T.C.

BİLECİK ŞEYH EDEBALİ ÜNİVERSİTESİ

LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

MATEMATİK ANABİLİM DALI

MODERN VE KUANTUM ŞİFRELEME ALGORİTMALARI

YÜKSEK LİSANS TEZİ

MURAT BAYRAM

TEZ DANIŞMANI

DOÇ. DR. ELİF ILGAZ ÇAĞLAYAN

BİLECİK, 2025

10593327

T.C.

BİLECİK ŞEYH EDEBALİ ÜNİVERSİTESİ

LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

MATEMATİK ANABİLİM DALI

MODERN VE KUANTUM ŞİFRELEME ALGORİTMALARI

YÜKSEK LİSANS TEZİ

MURAT BAYRAM

TEZ DANIŞMANI

DOÇ. DR. ELİF ILGAZ ÇAĞLAYAN

BİLECİK, 2025

10593327

BEYAN

MODERN VE KUANTUM ŞİFRELEME ALGORİTMALARI adlı yüksek lisans yeterlik tezi projesinin hazırlık ve yazımı sırasında bilimsel araştırma ve etik kurallarına uyduğumu, başkalarının eserlerinden yararlandığım bölümlerde bilimsel kurallara uygun olarak atıfta bulunduğumu, kullandığım verilerde herhangi bir tahrifat yapmadığımı, tezin herhangi bir kısmının Bilecik Şeyh Edebali Üniversitesi veya başka bir üniversitede başka bir tez çalışması olarak sunulmadığını, aksinin tespit edileceği muhtemel durumlarda doğabilecek her türlü hukuki sorumluluğu kabul ettiğimi ve vermiş olduğum bilgilerin doğru olduğunu beyan ederim.

<table>
<tr><td colspan="4">Bu çalışmanın,
Bilimsel Araştırma Projeleri (BAP), TÜBİTAK veya benzeri kuruluşlarca desteklenmesi durumunda; projenin ve destekleyen kurumun adı proje numarası ile birlikte, ETİK KURUL onayı alınması durumunda ise ETİK KURUL tarih karar ve sayı bilgilerinin beyan edilmesi gerekmektedir.</td></tr>
<tr><td>DESTEK ALINMIŞTIR</td><td>X</td><td>DESTEK ALINMAMIŞTIR</td><td></td></tr>
<tr><td colspan="4">Destek alındı ise;</td></tr>
<tr><td colspan="4">Destekleyen kurum; TÜBİTAK</td></tr>
<tr><td colspan="2">Desteğin Türü</td><td colspan="2">Proje Numarası</td></tr>
<tr><td colspan="2">1- BAP (Bilimsel Araştırma Projesi)</td><td colspan="2" rowspan="3">2210-A Yurt İçi Yüksek Lisans Burs Programı</td></tr>
<tr><td colspan="2">2- TÜBİTAK</td></tr>
<tr><td colspan="2">Diğer;……</td></tr>
<tr><td colspan="4">ETİK KURUL onayı var ise;</td></tr>
<tr><td colspan="2">ETİK KURUL karar tarih/sayı:</td><td colspan="2">……/……..</td></tr>
</table>

Murat Bayram

../../2025

İmza

ÖN SÖZ

Bu tezin ortaya çıkış sürecinde bilimsel rehberliği, sabrı ve yapıcı eleştirileriyle heraşamada yanımda olan değerli danışmanım Doç. Dr. Elif Ilgaz Çağlayan'a en içten teşekkürlerimi sunarım. Sevgili dostum Naz Bilgi Ekiz'e hesaplamalı süreçlerde verdiği destekiçin minnettarım. Makalem için tatilinin ortasında yardımıma koşan Prof. Dr. İlker İnam'a ve Kurumsal süreçlerde kolaylaştırıcı yaklaşımları için Lisansüstü Eğitim Enstitüsü'ne teşekkür ederim.

Bu çalışma, TÜBİTAK - BİDEB 2210-A Yurt İçi Yüksek Lisans Burs Programı tarafından desteklenmiştir. Sağladıkları maddi destek ve araştırma motivasyonu için TÜBİTAK'a teşekkür ederim.

Son olarak, her koşulda yanımda olan başta kıymetli annem Sultan Bayram olmak üzere aileme ve dostlarıma moral ve sabırları için teşekkür ederim. Onların desteği olmasaydı bu çalışma tamamlanamazdı.

MURAT BAYRAM

2025

ÖZET

MODERN VE KUANTUM ŞİFRELEME ALGORİTMALARI

Bu tez, modern ve kuantum şifreleme yaklaşımlarını matematiksel temeller üzerinden ortak bir çerçevede birleştirerek, iki alanın güvenlik anlayışlarını karşılaştırmalı biçimde tartışmaktadır.

Klasik bölümde, Sezar ve yerine koyma gibi tarihsel şemalar yalnızca giriş örnekleri olarak değil; modüler aritmetik ve grup eylemleri bağlamında incelenmiştir. Bu temel yaklaşımlardan hareketle Feistel mimarisi, sözde-rastgele permütasyon ve işlev kavramları ile indirgeme temelli ispat mantığı ele alınmış; ayrıca doğrusal olmayanlık, difüzyon ve karışıklık gibi tasarım ilkelerinin ölçülebilirliği tartışılmıştır. Veri Şifreleme Standardı özelinde S-kutuları cebirsel normal form ve diferansiyel olasılık gibi nicel ölçütlerle değerlendirilmiştir.

Kuantum bölümünde, güvenliğin fiziksel yasalara dayalı çerçevesi sunulmuştur. BB84 ve E91 protokolleri yan yana ele alınmış; BB84'te taban seçimi ve ölçüm istatistikleri, E91'de ise Bell türü korelasyonlar güvenlik temeli olarak değerlendirilmiştir. Doğruluk ve gizlilik hatalarının birlikte küçük tutulması gerektiği vurgulanmış; olası sızma senaryoları, gözlenen hata oranları ve E91 için uygun Bell testleri üzerinden üstten sınırlandırılmıştır. Sonlu örneklem koşullarında ise düzgünleştirilmiş eşitsizlikler kullanılarak elde edilebilecek gizli anahtar uzunluğunun nasıl kısaltılacağı belirlenmiştir.

Uygulamalı kısımda Python tabanlı bir BB84 simülasyonu geliştirilmiştir. Model; kanal kaybı, depolarizasyon, dedektör verimi, karanlık sayım, baz uyumsuzluğu ve kes-yeniden gönderme gibi dinleyici saldırılarını kapsamaktadır. Parametre tahmini, hata düzeltme sızıntısı ve gizlilik yükseltme adımlarını izleyen uçtan uca iş akışı sayesinde süzülmüş anahtar oranı ve etkin sızıntı büyüklükleri hesaplanmıştır. Yapılan hassasiyet analizleri, hata oranındaki küçük artışların anahtar üretimini orantısız şekilde azalttığını ve güvenli çalışma bölgesinin mesafe–gürültü–dedektör parametreleri arasında hızla daralan bir "işletim penceresi" oluşturduğunu göstermiştir.

Sonuçlar, düşük gürültü ve orta mesafe koşullarında pozitif anahtar üretiminin sürdürülebilir olduğunu; gürültü veya mesafenin artması halinde güvenliğin hızlı bir biçimde zayıfladığını ortaya koymaktadır.

Anahtar Kelimeler: DES, Kuantum Kriptografi, QKD, BB84 Protokolü, E91 Protokolü.

ABSTRACT

MODERN AND QUANTUM ENCRYPTION ALGORITHMS

This thesis integrates modern and quantum cryptographic approaches within a common mathematical framework and comparatively examines the security paradigms of both fields.

In the classical part, historical schemes such as Caesar and substitution ciphers are analyzed not merely as introductory examples but within the context of modular arithmetic and group actions. Building upon these foundations, the Feistel architecture, the notions of pseudorandom permutations and functions, and reduction-based proof techniques are discussed. Furthermore, the measurability of design principles such as nonlinearity, diffusion, and confusion is addressed. Specifically, in the case of the Data Encryption Standard, S-boxes are evaluated through quantitative measures such as algebraic normal form and differential probability.

In the quantum part, the security framework based on physical laws is presented. The BB84 and E91 protocols are examined side by side: BB84 relying on basis choice and measurement statistics, and E91 on Bell-type correlations. The necessity of simultaneously minimizing correctness and secrecy errors is emphasized, while potential eavesdropping scenarios are bounded using observed error rates and suitable Bell tests for E91. Under finite-size sampling conditions, smooth inequalities are employed to determine the extent to which the length of the secret key must be reduced.

In the applied section, a Python-based simulation of BB84 communication is developed. The model accounts for channel loss, depolarization, detector efficiency, dark counts, basis mismatch, and intercept–resend eavesdropping strategies. Following an end-to-end workflow of parameter estimation, error correction leakage, and privacy amplification, both the sifted key rate and effective leakage sizes are obtained. Sensitivity analyses reveal that even small increases in error rate disproportionately reduce key generation, while the secure operating region rapidly contracts into a narrow “operational window” defined by distance, noise, and detector parameters.

The results demonstrate that positive key generation remains sustainable under low-noise and medium-distance conditions, whereas increasing either noise or distance quickly undermines security.

Keywords: DES, Quantum Cryptography, QKD, BB84 Protocol, E91 Protocol.

İÇİNDEKİLER

Sayfa

TABLOLAR LİSTESİ

Sayfa

ŞEKİLLER LİSTESİ

Sayfa

KISALTMALAR VE SİMGELER LİSTESİ

OTP : One-Time Pad

AES : Advanced Encryption Standard

DES : Data Encryption Standard

QBER : Quantum Bit Error Rate

OTDR : Optical Time-Domain Reflectometry

LDPC : Low-Density Parity-Check

XOR : Exclusive OR

CHSH : Clauser - Horne - Shimony - Holt eşitsizliği

DDT : Difference Distribution Table

ort ± std : Ortalama ± Standart Sapma

leak$_{ec}$: Hata düzeltme sızıntısı

MDI-QKD : Measurement-Device-Independent Quantum Key Distribution

GİRİŞ

Çağımızda internet ve benzeri iletişim sistemlerinin, günlük hayatımızın önemli bir parçası haline gelmesi beraberinde bu sistemlerin güvenliğinin sağlanması gerekliliğini ortaya koymaktadır. Kişisel bilgilerin korunması, finansal işlemlerin güvenliği hatta devlet sırlarının korunması şifreleme algoritmaları olmadan mümkün olmaz.

Şifreleme teknolojileri hem modern hem de kuantum algoritmaları üzerine inşa edilmektedir. Modern şifreleme algoritmaları, veriyi karmaşık matematiksel işlemler kullanarak şifrelemekte ve çözmektedir, bu sayede verilerin güvenliği sağlanmaktadır. Diğer taraftan, kuantum mekaniği temelli şifreleme algoritmaları kuantum mekaniği prensipleriyle veriyi şifrelemekte ve çözmektedir. Bu prensipler neticesinde kuantum bilgisayarlar belirli matematiksel işlemleri geleneksel bilgisayarlardan çok daha hızlı yapabilmektelerdir. Bu durum, günümüzde kullanılan birçok şifreleme algoritmasının kuantum bilgisayarlarca kırılabileceği anlamına gelmektedir.

Bu tez, modern şifreleme algoritmalarının temellerini ve çalışma prensiplerini açıklamakla birlikte kuantum şifreleme algoritmalarının ne olduklarını, nasıl çalıştıklarını, avantajlarını ve dezavantajlarını matematiksel olarak ele alacak, sözde kodlar verilip, uygulamalar yapılacaktır. Tezde, kuantum mekaniğinin temel prensiplerine değinilecek ve kuantum şifreleme algoritmalarının bu prensiplerle geçerli olduğu gösterilecektir.

Tezde kullanılmış sözde kodların python tabanlı açık kaynak kodları github[1] linkinde mevcuttur.

[1] https://github.com/muratbayram1337

1. KLASİK ŞİFRELEME

Klasik şifreleme, tarihsel olarak gizli mesajlaşma için kullanılan şifreleme yöntemlerinin genel bir adıdır. Bu yöntemler, tarih boyunca birçok farklı kültürde kullanılmıştır.

Örneğin, eski Yunanlılar ve Romalılar, Sezar şifresi adı verilen basit bir yer değiştirme yöntemini kullanmışlardı (Kahn D., 1996). Bu yöntemde, mesajdaki her harf belirli bir sayı kadar kaydırılır ve böylelikle mesajın anlamı gizlenir. Orta Çağ'da ise, Vigenère şifresi adı verilen daha karmaşık bir şifreleme yöntemi popüler hale geldi. Bu şifrelemede ise, bir anahtar kelime kullanılarak metnin her harfi için farklı bir yer değiştirme şifresi oluşturulur. Klasik şifreleme yöntemleri, günümüz modern kriptografi teknikleriyle karşılaştırıldığında güvensizdir. Modern kriptografi yöntemleri, özellikle de simetrik ve asimetrik şifreleme yöntemleri, kompleks matematiksel algoritmalar kullanarak şifreleme yapar. Bu matematiksel algoritmalar, şifreleme ve deşifreleme işlemleri sırasında anahtar kullanarak verileri şifreler ve çözerler (Katz, 2020).

Klasik şifreleme yöntemleri, tarihte önemli bir yere sahip olmalarına rağmen, günümüzde artık güvensiz kabul edilmektedir, bunun nedeni, çok basit matematiksel algoritmalar kullanarak çalışmaları ve günümüz bilgi işlem teknolojileriyle kolayca çözülebilir olmalarıdır.

1.1. Klasik Şifreleme Algoritmaları

Kriptografinin temel amacı, iki kişinin güvensiz bir iletişim kanalı üzerinden iletişim kurmasını ve bu iletişimi dinleyen saldırganın iletilen mesajı anlayamamasını sağlamaktır (Katz, 2020). Bu iletişim kanalı telefon hattı, bilgisayar ağı veya benzeri olabilir. Gönderilmek istenen orijinal mesaj veya veri, açık metin olarak adlandırılır ve metin, sayılar veya diğer veri türlerinde olabilir. Bu fikirler matematiksel gösterim kullanılarak şöyle açıklanmaktadır:

Şifreleme sistemi: Sonlu bir alfabe A, verilsin. Açık metin uzayı $P \subseteq A^*$ ve şifreli metin uzayı $C \subseteq A^*$ sonlu kümelerdir. Anahtar uzayları iki parçadan oluşur: şifreleme anahtarları K_{enc} ve şifre çözme anahtarları K_{dec}. Bu iki uzay arasında birebir ve örten bir eşleme

$$\pi: K_{enc} \rightarrow K_{dec} \tag{1.1}$$

Tanımlıdır; $\pi(e)$, e anahtarının benzersiz çözme karşılığıdır (Goldreich, 2004).

Her $e \in K_{enc}$ için bir şifreleme dönüşümü

$$E_e : P \rightarrow C \tag{1.2}$$

ve her $d \in K_{dec}$ için bir şifre çözme dönüşümü

$$D_d : C \rightarrow P \tag{1.3}$$

Tanımlıdır. Doğruluk, uygun anahtar eşleşmesi altında şifre çözmenin, şifrelemenin tersi olmasını gerektirir (Boneh, 2020):

$$\forall e \in K_{enc}: \quad D_{\pi(e)} = E_e^{-1} \tag{1.4}$$

Bunun eşdeğer ifadeleri:

$$\forall p \in P: D_{\pi(e)}\left(E_e(p)\right) = p \tag{1.5}$$

ve

$$\forall c \in C: E_e\left(D_{\pi(e)}(c)\right) = c \tag{1.6}$$

Bu koşul nedeniyle her E_e bir bijeksiyondur ve dolayısıyla $|P| = |C|$.
Her $e \in K_{enc}$ için $(e, \pi(e))$ düzenli çifti bir anahtar çiftidir. Simetrik anahtarlı sistemlerde $K_{enc} = K_{dec}$ ve $\pi = id$ olup tipik olarak $e = \pi(e)$ (yani $e = d$) seçilebilir. Açık anahtarlı sistemlerde ise genel olarak $K_{enc} \neq K_{dec}$ ve $e \neq \pi(e)$ olabilir.
Şifreleme şeması, dönüşüm aileleri ile birlikte

$$E = \{E_e : e \in K_{enc}\} \tag{1.7}$$

$$D = \{D_d : d \in K_{dec}\} \tag{1.8}$$

ve eşleme π tarafından belirlenir. Bir şifreleme sistemi aşağıdaki sekizli ile özetlenir (Boneh, 2020):

$$\Sigma = (A, P, C, K_{enc}, K_{dec}, E, D, \pi) \tag{1.9}$$

1.1.1. Sezar Şifrelemesi

Sezar şifrelemesi, adını Roma İmparatoru Jül Sezar'dan alan ve tarih boyunca mektupların gizlenmesinde kullanılan en eski yöntemlerden biridir. Temel fikir, alfabetik bir dizin üzerinde her harfi sabit bir anahtar kadar boyunda kaydırmaktır; böylece düz metin okunamaz hale gelir. Şifrelemek için seçilen anahtar kadar ileri, çözmek için ise aynı miktarda geri kaydırma yapılır. Anahtarın 2 kabul edildiği senaryoda; yani her harf alfabede iki basamak sonraki karşılığıyla değiştirilir. Örneğin "abc" dizisi "cde" olur, "z" harfi başa sararak "b"ye dönüşür. Anahtar uzayı çok küçük olduğu ve harf sıklıkları değişmediği için

kaba kuvvet ve frekans analiziyle kolayca kırılabilir.

