T.C.
MALTEPE ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

BİLGİSAYAR GÜVENLİĞİ ÜZERİNE BİR ARAŞTIRMA VE ŞİFRELEME-DEŞİFRELEME ÜZERİNE UYGULAMA

Tarık TUNCAL

Yüksek Lisans Tezi

Tez Danışmanı
Prof. Dr. Şaban EREN

İSTANBUL – 2008

T.C.
MALTEPE ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

BİLGİSAYAR GÜVENLİĞİ ÜZERİNE BİR ARAŞTIRMA VE ŞİFRELEME-DEŞİFRELEME ÜZERİNE UYGULAMA

YÜKSEK LİSANS TEZİ

Tarık TUNCAL

Tez Danışmanı
Prof. Dr. Şaban EREN

İSTANBUL – 2008

Bu tez çalışması, Maltepe Üniversitesi Fen Bilimleri Enstitüsü Yönetim Kurulu'nun 28 / 04 / 2008 tarih ve 2008 / 15 sayılı kararıyla oluşturulan jüri tarafından ***Bilgisayar Mühendisliği Yüksek Lisansı Tezi*** olarak kabul edilmiştir.

JÜRİ

Prof. Dr.Şaban EREN
Danışman

Prof. Dr. İlhami Yavuz
Üye

Yrd. Doç. Dr. Şahin Uyaver
Üye

ÖZET

Yüksek Lisans Tezi, Bilgisayar Güvenliği Üzerine Bir Araştırma ve Şifreleme-Deşifreleme Üzerine Uygulama, T.C. Maltepe Üniversitesi, Fen Bilimleri Enstitüsü, Bilgisayar Mühendisliği Anabilim Dalı.

Tarihte asırlar boyu insanlar kendilerine rakip gördüğü kişilerin eline kendileri için özel saydıkları bilgilerin (gerek savaş için olsun, gerek de kişisel amaçlı olsun) geçmemesi için çabalamışlardır. Böylece günümüze göre ilkel sayılabilecek yöntemlerle şifrelemenin temelleri atılmıştır.

Günümüzde gelişen teknolojiyle beraber bilgi paylaşımı güvenliği hassas bir noktaya gelmiştir. Öyle ki, insanlar makineler sayesinde insan ömrünün yetmeyeceği karmaşıklıktaki şifreleme yöntemlerini insan ömrüne kıyasla kısa sürede kırıp ulaşılmak istenen bilgiye erişmeyi başarmaktadırlar.

Eliptik eğrilerin şifrelemede kullanılmaya başlanması sayesinde geliştirilen eliptik eğri kriptosistemlerin önerdiği anahtar uzunlukları yardımıyla daha küçük anahtar uzunluğuna sahip anahtarlarlarla daha etkili bir durum yaratılmıştır. 160 bitlik katsayılı bir eliptik eğri sistemi 1024 bitlik katsayılı RSA ile aynı seviyede kriptografik güvenlik sağlamaktadır. Anahtar boyutunun küçük olması işlem kolaylığı da sağlamıştır.

Birinci bölümde kriptolojiye kısa bir giriş yapılmıştır. Kriptografinin tanımı yapılarak kısaca şifrelemeden bahsedilmiştir. Şifrelemede kullanılan terminoloji incelenerek kriptografi tarihinden maddeler halinde bahsedilmiştir.

İkinci bölümde şifrelemede yararlanılabilecek bazı matematiksel alt yapıya kısaca değinilmiştir.

Üçüncü bölümde Klasik Kriptografi başlığı altında bazı basit kriptosistemler incelenmiştir. Bazı basit kriptosistemlere ait kriptoanalizlerden bahsedilmiştir.

Dördüncü bölümde kriptografi çeşitleri simetrik (gizli) ve asimetrik (açık) anahtarlı kriptografiler olmak üzere iki başlık altında incelenmiştir. Sonra kriptoanalizin amaç ve yöntemlerinden bahsedilmiştir.

Beşinci bölümde simetrik kriptosistemlerin genel yapısından kısaca bahsedilerek simetrik kriptosistemlerde kullanılan bazı algoritmalar incelenmiştir.

Altıncı bölümde asimetrik kriptosistemlerin genel yapısına kısaca değinilerek asimetrik kriptosistemlerde kullanılan bazı algoritmalardan bahsedilmiştir.

Yedinci bölümde eliptik eğriler kısaca incelenmiştir. Eliptik eğri üzerinde yapılacak işlemlerden kısaca bahsedilmiştir.

Sekizini bölümde eliptik eğri tabanlı kriptografi kısaca incelenmiştir. Sonra eliptik eğrilerde kullanılan bazı aritmetik işlemlerden bahsedilmiştir.

Dokuzuncu bölümde Base64 kodlaması kullanılarak yapılan simetrik şifreleme programı aracılığıyla örnek metinler üzerinde şifreleme ve deşifreleme işlemleri gerçekleştirilmiştir.

Bu tez 2008 yılında yapılmış olup 179 sayfadan oluşmaktadır.

Anahtar Kelimeler: Kriptografi, şifreleme, deşifreleme, kriptoanaliz, kriptosistem, gizli anahtar, açık anahtar, eliptik eğri

ABSTRACT

Master Thesis, A Research on Computer Security and an Application of Encryption-Decryption, T.C. Maltepe University, Graduate School of Natural and Applied Sciences, Department of Computer Engineering.

Throughout the history, people have been in search of ways to prevent their private data (both for wartime conditions and personal aims) from the possession of others whom they consider as their rivals. Therefore, these methods which might be regarded as primitive today have laid the foundation for modern cryptography.

Nowadays, with the developments in technology, the security concerns for private data sharing has reached to a very critical point. Besides, people manage to access the data –having complex encryption systems which cannot be decrypted in a lifespan– in a relatively short time with the help of computers.

The key lengths suggested by an Elliptic Curve Cryptosystem, developed by means of using Elliptic Curves in Cryptography, have resulted in the use of keys with smaller lengths which prove to be more efficient. An Elliptic Curve System of 160 bit constant provides the same level of security as DSA or RSA of 1024 bit constant. The smaller size of the key ensures the operational simplicity as well.

The first chapter consists of a brief introduction to cryptography. After this definition of cryptography, encryption, and with the terminological research on encryption, the history of cryptography is mentioned.

The second chapter refers to certain mathematical infrastructures which might be helpful.

In chapter three, under the heading of Classic Cryptography, certain simple cryptosystems and the cryptanalysis of these simple cryptosystems are studied.

In chapter four, cryptography methods are studied under two headings, namely symmetric (private) key and asymmetric (public) key cryptographies; this precedes the purpose and methods of cryptanalysis.

Chapter five consists of a brief explanation of the overall structure of private key cryptosystems and certain algorithms used in private key cryptosystems.

Chapter six refers to a brief explanation of the overall structure of public key cryptosystems and certain algorithms used in public key cryptosystems.

In chapter seven, elliptic curves are briefly studied. There is also a brief explanation concerning the operations on elliptic curve.

Chapter eighth, briefly explains elliptic curve based cryptography. After that, certain arithmetical operations used in the elliptic curves are studied.

In chapter nine, by means of symmetric encryption program which is formed by using Base 64 coding is introduced and model texts are encrypted and decrypted.

This thesis has been completed in 2008 and consists of 179 pages.

Keywords: Cryptography, encryption, decryption, cryptanalysis, cryptosystem, private key, public key, elliptic curve

TEŞEKKÜR

İlk olarak, tezime bilgisiyle yön veren tez danışmanım Prof. Dr. Şaban Eren'e ilgisi, öğütleri ve sabrı için teşekkür ederim. Hoş görüsüne ve rehberliğine çok şey borçluyum.

Ayrıca, bir süre destek olup değerli fikirlerini paylaşan Prof. Dr. Murat Esin ve Prof. Dr. Emin Anarım'a; uygulamada yardımlarından dolayı Sn. Mete Eminağaoğlu'na; çocukluğumdan beri bana yol gösteren Sn. Hüsnü Karagözoğlu'na ve Sn. Semra Demiray'a; manevi destekleri için arkadaşlarım Serkan Kemalbay'a ve Bahadır Karasulu'ya teşekkür ederim.

Son olarak hayatım boyunca hep yanımda olup verdikleri her türlü destekleri için babam Mehmet Tuncal'a, annem Nilgül Tuncal'a ve ablam İ. Neşe Tuncal'a teşekkür ederim.

İÇİNDEKİLER

Sayfa

KISALTMALAR

Kısaltma	İngilizcesi	Türkçesi
AES	Advanced Encryption Standard	İleri Şifreleme Standardı
ATM	Automatic Teller Machine	Otomatik Vezne Makinesi
DES	Data Encryption Standart	Veri Şifreleme Standardı
ECC	Elliptic Curve Cryptography	Eliptik Eğri Şifreleme
FIPS	Federal Information Processing Standard	Federal Bilgi İşleme Standardı
IDEA	International Data Encryption Algorithm	Uluslararası Veri Şifreleme Algoritması
LFSR	Linear Feedback Shift Register	Lineer Geribeslemeli Kaydırma Yazmacı
MAC	Message Autentication Code	Mesaj Doğrulama Kodu
NIST	National Institute of Standards and Technology	Ulusal Standartlar ve Teknoloji Enstitüsü
RSA	Rivest Shamir Adelman	Rivest Shamir Adelman
PKDS	Public-Key Distribution Schemes	Açık Anahtar Dağıtım Şeması
PKS	Public Key Schemes	Açık Anahtar Şeması

ŞEKİLLER

TABLOLAR

1. KRİPTOGRAFİ

1.1. Giriş

Kriptografi (kriptoloji, Yunanca κρυπτός (kryptós) "gizli" ve γράφω (gráfo) "yazmak" fiilinden türetilmiştir), mesaj gizliliği çalışmasıdır. Günümüzde, bilginin matematiksel çalışması ve özellikle bir yerden başka bir yere taşınması olarak bilişim kuramının bir bölümü (dalı) olmuştur. [5]

Şifreleme işlemi, en geleneksel kullanımda bilgileri belirli kişilerden saklama amacı güder; ama genellikle mesajın varlığını değil. Bu düşünce, rakip ve düşman kavramlarını gündeme getirir. Bu şartlarda, bilginin saklanması genellikle bir mesajın şifrelenmesi; yani, rakibin eline geçse bile (zorla ele geçirilmesi, mesajın bir kopyasına ulaşılması ya da mesajın gönderildiği kanalın dinlenilmesi) rakip tarafından kolayca anlaşılmayacak veya çözülemeyecek bir şekle dönüştürülmesi anlamındadır. Bu saklama işlemi, mesajı alması beklenen yetkili tarafından şifrelenmiş mesajı kolayca ve kısa sürede çözülme özelliğine sahip olmalıdır.

Şifreleme, modern olarak mesaj gizli olsun ya da olmasın mesajı göndereni doğrulamada (authentication) kullanılır; mesajı alan kişi, mesajın düşündüğü kişiden geldiğini ve bir başkası tarafından değiştirilmediğinden emin olabilmelidir. Bir mesajın değiştirilip değiştirilmediği problemi, veri bütünlüğü (data integrity) problemi olarak bilinir.

Şifreleme, bilgisayar bilimlerine de katkıda bulunur (özellikle, erişim denetimi ve bilgi gizliliği gibi şeyler için bilgisayar ve ağ güvenliğinde kullanılan tekniklerde). Şifreleme, günlük yaşamda kullanılan pek çok uygulamada da kullanılır; örneğin, ATM kartları güvenliği, bilgisayar şifreleri ve tümü şifrelemeye dayalı elektronik ticaret. [5]

1.2. Terminoloji

Bir kriptosistem ya da şifre, bir mesajın yetkili kişi(ler) dışında okunmasını engelleyecek matematiksel bir dönüşümdür. Şifreleme (encryption) süreci mesajı gönderen kişi tarafından gerçekleştirilir ve yola çıkan mesajı dinleyebilecek herhangi bir yetkisiz kişinin mesajı anlamasını engellemeyi hedefler. Deşifreleme süreci şifreleme sürecinden geçmiş mesajı alan yetkili kişi tarafından gerçekleştirilir ve gizlenmiş mesajdan (şifrelimetin - ciphertext) orjinal mesajın (düzmetin - plaintext) elde edilmesini hedefler. Buradaki en önemli nokta, şifrelenmiş metnin, eğer bu metin bir şekilde istenmeyen bir kişinin/kişilerin eline geçerse çok büyük zorluklarla çözülebilecek şekilde şifrelenmiş olması gerekliliğidir. Simetrik kriptosistemlerde bu özellik gönderici ve alıcının anahtar olarak isimlendirilen bir gizli bilgiyi ortaklaşa kullanmaları ile gerçekleştirilir. Söz konusu anahtara sahip olmayıp da mesajı ele geçiren biri mesajı çözmekte çok büyük güçlük çekecektir.

Şifreleme/deşifreleme süreçlerindeki tüm detayların gizliliğinin en iyi yaklaşım olacağı düşünülebilir. Maalesef şifreleme tarihi, ilk bakışta akla yatkın gibi görünen bu fikrin ne kadar zararlı olduğunun örnekleri ile doludur. Günümüzdeki standart bakış açısına göre şifreleme/deşifreleme algoritmasının; yani, kriptosistemin detayları kamuoyuna açık olmalı, sadece ve sadece *anahtar* gizli tutulmalıdır. Bu bakış açısı Kerckhoff Prensibi[1] olarak da bilinir. Söz konusu prensip artık o kadar yaygınlaşmıştır ve kabul görmüştür ki, bu prensibe göre kriptosistemler tasarlanabildiği öne sürülerek, bu prensibi ihlal eden kriptosistemler kabul edilmez bulunmaya başlanmıştır.

Kerckhoff Presibi'ne uymanın getirdiği en önemli faydalardan biri büyük ölçekli iletişimi pratikte mümkün kılan ve kamuoyuna açık algoritmaların standartlaştırmasıdır

[1] Auguste Kerckhoff (1835-1903). 1883'te yayınlanan, 64 sayfalık La Cryptographie Militaire (Askeri Kriptografi) adlı eseri yazan ve modern kriptografiye önemli katkılarda bulunan Hollanda doğumlu araştırmacı.

Klasik şifre ile ifade edilen elektronik bilgisayarların ortaya çıkmasından önce geliştirilen veya kullanılan şifre ifade edilmektedir. ENIGMA[2] ve II.Dünya Savaşı esnasında kullanılan benzerleri elektronik-öncesi kriptografi alanının doruktaki temsilcileridir.

Simetrik şifre, şifreleme için kullanılan anahtarın (encryption key) şifreyi çözmek (decryption key) için kullanılan anahtara denk olduğu şifredir. Uç durumlarda şifreleme ve şifre çözme için kullanılan anahtarlar birbirinin aynısıdır; yani, gönderici ve alıcı *aynı gizi* paylaşırlar. Tüm klasik şifreler bu şekilde tasarlanmıştır. Aslında, 1975 yılına dek başka türde bir şifre tasarlanabileceği düşünülmüyordu. 1975-1978 yılları arasında Merkle ve Hellman asimetrik şifre sistemlerine dair ilk fikirleri ve tasarımları ortaya koydular[3]. Bu modele göre şifreleme anahtarı (aynı zamada açık anahtar (public key) olarak anılır ve kamuoyuna açıktır), şifreyi çözme anahtarına (aynı zamanda gizli anahtar (private key) olarak anılır ve sadece şifreyi çözmeye yetkili kişinin bilgisindedir) dair çok az bilgi verir ve tersi de doğrudur. Asimetrik şifreler aynı zamanda matematiksel analiz bakımından da oldukça ilginçtirler; çünkü bu şifrelerin çözümünün zorluğu doğrudan bazı önemli matematiksel problemlerin çözümünün zorluğu ile bağlantılıdır. Simetrik şifreler, matematiksel bakımından, asimetrik şifrelere kıyasla daha basittirler.

Bir kriptosisteme karşı gerçekleştirilen saldırı, şifrelenmiş metni anahtar kullanmadan çözmeye çalışma işlemidir. Dört temel saldırı kategorisi vardır:

- Sadece şifrelimetin: Bu durumda şifreçözücünün elinde şifrelimetnin bir kısmına sahiptir ancak düzmetin ya da anahtara ilişkin hiçbir bilgiye sahip değildir. Saldırının iki aşaması vardır: *belli* bir mesajı çözmek veya anahtarın ne olduğunu bularak *tüm* mesajları çözmek.

[2] II. Dünya Savaşı sırasında Almanlar tarafında geliştirilen, rotor tabanlı, çok sayıda elektriksel olarak üretilmiş alfabe kullanabilen şifreleme cihazı/sistemi. Polonyalı matematikçilerin önbilgilerinin yardımı ile Londra'daki Bletchley Park'ta, matematikçi Alan Turing önderliğindeki Ultra kod adlı çalışma sonucunda çözülmüştür.

[3] ABD'deki Ulusal Güvenlik Ajansı, NSA (National Security Agency) bu tip şifre sistemini 1966 yılında geliştirdiğini iddia etmektedir ancak bununla ilgili bir kanıt yoktur.

- Bilinen düzmetin: Bu durumda şifreçözücü düzmetnin bir kısmına (veya tamamına) ve buna karşılık gelen şifrelimetne sahiptir. Hedef anahtarın ne olduğunu bulmaktır. Pek çok 'klasik' şifre bu tip saldırı karşısında çok zayıftır.
- Seçilmiş düzmetin: Şifreçözücünün elinde belli sayıda düzmetin seçme ve buna karşılık gelen şifrelimetni görme şansı vardır. Hedef anahtarı bulmaktır. Klasik şifreler bu tip saldırı karşısında çok zayıftır.
- Şifreleme anahtarı: Bu saldırı tipi, şifreleme anahtarının şifreçözme anahtarına dair çok az bilgi içerdiği asimetrik şifre sistemlerine yöneliktir. Hedef ilgili şifrelimetni ele geçirmeden önce şifreçözme anahtarına dair bilgi elde edebilmektir. [4]

1.3. Kriptografinin Kısa Tarihi

Kriptoloji çok eski çağlardan beri insanoğlu tarafından kullanılmaktadır. Bu tarihçeye kısaca bakacak olursak:

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

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

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

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

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

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

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

- 1917'de Joseph Mauborgne ve Gilbert Vernam mükemmel şifreleme sistemi olan "one-time pad"'i buldular.

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

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

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

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

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

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

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

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

- 1990'da Xuejia Lai ve James Massey: IDEA algoritmasını buldular.

- 1991'de Phil Zimmerman: PGP sistemini geliştirdi ve yayınladı.

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

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

- 2001'de NIST'in yarışmasını kazanan Belçikalı Joan Daemen ve Vincent Rijmen'e ait Rijndael algoritması, AES (Advanced Encryption Standard) adıyla standart haline getirildi. [17]

2. SAYI TEORİSİNE GİRİŞ

Bu bölümde kriptolama algoritmalarının matematik modellemesinde kullanılan modüler aritmetik kavramları üzerinde kısaca durulacaktır.

2.1. Modüler Aritmetik

Moduler aritmetik, günlük hayatta zaman zaman karşımıza çıkmaktadır. Moduler aritmetik, asimetrik şifrelerin oluşturulmasında ve kırılmasında yardımcı olabilecek bir dizi sayısal mantığı barındıran konu bütünlüğüdür.

Modüler aritmetiğin tanımı Tanım 2.1 ile verilmiştir.

Tanım 2.1: *a, b* ve *n* tam sayıları ve $n \neq 0$ şartı için, eğer *a* ve *b* 'nin farkı *n* 'nin *k* katı kadarsa

$$a - b = k \,.\, n \tag{2.1}$$

veya

$$a \equiv b \bmod n \tag{2.2}$$

şeklinde gösterilebilir.

Teorem 2.1: a_1, a_2 ve *n* tam sayıları ve $n \neq 0$ şartı için,

$$(a_1 \; op \; a_2) \bmod n \equiv [(a_1 \bmod n) \; op \; (a_2 \bmod n)] \bmod n$$

denkliği gösterilebilir, burada *op*, " + " veya " × " şeklinde bir operatör (işlemci) olabilir.

- Bir $a = b \bmod n$ eşitliği, *a* ve *b* aynı *n* ile bölündüğünde aynı kalanı verdiklerini ifade eder.
  - 2.2 denkleminde genellikle $0 \leq b \leq n-1$ 'dir.
  - 2.2 denkleminde *b* 'ye *a* mod *n* 'nin kalanı denir.
  - Örneğin, $100 = 34 \bmod 112$, $2 = 9 \bmod 7$

- Tamsayı modulo n ile yapılan aritmetikte bütün sonuçlar 0 ve n arasında olur.

Modüler aritmetikle ilgili bazı özellikler

2.1.1. Toplama

(a mod n) + (b mod n) = (a+b) mod n

2.1.2. Çıkartma

(a mod n) – (b mod n) = [a+(-b)] mod n

2.1.3. Çarpma

(a mod n)×(b mod n) = (a×b) mod n

- Tekrarlanan toplamdan türetilir

- Ne a ne de b sıfır değil iken a×b=0 olabilir
    - Örnek, 2×5 mod 10

2.1.4. Bölme

a/b mod n

- b nin tersi ile çarpmak gibidir: $a/b = a.b^{-1} \bmod n$

- eğer n asal ise $b^{-1} \bmod n$ vardır. $b.b^{-1} = 1 \bmod n$
    - örnek $2.3=1 \bmod 5$ bu nedenle $4/2=4.3=2 \bmod 5$ dir.

- Tamsayılarla modulo n toplama ve çarpma aşağıdaki kurallar ile bir geçişli halkadır.

2.1.5. Birleşme (Associativity)

[(a+b)+c] mod n = [a+(b+c)] mod n

2.1.6. Geçişlilik (Commutativity)

$(a+b) \bmod n = (b+a) \bmod n$

2.1.7. Dağılma (Distributivity)

$[(a+b)\times c] \bmod n = [(a\times c)+(b\times c)] \bmod n$

- Aynı zamanda, indirgeme tamsayılar halkasından tamsayı modulo n'lerin halkasına bir homomorfizm olduğu için, bir işlem ve sonra modulo n'yi indirgeyip indirgemeyeceği veya indirgedikten sonra yapacağı işlem seçilebilir.
    - $(a \pm b) \bmod n = [(a \bmod n) \pm (b \bmod n)] \bmod n$
    - $(a\times b) \bmod n = [(a \bmod n) \times (b \bmod n)] \bmod n$

- Eğer n, p doğal sayısı olmaya zorlanırsa bu form bir **Galois Field modulo p** ve **GF(p)** ile gösterilir ve bütün tamsayı aritmetiğindeki normal kurallar geçerlidir.

[6]

2.2 Asal Sayılar

Asal sayılar, Açık-anahtarlı kripto sistemlerinde büyük rol oynarlar. Asal sayılarda karşımıza çıkan en önemli fonksiyonlar, asal bir sayının oluşturulması ve bir sayının asal olup olmadığının test edilmesidir. Asal sayı oluşturma, verilmiş bir [r_1,r_2] tam sayılar aralığında asal sayı bulma işlemidir.

Modüler aritmetik'te asal sayılarla ilgi birkaç tanım ve teorem konuyu anlatabilmek açısından verilebilir.

Tanım 2.2: $as^{-1} \equiv 1 \bmod s$ şartını ve $1< a< s$ şartını sağlayan s tam sayısına a tabanına göre **sanki asal** (pseudoprime) sayı denir**.**

Teorem 2.2 (Euler Totient fonksiyonu): *n* tam sayısı için Euler Totient fonksiyonu φ(*n*), *n* den daha küçük olan ve *n* ile aralarında asal olan bütün pozitif tam sayıların sayısını verir.

- *p* asal ise φ (*p*) = *p*-1 ‘dir.

- $n=p\times q$ ve *p*, *q* asal sayılar ise $\varphi(n) = \varphi(p)\times\varphi(q) = (p-1)\times(q-1)$ dir.

Teorem 2.3 (Fermat teoremi): Eğer ***s*** bir asal sayı ve ortak bölenlerin en büyüğü *OBEB*(*a,s*)=1 ise *s*, *a* tabanına göre bir sanki asal (pseudo prime) sayıdır. [6]

2.3. Tek Yönlü Fonksiyon

$$F : X \rightarrow Y$$

$$\boldsymbol{F} : x \rightarrow f(x)=y$$

yalnız ve yalnız aşağıdaki şartları taşıdığı takdirde tek yönlü bir fonksiyondur:

- *f*(*x*) bütün *x* değerleri için polinomsal (çok terimli) zamanda çözümlenebilir olmalıdır.
- Verilen bir *y* değeri için *x* değeri polinomsal zamanda bulunamamalıdır.

Örneğin, $a^m \bmod n \equiv x$ bir modüler üs alma işlemidir ve kolaylıkla gerçekleştirilebilir; ancak, var olan *x* değerinden *m* değerini bulmak ayrık logaritma problemine girer ve bunun da hesaplanma süresi polinomsal çözümleme süresinden çok daha uzundur.

2.3.1. Kapı Tek Yönlü Fonksiyonlar (Trapdoor One-Way Functions)

Kapaklı tek yönlü fonksiyonlarda ise tek yönlü fonksiyonlara ek olarak analizciye başka bilgiler verilirse fonksiyon daha kolay tersinir hale getirilebilir.

Örneğin yalnız $a^m \bmod n$ değerini bilmekten öte buradaki n değerinin iki asal sayının çarpımı olduğunu ve anahtarların bu sayılara bağlı olduğunu bilmek buradan m değerini bulma aşamasında analizciye ipucu vermiş olur. [6]

2.4. Grup Teorisi

Verilen herhangi bir G grubu için bu gruba ait elemanların sayısına G grubunun **düzen**i denir ve $ord\,(G) = |G|$ sembolüyle gösterilir. Eğer H grubu G grubunun bir alt grubu ise $|H|$ değeri $|G|$ değerini böler. Böylece eğer G grubunun *düzeni* bir asal sayıysa G' nin tek alt grubu kendisidir. Bu durumda G grubu çarpmalı olarak yazılabilir.

Eğer G grubu çarpmalı olarak yazılabilirse ve $g \in G$ olmak üzere g sayısı G grubunun düzeni ise bu g sayısı $i \in \mathrm{N} \cup \{\infty\}$ ve $g^i = 1$ şartını sağlayan en küçük i değeridir. Burada $\forall j,l \in Z$:

$$g^j = g^l \Leftrightarrow j \equiv l \bmod ord(g)$$

dir.

G grubunun altgrubu olan tüm gruplar g elemanının bir üssüdür ve $\langle g \rangle$ ifadesiyle gösterilirler. Eğer $\langle g \rangle = G$ ise g sayısı G grubunun **üreteci** (jeneratörü) olur. Bir üreteci olan tüm gruplara **devirli grup** (cyclic group) adı verilir.

G grubunun düzeni p asal sayısı ise grup içerisinde yer alan 1 dışındaki tüm sayılar G grubunun üreteci olur. Diğer bir deyişle $\langle g \rangle$ nin düzeni 1 veya p sayısı olur.

Teorem 2.4 (Fermat teoremi): p bir asal sayı olsun. Her a tam sayısı için

$$a^p \equiv a \bmod p \tag{2.3}$$

denkliği; ve p ile bölünmeyen her a tam sayısı için ise

$$a^{p-1} \equiv 1 \bmod p \tag{2.4}$$

denkliği her zaman doğrudur. [6]

2.5. GF(p) ‘de üstel işlem

- Birçok kriptolama algoritması üstelleştirmeyi kullanır, b üssü ne göre büyüyen bir a sayısı(taban) mod p
    - $b = a^e \bmod p$

- üstelleştirme basit olarak bir n sayısı için O(n) çarpma olan tekrarlanan çarpmalardır.

- Daha iyi bir yöntem kare ve çarpma algoritmasıdır. Buna ait kaba kod,

let base = a, result =1
for each bit ei (LSB to MSB) of exponent
if ei=0 then
square base mod p
if ei=1 then
multiply result by base mod p
square base mod p (except for MSB)
required ae is result

şeklindedir.

- Bir n sayısı için sadece $O(\log_2 n)$ çarpma yapılır. [6]

2.6. GF(p) ‘de ayrık Logaritmalar

- Üstelleştirmede ters problem, bir modulo p sayısının ayrık logaritmasının bulunmasıdır.
    - Yani, $a^i = b \bmod p$ için *i* bulunur.

- Üstelleştirme nispeten kolay iken, ayrık logaritmanın bulunması genellikle kolay yolu olmayan zor bir problemdir.

- Bu problemde, eğer p asal ise, herhangi bir b! = 0 için her zaman bir ayrık logaritması olan bir α olduğu gösterilebilir.
  - α'nın ardışıl kuvvetleri mod p ile grup oluşturur
    α mod p, α^2 mod p, …, α^{p-1} mod p farklıdırlar ve 1 ila p-1 arasında değer alır.

- Öyle ki α ya **primitif kök** denir ve aynı zamanda bulmak nispeten zordur.

Herhangi bir *b* tamsayısı ve p nin primitif kökü olan α için bir *i* üssü bulunabilir ki;

$$b = \alpha^i \bmod p \quad 0 \le i \le (p-1)$$

dir.

Üs i ayrık logaritma veya indeks olarak gösterilir. [6]

2.7. En Büyük Ortak Bölen (OBEB (Greatest Common Divisor))

Teorem 2.5: *a* ve *n* tam sayıları için, ($a \in \{0, 1, \ldots, n-1\}$); eğer *a* ve *n* aralarında asal iki sayıysa *a* 'nın modül *n'* e göre yalnız bir tane tersi vardır ve a^{-1} sembolüyle gösterilir.

$OBEB(a,n) = 1 \Leftrightarrow \exists b \in [a,n-1],\ 1 = a \times b \bmod n$, yani $b = a^{-1}$ dir.

- a ve b'nin en büyük ortak böleni (a,b) a ve b'nin her ikisini de bölen en büyük sayıdır.

- **Euclid's Algoritması** iki a ve n (a<n) sayısının en büyük ortak bölenini bulmak için kullanılır,
  - Eğer a ve b nin böleni d ise, a-b ve a-2b 'yi bulur.

OBEB (GCD) bulunumuna dair kaba kod

GCD (a,n) is given by:

let g0=n

g1=a

gi+1 = gi-1 mod gi

when gi=0 then (a,n) = gi-1

gibidir.

Örneğin, OBEB (56,98) 'i bulalım. Yukardaki işlevi adım adım izlersek
g0=98 ve g1=56 olur. Buradan
g2 = 98 mod 56 = 42 elde edilir. g2, 0'dan farklı olduğundan
g3 = 56 mod 42 = 14 elde edilir. g3, 0'dan farklı olduğundan
g4 = 42 mod 14 = 0 elde edilir. g4 = 0 olduğundan OBEB (56,98)=14 olur.

Teorem 2.6 (Euler genellemesi): Her a ve n tam sayısı için;

$$OBEB\ (a,n) = 1 \text{ ise } a^{\varphi(n)} \equiv 1 \bmod n$$

dir.

Buradan eğer $a \times x \equiv 1 \bmod n$ ifadesini $OBEB\ (a,n)=1$ ile çalıştırırsak

$$x \equiv a^{\varphi(n)-1} \bmod n \qquad (2.5)$$

bize a sayısının modül n 'ye göre tersini verir.

Teorem 2.7 (Chinese Remainder Teoremi):

$X \equiv a_1 \bmod m_1$

$X \equiv a_2 \bmod m_2$

...

$X \equiv a_r \bmod m_r$ ve $OBEB(m_i, m_j) = 1,\ i \neq j$

benzerlik sistemleri için, X ' in en az bir çözümü vardır. Çözüm,

$$X = \sum a_i . M_i N_i \qquad (2.5)$$

ifadesiyle bulunur. 2.5 denkleminde

$$M = m_1 . m_2 . m_3 . \ldots . m_r$$
$$M_i = M / m_i$$
$$N_i = M_i^{-1} \bmod m_i$$

ifadeleriyle hesaplanır. [6]

3. KLASİK KRİPTOGRAFİ

3.1. Bazı Basit Kriptosistemler

Kriptografinin temel amacı, genellikle Alice ve Bob olarak adlandırılan iki kişinin güvenli olmayan bir kanal üzerinden Oscar olarak adlandırılan düşmanın ne söylendiğini öğrenmesi önlenerek haberleşmesini sağlamaktır. Bu kanal bir telefon hattı veya bilgisayar ağı olabilir. Alice'in Bob'a göndermek istediği bilgi (düzmetin) İngilizce bir metin, nümerik bir veri veya yapısı tamamen keyfe bağlı olan herhangi birşey olabilir. Alice, önceden kararlaştırılan bir anahtar kullanarak düzmetni şifreler ve şifrelenen şifreli metni kanal üzerinden gönderir. Oscar kanal içindeki şifreli metni gözler. Ancak düzmetnin ne olduğunu tespit edemez. Bob ise şifreleme anahtarını bildiği için şifreli metnin şifresini çözer ve düzmetni elde eder.

Bu kavram aşağıdaki matematiksel yaklaşım kullanılarak daha biçimsel olarak tanımlanmıştır:

Tanım 3.1: Bir kriptosistem aşağıdaki koşulları sağlayan bir beşliden (P, C, κ, E, D) meydana gelir.

1. P; Düzmetinlerin sonlu kümesidir.
2. C; Şifreli metinlerin sonlu kümesidir.
3. K; anahtar alan, anahtarların sonlu kümesidir.
4. Her $K \in \kappa$ olmak üzere bir $e_K \in E$ şifreleme kuralı ve buna bağlı olarak bir $d_K \in D$ şifre çözme kuralı vardır. Her $e_K : P \rightarrow C$ ve $d_K : C \rightarrow P$ fonksiyonları her düzmetin $x \in P$ için $d_K(e_K(x)) = x$ sağlar.

Burada temel özellik 4. koşuldur ve anlatılmak istenen e_K kullanılarak bir düzmetin (x) şifrelenir ve elde edilen şifreli metin daha sonra d_K kullanılarak çözülür ve orjinal düzmetin x elde edilir.

Alice ve Bob aşağıdaki protokolü belirli bir kriptosistem kullanarak çalıştıracaklardır. Öncelikle, rasgele bir anahtar seçerler ($K \in \mathcal{K}$). Anahtar seçme işlemi Alice ve Bob aynı yerdeyken ve Oscar tarafından gözetlenmiyorken ya da farklı yerlerdeyken ancak güvenli bir kanala eriştikleri zaman yapılabilir.

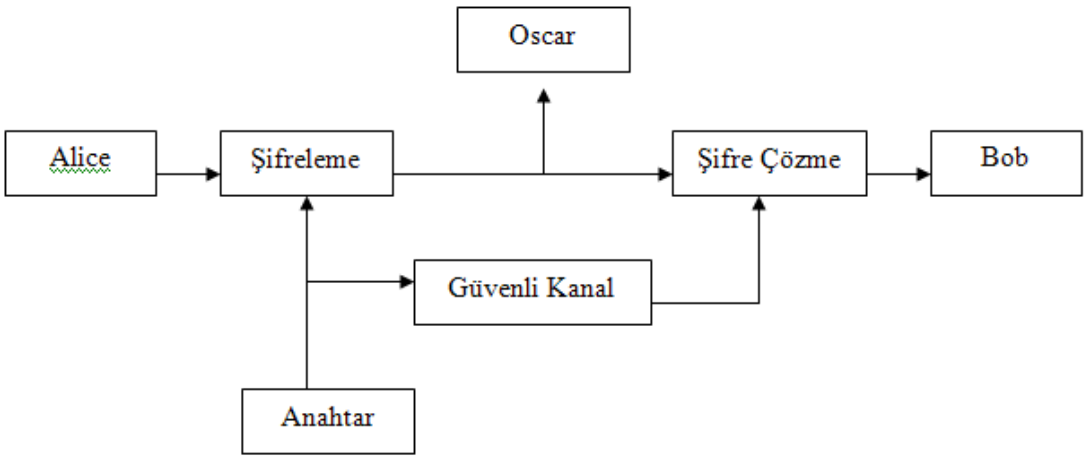
Daha sonra Alice'in Bob'a güvenli olmayan bir kanal üzerinden bir mesaj göndermek istediğini varsayalım ve bu mesaj örneğin şu şekilde olsun:
$x = x_1\, x_2 \ldots x_n \quad , n \geq 1, \; x_i \in P, \; 1 \leq i \leq n$

Her x_i önceden belirlenen K anahtarı ile e_k şifreleme kuralı kullanılarak şifrelenir. Alice, $y_i = e_k(x_i)$, $1 \leq i \leq n$'i hesaplar ve sonuç olarak şifreli metin

$$y = y_1\, y_2 \ldots y_n$$

elde edilerek kanal üzerinden gönderir.

Bob $y = y_1 y_2 \ldots y_n$'i aldığı zaman d_k şifre çözme kuralını kullanarak şifreli metni çözer ve $x = x_1 x_2 \ldots x_n$ düzmetni elde eder. Bu haberleşme kanalı Şekil 3.1'de gösterilmektedir.



Şekil 3.1: Haberleşme Kanalı

Açıkça görüldüğü üzere, her şifreleme fonksiyonu (e_k) bire-bir fonksiyon olmalıdır. Aksi halde, şifre çözme belirli bir yöntem kullanılarak geliştirilemez. Örneğin eğer,

$$y = e_k(x_1) = e_k(x_2)\ (x_1 \neq x_2)$$

olduğu bir durumda Bob, y'nin x_1 ile mi yoksa x_2 ile mi çözüleceğini bilemez. Eğer P=C ise, her şifreleme fonksiyonu bir permütasyondur. Yani, eğer düzmetinlerin ve şifreli metinlerin kümesi özdeş ise, her şifreleme fonksiyonu bu kümenin elemanlarını tekrardan düzenler.

Bazı klasik şifreleme yöntemleri konuyu açıklamak için verilebilir. [20]

3.1.1. Kaydırma Şifresi (The Shift Cipher)

Bu bölümde, modüler aritmetiğe dayanan kaydırma şifresini tanımlanacaktır. Ancak önce modüler aritmetikle ilgili bazı temel tanımlamaların gözden geçirilmesi gerekir.

Tanım 3.2: a ve b'nin tamsayılar, m'nin pozitif bir tamsayı olduğunu varsayalım. Eğer m, $(b-a)$' yı bölebiliyorsa

$$a \equiv b \pmod{m} \qquad (3.1)$$

yazılabilir. 3.1 denklemi "a, b'ye mod m'e göre denktir" şeklinde okunur. m tamsayısına modüler denir.

a ve b'yi m ile bölüm ve kalanı elde etmek için böldüğümüzü varsayalım. Kalan 0 ile m-1 arasında bir değer alacaktır. Bu;

$$0 \leq r_1 \leq m-1$$

ve

$$0 \leq r_2 \leq m-1$$

olduğu durumlarda

$$a = q_1 m + r_1$$

ve

$$b = q_2 m + r_2$$

eşitliklerini verir. Bu eşitliklerden sadece ve sadece,

$$r_1 = r_2$$

olduğu zaman

$$a \equiv b \pmod{m}$$

olabileceği kolayca görülebilir. b mod m ifadesini, a; m ile bölündüğünde kalanı göstermek için kullanacağız. Böylece,

$$a \equiv b \pmod{m}$$

sadece ve sadece

$$a \bmod m = b \bmod m$$

olduğu durumda doğrudur. Burda a yerine a mod m konulmuştur, a'ya mod m'e göre indirgenmiş denir. [20]

Uyarı: Birçok bilgisayar programlama dili a mod m'i, a ile aynı işarete sahip –m+1, ..., m-1 aralığında kalan olarak tanımlar. Örneğin, -18 mod 7, yukarda tanımlandığı gibi 3 yerine –4 olacaktır. Ancak a mod m'i her zaman n-negatif olarak tanımlamak daha kullanışlı olacaktır.

Aritmetik modül m: Z_m, toplama (+) ve çarpma ($\times$) işlemleri ile {0, ..., m-1} kümesi olarak tanımlanır. Z_m'deki toplama ve çarpma işlemleri çıkan sonuçların modül m'e indirgenmesi dışında bilinen toplama ve çarpma işlemleri ile aynıdır.

Örneğin, 11×13'ün Z_{16}'da hesaplanması istensin. Tamsayılar olarak

$$11 \times 13 = 143$$

tür. 143'ü mod 16'ya indirgemek için bölme işlemini gerçekleştirilir:

$$143 = 8 \times 16 + 15$$

elde edilir. Böylece

$$143 \bmod 16 = 15$$

tir; yani, Z_{16}'da $11 \times 13 = 15$'tir.

Z_m'de yapılan bu toplama ve çarpma işlemlerinin tanımları, aritmetikte birçok benzer kuralı karşılamaktadır. Aşağıda bu özellikler listelenmiştir:

1. Toplamanın kapalılık özelliği, herhangi bir a, b $\in$ Z_m için a+b $\in$ Z_m'dir.
2. Toplamanın değişme özelliği, herhangi a, b $\in$ Z_m için a+b = b+a'dır.
3. Toplamanın dağılma özelliği, herhangi a, b, c $\in$ Z_m için (a+b)+c = a+(b+c)
4. Toplamada etkisiz eleman 0'dır, herhangi bir a$\in Z_m$ için a+0 = 0+a = a
5. Herhangi bir a$\in Z_m$'in toplamaya göre tersi m-a'dır, a+(m-a) = (m-a)+a = 0
6. Çarpmanın kapalılık özelliği, herhangi a, b$\in Z_m$ için ab$\in Z_m$
7. Çarpmanın değişme özelliği, herhangi a, b$\in Z_m$ için ab = ba
8. Çarpmanın dağılma özelliği, herhangi a, b, c$\in Z_m$ için (ab)c = a(bc)
9. Çarpmada etkisiz eleman 1'dir, herhangi bir a$\in Z_m$ için ax1 = 1xa = a
10. Çarpmanın toplama üzerinde dağılma özelliği vardır, herhangi a, b, c$\in Z_m$ için, (a+b)c = (ac)+(bc) ve a(b+c) = (ab)+(ac) 'dir.

Z_m'de toplamaya göre tersi alınabildiği için, (11+13) mod 31=24. Diğer bir yolda, önce 11'den 18'i çıkarıp –7 elde edilebilir ve daha sonra –7 mod 31=24 bulunur.

