T.C.
ÜSKÜDAR ÜNİVERSİTESİ
BAĞIMLILIK VE ADLİ BİLİMLER ENSTİTÜSÜ

Danışman
Prof.Dr. Serhat ÖZEKES

SAYISAL VERİ VE DELİLLER İÇİN KÜP TABANLI ÇOK BOYUTLU ASİMETRİK ŞİFRELEME YÖNTEMİ

ADLİ BİLİMLER ANABİLİM DALI
ADLİ BİLİŞİM VE DİJİTAL DELİLLER BİLİM DALI
YÜKSEK LİSANS TEZİ

Burak BAYSAN

İSTANBUL – 2019

T.C.
ÜSKÜDAR ÜNİVERSİTESİ
BAĞIMLILIK VE ADLİ BİLİMLER ENSTİTÜSÜ

Danışman
Prof.Dr. Serhat ÖZEKES

SAYISAL VERİ VE DELİLLER İÇİN KÜP TABANLI ÇOK BOYUTLU ASİMETRİK ŞİFRELEME YÖNTEMİ

ADLİ BİLİMLER ANABİLİM DALI
ADLİ BİLİŞİM VE DİJİTAL DELİLLER BİLİM DALI
YÜKSEK LİSANS TEZİ

Burak BAYSAN

İSTANBUL – 2019

İÇİNDEKİLER

BİLİMSEL ETİK SAYFASI

Yüksek Lisans Tezi olarak sunduğum "Sayısal Veri ve Deliller İçin Küp Tabanlı Çok Boyutlu Asimetrik Şifreleme Yöntemi" adlı çalışmamın, tarafımdan bilimsel etik ve geleneklere aykırı düşecek hiçbir yardıma başvurmaksızın yazdığımı, intihal yapmadığımı ve yararlandığım eserlerin kaynakçada gösterilenlerden oluştuğunu, bunlara atıf yapılarak yararlanılmış olduğunu belirtir ve bunu onurumla doğrularım.

…../…../2019

Burak BAYSAN

T.C.
ÜSKÜDAR
ÜNİVERSİTESİ

YÜKSEK LİSANS TEZ SAVUNMA SINAVI TUTANAĞI

BAĞIMLILIK VE ADLİ BİLİMLER ENSTİTÜSÜ

GENEL BİLGİLER

Öğrenci No	:	174501030
Öğrenci Adı Soyadı	:	Burak Bayan
Anabilim Dalı	:	Adli Bilimler
Tez Danışmanı	:	Prof. Dr. Serhat Özekes
Tezin Başlığı	:	Sayısal Veri ve Deliller için Küp Tabanlı Çok Boyutlu Asimetrik Şifreleme Yöntemi

Toplantı Tarihi	: 03 Eylül 2019	Saati : 14:00	
Öğrenci Savunmaya : ☒ Geldi			

Üniversitemiz Lisansüstü Eğitim-Öğretim ve Sınav Yönetmeliğinin ilgili hükümleri uyarınca tez bilimsel olarak incelenmiş, adayın tez çalışmasını sunmasının ardından, adaya tez çalışması ile ilgili sorular yöneltilmiştir.

☒ Yapılan savunma sınavında adayın tez çalışması başarılı bulunarak **KABUL** edilmesine,

☐ Yapılan savunma sınavı sonunda tez çalışmasının **DÜZELTİLMESİNE, düzeltme için** adaya ..... ay **EK SÜRE** verilmesine (en fazla 3 ay)

☐ Yapılan savunma sınavının sonunda tezin **REDDEDİLMESİNE**

☒ **OY BİRLİĞİ** ☐ **OY ÇOKLUĞU**

İle karar verilmiştir.

Savunmada Tezin Başlığı : ☒ **Değişmedi** ☐ **Değişti**

Tezin Yeni Başlığı : ☐ **Değişmedi**

Öğrenci Savunmaya : ☐ **Gelmedi**

Üniversitemiz Lisansüstü Eğitim-Öğretim ve Sınav Yönetmeliğinin ilgili hükümleri uyarınca yukarıda belirtilen tarih ve saatte Tez Savunma Jürisi toplanmış ancak ilgili öğrenci savunma sınavına gelmemiştir. Adayın tez çalışmasını Jüri önünde sunmadığı için yapılan değerlendirmeler sonunda adayın tez çalışmasıyla ilgili aşağıdaki kararı,

☐ **OY BİRLİĞİ İLE REDDEDİLMİŞTİR.**

Tez Sınavı Jürisi	Unvanı, Adı Soyadı	İmza
Başkan	Prof. Dr. Sevil Atasoy	
Danışman Üye	Prof. Dr. Serhat Özekes	
Üye	Dr. Öğr. Üy. Kaan Yılancıoğlu	
Üye	Dr. Öğr. Üy. Mustafa Cem Kasapbaşı	
Üye	Dr. Öğr. Üy. Burcu Gürbüz	

(Tüm durumlarda jüri üyelerinin tez değerlendirme raporları gerekir.)

Sayı No :

Tarih : 03 / 09 / 2019

Yukarıda kimlik bilgileri belirtilen ve Anabilim Dalımız Yüksek Lisans Programı öğrencisinin Tez Savunma Sınav Tutanağı ve eklerinin Enstitü Yönetim Kurulunda görüşülmesi hususunda bilgilerinizi ve gereğini arz ederim.

Prof. Dr. Sevil Atasoy

Anabilim Dalı Başkanı
(Unvanı, Adı Soyadı, İmza)

Not: Bu forma orijinal raporlar (bir nüsha) eklenecektir.

