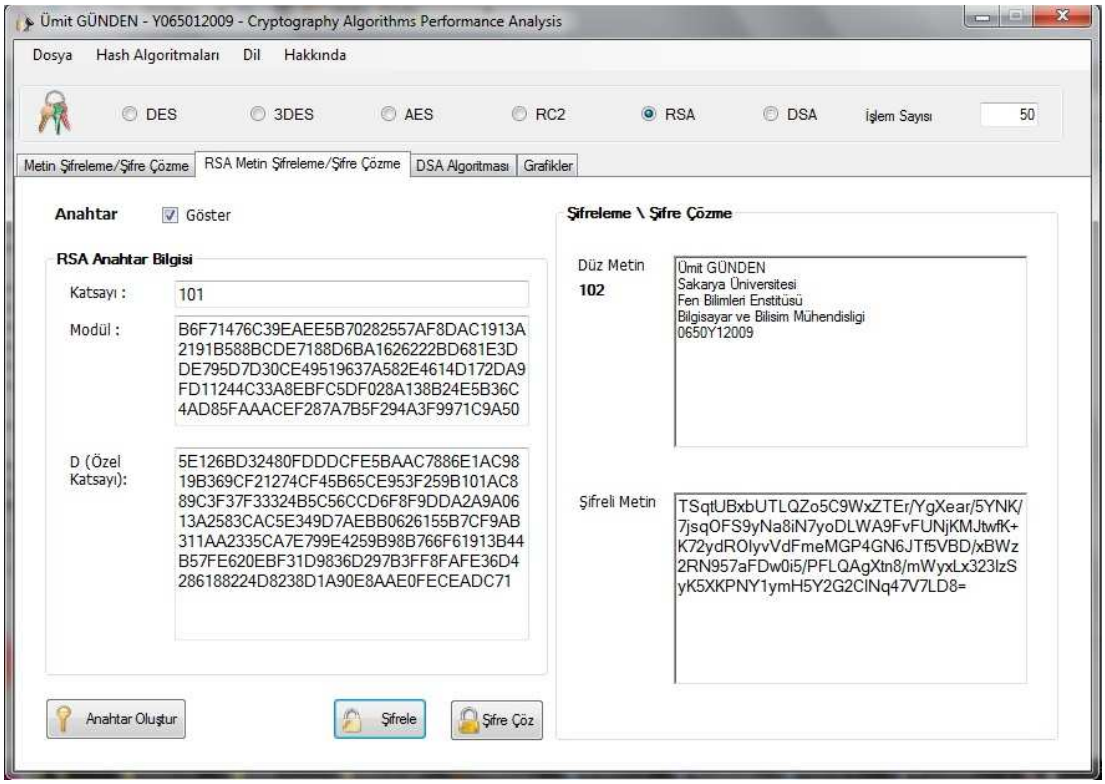
Şekil 4.2. Dosya bölümü

4.3.2. Metin şifreleme / şifre çözme

Tasarlanan programda simetrik algoritmalar ile ilgili şifreleme ve şifre çözme işlemleri bu bölümde yer almaktadır. DES, AES, 3DES, RC2, Blowfish, Twofish, IDEA, TEA bu bölümde kullanılmış olan simetrik algoritmalardır. Algoritmalar için gerekli initialization vektör ve anahtar üretimi her algoritma için otomatik olarak yapılmaktadır. Verilen metin, bu anahtar ve vektör bilgileri kullanılarak şifrelenmektedir. Şifreli metinden düz metnin elde edilmesi işlemi de benzer şekilde vektör ve anahtar bilgilerinin arayüzden girilmesinden sonra sağlanmaktadır. Şifreleme yada şifre çözme işlemi belirtilen iterasyon sayısında yapılmaktadır. İterasyon sayısı kullanıcı tarafından bu bölümde belirlenebilmektedir. Bu bölüme ait ekran görüntüsü Şekil 4.1. de gösterilmiştir.

4.3.3. RSA metin şifreleme / şifre çözme

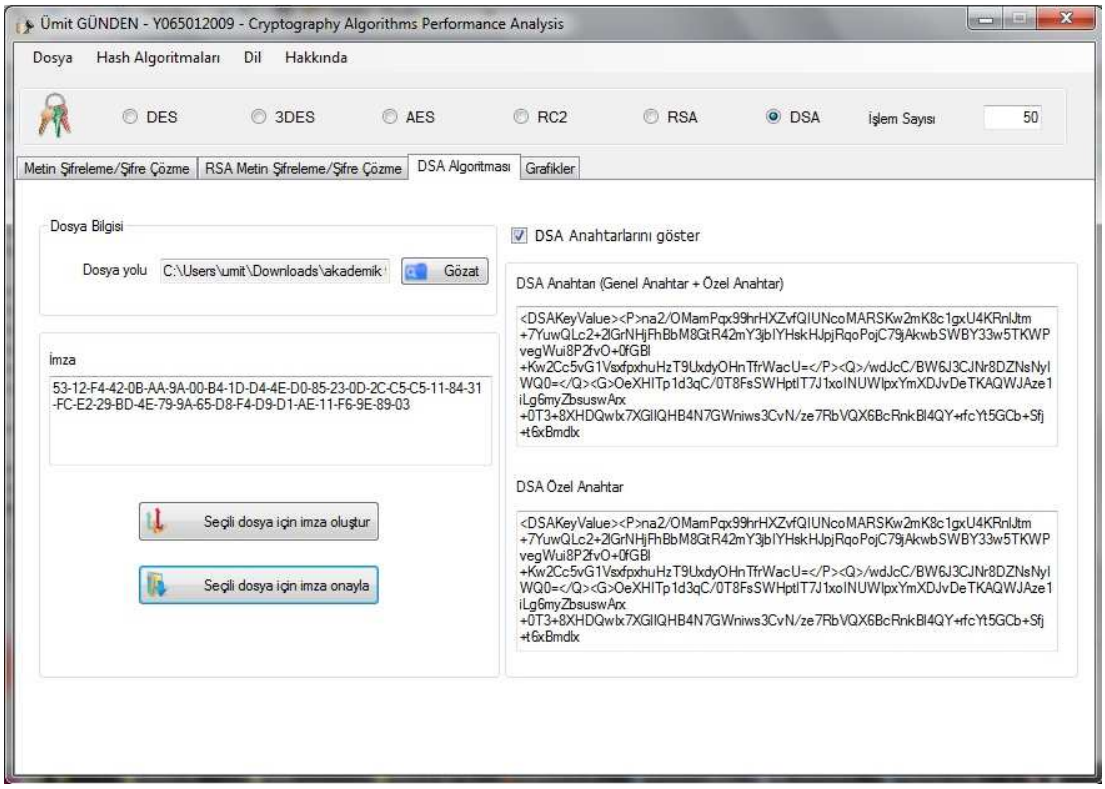
RSA asimetrik şifreleme algoritması için anahtar üretimi , şifreleme ve şifre çözme işlemleri bu bölümde yer almaktadır. Üretilen anahtara ait katsayı, D (özel katsayı) ve modül değerleri gösterilmektedir. Şifreleme ve şifre çözme işlemlerinde bu değerler kullanılmaktadır. Algoritma ile ilgili ayrıntılı bilgiler daha önceki bölümlerde verilmiştir. (Bkz Bölüm 2.1). Bölüm ile ilgili ekran görüntüsü aşağıda verilmiştir. (Şekil 4.3.)