Kaydırma şifresi Tanım 3.2'de gösterilmiştir. İngiliz alfabesinde 26 harf olduğundan şifreleme Z_{26}'da tanımlanmıştır. Kaydırma şifresi Tanım 3.2 ile bir kriptosistemi şekillendirir. Örneğin, bütün x$\in Z_{26}$'da tanımlanmıştır. Kaydırma şifresi yukarıda tanımlandığı gibi bir kriptosistemi şekillendirir. Örneğin, bütün x$\in Z_{26}$ için $d_K(e_K(x)) = x$'tir. [20]

Tanım 3.2 Kaydırma Şifresi (Shift Cipher):

P = C = $\mathcal{K}$ = Z_{26} olsun, $0 \le K \le 25$ için

$e_k(x) = (x+K) \bmod 26$

ve

$d_k(y) = (y-K) \bmod 26$ tanımlanır.

$(x,y \in Z_{26})$

UYARI: k = 3 için kriptosistem Caesar şifresi olarak bilinir.

Kaydırma şifresini İngiliz alfabesindeki harfleri mod 26'ya göre $A \rightarrow 0$, $B \rightarrow 1$, …, $Z \rightarrow 25$ şeklinde şifreleyerek kullanacağız. Bu durum Tablo 3.1'de gösterilmiştir.

Tablo 3.1: İngiliz Alfabesinin mod 26'ya göre tamsayı karşılıkları

A	B	C	D	E	F	G	H	I	J	K	L	M
0	1	2	3	4	5	6	7	8	9	10	11	12

N	O	P	Q	R	S	T	U	V	W	X	Y	Z
13	14	15	16	17	18	19	20	21	22	23	24	25

Örnek 3.1: Kaydırma şifresi k=11 ve düzmetin de

wewillmeetatmidnight.

olsun.

Öncelikle düzmetindeki her harf Tablo 3.1'i kullanılarak tamsayılara çevrilir; yani, w için 22, e için 4 elde edilir. Sonuçta,

```
22  4 22  8 11 11 12  4  4 19
 0 19 12  8  3 13  8  6  7 19
```

elde edilir.

Daha sonra herbirine mod 26'ya göre 11 eklenir; örneğin, (22+11) mod 26 = 7 gibi. Sonuç olarak,

```
 7 15  7 19 22 22 23 15 15  4
11  4 23 19 14 24 19 17 18  4
```

elde edilir.

Elde edilen bu tamsayılar şifreli metni elde etmek üzere Tablo 3.1 yardımıyla alfabetik karakterlere çevrilir; örneğin, 7 için H, 15 için P gibi. Bu değişim sonucunda,

```
HPHTWWXPPELEXTOYTRSE.
```

elde edilir.

Şifreli metnin şifresini çözmek için, Bob öncelikle harfleri tamsayılara çevirir, sonra her değerden mod 26'a göre 11'i çıkarır ve son olarak tamsayıları alfabetik sayılara dönüştürerek düzmetni elde eder.

Tablo 3.1 kullanılarak

HPHTWWXPPELEXTOYTRSE

şifreli metni tam sayılı ifadelere çevrilir.

```
 7 15  7 19 22 22 23 15 15  4
11  4 23 19 14 24 19 17 18  4
```

Sonra mod 26'ya göre sırayla her sayıdan 11 çıkarılınca

```
22  4 22  8 11 11 12  4  4 19
 0 19 12  8  3 13  8  6  7 19
```

elde edilir. Tekrar Tablo 3.1 yardımıyla tam sayılar alfabetik karakterlere çevrilerek

wewillmeetatmidnight.

elde edilir.

Örnek 3.1'de okunabilirliliği arttırmak şifreli metin için büyük harfler, düzmetin için de küçük harfler kullanılmıştır.

Eğer bir kriptosistem pratik olarak kullanılacaksa, bir takım özellikleri sağlamalıdır:

1. Her şifreleme fonksiyonu e_k ve her şifre çözme fonksiyonu d_k hesaplanabilir olmalıdır.
2. Oscar, şifreli metin y'i gördüğünde, kullanılan K anahtarını ya da düzmetin x'i belirleyememelidir.

İkinci özellik "güvenlik" fikrini tanımlamaktadır. Verilen bir şifreli metin y'den K anahtarını hesaplamaya çalışma işlemine kriptoanaliz denir. Eğer Oscar K'yi tespit edebilirse, Bob'un yaptığı gibi d_k ile y'nin şifresini çözebilir. K'nin tesbit edilmesi en az düzmetin x'in tahmin edilmesi kadar zor olmalıdır.

Shift Cipher, güvenli bir yöntem değildir. Çünkü "ayrıntılı anahtar arama yöntemi" ile kriptoanaliz edilebilir. Sadece 26 mümkün anahtar olabileceği için, anlamlı bir düzmetin elde edene kadar olası bütün şifre çözme kuralı (d_k)'yı denemek kolaydır. Bu Örnek 3.2'de gösterilmiştir. [20]

Örnek 3.2:
Verilen şifreli metin;

JBCRCLQRWCRVNBJENBWRWN,

olsun.

Şifre çözme anahtarları d_0, d_1, ...vb. kullanarak aşağıdaki düzmetinler elde edilmiştir:

```
jbcrclqrwcrvnbjenbwrwn
iabqbkpqvbqumaidmavqvm
hzapajopuaptlzhclzupul
gyzozinotzoskygbkytotk
fxynyhmnsynrjxfajxsnsj
ewxmxglmrxmqiweziwrmri
dvwlwfklqwlphvdyhvqlqh
cuvkvejkpvkogucxgupkpg
btujudijoujnftbwftojof
astitchintimesavesnine
```

Bu noktada, düzmetin belirlenebildi. Anahtar K=9'dur. Ortalama olarak bir düzmetin 26/2=13 şifre çözme kuralı denenerek hesaplanabilir.

Örnek 3.2'de görüldüğü gibi, bir kriptosistemin güvenli olması için, etraflı anahtar araması olursuz olmalıdır; yani, anahtar uzayı çok büyük olmalıdır. Ancak kuvvetlice beklendiği gibi büyük bir anahtar alanı da güvenliği garanti edebilmek için yeterli değildir. [20]

3.1.2. Değiştirme Şifresi (The Substitution Cipher)

Diğer bilinen kriptosistemlerden bir tanesi de değiştirme şifresi olup Tanım 3.3'te tanımlanmıştır.

Tanım 3.3: Değiştirme şifresi:

P = C = Z_{26} olsun. K; 26 sembolün 0,1, ..., 25 bütün olası permütasyonlarından oluşsun. Her $\pi \in \kappa$ permütasyonu için, π^{-1} π'nin ters permütasyonu olmak üzere;

$$e_{\pi}(x) = \pi(x),$$

ve

$$d_{\pi}(x) = \pi^{-1}(y)$$

ile tanımlanır.

Düzmetin karakterleri küçük harflerle ve şifreli metin karakterleri de büyük harflerle yazılmıştır. Bu durum Tablo 3.2'de görülmektedir.

Tablo 3.2: Örnek π fonksiyonu

a	b	c	d	e	F	g	h	i	j	k	l	m
X	N	Y	A	H	P	O	G	Z	Q	W	B	T

n	o	p	q	r	S	t	u	v	w	x	y	z
S	F	L	R	C	V	M	U	E	K	J	D	I

Şifre çözme fonksiyonu alfabetik sıranın tersidir.

Tablo 3.3: Örnek π^{-1} fonksiyonu

A	B	C	D	E	F	G	H	I	J	K	L	M
d	l	r	y	v	o	h	e	z	x	w	p	t

N	O	P	Q	R	S	T	U	V	W	X	Y	Z
b	g	f	j	q	n	m	u	s	k	a	c	i

Dolayısıyla Tablo 3.2'den yararlanarak d_π (A) = d, d_π(B) = l , … vb. elde edilir.

Bu şifre çözme fonksiyonu kullanılarak aşağıdaki şifreli metin deşifrelenebilir:

MGZVYZLGHCMHJMYXSSFMNHAHYCDLMHA.

Tablo 3.3 kullanılarak $d_{\pi^{-1}}$ (M) = t, $d_{\pi^{-1}}$ (G) = h, vb elde edilerek düz metin

thisciphertextcannotbedecrypted.

elde edilir.

Değiştirme şifresi için gerekli olan bir anahtar sadece 26 alfabetik karakterin bir permütasyonundan meydana gelmektedir. İlk harf için 26 seçim, ikinci harf için 25 seçim, vb seçimler yaparak permütasyon sayısı bulunabilir. Böylece bu permütasyonların sayısı 26! olarak elde edilir. Bu sayı 4.0×10^{26}'dan daha büyüktür. Böylelikle, bir bilgisayar için bile bir ayrıntılı anahtar araması mümkün değildir. Ancak daha sonra göreceğimiz diğer yöntemlerle Değiştirme şifresi kolayca kriptoanaliz edilebilir. [20]

3.1.3. Affine Şifresi

Değiştirme şifresi, kaydırma şifresinin 26 elemanının olası 26!'den sadece 26'sını içeren özel bir durumudur. Kaydırma şifresinin başka bir özel durumu da şimdi bahsedeceğimiz Affine şifresidir. Affine şifresinde, şifreleme fonksiyonları; $a,b \in Z_{26}$

$$e(x) = (ax+b) \mod 26 \qquad (3.2)$$

fonksiyonları şeklinde sınırlandırılır. Bu fonksiyonlara afin fonksiyonları denir; bundan dolayı Affine şifresi olarak adlandırılır. (a = 1 iken yerdeğiştirme şifresi elde edilir.)

Şifre çözmenin mümkün olması için, bir afin fonksiyonunun ne zaman bire bir olacağını sormak gereklidir. Başka bir deyişle, herhangi $y \in Z_{26}$ için,

$$ax+b \equiv y \pmod{26} \tag{3.3}$$

denkliğinin x için tek bir çözümünün olması istenir. Bu eşleşim

$$ax \equiv y-b \pmod{26} \tag{3.4}$$

ifadesine eşittir.

Bu durumda y, Z_{26}'da farklı değerler aldığı müddetçe, y-b'de Z_{26}'da farklı değerler alacaktır. Bu eşleşim her y için sadece ve sadece OBEB(a,26) = 1 olduğu zaman tek bir çözüme sahiptir (OBEB fonksiyonu argümanların en büyük ortak bölenini göstermektedir). Öncelikle, OBEB(a,26) = d > 1 olduğunu varsayalım. Buna göre, $ax \equiv 0 \pmod{26}$ denkliği Z_{26}'da x = 0 ve x=26/d olmak üzere en azından iki farklı çözüme sahiptir. Bu durumda, e(x) = ax+b mod 26 bire bir fonksiyon değildir ve buna bağlı olarak da doğru bir şifreleme fonksiyonu da olamaz.

Örneğin, OBEB(4,26) = 2 olduğu için, 4x+7 doğru bir şifreleme fonksiyonu değildir. Çünkü herhangi bir $x \in Z_{26}$ için x ve x+13 aynı değeri şifreleyecektir.

Şimdi de OBEB(a,26) = 1 olduğunu varsayalım. Herhangi x_1 ve x_2 için

$$ax_1 \equiv ax_2 \pmod{26}$$

olduğunu varsayalım. O zaman,

$$a(x_1-x_2) \equiv 0 \pmod{26},$$

olduğundan

$$26 \mid a(x_1-x_2)$$

elde edilir.

Bölmenin şu özelliğini kullanacağız: Eğer OBEB(a,b) = 1 ve a / bc ise, a / c'dir. Örneğin,

$$26 / a(x_1-x_2)$$

ve

$$\gcd(a,26) = 1$$

olduğu için

$$26 \mid (x_1-x_2)$$

elde edilir; dolayısıyla,

$$x_1 \equiv x_2 \pmod{26}$$

bulunur.

Bu noktada, eğer OBEB(a,26) = 1 ise $ax \equiv y \pmod{26}$ denkliğinin Z_{26}'da en çok bir çözümü vardır. Buna bağlı olarak, x'in Z_{26}'da değişmesine izin verilirse, ax mod 26; 26 farklı değer alır. Buda her değeri sadece bir kere alması anlamına gelmektedir. Herhangi bir $y \in Z_{26}$ için $ax \equiv y \pmod{26}$ denkliğinin y için tek bir çözümü vardır.

Teorem 3.1: $ax \equiv b \pmod{m}$ denkliğinin her $b \in Z_m$ için, sadece OBEB(a,m) = 1 olduğunda tek bir $x \in Z_m$ çözümü vardır.

26 = 2 x 13 olduğundan, OBEB(a,26) = 1 için $a \in Z_{26}$ değerleri, a = 1, 3, 5, 7, 9, 11, 15, 17, 19, 21, 23, ve 25'dir. b parametresi Z_{26}'da herhangi bir eleman olabilir. Buna bağlı olarak Affine şifresinin 12 x 26 = 312 anahtarı vardır. (Tabii ki bu da güvenlik için yeterli değildir.)

Tanım 3.4: $a \geq 1$ ve $m \geq 2$ tamsayılar olsun. Eğer gcd(a,m) = 1 ise a ve m aralarında asaldır. m ile aralarında asal olan Z_m'deki tamsayıların sayısı $\phi(m)$ ile gösterilir. (Bu fonksiyona Euler phi-fonksiyonu denir.)

Teorem 3.2: p_i'lerin ayrı asallar ve $e_i > 0$, $1 \leq i \leq n$ olduğu yerlerde

$$M = \prod_{i=1}^{n} p_i^{e_i} \tag{3.5}$$

olduğunu varsayalım. O zaman,

$$\phi(m) = \prod_{i=1}^{n} \left(p_i^{e_i} - p_i^{e_i - 1} \right) \tag{3.6}$$

elde edilir.

Z_m üzerindeki Affine şifresi içindeki anahtarların sayısı yukarıdaki formülde verilen $m\phi(m)$'dir (Şifreleme fonksiyonu e(x) = ax+b olduğu durumda b için seçimlerin sayısı m, ve a için seçimlerin sayısı $\phi(m)$'dir). Örneğin $m=60=2^2 \times 3 \times 5$ olduğu zaman

$$\phi(60) = \phi(2^2 \times 3 \times 5) = (2^2\text{-}2) \times (3\text{-}1) \times (5\text{-}1) = 2 \times 2 \times 4 = 16$$

ve Affine şifresindeki anahtarların sayısı da $60 \times 16 = 960$'tır.

Şimdi de Affine şifresinde, m=26 modülle şifre çözme işlemini düşünelim. $OBEB(a, 26) = 1$ olduğunu varsayalım. Şifre çözmek için, x için $y \equiv ax+b \pmod{26}$ denkliğinin çözülmesi gerekir. Yukarıda anlatılanlar denkliğin Z_{26}'da tek bir çözümü olacağından bahsetmektedir. Ancak bu çözüm bize çözümü bulmak için etkin bir yöntem sunmamaktadır. Gerekli olan şey bunu yapabilmek için etkin bir algoritmadır. Ancak, modüler aritmetikteki bazı sonuçlar bize istenilen şifre çözme algoritmasını sağlamaktadır.

Tanım 3.5: $a \in Z_m$ olsun. a'nın tersi $aa^{-1} \equiv a^{-1}a \equiv 1 \pmod{m}$ gibi bir $a^{-1} \in Z_m$'dir.

Z_{26}'da elemanların çarpmaya göre tersleri sırayla $1^{-1} = 1$, $3^{-1} = 9$, $5^{-1} = 21$, $7^{-1} = 15$, $11^{-1} = 19$, $17^{-1} = 23$, and $25^{-1} = 25$'tir. Örneğin, $7 \times 15 = 105 \equiv 1 \pmod{26}$; yani, $7^{-1} = 15$ bulunur.

Tanım 3.6: Affine Şifresi:

$P = C = Z_{26}$ ve

κ $\{(a,b) \in Z_{26} \times Z_{26} : \gcd(a,26) = 1\}$ olsun.

$K = (a,b) \in \kappa$ için,

$$e_K(x) = ax+b \bmod 26 \quad \text{ve}$$
$$d_K(y) = a^{-1}(y-b) \bmod 26 \quad \text{tanımlanır}$$

$(x,y \in Z_{26})$

$y \equiv ax+b \pmod{26}$ denkliği düşünülsün. Bu denklik $ax \equiv (y-b) \pmod{26}$'ya eşittir. OBEB(a,26)=1 olduğundan denkliğin her iki tarafı da a^{-1} ile çarpılabilir:

$$a^{-1}(ax) \equiv a^{-1}(y-b) \pmod{26} \qquad (3.7)$$

Modülo 26'ya göre çarpmanın birleşme özelliği (3.7) denkleminin sol tarafına uygulanırsa

$$a^{-1}(ax) \equiv (a^{-1}a)\, x \equiv 1x \equiv x$$

elde edilir. Sonuç olarak

$$x \equiv [a^{-1}(y-b)] \pmod{26}$$

elde edilir. Bu x için açık formüldür; yani, şifre çözme fonksiyonu,

$$d(y) = [a^{-1}(y-b)] \pmod{26}$$

olarak elde edilir. [20]

Örnek 3.3: Tanım 3.6'da verilen tanıma göre anahtar K = (7,3) olsun. Buna göre,

$$7^{-1} \bmod 26 = 15$$

tir. Bütün işlemleri Z_{26}'da gerçekleştirilen şifreleme fonksiyonu,

$$e_K(x) = 7x + 3$$

ve buna bağlı olarak şifre çözme fonksiyonu da,

$$d_K(y) = 15\,(y-3) = 15y - 19$$

olarak elde edilir. Bütün $x \in Z_{26}$ için $d_K(e_K(x)) = x$ doğrulamak iyi bir denetim olur. Z_{26}'da hesaplamalarla,

$$d_K(e_K(x)) = d_K(7x+3)$$
$$= (15\,(\,7x + 3\,) - 19) \bmod 26$$
$$= (105x + 45 - 19) \bmod 26$$
$$= ((26.4+1)x + 26) \bmod 26$$
$$= x$$

Örneklemek için *hot* düzmetnini şifrelensin. Öncelikle, Tablo 3.1'den yararlanılarak h, o, t harfleri modül 26'ya göre 7, 14 ve 19 olarak dönüştürülür. Bu değerler şifrelenirse,

h harfi için 7x7+3 mod 26 = 52 mod 26 = 0
o harfi için 7x14+3 mod 26 = 101 mod 26 = 23
t harfi için 7x19+3 mod 26 = 136 mod 26 = 6

elde edilir.

Böylece üç şifreli metin karakteri Tablo 3.1'e göre AXG'e karşılık gelen 0, 23 ve 6'dır. [20]

3.1.4. Vigenere Şifresi

Kaydırma şifresinde ve Kaydırma şifresinde bir anahtar seçildikten sonra her alfabetik karakter tek bir alfabetik karaktere haritalanır. Bu nedenle, bu kriptosistemlere *monoalfabetik* denir. Tanım 3.7'te Vigenere Şifresi olarak adlandırılan monoalfabetik olmayan bir kriptosistemi gösterilmiştir.

Tanım 3.7: Vigenere Şifresi:

m sabit pozitif bir tamsayı olsun. $P = C = \kappa = (Z_{26})^m$ tanımlansın.
Bir $K= (k_1,k_2,...,k_m)$ anahtarı için,

$$e_k(x_1,x_2,...,x_m) = (x_1+k_1, x_2+k_2,..., x_m+k_m) \text{ ve}$$

$$d_k(y_1,y_2,...,y_m) = (y_1-k_1, y_2-k_2,..., y_m-k_m) \text{ tanımlanır.}$$

Daha önceden Tablo 3.1'de tanımlanan A → 0, B → 1, …, Z → 25 bağlantıları kullanılarak, her K anahtarı; anahtar kelime (keyword) denilen m uzunluklu bir alfabetik dizi ile bağlanabilir. Vigenere Şifresi, her düzmetin elemanı m alfabetik karaktere eşit olduğu zaman bu karakterleri şifreler.

Bu durum Örnek 3.4'te açıklanmıştır.

Örnek 3.4:

m = 6 ve anahtar kelime de *CIPHER* olsun. Bunun nümerik eşdeğeri, Tablo 3.1'e göre anahtar K = (2,8,15,7,4,17)'dir. Düzmetnin aşağıdaki diziden oluştuğunu varsayalım:

```
Thiscryptosystemisnotsecure.
```

Bu düzmetin elemanlarını Tablo 3.1'den yararlanarak mod 26'ya göre 6'lı gruplar halinde yazıp, anahtar kelime eklenirse Tablo 3.4 elde edilir:

Tablo 3.4:

19	7	8	18	2	17	24	15	19	14	18	24
2	8	15	7	4	17	2	8	15	7	4	17
21	15	23	25	6	8	0	23	8	21	22	15

18	19	4	12	8	18	13	14	19	18	4	2
2	8	15	7	4	17	2	8	15	7	4	17
20	1	19	19	12	9	15	22	8	25	8	19

20	17	4
2	8	15
22	25	19

Bu alfabetik eşitliğin şifreli metin dizisi Tablo 3.4'e göre Tablo 3.1'den yararlanılarak

VPXZGIAXIVWPUBTTMJPWIZITWZT.

elde edilir.

Şifre çözmek için, aynı anahtar kelime kullanılır; ancak, toplama yerine mod 26'a göre çıkarma yapılır. Vigenere şifresinde, m uzunluğundaki anahtar kelimelerin sayısı 26^m 'dir. m uzunluğunda anahtar kelimeye sahip bir Vigenere şifresinde, bir alfabetik karakter; m alfabetik karakterlerden birine haritalanabilir. (anahtar kelimenin m farklı karakter içerdiğini varsayarsak.) Böyle bir kriptosisteme *polialfabetik* denir. Genel olarak, kriptoanaliz polialfabetik kriptosistemlerde, monoalfabetik kriptosistemlerde olduğundan daha zordur. [20]

3.1.5. Hill Şifresi

Bu bölümde Hill Şifresi olarak adlandırılan başka bir polialfabetik kriptosistemi tanımlanacaktır. Bu şifreleme 1929 yılında Lester S. Hill tarafından bulunmuştur. m pozitif bir tamsayı olsun ve $P = C = (Z_{26})^m$ olarak tanımlansın. Buradaki temel fikir, bir düzmetin elemanı içindeki m alfabetik karakterlerin, m lineer kombinasyonunu almaktır. Böylece, bir şifreli metin elemanı içinde m alfabetik karakter üretilir.

Örneğin, m = 2 ise, bir düzmetin elemanı $x = (x_1 , x_2)$ ve şifreli metin elemanı da $y = (y_1, y_2)$ olarak yazılabilir. Burada, y_1; x_1 ve x_2'nin lineer bir kombinasyonu olacaktır. Aynı durum y_2 için de geçerlidir.

Örneğin,

$$y_1 = 11x_1 + 3x_2$$

ve

$$y_2 = 8x_1 + 7x_2$$

ele alınsın.

Bunu daha kısa olarak (3.8)'deki gibi matris gösterimiyle yazmak da mümkündür:

$$(y_1,y_2) = (x_1,x_2) \begin{bmatrix} 11 & 8 \\ 3 & 7 \end{bmatrix} \tag{3.8}$$

Genel olarak, K anahtarı için $m \times m$'lik bir matris kullanılacaktır. $x = (x_1, \ldots, x_m) \in P$ ve $K \in \kappa$ için, $y = e_k(x) = (y_1, \ldots, y_m)$ (3.9)'daki gibi hesaplanır :

$$(y_1,y_2, \ldots, y_m) = (x_1,x_2, \ldots ,x_m) \begin{bmatrix} k_{11} & k_{12} & \ldots & k_{1m} \\ k_{21} & k_{22} & \ldots & k_{2m} \\ \vdots & \vdots & \vdots & \vdots \\ k_{m1} & k_{m2} & \ldots & k_{mm} \end{bmatrix} \tag{3.9}$$

Başka bir deyişle, $y = xK$ 'dir. Düzmetinden şifreli metin bir lineer dönüşüm ile elde edilir. Şifre çözmenin nasıl çalışacağını; yani, y'den x'in nasıl hesaplanacağının düşünülmesi gerekiyor. Şifre çözmek için K^{-1} ters matrisi kullanılır. Şifreli metin şifresinin çözümü için

$$x = yK^{-1} \tag{3.10}$$

formülü kullanılır.

Eğer $A=(a_{i,j})$, $l \times m$ matrisi ve $B=(b_{j,k})$ $m \times n$ matrisi ise, matris çarpımı $AB = (c_{i,k})$ aşağıdaki formülle tanımlanır:

$$c_{i,k} = \sum_{j=1}^{m} a_{ij} b_{jk} \text{ , } 1 \le i \le l \text{ ve } 1 \le k \le n$$

$m \times m$ birim matris I_m ile gösterilir. 2×2'lik birim matris ise

$$I_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

olarak tanımlanır.

Herhangi bir $l \times m$ A matrisi ve $m \times n$ B matrisi için $AI_m = A$ ve $BI_m = B$'dir. $m \times m$ A matrisinin tersi A^{-1} matrisi için $AA^{-1} = A^{-1}A = I_m$ 'dir. Her matrisin ters matrisi yoktur. Ancak eğer tersi varsa, bu tektir.

Elimizdeki bu açıklamalarla, (3.10)'da verilen şifre çözme fonksiyonunu çıkarmak kolaydır:

$y = xK$ denkleminde her iki taraf da K^{-1} ile çarpılarak:

$$y K^{-1} = (xK) K^{-1} = x (K K^{-1}) = xI_m = x \tag{3.11}$$

elde edilir.

(3.8)'deki şifreleme matrisinin Z_{26}'da tersi vardır:

$$\begin{bmatrix} 11 & 8 \\ 3 & 7 \end{bmatrix}^{-1} = \begin{bmatrix} 7 & 18 \\ 23 & 11 \end{bmatrix} \tag{3.12}$$

dir. Çünkü,

$$\begin{bmatrix} 11 & 8 \\ 3 & 7 \end{bmatrix}\begin{bmatrix} 7 & 18 \\ 23 & 11 \end{bmatrix} = \begin{bmatrix} 11\times 7+8\times 23 & 11\times 18+8\times 11 \\ 3\times 7+7\times 23 & 3\times 18+7\times 11 \end{bmatrix}$$

$$= \begin{bmatrix} 261 & 286 \\ 182 & 131 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

dir. Örnek 3.5'teki örnek yardımıyla Hill Şifresindeki şifreleme ve şifre çözmekonusuna açıklık getirilebilir. [20]

Örnek 3.5: Anahtarın,

$$\mathrm{K} = \begin{bmatrix} 11 & 8 \\ 3 & 7 \end{bmatrix}$$

olduğunu varsayalım.

(3.12)'de

$$K^{-1} = \begin{bmatrix} 7 & 18 \\ 23 & 11 \end{bmatrix}$$

olarak elde edilmişti.

july düzmetnini şifrelemek istediğimizi varsayalım. Şifrelemek için düzmetnin 2 elemanına sahibiz. Tablo 3.1'den yararlanarak ju'a ait (9,20) ve ly'e ait (11,24) sayısal elemanları elde edilir.

$$(9,20)\begin{bmatrix} 11 & 8 \\ 3 & 7 \end{bmatrix} = (\ 99+60,\ 72+140) = \ (3,\ 4)$$

ve

$$(11,24)\begin{bmatrix} 11 & 8 \\ 3 & 7 \end{bmatrix} = \ (\ 121+72,\ 88+168\ ) = (11,\ 22)$$

elde edilir.

Böylece Tablo 3.1'den yararlanarak elde edilen sonuçlarla july'ın şifrelenmiş şekli DELW (3, 4, 11, 22) elde edilmiş olur. Şifreyi çözmek için Bob (3.9) denkleminden yararlanarak (3,4) ve (11,22) elemanları için aşağıdaki gibi bir hesaplama yapar:

$$(3,4)\begin{bmatrix} 7 & 18 \\ 23 & 11 \end{bmatrix} = (\,9,\,20\,)$$

ve,

$$(11,22)\begin{bmatrix} 7 & 18 \\ 23 & 11 \end{bmatrix} = (\,11,\,24\,)$$

bulunur.

Böylece july düzmetni elde edilir.

Bu noktada şifre çözmenin; K'nin tersi olması durumunda mümkün olacağı gösterilmiştir. Bu nedenle tersi alınabilen K matrisleri ele alınır. Bir (kare) matrisin tersinin alınabilmesi determinant değerine bağlıdır. Determinant değeri sıfırdan farklı ise ilgili matrisin tersi mevcuttur denir. [20]

Tanım 3.8: 2×2'lik $A = (a_{i,j})$ matrisinin determinant değeri :

$$\det A = a_{1,1}\, a_{2,2} - a_{1,2}\, a_{2,1} \tag{3.13}$$

denklemiyle elde edilir.

Determinantlarda iki önemli özelliklerden bir tanesi $\det I_m = 1$, diğeri de

$$\det(AB) = \det A \times \det B \tag{3.14}$$

dir.

Bir K matrisinin determinantı sıfırdan farklı olduğu durumda tersi vardır. Ancak biz Z_{26}'da işlemlerimizi yaptığımız için; K matrisinin tersi sadece OBEB(det K,26) = 1 olduğu durumlarda vardır.

Bunu kanıtlamak için öncelikle OBEB(detK,26) = 1 olduğunu varsayalım. O zaman det K'nin Z_{26}'da tersi vardır. $1 \le i \le m$, $1 \le j \le m$ için K matrisinin i. sırası ve j. sütunu silinerek elde edilmiş bir K_{ij} matrisi tanımlansın. Değeri $(-1)^{i+j}$ det (K_{ji}) olan bir K* matrisi tanımlansın. (K* matrisine adjoint matris denir.)

$$K^{-1} = (\det K)^{-1} K^* \tag{3.15}$$

Buna bağlı olarak K'nın tersi alınabilir.

Tersine, K^{-1}'in K'nin tersi olduğunu varsayalım. Determinantların çarpım kuralı (3.14)'den;

$$1 = \det I = \det (K K^{-1}) = \det K \times \det K^{-1}$$

elde edilir. Böylece de K, Z_{26}'da tersi alınabilir bir matristir. [20]

Teorem 3.3: $A = (a_{i,j})$; Z_{26}'da $\det A = a_{1,1} a_{2,2} - a_{1,2} a_{2,1}$ 'in tersi alınabilen bir 2×2'lık matris olsun. O zaman,

$$A^{-1} = (\det A)^{-1} \begin{bmatrix} a_{22} & -a_{12} \\ -a_{21} & a_{11} \end{bmatrix} \tag{3.16}$$

dir.

Şimdi daha önce ele alınan Örnek 3.5'i göz önüne alalım:

$$\det\begin{bmatrix} 11 & 8 \\ 3 & 7 \end{bmatrix} = (11\times 7 - 8\times 3) \bmod 26$$

$$= (77-24) \bmod 26$$

$$= 53 \bmod 26 = 1$$

elde eldir.

$1^{-1} \bmod 26 = 1$'dir. Böylece daha önce elde edilen ters matris:

$$\begin{bmatrix} 11 & 8 \\ 3 & 7 \end{bmatrix}^{-1} = \begin{bmatrix} 7 & 18 \\ 23 & 11 \end{bmatrix}$$

elde edilir.

Tanım 3.9'da Hill Şifresi ayrıntılı olarak tanımlanmıştır.

Tanım 3.9: Hill Şifresi:

m herhangi bir sabit pozitif tamsayı olsun. $P = C = (Z_{26})^m$ ve

$\kappa = \{Z_{26}$'da mxm tersi alınabilir matrisler$\}$ olsun.

K anahtarı için,

$$e_K(x) = xK$$

ve

$$d_K(x) = yK^{-1}$$

tanımlanır.

3.1.6 Permütasyon Şifresi

Buraya kadar bahsedilen bütün kriptosistemler kaydırmayı içeriyordu. Düzmetin karakterleri yerine, farklı şifreli metin karakterleri konuluyordu. Permütasyon şifresinde ise düzmetin karakterleri değiştirilmez; ancak, tekrardan düzenlenerek yerleri değiştirilir. Permütasyon şifresi Tanım 3.10'de açıklanmıştır.

Tanım 3.10: Permütasyon Şifresi:

m herhangi bir sabit pozitif tamsayı olsun. $P = C = (Z_{26})^m$ ve

κ, $\{1, ..., m\}$'in tüm permütasyonlarından oluşsun. Bir π anahtarı için,

$$e_\pi(x_1,.....,x_m) = (x_{\pi(1)},....., x_{\pi(m)})$$

ve

$$d_\pi(y_1,.....,y_m) = (y_{\pi^{-1}(1)},....., y_{\pi^{-1}(m)})$$

tanımlanır.

(π^{-1}, π'nin ters permütasyonudur.)

Örnek 3.6'da Permütasyon şifresi ile ilgili bir örnek verilmiştir:

Örnek 3.6: m = 6 ve anahtarda aşağıdaki gibi π permütasyonu olsun:

1	2	3	4	5	6
3	6	1	5	2	4

Ters permütasyon π^{-1}' de aşağıdaki gibidir:

1	2	3	4	5	6
3	5	1	6	4	2

Düzmetin "shesellsseashellsbytheseashore." olsun.

Öncelikle düzmetin altı harf olacak biçimde gruplara ayrılırak

shesel | lsseas | hellsb | ythese | ashore

elde edilir.

Daha sonra bu altı harfli grupları π permütasyon tablosuna göre tekrardan düzenlenerek (ilk altılı grupta sırayla 1. harf s için 3. harf E'yi, 2. harf h için 5. harf E'yi, 3. harf e için 1. harf S'yi, 4. harf s için 6. harf L'yi, 5. harf e için 4. harf S'i ve 6. harf l için 2. harf H'yi yerlerine yazarak vs diğer altılı gruplar için aynısını uygulayarak)

EESLSH | SALSES | LSHBLE | HSYEET | HRAEOS

elde edilir.

Böylece şifreli metin

EESLSHSALSESLSHBLEHSYEETHRAEOS.

olur.

π^{-1} ters permütasyonu kullanılarak şifreli metnin şifresi benzer şekilde çözülür.

EESLSHSALSESLSHBLEHSYEETHRAEOS.

şifreli metni altılı gruplar halinde

EESLSH | SALSES | LSHBLE | HSYEET | HRAEOS

şeklinde yazılır. Daha sonra ilk altılı grupta sırayla 1. harf E yerine 3. harf s'yi, 2. harf E için 6. harf h'yi, 3. harf S için 1. harf e'yi, 4. harf L için 5. harf s'yi, 5. harf S için 2. harf e'yi ve 6. harf H için 4. harf l'yi yazarak ve diğer altılı ruplar için de bu yöntemi uygulayarak

```
shesel | lsseas | hellsb | ythese | ashore
```

elde edilir. Böylece düzetin "shesellsseashellsbytheseashore." olarak bulunur.

Gerçekte, Permütasyon şifresi; Hill şifresinin özel bir durumudur. {1, …, m} kümesinin verilen bir π permütasyonu ile

$$k_{ij} = \begin{cases} 1, & j = \pi(i) \\ 0, & \textit{ötekilerde} \end{cases} \tag{3.17}$$

formüllü ile bağlantılı bir $m \times m$'lik $K_\pi = (k_{ij})$ permütasyon matrisi tanımlanabilir.

Bir permütasyon matrisinin tüm satır ve sütunlarında mutlaka 1 vardır ve geri kalan değerler 0'dır. Permütasyon matrisini, birim matrisin satır ve sütunlarının permütasyonu ile elde edebiliriz.

Örnek 3.6'da kullanılan π permütasyonu için permütasyon matrisleri:

$$K_\pi = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

ve

$$K_\pi^{-1} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

şeklindedir. [20]

3.1.7 Akan (Stream) Şifresi

Bu noktaya kadar incelediğimiz tüm kriptosistemlerde, düzmetin elemanları aynı K anahtarı kullanılarak şifrelendi. Şifreli metin dizisi y (3.18)'deki gibi elde edilir:

$$y = y_1 y_2 ... = e_K(x_1)\, e_K(x_2) ... \tag{3.18}$$

Bu tipteki kriptosistemlere *blok şifreler* denir.

Başka bir yaklaşım ise stream şifrelerini kullanmaktır. Buradaki temel fikir bir $z = z_1 z_2 \ldots$ anahtar akımı meydana getirmek ve düzmetin dizisini $x = x_1\, x_2 \ldots$'i (3.19)'da tanımlanan kurala göre şifrelemek için kullanmaktır.

$$y = y_1 y_2 \ldots = e_{z_1}(x_1)\, e_{z_2}(x_2) \ldots \tag{3.19}$$

$K \in \kappa$ anahtar ve $x_1\, x_2 ...$ de düzmetin olsun. f_i fonksiyonu z_i (anahtar akımının i. elemanı)'yi oluşturmak için kullanılır. İlk i-1 düzmetin karakterleri:

$$z_i = f_i (K, x_1 .., x_{i-1})$$

dir.

Anahtar akım elemanı z_i x_i'i şifrelemek için kullanılır. Bu nedenle $x = x_1\, x_2 \ldots$düzmetnini şifrelemek için

$$z_1, y_1, z_2, y_2, \ldots$$

hesaplanır. $y_1 y_2 ...$ şifreli metin dizisinin şifresini çözmek için de

$$z_1, x_1, z_2, x_2, \ldots$$

hesaplanır.

Tanım 3.11: Bir Stream Şifresi aşağıdaki koşulları sağlayan (P, C, κ, L, F, E, D)'den oluşmaktadır.

1. P; düzmetinlerin sonlu kümesidir.
2. C; şifreli metinlerin sonlu kümesidir.
3. κ; anahtar alan, anahtarların sonlu kümesidir.
4. L; anahtar akım (keystream) alfabesi denilen sonlu bir kümedir.
5. $F = (f_1, f_2, ..)$ anahtar akım üreteci. $i \geq 1$ için;

$$f_i : \kappa \times P^{i-1} \rightarrow L$$

dir.

6. Her $z \in L$ için, $e_z \in E$ olmak üzere bir şifreleme fonksiyonu ve buna bağlı olarak $d_z \in D$ olmak üzere de bir şifre çözme fonksiyonu vardır. $e_z : P \rightarrow C$ ve $d_z : C \rightarrow P$ fonksiyonları her $x \in P$ düzmetni için $d_z(e_z(x)) = x$'tir.

Blok Şifreleme, Akan Şifrelemenin özel bir durumu gibi düşünülebilir: tüm $i \geq 1$ için $z_i = K$. Bir akan şifre eğer anahtar akım düzmetin dizisinden bağımsızsa eşzamanlıdır ve bir Akan şifre tüm $i \geq 1$ tamsayıları için d $(z_{i+d} = z_i)$ periyodu ile periyodiktir.

Akan şifreler çoğunlukla ikili alfabelerle gösterilirler. Örneğin, $P = C = L = Z_2$. Bu durumda, şifreleme ve şifre çözme işlemleri:

$$e_z(x) = (x + z) \mod 2$$

ve

$$d_z(y) = (y + z) \mod 2$$

dir. [20]

Örnek 3.7: m = 4 ve anahtar akım da:

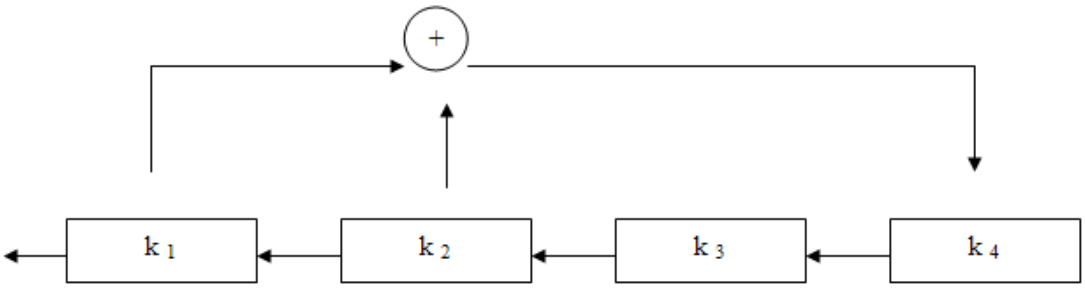
$$z_i + 4 = z_i + z_{i+1} \mod 2, (i \geq 1)$$

kuralı yaratılarak oluşturulmuş olsun.

Eğer anahtar akımı (0, 0, 0, 0)'dan farklı herhangi bir vektör ile başlatılmışsa, o zaman periyodu 15 olan bir anahtar akımı elde edilir. Örneğin, (1, 0, 0, 0) ile başlayan anahtar akım:

1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 1, ...

dir.



Şekil 3.2: Lineer Geribeslemeli Kaydırma Yazmacı (LFSR)

Bir kaydırma kaydını m safhada kullanırız. (k_1, …, k_m) vektörü kaydırma kaydını başlatmak için kullanılır. Her bir zaman biriminde, aşağıdaki işlemler gerçekleştirilir:

1. k_1 bir sonraki anahtar akım biti olarak bağlantı kurulur.
2. k_2, …, k_m sola doğru bir safha kaydırılır.
3. k_m'in yeni değeri,

$$\sum_{j=0}^{m-1} c_j k_{j+1}$$

ile hesaplanır. (Bu "lineer geribeslemedir.")

Örnek 3.8: Anahtar K = 8 ve düzmetin

```
rendezvous
```

olsun.

Öncelikle düzmetin Tablo 3.1'den yararlanarak bir sıra tamsayıya dönüştürülür:

```
17  4 13  3  4 25 21 14 20 18
```

Anahtar akım Tablo 3.1'e göre aşağıdaki gibidir. Burada ilk elemanın yerine anahtar konulmuştur, sonra düzmetnin boyutunu tamamlamak için ilgili olarak sırayla baştan elemanlar alınarak

```
8 17  4 13  3  4 25 21 14 20
```

elde edilir.

Bu iki dizideki ilgili elemanlar mod 26'ya göre toplanarak

25 21 17 16 7 3 20 9 8 12

elde edilir.

ve alfabetik şekilde şifreli metin Tablo 3.1'den yararlanılarak

ZVRQHDUJIM

olarak bulunur

Alice ise şifreli metni ilk olarak Tablo 3.1'den yararlanarak alfabetik diziyi nümerik diziye çevirerek

25 21 17 16 7 3 20 9 8 12

elde eder.

Daha sonra sırasıyla $x_1, x_2, \ldots$ hesaplar:

$$x_1 = d_8(25) = (25\text{-}8) \bmod 26 = 17$$

$$x_2 = d_{17}(21) = (21\text{-}17) \bmod 26 = 4$$

Her seferinde başka bir düzmetin karakteri bulur ve bulduğu her karakteri bir sonraki anahtar akım elemanını bulmak için kullanır.