Bu yöntemin Python dilindeki genel kodlama yapısı aşağıdaki şekillerde gösterildiği gibidir:

ALGORİTMA 1: SEZAR ŞİFRELEMESİ

```
    Girdi: mesaj (dizge), anahtar (tamsayı)
    Çıktı: sifrelenecek_mesaj (dizge)
1   sifrelenecek_mesaj ← ""
2   alfabe ← ['a','b','c','ç','d','e','f','g','ğ','h','ı','i','j','k','l','m','n','o','ö','p','r','s','ş','t','u','ü','v','y','z']
3   n ← |alfabe|
4   mesaj_kucuk ← Lower(mesaj)
5   foreach karakter ∈ mesaj_kucuk do
6       if karakter ∉ alfabe then
7           sifrelenecek_mesaj ← sifrelenecek_mesaj ‖ karakter
8   else
9           ind ← IndexOf(alfabe, karakter)
10  else
11          yeni_ind ← (ind + anahtar) mod n
12          sifrelenecek_mesaj ← sifrelenecek_mesaj ‖ alfabe[yeni_ind]
13  return sifrelenecek_mesaj
```

ALGORİTMA 2: SEZAR ŞİFRELEME ALGORİTMASI ANA AKIŞI

```
1   geçerli ← yanlış
2   while geçerli do
3     girdi ← Input (“Kaydırma değeri: “)
4       if girdi tamsayıya dönüştürülebilir then
5       anahtar ← Int(girdi)
6       geçerli ← doğru
7   else
8     print (“Kaydırma değeri tamsayı olmalıdır.”)
9       metin ← input(“şifrelenecek mesaj:”)
10  Şifrelenmiş ← sezar(metin, anahtar)
11  Print(“Şifrelenmiş mesaj: “ || sifrelenmiş)
```

Verilen kod bloğunda öncelikle, Sezar şifrelemesi algoritmasını uygulayacak fonksiyon tanımlanmıştır. Bu fonksiyonun ilk parametresi, şifrelemek istediğimiz mesajı temsil ederken, ikinci parametre ise kaydırma değerini ifade etmektedir. Şifrelemeyi gerçekleştirebilmek için kullanılacak alfabeye de bu kod içerisinde yer verilmiştir. Alfabede, Türkçe harfler de dâhil olmak üzere toplam 29 harf bulunmaktadır. Daha sonra, kullanıcı tarafından girilecek olan mesaj, for döngüsü ile eleman eleman gezilerek şifrelenmektedir. Şifreleme (kaydırma) 10. satırdaki adımda gerçekleşir. Ardından şifreli karakterin metne yazılması 11. satır ile gerçekleşir.

Daha sonra, bulunan indeksin alfabe listesi boyutuna göre mod alınması gerekmektedir. Bu işlem, kaydırma değerinin alfabe boyutundan büyük olması durumunda, listenin başına geri dönülmesini sağlamaktadır. Program, son olarak kullanıcının girdiği mesajın şifrelenmiş halini ekrana yansıtmaktadır.

Sezar şifrelemesi, kolay uygulanabilmesi ve anlaşılması nedeniyle tarih boyunca yaygın bir şekilde kullanılmıştır. Ancak, modern kriptografi teknikleriyle karşılaştırıldığında, basitçe kırılabilir bir yöntemdir. Şifreleme yaparken kullanılan kaydırma sayısı, deneme yanılma yoluyla bulunabilir ve bu sayede şifreli metin açık metne dönüştürülebilir (Stinson D. R., 2018).

1.1.2. Yerine Koyma Şifrelemesi

Bir alfabedeki her harfin, o alfabede bulunan farklı bir harfle değiştirildiği bir şifreleme yöntemidir. Bu yöntemde, bir kılavuz kullanılarak şifreleme yapılır ve aynı kılavuza sahip olan kişi şifrelenmiş mesajı çözebilir. Bu şifreleme yöntemi, basit ve anlaşılır olması nedeniyle tarih boyunca sıklıkla kullanılmıştır.

Açık Metin:

$$P=C=Z_{29} \tag{1.10}$$

Burada Z_{29}, 29 harflik alfabe üzerinde mod 29 işlemi ile temsil edilen tam sayıların kümesini temsil eder.

Şifreleme Anahtarı:

$$K=\{\pi|0,1,\ldots,28\} \tag{1.11}$$

Burada π 0'dan 28'e kadar olan sayıların permütasyonlarını temsil eder. Yani K, 29 farklı sembolün tüm olası permütasyonlarının kümesini temsil eder.

Şifreleme Fonksiyonu:

$$e_\pi(x)=\pi(x) \tag{1.12}$$

Burada $e_\pi(x)$, açık metin harfi x'i şifrelemek için kullanılan permütasyon fonksiyonunu temsil eder.

Çözme Fonksiyonu:

$$d_\pi(y)=\pi^{-1}(y) \tag{1.13}$$

Burada $d_\pi(y)$, şifreli metin harfi y'i çözmek için kullanılan ters permütasyon fonksiyonunu temsil eder. π^{-1} ise π'nin ters permütasyonunu temsil eder.

Yukarıdaki tanım, yerine koyma şifrelemesini matematiksel olarak tanımlamaktadır. P ve C sembollerinin tam sayılar kümesi Z_{29} üzerinde yerine koyma işlemi yapılır. Şifreleme anahtarı K, 0'dan 28'e kadar olan sayıların olası permütasyonlarını içerir. Şifreleme fonksiyonu $e_\pi\ (x)$, açık metin harfini permütasyon fonksiyonu ile şifreler ve çözme fonksiyonu $d_\pi\ (y)$, şifreli metin harfini ters permütasyon fonksiyonu ile çözer.

Şifreleme işlemini oluşturacak yerine koyma tablosu aşağıda verilmiştir. π şifreleme işlemini içeren bir fonksiyon olmak üzere, $e_\pi\ (a) = G$, $e_\pi\ (b)$ = Ğ, ..., $e_\pi\ (z) = A$ şeklinde tüm alfabe şifrelenir.

Tablo 1.1 Şifreleme Tablosu

a-G	b-Ğ	c-H	ç-İ	d-J	e-K
g-L	ğ-Ş	h-N	ı-O	i-P	j-R
l-Z	m-Ş	n-T	o-U	ö-Ü	p-V
r-M	s-Ş	t-C	u-D	ü-E	v-F
y-F	z-A				

Şifre çözme işlemi ters permütasyonla gerçekleştirilir. Bunun için ikinci satırlar önce yazılır ve ardından alfabetik sıraya göre sıralanır, böylelikle aşağıdaki sonuç elde edilir:

Tablo 1.2 Şifre Çözme Tablosu

A-z	B-ş	C-t	D-u	E-ü	F-y
G-a	Ğ-b	H-c	İ-ç	J-d	K-e
L-g	M-r	N-h	O-ı	P-i	R-j
S-ğ	Ş-m	T-n	U-o	Ü-ö	V-p
Y-s	Z-l				

Sonuç olarak, $d_\pi(A) = z$, $d_\pi(B) =$, ..., $d_\pi(Z) = s$ şeklinde şifreli metin ters permütasyon fonksiyonu ile çözülür (Menezes, vd., 1996).

Ayrıca, bu yerine koyma tablosunu anahtar kabul eden Python dilindeki genel kodlama yapısının sözde kodu şu şekildedir:

ALGORİTMA 3: YER DEĞİŞTİRME ŞİFRELEMESİ ALGORİTMASI

Girdi: metin (dizge)

Çıktı: sifreliMetin (dizge)

```
anahtar ← {'a': 'G', 'b': 'Ğ', 'c': 'H', 'ç': 'I', 'd': 'İ', 'e': 'J', 'f': 'K', 'g': 'L', 'ğ': 'S',
'h': 'N', 'ı': 'O', 'i': 'Ö', 'j': 'P', 'k': 'R', 'l': 'Z', 'm': 'Ş', 'n': 'T', 'o': 'U', 'ö': 'Ü',
'p': 'V', 'r': 'Y', 's': 'M', 'ş': 'B', 't': 'C', 'u': 'Ç', 'ü': 'D', 'v': 'E', 'y': 'F', 'z': 'A'}
sifreliMetin ← ""
foreach harf ∈ metin do
    if harf ∈ Keys(anahtar) then
        sifreliMetin ← sifreliMetin ‖ anahtar[harf]
else
    sifreliMetin ← sifreliMetin ‖ harf
return sifreliMetin
```

ALGORİTMA 4: YER DEĞİŞTİRME ŞİFRELEMESİ ALGORİTMASI ANA AKIŞI

```
metin ← Input("Şifrelenecek metni girin:")
sifreli ← yerDegistirme(metin)
Print("Şifreli metin: " || sifreli)
```

Ancak, modern kriptografi sistemleri bu yöntemin güvenliği açısından yetersiz olduğunu göstermiştir ve yerine daha karmaşık yöntemler kullanılmaktadır.

1.1.3. Doğrusal Şifreleme

Doğrusal (Affine) Şifreleme, yerine koyma şifresinin bir diğer örneği olsa da biraz farklılık göstermektedir. Bu şifreleme yönteminde, şifreleme süreci büyük ölçüde matematikseldir. Şifreleme işlemi, kullanılan alfabenin uzunluğuna göre mod m (kullanılan alfabenin uzunluğu) işlemi ile gerçekleştirilir. Şifreleme süreci, alfabedeki her karakterin 0 ila (m-1) aralığında bir tam sayıya dönüştürülmesiyle başlar. Bu işlem, şifrelemenin daha karmaşık hale gelmesini sağlar ve şifrelenmiş metnin çözülmesini zorlaştırır.

Tablo 1.3 Alfabe-Sayı Eşlemesi (Türk alfabesi, $m = 29$)

a-0	b-1	c-2	ç-3	d-4	e-5	f-6
g-7	ğ-8	h-9	ı-10	i-11	j-12	k-13
l-14	m-15	n-16	o-17	ö-18	p-19	r-20
s-21	ş-22	t-23	u-24	ü-25	v-26	y-27
z-28						

Bu şifreleme yönteminde a ve b, şifreleme için kullanılacak anahtar değerleridir. m, kullanılan alfabe uzunluğunu ifade eder ve Türk alfabesi kullanıldığında m = 29 olmalıdır. x, şifrelenecek harfin alfabe tablosundaki tam sayı değeridir. E(x) ise şifreleme sonrasında harfin aldığı yeni değerdir ve şifrelenmiş hali olarak kullanılır.

$$E(x)=(ax+b) \bmod m \tag{1.14}$$

Türk alfabesi üzerinde Şifre kelimesini (3,2) anahtarını kullanarak şifreleyelim:

Tablo 1.4 "Şifre" Kelimesinin "*a, b*" = (3, 2) Anahtarıyla Şifrelenmesi

Şifrelenecek harf	ş	i	f	r	e
x	22	11	6	20	5
$3x + 2$	68	35	20	62	17
$(3x + 2) \bmod 29$	10	6	20	4	17
Şifrelenmiş harfler	ı	f	r	d	o

Şifrelenmiş metnin şifresini çözmek için, şifreleme sürecinde kullandığımız fonksiyonun tersini uygulamamız gerekiyor. Bu örnekte, şifreleme fonksiyonu E (x) = (ax + b) mod m şeklindeydi. Bunun tersi,

D(x)=c(y-b) mod m dir. (1.15)

Şifre çözme işlemi için kullanılan yöntem, şifreleme işlemine benzerdir. Ancak, ters işlemi gerçekleştirmemiz gerekmektedir. Şifreleme anahtarları olan a ve b yerine, çözme anahtarları olan c ve d kullanılır. c, a'nın modüler çarpımsal tersidir, yani a.c = 1 mod m. y, şifrelenmiş harfin tabloda aldığı değerdir. D(x) ise şifresi çözülmüş harfin tam sayı karşılığıdır.

Şifreleme sonunda elde ettiğimiz şifreli metin "ıfrdo" olarak verilmiştir. Şifrenin çözülmesi için ilk adım, her harfin sayısal karşılığını bulmaktır. Daha sonra, şifrelemede kullanılan anahtarların tersi olan c sayısı hesaplanmalıdır. Bu hesaplama için a.c = 1 mod m koşulunun sağlanması gerekmektedir. Türk alfabesi kullanıldığı için m = 29 olmalıdır. Verilen örnekte a = 3 olduğundan, en küçük c tam sayısı 10 olarak bulunur.

Tablo 1.5 Şifre Çözme Adımlarının Tablosu

Şifrelenmiş harfler	ı	f	r	d	o
y	10	6	20	4	17
$10(y-2)$	80	40	180	20	150
$10(y-2) \bmod 29$	22	4	6	20	5
Şifresi çözülmüş harfler	ş	i	f	r	e

Doğrusal şifreleme algoritmalarında kullanılan anahtarlar, alfabenin uzunluğuna bağlı olarak bazı durumlarda işlevsiz hale gelebilir. Türk alfabesi 29 karakterden oluştuğu için kendisinden küçük herhangi bir sayı ile aralarında asal olma ihtimali yoktur ve seçilecek her (a,b) ikilileri için algoritma çalışacaktır. Ancak İngiliz alfabesi gibi 26 harfli bir alfabede durum böyle değildir. İngiliz alfabesi için (4,5) ikilisi anahtar olarak seçilirse, hem "e" hem de "r" harflerini "v" olarak şifreleyeceği için şifre çözülürken alıcı, gönderilen mesajda hangi harfin kullanıldığını tam olarak bilemeyecektir.

Bu sorun, a'nın çarpımsal tersinin biricik olmamasından kaynaklanmaktadır. Mod 26'da 1 elde etmek için 4 ile çarpılabilecek birden fazla sayı olduğundan, (4,5) anahtarı istenildiği gibi çalışmayacaktır. Bu alfabede olası tüm anahtarları hesaplamak da mümkündür. Bir lineer kongrüans olan $ax \equiv b \pmod m$ denklemi, a ve m aralarında asal olduğunda herhangi bir $b \in Z_m$ için eşsiz bir çözüm, $x \in Z_m$ bulunur (Niven, 2004). Bu teoreme göre $26 = 2 \cdot 13$ olduğundan, $ebob(a, 26) = 1$ olan $a \in Z_{26}$ değerleri a = 1, 3, 5, 7, 9, 11, 15, 17, 19, 21, 23 ve 25 olarak bulunur. b, Z_{26} içindeki herhangi bir eleman olabilir. Bu nedenle, $12 \cdot 26 = 312$ olası anahtarı bulunmaktadır.

Doğrusal şifrelemenin Python dilindeki genel kodlama yapısının sözde kodu şu

şekildedir:

ALGORİTMA 5: DOĞRUSAL ŞİFRELEME/ÇÖZME ALGORİTMASI

Girdi: a (tamsayı), b (tamsayı), metin (dizge), sifre (boole; False⇒ şifrele, True⇒ çöz)

Çıktı: sonuc (dizge)

```
alfabe ← "ABCÇDEFGĞHIİJKLMNOÖPRSŞTUÜVYZ"
n ← |alfabe|
sonuc ← ""
foreach karakter ∈ Upper(metin) do
    if karakter ∈ alfabe then
        if ¬sifre then
            s ← IndexOf(alfabe, karakter)
            s₁ ← (a·s + b) mod n
            sonuc ← sonuc ‖ alfabe[s₁]
    else
        a_ters ← None
        for i ← 1 to n − 1 do
            if (a·i) mod n = 1 then
                a_ters ← i; break
        if a_ters = None then
            döndür "a sabiti yanlış girildi"
        s ← IndexOf(alfabe, karakter)
        s₁ ← (a_ters·(s − b)) mod n
        sonuc ← sonuc ‖ alfabe[s₁]
    else
        sonuc ← sonuc ‖ karakter
döndür sonuc
```

ALGORİTMA 6: DOĞRUSAL ŞİFRELEME/ÇÖZME ALGORİTMASI ANA AKIŞI

```
metin ← Input("Şifrelemek istediğiniz metni girin:")
a ← 3
b ← 2
sifrelenmis_metin ← dogrusalSifreleme(a, b, metin, False)
Print("Şifrelenmiş metin: " ‖ sifrelenmis_metin)
```

```
    cozulmus_metin ← dogrusalSifreleme(a, b, sifrelenmis_metin, True)
6   Print("Çözülmüş metin: " ‖ cozulmus_metin)
```

1.2. Blok Şifreleme Algoritmaları

Modern veri güvenliğinde yaygın olarak kullanılan bir şifreleme yöntemidir. Bu yöntemde, veri belirli bir boyutta bloklara ayrılır ve ardışık olarak şifreleme algoritması tarafından işlenir. Shannon tarafından yayımlanan makale (Shannon, 1948), modern blok şifreleme yöntemlerinin temelini atmıştır. Makalede, karmaşıklık ve yayılma prensipleri üzerinde durulmuştur.

Karmaşıklık, açık metin ile şifreli metin arasındaki ilişkinin belirsizleştirilmesini ifade eder. Yani şifreleme işlemi sonucu elde edilen şifreli metin, açık metinle ilişkili herhangi bir yapıyı veya deseni açığa vurmamalıdır. Yayılma ise açık metindeki desenlerin şifreli metinde tespit edilemez hale getirilmesini sağlar.

Shannon, karmaşıklık ve yayılmanın yer değiştirme ve doğrusal dönüşüm gibi işlemlerle elde edilebileceğini önerir. Yer değiştirme, açık metindeki sembollerin veya bitlerin başka sembollerle veya bitlerle değiştirilmesini ifade ederken, doğrusal dönüşüm ise şifreleme işlemi sırasında matematiksel işlemlerle doğrusal dönüşümlerin uygulanmasını içerir. Bu prensipler, günümüzde kullanılan blok şifrelerin tasarımında temel alınmaktadır. İki ana blok şifre mimarisi, yer değiştirme-permütasyon ağları ve Feistel ağları, Shannon'ın prensiplerini uygulamak için yer değiştirme ve doğrusal dönüşümü kullanır.

Yer değiştirme-permütasyon ağları, açık metinde yer değiştirme ve ardışık bir permütasyon işleminin uygulandığı şifreleme yöntemleridir. Feistel ağları ise açık metni iki yarı bloğa böler ve ardışık turda yer değiştirme ve doğrusal dönüşüm işlemlerini uygulayarak şifreler. Her turda, sağ yarı blok sola taşınırken, sol yarı blok, bir dönüşüm fonksiyonu tarafından işlenir ve ardından sağ yarı blokla XOR (mOD2) işlemi uygulanır.

Bu işlem, her turda farklı bir anahtar kullanılarak tekrarlanır ve son turda şifrelenmiş bloklar tekrar birleştirilir.

1.1.1. Blok Şifreleme Algoritmalarının Özellikleri

Blok Boyutu: Açık metin, belirli bir boyutta bloklara ayrılır. Blok boyutu, algoritmanın veri işleme kapasitesini belirler ve genellikle 64, 128 veya 256 bit olur. Blok boyutu, güvenlik ve performans arasında bir denge kurmak için dikkatlice seçilmelidir. Daha büyük blok boyutları, daha fazla veri işleme kapasitesine ve daha güçlü bir şifreleme sağlayabilir, ancak aynı zamanda işlem süresini arttırır.