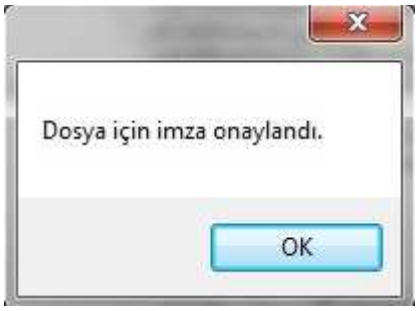
Şekil 4.3.RSA algoritması

4.3.4. DSA algoritması

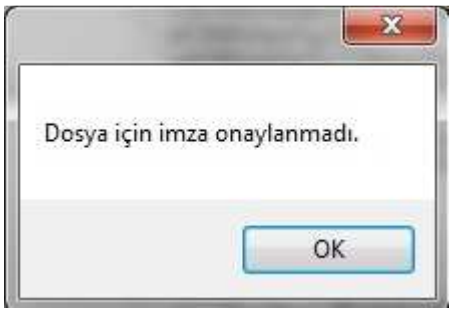
Bu bölümde DSA algoritmasının kullanımı ile ilgili işlemler yer almaktadır. DSA algoritması genel olarak dijital sertifika işlemlerinde kullanılmaktadır. Algoritma ile ilgili ayrıntılı bilgiler daha önceki bölümlerde verilmiştir. (bkz Bölüm 2.2.) İmza oluşturulacak olan dosya “Dosya Bilgisi” bölümünden seçilmektedir. Seçilen dosya için oluşturulmuş imza, public ve private anahtar bilgileri görüntülenebilmektedir. Seçili olan dosyaya ait imza onaylanması halinde Şekil 4.5. teki gibi onay mesajı gösterilmektedir. Onaylanmaması durumunda ise Şekil 4.6. daki mesaj gösterilmektedir. Bu şekilde o dosyaya ait olan imza onay işlemleri yapılmaktadır. Bölüm ile ilgili genel ekran görüntüsü Şekil 4.5. ‘te gösterildiği gibidir.



Şekil 4.4. DSA algoritması



Şekil 4.5. DSA algoritması olumlu onay mesajı



Şekil 4.6. DSA algoritması olumsuz onay mesajı

4.3.5. Grafikler

Şifreleme algoritmaları için anahtar üretimi, metin şifreleme ve şifre çözme işlemleri yapıldıktan sonra elde edilen değerler kullanılarak işlem zamanı, işlemci kullanımı ve hafıza kullanımı grafikleri oluşturulmaktadır. Her algoritmanın bu işlemler için harcamış oldukları zaman ayrıca bir tablo halinde görüntülenmektedir. Bölüm ile ilgili ekran görüntüsü Şekil 4.7. ‘de gösterilmektedir.

Şekil 4.7. Şifreleme algoritmalarının performans grafikleri

4.3.6. Hash algoritmaları

Bu bölümünde MD5, SHA, SHA256, SHA384, SHA512 hash algoritmaları kullanılarak şifreleme yapılabilmektedir. Tez çalışmasında hash algoritmalarının performans analizi incelenmemiştir. Genel kullanımı ile ilgili bilgi verilmiş ve tasarlanan uygulama da bu algoritmalar ile ilgili bir bölüm eklenmiştir. (Şekil 4.8.)

Şekil 4.8. Hash algoritmaları

4.3.7. Dil seçenekleri

Tasarlanan programda Türkçe ve İngilizce olmak üzere 2 adet dil seçeneği bulunmaktadır. İstenilirse programın dili bu bölümden değiştirilebilmektedir.

4.3.8. Hakkında

Bu bölümde programı hazırlayan kişi ile ilgili bilgiler yer almaktadır. (Şekil 4.9.)

Şekil 4.9. Hakkında

Kriptografık uygulamalarda kullanılmak üzere bir şifreleme algoritması seçilirken; bazen şifreleme ve çözme işlemlerindeki hızı, bazen algoritmanın kullandığı hafıza miktarı, bazen de algoritmanın saldırılara karşı sağlamlığı önemli olur. Algoritmaların şifreleme ve çözmede harcadığı zaman ve hafıza miktarını ölçmeye performans analizi, saldırılara karşı dayanıklılığını ölçmeye ise kriptanaliz adı verilir. Bu tez çalışmasında şifreleme algoritmaları kriptanaliz bakımından değerlendirilmemiş , performans analizleri ele alınmıştır.

4.4. Şifreleme Algoritmalarının Performans Analizi

Bu bölümde şifreleme algoritmalarının performans analizi yapılmıştır. Performans değerlendirmeleri, daha çok algoritmaların şifreleme, çözme hızları yani zaman karmaşıklıkları üzerine yoğunlaşmıştır. Hafıza karmaşıklığı ve CPU kullanımı da değerlendirilmiştir.

Burada kullanılan performans analizi yöntemi ise deneysel analiz yöntemidir. Bu yöntem karmaşık algoritmaların değerlendirilmesinde tercih edilen güvenilir bir yöntemdir.

Algoritmaların işlemci, bellek kullanımları ve şifreleme için harcadıkları zaman bilgisayar donanımına göre değişiklik gösterebilmektedir. Bu nedenle tasarlanan program farklı donanıma sahip bilgisayarlarda ayrı ayrı test edilmiş ve sonuçlar değerlendirilmiştir.

4.4.1. Performans test sonuçları

Şifreleme Algoritmalarının performans analizi farklı donanım özellikleri ve işletim sistemlerine sahip bilgisayarlarda test edilmiştir. Tablo 4.1.’de kullanılan

bilgisayarların donanım özellikleri ve işletim sistemi gösterilmiştir. Performans, her algoritmayla farklı karakter uzunluğuna sahip metinlerin 50 defa şifrelenip çözülmesiyle ölçülmüştür. Bulunan sonuçlar 50 iterasyonun ortalamasıdır.

Tablo 4.1. Test bilgisayarlarının özellikleri

	Test Bilgisayarı-1	Test Bilgisayarı-2	Test Bilgisayarı-3
CPU	Intel(R) Core(TM)2 Duo CPU T7500 @ 2.20GHz	Intel(R) Pentium(R) M Processor 1.70GHz	AMD Athlon(tm) XP 2800+
RAM	2048 MB	768 MB	768 MB
İşletim Sistemi	Microsoft Windows 7 Ultimate	Microsoft Windows XP Professional	Microsoft Windows XP Professional

