

T.C.

GEBZE YÜKSEK TEKNOLOJİ ENSTİTÜSÜ
MÜHENDİSLİK VE FEN BİLİMLERİ ENSTİTÜSÜ

GÜVENLİK PROTOKOLLERİNİN TASARIMI
VE ANALİZİ

FATİH BİRİNCİ
YÜKSEK LİSANS TEZİ
BİLGİSAYAR MÜHENDİSLİĞİ ANA BİLİM DALI

121 334

121334

T.C. YÜKSEKÖĞRETİM KURULU
DOKÜMANTASYON MERKEZİ

TEZ DANIŞMANI
YRD. DOÇ. DR. İBRAHİM SOĞUKPINAR

GEBZE

2002



Fatih Birinci'nin tez çalışması, G.Y.T.E. Mühendislik ve Fen Bilimleri Enstitüsü Yönetim Kurulu'nun 22 / 07 / 2002 tarih ve 2002 / 24 sayılı kararıyla oluşturulan jüri tarafından Bilgisayar Mühendisliği Anabilim Dalında YÜKSEK LİSANS Tezi olarak kabul edilmiştir.

JURI

Üye(Tez Danışmanı) : Yrd. Doç. Dr. İbrahim SOĞUKPINAR, GYTE

Üye : Yrd. Doç. Dr. F. Erdoğan SEVİLGİN, GYTE

Üye : Yrd. Doç. Dr. Oğuz KUCUR, GYTE

ONAY

GYTE Mühendislik ve Fen Bilimleri Enstitüsü Yönetim Kurulu'nun 03.04.2002..... tarih ve 2002/28. sayılı kararı.

ÖZET



“ABD Ulusal Standart ve Teknoloji Enstitüsü” (NIST) “Veri Şifreleme Standardı” (DES)’in yerine geçen “İleri Şifreleme Standardı” (AES)’in çalışma kiplerini belirlemeye çalışmaktadır. Bu çalışma kiplerinden en göze çarpanlar arasında OCB ve CTR gibi kipler de bulunmaktadır. Söz konusu çalışma kiplerinde DES çalışma kiplerindeki bazı zayıflıklar giderilerek, örneğin OCB çalışma kipinde olduğu gibi şifreleme ve bütünlüğü beraber sağlamak gibi çok güzel fikirler kullanılmıştır.

Ancak yeni çalışma kipleri, literatürde mevcut çalışma kiplerindeki şifrelemeye karşı uygulanan bazı aktif ve pasif saldırı yöntemleri açısından değerlendirilmemiştir. Bu çalışmada DES ve AES çalışma kipleri, güvenlik protokollerine karşı yapılan saldırı yöntemleri açısından değerlendirip karşılaştırılmıştır. Sonuç olarak DES çalışma kiplerine uygulanan bazı zayıflıkların AES çalışma kiplerinde olmadığı ispatlanmıştır. Ayrıca OCB ve CBC çalışma kipleri de performansları açısından karşılaştırılmıştır.

Bu çalışmada ayrıca DES kiplerinin analizleri yapılarak zayıf ve güçlü yönleri araştırılmış ve birbiriyle karşılaştırılmaları yapılarak eksik taraflarının yeni AES çalışma kipleriyle nasıl giderildiği gösterilmiştir. Elde edilen sonuçlar, güvenlik protokolü analizi ve tasarımı yapan kişilere önemli tavsiyeler olarak sunulmuştur.

SUMMARY



"National Institute of Standards and Technology" (NIST) is in process of generating new "Advanced Encryption Standard" (AES) modes of operation. . There are some proposals having good ideas such as "Counter" (*CTR*) and "Offset Codebook" (OCB) modes. In these new AES-modes of operation some precautions are taken to prevent some deficiencies of DES modes of operations. There are some proposals having good and ingenious ideas, for example, OCB mode provides confidentiality and integrity simultaneously.

However, these new modes of operations haven't been evaluated from point of view of the active attacks made to the encryption with these modes yet. In this thesis, DES and AES modes of operations evaluated and compared from point of view of the attacks made to the security protocols. It is proved that these new modes are secure against these attacks. Besides, OCB and CBC modes performance are compared.

In this thesis, DES modes' weak and strong aspects are analyzed and how these weak sides are corrected with new AES modes are presented. The results are given as recommendations to the security protocol designers and analyzers.

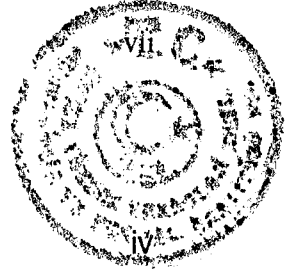


TEŞEKKÜR

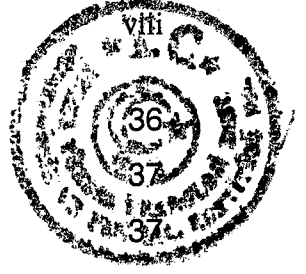
Öncelikle tez çalışmamın her aşamasında bana destek olan ve beni motive eden tez danışmanım Yrd. Doç. Dr. İbrahim Soğukpınara teşekkür ediyorum. Daha sonra, tez çalışmamı destekleyen ve gerekli izinleri sağlayan TUBİTAK-UEKAE ve sayın müdürleri Önder Yetiş ve Alparslan Babaoğlu'na teşekkür ediyorum. Son olarak da eşim Sevinç Birinci'ye ve aileme bana destek olmalarından dolayı teşekkürlerimi belirtmek isterim.



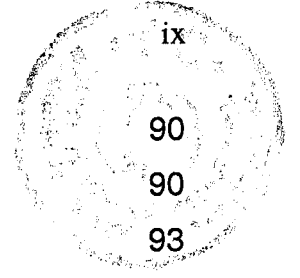
İÇİNDEKİLER DİZİNİ



ÖZET	
SUMMARY	v
TEŞEKKÜR	vi
İÇİNDEKİLER DİZİNİ	vii
SİMGELER VE KISALTMALAR DİZİNİ	ix
ŞEKİLLER DİZİNİ	xi
ÇİZELGELER DİZİNİ	xiii
1. GİRİŞ	1
1.1. Kuramsal Yaklaşımlar	2
1.1.1. Gizlilik	2
1.1.2. Bilgi Kaynağının Doğrulanması	2
1.1.3. Uç Doğrulama	3
1.1.4. Mesaj Bütünlüğünün Sağlanması	3
1.1.5. İnkâr Edememe	3
1.1.6. Kimliğinin Saklı Tutulması	4
1.1.7. Kullanılabilirlik	4
1.2. Güncel Protokoller ile İlgili Yapılan Bazı Çalışmalar	5
1.3. Gösterim	6
1.4. Güvenlik Protokolü Tasarımında En Uygun Çalışma Kipinin Seçilmesi ve Analizi	8
2. KRİPTOLOJİK SİSTEMLER	9
2.1. Simetrik Şifreleme Sistemleri	11
2.1.1. Blok Şifreleme Algoritmaları	12
2.1.2. Dizi Şifreleme Algoritmaları	23
2.2. Açık Anahtarlı Şifreleme Sistemleri	24
2.3. Güvenlik Protokollerinin Tehditleri	26
2.3.1. Aradaki Kişi Saldırısı	27
2.3.2. Yansıma Saldırısı	29
2.3.3. Nasihatçi Saldırısı	30
2.3.4. Tekrar Gönderme Saldırısı	30
2.3.5. Arada Bırakma Saldırısı	31
2.3.6. İleri Doğru Güvenlik Yetersizliği Saldırısı	33



2.3.7. Cebirsel Saldırılar	
2.4. Konu İle İlgili Yayınlar ve Çalışmalar	
2.4.1. Çalışma Kiplerinin Önemi	
2.4.2. Dizi Şifreleme Algoritmalarının Zayıflıkları	45
2.4.3. Mesajın Gizliliği	49
2.4.4. Mesajın Bütünlüğünün Korunması ve Kimlik Doğrulama	53
2.4.5. Mesajın Gönderildiğinin İnkâr Edilememesi	56
2.4.6. Anahtar Yönetimi	57
2.4.7. Gerçekleme Hataları	58
3. GÜVENLİK PROTOKOLÜ TASARIMINDA BLOK ŞİFRELEME KİPLERİNİN ANALİZİ VE SEÇİMİ	60
3.1. DES'in AES ile Değiştirilmesinin Nedenleri	60
3.2. DES Çalışma Kipleri	62
3.2.1. Elektronik Kod Kitabı (ECB) Çalışma Kipi	62
3.2.2. Şifre Bloğu Zincirleme (CBC) Çalışma Kipi	62
3.2.3. Şifre Geri Besleme (CFB) Çalışma Kipi	63
3.2.4. Çıkış Geri Besleme (OFB) Çalışma Kipi	63
3.3. Önerilen AES Çalışma Kiplerinden Bazıları	65
3.3.1. Ofset Kod Kitabı (OCB) Çalışma Kipi	65
3.3.2. Sayaç (CTR) Çalışma Kipi	68
3.4. İlk Vektör IV Konusunda Bazı Problemler	70
3.5. AES Çalışma Kiplerinin Güvelik Analizi ve Seçimi Üzerine Yapılan Özgün Çalışmalar	71
3.5.1. Kes - Yapıştır Saldırısı	71
3.5.2. Şifreli Mesajın Okunabilmesi Saldırısı	77
3.5.3. Oturum Kaçırma Saldırısı	77
3.5.4. IV'yi Kötüye Kullanma Saldırısı	78
3.6. Güvenlik Protokolü Tasarımında Çalışma Kipi Seçimi	79
3.7. Çalışma Kiplerinden Bazılarının Performans Karşılaştırılması	81
4. ÖRNEK GÜVENLİK PROTOKOLÜ ANALİZİ (IPSEC, SSL VE WEP ANALİZİ)	83
4.1. RC4 Algoritması ile İlgili Problemler	88
4.2. WEP Protokolündeki Problemlere Bazı Çözüm Önerileri	89
4.2.1. Şifreleme sistemi ve Çalışma kipi ile ilgili öneriler	90



4.2.2. Oturum Anahtarı Oluřturulması

4.2.3. Tavsiye edilen Kapsüle Alma Yöntemi

5. SONUÇLAR/ÖNERİLER

KAYNAKLAR

ÖZGEÇMİŐ

EKLER

Performans Karşılařtırma Kodu

90

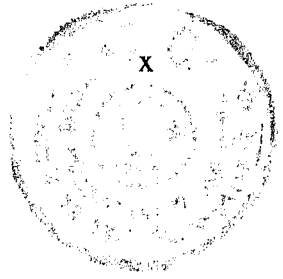
90

93

97

101





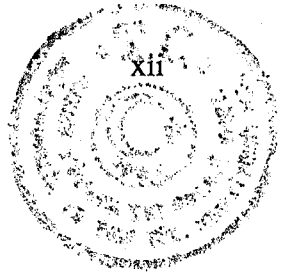
SİMGELER VE KISALTMALAR DİZİNİ

IV	Initial Vector (İlk Vektör)
K	Şifreleme işleminde kullanılan anahtar
NSPK	Needham-Schroeder Public-Key
NSSK	Needham-Schroeder Secret-Key
RSA	Rivest Shamir Adleman tarafından bulunan Açık Anahtarlı Şifreleme Algoritması
DH	Diffie-Hellman, Anahtar anlaşım protokolü.
CSP	Communicating Sequential Processes
IETF	Internet Engineering Task Force
DES	Data Encryption Standart (Veri Şifreleme Standardı)
3DES	DES algoritmasının ard arda Üç Kez Kullanılması.
AES	Advanced Encryption Standart (İleri Şifreleme Standardı)
CBC	Cipher Blok Chaining (Şifre Bloğu Zincirleme)
CFB	Cipher Feedback (Şifre Geri Besleme)
OCB	Offset Codebook Mode (Ofset Kod Kitabı)
CTR	Counter Mode (Sayaç)
MAC	Message Authentication Mode (Mesaj Doğrulama Kodu)
CBC-MAC	CBC ile Gerçekleştirilen MAC.
IP	Internet Protocol (İnternet Protokolü).
IPSec	IP Security. (IP Güvenliği)
SSL	Secure Socket Layer
802.11	Standarts for Wireless Networks (Kablosuz Ağ Standartları)
WEP	Wired Equivalent Privacy (Kablolu Denk Gizlilik)
RC4	Ron Rivest'in Geliştirdiği Dizi Şifreleme Algoritması.
NSA	National Security Agency
NIST	National Institute of Standards and Technology
DSA	Digital Signature Algorithm
ISO	Internet Standarts Organization
PEM	Privacy Enhanced Mail
PPTP	Noktadan Noktaya Tünel Protokolü
UDP	User Datagram Protocol
TCP	Transmission Control Protocol

MD2,4,5	Ron Rivest tarafından geliştirilen mesaj özetleme algoritması
HMAC	Keyed-Hashing for Message Authentication
CRC-32	Cyclic Redundancy Check
IKE	Internet Key Exchange
PCMCIA	Personal Computer Memory Card International Association
JDK	Java Development Kit
ESP	Encapsulation Security Payload
AH	Authentication Header
X.400	E-Mail için evrensel protokol.
FIPS	Federal Information Processing Standards.
" $\oplus$ "	İkillerin modüler 2 olarak toplanmasını simgelemektedir.
" "	İkillerin ardarda eklenmelerini simgelemektedir.



ŞEKİLLER DİZİNİ



Şekil	Sayfa
2.1.:Yerine Koyma Permütasyon Ağının Şematik Gösterimi.	13
2.2.: DES Algoritmasının Giriş ve Çıkışları.	14
2.3.: DES Algoritmasının İşlem Yollarının Şematik Gösterimi.	16
2.4.: DES Algoritmasının İç Fonksiyonunun Şematik Gösterimi.	17
2.5.: Eş Zamanlı Dizi Şifreleme Sisteminin Genel Yapısı.	23
2.6.: İkili Ekleme Dizi Şifreleme Sistemleri.	23
2.7.: Kendiliğinden Eş Zamanlı Dizi Şifreleme Sisteminin Genel Yapısı.	24
2.8.: NSSK Protokolünün Şematik Gösterimi.	34
2.9.: Burada Başka Bir Yasal Kullanıcının Trafiğini Çözmek Amacı İle Kes-Yapıştır Atağı Yapılması Gösterilmektedir. Burada Çizgili Bölge Şifreli Verileri, Gölge Bölge İse Verinin Yasal Kullanıcıya Ait Olduğunu Göstermektedir.	50
2.10.: Burada Yasal Başka Bir Kullanıcının Oturumunun Kaçırılması İle Bu Kullanıcıya Kötü Niyetli Bilgiler Gönderilmesi Gösterilmektedir. Burada Çizgili Bölge Şifreli Verileri, Gölge Bölge İse Verinin Yasal Kullanıcıya Ait Olduğunu Göstermektedir.	51
3.1.: DES Algoritmasının ECB Çalışma Kipi	61
3.2.: DES Algoritmasının CBC Çalışma Kipi	61
3.3.: DES Algoritmasının CFB Çalışma kipi	62
3.4.: DES Algoritmasının OFB Çalışma Kipi	63
3.5.: Teklif Edilen AES Çalışma Kipi Olan OCB Çalışma Kipi.	66
3.6.: Teklif Edilen AES Çalışma Kipi Olan CTR Çalışma Kipi.	67

ÇİZELGELER DİZİNİ

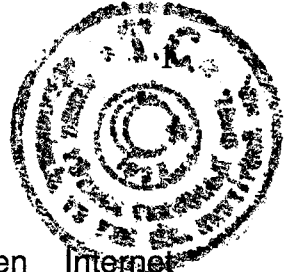


Çizelge

Sayfa

2.1.: DES Algoritmasının İlk Permütasyon Çizelgesi.	18
2.2.: DES Algoritmasının İlk Permütasyonunun Tersinin Çizelgesi.	18
2.3.: DES Algoritmasının Çevrimlerinde Kullanılan, E Genişletme ve P Permütasyon Çizelgeleri.	19
2.4.: DES Algoritmasının Anahtar Zamanlama Permütasyon Çizelgeleri.	19
2.5.: DES Algoritmasının S-Kutuları.	20
3.1.: DES, 3DES ve AES Algoritmalarının Karşılaştırması.	61
3.2.: Çalışma Kiplerinin Performans Karşılaştırması.	82
4.1.: İncelenen Protokollerin Karşılaştırması.	87





1. GİRİŞ

Geçtiğimiz yıllarda çok büyük bir gelişme gösteren İnternet tasarlanırken güvenlik faktörü göz ardı edilmişti. Bundan dolayı, İnternet'in, genel olarak bilgisayar ağlarının, güvenli hale getirilmesi önemli bir ihtiyaç haline gelmiştir. Bu nedenle, İnternet'in gelişmesiyle beraber kriptografik protokol olarak da adlandırılan güvenlik protokolleri büyük bir gelişme göstermiş ve önem kazanmıştır.

Bu bağlamda kriptoloji, bilgisayar ağları arasındaki iletişimin güvenli hale getirilmesinde gün geçtikçe daha önemli bir araç haline gelmiştir. Geçmiş gösterdi ki kriptolojinin sağladığı güvenlik servislerinin kullanmayan elektronik ticaret veya HTTP (Hyper Text Transfer Protokol) gibi protokolleri kullanan insanlar para ve zaman kaybı sebebi ile pişman olmuşlardır. Doğru şekilde kullanıldıkları takdirde çoğu kriptografik algoritmanın güvenli olduğu söylenmektedir. Eğer doğru şekilde kullanılmazlar ise, kriptografik algoritmanın ne kadar güçlü olduğu veya anahtarının ne kadar uzun olduğu fark etmez, kriptolojinin sağladığı güvenlik servisleri fazla bir işe yaramaz. Bu çalışmada, bazı popüler güvenlik protokollerinde yapılan hatalardan, güvenlik protokollerinin analizi ve tasarımı yapılırken dikkat edilmesi gereken hususlardan ve güvenlik protokollerinin nasıl ele alınması gerektiğinden bahsedilecektir. Ayrıca çalışma kipleri üzerine yapılan özgün çalışmalar bölüm (1.4.)'de kısaca anlatılmıştır, bu çalışmalar bölüm (3.)'de detaylandırılmıştır.

Güvenlik protokollerinin amacı dağıtık sistemlerde, tarafların kimliklerinin doğrulanması, taraflar arasında oturum anahtarının oluşturulması, gönderilen bilginin gizliliğinin ve bütünlüğünün sağlanması, gönderici tarafın kimliğinin saklanması ve gönderdiği bilgiyi inkar edememesi gibi çeşitli servisleri sağlamaktır. Güvenlik protokolleri, taraflar arasında mesaj değiş tokuşu ile gerçekleşmektedir ve genellikle üçüncü bir güvenilir tarafın katılımını gerektirirler. Bu protokoller, simetrik ve açık anahtarlı şifreleme algoritmaları, özetleme algoritmaları ve sayısal imza algoritmaları

gibi çeşitli kriptografik mekanizmaları kullanırlar. Bazı durumlarda zaman pulu gibi mekanizmalarda kullanılmaktadır.

1.1. Kuramsal Yaklaşımlar

Yukarıda güvenlik protokollerinin sağladığı önemli güvenlik servislerinden bahsedildi. Tipik olarak bir güvenlik protokolünden, uygulamaya bağlı olarak yukarıdaki güvenlik servislerinin bir alt setini sağlaması beklenmektedir.

1.1.1. Gizlilik

Gizliliğin uygulamaya göre farklılıklar gösterebilen değişik anlamları vardır. En genel anlamda gizlilik, saldırgan kişilerin dürüst kullanıcıların aktivitelerinden herhangi bir bilgi edinememesi şeklinde açıklanabilir. Bu servis genellikle simetrik algoritma kullanımı ile sağlanmaktadır. Açık anahtarlı algoritmalar ise daha çok oturum anahtarı değişiminde, oturum anahtarının gizliliğinin sağlanması ve sayısal imza amacı ile kullanılmaktadır.

1.1.2. Bilgi Kaynağının Doğrulanması

Bilgi kaynağının doğrulanması bir mesajın gerçekten gönderildiği iddia edilen taraf tarafından gönderildiğinin ispatlanması ile gerçekleşmektedir. Ayrıca, mesajın gerçek kaynağı tarafından gönderildikten sonra alıcı tarafa gelene kadar değişikliğe uğramadığı anlamına da gelmektedir. Bir güvenlik protokolünün bilgi kaynağı doğrulanması servisini sağlayabilmesi demek; eğer bir taraf bir mesajı diğer bir taraftan gelmiş olarak kabul ediyor ise gerçekte ikinci tarafın bu mesajı göndermiş olması manasına gelmektedir. Daha güçlü bir özellik ise ikinci tarafın gerçekte mesajı birinci tarafa göndermek için hazırlamış olmasıdır. Burada dikkat edilmesi gereken bir husus, ikinci tarafın mesajı göndermesi ile birinci tarafın mesajı alması arasında geçen zamandan bahsedilmemektedir. Öyleyse, bu anlatımda ikinci tarafın bir mesajı bir defa göndermesine karşılık birinci tarafın aynı mesajı iki

defa almasını önleyecek bir mekanizma mevcut değildir. Bu servis genellikle güçlü (kriptografik tekniklerin kullanıldığı) kimlik doğrulama tekniklerinin kullanımı ile sağlanmaktadır. Bu tekniklerde simetrik ve açık anahtarlı algoritmaların yanı sıra özetleme algoritmaları kullanımı ile sağlanmaktadır. Bu algoritmaların içinde gerçek rasgele sayı veya zaman pulu kullanımı ile de taraflar arasında gelip giden mesajların tazeliği sağlanmaktadır.

1.1.3. Uç Doğrulama

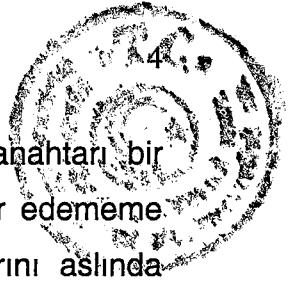
Açık literatürde, [Menezes, et al, 1997] ve [Schneier, 1996], uç doğrulamanın manası genelde belirsizdir. Aslında, uç doğrulama ile bilgi kaynağının doğrulanması arasındaki farklılıklar da belirsizdir. Uç doğrulama, bilgi alış verişi yapılacak tarafın kimliğinin gerçektende doğru olduğundan emin olunması anlamına gelmektedir. Aslında bu özellik, bilgi kaynağının doğrulanması servisi tarafından kapsamaktadır. Bu servisin sağlanması da yukarıdaki servis ile benzer şekilde sağlanır.

1.1.4. Mesaj Bütünlüğünün Sağlanması

Gönderilen bilginin bütünlüğünün sağlanması, gönderilen bilginin alıcı tarafa bozulmadan geldiğinin ispatlanması veya en azından bilginin bozulduğunun anlaşılması manasına gelmektedir. Aslında bilgi bütünlüğünün sağlanması özelliği de bilgi kaynağı doğrulama servisinin bir başka sonucu olarak görülebilir. Bu servisin sağlanması amacı ile genellikle anahtarlı veya anahtarsız özetleme algoritmaları kullanılmaktadır.

1.1.5. İnkâr Edememe

Birçok güvenlik sisteminin hedefinde, tanımlı kullanıcıların dürüst olduğu ve karşısındaki kullanıcıyı aldatmaya çalışmayacağı varsayılır. Gerçek dünyada durum böyle değildir ve tanımlı kullanıcılar da birbirlerini aldatmaya çalışabilmektedir. İnkâr edememe servisi tanımlı bir kullanıcıyı diğer bir tanımlı kullanıcının muhtemel aldatmasından korumayı hedeflemiştir. Bu servis genellikle sayısal imza kullanımı ile sağlanmaktadır. Burada dikkat edilmesi gereken husus imza atılan anahtarın sadece imza



atan taraf tarafından bilinmesinin sağlanmasıdır. Eğer imza anahtarı bir sertifika otoritesi tarafından üretilip kullanıcıya veriliyor ise inkar edememe servisi tam anlamı ile yürütülemez. Çünkü imzalama anahtarını aslında kullanıcı haricinde bilen bazı kişiler olabilir. Bu nedenden dolayı, mesela Amerika'da ki bazı kanunlar, imza atılan anahtarın kullanıcının akıllı kartının içinde üretilmesini ve buradan dışarı çıkarılmamasını zorunlu kılmaktadır. Daha sonra bu imzalama anahtarına karşılık gelen imza kontrol anahtarının bir sertifika otoritesi tarafında kullanıcı ile eşleştirilip sertifika haline getirilmesi gerekmektedir. Burada imzalama işlemi de yine akıllı kartın içinde gerçekleşmektedir ve dışarıya sadece imza verilmektedir. Bu şekilde işleyen bir sistemde, atılmış olan bir imzanın inkar edilmesi söz konusu olamaz, çünkü, imzalama anahtarı sadece ve sadece o kullanıcının akıllı kartında vardır, başkası tarafından bilinmemektedir ve dolayısı ile imzası taklit edilemez, [Schneier, 1996].

1.1.6. Kimliğin Saklı Tutulması

Bazı sistemlerde, örneğin grup imzalama sistemlerinde, işi gerçekleştiren kişinin kimliğinin gözlemcilere gizli tutulması gerekmektedir. Kimliğin saklı tutulması servisi bu amaçla kullanılmaktadır. Gözlemci kişi işi yapan kişinin belirli bir gurup içerisinde olduğunu anlamakla beraber, grup içinden kimin gerçekleştirdiğini bilememektedir.

1.1.7. Kullanılabilirlik

Makul varsayımlar altında bir protokolden hedeflediği servisleri gerçekleştirebilmesi beklenilmektedir. Örneğin, bir anahtar değişim protokolü için, makul varsayımlar altında oturumu için anahtar oluşturabilmesi beklenilmektedir. Bir sistemin böyle bir olguyu garantileyebilmesi için ise, saldırganın kabiliyetlerinin belirlenmesi ve belirli bir düzeyde sınırlandırılması gerekmektedir.

1.2. Güncel Protokoller ile İlgili Yapılan Bazı Çalışmalar



Protokol analizi ve tasarımı yapmak amacı ile açık literatürde bazı formal yöntemler keşfedilmiştir. Bunlardan bazıları, CSP protokol analizcisi (Ardışık İşlemlerin haberleşmesi), NRL protokol analizcisi [Meadows, 1994], vs, bu protokol analizcileri [Meadows, 2000]'de anlatılmıştır. Genelde, kriptografik protokollerin analizini yapmak için her zaman formal yöntemler başarılı olmayabilir. Bu nedenden dolayı kriptografik protokol tasarımcıları çok dikkatli olmalılardır ve açık literatürü ve kriptografik protokoller üzerine yapılmış çalışmaları çok yakından takip etmeleri gerekmektedir. Aksi taktirde, yaptıkları kriptografik protokoller muhtemelen hedeflediği güvenliği sağlamasını engelleyen güvenlik açıkları içerecektir.

[Bellovin, 1996]' de Steven M. Bellovin, IP katmanında çalışan bir şifreleme ve kimlik doğrulama protokolü olan IPSec protokolünün bir önceki versiyonunu incelemiş ve bu protokolde hedeflediği güvenlik servislerini sağlamasını engelleyen güvenlik açıkları keşfetmiştir. Daha sonra IETF (Internet Mühendislik Görev Gücü) Steven M. Bellovin'in bulduğu güvenlik açıklarını gidererek IPSec protokolünü günümüzdeki standart haline getirmişlerdir. Yazar makalesinde protokol tasarımı yapanların standart DES (Veri Şifreleme Standardı) çalışma kipleri ve IV (İlk Vektör) kullanımında çok dikkatli olmaları gerektiğinden bahsetmektedir. Bu makalede ayrıca, güvenliğe yönelik saldırıdan, oturum kaçırma saldırısından, tasarımcıların çok dikkatli olması gerektiği dizi şifreleme algoritmalarının zayıflıklarından bahsedilmektedir ve kimlik doğrulamanın önemi üzerinde durulmaktadır.

[Schneier and Wagner, 1997]'de SSL protokolü incelenmiştir. SSL protokolü İnternet istemci/sunucu haberleşme güvenliğinde kullanılan ve uygulama katmanında çalışan pratik, geniş uygulama alanı bulunan bağlantı değişimli bir mekanizmadır. Bu makale SSL 3.0 protokolünün kriptografik gücünün detaylı teknik analizini vermektedir. Ayrıca, SSL 2.0 da ki problemlerden ve SSL 3.0 dan bu problemlere getirilen çözümlerden bahsedilmektedir. Makalede, bu protokol için hala bazı teknik noktaların

araştırılması gerekmekte beraber genelde SSL 3.0 protokolünün günümüz pratik haberleşme güvenliğine değerli bir katkı olduğu vurgulanmaktadır.

[Borisov et al, 2001]'de kablosuz ağ standardı 802.11'da yer alan Kablolu Denk Gizlilik (WEP) protokolünün ciddi bazı güvenlik açıkları içerdiğinden ve acilen değiştirilmesi gerektiğinden bahsetmektedir. Bu protokol bağlantı katmanında haberleşme güvenliği sağlamaktadır, Makalede bulunan zayıflıkların kriptografik araçların yanlış kullanımından kaynaklandığı belirtilmektedir. Bu güvenlik hataları bu protokolün hedeflediği güvenliği sağlamasını engelleyen birkaç pratik saldırı yöntemine yol açmaktadır. Bu makalede yapılan hatalar ve saldırı yöntemleri detaylı anlatılmaktadır.

1.3. Gösterim

Bu çalışmada simetrik tabanlı şifreleme işlemi,

$$C = S_Enc_K (P), \quad (1.1.)$$

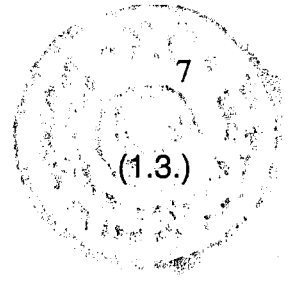
ile gösterilmektedir. Burada S_Enc , simetrik şifreleme algoritmasını, K şifreleme işlemindeki anahtarı, C , şifreli mesajı ve P , açık mesajı göstermektedir.

$$C = P_Enc_{açık} (P), \quad (1.2.)$$

$$P = P_Dec_{özel} (C),$$

denkleminde ise $P_Enc_{açık}$ açık anahtarlı şifreleme sistemi ve açık anahtar kullanılarak yapılan şifrelemeyi, $P_Dec_{özel}$ özel anahtarlı şifreleme sistemi ve özel anahtar kullanılarak yapılan şifrelemeyi göstermektedir.

Bu çalışmada ayrıca, blok şifreleme sistemlerinde şifreleme işlemi,



$$C_i = B_Enc_{K, IV}(P_i),$$

Şifre çözme işlemi de,

$$P_i = B_Dec_{K, IV}(C_i) \quad (1.4.)$$

şeklinde gösterilmektedir. Bu işlemlerde, B_Enc blok şifreleme algoritmasını, B_Dec ise blok şifre çözme algoritmasını, C_i i-inci şifreli mesaj bloğunu, P_i ise i-inci açık mesaj bloğunu göstermektedir.

Dizi şifreleme sistemlerinde ise şifreleme işlemi,

$$C_i = P_i \oplus \text{AnahtarDizisi}, \quad (1.5.)$$

Karşıtı olan şifre çözme işlemi ise

$$P_i = C_i \oplus \text{AnahtarDizisi}, \text{ ve} \quad (1.6.)$$

$$\text{AnahtarDizisi} = S_Enc_{K, IV}(), \quad (1.7.)$$

şeklinde gösterilmektedir. Bu işlemlerde, S_Enc dizi şifreleme algoritmasını, C_i i-inci şifreli mesaj bloğunu, P_i ise i-inci açık mesaj bloğunu göstermektedir.

Ayrıca her iki şifreleme sisteminde de “İlk Vektör” (IV) algoritmanın ilklendirilmesinde kullanılmaktadır, K ise şifreleme işleminde kullanılan anahtarı göstermektedir.

1.4. Güvenlik Protokolü Tasarımında En Uygun Çalışma Kipinin Seçilmesi ve Analizi

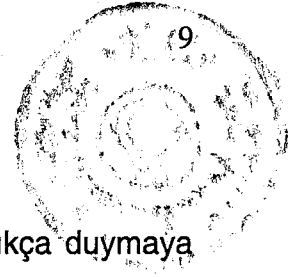
DES blok şifreleme algoritması çalışma kipleri ile beraber, bölüm 2. ve bölüm 3.'de anlatılacağı üzere uzun süredir kullanımda olan bir algoritma

olmakla beraber bu algoritmanın standart çalışma kiplerinin bazı zayıflıkları bulunmuştur. DES algoritmasının yerine geçen AES blok şifreleme algoritması ile beraber, DES çalışma kiplerinin yerine yeni AES çalışma kiplerinin belirlenmesi gerekmektedir. Bu amaç doğrultusunda NIST, AES algoritmasının çalışma kiplerini belirlemek için bir forum düzenlemektedir. Önerilen çalışma kipleri arasında en göze çarpanlar arasında OCB ve CTR gibi kipler bulunmaktadır. Bu çalışma kipleri içerisinde DES çalışma kiplerinde ki zayıflıkların bazılarının giderilmiş ve örneğin OCB çalışma kipinde olduğu gibi şifreleme ve bütünlüğü beraber sağlamak gibi çok güzel fikirler kullanılmıştır.

Fakat yeni çalışma kipleri literatürde, eski çalışma kiplerindeki şifrelemeye karşı yapılan bazı aktif ve pasif saldırı yöntemlerine açısından değerlendirilmemiştir.

Bu çalışmada DES ve AES çalışma kipleri detaylı bir şekilde incelenmiştir. Ayrıca, söz konusu çalışma kipleri güvenlik protokollerine yapılan kes-yapıştır gibi saldırı yöntemleri açısından değerlendirip karşılaştırılmıştır ve tasarlanan protokole uygun çalışma kipinin seçimi anlatılmıştır.

Sonuç olarak DES çalışma kiplerinde görülen bazı zayıflıkların AES çalışma kiplerinde olmadığı ispatlanmıştır. Ayrıca OCB, CTR ve CBC çalışma kipleri performans açısından da karşılaştırılmıştır.



2. KRİPTOLOJİK SİSTEMLER

Özellikle İnternet teknolojisinin yaygınlaşmasıyla adını sıkça duymaya başladığımız kriptoloji gitgide önem kazanmaktadır.

Yunanca Kryptos Logos (Gizli kelime) tamlamasından gelen Kriptoloji, toplumda haberleşmenin gizli tutulabilmesi üzerinde çalışan bilim olarak düşünülse de aslında anlamı daha geniştir, çözülebilmesi çok zor matematik problemleri ve mekanizmaları inceleyen Kriptolojiyi ve bu problemleri ve mekanizmaları çözmeyi hedefleyen ve saldırıları belirleyen kriptanalizi içermektedir.

Şifreleme, veriyi bir anahtarla kodlamaya verilen addır. Burada amaç, veriyi, şifreleme işleminde kullanılan anahtar bilgisi olmadan çözülebilmesi imkansızla mümkün olduğunca yakın şekilde kodlamaktır. Şifre çözme işlemi ise şifrelenmiş veriyi çözüp eski haline getirme işlemidir. Kriptolojide en çok kullanılan kelimelerden biri olan anahtar ise bir metni şifrelemekte veya açmakta kullanılan veri parçasına (sayı, kelime veya herhangi bir sayısal veri parçası) verilen isimdir.

Kriptoloji tarihte de çeşitli zamanlarda kullanılmıştır. Mesajın içerisindeki harflerin belli bir sayı kadar ötelenmesinden ibaret olan Sezar şifresi bundan binlerce yıl önce kullanılmaktaydı. Üç harf ötelemeli Sezar şifresinde "merhaba" kelimesi "öğtjçdç" haline gelmektedir. Bilgisayar programlayan pek çok kişinin bildiği benzer başka bir yöntem ise, bir mesajı belli bir anahtar kelime tekrarı ile modül 2 toplama işlemine tabi tutunca (XOR işlemi, işleme tutulan iki ikil birbirinin aynı ise 0, farklı ise 1 sonucunu veren bir Boole cebri işlemidir) şifrelenmiş, gene aynı anahtarı bu şifrelenmiş metine uygulayınca şifre çözülmüş olmaktadır. Bu tür basit şifreleme metotlarını kırmak için fazla çaba göstermeye gerek yoktur, iki metot da yeterince uzun bir şifreli metin elde olduğu zaman metinde geçen harfler sayılarak hangi harfin ne miktarda çıktığı hesaplanmaktadır, daha sonra her dilin kendine has olan harf kullanım oranları ile karşılaştırılarak kelimeler deşifre edilebilmektedir. Modül 2 toplama şifresinin en zayıf yönü ise sıfırlardan oluşan bir dizinin herhangi bir



anahtar ile modül 2 toplanması sonucunda sonucun anahtarın kendisi olmasıdır. Microsoft firması Word 6.0'a kadar şifreli olarak saklanan metinler için bu en basit XOR metodunu kullanmıştır üstelik anahtar gizlemeye gerek bile duymadan dosyanın içinde belirli bir pozisyona yerleştirerek kullanmıştır. (Microsoft Word yardım dosyasının sıkça sorulan sorular bölümünde şifrelenmiş bir metnin anahtar unutulduğu taktirde asla açılmayacağı yazar) Bu tür şifreler kişisel haberleşmelerde belli bir seviyede güvenlik sağlasa da ticaret gibi ciddi konularda kullanılmaları kesinlikle düşünülmemelidir.

İki tür krypto sistemi tanımlanmıştır. Birincisi gizli anahtarlı simetrik krypto sistemleri; bu sistemlerde, blok şifreleme algoritmaları, dizi şifreleme algoritmaları olmak üzere ikiye ayrılmaktadır. Simetrik krypto sistemleri, haberleşen taraflarda sadece iki tarafın bildiği bir gizli anahtar bulunmaktadır. Mesela, A (Ali) ve B (Bekir) bu tür bir şifreleme metodu kullanarak haberleşeceği zaman önce ortak ve gizli bir anahtar seçmektedirler, A, B'ye mesaj göndereceği zaman mesajı bu ortak gizli anahtarla şifrelemektedir ve B'de aynı anahtarla mesajı çözüp okumaktadır. DES, IDEA, RC5, CAST, BLOWFISH, 3DES, AES en çok kullanılan ve bilinen blok şifreleme algoritmalarıdır. RC4' da en çok bilinen ve kullanılan, yazılım olarak çok az yer tutan ve hızlı şifreleme yapabilen bir dizi şifreleme algoritmasıdır. Gizli anahtarlı simetrik krypto sistemlerinin avantajı hızlı olmasıdır, dezavantajı ise ortak anahtarların belirlenmesi ve taraflara iletilmesindeki zor olmasıdır.

İkinci metot açık anahtarlı asimetric krypto sistemleridir. Bu sistemlerde iki taraf da ikişer anahtara sahiptir. Anahtarlarından biri özeldir, kendisinden başka hiç kimse bilmez, açık anahtar ise haberleştiği herkes tarafından bilinmektedir. Ali ve Bekir bir açık anahtar sistemi ile haberleşeceği zaman, önce Ali Bekir'in açık anahtarını kullanarak mesajını şifrelemektedir. Bekir de gelen mesajı kendi özel anahtarını kullanarak açmaktadır. Özel anahtarı yalnızca Bekir bildiği için mesajı da yalnızca Bekir okuyabilmektedir.

Ali hiçbir zaman Bekir'in özel anahtarını bilmemektedir ve ihtiyaç duymamaktadır. Açık anahtarlı haberleşmenin avantajı gizli anahtarları iki tarafın bilmesine ihtiyaç duyulmaması, dolayısıyla gizli anahtarlı krypto



sistemindeki anahtar iletimi probleminin olmayışdır. Ancak açık anahtarlı kriptu algoritmaları gizli anahtarlılara göre 10 ila 100 kat daha yavaştır ve aslında bir bloktan fazla şifreleme yapmak için kullanılmaları da güvenlik açısından sakıncalıdır. Bu tip algoritmalar daha çok anahtar değış tokuş amacı ile kullanılmaktadırlar. En iyi bilinen açık anahtar algoritması RSA'dır.

Bu iki sisteminde birbirlerine avantaj ve dezavantajları olması, iki sistemin de beraber kullanıldığı hibrit sistemleri ortaya çıkarmıştır. Bunlardan en popüleri, kodu açık olan ve bedava dağıtılan PGP (Pretty Good Privacy) sistemidir. PGP kullanarak haberleşen Ali ve Bekir'in mesajlarının nasıl işlendiğine bakılacak olursa; Ali her mesajı için farklı bir gizli anahtar üretmektedir, bu rasgele yapılan fare ve klavye hareketleri ile belirlenmektedir. Öncelikle Ali'nin mesajı sıkıştırılıp, oluşturulmuş gizli anahtarla ve DES gibi simetrik bir şifreleme algoritması kullanılarak şifrelenmektedir. Daha sonra bu anahtar Bekir'in açık anahtarı ile ve RSA gibi bir algoritma kullanarak şifrelenip mesaja iliştirilmektedir. Bekir, kendisine gelen mesajdaki şifreli anahtar kendi özel anahtar ile açıp, açılan anahtar ve simetrik algoritmayı kullanarak da mesajın geri kalan kısmını çözmektedir. [Schneier, 1996] ve [Menezes, et al, 1997] kriptoloji konusunda çok iyi referans kitaplarıdır.