Anahtar Boyutu: Anahtar, şifreleme ve deşifreleme işlemlerinde kullanılan gizli bir değerdir. Anahtar boyutu, şifreleme algoritmasının güvenliğini etkileyen önemli bir faktördür. Daha uzun anahtar boyutları, daha güvenli bir şifreleme sağlar ancak daha fazla hesaplama gücü gerektirir ve işlem süresini arttırır. Dolayısıyla anahtar boyutu, güvenlik gereksinimleri ve performans hedefleri dikkate alınarak dikkatlice seçilmelidir.

Şifreleme Algoritması: Açık metin bloklarını anahtar ile matematiksel işlemlere tabi tutarak şifreli metin üretilen aşamadır. Bir şifreleme algoritması, açık metin bloklarını nasıl karıştıracağını, yer değiştireceğini, dönüştüreceğini ve matematiksel işlemlerin nasıl uygulanacağını belirler.

Döngü Sayısı: Algoritmanın güvenliğini artırmak için kullanılan döngü sayısını ifade eder. Her turda, blokların karıştırılması, yer değiştirilmesi, dönüştürülmesi gibi işlemler tekrarlanır. Daha fazla tur, daha güvenli bir şifreleme sağlayabilir, ancak aynı zamanda işlem süresini arttırır.

Değiştirme ve Permütasyon İşlemleri: Blokların yerlerini değiştirerek ve veri bloklarını dönüştürerek şifreleme güvenliği sağlanır. Bu işlemler, şifreleme algoritmasının karmaşıklığını ve güvenliğini artırarak, saldırganların veri bloklarının içeriğini anlamalarını zorlaştırır. Değiştirme işlemi, veri bloklarının yerlerini karıştırarak veya yer değiştirerek, şifrelenen verinin orijinal düzenini gizler. Permütasyon işlemi ise, veri bloklarını dönüştürerek, verinin yapısal değişiklikler geçirmesini sağlar.

1.2.1. Vigenère Şifrelemesi

Vigenère şifrelemesi, İlk olarak 16. yüzyılda Fransız diplomat Blaise de Vigenère tarafından tanımlanmıştır (Kahn D., 1968). Vigenère şifrelemesi, alfabetik karakterleri bir dizi Sezar şifrelemesiyle değiştirir. Her bir Sezar şifrelemesi, anahtar kelimenin bir harfi ile belirlenir (Menezes, vd., 1996). Örneğin, anahtar kelime "ANAHTAR" ve şifrelenecek metin "MERHABA" ise, ilk harf 'M' 'A' ile, ikinci harf 'E' 'N' ile ve böylece devam edilir. Bu durumu matematiksel olarak şöyle ifade edebiliriz;

$$C_i=(P_i+K_i) \bmod n \tag{1.16}$$

Burada C_i şifrelenmiş karakteri, P_i açık metnin i'nci karakterlerini, K_i mod n ise anahtar kelimenin i'nci karakterini temsil eder (Stinson D. R., 2005). Vigenère şifrelemesi, günümüz standartlarına göre güvenli kabul edilmez. Anahtarın yeterince uzun olmaması ve/veya anahtarın tekrar kullanılması, şifrelemenin kırılmasına yol açabilir. Bu, kripto analiz yöntemlerinden Kasiski testi ile mümkündür (Kasiski, 1863).

Vigenère şifrelemesi için Python dilinde yazılmış programın sözde kodu:

ALGORİTMA 7: VİGENERE ŞİFRELEME ALGORİTMASI

```
Girdi: acik_metin (dizge), anahtar (dizge)
Çıktı: sifrelenmis_metin (dizge)
sifrelenmis_metin ← ""
anahtar_uzunluk ← |anahtar|
alfabe ← "ABCÇDEFGĞHIİJKLMNOÖPRSŞTUÜVYZ"
alfabe_uzunluk ← |alfabe|
acik_metin ← Upper(acik_metin)
anahtar ← Upper(anahtar)
for i ← 0 to |acik_metin| − 1 do
    anahtar_karakter ← anahtar[i mod anahtar_uzunluk]
    acik_kod ← IndexOf(alfabe, acik_metin[i])
    anahtar_kod ← IndexOf(alfabe, anahtar_karakter)
    yeni_kod ← (acik_kod + anahtar_kod) mod alfabe_uzunluk
    yeni_karakter ← alfabe[yeni_kod]
    sifrelenmis_metin ← sifrelenmis_metin || yeni_karakter
retun sifrelenmis_metin
```

ALGORİTMA 8: VİGENERE ŞİFRELEME ALGORİTMASI ANA AKIŞI

```
anahtar ← "ANAHTAR"
acik_metin ← Input("Lütfen şifrelemek istediğiniz metni giriniz:")
sifrelenmis ← Vigenere_Sifreleme(acik_metin, anahtar)
Print("Şifrelenmiş Metin: "  ||  sifrelenmis)
```

Bu kodda, input fonksiyonu bir metin girmesini istemektedir. Girilen metin acik_metin değişkenine atanır ve daha sonra vigenere_sifreleme fonksiyonuna argüman olarak gönderilir. Şifreleme işlemi tamamlandıktan sonra, şifrelenmiş metin ekrana yazdırılır.

1.2.2. Feistel Ağı

1973'te Horst Feistel tarafından geliştirilen bir blok şifreleme yapısıdır (Feistel, 1973). Feistel ağı, kendi başında bir şifreleme algoritması değildir daha çok şifreleme algoritmaları için bir yapı iskeletidir. Bu yapı, şifrelemenin ve şifre çözmenin benzer işlemlerle gerçekleştirilebileceği şekilde tasarlanmıştır. Şekil 1.1'da görüldüğü gibi, Feistel ağı, n-bitlik giriş bloğunu iki parçaya ayırır, bu parçalar her zaman eşit olmak zorunda değildir, sol parça L_0, sağ parça R_0 ve n tur sayısı olmak üzere her bir turda (K_1, K_2, K_3, ..., K_n) anahtar dizisinden

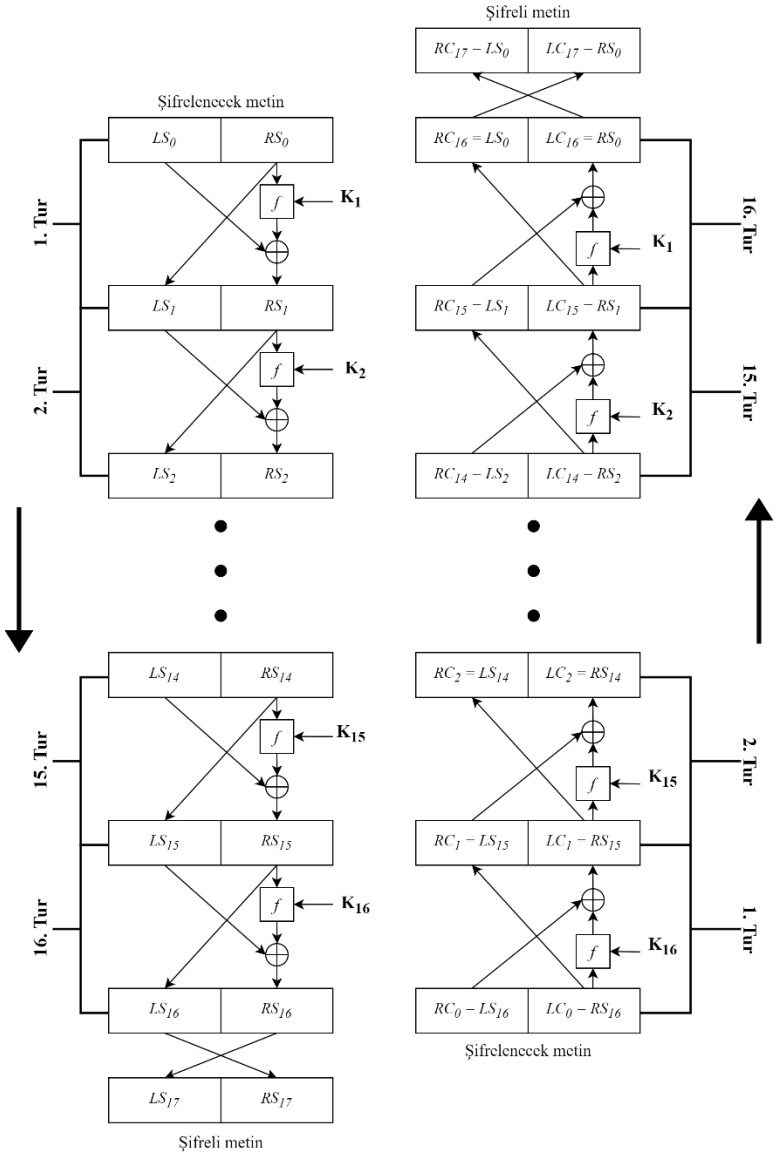
ilgili anahtar kullanılır, bu anahtarlar genellikle ana anahtardan bir algoritma yardımıyla elde edilir.

Her şifreleme turu, f fonksiyonu ile R_{n-1} işlenip bir önceki turun sol parçası, L_{n-1} ile XOR işlemine girer ve böylelikle bir sonraki turun sağ yarısı, Rn üretilir. Herhangi bir n turu için sağ ve sol parçalar şu şekilde ifade edilir:

$$R_n=L_{n-1}\oplus f(R_{n-1}, K_n) \tag{1.17}$$

$$L_n=R_{n-1} \tag{1.18}$$

Feistel ağının gücü, f fonksiyonunun zorluğuna dayanır fonksiyon genellikle S-kutuları, P-kutuları ve diğer lineer olmayan dönüşüm yöntemleri kullanılarak oluşturulur giriş bloğunun yarısını ve anahtarı alan fonksiyon bir dizi operasyonun ardından çıktı üretir bu çıktı her ne kadar karmaşık olursa olsun XOR işlemleri sayesinde geri çevrilebilirdir.



Şekil 1.1 Feistel Ağı'nın Çalışma Prensibi

Şekil 1.1'de, şifre çözme işleminin ilk turunu inceleyelim ve 16'ncı turun sonundaki 32-bitlik dönüşüme eşit olduğunu gösterelim: Oluşabilecek karışıklığı önlemek adına şifreleme kısmı için sol parçanın ve sağ parçanın yanına *S*, aynı şekilde çözme işlemlerinin gerçekleştiği yerlerde de C, ekleyeceğiz, yani L_i, R_i, yerine sırasıyla LS_i, RS_i, ve LC_i, RC_i yazıyor olacağız.

Gösterilmesi gereken,

$$LS_{16}=RS_{15} \tag{1.19}$$

$$RS_{16}=LS_{15} \oplus f(RS_{15},K_{16}) \tag{1.20}$$

Şifreleme kısmı,

$$LS_{16}=RS_{15}$$

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

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

$$=[LS_{15}\oplus f(RS_{15},K_{16})]\oplus f(RS_{15},K_{16}) \quad (1.21)$$

XOR şu özellikleri sağlar,

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

$$D\oplus D=0$$

$$E\oplus 0=E \quad (1.22)$$

Öyleyse, $LC_1 = RS_{15}$ ve $RC_1 = LS_{15}$ gerçeklenir. Şifreleme turunun ilk çıktısı, şifrelemenin on altıncı turunun girdisinin 32-bitlik dönüşüm uygulanmış halidir bu eşitlik 16 tur boyunca sağlanır ve genel bir terimle ifade edebiliriz,

$$LS_i=RS_{i-1} \quad (1.23)$$

$$RS_i=LS_{i-1}\oplus f(RS_{i-1},K_i) \quad (1.24)$$

Yeniden düzenlersek:

$$RS_{i-1}=LS_i \quad (1.25)$$

$$LS_{i-1}=RS_i\oplus f(RS_{i-1},K_i)=RS_i\oplus f(LS_i,K_i) \quad (1.26)$$

Böylece, i'nci iterasyonun girdilerini, çıktıların bir fonksiyonu olarak tanımladık ve bu denklemler, Şekil 1.9'un sağ tarafında gösterilen atamaları doğrulamaktadır. Gerçekleşen işlemleri bir örnekle açıklayalım,

Şifrelenecek mesaj = "9A4C5F12" Anahtar(K) = "B37E12" olsun. (24-bitlik bu anahtar değerinin sadece sağ tarafından 16-bitlik değeri kullanılacak) 7. tur için şifreleme ve şifre çözme işlemlerini uygulayalım,

LS_6 = 9A4C,

RS_6 = 5F12 ve

K_7 = 7E12 olur.

f fonksiyonunu RS_6 üzerine uygulayalım,

$f(RS_6) = RS_6 \oplus K_7$ açıkça yazarsak,

f(5F12) = 5F12⊕7E12 = 2100

Elde ettiğimiz bu değeri LS_6 ile XOR işlemine tabi tutalım,

$LS_6 \oplus f(RS_6) = 9A4C \oplus 2100 = BB4C$

Bu durumda, $LS_7 = RS_6 = 5F12RS_7 = BB4C$ ve bu turun sonunda elde edilen blok ”5F12BB4C” olur. Yine aynı tur için şifre çözme işlemleri de şu şekilde gerçekleşecektir,

$LC_7 = 5F12RC_7 = BB4C$ şifre çözme girdileri olmak üzere aynı f fonksiyonu ve $K_7 = 7E12$ anahtarını kullanarak işlemleri gerçekleştirelim,

$f(RC_7) = RC_7 \oplus K_7$

f(BB4C) = BB4C ⊕7E12 = C55E olarak bulunur.

Bu değeri, LC7 değeriyle XOR işlemine tabi tutalım,

LC7 ⊕ f (RC7) = 5F12 ⊕ C55E = 9A4C elde edilir.

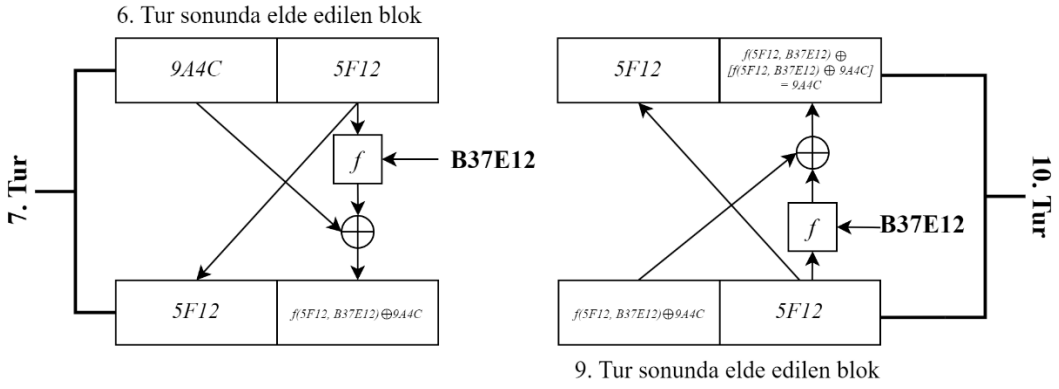
Sonuç olarak, LC6 = RC7 = BB4C ve RC6 = LS6 = 9A4C elde edilir.

Yani, şifre çözme işlemi sonucunda, başlangıçta (şifreleme işleminde) 6. turun sonunda elde edilen,

LS6 = 9A4C

RS6 = 5F12

Değerlerini geri kazanmış olduk. Şekil 1.2’de bu adımlar gözlemlenebilir.



Şekil 1.2 Feistel Ağı İçin Şifreleme ve Çözme Örneği

Klavyeden girilecek en fazla 8 karakter (64-bit) uzunluğunda veriyi yine klavyeden girilen 4 karakterlik (16-bit) anahtar ile şifreleyen ve çözen Python dilinde yazılmış bir örneğin sözde kod ile gösterimi,

ALGORİTMA 9: FEİSTEL AĞINDA BİR TUR

Girdi: sol, sag, anahtar

Çıktı: Güncellenmiş(sol, sag)

```
1  geçici ← sag
2  sag ← sol ⊕ TurFonksiyonu(sag, anahtar)
3  sol ← geçici
4  return (sol, sag)
```

ALGORİTMA 10: FEİSTEL AĞINDA TUR FONKSİYONU

Girdi: veri, anahtar

Çıktı: veri üzerinde anahtara bağlı dönüşüm

```
1  return veri ⊕ anahtar
```

ALGORİTMA 11: FEİSTEL AĞINDA ŞİFRELEME FONKSİYONU

Girdi: veri (64-bit tamsayı), anahtar (32-bit tamsayı), turlar (varsayılan 16)

Çıktı: Şifreli 64-bit tamsayı

```
1  sol ← veri ≫ 32                 // üst 32 bit
2  sag ← veri & 0xFFFFFFFF         // alt 32 bit
3  for i ← 1 to turlar do
4  |  (sol, sag) ← Feistel(sol, sag, anahtar)
5  return (sol ≪ 32) | sag
```

ALGORİTMA 12: FEİSTEL AĞINDA ÇÖZME FONKSİYONU

Girdi: veri (64-bit tamsayı), anahtar (32-bit tamsayı), turlar (varsayılan 16)

Çıktı: Çözülen 64-bit tamsayı

```
1  sol ← veri ≫ 32
2  sag ← veri & 0xFFFFFFFF
3  for i ← 1 to turlar do
4  |  (sag, sol) ← Feistel(sag, sol, anahtar)
5  return (sol ≪ 32) | sag
```

ALGORİTMA 13: FEİSTEL AĞINDA METNİ HEX'E ÇEVİRME FONKSİYONU

Girdi: metin (UTF-8 dizgesi)

Çıktı: Tamsayı olarak hex değeri

```
1  baytlar ← UTF-8.Encode(metin)
```

```
2  heks ← Hexlify(baytlar)
3  return Int(heks, 16)
```

ALGORİTMA 14: FEİSTEL AĞINDA HEX'İ METİNE ÇEVİRME FONKSİYONU

```
Girdi: hex_veri (tamsayı)
Çıktı: metin (UTF-8 dizgesi)
1  h ← Hex(hex_veri)              // "0x" ön ekli dize
2  h ← h'nin "0x" ön eki çıkarılmış hali
3  h ← ZFill(h, 16)               // en az 8 bayt olacak şekilde sıfırla doldur
4  baytlar ← Unhexlify(h)
5  return Decode(baytlar, UTF-8)
```

ALGORİTMA 15: FEİSTEL AĞI ANA PROGRAM AKIŞI

```
Girdi: Kullanıcıdan iki dize: metin_girdi (en çok 8 karakter), anahtar_girdi (tam 4
       karakter)
Çıktı: Açık metin, şifreli metin (hex), ve çözülen metin
1  if |metin_girdi| > 8 then
2  |   döndür Hata: "metin 8 karakteri aşmamalı"
3  if |anahtar_girdi| ≠ 4 then
4  |   döndür Hata: "anahtar 4 karakter olmalı"
5  metin_8 ← metin_girdi sağa \0 ile 8 karaktere tamamlanır
6  acik_metin ← MetniHexeCevir(metin_8)
7  anahtar ← MetniHexeCevir(anahtar_girdi)
8  sifreli ← Sifrele(acik_metin, anahtar, 16)
9  desifreli ← Desifrele(sifreli, anahtar, 16)
10 desifreli_dize ← HexiMetneCevir(desifreli)
11 desifreli_temiz ← sonda kalan \0 karakterleri kırpılmış desifreli_dize
12 Ekrana yaz: Açık Metin, Şifreli Metin (hex), Deşifre Edilmiş Metin
```

1.2.3. RSA Şifreleme Algoritması

Günümüz dijital iletişiminin güvenlik yapı taşlarını şekillendiren protokollerin merkezinde 20. yüzyılın sonlarına doğru ortaya çıkan bir buluş yer alır: açık anahtarlı şifreleme. Bu yöntemin ilk kez pratiğe dökülmüş ve yaygınlık kazanmış örneği olan RSA, 1977 yılında Massachusetts Institute of Technology (MIT) bünyesinde çalışan üç araştırmacı tarafından tanıtıldı. Bilgisayar bilimcisi Ron Rivest ve Adi Shamir ile matematikçi Leonard Adleman.