Tanım 3.12: Autokey Şifresi:

$P = C = \mathcal{K} = L = Z_{26}$, $z_1 = K$ ve $z_i = x_{i-1}$ $(i \geq 2)$ olsun. $0 \leq z \leq 25$ için,

$$e_z(x) = x + z \bmod 26$$

ve

$$d_z(y) = y - z \bmod 26$$

$(x, y \in Z_{26})$ tanımlanır.

Autokey Şifresinde sadece 26 olası anahtar olduğu için güvenli bir şifreleme değildir. [20]

3.2. Kriptoanaliz

Bu bölümde bazı kriptoanaliz tekniklerinden bahsedilecektir. Genel olarak Oscar'ın kullanılan kriptosistemi bildiği varsayılır ve buna Kerckhoff prensibi denir. Tabiki, Oscar kullanılan kriptosistemi bilmiyorsa şifreyi çözebilmesi daha zor olacaktır. Burada Kerckhoff'un prensibine göre bir kriptosistemi tasarlanacaktır. Yani Oscar'ın kullanılan kriptosistemi bildiği varsayılacaktır.

İlk olarak, kriptoanaliz tiplerini tanımlayalım:

Sadece şifrelimetin
Oscar, y şifrelimetin dizisine hükmeder.

Bilinen düzmetin
Oscar x düzmetin dizisine ve ona bağlı olarak da y şifrelimetin dizisine hükmeder.

Seçilen düzmetin
Oscar, şifreleme makinasına geçici olarak erişim sağlar. Böylece, bir x düzmetin dizisi seçebilir ve ona bağlı olarak bir y şifrelimetin dizisi yapılandırabilir.

Seçilen şifrelimetin
Oscar, şifre çözme makinasına geçici olarak erişim elde eder. Böylece, bir y şifrelimetin dizisi seçebilir ve ona bağlı olan bir x düzmetin dizisi yapılandırabilir.

Her durumda amaç kullanılan anahtarı belirlemektir. Bunlar arasındaki en zayıf tip kriptoanaliz sadece şifrelimetin olandır.

Beker ve Piper'a göre Tablo 3.5 verilebilir.

Tablo 3.5: 26 Harfin meydana gelme olasılıkları:

harf	olasılık	harf	olasılık
A	.082	N	.067
B	.015	O	.075
C	.028	P	.019
D	.043	Q	.001
E	.127	R	.060
F	.022	S	.063
G	.020	T	.091
H	.061	U	.028
I	.070	V	.010
J	.002	W	.023
K	.008	X	.001
L	.040	Y	.020
M	.024	Z	.001

Tablo 3.4'e göre 26 harf aşağıdaki gibi 5'li gruplara ayrılabilir:

1. E, 0.120 civarında olasılığı olan
2. T, A, O, I, N, S, H, R 0.06 ve 0.09 arasında olasılığı olanlar
3. D, L 0.04 civarında olasılığı olanlar
4. C, U, M, W, F, G, Y, P, B 0.015 ile 0.023 arasında olasılığı olanlar
5. V, K, J, X, Q, Z 0.01'den daha az olasılığı olanlar.

Bunlar dışında en çok yanyana kullanılan 30 ikili harfler: TH, HE, IN, ER, AN, RE, ED, ON, ES, ST, EN, AT, TO, NT, HA, ND, OU, EA, NG, AS, OR, TI, IS, ET, IT, AR, TE, SE, HI VE OF. Yine yanyana çokca kullanılan 12 harf: THE, ING, AND, HER, ERE, ENT, THA, NTH, WAS, ETH, FOR ve DTH. [20]

3.2.1. Affine Şifresinin Kriptoanalizi

Örnek 3.9: Affine şifresinden elde edilen şifreli metin aşağıdaki gibi olsun:

FMXVEDKAPHFERBNDKRXRSREFMORUDSDKDVSHVUFEDK
APRKDLYEVLRHHRH

Bu şifreli metnin frekans analizi Tablo 3.6'da verilmiştir.

Tablo 3.6: 26 Şifreli metin harflerinin kullanım sıklığı:

harf	frekans	harf	frekans
A	2	N	1
B	1	O	1
C	0	P	3
D	6	Q	0
E	5	R	8
F	4	S	3
G	0	T	0
H	5	U	2
I	0	V	4
J	0	W	0
K	5	X	2
L	2	Y	1
M	2	Z	0

Şifreli metinde sadece 57 karakter vardır. Ancak bu bir Affine şifresini kriptoanaliz etmek için yeterlidir. Ençok kullanılan şifreli metin karakterleri şunlardır: R (8 kere tekrar etmiş), D (6 kere tekrar etmiş), E, H, K (herbiri 5 kere tekrar etmiş) ve F, S, V (herbiri 4 kere tekrar etmiş). İlk tahmin olarak, e harfinin şifresi R ve t harfinin şifresi de D olarak düşünülebilir. (e ve t en çok kullanılan harflerdir.) Bu tahmin Tablo 3.1 yardımıyla nümerik olarak ifade edilirse, $e_k(4) = 17$ ve $e_k(19) = 3$ bulunur. a ve b

bilinmeyenler olmak üzere $e_k(x) = ax + b$ denklemi ele alınsın. Böylece iki bilinmeyenli iki eşitlik elde edilir:

$$4a + b = 17 \quad (3.20)$$

$$19a + b = 3 \quad (3.21)$$

(3.20) ve (3.21) denklem sistemi Z_{26}'da çözülürse a = 6 ve b = 19 olan tek bir çözüme sahiptir. Ancak bu doğru olmayan bir anahtardır. Çünkü, OBEB(a,26) = 2 > 1. Böylece ilk tahmin yanlış çıkmıştır.

Bir sonraki tahminimiz, e harfinin şifresinin R ve t harfinin şifresinin E olduğu durum olabilir. Ancak yukarda yapılan işlemlerle bu seferde a = 13 bulunur ki buda doğru bir tahmin etme değildir. Bir sonraki olasılık, e harfinin şifresinin R ve t harfinin şifresinin H olduğu durumdur. Ancak burada da a = 8 olduğu için yanlış bir tahmindir. Bu şekilde devam ederek, e harfinin şifresinin R ve t harfinin şifresinin K olduğunu varsayalım. Bu durumda da en azından olası bir anahtar elde edilir. Çünkü a = 3 ve b = 5 olarak hesaplanır. K = (3,5)'e göre şifreli metnin şifresi anlamlı bir dizi elde edilecek mi diye kontrol edilmek üzere çözülür. Bu şekilde (3,5)'in geçerliliği kontrol edilir.

Bu işlemleri gerçekleştirerek, Tanım 3.6'ya göre $d_k(y) = 9y - 19$ elde edilir ve verilen şifreli metnin şifresi aşağıdaki gibi çözülür:

algorithmsarequitegeneraldefinitionsofarit
hmeticprocesses

Doğru anahtarın bulduğu ispatlamış olundu. [20]

3.2.2. Kaydırma Şifresinin Kriptoanalizi

Örnek 3.10: Kaydırma şifresinden elde edilen şifreli metin aşağıdaki gibi olsun:

```
YIFQFMZRWQFYVECFMDZPCVMRZWNMDZVEJBTXCDDUMJ
NDIFEFMDZCDMQZKCEYFCJMYRNCWJCSZREXCHZUNMXZ
NZUCDRJXYYSMRTMEYIFZWDYVZVYFZUMRZCRWNZDZJJ
XZWGCHSMRNMDHNCMFQCHZJMXJZWIEJYUCFWDJNZDIR
```

Bu şifreli metnin frekans analizi Tablo 3.7'de verilmiştir.

Tablo 3.7: 26 Şifreli metin harflerinin kullanım sıklığı:

harf	frekans	harf	frekans
A	0	N	9
B	1	O	0
C	15	P	1
D	13	Q	4
E	7	R	10
F	11	S	3
G	1	T	2
H	4	U	5
I	5	V	5
J	11	W	8
K	1	X	6
L	0	Y	10
M	16	Z	20

Z, diğer şifreli metin karakterlerinden daha fazla kullanıldığı için, $d_k(Z) = e$ varsayımı yapılabilir. Geri kalan şifreli metin karakterlerinden en az 10 kere tekrarlananlar da: C, D, F, J, M, R, Y. Bu harflerin, t, a, o, i, n, s, h, r kümesinin şifrelenmiş şekli olduğu düşünülebilir. Ancak harflerin tekrar sayıları yeterli derecede bilgi vermemektedir.

Bu nedenle, özellikle –Z ve Z– şeklindeki ikililerin kullanım çokluklarına bakılmalıdır. Örneğin, DZ ve ZW (herbiri 4 kere kullanılmış); NZ ve ZU (herbiri 3 kere kullanılmış); ve RZ, HZ, XZ, FZ, ZR, ZV, ZC, ZD, VE ZJ (herbiri 2 kere

kullanılmış). WZ, 4 kere tekrarlanmış; ancak, ZW hiç tekrarlanmadığından dolayı, W diğer birçok karakterden daha az tekrarlanmıştır. Bu nedenle $d_K(W) = d$ varsayımı yapılabilir. DZ 4 kere ve ZD de 2 kere tekrarlandığı için $D_K(D) \in \{r,s,t\}$ olarak düşünülebilir. Ancak üç olasılıktan hangisinin doğru olduğu açık olarak görülmemektedir.

Eğer $d_K(Z) = e$ ve $d_K(W) = d$ tahminlemesi üzerinde hareket edilip tekrar şifreli metne bir göz atılırsa, ZRW ve RZW'nun şifreli metnin başlarına doğru bir yerlerde kullanıldığı ve RW'nun da daha sonra tekrar kullanıldığı farkedilir. R, şifreli metinde sıkça kullanıldığından ve nd'de oldukça kullanılan bir ikili olduğundan dolayı $d_K(R)=n$ olasılığı denenebilir.

Bu varsayımla

```
------end---------e----ned---e------------
YIFQFMZRWQFYVECFMDZPCVMRZWNMDZVEJBTXCDDUMJ

--------e----e---------n--d---en----e----e
NDIFEFMDZCDMQZKCEYFCJMYRNCWJCSZREXCHZUNMXZ

-e---n------n------ed---e---e--ne-nd-e-e--
NZUCDRJXYYSMRTMEYIFZWDYVZVYFZUMRZCRWNZDZJJ

-ed-----n-----------e----ed-------d---e--n
XZWGCHSMRNMDHNCMFQCHZJMXJZWIEJYUCFWDJNZDIR
```

elde edilir.

Bir sonraki adım, NZ ortak bir ikili olduğu ve ZN olmadığı için $d_K(N) = h$'ı denemek olabilir. Eğer bu doğruysa, o zaman ne – ndhe düzmetin parçası $d_K(C) = a$ olmasını gerektirir. Bu tahminlere bağlı olarak

```
------end-----a---e-a--nedh--e------a-----
YIFQFMZRWQFYVECFMDZPCVMRZWNMDZVEJBTXCDDUMJ

h-------ea---e-a---a---nhad-a-en--a-e-h--e
NDIFEFMDZCDMQZKCEYFCJMYRNCWJCSZREXCHZUNMXZ

he-a-n------n------ed---e---e--neandhe-e--
NZUCDRJXYYSMRTMEYIFZWDYVZVYFZUMRZCRWNZDZJJ


-ed-a---nh---ha---a-e----ed-----a-d--he--n
XZWGCHSMRNMDHNCMFQCHZJMXJZWIEJYUCFWDJNZDIR
```

elde edilir.

Şimdide ikinci en çok kullanılan şifreli metin karakteri olarak da M'yi düşünülsün. $d_K(M)=i$ ya da o olabilir. (CM) ai, ao'dan daha mantıklı geldiği için öncelikle $d_K(M)=i$ olması durumu denensin. Bu şekilde

```
-----iend-----a-i-e-a-inedhi-e------a---i-
YIFQFMZRWQFYVECFMDZPCVMRZWNMDZVEJBTXCDDUMJ

h-----i-ea-i-e-a---a-i-nhad-a-en--a-e-hi-e
NDIFEFMDZCDMQZKCEYFCJMYRNCWJCSZREXCHZUNMXZ

he-a-n-----in-i----ed---e---e-ineandhe-e--
NZUCDRJXYYSMRTMEYIFZWDYVZVYFZUMRZCRWNZDZJJ

-ed-a--inhi--hai--a-e-i--ed-----a-d--he--n
XZWGCHSMRNMDHNCMFQCHZJMXJZWIEJYUCFWDJNZDIR
```

elde edilir.

Bir sonraki aşamada o harfinin hangi harfle şifrelendiği belirlemeye çalışılacak. Mümkün olan olasılıklar: D, F, J, Y. Y diğerlerinden daha mümkünmüş gibi gözükmektedir. Bu nedenle $d_E(Y) = o$ olduğunu varsayalım.

Üç tane en çok kullanılan şifreli metin harfleri de D, F, J'dir ve bu harfler r, s, t 'nin herhangi bir sırasını şifrelemek için kullanılabilir. N M D 'den $d_E(D) = s$ olarak ve H N C M F 'den $d_E(F) = r$, $d_E(H) = c$ ve $d_E(J) = t$ olarak bulunur. Şifreli metin tekrar düzenlenirse

```
o-r-riend-ro--arise-a-inedhise--t---ass-it
YIFQFMZRWQFYVECFMDZPCVMRZWNMDZVEJBTXCDDUMJ

hs-r-riseasi-e-a-orationhadta-en--ace-hi-e
NDIFEFMDZCDMQZKCEYFCJMYRNCWJCSZREXCHZUNMXZ

he-asnt-oo-in-i-o-redso-e-ore-ineandhesett
NZUCDRJXYYSMRTMEYIFZWDYVZVYFZUMRZCRWNZDZJJ

-ed-ac-inhischair-aceti-ted--to-ardsthes-n
XZWGCHSMRNMDHNCMFQCHZJMXJZWIEJYUCFWDJNZDIR
```

elde edilir.

Artık Örnek 3.10 için düzmetni ve anahtarı belirlemek çok kolaydır: [20]

> Our friend from Paris examined his empty glass with surprise, as if evaporation had taken place while he wasn't looking. I poured some more wine and he settled back in his chair , face tilted up towards the sun.[4]

[4] P. Mayle, A Year in Provence. A. Knopf, Inc., 1989

3.2.3. Vigenere Şifresinin Kriptoanalizi

Bu bölümde Vigenere şifresinin kriptoanalizi için bazı yöntemleri tanımlanacak. İlk adım, m ile gösterilen anahtar kelimenin uzunluğunu belirlemektir. Bunun için kullanılan birkaç teknik vardır. Bunlardan ilki Kasiski testi olarak adlandırılır, ikincisi ise rastlantı indeksidir.

Kasiski testi ilk olarak 1863 yılında Friedrich Kasiski tarafından tanımlanmıştır. Düzmetnin iki özdeş bölümü düzmetinde x durum uzaklıkta oluştuğu zaman aynı şifreli metne şifrelenebilir ($x \equiv 0 \bmod m$). Tersine, eğer şifreli metnin iki özdeş bölümü gözlemlenirse, o zaman düzmetnin ilgili bölümüyle ilgili iyi bir şans vardır.

Kasiski testi şöyle çalşır. Öncelikle şifreli metinde en azından üç uzunluklu özdeş bölüm çiftleri aranır ve iki bölümün başlangıç durumları kaydedilir. Eğer çeşitli başlangıç durumları d_1, d_2, … elde edilmişse, m'nin d_i'lerin en büyük ortak bölenini böldüğü varsayılabilir.

m değeri için bir başka kanıt rastlantı indeksinden elde edilebilir. 1920'de Wolfie Friedman tarafından Tanım 3.13'teki gibi tanımlanmıştır.

Tanım 3.13: $\mathbf{x} = x_1x_2 \ldots x_n$, n alfabetik karakterli bir dizi olsun. $I_c(\mathbf{x})$ ile gösterilen $\mathbf{x}$'in rastlantı indeksi $\mathbf{x}$'in iki rastgele elemanının özdeş olması olasılığıyla tanımlanır. Sırasıyla $\mathbf{x}$'te *A, B, C, ... , Z*'nin sıklıkları $f_0, f_1, \ldots, f_{25}$ ile gösterilsin. $\mathbf{x}$'in iki elemanı $\binom{n}{2}$ yolla seçilir. Her i $(0 \le i \le 25)$ için her iki elemanın i olması için $\binom{f_i}{2}$ yol vardır. O zaman, rastlantı indeksi

$$I_c(x) = \frac{\sum_{i=0}^{25} f_i(f_i - 1)}{n(n-1)} \tag{3.22}$$

formülüyle elde edilir. [20]

Örnek 3.11: Vigenere şifresinden elde edilen şifreli metin:

```
CHREEVOAHMAERATBIAXXWTNXBEEOPHBSBQMQEQERBW
RVXUOAKXAOSXXWEAHBWGJMMQMNKGRFVGXWTRZXWIAK
LXFPSKAUTEMNDCMGTSXMXBTUIADNGMGPSRELXNJELX
VRVPRTULHDNQWTWDTYGBPHXTFALJHASVBFXNGLLCHR
ZBWELEKMSJIKNBHWRJGNMGJSGLXFEYPHAGNRBIEQJT
AMRVLCRREMNDGLXRRIMGNSNRWCHRQHAEYEVTAQEBBI
PEEWEVKAKOEWADREMXMTBHHCHRTKDNVRZCHRCLQOHP
WQAIIWXNRMGWOIIFKEE
```

olsun.

Öncelikle Kasiski testini deneyelim. CHR şifreli metin dizisi şifreli metin içinde 1, 166, 236 ve 286 pozisyonlarında başlayan dört yerde tekrarlanmıştır. Birinci tekrardan diğer üç tekrara olan mesafeler: 165, 235, ve 285'dir. Bu üç tamsayının OBEB'i 5'tir. Bu da anahtar kelime uzunluğudur.

Şimdide rastlantısal indislerin hesaplanmasıyla aynı sonucun elde edilip edilemeyeceğine bakalım. Tablo 3.7 ve 3.22 ifadesi yardımıyla m = 1 ile rastlantısal indeks 0.045'tir; m = 2 ile iki indis 0.046 ve 0.041'dir; m = 3 ile 0.043, 0.050, 0.047 elde edilir; m = 4 ile 0.042, 0.039, 0.046, 0.040 indisleri elde edilir; daha sonra m = 5 denenir ve 0.063, 0.068, 0.069, 0.061 ve 0.072 değerleri elde edilir. Bu da anahtar kelimenin uzunluğunun beş olduğunu kanıtlar.

Bu varsayım altında anahtar kelime nasıl belirlenecek? Bunun için Tanım 3.14 ile iki dizinin rastlantısal ortak indeksi incelenebilir. [20]

Tanım 3.14: $x = x_1x_2\ldots x_n$ ve $y = y_1y_2\ldots y_{n'}$, n ve n' alfabetik karakterlerin dizileri olsun. x ve y'nin $MI_c(x,y)$ ile gösterilen rastlantısal ortak indeksi, x'in rastgele bir elemanının y'nin rastgele bir elemanı ile özdeş olması olasılığını tanımlar. Eğer x ve y deki A, B, C, …, Z'nin frekansları $f_o,f_1,..,f_{25}$ ve $f_o',f_1',...f_{25}'$ olarak gösterilirse o zaman $MI_c(x,y)$ (3.23)'teki gibi ifade edilir: [19]

$$MI_c(x,y) = \frac{\sum_{i=0}^{25} f_i f_i'}{nn'} \tag{3.23}$$

K = (k_1,k_2,..,k_m) anahtar kelime olduğunu varsayalım. $MI_c(y_i,y_j)$'nin tahmin edilip edilemeyeceğine bakalım. y_i ve y_j'de rastgele birer karakter düşünülsün. Her iki karakterin *A* olma olasılığı $p_{-k_i} p_{-k_j}$, *B* olma olasılığı $p_{1-k_i} p_{1-k_j}$, vs. (her alt simge modülo 26'ya göre indirgenmiştir). Dolayısıyla,

$$MI_c(y_i,y_j) \approx \sum_{h=0}^{25} p_{h-k_i} p_{h-k_j} = \sum_{h=0}^{25} p_h p_{h+k_i-k_j}$$

olduğu tahmini hesaplanır. [20]

Dikkat edilirse bu tahmindeki değer sadece (k_i-k_j) mod 26 arasındaki farklılığa bağlıdır. Buna y_i ve y_j'nin ilişkili kaydırması denir. Aynı zamanda,

$$\sum_{h=0}^{25} p_h p_{h+l} = \sum_{h=0}^{25} p_h p_{h-l}$$

eşitliğinden,
l'nin ilişkili bir kaydırması, 26-*l*'nin ilişkili bir kaydırması gibi aynı MI_c tahminini kabul eder.

0 ile 13 arasındaki ilişkili kaydırmalar için yapılan tahminler Tablo 3.7'de gösterilmiştir.

Burada dikkat edilmesi gereken nokta, ilişkili kaydırma sıfırdan farklıysa bu tahminler 0.031 ve 0.045 arasında değişmektedir. İlişkili kaydırmanın sıfır olması durumunda 0.065 tahmin olarak kabul edilir. Bu durumu dikkate alarak, y_i ve y_j'nin ilişkili kaydırması olarak $1 = K_i - K_j$ gibi bir formül çıkartılabilir. y_i'nin sabitlediği varsayılsın ve e_0,e_1,e_2 ile y_j şifrelemesinin etkisi düşünülsün. Sonuç dizileri y_j^0,y_j^1, … vb. ile gösterilsin. MI_c (y_i,y_j^g) , $0 \le g \le 25$ indislerini hesaplamak kolaydır. Bu (3.24)'teki formül kullanılarak yapılır: [20]

$$MI_c\left(x, y^g\right) = \frac{\sum_{i=0}^{25} f_i f'_{i-g}}{nn'} \tag{3.24}$$

Tablo 3.8 Beklenen Rastlantısal İndisler**:**

İlişkili kaydırma	Beklenen MI_c değeri
0	0.065
1	0.039
2	0.032
3	0.034
4	0.044
5	0.033
6	0.036
7	0.039
8	0.034
9	0.034
10	0.038
11	0.045
12	0.039
13	0.043

g = l ve y_i ve y_j'nin ilişkisel kaydırmaları sıfır olduğu zaman, MI_c 0.065'e yakın bir değer olmalıdır. g ≠ l'nin değerleri 0.031 ile 0.045 arasında değişir.

Bu teknik kullanılarak y_i altdizilerinin herhangi iki ilişkisel kaydırmaları elde edilebilir. Bu da kolayca elde edilebilecek 26 olası anahtar kelimeye karşı gelmektedir.

Örnek 3.11'e yardımıyla bu durumu açıklamaya çalışalım.

Örnek 3.11 (Devamı)

Anahtar kelimenin uzunluğu Örnek 3.11'in ilk kısmında 5 olarak belirlenmişti. Şimdi de ilişkili kaydırmalar hesaplanmaya çalışılsın. Bilgisayar yardımı ile $1 \le i < j \le 5,0 \le g \le 25$ olduğu yerlerde; 260 $MI_c(y_i,y_j^g)$ değerini hesaplamak zor bir işlem değildir. Bu değerler Tablo 3.8'de gösterilmiştir. Her (i,j) çifti için, 0.065'e yakın olan $MI_c\left(y_i, y_j^g\right)$ değerleri aranmıştır.

Tablo 3.9 Elde edilen Ortak Rastlantısal İndisler:

i	j	$MI_c(y_i,y_j^g)$ değeri								
1	2	.028	.027	.028	.034	.039	.037	.026	.025	.052
		.068	.044	.026	.037	.043	.037	.043	.037	.028
		.041	.041	.034	.037	.051	.045	.042	.036	
1	3	.039	.033	.040	.034	.028	.053	.048	.033	.029
		.056	.050	.045	.039	.040	.036	.037	.032	.027
		.037	.036	.031	.037	.055	.029	.024	.037	
1	4	.034	.043	.025	.027	.038	.049	.040	.032	.029
		.034	.039	.044	.044	.034	.039	.045	.044	.037
		.055	.047	.032	.027	.039	.037	.039	.035	
1	5	.043	.033	.028	.046	.043	.044	.039	.031	.026
		.030	.036	.040	.041	.024	.019	.048	.070	.044
		.028	.038	.044	.043	.047	.033	.026	.046	
2	3	.046	.048	.041	.032	.036	.035	.036	.030	.024
		.039	.034	.029	.040	.067	.041	.033	.037	.05
		.033	.033	.027	.033	.045	.052	.042	.030	
2	4	.046	.034	.043	.044	.034	.031	.040	.045	.040
		.048	.044	.033	.024	.028	.042	.039	.026	.034
		.050	.035	.032	.040	.056	.043	.028	.028	
2	5	.033	.033	.036	.046	.026	.018	.043	.080	.050
		.029	.031	.045	.039	.037	.027	.026	.031	.039
		.040	.037	.041	.046	.045	.043	.035	.030	
3	4	.038	.036	.040	.033	.036	.060	.035	.041	.029
		.058	.035	.035	.034	.053	.030	.032	.035	.036
		.036	.028	.046	.032	.051	.032	.034	.030	
3	5	.035	.034	.034	.036	.030	.043	.043	.050	.025
		.046	.048	.041	.032	.036	.035	.036	.030	.024
		.027	.030	.072	.035	.034	.032	.043	.027	
4	5	.052	.038	.033	.038	.041	.043	.037	.048	.024
		.028	.036	.061	.033	.033	.032	.052	.034	.027
		.0392	.043	.033	.027	.030	.039	.048	.035	

Tablo 3.9'da 6 değer kutu içine alınmıştır. Buna göre bu değerler y_1 ve y_2'nin ilişkisel kaymasının 9, y_1 ve y_5'in ilişkisel kaymasının 16, y_2 ve y_3'ün ilişkisel kaymasının 13, y_2 ve y_5'in ilişkisel kaymasının 7, y_3 ve y_5'in ilişkisel kaymasının 20 ve y_4 ve y_5'in ilişkisel kaymasının 11 olduğunu gösteren güçlü kanıtlardır. Bu sonuçlardan k_1, k_2, k_3, k_4, k_5 beş tane bilinmeyen olmak üzere aşağıdaki eşitlikler elde edilir:

$$k_1 - k_2 = 9$$
$$k_1 - k_5 = 16$$
$$k_2 - k_3 = 13$$
$$k_2 - k_5 = 7$$
$$k_3 - k_5 = 20$$
$$k_4 - k_5 = 11$$

Bu, i = 2, 3, 4 ve 5 için k_i'lerin k_1 cinsinden yazılabilmesine olanak verir:

$$k_2 = k_1 + 17$$
$$k_3 = k_1 + 4$$
$$k_4 = k_1 + 21$$
$$k_5 = k_1 + 10$$

Herhangi $k_1 \in Z_{26}$ için, anahtar; (k_1, k_1+17, k_1+4, k_1+21, k_1+10) şeklindedir. İlk olarak k_1=A için anahtar kelimenin AREVK'nin devirli kaydırması olduğu düşünülebilir. Devam ederek anahtar kelimenin JANET olduğunu belirlemek uzun zaman almaz. Şifrenin tamamıyla çözülmüş şekli aşağıda verilmiştir:

> The almond tree was in tentative blossom. The days were longer, often ending with magnificient evenings of corrugated pink skies. The hunting season was over, with hounds and guns put away for six months. The vineyards were busy again as the well-organized farmers treated their vines and the more lackadaisical neighbors hurried to do the pruning they should have done in November.

[20]

3.2.4. Hill Şifresinin Kriptoanalizi

Hill şifresinin, sadece-şifreli metin analizi düzmetin analizinden daha zordur. İlk başta Oscar'ın kullanılan m değerini belirlediğini varsayalım. $1 \leq j \leq m$ için $y_i=e_K(x_j)$ olmak üzere $x_j = (x_{1,j} , x_{2,j} , ..., x_{m,j})$ ve $y_j = (y_{1,j} , y_{2,j} , ..., y_{m,j})$ değerlerini elde etmiş olsun. Eğer $X = (x_{i,j})$ ve $Y = (y_{i,j})$ $m \times m$'lik matrisleri tanımlanmışsa, K bilinmeyen anahtar olmak üzere Y = XK matris eşitliği yazılabilir. Y matrisi tersi alınabilir bir matris ise, Oscar $K = X^{-1}Y$ anahtarını hesaplayıp sistemi kırabilir. Değilse başka bir grup m düzmetin-şifreli metin çifti alınır.

Konuya açıklık getirmek için Örnek 3.12'yi göz önüne alalım.

Örnek 3.12:
m = 2 ile Tanım 3.9'a göre Hill Cipher kullanarak friday düzmetninin şifrelendiğini ve PQCFKU şifreli metninin elde edildiği varsayılsın.

Böylelikle, $e_K(5,17) = (15,16)$, $e_K(8,3) = (2,5)$, $e_K(0,24) = (10,20)$ elde edilir. İlk iki düzmetin-şifreli metin ikilisinden, (3.25)'teki matris eşitliğini elde edilir:

$$\begin{bmatrix} 15 & 16 \\ 2 & 5 \end{bmatrix} = \begin{bmatrix} 5 & 17 \\ 8 & 3 \end{bmatrix} K \qquad (3.25)$$

Teorem 3.3 kullanılarak kolayca (3.26)'daki matris hesaplanır:

$$\begin{bmatrix} 5 & 17 \\ 8 & 3 \end{bmatrix}^{-1} = \begin{bmatrix} 9 & 1 \\ 2 & 15 \end{bmatrix} \tag{3.26}$$

böylece,

$$K = \begin{bmatrix} 9 & 1 \\ 2 & 15 \end{bmatrix} \begin{bmatrix} 15 & 16 \\ 2 & 5 \end{bmatrix} = \begin{bmatrix} 7 & 19 \\ 8 & 3 \end{bmatrix}$$

elde edilir.

Eğer m bilinmiyorsa ne yapılmalı? O zaman, m'nin çok büyük olmadığı varsayılarak $m = 2,3,\ldots$ değerleri denenerek anahtar bulunmaya çalışılır. [20]

3.2.5 LFSR-tabanlı Akım Şifresinin Kriptoanalizi

Şifreli metin düzmetnin ve anahtar akımın modülo 2'ye göre toplamı olsun; yani, $y_i = (x_i + z_i) \bmod 2$ olsun. Anahtar akım, $c_0, c_1, \ldots, c_{m-1} \in Z_2$ (c_0=1) olmak üzere

$$z_{m+i} = \sum_{j=0}^{m-1} (c_j z_{i+j}) \bmod 2 \tag{3.27}$$

lineer özyineleme ilişkisi kullanılarak $z_1, \ldots, z_m$'den türetilir. [20]

Bu kriptosistemde bütün işlemler lineer olduğundan, kriptosistemin bir bilinmiş düzmetin saldırısına karşı savunmasız olduğundan kuşkulanılabilir; aynı Hill Şifrelemede olduğu gibi. Oscar, $x_1x_2\ldots x_n$ düzmetin dizisine ve ilişkili $y_1y_2\ldots y_n$ şifreli metin dizisine sahip olsun. O zaman Oscar anahar akım bitlerini

$$z_i = (x_i + y_i) \bmod 2 \ (1 \le i \le n) \tag{3.28}$$

yardımıyla hesaplayabilir. Oscar m değerini de biliyor olsun. Oscar bütün anahtar akımı tekrar inşa etmek için $c_0, \ldots, c_{m-1}$ anahtar akım değerlerini hesaplamalıdır.

Herhangi bir $i \ge 1$ için

$$z_{m+i} = \sum_{j=0}^{m-1} (c_j z_{i+j}) \bmod 2$$

m bilinmeyenli bir lineer denklem oluşur. Eğer $n \geq 2m$ ise, o zaman m bilinmeyenli m denklem vardır.

m bilinmeyenli denklem sistemi

$$\left(z_{m+1}, z_{m+2}, \ldots, z_{2m}\right) = \left(c_0, c_1, \ldots, c_{m-1}\right) \begin{bmatrix} z_1 & z_2 & \cdots & z_m \\ z_2 & z_3 & \cdots & z_{m+1} \\ \vdots & \vdots & \ddots & \vdots \\ z_m & z_{m+1} & \cdots & z_{2m-1} \end{bmatrix} \tag{3.29}$$

şeklinde yazılabilir.

Katsayı matrisinin modülo 2'ye göre tersi varsa,

$$\left(c_0, c_1, \ldots, c_{m-1}\right) = \left(z_{m+1}, z_{m+2}, \ldots, z_{2m}\right) \begin{bmatrix} z_1 & z_2 & \cdots & z_m \\ z_2 & z_3 & \cdots & z_{m+1} \\ \vdots & \vdots & \ddots & \vdots \\ z_m & z_{m+1} & \cdots & z_{2m-1} \end{bmatrix}^{-1} \tag{3.30}$$

elde edilir.

Bunu Örnek 3.13 ile açıklamaya çalışalım.

Örnek 3.13: Oscar'ın aşağıdaki şifrelimetni

101101011110010

ve buna bağlı olarak düzmetin dizisini

011001111111001

ele geçirdiğini varsayalım.

Oscar daha sonra anahtar akım bitlerini (3.27)'ye göre hesaplar:

110100100001010.

Aynı zamanda Oscar'ın, anahtar akımın 5-seviyeli bir LFSR kullanılarak oluşturulduğunu bildiğini varsayalım. Daha sonra (3.29)'e göre ilk 10 keystream bitinden elde edilen

$$(0, 1, 0, 0, 0) = (c_0, c_1, c_2, c_3, c_4)\begin{bmatrix} 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

matris eşitliğini çözsün.

$$\begin{bmatrix} 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}^{-1} = \begin{bmatrix} 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 \end{bmatrix}$$

olduğu denetlenebilir.

Bu (3.30)'a göre

$$(c_0, c_1, c_2, c_3, c_4) = (0, 1, 0, 0, 0)\begin{bmatrix} 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 \end{bmatrix}$$

$$= (1, 0, 0, 1, 0).$$

sonucunu sağlar. Böylece özyineleme

$$z_i + 5 = z_i + z_{i+3} \mod 2$$

anahtar akımını yaratmakta kullanılır. [20]

4. KRİPTOGRAFİ ÇEŞİTLERİ

Kriptografiyi iki başlıkta incelemek mümkündür. Bunlardan biri Simetrik Anahtarlı Kriptografi (Symmetric-key cryptography); diğeri ise Açık Anahtarlı Kriptografidir (Public-key cryptography).

4.1. Simetrik Anahtarlı Kriptografi

Simetrik Anahtarlı Kriptografi'de alıcı ve göndericinin aynı anahtarı paylaştığı şifreleme yöntemidir. Bu 1976'ya kadar açıkça bilinen tek şifreleme çeşidiydi.

Modern simetrik anahtarlı şifreleme çalışmaları blok şifrelerin (block cipher) ve akan şifrelerin (stream cipher) çalışması ve onların uygulamalarıyla ilgilidir. Blok bir şifre Alberti'nin çok alfabeli ikame şifrelerinin modern bir somutlaşmasıdır: Blok şifreler, girdi olarak bir düzmetin bloğu ve bir anahtar alır ve çıktı olarak aynı uzunlukta şifreli metin bloğu verir. Mesajlar daima bir bloktan daha uzun olduğundan blokları başarıyla birleştiren bazı yöntemler gereklidir. Diğerlerine göre görüşlere göre biri diğerinden daha güvenli olduğu düşünülen pek çok yöntem geliştirilmiştir. Bunlar uygulama biçimleri olup bir kriptosistemde bir blok şifreleme kullanırken özenle dikkat edilmelidir.

Veri Şifreleme Standardı (Data Encryption Standard (DES)) ve İleri Şifreleme Standardı (Advanced Encryption Standard (AES)) Amerika Birleşik Devletleri (A.B.D.) hükümetince kriptografi standartları olarak belirlenen blok şifreleme tasarımlarıdır. Resmi bir standart olarak karşı koyulduğu için, DES oldukça popüler kalmıştır; ATM şifrelemelerinde, e-posta güvenliğinde ve güvenli uzak erişimde gibi pek çok geniş uygulama alanında kullanılmaktadır. Pek çok blok şifreleme nitel olarak çok fazla çeşitlilikte tasarlanmış ve sunulmuştur. Çoğu tamamen kırılmıştır.

Blok çeşidine karşıt olarak akan şifreler biraz tek zamanlı blok (one-time-pad) gibi düzmetinde bit-bit ya da karakter-karakter birleştirilen keyfi uzunlukta akan bir

anahtar malzeme yaratır. Akan bir şifrede, çıktı akım şifreleri işleterek değişen bir iç durum taban alarak yaratılır. Durumların değişimi anahtarla denetlenir; bazı akan şifrelerdeyse düzmetin akımlarıyla da denetlenir. RC4 çok iyi bilinen bir akan şifreye örnektir.

Mesaj özet fonksiyonları olarak da adlandırılan şifreyle ilgili hash fonksiyonları (hash function) anahtar kullanmazlar; ama, şifreyle ilgili algoritmaların ilişkili ve önemli bir sınıfıdır. Girdi verisini alırlar (çoğu kez bütün bir mesajı) ve çıktı olarak kısa ve sabit uzunlukta bir hash verir; bunu da tek yönlü bir fonksiyonla yapar.

Mesaj doğrulama kodları (message autentication codes (MACs)) alındığında hash değerini doğrulamak için gizli bir anahtar kullanmak dışında hash fonksiyonları gibidirler. [5]

4.2. Açık Anahtarlı Kriptografi

Simetrik anahtarlı kriptosistemler, tipik olarak şifreleme ve deşifreleme için aynı anahtarı kullanırlar; gerçi bu mesaj ya da mesaj grupları diğerlerinden farklı bir anahtara sahip olabilir. Simetrik şifrelerin belirgin bir dezavantajı güvenli olarak kullanmak için anahtar yönetim gerekliliğidir. Haberleşen ekiplerin her ayrı çifti ideal olarak farklı bir anahtar paylaşırlar ve belki her şifreli metin de değişir. Anahtarların ihtiyaç sayısı, hepsini düzgün ve gizli tutacak karışık anahtar yönetim çizimlerine çok çabuk ihtiyaç duyan ağ üyelerinin sayısının karesi olarak yükselir.

Çığır açan bir 1976 gazetesinde, Whitfield Diffie ve Martin Hellman açık bir anahtar ve gizli bir anahtar olmak üzere iki farklı ama matematiksel olarak ilişkili anahtarların olduğu açık anahtarlı (daha genel olarak asimetrik anahtar (asymmetric key)) kriptografi düşüncesini önerdiler. Açık anahtarlı sistem, ister istemez ilişkili olsalar bile bir anahtarın (gizli anahtar) hesaplamasının diğerinden (açık anahtar) hesaplanmasının olmayacağı şekilde yapılmıştır. Ayrıca her iki anahtar da birbiriyle ilişkili olarak gizli bir şekilde yaratılır. Tarihçi David Kahn açık anahtarlı

kriptografiyi “Rönesansta çok alfabeli ikame yerine koymadan beri alandaki en devrimci yeni kavram ortaya çıktı” şeklinde tasvir etmiştir.

Açık anahtarlı kriptosistemde, açık anahtar özgürce dağıtılabilirken onun eşi gizli anahtar gizli kalmalıdır. Açık anahtar tipik olarak şifreleme için kullanılırken, gizli ya da özel anahtar da deşifreleme için kullanılır. Diffie ve Hellman açık anahtarlı şifrelemenin Diffie-Hellman anahtar değişim kuralını sunarak mümkün olduğunu gösterdiler.

1978’de Ronald Rivest, Adi Shamir ve Len Adleman bir başka açık anahtarlı sistem olan RSA’yı gerçekleştirdiler. 1997’de sonunda alenen bilinen asimetrik anahtar şifreleme GCHQ’da James H. Ellis tarafından bulunmuş ve her iki Diffie-Hellman ve RSA algoritmaları evvelce geliştirilmiştir (sırasıyla Malcolm J. Williamson ve Clifford Cocks). Diffie-Hellman ve RSA algoritmaları (ek olarak yüksek kaliteli açık anahtarlı şifrelerin ilk alenen bilinen örnekleri) geniş çapta kullanılmıştır. Geriye kalanlar Cramer-Shoup kriptosistemi, ElGamal şifreleme ve çeşitli eliptik eğri tekniklerini içerir.

Açık anahtarlı algoritmalar sayılar teorisinde hesaplanması karmaşık ve zor problemlere dayanır. Örneğin, RSA’nın zorluğu tamsayı çarpanlara ayırma problemiyle ilgiliyken, Diffie-Hellman ve DSA ayrık logaritma problemiyle ilgilidir. Çok yakın geçmişte, güvenliği eliptik eğrileri kapsayan sayı teorik problemlerine dayanan eliptik eğri kriptografi gelişmiştir. Belli başlı problemlerin zorluğundan dolayı, çoğu açık anahtarlı algoritma modüler çarpım ve üs alma gibi işlemleri kapsar. Sonuç olarak açık anahtarlı kriptosistemler hızlı yüksek kaliteli bir simetrik anahtar şifreleme algoritmalarında mesajın kendisi için kullanılan yaygın melez sistemlerdir. [5]

4.3. Kriptanaliz

Kriptoanalizin amacı, kriptografik bir plandaki bazı zayıflıkları ve güvensizliği bulmaktır. Kriptanaliz bir sistemi yıkmayı deneyen kötü niyetli bir saldırgan tarafından ya da bir sistemin savunmasızlıklarının olup olmadığını değerlendirmeye kalkışan bir sistem tasarımcısı (ya da diğerleri) tarafından üstlenilebilir. Her nasılsa modern uygulamalarda kriptografik algoritmalar ve uygulamalar, sistemin güvenliğinin herhangi güvencesini önermek için dikkatlice incelenmeli ve test edilmelidir.