2.1. Simetrik Şifreleme Sistemleri

Simetrik şifreleme sistemlerinin blok ve dizi şifreleme olmak üzere ikiye ayrıldığından yukarıda bahsedilmişti, burada bu sistemleri biraz daha detaylandırılacaktır. Blok şifreleme sistemleri adında anlaşılacağı gibi şifrelenecek açık veriyi bloklar halinde işlemektedir. Dizi şifreleme sistemlerinde ise anahtar ve algoritma kullanılarak periyodu çok uzun olan diziden istenildiği kadar üretilip, üretilen bu dizi açık mesaj ile modül 2 toplanarak şifrelenmiş olmaktadır.



2.1.1. Blok Şifreleme Algoritmaları

Bu tür algoritmalar veriyi bloklar halinde işlemektedirler. Blok şifreleme algoritmalarının en meşhuru DES algoritmasıdır. DES sistemi "Data Encryption Standart" (Veri Şifreleme Standardı) nın kısaltmasıdır ve fips46-1 numaralı Amerikan federal veri işleme standardıdır. Standart DES veri şifreleme algoritmasını tanımlar. IBM tarafından Lucifer adıyla 1970'lerin başlarında geliştirilmiştir. NSA (National Security Agency- Ulusal Güvenlik Ajansı) ve NIST (National Institute of Standards and Technology) tarafından desteklenmiştir. En çok bilinen ve kullanılan simetrik kriptoloji algoritmasıdır. DES 56 ikilik bir anahtar kullanır. DES sistemine karşı çeşitli saldırı metotları geliştirilmiş olsa da kırılması için en uygun yol en basit olanı, yani brute-force metodudur (kaba kuvvet), Bu metoda DES sistemi için 56 ikilik anahtarların tamamının ($2^{56} = 64$ trilyon) denenmesinden ibarettir. Uzun yıllar kullanıldıktan sonra 56 ikilik DES şifresi uygun donanım sağlandığı takdirde günümüzde birkaç saatlik çabalarla rahatlıkla kırılabilmektedir.

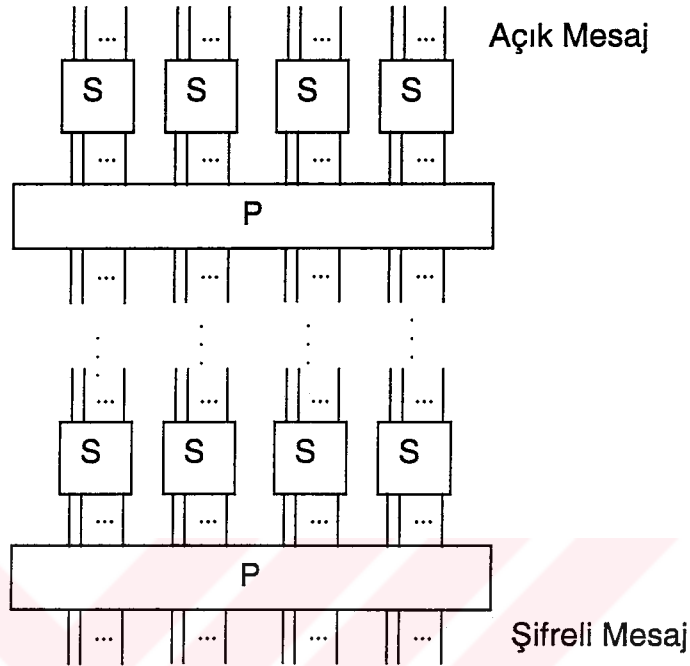
DES algoritmasının yapısını daha yakından incelersek, DES blok şifreleme algoritmasının tasarımı "Çarpım" ve "Feistel" tipi genel kavramları ile yakından alakalıdır.

Çarpım ve Feistel Şifreleme Sistemleri:

Çarpım şifreleme sistemlerinin temel fikri, tek başlarına yetersiz güvenlik sağlayan, fakat birleşimi daha güvenli olan birkaç basit operasyonun birleşimini kullanarak karmaşık bir şifreleme fonksiyonu oluşturmaktır. Bu basit işlemler, yerini değiştirme, modül 2 toplama gibi çevirici ve doğrusal dönüşüm, aritmetik işlemler, modüler çarpma ve basit yerine koyma gibi işlemleri içermektedir.

Çarpım şifreleme sistemi, tek başına yeterli güvenliği sağlayamayan iki veya daha fazla dönüşümü bir araya getirerek, yalnız başlarına sağlayamayacakları güvenliği sağlama işlemidir.

Yerine koyma permütasyonu ağı, bir dizi yerine koyma ve permütasyon evrelerini içeren bir çarpım şifreleme sistemidir.



Şekil 2.1.:Yerine Koyma Permütasyon Ağı'nın Şematik Gösterimi.

Yinelenen blok şifreleme, yuvarlanan fonksiyon adı verilen dahili bir fonksiyonun ardışık olarak tekrar etmesi şeklinde çalışan bir blok şifreleme sistemidir.

Feistel şifreleme sistemi, t ikil uzunluklarındaki L_0 , R_0 'den oluşan $2t$ ikil uzunluğundaki açık mesajları (L_0, R_0) , $r \geq 1$ olmak üzere r yineme sonrasında (R_r, L_r) şifreli mesajlarına dönüştüren bir yinelenen blok şifreleme sistemidir. $r \geq i \geq 1$ olmak kaydı ile i . Çevirim $(L_{i-1}, R_{i-1}) \xrightarrow{K_i} (L_i, R_i)$ işlemini $L_i = R_{i-1}$ ve $R_i = L_{i-1} \oplus f(R_{i-1}; K_i)$ olacak şekilde eşlemektedir. Bu işlemdeki K_i , her çevrimde şifreleme işlemindeki asıl anahtar K 'dan elde edilmektedir.

Tipik bir Feistel sisteminde genellikle $r \geq 3$ ve çift bir sayıdır. Feistel yapısının özelliği, şifreli mesajları (L_r, R_r) olacak şekilde değil de (R_r, L_r)

şeklinde çıkarmasıdır. Son çevrimden sonra bloklar eski dizilişlerindeki hallerine getirilmektedir.

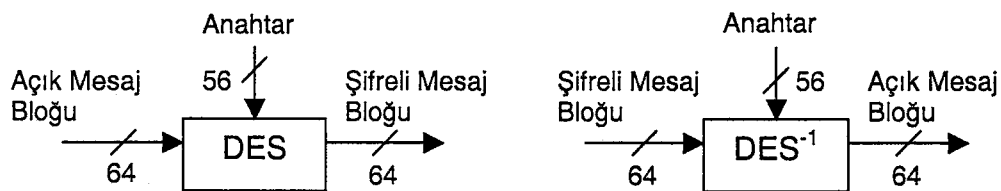
Şifre çözme işleminde ise aynı miktarda çevrim ve alt anahtarların ters olarak alınması ile gerçekleşmektedir.

DES Blok Şifreleme Algoritması:

DES blok şifreleme Algoritması Feistel tipinde bir şifreleme sistemi olup etkin uzunluğu 56 ikil olan anahtar kullanarak, 64 ikilden oluşan açık mesaj bloklarını işlemekte ve 64 ikil uzunluğunda şifreli mesaj blokları elde etmektedir. Aslında DES algoritmasının anahtarı 64 ikilden oluşmaktadır, fakat 8, 16, ... ve 64 üncü ikiller eşlik ikilleri olarak kullanılmaktadır. Dolayısı ile DES algoritmasının etkin anahtar uzunluğu 56 ikile düşmektedir. Her yerde inanılan ortak görüş eşlik ikillerinin kullanım amacının DES algoritmasının etkin anahtar uzunluğunu 64 ikilden 56 ikile indirmek olduğu yönündedir.

$$L_i = R_{i-1}; \quad (2.1.)$$

$$R_i = L_{i-1} \oplus f(R_{i-1}; K_i), f(R_{i-1}, K_i) = P(S(E(R_{i-1}) \oplus K_i)) \quad (2.2.)$$



Şekil 2.2.: DES Algoritmasının Giriş ve Çıktıları.

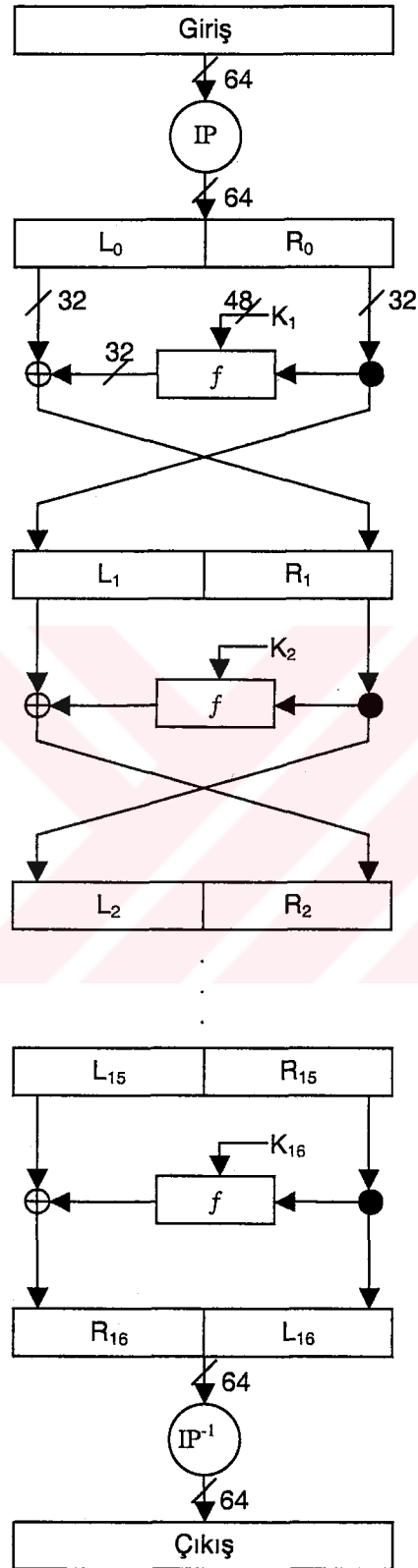
Veri Şifreleme Standardı (DES) Algoritması

GİRİŞ: açık-mesaj $m_1 \dots m_{64}$; 64 ikil anahtar $K = k_1 \dots k_{64}$ (8 eşlik ikili ile).

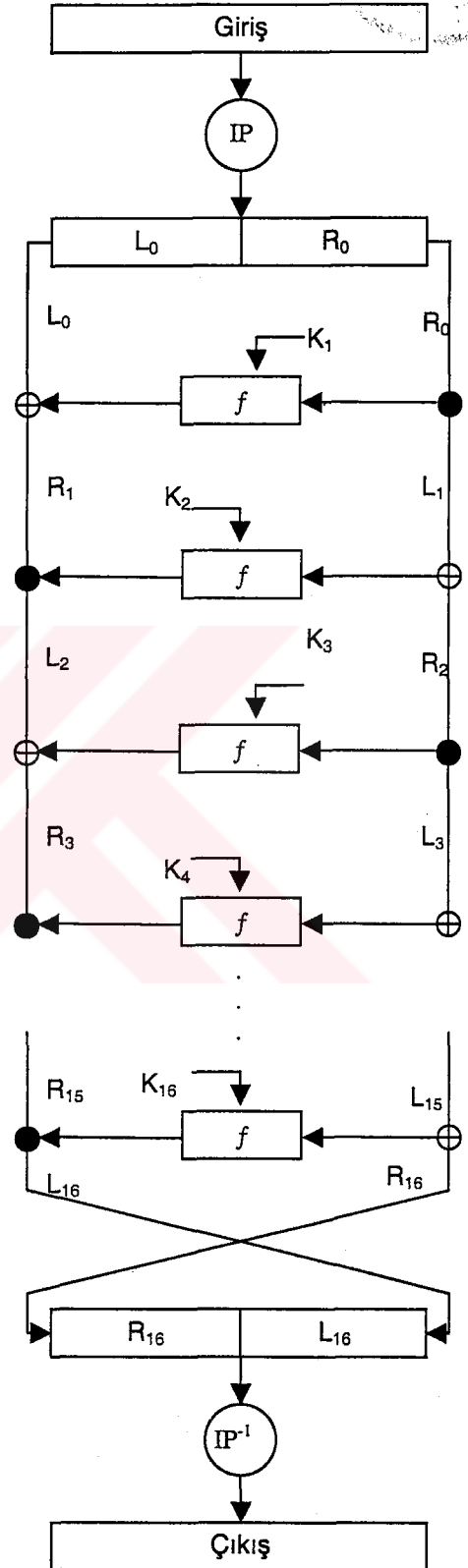
ÇIKIŞ: 64 ikil şifreli mesaj blokları $C = c_1 \dots c_{64}$.

- 1) (Anahtar Zamanlaması) K'dan Algoritma anahtar zamanlaması algoritması kullanılarak 16 tane 48 ikil çevrim anahtarları K_i üretilir.
- 2) $(L_0, R_0) \leftarrow IP(m_1 m_2 \dots m_{64})$. (Çizelge 2.1.kullanılarak açık mesaj permütasyondan geçirilir. Sonuç sağ ve sol olmak üzere 32 ikile ayrılır. $L_0 = m_{58} m_{50} \dots m_8$ ve $R_0 = m_{57} m_{49} \dots m_7$.)
- 3) (16 çevrim) $i, 1$ 'den 16'ya kadar (16 dahil), (2.1.) ve (2.2.) denklemler kullanılarak L_i ve R_i 'ler hesaplanır. Şekil 2.4.'de gösterilen $f(R_{i-1}, K_i) = P(S(E(R_{i-1}) \oplus K_i))$ aşağıdaki gibi hesaplanır:
 - a) $R_{i-1} = r_1 r_2 \dots r_{32}$, Çizelge 2.3. kullanılarak 32'den 48 ikile çıkartılarak genişletilir.
 $T \leftarrow E(R_{i-1})$, ($T = r_{32} r_{17} r_2 \dots r_{32} r_{17}$.)
 - b) $T' \leftarrow T \oplus K_i$. T' as sekiz tane 6 ikil karakter dizisi şeklinde gösterilir.
 $(B_1, \dots, B_8) = T'$.
 - c) $T'' \leftarrow (S_1(B_1), S_2(B_2), \dots, S_8(B_8))$. (Burada $S_i(B_i)$, $B_i = b_1 b_2 \dots b_6$ altı ikilini Çizelge 2.4.'de görüldüğü gibi 4 ikil ile eşler. Burada, $r = 2 b_1 + b_6$, ve c ise $b_2 b_3 b_4 b_5$ 'nin tamsayı değeridir, $0 \leq c \leq 15$. Örnek olarak $S_1(011011)$ için $r = 1$, $c = 13$, ve çıktıda 5'dir (ikil olarak 0101).
 - d) $T''' \leftarrow P(T'')$. (Çizelge 2.3.'deki permütasyon kullanılarak $T'' = t_1 t_2 \dots t_{32}$ 32 ikil $t_{16} t_7 \dots t_{25}$ 'ye dönüştürülür.)
- 4) $b_1 b_2 \dots b_{64} (R_{16}; L_{16})$. (Son bloklar yer değiştirilir, $L_{16}; R_{16}$.)
- 5) $C \leftarrow IP^{-1}(b_1 b_2 \dots b_{64})$. (IP^{-1} için Çizelge 2.2. kullanılır, $C = b_{40} b_8 \dots b_{25}$.)

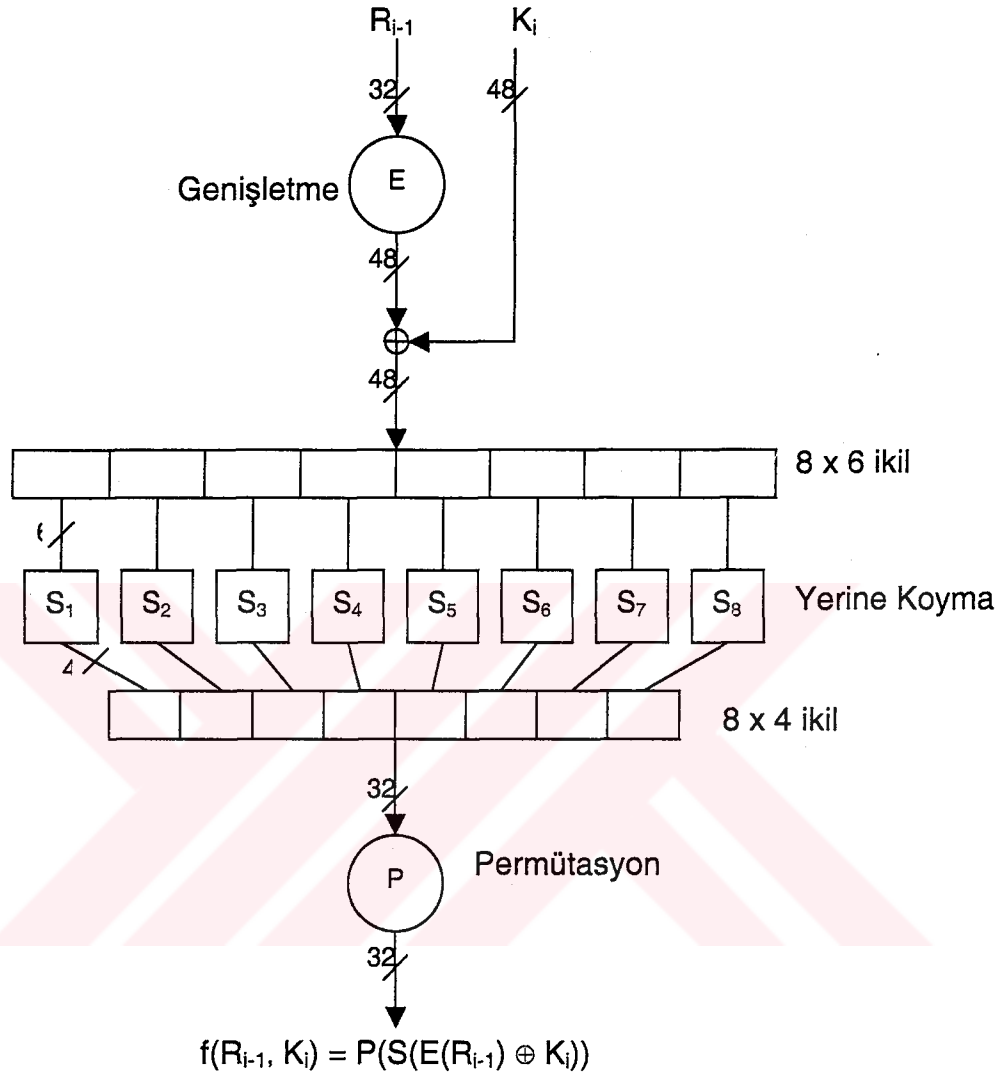
a) Çaprazlamalı Merdiven Modeli



b) Çaprazlamasız Merdiven Modeli



Şekil 2.3.: DES Algoritmasının İşlem Yollarının Şematik Gösterimi



Şekil 2.4.: DES Algoritmasının İç Fonksiyonunun Şematik Gösterimi.

DES Algoritmasının Anahtar Zamanlama Algoritması

GİRİŞ: 64 ikil anahtar (8 eşlik ikili ile).

ÇIKIŞ: 16 tane 48 ikil anahtarı K_i , $1 \leq i \leq 16$

- 1) v_i , $1 \leq i \leq 16$ aşağıdaki gibi tanımlanır. Eğer $i \in \{1, 2, 9, 16\}$ ise $v_i = 1$, değil ise $v_i = 2$

2) $T \leftarrow PC1(K)$. T, 28 ikilik iki parçaya bölünür, $(C_0; D_0)$. PC1 için Çizelge

2.4. kullanılır, $K: C_0 = k_{57}k_{49} \dots k_{36}$, $D_0 = k_{63}k_{55} \dots k_4$.

3) $i, 1$ 'den 16 'ya kadar (16 dahil), $K_i: C_i \leftarrow (C_{i-1} \ll v_i)$, $D_i \leftarrow (D_{i-1} \ll v_i)$,
 $K_i \leftarrow PC2(C_i, D_i)$ hesaplanır. PC2 için Çizelge 2.4., C_i 'nin $b_1b_2 \dots b_{56}$ ve
 D_i 'nin birbirine eklenmesinden 48 ikil seçmek için kullanılır: $K_i = b_{14}b_{17} \dots$
 b_{32} . ' $\ll$ ' dairesel ötelemeyi ifade etmektedir.

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

Çizelge 2.1.: DES Algoritmasının İlk Permütasyon Çizelgesi.

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

Çizelge 2.2.: DES Algoritmasının İlk Permütasyonunun Tersinin Çizelgesi.

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

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

Çizelge 2.3.: DES Algoritmasının Çevrimlerinde Kullanılan, E Genişletme ve P Permütasyon Çizelgeleri.

PC1						
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
Yukarısı C_i için Aşağısı D_i için						
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

PC2					
14	17	11	24	1	5
3	28	15	6	21	10
23	19	12	4	26	8
16	7	27	20	13	2
41	52	31	37	47	55
30	40	51	45	33	48
44	49	39	56	34	53
46	42	50	36	29	32

Çizelge 2.4.: DES Algoritmasının Anahtar Zamanlama Permütasyon Çizelgeleri.

Satır	Sütun numarası															
	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]	[12]	[13]	[14]	[15]

S1

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

S2

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

S3

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

S4

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

S5

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

S6

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

S7

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

S8

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

Çizelge 2.5.: DES Algoritmasının S-Kutuları.



DES 1970 yılların sonuna doğru çıkmış bir algoritma olmasından dolayı bugüne kadar çok ciddi ve uzun süredir açık literatürde analiz edilmiş ve çok ciddi bir zayıflığı bulunamamıştır. Yapılan en ciddi saldırı yöntemleri "Fark kript analizi" ve "Doğrusal kript analizi" saldırı yöntemleridir. Fark kript analizi yönteminde yaklaşık olarak seçilebilen 2^{40} açık mesaj ve karşılığı kapalı mesaj bloğuna ihtiyaç duyulmaktadır. Doğrusal kript analizi yöntemi ise biraz daha az miktarda şifreli mesaj bloğuna ihtiyaç duymaktadır ve Fark kript analizi yöntemine göre biraz daha başarılıdır. Fakat bu saldırı yöntemleri ile taranacak anahtar uzayı çok az miktarda daraltılabilmektedir. Dolayısı ile bu saldırı yöntemleri pratik olarak mümkün görünmemekle beraber aslında bu algorithmada bazı zayıflıkların olduğunu göstermektedir. Daha sonra bu saldırı yöntemleri blok şifreleme algoritmalarının tasarım ve testlerinde en çok dikkat edilen saldırılardan olmuşlardır.

DES algoritmasının değiştirilme ihtiyacı çok uzun süre piyasada kaldığından dolayı artık anahtar ve blok uzunlukları günümüz teknolojisi göz önüne alındığında yetersiz kaldığından dolayıdır. 56 ikil uzunluğundaki DES algoritmasının anahtar uzayının taranma işlemi günümüzde bazı özel yapım bilgisayarlar ile saniyeler mertebesine inmiş durumdadır. Bundan dolayı artık DES yerine anahtar uzunluğu daha fazla olan DES algoritmasının üç defa çalıştırılması anlamında 3DES algoritması kullanılmaktadır. Bu algoritmanın etkin anahtar uzunluğu çift anahtar kullanıldığı durumda 112 ikildir ve günümüz teknolojisi ile böyle bir anahtar uzayının makul bir zamanda taranması imkansız görünmektedir. Anahtar uzayındaki artışa rağmen 3DES algoritmasının blok uzunluğu hala 64 ikil kalmıştır. Bu blok uzunluğu bütünlük koruma bilgisi MAC'ın, 3DES-CBC ile yapıldığı düşünülürse çok az kalmaktadır. Çünkü ortada buluşma saldırı yöntemi ile bütünlük MAC bilgisinde 2^{32} de bir ihtimal ile eşdenik üretmek mümkün gözükmemektedir. Bu miktar ise gerçekten çok düşüktür.

Bunun yanında her bir açık mesaj bloğunun şifrelenmesi için DES blok şifreleme algoritmasının üç kere çalıştırıldığı düşünülürse 3DES algoritması gerçekten yavaş bir algoritmadır. Günümüzde veri büyüklüklerinin ne kadar çok arttığı düşünülürse, özellikle anında şifreleme ve şifre çözmenin

gerekli olduğu çevrimdeki hatlarda hızlı şifreleme ve şifre çözme ihtiyacı düşünülürse, daha hızlı çalışan bir algoritmaya ihtiyaç duyulduğu anlaşılabacaktır.

Ayrıca DES algoritmasının tasarımının IBM firması ile beraber NSA tarafından yapıldığı bilinmektedir. Bugüne kadar yapılan analizlerde önemli bir zafiyeti bulunamamasına karşın, bir arka kapısının olup olmadığından şüphelenilmektedir, [Schneier, 1996] ve [Mel and Baker, 2001].

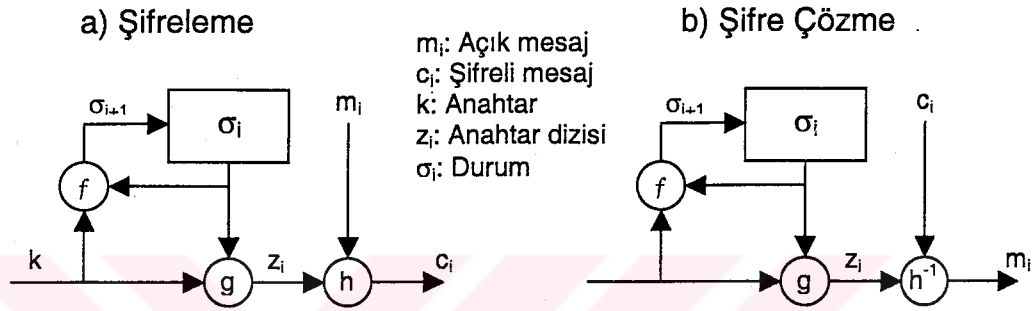
Bu nedenlerden dolayı NIST tarafından anahtar ve blok uzunluğu günümüz şartlarına daha uygun bir algoritma oluşturmak için 1998 yılında bir çalışma yapılmaya başlanılmıştır. NIST bu amaçla bütün dünyadan isteyen herkesin kendi tasarladığı blok şifreleme algoritmasını önerebileceği bir yarışma başlatmıştır ve seçim işleminde ve göz önüne alınacak kriterler yine bütün dünyanın gözü önünde şeffaflık içinde sürdürülmüştür. NIST böylelikle DES algoritmasının kapalı kapılar arkasında tasarlanmasından kaynaklanan kötü ününü gidermeye çalışmıştır. Sonuçta 2001 yılında yarışma sonuçlanmış ve Belçikalı iki tasarımcının ürünü olan Rijndael blok şifreleme algoritması birinci seçilerek AES algoritması olmuştur.

Bu algoritma fips-197 olarak standartlaşmıştır [Fips197, 2001]. Rijndael blok şifreleme algoritmasının tasarım kriterleri DES blok şifreleme algoritmasının tasarım kriterlerinden oldukça farklıdır. Bu algoritmanın detayları ve tasarım kriterleri için [Daemen and Rijmen, 1998] ve [Daemen and Rijmen, 2000]'e bakınız.

Rijndael blok şifreleme algoritması AES olarak standartlaştıktan sonra, bu algoritmaya yapılan analizlerde arttı. Bunu sonucu olarak bu algoritmanın dokuz çevrime kadar azaltılarak zayıflatılmış sürümlerine karşı bazı saldırı yöntemleri bulunmuştur. Zaman NIST'in Rijndael blok şifreleme algoritmasını AES olarak seçmesinin doğrumu yanlış mı olduğunu gösterecektir. Umarız bu algoritma kullanımı iyice yaygınlaştıktan sonra zayıf bulunup değiştirilmeye çalışılmaz.

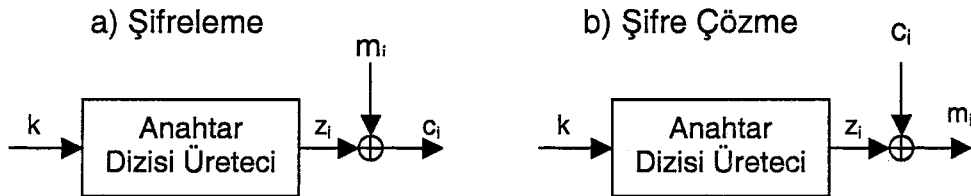
2.1.2. Dizi Şifreleme Algoritmaları

Dizi şifreleme sistemleri, blok şifreleme sistemlerinin aksine açık metni genellikle ikil bazında işleme tutmaktadır. Dizi şifreleme sistemleri eş zamanlı ve kendiliğinden eş zamanlı olmak üzere ikiye ayrılır. Eş zamanlı dizi şifreleme sistemlerinde anahtar dizisi, açık mesaj ve şifreli mesajdan bağımsız olarak üretilmektedir. Şekil 2.5.'de eş zamanlı dizi şifreleme sisteminin genel yapısı verilmektedir.



Şekil 2.5.: Eş Zamanlı Dizi Şifreleme Sisteminin Genel Yapısı.

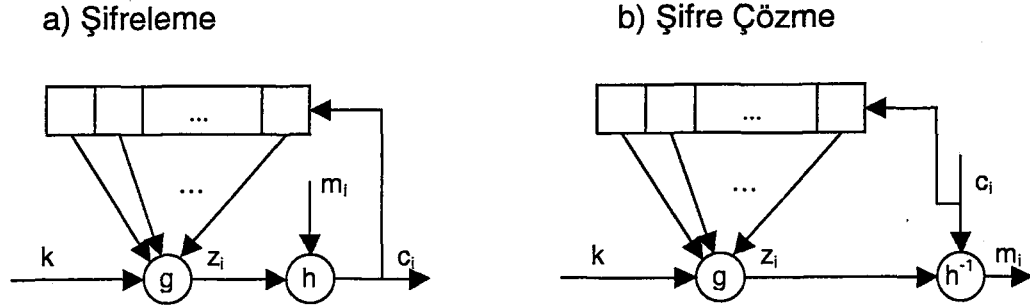
Bu tip dizi şifreleme sistemlerinde alıcı ve gönderici arasında uyumluluğun sağlanması gerekmektedir. Bundan dolayı, saldırganın araya ikil sokması veya silmesinden dolayı uyum bozulmaktadır ve tekrardan sağlanması gerekmektedir. Bozulan şifreli mesaj ikili sadece açık mesajın o ikiline karşılık gelen ikili bozmaktadır.



Şekil 2.6.: İkili Ekleme Dizi Şifreleme Sistemleri.

Bu tip dizi şifreleme sistemlerinin en çok kullanılanları ikili ekleme dizi şifreleme sistemleridir. Sözü edilen işlem Şekil 2.6.'da da görüldüğü gibi modül 2 toplamadır.

Kendiliğinden eş zamanlı dizi şifreleme sistemlerinde ise anahtar dizisi anahtar ve belirli miktardaki önceki şifreli mesaj ikililerinin bir fonksiyonu olarak üretilmektedir. Şekil 2.7.'de kendiliğinden eş zamanlı dizi şifreleme sisteminin genel yapısı verilmektedir.



Şekil 2.7.: Kendiliğinden Eş Zamanlı Dizi Şifreleme Sisteminin Genel Yapısı.

Bu tip dizi şifreleme sistemlerinden alıcı ve gönderici arasındaki uyum otomatik sağlanmaktadır. Şifreli mesajdaki bir ikil bozulma mesaja geri besleme miktarı artı bir şekilde yansımaktadır. Dolayısı ile saldırganın yaptığı değişiklikler, büyüyerek açık mesaja geçmektedir. Buda aktif saldırıların belirlenmesinde yardımcı olmaktadır. Bu tip dizilerde önceki şifreli mesaj ikileri şifrelemede kullanıldığından şifreli mesajın istatistiği daha iyi olmaktadır. Ayrıca aynı anahtar ile yapılan şifrelemelerde aynı anahtar dizisi kullanımından da kaçınılmış olmaktadır.

2.2. Açık Anahtarlı Şifreleme Sistemleri

1977 yılında yapılan önemli çalışmalardan biri de, Ron Rivest, Adi Shamir ve Leonard Adleman tarafından RSA (soyadlarının baş harfleri) isimli kriptosisteminin icadı edilmesidir. Çok karmaşık olmayan modüler aritmetik ve üs hesaplamalarına dayanan matematiksel işlemlerin sonucunda iki büyük asal sayıdan birer tane açık ve özel anahtar üreten bu kriptosistemin en büyük özelliklerinden biri özel anahtarın açık anahtarı oluşturan parçalardan üretilmesinin olanak dışı olmasıdır.



RSA, DES alternatifi bir algoritma değildir. Bir yukarı ki bölümde anlatıldığı gibi daha çok anahtar dağıtım ve sayısal imza amaçlı bir algoritmadır. RSA ile bir verinin tamamını şifrelemek, gereken işlemlerin çok uzun zaman alması nedeniyle oldukça verimsiz ve ayrıca güvensizdir. Ancak, ondan çok daha hızlı DES gibi simetrik bir algoritma ile verinin şifrelenmesi ve daha sonra bu şifrelemede kullanılan anahtarının RSA ile şifrelenerek şifreli veriye eklenmesi, özellikle anahtar dağıtım işlemi oldukça hızlandırmaktadır.

RSA krypto sistemini kırmak için bir büyük sayıyı oluşturan iki asal çarpanın bulunması gerekmektedir. Önceleri 125 basamaktan oluşan bir sayı için ideal çözümün bulunması bir kaç trilyon yıl sürmekteydi. Fakat 1994'de yapılan bir denemede, dünya üzerindeki 1600 bilgisayarda sekiz ay süren bir çalışma sonucunda, 129 basamaklı bir sayının iki asal çarpanı bulunabilmiştir.

RSA algoritmasına alternatif olması amacıyla bir çok şifreleme sistemi üretildi, fakat bunlardan bir kısmın güvensiz olduğu ortaya çıktı.

ElGamal tarafından üretilen ve Schnorr tarafından geliştirilen bir krypto sistem NIST'in projesinde yer alan DSS imza sistemini oluşturdu. DSA sayısal imzalama algoritması, RSA algoritmasının aksine şifreleme yapmak için kullanılamamaktadır. İmzalanacak veri kriptolojik özetleme algoritmalarından geçirildikten sonra çıkan veri üzerine imza atılabilmekte, imzanın ise sadece doğru olup olmadığı hesaplanabilmektedir. DSA algoritması bu özelliğinden dolayı Amerikan krypto algoritmaları ihraç kuralları dışında tutulmuştur.

Fakat, RSA'nin en büyük avantajlarından birisi hem anahtar değiş tokuşu için kullanılabilmekte, hem de dijital imzayı üretmekte kullanılabilir. DSA algoritması ise sadece sayısal imza üretmek için kullanılabilir.

RSA, ISO (Internet Standards Organization) ve CCITT (Consultative Committee in International Telegraphy and Telephony) tarafından standart bir kriptosistem olarak kabul edilmiştir. İnternet'te PEM (Privacy Enhanced Mail) ve PGP tarafından da kullanılmaktadır. Fakat ABD hükümeti, NSA'ya ABD dışına çıkacak kriptosistemleri kontrol yetkisini tanımıştır ve NSA da neredeyse tüm önemli kriptosistemlerin ABD dışına çıkartılmasını veya bu algoritmaların Amerika dışında belli bir anahtar uzunluğunun üzerinde kullanılmasını yasaklamıştır.

Bu yüzden DES algoritması Amerika dışında uzun yıllar 40 ikil uzunluğunda kullanılmak zorunda bırakılmıştır. Daha sonra Amerikan hükümeti ihraç edilebilen anahtar uzunluğunu 128 ikile çıkartmıştır.

Bir sonraki bölümde güvenlik protokollerine yapılabilecek muhtemel saldırı çeşitlerini inceleyeceğiz.

2.3. Güvenlik Protokollerinin Tehditleri

Burada anlatılacak saldırı tipleri güvenlik protokollerinin ne kadar çeşitli ve zekice hazırlanmış saldırı yöntemlerine maruz kalabileceklerini göstermektedir. Bunların yanında daha birçok bilinen ve büyük bir olasılıkla var olduğundan şüphelenilen daha nice saldırı yöntemleri vardır. Birçok saldırı yöntemi yukarıda anlatılan senaryoların birleşimini kullanmaktadır. Örneğin, bölüm (2.3.5.)'de anlatılan NSPK protokolüne uygulanan Lowe saldırısı arada bırakma ve nasihatçi saldırılarının birleşimini kullanmaktadır. Aslında bir protokoldeki zayıflığı bulmak çok derin bir satranç problemini çözmek gibidir. Çünkü, bağlantılar genellikle çok incedir ve fark edilmesi zekilik, deneyim ve bazen de biraz şans gerektirmektedir, fakat bulunduğu zamanda problem genellikle nettir.

Güvenlik protokollerine karşı uygulanabilecek saldırı yöntemlerini sıralarken, saldırganların kullanabilecekleri muhtemel ve iyi bilinen birkaç strateji değişik tipte saldırı yöntemleri anlatılacak ve incelenecektir. Ayrıca burada sadece protokol tasarımında yapılan hatalardan dolayı

gerçekleştirilebilen saldırı yöntemleri yer alacaktır. Diğer saldırı yöntemleri, kriptanaliz, zamanlama, cihazdaki veya elektrik kullanımındaki değişim ve elektro manyetik yayılmadan kaynaklanan saldırı yöntemleridir.

Kriptografik protokollere karşı yapılan bazı saldırı yöntemleri aşağıda sıralanmıştır. Bunlardan bazıları, aradaki-kişi saldırısı, yansıma saldırısı, tekrar gönderme saldırısı, arada bırakma saldırısı, ileri doğru güvenlik yetersizliği saldırısı ve bazı cebirsel saldırılar olarak sıralanabilir. Protokol analizi ve tasarımı yapan kişiler protokollerinin bu saldırılara karşı güçlü olup olmadığını araştırmalı ve protokollerinde gerekli düzeltmeleri yapmalılardır.

2.3.1. Aradaki Kişi Saldırısı

İsminden de anlaşılacağı gibi bu saldırı yönteminde saldırgan yasal iki kullanıcı arasına girmektedir. Eğer güvenlik protokolü zayıf bir şekilde tasarlanmış ise yasal kullanıcıların gönderdiği mesajları değiştirmekte veya onun gitmesini engelleyip kendi oluşturduğu mesajı gönderebilmektedir.

Bunun daha açık gösterilebilmesi amacı ile Ali'nin doğal bir protokol kullanarak X gizli mesajını Bekir'e birbirlerinin açık anahtarına ihtiyaç duymadıkları açık anahtarlı bir sistem kullanarak göndermek istediği farz edilecek. Şifreleme ve şifre çözme işleminin birbirinin tersi olduğu ve şifreleme sırası değiştirilebilen RSA gibi açık anahtarlı bir algoritma kullanarak aşağıdaki şekilde haberleştikleri farz edilecek.

1. Ali, Bekir'e Alinin açık anahtarı ile şifrelediği mesajı göndermektedir.

$$P_{Enc_{P_{Ali}}}(X) \quad (2.3.)$$

Burada P_{Ali} Ali'nin açık anahtarı anlamına gelmektedir.

2. Bekir bu mesajı aldığı zaman bu mesajın şifresini çözmemektedir, çünkü bu mesaj ancak Alinin özel anahtarı ile çözülebilmektedir.

Yapabileceği şey bu mesajı kendi açık anahtarı ile tekrar şifreleyip Ali'ye göndermektir.

$$P_{EncP_{Bekir}}(P_{EncP_{Ali}}(X)) \quad (2.4.)$$

Burada P_{Bekir} Bekir'in açık anahtarı anlamına gelmektedir.

3.RSA algoritmasının sıra değiştirebilme özelliğini kullanarak,

$$P_{EncP_{Ali}}(P_{EncP_{Bekir}}(X)) \quad (2.5.)$$

elde edilir. Dolayısı ile Ali kendi şifresini çözüp

$$P_{EncP_{Bekir}}(X) \quad (2.6.)$$

mesajını elde eder.

4.Ali bu mesajı Bekir'e gönderdiği zaman bu mesajı sadece Bekir çözebilecektir.

İlk bakışta bu protokolün Ali'nin, Bekir ile güvenli görüşebilmesini sağlayan mantıklı ve çalışır bir protokolmüş gibi gözükmemektedir. Fakat aradaki saldırgan mesajlar üzerinde oynayarak Ali'nin gönderdiği mesajı ele geçirebilmektedir. Saldırı yöntemi aşağıdaki gibidir.

1.Saldırgan (2.4.) numaralı mesajı keser ve bu mesajın üzerine kendi açık anahtarı $P_{saldırgan}$ ile şifreleyerek gönderir.

$$P_{EncP_{saldırgan}}(P_{EncP_{Ali}}(X)) \quad (2.7.)$$

2. Bu mesajı Ali'ye gönderir ve Ali bu mesajın Bekir'den değil de saldırgandan geldiğini hiçbir şekilde bilemez. Rasgele bir dizi bir diğerine benzeyebilir. Dolayısı ile Ali kendi şifresini çözerek Bekir'e

$P_{Enc_{P_{saldirgan}}}(X)$

(2.8.)

mesajını gönderir.

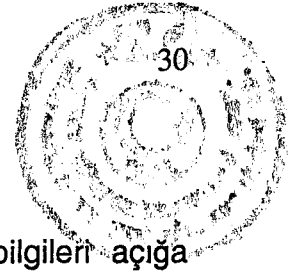
3. Saldırgan bu mesaja kendi özel anahtarı ile şifre çözme işlemine tabi tuttuğunda X mesajını elde edecektir.