Geliştirdikleri algoritma, soyadlarının baş harflerinden türetilen RSA adıyla anılmaya başlandı RSA algoritması, Diffie ve Hellman'ın kuramsal çerçevesini çizdiği açık anahtarlı şifreleme fikrini, sağlam matematiksel yapılar üzerine inşa ederek pratik bir çözüme dönüştürmüştür. Algoritma günümüzde dijital güvenlik protokollerinde (örneğin TLS/SSL, dijital imza standartları) yoğun biçimde kullanılmaktadır.

RSA'nın güvenliği, tam sayı çarpanlara ayırma probleminin (Integer Factorization Problem, IFP) zorluğuna dayanır. Bu problem, etkin bir çözüm yönteminin bilinmemesi ve herhangi bir çokterimli zamanda çözülebilir deterministik bir algoritma bulunmaması nedeniyle hesaplama teorisi açısından güvenilirliğini korumaktadır.

Tanım (Euler'in Totient Fonksiyonu): $\varphi : \mathbb{Z}^+ \to \mathbb{Z}^+$ fonksiyonu, $\varphi(n)$ değerini, $1 \leq k \leq n$ aralığındaki tam sayılar içinden $OBEB(k, n) = 1$ koşulunu sağlayan k tamsayılarının sayısı olarak tanımlar.

Bu fonksiyon çarpımsaldır; yani $OBEB(m, n) = 1$ ise $\varphi(mn) = \varphi(m)\varphi(n)$'dir. RSA için bu fonksiyonun en önemli özelliği, asal sayılar için hesaplanmasındaki basitliğidir:

- Eğer p bir asal sayı ise, $\varphi(p) = p - 1$'dir.
- Eğer $n = p \cdot q$ ve p, q birbirinden farklı iki asal sayı ise, $\varphi(n) = \varphi(p) \cdot \varphi(q) = (p - 1)(q - 1)$ olur.

Bu özellik kritiktir, çünkü p ve q bilindiğinde $\varphi(n)$'yi hesaplamak kolayken, sadece n bilindiğinde $\varphi(n)$'yi hesaplamak, n'yi çarpanlarına ayırmak kadar zordur.

RSA protokolü, üç temel algoritmadan oluşur: Anahtar Üretimi, Şifreleme ve Çözme.

Anahtar Üretim Algoritması

1. Asal Seçimi: k-bitlik iki büyük ve birbirinden farklı asal sayı, p ve q, rastgele seçilir. Bu seçim, olasılıksal asallık testleri (örn. Miller-Rabin) ile yapılır.
2. Modül Hesaplanması: Modül $n = p \cdot q$ olarak hesaplanır. n'nin bit uzunluğu (yaklaşık 2k), anahtar boyutunu belirler.
3. Totient Hesaplanması: Euler'in totient değeri $\varphi(n) = (p - 1)(q - 1)$ olarak hesaplanır.
4. Açık Üs Seçimi: $1 < e < \varphi(n)$ koşulunu sağlayan ve $OBEB(e, \varphi(n)) = 1$ olan bir e tamsayısı seçilir. Verimlilik açısından e genellikle $2^{16} + 1 = 65537$ gibi küçük bir asal sayı olarak seçilir.
5. Özel Üs Hesaplanması: $d \equiv e^{-1} \pmod{\varphi(n)}$ olacak şekilde d tamsayısı, Genişletilmiş Öklid Algoritması kullanılarak hesaplanır.
6. Çıktı: Açık anahtar $PK = (n, e)$ ve özel anahtar $SK = (n, d)$ olarak tanımlanır. p, q ve $\varphi(n)$ değerleri gizli tutulmalı ve güvenli bir şekilde imha edilmelidir.

Şifreleme Algoritması

Bir E düz metnini, $E \in \{0, 1, \ldots, n-1\}$ olacak şekilde bir tamsayı olarak kabul ederek, şifreli metin C şu şekilde hesaplanır:

$C = \text{Şifrele}(PK, E) \equiv E^e \pmod{n}$

Bu işlem, modüler üs alma (modular exponentiation) algoritmaları (örn., tekrarlı kare alma) ile verimli bir şekilde gerçekleştirilir.

Çözme Algoritması

Bir C şifreli metnini deşifre etmek için, alıcı kendi özel anahtarını SK = (n, d) kullanarak orijinal metin E'yi şu şekilde geri elde eder:

$E = \text{Çözme}(SK, C) \equiv C^d \pmod{n}$

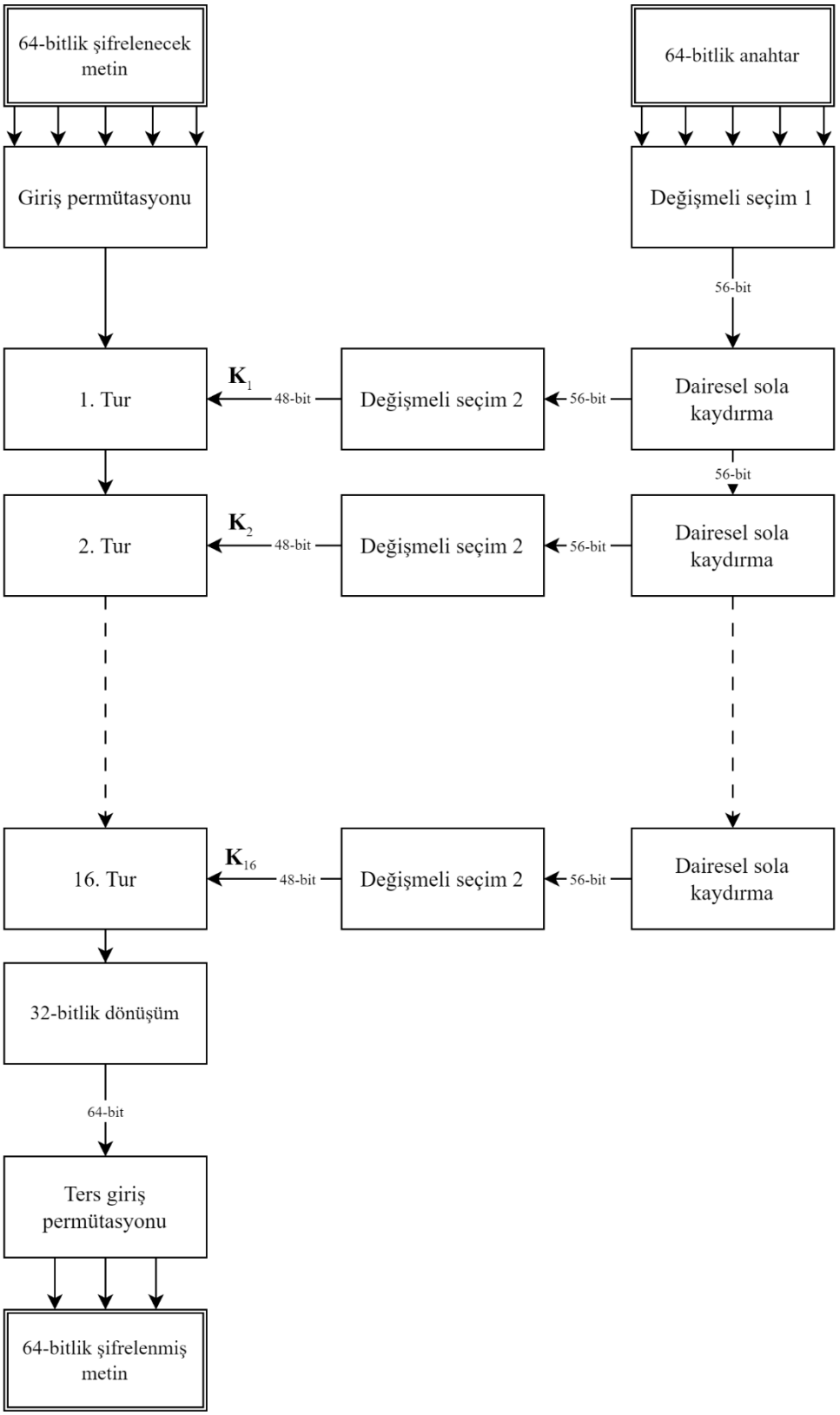
1.2.4. Veri Şifreleme Standardı

Veri Şifreleme Standardı (DES) algoritması, International Business Machines (IBM) tarafından geliştirilmiş ve 1977 yılında Amerikan Ulusal Standartlar Enstitüsü (NIST) tarafından standart olarak kabul edilmiştir (Standard, 1977). DES, 64-bit blok şifreleme algoritmasıdır ve 56-bitlik simetrik anahtar kriptografisi kullanır bununla birlikte bir Feistel ağı olduğundan her bir turda gerçekleşen şifreleme ve deşifreleme işlemlerini matematiksel olarak şu şekilde ifade edebiliriz:

$$L_{i+1}=R_i, \tag{1.27}$$

$$R_{i+1}=L_i \oplus f(R_i,K_i) \tag{1.28}$$

L_i ve R_i sırasıyla, i'nci tura giren sol ve sağ 32-bitlik bloklardır. K_i i'nci tura özgü 48-bitlik anahtar. f bir dizi bit değiştirme, taşıma ve yerine koyma işlemlerini ifade eder.



Şekil 1.3 DES'in Çalışma Prensibi

Şekil 1.3'de DES'in nasıl çalıştığı gösterilmiştir, şeklin sol kısmında şifreleme algoritmasına alınan düz metin giriş permütasyonundan geçer hem permütasyon hem de yer

değiştirme fonksiyonlarını içeren aynı fonksiyonun on altılı turunu takip eder, çıktının sağ ve sol yarıları 32-bitlik dönüşümle öncül çıktı oluşturulur ve bu 64-bitlik şifreli metni oluşturmak için ters giriş permütasyonundan geçirilir. Giriş ve ters giriş permütasyonları dışında DES, Şekil 1.1'da gösterilen Feistel ağı yapısıdır.

Giriş permütasyonu, giriş permütasyonu ve tersi tablo 1.6 ve tablo 1.7'de gösterilmiştir. Bu tablolar, 64-bitten oluşur ve her bir değerin pozisyonunu belirlidir.

Tablo 1.6 Giriş Permütasyonu

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

Tablo 1.7 Ters Giriş Permütasyonu

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

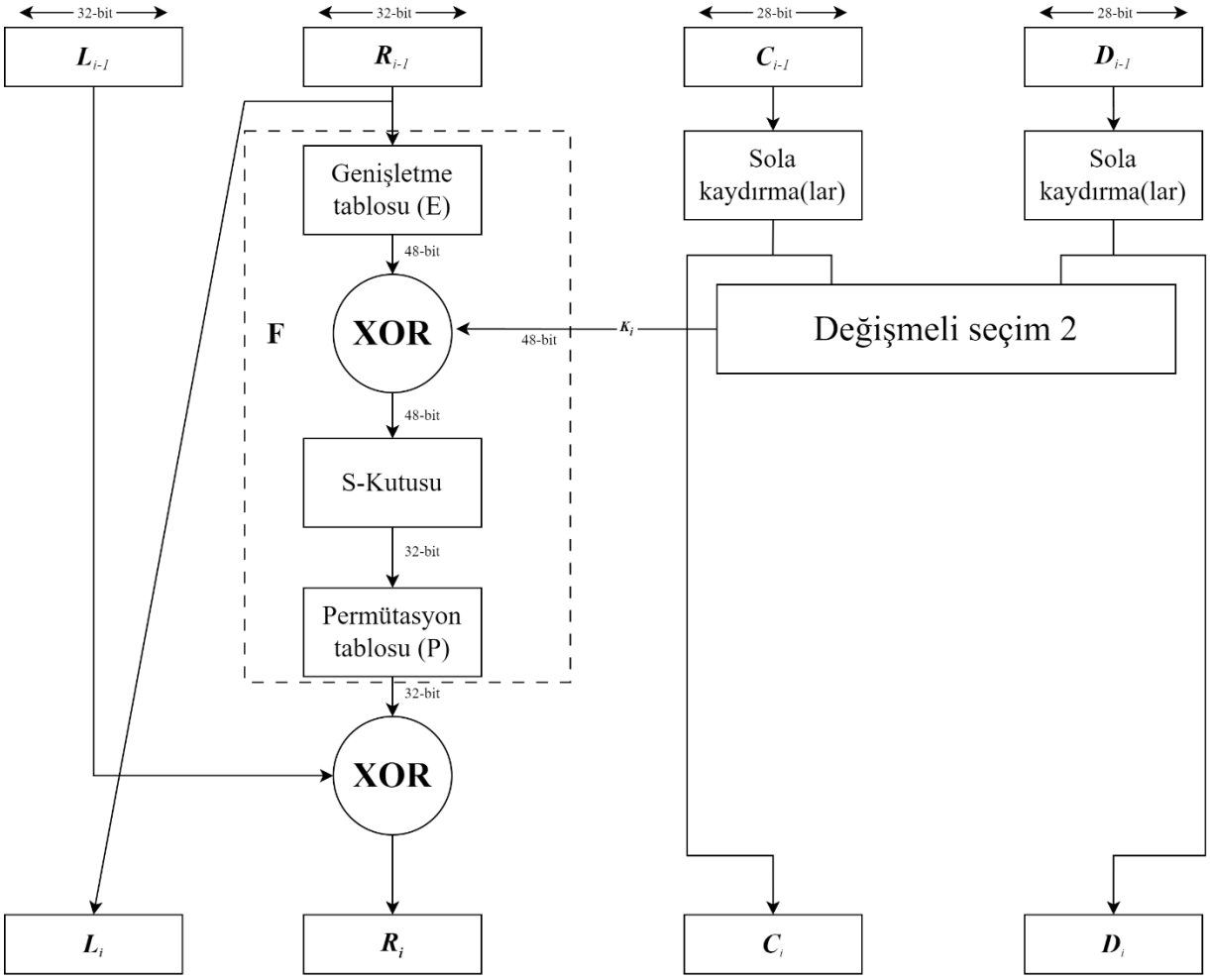
Tablo 1.8 Genişletme Tablosu (E)

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

Tablo 1.9 Permütasyon Fonksiyonu (P)

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

Genişletme tablosu, 32 farklı karakterden oluşan 48-bitlik bir yapıdır tablo 1.8'de görüleceği üzere bazı karakterler tekrar eder böylelikle genişletme tablosuna giren 32-bitlik değer, yerine koyma işlemlerinden sonra 48-bit olarak çıkar.



Şekil 1.4 DES'in Tek Bir Turunun Çalışma Prensibi

Şekil 1.4 tek bir turun iç yapısını göstermektedir. DES'in her turundaki F fonksiyonunun doğrusal olmayan tek bileşeni Şekil 1.5'de gösterilen sekiz adet S-kutusudur $S_i : \{0,1\}^6 \rightarrow \{0,1\}^4$ (i= 1, . . . , 8) 32 bitlik sağ yarı R önce genişletme $E : \{0,1\}^{32} \rightarrow \{0,1\}^{48}$ ile 48 bite

çıkarılır, alt anahtar K_i ile XOR'lanır ve ortaya çıkan 48 bit, sekiz 6-bitlik bloğa bölünür. Her blokta satır indisi uç bitlerden $r = 2b_1+b_6$, sütun indisi orta dört bitten $c=8b_2+4b_3+2b_4+b_5$ hesaplanır; tablo girdisi S_i (r, c) 4 bit üretir. Sekiz S-kutusundan çıkan 32 bit, sabit *P* permütasyonu ile karıştırılarak $F(R, K_i) = P(S(E(R) \oplus K_i))$ elde edilir; standart, E, S_1 örneği ve P tablolarını ve satır/sütun seçim kuralını ayrıntılı olarak verir (NIST, 1999; Menezes A. J vd., 1996)

Tablo 1.10 Bir 6×4 S-kutusu İçin Fark Dağılım Tablosu (DDT)

$$DDT_S(a,\beta) = \left|x \in 0,1^6 : S(x) \oplus S(x \oplus a) = \beta\right|$$

Maksimum diferansiyel olasılık (MDP) $\max_{a \neq 0,\beta} DDT_S(a,\beta)/64$'dür. DES ölçütlerinden (S.7), herhangi bir sabit $a \neq 0$ için aynı β'ya giden giriş çiftlerinin sayısını 32 giriş çifti içinde en çok 8'le sınırlar; bu, her etkin S-kutu başına olasılığın $\leq 1/4$ olduğu anlamına gelir. Coppersmith'in analizine göre bu ve P ölçütleri birlikte, 12–16 tur boyunca tur başına ortalama ≥ 1.6 etkin S-kutu olacak şekilde zorlayarak olası diferansiyel yolların toplam olasılığını üstel olarak bastırır (Coppersmith, 1994; Biham ve Shamir, 1991)

Coppersmith'in yayımladığı sekiz ölçüt, DES S-kutularını diferansiyel kriptanalize (ve dolaylı olarak diğer istatistiksel saldırılara) dayanıklı kılmak için seçilmiştir:

S.1 Boyut: 6→4.