Her şifreleme yönteminin kırılabilirliği yanlış bir anlamadır. Bell laboratuarlarında İkinci Dünya Savaşı işiyle bağlantılı olarak Claude Shannon anahtar malzemeleri tekrar kullanılmayan, bütün mümkün saldırganlardan saklanan ve mesaja eşit ya da daha büyük uzunlukta tamamen rasgele olarak sağlanan tek zamanlı blok şifrelerin kırılamaz olduğunu ispatlamıştır. Tek zamanlı blok şifreler dışında çoğu şifreler kaba kuvvet saldırısıyla yeterli hesaplama girişimiyle kırılabilir; ama, çabanın değeri, şifreyi kullanmak için gereken çabaya kıyaslamayla üssel olarak bağlıdır. Böyle durumlarda, etkin güvenlik başarılıdır; eğer, gereken çabanın herhangi bir hasmın yeteneklerinin ötesinde olduğu ispatlanırsa. Bu demektir ki, şifreyi kırmak için hiçbir etkin yöntem olmadığı gösterilmelidir (zaman alan kaba kuvvet yöntemine karşıt olarak). Böyle bir gösterim şu ana kadar yapılamadığından, tek zamanlı blok şifreler sadece teorik olarak kırılamaz olarak kalır.

Kriptanalitik saldırılarının geniş bir çeşitliliği vardır ve pek çok yolla sınıflandırılabilir. Yaygın bir ayrım, saldırganın neleri bildiğine ve hangi yeteneklerin uygun olduğuna bağlıdır.

- Sadece şifreli metin saldırılarında (ciphertext-only attack), kriptanalist sadece şifreli metine erişir (iyi modern şifreler genellikle sadece şifreli metin ataklarına etkin olarak bağışıklıdır).

- Bilinen bir düzmetin saldırısında (known-plaintext attack), kriptanalist şifreli metne ve ilgili olan düzmetne erişir (ya da pek çok böyle çiftlere).

- Seçilmiş bir düzmetin saldırısında (chosen-plaintext attack), kriptanalist bir düzmetin seçer ve ilgili olan şifreli metni öğrenir (muhtemelen birçok defa).

- Seçilmiş bir şifreli saldırıda (chosen-ciphertext attack), kriptanalist şifreli bir metin seçmeli ve onun düzmetnini öğrenmelidir.

Simetrik anahtar şifreleme kriptanalizi, mükemmel bir şifre karşısındaki herhangi bir saldırıdan daha etkin olan blok şifrelere ve akan şifrelere karşı saldırıların araştırılması tipik olarak gereklidir.

Açık anahtarlı algoritmalar çeşitli problemlerin hesaplama güçlüğüne dayanır. Bunların en bilineni tamsayı çarpanlara ayırmadır; ama, ayrık logaritma problemi de önemlidir. Pek çok açık anahtarlı kriptanaliz bu hesaplamalı problemleri ya da etkin olarak bir kısmını çözmek için sayısal algoritmalarla ilgilenir. Örneğin, ayrık algoritmaların eliptik ağri tabanlı sürümünü çözmek için bilinen en iyi algoritma en azından aşağı yukarı eşit boyutlu problemler için çarpanlara ayırmak için bilinen en iyi algoritmadan daha fazla zaman kazandırır. Böylece, eşit olan diğer şeyler için, eşit güçte saldırı direncine ulaşmak için çarpanlara ayırma şifreleme teknikleri eliptik eğri tekniğinkinden daha büyük anahtar kullanmalıdır. Bu nedenle, eliptik eğrilere dayanan açık anahtarlı kriptosistemler 1990'ların ortalarında icadından beri popüler oldular. [5]

4.3.1. Kriptanalitik Atakların Amaçları

Ayrılmış Ataklar (Distinguishing Attacks):
Ayrılmış Atakların başarılı olabilmesi için şifre sisteminin çıktısının rastgele bir permutasyonun çıktısından ayırılması olası olmalıdır.

Kısmi açık metin bilgisi (Partial Knowledge of the Plaintext):
Bu atakta kısmi açık metin bilgisine (Şifre sisteminin girdisi için herhangi tahmin) sahip olunur.

Deşifreleme (Decryption):
Bu durumda saldıran şifrelenmiş trafiğin bir kısmını deşifre etme yeteneğine sahiptir.

Şifreleme (Encryption(Forgery)):
Bu durumda saldıran anlamlı mesajları bilinmeyen gizli anahtar ile şifreleme olanağına sahiptir. Bu gizli anahtar bilgisine sahip olduğu anlamına gelmez. Bu atağa meyilli olan şifre sistemleri gerçekliğini kanıtlama/kimlik belirtme işlemlerinde kullanım için uygun değillerdir.

Kısmı Anahtar Edinimi (Partial Key Recovery):
Bu atakta gizli anahtarın belli bir kısmı saldıran tarafından ele geçirilir. Belki bu anahtarın geriye kalan kısmı çok büyük olabilir fakat bu arzulanan bir durum değildir. Çünkü tüm anahtarın genellikle ele geçirilmesi için ilk basamaktır.

Tüm Anahtar Edinimi (Total Key Recovery):
Bir kriptosistem için en korkunç kriptanalitik atak çeşidir. [18]

4.3.2. Kriptanaliz Metodları (Methods of Cryptanalysis)

Eğer şifre sistemi temiz ve basit bir yapıya sahip ise hala kalem ve kağıt kriptanalistin elindeki en güçlü silahlardır. Bir çok durumda kağıt analizi bilgisayar analizinden gelen geribildirimlere (özel istatistiksel özellikler ve düzensizlikler arama gibi) ihtiyaç duyar. Araştırmacı bakış açısından bir şifre sistemi kırıldığının dile getirilmesi, bu sistemin tasarımı değiştirmeye yol açacak bir zayıflığının bulunması demektir. Bu değişiklik, ek döngülerin eklenmesi veya döngü anahtarlarını oluşturan algoritmanın ve bazı iç yapıların değişmesi anlamına gelebilir. Şifre sisteminin tamamen kırılması ise bu tür sistemi tamir etmek yerine

baştan tasarlamanın daha anlamlı veya kolay olduğu durumlardır. Tipik kriptanaliz metotlarını şöyle sıralanabilir:

Etraflı Arama (Exhaustive Search):
Etraflı Arama, şifre sistemleri üzerine en açık ve en doğrudan uygulanabilir bir metottur. Tüm olası gizli anahtarları bilinen kısa açık/kapalı metin örnekleri üzerinde dener. Doğru gizli anahtar bilinen açık bir metinden doğru kapalı metnin elde edilmesini sağlar. Günümüz hesaplama imkanlarına göre modern blok şifre sistemlerinin anahtar uzunlukları (128-bit ve yukarısı) bu tip atağı imkansız kılacak şekilde seçilmektedir. DES in en önemli zayıflığı 56-bit olan kısa anahtar uzunluğudur ve günümüz koşulları düşünüldüğünde bu anahatar uzunluğu etraflı aramayı mümkün kılmaktadır.

Sözlük Atakları (Dictionary Attacks):
Bu da basit fakat blok şifre sistemleri için önemli bir atak çeşididir. Eğer şifrelenen metinlerin uzunlukları kısa ise saldıran birçok metin toplar ve farklı metinlerin tekrarlama analizini yapar. En uç noktada bu atağa zayıf şifre sistemi basit değiştirmeli şifre sistemidir (simple substitution cipher).

Eş Tanımlama (Equivalent Description):
Bazen şifre sistemi tasarlayanlar sistemin veya parçalarının basit eşdeğer tanımlarını gözden kaçırmaları, bu atak tarafından sömürülür.

Değişmezler için Devirlilik veya Arama (Periodicity or Search for Invariants):
Değişmezler, şifreleme boyunca değişmeyen özellikler olarak düşünülebilir ve şifre sisteminin istenilmeyen bir özelliğidir. Eğer kriptanalist sistemin herhangi değişmezini ya da yakınsamasını bulmayı başarırsa bir ayrılmış saldırı için malzeme edinmiş olur. Her çeşit devirli davranış veya şifrelenmeler arasındaki korelasyon mutlaka engellenmelidir.

Doğum Günü Paradoks (Birthday Paradox):

Blok şifreleme sistemlerinden açık anahtar şifre sistemlerine uzunan sayılamayacak kadar önemli bir olasılıklı paradoksudur.

Ortada buluşma Atağı (Meet-in-the-Middle Attack):

Bu metod şifreleme sistemini alt ve üst olmak üzere böler. Sonra kısmi tahmini ile yukarıdan ortaya ve baştan ortaya kısmi deşifreleme yapar. Sonuç karşılaştırılır ve eğer uyumlu ise aday anahtar saklanır. Aksi taktirde tahmin edilen anahtar yanlış olur.

İstatistiksel Yaklaşımlar (Statistical Approaches):

Bu metodlar kapalı ve açık metin arası ilişki veren istatistiksel örnekleri arar. Bir tür ayıran ataktır ve diğer ataklar için ilk adımdır. İstatistiksel yaklaşımlar hem oluşması yüksek olasılıkta olayları hem de gerçekleşmesi imkansız olayları bulmaya yöneliktir.

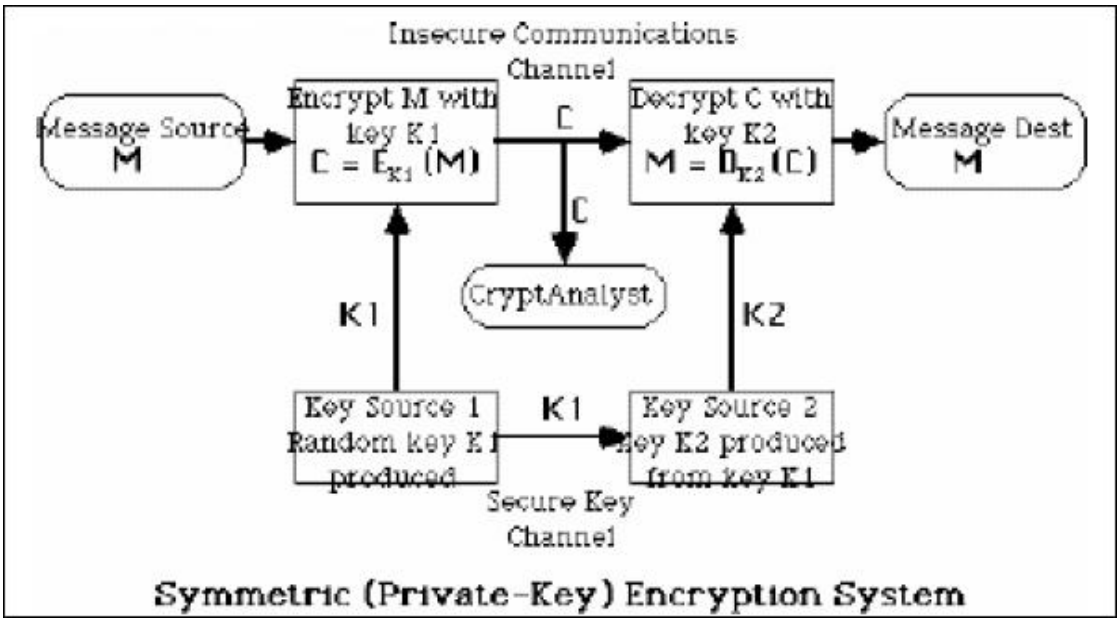
Özel bir Atağa göre Zayıf Anahtarlar (Weak Keys with Respect to a Particular Attack):

Bazı durumlarda bir zayıf anahtar kümesinin bir şifre sisteminin analizini özel bir atak modeli düşünüldüğünde kolaylaştırması mümkün olmaktadır. Eğer bir kriptosistemin tüm anahtar uzayına göre yüksek bir oranda zayıf anahtarlara sahip ise tekrar tasarlanması bile söz konusu olabilir.

Örnek olarak DES'te dört tane zayıf ve on iki tane yarı-zayıf anahtar bulunmaktadır. Gizli anahtar K olmak üzere DES'i E_K olarak tanımlarsak dört tane zayıf anahtar için $E_K(E_K(m)) = m$ ve on iki tane yarı-zayıf anahtardan iki tanesi için $E_{K1}(E_{K2}(m)) = m$ sağlanmaktadır. IDEA blok şifre sistemi için 2^{128} anahtar uzayı üzerinde 2^{63} elemana sahip bir zayıf anahtar kümesi bulunmaktadır. [18]

5. GİZLİ ANAHTARLI (SİMETRİK) KRİPTOSİSTEMLER

Gizli anahtarlı kriptografik sistemler tarihin ilk devirlerinden beri dünyada kullanımı süregelen kriptografik sistemlerdir. Bu sistemlerde şifreleme algoritması ve deşifreleme algoritması birbirinin tersi şeklindedir. Öncelikle haberleşecek iki grup aralarında gizli bir anahtar tespit ederler. Eğer bu iki grup birbirlerine yakın yerlerde yer almıyorlarsa güvenli bir haberleşme kanalı veya güvenilir bir kurye yoluyla anahtarları birbirlerine ulaştırabilirler. Bir taraf şifreleme algoritmasında girdi olarak açık metin (*P*) ve anahtarı (*K*) uygular, ardından şifreli metin (*C*)'yi elde eder ve mesajın alıcısına gönderir. Mesaj alıcısı ise deşifreleme algoritmasının girdileri olarak şifreli metin (*C*)'yi ve aynı (*K*) anahtarını kullanır ve ardından çıktı olarak açıkmetin (*P*)'yi elde eder (Şekil 5.1).

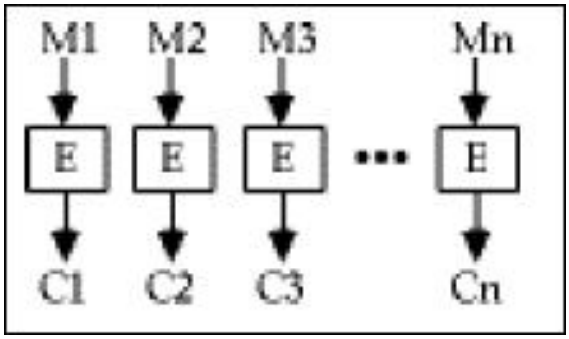


Şekil 5.1: Gizli-anahtarlı kriptosistem ile haberleşme

Gizli-anahtarlı kriptosistemleri uygulama sahalarında ikiye ayrılır;

i. Blok Şifreleme: Şifreleme ve deşifreleme işleminde metinler sabit uzunluklu dizilere bölünüp blok blok işleme tabi tutulur (örneğin 8, 16, 32 bit veya bayt). Anahtar uzunluğu ise yine sabittir. Blok şifrelemeye örnek olarak IBM tarafından

1976 yılında tasarlanan ve A.B.D. Teknoloji Standartları Enstitüsü NIST tarafından her dört yılda bir güvenliği onaylanan DES (Veri Şifreleme Standardı (Data Encryption Standard)) algoritması verilebilir. DES algoritması şifrelenecek metni 64 bitlik bloklar halinde şifreler, kullandığı anahtar boyu ise yine 64 bittir. Yalnız burada anahtarın işaret bitlerinin ayıklanmaları durumunda anahtar boyunun 56 bite indiğini hatırlatmak gerekir. Diğer bilinen blok şifrelemeli algoritmaları ise FEAL, IDEA ve RC5 örnek olarak gösterilebilir. Çalışılan çoğu modern şifreleyici bu formdadır (Şekil 5.2).



Şekil 5.2: Blok Şifreleyici

ii. Dizi Şifreleme: Bu çeşit şifrelemede algoritmanın girdisi yalnızca anahtardır. Algoritma anahtardan rastgele bir diziye çok benzeyen kayan anahtar dizisi üretir. Daha sonra kayan anahtar dizisinin elemanları ile açık metin veya kapalı metin dizisinin elemanları ikili tabanda toplanarak şifreleme veya deşifreleme işlemi tamamlanır. Dizi şifreleme algoritmalarına örnek olarak RC4 algoritması gösterilebilir.

- Mesajı bit bit işler (dizi olarak).

- En meşhur olanı Vernam şifreleyicisidir (aynı zamanda tek zamanlı blok (one-time pad) denir).

- 1917'de AT&T de çalışan Vernam tarafından geliştirilmiştir.

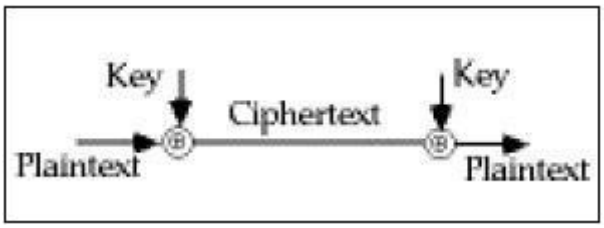
- Basit olarak mesaj bitlerini rastgele anahtar bitlerine ekler (Şekil 5.3.1).

- Mesaj biti kadar anahtar biti gerekir. Prtatikte zordur (Örnek, Pratikte mag teyp veya CDROM da dağıtılır).

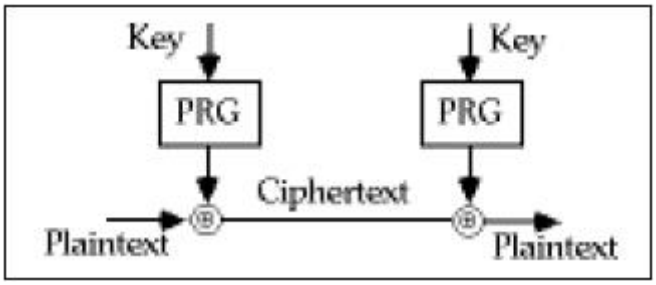
- Anahtar tamamen rasgele olduğu için koşulsuz güvenlik sağlanır.

- Böyle büyük bir anahtar dağıtımı güç olduğu için anahtar dizisi daha küçük (taban) bir anahtardan üretilebilir. Bunun için rasgele sembol fonksiyonları kullanılır (Şekil 5.3.2).

- Her ne kadar bu çok çekici gözükse de pratikte iyi bir kriptografik güçlü rasgele fonksiyon bulmak çok güçtür. Bu hala birçok araştırmacının konusudur. [6]



Şekil 5.3.1



Şekil 5.3.2

5.1. Simetrik Şifreleme Algoritmaları

Geleneksel simetrik blok şifreleme algoritmaları (örn. DES) 1973'de IBM'de çalışan Hort Feistel Tarafından geliştirilen Feistel networküne dayanır. Şekil 5.4'te gösterilen bu algoritmada 2w bit uzunluğunda olan şifresiz metin iki eşit sol ve sağ parçaya ayrılır. Her bir turda ana şifreden üretilen alt şifre ile sağ tarafa F fonksiyonu uygulanır. Bunun sonucu ise sol taraf ile EXOR mantıksal işlemine tabi tutulur. Daha sonrada elde edilen sonuçlar çaprazlanır. Yani sağ taraf sola sol taraf sağa geçer. Böylece turlar devam eder. Asıl anahtardan alt anahtarlar her turda üretilerek F fonksiyonuna girdi olarak kullanılır.
Feistel algoritmasının önemli parametreleri aşağıda açıklanmıştır.

Blok uzunluğu: Büyük blok uzunluğu daha fazla güvenlik anlamındadır; fakat, şifreleme/deşifreleme hızını azaltır. Genel olarak 64 bitlik blok genişliği kullanılır.

Anahtar Uzunluğu: Büyük anahtar genişliği daha fazla güvenlik anlamındadır; fakat, şifreleme/deşifreleme hızını azaltır. Çok kullanılan anahtar uzunluğu 128 bittir.

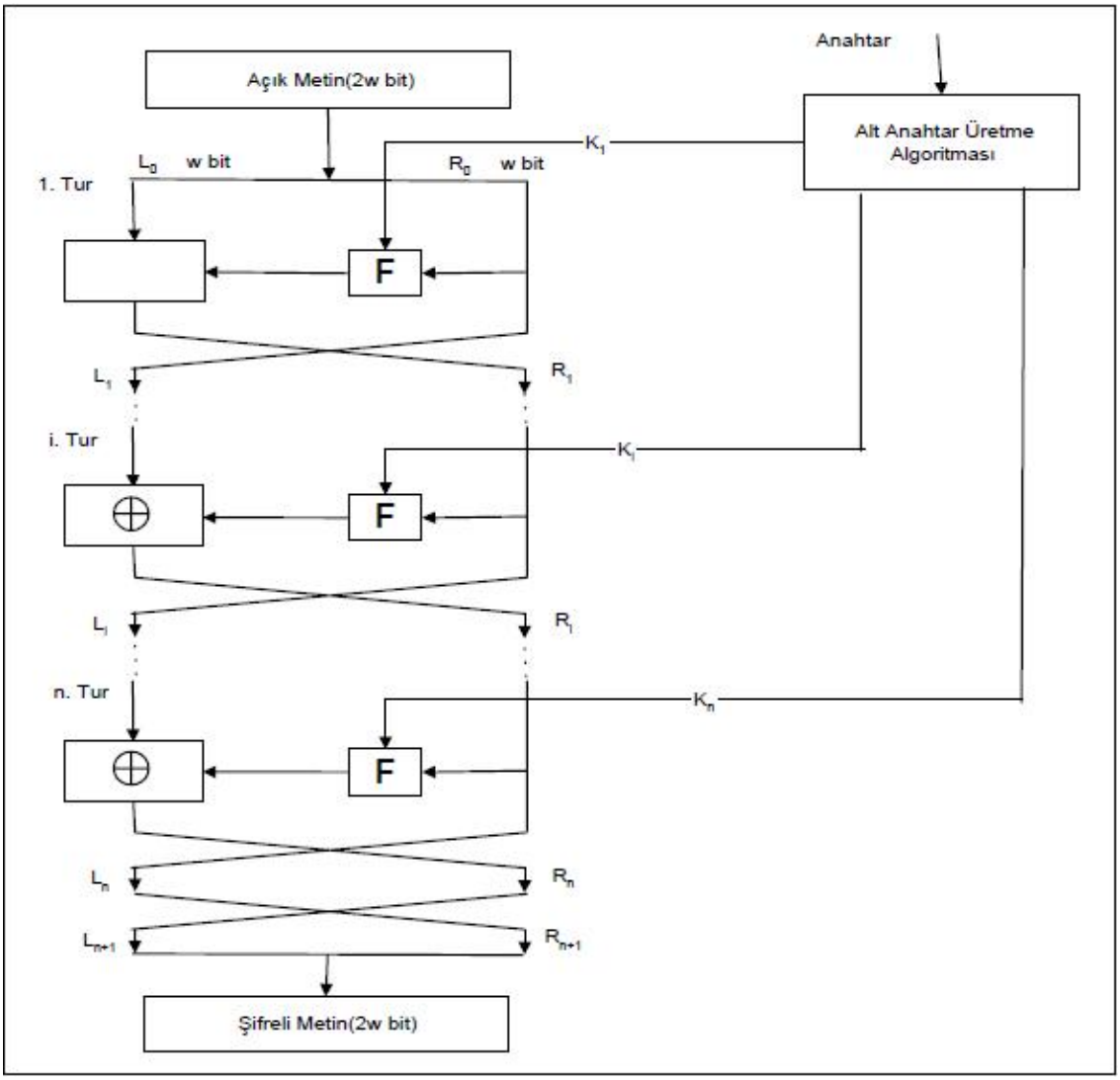
Tur Sayısı: Fazla tur sayısı şifreleme güvenliğini artırır. Genel olarak 16 Tur kullanılır.

Alt Anahtar Üretme Algoritması: Karmaşıklığı fazla olan bir alt anahtar üretimi kirptoanalizi zorlaştırır.

Tur Fonksiyonu: Fazla karmaşık olan tur fonksiyonu kriptoanalizi zorlaştırır. Feistel şifreleyici için diğer özellikler.

Hızlı yazılım şifreleme/deşifreleme: Çoğu uygulamada şifreleme uygulamaları veya donanım gerçeklemesi şeklinde kullanım fonksiyonlarının içine koyulur. Dolayısı ile algoritmanın icra hızı düşünülmesi gerekir.

Analiz Kolaylığı: Her ne kadar algoritmanın olası kriptoanaliz saldırılarına karşı olabildiğince karmaşık olması istenirse, bu özellik algoritmanın anlaşılabilirliğini de azaltır. Örneğin DES kolay analiz edilen bir algoritma değildir. Feistel şifreleyicinin deşifreleme algoritması da aynıdır. Şifreli metin giriş olarak kullanılırken alt anahtar tersinden kullanılır. Yani önce K_n, en son olarak da K_1 kullanılır. Bu özellik nedeniyle Şifreleme ve deşifrelemede farklı algoritma kullanılması gerekmez. [6]



Şekil 5.4: Klasik Feistel Network

5.1.1. Veri Şifreleme Standardı (Data Encryption Standart (DES))

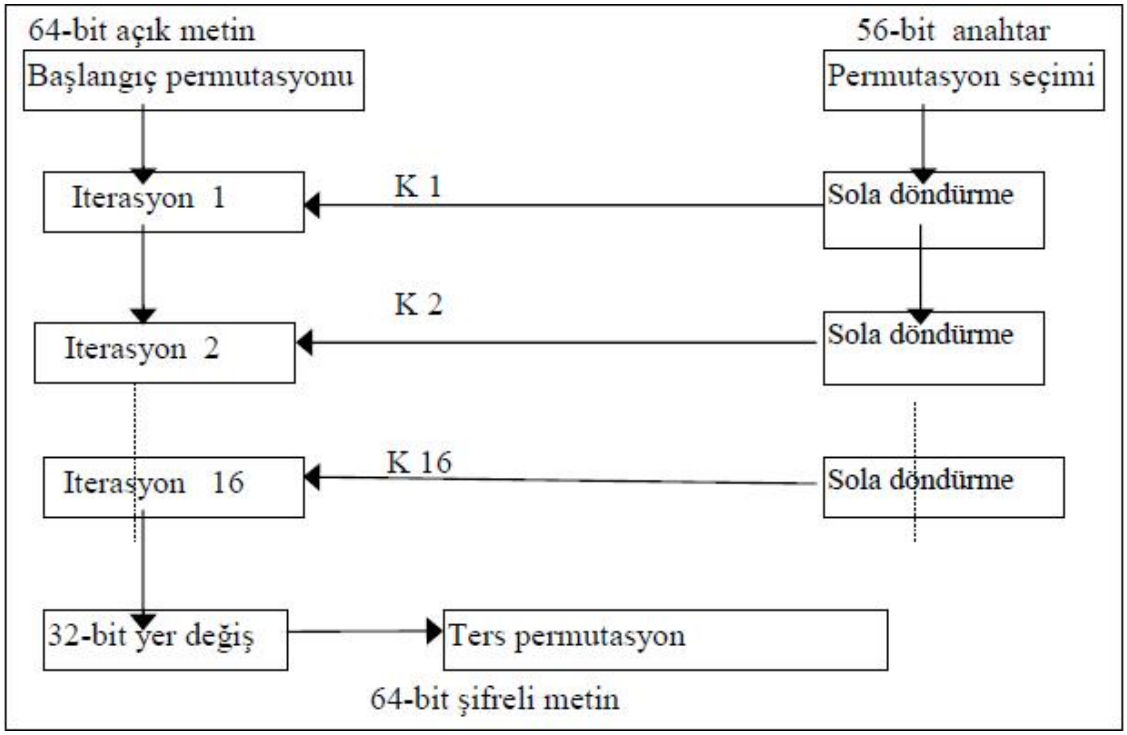
DES 1974 yılında IBM tarafından geliştirilmiş ve 1977 yılında yasal olarak atanmıştır. Basit blok şema Şekil 5.5'te gösterilmiştir. Temeli Feistel networküne dayanır.

DES bir blok şifrelemedir, 64 bit bloklardaki veriyi şifreler. Düzmetnin 64 bitlik bloğu bir algoritmaya sokulur ve 64 bitlik şifrelenmiş bir ifade elde edilir. Şifrelemede ve şifreyi çözerken her ikisinde de aynı algoritma ve anahtarlar (key) kullanılır.

Anahtar uzunluğu 56 bittir (Anahtar genellikle 64 bit olarak ifade edilir, fakat her sekizinci bit parity biti olarak kullanılır ve ihmal edilir). Anahtar herhangi bir 56 bit sayı olabilir ve her zaman değiştirilebilir.

Algoritmanın Özeti:

DES 64 bit blok düzmetinde işlem görür. Düzmetin, ilk permutasyondan sonra yarısı sağda yarısı solda her biri 32 bit uzunluğunda iki parçaya bölünür. Daha sonra *f* fonksiyonu ve anahtar ile birleştirilerek sonraki adıma geçilir. Aynı işlem 16 kez tekrarlanır ve 16. turun sonunda, sağ ve sol parçalar birleştirilir. Son permutasyondan sonra (başlangıçtaki permutasyonun tersi) algoritma tamamlanarak biter.



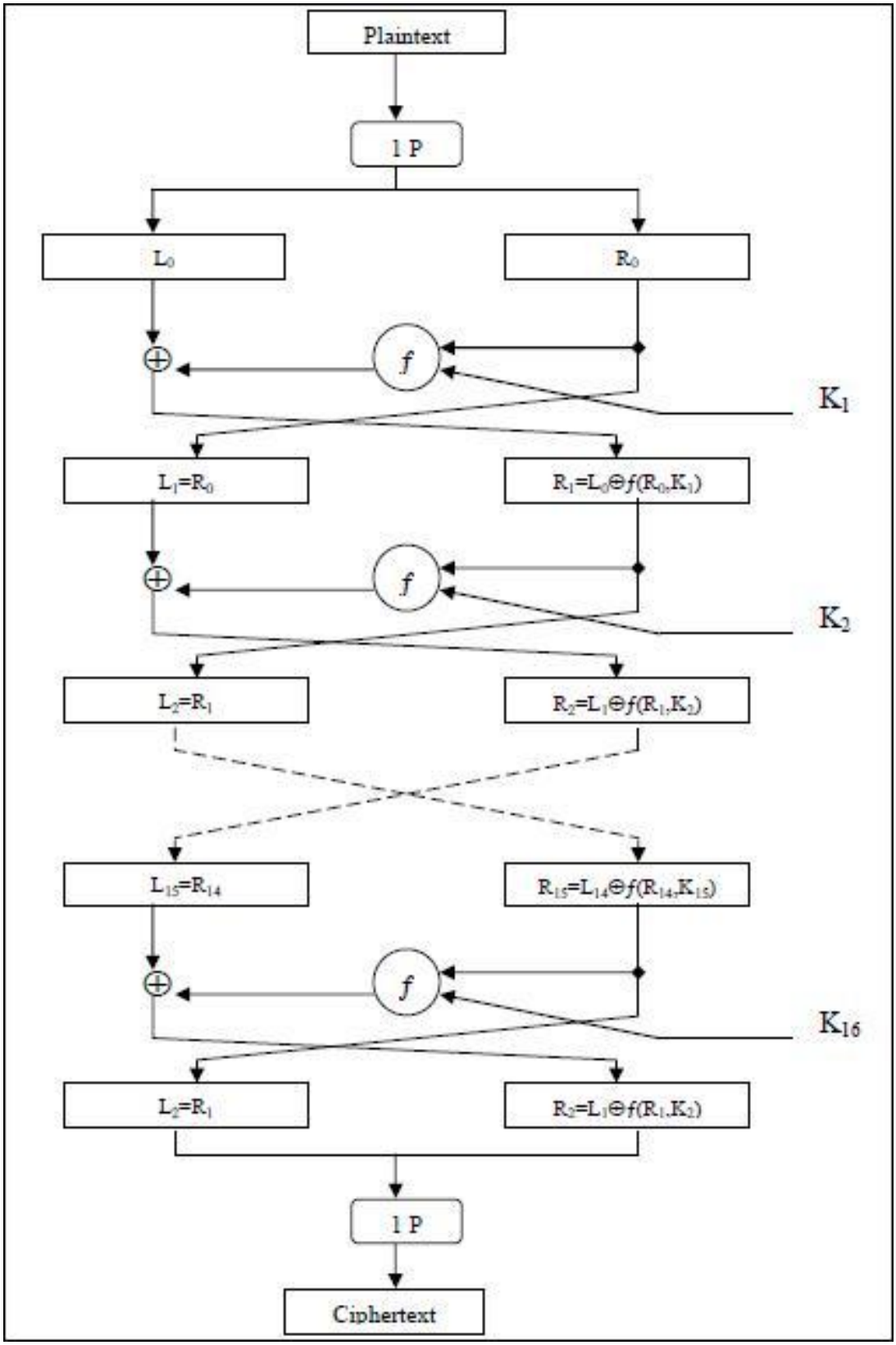
Şekil 5.5: DES Algoritmasının Genel Yapısı

Her bir tur da anahtar bitleri değiştirilir ve anahtarın 56 bitinden 48 biti seçilir. Verinin sağ yarımı genişleme permutasyonu (expansion permutation) yoluyla 32 bitten 48 bite genişletilir.

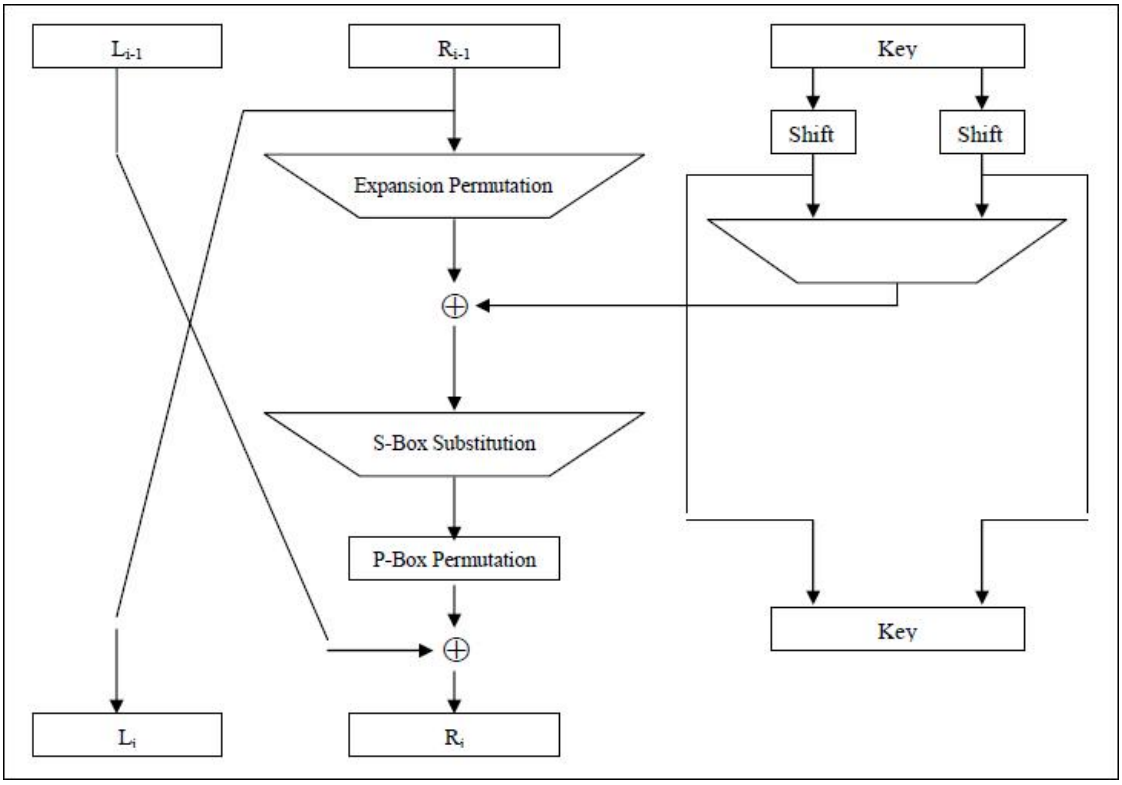
Genişletilen kısım seçilen 48 bit anahtarla XOR işlemine sokulur. Daha sonra 32 yeni bit üreten 8 S-Box içerisine gönderilir ve tekrar değiştirilir. Bu dört işlem *f* fonksiyonunu oluşturur. *f* fonksiyonunun çıktısı verinin sol yarımı ile XOR işlemine tabi tutulur. Sonuçta elde edilen değer yeni sağ yarım olmakta ve sol yarım ise sağ yarımın eski hali olmaktadır. (5.1)'de gösterilen bu işlem 16 kez tekrar eder.

$$L_i = R_{i-1},\ L_i = R_{i-1} \oplus f\left(R_{i-1}, K_i\right) \tag{5.1}$$

Şekil 5.6 ile DES Algoritmasının genel yapısı görülebilir. Şekil 5.7 ile DES Algoritmasının işleyişindeki bir tur görülebilir.



Şekil 5.6: DES Algoritması



Şekil 5.7: DES'in bir turu

Başlangıç Permutasyonu:

Başlangıç permutasyonu tur 1' den önce meydana gelir. Şifrelemeden önce 64 bitlik düzmetin 32 bitlik iki parçaya bölünür. Tüm çift bitler sol tarafta ve tek pozisyondaki bitler de sağ tarafta yer alır. Tablo 5.1'de tanımlandığı gibi giriş bloklarının yerleri değiştirilir. Tabloda görüldüğü gibi örneğin; başlangıç değişiminde düzmetnin 1. pozisyonundaki bite 58 nolu bit taşınmış, 2. pozisyonuna 50 nolu bit atanmış vb. …

Başlangıç permutasyonu ve benzer şekilde sonuç permutasyonu DES' in güvenliğine etki etmez.

Tablo 5.1: Başlangıç Permutasyonu

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

Anahtar Dönüşümü:

Başlangıçta, 64 bitlik DES anahtarı her sekiz bit ihmal edildiği için 56 bite düşürülür. Bu Tablo 5.2'de tanımlanmıştır. İhmal edilen bu bitler anahtarı kontrol etmek için eşitlik kontrolünde kullanılır. 56 bitlik anahtar elde edildikten sonra DES' in 16 turunun her biri için farklı 48 bit altanahtar üretilir. Bu alt-anahtarlar (K_i) şu şekilde belirlenir.

Tablo 5.2: Anahtar Permutasyonu

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

İlk olarak 56 bitlik anahtar 28 bitlik iki parçaya bölünür. Turun ihtiyacına göre parçaların bir veya iki biti değiştirilir. Değiştirilecek bit sayıları Tablo 5.3'te belirtilmiştir.

Tablo 5.3: Turların her biri için değiştirilen anahtar bitlerinin sayısı

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
0	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

Değiştirmeden sonra, 56 bitten 48 biti seçilir. Bu işlemde bitlerin altkümesi seçildiği için, bitlerin düzeni değişir. Bu işlem sıkıştırma permutasyonu (compression permutation) olarak adlandırılır. Tablo 5.4'te sıkıştırma permutasyon tanımlanmıştır.

Tablo 5.4: Sıkıştırma Permutasyonu

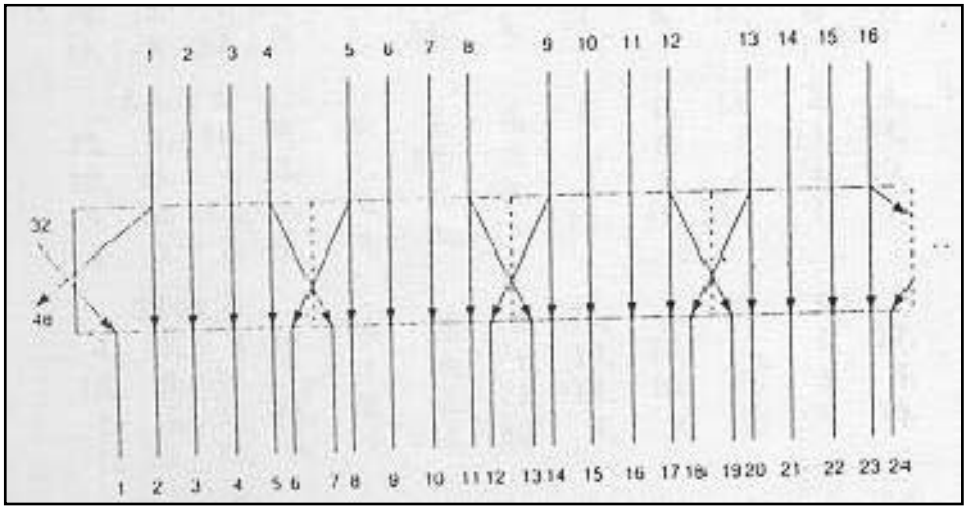
14 17 11 24 1 5	3 28 15 6 21 10
23 19 12 4 26 8	16 7 27 20 13 2
41 52 31 37 47 55	30 40 51 45 33 48
44 49 39 56 34 53	46 42 50 36 29 32

Genişleme permutasyonu:

Bu işlemde verinin sağ yarısı (R_i) 32 bitten 48 bite genişletilir; çünkü, bu işlem tekrar eden belirli bitleri en uygun şekilde değiştirir. Bu işlem iki amaç için yapılır. XOR işlemi için sağ yarımı anahtar ile aynı uzunlukta yapmak ve yerine koyma (substitution) işlemi sırasında sıkıştırılabilen daha uzun sonuç sağlamak.

Şekil 5.8'de genişleme permutasyonu tanımlanmıştır. Her 4 bit giriş bloğu için, birinci ve dördüncü bitlerin her biri çıkış bloğundan iki biti gösterir, ikinci ve üçüncü bitler ise çıkış bloğundan birer bit gösterir.

Tablo 5.5'te çıktı pozisyonlarının hangi girdi pozisyonlarına göre nasıl yerleştirildiği görülmektedir. Örneğin; girdi bloğunun 3. pozisyonu çıktı bloğunun 4. pozisyonuna karşılık gelmektedir ve girdi bloğunun 21. pozisyonu çıktı bloğunun 32. pozisyonuna karşılık gelmektedir.



Şekil 5.8: Genişleme Permutasyonu

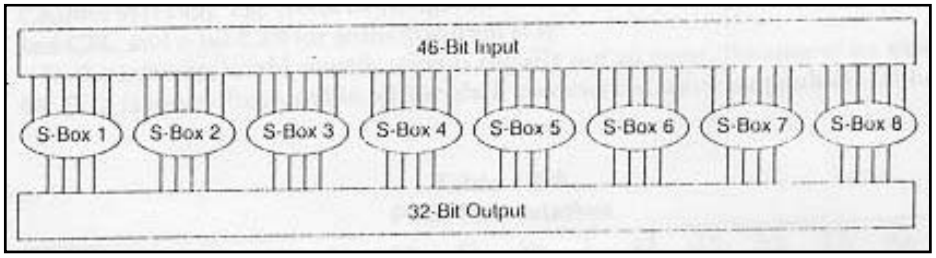
Tablo 5.5: Genişleme Permutasyonu

1	2	3	4				5	6	7	8				9	10	11	12				13	14	15	16				17	18	19	20	
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		

21	22	23	24				25	26	27	28			
20	21	22	23	24	25	24	25	26	27	28	29	28	29	30	31	32	1		

S-Box Yerine Koyma:

Sıkıştırılmış anahtar genişletilmiş blok ile XOR edildikten sonra, 48 bit yerine koyma işlemine taşınır. Yerine koymalar, sekiz tane yerdeğiştirme kutusu (substitution boxes) veya S-Box'lar tarafından icra edilir. Her bir S-Box'ta 6 bit giriş ve 4 bit çıkış vardır ve sekiz farklı S-Box mevcuttur. 48 bit sekiz tane 6 bitlik alt bloklara bölünür. Her bir ayrılan blok, ayrılmış S-Box tarafından işletilir. Birinci blok S-Box 1, ikinci blok S-Box 2 tarafından işleme sokulur.