Bu saldırı yönteminin çalışmasının nedeni kullanılan protokolde mesaj bütünlüğünü koruyan bir mekanizmanın yer almamasıdır. Ali'nin kendisine gelen mesajın gerçekten de Bekir tarafından gönderildiğini gösteren bir mekanizma bulunmamaktadır. Buna benzer bir problemde mesaj bütünlüğü korunmamış Diffie-Hellman anahtar anlaşımı protokolünde mevcuttur.

2.3.2. Yansıma Saldırısı

Burada kullanılan hile adından da anlaşıldığı gibi yasal bir kullanıcının bir diğerine gönderdiği mesajı saldırgan göndericiye geri göndermektedir. Bazen bu yöntem ile gönderici taraf yanılarak karşı taraftan doğru bir cevap aldığını zannedebilmektedir. Örnek olarak göndericinin "parola nedir?" mesajına "parola nedir?" ile cevap verildiği zaman cevap olarak kendi parolasını gönderebilecektir. Dolayısı ile parolanızı karşıya göndermiş olacaksınız. Bu saldırı size çok basitmiş gibi gelebilir, fakat dost düşman tanıma protokollerinde bu tür saldırılar aktif olarak kullanılmaktadır.

Bu saldırı yöntemi protokoldeki simetriye bağlıdır. Yukarıdaki basit örnekte nöbetçi erin kendini tanıtmaya şartladığını varsayılıyor. Nöbetçilerin ve cevaplayıcıların farklı parolaya sahip olması ile bu simetri bozulabilir.



2.3.3. Nasihatçi Saldırısı

Burada saldırgan, dürüst kullanıcının yanlışlıkla bazı bilgileri açığa çıkarmasını sağlamaya çalışmaktadır. Dürüst kullanıcı protokolün bazı adımlarını, saldırganın başka şekilde elde edemeyeceği bazı bilgileri açığa çıkaracak şekilde işletmesi ile gerçekleştirilmektedir. Burada saldırgan nasihatçi ve protokol işleyişindeki aralığın birleşimini kullanmaktadır.

Bu tip bir saldırıda, saldırgan başka bir zamanda işletilen aynı veya başka bir protokolün adımlarını kullanabilecektir. Farklı bir protokolün adımlarını kullanabilmesi çok nadir bir işlemdir, çünkü, her iki protokolünde benzer yapıya sahip olmaları gerekmektedir. Bazen benzer yapıya sahip olmaları bile yetersiz olabilmektedir. Kullanılan anahtarlarında her iki protokolde aynı olması zaten doğal bir zayıflıktır.

2.3.4. Tekrar Gönderme Saldırısı

Bu saldırı yönteminde protokolün bazı adımlarını izleyecek ve sonraki bir zamanda daha önce kaydettiği protokol adımlarını dürüst kullanıcılara göndererek aldatmaya çalışacaktır. Eğer protokolün değişik zamanlarda yapılan protokol adımlarını tanıyacak veya mesajın tazeliğini kontrol edecek bir mekanizması yok ise saldırgan dürüst kullanıcıyı kandırarak protokolün bazı adımlarını veya hepsini işlettirebilecektir.

Protokolde kullanılacak rasgele sayılar, protokolün o anını tanıyıcı sayaç benzeri mekanizmalar, zaman pulları bu tip saldırıları engellemek amacı ile kullanılmaktadır.

2.3.5. Arada Bırakma Saldırısı

Bu saldırı yöntemi belki de en zekice tasarlanmış saldırı yöntemleridir. Saldırganın iki veya daha fazla bütün protokol işleyişinin bir yolunu bulup örtüşmesini sağlamaktadır.

Burada bölüm (2.3.3.)'de anlatılan nasihatçi ve arada bırakma saldırılarının birleşimi olan meşhur bir saldırı yöntemini anlatılacaktır.

Söz konusu protokol "Needham-Schroeder Public-Key" (NSPK) protokolüdür. Adından da anlaşılacağı gibi açık anahtarlı şifreleme algoritması kullanmaktadır. Bu protokolün işleyişi aşağıdaki gibidir.

$$\begin{aligned}
 \text{Mesaj 1} \quad A &\rightarrow B: P_{\text{Enc}_{P_B}}(A \mid N_A) \\
 \text{Mesaj 2} \quad B &\rightarrow A: P_{\text{Enc}_{P_A}}(N_A \mid N_B) \\
 \text{Mesaj 3} \quad A &\rightarrow B: P_{\text{Enc}_{P_B}}(N_B)
 \end{aligned} \tag{2.9.}$$

İlk adımda Ali rasgele bir sayı olan N_A üretir ve kendi adı ve bu sayıyı Bekir'in açık anahtarı ile şifreleyerek Bekir'e gönderir. Bu mesajın şifresini sadece ama sadece Bekir çözebilmektedir ve kendi ürettiği N_B rasgele sayısını da Ali'nin gönderdiğine ekler ve Ali'nin açık anahtarını kullanarak şifreleyerek Ali'ye gönderir. Ali bu mesajı aldığı anda şifresini çözer ve ilk rasgele sayının kendi ürettiği sayı olup olmadığını kontrol eder, eğer doğru ise, Bekir'in ürettiği rasgele sayı olan N_B 'yi yine Bekir'in açık anahtarı ile şifreleyerek Bekir'e gönderir. Bekir bu mesajı aldığı anda şifresini çözer ve kendi sayısını olup olmadığını kontrol eder. Eğer her şey olunda giderse iki tarafta:

1. Kiminle mesajlaştıklarından,
2. N_A ve N_B sayılarının her ikisinde olduğundan,
3. Başka kimsenin N_A ve N_B değerlerini bilmediğinden emin olacaklardır.

Bu yaklaşım Ali'nin ve Bekir'in açık anahtarları ile şifrelenmiş mesajları sadece kendileri açabileceklerinden makul gibi gözükmemektedir. Ali ve Bekir bu değerleri sonradan tekrar kimlik doğrulama amaçlı veya bu değerlerin özetini alarak gizli haberleşmelerindeki ortak anahtarı oluşturmak için kullanabilecekleridir.



Yıllarca bu protokolün bahsedilen özellikleri sağladığı herkes tarafından kabul görmüştür. Daha sonra bu protokole G. Lowe tarafından yapılan zekice bir saldırı yöntemi bulunmuştur. Bu yöntem aşağıdaki gibidir.

Mesaj $\alpha.1$ $A \rightarrow S: P_{EncP_S} (A | N_A)$

Mesaj $\beta.1$ $S(A) \rightarrow B: P_{EncP_B} (A | N_A)$

Mesaj $\beta.2$ $B \rightarrow S(A): P_{EncP_A} (N_A | N_B)$

Mesaj $\alpha.2$ $S \rightarrow A: P_{EncP_A} (N_A | N_B)$ (2.10.)

Mesaj $\alpha.3$ $A \rightarrow S: P_{EncP_S} (N_B)$

Mesaj $\beta.3$ $S(A) \rightarrow B: P_{EncP_B} (N_B)$

Bu saldırı yönteminde saldırganın aslında tanınan bir kullanıcı olduğu, öbür kullanıcıları tanıdığı ve sertifikalı bir açık anahtara sahip olduğu varsayılmaktadır. Ali'nin suçsuz olduğu ve protokolü saldırganın dürüst bir kullanıcı olarak gördüğü için saldırgan ile haberleşmek için başlattığı farz edilecek. Fakat saldırganın amacı başkadır, Ali'ye beklenen şekilde cevap vermek yerine, Ali'nin gönderdiği rasgele sayıyı kullanarak Bekir ile yeni bir protokol anı başlatacaktır, fakat kendi ismi yerine Ali'nin ismini yazacaktır. $S(A)$ gösterimi, saldırganın sanki Ali hazırlamış gibi göstermek istediği mesajlar için kullanılmaktadır. Bekir bu mesaja dürüst şekilde kendi rasgele sayısı olan N_B ile ve tabi ki Ali'nin açık anahtarı ile şifreleyerek cevap vermektedir. Çünkü, protokolü Ali'nin başlattığını düşünmektedir. Bu mesaj aslında Ali'nin saldırganda beklediği mesajdır ve mesajı çözer ve içindeki N_B rasgele sayısını saldırganın açık anahtarı ile şifreleyerek saldırgana gönderir. Saldırgan kendi özel anahtarını kullanarak bu mesajı çözer ve N_B değerini elde eder. Saldırgan daha sonra N_B sayısını Bekir'in açık anahtarı ile şifreleyerek son mesajı hazırlayarak Bekir'e gönderir.

Dolayısı ile bu protokol adımlarının sonucunda, Ali, saldırgan ile N_A ve N_B değerleri üzerinde anlaştıklarını, Bekir ise Ali ile N_A ve N_B değerleri üzerinde anlaştıklarını sanmaktadır. Bekir, dürüst olan Ali ile protokolü çalıştırdığını sanmaktadır, ama durum hiçte sandığı gibi değildir. Bu saldırı en azından Ali ve Bekir açısından yanlış anlamalara yol açmaktadır.

Bu saldırı yöntemi açık literatürde çok ilgi toplamış ve bazı anlaşmazlıklara yol açmıştır. Sonuç olarak bu protokol gönderilen mesajların gerçekte kimin tarafından gönderildiğini gösteren bir nevi kimlik doğruma yapısına sahip olmadığından dolayı bu saldırıya maruz kalmaktadır. Buda bize kimlik doğrulamanın ne kadar önemli olduğunu göstermektedir.

Buradaki saldırının nasıl çalıştığına dikkatlice bakacak olursak, 2. ve 3. adımlarda saldırgan Ali'yi kullanarak başka şekilde elde edemeyeceği bilgileri elde etmektedir. Dolayısı ile Ali saldırgan tarafından Bekir ile olan protokol adımlarını tamamlamakta kullanılmaktadır.

2.3.6. İleri Doğru Güvenlik Yetersizliği Saldırısı

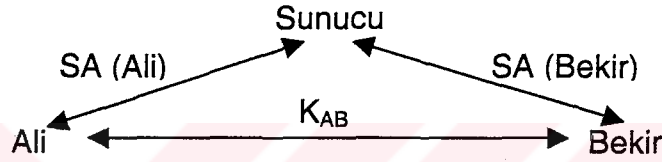
Yukarıda bahsedilen saldırı yöntemlerinin hepsinde de saldırganın herhangi bir kriptoloji işlemi yapması gerektirmemektedir. Saldırgan trafikte gidip gelen mesajlar ile oynayarak protokolün istediği bazı bilgileri kendine vermesini sağlamaktadır. Şifreleme algoritması veya başka nedenlerin sonucundaki başarılı kriptoloji analizinden veya protokoldeki hatalardan dolayı bazen anahtar bilgisinin saldırganın eline geçmesi engellenememektedir. Başka nedenler, hırsızlık, protokol veya şifreleme algoritmalarındaki arka kapılardan ve rüşvet olarak sıralanabilir. Çözümüne yönelik bir iskeletin anahtarın istenmeyen kişilerin eline geçtiğinde ne yapılması gerektiği ve yeni anahtarın kısa zamanda oluşturulması ve dağıtılması gibi işlemleri önceden modellemesi gerekmektedir. Daha sonra kullanıcılar anahtar ele geçmelerini engellemek için ve daha sonraki işlemlerde neler yapılması gerektiğini bildiklerinden dolayı sistemi en kısa zamanda tekrar ayağa kaldıracaklardır.

Bazı sonuçlar açıkça görülmektedir;

1. Açığa çıkmış bir anahtar ile şifrelenen mesajlarda açığa çıkmış demektir.

2. Sisteme elektronik olarak yeni dağıtılacak anahtarlar, eğer daha önce yine elektronik olarak gönderilmiş anahtar kullanılarak şifreleniyor ise gönderilen anahtarlar birbirine zincir gibi bağlı durumdadır. Bu durumda eğer anahtarlardan biri ele geçerse, bu anahtar kullanılarak şifrelenen anahtar, ve bundan sonra elektronik olarak gönderilen anahtarlarda ele geçmiş demektir. Aslında, böyle bir durum için anahtar arasındaki bağlantılarını gösteren çizgenin geçişli kapatması oluşturulmalıdır.

3. Bu açık sonuçların yanı sıra, daha zekice hazırlanmış olan senaryolarda olabilmektedir.



Şekil 2.8.: NSSK Protokolünün Şematik Gösterimi.

Şekil 2.8.'de gösterilen NSSK protokolünde bir sunucu yer almaktadır ve bütün kullanıcıların sunucu ile uzun zamanlı kullanım süresine paylaşılmış gizli anahtarları mevcuttur. Sunucu ile Ali arasında $SA(Ali)$, sunucu ile Bekir arasında $SA(Bekir)$ daha önceden paylaştırılmıştır. Aşağıda NSSK protokolünün adımları yer almaktadır:

Mesaj 1: $A \rightarrow S: A \mid B \mid N_A$

Mesaj 2: $S \rightarrow A: S_Enc_{SA(Ali)}(N_A \mid B \mid K_{AB} \mid S_Enc_{SA(Bekir)}(K_{AB} \mid A))$

Mesaj 3: $A \rightarrow B: S_Enc_{SA(Bekir)}(K_{AB} \mid A)$ (2.11.)

Mesaj 4: $B \rightarrow A: S_Enc_{K_{AB}}(N_B)$

Mesaj 5: $A \rightarrow B: S_Enc_{K_{AB}}(N_B-1)$

N_A ve N_B (2.9.) numaralı denklemde açıklanmaktadır. İlk mesajda Ali, sunucuya Bekir ile görüşmek istediğini bildirmekte ve rasgele oluşturduğu N_A sayısını birlikte yollamaktadır. İkinci mesajda sunucu N_A değerini, Bekir'in

kimliğini, Ali ile Bekir'in aralarındaki şifreli görüşmeyi sağlayacakları K_{AB} anahtarını ve K_{AB} anahtarı ve Ali'nin kimliğinin ard arda eklenmiş haininin sunucu ile Bekir arasında paylaşılmış anahtar olan $SA(Bekir)$ ile şifrelenmiş halinin ard arda eklenmiş hallerinin sunucu ile Ali arasında paylaşılmış anahtar olan $SA(Ali)$ ile şifrelenmiş mesaj olan mesaj 2'yi sunucu Ali'ye gönderir. Üçüncü mesajda Ali, sunucunun göndermiş olduğu mesajın içinde yer alan K_{AB} anahtarının ve Ali'nin kimliğinin ard arda eklenmiş haininin sunucu ile Bekir arasında paylaşılmış anahtar olan $SA(Bekir)$ ile şifrelenmiş mesajı Bekir'e gönderir. Dördüncü mesajda Bekir, kendi oluşturduğu rasgele sayı olan N_B 'yi Alinin bir önceki mesajda göndermiş olduğu K_{AB} anahtarı ile şifreleyerek Ali'ye gönderir. Alide son mesajda bu değerin bir eksiğini aynı anahtar ile şifreleyerek Bekir'e gönderir.

Zayıflatılmış NSSK protokolündeki zayıflıklar bu konudaki en başlıca örneklerdendir. Bu protokolde, herhangi bir açığa çıkmanın daha sonrasına etki etmemesi olan ileri doğru güvenlik yetersizliğinin başlıca amacı çöğnenmektedir. Bu protokolün zayıflatılmış versiyonunda protokolü başlatan tarafın gönderdiği rasgele sayı çıkartılmaktadır, bu durumda başlatıcı taraf kendisine geri gelen mesajı tazeliğini kontrol edecek bir mekanizmadan yoksun hale gelmektedir. Buda saldırganın eski bir protokol adımını kullanarak protokolü başlatanı aldatabileceğini göstermektedir.

Bu çok genel bir hatadır. Bir açığa çıkmadan sonra, oturum anahtarları dahil bütün açığa çıkmış anahtarları değıştirsek bile hala saldırgan için protokolün daha sonraki işleyişini tehlikeye atacak bir yol bulunabilmektedir. Kerberos tarafından bu saldırı yöntemine karşı rasgele sayı yerine zaman pulu yöntemi kullanmak benimsenmiştir. Böyle bir saldırı yöntemi ile başa çıkmanın başka bir yöntemi de Yahalom protokolündeki gibi protokolün iki yönünde farklı rasgele sayı kullanmaktadır. Aşağıda bu protokolün adımları yer almaktadır.

Mesaj 1: $A \rightarrow B: A \mid N_A$

Mesaj 2: $B \rightarrow S: B \mid S_Enc_{SA(Bekir)}(A \mid N_A \mid N_B)$ (2.12.)

Mesaj 3: $S \rightarrow A: S_Enc_{SA(Ali)}(B \mid K_{AB} \mid N_A \mid N_B) \mid S_Enc_{SA(Bekir)}(A \mid K_{AB})$

Mesaj 4: $A \rightarrow B: S_{\text{Enc}_{SA}(\text{Bekir})}(A \parallel K_{AB}) \parallel S_{\text{Enc}_{KAB}}(N_B)$

Burada istek sunucuya ulaşmadan Ali ve Bekir'in N_A , N_B rasgele sayılarını oluşturmaları gerekmektedir. Dolayısı sunucunun Bekir'in isteği ile gönderdiği anahtarın tazeliği konusunda ellerinde delil bulunmaktadır.

2.3.7. Cebirsel Saldırıları

Cebirsel özdeşlikler kriptolojik fonksiyonlarda sıkça gözüken bir özelliktir. Bazen bu özellik protokolün doğru bir şekilde çalışması için şarttır. Aslında sonlu alanlardaki üst alma gibi matematiksel işlemlerin değişme özelliği Diffie-Hellman anahtar oluşturma algoritmasının çalışabilmesi için gereklidir. Fakat bazı durumlarda saldırgan bu tip özdeşlikleri kullanarak protokolün güvenliğini tehlikeye atmaktadır.

Bu tip saldırı yöntemlerini önceden algılayabilmek için protokoldeki bu tip özdeşliklerin modellenerek ortaya çıkarılması gerekmektedir. Aslında bu demek oluyor ki, saldırganın protokoldeki özdeşlikleri bildiği ve cebirsel bazı işlemler yapabileceği farz edilmektedir. Protokoldeki bütün özdeşliklerin bulunup modellendiğinden her zaman emin olunamaz. Bu konuda da kolay bir metot bulunmamaktadır.

2.4. Konu İle İlgili Yayınlar ve Çalışmalar

Bu bölümde popüler güvenlik protokollerinden bazılarını ve bu protokollerin saldırıya açık güvenlik servisleri incelenecektir.

2.4.1. Çalışma Kiplerinin Önemi

DES in standart çalışma kipleri, "Elektronik Kod Kitabı" (ECB), "Şifre Blok Zincirleme" (CBC), "Şifre Geri Besleme" (CFB), "Çıktı Geri Besleme" (OFB) çalışma kiplerinden oluşmaktadır. DES ve bu çalışma kipleri 20 seneden beri kullanılmaktadır ve bazı zayıflıkları tespit edilmiştir. Dolayısı ile DES'in yerine geçen AES algoritmasının çalışma kiplerinde, DES çalışma

kiplerindeki zayıflıkların giderilmesi gerekmektedir. Bu maksatla NIST, AES'in çalışma kiplerinin belirlenmesi için bir yarışma düzenlemiştir. Öneriler arasında en dikkat çekenler arasında [Rogaway, et al, 2001]'de anlatılan "Baskı Kod Kitabı" (OCB) ve [Lipmaa, et al, 2001]'de anlatılan "Sayaç" (CTR) gibi çalışma kipleri mevcuttur. AES'e önerilen bu yeni çalışma kiplerinde şifrelemeyi ve bütünlüğü beraberce sağlamak gibi çok güzel fikirler kullanılmaktadır. Bu çalışma kipleri bölüm 3.'de anlatılmıştır. Bu çalışmada, yeni AES çalışma kiplerini detaylı olarak incelenecektir ve DES çalışma kipleri kullanan popüler protokollerdeki bazı problemlerin AES' e önerilen çalışma kipleri kullanımı ile çözülebileceği ispatlanacaktır.

[Bellare, 1996]'de anlatıldığı gibi DES çalışma kipinde kullanılan blok şifreleme algoritmalarını dikkatlice ele almak gerekmektedir. En çok kullanılan DES çalışma kipi CBC [Dworkin, 2001] çalışma kipidir. Bu kipte şifreli metin bloğu aşağıda gösterildiği gibi, açık metin bloğu ve bir önceki şifreli metin bloğunun modül 2 de toplanmasının blok şifreleme algoritması ile şifrelenmesi ile elde edilir.

$$C_i = B_Enc_{K, IV} (P_i \oplus C_{i-1}), \quad (2.13.)$$

Karşıtı olan şifre çözme işlemi ise aşağıdaki gibidir.

$$P_i = C_{i-1} \oplus B_Dec_{K, IV} (C_i), \quad (2.14.)$$

İlk açık metin bloğunun şifrelenmesi işleminde C_0 yerine IV kullanılmaktadır. Mesajı alan tarafın şifre çözme işlemini düzgün yapabilmesi amacı ile şifreleyici ve alıcı taraflar IV üzerinde daha sonra anlaşılacakları gibi, IV şifreleyici tarafından şifreli veya açık olarak da karşı tarafa gönderilebilir. Burada unutulmaması gereken şifrelenecek ilk açık metin bloğunun genellikle şifreli mesajın doğru şekilde açılabilmesi için bir çeşit mesaj uzunluğu bilgisi ve algoritmanın blok uzunluğunun katı olmayan mesajlar için ekleme miktarı gibi bilgileri de içeren mesaj başlığı olmasıdır.

En popüler DES çalışma kipi olan CBC'nin mesaj bütünlüğünün kullanılmadığı durumlarda aşağıda anlatacağımız bazı ilginç özellikleri vardır.

1. Şifrelenecek açık metin bloklarının $P_1, \dots, P_n$ ve CBC çalışma kipinde şifreleme işlemi sonucundaki şifreli mesaj bloklarının da $C_1, \dots, C_n$ oldukları farz edilecek. Bu durumda $C_1, \dots, C_k, , k < n$ şifreli mesaj bloklarının şifre çözme işlemi sonucunda $P_1, \dots, P_k$ açık mesaj blokları elde edilecektir. Bu demek oluyor ki, saldırgan kişi şifreli mesaj bloklarını sonundan kısaltabilir.
2. Bu özellik birinci özelliğin genelleştirilmiş hali olarak da anlaşılabilir. Yukarıda tanımlanan şifreli mesaj bloklarının herhangi ardışık bir dizisi olan $C_k, \dots, C_l, 1 < k < l < n$ şifreli mesaj bloklarının şifre çözme işlemi sonucunda $P_k, \dots, P_l$ açık mesaj blokları elde edilecektir. Bu demek oluyor ki, saldırgan kişi şifreli mesaj bloklarının önünden ve arkasından kısaltarak herhangi bir ardışık alt dizisinin kullanabilir.
3. CBC çalışma kipinin şifreli mesajdaki hataların, açık mesajdaki hatalara etkisinin sınırlı olmasıdır. Eğer bir şifreli mesaj bloğu iletim sırasında bozulmuş ise, açık mesaj bloklarından karşılık gelen blok ve bir sonraki blok zarar görürler. Fakat bunların dışındaki açık mesaj blokları aşağıda anlatılacağı gibi zarar görmezler. Eğer bir şifreli mesaj bloğu silinir ise, sadece karşılık gelen açık mesaj bloğunun bir sonraki zarar görür.

$$\begin{aligned} P_i &= C_{i-1} \oplus B_Dec_K(C_i) \\ P_{i+1} &= C_i \oplus B_Dec_K(C_{i+1}) \\ P_{i+2} &= C_{i+1} \oplus B_Dec_K(C_{i+2}) \end{aligned} \quad (2.15.)$$

Farz edilecek ki şifreli mesaj bloğu C_i hata içeriyor. Bu durumda açık mesaj bloğu tahmin edilemez biçimde tamamen hatalı olacaktır, çünkü bu açık mesaj bloğu, (2.15.) numaralı denklemden de anlaşıldığı gibi C_i şifreli mesaj bloğunun şifre çözme işlemi sonucunda bulunmaktadır. P_{i+1} açık mesaj bloğu tahmin edilebilir şekilde hata içermektedir, çünkü yine (2.15.) numaralı denklemden de anlaşıldığı gibi C_i şifreli mesaj bloğu ile modül 2

toplama sonucunda bulunmaktadır. Fakat diğer açık mesaj blokları hatasız olarak deşifre edilebilecektir, çünkü bunlar hatalı olan C_i şifreli mesaj bloğundan bağımsızlardır.



Eğer mesaj bütünlüğü korunmuyor ise yukarıdaki özellikler beraber düşünüldüğü zaman, saldırgan kişi şifreli, mesaj üzerinde kes-yapıştır işlemi gerçekleştirebilmektedir. Aynı anahtar ile şifrelenmiş farklı mesajlardan alınan şifreli mesaj blokları bir araya getirilebilir. Sadece ekleme noktasından sonraki açık mesaj bloğu zarar görecektir. Eğer IV'de kopyalanabiliyor ise bu açık mesaj bloğu da kurtarılabilmektedir.

Eğer kriptografik mesaj bütünlüğü algoritmaları kullanımı ile yukarıdaki mesajın bütünlüğü korunsaydı, saldırgan yukarıda anlatılan işlemlerden herhangi bir tanesini uygulayamayacaktı.

AES öneri çalışma kipi olan CTR çalışma kipi CBC çalışma kipinden oldukça farklıdır. CTR çalışma kipi daha çok blok algoritmaları blokları uzun olan dizi şifreleme algoritmaları gibi çalıştırır. Bu çalışma kipinin kullanıcıları eğer mesaj bütünlüğünü korumuyorlar ise anahtar dizisi tekrar kullanımı atağı gibi dizi şifreleme sistemlerinin sahip oldukları zayıflıklarının farkında olmalılardır. Aslında CTR çalışma kipinin mazisi 1979'lara kadar uzanmaktadır.

CTR çalışma kipinin avantajları aşağıda sıralanmıştır:

1. Bu çalışma kipi donanım olarak verimli bir şekilde gerçekleştirilebilmektedir. Çünkü, CBC gibi geri beslemeli olan çalışma kiplerindeki şifreleme hızı, altta yatan blok şifreleme algoritmasının hızı ile sınırlıdır. Bunun nedeni ise, C_{i+1} şifreli metin bloğunun şifrelenmesine geçilmeden önce C_i şifreli metin bloğunun şifrelenme işlemini tamamen bitmiş olması gerektiğinden kaynaklanmaktadır. Buna karşın CTR çalışma kipinde şifreli metin blokları birbirlerine işlemsel olarak bağlı olmadıklarından dolayı, donanım olarak C_1 , C_2 , ..., C_i şifreli metin blokları aynı anda hesaplanabilirler. [Gaj and

Chodowiec, 2000] makalesinde bunu ispatlamış ve AES finalistlerinin dördü için 30 ila 100 kat hız artışı elde edilmiştir.

2. Ayrıca yazılım olarak da verimli bir şekilde gerçekleştirilebilmektedir. Çünkü, modern işlemcilerin etkin ardışık düzenleri, birden çok yönergeleri ve ayrıca yazım alanları yeterince uzun ve çok. C_i ile C_j arasındaki işlem bağımlılığını ortadan kaldırmakla CTR çalışma kipi şifrelemesi yukarıda anlatılan işlemcilerin özelliklerinden azami bir şekilde istifade etmektedir. [Lipmaa, 1998]'da bu çalışma kipinin Pentium III, Itanium, Alpha ve Motorola Altivec işlemcilerde iyi bir şekilde gerçekleştirilmiş CBC'den dört kata kadar daha hızlı çalıştığı gösterilmiştir.
3. M açık mesajını şifrelemek için yapılan kriptolojik işlerin açık mesaj M'den bağımsız olmasından dolayı, ön hesaplama yapılabilmektedir. Buda bazı şartlarda hız artışı anlamına gelmektedir. Bu özellik aslında CTR çalışma kipinin blok şifreleme algoritmasını dizi şifreleme algoritması gibi kullanmasından kaynaklanmaktadır. Daha açık bir şekilde anlatmak gerekirse, CTR çalışma kipi kullanılarak çok uzun miktarlarda rasgele benzeri dizi elde edilir ve bu dizi gerektiği zaman açık mesaj M ile modül 2 de toplanarak şifrelenmiş olur.
4. CTR çalışma kipinde i-inci şifreli mesaj bloğu C_i rasgele erişim mantığı ile şifrelenebilir veya şifresi çözülebilir. Bu özellik disk şifrelenmesi gibi işlerde çok önem kazanmaktadır. CBC çalışma kipinde ise C_i şifreli mesaj bloğunun elde edilebilmesi için ilk önce C_{i-1} şifreli mesaj bloğunun elde edilebilmesi gerekmektedir. Bundan dolayı disk şifrelenmesi gibi uygulamalarda CBC çalışma kipi verimli bir şekilde kullanılamamaktadır.
5. Yukarıda anlatılan özellikler CTR çalışma kipinde güvenlikten ödün verilerek sağlanmamıştır. CTR çalışma kipi için somut güvenlik sınırları CBC çalışma kipinde olduğu kadar iyi belirlenmiştir.

6. Şifreleme ve şifre çözme işlemlerinin her ikisi de altta yatan blok şifreleme algoritmasının şifreleme işlemine ihtiyaç duyar. Dolayısı ile blok şifreleme algoritmasının şifre çözme işlemini gerçeklemeye gerek yoktur. Bu durum özellikle altta yatan blok şifreleme algoritmasının şifreleme ve şifre çözme işlemlerinin birbirinden çok farklılık gösterdiği durumda yarar sağlamaktadır.
7. CTR çalışma kipi kullanılarak istenilen uzunluktaki mesaj şifrelenebilir. CBC çalışma kipinin aksine şifrelenecek mesajın blok şifreleme algoritmasının katı olmadığı durumlarda bile ekleme yapılmasına gerek duyulmaz, dolayısı ile ikil israf edilmemiş olur.

CTR çalışma kipinin dezavantajları aşağıda sıralanmıştır:

1. CTR çalışma kipi sadece mesaj gizliliğini sağlamaktadır, mesaj bütünlüğünü sağlamamaktadır. Fakat, normalde mesaj bütünlüğünün sağlanmasının çalışma kipinin sorumluluğunda olmadığı kabul edilmektedir. CBC çalışma kipi de anlamlı bir şekilde mesaj bütünlüğü korunmasında başarısız olmaktadır. Eğer CRT çalışma kipinin yanında mesaj bütünlüğünün sağlanması isteniliyor ise, genel yöntem blok şifreleme algoritmasının mesaj doğrulama kodu şeklinde kullanılması yönünde olmaktadır.
2. Eğer şifreli mesaj bloklarından herhangi birinde bir ikilik bir bozulma olmuş ise, şifre çözme işleminden sonra sadece bu bloğa karşılık gelen açık mesaj bloğunun aynı pozisyonundaki ikili bozulur. Aslında bu ne iyi nede kötü demek oluyor. Fakat eğer mesaj bütünlüğü korunmuyor ve gönderilen mesajın bazı yerleri tahmin edilebiliyor ise, saldırgan şifreli metin bloklarındaki ikilleri ters çevirerek açık metni istediği gibi değiştirebilmektedir. Bu kötü özellikte dizi şifreleme sistemlerinin zayıflıklarından CTR çalışma kipine geçmektedir.

3. Gönderici tarafın sayacın geldiği son durum bilgisini saklaması gerekmektedir. Fakat sayaç yerine rasgele bir değer kullanılabilir. Bu durumda son durumun saklanmasına gerek yoktur. Aslında CTR çalışma kipinde kullanılan sayaç bilgisinin, CBC çalışma kipinde kullanılan IV'den fazla bir farkı yoktur.
4. CTR çalışma kipinin yukarıdaki özellik ile çok yakından ilgili bir başka dezavantajı da kullanıcı hatalarına karşı çok hassas olmalarıdır. Örneğin, CTR çalışma kipinde sayacın bir başka mesajın şifrelenmesi amacı ile tekrar kullanılması, şifrelenmiş her iki açık mesajında saldırgan tarafından ele geçebilmesi gibi çok hayati sonuçlara neden olabilmektedir. Çünkü, söz konusu iki şifreli mesajın birbirleri ile modül 2 toplamlarından, bunlara karşılık gelen iki açık mesajın modül 2 toplamları çıkmaktadır. Bu durumda her iki mesajında bulunabileceğini gösteren değişik makaleler mevcuttur. Farklı mesajlar için sayacın tekrar kullanılmasının yanı sıra, bu mesajların herhangi bloklarında bile kullanılan sayaçların aynı olmaması gerekmektedir. Aksi takdirde sonuç yukarıdaki ile aynı olacaktır.
5. Doğru kullanıldığı takdirde CTR çalışma kipinin güvenliği aslında altta yatan blok şifreleme algoritmasının güvenliğine bağlıdır. Birbirini izleyen blokların sayaç, sayaç+1 Hamming farkları genellikle çok azdır. Bu saldırganın Hamming farklarının az olduğunu bildiği açık mesaj bloklarını toplayabilir. Buda fark kriptanalizi ile saldırganın anahtar hakkında bir şeyler öğrenebileceği anlamına gelmektedir. Fakat saldırgan bu işi ancak altta yatan blok şifreleme algoritması fark kriptanalizi yönünden zayıf ise gerçekleştirebilir. Bu aslında çalışma kipinin sorumluluğunda olan bir şey değildir. CBC çalışma kipinde IV yerine sayaç kullanmanın kötü bir fikir olduğu zaten bilinmektedir. Bu yüzden de CTR çalışma kipini bu yönden şüphe ile bakılmaktadır.

Başka bir AES öneri çalışma kipi olan OCB ise yukarıda bahsedilen çalışma kiplerinden oldukça farklıdır. Bu çalışma kipinde çok güzel ve farklı tasarım kriterleri kullanılmıştır. Örneğin bu çalışma kipi, mesaj gizliliğini ve mesaj bütünlüğünü aynı anda sağlamaktadır. Bu durumda mesaj bütünlüğünü otomatik olarak sağladığı için yukarıda anlatılan kes-yapıştır saldırı yöntemi bu çalışma kipi kullanılan şifreli mesajlara uygulanamayacaktır.

OCB çalışma kipinin avantajları aşağıda sıralanmıştır:

1. OCB çalışma kipi mesaj bütünlüğünü ve mesaj gizliliğini beraberce sağlamaktadır. CTR ve CBC gibi sadece mesaj gizliliğini sağlayan çalışma kiplerinin yanında mesaj bütünlüğünün korunması da talep ediliyor ise bu durumda genellikle altta yatan blok şifreleme algoritmasının CBC-MAC olarak kullanılması yöntemi seçim yapılmaktadır.
2. Gizliliğin sağlandığı çalışma kiplerinden sadece iki fazla şifreleme algoritması kullanmaktadır. Buda gizlilik ve bütünlüğü beraber düşündüğümüzde, diğerlerine göre üstünlük sağlamaktadır. Çünkü, sadece gizliliği sağlayan çalışma kiplerinde bütünlük hesaplanması için ayrıca işlemlerin yapılması gerekmektedir.
3. OCB çalışma kipinde kullanılarak istenilen uzunluktaki mesaj şifrelenebilir. CBC çalışma kipinin aksine şifrelenecek mesajın blok şifreleme algoritmasının katı olmadığı durumlarda bile ekleme yapılmasına gerek duyulmaz, dolayısı ile ikil israf edilmemiş olur.
4. Diğer çalışma kiplerinde olduğu gibi OCB çalışma kipi de tekrar etmeyen ilk vektör kullanımı gerektirmesine karşın CBC çalışma kipinin aksine bu değer OCB çalışma kipinde sayaç olarak kullanılabilir.

5. Bu çalışma kipinde kullanılan baskı değeri sadece anahtara bağlıdır. Bu nedenden dolayı bir anahtar için bir kere hesaplanmaması yeterlidir.
6. OCB çalışma kipi mesaj gizliliği ve bütünlüğünü beraber sağlamasına karşın sadece bir anahtar kullanmaktadır.
7. OCB çalışma kipinde i-inci şifreli mesaj bloğu C_i rasgele erişim mantığı ile şifrelenebilir veya şifresi çözülebilir. Bu özellik disk şifrelenmesi gibi işlerde çok önem kazanmaktadır. CBC çalışma kipinde ise C_i şifreli mesaj bloğunun elde edilebilmesi için ilk önce C_{i-1} şifreli mesaj bloğunun elde edilebilmesi gerekmektedir. Bundan dolayı disk şifrelenmesi gibi uygulamalarda CBC çalışma kipi verimli bir şekilde kullanılamamaktadır. Dolayısı ile OCB çalışma kipi donanım gerçeklemelerinde CBC çalışma kipinin aksine paralel çalıştırılabileceğinden dolayı CBC çalışma kipine üstünlük sağlamaktadır.
8. Bu çalışma kipinin üç değişik tipi mevcuttur.
9. Yukarıda anlatılan özellikler OCB çalışma kipinde, CTR çalışma kipinde olduğu gibi güvenlikten ödün verilmeden sağlanmıştır. OCB çalışma kipi için somut güvenlik sınırları CBC ve CTR çalışma kiplerinde olduğu kadar iyi belirlenmiştir.

2.4.2. Dizi Şifreleme Algoritmalarının Zayıflıkları

Sadece Vernam şifreleme sistemi olarak bilinen şifreleme sisteminin kırılmaz olduğu söylenmektedir. Vernam şifreleme sisteminde tamamen rasgele olarak üretilen ikil dizisi açık mesaj ikilleri ile modül 2 de toplanırlar. Fakat bu sistemde kullanılan tamamen rasgele olan ikil dizilerinin mesajın uzunluğunda olması gerekmektedir. Dolayısı ile burada kullanılan ikil dizilerinin üretilmesi bir yana özellikle dağıtılması büyük problem olmaktadır.

Aslında Vernam şifreleme sisteminde eğer gerçekten rasgele ikil dizisi kullanıldı ise ve daha önemlisi bu dizi sadece bir kere kullanıldı ise kanıtlanmış güvenlik seviyesine sahip olduğunu söylemek daha yerinde olur. Daha açık bir ifade ile anlatmak gerekirse, rasgele ikil dizisi ile modül 2 toplanan ve rasgele olmayan açık mesaj ikil dizisinin sonucunda gerçek rasgele ikil dizisi ortaya çıkar. Eğer bahsedilen ikil dizisi birden fazla kullanılmış ise kırılmaz olarak bahsedilen bu şifreleme sistemi [David, 1996] kitapta bahsedildiği gibi kırılabilir.

Günümüzde böyle sistemlerin yerine anahtar olarak adlandırılan ve yukarıda bahsedilen ikil dizisine göre çok daha kısa rasgele değerler kullanılarak istenilen uzunlukta Golomb'un gerçek olduğu varsayılan rasgelelik kriterlerine uyan ikil dizileri üretilebilmektedir. Bir kural çerçevesinde üretilen ve Golomb'un gerçek olduğu varsayılan rasgelelik kriterlerine uyan böyle dizilere "Sanki rasgele" ikil dizisi adı verilmektedir. Çünkü aynı anahtar verildiği taktirde sonuçta aynı dizi olmaktadır. RC4 dizi şifreleme algoritması böyle bir algoritmadır. Bu algoritma ile istenilen uzunluktaki açık metin şifrelenebilir. Aslında 128 ikillik bir anahtar ile pratik olarak sonsuz uzunluktaki açık metin şifrelenebilir. Buradaki pratik kelimesi günümüz teknolojisini kullanarak böyle bir işlemin çok uzun zaman alacağı anlamına gelmektedir.

802.11 kablosuz ağlarında kullanılan "Kabloluya Denk Gizlilik " (WEP) protokolünde de RC4 dizi şifreleme algoritması kullanılmaktadır. Dizi şifreleme sistemlerinin bilinen en zayıf hatası, eğer aynı anahtar dizisi birden fazla açık mesajı şifrelemek için kullanılıyor ise bu aslında her iki mesajın da hattı dinleyen bir saldırgan tarafından ele geçirilebileceği anlamına gelmektedir.

$$C_1 = P_1 \oplus \text{AnahtarDizisi},$$

$$C_2 = P_2 \oplus \text{AnahtarDizisi}, \quad (2.16.)$$

$$\begin{aligned} C_1 \oplus C_2 &= (P_1 \oplus \text{AnahtarDizisi}) \oplus (P_2 \oplus \text{AnahtarDizisi}) \\ &= P_1 \oplus P_2 \oplus \text{AnahtarDizisi} \oplus \text{AnahtarDizisi}. \end{aligned}$$

$$= P_1 \oplus P_2$$

Başka bir deyişle aynı anahtar dizisi ile şifrelenmiş iki şifreli mesajın modül 2 toplamı, anahtar dizilerinin birbirlerini iptal etmelerinden sonra bize açık mesajların modül 2 toplamını vermektedir. Buda demek oluyor ki anahtar dizisi tekrar kullanımı saldırganı birtakım atak yöntemleri vermektedir. Örneğin, eğer bir açık metin biliniyor ise diğeri hemen bulunabilir. Aslında bakılırsa hangi dilde olursa olsun açık mesajlar genellikle artıklık içermektedir buda sadece $P_1 \oplus P_2$ 'dan P_1 ve P_2 'nin her ikisinin de hesaplanabileceği anlamına gelmektedir. Örneğin iki İngilizce açık mesajın modül 2 toplamından her iki mesajında elde edilme yöntemleri [Dawson and Nielsen, 1996]'de anlatılmıştır.