S.2 Her çıkış biti, giriş bitlerinin herhangi bir doğrusal kombinasyonuna fazla yakın olmamalı (olasılık $\approx 1/2$).

S.3 Uç bitler sabitlenip orta dört bit gezildiğinde her satır tam bir permütasyondur; çıktılar 0,1, ...,15 kümesinin bir permütasyonudur.

S.4 Tek bit giriş farkı en az iki bitlik çıkış farkı doğurmalı.

S.5 Orta iki bitte fark varsa, çıkışta en az iki bit değişmeli.

S.6 İlk iki bit fark, son iki bit aynı ise, çıkış aynı olamaz.

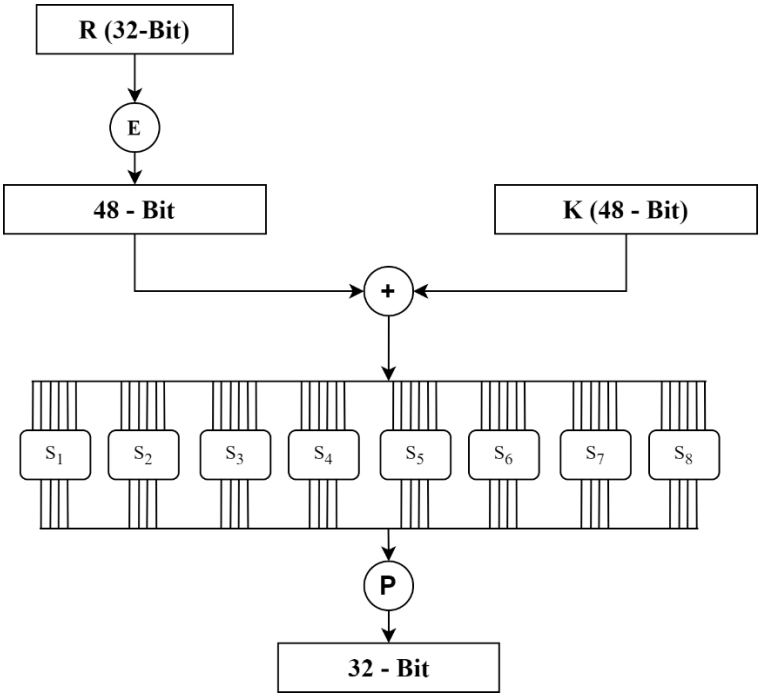
S.7 Her $a \neq 0$ için aynı β'yı üreten giriş çiftlerinin sayısı ≤ 8 (MDP $\leq 1/4$).

S.8 "Üç bitişik etkin S-kutu" vakasına karşı ek kısıt (özellikle β=0 için).

Ayrıca P permütasyonu için (P-1...P-3) dağıtım ölçütleri, bit yayılımını hızlandırmak üzere tanımlanmıştır (Coppersmith, 1994; NIST, 1999)

(**S.3**) sayesinde her satırın {0, 1, . . ., 15} çıktılarının permütasyon olması, tek-tur tersine çevirmeleri zorlaştırır ve istatistiksel tekdüzeliği artırır; literatürde DES S-kutularının satırpermutasyon özelliği ve Strict Avalanche Criterion (SAC) ile Bit Independence Criterion

(BIC) bakımından iyi performansı klasik olarak vurgulanır (Webster ve Tavares, 1985; Menezes, vd., 2018).



Şekil 1.5 S-Kutuları

2. KUANTUM ŞİFRELEME

Kuantum şifreleme, bilgi güvenliğini matematiğin zor çözümlerine değil, doğrudan kuantum mekaniğinin yasalarına dayandıran yöntemlerdir (Bennett ve Brassard, 2014). Temel yapıtaşları süperpozisyon, dolanıklık ve no-cloning teoremidir (Ekert, 1991). Bu yaklaşımın en büyük uygulaması Kuantum Anahtar Dağıtımı (QKD)'dir. İki taraf, foton temelli bir kuantum kanal üzerinden ortak bir gizli anahtar türetir ardından klasik kanalda yürütülen hata düzeltme ve gizlilik artırımı adımlarıyla anahtar arındırılıp kullanılabilir hâle getirilir. Hazırlama-ölçme sınıfının öncüsü BB84'tür; dolanıklık kullanılan kısım ise E91 ile örneklenir. Elde edilen anahtar, örneğin One-Time Pad (OTP) ile kullanıldığında, bilgi kuramsal düzeyde şifreleme güvenliği sağlar (Bennett ve Brassard, 2014).

Kuantum şifrelemenin güvenliği, Hilbert uzayında tanımlı kuantum durumlarının ölçüm kurallarına dayanır. Yanlış tabanda yapılan ölçümler Born kuralı gereği geri dönüşsüz hatalara yol açar; bu da izinsiz dinlemeyi kaçınılmaz biçimde görünür kılar. Dolanıklığa dayalı protokollerde ise güvenlik, gözlenen korelasyonların Bell eşitsizliklerini ihlal etmesi ile test edilir. Bu nedenle kuantum şifreleme güvenliği yalnızca teorik bir kavram olarak kalmaz, aynı zamanda deneysel olarak da doğrulanabilir.

Bu çerçevede kuantum bilgisayarların güncel gelişim düzeyi de önem arz etmektedir. Günümüzde süperiletken kübitler (IBM, Google), iyon tuzağı sistemleri (IonQ, Honeywell), fotonik çözümler (Xanadu) ve topolojik kübit yaklaşımları üzerine yoğun araştırmalar yapılmaktadır (Bayram ve Ilgaz Çağlayan, 2024). Bununla birlikte mevcut cihazlar, NISQ (Noisy Intermediate-Scale Quantum) dönemi olarak bilinen aşamadadır. Yüzlerce kübitlik sistemler geliştirilmiş olsa da hata düzeltme kapasiteleri sınırlı olduğundan uzun süreli güvenilir hesaplamalar henüz mümkün değildir (Preskill, 2018). Buna rağmen kuantum bilgisayarlar laboratuvar ölçeğinden çıkarak ticari ve bulut tabanlı servisler halinde kullanılmaya başlanmış; kriptografi ve optimizasyon gibi alanlarda test edilmektedir.

2.1. Kuantum Şifreleme Algoritmaları

Kuantum mekaniğinin ölçüm bozunumunu ve kopyalanamazlık ilkesini kullanarak bilgi kuramsal güvenlikte anahtar üretimi ve iletimini mümkün kılan protokolleri kapsar; pratikte gizli iletim, üretilen anahtarın OTP ile kullanılmasıyla elde edilir. Hazırla ve ölç (prepare-and-measure) sınıfının en bilinen örneği olan BB84, karşılıklı bazlarda örnekleme ve hata istatistikleriyle Eve'in varlığını saptayıp gizlilik yükseltmesi yapan bir anahtar uzlaşımı sağlar (Bennett ve Brassard, 2014). Dolanıklık tabanlı E91 ise güvenliği CHSH ihlali gibi Bell eşitsizliği testlerine bağlayarak kaynağa güvenilmeyen senaryolara daha doğal bir yol sunar

(Ekert, 1991).

Gerçekçi kurulumlar, çok fotonlu zayıf koherent kaynakları Decoy-State tekniğiyle yönetir, algılayıcı açıklıklarını MDI-QKD ile kapatır, Continuous-variable (CV) varyantlarla fiber/serbest-uzay kanallarda verimliliği artırır ve tümünü sonlu-anahtar ile evrensel-birleşik (composable) güvenlik çerçevesinde analiz eder (Pirandola, vd., 2020). Bu protokoller kuantum şifreleme olarak anılsa da, post-quantum (lattice-code, hash tabanlı) algoritmalarla amaç farklıdır: QKD fizik yasalarıyla anahtar üretir; post-quantum ise klasik şifrelemeyi kuantum saldırılara dayanıklı hale getirir ikisi birlikte uçtan uca güvenliğe tamamlayıcı katkı verir (Pirandola, vd., 2020). Uygulamada gizli anahtar oranı; QBER, algılayıcı verimi, kanal kaybı ve hata düzeltme sızıntısı gibi metriklerin sıkı kontrolüyle optimize edilir; cihaz tarafı kusurlar için modellemeli veya cihazdan-bağımsız analizler tercih edilir (Pirandola, vd., 2020).

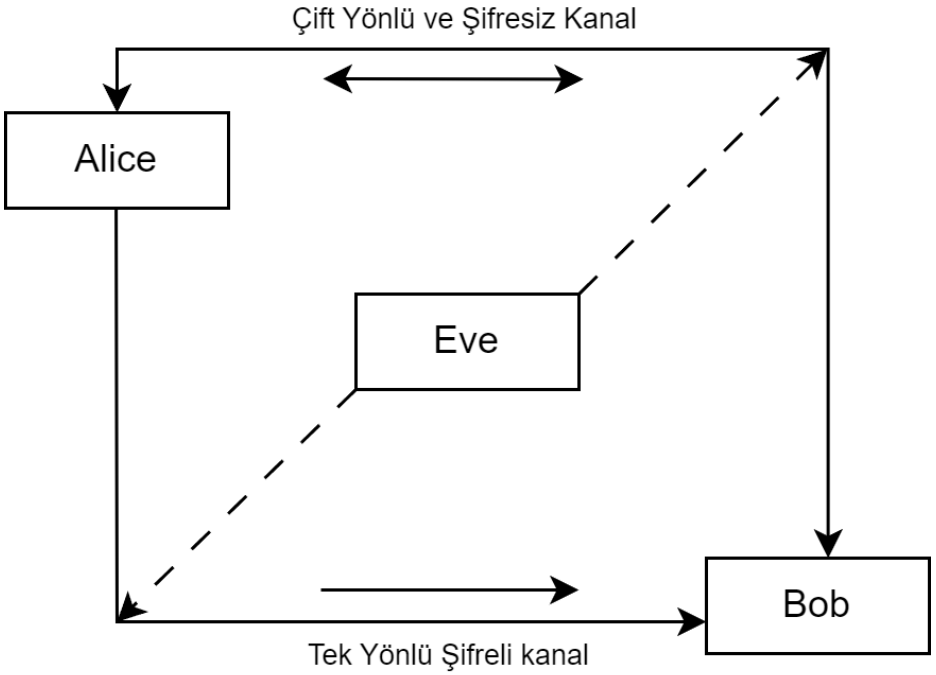
2.1.1. Kuantum Anahtar Dağıtımı (QKD)

Kuantum Anahtar Dağıtımı (QKD), kuantum kriptografi alanında geliştirilen ve geleneksel kriptografik yöntemlere kıyasla daha yüksek güvenlik hedefleyen yenilikçi bir tekniktir ve BB84, E91, B92, SARG04, Six-State, Decoy-State BB84, DPS, RRDPS, COW, MDI, TF, DI, CV, CV-MDI gibi algoritmaların anahtarlarının üretiminde kullanılır. Geleneksel Kriptografik protokoller, güvenliklerini karmaşık matematiksel problemlerin çözüm zorluğuna dayandırır; buna örnek olarak büyük asal sayıların çarpanlarına ayrılması gösterilebilir. Bu tür problemler klasik bilgisayarlarla çözülemez olarak kabul edilse de kuantum bilgisayarların gelişimi, bu geleneksel yöntemlerin güvenliğini tehlikeye atmaktadır. Buna karşılık, QKD'nin güvenlik altyapısı kuantum mekaniğinin temel ilkelerine dayanır. Heisenberg'in Belirsizlik İlkesi ve kuantum durumlarının kopyalanamaması, bu alandaki temel güvenlik sağlayıcı prensipler olarak öne çık-maktadır. Özellikle bu prensipler, kuantum anahtar dağıtım sürecinde, dinleyicinin (Eve olarak adlandırılan saldırganın) sistemden bilgi çalmaya çalışması durumunda girişimin tespit edilebileceğini garanti eder. Bunun sebebi kuantum mekaniği prensiplerine göre, bir kübitin durumunu ölçmeye çalışmak, o durumu kaçınılmaz olarak bozar; bu bozulma iletişimin her iki tarafı (Alice ve Bob) tarafından fark edilir.

QKD'nin en bilinen protokollerinden biri olan BB84 protokolü, 1984 yılında Bennett ve Brassard tarafından önerilmiştir (Bennett ve Brassard, 2014). Bu protokol, kuantum bitlerinin dört farklı kuantum durumu üzerinden kodlanmasını temel alır ve bu durumlar iki farklı bazdan seçilir; böylelikle olası bir dinleyicinin hangi bazda ölçüm yapacağını bilememesi nedeniyle, saldırı girişimleri tespit edilebilir. Bir diğer önemli QKD protokolü olan Ekert91 (E91) protokolü ise, 1991 yılında Ekert tarafından geliştirilmiştir (Ekert, 1991). E91 protokolü,

Bell'in teoremine dayanan kuantum dolanıklık prensibini kullanarak anahtar dağıtımı gerçekleştirir. Bu protokol, BB84'e kıyasla daha karmaşık bir yapı sunmakla birlikte, kuantum dolanıklığın doğasında bulunan doğrulama mekanizmaları sayesinde, daha güçlü bir güvenlik altyapısı sağlar.

Kuantum şifrelemenin üç ana karakteri vardır; bunlar; Alice (gönderen), Bob (alıcı) ve Eve (dinleyici) iletişimleri genel olarak Şekil 2.1'de gösterildiği gibidir.



Şekil 2.1 Kuantum İletişim Örneği

Çift yönlü şifresiz kanal üzerinden Alice ve Bob'un kullandığı bazlar paylaşılır. Alice, Bob'a tek yönlü şifreli kanal aracılığıyla mesaj gönderir. Eve'nin amacı mesajları Alice ve Bob'un haberi olmadan okumaktır. Kuantum şifreleme protokolleri, kuantum mekaniğinin bazı temel ilkelerinden faydalanarak çalışır bunlar:

Süperpozisyon İlkesi: Kuantum mekaniğinde, bir sistem birden fazla durumla aynı anda sahip olabilir. Örneğin, bir kübit hem $|0\rangle$ hem de $|1\rangle$ durumlarında bulunabilir. Bir kübitin süperpozisyonu denklem (2.1)'de gösterildiği gibi ifade edilir.

$$\psi=\alpha 0+\beta 1 \tag{2.1}$$

Burada α ve β kompleks sayılar olup $|\alpha|^2 + |\beta|^2 = 1$ koşulu sağlanır.

Ölçüm İlkesi: Bir kuantum sistem ölçüldüğünde sistem süperpozisyonunu koruyamaz ve belirli bir durum alır. Bir kübitin ölçümü onu $|0\rangle$ ve $|1\rangle$ arasındaki durumlardan birini almaya zorlar.

Dolanıklık İlkesi: İki veya daha fazla kuantum sistem birbiriyle dolanık hale gelebilir bu noktada sistemler birbirlerinden bağımsız olarak ele alınamaz. Dolanıklık, kuantum şifreleme protokollerinin güvenliğinde elzemdir.

2.1.2. BB84 Protokolü

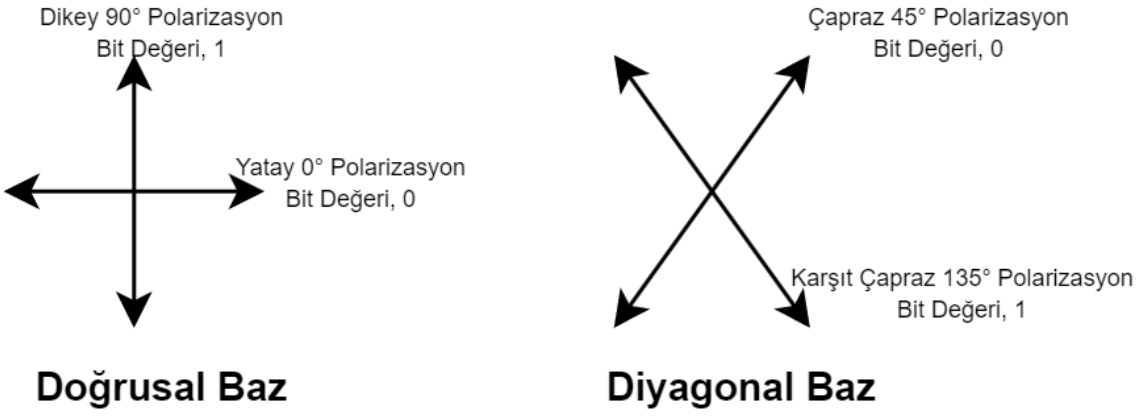
Bu protokol, 1984'te Charles Bennett ve Gilles Brassard tarafından tasarlanan ilk kuantum anahtar dağıtım (QKD) protokolüdür (Bennett ve Brassard, 2014). BB84 protokolü, rastgele bitlerden oluşan gizli anahtarın dağıtımı için tek foton kullanmaktadır, bu foton olası dört polarizasyon durumundan biriyle polarize edilir ve iki eşlenik bazdan biri seçilir. Bu bazlar, dikey yatay polarizasyon için doğrusal baz, çapraz ve karşıt-çapraz polarizasyon için diyagonal baz olarak adlandırılır. BB84 protokolünde kullanılan kuantum durumları Dirac notasyonuyla şu şekilde ifade edilir;

Doğrusal Baz:

$$|0\rangle=\begin{pmatrix}1\\0\end{pmatrix}, \qquad |1\rangle=\begin{pmatrix}0\\1\end{pmatrix} \tag{2.2}$$

Diyagonal Baz:

$$|+\rangle=\frac{1}{\sqrt{2}}\begin{pmatrix}1\\1\end{pmatrix}, \qquad |-\rangle=\frac{1}{\sqrt{2}}\begin{pmatrix}1\\-1\end{pmatrix} \tag{2.3}$$



Şekil 2.2 BB84 Protokolü İçin Bazlar ve Bit Değerleri

BB84 protokolünde iletişim örneği:

- Alice, 0 ve 1 bitlerinden oluşan bir bit dizisi seçer. Örneğin, 0, 1, 1, 0, 0, 1, 0, 1 dizisi

seçilsin.