ÖNSÖZ / TEŞEKKÜR

Bu çalışmanın ortaya çıkış sürecinin asıl emektarları olan; [1]

Adalet Bakanlığı

Temel Haklar, Eğitim ve Öğretime verdiği desteğiyle ülkemizi uluslararası alanda öne çıkaran yaklaşımları ve sağladığı imkânlar için teşekkür ederim.

Dr.Öğr. Üyesi Aylin YALÇIN SARIBEY

Üsküdar Üniversitesi Bağımlılık ve Adli bilimler Enstitüsü, Bölüm Başkanı

Bilişim etiği hakkında yazdığım bir çalışma sonuna oldukça silik font rengiyle ve ikilik sistemde yazdığım iletiyi fark edip ikilik sistemde yanıt veren hayranlık uyandıran zekâsı ve yardımları için, güler yüzlü ve inanılmaz içten yaklaşımları için, çalışmalarımı detaylıca okuyarak bir fizikçi-matematikçi olarak sözel yönden çekilen zorluğu giderme çabası için teşekkür ederim.

Ablam Banu Deniz TERLEMEZ

Bugün bu satırları yazmamı sağlayan ve afacan merakıyla ablasının birçok elektronik / mekanik eşyasından kimisini yeniden tamir edebilse de çoğunu hurdaya gönderen, ablasını psikolog olmaya sürükleyebilecek kadar anlaşılması güç, ablasının ismini koyduğu bir çocuk var, iyi ki varsın o çocuğun abılası.

Av. Gülderen ERTAŞ

İstanbul Barosu Avukatı,

Matematikçi

Unuttuklarımı hatırlatmak, bilmediklerimi öğretmek için her hafta şehirlerarası yolculuklar yapan, sadece eğitimimde ve hayatımda değil yaşamımda var oluşundan mutluluk duyduğum "insan". Paylaşmanın, insan olmanın, var olan her şeyi sevebilmenin, vazgeçmemenin, inadına var olmanın ve iyilikte direnmenin öğreticisi. O olmasaydı mümkün değildi…

1 Harf sırasına göre…

Dr.Öğr. Üyesi Kaan YILANCIOĞLU

Üsküdar Üniversitesi Bağımlılık ve Adli bilimler Enstitüsü, Bölüm Başkanı

Enstitü giriş mülakatımdan başlayan ve her zaman ortaya koyduğu ilgisi / yardımları için, Enstitümüzdeki her karşılaşmamızda bir hoca otoritesi yerine dostane yaklaşımlarla akademik kaygılarıma engel olduğu için, tez-makale ve her çalışmada yaşadığım sorunlarıma bıkmadan yardımlarını sunduğu için teşekkür ederim.

Mehmet KIRAY

Hukuk alanındaki yüksek lisans sınavını 3.lükle kazandığımda "*...Burak, istediğin bu mu?. Sen kendi alanında bir şeyler yapmalı ve fayda üretmelisin…*" demişti. Bir yıl sonra önerisini dinlememem sayesinde üniversitem ve ülkemin en değerli akademisyenleriyle keyif aldığım ve fayda sağlayabilecek çalışmalar yaparken buldum kendimi. Bir eğitimci olarak, öğrencilerine ve topluma fayda sağlama çabasıyla evrensel hukuku savunduğu, bilginin üreticisi olan insanları yetiştirdiği için teşekkür ederim. Eğitimci yaklaşımları olmasaydı bu adımları gerçekleştirmem mümkün olmazdı. Birçok kurumca zaten takdir edilen bu felsefesini anlayabilecek yetkinlikte insanlarla karşılaşmasını dilerim, başarıya inanmaya ve doğru yönlendirilmeye ihtiyaç duyan daha birçok insan var…

Annem Rahime USLU

Asla vazgeçmediği, her zaman inandığı için, disiplini ve ahlakı öğrettiği için, bu çalışma da dahil üzere olmak yazabildiğim ve yazacağım her cümleye sebep olduğu için teşekkür ederim. Yapacağım her iyi şeyde sen varsın.

Seher EREN

Üsküdar Üniversitesi Bağımlılık ve Adli bilimler Enstitüsü, Yazı İşleri

Enstitümüzle ilişkim sırasında her sorunumda ilgili, yardımcı, güler yüzlü yaklaşımları ve her karşılaşmamızda hayretle-imrenerek gözlemlediğim yardımcı olma çabaları, tarafımı BABE ailesinde hissettirmesi, yaşadığım karışık süreçlerde adım-adım takip ederek olası sorunları öncesinde çözümlediği için teşekkür ederim.

Prof.Dr. Serhat ÖZEKES
Üsküdar Üniversitesi Mühendislik ve Doğa Bilimleri Fakültesi Dekan Yardımcısı
Üsküdar Üniversitesi Bilgisayar Mühendisliği Bölüm Başkanı

Olağanüstü çabaları, sıfırdan bir araştırmacı yetiştirdiği, haftalık toplantılarımızda gerek bilim gerek kriptoloji üzerine yaptığımız keyifli ve verimli sohbetleri, inanılması güç mütevaziliği, tezim ve makalelerimize olan inancı ve güveni, bilgiyi akademikleştirmeyi öğretmesi, çalışma alanımızı ilgilendiren konular hayatımda boyunca anlatmak istediğim anlayan tek insan olması ve bilgiye dair yeni bir tezin konusu olabilecek varlığı için teşekkür ederim.