Bundan başka eğer elimizde aynı anahtar dizisi kullanılarak şifrelenmiş n tane şifreli mesaj var ise [Tutte, 1998]'de anlatıldığı gibi n büyüdükçe açık mesajların bulunması daha da kolay hale gelmektedir.

Bu saldırıları, yani anahtar dizisi tekrar kullanımını önlemek için WEP protokolünde paket başına farklı IV kullanılmaktadır. Böylelikle her pakette kullanılan anahtar dizisi farklı olacaktır. WEP protokolü anahtar dizisini IV ve anahtarın bir fonksiyonu olarak üretmektedir. Burada IV her bir paket için değişmektedir, anahtar ise bütün paketler için aynıdır. Böylelikle her bir paket farklı bir anahtar dizisi ile şifrelenmektedir. Burada kullanılan IV, alıcı tarafında aynı anahtar dizisinin oluşturulabilmesi amacı ile paketin şifrelenmemiş kısmında yer almaktadır. Buda demek oluyor ki aslında IV saldırgan kişi tarafından da bilinmektedir, ama şifrelemede kullanılan anahtar saldırgan tarafından bilinmediğinden dolayı gizlilik korunmuş olmaktadır. Burada kullanılan paket başına farklı IV kullanımı anahtar tekrar kullanımlarını önlemek amacı iledir. Fakat WEP protokolü bu amacına ulaşamamaktadır. [Borisov et al, 2001]'de anlatıldığına göre WEP protokolüne karşı uygulanan bazı gerçekçi saldırılar mevcuttur. Bu makalede anahtar tekrar kullanımlarının nasıl saptanacağı ve tipik açık mesajların beklenen dağıtılma yöntemleri hakkında bazı bilgiler kullanılarak bu durumları saptama yöntemlerinden de bahsedilmektedir.

WEP standardı, her paket için farklı bir IV kullanılmasını tavsiye etmektedir, zorunlu kılmamaktadır. Fakat bundan başka IV'lerin nasıl seçileceği konusunda bir şey belirtmemektedir. Gerçekte ise bazı WEP gerçeklemleri bu işi kötü biçimde yapmaktadır. Örneğin, Bu makalede inceledikleri bazı PCMCIA kartları tekrar ilklendirildiklerinde IV'yi sıfırlamakta ve her bir paketi şifreledikten sonra IV'yi bir arttırmaktadır. Çok sık olması beklenen bir durum olan bu kartların diz üstü bilgisayara her takılışlarında kendilerini ilklendirmekteydiler. Dolayısı ile düşük numaralı IV'lerden oluşan anahtar dizilerinin, herhangi bir anahtar yaşam süresi içerisinde birçok defa kullanılmaları kaçınılmaz olmaktadır.

Aslında WEP standardı IV konusunda bütün WEP gerçeklemlerini korumasız bırakan çok ciddi bir tasarım hatası içermektedir. Örneğin, nerede ise aynı anahtar dizisi kullanımını birkaç mesaj için garantileyecek şekilde bu standartta IV 24 ikil olarak alınmıştır. Daha önce bahsettiğimiz gibi, saldırgan aynı IV kullanan iki paket bulduğu zaman, bu paketlerdeki her iki açık mesajı da çözebilir.

Buna benzer bir durum da 1998 yılında Microsoft firmasının bir ürünü olan "Noktadan Noktaya Tünel Protokolü" (PPTP) protokolünde ortaya çıktı. Yazılım bir protokol olan PPTP protokolü, kullanıcıların noktadan noktaya bağlantılarını tünel açmalarına olanak sağlayarak bir nevi sanal özel ağ görevi görmektedir.

PPTP protokolü istemci/sunucu tipinde çalışan bir protokoldür. İstemci ile sunumcu arasındaki mesajlar RC4 dizi şifreleme algoritması kullanılarak şifrelenebilmektedir. Microsoft bu protokolün dizayn kriteri olarak, istemci ile sunucu arasındaki şifrelemesinin sunucu ile istemci arasındaki şifrelemeden bağımsız olmasını tasarlamış ve şifreleme algoritması olarak yazılım olarak hızlı çalışan RC4 dizi şifreleme algoritması seçilmiş. Bu protokoldeki problem [Schneier and Mudge, 1998] ve [Schneier and Mudge, 1999]'de de anlatıldığı gibi istemci-sunucu ile sunucu-istemci arasındaki RC4 dizi şifrelemesinde aynı anahtarın kullanılmış olmasıdır.

Daha dikkatli bir şekilde incelenirse, her bir tarafın birini şifreleme işleminde diğerini ise şifre çözme işleminde kullanacağı iki anahtar dizisi olduğu sonucuna varılır. Fakat bu iki anahtar dizisi de aynı olmaktadır. Örneğin, istemcinin sunucuya göndereceği ilk mesaj bu anahtar dizisinin ilk 128 ikilini kullansın. Sunucu ise istemci için hazırlayacağı mesajda yine aynı anahtar dizisinin ilk 128 ikilini kullanacaktır. Bu durumda bu iki mesajın modül iki toplamlarından açık mesajların modül 2 toplamı kalmaktadır.

2.4.3. Mesajın Gizliliği

Mesaj gizliliği servisine yapılan bazı saldırılar yukarıda anlatıldığı gibi anahtar dizisi tekrar kullanımı gibi dizi şifreleme sistemlerinin zayıflıklarından kaynaklanmaktadır. Güvenlik protokolü analizi ve özellikle tasarımı yapan kişilerin dizi şifreleme sistemlerinin bu zayıflıkları konusunda çok dikkatli olmaları gerekmektedir.

ECB DES çalışma kipi birden fazla açık mesaj bloğunun şifrelenmesinde kullanılmamalıdır. Bir mesajın ECB çalışma kipinde şifrelenmesi sırasında aynı açık mesaj bloklarının şifrelenmiş halleri de aynı olacaktır. Bu ise kriptografik açıdan tehlikelidir. En çok kullanılan DES çalışma kipi olan CBC çalışma kipinde ise aynı anahtar ve aynı IV kullanıldığı taktirde bile şifreli mesaj blokları farklı olmaktadır. Dolayısı CBC gibi geri beslemeli çalışan çalışma kipleri vasıtası ile blok şifreleme sistemlerinde, dizi şifreleme sistemlerindeki anahtar tekrar kullanımı gibi bazı zayıflıklar görülmemektedir.

Fakat blok şifreleme sistemi kullanan bazı protokollerde de bazı gizlilik servis saldırıları yapılabilmektedir. [Bellovin, 1996]'de blok şifreleme algoritması kullanılarak tasarlanmış olan IPSec protokolünün bir önceki sürümü incelenmiştir ve gizlilik servisine karşı yapılabilecek bazı saldırı yöntemleri sıralanmıştır. Bu saldırı yöntemlerine kısaca değinilecektir.

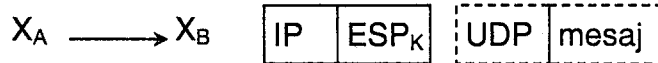
Burada saldırgan kişinin A ve B gibi her iki ana bilgisayarda da yasal bir kullanıcısının olduğunu, mesajlaşmada mesaj bütünlüğünün korunmadığını,

saldırgan kişinin hattı dinleyebildiği, mesaj enjekte edebildiği ve ana bilgisayar başına anahtarlamının kullanıldığını varsayacağız.



A ana bilgisayarındaki L_A yasal kullanıcılarından B ana bilgisayarındaki L_B yasal kullanıcılarına hilesiz bir mesaj gönderildiğini varsayalım. Saldırgan bu mesajı kaydeder ve A ana bilgisayarındaki X_A kendi yasal kullanıcılarından B makinesinde ki X_B kendi yasal kullanıcılarına bir UDP mesajı gönderir ve bu mesajı da yakalayıp kaydeder. Daha sonra Şekil 2.9.'de de görüldüğü gibi ilk kaydettiği mesajın şifreli kısmını ikinci kaydettiği mesajın gövdesine kopyalayarak ve uzunlukların tutması için gerekli eklemeleri yaptıktan sonra bu yeni mesaj saldırıdan tarafından B ana bilgisayarındaki X_B kullanıcılarına gitmek üzere hatta enjekte edilir. Önceki bölümde anlatılan CBC çalışma kipiinin hataların fazla büyümemesi özelliğinden dolayı saldırıdan tarafından kaydedilen ilk mesajın TCP başlığı hasar görür (eğer saldırıdan tarafından IV'de kopyalanabilirse TCP başlığı da zarar görmez), fakat gövde kısmı açık mesaj haline gelmiş durumdadır. Eğer IPv4 kullanılıyor ise saldırıdan tarafından UDP sağlamasının doğrulamasının yapılmaması istenilebilir. Eğer bunun mümkün olmadığı IPv6 kullanılıyor ise saldırıdan tarafından bu sağlamayı aldatabilmesi için ortalama 2^{16} deneme yapması yeterli olacaktır.

İzlenen Veriler



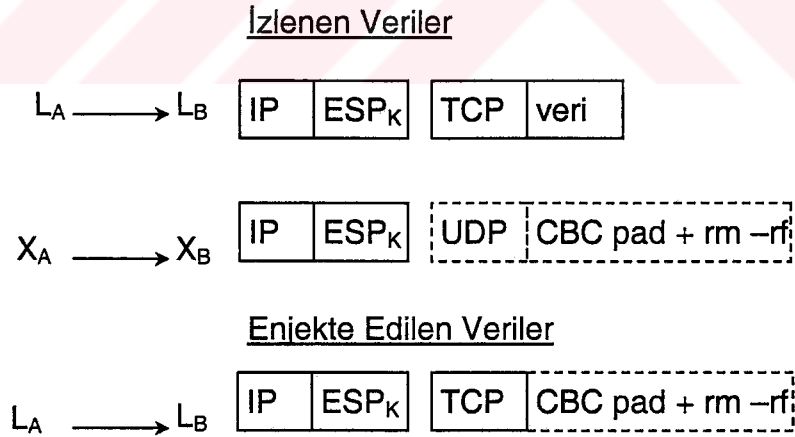
Enjekte Edilen Veriler



Şekil 2.9.: Burada Başka Bir Yasal Kullanıcının Trafiğini Çözmek Amacı İle Kes-Yapıştır Atağı Yapılması Gösterilmektedir. Burada Çizgili Bölge Şifreli Verileri, Gölge Bölge İse Verinin Yasal Kullanıcıya Ait Olduğunu Göstermektedir.

Saldırgan aynı tip teknikleri, Şekil 2.10.'de gösterildiği gibi, yasal bir başka kullanıcının şifreli oturumuna virüs gibi kötü niyetli veriler yollamak amacı ile de kullanabilir. Bu saldırı biçiminde de saldırgan yasal kullanıcılar L_A ve L_B 'den ve kendi kullanıcıları X_A ve X_B tarafından gönderilen verileri hatta gözleyerek kaydeder. İkinci paket B anabilgisayarındaki L_B kullanıcısına gönderilmek üzere hazırlanmış kötü niyetli veriler içermektedir. Eğer L_B kullanıcısına giden yasal paket silinebilir ise yerine saldırgan tarafından hazırlanan paket gönderilebilir. Eğer silinmiyor ise, yeni bir paket oluşturularak L_B kullanıcısına gitmek üzere hatta enjekte edilebilir.

Her şeyden önce TCP protokolü uzunluk bilgisi içermemektedir. Bu şekilde hatta izlenen paketlere eklemeler yapmak mümkün hale gelmektedir. Çok önemli olmamakla beraber ekleme yapılan bölgede CBC çalışma kipi şifrelemesinden dolayı veri kaybını önlemek için ekleme yapılması gerekebilmektedir. Eklenen bu önemsiz bilgiler şifre çözme işleminden sonra çöp veri olacaktır. Saldırganın kabuk bilgi sistemi gibi bilinen bir oturumu geri yüklemek için birkaç sekiz ikillik veri göndermesi de gerekebilecektir.



Şekil 2.10.: Burada Yasal Başka Bir Kullanıcının Oturumunun Kaçırılması İle Bu Kullanıcıya Kötü Niyetli Bilgiler Gönderilmesi Gösterilmektedir. Burada Çizgili Bölge Şifreli Verileri, Gölge Bölge İse Verinin Yasal Kullanıcıya Ait Olduğunu Göstermektedir.

IV bilgisi genellikle açık olarak gönderildiğinden ve şifre çözme işlemi sonucunda ilk açık mesaj bloğu olan P_1 'in, ilk şifreli mesaj bloğu C_1 'in deşifre

işlemi sonucunda çıkan bilgi ile IV'nin modül 2 toplamının sonucunda hesaplanmasından dolayı saldırgan P_1 açık mesaj bloğuna istediği şekilde değişiklikler yapabilir.

$$P_1 = IV \oplus B_Dec(C_1) \quad (2.17.)$$

UDP protokolü için bütün başlık bilgisi bir bloktan oluşmaktadır. Dolayısı ile saldırgan bu paketi başka bir bağlantıya yönlendirebilir. Sağlama bilgisi yüzünden TCP başlıklarını değiştirmek biraz daha zor hale gelmektedir, fakat yine de mümkündür [Ziemba, et al, 1995]. IP başlığının da sağlama bilgisi vardır. Fakat bu başlıklarda paket belirleme alanı gibi bazı ayrılmış alanlar vardır. Bu alanlar üzerinde oynama yapılarak sağlama doğru hale getirilebilir, çünkü sadece 16 ikiden oluşmaktadır.

Yine [Bellovin]'de anlatılan kısa blok saldırısı IPsec protokolünün gizliliğine yönelik yapılan başka bir aktif saldırı yöntemidir. Bu saldırı yöntemi bir sekiz ikil gibi çok kısa bilgilerin şifrenmesi veya son blokta kısa bilgi bulunması durumunda yapılabilmektedir. Bu kısa blokların tam bir blok haline getirilmesi için genellikle rasgele ekleme yapılmaktadır. Bu atak yönteminde bu son şifreli mesaj bloğu yerine açık ve şifreli halini bildiği yeni bir blok ile değiştirir. Burada unutulmaması gereken eklemenin tamamen rasgele oluşu ve saldırgan tarafında tahmin edilemeyeceğidir. Doğru tahin TCP protokolünün sağlamasının doğruluğu ile anlaşılabilir. Hatalı blok alıcının TCP yığınının sessizce düşmesine neden olacaktır. Doğru paket ise alındı bilgisi gönderilmesine neden olacaktır. [Bellovin, 1996]'da yüksek ihtimal ile buna benzer başka saldırı yöntemlerinin olduğundan şüphelenilmektedir.

[Schneier and Wagner, 1997]'de ise SSL 3.0 protokolü incelenmiş ve gizliliği mükemmel bir şekilde koruduğundan bahsedilmektedir. Fakat ihraç-zayıflamasına uğramış sürümü düşük düzeyde gizlilik sağlamaktadır.

2.4.4. Mesajın Bütünlüğünün Korunması ve Kimlik Doğrulama



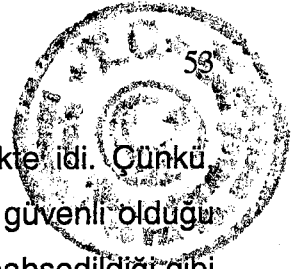
[Bellovin, 1996] ve [Borisov et al, 2001]'de zayıf veya yetersiz bütünlük kontrolü kullanılmasından kaynaklanan benzer problemlerden bahsedilmektedir. Ayrıca Microsoft'un parola dosyası ve SSL 2.0 protokolünde de durum farklı değil.

Microsoft NT işletim sisteminin tasarımını yaparken parolaların nasıl korunacağı konusunda özetleme fonksiyonu kullanmaya karar verir. O zamanlarda en popüler özetleme fonksiyonları MD2 ve MD5 dir. Bu iki özetleme algoritmasının tasarım kriterleri açıktır ve akademik çevreler tarafından iyi test edilmiştir. Fakat Microsoft özetleme algoritmasını kendi geliştirme karar verir ve LANMAN özetleme algoritması geliştirerek kullanır. Daha sonra bu özetleme algoritmasının saniyeler mertebesinde kırılabilen çok zayıf bir özetleme algoritması olduğu keşfedilir [Petteri, 1997].

Daha sonra Microsoft bu özetleme algoritmasını değiştirmeye karar verir ama bu kez de MD4 algoritmasını seçer. Halbuki araştırmacılar MD4 konusunda şüphelerinin olduğunu ve kullanılmaması gerektiğini belirtmişlerdir. Aslında MD4 özetleme algoritmasının geliştiricileri bile bu özetleme algoritmasının kullanılmaması gerektiğini belirtmişlerdir.

SSL 3.0 da uygulama verisinin bütünlüğünün korunması amacı ile kriptografik MAC kullanılmaktadır. SSL tasarımcıları HMAC [Krawczyk, et al, 1997] kullanmayı tercih etmişlerdir. HMAC, basit, hızlı, özetleme algoritması tabanlı ve güvenli olduğuna dair güçlü teorik altyapıya sahiptir. HMAC, mesaj bütünlüğü ve doğrulanması konusunda altın bir standart olma yolundadır ve SSL 3.0 protokolü için çok iyi bir seçimdir. Eğer kriptoloji sahasında beklenilmeyen bir gelişme olmaz ise, HMAC standardının yakın gelecekte de güvenliğini sürdüreceği tahmin edilmektedir.

Genelde bakıldığı zaman SSL 3.0 protokolünün bütünlüğüne yapılan saldırılara karşı oldukça güvenli olduğu görülmektedir. SSL 2.0 ise güvensiz



MAC kullanmaktan dolayı çok ciddi tasarım hataları içermekte idi. Çünkü mesajın özetini alıp, mesaj ve özeti daha sonra şifrelemenin güvenli olduğu düşünülmekte idi. Bu tip mesajlar aslında [Bellovin, 1996]'de bahsedildiği gibi seçilebilen açık mesajlar ile kes-yapıştır saldırılarına maruz kalmaktadır.

WEP protokolü [Borisov, et al, 2001] ve [Arbaugh, et al, 2001]'de anlatıldığı dibi, paketin iletim sırasında değişmediğini anlamak için bütünlük sağlama alanı kullanır. Bütünlük sağlama alanı CRC-32 işlemi ile hesaplanmaktadır. Ayrıca bu alan pakette şifrelenecek bilgiler arasında yer almaktadır. Bu makalede CRC-32 sağlama işleminin kullanılmasının çok yetersiz kaldığından ve saldırganın mesajı değiştirmesine engel olamadığından bahsedilmektedir. Kriptografik açıdan ve tasarım kriteri olarak CRC-32 mesajda iletimi sırasında oluşabilecek rasgele ve kasıtsız değişimleri saptamak için kullanılmaktadır ve mesaja yapılan kasıtlı değişiklikleri saptamakta yetersiz kalabilmektedir. Mesaj paketi bir dizi şifreleme algoritması ile şifrelenince bu özellik aşağıda anlatılacağı gibi mesaj paketini önemli saldırılara maruz bırakmaktadır.

WEP protokolünde kullanılan mesaj bütünlük sağlama fonksiyonu, mesajın doğrusal bir fonksiyonudur. Bunun nedeni de bütünlük sağlama işleminin aşağıdan da görülebileceği gibi tamamen mesaj üzerinde modül 2 toplama işlemleri ile hesaplanmasıdır.

$$\text{CRC}(x \oplus y) = \text{CRC}(x) \oplus \text{CRC}(y) \quad (2.18.)$$

Yukarıdaki işlem bütün x ve y'ler için geçerlidir. Bu özellik bütün CRC sağlamalarının ortak bir özelliğidir. Bu özelliğin önemli bir sonucu da bütünlük sağlamasına zarar verilmeden şifreli metin üzerinde kontrollü değişiklikler yapabilmesidir. Buda demek oluyor ki saldırgan ortaya çıkma korkusu olmadan şifreli mesaj üzerinde istediği değişikliği yapabilecektir. Bu da WEP protokolünün üç görevinden biri olan bütünlüğü sağlama işleminin başaramadığı anlamına gelmektedir.



WEP protokolü için keşfedilen diğer bir zayıflık ise, bu protokolün güvenli erişim sağlayamamasıdır. Bunun önemli bir nedeni de WEP protokolünde kullanılan bütünlük sağlama fonksiyonunun herhangi bir anahtara bağlı olmadan sadece açık mesajın bir fonksiyonu olmasından kaynaklanmaktadır. Bundan dolayı açık mesajı bilen veya tahmin edebilen bir saldırgan bütünlük sağlama bilgisini de elde edebilmektedir. WEP protokolünün bütünlük sağlama fonksiyonunun bu özelliği saldırgana erişim hakkı edinmekte yardımcı olmaktadır. Eğer saldırgan şifreli mesaja karşılık gelen bütün açık mesajı bir şekilde biliyor veya elde edebiliyor ise, herhangi istediği bir bilgiyi trafiğe enjekte de edebilir.

Bunun nedeni aynı anda hem açık mesajı hem de şifreli mesajı bilmek, aslında anahtar dizisini de bilmek anlamına gelmektedir. Saldırganın anahtar dizisini bilmesi ise, aynı IV kullanımı ile istediği herhangi bir açık mesajı şifreleyebilmesi anlamına gelmektedir.

$$P \oplus C = P \oplus (P \oplus \text{Anahtar Dizisi}) = \text{Anahtar Dizisi} \quad (2.19.)$$

[Borisov, et al, 2001] makalesinde ayrıca WEP protokolünün kimlik doğrulama protokolüne karşı da bir saldırı yapılabileceğinden bahsetmektedir. Bu atak aslında yukarıdaki mesaj oluşturma atağının bir benzeridir ve WEP protokolündeki ortak anahtar ile kimlik doğrulama işlemine saldırmaktadır.

Mobil istasyon sisteme dahil olmadan önce bir nevi kimlik doğrulama işlemi yapmak zorundalar. Bu işlem için mobil istasyonların ortak ile anahtar kimlik doğrulaması talebinde bulunması gerekmektedir. Kabul noktası buna karşı 128 sekiz ikil uzunluğunda rasgele ve şifresiz bir sayı ile meydan okumada bulunur. Mobil istasyon ise bu meydan okumaya karşılık olarak kendisine gelen 128 sekiz ikilin ortak anahtarın kullanımı ile şifrelenmiş halini göndermesi gerekmektedir. Eğer kabul noktası, şifre çözme işleminden sonra kendi gönderdiği rasgele sayıyı buluyor ise, kimlik doğrulama işlemi başarı ile sonuçlanmış demektir. Burada gönderilen rasgele sayının şifreli halini

hesaplayabilme kabiliyetinin mobil istasyonun anahtara sahip olduğu anlamına geldiği düşünülmüştür.

Fakat yukarıda gösterildiği gibi anahtar bilgisine sahip olunmadan da şifreli WEP mesajları gönderilebilmektedir. Bütün ihtiyaç duyulan yeterli uzunlukta bir açık / şifreli mesaj çifti bulabilmektir. Bunu elde etmekte sanıldığı gibi hiçte güç değildir. Yapılması gereken tek şey başarılı bir kimlik doğrulama işleminin gidip gelem mesajlarını kaydedebilmektir. Saldırgan böylelikle, hem kabul noktası tarafından gönderilen rasgele bilgiyi ve mobil istasyon tarafından gönderilen rasgele bilginin şifrelenmiş halini elde edebilmektedir. Bu bilgiler ışığında cevabı şifrelemek için gerekli olan anahtar dizisini hesaplamak çok kolaydır. Bütün meydan okumalar aynı uzunlukta olduğundan dolayı, saldırının kabul noktası tarafından gönderilen meydan okumayı bu anahtar dizisi ile şifrelemesi için yapması gereken sadece modül 2 toplamdır.

2.4.5. Mesajın Gönderildiğinin İnkâr Edilememesi

İnkâr edememe güvenlik protokollerindeki önemli servislerden biridir. Bu servisin amacı, bir katılımcıyı diğer bir katılımcının muhtemel aldatmasına karşı korumaktır. Aslında yasal kullanıcıların birbirlerini aldatmaya çalışmayacaklarını düşünmemiz çok gerçekçi olmaz. İnkâr edememe servisinin mesaj kimlik doğrulama servisi ile yakından alakası vardır. Dolayısı ile bu özellik kimlik doğrulama servisinin bir çeşidi olarak algılanıp saldırının kabiliyetleri çerçevesinde incelenebilir.

Sayısal imzalama yöntemlerindeki inkâr edememe tarafı açık anahtarlı şifreleme sistemlerinin başlıca avantajları arasındadır. Bu özellik sayesinde, imza atan kişinin daha sonra imzayı atmadığını iddia etmesini önlenilebilir.

İncelediğimiz protokollerden IPSec anahtar yönetimi hariç açık anahtarlı sistemler kullanmadıklarından dolayı bu servisi desteklememektedirler.

2.4.6. Anahtar Yönetimi

Anahtar yönetimi güvenlik protokollerinin kalbi olarak bilinmektedir. Çünkü burada bulunacak bir zafiyet saldırıya anahtar hakkında bilgi vermesi ile sonuçlanacaktır. Anahtar hakkında saldırının bilgi sahibi olması demek şifreli mesajları çözebilmesi anlamına gelmektedir. Çünkü gizlilik gibi servislerin güvenliği tamamen alttaki şifreleme algoritmasının kullandığı anahtara bağlıdır. WEP protokolü gibi anahtar yönetim standardının olmadığı gibi bazı durumlarda sağlayıcıların buldukları mesaj bütünlüğü sağlanmamış Diffie-Hellman anahtar anlaşım protokolü gibi çözümler durumu daha da kötüleştirmektedir.

IPSec protokolünün anahtar dağıtım işleminde IKE standart anahtar değişim protokolü kullanılmaktadır. Bu protokolün NRL protokol analizcisi ile biçimsel analizi [Meadows, 1999]'da yer almaktadır. IETF tarafından standartlaştırılmış bu anahtar değişim protokolü gerçekte güvenli bir protokoldür [Harkins and Carrel, 1998].

[Schneier, Wagner, 1997] de anlatıldığına göre SSL el sıkışma protokolü bazı ciddi hatalar içermekteydi, ama daha sonra bu hataların büyük ölçüde giderildiğinden bahsedilmektedir. Fakat bu hatalar bütün gerçeklemlerde yoktu.

[Borisov, et al, 2001] 802.11 standardının ortak anahtarın nasıl dağıtılacağı konusunda bir standardı olmadığından bahsetmektedir. 802.11 harici bir mekanizma ile dağıtılmış geniş çaplı paylaşılmış 4 anahtar dizisinin varlığına güvenmektedir. Her bir mesaj hangi diziden anahtar ile şifrelendiğini belirtmek için bir anahtar belirleme alanı içermektedir. Ayrıca bir tek anahtar içeren dizi ile de kullanabilecektir. Fakat bu seçenek fazla rağbet görmektedir. Pratikte birçok kuruluş bütün ağ için tek bir anahtar kullanmaktadır.

Bu uygulama sistemin güvenliğini ciddi biçimde yaralamaktadır. Çünkü, birçok kullanıcı tarafından paylaşılan paylaşılmış anahtarın güvenliğini

sağlamak çok zordur. Bazı sistem yöneticileri bu nedenden dolayı kullanıcılara anahtar vermektense sistemlerine sabitlemeyi tercih ediyorlar. Bu sadece göreceli olarak bir nevi güvenlik sağlamaktadır, çünkü aslında anahtar son kullanıcının bilgisayarında bulunmaktadır. Ayrıca bir tek anahtarın birçok kullanıcı tarafından kullanılması IV lerin aynı olma şansını arttırdığından dolayı saldırganın atak yapmasını kolaylaştırmaktadır. Bazı gerçeklemlerde ise sistem her açılışında veya PCMCIA kartları her takılışında IV'yi sıfırlamaktadır buda aynı anahtar dizisi kullanımını arttırmaktadır.

802.11 deki anahtar yönetiminin standart hale getirilmesi üzerinde önemle durulan bir konu haline gelmiştir. Eğer mümkünse her mobil istasyonun kendine özgü bir anahtarı olsun. Bu protokolde güvenli, kullanımı kolay bir otomatik anahtar yönetim mekanizmasının kurulması araştırmaya açık bir noktadır. Burada unutulmaması gereken sadece tek başına güvenli bir anahtar yönetim mekanizmasının kullanılması yukarıda anlatılan problemleri çözmemektedir.

2.4.7. Gerçekleme Hataları

Algoritma veya sistem ne kadar iyi tasarlanmış olursa olsun, eğer doğru şekilde gerçekleştirilmemiş ise hiçbir güvenlik sağlamaz. Bu konuda güvenlik protokolleri tasarımcıları geçmişte yaşanan gerçekleme hatalarından çok şey öğrendiler.

Sun Microsystem JDK 1.1 sürümünü yayınladığında, bu sürüm sayısal imza algoritması olan DSA gerçeklemesi içermekteydi. Sun DSA'i seçmişti, çünkü RSA, El Gamal veya Eliptik Eğri alternatifleri Amerikan ihraç kontrollerine tabi tutuluyordu. DSA sayısal imza algoritması alternatiflerinin aksine sadece imzalama işleminde kullanılabiliyor, şifreleme işlemi yapamıyor. Ayrıca DSA sayısal imza algoritmasının herhangi bir lisansı bulunmamaktadır.

DSA sayısal imzala algoritmasında kullanılan ve rasgele olarak üretilmesi gereken k değeri mevcuttur. Her bir sayısal imzanın farklı bir rasgele k değeri ile üretilmesi gerekmektedir. Eğer bir gerçekleştirme yanlışlıkla iki imzayı da aynı k değerini kullanarak atarsa, o zaman imza atma işleminde kullanılan özel anahtarın saldırgan tarafından hesaplanması çok kolay hale gelmektedir. Normalde ise saldırganın DSA algoritmasını kırması için, yani imza değerinden özel anahtarı elde edebilmesi veya imzayı bir şekilde taklit edebilmesi için, ayrık logaritma problemini çözmesi gerekmektedir. Bu problem ise NP-tam bir problemdir ve günümüz teknolojisini kullanarak çözülmesi çok uzun zaman sürmektedir. Fakat eğer iki imzada aynı k değeri kullanılıyor ise normal lise bilgisi ile çözülebilmektedir. Java geliştiricileri ilk önce sabit bir k kullanmışlardı, her imza için ayrı bir k kullanmaları gerektiğini de fark etmişlerdi. Fakat unuttular ve Java bu sabit k değeri ile piyasaya sürüldü ve bütün imzalar aynı k değeri kullanılarak hesaplandı. Programcılar DSA algoritmasını bilmiyor değillerdi. Her imza için farklı bir k değeri kullanılması gerektiği bilinmekle beraber, sadece gerçekleştirme hatası yapıldı. Bu hata Sun tarafından JDK 1.1.2'de düzeltildi.

Birçok uygulama için gerçekleştirme hatası, uygulamanın düzgün görünmemesinden kaynaklanır, örneğin, pencere boyutu ayarlanamıyordur veya font ekranda yanlış çıkıyordur. Fakat kriptolojide gerçekleştirme hatası, uygulamanın manasız olması ile sonuçlanmaktadır. Aslında, hatalar uygulamayı tehlikeli yapmaktadır, çünkü, kullanıcılar, kullandıkları uygulama servislerinin ve güvenli olduklarını sanmaktadırlar.

3. GÜVENLİK PROTOKOLÜ TASARIMINDA BLOK ŞİFRELEME KİPLERİNİN ANALİZİ VE SEÇİMİ

Bu bölümde eski DES çalışma kipleri ve AES için oluşturulmaya çalışılan yeni çalışma kipleri incelenecektir. Yeni çalışma kiplerinin oluşturulmasının nedeni, DES çalışma kiplerinde görülen bazı zayıflıkları gidermek ve 'bütünlük korumalı gizlilik' gibi birtakım yeni fikirlerin de endüstri tarafından kullanılabilir hale gelmesini sağlamaktır. Yeni çalışma kipleri oluşturulurken DES çalışma kiplerindeki zayıflıklar giderilmeye çalışılmıştır.

Yeni çalışma kiplerinin tasarım kriterlerine bölüm (2.4.)'de etraflıca değinildi. Bunlardan bazıları, CBC gibi geri beslemeli çalışma kipleri ile açık mesaj bloklarının paralel olarak şifrelenilememesi, bu sebeble yazılım ve donanım hız artırımı problemleri ve gizliliğin yanı sıra bütünlüğün de standart hale gelmesi olarak sıralanabilir.

Ancak yeni çalışma kipleri, literatürde mevcut CBC gibi eski çalışma kiplerindeki şifrelemeye karşı yapılan bazı aktif ve pasif saldırı yöntemlerine açısından değerlendirilmemiştir. Bu çalışmada DES ve AES çalışma kipleri, güvenlik protokollerine yapılan saldırı yöntemleri açısından değerlendirip karşılaştırılmıştır. Sonuç olarak DES çalışma kiplerinde karşılaşılan kimi zayıflıkların AES çalışma kiplerinde görülmediği ispatlanmıştır.

3.1. DES'in AES ile Değiştirilmesinin Nedenleri

DES blok şifreleme algoritmasının AES ile değiştirilmesinin nedenleri bölüm 2.1.1. de detaylı bir şekilde anlatılmıştır. Burada ise kısaca özetlenecektir.

Çizelge 3.1. de görülebileceği üzere, DES blok şifreleme algoritması 1970'li yılların sonuna doğru çıkmış bir algoritmadır. Anahtar uzunluğu 56 ikil, blok uzunluğu ise 64 ikildir. Günümüz teknolojisi ile 2^{56} adet anahtarın hepsinin denenmesi artık çok kısa sürmektedir.

Bu nedenden dolayı, DES yerine üç tane DES algoritmasının arka arkaya çalıştırılması anlamına gelen 3DES algoritması kullanılmaktadır. Bu yöntem ile algoritmanın anahtar uzunluğu 112 ikile çıkmakla beraber blok uzunluğu 64 ikil kalmaktadır. Anahtar uzayının iyileştirilmesine karşın 3DES algoritması çok yavaş bir algoritmadır. Bu nedenlerden dolayı NIST tarafından anahtar ve blok uzunluğu günümüz şartlarına daha uygun ve değişken bir algoritma oluşturmak için 1998 yılında bir çalışma yapılmaya başlanılmıştır. NIST bu amaçla bütün dünyadan isteyen herkesin kendi tasarladığı blok şifreleme algoritmasını önerebileceği bir yarışma başlatmıştır ve seçim işleminde ve göz önüne alınacak kriterler yine bütün dünyanın gözü önünde şeffaflık içinde sürdürülmüştür. Sonuçta 2001 yılında yarışma sonuçlanmış ve Belçikalı iki tasarımcının ürünü olan Rijndael blok şifreleme algoritması birinci seçilerek AES algoritması olmuştur.

Özellik	DES	3DES	AES	Yorum
Anahtar Uzunluğu	56 ikil	112 ikil	128, 192, 256 ikil	Anahtar uzunluğunun değişken olması AES için önemli bir avantajdır.
Blok Uzunluğu	64 ikil	64 ikil	128, 192, 256 ikil	Blok uzunluğunun değişken olması da AES için önemli bir avantajdır.
Çevrim Türü	Feistel	Feistel	Katman	Daha az çevrim kullanarak daha hızlı işlemler kullanılmıştır.
Çevrim sayısı	16	3 x 16	Anahtar ve blok uzunluğuna göre 10 ile 14 arası	Daha az çevrim kullanarak daha hızlı işlemler kullanılmıştır.
Hızı	Orta	Yavaş	Hızlı	AES kullandığı işlemler ve çevrim sayısının az olmasından dolayı hızlıdır.

Çizelge 3.1.: DES, 3DES ve AES Algoritmalarının Karşılaştırması.

3.2. DES Çalışma Kipleri

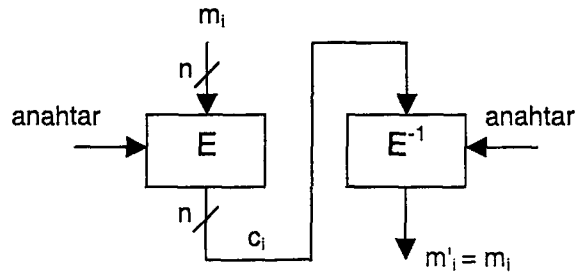
Bu bölümde DES çalışma kipleri incelenecektir.

3.2.1. Elektronik Kod Kitabı (ECB) Çalışma Kipi

Bu çalışma kipi DES çalışma kipleri arasındaki en zayıf çalışma kipi ve bir bloktan fazla açık mesajın şifrenmesi amacı ile kullanımı kesinlikle tavsiye edilmemektedir. Bu çalışma kipi işleyişi Şekil 3.1.'de görüldüğü gibidir.

a) Şifreleme

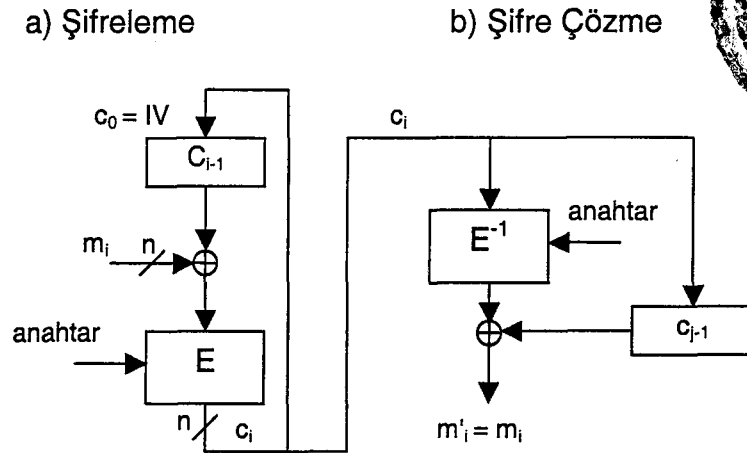
b) Şifre Çözme



Şekil 3.1.: DES Algoritmasının ECB Çalışma Kipi

3.2.2. Şifre Bloğu Zincirleme (CBC) Çalışma Kipi

En çok kullanılan DES çalışma kipi. Şifreli mesajın geri beslenmesi ile çalışmaktadır. Kendiliğinden uyumlu bir çalışma kipi. Şifreli mesajda oluşan hataları açık mesaja blok uzunluğu kadar büyütür. Şifreleme ve şifre çözme algoritmasını kullanmaktadır. Bu çalışma kipi işleyişi Şekil 3.2.'de görüldüğü gibidir.



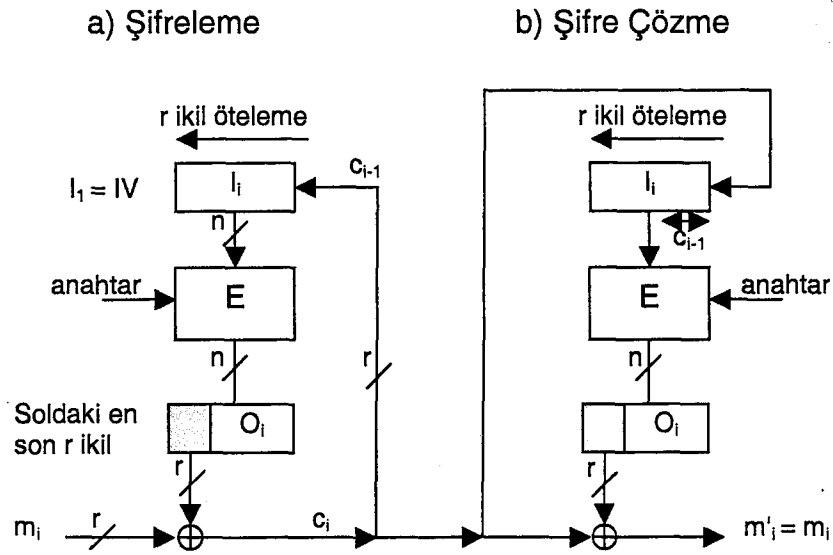
Şekil 3.2.: DES Algoritmasının CBC Çalışma Kipi

3.2.3. Şifre Geri Besleme (CFB) Çalışma Kipi

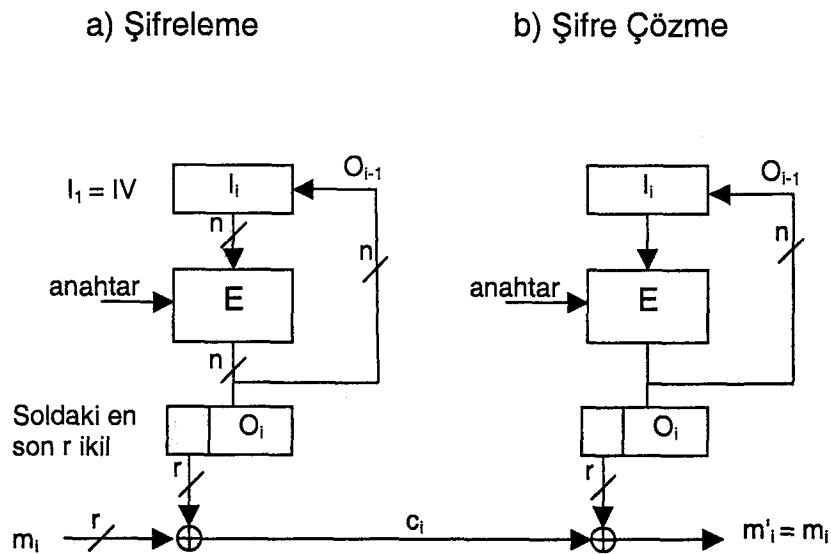
En çok kullanılan DES çalışma kiplerinin ikincisidir. CBC çalışma kipinde olduğu gibi şifreli mesajın geri beslenmesi ile çalışmaktadır. Ancak mesajı algoritmanın blok uzunluğundan daha kısa uzunluklarda işleyebilmektedir. CFB çalışma kipi de kendiliğinden uyumlu bir çalışma kipi. Şifreli mesajda oluşan hataları açık mesaja öteleme uzunluğu kadar büyütür. CBC çalışma kipinin aksine sadece şifreleme algoritmasını kullanmaktadır. Bu çalışma kipinin işleyişi Şekil 3.3.'de görüldüğü gibidir.