- Alice, her bir bit için rastgele bir baz seçer. Doğrusal baz (ör. yatay ve dikey) şeklinde polarize eder. Benzer şekilde diyagonal baz seçimi de 0 ve 1 bitleri, sırasıyla $+45^{\circ}$ ve -45° ile şeklinde polarize edilir. Örneğin, ×, +, +, ×, +, ×, +, × bazları seçilsin.
- Alice, her bir bitini seçtiği baz kullanarak kuantum durumuna dönüştürür ve Bob'a gönderir.
- Bob, Alice'ten gelen her foton için rastgele bir baz seçer.
  Örneğin, +, +, +, ×, ×, +, ×, × bazları seçsin.
- Bob, seçtiği bazları kullanarak her bir fotonu ölçer ve elde ettiği bit değerlerini kaydeder. Eğer Bob'un seçtiği baz, Alice'in seçtiği baz ile aynıysa doğru bit değeri ölçülmüş olur.
- Alice ve Bob, seçtikleri bazları birbirleriyle paylaşır, bu bilgiyi güvenli bir kanal üzerinden paylaşırlar ve bit değerlerini paylaşmazlar.
- Alice ve Bob, aynı baz kullandıkları bitleri seçer ve bu bitleri güvenli anahtar olarak kullanır.

Tablo 2.1 BB84 Örnek Eşleme Tablosu: Baz/Bit Durumları, Süzme (Bit Koru) ve Elde Kalan Bitler

	1	2	3	4	5	6	7	8
Alice'in Bazları	+	×	×	×	+	×	+	×
Alice'in Bitleri	$\|0\rangle$	$\|-\rangle$	$\|1\rangle$	$\|+\rangle$	$\|0\rangle$	$\|-\rangle$	$\|0\rangle$	$\|-\rangle$
Bob'un Bazları	+	+	×	×	×	+	+	×
Bob'un Bitleri	$\|0\rangle$	$\|0\rangle$	$\|1\rangle$	$\|+\rangle$	$\|-\rangle$	$\|1\rangle$	$\|0\rangle$	$\|-\rangle$
Bit Koru	Evet	Hayır	Evet	Evet	Hayır	Hayır	Evet	Evet
Korunan Bitler	0		1	0			0	1

BB84 protokolünde güvenliğin en önemli ölçütlerinden biri, kuantum bit hata oranı (Quantum Bit Error Rate, QBER) olarak bilinen parametredir.

$$\mathrm{QBER}=\frac{N_{\mathrm{hata}}}{N_{\mathrm{toplam}}} \tag{2.4}$$

Burada N_{hata} hata sayısını, N_{toplam} ise toplam kübit sayısını temsil eder. QBER, gönderilen ve alınan kübitler arasındaki uyuşmazlıkları ölçer ve bu oran ne kadar düşükse, anahtarın güvenliği o kadar yüksek olur. QBER'i etkileyen unsurlar kuantum kanallarındaki

gürültü, fiziksel cihazlardaki kusurlar ve potansiyel saldırganların dinleme girişimleri olarak sayılabilir.

Yapılan çalışmalarda, QBER'in %11'e kadar olan değerlerinde BB84 protokolü güvenli olarak kabul edilmektedir (Watanabe, vd., 2006). Bunun ötesinde, hata oranının %25'in üzerine çıkması durumunda (Nikolopoulos ve Alber, 2004), (bazı yöntemlerle (Xu, vd., 2010) bu oran %17.7'ye kadar düşürülebilir) protokolün güvenliğinin ihlal edildiği varsayılarak %25 ve üzerindeki hata oranlarında bir saldırganın (Eve) kuantum kanallarına müdahale ettiğini kübitlerin büyük bir kısmını başarılı bir şekilde dinlediğini veya manipüle ettiğini ifade eder.

BB84 protokolünün anahtar oluşturma oranı (key rate), sistemin güvenli bir şekilde ne kadar veri üretebileceğini belirler. Bu oran, hata oranı ve kanaldaki diğer zorluklarla doğrudan ilişkilidir. Matematiksel olarak, güvenli anahtar oranı genellikle şu formülle ifade edilir:

$$R=Q\cdot[1-h(e)] \tag{2.5}$$

Burada R güvenli anahtar oranını, Q kuantum bitlerinin başarılı iletim oranını ve e hata oranını temsil eder. h(e) ise Shannon entropisi ile ilişkili bir fonksiyondur ve

$$h(e)=-e\log_2 e-(1-e)\log_2(1-e) \tag{2.6}$$

şeklinde tanımlanır. Bu formül, hata oranının artmasıyla güvenli anahtar oranının azaldığını gösterir (Sano, vd., 2010).

2.1.3. E91 Protokolü

E91 protokolü, 1991 yılında Artur Ekert tarafından önerilen ve kuantum mekaniğinin temel prensiplerine dayanan bir kuantum anahtar dağıtım (QKD) protokolüdür (Ekert, 1991). Bu protokol, BB84 gibi önceki protokollerden farklı olarak, kuantum dolanıklık ve Bell teoremi üzerine inşa edilmiştir. Alice ve Bob, güvenli bir anahtar oluşturmak için kuantum dolanık çiftleri kullanır. Bu dolanık çiftler, genellikle Bell durumları olarak bilinen özel kuantum süperpozisyon durumlarında bulunur. Dört Bell durumu şu şekilde tanımlanabilir:

$$|\psi^+\rangle=\frac{1}{\sqrt{2}}(|01\rangle+|10\rangle), \tag{2.7}$$

$$|\psi^-\rangle=\frac{1}{\sqrt{2}}(|01\rangle-|10\rangle), \tag{2.8}$$

$$|\phi^+\rangle=\frac{1}{\sqrt{2}}(|00\rangle+|11\rangle), \tag{2.9}$$

$$|\phi^-\rangle=\frac{1}{\sqrt{2}}(|00\rangle-|11\rangle), \tag{2.10}$$

Bu dört Bell durumu, iki kübitin birbirine dolanık olduğunu ifade eder ve bu durumlar arasındaki fark, kübitlerin aynı veya zıt durumlarda olmasına bağlıdır. E91 protokolünde genellikle $|\psi^-\rangle$ durumu kullanılır, ancak diğer Bell durumları da aynı prensiplere dayanarak kullanılabilir. Klasik sistemlerde, Bell parametresi S, maksimum 2 değerini alabilir ancak, kuantum dolanık durumlarda bu parametre, $2\sqrt{2}$ değerine kadar çıkabilir. Bell parametresi S, denklemle hesaplanır:

$$S = E(\theta_A,\theta_B) - E(\theta_A,\theta'_B) + E(\theta'_A,\theta_B) + E(\theta'_A,\theta'_B), \quad (2.11)$$

Burada, E (θ_A, θ_B) ifadesi, Alice ve Bob'un θ_A ve θ_B açılarında yaptıkları ölçümlerin korelasyonlarını temsil eder ve şu şekilde tanımlanır:

$$E(\theta_A,\theta_B) = P(\text{Alice} = \text{Bob}|\theta_A,\theta_B) - P(\text{Alice} \neq \text{Bob}|\theta_A,\theta_B). \quad (2.12)$$

P (Alice = Bob|θ_A, θ_B) Alice ve Bob'un aynı sonucu elde etme olasılığını temsil eder. Kuantum mekaniğine göre, Bell parametresinin S değeri şu aralıkta olacaktır:

$$2<|S|\leq 2\sqrt{2}. \quad (2.13)$$

E91 protokolünde iletişim adım adım şu şekilde gerçekleşir:

- Alice ve Bob, birbirlerine göndermek üzere kuantum dolanık kübit çiftleri alır. Bu dolanık kübitler genellikle Bell durumundadır. Yani, Alice'in kübiti bir durumda ölçüldüğünde,

Bob'un kübiti de buna göre anında belirlenir.

- Alice ve Bob, kübitleri hangi tabanda ölçüm yapacaklarını rastgele seçerler:

$$|0\rangle = \begin{pmatrix}1\\0\end{pmatrix}, \qquad |1\rangle = \begin{pmatrix}0\\1\end{pmatrix} \quad (2.14)$$

$$|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle+|1\rangle), \quad |-\rangle = \frac{1}{\sqrt{2}}(|0\rangle-|1\rangle), \quad (2.15)$$

- Alice ve Bob, seçtikleri tabanlarda ölçüm yapar. Eğer Alice ve Bob aynı tabanı seçmişse, kübitlerin ölçüm sonuçları tamamen korele olacaktır; Alice $|0\rangle$ ölçtüğünde Bob kesinlikle $|1\rangle$ ölçecektir. Korelasyon fonksiyonu deneyden toplanan verilerle hesaplanır.
- Alice ve Bob, klasik bir iletişim kanalı üzerinden hangi tabanları kullandıklarını karşılaştırır. Taban seçimlerini paylaştıktan sonra, aynı tabanı seçtikleri ölçümlerden kalanlar anahtar bitlerini oluşturur; farklı tabanlarda ölçülmüş sonuçlar atılır.

- Alice ve Bob, seçilmemiş sonuçlardan yararlanarak S parametresini hesaplayıp protokolün güvenliğini doğrular.
- S değeri uygun aralıkta ise, belirlenmiş ölçüm sonuçlarını birleştirerek bir ham anahtar oluştururlar. Aynı tabanda yapılan ölçümler anahtarı oluşturur; farklı tabanlarda alınan sonuçlar test bitleri olarak kullanılır.
- Hata düzeltme işlemlerinden ardından, Eve'in olası bilgi kazanımını minimuma indirmek ve ham anahtarın güvenliğini artırmak için gizlilik güçlendirme işlemleri uygulanır.

Bilgi kaybı tahmini adı verilen süreçte Von Neumann Entropisi ve karşılıklı bilgi gibi matematiksel modeller ve Bell eşitsizliği kullanılarak Eve'in bilgi kazanımı ölçülür. Anahtarı daha kısa fakat daha güvenli hale getirmek içinse evrensel bir karma fonksiyonu kullanılır. Örneğin, n bit uzunluğunda bir ham anahtar için Eve'in en fazla m bit bilgiye sahip olacağı tahmin ediliyorsa, Alice ve Bob $n - m$ bit uzunluğunda bir güvenli anahtar elde ederler (Scarani vd., 2009). Matematiksel olarak, evrensel bir karma fonksiyon $h(x)$ aşağıdaki gibi ifade edilebilir:

$$h(x) = \left(h_1(x), h_2(x), \ldots, h_k(x)\right) \tag{2.16}$$

- Gizlilik güçlendirme işlemi sonrasında Alice ve Bob güvenli bir son anahtara sahip olurlar. Bu anahtar; tek kullanımlık şifre (OTP) ya da AES/DES gibi simetrik şifreleme algoritmalarında kullanılabilir.
- Eğer ilk turda yeterince uzun ve güvenli bir anahtar elde edilemezse, Alice ve Bob protokolü tekrar ederler. Bu, kuantum kanalındaki gürültü seviyesine ve hata oranlarına bağlı olarak birden fazla kez gerekebilir. Protokol tekrarlanarak, nihai güvenli anahtar uzunluğu artırılır.

Bu adımların her biri, E91 protokolünün güvenliğini sağlamak için gereklidir. Bu süreçte kullanılan matematiksel araçlar, klasik kriptografinin ötesine geçerek kuantum mekaniğinin prensiplerinden yararlanır ve bu sayede çok daha güçlü bir yapı inşa eder.

BB84 protokolünde olduğu gibi E91 protokolü için de bir kuantum bit hata oranı (QBER) tanımlanır; BB84 protokolünden farklı olarak, iki dolanık fotondan elde edilen ölçümlere ilişkin rastlantısal olaylar incelenir. Rastlantı olasılığı $P_{\text{rastlantı}}$ iki parçaya ayrılır. $P_{\text{asıl}}$,

dolanık durumdaki bir çift fotondan kaynaklanan gerçek rastlantı olasılığını ifade ederken, P_{sahte} ise ideal bir kaynaktan yayılan fotonlar ile fononlar gibi mekanik kaynaklı sayım nedeniyle oluşabilecek sahte rastlantı olasılığını ifade eder:

$$P_{\text{rastlantı}}=P_{\text{asıl}}+P_{\text{sahte}} \tag{2.17}$$

ve QBER şöyle ifade edilir (Waks, 2002):

$$\text{QBER}=\frac{P_{\text{sahte}}/2+\mu\, P_{\text{asıl}}}{P_{\text{rastlantı}}} \tag{2.18}$$

E91 protokolünün bir diğer kritik yönü, güvenli anahtar oluşturma oranıdır (key rate). Bu oran, protokolün güvenli bir şekilde ne kadar veri üretebileceğini belirler ve QBER ile doğrudan ilişkilidir. Matematiksel olarak, güvenli anahtar oluşturma oranı aşağıdaki gibi ifade edilir:

$$R=S\times(1-2\times\text{QBER}) \tag{2.19}$$

Burada R güvenli anahtar oranını, S ise Bell eşitsizliğini kullanarak elde edilen verimliliği temsil eder. Bu oran, hata oranının artması durumunda güvenli anahtar oluşturmanın azalacağını gösterir. Ancak, Bell eşitsizliklerinin doğru bir şekilde test edilmesi ve hata oranlarının etkin bir şekilde yönetilmesi, E91 protokolünün güvenli bir şekilde çalışmasını sağlar.

2.2. BB84 Protokolü İçin Python Tabanlı İletişim Simülasyonu

BB84 protokolü, Python ile vektörize edilmiş bir simülasyon üzerinden incelenmiştir. Kuantum kanal modeli, depolarizasyon (p_{dep}) ve kayıp (p_{kayip}) etkilerini kapsayacak şekilde basitleştirilmiştir. Dinleyici (Eve) için intercept-resend saldırı modeli etkinleştirdi (Eve açık).

- Alice, n adet rasgele bit ve karşılık gelen baz (Z veya X) seçer.
- Bob, her deneme için bağımsız bir baz seçer.
- Kuantum kanal: p_{kayip} ile tespit yok; p_{dep} ile aynı bazda ölçümlerde hata (yaklaşık) $p_{\text{dep}}/2$.
- Baz eleme (Sifting): Alice ve Bob bazları eşleşen ve tespit edilen denemeler anahtar adaylarını oluşturur.
- Parametre tahmini: Sifted anahtarın belirli bir oranı örneklenerek QBER tahmini yapılır.
- Hata düzeltme sızıntısı basit modelle $\text{leak}_{ec} = 1.15\ h(Q)$ alındı.
- Gizlilik yükseltme: Evrensel karma ailesi ile anahtar kısaltma uygulanacağı varsayılır; bu simülasyonda yalnızca uzunluk hesabı yapılmıştır.

2.2.1 Formüller

İkili entropi fonksiyonu

$$h(Q)=- Q\log_2 Q - (1-Q)\log_2(1-Q). \tag{2.20}$$

Asimptotik gizli anahtar oranı

Baz eleme oranı $q = \frac{|Ksifted|}{n}$ olmak üzere,

$$R\approx q[\, 1-h(Q)-leak_{ec}\,], \qquad leak_{ec}=1.15\, h(Q). \tag{2.21}$$

Pratikte anahtar uzunluğu $|K_{gizli}|\approx|K_{sifted}|\cdot\max 0,\ 1-h(Q)-1.15\, h(Q)$ olarak alınmıştır.

Sonlu-anahtar kısa notu

Güven düzeyi ε için sonlu örneklem etkileri, parametre tahmini örnek sayısı ve istatistiksel sapma terimleri üzerinden anahtar oranını düşürür. Bu çalışmada asimptotik yaklaşım kullanılmış olup, sonlu-anahtar düzeltmeleri yalnızca tartışma bölümün de nitel olarak ele alınmıştır (Scarani vd., 2009).

2.2.2 Simülasyon Yapılandırması

Tohum: np.random.default_rng(42).

Python: 3.10+.

Paketler: numpy, pandas, matplotlib.