Prof.Dr. Sevil ATASOY
Üsküdar Üniversitesi Rektör Yardımcısı
Üsküdar Üniversitesi Bağımlılık ve Adli bilimler Enstitüsü Müdürü

Enstitümüzde var olmamı sağlayan ve bilime sarılmamı mümkün kıldığı için, bir bilgi dağı olmasıyla sayısız akademisyene idolken enstitümüzdeki toplantılarımızda "*...bu çocuk yemek yedi mi ?..*" endişeleriyle bilginin insan için olduğunu öğrettiği, üstün eğitimci ve bilim insanı zekâsı için teşekkür ederim.

Dr.Öğr. Üyesi Tuğba ÜNSAL
Üsküdar Üniversitesi Bağımlılık ve Adli bilimler Enstitüsü, Bölüm Başkanı

Enstitü giriş mülakatımdan başlayan süreçte güler yüzlü yaklaşımı için, erken yaşlar da elde ettiği akademik kariyeriyle birçok öğrencisine umut veren bir örnek oluşturduğu için teşekkür ederim.

Ve son olarak yaşamımın zor anlarında bile pozitif kazanımların farkında olmamı sağlayan, beni ben yapan, kaotik bir fonksiyonun *negatif* veya pozitif sabit değerlerine teşekkür ederim: Aaron WENDEL, Bahadır BADEM, Barış BAYRAM, Cüneyt AĞGİL, Demet GÜL, Dilek GÜL, Erdi GÜRSOY, Erol AŞIK, Faruk YÜZÜER, Fikri DAL, Görkem ÇETİN, Gül Melis TAZE Gürhan SOLAKOĞLU, Halil DOĞAN, Harun KADIOĞLU, James ZELLER, Kayhan AMUCA, Mevlüt BAYSAN, Murat ETİZ, Olcay ÖZDEMİR, Ömer BAĞLAN, Ramazan TERLEMEZ, Rıdvan ÖZDEMİR, Serkan DENİZ, Todd BRIAN.

ÖZET

Bu çalışma kriptoloji ve adli bilimler arasındaki ilişkinin detaylı incelemesini, bir kriptosistem algoritmasını (Cube) ve bu kriptosistem için sayısal imza değerleri üretebilen bir kriptografik mesaj özetleme algoritmasını (ICDS) içerir. Cube, asimetrik şifreli metin çıkışıyla yeni kriptografi sınıfında olan ve 3-boyutlu geometrik küp formunda blok yapısına sahip bir kriptosistemdir. Cube'ün şifreli metin asimetrisi, aynı düz metin ve aynı anahtar ile rastlantısal veya seçilebilir 256 farklı şifreli metin çıkışını sağlar. Cube verilere ait bit değerlerini 3-boyutlu koordinatlarda konumlandıran alt algoritmalara ve bu algoritmaların rastlantısal seçilmesini sağlayan tasarıma sahiptir. ICDS kriptografik mesaj özetleme algoritması, sayısal imza algoritmalarında, veri bütünlüğü ispatında, kaynak kontrolünde, veri işaretlemede, kimlik ispatında, şifre örtmede ve rastlantısal sayı üreteci (PRNG) olarak uygulanabilecek tasarımda geliştirilmiştir. ICDS'in 128-65336 bit aralığında özet değeri üretebilen esnek yapısı ve uygulamaya bağlı anahtarlı (MAC) veya anahtarsız (MDC) kullanılabilen özellikleri vardır. Cube, literatürde yer alan ve bu çalışma kapsamında tasarlanan testlerde yüksek güvenlik ve performans sonuçları elde etmiştir. ICDS, literatürde en çok bilinen ve güvenilir kabul edilen mesaj özetleme algoritmalarıyla aynı koşullarda analiz edildiğinde daha yüksek güvenlik ve performans sonuçları elde etmiştir.

Anahtar Kelimeler: kripto, adli, bilişim, delil, imza, özet, asimetrik, cube, icds

ABSTRACT

This study includes a detailed examination of the relationship between cryptology and forensic sciences, a cryptosystem algorithm (Cube), and a cryptographic message hashing algorithm (ICDS) capable of generating digital signature values for this cryptosystem. Cube is a new cryptography class with asymmetric ciphertext output and a 3-dimensional geometric cube-shaped block structure. Cube's ciphertext asymmetry provides random or selectable 256 different ciphertext outputs with the same plaintext and key pair. The cube has sub-algorithms that coordinate the bit values of the data in 3-dimensionally and has the design that allows these algorithms to be selected randomly. ICDS cryptographic message hashing algorithm has been developed to be used in digital signature algorithms, data integrity proof, source control, data marking, identity proof, password covering and random number generator (PRNG). The ICDS has a flexible structure that can generate a hash value in 128-65336 bit range, and that can be used with key-based (MAC) or keyless (MDC) applications. Cube has achieved high security and performance results in the literature tests and tests designed within this study. The ICDS has achieved higher security and performance results when analyzed under the same conditions as message hashing algorithms, which are well known and trusted in the literature.