3.2.4. Çıkış Geri Besleme (OFB) Çalışma Kipi

CFB çalışma kipine benzemektedir. Fakat şifreli mesajın geri beslemesi bu çalışma kipinde yoktur. Bunun yerine algoritma çıktısı geri beslenmektedir. Dolayısı ile bu çalışma kipi blok şifreleme algoritmasını dizi şifreleme şeklinde çalıştırmaktadır. Bu nedenle kullanıcılarının dizi şifreleme sistemlerinin zayıflıklarından haberdar olmaları gerekmektedir. Bu çalışma kipi de mesajı blok uzunluğu ve daha az uzunluklarda çalıştırabilmektedir.



Bu çalışma kipinin önemli bir dezavantajı vardır. Yukarıda, bu çalışma kipi algoritmayı dizi şifreleme sistemi gibi çalıştırır denilmişti. OFB çalışma kipinin ürettiği anahtar dizisinin periyodu değişkendir ve önceden tahmin edilememektedir. Dolayısı ile bazı anahtarlar için periyodu çok küçük olabilmektedir. Böyle bir durumda dizi şifreleme sistemlerindeki 'anahtar dizisi tekrar kullanımı'na benzer bir durum ortaya çıkmaktadır ve şifrelenen mesajların anahtar bilgisi olmadan da çözülebilme ihtimali vardır. Bu çalışma kipinin işleyişi Şekil 3.4.'de görüldüğü gibidir.



3.3. Önerilen AES Çalışma Kiplerinden Bazıları

Blok şifreleme algoritmalarında DES çalışma kiplerinin kullanımındaki bazı sakıncalar bölüm (2.4.)'de ayrıntılı bir şekilde anlatıldı. DES çalışma kipleri uzun zamandır kullanıldığı için bilim çevreleri tarafından güvenlikleri ve açıkları çok iyi tahlil edilmiştir. NIST, DES çalışma kiplerinin bu zayıflıklarını gidermek amacıyla bir AES blok algoritması ile çalışacak yeni çalışma kiplerinin bulunması için yarışma düzenlemiştir. AES algoritmasına önerilen çalışma kipleri arasında OCB ve CTR gibi güzel tasarım kriterleri bulunan çalışma kipleri de mevcuttur.

3.3.1. Ofset Kod Kitabı (OCB) Çalışma Kipi

OCB çalışma kipi, hem mesaj bütünlüğünü hem de mesaj gizliliğini sağlamaktadır. Eğer IPsec protokolünde CBC çalışma kipi yerine OCB çalışma kipi kullanılsaydı, [Bellare, 1996] de anlatılan saldırı yöntemleri bu protokole karşı başarısız olacaktı. Çünkü, OCB çalışma kipi mesaj bütünlüğünü de sağlamaktadır. Zaten bu makalede de söz konusu saldırı yöntemlerinin büyük bir kısmı, mesaj bütünlüğünün korunmadığı farz edildiğinde geçerlilik kazanmaktadır. Bu çalışma kipi bölüm (2.4.1.)'de anlatılmıştır. Bu çalışma kipi [Rogaway, et al, 2001]'de detaylı anlatılmaktadır. Bu çalışma kipi işleyişi Şekil 3.5.'de görüldüğü gibidir.

OCB Çalışma Kipi Algoritması

1. Şifreleme işlemi

GRİŞ: Açık mesaj M , K ve IV .

ÇIKIŞ: Şifreli mesaj C ve etiket.

a. $M, M_1, \dots, M_m$ şeklinde açık mesaj bloklarına bölünür.

b. $L \leftarrow E_K(0^n)$

c. $R \leftarrow E_K(IV \oplus L)$

d. $i, 1\text{'den } m\text{'e kadar } Z_i = \gamma_i \cdot L \oplus R$

e. $i, 1\text{'den } (m - 1)\text{'e kadar } C_i \leftarrow E_K(M_i \oplus Z_i) \oplus Z_i$

f. $X[m] \leftarrow \text{uzunluk}(M_m) \oplus L \cdot x^{-1} \oplus Z_m$

g. $Y[m] \leftarrow E_K(X_m)$

h. $C_m \leftarrow Y_m \oplus M_m$

i. $C \leftarrow C_1, \dots, C_m$

j. $\text{Etiket} \leftarrow M_1 \oplus \dots \oplus M_{m-1} \oplus C_m \cdot 0^* \oplus Y_m$

k. $T \leftarrow E_K(\text{Etiket} \oplus Z_m)$ [ilk τ ikil]

l. $\text{ÇIKIŞ } C \leftarrow C \parallel T$

2. Şifre çözme işlemi

GİRİŞ: Şifreli mesaj C , K ve IV .

ÇIKIŞ: Açık-mesaj M ve bütünlük doğruluğu bilgisi.

a. $C, C_1, \dots, C_m$ şeklinde şifreli mesaj bloklarına bölünür.

b. $L \leftarrow E_K(0^n)$

c. $R \leftarrow E_K(IV \oplus L)$

d. $i, 1\text{'den } m\text{'e kadar } Z_i = \gamma_i \cdot L \oplus R$

e. $i, 1\text{'den } (m - 1)\text{'e kadar } M_i \leftarrow E_K(C_i \oplus Z_i) \oplus Z_i$

g. $Y[m] \leftarrow E_K(X_m)$

h. $M_m \leftarrow Y_m \oplus C_m$

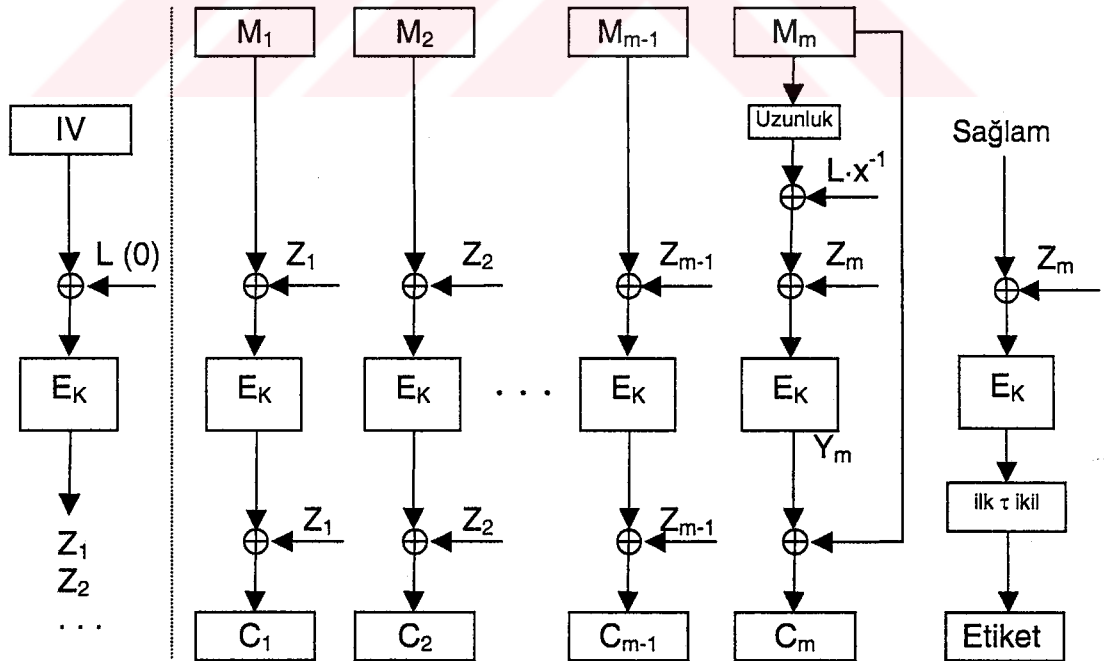
i. $M \leftarrow M_1, \dots, M_m$

j. $\text{Etiket} \leftarrow M_1 \oplus \dots \oplus M_{m-1} \oplus C_m \oplus Y_m$

k. $T' \leftarrow E_K(\text{Etiket} \oplus Z_m)$ [ilk τ ikil]

l. Eğer $T' = T$ ise ÇIKIŞ $\leftarrow M$, değilse ÇIKIŞ $\leftarrow$ geçersiz.

Etiket T için 64 ikil uzunluk önerilmektedir. 32 ikil uzunluktaki etiketler perakende bankacılıkta standart olarak kullanılıyor. IPsec'de 96 ikilik etiket kullanılırken, tasarımcıları 80 ikilin üzerindeki etiketlerin sağladığı ekstra güvenliğin tartışılır olduğunu belirtmekle beraber, şifreli mesajı uzattığını da vurguluyorlar.

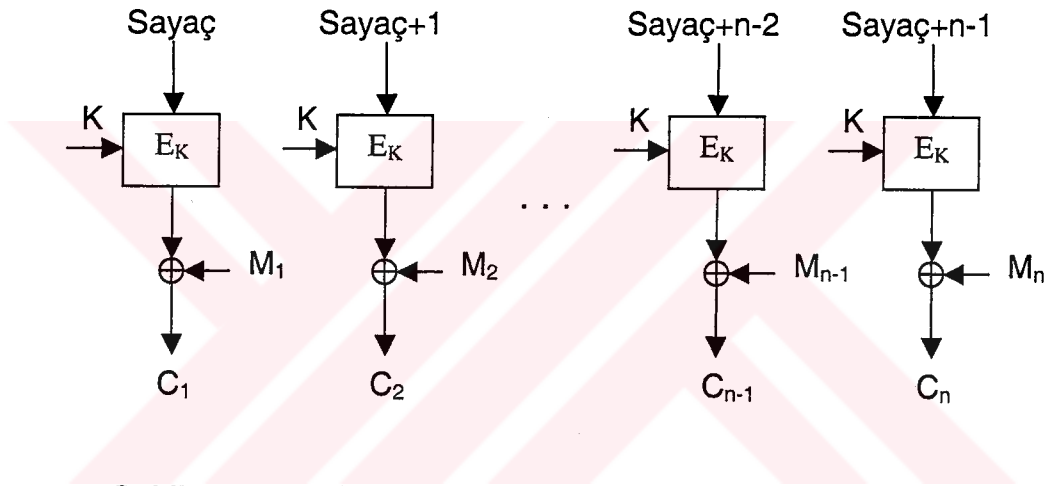


Şekil 3.5.: Teklif Edilen AES Çalışma Kipi Olan OCB Çalışma Kipi.

Şekil 3.5.'de $L(0) = B_Enc_K[0^n]$, $Z_i = Z_{i-1} \oplus L(nty(i))$ ve $L(i)$, $L(i-1)$ 'in kaydırma ve şartlı modül 2 toplaması sonucunda elde edilmektedir. Bu çalışma kipi hakkındaki detaylar ve tanımlar [Rogaway, et al, 2001]'de mevcuttur.

3.3.2. Sayaç (CTR) Çalışma Kipi

AES öneri çalışma kipi olan CTR çalışma kipi, OCB çalışma kipinden oldukça farklıdır. Sadece mesaj gizliliğini sağlamaktadır. Bu çalışma kipinin avantajları ve dezavantajları bölüm (2.4.1.)'de anlatılmıştır. Detayları ise [Lipmaa, et al, 2001]'de bulmak mümkündür.



Şekil 3.6.: Teklif Edilen AES Çalışma Kipi Olan CTR Çalışma Kipi.

CTR Çalışma Kipi Algoritması

1. Şifreleme işlemi

GİRİŞ: Açık mesaj M , K ve IV .

ÇIKIŞ: Şifreli mesaj C .

a. M , M_1 , ..., M_m şeklinde açık mesaj bloklarına bölünür.

b. i , 1'den m 'e kadar $C_i = E_K(\text{Sayaç}) \oplus M_i$

c. $C \leftarrow C_1, \dots, C_m$

d. ÇIKIŞ C

2. Şifre çözme işlemi

GİRİŞ: Şifreli mesaj C, K ve IV.

ÇIKIŞ: Açık-mesaj M.

a. C, $C_1, \dots, C_m$ şeklinde şifreli mesaj bloklarına bölünür.

b. i, 1'den m'e kadar $M_i = E_K(\text{Sayaç}) \oplus C_i$

c. $M \leftarrow M_1, \dots, M_m$

d. ÇIKIŞ M

3.4. İlk Vektör IV Konusunda Bazı Problemler

Simetrik şifreleme algoritmalarında kullanılan ilk vektör, şifreleme işlemine rasgelelik katmakla birlikte aynı anahtar kullanıldığı zaman bile aynı anahtar dizisi kullanımının önüne geçmektedir. 'Bilinen Rasgele Sayı' olarak adlandırılmasının nedeni, genellikle açık olarak gönderildiği için herkes tarafından bilinmesidir. Eğer IV, açık olarak değil de şifreli olarak karşı tarafa gönderilirse veya açık olarak gönderilip işleme girmeden önce bir nevi şifrelemeden geçirilirse, saldırgan şifreleme işleminde gerçekte hangi IV kullanıldığını bulamamaktadır. Çünkü saldırgan IV'nin şifrelenmesinde kullanılan anahtar bilgisine sahip değildir. Yalnız bu yöntem kullanıldığı halde, dizi şifreleme sistemlerinde saldırgan aynı anahtar dizisi kullanımını takip edebilecektir. Zira, aynı anahtar ile aynı IV'nin şifreli halleri de aynı olacaktır.

Blok şifreleme sistemlerinde, örneğin CBC çalışma kipi kullanılırsa, aynı IV ve aynı anahtar kullanıldığı taktirde bile şifrelenecek mesajların ilk açık

mesaj blokları farklı olduğu zaman şifreli mesajlar tamamen farklı olmaktadır. Yukarıdaki paragrafta bahsedilen basit işlem kullanıldığında, mesaj gizliliği bölümünde (2.4.3.) sözü geçen CBC çalışma kipi kullanılan şifreleme işlemindeki ilk açık mesaj bloğunun değiştirilebilmesi saldırısı, (2.17.) aşağıdaki denklemde gösterildiği gibi uygulanamaz hale gelmektedir. Çünkü, artık saldırgan şifreleme işleminde kullanılan gerçek IV değerini bilmemekte ve tahmin edememektedir.

$$P_1 = B_Enc_K (IV) \oplus B_Dec_K (C_1) \quad (3.1.)$$

Bunun nedeni P_1 'in, C_1 şifreli mesaj bloğunun şifre çözme işlemi sonucunda çıkan değerin IV ile değil, IV'nin şifrelenmesinden çıkan değer ile modül 2 toplamından hesaplanıyor olmasıdır.

Genellikle daha pratik olduğu için IV, 802.11 WEP protokolünde olduğu gibi sayaç gibi, yani birer birer artan şekilde kullanılmaktadır. IV bu şekilde kullanıldığı zaman saldırgan bir sonraki IV'nin ne olduğunu bilmektedir. Ayrıca, CBC çalışma kipindeki IV'nin sayaç olarak kullanılmasının sakıncalı olduğu zaten bilinmektedir [Rogaway, 1995]. Bunun yerine en basitinden geri beslemeli ötemeli yazıcılar kullanılabilir. Bu basit yöntemin kullanılmasıyla, birbiri ardınca kullanılan IV'lerin Hamming ağırlıklarının beklenen değere yakın olması amaçlanmıştır.

3.5. AES Çalışma Kiplerinin Güvelik Analizi ve Seçimi Üzerine Yapılan Özgün Çalışmalar

Bu bölümde NIST'in yarışmasına AES çalışma kipi olarak önerilen ve henüz standartlaşmamış OCB ve CTR çalışma kiplerinden bahsedilecektir. Ayrıca bu çalışma kiplerinin bölüm (2.4.) ve (3.2.)'de anlatılan DES çalışma kipine uygulanan saldırı yöntemlerine karşı güvenli oldukları ispatlanacaktır.

Aşağıda [Bellovin, 1996]' deki kes-yapıştır, şifreli mesajın okunması, oturum kaçıрма, IV'yi kötüye kullanma saldırı yöntemlerinin OCB ve CTR

alıřma kipleri kullanıldıđı taktirde uygulanamayacađı madde madde aıklanmıř ve ispatlanmıřtır.

3.5.1. Kes - Yapıřtır Saldırısı

Bölüm (2.4.1)'de de aıklandıđı gibi DES CBC alıřma kipine, mesaj bütünlüğü'nün sađlanmadıđı durumlarda kes-yapıřtır saldırısı (ele geirilen mesajlardaki IV farklı olduđu durumlarda dahi) uygulanabilmektedir. [Bellovin, 1996]'da bu saldırı yöntemi IPsec protokolünün bir önceki sürümü için incelenmiř ve kes yapıřtır saldırı yönteminin ok ciddi aıklıklar oluřturduđu gösterilmiřtir.

Bu alıřma kipinde herhangi bir řifreli mesajın birbirini izleyen alt řifreli mesaj blok kümesinin řifresi, bařarılı bir řekilde özölebilmektedir.

Bununla beraber OCB alıřma kipi, mesaj bütünlüğü'nü ve gizliliğini beraberce sađlayan bir alıřma kipi olduđundan kes-yapıřtır saldırı yöntemi bu alıřma kipi için kullanılamamaktadır. OCB alıřma kipinin mesaj bütünlüğü'nü sađlayan özelliğini kullanmadığımız durumda dahi, eđer řifrelenecek her bir mesajlar için farklı IV kullanılıyor ise, kes-yapıřtır saldırı yöntemi bu alıřma kipi için kullanılamamaktadır. Bunun nedeni ařađıda aıklanmaktadır.

CBC alıřma kipinde IV, sadece ilk aık mesaj bloğunun řifrelenmesinde etkin rol oynamaktadır. Daha sonraki bloklarda ise (2.13.) numaralı denklemden ve bölüm (3.2.2)'den anlařılacađı gibi bir önceki řifreli mesaj blođu IV görevini görmektedir. Bölüm (2.4.1)'de anlatılan ve CBC alıřma kipine uygulanan kes-yapıřtır saldırı yönteminin bařlıca nedeni de CBC alıřma kipinin bu özelliğinden kaynaklanmaktadır.

OCB alıřma kipinde durum biraz daha farklıdır. Kullanılan IV sadece ilk aık mesaj bloğunun řifrelenmesinde etkin deđildir. Kullanılan IV'nin türevleri her bir aık mesaj blođu için farklıdır ve řifrelenmelerinde etkin rol oynamaktadır.

Eğer saldırgan, ele geçirdiği OCB çalışma kipi kullanılarak şifrelenmiş bir mesajın birbirini izleyen alt mesaj bloklarını içeren yeni bir mesajı hatta enjekte etmek istiyor ise bu şifreli mesaj blokları ilk şifreli mesaj bloğundan başlamak zorundadır. Bunun nedeni aşağıda detaylı bir şekilde anlatılacak olan, OCB çalışma kipi kullanılarak yapılan şifreleme işlemindeki anahtar bilgisi bilinmeden, söz konusu birbirini izleyen şifreli mesaj bloklarının sıra numaralarının değiştirilemeyeceği gerçeğidir. Örneğin, saldırganın OCB çalışma kipi kullanılarak şifrelenmiş C_2 , C_3 , C_4 şifreli mesaj bloklarına sahip olduğu varsayıldığında, bu şifreli mesaj bloklarının şifrelenme yöntemi aşağıdaki gibidir.

$$\begin{aligned}
 C_2 &= Z_2 \oplus E_K (Z_2 \oplus M_2)), \\
 C_3 &= Z_3 \oplus E_K (Z_3 \oplus M_3)), \\
 C_4 &= Z_4 \oplus E_K (Z_4 \oplus M_4)), \\
 Z_2 &= Z_1 \oplus L (ntz (2)), \text{ ntz } (2) = 1, \\
 Z_3 &= Z_2 \oplus L (ntz (3)), \text{ ntz } (3) = 0, \\
 Z_4 &= Z_3 \oplus L (ntz (4)), \text{ ntz } (4) = 2.
 \end{aligned} \tag{3.2}$$

Burada $L (i)$, $L (i-1)$ 'in kaydırma ve şartlı modül 2 toplamı ile ve $ntz (i)$ ise 2^z 'nin i 'yi böldüğü en büyük z değeridir. M_2 , M_3 , M_4 ise C_2 , C_3 , C_4 şifreli mesaj bloklarına karşılık gelen açık mesaj bloklarıdır. Daha sonra saldırganın kes-yapıştır saldırısında kullanmak üzere C_1' , C_2' , C_3' olarak bu şifreli mesaj bloklarının sıra numaralarını değiştirmeye çalıştığı farz edilecek. Saldırgan bu mesajların mesaj çözme işlemi sonucunda, şifreleme işleminde kullanılan anahtarı bilmeden açık mesaj blokları $M_1' = M_2$, $M_2' = M_3$, $M_3' = M_4$ 'ü elde etmeyi amaçlamaktadır. Bu nedenle uygun bir IV' bulması gerekmektedir. C_1' , C_2' , C_3' şifreli mesaj bloklarının şifrelenme yöntemi aşağıdaki gibidir.

$$\begin{aligned}
 C_1' &= Z_1' \oplus E_K (Z_1' \oplus M_1')), \\
 C_2' &= Z_2' \oplus E_K (Z_2' \oplus M_2')), \\
 C_3' &= Z_3' \oplus E_K (Z_3' \oplus M_3')), \text{ ve} \\
 Z_1' &= E_K (IV' \oplus L (ntz (1))), \text{ ntz } (1) = 0, L (0) = E_K (0), \\
 Z_2' &= Z_1' \oplus L (ntz (2)), \text{ ntz } (2) = 1.
 \end{aligned} \tag{3.3}$$



$$Z_3' = Z_2' \oplus L(\text{ntz}(3)), \text{ntz}(3) = 0,$$

Burada dikkat edilmesi gereken, saldırganın C_1' şifreli mesaj bloğunun şifre çözme işlemi sonucunda $M_1' = M_2$ açık mesaj bloğunu elde etmeyi amaçlamasıdır. C_1' şifreli mesaj bloğunun şifrelenme yöntemi aşağıdaki gibidir.

$$\begin{aligned} C_2 &= Z_2 \oplus E_K(Z_2 \oplus M_2), \\ C_1' &= Z_1' \oplus E_K(Z_1' \oplus M_1'), \text{ ve} \\ Z_2 &= Z_1 \oplus L(\text{ntz}(2)), \text{ntz}(2) = 1, \\ Z_1' &= E_K(IV' \oplus L(\text{ntz}(1))), \text{ntz}(1) = 0, L(0) = E_K(0). \end{aligned} \quad (3.4.)$$

Saldırganın başarılı olabilmesi için $Z_1' = Z_2$ olmalıdır. Z_1' bir IV' ve $E_K(0)$ 'ın modül 2 toplamlarının K anahtarı ile şifrelenmesinden elde edildiği için, saldırgan burada kullanılan ilk vektörü (IV') K anahtarını bilmeden hesaplayamamaktadır. Dolayısı ile bu IV' bilgisine sahip olamadan da şifreli mesaj bloklarının sıra numaralarını değiştirememektedir. Benzer şekilde saldırgan, şifreli mesaj bloklarını sıra numaralarını değiştirerek ilk numaradan değil herhangi bir numaradan da başlatamaz. Burada saldırganın ele geçirdiği OCB çalışma kipi ile şifrelenmiş bir şifreli mesajın, birbirini izleyen alt şifreli mesaj bloklarının sıra numaralarını değiştiremeyeceği ispatlandı.

Buna benzer şekilde saldırganın farklı IV ve OCB çalışma kipi kullanılarak şifrelenmiş M^1 ve M^2 mesajlarını ele geçirdiği ve bu mesajların bazı parçalarını birbirine ekleyerek yeni bir mesaj oluşturmak istediği farz edilecek.

$$\begin{aligned} M^1: IV^1, C_i^1 &= Z_i^1 \oplus E_K(Z_i^1 \oplus M_i^1), 0 < i < n \text{ ve } n \in \mathbb{N}, \\ M^2: IV^2, C_j^2 &= Z_j^2 \oplus E_K(Z_j^2 \oplus M_j^2), 0 < j < m \text{ ve } m \in \mathbb{N}, \\ Z_1^{1,2} &= E_K(E_K(0) \oplus IV^{1,2}), \\ Z_i^{1,2} &= Z_{i-1}^{1,2} \oplus L(\text{ntz}(i)). \end{aligned} \quad (3.5.)$$



Saldırgan, şifrelemede kullanılan K anahtar bilgisine sahip olmadığından yukarıda ispat edildiği üzere M^1 mesajının ilk şifreli mesaj bloğu olan C_1^1 ve ilgili IV^1 den başlamak durumundadır. Saldırganın ikinci mesajın bir parçasını ilk mesajın baştan bir parçasına eklemek istediği farz edilecek, örneğin, $(IV^1), C_1^1, C_2^1, C_3^1, (IV^2), C_1^2, C_2^2, C_3^2$. Daha sonra saldırgan bu oluşturduğu yeni mesajı hatta enjekte etmek maksadı ile kullanacaktır. Aşağıda, saldırganın bu işlemi yapmasının da mümkün olmadığı ispatlanacaktır.

Şekil 3.5.'de, Z_i^1 lerin Z_{i-1}^1 lere ve Z_1^1 inde IV^1 ve $E_K(0)$ 'ın modül 2 toplamlarının şifrlenmesine bağlı olduklarından bahsedilmiştir. Öyleyse, IPsec sistemi C_1^1, C_2^1, C_3^1 şifreli mesaj bloklarının şifresini doğru biçimde açacaktır ama C_1^2, C_2^2, C_3^2 şifreli mesaj bloklarını doğru şekilde açamayacaktır. Çünkü, saldırgan şifrelemede kullanılan anahtar bilgisine sahip değildir ve bu bilgi olmadan da M^2 mesajının bloklarını doğru şekilde ayarlayamamaktadır. Aslında anahtar bilgisine sahip olmak zaten mesajlara ulaşmak anlamına gelmektedir. 'Doğru şekilde ayarlayamaz' sözünü bu anlamın yüklenmesi gerekmektedir.

$$\begin{aligned} C_3^1 &= Z_3^1 \oplus E_K(Z_3^1 \oplus M_3^1), \\ C_1^2 &= Z_1^2 \oplus E_K(Z_1^2 \oplus M_1^2), \text{ ve} \\ Z_3^1 &= Z_2^1 \oplus L(\text{ntz}(3)), \text{ntz}(3) = 0, \\ Z_1^2 &= E_K(IV^2 \oplus L(\text{ntz}(1))), \text{ntz}(1) = 0, L(0) = E_K(0), \end{aligned} \quad (3.6.)$$

Bu da gösteriyor ki saldırganın yeni oluşturmaya çalıştığı karma şifreli mesaj blokları hem IV^1 yani Z_1^1 lere hem de IV^2 yani Z_1^2 lere bağlı olacaktır. Oysa, yukarıda anlatıldığı gibi şifreleme işleminde kullanılan anahtar bilgisine sahip olmadan bunun başarılması mümkün değildir. Çünkü, Z_i^1 ler hem IV^2 ye hem de $E_K(0)$ 'ın şifrlenmiş haline bağlıdır. Şifreleme işlemindeki anahtar bilgisi olmadan bu değerlerle oynamak söz konusu olamaz.

CTR çalışma kipinde eğer sayaç açık olarak gönderiliyor ve bu şekilde kullanılıyor ise saldırgan şifreli mesaj bloklarının sırasını değiştirebilir. Çünkü, CTR çalışma kipinde şifreli mesaj blokları aşağıdaki gibi oluşturulmaktadır.

$$\begin{aligned} C_1 &= B_Enc(\text{sayaç}) \oplus P_1, \\ C_2 &= B_Enc(\text{sayaç}+1) \oplus P_2, \\ C_3 &= B_Enc(\text{sayaç}+2) \oplus P_3, \\ &\dots \end{aligned} \quad (3.7.)$$

Saldırgan sayaç bilgisine sahip olduğu müddetçe, şifreli mesaj bloklarının istediği yerinden başlayabilir. Tek yapması gereken bu ardışık şifreli mesaj bloklarının önüne, ilgili sayaç bilgisini koymasıdır. Ancak sayaç bilgisi şifreli olarak gönderiliyor veya açık gönderilip OCB çalışma kipinde olduğu gibi bir nevi şifreleme işleminden sonraki hali asıl sayaç olarak kullanılıyor ise, saldırgan (yukarıda anlatılan) şifreli mesaj bloklarının istediğinden başlayamayacak, tıpkı OCB çalışma kipinde olduğu gibi ilk şifreli mesaj bloğundan başlamak zorunda kalacaktır. Çünkü saldırgan, şifreleme işleminde kullanılan asıl sayaç bilgisini anahtar bilgisi olmadan tahmin edemeyecektir.

Ayrıca, eğer sayaç bilgisi açık gönderiliyor, iki mesajı şifrelemek için kullanılan sayaç bilgileri bu iki mesajın bir bölümünde örtüşüyor ve mesaj bütünlüğü korunmuyor ise bu durumda saldırgan kes-yapıştır saldırısı yapabilmektedir.

$$\begin{aligned} C_{1, \text{sayaç}}, C_{2, \text{sayaç}+1}, \dots, C_{n, \text{sayaç}+n-1} \\ C'_{1, \text{sayaç}+k}, C'_{2, \text{sayaç}+k+1}, \dots, C'_{n, \text{sayaç}+k+n-1} \end{aligned} \quad (3.8.)$$

Burada eğer $1 < k < n$ ise bazı açık mesaj bloklarının şifrlenmesinde aynı sayaç bilgisi kullanılıyor demektir. Bu da saldırganın bu bloklar için kes-yapıştır saldırı yöntemini uygulayabileceği anlamına gelmektedir.

Ancak sayaç bilgisi şifreli gönderiliyor veya açık gönderilip bir nevi şifreleme işleminden sonraki hali asıl sayaç olarak kullanılıyor ise, saldırgan mesajlarda sayaç bilgisinin örtüşüp örtüşmeyeceğini tahmin edememektedir. Çünkü, şifreleme işleminde kullanılan anahtarı bilmemektedir. Bu sebeple kes yapıştır saldırı yöntemini kullanamayacaktır. [Lipmaa, et al, 2001]'de sayaç bilgisinin beş değişik kullanım biçimi anlatılmaktadır.

3.5.2. Şifreli Mesajın Okunabilmesi Saldırısı

Bu saldırı yöntemine bölüm (2.4.3.)'de ayrıntılı bir şekilde yer verilmiştir. Bu saldırı yönteminde saldırgan, yasal kullanıcı ve kendisi tarafından gönderilen, kötü niyetli veri içeren iki mesajı hattı dinleyerek kaydetmekte ve bu mesajları kes-yapıştır saldırı yöntemi yardımıyla yeni bir mesaj oluşturmak için kullanmaktadır.

Burada yapılan şifreleme işleminde, CBC yerine OCB çalışma kipi kullanıldığında saldırgan bu saldırı yönteminde başarısız olacaktır. Bunun nedeni, saldırganın kaydettiği mesajlardaki IV'lerin farklı olması ve bölüm (3.5.1.)'de ispatlandığı gibi OCB çalışma kipi ile şifrelenmiş mesajlara kes-yapıştır saldırı yönteminin başarı ile uygulanamamasıdır.

CTR çalışma kipi kullanıldığı durumda da saldırgan bu saldırı yöntemini uygulayamamaktadır. Çünkü, her mesaj için farklı bir sayaç kullanılmaktadır ve yukarıdaki madde de ispatlandığı gibi CTR çalışma kipine kes-yapıştır saldırı yönteminin uygulanması mümkün değildir. Yalnız, sayaç bilgisinin açık gönderildiği durumlarda farklı mesajlar için sayaç örtüşmesi oluyor ise yine bir önceki maddede ispatlandığı gibi saldırgan kes-yapıştır atağını kullanabilmektedir.

CTR çalışma kipi kullanıcıları açık sayaç bilgisini kullanmasını tercih ediyor ve mesaj bütünlüğü kullanmıyorlarsa, farklı mesajlardaki sayaç örtüşmesi konusunda çok dikkatli olmalılardır.

3.5.3. Oturum Kaçırma Saldırısı

Bölüm (2.4.3.)'de detaylandırılan bu saldırı yöntemi bölüm (3.5.2.)'de anlatılan saldırı yöntemine çok benzemektedir. Bu saldırı yönteminde yapılan şifreleme işleminde, CBC yerine OCB çalışma kipi kullanıldığı zaman, saldırgan bu saldırı yönteminde yine başarısız olacaktır. Bunun nedeni, saldırganın kaydettiği mesajlardaki IV'lerin farklı olması ve bölüm (3.5.1.)'de ispatlandığı gibi OCB çalışma kipi ile şifrelenmiş mesajlara kes-yapıştır saldırı yönteminin başarı ile uygulanamamasıdır.

CTR çalışma kipinde de durum farklı değildir. Yalnız bir önceki maddede değinildiği gibi farklı mesajlardaki sayaç bilgisi örtüşmesi konusunda dikkatli olunmalıdır.

3.5.4. IV'yi Kötüye Kullanma Saldırısı

CBC çalışma kipi ile yapılan şifreleme işleminde ilk açık mesaj bloğunun şifreleme işlemi (2.17.) numaralı denklem ile gösterilmiştir. Bu denklem dikkatlice incelendiğinde, IV üzerinde yapılan herhangi bir değişikliğin ilk açık mesaj bloğu olan P_1 bloğunun aynı ikil pozisyonlarını etkilediği anlaşılmaktadır. Genellikle ilk mesaj bloğu protokol başlığı olduğu için saldırgan, başlığını (ilk bloğunu) tahmin ettiği mesajlar üzerinde istediği değişikliği yapabilmektedir.

Bu saldırının CBC çalışma kipinde etkili olmasının bir diğer nedeni de, mesaj bütünlüğü koruma işleminin isteğe bağlı olması sebebi ile çoğu zaman kullanılmamasıdır.

Şayet OCB çalışma kipi tercih edilmiş olsaydı bu saldırı yöntemi OCB çalışma kipinin bütünlük koruma özelliği kullanılmadığı durumlarda bile uygulanamayacaktı. Bunun nedeni, herhangi bir şifreli mesaj çözme işleminde IV'nin direk olarak değil aşağıda gösterildiği gibi bir dizi şifreleme işleminden sonra kullanılmasıdır.

$$\begin{aligned}
C_1 &= Z_1 \oplus B_Enc_{K, IV} (Z_1 \oplus M_1), \\
M_1 &= Z_1 \oplus B_Dec_{K, IV} (Z_1 \oplus C_1), \\
Z_1 &= E_{K, IV} (IV \oplus L (ntz (1))), \text{ ntz } (1) = 0, L (0) = E_K (0),
\end{aligned}
\tag{3.9.}$$

Dolayısı ile saldırgan OCB çalışma kipi kullanıldığı durumda bu saldırı yöntemini de şifreleme işleminde kullanılan anahtar bilgisini bilmeden uygulayamamaktadır. CTR çalışma kipinde ilk açık mesajın şifresinin çözülme işlemi aşağıdaki gibidir,

$$P_1 = B_Enc_{K, IV} (\text{sayaç}) \oplus C_1 \tag{3.10.}$$

Saldırganın, sayaç (IV) bilgisine sahip olmasına rağmen anahtar bilgisine de sahip olmadan sayaç bilgisinin şifreli halini bilinmesi imkansızdır. Burada CBC çalışma kipinde olduğu gibi sayaç bilgisinden gelen bir zayıflık olmamasına karşın, blok şifreleme algoritmasının dizi şifreleme gibi çalışması sonucu bu çalışma kipinde dizi şifreleme sistemlerinin zayıflıkları aynen görülmektedir. Bu sebeple mesaj bütünlüğün korunmadığı durumlarda saldırgan şifreli mesaj ikilerini değiştirerek açık mesajı değiştirmiş olmaktadır. Bu hal CBC çalışma kipindeki durumda daha kötüdür. Çünkü bu saldırı, sadece ilk açık mesaj bloğunu değil mesajın tümünü etkilemektedir.

3.6. Güvenlik Protokolü Tasarımında Çalışma Kipi Seçimi

Bu bölümde güvenlik protokolleri için kullanılacak blok şifreleme çalışma kipinin seçimi üzerinde durulacaktır. Aşağıda çalışma kiplerinin seçimi ve nedenleri madde madde açıklanmıştır.

- a. Bölüm 3.5. de anlatıldığı üzere CBC çalışma kipine bazı saldırılar düzenlenebilmektedir. Bu saldırı yöntemlerinin çoğu yeterli mesaj bütünlüğü koruması sağlanmadığından kaynaklanmaktadır. Halbuki, OCB çalışma kipi hem mesaj bütünlüğünü hem de mesaj gizliliğini

sağlamaktadır. Dolayısı ile bu tip bir çalışma kipi kullanımı ile bir çok saldırı yöntemi önlenmiş olacaktır.

- b. Bölüm 3.5.1. de anlatıldığı gibi OCB çalışma kipinin mesaj bütünlüğü koruma özelliğini kullanmadığımız durumlarda bile Bölüm 3.5.1., 3.5.2., 3.5.3., 3.5.4. de anlatılan saldırı yöntemleri OCB çalışma kipinde kullanıkan tasarım kriterleri sayesinde uygulanamamaktadır.
- c. OCB çalışma kipi kadar güçlü olmamasına karşın tamamen patentsiz olarak kullanılabilen CTR-MAC ise dizi şifreleme sisteminin zayıflıklarına sahiptir, özellikle aynı kayan anahtar dizisi kullanımı konusunda dikkatli olunmalıdır.
- d. Bölüm 3.5.4. de anlatıldığı üzere, IV'yi kötüye kullanma saldırısı, IV'yi şifreleme işleminden sonra aktif olarak kullanmaya başladığından dolayı OCB çalışma kipine uygulanamamaktadır. CTR çalışma kipinde ise yine dizi şifreleme sistemlerinden kaynaklanan bir zayıflık mevcuttur.
- e. Mesaj bütünlüğü koruma yöntemi çok önemlidir ve güvenlik protokollerinde mutlaka kullanılmalıdır. OCB çalışma kipi bu yönden de avantaj sağlamaktadır.
- f. Ayrıca OCB çalışma kipi mesaj gizliliğinin yanısıra mesaj bütünlüğünü nerede ise masrafsız sağlamaktadır. Çizelge 3.2. den de görülebileceği gibi OCB çalışma kipi önemli hız avantajı sağlamaktadır.

Yukarıda anlatılan nedenlerden dolayı OCB çalışma kipi önemli avantajlar sağlamaktadır. Fakat OCB çalışma kipi patentli bir çalışma kipidir. Bu nedenden dolayı OCB çalışma kipi kadar avantaj sağlayamayan ve dizi şifreleme sistemlerinin zayıflıklarına sahip olan fakat tamamen patentsiz olan CTR-MAC dikkatli kullanılması şartı ile OCB çalışma kipine alternatif olmaktadır.

3.7. Çalışma Kiplerinden Bazılarının Performans Karşılaştırılması

Bu bölümde AES-OCB, AES-CBC ve AES-CTR şifreleme işlemleri performans açısından karşılaştırılacaktır. OCB çalışma kipi gizliliği ve bütünlüğü beraber sağlamaktadır, CBC ve CTR çalışma kipleri ise sadece gizliliği sağlamaktadırlar. Anlamlı bir karşılaştırma olması amacı ile CBC ve CTR çalışma kipleri ile yapılan şifreleme işlemine, bütünlüğü sağlaması için CBC-MAC değeri eklenecektir.

Şifreleme algoritmasının CBC çalışma kipinde çalıştırılarak elde edilen son bloğu, özet değer yani CBC-MAC olarak adlandırılır. Bu işlemdeki anahtar bilgisi, farklı bir anahtar olabileceği gibi, şifreleme anahtarından da elde edilebilmektedir.

Değerlendirme işlemi Pentium III 700 Mhz işlemci ve 128 MB RAM donanımlarına sahip bir bilgisayarda, Borland C++ 5.5.1 derleyicisi kullanılarak gerçekleştirilmiştir ve sonuçta 1 KByte açık mesaj şifrelenmiştir.

Çizelge 3.2.'de gösterildiği gibi AES-OCB şifrelemesi, AES-CBC ve AES-CTR şifrelemelerinden sadece iki algoritma çağırımı fazla iş gerektirmekle birlikte ekstra olarak anahtar hazırlığı yapılması gerekmektedir, bu işlem de çok hızlıdır. Yazılım olarak deneysel elde edilen sonuçlar da bunu desteklemektedir. Deneysel sonuçlarda elde edilen değerler birbirine çok yakın olmakla beraber, AES-OCB, diğerlerinden ihmal edilebilir şekilde çok az yavaştır. Buna rağmen OCB, hem gizliliği hem de bütünlüğü sağlayan bir çalışma kipidir. Karşılaştırmanın manalı olabilmesi için diğer çalışma kiplerine CBC-MAC bütünlük değeri eklenerek karşılaştırılması gerekmektedir. Bu durumda AES-OCB, AES-CBC ve AES-CTR şifrelemelerinden neredeyse iki kat daha hızlı olmaktadır.