BB84 Protokolü için Python Tabanlı İletişim Simülasyonu Sözde Kodu

ALGORİTMA 16: ALİCE BİT HAZIRLIĞI

Parametreler: N, tohum

Çıktı: (bit_A, baz_A)

bit_A ← uzunluk *N* rastgele 0/1 dizi (tohum opsiyonel olarak kullanılır)

baz_A ← uzunluk *N* rastgele baz dizi (Z veya X)

return (bit_A, baz_A)

ALGORİTMA 17: DİNLEYİCİ (EVE) KONTROLÜ

Parametreler: bit, baz

Çıktı: ($\hat{b}$, eve_baz)

eve_baz ← rastgele {Z, X}

if eve_baz = baz **then** $\hat{b}$ ← bit

else $\hat{b}$ ← rastgele {0, 1}

4 **return** ($\hat{b}$, eve_baz)

ALGORİTMA 18: KANAL VE BOB BAZLARI

Parametreler: b_0, z_0, p_{kayip}, p_{dep} eve_aktif, bob_baz

Çıktı: ölçüm sonucu b* veya None

if eve_aktif **then**
  (b, z) ← 19(b_0, z_0)
else
  (b, z) ← (b_0, z_0)
if rastgele_olasılık() < p_{kayip} **then**
  return None
if bob_baz = z **then**
  b* ← b
  if rastgele_olasılık() < p_{dep} **then**
    b* ← 1 – b*
  return b*
else
  return rastgele {0, 1}

ALGORİTMA 19: BOB ÖLÇÜM AŞAMASI

Parametreler: bit_A, baz_A, p_{kayip}, p_{dep}, eve_aktif

Çıktı: (baz_B, olcum_B, kayip)

N ← |bit_A| ; baz_B ← uzunluk N rastgele {Z, X}
olcum_B ← uzunluk N boş dizi ; kayip ← uzunluk N **False**
for i ← 1 **to** N **do**
s ← 20(bit_A[i], baz_A[i], p_{kayip}, p_{dep}, eve_aktif, baz_B[i])
if s = None **then** kayip[i] ← **True**
else olcum_B[i] ← s
return (baz_B, olcum_B, kayip)

ALGORİTMA 20: SÜZME (SİFTİNG) BÖLÜMÜ

Parametreler: bit_A, baz_A, baz_B, olcum_B, kayip

Çıktı: (alice_suz, bob_suz, I)

I ← { i : ¬kayip[i] **ve** baz_A[i] = baz_B[i] }

2 alice_suz ← [bit_A[i] : i ∈ I], bob_suz ← [olcum_B[i] : i ∈ I]

3 **return** (alice_suz, bob_suz, I)

ALGORİTMA 21: QBER TAHMİNİ VE ÖRNEK

Parametreler: alice_suz, bob_suz, ρ, tohum

Çıktı: (Q, alice_kalan, bob_kalan)

1 n ← |alice_suz| ; m ← max(1, ⌊ρn⌋)

2 J ← tohumla rastgele m indeks

3 hatalar ← |{ j ∈ J : alice_suz[j] ≠ bob_suz[j] }| ; Q ← hatalar / m

4 K ← {0, …, n–1} \ J

5 alice_kalan ← [alice_suz[k] : k ∈ K], bob_kalan ← [bob_suz[k] : k ∈ K]

6 **return** (Q, alice_kalan, bob_kalan)

ALGORİTMA 22: HATA DÜZELTME (SOYUT SIZINTI MODELİ)

Parametreler: alice_kalan, bob_kalan, Q, leak_ec_faktor

Çıktı: (bob_duz, leak_ec)

1 n ← |alice_kalan| ; leak_ec ← ⌊leak_ec_faktor · n · h_2(Q)⌋

2 bob_duz ← alice_kalan

3 **return** (bob_duz, leak_ec)

ALGORİTMA 23: GİZLİLİK YÜKSELTME (PRIVACY AMPLIFICATION)

Parametreler: alice_duz, Q, leak_ec, k

Çıktı: (final_anahtar, L)

1 n ← |alice_duz| ; R ← max{0, 1 – 2·h_2(Q)}

2 L ← ⌊max{0, nR – leak_ec – k}⌋

3 **if** L ≤ 0 **then**

4 | **return** ([ ], 0)

5 final_anahtar ← evrensel karma (Toeplitz)(alice_duz, L)

6 **return** (final_anahtar, L)

ALGORİTMA 24: BB84 ALGORİTMASI ANA AKIŞI (ALT ALGORİTMALAR İLE BİRLİKTE)

Parametreler: N, p_{kayip}, p_{dep}, eve_aktif, ρ, leak_ec_faktor, k, tohum

Çıktı: final_anahtar, metrikler

1 (bit_A, baz_A) ← 18(N, tohum)

2 (baz_B, olcum_B, kayip) ← 21(bit_A, baz_A, p_{kayip}, p_{dep}, eve_aktif)

3 (alice_suz, bob_suz, I) ← 22(bit_A, baz_A, baz_B, olcum_B, kayip)

4 (Q, alice_kalan, bob_kalan) ← 23(alice_suz, bob_suz, ρ, tohum)

5 (bob_duz, leak_ec) ← 24(alice_kalan, bob_kalan, Q, leak_ec_faktor)

6 (final_anahtar, L) ← 25(bob_duz, Q, leak_ec, k)

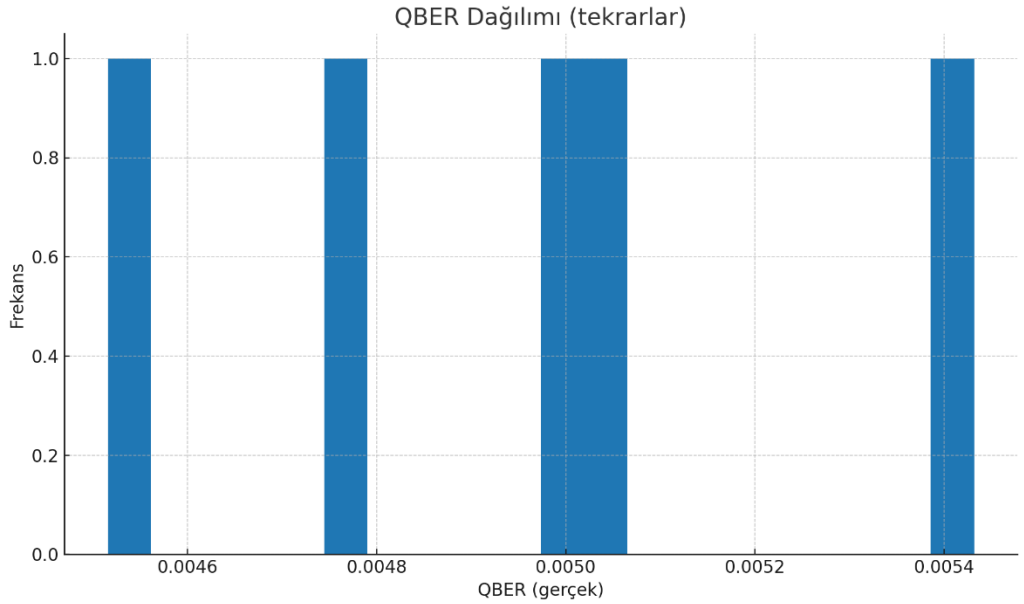
7 **Metrikler:** N, |alice_suz|, |alice_kalan|, Q, leak_ec, L, L/N, p_{kayip}, p_{dep}, eve_aktif

---

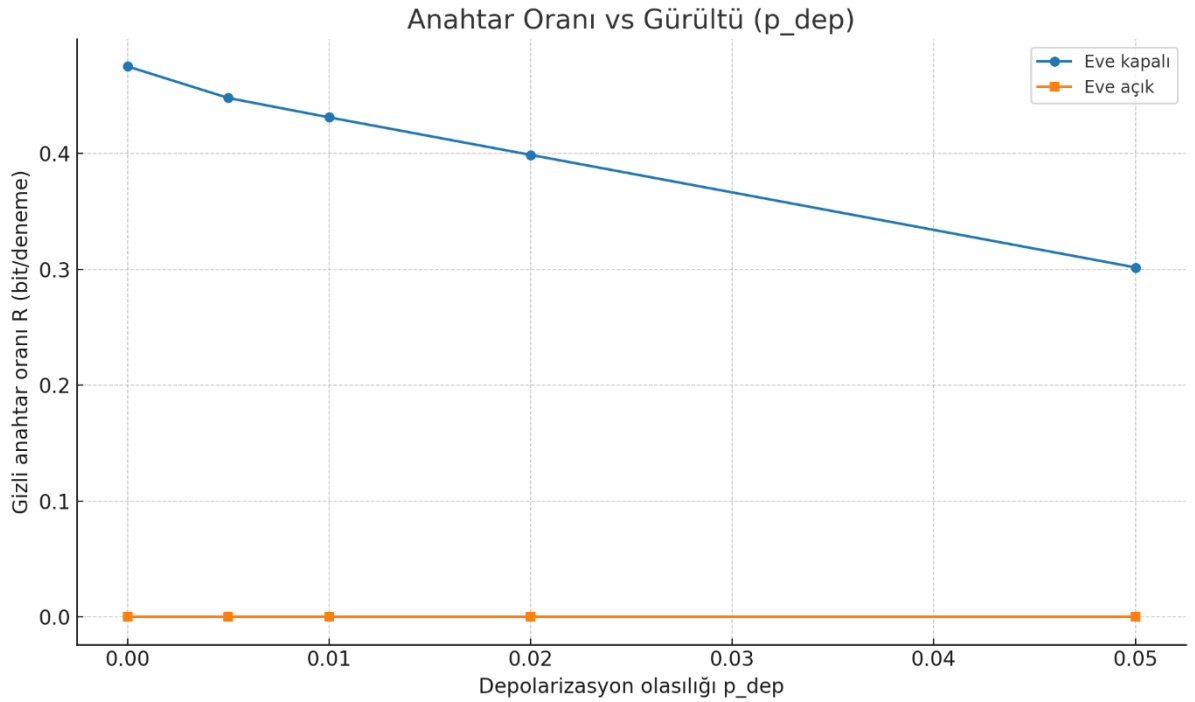
Parametre aralıkları:

- $n \in \{10^5, 2 \times 10^5\}$,
- $p_{dep} \in \{0, 0.005, 0.01, 0.02, 0.05\}$,
- $p_{kayip} \in \{0, 0.02, 0.05, 0.1\}$,
- Eve $\in \{0, 1\}$,
- Her noktada en az 5 tekrar.

2.2.3 Bulgular



Şekil 2.3 Tekrarlar üzerinden QBER dağılımı. Eve açıkken QBER artışı beklenir; histogram bu farkı görünür kılar.



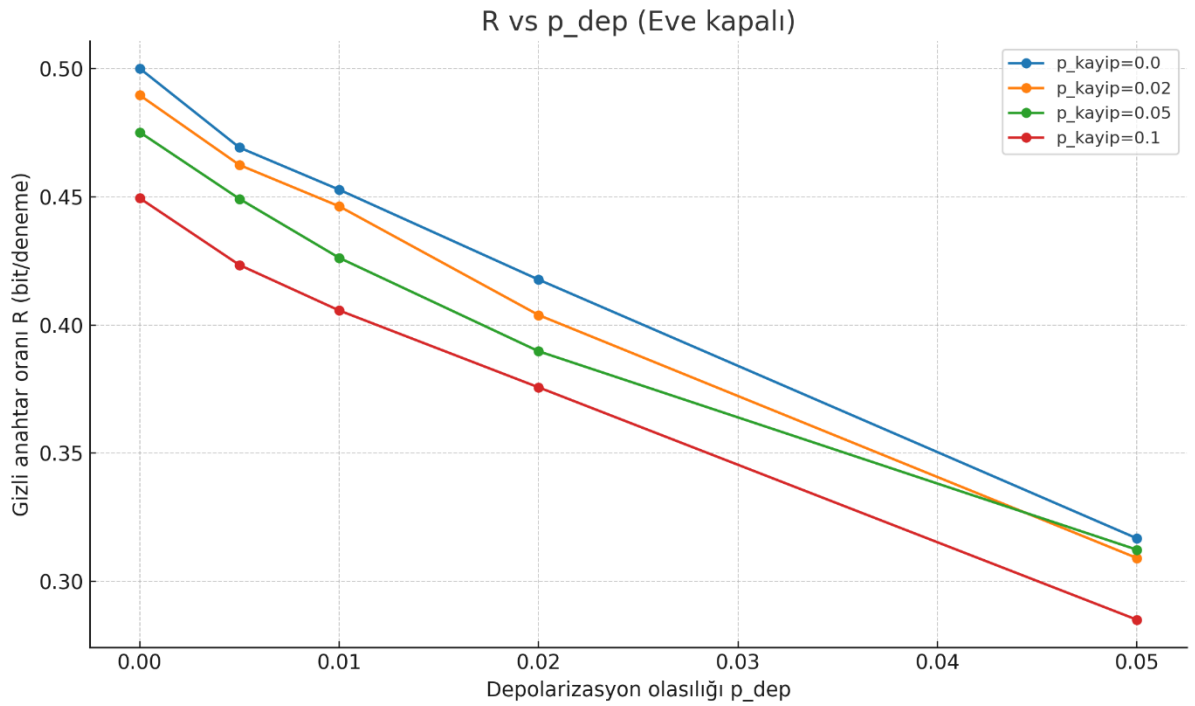
Şekil 2.4 Gürültü (p_{dep}) arttıkça gizli anahtar oranı R'nin düşmesi. Eşik civarında R ≈ 0 olur.

Tablo 2.2 Örnek Özet Metrikler

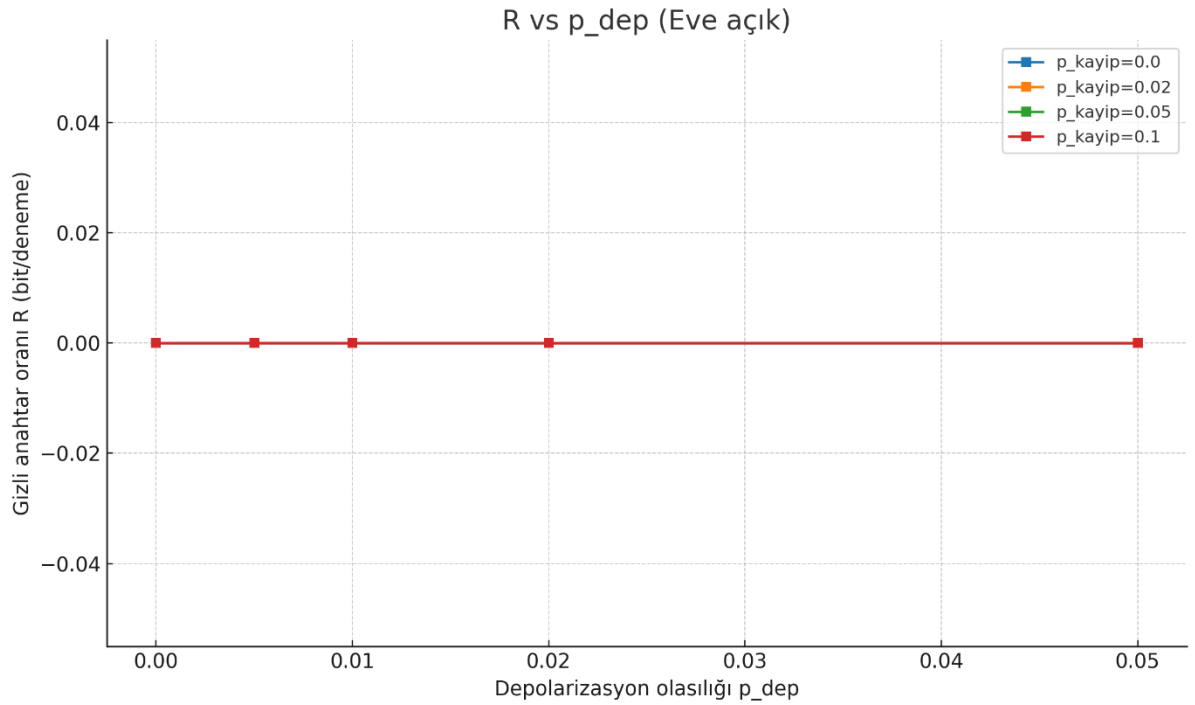
Büyüklük	Ortalama	Standart Sapma
QBER (gerçek)	0.004954	0.000341
Sifted uzunluğu	95079.8	150.0
R (bit/deneme)	0.428952	0.002971

2.2.4 Hassasiyet Analizi

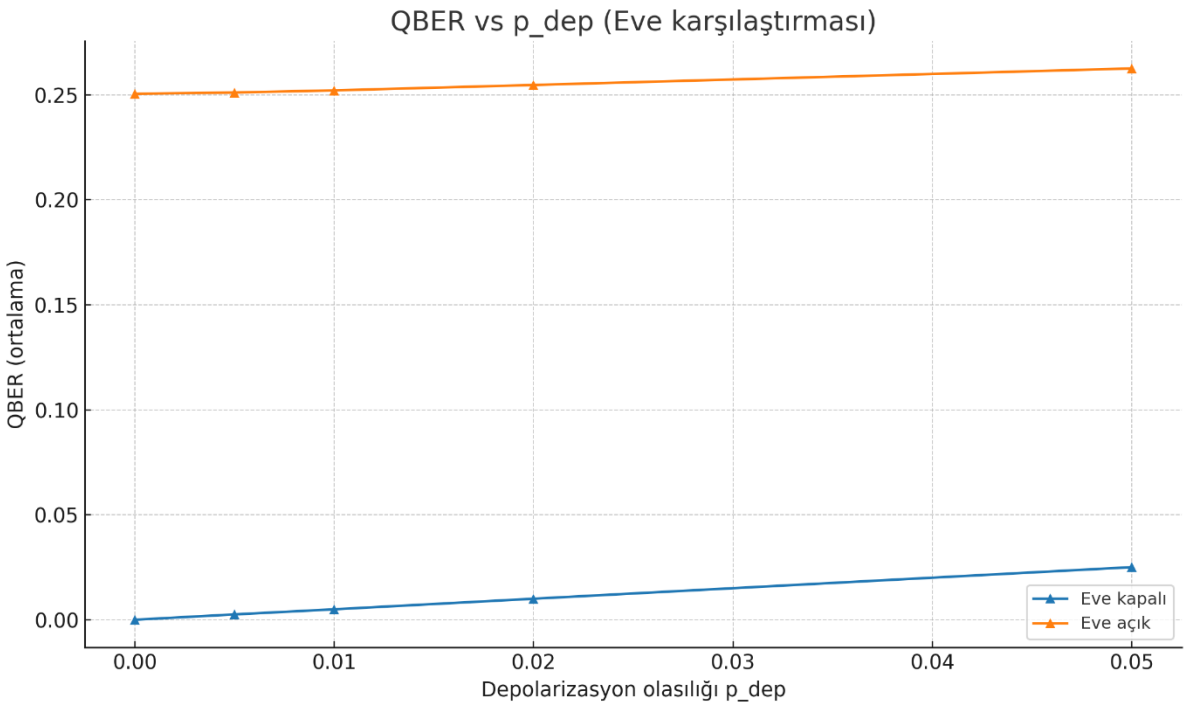
Bu alt bölümde hassasiyet analizi gerçekleştirilmiştir. Parametre ızgarası $n \in 10^5, 2\times10^5$, $p_{dep} \in 0, 0.005, 0.01, 0.02, 0.05$, $p_{kayip} \in 0, 0.02, 0.05, 0.1$ ve Eve $\in 0,1$ olarak seçilmiş; her noktada en az 5 tekrar ile, sabit tohum (42) kullanılmıştır. Gürültü ve kaybın artmasıyla R'nin (gizli anahtar oranı) beklenen şekilde azaldığı, dinleyici (Eve) açıkken eğrilerin aşağı kaydığı gözlenmiştir.