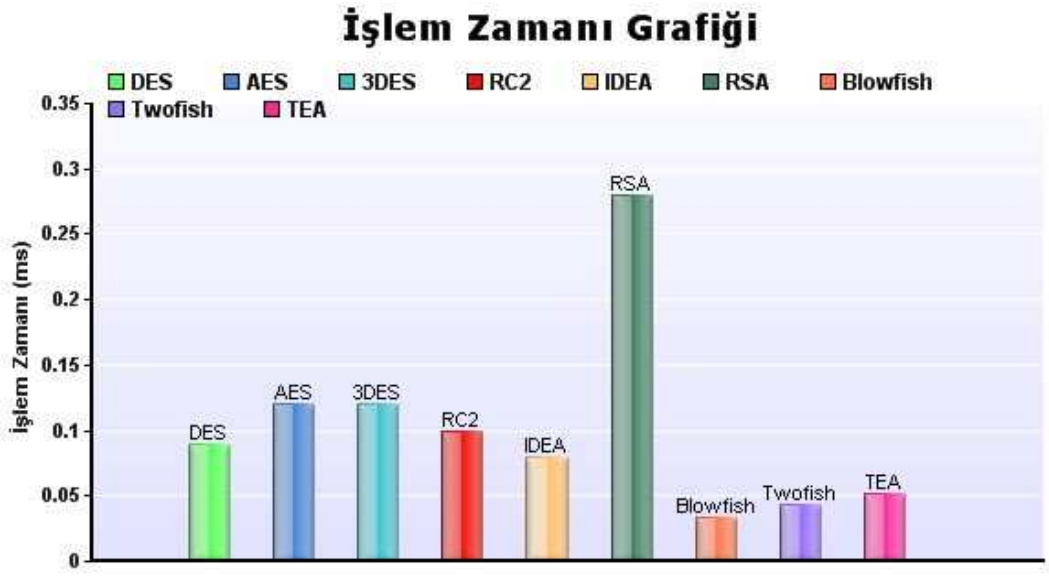
4.4.1.1. 102 Byte'lık metnin şifrelenmesi

102 Byte karakter uzunluğuna sahip metnin şifrelenmesi sonucunda elde edilen işlem zamanı, CPU kullanımı ve RAM kullanımına ait tüm değerler Tablo 4.2. 'de verilmektedir. Tabloda yer alan işlem zamanına ait değerler milisaniye, işlemci kullanımına ait değerler yüzde (%) , hafıza kullanımına ait değerler ise megabyte (MB) türünden ifade edilmiş değerlerdir. Ayrıca metnin şifrelenmesi sonucu elde edilen işlem zamanına ait değerler karşılaştırmalı bir şekilde Şekil 4.10. , Şekil 4.11. ve Şekil 4.12.' te grafiksel olarak gösterilmiştir. Grafikte gösterilmiş olan bu değerler de milisaniye türünden değerlerdir.

Tablo 4.2. 102 Byte ‘lık metni şifreleme işlemi performans değerleri

Test Bilgisayarı	Algoritma	İşlem Zamanı (ms)	RAM Kullanımı (MB)	CPU Kullanımı (%)
Bilgisayar 1	DES	0,29	38,77	12,83
	AES	0,34	39,67	13,88
	3DES	0,60	44,45	18,58
	RC2	0,48	42,25	15,05
	RSA	0,82	51,69	27,29
	IDEA	0,28	37,43	12,38
	TEA	0,13	23,16	6,0
	Blowfish	0,10	19,14	5,2
	Twofish	0,12	21,07	5,4
Bilgisayar 2	DES	0,09	31,37	3,75
	AES	0,12	30,77	4,00
	3DES	0,12	39,44	7,00
	RC2	0,10	35,20	5,00
	RSA	0,28	38,05	8,33
	IDEA	0,108	37,01	4,5
	TEA	0,052	22,17	3,5
	Blowfish	0,034	17,77	3,5
	Twofish	0,044	19,60	3,4
Bilgisayar 3	DES	0,40	34,13	9,33
	AES	0,50	32,99	9,82
	3DES	1,62	38,49	14,00
	RC2	0,79	36,21	11,00
	RSA	7,80	41,25	16,00
	IDEA	0,069	25,88	6
	TEA	0,09	27,67	8,2
	Blowfish	0,13	31,02	8,8
	Twofish	0,08	20,8	7

Şekil 4.10. 102 Byte'lık metni şifreleme bilgisayar-1 işlem zamanı



Şekil 4.11. 102 Byte'lık metni şifreleme bilgisayar-2 işlem zamanı

Şekil 4.12. 102 Byte'lık metni şifreleme bilgisayar-3 işlem zamanı

102 Byte karakter uzunluğuna sahip metnin şifreli halinin, şifreleme işleminde kullanılan anahtar ve IV değerleri kullanılarak şifre çözme işlemi sonucunda elde edilen işlem zamanı, CPU kullanımı ve RAM kullanımı değerleri de Tablo 4.3.'te verildiği gibidir.

Tablo 4.3. 102 Byte'lık metnin şifre çözme işlemi performans değerleri

Test Bilgisayarı	Algoritma	İşlem Zamanı (ms)	RAM Kullanımı (MB)	CPU Kullanımı (%)
Bilgisayar 1	DES	0,28	40,98	12,99
	AES	0,35	42,56	12,89
	3DES	0,51	48,14	20,13
	RC2	0,48	44,93	16,96
	RSA	13,04	52,03	36,00
	IDEA	0,27	38,44	12,52
	TEA	0,14	32,51	6,5
	Blowfish	0,11	29,14	5,0
	Twofish	0,13	31,17	6,0

Tablo 4.3.(Devam) 102 Byte'lık metnin şifre çözme işlemi performans değerleri

Bilgisayar 2	DES	0,10	32,49	6,00
	AES	0,11	31,27	6,00
	3DES	0,14	38,29	8,00
	RC2	0,12	34,56	5,00
	RSA	28,49	45,60	12,00
	IDEA	0,09	29,14	5,5
	TEA	0,06	22,57	3,5
	Blowfish	0,04	20,16	3
	Twofish	0,05	21,17	3,5
Bilgisayar 3	DES	0.40	33.85	10.00
	AES	0.43	34.83	12.00
	3DES	0.51	39.30	15.00
	RC2	0.44	35.41	13.00
	RSA	33.27	44.73	16.00
	IDEA	0,074	16,14	5,0
	TEA	0,14	21,82	6,0
	Blowfish	0,21	27,71	8,0
	Twofish	0,17	24,32	7,0

4.4.1.2. 1 MB'lık metnin şifrelenmesi

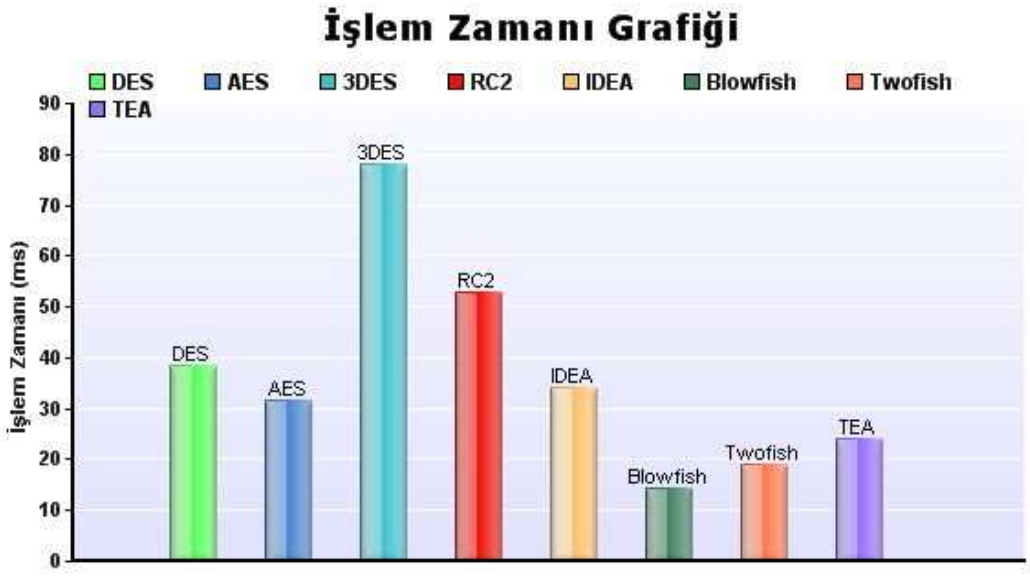
Asimetrik şifreleme algoritmaları boyutu uzun dosyaların şifrelenmesi için genel olarak uygun değildir. RSA algoritması 112 byte' tan daha uzun metinlerin şifrelenmesinde kullanılamamaktadır.