Keywords: crypto, forensic, informatics, evidence, signature, hash, asymmetric, cube, icds

KISALTMALAR

3DES	:	Triple Data Encrypt Standart *Üçlü Veri Şifreleme Standardı*
ABD	:	Amerika Birleşik Devleri
AES	:	Advanced Encrypt Standart *Gelişmiş Şifreleme Standardı*
ASCII	:	American Standard Code for Information Interchange *Veri Değişimi İçin Amerikan Standart Kodu*
CMK	:	Ceza Mahkemesi Kanunu
DES	:	Data Encryption Standart *Veri Şifreleme Standardı*
DSA	:	Digital Signature Algorithm *Sayısal İmza Algoritması*
FBI	:	Federal Bureau of Investigation *Federal Soruşturma Bürosu*
ICDS	:	Inconvertible Data Signature *Dönüştürülemez Veri İmzası*
MAC	:	MAC: Message Authentication Code *Mesaj Yetkilendirme Kodu*
MD	:	Message Digest *Mesaj Özeti*
MDC	:	Modification Detection Code *Değişiklik Denetleme Kodu*
NIST	:	National Institute of Standards and Technology *Ulusal Standartlar ve Teknoloji Enstitüsü*
NSA	:	National Security Agent *Ulusal Güvenlik Kurumu*
PGP	:	Pretty Good Privacy *Oldukça İyi Mahremiyet-Gizlilik*
SHA	:	Secure Hash Algorithm *Güvenli Özetleme Algoritması*
TCK	:	Türk Ceza Kanunu

SİMGELER

$\mathcal{R}$: Halka cismi.

G : Grup cismi.

$\mathcal{F}$: Alan cismi.

$GF(2^n)$: 2^n düzenli Galois Alanı.

$\oplus$: Exclusive – OR / XOR işlemi

$\vee$: OR – mantıksal VEYA işlemi

$\wedge$: AND – mantıksal VE işlemi

$\ll$: Bit düzeyinde sola kaydırma işlemi.

$\gg$: Bit düzeyinde sağa kaydırma işlemi.

$\boxplus$: $GF(2^n)$ alanında toplama işlemi

$\boxminus$: $GF(2^n)$ alanında çıkarma işlemi

$\sim$: Değerin tersi, bire tümleyeni.

x‖y : x ve y değerlerinin yan yana getirilerek-birleştirilerek yeni değer oluşturması.

$\mathbb{Z}^+$: Pozitif tam sayılar kümesi.

$\mathcal{P}$: Plaintext / Açık metin.

$\mathcal{C}$: Ciphertext / Şifrelenmiş metin.

$\mathcal{K}$: Key – Cipher / Anahtar.

$\mathcal{H}$: Hash / Mesaj özet değeri.

$\mathcal{W}$: $GF(2^n)$ alanında x-bit uzunlukta kelime.

$A \cup B$: A ve B'nin birleşimi.

$A \cap B$: A ve B'nin kesişimi.

$A - B$: A'nın B'den farkı, $\{a: a \in A, a \notin B\}$.

TABLOLAR

ŞEKİLLER

1. GİRİŞ

1.1. Çalışma Düzeni

Bu çalışmada adli bilimler ve kriptografi arasındaki ilişki incelenirken, her iki alanı ilgilendirir detaylı bilgiler genelden özele doğru olacak şekilde bölümlendirilmiştir. Adli bilimlerin ve kriptografinin yöntemlerine dair gereksinim duyulabilecek incelemeler ve bilgilerin yer aldığı bölümlerden sonra çalışma başlığını oluşturan ICDS mesaj özetleme algoritmasının ve Cube kriptosisteminin yer aldığı bölümler oluşturulmuştur.

Çalışma kapsamında geliştirilen Cube ve ICDS'in kendilerine özgül analiz araçları, ölçekleri ve sınıfsal özellikleri bulunması sebebiyle her iki yöntemin kendi bölümlerinde ayrı – ayrı "Analiz" bölümleri oluşturularak deneylerde kullanılan analiz araçları, test sürecinin detaylı bilgileri ve analizlerden elde edilen sonuçlar verilmiştir.

8. TARTIŞMA bölümünde çalışma kapsamında elde edilen bütün analiz sonuçlarının geliştirilen Cube kriptosistemi ve ICDS mesaj özetleme algoritmasıyla birlikte değerlendirmesi verilmiştir.

1.2. Amaçlar

Bu çalışmanın amacı kriptografinin ve bilgi güvenliğinin adli bilimler alanındaki yerinin anlaşılabilmesi, mevcut yöntem ve yaklaşımlara ait bilgilerin detaylı ancak anlaşılabilir tek kaynakta bir araya getirilmesi, adli bilimler ve bilgi güvenliği alanlarının özel gereksinimlerinin tespit edilmesi ve tespiti yapılan gereksinimlere gerekli alt yapıyı sunacak ve sürdürülebilir araştırma-geliştirme kaynağı olabilecek bir kriptosistemin (Cube) geliştirilmesi, geliştirilecek bu kriptosistemde ve bilgi güvenliği-

güvenirliği gereksinimlerine yanıt verebilecek bir kriptografik mesaj özetleme algoritmasının geliştirilmesidir.

1.3. Ön Bilgiler

Teknolojinin süratli bir şekilde gelişimini sürdürmesi ve her geçen gün kurumsal ve bireysel alanda daha fazla zorunluluk halini alması, sadece yaşamı kolaylaştıran imkânları sunmakla kalmamış, beraberinde sınırları ve sınıfı belirsiz tehditleri de yanında getirmiştir. Bu çalışmanın yapıldığı tarihten çokta uzak olmayan bir zaman öncesine kadar sanal dünya – gerçek dünya ayrımından söz edilebilirken, bugün bu ayrım tamamen ortadan kalkmıştır. Bireyler tarafında sanal ve/veya teknolojik alandan uzak durabilmek bir tercih hakkı olmaktan çıkmıştır. Kurumlar, vatandaşları tercihinden bağımsız olarak, sağlık, yargı, finans ve sosyal bilgilerin tümünü bu sanal alanda barındırmaya ve işlemeye başlamıştır. Teknolojik alanın gerçek dünyadan ayrımının ortadan kalktığı ve sanıyı temsil eden "*Sanal"* sözcüğünün anlam ifade etmediği bir dünya içerisindeyiz. Bu dünya artık hem insan yaşamının etki ettiği, hem de etkilendiği bir alan ve bir makro düzendir. İnsan doğasının kaotik doğal denge olduğu yaklaşımlar bir yana teknoloji ve siber ağlar insanın kendi eliyle oluşturduğu yapay ve kaotik bir dünya, bir doğa haline gelmiştir.