Şekil 5.9: S-Box Yerine Koyma

Her bir S-Box 4 satır ve 16 sütundan oluşan bir tablodur. Kutulardaki her bir girdi 4 bitlik sayıdır. Tablo 5.6'da sekiz S-Box'ın tümü gösterilmiştir.

P-Box Permutasyonu:

S-Box yerine koyma işleminden sonra elde edilen 32 bitlik çıktı P-Box'ta uygun bir şekilde değiştirilir. Bu değişiklikte girdi pozisyonuna göre çıktı pozisyonu tasarlanır. Hiçbir bit iki kez kullanılmaz ve hiçbir bit ihmal edilmez. Bu işlem straight permutation olarak çağrılır. Tablo 5.7'de her bir bitin taşındığı pozisyon gösterilmektedir. Örneğin, 21. bit 4. bite taşınmış ve 4. bit 31. bite taşınmıştır.

En sonunda başlangıçtaki 64 bitlik verinin sol yarımı ile P-Box permutasyonu sonucunda elde edilen 32 bitlik veri XOR işlemine sokulmaktadır. Sol ve sağ yarımlar değiştirilerek bir sonraki tur başlamaktadır.

Tablo 5.6: S-Box'lar

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
S1 0:	E	4	D	1	2	F	B	8	3	A	6	C	5	9	0	7
1:	0	F	7	4	E	2	D	1	A	6	C	B	9	5	3	8
2:	4	1	E	8	D	6	2	B	F	C	9	7	3	A	5	0
3:	F	C	8	2	4	9	1	7	5	B	3	E	A	0	6	D
S2 0:	F	1	8	E	6	B	3	4	9	7	2	D	C	0	5	A
1:	3	D	4	7	F	2	8	E	C	0	1	A	6	9	B	5
2:	0	E	7	B	A	4	D	1	5	8	C	6	9	3	2	F
3:	D	8	A	1	3	F	4	2	B	6	7	C	0	5	E	9
S3 0:	A	0	9	E	6	3	F	5	1	D	C	7	B	4	2	8
1:	D	7	0	9	3	4	6	A	2	8	5	E	C	B	F	1
2:	D	6	4	9	8	F	3	0	B	1	2	C	5	A	E	7
3:	1	A	D	0	6	9	8	7	4	F	E	3	B	5	2	C
S4 0:	7	D	E	3	0	6	9	A	1	2	8	5	B	C	4	F
1:	D	8	B	5	6	F	0	3	4	7	2	C	1	A	E	9
2:	A	6	9	0	C	B	7	D	F	1	3	E	5	2	8	4
3:	3	F	0	6	A	1	D	8	9	4	5	B	C	7	2	E
S5 0:	2	C	4	1	7	A	B	6	8	5	3	F	D	0	E	9
1:	E	B	2	C	4	7	D	1	5	0	F	A	3	9	8	6
2:	4	2	1	B	A	D	7	8	F	9	C	5	6	3	0	E
3:	B	8	C	7	1	E	2	D	6	F	0	9	A	4	5	3
S6 0:	C	1	A	F	9	2	6	8	0	D	3	4	E	7	5	B
1:	A	F	4	2	7	C	9	5	6	1	D	E	0	B	3	8
2:	9	E	F	5	2	8	C	3	7	0	4	A	1	D	B	6
3:	4	3	2	C	9	5	F	A	B	E	1	7	6	0	8	D
S7 0:	4	B	2	E	F	0	8	D	3	C	9	7	5	A	6	1
1:	D	0	B	7	4	9	1	A	E	3	5	C	2	F	8	6
2:	1	4	B	D	C	3	7	E	A	F	6	8	0	5	9	2
3:	6	B	D	8	1	4	A	7	9	5	0	F	E	2	3	C
S8 0:	D	2	8	4	6	F	B	1	A	9	3	E	5	0	C	7
1:	1	F	D	8	A	3	7	4	C	5	6	B	0	E	9	2
2:	7	B	4	1	9	C	E	2	0	6	A	D	F	3	5	8
3:	2	1	E	7	4	A	8	D	F	C	9	0	3	5	6	B

Tablo 5.7: P-Box Permutasyonu

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

Sonuç Permutasyonu:

Sonuç permutasyonu başlangıç permutasyonunun tersi şekilde çalışır ve Tablo 5.8'de tanımlanmıştır. DES'in son turundan sonra elde edilen sağ ve sol yarımlar birleştirilerek ($R_{16}L_{16}$) sonuç permutasyonuna girdi olur. Bu algoritma şifrelemede ve şifreyi çözmede her ikisinde de kullanılır. [6]

Tablo 5.8: Sonuç Permutasyonu

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

5.1.2. Uluslararası Veri Şifreleme Algoritması (International Data Encryption Algorithm (IDEA))

Simetrik blok şifreleme algoritması olan IDEA 1991'de Swiss Federal Institute of Technology'de geliştirilmiştir. 128 Bit anahtar uzunluğu kullanılır. IDEA alt anahtar üretim ve tur fonksiyonları bakımından DES'ten farklıdır. S-boxes kullanılmaz. XOR, 16 bit tam sayı toplama ve 16 bit tamsayı çarpma matematik işlemlerini kullanır. Kriptoanalizi zor olan bir algoritmadır. Alt anahtar üretim algoritması sadece dairesel kaydırma üzerinedir, fakat her bir sekiz turda altı alt anahtar üreten karmaşık bir yapıya sahiptir. İlk 128 bit anahtar kullanan algoritma olduğu için kriptoanalistlerin üzerinde çok çalıştıkları bir algoritmadır. [6]

5.1.3. BlowFish

Blowfish, bağımsız kriptocu olan Bruce Schneier tarafından 1993'te geliştirilmiş ve kısa zamanda DES'e en popüler alternatif haline gelmiştir. Kolay programlanabilen ve hızlı çalışan bir algoritmadır.

Aynı zamanda 5K'den az bellekte çalışan çok karmaşık bir algoritmadır. Anahtar uzunluğu değişkendir ve 448 bit kadar olabilir. Pratikte 128 bit anahtar kullanılır ve 16 tur kullanır.

Blowfish DES gibi S-Box ve XOR fonksiyonu kullanır; fakat, aynı zamanda ikili toplama da kullanır. Sabit S-Box'lar kullanan DES'in tersine, Blowfish anahtarın bir fonksiyonu olarak üretilen dinamik S-Box'lar kullanır. Blowfish'de alt anahtar ve S-Box'lar, Blowfish algoritmasının anahtar üzerinde tekrarlanarak uygulanmasıyla elde edilir. Alt anahtar ve S-Box'ların üretilmesi için Blowfish şifreleme algoritmasının toplam 512 kere yürütülmesi gerekir. Dolayısı ile çok sık gizli anahtar değişimi gerektiren uygulamalarda Blowfish kullanılması uygun değildir. [6]

5.1.4. RC5

RC5, 1994'te RSA asimetrik şifreleme algoritmasını geliştirenlerden biri olan Ron Rivest tarafından geliştirilmiş ve RC5 aşağıdaki özelliklere sahiptir:

Donanım veya yazılım ile gerçeklenmeye uygundur: Mikro işlemcilerde bulunan primitif hesaplama operatörlerine sahiptir.

Hızlılık: Basit ve kelime yönelimlidir. Temel işlemler bir anda verinin bütün kelimesi üzerinde yapılır.

Değişik kelime uzunluklu işlemcilere adapte edilebilirlik: Bir kelimdeki bit sayısı RC5'te parametredir. Farklı kelime uzunluklu farklı algoritmalar oluşturur.

Değişken sayıda tur: Değişken tur sayısı RC5'in diğer parametresidir. Bu parametre daha fazla hız ile daha fazla güvenlik arasında değişim yapar.

Değişken anahtar uzunluğu: Anahtar uzunluğu RC5'in üçüncü parametresidir. Bu parametre dedaha fazla hız ile daha fazla güvenlik arasında değişim yapar.

Basitlik: RC5 kolay programlama için basit bir yapıya sahiptir.

Düşük bellek gereksinimi: Düşük bellek gereksinimi RC5'i smart kartlar ve sınırlı belleğe sahip diğer benzer cihazlarda kullanımını sağlar.

Yüksek güvenlik: RC5 uygun parametreler ile yüksek güvenlik sağlar.

Veri bağımlı döndürmeler: Verinin miktarına bağlı olarak döndürme gerçekleştirir. Bu algoritmanın kriptoanalistlere karşı gücünü artırır. [6]

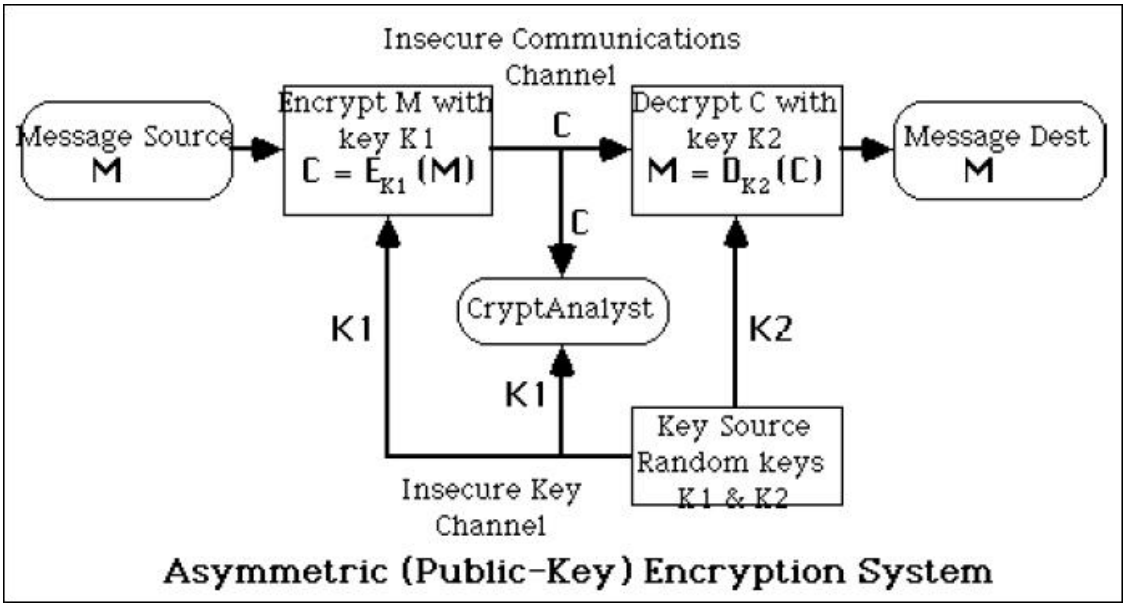
5.1.5. CAST-128

CAST 1997'de Entrust Teknolojiler'den Carlise Adams ve Stafford Tavares Tarafından geliştirilen bir tasarım prosedürüdür. Bir özel algoritma 8 bit artımlar ile 40 bitten 128 bit'e kadar değişen anahtar uzunlukları kullanır. CAST, DES'te kullanılanlardan daha uzun olan sabit S-Box'lar kullanır. Bu S-Box'ların tasarımı kriptoanaliste karşı önemlidir. CAST'taki alt anahtar üretimi diğer blok şifreleyicilerden farklıdır. Doğrusal olmayan S-Box'lar kullanılarak alt anahtar üretimi yapılır. CAST-128'in diğer enteresan özelliği tur'dan tur'a değişen F tur fonksiyonudur. [6]

6. AÇIK ANAHTARLI (ASİMETRİK) KRİPTOSİSTEMLER

Gizli-anahtarlı kriptosistemlerinin aksine Açık-anahtarlı kripto sistemlerinin kullanımı henüz çok yenidir. Açık-anahtarlı kriptosistemleri üzerine ilk öneri 1976 yılında Diffie ve Hellman tarafından yapılmıştır. Ardından 1977 yılında Rivest, Shamir ve Adleman RSA Kriptosistemi adlı yeni bir Açık-anahtarlı kriptosistemini bulmuşlardır. 1978 yılından beri kripto dünyasına değişik teklifler yapıla gelmiştir. Bunlardan daha önemlileri El-Gamal tarafından tasarlanan El-Gamal Açık-anahtarlı kripto algoritması ve eliptik eğri Açık-anahtarlı kripto sistemleridir. Temelde Açık-anahtarlı kriptosistemlerinin gayesi belli bir anahtar üzerinde anlaşmanın ve karşı tarafa bu anahtarı güvenli olarak ulaştırabilmenin zorluğunu ortadan kaldırmaktır. Burada tek yönlü bir mesajlaşma söz konusudur. Mesaj alıcısı sadece kendisinin bileceği "Gizli-anahtar" ve diğer kişilere dağıtabileceği bir "Açık-anahtar"dan oluşan anahtar çifti belirler. Kullanılan anahtar üretim algoritmasına göre bu iki anahtar arasında matematiksel bir bağlantı mutlaka olabilecektir; fakat, asıl amaç bilinen açık anahtardan gizli anahtarın hesaplanmasının polinomsal (çok terimli) zamanda imkansız olabilmesidir.

Şekil 6.1 ile Açık anahtarlı kriptosisteminin işleyişi görülebilir.



Şekil 6.1: Açık anahtarlı kriptosistemi

Mesaj göndericisi alıcıya ait herkesçe bilinen açık anahtarı kullanarak Açık-anahtarlı kripto algoritmasıyla göndereceği mesajı kapatır ve alıcıya gönderir, mesajın alıcısı ise yalnız kendisinin bildiği gizli anahtar ile deşifreleme algoritmasını kullanarak mesajı açabilir. Gizli anahtar yalnız alıcı tarafından bilindiği için başka birinin bu mesajı açması mümkün olamayacaktır. Gizli anahtarın açık anahtardan polinomsal zamanda türetilmesini imkansız kılmak için Diffie ve Hellman'ın "tek-yönlü fonksiyon" mantığı üzerine kurulu Anahtar değişim protokolü (Key Exchange Method) vardır.

Özellikler

- Geleneksel Gizli anahtarlı kriptografi gönderici ve alıcının birlikte paylaştığı tek bir anahtar kullanır.

- Eğer bu anahtar açıklanırsa haberleşme tehlikeye düşer

- Açık anahtarlı (veya çift anahtarlı, asimetrik) kriptografi iki anahtar kullanmayı gerektirir:

- Mesajları şifrelemekte ve imzaları doğrulamakta kullanılan herkes tarafından bilinebilen bir açık anahtar
- Mesajları deşifrelemekte ve imza oluşturmakta kullanılan sadece alıcı tarafından bilinen bir gizli anahtar

- Açık anahtar özel (gizli) anahtardan ve şifreleme hakkındaki diğer bilgilerinden kolaylıkla hesaplanır (Bu bir polinomsal zaman problemidir (P-time)).

- Bununla beraber açık anahtarın ve şifrelemenin bilinmesi, gizli anahtarı hesaplamak için hala hesaplama bakımından verimsizdir (NP-time problem).

- Böylece açık anahtar, kendisi ile güvenli haberleşmek isteyen herhangi birisine dağıtılabilir.

- Açık anahtarlı algoritmaların üç önemli sınıfı vardır.
  - Açık Anahtar Dağıtım Şeması (Public-Key Distribution Schemes (PKDS)) burada şema bilginin bir kısmının güvenli olarak değiştirilmesi için kullanılır (değer iki tarafa bağlıdır).
  - Bu değer gizli anahtar şeması için bir oturum anahtarı olarak kullanılır.
  - İmza Şeması (Signature Schemes) Sadece sayısal imza üretmek için kullanılır. Burada gizli anahtar imzayı üretmekte, açık anahtar ise doğrulamakta kullanılır
  - Açık Anahtar Şeması (Public Key Schemes (PKS)), şifrelemek için kullanılır. Burada açık anahtar mesajları şifreler, gizli anahtar mesajları deşifreler.
  - Herhangi bir açık anahtar şeması, gerekli olan oturum anahtarlı mesajı seçmek suretiyle PKDS olarak kullanılabilir.
  - Çoğu açık anahtar şeması aynı zamanda imza şemasıdır (sağlanan şifreleme ve deşifreleme her iki sırada yapılabilir). [6]

6.1. Diffie-Hellman Açık anahtar Dağıtım Şeması

- İlk açık anahtar tipi şema PKDS olup 1976 da Diffie & Hellman tarafından yayınlanmıştır.

- Bu zamanda mükemmel bir kriptografi üst bakışıdır.

- Açık anahtar dağıtım şemasıdır.
  - Herhangi bir keyfi mesajı değiştirmek için kullanılmaz
  - Değeri üyelere bağlı olan bir anahtardır (ve onların açık ve gizli anahtar bilgisi)

- Sonlu bir alanda (Galois) ya bir asal sayının tamsayı modulu veya bir polinomsal alan üzerinde üstelleştirilmesine dayanır.
  - nb üstelleştirme $O((\log n)^3)$ işlem mertebesindedir.

- Güvenliği bu alanlardaki logaritmik hesaplamanın güçlüğüne dayanır
  - nb ayrık logaritma $O(e^{\log n \log \log n})$ işlem mertebesindedir.

- Diffie-Hellman PKDS aşağıdaki şekilde çalışır.
  - Güvensiz bir iletişim kanalı üzerinden bazı anahtarları değiştirmek isteyen iki kişi A ve B olsun, Bunlar;
  - Büyük bir asal sayı seçerler. p (~200 dijit) ve
  - α bir mod p pirimitif elemandır
  - A'nın x_A gibi bir gizli sayısı vardır ($x_A < p$).
  - B'nin x_B gibi bir gizli sayısı vardır ($x_B < p$).
  - A ve B açıklayacakları y_A ve y_B'yi sırasıyla hesaplarlar
    - $y_A = \alpha^{x_A} \bmod p$, $y_B = \alpha^{x_B} \bmod p$
  - Sonra anahtar aşağıdaki şekilde hesaplanır
    - $K_{AB} = \alpha^{x_A x_B} \bmod p$ (ortak Gizli Anahtar)

        - $= y_A^{x_B} \bmod p$ (B hesaplayabilir)
        - $= y_B^{x_A} \bmod p$ (A hesaplayabilir)
    - A ve B arasında güvenli haberleşme için bir gizli anahtarlı şifreleyicide kullanılabilir.

- nb: Eğer iki kişi sonradan haberleşmek isterse kendi açık anahtarlarını değiştirmedikçe aynı gizli anahtara sahip olacaklardır. [6]

6.2. RSA Açık anahtarlı Kriptosistem

- En çok bilinen ve en pratik açık anahtarlı tasarım olarak kabul edilen algoritma 1977'de Rivest, Shamir & Adleman tarafından önerildi.

- Mesajları şifrelemek, Anahtar değiştirmek ve sayısal imza oluşturmak için kullanılan bir açık anahtarlı tasarımdır.

- Tamsayı modulo bir sonlu alan (Galois) içinde tamsayı modulo üzerinde üsselleştirmeye dayanır.
    - nb üstelleştirme işlemleri $O((\log n)^3)$ mertebesindedir.

- Güvenliği, büyük sayıların çarpanlarının hesaplanmasının zorluğuna bağlıdır.
    - nb faktörizasyon işlemleri $O(e^{\log n \log \log n})$ mertebesindedir.
    - (Ayrık logaritma ile benzerdir.)

- Algoritma Kuzey Amerika'ya patentlidir (Bu nedenle dünyanın başka bir yerinde patentlenemez).
    - Bu yöntemin uygulanmasında yasal zorlukların kaynağıdır.

- RSA, modüler aritmetiği kullanarak üstelleştirmeye dayanan bir açık anahtarlı şifreleme algoritmasıdır.

- Yöntemin uygulanması için önce anahtarların üretilmesi gerekir.

- Her bir kullanıcı tarafından anahtar üretimi aşağıdakileri içerir:
  - Rasgele çok büyük iki asal sayı seçilir (~100 digit), p, q
  - n = p.q hesaplanır
  - $\varphi(n) = (p-1).(q-1)$ hesaplanır.
  - Rasgele bir şifreleme anahtarı seçilir; öyle ki: OBEB($\varphi(n)$,e) =1; e<$\varphi(n)$.
  - Deşifreleme anahtarı d hesaplanır: $d = e^{-1} \bmod \varphi(n)$, $0 \leq d \leq n$.
  - Açık Anahtar: KA={e,n}
  - Gizli Anahtar: KG={d,n}

- Şifreli metin C'yi elde etmek için M mesajının şifrelenmesi: $C=M^e \bmod n$, $0 \leq d \leq n$.

- M Mesajını elde etmek için C şifreli metnin deşifre edilmesi: $M = C^d \bmod n$ dir.

- RSA sistemi aşağıdaki sonuca dayanır:
  Eğer n = pq burada p, q farklı büyük asal sayılardır buradan
  X $\varphi(n) = 1 \bmod n$
  Bütün x'ler p veya q tarafından bölünemezler
  ve $\varphi(n) = (p-1)(q-1)$ 'dir. [6]

6.2.1. RSA Örneği

- p=7 ve q= 17 olan iki asal sayı seçilir.

- n = p.q = 119 değeri hesaplanır.

- $\varphi(n) = (p-1)(q-1) = 96$ hesaplanır.

- Bir e sayısı seçilir, öyle ki φ(n) =96 ve ebob(φ(n) ,e) =1; e < φ(n), buradan e=5 seçilir,

- Öyle bir d sayısı belirlenir ki, de = 1 mod 96 ve d < 96 d için doğru değer d=77 dir.

- Çünkü 77 x 5 = 385 = 4 x 96 + 1

- Sonuçta anahtarlar; açık anahtar KA = {5,119}; gizli anahtar; KG = {77,119} olacaktır.

- Şifreleme için ise;
  Açık metin olarak M= 19 seçilsin;
  $C=M^e \bmod n$, 195 ≡ 66mod 119 elde edilir. Şifreli metin 66'dır.

- Deşifreleme için;
  $M = C^d$ Mod n; 6677 ≡ 19mod 119 elde edilir ki açık metinin 19 olduğu sonucuna ulaşılır. [6]

6.2.2. RSA'nın Güvenliği

- RSA algoritmasının güvenliği, n'nin modülünün çarpanlarına ayrılmasının zorluğuna dayanır.

- En iyi bilinen çarpanlarına ayırma algoritması olan (Brent-Pollard) n sayısı üzerindeki en büyük çarpan p ise $O\left(\frac{e^{\sqrt{2\ln p \ln\ln p}}}{\ln p}\right)$ işlem mertebesindedir (Tablo 6.1).

- Bu 200 dijit uzunluğunda olan n için 1-100 MIPS'lik modern bilgisayar için sayı 106'ya bölünerek saniye cinsinden zaman hesaplanır.
    - nb: halen 1e+14 işlem hesaplama için elverişlilik limiti olarak kabul edilir ve 3e+13 usec/yıl alır.

- Fakat çoğu bilgisayarlar 32/64 bitten daha büyük sayılar üzerinde doğrudan işlem yapamazlar.

- Bu nedenle büyük sayılar üzerinde işlem yapmak için kütüphaneleri kullanılır. [6]

Tablo 6.1

n'deki onlu dijit sayısı	n'nin çarpanlarına ayrılmasındaki işlem(bit) sayısı
20	7200
40	3.11e+06
60	4.63e+08
80	3.72e+10
100	1.97e+12
120	7.69e+13
140	2.35e+15
160	5.92e+16
180	1.26e+18
200	2.36e+19

6.2.3. Multi-Precision Arithmetic

- Çoklu kelime(multiple precision) sayılar üzerinde çalışan fonksiyon kütüphanelerini kapsar.

- Klasik referanslar "yarı nümerik algoritmalar" olarak bilinir
    - Dijit dijit çarpma yapılır
    - Kare alma ve çarpma ile üs alma işlemi yapılır

- Bilinen kütüphaneler kullanılıp tekerlek yeniden keşfedilmeye uğraşılmamalıdır.

- Modülo aritmetiği özellikle modülo indirgemeler ile özel hünerler kullanabilir. [6]

6.2.4. Daha Hızlı modülo İndirgeme

Chivers (1984), multi-precision aritmetik işlemleri yaparken modülo indirgemeleri yapmanın hızlı bir yolunu göstermiştir.

Bir b tabanlı n karakterli tamsayı A(a_0, …, a_{n-1}) verilsin, tamsayı A (6.1)'deki gibi gösterilebilir:

$$A = \sum_{i=0}^{n-1} a_i b^i \tag{6.1}$$

Buradan

$$A \equiv \left\{ \sum_{i=0}^{n-2} a_i b^i + a_{n-1} b^{n-1} \left(\mathrm{mod}\ jm\right) \right\} \left(\mathrm{mod}\, m\right) \tag{6.2}$$

elde edilebilir.

(6.2)'deki ifade, bir sayının En yüksek Anlamlı Dijitinin çıkartılabileceğini ve kalan dijitlere eklenebilen mod m kalanının asıl sayıya mod m uyumlu olan bir sayıda sonuçlanacağını gösterir.

Bir sayıyı indirgemek için kullanılan Chivers algoritması aşağıdaki gibidir:

1. R = (bd, 2.bd, … , (b-1).bd)(mod m) şeklinde Bir dizi düzenle
2. FOR i = n-1 to d do
WHILE A[i] != 0 do
j = A[i];
A[i] = 0;
A = A + bi-d.R[j];
END WHILE
END FOR

Burada; A[i] A sayısını i'inci karakteridir, R[j] R dizisinden j. tamsayı kalandır. A'daki sembol sayısı n, Modüldeki sembol sayısı d'dir. [6]

6.2.5. RSA 'in Hızlandırılması – Değişik Çarpma Teknikleri

- Geleneksel çarpma $O(n^2)$ mertebesinde bit işlemi yapar, daha hızlı teknikler aşağıdakileri içerir:

- Schonhage-Strassen Tamsayı Çarpma Algoritması:
    - Herbir tamsayı bloklara bölünür ve bir polinomun katsayıları olarak kullanılır.
    - Bu polinomların uygun noktalarda değeri hesaplanır, sonuç değerler çarpılır
    - Çarpım polinomun katsayılarını oluşturmak için bu değerlerin enterpolasyonu alınır.
    - Orijinal tamsayının çarpımını bulmak için katsayılar birleştirilir
    - Enterpolasyon fazını hızlandırmak için Ayrık Fourier dönüşümü ve Katlama (konvolüsyon) dönüşümü kullanılır.
    - O(n log n) bit işleminde çarpma yapılablilir.

- Özel donanım kullanılabilir Çünkü:
    - Elde propagasyon gecikmesi nedeniyle geleneksel aritmetik birimler büyütülemez.
    - Böylece O(n) bit işlemde çarpma yapmak için O(n) kapılı ya paralel elde saklama veya gecikmeli elde-saklama teknikleri kullanılır.
    - Veya, O(log n) bit işlemde çarpma yapmak için O(n2) kapılı, paralel-paralel teknikler kullanılır. [6]

6.2.6. RSA ve Çin Kalan Teoremi (Chinese Remainder Theorem)

- RSA için deşifreleme hızında anlamlı bir iyileştirme, sırasıyla modulo p ve modulo q çalıştırmak için Çin kalan teoremini kullanarak yapılır.
    - P ve q yarı büyüklükte olduğu için, n = p.q nin büyüklüğü yarıdır ve böylece aritmetik çok daha hızlıdır.

- Deşifreleme hesabından, iki denklem üretmek suretiyle RSA'de Çin kalan teoremi kullanılır.

$$M = Cd \bmod R$$

Aşağıdaki gibidir:

$$M1 = M \bmod p = (C \bmod p)d \bmod (p-1)$$
$$M2 = M \bmod q = (C \bmod q)d \bmod (q-1)$$

Buradan denklem çiftinin

$$M = M1 \bmod p = M2 \bmod q$$

dur.

Çin Kalan Teoremi ile aşağıda verilen tek bir çözümü vardır:

$$M = [((M2 + q - M1)u \bmod q] p + M1$$

dir.

Burada (p.u) mod q = 1 dir. [6]

6.2.7. Pratikte RSA Gerçeklenmesi

- Yazılım ile gerçeklemeler

- Genellikle 256-512 bit blok uzunluğunda 1-10 bits/saniye de icra edilir.
- Gerçeklemenin iki ana şekli:
  - Mikrobilgisayarlarda, hibrid bir algoritmada anahtar değiştirme mekanizmasının parçası olarak.
  - Daha büyük makinalarda güvenli bir posta sisteminin elemanları olarak.

- Donanım Gerçeklemeleri
  - Genellikle 256-512 bit blok uzunluğunda 100-10000 bits/saniye de icra edilir.
  - Bütün bilinen gerçeklemeler büyük bit uzunluklu geleneksel Aritmetik Mantık Birimdir. [6]

6.3. El Gamal

- Diffie-Hellman anahtar dağıtım şeması mesajların güvenli değiştirilmesini sağlar.

- 1985 de ElGamal tarafından geliştirilmiştir.

- Diffie-Hellman yöntemindeki gibi güvenliği faktör işlemli logaritmaların zorluğuna dayanır.

- **Anahtar Üretimi**
  - Büyük bir asal sayı seçerler. p (~200 dijit).
  - α bir mod p pirimitif elemandır.
  - A'nın x_A gibi bir gizli sayısı vardır ($x_A < p$).
  - B'nin x_B gibi bir gizli sayısı vardır ($x_B < p$).
  - A ve B açık layacakları y_A ve y_B'yi sırasıyla hesaplarlar.
    - $y_A = \alpha^{x_A} \mod p$ $y_B = \alpha^{x_B} \mod p$.

- M mesajını C şifreli metne kriptolamak için,
  - Rasgele bir **k** sayısı seçilir, 0 <= k <= p-1.
  - **K** mesaj anahtarı aşağıdaki şekilde hesaplanır.
    - $K = y_B^k \bmod p$
  - Şifreli metin çifti: C = {c1,c2} aşağıdaki gibi hesaplanır.
    - $C_1 = [\alpha]^k \bmod p$, $C_2 = K.M \bmod p$

- Mesajı deşifrelemek için
  - **K** mesaj anahtarı çıkartılır
    - $K = C_1^{x_B} \bmod p = [\alpha]^{kx_B} \bmod p$
  - M için aşağıdaki denklem çözülerek **M** elde edilir:
    - $C_2 = K.M \bmod p$

[6]

7. ELİPTİK EĞRİ

Kriptografide kriptografik sistemleri oluşturmak için eliptik eğri kullanımı yaklaşık 20 sene önce Neal Koblitz[5] ve Victor S. Miller[6] tarafından bağımsız olarak önerilmiştir.

7.1. Giriş

Eliptik bir eğri, çözümleri topolojik olarak bir *torus*a denk olan uzayın bir bölgesine sıkıştırılan bir kübik eğri çeşididir.

Tanım 7.1: Bir K cismi üzerinde eliptik bir eğri E, $a_1, a_2, a_3, a_4, a_6 \in K$ ve $\Delta \neq 0$ olmak üzere

$$E : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6 \tag{7.1}$$

denklemi ile tanımlanır. Burada Δ, E'nin diskriminantıdır ve (7.2) ifadesiyle elde edilir.

$$\left.\begin{aligned} \Delta &= -d_2^2 d_8 - 8d_4^3 - 27d_6^2 + 9d_2d_4d_6 \\ d_2 &= a_1^2 + 4a_2 \\ d_4 &= 2a_4 + a_1a_3 \\ d_6 &= a_3^2 + 4a_6 \\ d_8 &= a_1^2a_6 + 4a_2a_6 - a_1a_3a_4 + a_2a_3^2 - a_4^2 \end{aligned}\right\} \tag{7.2}$$

Eğer L, K cisminin bir genişlemesi ise, E üzerinde L-rasyonel nokta kümesi ∞ sonsuzda bir nokta olmak üzere

$$\mathrm{E(L)} = \left\{(\mathrm{x}, \mathrm{y}) \in L \times L : y^2 + a_1xy + a_3y - x^3 - a_2x^2 - a_4x - a_6 = 0\right\} \cup \{\infty\}$$

ile gösterilir. [3]

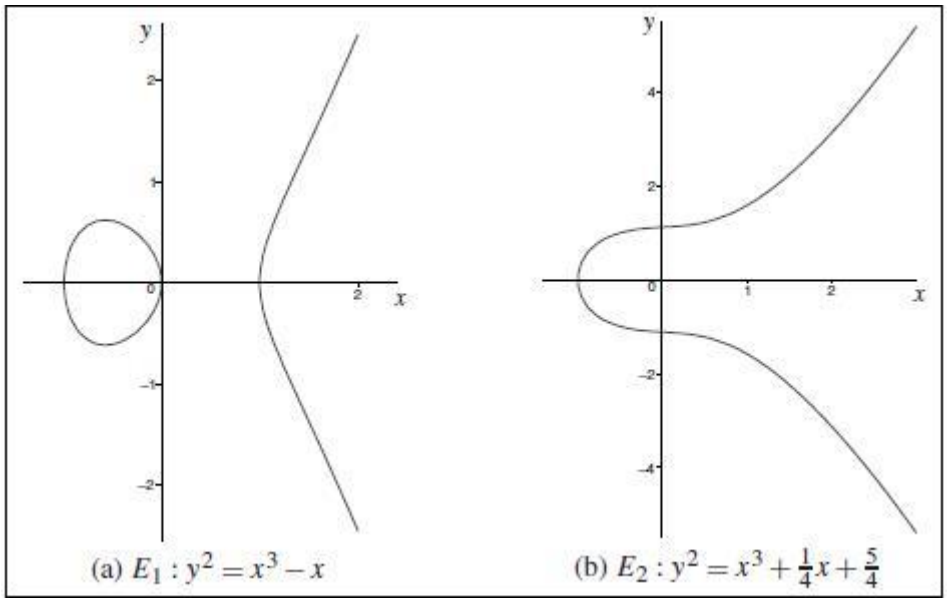
(7.1) ifadesi Weierstrass denklemi olarak adlandırılır.

---

[5] N. Koblitz, *Elliptic curve cryptosystems*, in *Mathematics of Computation* 48, 1987, pp. 203–209

[6] V. Miller, Use of elliptic curves in cryptography, CRYPTO 85, 1985

Şekil 7.1 ile $y^2 = x^3 - x$ ve $y^2 = x^3 + \frac{1}{4}x + \frac{5}{4}$ eliptik eğrileri gösterilebilir. [3]



Şekil 7.1: *R* üzerinde eliptik eğriler.

Tanım 7.2: *K*, karakteristiği 2 ve 3'ten farklı olan bir cisim ve $a, b \in K$ olmak üzere $x^3 + ax + b$ çoklu (katlı) kökü olmayan kübik bir çokterimli (polinom) olsun. *K* üzerinden eliptik bir eğri, "sonsuzda bir nokta" olarak adlandırılan ve *O* ile gösterilen tek bir eleman ile beraber

$$y^2 = x^3 + ax + b \tag{7.3}$$

denklemini sağlayan ve $x, y \in K$ olan (x, y) noktalar kümesidir. [19]

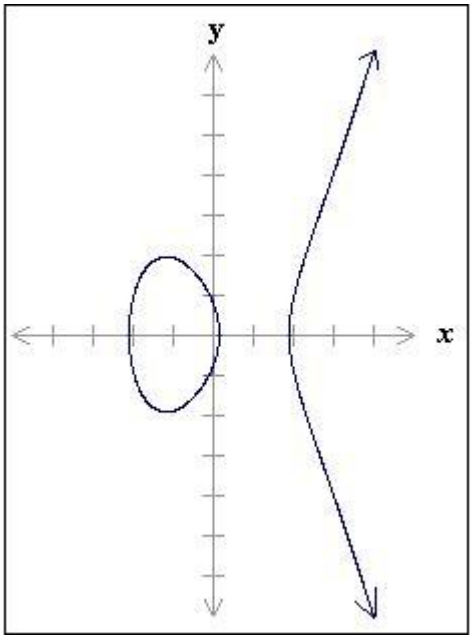
7.2. Reel Sayılar Üzerinde Eliptik Eğri Grupları

Reel sayılar üzerinde eliptik bir eğri, *x*, *y*, *a* ve *b* reel sayılar olmak üzere

$$y^2 = x^3 + ax + b$$

şeklindeki bir eliptik eğri denklemini sağlayan (x, y) noktalar kümesi olarak tanımlanabilir.

a ve b sayılarının değişik seçimleri farklı eliptik eğriler yaratır. $a = -4$ ve $b = 0.67$ için eliptik eğri denklemi $y^2 = x^3 - 4x + 0.67$ olur ve Şekil 7.2'deki şekil elde edilir.



Şekil 7.2: $y^2 = x^3 - 4x + 0.67$ denklemine ait eliptik eğri

Eğer $x^3 + ax + b$ tekrarlı çarpan içermiyorsa ya da başka bir deyişle eğer $4a^3 + 27b^2 \neq 0$ ise, $y^2 = x^3 + ax + b$ eliptik eğrisi bir grup oluşturarak kullanılabilir. Reel sayılar üzerindeki bir eliptik eğri grubu, "sonsuzdaki nokta" olarak adlandırılan bir özel nokta O ile beraber eliptik eğriyle ilgili noktaları içerir. [7]

7.2.1. Eliptik Eğri Toplamı: Geometrik Yaklaşım

Eliptik eğri grupları toplamsal gruplardır; yani, basit fonksiyonların toplamadır. Eliptik eğrilerde iki noktanın toplamı geometrik olarak tanımlanır.

Bir $P = (x_P, y_P)$ noktasının negatifi, o noktanın x-eksenine göre yansımasıdır: $-P$ noktası $(x_P, -y_P)$'dir. Eliptik eğri üzerindeki her P noktası için $-P$ de eliptik eğri üzerindedir. [8]

Tanım 7.3 Toplama Kuralı: $y^2 = x^3 + ax + b$ bir E eğrisi ve $P_1 = (x_1, y_1)$ ve $P_2 = (x_2, y_2)$ iki nokta olsun.

$$x_3 = m^2 - x_1 - x_2$$
$$y_3 = m(x_1 - x_3) - y_1$$

ve

$$m = \begin{cases} \dfrac{(y_2 - y_1)}{(x_2 - x_1)} & eğer\ P_1 \neq P_2 \\ \dfrac{(3x_1^2 + a)}{(2y_1)} & eğer\ P_1 = P_2 \end{cases}$$

olmak üzere

$$P_1 + P_2 = P_3 = (x_3, y_3)$$

bulunur. Eğer m eğimi sonsuz ise, $P_3 = \infty$ olur. Bütün P noktaları için $\infty + P = P$ kabul edilir.