AES-OCB şifrelemesinde bir adet şifreleme anahtarı kullanılmasına karşın, AES-CBC ve AES-CTR şifrelemelerinde CBC-MAC değeri için ayrı bir anahtar kullanılırsa iki şifreleme anahtarına ihtiyaç duyulacaktır.

Çalışma kipleri	AES-OCB	AES-CBC	AES-CTR	AES CBC-MAC
Yazılım hızı	~1 birim (birden büyük)	1 birim	1 birim	1 birim
Şifreleme algoritma kullanım sayısı	$\lceil M /n \rceil + 2$	$\lceil M /n \rceil$	$\lceil M /n \rceil$	$\lceil M /n \rceil$
Anahtar hazırlığı	var	Yok	Yok	yok
Kullandığı anahtar sayısı	1 tane	1 tane	1 tane	1 tane, şifreleme anahtarından elde edilebilir
Şifreleme algoritması	E, E ⁻¹	E, E ⁻¹	E	E
IV kullanımı	Aynı anahtarla en çok bir defa kullanılan rasgele sayı	Aynı anahtarla en çok bir defa kullanılan rasgele sayı	Aynı anahtarla en çok bir defa kullanılan sayaç	Aynı anahtarla en çok bir defa kullanılan rasgele sayı

Çizelge 3.2.: Çalışma Kiplerinin Performans Karşılaştırması.

4. ÖRNEK GÜVENLİK PROTOKOLÜ ANALİZİ (IPSEC, SSL VE WEP ANALİZİ)



RC4 algoritması güvenli olarak kullanılabilen etkili bir dizi şifreleme algoritmasıdır. Örneğin SSL protokolü içinde gerçekleşen RC4 algoritması [Fluhrer, et al, 2001]'da anlatılan saldırı yönteminden etkilenmemektedir. Yinede RC4 kullanıcılarının dikkatli olması gereken bazı hususlar mevcuttur. Bu hususlara dikkat edilmediği takdirde WEP protokolündeki RC4 gerçeklemesine karşı uygulanabilen [Fluhrer, et al, 2001] saldırı yöntemleri ile protokol güvensiz hale gelebilmektedir.

SSL protokolündeki RC4 gerçeklemesinin bu saldırı yönteminden etkilenmemesinin bir nedeni, IV ve anahtar kullanılmadan önce MD5 ve SHA1 özetleme algoritmalarının ikisinden birden geçirilmesidir. Böylelikle, farklı oturumlar bağımsız anahtarlara sahip olmaktadır ve algoritma her paketten sonra tekrar anahtarlama gerektirmemektedir.

[Walker, 2000]'de bu saldırı yönteminin sadece RC4 değil bütün dizi şifreleme sistemlerini kullanan protokollere karşı uygulanabileceğinden bahsedilmektedir. Dizi şifreleme algoritması kullanan bazı protokollere karşı uygulanamamasının nedeni, bu saldırı yönteminin söz konusu protokoller için çok verimsiz olmasıdır. Bu saldırı yönteminin diğer protokollerde değil de WEP protokolünde etkili olması ise başka gerçeklemelerde dizi şifreleme algoritmasının kullanım şeklinin WEP protokolündeki gibi olmamasından kaynaklanmaktadır. Örneğin, SSL protokolünde şifreleme işleminde kullanılan anahtarlar çok önemsiz bir ihtimal ile birbirlerine bağlı olacak şekilde tasarlanmıştır. SSL protokolünde bütün bir oturum için 128 ikilik bir ilk vektör seçilmektedir ve anahtar her paketten sonra değişikliğe uğratılmaktadır. SSL protokolünün bu şekilde gerçekleştirilebilmesi, gönderilen bilginin hatasız olarak alıcı tarafa ulaştırılabildiği bir ortamda kullanılmasıyla kaynaklanmaktadır. Böylelikle, şifreleyici ve şifre çözücü arasında senkronizasyon kaybı olmayacağı garanti edilebilmektedir.



Ayrıca SSL protokolü, anahtarını değişimden geçireceği zaman WEP protokolündeki gibi sadece 24 ikilik bir ilk vektörle değil anahtar uzunluğu ile aynı (bu değer genelde 128 ikidir) uzunlukta ilk vektör kullanmaktadır. Bu durumda saldırgan, WEP protokolündeki gibi bir avantaj sağlayabilmek için 2^{12} şifreli mesaj dizisini değil de SSL protokolü ile şifrelenmiş 2^{64} şifreli mesaj dizisini elde etmek zorunda kalacaktır. Buda demek oluyor ki SSL, aynı dizi şifreleme algoritmasını kullanmasına karşın saldırgana 2^{52} yani 4,000,000,000,000,000 kere daha fazla masrafa mal olmaktadır.

Bu karşılaştırmanın anlamlı olabilmesi için, SSL protokolü ile WEP protokolünün bir çok açıdan ciddi farklılıklar gösterdiği belirtilmelidir. Bu noktada, IPSec protokolündeki kapsül içine alma işlemi sırasında karşılaşılan problemlerin 802.11 deki problemlere benzemesi sebebiyle, bu protokolün aynı anahtar kullanımını önlemek için aldığı tedbirlerden bahsedilecektir.

1. Sanal bağlantılar üzerindeki her bir oturumun her yönü için benzersiz bir anahtar kullanmaktadır. 3DES şifreleme işlemindeki varsayılan şifreleme algoritmasıdır. Bu algoritmanın efektif anahtar gücü 112 ikidir. Gerçek anahtar uzunluğu 168 ikidir, fakat 3DES yapısı yüzünden ortadaki kişi saldırısı ile bu anahtar uzayı 112 ikile daraltılmaktadır. Demek oluyor ki, iki rasgele oluşturulmuş anahtarın ciddi bir olasılıkla aynı olması için $2^{56}/2 = 2^{55}$ oturum geçmesi gerekmektedir. Saldırganın böyle bir durumu ümit edebilmesi için gerçekten çok büyük miktardaki şifreli mesajları veri tabanında saklayabilmesi gerekmektedir. Ayrıca, 2^{55} şifreli mesajı ele geçirebilmesi demek, her saniyede 2^{36} yeni oturum oluşturan bir sistemde yirmi yıldan fazla bütün şifreli mesajları kayıt yapması anlamına gelmektedir. Bu durumun pratikte, günümüz teknolojisi ile gerçekleştirilmesi neredeyse imkansızdır.
2. IPSec protokolü 3DES blok şifreleme algoritmasını kullanırken, her bir oturum 64 ikil uzunluğundaki kendi IV'lerini kullanmaktadır. Eğer AES kullanılacak ise bu değer 128 ikil olmalıdır. Yani 3DES blok

algoritması için 2^{32} paketten önce ciddi bir oranla aynı IV kullanımı beklenilmemelidir. IPSec protokolü böyle büyük bir IV uzayı kullanarak, IV tekrar kullanımından kaynaklanan problemleri de çözmektedir. Ancak saldırgan her anahtar için veri tabanında 2^{32} şifreli mesaj üreterek saklayabilmekte ve bu olasılığı arttırabilmektedir. Fakat bu işlem de günümüz teknolojisiyle ve pratikte neredeyse imkansızdır.

3. IPSec her anahtar için sadece 2^{32} paket şifrlenmesine izin vermektedir. Saldırganın her nasılsa yeteri kadar çok şifreli mesaj topladığı varsayılsa bile her anahtar için olasılıklar birbirine eklenmektedir. Bu da saldırganın işini zorlaştıracaktır.
4. IPSec protokolü CBC çalışma kipini kullanmaktadır. CBC çalışma kipi ise IV'yi anahtarı değişimden geçirmek için değil, bir nevi geri besleme olarak kullanmaktadır. Bu besleme herkes tarafından bilinen ve aynı anahtar ile şifrelenen mesajları farklılaştırmak için kullanılmaktadır. Böylelikle mesajlar içindeki örüntüler şifreli mesajdan ayırt edilemez hale gelmektedir. CBC çalışma kipinde ki IV tekrar kullanımı, WEP protokolündeki IV kullanımı gibi çok kötü sonuçlara neden olmamaktadır. Buna rağmen, CBC çalışma kipinin de bölüm (2.4.)'de anlatılan bazı zayıflıkları vardır.

WEP protokolündeki kapsül içine alma işlemi, bağlantı başına farklı anahtar ile yapılırsa bile WEP protokolündeki problemler yine devam edecektir. Aynı zamanda, WEP protokolünde IV tekrar kullanımının çok az bir olasılık ile gerçekleşmesini ve IV'nin tekrar kullanımının tehlikeye girdiği noktada taban anahtarın değiştirilmesini sağlamak da gerekecektir. Bu söylenirken WEP protokolünde başka ciddi problemlerin olmadığı farz edilmektedir.

WEP protokolünde her bir uç farklı bir anahtar kullanıyor ve birbirlerinin anahtarını bir şekilde biliyorlar ise, düzeltilmiş WEP belki de tekrar kullanımını önlemek amacı ile IV'yi rasgele üretmek yerine bütün halkayı bitirene kadar bir kurala bağlı olarak (mesela doğrusal modüler üreteç veya yine sayaç şeklinde) kullanabilir. Güvenlik açısından bakıldığında böyle bir gerçeğin tamamen rasgele bir IV kullanılmadığı zaman güvenli olup olmayacağına dair bir delil bulunmamaktadır. Ayrıca kullanılan algoritma RC4 dizi şifreleme algoritması olduğu zaman, IV'nin kurallı kullanımı saldırgan açısından zayıf anahtar saldırısı uygulamasına imkan verebilmektedir. Bu saldırılara bölüm (4.1.)'de daha geniş yer verilecektir.

WEP protokolündeki problemler, özel olarak tasarlanan bu protokolün tasarımcıları tarafından, literatürde mevcut saldırı yöntemlerinin bu protokolde açık bir şekilde gözükmemesinden ve açık bir tehdit oluşturmamasından kaynaklanmaktadır. Zamanla bu protokolün tasarlanan güvenlik hedeflerine ulaşamadığı anlaşıldı. Genel bir kural olarak, böyle özel bir şekilde tasarlanan protokollerin akademik çevreler tarafından test edilip güvenli oldukları kanıtlanmadan kullanılmamaları gerektiği anlaşılmıştır. IPSec protokolü gibi birçok güvenlik protokolü, ilk çıktıkları zamandan itibaren bulunan zayıflıkları giderilip üzerlerinde birçok değişiklikler yapıldıktan sonra kabul edilir hale gelmişleridir. IPSec, SSL ve WEP protokollerinin karşılaştırması Çizelge 4.1.'de verilmiştir.

Protokol	SSL	WEP	IPSEC	Yorumlar
Şifreleme Algoritması	RC4	RC4	Herhangi bir blok şifreleme algoritması kullanılabilir.	Blok şifreleme algoritmaları tercih edilmelidir. AES blok şifreleme algoritması iyi bir seçimdir.
IV Uzunluğu	128	24	Algoritmanın blok uzunluğu kadar (AES için en az 128 ikil)	En az 128 ikil uzunluğunda olmalıdır. Rasgele üretilmesi ve sayaç tabanlı olmaması tercih sebebidir. Aynı anahtar ile birden fazla kullanılmamalıdır.
Kimlik Doğrulama ve Bütünlük Kontrolü	SSL3.0'da HMAC-MD5 HMAC-SHA-1 kullanılıyor. SSL2.0'da ise güvensiz MAC kullanılmıştı.	CRC-32 ikil	Hem ESP hem de AH'da HMAC-MD5 HMAC-SHA-1 HMAC-RIPEMD-160	Anahtarlı özetleme olan, HMAC, veya bütünlüğü de koruyan OCB gibi bir çalışma kipi tercih edilmelidir. Özet minimum 128 ikil olmalıdır.
İnkâr Edememe	Desteklemiyor	Desteklemiyor	Sadece Anahtar yönetiminde destekleniyor.	Sadece açık anahtarlı şifreleme sistemleri ile mümkündür.
Anahtar Yönetimi	Standart bir anahtar yönetimi mevcuttur (El sıkışma protokolü)	Standart bir anahtar yönetimi yoktur.	Standart bir anahtar yönetimi mevcuttur. (IKE)	Protokol ile birlikte standart bir anahtar yönetimi sağlanmalıdır.
Saldırıları	RC4'dan kaynaklanan bir zayıflık yoktur.	RC4'dan ve IV kullanımından kaynaklanan bazı zayıflık mevcuttur.	Bütünlük kontrolü yapılmadığı durumlarda çalışma kipinden kaynaklanan problemler düzeltilmiştir.	Eski DES çalışma kiplerinin bazı zayıflıkları vardır. OCB gibi yeni AES çalışma kipleri tercih edilmelidir.

Çizelge 4.1.: İncelenen Protokollerin Karşılaştırması.

Akla gelebilecek bir diğer soru da açık mesajın önüne, bir 'önleyici rasgele dizi' eklemenin WEP kapsüle alma işlemindeki güvenliğini artırıp arttırmayacağıdır. Önleyici rasgele dizi, açık mesajların önüne eklenir ve şifrelendiği zaman sonrasında gelen açık mesajların ilintisini azaltarak IV gibi çalışır. Dolayısıyla cevap da hayırdır. Çünkü, anahtar dizisi üretici RC4 dizi şifreleme algoritmasıdır ve bu algorithmada önceki verilerin sonrakileri etkileyecek geri besleme mekanizması bulunmamaktadır. Bu nedenle önleyici rasgele dizi, anahtar dizisini değiştirmemektedir. Önleyici rasgele dizi, daha çok blok şifreleme algoritmalarının CBC, OFB ve CFB gibi geri beslemeli çalışma kiplerinde etkili olan bir mekanizmadır.

4.1. RC4 Algoritması ile İlgili Problemler

[Fluhrer, et al, 2001]'de RC4 dizi şifreleme algoritmasına yapılan ciddi bir saldırı yönteminden bahsedilmektedir. Bu makalede RC4 dizi şifreleme algoritmasının anahtar-zamanlama algoritmasında bazı zayıflıklar olduğundan bahsedilmektedir. Büyük miktardaki bir şifreli mesaj bilgisi kullanılarak şifreleme işleminde kullanılan anahtar bilgisine ulaşılabilir. WEP protokolündeki gönderilen şifreli mesaj kullanılarak anahtar bilgisine ulaşmak için ağdan büyük miktarda şifreli mesaj paketi toplamak gerekmektedir. Başarılı saldırıların ağdan toplanan 1.000.000 paket ile yapılabildiği rapor edilmiştir.

Bu probleme en kolay çözüm şifreleme işleminde kullanılan bir anahtarın sadece sınırlı sayıda açık mesajın şifrelenme işleminde kullanılmasıdır. Aslında WLAN (Wireless Local Area Network) sağlayıcıları tarafından önerilen çözüm, karmaşık kimlik doğrulama ve anahtar değişim teknikleri içermektedir. Bununla birlikte kullanılan anahtarın çok sık değiştirilmesi de önerilmektedir. Bu şekildeki çözümler ile sadece anahtarın belli bir miktar açık mesajdan fazlasının şifrelenmesi önlenmektedir. Bölüm (2.4.2.)'de bahsedilen WEP protokolünden gelen zayıflıklar hala giderilmemiştir.



Aslında bu anlatılan ve RC4 dizi şifreleme algoritmasının WEP protokolünde kullanımının kötü bir seçim olduğunu gösteren zayıflıklar birçok dizi şifreleme sistemine uygulanabilmektedir. Aşağıda RC4 dizi şifreleme sisteminin diğer bazı zayıflıklarından bahsedilecektir.

[Roos, 1995]' da anlatıldığı gibi RC4 dizi şifreleme algoritmasının her 256 anahtarından biri zayıf anahtardır. Demek oluyor ki anahtar-zamanlama algoritması böyle anahtarları olması gerektiği kadar anahtar ile bağlantılı olarak karıştıramamaktadır. Böylelikle bu anahtarlar ile şifrelenmiş mesajların kriptoloji çok daha kolay hale gelmektedir. WEP protokolünde RC4 dizi şifreleme algoritması kullanılmaya devam edilirse bununla beraber bu probleme standart bir çözüm WEP standardı ile birlikte verilmelidir.

Ciddi olmayan ama not edilmesi gereken son bir problem de şudur ki; RC4 dizi şifreleme algoritmasının ürettiği anahtar dizisi, fark edilebilen sapmalar içermektedir. Yani belirli bir uzunlukta şifreli bir mesajın RC4 ile şifrelenip şifrelenmediği anlaşılmaktadır. Fakat bu problem WEP veya başka bir protokolda etkili bir saldırı yöntemi olamamıştır.

4.2. WEP Protokolündeki Problemlere Bazı Çözüm Önerileri

WEP algoritmasının anahtar uzunluğunu 40 ikiden 112 veya 128 ikile çıkarmak, yapılan bu saldırıları önlemeyecektir. Çünkü, WEP protokolünün zayıflıkları kullandığı anahtar uzunluğu ile değil, WEP protokolünün kriptolojiyi nasıl kullandığı ile ilgilidir. WEP protokolündeki RC4 algoritmasının kullanıldığı çevre koşulları ele alındığı zaman, kötü bir seçenek olduğu anlaşılmaktadır. Durum böyle olunca kabloluya denk gizlilik protokolünün hedeflediği servisleri gerçekleştirebilmesi için WEP kapsüle alma işleminin ciddi bir şekilde yeniden tasarlanması gerekmektedir.



4.2.1. Şifreleme sistemi ve Çalışma kipi ile ilgili öneriler

NIST'in belirlediği AES algoritması büyük bir çabanın ürünüdür ve her türlü seçim işlemi de literatüre açık şekilde gerçekleştirilmiştir. Bu algoritmanın bazı zayıflıkları bulunmasına karşın en az literatürdeki diğer algoritmalar kadar güçlü olduğu kabul edilmektedir. AES, 8-ikil işlemcilerden süper bilgisayarlara kadar çeşitli çevrelerde verimli bir şekilde gerçekleştirilebilmektedir. WEP protokolündeki RC4 kullanımının sakıncaları sebebiyle yeni tasarlanan WEP protokolünde en az 128 ikil AES kullanımının zorunlu şifreleme algoritması olarak seçilmesi tavsiye edilmektedir.

Bu öneride ayrıca WEP protokolünün AES ile beraber OCB çalışma kipinde kullanılması da tavsiye edilmektedir. Bölüm (2.4.1.)'de anlatılan bu çalışma kipinde gizliliğin yanında ayrıca saldırganın mesaj üzerinde değişiklik yapmasını önleyen bir mekanizma olan mesaj bütünlüğü de otomatik olarak sağlanmaktadır.

Sevindirici bir haber olarak 802.11i standardı AES-OCB şifreleme algoritma ve çalışma kipini mesaj gizliliği ve bütünlüğü için standart olarak kabul edilmiştir.

4.2.2. Oturum Anahtarı Oluşturulması

[Walker, 2000]'da anlatılan oturum anahtarı oluşturma algoritmasının geliştirilmesi gerekmektedir. IPSec ve SSL protokollerindeki fikirler kullanılabilir.

4.2.3. Tavsiye edilen Kapsüle Alma Yöntemi

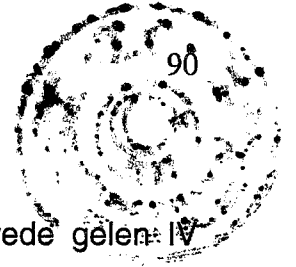
Yine [Walker, 2000]'da kapsüle alma işlemi için aşağıdaki yöntem önerilmektedir.



1. Şifreleme işlemini gerçekleştiren taraf 128 ikil uzunluğundaki IV'i rasgele olarak üretir. Sonraki şifrelenecek çerçeveler için son şifre blok yeni IV olarak kullanılacaktır.
2. 32 ikillik bir dizi numarası kullanılacak ve bu değer ilk şifrelenecek blok içinde yer olacaktır. Bu numara kullanılmakta olan anahtarın kaç tane çerçevenin şifrlenmesinde kullanıldığını göstermektedir. El ile yüklenen anahtarlar kullanılacağı zaman bu alan kullanılmayacak ve sifira sabitlenecektir. Dinamik anahtarlar kullanılacağı zaman bu değer bir den başlayıp sayaç gibi arttırılacaktır.
3. Bu öneride LLC (Logical Link Control) verisinin de şifrlenmesi gerekmektedir.
4. Mesaj bütünlüğünün korunması için 128 ikil uzunluğunda OCB veri etiketi kullanılacaktır.

Dolayısı ile kapsüle alma işlemi esnasında 36 sekiz ikil uzunluğunda veri çerçeveye eklenmiş olacaktır. Bu büyüklükteki veri, güvenlik düşünüldüğünde göze alınmaya değer bir miktardır.

Şifreleme işlemi yapan taraf ilk önce sayacın belirlenmiş maksimum değere ulaşip ulaşmadığını kontrol edecektir. Eğer standartta belirlenmiş bu değere ulaşmış ise yeni bir anahtar oluşturulması gerekmektedir. Bu amaç ile yeniden anahtar oluşturma protokolü işletilmeli veya önceden anlaşılmış bir diğer anahtara geçilmelidir ve sayaç başa döndürülmelidir. Eğer sayaç belirlenen değerden az ise şifrelemeye o anki anahtar ile devam edilebilecektir. Şifreleme işleminden sonra bu sayaç değeri bir arttırılacaktır. Daha sonra OCB çalışma kipinde şifreleme için rasgele IV seçilmeli ve etiket ve mesaj şifrlenmelidir. Bu şifrelemenin sonucunda şifreli mesaj ve mesaj doğrulama etiketi ortaya çıkacaktır. Şimdi çerçeve gönderilmeye hazır hale gelmiştir.



Alınan şifreli mesajın şifre çözme işleminde ise çerçevede gelen IV kullanılarak OCB çalışma kipinde şifre çözme işlemi gerçekleştirilir ve mesaj doğruluğunu kontrol eder. Eğer oturum dinamik anahtar kullanıyor ise son olarak da sayaç bilgisinin daha önce aldığı çerçevedekinden büyük olduğunu kontrol edilecektir. Yani alıcı taraf pencere boyunun bir olup olmadığını kontrol edecektir. Bunun amacı tekrar gönderme saldırılarını önlemektir. Tekrar gönderme saldırısının önlenmesi el ile yapılan anahtarlama mümkün değildir dinamik anahtarlama mümkündür.



5. SONUÇLAR/ÖNERİLER

Güvenlik protokolü tasarılmanın ne kadar zor bir iş olduğu ve birçok hatayı ve düzeltmeyi ardından getirdiği kriptoloji ile ilgili olanlar tarafından iyi bilinen bir gerçektir. Güvenlik protokollerinin tasarımının yapılabilmesi için ağ protokolleri deneyimli olmak yetersizdir, bunun haricinde birçok özel deneyiminde sahip olunması gerekmektedir.

Bazı güvenlik protokollerinde güvenlik için gerekli bazı noktalar açık bırakılmaktadır, örneğin WEP protokolünün anahtar yönetimi gibi, buda beraberinde gerçeklemeden gerçeklemeye değişiklikler getirmektedir. Dolayısı ile üründen ürüne güvenlik farklılıklar göstermektedir. Diğer taraftan, eğer standart bütün her şeyi çok detaylı bir şekilde belirler ise, sonuçtaki standardı gerçeklemek çok zor olabilir veya standart esnek olmayabilir. Örnek olarak çok detaylandırılmış bir standart olan X.400 standardının hiçbir zaman tamamı gerçekleştirilmemiş sadece alt kümeleri gerçekleştirilmiştir.

Tasarımcılar açık literatürdeki mümkün olan en iyi algoritmayı kullanmalılardır. Bugün açık literatürde bilinen güçlü şifreleme algoritması mevcuttur. Bazılarının lisansı vardır ama çoğuna lisansız olarak ulaşılabilip kullanılabilir. Dolayısı ile tasarımcıların, bölüm (2.4.4.)'de de anlatıldığı gibi Microsoft'un NT parolaları için yaptığı gibi bilinmeyen veya güvensiz algoritmaları tercih etmemeleri gerekmektedir.

Bölüm (2.4.) ve [Bellovin, 1996]'de anlatıldığı gibi birçok saldırı yöntemine karşı en etkili korunma yöntemi kriptolojik bütünlük kontrolü yapmaktır. Bütünlük koruması yapmadan sadece şifreleme yapmak gerçekte manasız olmaktadır ve bölüm (2.4.)'de de anlatıldığı gibi birçok saldırıya maruz kalmaktadır. Hatta WEP protokolündeki gibi güçsüz özet kullanımı bile birçok saldırıya maruz kalmaktadır. Bundan dolayı gönderilen her mesajın güçlü kriptolojik teknikler kullanılarak bütünlüğünün sağlanması çok önemlidir. Eğer mesajın düzgün bir şekilde bütünlüğü korunmuş ise

kesilip yerine başka bir şey konulamaz veya değiştirilemez. Güncel protokollerin bütünlük korunması için ayrı bir mekanizmaları yoktur. Fakat, kimlik doğrulama teknikleri mesajın bütünlüğünü de korumaktadır. IPSec ve benzeri protokoller için genelde bağlantı başına anahtar kullanılması önerilmektedir.

Bellovin makalesinde ayrıca, seçilebilen açık mesaj saldırılarından korunmak için çok basit bir güvenlik politikası önermektedir, hiçbir zaman açık mesaja şifrelisi ile karşılık verilmemelidir.

İkinci genel bir korunma tekniği, bir anahtarın tek bir bağlantı için kullanılmasıdır. Yeni bir bağlantı için yeni bir anahtar seçilmelidir. Böyle bir durumda saldırgan, bağlantılar arasında kes-yapıştır saldırısını gerçekleştiremeyecektir. Eğer bu mümkün değilse anahtarlar makul aralıklar ile değiştirilmelidir. Özellikle dizi şifreleme algoritmalarında, belli miktar zaman geçmişse, alınan veri miktarına belli bir sınırı geçmişse ve eğer alınan dizi numaraları arasında büyük fark varsa , kullanılan anahtar mutlaka değiştirilmelidir. Geri gönderme saldırılarına da önlem alınması gerekir. Eğer bağlantı başına anahtar kullanılmıyorsa, paket kimlik doğrulama tekniğinin geçerli bir paketin saldırgan tarafından tekrar reddetmesini engelleyecek mekanizması yoktur.

[Schneier and Wagner, 1997]'de de ifade edildiği gibi SSL protokolünün el sıkışma kısmında bazı problemler vardır. Bu problemler sadece aktif bir saldırı olduğunda önem kazanmaktadır. Ayrıca bu zayıflıklar bütün gerçeklemede görünmemektedir, gerçeklemeden gerçeklemeye farklılıklar göstermektedir. Protokoldeki bir hata her zaman gerçeklemenin tehlike altında olduğu anlamına gelmemektedir. Yinede, eğer bir standart protokolüne muhtemel yapılabilecek saldırıları açıkça uyarmıyorsa, veya en iyisi bunları engellemiyorsa, bu protokol yoruma çok açık demektir. Çünkü, bu protokolü gerçekleyenlerin saldırıları her zaman öngörmesi ve önlem alması beklenemez.



[Schneier and Wagner, 1997]'da yapılan analizler SSL 3.0 protokolündeki birkaç aktif saldırı yöntemini açığa çıkarmıştır. Bu saldırılar bütün gerçeklemeler için geçerli değildir. En önemli yeni saldırı yöntemleri sistem-belirtim-düşürme ve anahtar değişim algoritması aldatma saldırılarıdır. Bunların ilkinde saldırgan protokolün ilk adımlarındaki SSL'in hangi versiyonunun kullanılacağını belirten belirtimi değiştirerek zayıf versiyon, mesela ihraç zayıflatılmasına uğramış veya eski sürüm kullanımını sağlaması gibi, kullanımını sağlayabilmektedir. Neyse ki bu saldırıların düzeltilmesi zor değildir.

SSL 2.0'nin kayıt katmanındaki anahtar değişim protokolüne de birçok aktif saldırı yapılabilmektedir. SSL 3.0 bu hataları düzeltmiştir, dolayısı ile çok daha güvenli bir protokoldür. SSL 3.0, ihraç zayıflatılmasına uğramış sürümünde de daha iyi bir mesaj bütünlüğü koruması sağlamaktadır. SSL 2.0' da ise 40 ikiden oluşan güvensiz bir mesaj doğrulama kodu kullanılmaktaydı. SSL 3.0 ise her zaman 128 ikilik HMAC kullanmaktadır. Genel olarak bakıldığında zaman SSL 3.0, SSL 2.0'a göre oldukça daha güvenli bir protokoldür ve çok önemli olmayan bazı gelişmelere açık olmakla beraber genelde SSL 3.0'ın pratik haberleşme güvenliğine kazandırılmış önemli güvenlik bir protokolü olarak bakılmaktadır.

[Borisov et al, 2001]'de protokol tasarımı ve analizi işinde protokollerde kullanılan kriptolojik temellerin ve özelliklerinin çok iyi anlaşılmış olmasının öneminden bahsedilmektedir. Örneğin, sadece mühendislik yönü ile olaya bakarsan CRC-32 ve RC4'un kullanımı hız ve gerçekleştirme kolaylığı açısından makul gözükmemektedir. Fakat bu makalede anlatılan saldırı yöntemlerinin çoğu dizi şifreleme sistemlerinin zayıflıklarından ve CRC-32'nin mesaj bütünlüğünü korumada yetersiz kalmasından kaynaklanmaktadır. Diğer algoritmalar örneğin AES ve OCB çalışma kipi seçilmiş olsaydı, başka bir iyileştirme yapmadan bile en azından bu makalede anlatılan saldırı yöntemlerinin uygulanması çok daha zor hale gelecekti. Tabi ki kriptoloji ile direk alakalı olmayan daha ince mühendislik etkileşimleri de vardır. Örneğin, bir protokolün kabul edeceği durumsuz ve geniş görüşlü olmak gibi kavramlar ağ

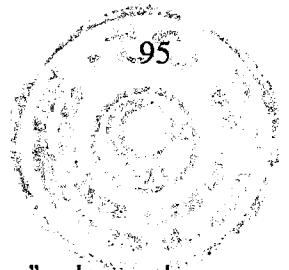


mühendisliğinde iyi yerleşmiş kurallardır. Fakat güvenlik açısından olaya baktığımızda saldırgana daha fazla hareket ve imkan olanağı verdiğimizden bunların ikisi de tehlikelidir. Güvenlik kavramı bütün sistemin bir özelliğidir ve alınan her bir karar güvenlik düşünülerek gözden geçirilmelidir.

Bu tip güçlüklerle karşılaşan birçok güvenlik uzmanı çok ciddi hatalar yapabilmektedir. Bu gerçek anlaşıldıktan sonra, önerilen yol başkalarının deneyimlerinden yararlanmak, daha önce yapılmış hatalardan ders almaktır ve yapılan protokolün açık literatürde herkesin yorumuna sunulmasıdır. Eski ve güvelliği test edilmiş tasarımların mümkün olduğunca kullanılması gerekmektedir. Örneğin, WEP protokolü, IPsec ve Microsoft PPTP protokollerinden kazanılan deneyimleri kullanmış olsaydı bu hatalara düşmeyecekti. IPsec ve Microsoft PPTP protokollerinin amaçları farklı olmasına karşın, IPsec protokolü de bağlantı katmanı güvenlik sağlamaktadır ve bundan dolayı WEP protokolü ile ortak birçok uğraşım alanları vardır. Protokolden alıntı yapılmasa bile gelişiminde kazanılan deneyimlerden yararlanmak çok önemlidir. IPsec ve Microsoft'un daha önce yayınlanmış problemlerin bir çoğu WEP protokolünün problemleri ile benzerlik göstermektedir.

Son olarak da AES ve bu algoritma ile beraber gelen yeni çalışma kiplerinin, özellikle OCB gibi gizliliği ve bütünlüğü beraber sağlayan çalışma kiplerinin kullanımının teşvik edilmesi gerekmektedir. NIST'in yeni belirleyeceği AES çalışma kiplerinden en az birinin bu tip bir çalışma kipi olması umulmaktadır.

KAYNAKLAR



[1] Meadows, C., "The NRL Protocol Analyzer: An Overview", Journal Logic Programming, 1994:19, 20:1-679.

[2] Bellare S. M., "Problem Areas for the IP Security Protocols", Proceedings of the Sixth USENIX Security Symposium, Usenix Association, 1996, pp. 205-214. <ftp://ftp.research.att.com/dist/smb/bades.ps>.

[3] Schneier B., Wagner D., "Analysis of the SSL 3.0 protocol", The Second USENIX Workshop on Electronic Commerce Proceedings, USENIX Press, November 1996, pp. 29-40. Revised on April 15, 1997.

[4] Borisov N., Goldberg I., Wagner D., "Intercepting Mobile Communications: The Insecurity of 802.11", Proceedings of the Seventh Annual International Conference on Mobile Computing and Networking, July 16-22, 2001.

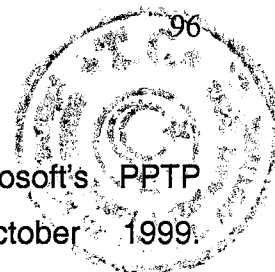
[5] Schneier, B., and Mudge, "Cryptanalysis of Microsoft's Point-to-Point Tunneling Protocol (PPTP)", 5th ACM Conference on Computer and Communications Security, pp. 132-140, Nov, 1998. <http://www.counterpane.com/pptp.pdf>

[6] Menezes A. J., van Oorschot P. C., Vanstone S. A., "Handbook of Applied Cryptography", CRC Press, 1997.

[7] Schneier B., "Applied Cryptography: Protocols, Algorithms and Source Code in C", John Wiley and Sons, Inc., NY, USA, second edition, 1996.

[8] Mel, H. X., Baker D., "Cryptography Decrypted", Addison Wesley, 2001.

[9] Dworkin, M., "Recommendation for Block Cipher Modes of Operation", National Institute of Standards and Technology (NIST), December 2001.



[10] Schneier, B., and Mudge, "Cryptanalysis of Microsoft's PPTP Authentication Extensions", Counterpane System, October 1999.
<http://www.counterpane.com/pptpv2.pdf>

[11] David K., "The Code Breakers", 1996

[12] Dawson, E., and Nielsen, L., "Automated cryptanalysis of XOR plaintext strings", *Cryptologia*, (2): 165–181, April 1996.

[13] Tutte, W. F., "A transcript of Tutte's lecture at the University of Waterloo", June 19, 1998.

[14] Ziemba, P., Reed, D., and Traina, P., "Security considerations for IP fragment filtering, Request for Comments (Informational) RFC 1858, Internet Engineering Task Force", October 1995.

[15] Petteri, L., "NT Server Security", Helsinki University of Technology,
http://www.tcm.hut.fi/Opinnot/Tik-110.501/1997/nt_server.html

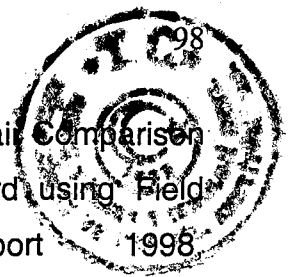
[16] Meadows, C., "Invariant Generation Techniques in Cryptographic Protocol Analysis", *Proceedings of the 13th Computer Security Foundation Workshop*, IEEE Computer Society Press, July 2000.

[17] Meadows, C., "Analysis of the Internet Key Exchange Protocol Using the NRL Protocol Analyzer", *Proceedings of the 1999 IEEE Symposium on Security and Privacy*, IEEE Computer Society Press, May 1999.

[18] Atkinson, R., and Kent, S., "Security architecture for the Internet Protocol. Internet Request for Comment RFC 2401", Internet Engineering Task Force, Nov. 1998.



- [19] Fluhre, S., Martin, I., and Shamir, A., "Weaknesses in the Key Scheduling Algorithm of RC4", Eight Annual Workshop on Selected Areas in Cryptography, August 2001.
- [20] Walker, J., "Unsafe at any key size: an analysis of the WEP encapsulation," Tech. Rep. 03628E, IEEE 802.11 committee, March 2000.
<http://grouper.ieee.org/groups/802/11/Documents/DocumentHolder/0-362.zip>.
- [21] Roos, A., "A Class of Weak Keys in the RC4 Stream Cipher", Vironix software Laboratories, Westville, South Africa, September 22, 1995.
<http://www.itribe.net/CTRS/Papers/CTRS-0008.txt>
- [22] Rogaway, P., Bellare, M., Black, J., Krovetz, T., "OCB: A Block-Cipher Mode of Operation for Efficient Authenticated Encryption", Eighth ACM Conference on Computer and Communications Security (CCS-8), ACM Press, pp. 196-205, 2001. <http://csrc.nist.gov/encryption/aes/>
- [23] Arbaugh, W. A., Shankar, N., and Wan, Y. J., "Your 802.11 wireless network has no clothes", March 2001.
<http://www.cs.umd.edu/~waa/wireless.pdf>
- [24] Krawczyk, H., Bellare, M., Canetti, R., "HMAC: Keyed-Hashing for Message Authentication", February 1997.
- [25] Lipmaa, H., Rogaway, P., Wagner, D., "Comments to NIST concerning AES Modes of Operations: CTR-Mode Encryption", 2001
- [26] Lipmaa H., "IDEA: A Cipher for Multimedia Architectures?" In Stafford Tavares and Henk Meijer, editors, Selected Areas in Cryptography '98, volume 1556 of Lecture Notes in Computer Science, pages 248–263, Kingston, Canada, Springer-Verlag, 17–18 August 1998.



[27] Gaj, K., and Chodowiec, P., "Fast Implementation and Fair Comparison of the Final Candidates for Advanced Encryption Standard using Field Programmable Gate Arrays", Technical report 1998.
<http://ece.gmu.edu/crypto/publications.htm>.

[28] Harkins, D., Carrel, D., "RFC 2409; The Internet Key Exchange (IKE)", November 1998.

[29] Rogaway, P., "Problems with Proposed IP Cryptography", April 3, 1995.
<http://www.cs.ucdavis.edu/~rogaway/papers/draft-rogaway-ipsec-comments-00.txt>.

[30] FIPS197, "Specification for the Advanced Encryption Standard (AES)", November 26, 2001. <http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf>

[31] Daemen, J., Rijmen, V., "AES Proposal: Rijndael", June 11, 1998
<http://csrc.nist.gov/encryption/aes/rijndael/Rijndael.pdf>

[32] Daemen, J., Rijmen, V., "The design of Rijndael", 2000.
<http://www.esat.kuleuven.ac.be/~rijmen/rijndael/>

ÖZGEÇMİŞ



Fatih Birinci, 2 Mart 1973 tarihinde Ankara'da doğdu. İlk, orta ve lise eğitimini Ankara'da tamamladı. Daha sonra Orta Doğu Teknik Üniversitesi, Fen Edebiyat Fakültesi Matematik bölümünü birincilikle bitirdi. Yine aynı üniversite ve bölümde yüksek lisans eğitimini tamamladı. Tezin konusu, «Dizi Şifreleme Sistemlerinde İlişkiye Bağışıklık Atağı» idi. Yüksek lisansın ardından askerlik eğitimini tamamladı. 1997 tarihinden itibaren TÜBİTAK Ulusal Elektronik ve Kriptoloji Araştırma Enstitüsü'nde uzman araştırmacı olarak çalışmaktadır. Evli ve bir çocuk babasıdır.