Bu yenidünya beraberinde yeni tip tehditleri de getirmiştir. Bu tehditler yeni tip suçlar ve yeni tip suçlular olarak ortaya çıkmıştır. Bilgi ve zekâ temelli bu yenidünyanın kriminalist gelişimi de aynı temelde olmuştur. Daha derinlemesine yapılacak değerlendirmelerde bireyler tarafında kriminalist yaklaşımla ele alınan eylemlerin, kurumlar tarafında meşru olarak ele alınabildiği bilinmektedir. Bağımsız kurumların,

bağımsız meşrutiyet hak iddiaları veya iddia çabasında olmaya ihtiyaç duymaksızın eylemler içerisinde bulunması bu dünyanın tehditleridir.

Ulusal kurumların bir araya gelerek, küresel olarak bilgi güvenliği temelinde çalışmalar yapması ve bu çalışmalarını standartlar olarak sunması bireyler tarafında çok bilinmeyen ancak hassas bir noktadır. Bilgi güvenliği temelinde en önemli alanı oluşturan kriptografik fonksiyonların en çok bilinenleri bu ulusal kurumların standart olarak ilan ettikleri olmuştur. Diğer taraftan bakıldığında bu çalışmaları standart olarak sunan kurumların, küresel olarak bilgi güvenliğiyle ve bireysel/kurumsal mahremiyetle bağdaşmayan eylemleri ortaya çıkmaktadır. Bu çelişkilerin ortaya çıkışının altında yatan temel nedense bilginin yeni güç olduğunun haklı gerçeğidir.

Bilgi güvenliği denildiğinde akla ilk gelen kriptolojidir. Kriptoloji içerisinde, şifreleme bilimi anlamına gelen kriptografi ve şifrelenmiş verinin ve/veya şifreleme yönteminin çözümü/analizi anlamına gelen kriptanaliz alt sınıflarını barındırır. Temeline bakıldığında, varoluşunu ve keşiflerini ölümsüzleştirmek isteyen insanoğlunun yazıyla iz bırakmayı öğrenmesi kriptografinin başlangıcını da beraberinde getirmiştir. Çünkü insanoğlunun yazıyı keşfinin devamında bu izlerin var oluşuna tehdit olabileceğini keşfetmesi çok uzun sürmez. Bu son keşif ile birlikte bugün kriptolojinin dediğimiz alan tarih içerisinde evrimine başlamıştır.

M.Ö.60-50 Julius CAESAR'ın Sezar yöntemi, M.S.1586 Blaise de VIGENERE'nin Vigenere'sı, 1790'da Thomas JEFFERSON'ın Strip Chipher makinesi, 1917'de Joseph MAYBORGNE ve Gilbert VERNAM'ın One-Time-Padi, 2. Dünya savaşında Almanların Arthur SCHERBIUS ile Enigma cihazı, 1970'de Horst FEISTEL'in Lucifer algoritması, 1976'da Whitfield DIFFIE ve Martin HELLMAN'ın açık anahtar/asimetrik şifresi (1), 1978'de R.L.RIVEST, A.SHAMIR ve L.M.

ADLERMAN'ın RSA açık anahtar/asimetrik şifre algoritması (2) ve 1991'de Phil ZIMMERMAN'ın PGP sistemi tarihte önemli yer tutan örnekler olmuştur.

Tarihte önceleri devletler tarafında askeri amaçlarla kullanımı rutinleşen kriptoloji devamında ticari ve bireysel alanlarda da kendisine yer bulmuştur.

Adli bilimler açısından bakıldığında kriptoloji, sayısal delillerin bütünlüğünü ispat aracı olan mesaj özetleme-hash fonksiyonlarını sağladığı gibi soruşturulan bir vakada şifrelenerek saklanan delillerin çözümü aşamasında da aşılması gereken bir sorun olarak ortaya çıkmaktadır. Hiç kuşkusuz ki aynı sorun ve ikircikli durum sadece adli bilimler alanında değil, kurumsal yapıdaki alanların genelinde mevcuttur. Bu alana dair en kapsamlı araştırmaların yapılacağı disiplin dalı ise hiç kuşkusuz Adli bilimler olacaktır. Teorik ve pratik araştırmanın küresel veya ulusal kapsamda ele alınabilmesini mümkün kılan multidisipliner bir alan olan Adli bilimler araştırmanın sosyolojik, normsal ve etik yönlerden de ele alınmasına imkân sağlar. Fen bilimleri ile sosyal bilimlerin sentezini pozitivist çerçevede sağlama çabasında olan adli bilimlerin ürettiği bilgi sayıca zengin birçok alandan beslenirken bu alanları da besler.

Bu çalışma kapsamında ele alınan "*Sayısal Veri ve Deliller İçin Küp Tabanlı Çok Boyutlu Asimetrik Şifreleme Yöntemi*" konusu da adli bilimlerin bu zengin alan sentezi sonucu tanımlanabilen gereksinimlerini karşılamak üzere ele alınmıştır. Araştırma kapsamı sayısal deliller odaklı olarak başlamış, sonuçları odaklanan alan ile ilişkili diğer alanlar olan matematik, kriptografi ve bilgisayar gibi mühendislik alanlarına doğrudan ulaşmıştır.