Ayrıca toplama kuralı, birleşme

$$(P + Q) + R = P + (Q + R)$$

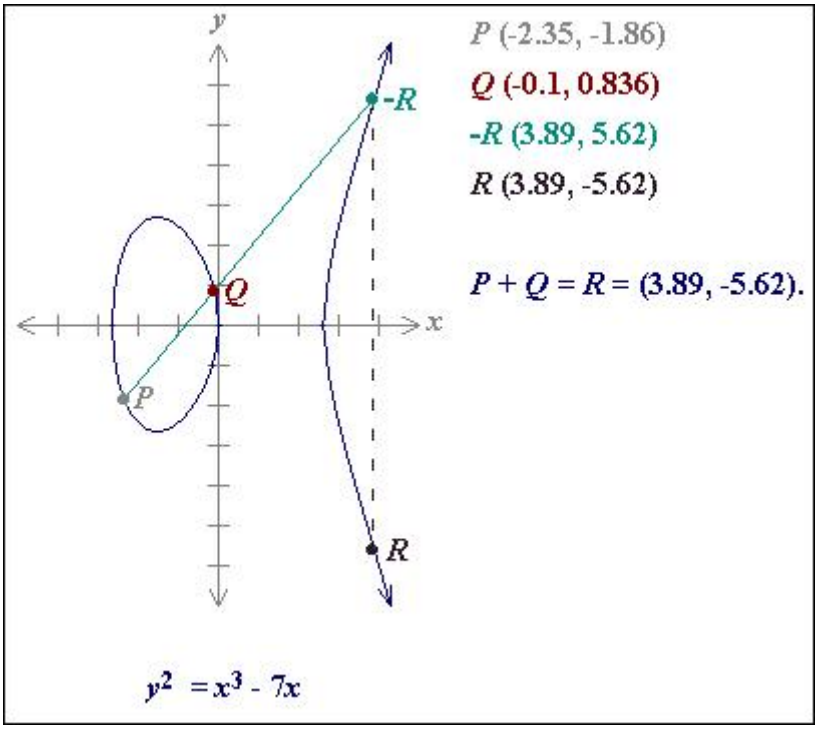
ve değişme

$$P + Q = Q + P$$

özelliklerine sahiptir. [21]

7.2.1.1. *P* ve *Q* Ayrı Noktaları Toplama

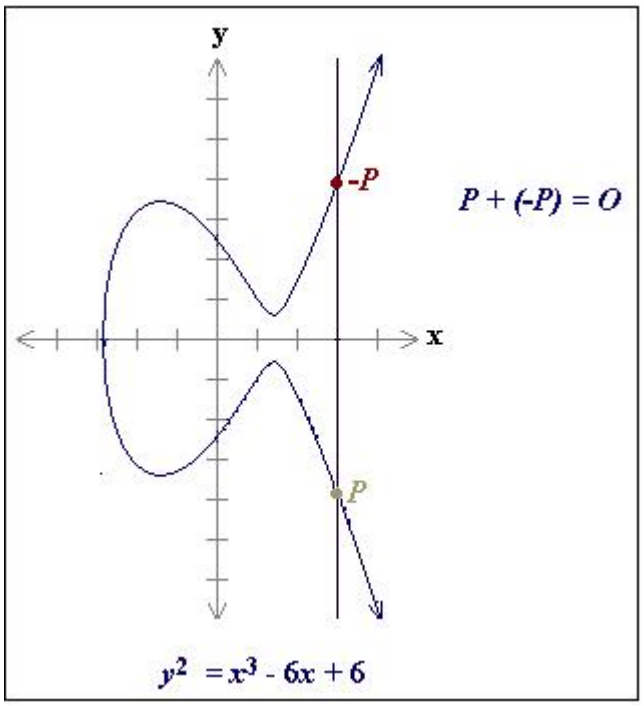
P ve Q eliptik eğri üzerinde iki nokta ve $P \neq -Q$ olsun. P ve Q noktalarını eklemek için iki nokta boyunca bir doğru çizilir. Bu doğru eğriyi $-R$ olarak adlandırılacak olan bir noktada daha keser. $-R$ noktası, x-eksenine göre R noktasına yansıtılır. Eliptik eğri grubunda toplama kuralı, $P + Q = R$'dir. Bu durum Şekil 7.3'teki $y^2 = x^3 - 7x$ eğrisi üzerinde gösterilebilir. [8]



Şekil 7.3

7.2.1.2. *P* ve – *P* Noktalarını Toplama

P'den $-P$'ye olan doğru, Şekil 7.4'teki gibi eliptik eğriyi üçüncü bir noktada kesmeyen dikey bir doğrudur; bundan dolayı, *P* ve $-P$ normal olarak eklenemez. Bu yüzden eliptik eğri grubu sonsuzda bir *O* noktası içerir. Tanıma göre, $P+(-P)=O$ olur. Bu eşitliğe göre de $P+O=P$ yazılabilir. Burada *O*, eliptik eğri grubunun toplama işlemine göre birim elamanı olarak adlandırılır. Bütün eliptik eğriler bir birim elamana sahiptir. [9]



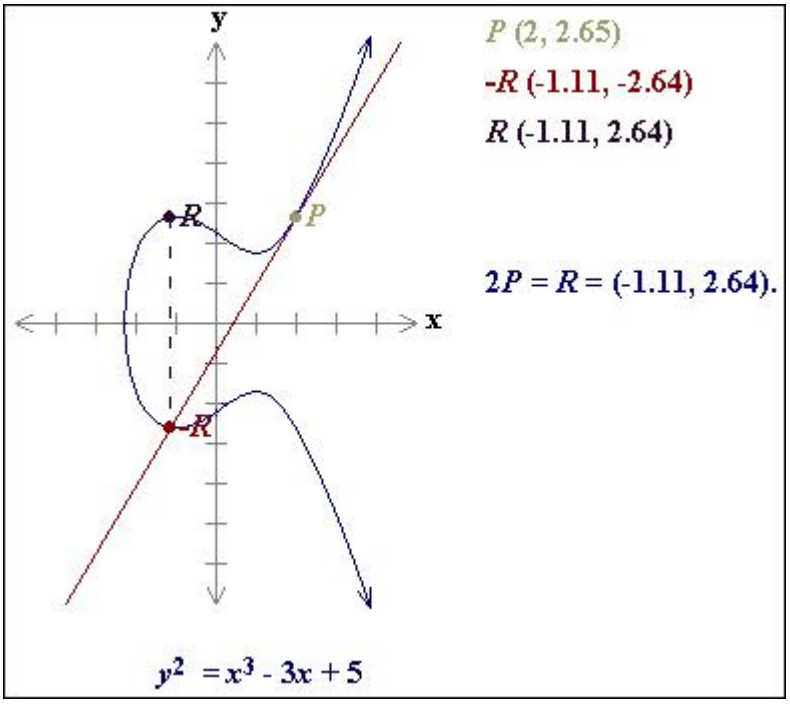
Şekil 7.4

7.2.1.3. *P* Noktasını İkiye Katlama

$P=(x_P, y_P)$ noktasını kendisine eklemek için, P noktasında eğriye teğet bir doğru çizilir. Eğer $y_P \neq 0$ ise, teğet doğru eğriyi bir $-R$ noktasında keser. Şekil 7.5'te görüldüğü gibi, $-R$ x-eksenine göre yansıtılarak R bulunur. Bu işlem, P noktasını ikiye katlama olarak adlandırılır ve

$$P + P = 2P = R$$

ile ifade edilir. [10]



Şekil 7.5

7.2.1.4. $P=(x_P, y_P)$'yi $y_P = 0$ iken İkiye Katlamak

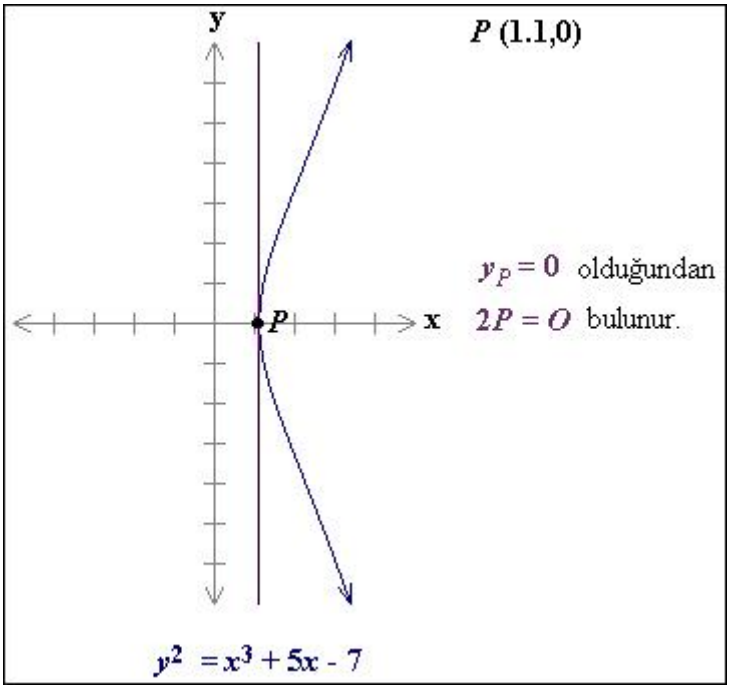
Eğer bir $P=(x_P, y_P)$ noktasında $y_P = 0$ ise, eliptik eğriye P noktasındaki teğet doğru Şekil 7.6'daki gibi dikeydir ve eğriyi herhangi bir noktada kesmez. Tanıma göre böyle bir P noktası için $2P = O$ olur.

Bu durumda $3P$ bulunmak istenirse

$$3P = 2P + P = O + P = P$$

bulunur.

Devam edilirse, $4P = 0$, $5P = P$, … gibi sonuçlar elde edilir. [11]



Şekil 7.6

7.3. F_p Üzerinde Eliptik Eğri Grupları

Reel sayılar üzerinde hesaplama yavaştır ve yuvarlama hatası yüzünden yanlışlıklar doğar. Kriptografik uygulamalar hızlı ve hassas aritmetiğe ihtiyaç duyar; böylece pratikte F_p ve F_{2^m} sonlu cisimleri üzerinde eliptik eğri grupları kullanılır.

F_p cismi 0'dan $p-1$'e kadar olan sayıları kullanır ve hesaplamalar p'ye bölümden kalanları alarak sonlanır. Örneğin F_{23} cismi 0 ile 22 sayıları arasındaki tamsayılardan oluşur ve bu cisimdeki herhangi işlem 0 ile 22 sayıları arasındaki tamsayılarla sonuçlanır.

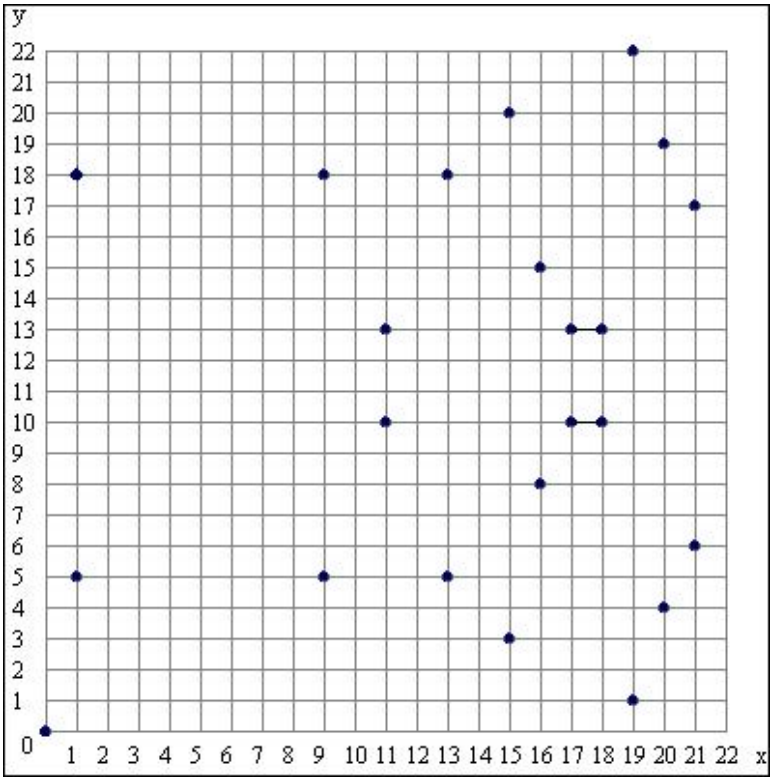
Temel alınan F_p cismi ile bir eliptik eğri, a ve b değişkenleri F_p cismine ait olacak şekilde seçilerek oluşturulur. Eliptik eğri, modülo p'ye göre eliptik eğri denklemini sağlayan bütün (x, y) noktalarını içerir (x ve y, F_p'ye ait sayılardır).

Eğer $x^3 + ax + b$ tekrarlı kök içermiyorsa (yani, $\left(4a^3 + 27b^2\right) \neq 0 \left(\text{mod } p\right)$ ise), eliptik eğri bir grup oluşturabilir. F_p üzerindeki eliptik bir eğri, sonsuzdaki nokta olarak adlandırılan bir O noktası ile beraber ilgili eliptik eğrideki noktaları içerir. Böyle bir eliptik eğride sonlu sayıda nokta vardır. [12]

Örnek 7.1 ile bunu inceleyelim.

Örnek 7.1: F_{23} cismi üzerinde eliptik bir eğri göz önüne alınsın. $a = 1$ ve $b = 0$ için eliptik eğri denklemi $y^2 = x^3 + x$ olur. $y^2 = x^3 + x$ eliptik eğri denklemini sağlayan noktalar modülo 23'e göre sırayla (0,0), (1,5), (1,18), (9,5), (9,18), (11,10), (11,13), (13,5), (13,18), (15,3), (15,20), (16,8), (16,15), (17,10), (17,13), (18,10), (18,13), (19,1), (19,22), (20,4), (20,19), (21,6) ve (21,17) bulunur.

F_{23} üzerinde $y^2 = x^3 + x$ eğrisinin grafiksel gösterimi Şekil 7.7'deki gibidir.



Şekil 7.7: F_{23} üzerinde $y^2 = x^3 + x$

Her x değeri için iki nokta vardır. $y = 11.5$ için grafik simetriktir. Reel sayılardaki eliptik eğrilerde her noktanın negatifi x-eksenine göre yansıtılarak alınıyordu. Örnek 7.1'de y değerlerindeki negatif bileşenler modülo 23'e göre bulunur. Örneğin, $P = (x_P, y_P)$ noktasının negatifi $-P = \left(x_P, \left((-y_P) \pmod{23}\right)\right)$'dir. [13]

7.3.1. F_p Üzerinde Eliptik Eğri Gruplarında Aritmetik

7.3.1.1. Farklı P_1 ve P_2 Noktalarını Toplama

$y^2 = \left(x^3 + ax + b\right) \pmod{p}$ eğrisi göz önüne alınsın. $P = (x_P, y_P)$ noktasının negatifi $-P = \left(x_P, \left((-y_P) \pmod{p}\right)\right)$'dir. Eğer $P_1 = (x_1, y_1)$ ve $P_2 = (x_2, y_2)$ noktaları $P_1 \neq -P_2$ olacak şekilde birbirlerinden farklılarsa

$$x_3 = \left(m^2 - x_1 - x_2\right) \pmod{p}$$
$$y_3 = \left(m(x_1 - x_3) - y_1\right) \pmod{p}$$

ve

$$m = \left(\frac{(y_2 - y_1)}{(x_2 - x_1)}\right) \pmod{p}$$

olmak üzere

$$P_1 + P_2 = P_3 = (x_3, y_3)$$

bulunur. Burada m, P_1 ve P_2'den geçen doğrunun eğimidir. [14]

7.3.1.2. P Noktasını İkiye Katlama

$y^2 = \left(x^3 + ax + b\right) \pmod{p}$ eğrisi göz önüne alınsın. $P = (x_1, y_1)$ noktası için $y_1 \neq 0$ ise,

$$x_3 = \left(m^2 - 2x_1\right) \pmod{p}$$
$$y_3 = \left(m(x_1 - x_3) - y_1\right) \pmod{p}$$

ve

$$m = \left(\frac{\left(3x_1^2 + a\right)}{\left(2y_1\right)} \right) \left(\mathrm{mod}\, p\right)$$

olmak üzere

$$2P = R = \left(x_3, y_3\right)$$

bulunur. [14]

7.4. F_{2^m} Üzerinde Eliptik Eğri Grupları

F_{2^m} cisminin elemanları *m*-bitlik dizilerdir. F_{2^m} bit dizileri üzerinde işlem gördüğünden, bilgisayarlar bu cisimlerde çok verimli bir şekilde aritmetik yapabilirler.

Esas cisim F_{2^m} ile birlikte eliptik bir eğri F_{2^m}'den *a* ve *b*'yi seçerek oluşturulur (burada tek şart *b*'nin 0'dan farklı olmasıdır). Karakteristiği 2 olan F_{2^m} cisminin bir sonucu olarak, eliptik eğri denklemi, ikili sunum için (7.4)'tek gibi düzenlenebilir.

$$y^2 + xy = x^3 + ax^2 + b \tag{7.4}$$

Eliptik eğri, F_{2^m} üzerinde eliptik eğri denklemini sağlayan bütün $\left(x, y\right)$ noktalarını içerir (*x* ve *y* de F_{2^m}'nin elemanıdır). F_{2^m} üzerindeki eliptik eğri grubu sonsuzdaki bir *O* noktası ile beraber ilgili eliptik eğri üzerindeki noktaları içerir. Böyle bir eliptik eğride sonlu sayıda nokta vardır. [15]

7.4.1. F_{2^m} Üzerindeki Eliptik Eğri Grubunda Aritmetik

F_{2^m} üzerindeki eliptik eğri grubu sonlu sayıda notla içerir ve bunun aritmetiği yuvarlama hatası içermez. F_{2^m} aritmetiği bilgisayarlar tarafından çok verimli bir şekilde yapılır.

7.4.1.1. Farklı P_1 ve P_2 Noktalarını Toplama

$P=(x_P, y_P)$ noktasının negatifi $-P=(x_P, x_P+y_P)$'dir. $P_1=(x_1, y_1)$ ve $P_2=(x_2, y_2)$ noktaları $P_1 \neq -P_2$ olacak şekilde birbirlerinden farklılarsa

$$x_3 = m^2 + m + x_1 + x_2 + a$$
$$y_3 = m(x_1 + x_3) + x_3 + y_1$$

ve

$$m = \frac{(y_1 - y_2)}{(x_1 + x_2)}$$

olmak üzere

$$P_1 + P_2 = P_3 = (x_3, y_3)$$

bulunur.

Reel sayılardaki gibi sonsuzdaki nokta O için $P+(-P)=O$ olur. Ayrıca bütün P noktaları için de $P+O=P$ olur. [16]

7.4.1.2. P Noktasını İkiye Katlama

$P=(x_1, y_1)$ için eğer $x_1=0$ ise, $2P=0$ olur.

$x_1 \neq 0$ kabul edilirse,

$$x_3 = m^2 + m + a$$
$$y_3 = x_1^2 + (m + x_1) \times x_3$$

ve

$$m = x_1 + \frac{y_1}{x_1}$$

olmak üzere

$$2P = R = (x_3, y_3)$$

bulunur. m burada da eğimdir. [16]

8. ELİPTİK EĞRİ ŞİFRELEME

8.1. Eliptik Eğri Tabanlı Kriptografi

Eliptik eğrilerin kriptografide kullanılması birbirinden bağımsız olarak 1986'da Miller[7] ve 1987'de Koblitz[8] tarafından önerilmiştir.

Bilgisayarların işlem yapma yetenek ve hızından yararlanılarak düzenlenecek bir kriptanaliz işleminde, n bit uzunluğundaki anahtarı kırmak için yaklaşık $(2^n - 1)$ adet işlem yapılması gerekir. Bu değer, bilgisayarların günümüzde sahip oldukları kapasite ve bilgisayarın hızlı gelişimi için bir ölçüt tanımlayan Moore yasası[9] uyarınca hesaplandığında, ortaya çıkan eşdeğer güvenlik seviyesindeki anahtar uzunlukları Tablo 8.1'de sunulmuştur.

Tablo 8.1'de yer alan üç sütunun her birisinde sunulan kriptosistemler, farklı uzunluklardaki anahtar boyları için, her satırda eşdeğer güvenlik seviyesini sağlamaktadır. Çünkü, bu kriptosistemler farklı matematiksel problemler üzerine kurulmuştur ve matematiksel problemlerin her birisi farklı karmaşıklık seviyesindedir.

Tablo 8.1'de görüldüğü gibi, eşdeğer güvenlik düzeyinin sağlanması için 256 bit uzunluğunda anahtar kullanan simetrik kriptosisteme karşın; faktörizasyon problemini kullanan RSA ve/veya ayrık logaritma problemini kullanan Diffie-Hellmann (DH) için anahtar uzunluğunun bundan 60 kat ve eliptik eğri kriptosisteminse 2 kat daha uzun anahtar kullanması gerekmektedir. Öte yandan; daha anlamlı bir kıyaslama için, RSA-DH ikilisinin gereksinim duyduğu anahtar uzunluğu, eliptik eğri tabanlı kriptosistemin 29.4 katıdır.

[7] V. Miller, "Use of Elliptic Curves in Cryptography", Lecture Notes in Computer Science, Advances in Cryptology citation 1986, s. 417- 426.
[8] N. Koblitz, "Elliptic Curve Cryptosystems", 1987, s. 203-209.
[9] http://www.intel.com/research/silicon/mooreslaw.html

Ayrıca, eliptik eğrilerle aritmetik işlem maliyetleri, asimetrik ailesinin diğer üyelerine göre daha düşüktür. Bu nedenle, Amerikan Ulusal Teknoloji ve Standartlar Enstitüsü (National Institute of Standards and Technology – NIST), 2008 yılına kadar sayısal imza uygulamalarında faktörizasyonu kullanan RSA için 2048 veya 3072 bit anahtar uzunluğunu, eliptik eğri tabanlı kriptografiyi kullananan ECDSA'da da 224 veya 283 bit uzunluğundaki eliptik eğrileri önermektedir[10].

Tablo 8.1: Eşdeğer güvenlik seviyeleri için NIST tarafından önerilen anahtar uzunlukları[11].

Simetrik Kriptosistemlerde Anahtar Uzunluğu (bit)	RSA ve Diffie-Hellmann Anahtar Uzunluğu (bit)	Eliptik Eğri Anahtar Uzunluğu (bit)
80	1024	160
112	2048	224
128	3072	256
192	7680	384
256	15360	521

Mobil dünya uygulamaları olan e-devlet, e-finans, e-ticaret; internet, kablosuz iletişim ortamları ve mobil ekipmanların yarattığı alt yapıyı kullanır. Bu altyapıda gözlemlenen en büyük sıkıntı, bant genişliği, güç, hesaplama ve bellek kapasitelerindeki sınırlamalardan kaynaklanmaktadır. Tüm bu sınırlamalara rağmen yüksek güvenlik seviyelerini sağlayabilecek çözümlere gereksinim vardır. EEK, asimetrik kriptografinin bir bileşeni olarak, sanal dünyanın tüm güvenlik

[10] W. T. Polk, D. F. Dodson, W. E. Burr, "Cryptographic Algorithms and Key Sizes for Personal Identity Verification", April 2005 http://csrc.nist.gov/publications/nistpubs/800-78/sp800-78-final.pdf, Information Technology Laboratory National Institute of Standards and Technology, MD, 20899-8930, s. 6.

[11] National Security Agency, Central Security Service, "The Case for Elliptic Curve Cryptography", http://www.nsa.gov/ia/industry/crypto_elliptic_curve.cfm

gereksinimlerini karşılayabilecek yetkinliktedir ve alternatiflerine göre daha kısa anahtar boyları ile yüksek güvenlik seviyelerini sağlayabilmektedir. [1]

8.2. Eliptik Eğrilerde Aritmetik İşlemler

Eliptik eğriler, 3. dereceden polinomlar kullanılarak tanımlanır. Polinom, birbirini dik kesen eksenlerin oluşturduğu iki boyutlu düzlem ya da üç boyutlu uzay olan, kartezyen (Euclidean) koordinat sistemi üzerinde eliptik eğriyi tanımlar.

Herhangi bir nesne üzerinde bir noktayı tanımlamak için gereksinim duyulan sayısal değerlerin adedi, koordinat sisteminin boyutunu belirler. Örneğin, bir dörtgen iki boyutluyken, küp üç boyutludur. Tanım genelleştirildiğinde; n boyutlu bir koordinat sistemi; $R^n = (x_1, x_2, \cdots, x_n)$ gibi n adet gerçel sayı ile gösterilebilir.

Bu bölümde, eliptik eğri (EE) aritmetiği, iki ve üç boyutlu kartezyen koordinat sistemleri olan;

- İki boyutlu
    - Afin (Affine)
- Üç boyutlu
    - Projektif (Projective)
    - Jacobian
    - Modified Jacobian
    - Chudnovsky Jacobian

koordinat sistemleri için incelenecektir.

Farklı koordinat sistemleri arasında geçişi tanımlayan eşleşme fonksiyonları kullanılarak, aynı eliptik eğri için gerekli aritmetik işlemler farklı koordinat sistemlerinde yapılabilir.

$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$ "Weierstrass" eşitliğiyle eliptik eğri 3. dereceden polinom olarak, iki boyutlu koordinat sisteminde $\mathbb{F}$ cismi üzerinde

$a_1, a_2, a_3, a_4, a_5, a_6 \in \mathbb{F}$ olarak tanımlanır. Diskriminant $\Delta \neq 0$ olmak zorundadır, böylece, polinom birden fazla (x, y) çözüm kümesine sahip, tekil-olmayan bir eliptik eğriyi tanımlar. Çözüm kümesinin içindeki tüm (x, y) elemanları $\mathbb{F}$ cismi altında bir L halkası oluşturur, $L \supseteq \mathbb{F}$ ve $E(L)$ olarak gösterilir. L halkasının içindeki (x, y) nokta çiftleri, kiriş ve teğet kuralının uygulandığı nokta toplama işlemiyle elde edilir. $(x, 0)$ noktası, y eksenine paralel bir doğruyu tanımladığı için eliptik eğriyi sonsuzda keser[12].

$\mathbb{F}_p$, asal bir cisim olduğunda, eliptik eğriyi tanımlayan polinom $E : y^2 = x^3 + ax + b$ normal formunda olup, $a, b \in \mathbb{F}_p$, $\Delta = -16(4a^3 + 27b^2)$'dir. Eliptik eğri için çözüm kümesi; $E(L) = \{O\} \cup \{(x, y) \in LxL \mid y^2 = x^3 + Ax + B\}$, L halkasını oluşturur $\Delta \neq 0$.

Eliptik eğriyi tanımlayan polinomu sağlayan (x, y) değerleri, nokta çiftleme (point doubling) ve nokta toplama (point addition) operasyonlarının ardışık olarak sonsuzdaki noktaya ulaşıncaya kadar tekrarlanması sonucu elde edilir[13]. Bu ardışık nokta toplama, eliptik eğri üzerinde seçilen bir $P_1 = (x_1, y_1)$ noktasından başlar. İlk işlem nokta çiftlemedir: $2P = P_1 + P_1$ ve $2P$ elde edildikten sonra, $3P = 2P + P_1$ nokta toplama işlemiyle ardışık olarak devam edilir.

Nokta toplama ve çiftleme iyi bilinen ve kolay uygulanabilen işlemler olmasına rağmen, işlemsel maliyetleri çok yüksektir. Bu yüksek maliyetler, eliptik eğri tabanlı kriptografinin akıllı kartlar gibi kısıtlı kaynakların kullanıldığı ortamlarda yazılım uygulaması olarak kullanımını hız probleminden dolayı engellemekte ve donanıma gömülü olarak kullanımını zorunlu kılmaktadır.

Bu bölüm içinde, öncelikle bir nokta toplama ve nokta çiftleme algoritmasını oluşturan temel aritmetik operasyonlar, gereksinim duydukları işlemci süreleri dikkate alınarak milisaniye cinsinden değerlendirilecektir. Böylece; seçilen bir cisim

[12] Lawrance C. Washington, "Elliptic Curves, Number Theory and Cryptography", 2003, s. 9.
[13] Lawrance C. Washington, "Elliptic Curves, Number Theory and Cryptography", 2003, s. 12.

üzerinde çalışmayı sağlayacak aritmetik operasyonlar, maliyeti yüksek olandan daha az maliyetli olana doğru sıralanabilecektir. [1]

8.2.1. Nokta Toplama Aritmetiğinde İşlem Maliyetleri

Asal bir cisim olan $\mathbb{F}_p$ üzerinde tanımlı eliptik eğri için nokta toplama ve nokta çiftleme işlemi anahtar çiftinin oluşturulması, mesajın şifrelenmesi/deşifrelenmesi, sayısal imzanın oluşturulması ve doğrulanmasında, yoğun olarak kullanılmaktadır. Bu operasyonlarda maliyeti oluşturan aritmetik işlem tipleri ve her bir aritmetik işlem tipinin gereksinim duyduğu hesaplama maliyetinin tanımlanması, maliyeti düşürülecek aritmetik işlemlerin belirlenmesi için önemlidir. Bu durumda:

Nokta toplama aritmetiğindeki temel aritmetik işlemlerin zamansal maliyetleri; işlemcisi Pentium 4, hızı 1.7GHz, ana bellek kapasitesi 512MB, önbelleği 256KB, işletim sistemi Windows XP olan bir bilgisayar ve kodlamada ANSI C derleyicisiyle, CRYMPIX[14] (sürüm 0.1.2.1) yazılım kütüphanesi kullanılarak 256 bit ve 512 bit uzunluğundaki sayılar için analiz edilmiştir. Analiz sonuçları Tablo 8.2'de sunulmuş olup, çizelgede yer alan temel aritmetik işlemler aşağıdaki gibidir.

- Tersini-alma (Inversion - I); modüler aritmetikte çarpmaya göre elemanın tersinin bulunmasını tanımlar. Bir başka deyişle, a sayısının $(\mathrm{mod}\, p)$'ye göre tersinin alınması isteniyorsa $a.a^{-1} \equiv 1(\mathrm{mod}\, p)$ denkliğini sağlayan a^{-1} sayısı aranır. Bulunması çok zaman alan ve en yüksek maliyetli işlemdir. Tek bir çarpma işlemine karşı, yaklaşık 19 kat daha yüksek maliyetlidir $(1I \approx 19M)$. Bu nedenle, eliptik eğri aritmetiğinde bu operasyondan mümkün olduğunca kaçınılmaya çalışılmalıdır.

[14] http://is3.iyte.edu.tr/ , CRYMPIX yazılım geliştirme kütüphanesi.

- Çarpma (Multiplication - M); tersini alma işlemine göre daha az maliyetlidir. Bu nedenle tersini alma işlemi yerine daha fazla çarpma ve/veya toplama işleminin kullanılması tercih edilebilir.

- Toplama ve çıkarma (addition & subtraction); maliyetleri çok düşüktür; izleyen bölümlerde yer alan maliyet analizlerinde dikkate alınmamıştır.

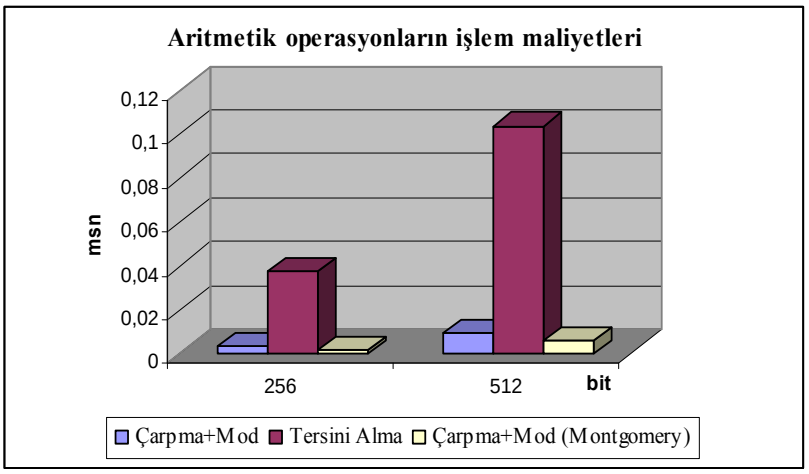
- İndirgeme (Reduction), her çarpma işleminden sonra gerçekleştirilmesi gereken bir aritmetik operasyondur, mod alma operasyonu olarak da isimlendirilebilir. Çarpma işlemi sonucunda elde edilen değerin, $\mathbb{F}_p$ cismi üzerinde bulunmasını garanti eder.

Tablo 8.2: Nokta toplama operasyonunda temel aritmetik işlem maliyetleri.

Temel	**İşlem maliyeti (msn)**	
Aritmetik İşlemler	**256 bit**	**512 bit**
Çarpma+Mod	0,003139	0,009414
Tersini Alma	0,037631	0,103347
Çarpma+Mod (Montgomery)	0,001938	0,006182

Tablo 8.2'deki değerler, Şekil 8.1'de grafik olarak da sunulmuştur. Şekilde görüldüğü gibi, Montgomery yöntemi[15] çarpma ve indirgeme işlemlerinde kullanılıyorsa; maliyet, standart çarpma ve indirgeme metoduna göre daha azdır. [1]

[15] D. Hankerson., A. Menezes, S. Vanstone, "Guide to Elliptic Curve Cryptography", 2004, s. 38- 42.



Şekil 8.1: Aritmetik operasyonların işlemsel maliyetleri.

8.2.2. Afin Koordinat Sisteminde Nokta Toplama Aritmetiği ve Maliyeti

Sonlu bir $\mathbb{F}_p$ cismi üzerinde gerçekleştirilen iki boyutlu Afin geometri, Afin düzlemi tanımlar. $\mathbb{F}_p$'nin elemanı olan, $E: y^2 = x^3 + ax + b$ eliptik eğri eşitliğini sağlayan (x, y) nokta çiftlerinin oluşturduğu noktalar kümesi ve bu noktaların biraraya gelmesiyle oluşan doğrular kümesi, Afin düzlemi oluşturur.

Sonsuzdaki nokta ve sonsuzdaki doğrunun da Afin düzlemi oluşturan kümeye dahil edilmesiyle Projektif düzlem (projective plane) tanımlanır. n elemanlı Afin düzlemin varlığı, yine n elemanlı Projektif düzlemin varlığına bağlıdır[16]. Bu durum Afin ve Projektif düzlemler arasındaki ilişkiyi tanımlayan eşleştirme fonksiyonlarının varlığını da gösterir[17].

$A_{\mathbb{F}}^2 = \{(x, y) \in \mathbb{F}x\mathbb{F}\}$; $\mathbb{F}_p$ cismi üzerindeki iki boyutlu Afin düzlemi tanımlar ve aynı cisim üzerindeki eliptik eğri $E: y^2 = x^3 + ax + b$ $(a, b \in \mathbb{F}_p, 4a^3 + 27b^2 \neq 0)$ olarak gösterilir.

[16] http://mathworld.wolfram.com/AffinePlane.html
[17] D. Hankerson., A. Menezes, S. Vanstone, "Guide to Elliptic Curve Cryptography", 2004, s. 86,87.

Afin koordinatlarda toplama işlemi, $E(\mathbb{F}_p)$ eliptik eğrisi üzerinde yeralan iki nokta $P=(x_1,y_1)$, $Q=(x_2,y_2)$ için $P+Q=(x_3,y_3)$ olarak gösterilebilir.

Tablo 8.3'te, nokta toplama ($P \neq Q$) işlemine ilişkin gerekli aritmetik ve maliyetler gösterilmiştir[18]. Yukarıda tersini alma işlem maliyetinin yüksekliği tanımlanmıştı ve bu işlemden kaçınılması gerektiği belirtilmişti. Afin koordinat sisteminde, eliptik eğri aritmetiği kolay uygulanabilir bir özellikte olmakla beraber; tersini alma işleminin gerekliliği, maliyetler için yine sorun oluşturmaya devam eder.

Tablo 8.3: Afin koordinatlarda $P \neq Q$ ve $P+Q=(x_3,y_3)$ için aritmetik işlemler ve maliyetleri.

$\lambda=(y_2-y_1)/(x_2-x_1)$	$1I \cong 19M$
$x_3=\lambda^2-x_1-x_2$	$1M$
$y_3=\lambda(x_1-x_3)-y_1$	$1M$
Toplam	$1I+2M \cong 21M$

Tablo 8.4'te nokta çiftleme $P=Q$ işlemine ilişkin gerekli aritmetik işlemler ve maliyetleri tanımlanmıştır. Tersini alma işleminin kullanımı maliyeti yükseltmekte ve $1I+3M \cong 22M$ olduğu görülmektedir. [1]

[18] Dale Husemöller D., "Elliptic Curves", Graduate Texts in Mathematics, 1987, s. 22-25.

Tablo 8.4: Afin koordinatlarda $P = Q$ ve $P + Q = (x_3, y_3)$ için aritmetik işlemler ve maliyetleri.

$\lambda = (3x_1^2 + a)/(2y_1)$	$1I + 1M \cong 20M$
$x_3 = \lambda^2 - 2x_1$	1M
$y_3 = \lambda(x_1 - x_3) - y_1$	1M
Toplam	$1I + 3M \cong 22M$

8.2.3. Projektif Koordinat Sisteminde Nokta Toplama Aritmetiği ve Maliyeti

Bir üst boyutta Afin uzayın içindeki doğruların biraraya getirilmesiyle Projektif uzay tanımlanır. $\mathbb{F}$ cismi üzerinde $x, y, z \in \mathbb{F}$ ve değerlerden en az birisinin sıfırdan farklı olduğu durumda; (x, y, z) üçlülerinin oluşturduğu eşdeğer sınıf, iki boyutlu Projektif uzayı $P_{\mathbb{F}}^2$ tanımlar. (x_1, y_1, z_1) ve (x_2, y_2, z_2) üçlülerinin, eğer $\lambda \in \mathbb{F}$ değeriyle $(x_1, y_1, z_1) = (\lambda x_2, \lambda y_2, \lambda z_2)$ denkliği tanımlanabiliyorsa, eşdeğer olduğu da söylenebilir. Bir üçlünün eşdeğer sınıfı sadece x, y ve z arasındaki sıfır olma oranına bağlıdır. Bu nedenle, bu bağımlılık $(x : y : z)$ olarak gösterilir.

Eğer $(x : y : z)$, $z \neq 0$ olan bir noktaysa, $P_{\mathbb{F}}^2$ üzerinde, $(x : y : z) = (x/z : y/z : 1)$ sonlu sayıda noktayı tanımlar. Bununla beraber $z = 0$ ise, z 'e bölme işlemi x ve y koordinatları hangi değeri alırsa alsın, sonsuzu (∞) verir, bu nedenle $(x : y : 0)$ üçlüsü Projektif uzayda sonsuzdaki noktayı tanımlar[19].

$A_{\mathbb{F}}^2 \to P_{\mathbb{F}}^2$, ek olarak $(x, y) \mapsto (x : y : 1)$ ile Afin koordinat sisteminden Projektif koordinat sistemine geçişi tanımlar[20].

[19] D. Hankerson., A. Menezes, S. Vanstone, "Guide to Elliptic Curve Cryptography", 2004, s. 87.
[20] Lawrence C. Washington, "Elliptic Curves Number Theory and Cryptography", 2003, s. 18-20.

Afin koordinat sisteminde, $\mathbb{F}_p$ cismi üzerinde eliptik eğriyi tanımlayan polinomu $E_A : y^2 = x^3 + ax + b$, Projektif koordinat sistemine geçirmek için $x = X/Z$, $y = Y/Z$ konulduğunda $E_P : Y^2Z = X^3 + aXZ^2 + bZ^3$ eşitliği elde edilir.

$P = (X_1, Y_1, Z_1)$ ve $Q = (X_2, Y_2, Z_2)$ olarak eliptik eğri üzerindeki iki noktanın toplamı $P + Q = (X_3, Y_3, Z_3)$ olarak alındığında, eğer $P \neq Q$ ise; Projektif koordinat sisteminde nokta toplama işlemini sağlayacak formüllerin de eliptik eğriyi tanımlayan polinom gibi düzenlenmesi gerekir.

$x_3 = \lambda^2 - x_1 - x_2$, $y_3 = \lambda(x_1 - x_3) - y_1$ ve $\lambda = (y_2 - y_1)/(x_2 - x_1)$ olarak Afin koordinat sisteminde uygulanmaktadır.

$x = X/Z$ ve $y = Y/Z$, $x_3 = \lambda^2 - x_1 - x_2$, $y_3 = \lambda(x_1 - x_3) - y_1$ ve $\lambda = (y_2 - y_1)/(x_2 - x_1)$ eşitliklerinde yerlerine konulup, gerekli sadeleştirmeler yapıldığında;

$$x_3' = \lambda^2 - \frac{x_1}{z_1} - \frac{x_2}{z_2},\ y_3' = \lambda\left(\frac{x_1}{z_1} - \frac{x_3}{z_3}\right) - \frac{y_1}{z_1},\ \lambda = \left(\frac{\frac{y_2}{z_2} - \frac{y_1}{z_1}}{\frac{x_2}{z_2} - \frac{x_1}{z_1}}\right) \text{ ve } z_3 = (x_2z_1 - x_1z_2)^3 z_1z_2$$

elde[21] edilir. Tablo 8.5'te Projektif koordinat sisteminde nokta toplama işlemini sağlayan bu formülleri gerçekleştirecek algoritmalarda yer alan aritmetik işlem maliyetleri gösterilmektedir.

[21] H. Cohen, A. Miyaji, T. Ono, "Efficient elliptic curve exponentiation using mixed coordinates", 1998, Advances in Cryptology - ASIACRYPT: International Conference on the Theory and Application of Cryptology, s. 3.

Tablo 8.5: Projektif koordinatlarda $P \neq Q$ ve $P+Q=(x_3, y_3)$ için aritmetik işlemler ve maliyetleri.

$u = y_2 z_1 - y_1 z_2$	$2M$
$v = x_2 z_1 - x_1 z_2$	$2M$
$A = u^2 z_1 z_2 - v^3 - 2v^2 x_1 z_2$	$6M$
$x_3 = vA$	$1M$
$y_3 = u(v^2 x_1 z_2 - A) - v^3 y_1 z_2$	$2M$
$z_3 = v^3 z_1 z_2$	$1M$
Toplam	$14M$

$P = Q$ olduğu durumda, Projektif koordinat sisteminde nokta çiftleme işlemi için:

Afin koordinat sisteminde kullanılan formüllerdeki $x_3 = \lambda^2 - 2x_1$, $y_3 = \lambda(x_1 - x_3) - y_1$, $\lambda = (3x_1^2 + a)/(2y_1)$ içindeki $x = X/Z$ ve $y = Y/Z$ olacak şekilde atanarak, eşitlikler sadeleştirilir ve;

$$x_3' = \lambda^2 - 2\frac{x_1}{z_1},\ y_3' = \lambda\left(\frac{x_1}{z_1} - \frac{x_3}{z_3}\right) - \frac{y_1}{z_1},\ \lambda = \left(3\left(\frac{x_1}{z_1}\right)^2 + a\right) / \left(2\frac{y_1}{z_1}\right)$$

ve

$$z_3 = 8(y_1 z_1)^3$$

olarak Projektif koordinat sistemi için elde edilir[22]. Tablo 8.6'da yeni eşitliklere ilişkin işlem maliyetleri sunulmuştur. Tablo 8.6'da NIST[23]'in önerdiği eliptik eğrilerin kullanılması durumunda; $w = az_1^2 + 3x_1^2$ için işlem maliyetinin $2M$'e indirgeneceği söylenebilir. Çünkü, NIST eliptik eğrilerinde $E: y^2 = x^3 + ax + b$ ve

[22] H. Cohen, A. Miyaji, T. Ono, "Efficient elliptic curve exponentiation using mixed coordinates", 1998, Advances in Cryptology - ASIACRYPT: International Conference on the Theory and Application of Cryptology, s. 3.
[23] National Institute of Standards and Technology, "Digital Signature Standard", FIPS Publication 186-2, February 2000. http://csrc.nist.gov/fips

her zaman $a = 3$'tür. Bu durumda $w = az_1^2 + 3x_1^2$ işleminde $a.z_1^2$ işleminde çarpma yerine toplama işlemi yapılabilir.

Tablo 8.6: Projektif koordinatlarda $P = Q$ ve $P + Q = (x_3, y_3)$ için aritmetik işlemler ve maliyetleri.

$w = az_1^2 + 3x_1^2$	$3M$, Nist eğrisi → $2M$
$s = y_1 z_1$	$1M$
$B = x_1 y_1 s$	$2M$
$h = w^2 - 8B$	$1M$
$x_3 = 2hs$	$1M$
$y_3 = w(4B - h) - 8y_1^2 s^2$	$4M$
$z_3 = 8s^3$	$1M$
Toplam	$13M$, Nist eğrisi → $12M$

Skaler çarpma ya da ardışık olarak nokta toplama işlemi, Afin koordinat sisteminde tanımlı bir başlangıç noktasıyla başlar. Eğer işlem maliyetleri dikkate alınarak, skaler çarpma sürecinin Projektif koordinat sisteminde uygulanması tercih edilirse; örneğin; P başlangıç noktasından başlayarak k defa nokta toplama işlemi yapılacak ve $Q = k.P$ noktası elde edilecekse, Q noktasının tekrar Afin koordinat sisteminde olması istenir. Bu nedenle, nokta değerlerinin koordinat sistemleri arasında dönüştürülebilmesi gerekir. Projektif koordinat sistemindeki bir noktanın, Afin koordinat sistemine dönüşümü;

$$\text{Projektif } (X, Y, Z) \Rightarrow x_A = X.Z^{-1}, \quad y_A = Y.Z^{-1}$$

olarak tanımlanabilir[24] ve maliyeti $2M + I \cong 21M$ 'dir. [1]

[24] Cohen H., Miyaji A., Ono T., "Efficient elliptic curve exponentiation using mixed coordinates", 1998, Advances in Cryptology -- ASIACRYPT: International Conference on the Theory and Application of Cryptology, s. 3.