1 Megabyte (MB) karakter uzunluğuna sahip metnin şifrelenmesi sonucunda elde edilen işlem zamanı, CPU kullanımı ve RAM kullanımına ait tüm değerler Tablo 4.4. 'de verildiği gibidir. Tabloda yer alan işlem zamanına ait değerler milisaniye, işlemci kullanımına ait değerler yüzde (%) , hafıza kullanımına ait değerler ise megabyte (MB) türünden ifade edilmiş değerlerdir. Ayrıca metnin şifrelenmesi sonucu elde edilen işlem zamanına ait değerler karşılaştırmalı bir şekilde Şekil 4.13. , Şekil 4.14.

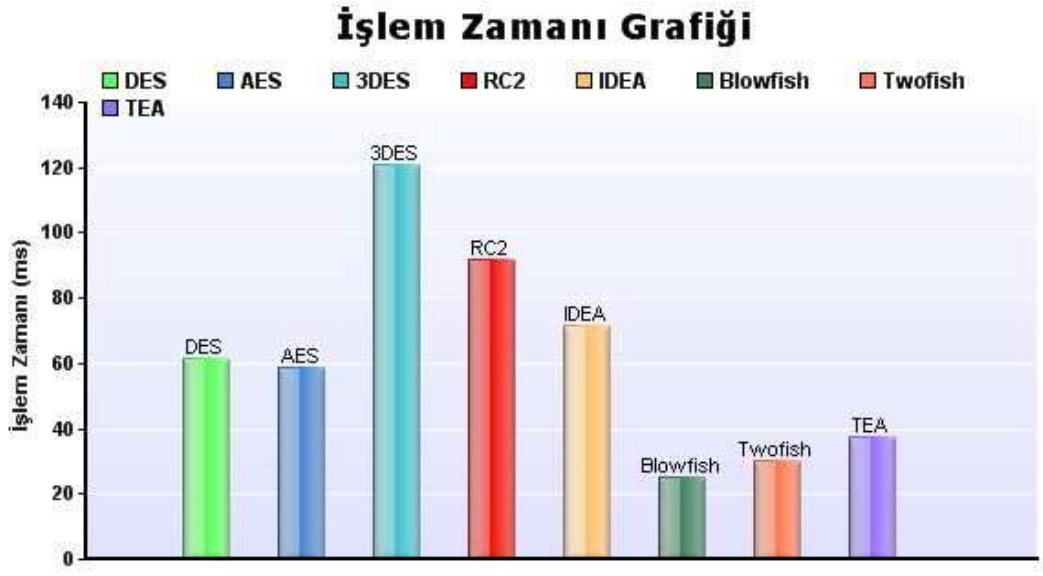
ve Şekil 4.15.’ te grafiksel olarak gösterilmiştir. Grafikte gösterilmiş olan bu değerler de milisaniye türünden değerlerdir.

Tablo 4.4. 1 MB’lık Metni şifreleme işlemi performans değerleri

Test Bilgisayarı	Algoritma	İşlem Zamanı (ms)	RAM Kullanımı (MB)	CPU Kullanımı (%)
Bilgisayar 1	DES	38,55	82,69	20,73
	AES	31,68	81,78	18,82
	3DES	78,17	93,37	33,92
	RC2	52,99	91,06	26,36
	IDEA	34,12	77,42	18,42
	TEA	24,16	39,06	10,02
	Blowfish	14,27	28,11	7,1
	Twofish	19,17	35,37	8,5
Bilgisayar 2	DES	61,34	35,87	6,00
	AES	59,05	34,77	6,00
	3DES	121,18	74,60	11,00
	RC2	91,91	68,02	9,00
	IDEA	71,61	46,71	7,5
	TEA	37,44	27,18	4,0
	Blowfish	25,21	21,17	3,5
	Twofish	30,07	22,54	3,75
Bilgisayar 3	DES	83,09	69,35	8,00
	AES	84,58	73,84	8,00
	3DES	120,35	84,97	14,00
	RC2	95,85	80,64	10,00
	IDEA	14,33	51,96	5,0
	TEA	18,47	55,60	7,0
	Blowfish	29,02	62,53	8,0
	Twofish	15,61	53,81	5,0



Şekil 4.13. 1 MB'lık metni şifreleme bilgisayar-1 işlem zamanı



Şekil 4.14. 1 MB'lık metni şifreleme bilgisayar-2 işlem zamanı

Şekil 4.15. 1 MB'lık metni şifreleme bilgisayar-3 işlem zamanı

1 MB karakter uzunluğuna sahip metnin şifrelenmesi sonucunda elde edilen şifreli halinin, aynı anahtar ve IV değerleri kullanılarak şifre çözme işlemi sonucunda elde edilen işlem zamanı, CPU kullanımı ve RAM kullanımı değerleri Tablo 4.5.' de verildiği gibidir.

Tablo 4.5. 1 MB'lık metnin şifre çözme işlemi performans değerleri

Test Bilgisayarı	Algoritma	İşlem Zamanı (ms)	RAM Kullanımı (MB)	CPU Kullanımı (%)
Bilgisayar 1	DES	25,69	77,39	30,75
	AES	25,52	74,05	24,15
	3DES	50,52	86,93	54,14
	RC2	28,60	80,34	37,45
	IDEA	23,77	74,12	29,02
	TEA	13,84	48,61	15,4
	Blowfish	11,02	31,14	12,0
	Twofish	12,01	36,17	14,0

Tablo 4.5. (Devam) 1 MB'lık metnin şifre çözme işlemi performans değerleri

Bilgisayar 2	DES	33,64	45,25	6,00
	AES	32,95	41,43	5,00
	3DES	45,21	67,21	12,00
	RC2	36,66	60,28	7,00
	IDEA	30,21	40,71	5,6
	TEA	22,14	28,16	4,2
	Blowfish	14,71	20,16	3,2
	Twofish	18,26	21,17	4,0
Bilgisayar 3	DES	42.21	65.28	6.00
	AES	44.60	67.59	6.00
	3DES	55.89	93.67	10.00
	RC2	48.63	67.52	6.50
	IDEA	17,82	32,02	4,0
	TEA	24,77	42,75	5,0
	Blowfish	32,16	56,87	6,0
	Twofish	27,41	46,75	6,0