Çalışma kapsamı fen bilimlerinde ve sosyal nilimlerde yanıtlanması gereken birden fazla hipotez üzerine kuruludur. Her bir hipotezin sonuçlanması için önceki çalışmaların ayrı-ayrı ele alınması ve ortaya konulması gerekmiştir. Çalışma

gerekçelerini ortaya çıkaran; geçmiş çalışmaların geçerliliklerini – güvenirliklerini yitirmiş durumları ve buna karşın halen adli bilimler alanında başvurulan kaynaklar olmasıdır. Adli bilim hukuk alanındaki ilişkisi bu geçmiş çalışmalar ile soruşturma yapılmasına karşıt bir durumdur.

Bu çalışmada adli bilişim ve veri güvenliği alanlarında kullanılabilecek ICDS: Inconvertible Data Signature başlıklı yeni nesil mesaj özetleme – hash fonksiyonu, CUBE isimli yeni nesil asimetrik kriptosistemi tasarlanmıştır.

Cube, bir veri şifreleme yöntemidir. Kriptografi alanında yeni bir alt alana aittir. Bu alan şifrelenmiş metnin (ciphertext) asimetrisidir. Asimetrik şifrelenmiş metin (ciphertext), açık metin (plaintext) ve anahtar veri (key) sabit olduğunda, her bir şifreleme işlemi sonunda farklı şifrelenmiş metin (ciphertext) elde edilmesini temsil eder. Cube ayrıca geliştirilmiş blok yapısına ve verilerin 3-boyutlu alanda permütasyonunu içeren alt fonksiyonlara sahiptir.

ICDS (Inconvertible Data Signature), bir mesaj özetleme-hash fonksiyonudur. ICDS, kimlik ispatından sayısal delillerin bütünlüğü ispatına uzanan geniş bir alandaki gereksinimler ele alınarak tasarlanmıştır. Bugün en çok bilinen ve en güvenilir mesaj özetleme fonksiyonlarına göre özet değeri boyutu ve parametrik yapısıyla daha gelişmiş tasarıma sahiptir.

1.4. Ceza Yargılamasında Deliller ve Bilirkişilik

Delil, maddi gerçeğin ve dava konusu olayın ortaya çıkarılmasında ispat amacıyla kullanılan araçlardır. Burada dava konusu ceza yargılamasıysa belirli niteliklere sahip olması koşuluyla ceza yargılamasında delil serbestîsi vardır. Ceza yargılamasında, hukuk yargılamasında olduğu gibi kesin deliller yoktur. Yalnızca duruşmanın nasıl

yapıldığına dair tutanak ispat aracı olabilir ve sahteliği kanıtlanmadığı sürece bu tutanak yargılamaya ilişkin ispat aracı olmaktadır.

Bir nesnenin delil olabilmesi için yukarıda belirtildiği gibi belli özellikleri taşıması ve belirli bir hususun ispatı konusunda yargıçta vicdani kanaat oluşturması gerekir.

Ceza yargılamasında her ne kadar delil serbestîsi kuralı esas olsa da, bu deliller şu özelliklere haiz olmalıdır.

- Yargılama konusu olayın tümünü veya bir parçasını ispat edebilecek nitelikte olmalıdır
- Beş duyu organı ile algılanabilecek nitelikte olmalıdır
- Hukuka uygun yollardan elde edilmiş olmalıdır
- Sağlam ve güvenilir olmalıdır
- Müşterek olmalı, müşterekliği sağlanmalıdır. İddia, savunma ve yargı makamının bilgisine sunulmalıdır
- Akılcı ve bilimsel olarak kabul edilebilir olmalıdır

Delilin müşterekliği, muhtevasını sadece yargıcın öğrenmesinin yetmediğini, dava taraflarının da öğrenmesi ve mütalaa niteliğindeki hükümleri ile kolektif hüküm verme faaliyetine katılabilmesini anlatmaktadır. Bu delilin ortaya konulup tartışılmasıyla mümkün olur. Yargıcın olay hakkındaki şahsi bilgisine[2] dayanarak karar vermesi, müdafaa hakkına da dokunur. Herkesçe bilinen maruf ve meşhur şeylere ilişkin bilgilerini yargıç ancak şu şartla dikkate alabilir;

- Bilginin mefruz hüküm ve mücerret kaide olmayıp, müşahhas olması
- Hakikat olduğunun tartışılamayacağının yerleşmiş, yayılmış bulunması.

2 Olay Hakkında Şahsi Bilgi ile anlatılmak istenen herkesçe bilinmeye hususlara ait bilgilerdir.

Adli bilişiminde dâhil olduğu ceza yargılamasında deliller üçe ayrılır:

1. Beyan Delili; Uyuşmazlık konusu maddi olaya ilişkin olarak sanık, mağdur veya üçüncü kişilerin sözlü açıklamalarıdır
    a. Sanık açıklamaları (İfade ve sorgu.)
    b. Tanık açıklamaları
    c. Sanık ve tanık dışığında kalanların açıklamaları
2. Belge Delili; Olay anında olayın bire bir belirli şekillerle bir nesne üzerine aktarılmasıdır.
    a. Yazılı belge (Sorgu tutanağı, adli sicil kaydı v.b.)
    b. Şekil tespit eden belge (Fotoğraf, resim, kroki, plan v.b.)
    c. Ses tespit eden belge (Ses kayıtları ve C.M.K. m. 147/1-b)
3. Belirti Delili; Olaydan geriye kalan her türlü iz ve eserdir. Sanığın iradesi dışığında olaydan geriye kalan iz ve eserlere doğal delil denir. Belirti delilleri kriminalistik birimin verilerine göre değerlendirilir. Olay yerinde bulunan bütün belirti delilleri birer dizsiz tanıktır. Belirti delillerinin anlamlandırılması, başka bir anlatımla konuşturulabilmesi ancak keşif veya bilirkişi incelemesiyle mümkün olabilmektedir (parmak izi, fren izi, sanık veya mağdurun kanı, tabanca v.b.) (3, 4).