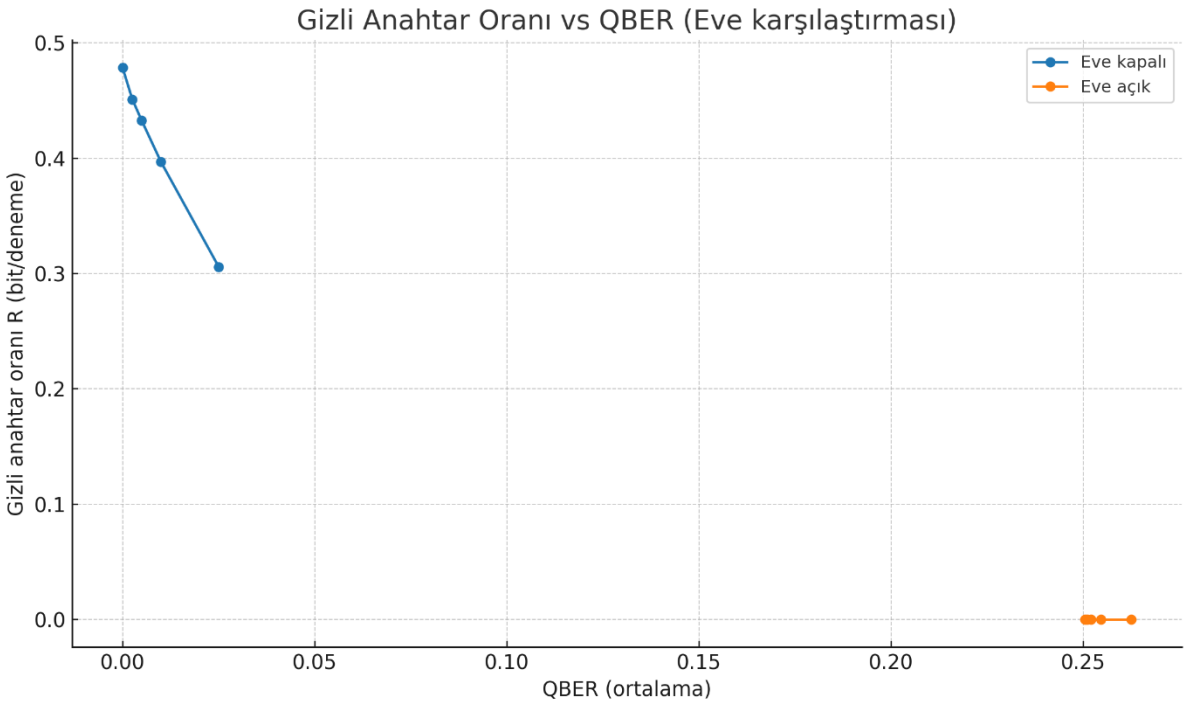
Şekil 2.5 Eve kapalıyken R'nin p_{dep}'e bağlı değişimi; farklı p_{kayip} değerleri için ortalama değerler.



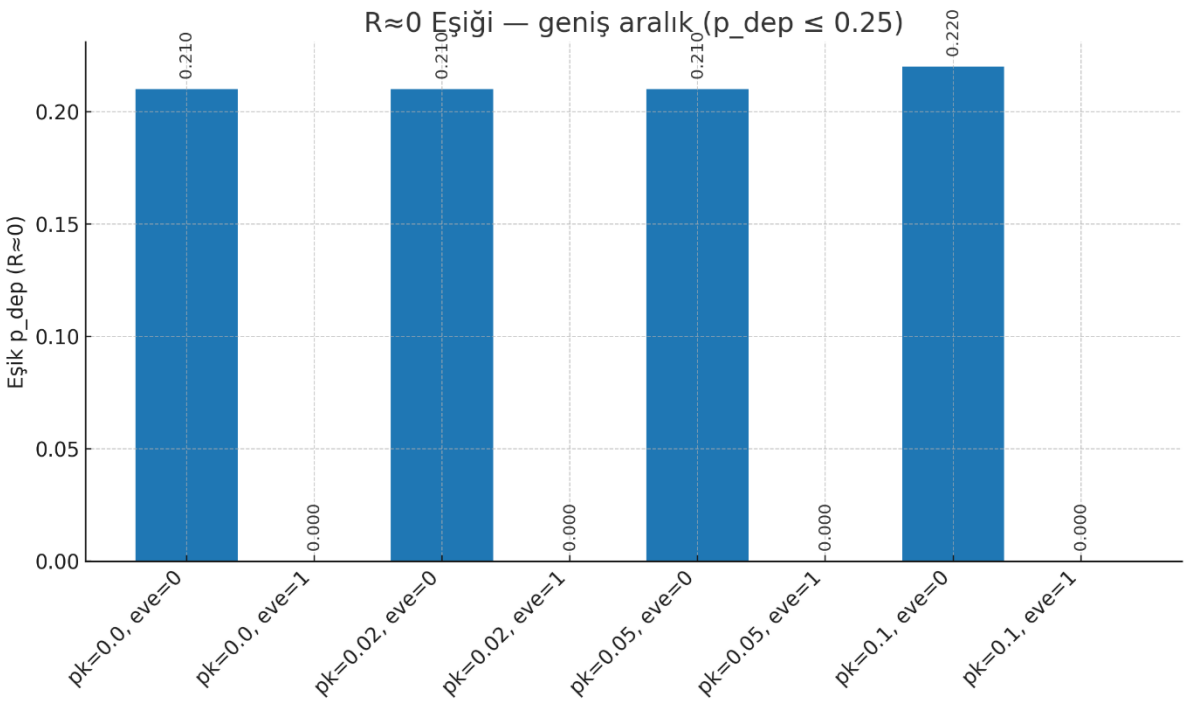
Şekil 2.6 Eve açıkken R'nin p_{dep}'e bağlı değişimi; farklı p_{kayip} değerleri için ortalama değerler.



Şekil 2.7 QBER'in p_{dep}'e bağlı karşılaştırması (Eve açık/kapalı). Dinleyici varlığında QBER artmaktadır.



Şekil 2.8 R ile QBER arasındaki ilişki (Eve açık/kapalı). QBER yükseldikçe R monoton biçimde düşer.



Şekil 2.9 R ≈ 0 eşiğini veren p_{dep} değerleri (çubuk grafiği), p_{kayip} ve Eve durumuna göre.

2.2.5 Bulguların Tartışılması

Asimptotik model $R \approx q\,[1\text{-}h(Q)\text{-}leak_{ec}]$ ve $leak_{ec}=1.15\,h(Q)$ altında, deney ızgarası üzerinden gözlenen temel eğilimler şöyledir:

(i) Kayıp p_{kayip} arttıkça sifting oranı $q \simeq 1/2\left(1\text{-}p_{kayip}\right)$ ile ölçeklenerek düşer;

(ii) Depolarizasyon p_{dep} arttıkça QBER Q artar ve R monoton azalır;

(iii) Eve açık senaryosunda Q≈25% bandına yerleşerek R'yi asimptotik sınırda sıfırlar (Scarani, vd., 2009)

Tablo 2.3, seçili $\left(p_{kayip},p_{dep}\right)$ çiftleri için sayısal özetleri sunar. Eve kapalı durumda test aralığımızda ($p_{dep}\leq 0.05$) R>0 korunurken, Eve açıkken tüm aralıkta R≈0 elde edilmiştir. Bu davranış, BB84'te intercept-resend saldırısının sifted anahtarda ~25% QBER ürettiğine dair klasik sonucu doğrular (Bennett ve Brassard, 2014; Scarani, vd., 2009). Ayrıca q'nun ölçülen değerlerinin, beklenti $q_{bek}=1/2\left(1\text{-}p_{kayip}\right)$ ile $\sim 10^{-3}$ mertebesinde uyumlu olduğu görülmüştür.

Sonuçlar, pratik hata düzeltme verimliliği f_{ec} iyileştikçe (ör. 1.15 → 1.05) *R*'nin artacağını; öte yandan sonlu-anahtar düzeltmelerinin (güven düzeyi ε) R'yi aşağı çekeceğini

ima eder. (Tomamichel, vd., 2012; Devetak ve Winter, 2005) Decoy-State yaklaşımıyla çoklu foton olaylarının ayrıştırılması, yüksek kayıp rejiminde güvenli oranları iyileştirmenin bilinen bir yoludur (Lo, vd., 2005; Hwang, 2003).

Bu model, optik kaynak ve dedektör düzeyi ayrıntılarını içermemektedir. Depolarizasyonun hata modeli ve intercept-resend saldırısı idealize edilmiştir. Gelecekte Decoy-State (μ-weak/strong) iskeleti genişletilerek foton sayısı istatistiklerine dayalı anahtar oranı analizi eklenebilir.

Tablo 2.3 Seçili parametre çiftleri için ortalama ± std metrikler (n = 2×10^5). q := $\frac{|\text{Ksifted}|}{\text{n}}$.

p_{kayip}	p_{dep}	Eve	q	QBER (ort±std)	R (ort±std)
0.02	0.000	Kapalı	0.4895	0.0000 ± 0.0000	0.4895 ± 0.0005
0.02	0.000	Açık	0.4903	0.2507 ± 0.0008	0.0000 ± 0.0000
0.02	0.010	Kapalı	0.4896	0.0050 ± 0.0003	0.4426 ± 0.0035
0.02	0.010	Açık	0.4902	0.2525 ± 0.0010	0.0000 ± 0.0000
0.02	0.050	Kapalı	0.4910	0.0249 ± 0.0004	0.3123 ± 0.0077
0.02	0.050	Açık	0.4897	0.2621 ± 0.0020	0.0000 ± 0.0000
0.05	0.000	Kapalı	0.4750	0.0000 ± 0.0000	0.4750 ± 0.0006
0.05	0.000	Açık	0.4735	0.2514 ± 0.0012	0.0000 ± 0.0000
0.05	0.010	Kapalı	0.4746	0.0049 ± 0.0002	0.4301 ± 0.0073
0.05	0.010	Açık	0.4741	0.2518 ± 0.0014	0.0000 ± 0.0000
0.05	0.050	Kapalı	0.4745	0.0248 ± 0.0006	0.3118 ± 0.0038
0.05	0.050	Açık	0.4746	0.2622 ± 0.0015	0.0000 ± 0.0000

3. SONUÇ VE GELECEK ÇALIŞMALAR

Bu tez, klasik şifrelemenin yapısal ilkelerini (karışıklık/difüzyon, indirgeme temelli güvenlik) ile kuantum kriptografinin bilgi kuramsal güvenliğini tek bir çerçevede ele alarak, kavramsal sürekliliği ve ayrışma noktalarını açıkça ortaya koymuştur. Klasik tarafta Vigenère, Feistel ağı ve DES üzerinden cebirsel/mimari ilkeler sistematikleştirilmiş; kuantum tarafta quantum key distribution (QKD) ailesinin tipik temsilcileri olan BB84 ve E91 protokolleri ölçüm, baz eleme (sifting), hata düzeltme ve gizlilik yükseltme adımları açısından karşılaştırılmıştır. Bu bütüncül yaklaşım, hesaplamalı güvenlik ile bilgi kuramsal güvenlik arasındaki farkı uygulamalı örneklerle somutlaştırmıştır.

Yöntemsel/Kavramsal Katkılar

Klasik şifreleme mimarileri ile QKD akışının ortak güvenlik dilinde formülasyonu, okurun iki paradigma arasında yerelleştirilebilir kavram eşleşmeleri kurmasına olanak sağlamıştır (ör. Feistel'deki difüzyon-BB84'teki rastgele baz seçimi ve sifting ile işaret-gürültü yönetimi). Simülasyonda, kanal parametreleri (p_{dep}, p_{kayip}) ve saldırı anahtarlarının (Eve açık/kapalı) etki profilleri ayrı ayrı izlenerek tasarım uzayı görselleştirilmiş bu sayede sistem tasarımının iyileştirilebileceği nicel olarak ayrıştırılmıştır. Uygulamalı güvenlik bağlamında, asimptotik sınırların yanında sonlu-anahtar rejiminin önemine dikkat çekilmiş örneklem büyüklüğündeki azalmaların (kayıp ve sifting etkisi) istatistiksel belirsizlik yoluyla R üzerinde ikinci dereceden fakat kritik bir rol oynadığı gösterilmiştir.

Gelecek Çalışmalar

Decoy-State BB84: Zayıf koherent kaynak altında tek-foton kesitlenmesini modele eklemek, photon number splitting (PNS) saldırılarına karşı güvenlik kanıtını simülasyonla birleştirmek (Hwang, 2003; H. Lo, vd. 2005; Wang, 2005).

Sonlu-anahtar Optimizasyonu: Güven düzeyi ε için smooth min-entropy tabanlı anahtar oranlarını ve istatistiksel sapma terimlerini (Serfling/Hoeffding) entegre etmek; örneklem-anahtar ayrımını optimizasyonla belirlemek (Elkouss, vd., 2010; Martinez-Mateo, vd., 2014; Yang, vd., 2017).

Cihaz modellemeli Güvenlik: Dedektör verimi, karanlık sayım ve zamanlama

sapmalarını içeren measurement device modeli; MDI-QKD varyantlarıyla deneysel gerçekçiliği artırmak (Lo, vd., 2012; Liu, vd., 2013).

Özetle, çalışma BB84'ün gürültü ve saldırı koşulları altındaki davranışını hem kuramsal sınırlar hem de simülasyon verileriyle uyumlu biçimde ortaya koymuş; pratik bir QKD hattının tasarımında en yüksek kaldıraç etkisine sahip parametreleri (hata düzeltme verimliliği, kayıp yönetimi, örneklem stratejisi) belirginleştirmiştir. Belirlenen sınırlılıkların giderilmesi ve önerilen yol haritasının izlenmesi, bilgi kuramsal güvenliğin sahadaki uygulanabilirliğini ölçülebilir biçimde güçlendirecektir.

KAYNAKÇA

Bayram, M., & Ilgaz Çağlayan, E. (2024). Comparative Analysis of BB84 and E91 Quantum Cryptography Protocols. *Kocaeli Science Congress 2024 (KOSC-2024)*, 104-109. doi:https://doi.org/10.5281/zenodo.14316751

Bennett, C. H., & Brassard, G. (2014). Quantum cryptography: Public key distribution and coin tossing. *Theoretical computer science*, 7-11.

Biham, E., & Shamir, A. (1991). Differential cryptanalysis of DES-like cryptosystems. *Journal of CRYPTOLOGY*.

Boneh, D. v. (2020). *A Graduate Course in Applied Cryptography.*

Coppersmith, D. (1994). The Data Encryption Standard (DES) and its strength against attacks. *IBM journal of research and development*.

Devetak, I., & Winter, A. (2005). Distillation of secret key and entanglement from quantum states. *Proceedings of the Royal Society A: Mathematical, Physical and engineering sciences*, 207-235.

Ekert, A. K. (1991). Quantum cryptography based on Bell's theorem. *Physical review letters*, 661.

Elkouss, D., Martinez-Mateo, J., & Martin, V. (2010). Information reconciliation for quantum key distribution. *arXiv preprint arXiv:1007.1616*.

Feistel, H. (1973). Cryptography and computer privacy. *Scientific american*.

Goldreich, O. (2004). *Foundations of Cryptography, Volume 2: Basic Applications.* Cambridge University Press.

Hwang, W.-Y. (2003). Quantum key distribution with high loss: toward global secure communication. *Physical review letters*.

Kahn, D. (1996). *The CodeBreakers – The Story of Secret Writing.*

Kasiski, F. W. (1863). *Die Geheimschriften und die Dechiffrir-Kunst. Mit besonderer Berücksichtigung.* ES Mittler und sohn.

Katz, J. v. (2020). *Introduction to Modern Cryptography.* CRC Press.

Liu, Y., Chen, T.-Y., Wang, L.-J., Liang, H., Shentu, G.-L. (2013). Experimental measurement-device-independent quantum key distribution. *Physical review letters*.

Lo, H.-K., Curty, M., & Qi, B. (2012). Measurement-device-independent quantum key distribution. *Physical review letters*.

Lo, H.-K., Ma, X., & Chen, K. (2005). Decoy state quantum key distribution. *Physical review letters*.

Martinez-Mateo, J., Pacher, C., Peev, M., Ciurana, A., & Martin, V. (2014). Demystifying the information reconciliation protocol cascade.

Menezes, A. J., Oorschot, P. C., & Vanstone, S. A. (1996). *Handbook of Applied Cryptography.* CRC Press.

Menezes, A. J., Van Oorschot, P. C., & Vanstone, S. A. (2018). Handbook of applied cryptography.

Nikolopoulos, G. M., & Alber, G. (2004). Robustness of the BB84 quantum key distribution protocol against general coherent attacks.

NIST. (1999). Data Encryption Standard (DES). https://csrc.nist.gov/pubs/fips/46-3/final

Niven, I. Z. (2004). *An Introduction to the Theory of Numbers.*

Pirandola, S., Andersen, U. L., Banchi, L., Berta, M., Bunandar, D., Colbeck, Ottaviani, C. (2020). Advances in quantum cryptography. *Advances in optics and photonics*.

Preskill, J. (2018). Quantum {C}omputing in the {NISQ} era and beyond. *{Quantum}*, 79. doi:10.22331/q-2018-08-06-79

Sano, Y., Matsumoto, R., & Uyematsu, T. (2010). Secure key rate of the BB84 protocol using finite sample bits. *Journal of Physics A: Mathematical and Theoretical*.

Scarani, V., Bechmann-Pasquinucci, H., Cerf, N. J., Dusek, M., Lutkenhaus, N., Peev, M. (2009). The security of practical quantum key distribution. *Reviews of modern physics*, 1301-1350.

Shannon, C. E. (1948). A mathematical theory of communication. *The Bell system technical journal*.

Standard, D. E. (1977). *Federal information processing standards publication.* US Department of Commerce.

Stinson, D. R. (2005). *Cryptography: theory and practice.* CRC.

Stinson, D. R. (2018). *Cryptography: Theory and Practice.* CRC Press.

Tomamichel, M., Lim, C. C., Gisin, N., & Renner, R. (2012). Tight finite-key analysis for quantum cryptography. *Nature communications*, 634.

Waks, E. a. (2002). Waks, Edo; Zeevi, Assaf; Yamamoto, Yoshihisa. *Physical Review A*.

Wang, X.-B. (2005). Beating the photon-number-splitting attack in practical quantum cryptography. *Physical review letters*.

Watanabe, S., Matsumoto, R., & Uyematsu, T. (2006). Noise tolerance of the BB84 protocol with random privacy amplification. *International Journal of Quantum Information*, 935-946.

Webster, A. F., & Tavares, S. E. (1985). On the Design of S-Boxes.

Xu, F., Qi, B., & Lo, H.-K. (2010). Experimental demonstration of phase-remapping attack

in a practical quantum key distribution system. *New Journal of Physics*.

Yang, S.-S., Bai, Z.-L., Wang, X.-Y., & Li, Y.-M. (2017). FPGA-based implementation of size-adaptive privacy amplification in quantum key distribution. *IEEE Photonics Journal*.