8.2.4. Jacobian Koordinat Sisteminde Nokta Toplama Aritmetiği ve Maliyeti

Jacobian koordinat sisteminde, $x = \frac{X}{Z^2}$ ve $y = \frac{Y}{Z^3}$ olarak alınır. Yeni x ve y değerleri $E_A : y^2 = x^3 + ax + b$ eşitliğinde yerine atanıp, eşitlik yeniden sadeleştirilip düzenlendiğinde, Jacobian koordinat sisteminde eliptik eğriyi tanımlayan $E_J : Y^2 = X^3 + aXZ^4 + bZ^6$ polinomu elde edilir[25].

$P = (X_1, Y_1, Z_1)$, $Q = (X_2, Y_2, Z_2)$ ve $P + Q = (X_3, Y_3, Z_3)$ olduğunda; $P \neq Q$ ise Jacobian koordinat sisteminde nokta toplama işlemini sağlayan denklemleri bulmak için, yine $x_3 = \lambda^2 - x_1 - x_2$, $y_3 = \lambda(x_1 - x_3) - y_1$ ve $\lambda = (y_2 - y_1)/(x_2 - x_1)$ Afin koordinat denklemlerinde yeni $x = \frac{X}{Z^2}$ ve $y = \frac{Y}{Z^3}$ değerleri kullanılıp, bölenler sadeleştirildiğinde Tablo 8.7'de sunulan denklemlere[26] ve maliyetlere ulaşılır.

[25] Hankerson D., Menezes A., Vanstone S., "Guide to Elliptic Curve Cryptography", 2004, Springer, s. 88.

[26] H. Cohen, A. Miyaji, T. Ono, "Efficient elliptic curve exponentiation using mixed coordinates", 1998, Advances in Cryptology - ASIACRYPT: International Conference on the Theory and Application of Cryptology, s. 4.

Tablo 8.7: Jacobian koordinatlarda $P \neq Q$ ve $P+Q=(x_3, y_3)$ için aritmetik işlemler ve maliyetleri.

$u_1 = x_1 z_2^2$	$2M$
$u_2 = x_2 z_1^2$	$2M$
$h = u_2 - u_1$	
$s_1 = y_1 z_2^3$	$2M$
$s_2 = y_2 z_1^3$	$2M$
$r = s_2 - s_1$	
$x_3 = r^2 - h^3 - 2u_1 h^2$	$4M$
$y_3 = r(u_1 h^2 - x_3) - s_1 h^3$	$2M$
$z_3 = z_1 z_2 h$	$2M$
Toplam	$16M$

Eğer $P = Q$ ise Tablo 8.8'de sunulan, Jacobian koordinat sisteminde doğru nokta çiftleme denklemlerinin elde edilebilmesi için, Afin koordinat sisteminde kullanılan $x_3 = \lambda^2 - 2x_1$, $y_3 = \lambda(x_1 - x_3) - y_1$, ve $\lambda = (3x_1^2 + a)/(2y_1)$ denklemlerinde $x = \frac{X}{Z^2}$ ve $y = \frac{Y}{Z^3}$ değerleri yerleştirilip, bölenleri sadeleştirilir.

Tablo 8.8: Jacobian koordinatlarda $P = Q$ ve $P + Q = (x_3, y_3)$ için aritmetik işlemler ve maliyetleri.

$s = 4x_1 y_1^2$	$2M$
$m = 3x_1^2 + az_1^4$	$4M$, Nist eğrisi → $3M$
$t = -2s + m^2$	$1M$
$x_3 = t$	
$y_3 = m(s - t) - 8y_1^4$	$2M$
$z_3 = 2y_1 z_1$	$1M$
Toplam	$10M$, Nist eğrisi → $9M$

Jacobian koordinat sisteminde nokta çiftleme işlemi, Projektif koordinat sistemine göre daha hızlı gerçekleştirilebilirken, nokta toplama işlemi daha yavaştır.

Jacobian koordinat sisteminde tanımlı bir noktanın, Afin koordinat sistemindeki eşdeğerine dönüştürülebilmesi için; $\text{Jacobian}(X, Y, Z) \Rightarrow x_A = X.Z^{-2}$, $y_A = Y.Z^{-3}$ işlemleri gereklidir ve maliyeti $4M + I = 23M$ değerindedir.

İzleyen bölümlerde sunulan Modified Jacobian ve Chudnovsky Jacobian koordinat sistemleri; Jacobian koordinat sisteminin nokta toplama ve nokta çiftleme süreçlerinde maliyetleri biraz daha düşürebilmek için tasarlanmış varyasyonlardır. Temelde aynı nokta toplama ve çiftleme denklemleri kullanılır. [1]

8.2.4.1. Chudnovsky Jacobian koordinat sisteminde nokta toplama aritmetiği

Jacobian koordinat sisteminde daha hızlı nokta toplama işlemi yapılabilmesi için (X, Y, Z, Z^2, Z^3) beşlisi kullanılır[27].

Skaler çarpma işleminde, nokta toplama işlemi ardışık olarak tekrar eder. Bu süreç içinde, son hesaplanan $n.P$ değeri, $(n+1)P = P + nP$ işleminde kullanılacağı için, $n.P$ deki $(x_2, y_2, z_2, z_2^2, z_2^3)$ değerleri $(n+1)P$ hesaplama sürecinde hazır değerler olarak sisteme girer ve bu çarpma operasyonları elimine edilebilir.

Chudnovsky Jacobian koordinat sisteminde; $P = (x_1, y_1, z_1, z_1^2, z_1^3)$, $Q = (x_2, y_2, z_2, z_2^2, z_2^3)$ ve $P + Q = (x_3, y_3, z_3, z_3^2, z_3^3)$ olarak tanımlandığında[80];

$P \neq Q$ ise, Chudnovsky Jacobian koordinat sisteminde nokta toplama işlemini sağlayacak denklemler, Jacobian koordinat sisteminde kullanılan denklemlerle aynı olup, maliyetleriyle beraber Tablo 8.9'da sunulmuştur. z_1^2, z_1^3 ve z_2^2, z_2^3 değerleri hazır olarak işlemler başlatıldığı için, aritmetik işlem maliyeti azalmaktadır.

[27] Cohen H., Miyaji A., Ono T., "Efficient elliptic curve exponentiation using mixed coordinates", 1998, Advances in Cryptology -- ASIACRYPT: International Conference on the Theory and Application of Cryptology, s. 4.

Tablo 8.9: Chudnovsky Jacobian koordinatlarda $P \neq Q$ ve $P + Q = (x_3, y_3)$ için aritmetik işlemler ve maliyetleri.

$u_1 = x_1(z_2^2)$	$1M$
$u_2 = x_2(z_1^2)$	$1M$
$h = u_2 - u_1$	
$s_1 = y_1(z_2^3)$	$1M$
$s_2 = y_2(z_1^3)$	$1M$
$r = s_2 - s_1$	
$x_3 = r^2 - h^3 - 2u_1h^2$	$4M$
$y_3 = r(u_1h^2 - x_3) - s_1h^3$	$2M$
$z_3 = z_1z_2h$, $z_3^2 = z_3z_3$, $z_3^3 = z_3^2z_3$	$4M$
Toplam	$14M$

Eğer $P = Q$ ise; Chudnovsky Jacobian koordinat sistemi için nokta çiftleme denklemleri Jacobian koordinat sisteminde kullanılan denklemlerle aynı olup[28] Tablo 8.10'da işlem maliyetleriyle beraber sunulmuştur. [1]

[28] Cohen H., Miyaji A., Ono T., "Efficient elliptic curve exponentiation using mixed coordinates", 1998, Advances in Cryptology -- ASIACRYPT: International Conference on the Theory and Application of Cryptology, s. 4.

Tablo 8.10: Chudnovsky Jacobian koordinatlarda $P = Q$ ve $P + Q = (x_3, y_3)$ için aritmetik işlemler ve maliyetleri.

$s = 4x_1 y_1^2$	$2M$
$m = 3x_1^2 + a(z_1^2)^2$	$3M$, Nist eğrisi → $2M$
$t = m^2 - 2s$	1M
$x_3 = t$	
$y_3 = m(s-t) - 8y_1^4$	$2M$
$z_3 = 2y_1 z_1$, $z_3^2 = z_3 z_3$, $z_3^3 = z_3^2 z_3$	$3M$
Toplam	$11M$, Nist eğrisi → $10M$

8.2.4.2. Modified Jacobian koordinat sisteminde nokta toplama aritmetiği

Jacobian koordinat sisteminde daha hızlı nokta çiftleme işleminin yapılabilmesi için (X, Y, Z, aZ^4) dörtlüsünün kullanımı önerilir. Chudnovsky Jacobian'da önerildiği gibi, algoritmik olarak, ardışık olan süreç içinde girdi parametresi olarak (x_1, y_1, z_1, az_1^4) değerlerinin hazır verilmesiyle, çarpma işleminden tasarruf edilmesi sağlanır[29].

$P = Q$ olduğu durumda, Modified Jacobian koordinat sisteminde, Jacobian koordinat sistemiyle aynı denklemler[30] kullanılmakla beraber, $a.z_1^4$ değeri hazır geldiğinden Tablo 8.11'de sunulduğu gibi işlem maliyeti azalır. [1]

[29] Yuome Hitchcook, Edward Dawson, Andrew Clark, Paul Montague, "Implementing an Efficient Elliptic Curve Cryptosystem over GF(p) on a Smart Card", The 10th Biennial Computational Techniques and Applications Conference, 2001, s. 3.

[30] Cohen H., Miyaji A., Ono T., "Efficient elliptic curve exponentiation using mixed coordinates", 1998, Advances in Cryptology -- ASIACRYPT: International Conference on the Theory and Application of Cryptology, s. 3.

Tablo 8.11: Modified Jacobian koordinatlarda $P = Q$ ve $P + Q = (x_3, y_3)$ için aritmetik işlemler ve maliyetleri.

$s = 4x_1 y_1^2$	$2M$
$m = 3x_1^2 + (az_1^4)$	$1M$
$t = m^2 - 2s$	$1M$
$x_3 = t$	
$y_3 = m(s-t) - 8y_1^4$	$2M$
$z_3 = 2y_1 z_1$, $az_3^4 = 2.8y_1^4.(az_1^4)$	$2M$
Toplam	$8M$

8.2.5. Skaler Çarpmada İşlemlerin Farklı Koordinat Sistemlerinde Sağlanması

Skaler çarpma işlemi; eliptik eğri üzerinde seçilen bir $P = (x, y)$ noktasından başlayarak, k defa nokta çiftleme işlemiyle başlayıp, nokta toplama işlemiyle devam eden ve sonuçta varılan $Q = k.P$ noktasına ulaşma sürecini tanımlar. Bu sürecin en hızlı ve en az işlem ve zaman maliyetiyle sağlanıyor olması eliptik eğri tabanlı kriptografi uygulamalarının kullanılabilirliliğini belirleyen önemli bir parametredir. Bu nedenle, skaler çarpma sürecinde, hangi koordinat sisteminde ya da sistemleri üzerinde aritmetiğin gerçekleştirileceğine karar verilmelidir. [1]

8.2.6. Nokta Toplama ve Nokta Çiftleme Uygulamaları

Tez çalışması uyarınca, yukarıda verilmiş tanımlar ve sunulmuş olan koordinat sistemleri doğrultusunda olmak üzere; NIST eğrileri kullanılarak[31] nokta toplama ve

[31] National Institute of Standards and Technology, "Digital Signature Standard", FIPS Publication 186-2, February 2000. http://csrc.nist.gov/fips

çiftleme işlemlerini sağlayan algoritmalar, Crympix kriptografik yazılım kütüphanesi[32] (sürüm 0.1.2.1) yardımıyla kodlanmış ve milisaniye cinsinden hız değerleri gözlemlenmiştir.

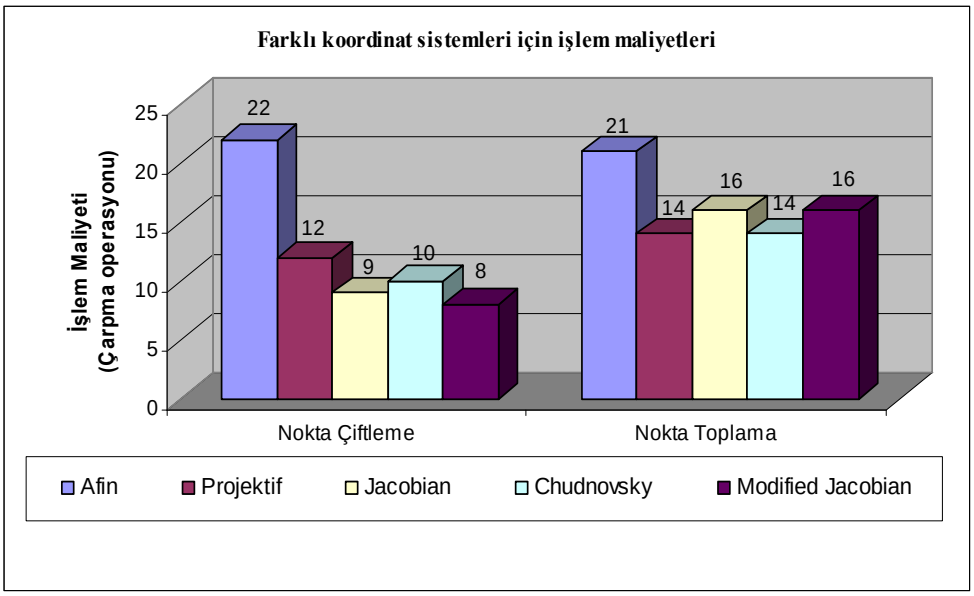
Yapılan çalışmada elde edilen sonuçlar Şekil 8.2'de bir grafikte toplandığında, farklı koordinat sistemleri üzerinde eliptik eğri aritmetiğinin algoritmik analizlerine uyumlu zaman gereksinimleri gösterdiği görülmüştür.

Afin koordinat sistemi, diğer koordinat sistemlerine göre daha az sayıda işlem adımı içermektedir, bu nedenle daha kolay uygulanabilir. Eliptik eğri üzerindeki noktaların (x, y) gibi az sayıda parametreyle tanımlanabilmesi ve haberleşmede en az bant genişliğine gereksinim göstermesi avantajları olarak görülebilir[33].

Diğer koordinat sistemleri noktayı tanımlarken fazladan parametre kullandıkları için daha büyük bir bant genişliğine gereksinim duyar. Örneğin, Jacobian ve Projektif koordinat sistemleri üçlü, Chudnovsky Jacobian beşli, Modified Jacobian dörtlü parametre gereksinimindedir.

Jacobian koordinat sistemi, Projektif koordinat sistemine göre daha hızlı nokta çiftleme işlemi yaparken, nokta toplamada daha yavaştır. Jacobian koordinat sisteminin varyasyonu olan Modified Jacobian nokta çiftleme işlemine, Chudnovsky Jacobian ise nokta toplama işlemine odaklanarak işlem maliyetlerini daha düşük bir değere getirir. [1]

[32] Koltuksuz, A., Hişil, H., Crympix: Cryptographic Multiprecision Library, Springer-Verlag, LNCS 3733, vol. 3733/2005, s. 884-893.
[33] Yuome Hitchcook, Edward Dawson, Andrew Clark, Paul Montague, "Implementing an Efficient Elliptic Curve Cryptosystem over GF(p) on a Smart Card", The 10th Biennial Computational Techniques and Applications Conference, 2001, s. 3.



Şekil 8.2: Farklı koordinat sistemleri için milisaniye olarak işlemsel zaman değerleri.

8.2.7. Farklı Koordinat Sistemleri Arasında Geçiş Maliyetleri

Skaler çarpma sürecinde, eliptik eğrilerde nokta toplama ve nokta çiftleme işlemleri için en az işlem maliyetine sahip koordinat sistemlerinin birarada kullanımı (mixed coordinate systems) Cohen[34] tarafından önerilmiştir.

Eliptik eğri üzerindeki bir P başlangıç noktası Afin koordinat sisteminde verilir. $Q = k.P$'de tanımlı Q noktasına ulaşıncaya kadar gereksinim duyulan hesaplamaların en hızlı ve en az işlem maliyetiyle tamamlanması, farklı koordinat sistemlerinin kullanımıyla sağlanacaksa, P değeri istenilen farklı koordinat sistemlerine dönüştürülerek skaler çarpma algoritması yürütülmelidir. Q noktasına ulaşıldığında da, değerin yine Afin koordinat sistemine dönüştürülmesi gerekecektir.

Afin koordinat sisteminde nokta(x, y) şeklinde iki sayısal değer kullanılarak tanımlanabilmektedir. Afin'de tanımlı bir nokta $z = 1$ kabul ederek, Projektif ya da Jacobian koordinat sistemine geçiş yapılabilir.

[34] H. Cohen, A. Miyaji, T. Ono, "Efficient elliptic curve exponentiation using mixed coordinates", 1998, Advances in Cryptology - ASIACRYPT: International Conference on the Theory and Application of Cryptology, s. 5.

Projektif koordinat sisteminde nokta $\left(\frac{x}{z},\frac{y}{z},1\right)$ alındığına tekrar Afin koordinat sistemine geçişi için $X_A=\left(\frac{x}{z}\right).z^{-1}$, $Y_A=\left(\frac{y}{z}\right).z^{-1}$ işlemleri yapılmalıdır.

Öte yandan, Jacobian koordinat sisteminde nokta $\left(\frac{x}{z^2},\frac{y}{z^3}\right)$ olduğundan, tekrar Afin koordinat sistemine geçişte $X_A=\left(\frac{x}{z^2}\right).z^{-2}$, $Y_A=\left(\frac{y}{z^3}\right).z^{-3}$ işlemleri yapılmalıdır.

Chudnovsky Jacobian $\left(x,y,z,z^2,z^3\right)$ ve Modified Jacobian: $\left(x,y,z,a.z^4\right)$ olup, Afin koordinat sistemine geçişte Jacobian'da tanımlı işlemlere gereksinim vardır.

Koordinat sistemleri arasındaki nokta dönüşüm maliyetleri Tablo 8.12'de gösterilmiştir[35].

Tablo 8.12: Farklı koordinat sistemleri arasında nokta dönüşüm maliyetleri.

	Afin	Projektif	Jacobian	Chudnovsky	Modified Jacobian
Afin					
Projektif	$2M+I$		$2M+I$	$2M+I$	$2M+I$
Jacobian	$4M+I$	$4M+I$		$2M$	$3M$
Chudnovsky	$4M+I$	$4M+I$			$3M$
Modified Jacobian	$4M+I$	$4M+I$		$2M$	

[35] Yuome Hitchcook, Edward Dawson, Andrew Clark, Paul Montague., "Implementing an Efficient Elliptic Curve Cryptosystem over GF(p) on a Smart Card", The 10th Biennial Computational Techniques and Applications Conference, 2001, s. 3.

Tablo 8.12'de de görüldüğü gibi; Afin ve Projektif koordinat sistemleri arasında geçiş için tersini alma operasyonu, maliyeti çok yükseltmektedir. Fakat, Jacobian koordinat sistemi ve varyasyonları olan Modified Jacobian ve Chudnovsky Jacobian koordinat sistemleri arasında geçişlerde tersini alma operasyonu gerekmemektedir. Bu en düşük geçiş ve hesaplama maliyetleri nedeniyle; skaler çarpma işleminde, nokta çiftleme işleminde Modified Jacobian, nokta toplama işleminde de Chudnovsky Jacobian tercih edilebilir. [1]

9. UYGULAMA

9.1. Simetrik Şifreleme Programı

5. Bölümde anlatılan Simetrik Şifrelemeye ait bir uygulama olarak Simetrik Şifreleme Programı, Base64 kodlaması kullanılarak oluşturulmuştur. Kısaca Base64, ikili verilerin (banary data) sadece ASCII karakterlerini kullanan ortamlarda iletilmesine ve saklanmasına yarayan bir kodlama şemasıdır. Kodlama esnasında 3 baytlık veriler 6 bitlik dörtlü gruplara ayrılır. Her 6 bitlik grup 0 ile 63 arasında bir sayı oluşturur ($2^6 = 64$). Tablo 9.1'e göre her sayı bir ASCII yazdırma karakterine dönüştürülür.

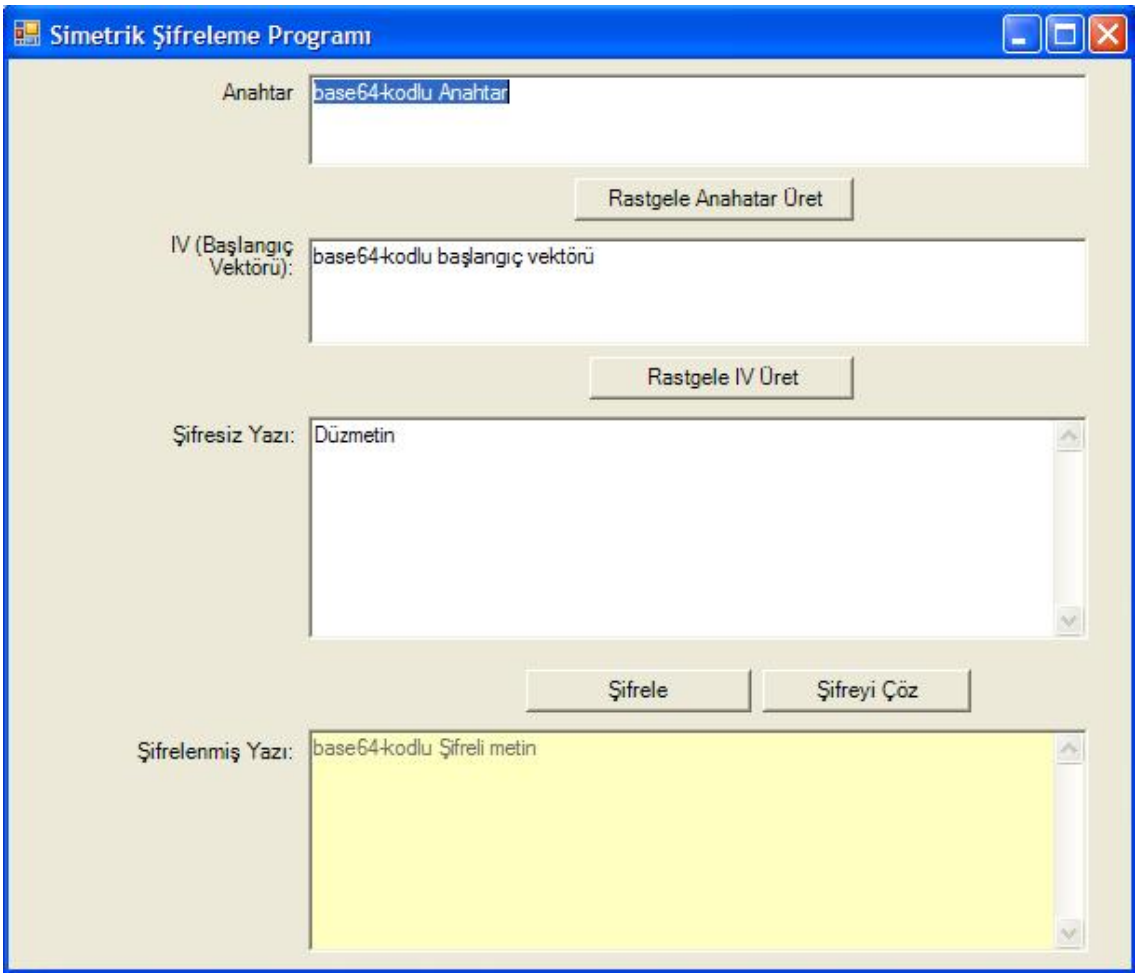
Tablo 9.1: Base64 kodlamasında kullanılan karakterler

Sayı	Karakter	Sayı	Karakter	Sayı	Karakter	Sayı	Karakter
0	A	16	Q	32	g	48	w
1	B	17	R	33	h	49	x
2	C	18	S	34	i	50	y
3	D	19	T	35	j	51	z
4	E	20	U	36	k	52	0
5	F	21	V	37	l	53	1
6	G	22	W	38	m	54	2
7	H	23	X	39	n	55	3
8	I	24	Y	40	o	56	4
9	J	25	Z	41	p	57	5
10	K	26	a	42	q	58	6
11	L	27	b	43	r	59	7
12	M	28	c	44	s	60	8
13	N	29	d	45	t	61	9
14	O	30	e	46	u	62	+
15	P	31	f	47	v	63	/

Bir Base64 kodlamasının uzunluğu 4'ün katları şeklindedir; uzunluğu 4'ün katı olmayan hiçbir metin geçerli bir Base64 metni değildir. Uzunluğu 4'ün katı olmayan bir Base64 metnine uzunluğu 4'ün katı olacak şekilde '=' eklenmelidir. Örneğin, uzunluğu 10 olan bir çıktının sonuna '==' eklenmelidir.

Örneğin, “Tarik” için Base64 kodlaması “VGFyaWs=” olacaktır.

Program çalıştırıldığında Şekil 9.1’deki görüntü ile karşılaşılır.



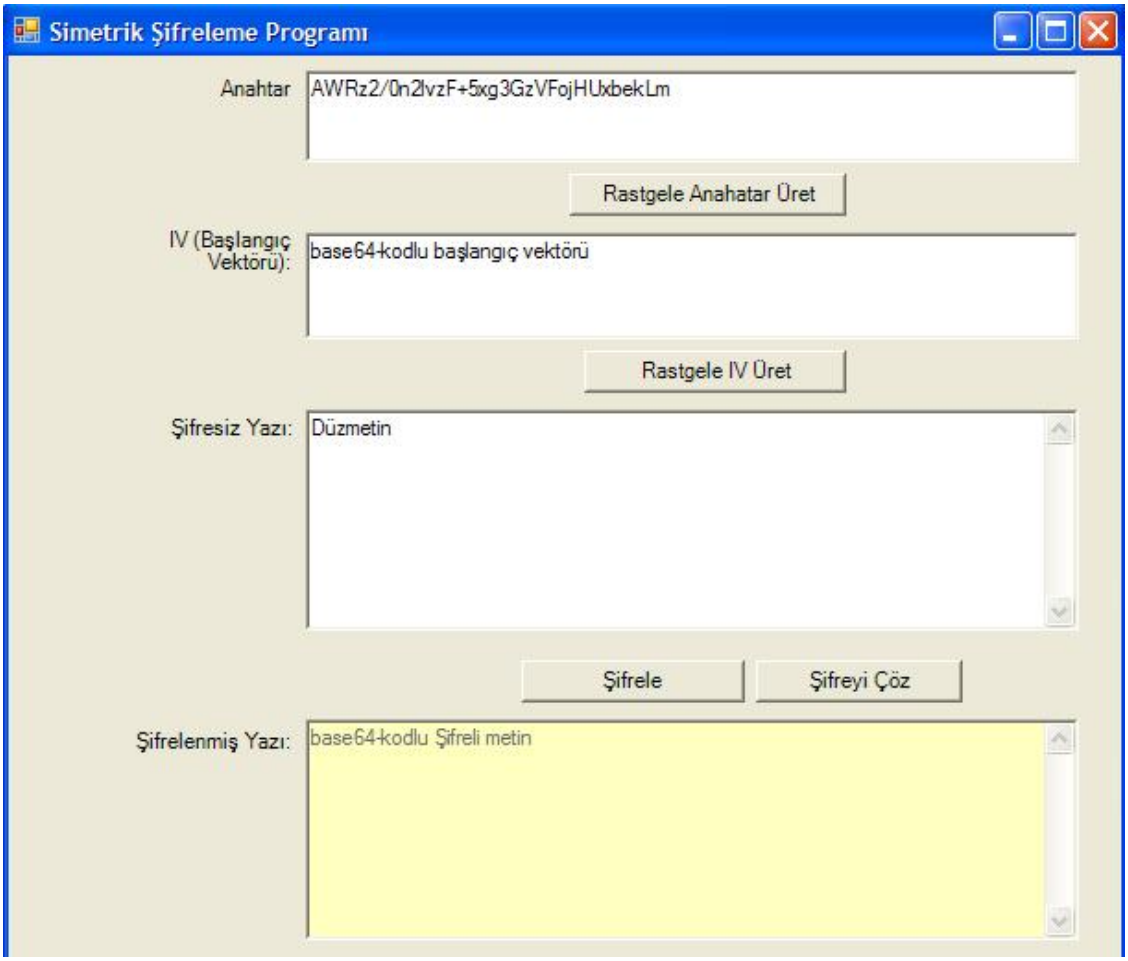
Şekil 9.1: Simetrik Şifreleme Programı başlangıç ekranı

Burada “Anahtar” kısmına Base64 kodlaması ile oluşturulan bir kod girilebileceği gibi, “Rastgele Anahtar Üret” tuşuna basarak da Base64 kodlamasıyla rastgele oluşturulan bir anahtar da otomatik olarak anahtar metin kutusuna yerleşir.

“Rastgele Anahtar Üret” tuşuna basarak Şekil 9.2’de görüldüğü gibi Base64 kodlu

AWRz2/0n2lvzF+5xg3GzVFojHUxbekLm

anahtar kodu elde edilmiş olsun. Oluşturulan bu anahtar, anahtar metin kutusuna otomatik olarak yerleşir.



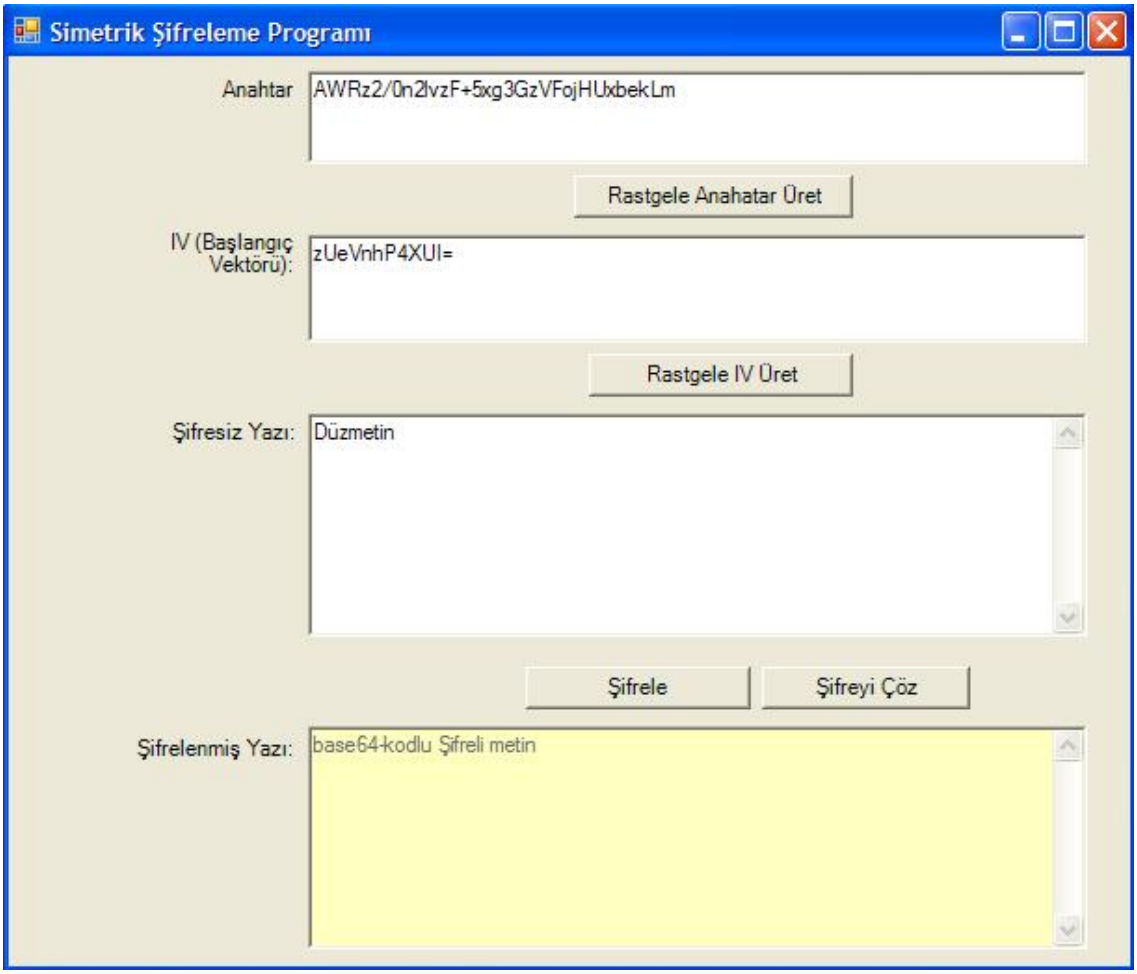
Şekil 9.2: "Rastgele Anahtar Üret" tuşu ile üretilmiş anahtar

Base64 kodlamasıyla oluşturulabilecek bir başlangıç vektörü "IV (Başlangıç Vektörü)" kısmına girilebileceği gibi, "Rastgele IV Üret" tuşuna basarak da bir başlangıç vektörü oluşturulabilir.

"Rastgele IV Üret" tuşuna basılarak Şekil 9.3'te görüldüğü gibi Base64 kodlu

zUeVnhP4XUI=

başlangıç vektörü elde edilmiş olsun.



Şekil 9.3: "Rastgele IV Üret" tuşu ile üretilmiş başlangıç vektörü

"Şifresiz Yazı" kısmına programla şifrelenmesi istenecek bir metin girilebilir. Örneğin, Base64 kodlaması Türkçe karakterler kabul etmediğinden

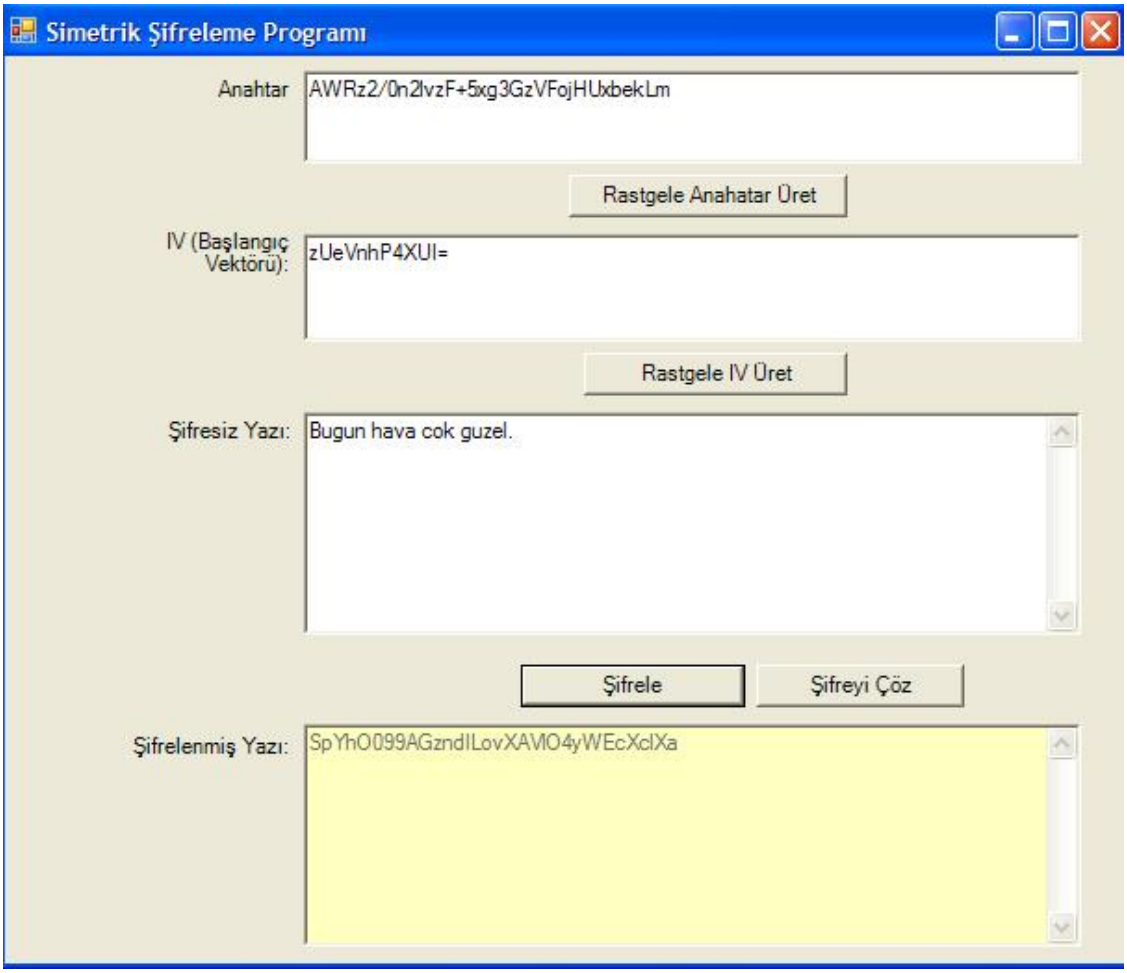
Bugun hava cok guzel.

şeklinde bir metin girildiğini varsayalım.

"Şifrele" tuşuna basıldığında EK-1'de sunulan program yardımıyla girilen düzmetin

SpYhO099AGzndILovXAVlO4yWEcXclXa

şeklinde şifrelenir. Girilen metnin şifreli durumunu Şekil 9.4'te görüldüğü gibi "Şifrelenmiş Yazı" kısmında görüntülenir.



Şekil 9.4: Verilen bilgilere göre "Şifrele" tuşuna basarak oluşturulan şifreli metin

Şifrelenmiş bir metnin deşifre edilmesi için ise aşağıdaki adımlar izlenir:

1. "Anahtar" için daha önce oluşturulan

   'AWRz2/0n2lvzF+5xg3GzVFojHUxbekLm',

   anahtarı anahtar metin kutusuna,

2. "IV (Başlangıç Vektörü)" için daha önce oluşturulan

‘zUeVnhP4XUI=’

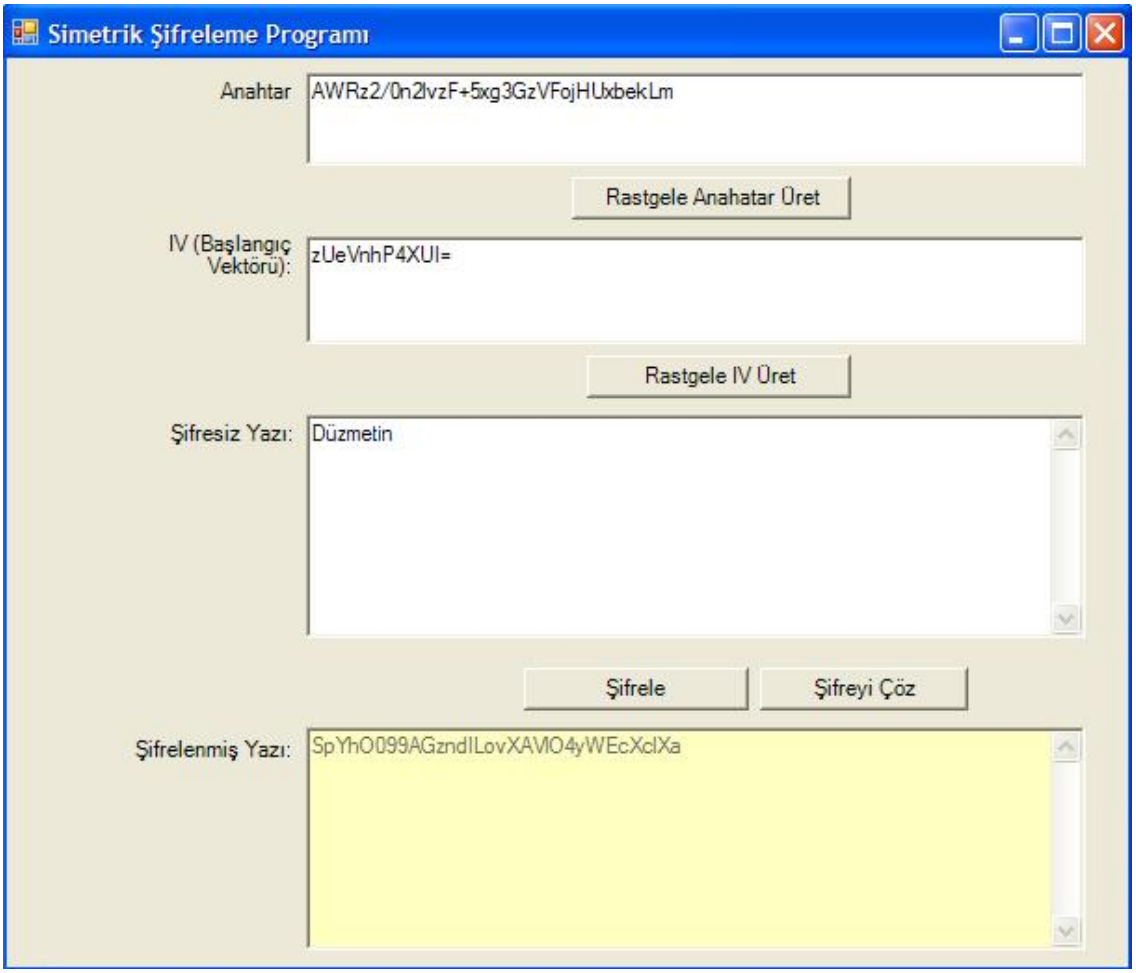
başlangıç vektörü başlangıç vektörü metin kutusuna,

3. “Şifrelenmiş Yazı” için daha önce şifrelenmiş

‘SpYhO099AGzndILovXAVlO4yWEcXclXa’

şifreli metni girilmiş olsun.

Bu durum Şekil 9.5’te görülmektedir.