EKLER



Performans Karşılaştırma Kodu

```
#include <stdio.h>
#include <stdlib.h>
#include <mem.h>
#include <dos.h>
#include <string.h>

#define MESAJ_UZNLK 32
#define ETIKET_UZNLK 16

#define MR_ECB 0
#define MR_CBC 1
#define MR_CFB1 2
#define MR_CFB2 3
#define MR_CFB4 5
#define MR_PCFB1 10
#define MR_PCFB2 11
#define MR_PCFB4 13
#define MR_OFB1 14
#define MR_OFB2 15
#define MR_OFB4 17
#define MR_OFB8 21
#define MR_OFB16 29
#define MR_CTR 30
#define MR_OCB 40

#define FALSE 0
#define TRUE 1
#define BYTE unsigned char
#define mr_unsign32 unsigned int
#define WORD mr_unsign32
#define NB 4

/* define this for a quick space/time performance boost */
#define LOTS_OF_MEMORY

/* rotates x one bit to the left */

#define ROTL(x) (((x)>>7)|((x)<<1))

/* Rotates 32-bit word left by 1, 2 or 3 byte */

#define ROTL8(x) (((x)<<8)|((x)>>24))
#define ROTL16(x) (((x)<<16)|((x)>>16))
#define ROTL24(x) (((x)<<24)|((x)>>8))
#define ON_HES_BLOKLAR 31 /* 0 and 31 arasinda olmalı */

typedef int BOOL;
typedef struct {
    int Nk,Nr;
    int mode;
    mr_unsign32 fkey[60];
    mr_unsign32 rkey[60];
    char f[16];
} aes;

typedef unsigned char block[16];
```

```

/*-----*/
void ocb(aes *,BYTE *);
typedef struct _anahtarstruct anahtarstruct;
struct _anahtarstruct {
    unsigned et_uznlik; /* Dogrulanacak Etiket uzunlugu */
    block L[ON_HES_BLOKLAR+1]; /* On hesap. L(i) degerleri, L[0] = L */
    block L_inv; /* On hesap. L/x value */
};
/*-----*/
anahtarstruct * /* anahtarstruct NULL veya ilklendirilmis */
ocb_aes_init(aes *a, char *sif_anahtar, /* AES anahtari */
             int anahtar_uznlik, /* Anahtar uzunlugu */
             unsigned et_uznlik, /* Etiket uzunlugu */
             anahtarstruct *anahtar, /* OCB anahtar yapisi */
             char *iv); /* IV */

anahtarstruct *ocb_bitir(anahtarstruct *anahtar);
/*-----*/
void ocb_aes_sifrele(aes *a,anahtarstruct *anahtar,/* Ilklendirilmis anahtarstruct */
                    char *IV, /* 16-byte IV */
                    char *pt, /* Acikmesaj Arabellegi */
                    unsigned pt_uznlik, /* pt'nin sekizli uzunlugu */
                    char *ct, /* Sifrelimesaj Arabellegi */
                    char *etiket); /* Etiket Arabellegi */
/*-----*/
int /* Etiket yanlissa o doner */
ocb_aes_sifreCoz(aes *a,anahtarstruct *anahtar,/* Ilklendirilmis anahtarstruct */
                 char *IV, /* 16-byte IV */
                 char *ct, /* Acikmesaj Arabellegi */
                 unsigned ct_uznlik, /* ct'nin sekizli uzunlugu */
                 char *pt, /* Sifrelimesaj Arabellegi */
                 char *etiket); /* Etiket Arabellegi */
/*-----*/
static BYTE InCo[4]={0xB,0xD,0x9,0xE}; /* Inverse Coefficients */

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

```

126,110,72,195,163,182,30,66,58,107,40,84,250,133,61,186,
43,121,10,21,155,159,94,202,78,212,172,229,243,115,167,87,
175,88,168,80,244,234,214,116,79,174,233,213,231,230,173,232,
44,215,117,122,235,22,11,245,89,203,95,176,156,169,81,160,
127,12,246,111,23,196,73,236,216,67,31,45,164,118,123,183,
204,187,62,90,251,96,177,134,59,82,161,108,170,85,41,157,
151,178,135,144,97,190,220,252,188,149,207,205,55,63,91,209,
83,57,132,60,65,162,109,71,20,42,158,93,86,242,211,171,
68,17,146,217,35,32,46,137,180,124,184,38,119,153,227,165,
103,74,237,222,197,49,254,24,13,99,140,128,192,247,112,7);
static BYTE fbsub[]=
{99,124,119,123,242,107,111,197,48,1,103,43,254,215,171,118,
202,130,201,125,250,89,71,240,173,212,162,175,156,164,114,192,
183,253,147,38,54,63,247,204,52,165,229,241,113,216,49,21,
4,199,35,195,24,150,5,154,7,18,128,226,235,39,178,117,
9,131,44,26,27,110,90,160,82,59,214,179,41,227,47,132,
83,209,0,237,32,252,177,91,106,203,190,57,74,76,88,207,
208,239,170,251,67,77,51,133,69,249,2,127,80,60,159,168,
81,163,64,143,146,157,56,245,188,182,218,33,16,255,243,210,
205,12,19,236,95,151,68,23,196,167,126,61,100,93,25,115,
96,129,79,220,34,42,144,136,70,238,184,20,222,94,11,219,
224,50,58,10,73,6,36,92,194,211,172,98,145,149,228,121,
231,200,55,109,141,213,78,169,108,86,244,234,101,122,174,8,
186,120,37,46,28,166,180,198,232,221,116,31,75,189,139,138,
112,62,181,102,72,3,246,14,97,53,87,185,134,193,29,158,
225,248,152,17,105,217,142,148,155,30,135,233,206,85,40,223,
140,161,137,13,191,230,66,104,65,153,45,15,176,84,187,22};
static BYTE rbsub[]=
{82,9,106,213,48,54,165,56,191,64,163,158,129,243,215,251,
124,227,57,130,155,47,255,135,52,142,67,68,196,222,233,203,
84,123,148,50,166,194,35,61,238,76,149,11,66,250,195,78,
8,46,161,102,40,217,36,178,118,91,162,73,109,139,209,37,
114,248,246,100,134,104,152,22,212,164,92,204,93,101,182,146,
108,112,72,80,253,237,185,218,94,21,70,87,167,141,157,132,
144,216,171,0,140,188,211,10,247,228,88,5,184,179,69,6,
208,44,30,143,202,63,15,2,193,175,189,3,1,19,138,107,
58,145,17,65,79,103,220,234,151,242,207,206,240,180,230,115,
150,172,116,34,231,173,53,133,226,249,55,232,28,117,223,110,
71,241,26,113,29,41,197,137,111,183,98,14,170,24,190,27,
252,86,62,75,198,210,121,32,154,219,192,254,120,205,90,244,
31,221,168,51,136,7,199,49,177,18,16,89,39,128,236,95,
96,81,127,169,25,181,74,13,45,229,122,159,147,201,156,239,
160,224,59,77,174,42,245,176,200,235,187,60,131,83,153,97,
23,43,4,126,186,119,214,38,225,105,20,99,85,33,12,125};
static WORD rco[]=
{1,2,4,8,16,32,64,128,27,54,108,216,171,77,154,47};
static WORD ftable[]=
{0xa56363c6,0x847c7cf8,0x997777ee,0x8d7b7bf6,0xdf2f2ff,0xbd6b6bd6,
0xb16f6fde,0x54c5c591,0x50303060,0x3010102,0xa96767ce,0x7d2b2b56,
0x19fefee7,0x62d7d7b5,0xe6abab4d,0x9a7676ec,0x45caca8f,0x9d82821f,
0x40c9c989,0x877d7dfa,0x15fafaef,0xeb5959b2,0xc947478e,0xbf0f0fb,
0xecadad41,0x67d4d4b3,0xfda2a25f,0xeaafaf45,0xbf9c9c23,0xf7a4a453,
0x967272e4,0x5bc0c09b,0xc2b7b775,0x1cfdfe1,0xae93933d,0x6a26264c,
0x5a36366c,0x413f3f7e,0x2f7f7f5,0x4fcccc83,0x5c343468,0xf4a5a551,
0x34e5e5d1,0x8f1f1f9,0x937171e2,0x73d8d8ab,0x53313162,0x3f15152a,
0xc040408,0x52c7c795,0x65232346,0x5ec3c39d,0x28181830,0xa1969637,
0xf05050a,0xb59a9a2f,0x907070e,0x36121224,0x9b80801b,0x3de2e2df,
0x26ebecbd,0x6927274e,0xcdb2b27f,0x9f7575ea,0x1b090912,0x9e83831d,
0x742c2c58,0x2e1a1a34,0x2d1b1b36,0xb26e6edc,0xee5a5ab4,0xfba0a05b,
0xf65252a4,0x4d3b3b76,0x61d6d6b7,0xceb3b37d,0x7b292952,0x3ee3e3dd,

0x712f2f5e,0x97848413,0xf55353a6,0x68d1d1b9,0x0,0x2cededc1,
0x60202040,0x1ffcfe3,0xc8b1b179,0xed5b5bb6,0xbe6a6ad4,0x46cbcb8d,
0xd9bebe67,0x4b393972,0xde4a4a94,0xd44c4c98,0xe85858b0,0x4acfcf85,
0x6bd0d0bb,0x2aefefc5,0xe5aaaa4f,0x16fbfbcd,0xc5434386,0xd74d4d9a,
0x55333366,0x94858511,0xcf45458a,0x10f9f9e9,0x6020204,0x817f7ffe,
0xf05050a0,0x443c3c78,0xba9f9f25,0xe3a8a84b,0xf35151a2,0xfea3a35d,
0xc0404080,0x8a8f8f05,0xad92923f,0xbc9d9d21,0x48383870,0x4f5f5f1,
0xdfbcbcb63,0xc1b6b677,0x75dadaaf,0x63212142,0x30101020,0x1affffe5,
0xef3f3fd,0x6dd2d2bf,0x4ccdc81,0x140c0c18,0x35131326,0x2fececc3,
0xe15f5fbe,0xa2979735,0xcc444488,0x3917172e,0x57c4c493,0xf2a7a755,
0x827e7efc,0x473d3d7a,0xac6464c8,0xe75d5dba,0x2b191932,0x957373e6,
0xa06060c0,0x98818119,0xd14f4f9e,0x7fcdca3,0x66222244,0x7e2a2a54,
0xab90903b,0x8388880b,0xca46468c,0x29eeec7,0xd3b8b86b,0x3c141428,
0x79dedea7,0xe25e5ebc,0x1d0b0b16,0x76dbdbad,0x3be0e0db,0x56323264,
0x4e3a3a74,0x1e0a0a14,0xdb494992,0xa06060c,0x6c242448,0xe45c5cb8,
0x5dc2c29f,0x6ed3d3bd,0xfacac43,0xa66262c4,0xa8919139,0xa4959531,
0x37e4e4d3,0x8b7979f2,0x32e7e7d5,0x43c8c88b,0x5937376e,0xb76d6dda,
0x8c8d8d01,0x64d5d5b1,0xd24e4e9c,0xe0a9a949,0xb46c6cd8,0xfa5656ac,
0x714f4f3,0x25eaeacf,0xaf6565ca,0x8e7a7af4,0xe9aeae47,0x18080810,
0xd5baba6f,0x887878f0,0x6f25254a,0x722e2e5c,0x241c1c38,0xf1a6a657,
0xc7b4b473,0x51c6c697,0x23e8e8cb,0x7cddda1,0x9c7474e8,0x211f1f3e,
0xdd4b4b96,0xdcdbdb61,0x868b8b0d,0x858a8a0f,0x907070e0,0x423e3e7c,
0xc4b5b571,0xaa6666cc,0xd8484890,0x5030306,0x1f6f6f7,0x120e0e1c,
0xa36161c2,0x5f35356a,0xf95757ae,0xd0b9b969,0x91868617,0x58c1c199,
0x271d1d3a,0xb99e9e27,0x38e1e1d9,0x13f8f8eb,0xb398982b,0x33111122,
0xbb6969d2,0x70d9d9a9,0x898e8e07,0xa7949433,0xb69b9b2d,0x221e1e3c,
0x92878715,0x20e9e9c9,0x49cece87,0xff5555aa,0x78282850,0x7adfdfa5,
0x8f8c8c03,0xf8a1a159,0x80898909,0x170d0d1a,0xdabfbf65,0x31e6e6d7,
0xc6424284,0xb86868d0,0xc3414182,0xb0999929,0x772d2d5a,0x110f0f1e,
0xcbb0b07b,0xfc5454a8,0xd6bbbb6d,0x3a16162c};
static WORD rtable[]=
{0x50a7f451,0x5365417e,0xc3a4171a,0x965e273a,0xcb6bab3b,0xf1459d1f,
0xab58faac,0x9303e34b,0x55fa3020,0xf66d76ad,0x9176cc88,0x254c02f5,
0xfcd7e54f,0xd7cb2ac5,0x80443526,0x8fa362b5,0x495ab1de,0x671bba25,
0x980eea45,0xe1c0fe5d,0x2752fc3,0x12f04c81,0xa397468d,0xc6f9d36b,
0xe75f8f03,0x959c9215,0xeb7a6dbf,0xda595295,0x2d83bed4,0xd3217458,
0x2969e049,0x44c8c98e,0x6a89c275,0x78798ef4,0x6b3e5899,0xdd71b927,
0xb64fe1be,0x17ad88f0,0x66ac20c9,0xb43ace7d,0x184adf63,0x82311ae5,
0x60335197,0x457f5362,0xe07764b1,0x84ae6bbb,0x1ca081fe,0x942b08f9,
0x58684870,0x19fd458f,0x876cde94,0xb7f87b52,0x23d373ab,0xe02024b72,
0x578f1fe3,0x2aab5566,0x728ebb2,0x3c2b52f,0x9a7bc586,0xa50837d3,
0xf2872830,0xb2a5bf23,0xba6a0302,0x5c8216ed,0x2b1ccf8a,0x92b479a7,
0xf0f207f3,0xa1e2694e,0xcdf4da65,0xd5be0506,0x1f6234d1,0x8afea6c4,
0x9d532e34,0xa055f3a2,0x32e18a05,0x75ebf6a4,0x39ec830b,0xaaef6040,
0x69f715e,0x51106ebd,0xf98a213e,0x3d06dd96,0xae053edd,0x46bde64d,
0xb58d5491,0x55dc471,0x6fd40604,0xff155060,0x24fb9819,0x97e9bdd6,
0xcc434089,0x779ed967,0xbd42e8b0,0x888b8907,0x385b19e7,0xdbeec879,
0x470a7ca1,0xe90f427c,0xc91e84f8,0x0,0x83868009,0x48ed2b32,
0xac70111e,0x4e725a6c,0xfbff0efd,0x5638850f,0x1ed5ae3d,0x27392d36,
0x64d90f0a,0x21a65c68,0xd1545b9b,0x3a2e3624,0xb1670a0c,0xfe75793,
0xd296eeb4,0x9e919b1b,0x4fc5c080,0xa220dc61,0x694b775a,0x161a121c,
0xaba93e2,0xe52aa0c0,0x43e0223c,0x1d171b12,0xb0d090e,0xad78bf2,
0xb9a8b62d,0xc8a91e14,0x8519f157,0x4c0775af,0xbbdd99ee,0xfd607fa3,
0x9f2601f7,0xbcf5725c,0xc53b6644,0x347efb5b,0x7629438b,0xdcc623cb,
0x68fcedb6,0x63f1e4b8,0xcadc31d7,0x10856342,0x40229713,0x2011c684,
0x7d244a85,0xf83dbbd2,0x1132f9ae,0x6da129c7,0x4b2f9e1d,0xf330b2dc,
0xec52860d,0xd0e3c177,0x6c16b32b,0x99b970a9,0xfa489411,0x2264e947,
0xc48cfca8,0x1a3ff0a0,0xd82c7d56,0xef903322,0xc74e4987,0xc1d138d9,
0xfea2ca8c,0x360bd498,0xcf81f5a6,0x28de7aa5,0x268eb7da,0xa4bfad3f,
0xe49d3a2c,0xd927850,0x9bcc5f6a,0x62467e54,0xc2138df6,0xe8b8d890,

```

0x5ef7392e,0xf5afc382,0xbe805d9f,0x7c93d069,0xa92dd56f,0xb31225cf,
0x3b99acc8,0xa77d1810,0x6e639ce8,0x7bbb3bdb,0x97826cd,0xf418596e,
0x1b79aec,0xa89a4f83,0x656e95e6,0x7ee6ffaa,0x8cfbc21,0xe6e815ef,
0xd99be7ba,0xce366f4a,0xd4099fea,0xd67cb029,0xafb2a431,0x31233f2a,
0x3094a5c6,0xc066a235,0x37bc4e74,0xa6ca82fc,0xb0d090e0,0x15d8a733,
0x4a9804f1,0xf7daec41,0xe50cd7f,0x2ff69117,0x8dd64d76,0x4db0ef43,
0x544daacc,0xdf0496e4,0xe3b5d19e,0x1b886a4c,0xb81f2cc1,0x7f516546,
0x4ea5e9d,0x5d358c01,0x737487fa,0x2e410bfb,0x5a1d67b3,0x52d2db92,
0x335610e9,0x1347d66d,0x8c61d79a,0x7a0ca137,0x8e14f859,0x893c13eb,
0xee27a9ce,0x35c961b7,0xede51ce1,0x3cb1477a,0x59dfd29c,0x3f73f255,
0x79ce1418,0xbf37c773,0xeacdf753,0x5baafd5f,0x146f3ddf,0x86db4478,
0x81f3afca,0x3ec468b9,0x2c342438,0x5f40a3c2,0x72c31d16,0xc25e2bc,
0x8b493c28,0x41950dff,0x7101a839,0xdeb30c08,0x9ce4b4d8,0x90c15664,
0x6184cb7b,0x70b632d5,0x745c6c48,0x4257b8d0};

```

```

/*-----*/

```

```

#ifdef LOTS_OF_MEMORY

```

```

static WORD ftable1[] =

```

```

{0x6363c6a5,0x7c7cf884,0x7777ee99,0x7b7bf68d,0xf2f2ff0d,0x6b6bd6bd,
0x6f6fdeb1,0xc5c59154,0x30306050,0x1010203,0x6767cea9,0x2b2b567d,
0xfefee719,0xd7d7b562,0xabab4de6,0x7676ec9a,0xcaca8f45,0x82821f9d,
0xc9c98940,0x7d7dfa87,0xfafaef15,0x5959b2eb,0x47478ec9,0xf0f0fb0b,
0xadad41ec,0xd4d4b367,0xa2a25ffd,0xafaf45ea,0x9c9c23bf,0xa4a453f7,
0x7272e496,0xc0c09b5b,0xb7b775c2,0xfdfde11c,0x93933dae,0x26264c6a,
0x36366c5a,0x3f3f7e41,0xf7f7f502,0xcccc834f,0x3434685c,0xa5a551f4,
0xe5e5d134,0xf1f1f908,0x7171e293,0xd8d8ab73,0x31316253,0x15152a3f,
0x404080c,0xc7c79552,0x23234665,0xc3c39d5e,0x18183028,0x969637a1,
0x5050a0f,0x9a9a2fb5,0x7070e09,0x12122436,0x80801b9b,0xe2e2df3d,
0xebebcd26,0x27274e69,0xb2b27fcd,0x7575ea9f,0x909121b,0x83831d9e,
0x2c2c5874,0x1a1a342e,0x1b1b362d,0x6e6edcb2,0x5a5ab4ee,0xa0a05bfb,
0x5252a4f6,0x3b3b764d,0xd6d6b761,0xb3b37dce,0x2929527b,0xe3e3dd3e,
0x2f2f5e71,0x84841397,0x5353a6f5,0xd1d1b968,0x0,0xededc12c,
0x20204060,0xfcfce31f,0xb1b179c8,0x5b5bb6ed,0x6a6ad4be,0xcbcb8d46,
0xbebe67d9,0x3939724b,0x4a4a94de,0x4c4c98d4,0x5858b0e8,0xcfcf854a,
0xd0d0bb6b,0xefefc52a,0xaaaa4fe5,0xfbfbed16,0x434386c5,0x4d4d9ad7,
0x33336655,0x85851194,0x45458acf,0xf9f9e910,0x2020406,0x7f7ffe81,
0x5050a0f0,0x3c3c7844,0x9f9f25ba,0xa8a84be3,0x5151a2f3,0xa3a35dfe,
0x404080c0,0x8f8f058a,0x92923fad,0x9d9d21bc,0x38387048,0xf5f5f104,
0xbcbcb63df,0xb6b677c1,0xdadaaf75,0x21214263,0x10102030,0xfffff51a,
0xf3f3fd0e,0xd2d2bf6d,0xcdcd814c,0xc0c01814,0x13132635,0xecccc32f,
0x5f5fbee1,0x979735a2,0x444488cc,0x17172e39,0xc4c49357,0xa7a755f2,
0x7e7efc82,0x3d3d7a47,0x6464c8ac,0x5d5dbaef,0x1919322b,0x7373e695,
0x6060c0a0,0x81811998,0x4f4f9ed1,0xdcda37f,0x22224466,0x2a2a547e,
0x90903bab,0x88880b83,0x46468cca,0xeeeeec729,0xb8b886bd3,0x1414283c,
0xdede779,0x5e5ebce2,0xb0b0161d,0xdbdbad76,0xe0e0db3b,0x32326456,
0x3a3a744e,0xa0a0141e,0x494992db,0x6060c0a,0x2424486c,0x5c5c8e4,
0xc2c29f5d,0xd3d3bd6e,0xacac43ef,0x6262c4a6,0x919139a8,0x959531a4,
0xe4e4d337,0x7979f28b,0xe7e7d532,0xc8c88b43,0x37376e59,0x6d6ddab7,
0x8d8d018c,0xd5d5b164,0x4e4e9cd2,0xa9a949e0,0x6c6cd8b4,0x5656acfa,
0xf4f4f307,0xaeaeacf25,0x6565caaf,0x7a7af48e,0xaeae47e9,0x8081018,
0xbaba6fd5,0x7878f088,0x25254a6f,0x2e2e5c72,0x1c1c3824,0xa6a657f1,
0xb4b473c7,0xc6c69751,0xe8e8cb23,0xdddda17c,0x7474e89c,0x1f1f3e21,
0x4b4b96dd,0xbdbdb61dc,0x8b8b0d86,0x8a8a0f85,0x7070e090,0x3e3e7c42,
0xb5b571c4,0x6666ccaa,0x484890d8,0x3030605,0xf6f6f701,0xe0e01c12,
0x6161c2a3,0x35356a5f,0x5757aef9,0xb9b969d0,0x86861791,0xc1c19958,
0x1d1d3a27,0x9e9e27b9,0xe1e1d938,0xf8f8eb13,0x98982bb3,0x11112233,
0x6969d2bb,0xd9d9a970,0x8e8e0789,0x949433a7,0x9b9b2db6,0x1e1e3c22,
0x87871592,0xe9e9c920,0xcece8749,0x5555aaff,0x28285078,0xdfdfa57a,
0x8c8c038f,0xa1a159f8,0x89890980,0xd0d01a17,0xbfbf65da,0xe6e6d731,
0x424284c6,0x6868d0b8,0x414182c3,0x999929b0,0x2d2d5a77,0xf0f01e11,

```



0xb0b07bcb,0x5454a8fc,0xbbbb6dd6,0x16162c3a};
static WORD rtable1[]=
{0xa7145150,0x65417e53,0xa4171ac3,0x5e273a96,0x6bab3bcb,0x459d1ff1,
0x58faacab,0x3e34b93,0xfa302055,0x6d76adf6,0x76cc8891,0x4c02f525,
0xd7e54ffc,0xcb2ac5d7,0x44352680,0xa362b58f,0x5ab1de49,0x1bba2567,
0xeea4598,0xc0fe5de1,0x752fc302,0xf04c8112,0x97468da3,0xf9d36bc6,
0x5f8f03e7,0x9c921595,0x7a6dbfeb,0x595295da,0x83bed42d,0x217458d3,
0x69e04929,0xc8c98e44,0x89c2756a,0x798ef478,0x3e58996b,0x71b927dd,
0x4fe1beb6,0xad88f017,0xac20c966,0x3ace7db4,0x4adf6318,0x311ae582,
0x33519760,0x7f536245,0x7764b1e0,0xae6bbb84,0xa081fe1c,0x2b08f994,
0x68487058,0xfd45f19,0x6cde9487,0xf87b52b7,0xd373ab23,0x24b72e2,
0x8f1fe357,0xab55662a,0x28ebb207,0xc2b52f03,0x7bc5869a,0x837d3a5,
0x872830f2,0xa5bf23b2,0x6a0302ba,0x8216ed5c,0x1ccf8a2b,0xb479a792,
0xf207f3f0,0xe2694ea1,0xf4da65cd,0xbe0506d5,0x6234d11f,0xfea6c48a,
0x532e349d,0x55f3a2a0,0xe18a0532,0xebf6a475,0xec830b39,0xef6040aa,
0x9f715e06,0x106ebd51,0x8a213ef9,0x6dd963d,0x53eddae,0xbde64d46,
0x8d5491b5,0x5dc47105,0xd406046f,0x155060ff,0xfb981924,0xe9bdd697,
0x434089cc,0x9ed96777,0x42e8b0bd,0xb890788,0x5b19e738,0xeec879db,
0xa7ca147,0xf427ce9,0x1e84f8c9,0x0,0x86800983,0xed2b3248,
0x70111eac,0x725a6c4e,0xff0efdfb,0x38850f56,0xd5ae3d1e,0x392d3627,
0xd90f0a64,0xa65c6821,0x545b9bd1,0x2e36243a,0x670a0cb1,0xe757930f,
0x96eeb4d2,0x919b1b9e,0xc5c0804f,0x20dc61a2,0x4b775a69,0x1a121c16,
0xba93e20a,0x2aa0c0e5,0xe0223c43,0x171b121d,0xd090e0b,0xc78bf2ad,
0xa8b62db9,0xa91e14c8,0x19f15785,0x775af4c,0xdd99eebb,0x607fa3fd,
0x2601f79f,0xf5725cbc,0x3b6644c5,0x7efb5b34,0x29438b76,0xc623cbdc,
0xfcedb668,0xf1e4b863,0xdc31d7ca,0x85634210,0x22971340,0x11c68420,
0x244a857d,0x3dbbd2f8,0x32f9ae11,0xa129c76d,0x2f9e1d4b,0x30b2dcf3,
0x52860dec,0xe3c177d0,0x16b32b6c,0xb970a999,0x489411fa,0x64e94722,
0x8cfca8c4,0x3ff0a01a,0x2c7d56d8,0x903322ef,0x4e4987c7,0xd138d9c1,
0xa2ca8cfe,0xbd49836,0x81f5a6cf,0xde7aa528,0x8eb7da26,0xbfad3fa4,
0x9d3a2ce4,0x9278500d,0xcc5f6a9b,0x467e5462,0x138df6c2,0xb8d890e8,
0xf7392e5e,0xafc382f5,0x805d9fbe,0x93d0697c,0x2dd56fa9,0x1225cfb3,
0x99acc83b,0x7d1810a7,0x639ce86e,0xbb3bdb7b,0x7826cd09,0x18596ef4,
0xb79aec01,0x9a4f83a8,0x6e95e665,0xe6ffaa7e,0xcfbcb2108,0xe815efe6,
0x9be7bad9,0x366f4ace,0x99fead4,0x7cb029d6,0xb2a431af,0x233f2a31,
0x94a5c630,0x66a235c0,0xbc4e7437,0xca82fca6,0xd090e0b0,0xd8a73315,
0x9804f14a,0xdaec41f7,0x50cd7f0e,0xf691172f,0xd64d768d,0xb0ef434d,
0x4daacc54,0x496e4df,0xb5d19ee3,0x886a4c1b,0x1f2cc1b8,0x5165467f,
0xea5e9d04,0x358c015d,0x7487fa73,0x410bfb2e,0x1d67b35a,0xd2db9252,
0x5610e933,0x47d66d13,0x61d79a8c,0xca1377a,0x14f8598e,0x3c13eb89,
0x27a9ceee,0xc961b735,0xe51ce1ed,0xb1477a3c,0xdfd29c59,0x73f2553f,
0xce141879,0x37c773bf,0xcdf753ea,0xaafd5f5b,0x6f3ddf14,0xdb447886,
0xf3afca81,0xc468b93e,0x3424382c,0x40a3c25f,0xc31d1672,0x25e2bc0c,
0x493c288b,0x950dff41,0x1a83971,0xb30c08de,0xe4b4d89c,0xc1566490,
0x84cb7b61,0xb632d570,0x5c6c4874,0x57b8d042};
static WORD ftable2[]=
{0x63c6a563,0x7cf8847c,0x77ee9977,0x7bf68d7b,0xf2ff0df2,0x6bd6bd6b,
0x6fdeb16f,0xc59154c5,0x30605030,0x1020301,0x67cea967,0x2b567d2b,
0xfef719fe,0xd7b562d7,0xab4de6ab,0x76ec9a76,0xca8f45ca,0x821f9d82,
0xc98940c9,0x7dfa877d,0xfaef15fa,0x59b2eb59,0x478ec947,0xf0fb0bf0,
0xad41ecad,0xd4b367d4,0xa25ffda2,0xaf45eaaf,0x9c23bf9c,0xa453f7a4,
0x72e49672,0xc09b5bc0,0xb775c2b7,0xfde11cfd,0x933dae93,0x264c6a26,
0x366c5a36,0x3f7e413f,0xf7f502f7,0xcc834fcc,0x34685c34,0xa551f4a5,
0xe5d134e5,0xf1f908f1,0x71e29371,0xd8ab73d8,0x31625331,0x152a3f15,
0x4080c04,0xc79552c7,0x23466523,0xc39d5ec3,0x18302818,0x9637a196,
0x50a0f05,0x9a2fb59a,0x70e0907,0x12243612,0x801b9b80,0xe2df3de2,
0xebcd26eb,0x274e6927,0xb27fcd2,0x75ea9f75,0x9121b09,0x831d9e83,
0x2c58742c,0x1a342e1a,0x1b362d1b,0x6edcb26e,0x5ab4ee5a,0xa05bfba0,
0x52a4f652,0x3b764d3b,0xd6b761d6,0xb37dceb3,0x29527b29,0xe3dd3ee3,
0x2f5e712f,0x84139784,0x53a6f553,0xd1b968d1,0x0,0xedc12ced,



0x20406020,0xfce31ffc,0xb179c8b1,0x5bb6ed5b,0x6ad4be6a,0xcb8d46cb,
0xbe67d9be,0x39724b39,0x4a94de4a,0x4c98d44c,0x58b0e858,0xcf854acf,
0xd0bb6bd0,0xefc52aef,0xaa4fe5aa,0xfbed16fb,0x4386c543,0x4d9ad74d,
0x33665533,0x85119485,0x458acf45,0xf9e910f9,0x2040602,0x7ffe817f,
0x50a0f050,0x3c78443c,0x9f25ba9f,0xa84be3a8,0x51a2f351,0xa35df5a3,
0x4080c040,0x8f058a8f,0x923fad92,0x9d21bc9d,0x38704838,0xf5f104f5,
0xbc63dfbc,0xb677c1b6,0xdaaf75da,0x21426321,0x10203010,0xffe51aff,
0xf3fd0ef3,0xd2bf6dd2,0xcd814ccd,0xc18140c,0x13263513,0xecc32fec,
0x5fbee15f,0x9735a297,0x4488cc44,0x172e3917,0xc49357c4,0xa755f2a7,
0x7efc827e,0x3d7a473d,0x64c8ac64,0x5dbae75d,0x19322b19,0x73e69573,
0x60c0a060,0x81199881,0x4f9ed14f,0xdc3a37dc,0x22446622,0x2a547e2a,
0x903bab90,0x880b8388,0x468cca46,0xeec729ee,0xb86bd3b8,0x14283c14,
0xde779de,0x5ebce25e,0xb161d0b,0xdbad76db,0xe0db3be0,0x32645632,
0x3a744e3a,0xa141e0a,0x4992db49,0x60c0a06,0x24486c24,0x5cb8e45c,
0xc29f5dc2,0xd3bd6ed3,0xac43efac,0x62c4a662,0x9139a891,0x9531a495,
0xe4d337e4,0x79f28b79,0xe7d532e7,0xc88b43c8,0x376e5937,0x6ddab76d,
0x8d018c8d,0xd5b164d5,0x4e9cd24e,0xa949e0a9,0x6cd8b46c,0x56acfa56,
0xf4f307f4,0xeacf25ea,0x65caaf65,0x7af48e7a,0xae47e9ae,0x8101808,
0xba6fd5ba,0x78f08878,0x254a6f25,0x2e5c722e,0x1c38241c,0xa657f1a6,
0xb473c7b4,0xc69751c6,0xe8cb23e8,0xdda17cdd,0x74e89c74,0x1f3e211f,
0x4b96dd4b,0xbd61dcbd,0x8b0d868b,0x8a0f858a,0x70e09070,0x3e7c423e,
0xb571c4b5,0x66ccaa66,0x4890d848,0x3060503,0xf6f701f6,0xe1c120e,
0x61c2a361,0x356a5f35,0x57aef957,0xb969d0b9,0x86179186,0xc19958c1,
0x1d3a271d,0x9e27b99e,0xe1d938e1,0xf8eb13f8,0x982bb398,0x11223311,
0x69d2bb69,0xd9a970d9,0x8e07898e,0x9433a794,0x9b2db69b,0x1e3c221e,
0x87159287,0xe9c920e9,0xce8749ce,0x55aaff55,0x28507828,0xdfa57adf,
0x8c038f8c,0xa159f8a1,0x89098089,0xd1a170d,0xbf65dabf,0xe6d731e6,
0x4284c642,0x68d0b868,0x4182c341,0x9929b099,0x2d5a772d,0xf1e110f,
0xb07cbcb0,0x54a8fc54,0xbb6dd6bb,0x162c3a16};
static WORD rtable2[]=
{0xf45150a7,0x417e5365,0x171ac3a4,0x273a965e,0xab3bcb6b,0x9d1ff145,
0xfaacab58,0xe34b9303,0x302055fa,0x76adf66d,0xcc889176,0x2f5254c,
0xe54ffcd7,0x2ac5d7cb,0x35268044,0x62b58fa3,0xb1de495a,0xba25671b,
0xea45980e,0xfe5de1c0,0x2fc30275,0x4c8112f0,0x468da397,0xd36bc6f9,
0x8f03e75f,0x9215959c,0x6dbfeb7a,0x5295da59,0xbed42d83,0x7458d321,
0xe0492969,0xc98e44c8,0xc2756a89,0x8ef47879,0x58996b3e,0xb927dd71,
0xe1beb64f,0x88f017ad,0x20c966ac,0xce7db43a,0xdf63184a,0x1ae58231,
0x51976033,0x5362457f,0x64b1e077,0x6bbb84ae,0x81fe1ca0,0x8f9942b,
0x48705868,0x458f19fd,0xde94876c,0x7b52b7f8,0x73ab23d3,0x4b72e202,
0x1fe3578f,0x55662aab,0xebb20728,0xb52f03c2,0xc5869a7b,0x37d3a508,
0x2830f287,0xbf23b2a5,0x302ba6a,0x16ed5c82,0xcf8a2b1c,0x79a792b4,
0x7f3f0f2,0x694ea1e2,0xda65cdf4,0x506d5be,0x34d11f62,0xa6c48afe,
0x2e349d53,0xf3a2a055,0x8a0532e1,0xf6a475eb,0x830b39ec,0x6040aaef,
0x715e069f,0x6ebd5110,0x213ef98a,0xdd963d06,0x3eddae05,0xe64d46bd,
0x5491b58d,0xc471055d,0x6046fd4,0x5060ff15,0x981924fb,0xbdd697e9,
0x4089cc43,0xd967779e,0xe8b0bd42,0x8907888b,0x19e7385b,0xc879dbee,
0x7ca1470a,0x427ce90f,0x84f8c91e,0x0,0x80098386,0x2b3248ed,
0x111eac70,0x5a6c4e72,0xefdfbff,0x850f5638,0xae3d1ed5,0x2d362739,
0xf0a64d9,0x5c6821a6,0x5b9bd154,0x36243a2e,0xa0cb167,0x57930fe7,
0xeb4d296,0x9b1b9e91,0xc0804fc5,0xdc61a220,0x775a694b,0x121c161a,
0x93e20aba,0xa0c0e52a,0x223c43e0,0x1b121d17,0x90e0b0d,0x8bf2adc7,
0xb62db9a8,0x1e14c8a9,0xf1578519,0x75af4c07,0x99eebbdd,0x7fa3fd60,
0x1f79f26,0x725cbcf5,0x6644c53b,0xfb5b347e,0x438b7629,0x23cbdccc6,
0xedb668fc,0xe4b863f1,0x31d7cad,0x63421085,0x97134022,0xc6842011,
0x4a857d24,0xbbd2f83d,0xf9ae1132,0x29c76da1,0x9e1d4b2f,0xb2dcf330,
0x860dec52,0xc177d0e3,0xb32b6c16,0x70a999b9,0x9411fa48,0xe9472264,
0xfca8c48c,0xf0a01a3f,0x7d56d82c,0x3322ef90,0x4987c74e,0x38d9c1d1,
0xca8cfea2,0xd498360b,0xf5a6cf81,0x7aa528de,0xb7da268e,0xad3fa4bf,
0x3a2ce49d,0x78500d92,0x5f6a9bcc,0x7e546246,0x8df6c213,0xd890e8b8,
0x392e5ef7,0xc382f5af,0x5d9fbe80,0xd0697c93,0xd56fa92d,0x25cfb312,

```

0xacc83b99,0x1810a77d,0x9ce86e63,0x3bdb7bbb,0x26cd0978,0x596ef418,
0x9aec01b7,0x4f83a89a,0x95e6656e,0xffaa7ee6,0xbc2108cf,0x15efe6e8,
0xe7bad99b,0x6f4ace36,0x9fead409,0xb029d67c,0xa431afb2,0x3f2a3123,
0xa5c63094,0xa235c066,0x4e7437bc,0x82fca6ca,0x90e0b0d0,0xa73315d8,
0x4f14a98,0xec41f7da,0xcd7f0e50,0x91172ff6,0x4d768dd6,0xef434db0,
0xaacc544d,0x96e4df04,0xd19ee3b5,0x6a4c1b88,0x2cc1b81f,0x65467f51,
0x5e9d04ea,0x8c015d35,0x87fa7374,0xbfb2e41,0x67b35a1d,0xdb9252d2,
0x10e93356,0xd66d1347,0xd79a8c61,0xa1377a0c,0xf8598e14,0x13eb893c,
0xa9ceee27,0x61b735c9,0x1ce1ede5,0x477a3cb1,0xd29c59df,0xf2553f73,
0x141879ce,0xc773bf37,0xf753eacd,0xfd5f5baa,0x3ddf146f,0x447886db,
0xafca81f3,0x68b93ec4,0x24382c34,0xa3c25f40,0x1d1672c3,0xe2bc0c25,
0x3c288b49,0xdf4195,0xa8397101,0xc08deb3,0xb4d89ce4,0x566490c1,
0xcb7b6184,0x32d570b6,0x6c48745c,0xb8d04257};
static WORD ftable3[] =
{0xc6a56363,0xf8847c7c,0xee997777,0xf68d7b7b,0xff0df2f2,0xd6bd6b6b,
0xdeb16f6f,0x9154c5c5,0x60503030,0x2030101,0xcea96767,0x567d2b2b,
0xe719fefe,0xb562d7d7,0x4de6abab,0xec9a7676,0x8f45caca,0x1f9d8282,
0x8940c9c9,0xfa877d7d,0xef15fafa,0xb2eb5959,0x8ec94747,0xfb0bf0f0,
0x41ecadad,0xb367d4d4,0x5fda2a2,0x45eaaafaf,0x23bf9c9c,0x53f7a4a4,
0xe4967272,0x9b5bc0c0,0x75c2b7b7,0xe11cfdfd,0x3dae9393,0x4c6a2626,
0x6c5a3636,0x7e413f3f,0xf502f7f7,0x834fcccc,0x685c3434,0x51f4a5a5,
0xd134e5e5,0xf908f1f1,0xe2937171,0xab73d8d8,0x62533131,0x2a3f1515,
0x80c0404,0x9552c7c7,0x46652323,0x9d5ec3c3,0x30281818,0x37a19696,
0xa0f0505,0x2fb59a9a,0xe090707,0x24361212,0x1b9b8080,0xdf3de2e2,
0xcd26eb6b,0x4e692727,0x7fdb2b2,0xea9f7575,0x121b0909,0x1d9e8383,
0x58742c2c,0x342e1a1a,0x362d1b1b,0xdc26e6e,0xb4ee5a5a,0x5bfa0a0,
0xa4f65252,0x764d3b3b,0xb761d6d6,0x7dceb3b3,0x527b2929,0xdd3ee3e3,
0x5e712f2f,0x13978484,0xa6f55353,0xb968d1d1,0x0,0xc12ceded,
0x40602020,0xe31ffcfc,0x79c8b1b1,0xb6ed5b5b,0xd4be6a6a,0x8d46cbcb,
0x67d9bebe,0x724b3939,0x94de4a4a,0x98d44c4c,0xb0e85858,0x854acfcf,
0xbb6bd0d0,0xc52aefef,0x4fe5aaaa,0xed16fbfb,0x86c54343,0x9ad74d4d,
0x66553333,0x11948585,0x8acf4545,0xe910f9f9,0x4060202,0xfe817f7f,
0xa0f05050,0x78443c3c,0x25ba9f9f,0x4be3a8a8,0xa2f35151,0x5dfea3a3,
0x80c04040,0x58a8f8f,0x3fad9292,0x21bc9d9d,0x70483838,0xf104f5f5,
0x63dfbc6b,0x77c1b6b6,0xaf75dada,0x42632121,0x20301010,0xe51affff,
0xfd0ef3f3,0xbf6dd2d2,0x814ccdc,0x18140c0c,0x26351313,0xc32fecec,
0xbee15f5f,0x35a29797,0x88cc4444,0x2e391717,0x9357c4c4,0x55f2a7a7,
0xfc827e7e,0x7a473d3d,0xc8ac6464,0xbae75d5d,0x322b1919,0xe6957373,
0xc0a06060,0x19988181,0x9ed14f4f,0xa37fcdcd,0x44662222,0x547e2a2a,
0x3bab9090,0xb838888,0x8cca4646,0xc729eeee,0x6bd3b8b8,0x283c1414,
0xa779dede,0xbce25e5e,0x161d0b0b,0xad76dbdb,0xdb3be0e0,0x64563232,
0x744e3a3a,0x141e0a0a,0x92db4949,0xc0a0606,0x486c2424,0xb8e45c5c,
0x9f5dc2c2,0xbd6ed3d3,0x43efacac,0xc4a66262,0x39a89191,0x31a49595,
0xd337e4e4,0xf28b7979,0xd532e7e7,0x8b43c8c8,0x6e593737,0xdab76d6d,
0x18c8d8d,0xb164d5d5,0x9cd24e4e,0x49e0a9a9,0xd8b46c6c,0xacfa5656,
0xf307f4f4,0xcf25eaea,0xcaaf6565,0xf48e7a7a,0x47e9aeae,0x10180808,
0x6fd5baba,0xf0887878,0x4a6f2525,0x5c722e2e,0x38241c1c,0x57f1a6a6,
0x73c7b4b4,0x9751c6c6,0xcb23e8e8,0xa17cdddd,0xe89c7474,0x3e211f1f,
0x96dd4b4b,0x61dcbdbd,0xd868b8b,0xf858a8a,0xe0907070,0x7c423e3e,
0x71c4b5b5,0xccaa6666,0x90d84848,0x6050303,0xf701f6f6,0x1c120e0e,
0xc2a36161,0x6a5f3535,0xae9f5757,0x69d0b9b9,0x17918686,0x9958c1c1,
0x3a271d1d,0x27b99e9e,0xd938e1e1,0xeb13f8f8,0x2bb39898,0x22331111,
0xd2bb6969,0xa970d9d9,0x7898e8e,0x33a79494,0x2db69b9b,0x3c221e1e,
0x15928787,0xc920e9e9,0x8749cece,0xaa5f5555,0x50782828,0xa57adfdf,
0x38f8c8c,0x59f8a1a1,0x9808989,0xa170d0d,0x65dabfbf,0xd731e6e6,
0x84c64242,0xd0b86868,0x82c34141,0x29b09999,0x5a772d2d,0x1e110f0f,
0x7bcb0b0,0xa8fc5454,0x6dd6bbb,0x2c3a1616};
static WORD rtable3[] =
{0x5150a7f4,0x7e536541,0x1ac3a417,0x3a965e27,0x3bcb6bab,0x1ff1459d,
0xacab58fa,0x4b9303e3,0x2055fa30,0xadf66d76,0x889176cc,0xf5254c02,