Çözümü özel veya teknik bilgiyi gerektiren bir konuda, Cumhuriyet savcısı, mahkeme veya yargıcın talebiyle, kendisinden özel veya teknik konuda görüşü alınan kişiye bilirkişi denmektedir. Görüşüne başvurulan kişinin bilirkişi olarak kabul edilmesi belirlenen konuda uzmanlığa sahip olması, Cumhuriyet savcısı, yargıç veya mahkeme

tarafından görevlendirilmesi ile bağlıdır. Bu iki özellikten birisini taşımayan kişilerin beyanı ancak tanık beyanı olarak kabul edilebilir.

Bilirkişi beyanını kendisine konu hakkında soru sorulması üzerine, inceleme konusu üzerinde olguları tespit ederek görüşünü yazılı veya sözlü olarak açıklar. Ceza Muhakemesi Kanununda bilirkişiliğe ilişkin hüküm bulunmayan hallerde tanıklara ilişkin hükümlere, reddinde ise hâkimlerin reddine ilişkin hükümlere atıf yapıldığı görülür.

Delillerin reddi veya sınırlandırılması şu hallerde yapılabilir:

1. Delil gösterilmesine yasal olarak olanak yoksa
2. Olay delil gerektirmeyecek biçimde açıksa
3. İspatlanması istenen olayın karara etkisi yoksa yada daha önce ispatlanmış bir hususa ilişkinse
4. Delil amaca elverişli değilse
5. Delilin elde edilmesi olanaksızsa
6. Delil gösterilmesi isteği işi/yargılamayı uzatmak amaçlıysa
7. İleri sürülen olay gerçek olarak kabul edilemeyecek nitelikteyse

Bu hallerin kanundaki düzenlenmiş halleri şu şekilde karşılığını bulur;

1. Delil, kanuna aykırı elde edilmişse
2. Delil ile ispat edilmek istenilen olayın karara etkisi yoksa
3. İstem, sadece davayı uzatmak maksadıyla yapılmışsa

Olayın ispatına ilişkin asıl delillere doğrudan delil denir[3]. Olayı doğrudan ispatlamayanlar doğrudan delili destekleyen delillere de dolaylı delil denir[4].

Ceza yargılamasının en önemli ilkelerinden birisi de "*in dubio pro reo*"[5] ilkesidir. Bu ilkenin özü göz önünde tutulması gereken herhangi bir meselede baş gösteren kuşkunun, sanığın yararına değerlendirilmesidir. Oldukça geniş bir uygulama alanı bulan bu kural;

1. Bir suçun gerçekten işlenip işlenmediği
2. Suç işlenmişse gerçekleştirme biçimi
3. Dava koşulları
4. Cezayı kaldıran veya hafifleten nedenlerin bulunup bulunmadığı

hususlarında sanık lehine uygulanır (5).

Ceza yargılamasında ispat külfeti, soruşturma evresinde Cumhuriyet savcısının sanığın lehinde ve aleyhindeki delilleri toplaması görevi olması, hakimin delil araştırmasının da kabul edilmesi karşısında ceza muhakemesinde ispat külfeti diye bir mesele yoktur. Ayrıca duruşmaların tek oturumda bitirilmesini amaçlayan düzenlemelerin felsefesi düşünüldüğünde; davaları bilmeyi sanıkların kabulüne bağlamak ve yargılamayı uzatmamak gerekir. Ancak bununda istisnaları vardır, bunlar;

1. Resmi bir belgenin sahte olduğunu iddia eden bunu ispat edecektir
2. Hakaret suçlarında isnat olunan maddenin doğruluğunun ispatı bunu ispat edene bağlıdır (T.C.K. m. 127)

3 Olaya bizzat şahit olan tanığın beyanı gibi.

4 Sanığı olay günü başka bir yerde gördüğünü söyleyen tanığın beyanı gibi.

5 Şüpheden sanık faydalanır.

3. Haksız mal edinme ile suçlanan kişi edindiği bu malları hukuka uygun yollardan iktisap ettiğini ispatlamak zorundadır (3628 sy. K. M. 4, 13)

Karine bir olayın doğruluğunun önceden bir surette kabul edilmesidir ve ikiye ayrılır;

1. Kanuni karine[6]
2. Basit Karine[7]

Hukuka aykırı olarak elde edilen delillerin dosyadan çıkarılıp çıkartılmayacağı konusu bu çalışmanın yapıldığı tarihte üzerinde mutabakata varılmayan bir konudur. Konuya ilişkin olarak Ceza Muhakemesi Kanununda iki hüküm yer alır, madde 206/1'e göre delil hukuka aykırı olarak elde edilmişse ikamesi talebi reddedilecektir. İkame talebi ret olunan delilin alınıp dosyaya konulması mümkün değildir. Ancak madde 206 sadece irat ve ikameyi engellediğinden bu hüküm hukuka aykırı delillerin dosyaya girmesine engel olmayacaktır. Nitekim Ceza Muhakemesi Kanunu hukuka aykırı olarak elde edilen delilin dosyadan çıkartılması anlayışını kabul etmediğini açıkça belirtmiştir. Buna göre mahkûmiyet hükmünün gerekçesinde hakin dosya içerisinde bulunan ve hukuka aykırı yöntemlerle elde edilen delilleri ayrıca ve açıkça gösterecektir (C.M.K. m.230/1-b). O halde Ceza Muhakemesi Kanununa göre dosyada bulunan ve hukuka aykırı olarak elde edildiği belirlenen delillerin dosyadan çıkarılması mümkün değildir.