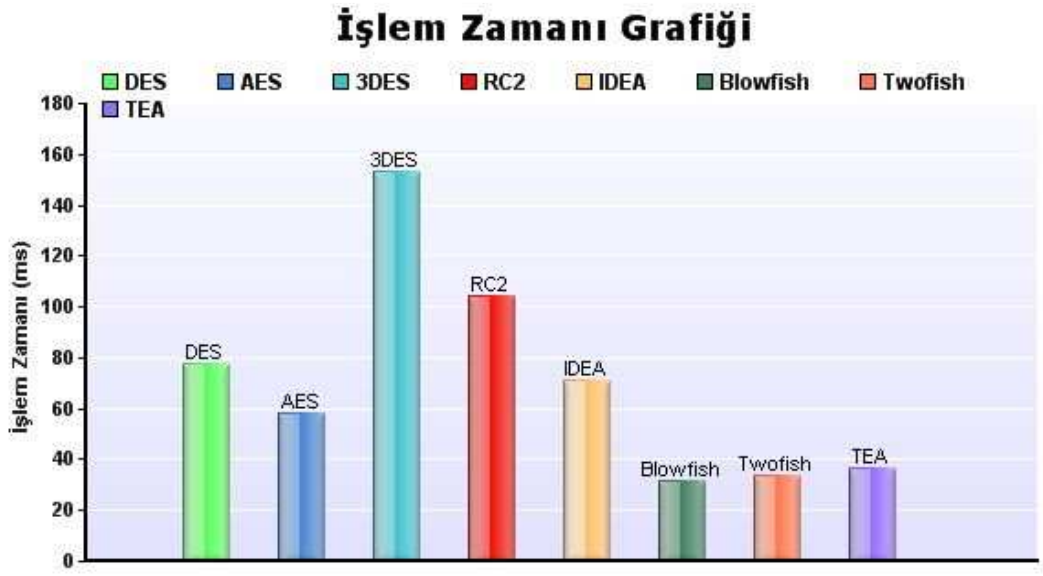
1027 Kilobyte boyutundaki metin dosyası şifrelendikten sonra boyutu 1348 Kilobyte olarak bir artış göstermektedir.

4.4.1.3. 2 MB'lık metnin şifrelenmesi

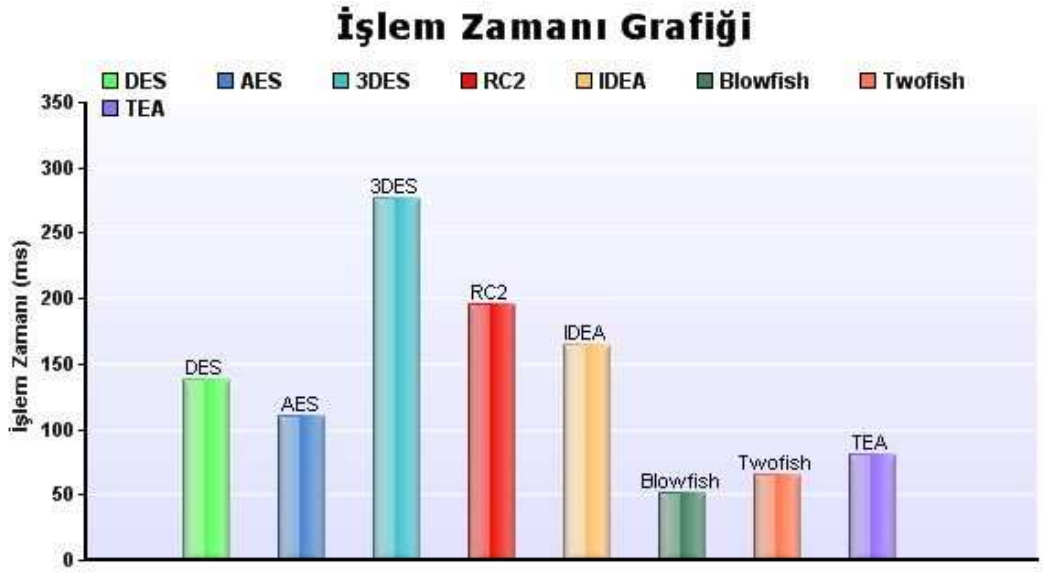
2 Megabyte (MB) karakter uzunluğuna sahip metnin şifrelenmesi sonucunda elde edilen işlem zamanı, CPU kullanımı ve RAM kullanımına ait tüm değerler Tablo 4.6. 'de verildiği gibidir. Tabloda yer alan işlem zamanına ait değerler milisaniye, işlemci kullanımına ait değerler yüzde (%) , hafıza kullanımına ait değerler ise megabyte (MB) türünden ifade edilmiş değerlerdir. Ayrıca metnin şifrelenmesi sonucu elde edilen işlem zamanına ait değerler karşılaştırmalı bir şekilde Şekil 4.16. , Şekil 4.17. ve Şekil 4.18.' de grafiksel olarak gösterilmiştir. Grafikte gösterilmiş olan bu değerler de milisaniye türünden değerlerdir.

Tablo 4.6. 2 MB'lık metni şifreleme işlemi performans değerleri

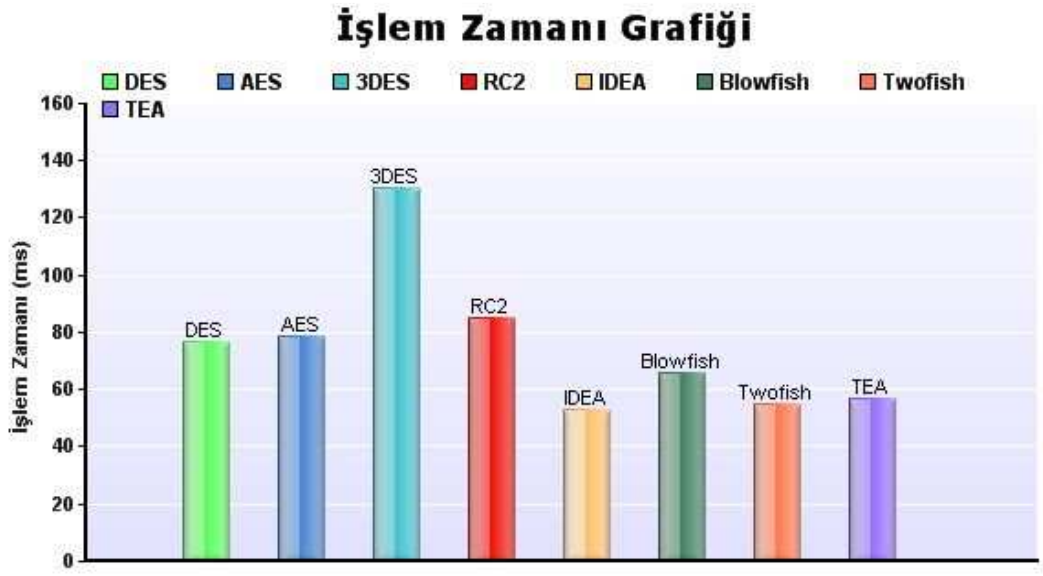
Test Bilgisayarı	Algoritma	İşlem Zamanı (ms)	RAM Kullanımı (MB)	CPU Kullanımı (%)
Bilgisayar 1	DES	77,76	85,12	35,02
	AES	58,36	84,19	30,42
	3DES	153,30	107,20	51,47
	RC2	104,58	102,68	42,89
	IDEA	71,03	82,11	33,02
	TEA	36,85	41,16	16,04
	Blowfish	31,44	31,13	11,72
	Twofish	34,17	36,10	13,67
Bilgisayar 2	DES	138,90	64,19	5,00
	AES	111,20	62,15	4,00
	3DES	277,65	103,73	10,00
	RC2	196,17	93,55	6,00
	IDEA	164,61	80,17	6,50
	TEA	81,21	39,08	3,50
	Blowfish	51,17	27,12	3,00
	Twofish	65,91	33,02	3,00
Bilgisayar 3	DES	76.60	72.06	6.00
	AES	78.86	70.51	10.00
	3DES	130.25	92.29	12.00
	RC2	85.30	76.42	8.00
	IDEA	53,01	62,41	6,10
	TEA	57,22	66,17	8,10
	Blowfish	65,71	69,31	6,0
	Twofish	54,96	64,12	5,5