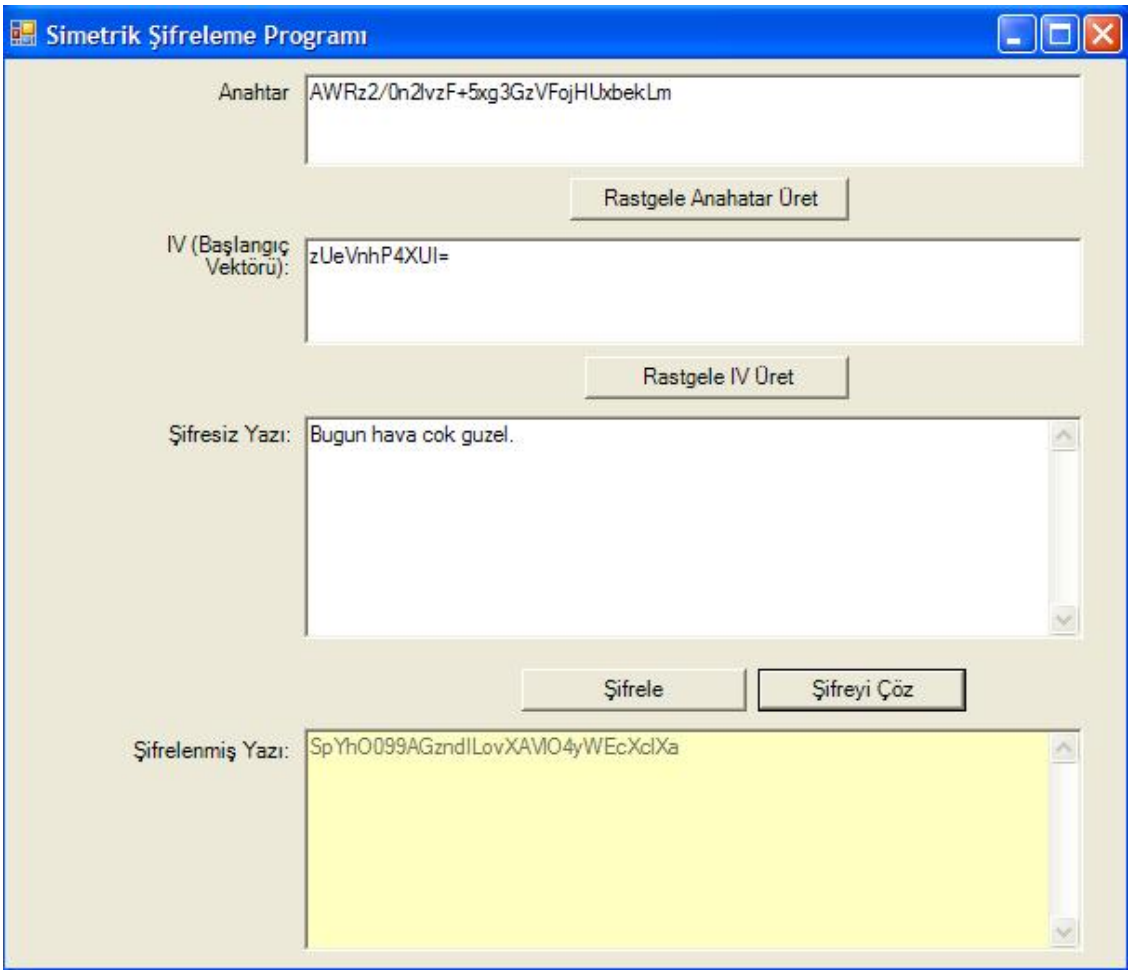


Şekil 9.5

“Şifreyi Çöz” tuşuna basıldığında düzmetin olarak girilen

Bugun hava cok guzel.

metni elde edilir. Bu durum Şekil 9.6 ile görülebilir.



Şekil 9.6: Verilen bilgilere göre “Şifreyi Çöz” tuşuna basılarak deşifre edilen düzmetin

Şimdi daha uzun bir metin girerek programı deneyelim. Metin olarak Türkçe karakter problemiyle karşılaşmamak için Dan Brown’un Digital Fortress adlı kitabından 61’inci bölüme ait aşağıdaki metni alalım.

“*Jabba lay on his back lodged halfway inside a dismantled mainframe computer. There was a penlight in his mouth, a soldering iron in his hand, and a large schematic blueprint propped on his belly. He had just finished attaching a new set of attenuators to a faulty motherboard when his cellular phone sprang to life.*

"Shit," he swore, groping for the receiver through a pile of cables. "Jabba here."

"Jabba, it's Midge."

He brightened. "Twice in one night? People are gonna start talking."

"Crypto's got problems." Her voice was tense.

Jabba frowned. "We been through this already. Remember?"

"It's a power problem."

"I'm not an electrician. Call Engineering."

"The dome's dark."

"You're seeing things. Go home." He turned back to his schematic.

"Pitch black!" she yelled.

Jabba sighed and set down his penlight. "Midge, first of all, we've got aux power in there. It would never be pitch black. Second, Strathmore's got a slightly better view of Crypto than I do right now. Why don't you call him?"

"Because this has to do with him. He's hiding something."

Jabba rolled his eyes. "Midge sweetie, I'm up to my armpits in serial cable here. If you need a date, I'll cut loose. Otherwise, call Engineering."

"Jabba, this is serious. I can feel it."

She can feel it? It was official, Jabba thought, Midge was in one of her moods. "If Strathmore's not worried, I'm not worried."

"Crypto's pitch black, dammit!"

"So maybe Strathmore's stargazing."

"Jabba! I'm not kidding around here!"

"Okay, okay," he grumbled, propping himself up on an elbow. "Maybe a generator shorted out. As soon as I'm done here, I'll stop by Crypto and--"

"What about aux power!" Midge demanded. "If a generator blew, why is there no aux power?"

"I don't know. Maybe Strathmore's got TRANSLTR running and aux power is tapped out."

"So why doesn't he abort? Maybe it's a virus. You said something earlier about a virus."

"Damn it, Midge!" Jabba exploded. "I told you, there's no virus in Crypto! Stop being so damned paranoid!"

There was a long silence on the line.

"Aw, shit, Midge," Jabba apologized. "Let me explain." His voice was tight. "First of all, we've got Gauntlet--no virus could possibly get through. Second, if there's a power failure, it's hardware-related--viruses don't kill power, they attack software and data. Whatever's going on in Crypto, it's not a virus."

Silence.

"Midge? You there?"

Midge's response was icy. "Jabba, I have a job to do. I don't expect to be yelled at for doing it. When I call to ask why a multi billion-dollar facility is in the dark, I expect a professional response."

"Yes, ma'am."

"A simple yes or no will suffice. Is it possible the problem in Crypto is virus-related?"

"Midge... I told you--"

"Yes or no. Could TRANSLTR have a virus?"

Jabba sighed. "No, Midge. It's totally impossible."

"Thank you."

He forced a chuckle and tried to lighten the mood. "Unless you think Strathmore wrote one himself and bypassed my filters."

There was a stunned silence. When Midge spoke, her voice had an eerie edge. "Strathmore can bypass Gauntlet?"

Jabba sighed. "It was a joke, Midge." But he knew it was too late."

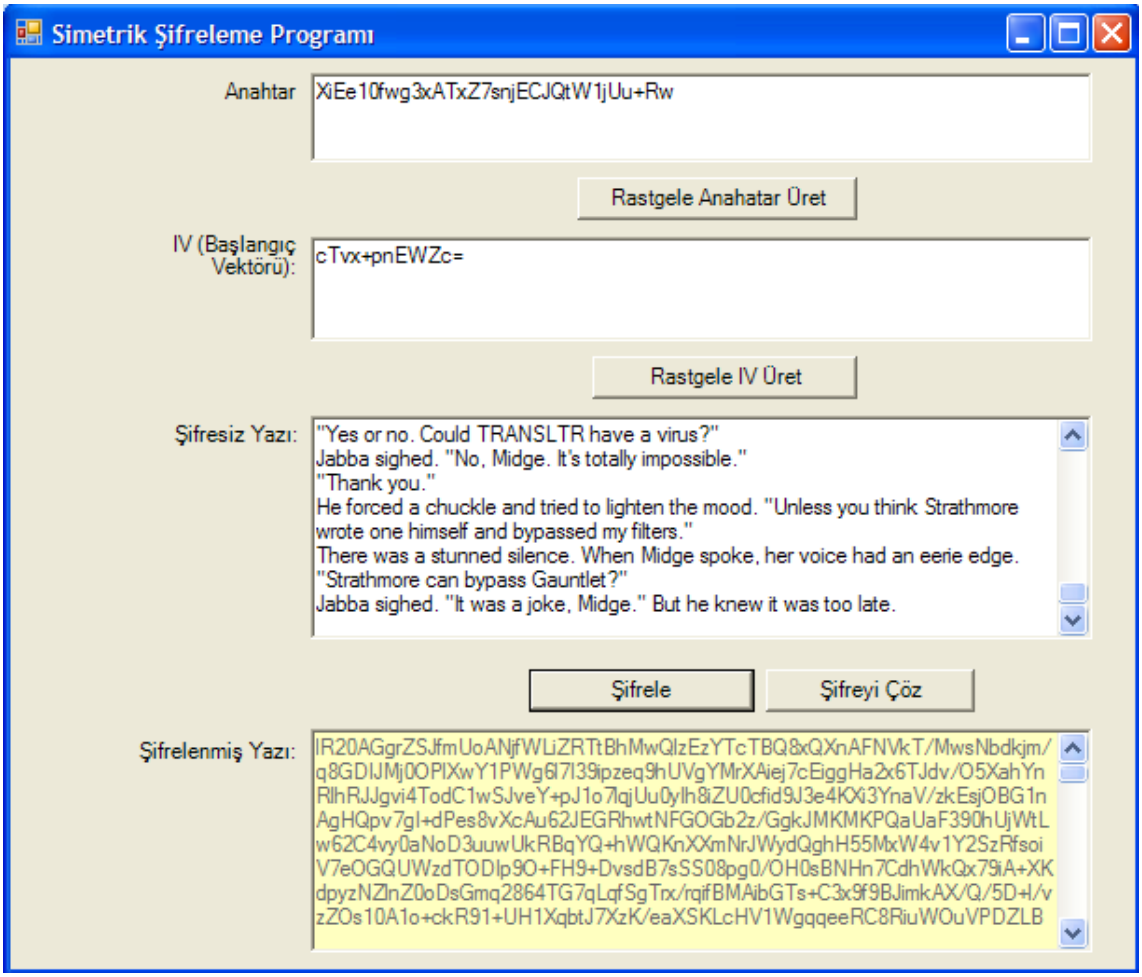
Bu metin

XiEe10fwg3xATxZ7snjECJQtW1jUu+Rw

anahtarı ve

cTvx+pnEWZc=

başlangıç vektörüyle Base64 kodlamasına göre şifrelenirse Şekil 9.7'de görüldüğü gibi "Şifrelenmiş Yazı" kısmında



Şekil 9.7: Metnin şifrelenmiş hali

"lR20AGgrZSJfmUoANjfWLiZRTtBhMwQlzEzYTcTBQ8xQXnAFNVkT/MwsNbdkjm/q8GDIJMj0OPlXwY1PWg6l7I39ipzeq9hUVgYMrXAiej7cEiggHa2x6TJdv/O5XahYnRlhRJJgvi4TodC1wSJveY+pJ1o7lqjUu0yIh8iZU0cfid9J3e4KXi3YnaV/zkEsjOBG1nAgHQpv7gI+dPes8vXcAu62JEGRhwtNFGOGb2z/GgkJMKMKPQaUaF390hUjWtLw62C4vy0aNoD3uuwUkRBqYQ+hWQKnXXmNrJWydQghH55MxW4v1Y2SzRfsoiV7eOGQUWzdTODIp9O+FH9+DvsdB7sSS08pg0/OH0sBNHn7CdhWkQx79iA+XKdpyzNZlnZ0oDsGmq2864TG7qLqfSgTrx/rqifBMAibGTs+C3x9f9BJi

mkAX/Q/5D+l/vzZOs10A1o+ckR91+UH1XqbtJ7XzK/eaXSKLcHV1WgqqeeRC8R
iuWOuVPDZLBZj0lUZcsv0XbOtRFsRApkfM78hH5XRAT4P2b677xoXEbo6VH2
Vgz0LjYDTujBaybd8031ZDM1fy48FbEHRes3IPLjkHhyoM8tMl4Qd9k/dq/sT5p40
EjVvptfSjJnNGeld2dkfxT/jQcpRNOWbkpLcdpJSMXOhoaUGoVdXdRnOoPG3fU
X9Sq5DjIF2RPL1DjkwbGVpcUVxICXNxnd4CDZj6WMiJf5EPxz27MEtE7/KG38
wCgtRjdPPiWzbMRMiVYCbnzuADEpzIeZCZTLzHTe/QgyJDtCrxGcMQPixvh0S
p/YqrnA/00SetskimnX4b6keVGzVaHi35S+uV3azFU1RX5+7s19c6GjJHNpXyQZt
YqYNzv2drFUjkqiZgmg741Ls3Ze4JqAGR9FAlugdV6+FCvBDe31e8gcasRjo+KiM
13zozB6EEGtb+4MZOKOJFawiddrz4rcGnHM3iowfkHi+fH2UHc5GhW2K4tr22O
aULArFswFpUpVdapTx5RgqsEZYFjtATiRY2YE9Rbk3UcCfBll+QzPx7wcqmqhk
DybjHdfSs2rtbzYwHW7NHbJiclvvsd0oetpiIi+plS4EyM8JtBY6/lCJyPcdRyNLqkG
RXnNa3tn2CW1oJBpOI3QYXoOW0kwsj60ihrMkm/cKTgwC0CQ2cD2jfgLWOB
G4t8jozGvoZsygbEcnc44Fk0Dto2qtmJo32hkwUe/XOX9mw3X9xC44+/rzjJkFE5Z7
bArTeConzAu/Q3erClaCZMG8tGs228w3SqzgJKH19AKqVWfHM8a1AByHpx/Eo
KmLLGOfOfhkn7hu37Cz5pKOMxwGcHuFTqNJjXe8rJtVSSiFPNvMXNQSCXrD
Msa6S4zuyQdci54XOPxqU3VEh03jkfY1mGUjTcMtcdOMzHeRFSF+K+cXmr2kD
0wUPHcHygQ7qzaURKdZAzAhBKeAYJw+Nm2zeNALFNl9T5VCCAJ5moPA0o
6LFIzCWfSDxQwZb6nFJgxhik5OL+tAX1Jkmr3aaK+58BkZFfs04xV/j5lw234pQo
1p5cqWbEZ5pgnc7n0Ntbm9CbTT1P0EdgqCBO7JDEnnyhcKNWd5HC864erjLryp
XynoQNVJfaCmsZVuPUNyfONhMOnK/uin8Z7dhiF8/RCu3LB+B2pteshxgL7BTS
SUuByr5zZxPd0hY24ipIlxOozxHIIimC3ongVPpe9DAWSfdAcjP8Nym5Khz06Oh
XMmH6Xv0emvsDLucJwinySmQ4mpYAa905ejxXGoXMd/vUXZGWVIgx0s41m/
POG+Sj2cH+ipnlCONTCJuX919CD+25wOgKQ/0aS0CG9uiFV3dspFqaY1CSI+G5
fsmh1KIw21PXObDMbCwLfwML93Wc1fSIhccXiynVZPsvSnoqEcuQdNbS/EKao
B6cvkvZrzs6AOqdHXZ6zEe6McOvcQM6F9llXTqYXeHdpAcm23Q+Uw+d75+wC
Leja97l/iKk/QAD4FD89heZ04qqy2pzCD/bIw2HOHbf/+4pj/+/HjEVm4P1QysdEaF
ulNCHxpy/3j+mdzO2INfWkEAagz9WZ+I0KEUk0WkdsIm7mJOMGmYk2q6Vgnd
ltGjJE7KP1w+X2D9PNaF4d74SQOt88UGYS6PQiEn8cxh+6UPS4pbWudWP+Bv4
v8dtHrKDwvy4ufLXlJ966qJYe11EZU/kW3c/MhlrUACL7Pqieae68ZUbyfsrWgS2h
sXAV/OQYlSuTSHPkqh6PCsmMdYRqB4q2T41lQNr7RLBXn/eR2z6iBzRZuOvS7
clBSBh43j2NMHdE5YeKRvQ/x/zNeMYOFKbttnO/W7o3lbkPMzGwnoy9vRE5uy
9PaGXZZLhrm7lQn5ljl2DmtcAtWb6Ww116bana/5cRvsQ+itvMKD57DwJJ7uIiJL/

f+G/CEB+glS92EX7+cXFcyoY4QNsSJ4C7nRPHQThu63nUJuygacuby8hZn3sJToq
ZpcexvOE6aHumjE/aUGwPgZQOOajj9FtLzqSvkQuurr+3175Y2vwxC36eZ1mQZ
Di8SWFVf+WaT2gWODavhYaYybcWjVP42fXHUqQGJkz0Rei+sUazCCe7kZtjB
BfiaCEsOswkCY15vhQz4cO5IxP2a+vGqCYauFMNro0QE0FAyU2cFg/e1RM1lTi
NYLG9TpQThiNEnwLYcfeeQW4HNh/7bV2uSHtRgS9QgfaiBkgFU+u4TTFlZKy2
ri0iCTcMmkoiRwiwTTKCC/gLCoPsOCtp8mF6pvhYX8JlaHgNtpBuwIkN6SqGjhII
frrJAkTB5TDVo81NYjNg/DMtepATCPegI8GZKnmn3k0ROJRZacenEHN1lTZ6A
CNP52C6yjdXo2+FBI14Xp4dYVOl5JqZ7ZQ8RZeTB8IBthFUXNWGq6gP5j4bkio
WUkssVMbUX7BJ8//et95zbzRgE0ZSFI4WNPHN1OEWJje/Yqa2OmPW+S4abQ9
ZeyKt1gfa5tCo1zNz2eWCtxThvzBpICnZm9dfPVQhIbvOJuSQEVjMjkLOjBgFluZ
HfW4yKcVXEQ8XUfWC9jHsAEYQoCFpwqAVeBwW7i1o4I2imrcDDbBrrIAHw
OQZltj1tFV52sDi9M6LuZjgV5ApVhf0UzDIj03W4bNBaaegnl3QvBhPhEqB9qMx
S8sDnp65a5KnQJh7sgnqirpJxUG48vLhdleBlxludf+rF3NbhUeQPT8pX/mb1KBa+a
zOH+vYXASUYVafTl1yH7mLJWYqCRLAEGJXv2XQmHRPf9+eniM2tMn2ExKl
IOBLXBVnRo0MvfcnyO9nVZz7+x/FiMGmjNmz1F6UIRSZXOssCLPYi4OLVikg
70OvHSI4/ONy3UiCbV11TIJ+D0eLyxt9LzKN0lA5VuhuHnch0/CVfZgJlN23xkmI
xQwltABPOyx1RR9q0jpuwedPaXE/Tq4qD81khmXRI1/hjWtp1TIUoNOgLjrpaVm
NY3Hutzwh0vB75stcVWzXSFQ3STFdu2ayaBlGZbpxazQZd9423dbpcC2sf9tRqUi
cx2Gm7ja35jysWQ1Y10FLObFq5j+945XUtEo+i2TSAV03IVb+6SBmUciXu72l9js
+N2L9V86MlyC4QNWw7DvpoqOiRtsfr4neBXuje1+m0z7ZWSziKqwjMC8YP8Kd
6tmesT10TJR4olOOqJyVkF0SUw2MwdFG1Y6MeDa+pInXobwe7n3geCNkClQ1l
PAj+qRD4uVMZNbregpXic2c/KfwDE7Mek7SZy8+dMaEwuiRLF1bev69tZ4wBsf
LbTcSpR5zjmHDN/MYR7/3mte9wKf6IXbhry/0cB4pXy1c0qZzQ8ARqUbfGW/F1
hCpRGeAdzKMzeU5L01T5ksg4tKnyU9uNSP+3sUM0mxQ8uIynLpnZIZ+/GLjhnj
5DpDW+NtLX6E+85ZWhb8BxiVQrb0IUr9jDy6rsJ+HR+TF+TWDY8Ok6vt5cihiF
WtIkFcvJXs618XQSMUr323I8q1+4uZxcZJf5Vv1dywlUG3io6n/C1n/8Q0J6YXNz
O6AvShj+thf1UrVgJsJhKPlNbQKU/vGpIT8bTX8lDxt9DNlLsC19DuF7RVdffpQD
roLMhZ+Y80rFIFbS7N2oEHPDIJ3XtKSCPo/V0cApI9cvrc3VeGC2Rtpw6mZ8eSZ
KAnRkQEKrfrkNyiPf2KnhfPTd2oFpd8mAbX5o43X2VG8GxfoUNJjizg==”

şifreli metni elde edilir. Benzer şekilde daha önce şifrelenmiş bir metin oluşturulan simetrik şifreleme anahtarı ve başlangıç vektörü yardımıyla deşifre edilerek düzmetin

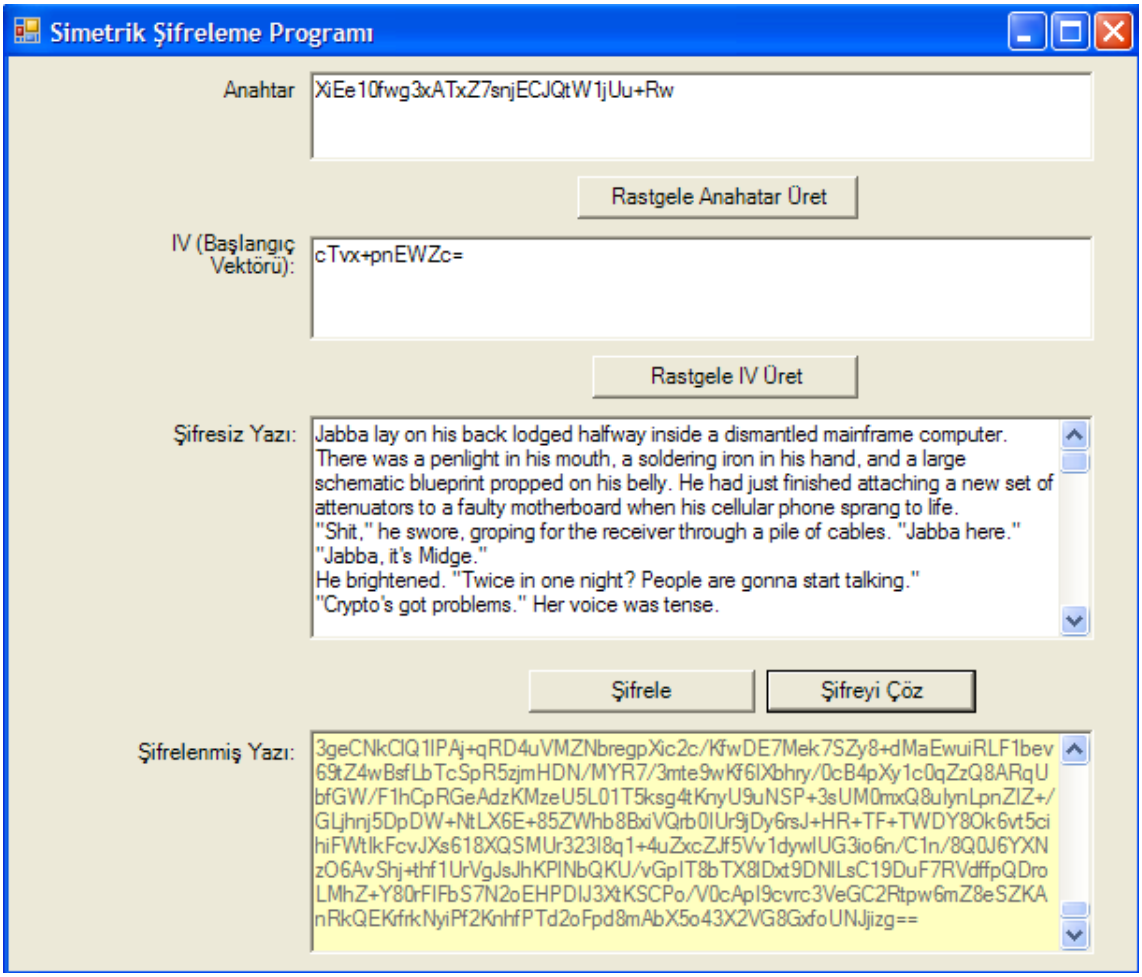
haline dönüştürülür. Örneğin, Dan Brown'un Digital Fortress adlı kitabından 61'inci bölüme ait önceden elde ettiğimiz şifrelenmiş hali "Şifrelenmiş Yazı" metin kutusuna girilip daha önce düzmetni şifrelemek için kullanılan

XiEe10fwg3xATxZ7snjECJQtW1jUu+Rw

anahtarı ve

cTvx+pnEWZc=

başlangıç vektörü kullanılarak "Şifreyi Çöz" tuşuna basılarak düzmetin haline dönüştürülür. Bu durum Şekil 9.8'de görülmektedir.



Şekil 9.8: Metnin önceki bilgilerle deşifre edilmiş hali

10. SONUÇ

Bu çalışmada kriptografiye kısa bir giriş yapılarak, kısaca terminolojiden ve kriptografi tarihinden bahsedilmiştir, kriptografi için gerekli olabilecek matematiksel alt yapıya değinilmiş, Klasik Kriptografi olarak adlandırılan bazı basit kriptosistemlerden ve bunların bazılarının kriptoanalizinden bahsedilmiştir. İlerleyen bölümde kriptografi çeşitleri iki ana başlık altında kısaca anlatılmış, Simetrik ve Asimetrik Kriptosistemler iki ayrı bölüm altında başlıca incelenmiş ve eliptik eğrilerden bahsedilerek Eliptik Eğri Şifrelemeye değinilmiştir. TripleDES algoritmasına bir uygulama olarak Base64 kodlaması anlatılarak yapılan Simetrik Şifreleme Programı ile örnekler verilmiştir.

Hızla gelişen sayısal teknoloji, iletişim konusunda güvenliğin sağlanması için şifreleme algoritmalarına ihtiyaç duymaktadır. Kriptografi bu ihtiyaçlara cevap verebilecek birçok şifreleme algoritmaları geliştirmektedir.

Günümüzde simetrik şifreleme algoritmaları şifreleme ve deşifreleme işlemlerini daha hızlı gerçekleştirebildiklerinden dolayı daha avantajlı gözükmektedir. Simetrik kriptosistemlerin mutlak biçimde korumak zorunda oldukları anahtarların koruma ve dağıtım maliyetinin ne kadar yüksek ve koruma işleminin ne kadar zor olduğu kolayca görülebilir. Sadece bu nedenden dolayı, karşılıklı haberleşme içinde olan iki tarafın güvenli dağıtım kanalları oluşturması özellikle güncel bankacılık sisteminde, yaygın görülen bir örnektir.

Asimetrik şifreleme algoritmalarında gizli anahtar kullanılması anahtarları koruma ve dağıtım maliyetini ortadan kaldırmaktadır. Bu algoritmalar bu avantajı sağlarken şifreleme ve deşifreleme süreleri çok uzun olduğundan dolayı bazı ortamlarda kullanılmak istenmez. Kriptografi bilimi bu dezavantajı ortadan kaldırmak için yeni asimetrik şifreleme algoritmaları geliştirmektedir. Örneğin, RSA şifreleme algoritması daha fazla güvenlik sağlaması için uzun anahtar değerleri kullanmaktadır. Bu da yeni oluşturulan sistemlerde pek istenmeyen bir durumdur. Bundan dolayı Eliptik Eğri şifreleme algoritması geliştirilmiştir. Eliptik Eğri

Kriptosistem daha düşük anahtar değerleriyle aynı güvenlik seviyesini sağlamıştır. Bu da güncel uygulamalarda çok önemli bir avantajdır.

Asimetrik kriptosistemler açısından böyle bir algoritmanın geliştirilmesi, bu sistem için çok önemlidir ve asimetrik kriptosistemlerin daha çok kullanılmasını sağlayacaktır.

Günümüz ve yakın gelecek için eliptik eğri kriptosistemlere ilişkin projeler üç alanda yoğunlaşmaktadır:[36]

- Kısıtlı kaynaklara sahip taşınabilir araçlardan, yüksek verimlilikteki ağ sunucularına kadar her ortam için, kriptografik çözüm üretiminde kullanılabilecek eliptik eğri kriptografik yazılım geliştirme kütüphanelerinin oluşturulması.

- Kriptografik çözüme gereksinimi olan ortamların tümünde kullanılabilir olan, donanımla tümleştirilmiş eliptik eğri kriptosistem çözümlerinin yaratılması.

- Eliptik eğri kriptosistemlerin, günlük hayatta daha çok yer almasını ve standartlaşmasını sağlayacak çalışmaların yapılması ve bu çalışmalar sonucunda internet üzerinde eliptik eğri kriptosistemler kullanılarak güvenliğin arttırılmasının sağlanması.

Günümüzde önemli sayılabilecek veriler, bilgisayarların yaygınlaşmasıyla birlikte sayısal ortama aktarılmakta ve saklanılan verilere dünyanın neresinde olunursa olunsun erişilebilmektedir. Bu nedenlerden dolayı kişisel kullanıcıların sayısal ortamlarda güvenli bir şekilde kendi verilerine erişebilmelerini sağlayacak çalışmaların yapılması gelecek için yararlı olacaktır.

[36] http://research.sun.com/projects/crypto/

KAYNAKLAR

1. Atay, S., Ege Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Anabilim Dalı Eliptik Eğri Kriptosistem Yazılım Uygulamalarında Hız Problemi Doktora Tezi, 2006
2. Brown, D., *Digital Fortress*, St. Martin's Press, ISBN: 0-312-20715-8, New York, 1998
3. Hankerson, D., Menezes, A., Vanstone, S., *Guide to Elliptic Curve Cryptography*, Springer, ISBN: 0-387-95273-X, New York, 2004
4. http://odevler.gencbilim.com/oddevlerfckgw/gencbilim_matematik_433.zip, (05.04.2007)
5. http://en.wikipedia.org/wiki/Cryptography, (18.01.2007)
6. http://www.bilmuh.gyte.edu.tr/~ispinar/BM550/SECURITYCRS9-13.pdf, (09.02.2004)
7. http://www.certicom.com/index.php?action=ecc_tutorial,ecc_tut_2_0, (31.12.2007)
8. http://www.certicom.com/index.php?action=ecc_tutorial,ecc_tut_2_1, (31.12.2007)
9. http://www.certicom.com/index.php?action=ecc_tutorial,ecc_tut_2_1_2, (31.12.2007)
10. http://www.certicom.com/index.php?action=ecc_tutorial,ecc_tut_2_1_3, (31.12.2007)
11. http://www.certicom.com/index.php?action=ecc_tutorial,ecc_tut_2_1_4, (31.12.2007)
12. http://www.certicom.com/index.php?action=ecc_tutorial,ecc_tut_3_0, (31.12.2007)
13. http://www.certicom.com/index.php?action=ecc_tutorial,ecc_tut_3_1, (31.12.2007)
14. http://www.certicom.com/index.php?action=ecc_tutorial,ecc_tut_3_2, (31.12.2007)
15. http://www.certicom.com/index.php?action=ecc_tutorial,ecc_tut_4_0, (31.12.2007)

16. http://www.certicom.com/index.php?action=ecc_tutorial,ecc_tut_4_2, (31.12.2007)
17. http://www.kamusm.gov.tr/tr/Bilgideposu/Belgeler/teknik/aaa/index.html?kriptolojinedir.html, (12.01.2007)
18. http://www3.iam.metu.edu.tr/iam/images/6/69/Kriptolojiyegiriş-ersanali.pdf, (19.05.2007)
19. Koblitz, N., 1994, *A Course in Number Theory and Cryptography 2nd Edition*, Springer-Verlag, ISBN: 0-387-94293-9, New York
20. Stinson, D. R., 1995, *Theory and Practice*, CRC Press, ISBN: 0-8493-8521-0, New York
21. Trappe, W., 2002, L.C. Washington, *Introduction to Cryptography with Coding Theory*, Prentice Hall, ISBN: 0-13-061814-4, New Jersey

ÖZGEÇMİŞ

Tarık Tuncal, Türkiye Cumhuriyeti vatandaşı olarak 1980 yılında İstanbul'da doğdu. İlk okulu 1991 yılında Koza İlk Okulu'nda bitirdi. Bahçelievler Anadolu Lisesi'nde 1995 yılında Orta Okulu ve 1998 yılında Lise öğrenimini tamamladı. 2003 yılında T.C. Yıldız Teknik Üniversitesi'nde Matematik Bölümü'nde lisans eğitimini bitirdi. 2004 yılında başladığı T.C. Maltepe Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Yüksek Lisans öğrenimine devam etmektedir.

EKLER

EK-A

```
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Data;

using System.Security.Cryptography;
using System.IO;

public class Form1 : System.Windows.Forms.Form
{
   SymmetricAlgorithm alg = new TripleDESCryptoServiceProvider();

   //SymmetricAlgorithm alg = new RijndaelManaged();

   public Form1()
   {
      InitializeComponent();
   }

   static void Main()
   {
      Application.Run(new Form1());
   }

   private void Encrypt_Click(object sender, System.EventArgs e)
   {
      ICryptoTransform enc = alg.CreateEncryptor(
```

```
      Convert.FromBase64String(tbKey.Text),
      Convert.FromBase64String(tbIV.Text)
    );

    System.Text.ASCIIEncoding ascii = new System.Text.ASCIIEncoding();
    byte[] plaintext = ascii.GetBytes(tbPlaintext.Text);

    byte[] cyphertext = PerformCryptoOperation(enc, plaintext);

    tbCyphertext.Text = Convert.ToBase64String(cyphertext);
}

private void Decrypt_Click(object sender, System.EventArgs e)
{
    ICryptoTransform dec = alg.CreateDecryptor(
Convert.FromBase64String(tbKey.Text),
      Convert.FromBase64String(tbIV.Text)
    );

    byte[] cyphertext = Convert.FromBase64String(tbCyphertext.Text);

    byte[] plaintext = PerformCryptoOperation(dec, cyphertext);

    System.Text.ASCIIEncoding ascii = new System.Text.ASCIIEncoding();
    tbPlaintext.Text = ascii.GetString(plaintext);

}

private byte[] PerformCryptoOperation(ICryptoTransform op,
  byte[] input)
{
    MemoryStream msOutput = new MemoryStream();
```

```
    CryptoStream encStream =
      new CryptoStream(msOutput, op, CryptoStreamMode.Write);

    encStream.Write(input, 0, input.Length);

    encStream.Close();

    return msOutput.ToArray();

}

private void RandomKey_Click(object sender, System.EventArgs e)
{
    RNGCryptoServiceProvider rng = new RNGCryptoServiceProvider();

    byte[] keybytes = new byte[alg.KeySize / 8];
    rng.GetBytes(keybytes);

    tbKey.Text = Convert.ToBase64String(keybytes);
}

private void RandomIV_Click(object sender, System.EventArgs e)
{
    RNGCryptoServiceProvider rng = new RNGCryptoServiceProvider();

    byte[] ivbytes = new byte[alg.BlockSize / 8];
    rng.GetBytes(ivbytes);

    tbIV.Text = Convert.ToBase64String(ivbytes);

}
```

```
private System.Windows.Forms.TextBox tbKey;
private System.Windows.Forms.Button RandomKey;
private System.Windows.Forms.Label label1;
private System.Windows.Forms.Label label2;
private System.Windows.Forms.TextBox tbPlaintext;
private System.Windows.Forms.Label label3;
private System.Windows.Forms.TextBox tbCyphertext;
private System.Windows.Forms.Button Encrypt;
private System.Windows.Forms.Button Decrypt;
private System.Windows.Forms.Label label4;
private System.Windows.Forms.Button RandomIV;
private System.Windows.Forms.TextBox tbIV;
private System.ComponentModel.Container components = null;

private void InitializeComponent()
{
   this.tbKey = new System.Windows.Forms.TextBox();
   this.RandomKey = new System.Windows.Forms.Button();
   this.label1 = new System.Windows.Forms.Label();
   this.label2 = new System.Windows.Forms.Label();
   this.tbPlaintext = new System.Windows.Forms.TextBox();
   this.label3 = new System.Windows.Forms.Label();
   this.tbCyphertext = new System.Windows.Forms.TextBox();
   this.Encrypt = new System.Windows.Forms.Button();
   this.Decrypt = new System.Windows.Forms.Button();
   this.label4 = new System.Windows.Forms.Label();
   this.RandomIV = new System.Windows.Forms.Button();
   this.tbIV = new System.Windows.Forms.TextBox();
   this.SuspendLayout();
   //
   // tbKey
```

```
//
this.tbKey.Location = new System.Drawing.Point(161, 8);
this.tbKey.Multiline = true;
this.tbKey.Name = "tbKey";
this.tbKey.Size = new System.Drawing.Size(419, 48);
this.tbKey.TabIndex = 0;
this.tbKey.Text = "base64-kodlu Anahtar";
//
// RandomKey
//
this.RandomKey.Location = new System.Drawing.Point(304, 64);
this.RandomKey.Name = "RandomKey";
this.RandomKey.Size = new System.Drawing.Size(150, 23);
this.RandomKey.TabIndex = 1;
this.RandomKey.Text = "Rastgele Anahatar Üret";
this.RandomKey.Click += new System.EventHandler(this.RandomKey_Click);
//
// label1
//
this.label1.Location = new System.Drawing.Point(83, 6);
this.label1.Name = "label1";
this.label1.Size = new System.Drawing.Size(72, 23);
this.label1.TabIndex = 2;
this.label1.Text = "Anahtar";
this.label1.TextAlign = System.Drawing.ContentAlignment.MiddleRight;
//
// label2
//
this.label2.Location = new System.Drawing.Point(33, 190);
this.label2.Name = "label2";
this.label2.Size = new System.Drawing.Size(122, 23);
this.label2.TabIndex = 4;
```

```
        this.label2.Text = "Şifresiz Yazı:";
        this.label2.TextAlign = System.Drawing.ContentAlignment.MiddleRight;
        //
        // tbPlaintext
        //
        this.tbPlaintext.Location = new System.Drawing.Point(161, 192);
        this.tbPlaintext.Multiline = true;
        this.tbPlaintext.Name = "tbPlaintext";
        this.tbPlaintext.ScrollBars = System.Windows.Forms.ScrollBars.Vertical;
        this.tbPlaintext.Size = new System.Drawing.Size(419, 120);
        this.tbPlaintext.TabIndex = 3;
        this.tbPlaintext.Text = "Düzmetin";
        //
        // label3
        //
        this.label3.Location = new System.Drawing.Point(12, 360);
        this.label3.Name = "label3";
        this.label3.Size = new System.Drawing.Size(143, 23);
        this.label3.TabIndex = 6;
        this.label3.Text = "Şifrelenmiş Yazı:";
        this.label3.TextAlign = System.Drawing.ContentAlignment.MiddleRight;
        //
        // tbCyphertext
        //
        this.tbCyphertext.BackColor =
System.Drawing.Color.FromArgb(((int)(((byte)(255)))), ((int)(((byte)(255)))),
((int)(((byte)(192)))));
        this.tbCyphertext.ForeColor = System.Drawing.Color.DimGray;
        this.tbCyphertext.Location = new System.Drawing.Point(161, 360);
        this.tbCyphertext.Multiline = true;
        this.tbCyphertext.Name = "tbCyphertext";
        this.tbCyphertext.ScrollBars = System.Windows.Forms.ScrollBars.Vertical;
```

```
this.tbCyphertext.Size = new System.Drawing.Size(419, 120);
this.tbCyphertext.TabIndex = 5;
this.tbCyphertext.Text = "base64-kodlu Şifreli metin";
//
// Encrypt
//
this.Encrypt.Location = new System.Drawing.Point(278, 328);
this.Encrypt.Name = "Encrypt";
this.Encrypt.Size = new System.Drawing.Size(121, 23);
this.Encrypt.TabIndex = 7;
this.Encrypt.Text = "Şifrele";
this.Encrypt.Click += new System.EventHandler(this.Encrypt_Click);
//
// Decrypt
//
this.Decrypt.Location = new System.Drawing.Point(405, 328);
this.Decrypt.Name = "Decrypt";
this.Decrypt.Size = new System.Drawing.Size(112, 23);
this.Decrypt.TabIndex = 8;
this.Decrypt.Text = "Şifreyi Çöz";
this.Decrypt.Click += new System.EventHandler(this.Decrypt_Click);
//
// label4
//
this.label4.Location = new System.Drawing.Point(37, 93);
this.label4.Name = "label4";
this.label4.Size = new System.Drawing.Size(118, 25);
this.label4.TabIndex = 11;
this.label4.Text = "IV (Başlangıç Vektörü):";
this.label4.TextAlign = System.Drawing.ContentAlignment.MiddleRight;
//
// RandomIV
```

```
//
this.RandomIV.Location = new System.Drawing.Point(312, 160);
this.RandomIV.Name = "RandomIV";
this.RandomIV.Size = new System.Drawing.Size(142, 23);
this.RandomIV.TabIndex = 10;
this.RandomIV.Text = "Rastgele IV Üret";
this.RandomIV.Click += new System.EventHandler(this.RandomIV_Click);
//
// tbIV
//
this.tbIV.Location = new System.Drawing.Point(161, 96);
this.tbIV.Multiline = true;
this.tbIV.Name = "tbIV";
this.tbIV.Size = new System.Drawing.Size(419, 56);
this.tbIV.TabIndex = 9;
this.tbIV.Text = "base64-kodlu başlangıç vektörü";
//
// Form1
//
this.AutoScaleBaseSize = new System.Drawing.Size(5, 13);
this.ClientSize = new System.Drawing.Size(603, 489);
this.Controls.Add(this.label4);
this.Controls.Add(this.RandomIV);
this.Controls.Add(this.tbIV);
this.Controls.Add(this.Decrypt);
this.Controls.Add(this.Encrypt);
this.Controls.Add(this.label3);
this.Controls.Add(this.tbCyphertext);
this.Controls.Add(this.label2);
this.Controls.Add(this.tbPlaintext);
this.Controls.Add(this.label1);
this.Controls.Add(this.RandomKey);
```

```
        this.Controls.Add(this.tbKey);
        this.Name = "Form1";
        this.StartPosition = System.Windows.Forms.FormStartPosition.CenterScreen;
        this.Text = "Simetrik Şifreleme Programı";
        this.ResumeLayout(false);
        this.PerformLayout();

    }
}
```