```

```

0x4ffcd7e5,0xc5d7cb2a,0x26804435,0xb58fa362,0xde495ab1,0x25671bba,
0x45980eea,0x5de1c0fe,0xc302752f,0x8112f04c,0x8da39746,0x6bc6f9d3,
0x3e75f8f,0x15959c92,0xbfeb7a6d,0x95da5952,0xd42d83be,0x58d32174,
0x492969e0,0x8e44c8c9,0x756a89c2,0xf478798e,0x996b3e58,0x27dd71b9,
0xb6b64fe1,0xf017ad88,0xc966ac20,0x7db43ace,0x63184adf,0xe582311a,
0x97603351,0x62457f53,0xb1e07764,0xbb84ae6b,0xfe1ca081,0xf9942b08,
0x70586848,0x8f19fd45,0x94876cde,0x52b7f87b,0xab23d373,0x72e2024b,
0xe3578f1f,0x662aab55,0xb20728eb,0x2f03c2b5,0x869a7bc5,0xd3a50837,
0x30f28728,0x23b2a5bf,0x2ba6a03,0xed5c8216,0x8a2b1ccf,0xa792b479,
0xf3f0f207,0x4ea1e269,0x65cdf4da,0x6d5be05,0xd11f6234,0xc48afea6,
0x349d532e,0xa2a055f3,0x532e18a,0xa475ebf6,0xb39ec83,0x40aaef60,
0x5e069f71,0xbd51106e,0x3ef98a21,0x963d06dd,0xddae053e,0x4d46bde6,
0x91b58d54,0x71055dc4,0x46fd406,0x60ff1550,0x1924fb98,0xd697e9bd,
0x89cc4340,0x67779ed9,0xb0bd42e8,0x7888b89,0xe7385b19,0x79dbeec8,
0xa1470a7c,0x7ce90f42,0xf8c91e84,0x0,0x9838680,0x3248ed2b,
0x1eac7011,0x6c4e725a,0xfdfbfff0,0xf563885,0x3d1ed5ae,0x3627392d,
0xa64d90f,0x6821a65c,0x9bd1545b,0x243a2e36,0xcb1670a,0x930fe757,
0xb4d296ee,0x1b9e919b,0x804fc5c0,0x61a220dc,0x5a694b77,0x1c161a12,
0xe20aba93,0xc0e52aa0,0x3c43e022,0x121d171b,0xe0b0d09,0xf2adc78b,
0x2db9a8b6,0x14c8a91e,0x578519f1,0xaf4c0775,0xeebbdd99,0xa3fd607f,
0xf79f2601,0x5cbcf572,0x44c53b66,0x5b347efb,0x8b762943,0xcbdcc623,
0xb668fced,0xb863f1e4,0xd7cad3c1,0x42108563,0x13402297,0x842011c6,
0x857d244a,0xd2f83dbb,0xae1132f9,0xc76da129,0x1d4b2f9e,0xdcf330b2,
0xdec5286,0x77d0e3c1,0x2b6c16b3,0xa999b970,0x11fa4894,0x472264e9,
0xa8c48cfc,0xa01a3ff0,0x56d82c7d,0x22ef9033,0x87c74e49,0xd9c1d138,
0x8cfea2ca,0x98360bd4,0xa6cf81f5,0xa528de7a,0xda268eb7,0x3fa4bfad,
0x2ce49d3a,0x500d9278,0x6a9bcc5f,0x5462467e,0xf6c2138d,0x90e8b8d8,
0x2e5ef739,0x82f5afc3,0x9fbe805d,0x697c93d0,0x6fa92dd5,0xcfb31225,
0xc83b99ac,0x10a77d18,0xe86e639c,0xdb7bbb3b,0xcd097826,0x6ef41859,
0xec01b79a,0x83a89a4f,0xe6656e95,0xaa7ee6ff,0x2108cfbc,0xefe6e815,
0xbad99be7,0x4ace366f,0xad4099f,0x29d67cb0,0x31afb2a4,0x2a31233f,
0xc63094a5,0x35c066a2,0x7437bc4e,0xfca6ca82,0xe0b0d090,0x3315d8a7,
0xf14a9804,0x41f7daec,0x7f0e50cd,0x172ff691,0x768dd64d,0x434db0ef,
0xcc544daa,0xe4df0496,0x9ee3b5d1,0x4c1b886a,0xc1b81f2c,0x467f5165,
0x9d04ea5e,0x15d358c,0xfa737487,0xfb2e410b,0xb35a1d67,0x9252d2db,
0xe9335610,0x6d1347d6,0x9a8c61d7,0x377a0ca1,0x598e14f8,0xeb893c13,
0xcdee27a9,0xb735c961,0xe1ede51c,0x7a3cb147,0x9c59dfd2,0x553f73f2,
0x1879ce14,0x73bf37c7,0x53eacdf7,0x5f5baafd,0xdf146f3d,0x7886db44,
0xca81f3af,0xb93ec468,0x382c3424,0xc25f40a3,0x1672c31d,0xbc0c25e2,
0x288b493c,0xff41950d,0x397101a8,0x8deb30c,0xd89ce4b4,0x6490c156,
0x7b6184cb,0xd570b632,0x48745c6c,0xd04257b8};

```

```
#endif
```

```
/*-----*/
```

```
void arttir(aes *a){ /*CTR calisma kipi icin a->f bir arttirilacak */
    int i;
```

```
/* printf("IV=\t");
   for (i=0;i<16;i++) printf("%02x",a->f[i]);
   printf("\n");
*/
```

```
for(i=15; i>=0; i--){
    a->f[i]++;
    if(a->f[i] != 0)break;
}
```

```
}
```

```
/*-----*/
```

```
static void modul2Topla(void *dst, void *src1, void *src2)
```

```
/* 128-ikil Modul 2 toplama: *dst = *src1 ^ *src2. Pointer'lar 32-ikil olmalı */
```

```
{
```

```

((unsigned *)dst)[0] = ((unsigned *)src1)[0] ^ ((unsigned *)src2)[0];
((unsigned *)dst)[1] = ((unsigned *)src1)[1] ^ ((unsigned *)src2)[1];
((unsigned *)dst)[2] = ((unsigned *)src1)[2] ^ ((unsigned *)src2)[2];
((unsigned *)dst)[3] = ((unsigned *)src1)[3] ^ ((unsigned *)src2)[3];
}
/*-----*/
static void sola_kaydir(unsigned char *x)
/* 128-ikil, 1 ikil sola kaydir: *x <<= 1. */
{
    int i;
    for (i = 0; i < 15; i++) {
        x[i] = (x[i] << 1) | (x[i+1] & 0x80 ? 1 : 0);
    }
    x[15] = (x[15] << 1);
}
/*-----*/
static void saga_kaydir(unsigned char *x)
/* 128-ikil, 1 ikil saga kaydir: *x >>= 1 */
{
    int i;
    for (i = 15; i > 0; i--) {
        x[i] = (x[i] >> 1) | (x[i-1] & 1 ? 0x80u : 0);
    }
    x[0] = (x[0] >> 1);
}
/*-----*/
static int ntz(unsigned i)
/* integer i nin sonunda kac bit oldugunu sayar */
{
    int rdeg = 0;
    while ((i & 1) == 0) {
        i >>= 1;
        rdeg++;
    }
    return rdeg;
}
/*-----*/
static WORD pack(BYTE *b)
/* pack bytes into a 32-bit Word */
{
    return ((WORD)b[3]<<24)|((WORD)b[2]<<16)|((WORD)b[1]<<8)|((WORD)b[0]);
}
/*-----*/
static void unpack(WORD a,BYTE *b)
/* unpack bytes from a word */
{
    b[0]=(BYTE)a;
    b[1]=(BYTE)(a>>8);
    b[2]=(BYTE)(a>>16);
    b[3]=(BYTE)(a>>24);
}
/*-----*/
/*
static BYTE xtime(BYTE a)
{
    BYTE b;
    if (a&0x80) b=0x1B;
    else b=0;
    a<<=1;
    a^=b;
    return a;
}
*/

```

```

*/
/*-----*/
static BYTE bmul(BYTE x,BYTE y)
{ /* x,y= AntiLog(Log(x) + Log(y)) */
  if (x && y) return ptab[(ltab[x]+ltab[y])%255];
  else return 0;
}
/*-----*/
static WORD SubByte(WORD a)
{
  BYTE b[4];
  unpack(a,b);
  b[0]=fbsub[b[0]];
  b[1]=fbsub[b[1]];
  b[2]=fbsub[b[2]];
  b[3]=fbsub[b[3]];
  return pack(b);
}
/*-----*/
static BYTE product(WORD x,WORD y)
{ /* dot product of two 4-byte arrays */
  BYTE xb[4],yb[4];
  unpack(x,xb);
  unpack(y,yb);
  return bmul(xb[0],yb[0])^bmul(xb[1],yb[1])^bmul(xb[2],yb[2])^bmul(xb[3],yb[3]);
}
/*-----*/
static WORD InvMixCol(WORD x)
{ /* matrix Multiplication */
  WORD y,m;
  BYTE b[4];

  m=pack(lnCo);
  b[3]=product(m,x);
  m=ROTL24(m);
  b[2]=product(m,x);
  m=ROTL24(m);
  b[1]=product(m,x);
  m=ROTL24(m);
  b[0]=product(m,x);
  y=pack(b);
  return y;
}
/*-----*/
void aes_reset(aes *a,int mode,char *iv)
{ /* reset mode, or reset iv */
  int i;
  a->mode=mode;
  for (i=0;i<4*NB;i++)
    a->f[i]=0;
  if (mode!=MR_ECB && iv!=NULL)
  {
    for (i=0;i<4*NB;i++)
      a->f[i]=iv[i];
  }
}
/*-----*/
void aes_getreg(aes *a,char *ir)
{
  int i;

```

```

    for (i=0;i<4*NB;i++) ir[i]=a->f[i];
}
/*-----*/
BOOL aes_init(aes* a,int mode,int nk,char *key,char *iv)
{ /* Key=nk bytes */
  /* currently NB,nk = 16, 24 or 32 */
  /* Key Scheduler. Create expanded encryption key */
  int i,j,k,N,nr;
  WORD CipherKey[8];

  nk/=4;

  if (nk!=4 && nk!=6 && nk!=8) return FALSE;

  /* nr is number of rounds */
  nr=6+nk;

  a->Nk=nk; a->Nr=nr;

  aes_reset(a,mode,iv);

  N=NB*(nr+1);

  for (i=j=0;i<nk;i++,j+=4)
  {
    CipherKey[i]=pack((BYTE *)&key[j]);
  }
  for (i=0;i<nk;i++) a->fkey[i]=CipherKey[i];
  for (j=nk,k=0;j<N;j+=nk,k++)
  {
    a->fkey[j]=a->fkey[j-nk]^SubByte(ROTL24(a->fkey[j-1]))^rco[k];
    if (nk<=6)
    {
      for (i=1;i<nk && (i+j)<N;i++)
        a->fkey[i+j]=a->fkey[i+j-nk]^a->fkey[i+j-1];
    }
    else
    {
      for (i=1;i<4 && (i+j)<N;i++)
        a->fkey[i+j]=a->fkey[i+j-nk]^a->fkey[i+j-1];
      if ((j+4)<N) a->fkey[j+4]=a->fkey[j+4-nk]^SubByte(a->fkey[j+3]);
      for (i=5;i<nk && (i+j)<N;i++)
        a->fkey[i+j]=a->fkey[i+j-nk]^a->fkey[i+j-1];
    }
  }
}

/* now for the expanded decrypt key in reverse order */

for (j=0;j<NB;j++) a->rkey[j+N-NB]=a->fkey[j];
for (i=NB;i<N-NB;i+=NB)
{
  k=N-NB-i;
  for (j=0;j<NB;j++) a->rkey[k+j]=InvMixCol(a->fkey[i+j]);
}
for (j=N-NB;j<N;j++) a->rkey[j-N+NB]=a->fkey[j];

return TRUE;
}
/*-----*/
static void aes_ecb_encrypt(aes *a,char *buff)

```

```

{
    int i,j,k;
    WORD p[4],q[4],*x,*y,*t;

    for (i=j=0;i<NB;i++,j+=4)
    {
        p[i]=pack((BYTE *)&buff[j]);
        p[i]^=a->fkey[i];
    }

    k=NB;
    x=p; y=q;

    /* State alternates between x and y */
    for (i=1;i<a->Nr;i++)
    { /* Nr is number of rounds. May be odd. */
#ifdef LOTS_OF_MEMORY
        y[0]=a->fkey[k]^ftable[(BYTE)x[0]]^
            ftable1[(BYTE)(x[1]>>8)]^
            ftable2[(BYTE)(x[2]>>16)]^
            ftable3[x[3]>>24];
        y[1]=a->fkey[k+1]^ftable[(BYTE)x[1]]^
            ftable1[(BYTE)(x[2]>>8)]^
            ftable2[(BYTE)(x[3]>>16)]^
            ftable3[x[0]>>24];
        y[2]=a->fkey[k+2]^ftable[(BYTE)x[2]]^
            ftable1[(BYTE)(x[3]>>8)]^
            ftable2[(BYTE)(x[0]>>16)]^
            ftable3[x[1]>>24];
        y[3]=a->fkey[k+3]^ftable[(BYTE)x[3]]^
            ftable1[(BYTE)(x[0]>>8)]^
            ftable2[(BYTE)(x[1]>>16)]^
            ftable3[x[2]>>24];
#else
        y[0]=a->fkey[k]^ftable[(BYTE)x[0]]^
            ROTL8(ftable[(BYTE)(x[1]>>8)])^
            ROTL16(ftable[(BYTE)(x[2]>>16)])^
            ROTL24(ftable[x[3]>>24]);
        y[1]=a->fkey[k+1]^ftable[(BYTE)x[1]]^
            ROTL8(ftable[(BYTE)(x[2]>>8)])^
            ROTL16(ftable[(BYTE)(x[3]>>16)])^
            ROTL24(ftable[x[0]>>24]);
        y[2]=a->fkey[k+2]^ftable[(BYTE)x[2]]^
            ROTL8(ftable[(BYTE)(x[3]>>8)])^
            ROTL16(ftable[(BYTE)(x[0]>>16)])^
            ROTL24(ftable[x[1]>>24]);
        y[3]=a->fkey[k+3]^ftable[(BYTE)x[3]]^
            ROTL8(ftable[(BYTE)(x[0]>>8)])^
            ROTL16(ftable[(BYTE)(x[1]>>16)])^
            ROTL24(ftable[x[2]>>24]);
#endif
        k+=4;
        t=x; x=y; y=t; /* swap pointers */
    }

    /* Last Round */

    y[0]=a->fkey[k]^(WORD)fbsub[(BYTE)x[0]]^
        ROTL8((WORD)fbsub[(BYTE)(x[1]>>8)])^
        ROTL16((WORD)fbsub[(BYTE)(x[2]>>16)])^

```

```

    ROTL24((WORD)fbsub[x[3]>>24]);
    y[1]=a->fkey[k+1]^((WORD)fbsub[(BYTE)x[1]]^
    ROTL8((WORD)fbsub[(BYTE)(x[2]>>8]))^
    ROTL16((WORD)fbsub[(BYTE)(x[3]>>16]))^
    ROTL24((WORD)fbsub[x[0]>>24]);
    y[2]=a->fkey[k+2]^((WORD)fbsub[(BYTE)x[2]]^
    ROTL8((WORD)fbsub[(BYTE)(x[3]>>8]))^
    ROTL16((WORD)fbsub[(BYTE)(x[0]>>16]))^
    ROTL24((WORD)fbsub[x[1]>>24]);
    y[3]=a->fkey[k+3]^((WORD)fbsub[(BYTE)x[3]]^
    ROTL8((WORD)fbsub[(BYTE)(x[0]>>8]))^
    ROTL16((WORD)fbsub[(BYTE)(x[1]>>16]))^
    ROTL24((WORD)fbsub[x[2]>>24]);

    for (i=j=0;i<NB;i++,j+=4)
    {
        unpack(y[i],(BYTE *)&buff[j]);
        x[i]=y[i]=0; /* clean up stack */
    }
}
/*-----*/
static void aes_ecb_decrypt(aes *a,char *buff)
{
    int i,j,k;
    WORD p[4],q[4],*x,*y,*t;

    for (i=j=0;i<NB;i++,j+=4)
    {
        p[i]=pack((BYTE *)&buff[j]);
        p[i]^=a->rkey[i];
    }

    k=NB;
    x=p; y=q;

    /* State alternates between x and y */
    for (i=1;i<a->Nr;i++)
    { /* Nr is number of rounds. May be odd. */
#ifdef LOTS_OF_MEMORY
        y[0]=a->rkey[k]^rtable[(BYTE)x[0]]^
        rtable1[(BYTE)(x[3]>>8)]^
        rtable2[(BYTE)(x[2]>>16)]^
        rtable3[x[1]>>24];
        y[1]=a->rkey[k+1]^rtable[(BYTE)x[1]]^
        rtable1[(BYTE)(x[0]>>8)]^
        rtable2[(BYTE)(x[3]>>16)]^
        rtable3[x[2]>>24];
        y[2]=a->rkey[k+2]^rtable[(BYTE)x[2]]^
        rtable1[(BYTE)(x[1]>>8)]^
        rtable2[(BYTE)(x[0]>>16)]^
        rtable3[x[3]>>24];
        y[3]=a->rkey[k+3]^rtable[(BYTE)x[3]]^
        rtable1[(BYTE)(x[2]>>8)]^
        rtable2[(BYTE)(x[1]>>16)]^
        rtable3[x[0]>>24];
#else
        y[0]=a->rkey[k]^rtable[(BYTE)x[0]]^
        ROTL8(rtable[(BYTE)(x[3]>>8)])^
        ROTL16(rtable[(BYTE)(x[2]>>16)])^
        ROTL24(rtable[x[1]>>24]);

```

```

y[1]=a->rkey[k+1]^rtable[(BYTE)x[1]]^
    ROTL8(rtable[(BYTE)(x[0]>>8)])^
    ROTL16(rtable[(BYTE)(x[3]>>16)])^
    ROTL24(rtable[x[2]>>24]);
y[2]=a->rkey[k+2]^rtable[(BYTE)x[2]]^
    ROTL8(rtable[(BYTE)(x[1]>>8)])^
    ROTL16(rtable[(BYTE)(x[0]>>16)])^
    ROTL24(rtable[x[3]>>24]);
y[3]=a->rkey[k+3]^rtable[(BYTE)x[3]]^
    ROTL8(rtable[(BYTE)(x[2]>>8)])^
    ROTL16(rtable[(BYTE)(x[1]>>16)])^
    ROTL24(rtable[x[0]>>24]);
#endif
    k+=4;
    t=x; x=y; y=t;    /* swap pointers */
}

/* Last Round */
y[0]=a->rkey[k]^rsub[(BYTE)x[0]]^
    ROTL8((WORD)rsub[(BYTE)(x[3]>>8)])^
    ROTL16((WORD)rsub[(BYTE)(x[2]>>16)])^
    ROTL24((WORD)rsub[x[1]>>24]);
y[1]=a->rkey[k+1]^rsub[(BYTE)x[1]]^
    ROTL8((WORD)rsub[(BYTE)(x[0]>>8)])^
    ROTL16((WORD)rsub[(BYTE)(x[3]>>16)])^
    ROTL24((WORD)rsub[x[2]>>24]);
y[2]=a->rkey[k+2]^rsub[(BYTE)x[2]]^
    ROTL8((WORD)rsub[(BYTE)(x[1]>>8)])^
    ROTL16((WORD)rsub[(BYTE)(x[0]>>16)])^
    ROTL24((WORD)rsub[x[3]>>24]);
y[3]=a->rkey[k+3]^rsub[(BYTE)x[3]]^
    ROTL8((WORD)rsub[(BYTE)(x[2]>>8)])^
    ROTL16((WORD)rsub[(BYTE)(x[1]>>16)])^
    ROTL24((WORD)rsub[x[0]>>24]);

for (i=j=0;i<NB;i++,j+=4)
{
    unpack(y[i],(BYTE *)&buff[j]);
    x[i]=y[i]=0; /* clean up stack */
}

}
/*-----*/
mr_unsign32 aes_encrypt(aes* a, char *buff)
{
    int j,bytes;
    char st[16];
    mr_unsign32 fell_off;

/* Supported Modes of Operation */

    fell_off=0;
    switch (a->mode)
    {
    case MR_ECB:
        aes_ecb_encrypt(a,buff);
        return 0;
    case MR_CBC:
        for (j=0;j<4*NB;j++) buff[j]^=a->f[j];
        aes_ecb_encrypt(a,buff);

```



```
memcpy(&a->f[0], buff, 4*NB);
return 0;
```

```
case MR_CFB1:
case MR_CFB2:
case MR_CFB4:
    bytes=a->mode-MR_CFB1+1;
    for (j=0;j<bytes;j++) fell_off=(fell_off<<8)|a->f[j];
    for (j=0;j<4*NB;j++) st[j]=a->f[j];
    for (j=bytes;j<4*NB;j++) a->f[j-bytes]=a->f[j];
    aes_ecb_encrypt(a,st);
    for (j=0;j<bytes;j++)
    {
        buff[j]^=st[j];
        a->f[16-bytes+j]=buff[j];
    }
    return fell_off;
```

```
case MR_OFB1:
case MR_OFB2:
case MR_OFB4:
case MR_OFB8:
case MR_OFB16:
```

```
    bytes=a->mode-MR_OFB1+1;
    aes_ecb_encrypt(a,(a->f));
    for (j=0;j<bytes;j++) buff[j]^=a->f[j];
    return 0;
```

```
case MR_PCFB1: /* error propagating CFB */
case MR_PCFB2:
case MR_PCFB4:
    bytes=a->mode-MR_PCFB1+1;
    for (j=0;j<bytes;j++) fell_off=(fell_off<<8)|a->f[j];
    for (j=0;j<4*NB;j++) st[j]=a->f[j];
    for (j=bytes;j<4*NB;j++) a->f[j-bytes]=a->f[j];
    aes_ecb_encrypt(a,st);
    for (j=0;j<bytes;j++)
    {
        buff[j]^=st[j];
        a->f[16-bytes+j]=buff[j]^st[16-bytes+j];
    }
    return fell_off;
```

```
case MR_CTR: /*CTR calisma kipi sifre cozme kullanmaz!!!*/
    for (j=0;j<4*NB;j++) st[j] = a->f[j];
    memcpy(st, &a->f[0], 4*NB);
    aes_ecb_encrypt(a, st);//burada sayac+n sifreleniyor
    for (j=0;j<4*NB;j++) buff[j] ^= st[j];
    arttir(a); //a->f (IV) bir arttirilir.
    return 0;
```

```
case MR_OCB:
```

```
    return 0;
```

```
default:
    return 0;
```

```
    }
}
```



```
/*-----*/
mr_unsign32 aes_decrypt(aes *a, char *buff)
{
    int j, bytes;
    char st[16];
    mr_unsign32 fell_off;

    /* Supported modes of operation */
    fell_off = 0;
    switch (a->mode)
    {
        case MR_ECB:
            aes_ecb_decrypt(a, buff);
            return 0;
        case MR_CBC:
            memcpy(st, &a->f[0], 4*Nb);
            memcpy(&a->f[0], buff, 4*Nb);
            aes_ecb_decrypt(a, buff);
            for (j=0; j<4*Nb; j++)
            {
                buff[j]^=st[j];
                st[j]=0;
            }
            return 0;
        case MR_CFB1:
        case MR_CFB2:
        case MR_CFB4:
            bytes = a->mode - MR_CFB1 + 1;
            for (j=0; j<bytes; j++) fell_off = (fell_off << 8) | a->f[j];
            for (j=0; j<4*Nb; j++) st[j] = a->f[j];
            for (j=bytes; j<4*Nb; j++) a->f[j-bytes] = a->f[j];
            aes_ecb_encrypt(a, st);
            for (j=0; j<bytes; j++)
            {
                a->f[16-bytes+j] = buff[j];
                buff[j]^=st[j];
            }
            return fell_off;
        case MR_OFB1:
        case MR_OFB2:
        case MR_OFB4:
        case MR_OFB8:
        case MR_OFB16:
            bytes = a->mode - MR_OFB1 + 1;
            aes_ecb_encrypt(a, (a->f));
            for (j=0; j<bytes; j++) buff[j] = a->f[j];
            return 0;
        case MR_PCFB1: /* error propagating CFB */
        case MR_PCFB2:
        case MR_PCFB4:

            bytes = a->mode - MR_PCFB1 + 1;
            for (j=0; j<bytes; j++) fell_off = (fell_off << 8) | a->f[j];
            for (j=0; j<4*Nb; j++) st[j] = a->f[j];
            for (j=bytes; j<4*Nb; j++) a->f[j-bytes] = a->f[j];
            aes_ecb_encrypt(a, st);
            for (j=0; j<bytes; j++)
            {
                a->f[16-bytes+j] = buff[j]^st[16-bytes+j];
            }
    }
}
```

```

        buff[j]^=st[j];
    }
    return fell_off;
default:
    return 0;
}
}
/*-----*/
void aes_end(aes *a)
{ /* clean up */
    int i;
    for (i=0;i<NB*(a->Nr+1);i++)
        a->fkey[i]=a->rkey[i]=0;
    for (i=0;i<4*NB;i++)
        a->f[i]=0;
}
/*-----*/
anahtarstruct * ocb_aes_init(aes *a, char *sif_anahtar, /* anahtarstruct NULL veya ilklendirilmiş */
                             int anahtar_uznltk, /* AES anahtarı */
                             unsigned et_uznltk, /* Anahtar uzunlugu */
                             anahtarstruct *anahtar, /* Etiket uzunlugu */
                             char *iv) /* OCB anahtar yapisi */
{
    unsigned char tmp[16] = {0,};
    unsigned first_bit, last_bit, i;

    if (anahtar == NULL)
        anahtar = (anahtarstruct *)malloc(sizeof(anahtarstruct));
    if (anahtar != NULL) {
        memset(anahtar, 0, sizeof(anahtarstruct));

        /* AES anahtarları ilklendirilir.*/
        aes_init(a, MR_OCB, anahtar_uznltk, sif_anahtar, iv);

        /* L[i]-değerleri hesaplanır. L[0] = L */
        aes_ecb_encrypt(a, tmp);
        for (i = 0; i <= ON_HES_BLOKLAR; i++) {
            memcpy(anahtar->L + i, tmp, 16); /* tmp L[i]'ye kopyalanır */
            first_bit = tmp[0] & 0x80u; /* tmp, x ile carpılır */
            sola_kaydir(tmp);
            if (first_bit)
                tmp[15] ^= 0x87;
        }

        /* L_inv = L . x^{-1} hesaplanır */
        memcpy(tmp, anahtar->L, 16);
        last_bit = tmp[15] & 0x01;
        saga_kaydir(tmp);
        if (last_bit) {
            tmp[0] ^= 0x80;
            tmp[15] ^= 0x43;
        }
        memcpy(anahtar->L_inv, tmp, 16);

        /* Bu oturma için etiket uzunluğu sabitlenir */
        anahtar->et_uznltk = et_uznltk;
    }
}

```



```
return anahtar;
}
/*-----*/
void ocb_aes_sifrele(aes *a, anahtarstruct *anahtar, /* ilklendirilmis anahtarstruct */
    char *IV, /* 16-byte IV */
    char *pt, /* Acikmesaj Arabelleği */
    unsigned pt_uznlik, /* pt'nin sekizli uzunlugu */
    char *ct, /* Sifrelimesaj Arabelleği */
    char *etiket) /* Etiket Arabelleği */
{
    unsigned i; /* Bkok sayaci */
    block tmp, tmp2; /* Gecici arabellek */
    block *pt_blk, *ct_blk; /* pt ve ct nin adreslerini tutar */
    block Ofset; /* Gecerli blok icin Ofset (Z[i]) */
    block saglama; /* Etiket hesabi icin saglama */

    /*
     * Ilklendirmeler
     */
    i = 1; /* ilk blokdan baslar */
    pt_blk = (block *)pt - 1; /* pt_blk[1]'un ilk blogu gosterir */
    ct_blk = (block *)ct - 1; /* ct_blk[1]'un ilk blogu gosterir */
    memset(saglama, 0, 16); /* saglama sifirlanir */

    /* Z[0] hesaplanir */
    modul2Topla(Ofset, IV, anahtar->L);
    aes_ecb_encrypt(a, Ofset);

    /*
     * bloklar 1 .. m-1 islenir
     */
    while (pt_uznlik > 16) {

        /* Ofset guncellenerek (Z[i-1] den Z[i]) bulunur */
        modul2Topla(Ofset, anahtar->L + ntz(i), Ofset);

        /* Acikmesaj blogu Z[i] ile modul 2 toplanir */
        modul2Topla(tmp, Ofset, pt_blk + i);

        /* Blok sifrelenir */
        aes_ecb_encrypt(a, tmp);

        /* tekrar Z[i] ile modul 2 toplanarak sifreli mesaj pointer'a yazilir. */
        modul2Topla(ct_blk + i, Ofset, tmp);

        /* saglama guncellenir */
        modul2Topla(saglama, saglama, pt_blk + i);

        /* dongu degiskenleri guncellenir */
        pt_uznlik -= 16;
        i++;
    }

    /*
     * m. blok islenir.
     */

    /* Ofset guncellenerek (Z[i-1] den Z[i]) bulunur */
    modul2Topla(Ofset, anahtar->L + ntz(i), Ofset);
```

```
/* L .  $x^{-1}$  ve  $Z[m]$  modul 2 toplanir */
modul2Topla(tmp, Ofset, anahtar->L_inv);
```

```
/* son blogun ikil uzunlugu eklenir */
tmp[15] ^= (pt_uznlik << 3);
```

```
/* sifrelenir. */
aes_ecb_encrypt(a,tmp);
```

```
/* 'pt' block-cipher cikisi ile modul 2 toplanir */
memcpy(tmp2, pt_blk + i, pt_uznlik);
modul2Topla(tmp2, tmp2, tmp);
memcpy(ct_blk + i, tmp2, pt_uznlik);
```

```
/* saglama bilgisine acikmesajin pt_uznlik bilgisi ile */
/* sifreli mesajin son (16 - pt_uznlik) sekizlisi eklenir */
memcpy(tmp, pt_blk + i, pt_uznlik);
modul2Topla(saglama, saglama, tmp);
```

```
/*
 * Etiket hesaplanir
 */
```

```
modul2Topla(saglama, saglama, Ofset);
aes_ecb_encrypt(a,saglama);
memcpy(etiket, saglama, anahtar->et_uznlik);
```

```
}
/*-----*/
```

```
int /* Etiket yanlissa 0 doner */
ocb_aes_sifreCoz(aes *a, anahtarstruct *anahtar, /* Ilklendirilmis anahtarstruct */
char *IV, /* 16-byte IV */
char *ct, /* Acikmesaj Arabellegi */
unsigned ct_uznlik, /* ct'nin sekizli uzunlugu */
char *pt, /* Sifrelimesaj Arabellegi */
char *etiket) /* Etiket Arabellegi */
```

```
{
unsigned i; /* Bkok sayaci */
block tmp, tmp2; /* Gecici arabellek */
block *ct_blk, *pt_blk; /* pt ve ct nin adreslerini tutar */
block Ofset; /* Gecerli blok icin Ofset ( $Z[i]$ ) */
block saglama; /* Etiket hesabi icin saglama */
```

```
/*
 * Ilklendirme
 */
```

```
i = 1; /* ilk blokdan baslar */
ct_blk = (block *)ct - 1; /* pt_blk[1]'un ilk blogu gosterir */
pt_blk = (block *)pt - 1; /* ct_blk[1]'un ilk blogu gosterir */
memset(saglama, 0, 16); /* saglama sifirlanir */
```

```
/*  $Z[0]$  hesaplanir */
modul2Topla(Ofset, IV, anahtar->L);
aes_ecb_encrypt(a,Ofset);
```

```
/*
 * bloklar 1 .. m-1 islenir
 */
while (ct_uznlik > 16) {
```

```
/* Ofset guncellenerek (Z[i-1] den Z[i]) bulunur */
modul2Topla(Ofset, anahtar->L + ntz(i), Ofset);
```

```
/* sifrelimesaj blogu ile Z[i] modul 2 toplanir */
modul2Topla(tmp, Ofset, ct_blk + i);
```

```
/* bir sonraki sifreli mesaj desifrelenir */
aes_ecb_decrypt(a,tmp);
```

```
/* Z[i] ile tekrar modul 2 toplanarak, acikmesaj pointer'a yazilir */
modul2Topla(pt_blk + i, Ofset, tmp);
```

```
/* saglama guncellenir */
modul2Topla(saglama, saglama, pt_blk + i);
```

```
/* dongu degiskenleri guncellenir */
ct_uznlik -= 16;
i++;
```

```
}
```

```
/*
 * m. blok islenir.
 */
```

```
/* Ofset guncellenerek (Z[i-1] den Z[i]) bulunur */
modul2Topla(Ofset, anahtar->L + ntz(i), Ofset);
```

```
/* L .  $x^{-1}$  ve Z[m] modul 2 toplanir */
modul2Topla(tmp, Ofset, anahtar->L_inv);
```

```
/* son blogun ikil uzunlugu eklenir */
tmp[15] ^= (ct_uznlik << 3);
```

```
/* sifrelenir. */
aes_ecb_encrypt(a,tmp);
```

```
/* son sifreli mesaj blogu, C[m]'den */
memset(tmp2, 0, 16);
memcpy(tmp2, ct_blk + i, ct_uznlik);
modul2Topla(tmp, tmp2, tmp);
memcpy(pt_blk + i, tmp, ct_uznlik);
```

```
/* yukaridaki modul 2 toplamadan sonra, tmp, acikmesajin ct_uznlik sekizkine sahiptir */
/* (16 - ct_uznlik) block-cipher sekizlisi, saglama inin kullanılacaktır */
modul2Topla(saglama, saglama, tmp);
```

```
/*
 * etiket ve dogrulugu hesaplanir
 */
```

```
modul2Topla(saglama, saglama, Ofset);
aes_ecb_encrypt(a,saglama);
return (memcmp(etiket, saglama, anahtar->et_uznlik) == 0 ? 1 : 0);
```

```
}
```

```
/*-----*/
```

```
anahtarstruct *ocb_bitir(anahtarstruct *anahtar)
```

```
{
```

```
if (anahtar) {
    memset(anahtar, 0, sizeof(anahtarstruct));/*kullanilan anahtarin free yapilmadan */
```

```

        free(anahtar);
    }
    return NULL;
}
/*-----*/
int main()
{
    aes a;
    anahtarstruct *anahtarlar = NULL;
    int i,j,nk, etiket_degeri;
    BYTE y,x,m;
    char anahtar[32];
    char mesaj[MESAJ_UZNLK]={0,};
    char sifre[MESAJ_UZNLK]={0,};
    char etiket[ETIKET_UZNLK];
    char iv[16];
    char sec[10];

    struct time t;

    for (i=0;i<32;i++) iv[i]=0;
    iv[15]=1;
    for (i=0;i<32;i++) anahtar[i]=i;
    for (i=0;i<MESAJ_UZNLK;i++) mesaj[i]=i;

    printf("Lutfen sifreleme modunu belirleyiniz :>");
    scanf("%s", &sec);

    printf("IV=\t\t");
    for (i=0;i<4*NB;i++) printf("%02x",iv[i]);
    printf("\n");

    gettimeofday(&t);
    printf("Sifreleme baslama zamani: %2d:%02d:%02d.%02d\n",
        t.ti_hour, t.ti_min, t.ti_sec, t.ti_hund);

    if(!strcmp(sec, "OCB")){

        for (nk=16;nk<=32;nk+=8)
        {
            printf("\nAnahtar Size= %d bits\n",nk*8);          printf("Anahtar=\t");

            for (i=0;i<4*NB;i++) printf("%02x",anahtar[i]);
            printf("\n");
            printf("A. Mesaj=\t");
            for (i=0;i<MESAJ_UZNLK;i++) printf("%02x",(unsigned char)mesaj[i]);
            printf("\n");

            anahtarlar = ocb_aes_init(&a, anahtar, nk, ETIKET_UZNLK, NULL, iv);
            ocb_aes_sifrele(&a, anahtarlar, iv, mesaj, MESAJ_UZNLK, sifre, etiket);

            printf("Sifrele=\t");
            for (i=0;i<MESAJ_UZNLK;i++) printf("%02x",(unsigned char)sifre[i]);
            printf("\n");

            printf("Etiket=\t\t");
            for (i=0;i<4*NB;i++) printf("%02x",(unsigned char)etiket[i]);
            printf("\n");
        }
    }
}

```





```
for (i=0;i<MESAJ_UZNLK;i++) mesaj[i]=i;

etiket_degeri=ocb_aes_sifreCoz(&a, anahtarlar, iv, sifre, MESAJ_UZNLK,
mesaj, etiket);
if (!etiket_degeri) printf("\nSifre cozme islemi basarisiz!!!\n");

printf("A. Sifre Coz=\t");
for (i=0;i<MESAJ_UZNLK;i++) printf("%02x", (unsigned char)mesaj[i]);
printf("\n");

ocb_bitir(anahtarlar);
aes_end(&a);
} /* for */
} /* if */

else if(!strcmp(sec, "CTR")){

for (nk=16; nk<=32; nk+=8)
{
printf("\nAnahtar Size= %d bits\n", nk*8);
if (!aes_init(&a, MR_CTR, nk, anahtar, iv))
{
printf("Failed to Initialize\n");
return 0;
}

printf("Anahtar=\t");
for (i=0; i<4*NB; i++) printf("%02x", (unsigned char)anahtar[i]);
printf("\n");
printf("Acik Mesaj=\t");
for (i=0; i<MESAJ_UZNLK; i++) printf("%02x", (unsigned char)mesaj[i]);
printf("\n");

for (i=0; i<MESAJ_UZNLK; i+=16) aes_encrypt(&a, mesaj+i);
printf("Sifrele=\t");
for (i=0; i<MESAJ_UZNLK; i++) printf("%02x", (unsigned char)mesaj[i]);
printf("\n");

aes_reset(&a, MR_CTR, iv);
for (i=0; i<MESAJ_UZNLK; i+=16) aes_encrypt(&a, mesaj+i);
printf("Sifre Coz=\t");
for (i=0; i<MESAJ_UZNLK; i++) printf("%02x", (unsigned char)mesaj[i]);
printf("\n");

aes_end(&a);

} /* for */
} /* else if */
else if(!strcmp(sec, "CBC")){
for (nk=16; nk<=32; nk+=8)
{
printf("\nAnahtar Size= %d bits\n", nk*8);
if (!aes_init(&a, MR_CBC, nk, anahtar, iv))
{
printf("Failed to Initialize\n");
return 0;
}

printf("Anahtar=\t");
for (i=0; i<4*NB; i++) printf("%02x", (unsigned char)anahtar[i]);
```

**İ.C. YÜKSEKÖĞRETİM KURULU
DOKÜMANTASYON MERKEZİ**



```
        printf("\n");
        printf("Acik Mesaj=\t");
        for (i=0;i<MESAJ_UZNLK;i++) printf("%02x",(unsigned char)mesaj[i]);
        printf("\n");

        for (i=0; i<MESAJ_UZNLK; i+=16) aes_encrypt(&a,mesaj+i);
        printf("Sifrele=\t");
        for (i=0;i<MESAJ_UZNLK;i++) printf("%02x",(unsigned char)mesaj[i]);
        printf("\n");
        aes_reset(&a,MR_CBC,iv);

        for (i=0; i<MESAJ_UZNLK; i+=16) aes_decrypt(&a,mesaj+i);
        printf("Sifre Coz=\t");
        for (i=0;i<MESAJ_UZNLK;i++) printf("%02x",(unsigned char)mesaj[i]);
        printf("\n");

        aes_end(&a);
    } /* for */
} /*else if*/

gettime(&t);
printf("Sifreleme bitis zamani: %2d:%02d:%02d.%02d\n",
        t.ti_hour, t.ti_min, t.ti_sec, t.ti_hund);

scanf("%d",&i);
return 0;
}
```