Şekil 4.16. 2 MB'lık metni şifreleme bilgisayar-1 işlem zamanı



Şekil 4.17. 2 MB'lık metni şifreleme bilgisayar-2 işlem zamanı



Şekil 4.18. 2 MB'lık metni şifreleme bilgisayar-3 işlem zamanı

2 Megabyte karakter uzunluğuna sahip metnin şifreli halinin, şifreleme işleminde kullanılan anahtar ve IV değerleri kullanılarak şifre çözme işlemi yapılması sonucunda elde edilen işlem zamanı, CPU kullanımı ve RAM kullanımı değerleri Tablo 4.7.'de verildiği gibidir.

Tablo 4.7. 2 MB'lık metnin şifre çözme işlemi performans değerleri

Test Bilgisayarı	Algoritma	İşlem Zamanı (ms)	RAM Kullanımı (MB)	CPU Kullanımı (%)
Bilgisayar 1	DES	52,34	88,95	13,00
	AES	50,47	84,50	12,70
	3DES	62,70	105,19	54,83
	RC2	58,04	100,28	30,74
	IDEA	49,64	85,12	12,5
	TEA	26,71	45,17	6,5
	Blowfish	21,37	35,77	5,0
	Twofish	24,96	42,16	6,0

Tablo 4.7. (Devam) 2 MB'lık metnin şifre çözme işlemi performans değerleri

Bilgisayar 2	DES	74,09	70,08	8,00
	AES	72,99	68,07	8,00
	3DES	83,34	104,73	14,00
	RC2	78,78	91,27	12,00
	IDEA	66,41	64,07	7,0
	TEA	45,40	44,17	5,0
	Blowfish	30,02	30,76	3,5
	Twofish	37,76	38,92	4,0
Bilgisayar 3	DES	84.72	63.58	8.00
	AES	82.75	60.98	7.00
	3DES	105.18	105.33	12.00
	RC2	95.22	97.47	9.00
	IDEA	35,54	30,08	4,0
	TEA	49,41	44,16	5,0
	Blowfish	65,17	53,17	7,0
	Twofish	55,40	48,12	6,0

2054 Kilobyte boyutundaki metin dosyası şifrelendikten sonra boyutu 2696 Kilobyte olarak bir artış göstermektedir.

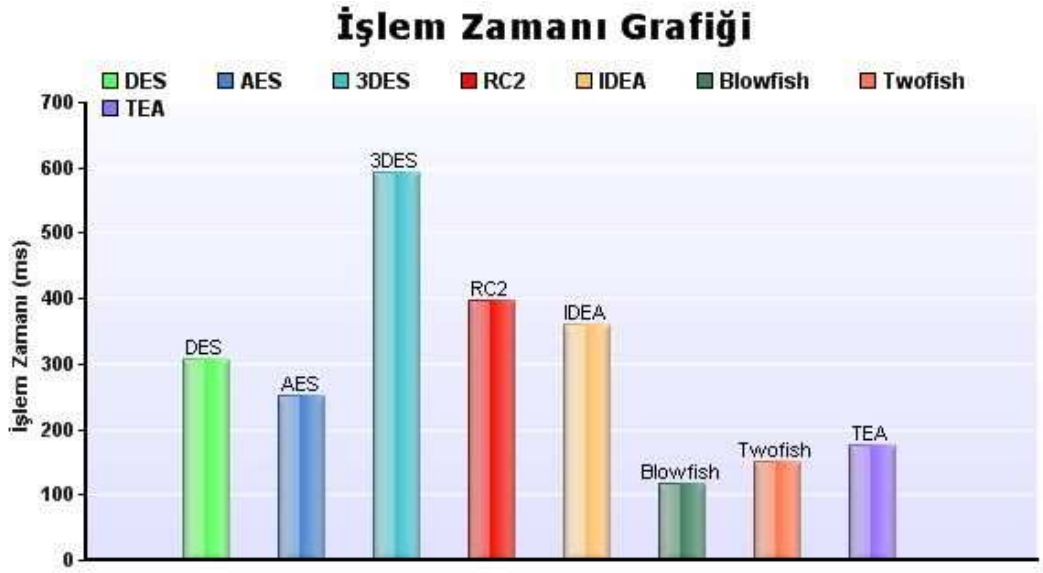
4.4.1.4. 5 MB'lık metnin şifrelenmesi

5 Megabyte (MB) karakter uzunluğuna sahip metnin şifrelenmesi sonucunda elde edilen işlem zamanı, CPU kullanımı ve RAM kullanımına ait tüm değerler Tablo 4.8. 'de verildiği gibidir. Tabloda yer alan işlem zamanına ait değerler milisaniye, işlemci kullanımına ait değerler yüzde (%) , hafıza kullanımına ait değerler ise megabyte (MB) türünden ifade edilmiş değerlerdir. Ayrıca metnin şifrelenmesi sonucu elde edilen işlem zamanına ait değerler karşılaştırmalı bir şekilde Şekil 4.19. , Şekil 4.20. ve Şekil 4.21.' de grafiksel olarak gösterilmiştir. Grafikte gösterilmiş olan bu değerler de milisaniye türünden değerlerdir.

Tablo 4.8. 5 MB'lık metni şifreleme işlemi performans değerleri

Test Bilgisayarı	Algoritma	İşlem Zamanı (ms)	RAM Kullanımı (MB)	CPU Kullanımı (%)
Bilgisayar 1	DES	353,30	125,67	34,27
	AES	307,39	92,21	32,28
	3DES	944,51	261,93	54,28
	RC2	576,35	156,37	36,43
	IDEA	341,11	120,34	32,07
	TEA	158,37	58,14	15,42
	Blowfish	121,82	43,39	13,64
	Twofish	146,19	54,18	17,28
Bilgisayar 2	DES	307,84	93,87	12,00
	AES	252,81	90,23	10,00
	3DES	594,79	228,58	23,99
	RC2	397,44	165,91	14,00
	IDEA	361,17	109,87	14,00
	TEA	176,86	59,67	7,00
	Blowfish	117,21	45,14	5,5
	Twofish	151,17	54,51	6,2
Bilgisayar 3	DES	356,32	89,35	12,5
	AES	353,31	82,73	10,00
	3DES	552,97	233,88	28,00
	RC2	397,91	181,67	18,99
	IDEA	125,96	65,40	10,00
	TEA	132,87	70,12	12,00
	Blowfish	147,71	79,14	14,00
	Twofish	131,08	67,87	13,00

Şekil 4.19. 5 MB'lık metni şifreleme bilgisayar-1 işlem zamanı



Şekil 4.20. 5 MB'lık metni şifreleme bilgisayar-2 işlem zamanı

Şekil 4.21. 5 MB'lık metni şifreleme bilgisayar-3 işlem zamanı

5 Megabyte karakter uzunluğuna sahip metnin şifreli halinin, şifreleme işleminde kullanılan anahtar ve IV değerleri kullanılarak şifre çözme işlemi yapılması sonucunda elde edilen işlem zamanı, CPU kullanımı ve RAM kullanımı değerleri Tablo 4.9.'da verildiği gibidir.

Tablo 4.9. 5 MB'lık metnin şifre çözme işlemi performans değerleri

Test Bilgisayarı	Algoritma	İşlem Zamanı (ms)	RAM Kullanımı (MB)	CPU Kullanımı (%)
Bilgisayar 1	DES	312,62	96,70	18,56
	AES	292,42	94,28	16,88
	3DES	349,50	272,90	42,54
	RC2	341,39	170,44	36,00
	IDEA	301,12	93,10	18,00
	TEA	161,37	48,41	9,50
	Blowfish	132,14	38,91	7,50
	Twofish	147,13	44,16	8,70

Tablo 4.9. (Devam) 5 MB'lık metnin şifre çözme işlemi performans değerleri

Bilgisayar 2	DES	177,40	179,54	14,00
	AES	173,49	178,97	12,00
	3DES	189,69	217,11	20,00
	RC2	182,96	195,38	18,00
	IDEA	159,14	162,14	11,4
	TEA	108,12	110,42	8,00
	Blowfish	72,18	101,84	5,50
	Twofish	88,71	106,12	6,50
Bilgisayar 3	DES	241,34	157,02	12,00
	AES	237,47	153,28	8,00
	3DES	258,00	272,95	24,00
	RC2	244,94	222,43	20,00
	IDEA	144,4	99,04	5,00
	TEA	194,0	124,02	7,00
	Blowfish	226,0	132,4	8,00
	Twofish	201,0	128,12	7,00

5124 Kilobyte boyutundaki metin dosyası şifrelendikten sonra boyutu 6920 Kilobyte olarak bir artış göstermektedir.

BÖLÜM 5. SONUÇLAR VE ÖNERİLER

Bilgi güveliği kapsamında değerlendirilen bütün kavramlar bu güvenliği sağlayacak şifreleme tekniklerini kullanırlar. Bilişim teknolojilerindeki gelişmelere paralel olarak bilgi güvenliği disiplinler arası bir konu olduğundan gün geçtikçe önemi artmaktadır. Bu çalışmada bilgi güvenliğini sağlama amaçlı olarak kullanılan simetrik ve asimetrik algoritmalardan en sık kullanılanları zaman karmaşıklığı, işlemci karmaşıklığı ve hafıza karmaşıklığı bakımından incelenmiş ve uygulamaları gerçekleştirilmiştir. Simetrik şifreleme algoritmalarından DES, AES, 3DES, RC2, Blowfish, Twofish, IDEA, TEA şifreleme algoritmaları ve asimetrik şifreleme algoritmalarından RSA algoritması incelenmiştir. Çalışma aynı zamanda şifreleme algoritmalarının teknik özelliklerinin belirlenmesi çalışması olarak ta kullanılabilir. Çalışmada sayısal imzalama ve imza doğrulama işlemlerinde en çok kullanılan DSA algoritması ve hash algoritmaları da incelenmiştir.

Çalışma sonucunda elde edilen verilere göre küçük boyuttaki verilerin şifrelenmesi işleminde DES algoritması AES algoritmasından daha performanslı bir şekilde çalışmaktadır. Şifrelenecek verinin boyutu 1MB ve üzeri olduğu durumlarda AES şifreleme algoritması DES şifreleme algoritmasından daha performanslı bir şekilde çalışmaktadır. 1 MB ‘tan küçük verilerin şifrelenmesi işlemi sonucunda elde edilen sonuçlara göre algoritmaların performans sıralamasının büyükten küçüğe doğru Blowfish, Twofish, TEA, IDEA, DES, AES, RC2, 3DES, RSA algoritmaları şeklinde olduğunu gösterir (Tablo 4.2 - 4.9). Simetrik şifreleme algoritmaları asimetrik şifreleme algoritmalarına göre çok daha performanslı bir şekilde çalışmaktadır. Çalışmada algoritmalar farklı donanım özelliklerine (AMD, Intel işlemci) ve farklı işletim sistemlerine sahip bilgisayarlarda test edilmiştir. Kullanılacak algoritmalar seçilirken bilgisayar özellikleri de göz önünde bulundurulmalıdır. Özellikle AMD işlemciye sahip bilgisayarlarda asimetrik şifreleme algoritması olan RSA, diğer simetrik şifreleme algoritmaları olan Blowfish,

Twofish, TEA, IDEA, DES, AES, 3DES ve RC2'ye göre çok daha yavaş çalışmaktadır. Şifrelenecek verinin boyutu 1 MB'tan fazla olması durumunda elde edilen sonuçlara göre algoritmaların performans sıralamasının büyükten küçüğe doğru Blowfish, Twofish, TEA, AES, IDEA, DES, RC2, 3DES, RSA algoritmaları şeklinde olduğunu gösterir (Tablo 4.2 - 4.9). Metinler şifrelendikten sonra şifreli hallerinde boyut olarak bir artış olmaktadır. Metinlerin şifreli halleri daha fazla karakter içermektedir. Burada gösterilen performans sonuçlarının algoritmaların güvenilirliğiyle bir ilgisi bulunmamaktadır.

Asimetrik şifreleme algoritmaları ile yapılan işlemler (şifreleme, deşifreleme, sayısal imzalama ve imza doğrulama işlemleri) yavaş işlemlerdir. Kullanılan algoritma ve uygulamanın çalıştırıldığı platform işlemlerin hızını belirleyen önemli faktörlerdendir. Ancak her ne şart altında olursa olsun, tek anahtarlı simetrik algoritmalar (DES, AES, 3DES, RC2, Blowfish, Twofish, IDEA, TEA) çok daha hızlıdır (Tablo 4.2).

Daha ileri çalışmalara öneri olarak algoritmalar stream ve blok şifreleme algoritmaları olarak sınıflandırılabilir, hash algoritmalarının performans analizleri de yapılabilir. Ayrıca şifreleme algoritmalarının değerlendirilmesinde kriptanaliz kriterleri belirlenerek saldırılara karşı dayanılıklığı da incelenebilir.

KAYNAKLAR

[1] GÜVENOĞLU, E., Görüntü Şifreleme Algoritmaları ve Performans Analizleri, Y.Lisans, Trakya Üniversitesi, Bilgisayar Mühendisliği Anabilim Dalı, sf. 1-5 ,10-14, 2006.

[2] SMART N., Cryptography: An Introduction, McGraw – Hill Companies, Inc, 2003.

[3] ERKOÇ, K., Kriptoloji ve Bilgi Güvenliği, Y.Lisans, Sakarya Üniversitesi, Bilgisayar ve Bilişim Mühendisliği, sf. 7, 16, 28-30, 44-46, 48, 2004.

[4] CW Kriptoloji – Kriptografi Seminerleri Sonuç Bildirgesi, sf. 8-9, 2009.

[5] KOBLITZ, N., A Course in Number Teory and Cryptography, Springer-Verlag, New York , 1994.

[6] HELLMAN, M.E., An Overview of Puplic Key Cryptography, IEEE Communications Society Magazine, 1978.

[7] YERLİKAYA, T., Şifreleme Algoritmalarının Analizi, Doktora, Trakya Üniversitesi, Bilgisayar Mühendisliği Bölümü, Sf. 8, 22-27, 2006.

[8] ATAY, S., Eliptik Eğri Tabanlı Kriptografik Protokol ve Akıllı Kart Üzerinde Bir Uygulama, Ağ ve Bilgi Güvenliği Sempozyumu, 2005.

[9] ERHAN M. , RSA Algoritmasını Kullanan Şifreleme / Deşifreleme Yazılımının Tasarımı, Y.Lisans, İstanbul Teknik Üniversitesi, Fen Bilimleri Enstitüsü, 1993.

[10] YILDIRIM, M., DES ve DES Benzeri Şifreleme Sistemlerinin Diferansiyel Kripto Analizi, Y.Lisans, İstanbul Teknik Üniversitesi, Fen Bilimleri Enstitüsü, sf.8-13, 1995.

[11] ÜLGER C., Holotransformasyon Metodu ile Veri Şifreleme, Doktora, Marmara Üniversitesi, Fen Bilimleri Enstitüsü, 1999.

[12] ANDİÇ, E., Bilgisayar Haberleşmesinde Şifreleme(Kripto) Yazılımıyla Güvenliğin Sağlanması, Y.Lisans, Marmara Üniversitesi, Fen Bilimleri Enstitüsü, sf. 9, 2002.

[13] ÇALIŞKAN, E.M., Şifreleme Algoritmalarının Performans-Kripto

Analizleri ve Eğitimde Kullanılması, Y.Lisans, Marmara Üniversitesi, Fen Bilimleri Enstitüsü, sf. 66-72, 2004.

[14] DERELİOĞLU E.A. , Değişken Uzunluklu RSA Şifreleme Sistemlerinin Tasarımı ve Gerçeklenmesi , Y.Lisans, Yıldız Teknik Üniversitesi, Fen Bilimleri Enstitüsü, 2005.

[15] ORDU L., AES Algoritmasının FPGA Üzerinde Gerçeklenmesi ve Yan Kanal Analizi Saldırılarına Karşı Güçlendirilmesi, Y.Lisans, İstanbul Teknik Üniversitesi ,Fen Bilimleri Enstitüsü, 2006.

[16] KAYIŞ H., AES Uygulaması'nın FPGA Gerçeklemelerine Karşı Güç Analizi Saldırısı, Y.Lisans, İstanbul Teknik Üniversitesi, Fen Bilimleri Enstitüsü, 2006.

[17] BULUŞ, H.N., Temel Şifreleme Algoritmaları ve Kriptanalizlerinin İncelenmesi, Y.Lisans, Trakya Üniversitesi, Fen Bilimleri Enstitüsü, sf. 2-19, 2006.

[18] BUCHMANN, A. Johannes, "Introduction to Cryptography", Sayfa No:1-21, 2000.

[19] MERK I c., U.C., HELLMAN, M.. "Hiding Information and Signatures in Trapdoor Knapsacks"', IEEE Transactions on Information Theory IT-24, No:5, sf. 525-530., September 1978.

[20] WAGSTAFF, S., Cryploanalysis of Number Theoretic Ciphers, Chapman& Hall/Crc, Boca Raton- Florida, 2003.

[21] FAQ Editor, Answers to Frequently asked Questions About Today's Cryptography, Version 3.0, RSA Laboratories, September 1993.

[22] SCHNEIDER, B. "Applied Cryptography Second Edition", John Wiley & Sons, Inc., New York, 1996.

[23] MONTGOMERY, P., "Modular Multiplication Without Trial Division,", Mathematics of Computation, 44, 1985.

[24] TEKTAŞ, M. , BABA, F. , ÇALIŞKAN M., "Şifreleme Algoritmalarının Sınıflandırılması Ve Bir Kredi Kartı Uygulaması", 3RD Internatıonal Advanced Technologıes Symposıum, August 18-20, Ankara, 2003.

[25] EL GAMAL, T., "A Public-Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. Advances in Cryptology: Proceedings of CRYPTO 84", Springer Verlag, pp. 10-18, 1988.

[26] P1398, "Standard Specifications For Public−Key Cryptography", IEEE, October 1998.

[27] QUISQUATER, J.J. and COUVREUR, C., "Fast Decipherment Algorithm

for RSA Public- Key Cryptosystem", Electronics Letters, vol. 18, no. 21, pp. 905- 907. October 1982.

[28] KALISKI B., “The Mathematics of the RSA Public-Key Cryptosystem”, RSA Laboratories NY, 2001.

[29] KNUTH, D.E., “Art of Computer Programming Volume 2 / Seminumerical Algorithms”, Addison Wesley, 1969.

[30] ÖKTEN, Ö., "Standard Digital Signature Algorithm (DSA) and Secure Hash Algorithm (SHA) in Public Key Cryptology, ODTU Fen Bilimleri Enstitüsü, Y.Lisans, 1997.

[31] DIFFIE, W.. HELLMAN, M.E., "New Directions in Cryptography", IEEE Transaction on Information Theory, Vol:22, No:6., p 644-654, November 1976.

[32] http://www.bidb.itu.edu.tr/?d=891, Ocak 2010.

[33] DOĞAN, A.Y., AES Algoritmasının FPGA Üzerinde Düşük Güçlü Tasarımı, Y.Lisans, İstanbul Teknik Üniversitesi, Fen Bilimleri Enstitüsü, sf. 28, 2008.

[34] http://en.wikipedia.org/wiki/RC2, Ocak 2010.

[35] http://www.ietf.org/rfc/rfc2268.txt, Ocak 2010.

[36] http://groups.google.com/group/sci.crypt/msg/f383f5dae68ebc70, Ocak 2010.

[37] MENEZES, A., VANSTONE, S., Handbook of Applied Cryptography, CRC Pres, 1996.

[38] http://www.bilgisayarkavramlari.com/2008/04/17/blowfish-sifreleme-balon-baligi-sifrelemesi-blowfish-cipher, Ağustos 2010

[39] http://tr.wikipedia.org/wiki/Blowfish, Ağustos 2010

[40] ab.org.tr/ab08/kitap/Bildiriler/158_121_AB08.pdf, Ağustos 2010

[41] http://en.wikipedia.org/wiki/Twofish, Ağustos 2010

[42] http://www.bilgisayarkavramlari.com/2009/06/11/idea-uluslar-arasi-sifreleme-algoritmasi, Ağustos 2010

[43] http://en.wikipedia.org/wiki/International_Data_Encryption_Algorithm, Ağustos 2010

[44] http://en.wikipedia.org/wiki/Tiny_Encryption_Algorithm, Ağustos 2010

[45] http://www.bilgisayarkavramlari.com/2009/06/10/tea-tiny-encryption-

algorithm, Ağustos 2010

ÖZGEÇMİŞ

Ümit GÜNDEN, 1983 ‘te Samsun’ da doğdu. İlk ve orta eğitimini Zonguldak’ta , lise eğitimini Ordu’da tamamladı. 2001 yılında Ünye Anadolu Öğretmen Lisesi’nden mezun oldu. 2001 yılında başladığı Sakarya Üniversitesi Bilgisayar Mühendisliği bölümünü 2006 yılında bitirdi. 2006 – 2010 yılları arasında İstanbul’da özel sektörde Yazılım Mühendisi olarak çalıştı. Şu anda Yazılım Mühendisi olarak çalışma hayatına devam etmektedir.