Tüm bunlarla birlikte ceza yargılamasının temel amacı maddi gerçeğin ortaya çıkarılmasıdır. Ancak Alman Federal Yüksek Mahkemesinin kararlarında, ceza

6 15 yaşından küçüğün rızasının mevcut olmaması gibi.

7 Her delil basit karineyi gerektirir. Fotoğraftaki görüntünün doğru olduğunun kabulü gibi…

yargılamasının amacı yalnızca maddi gerçeği bulmak değil, maddi ve özellikle usulü hukuk normlarının dikkate alınmasını ve hakların korunmasını gerektirir. Buradan hareket ile Alman hukukunda maddi gerçeğin ortaya çıkarılmasından ziyade tali bir amaç olduğu ve asli hedefin hukuksal barışın tesisine yönelik hüküm verilmesi olduğu görülür.

Yine ceza yargılamasının en önemli ilkelerinden birisi olan "*in dubio pro reo"* [8] kuralı uyarınca sanığın bir suçtan cezalandırılmasının temel koşulu suçun kuşkuya yer vermeyen bir kesinlikle ispat edilmesine bağlıdır. Ceza yargılamasında mahkûmiyet, büyük ve küçük bir olasılığa değil, her türlü kuşkudan uzak bir kesinliğe dayanmalıdır. Bu itibarla, birbiriyle çelişkili ve kesin bir kanaat vermekten uzak kanıtlara dayanılarak karar verilmesi isabetsiz olur (6).

Tüm bunlar ışığında bilirkişi görüşünün niteliği için delil olduğu yönünde doktrinler olduğu gibi delil değerlendirme aracı olduğuna dair doktrinler de bulunmaktadır. Bir duruma göre bilirkişi raporlarının bağlayıcı olmadığı ele alınırsa delillerin değerlendirilmesinde yargıca yardımcı bir araç olmaktadır. Buradan hareketle de bilirkişi yargıcın yardımcısıdır denilebilir.

Bilirkişiye herhalde başvurma zorunluluğu yoktur. Ancak adli bilişim ve/veya dijital deliller konusunda yasal bir zorunluluğun özel yasal düzenlemeyle mevcut olmaması karşısında, bir yargıç veya yavcının hayli karmaşık ve ileri teknik bilgileri gerektiren bu tür bilgileri elde etme veya tek başına yorumlama imkânı yoktur. Bu sebeple bilirkişiye başvuruyu zorunlu kılan haller adli bilişim ve/veya dijital deliller içinde geçerli olacaktır.

8 Şüpheden sanık faydalanır.

Bu haller şöyledir;

1. Genel zorunluluk halleri;
    a. Çözümü uzmanlığı gerektiren
    b. Özel bilgiyi gerektiren
    c. Teknik bilgiyi gerektiren

nedenlerin bulunması halinde yargıç, mahkeme veya Cumhuriyet savcısı bu alanlarda bilgiye ihtiyacı olduğunu düşünürse bilirkişiye başvuracaktır.

2. Özel Hukuk Halleri; Yargıç, mahkeme veya Cumhuriyet savcısı önüne gelen olayın uzmanlık, özel veya teknik bilgi isteyip istemediği konusunda kanaat getirmesine veya fikir yürütmesine gerek olmayan durumlardır. Bu durumlarda kanun uzmanlığı özel veya teknik bilgiye ihtiyaç olduğunu baştan kabul etmiştir. Ceza Muhakemesi Kanunuyla özel kanunlarda belirtilen bilirkişiye başvurulması zorunlu haller şunlardır;
    a. Sahte para ve değerler üzerinde inceleme (C.M.K. m.73)
    b. Şüpheli veya sanığın şuurunun tetkiki (C.M.K. m.74/1)
    c. Akıl hastalığı (T.C.K. m.32)
    d. Şüpheli veya sanığın akıl hastası olması (T.C.K. m.57)
    e. Üçüncü kişilerin akıl hastalığı (T.C.K. m.175, 194)
    f. Cinsel Dokunulmazlığı ihlal edilenin ruh sağlığı (T.C.K. m.102/3-a)
    g. Sağır dilsizlik (T.C.K. m.33)
    h. Uyuşturucu ve alkol muayenesi (T.C.K. m.34, 57/7)
    i. Ölü muayenesi ve otopsi (C.M.K. m.86, 88)

j. Zehirlenme şüphesi üzerine inceleme (C.M.K. m.89)

k. Şüpheli veya sanığın beden muayenesi ve örnek alma (C.M.K. m.75, 77)

l. Moleküler genetik inceleme (C.M.K. m.79)

m. Beden muayenesi ve vücuttan örnek alınması (C.M.K. m.75, 77)

n. Fizik kimliğinin tespiti (C.M.K. m.81)

o. Tıbbi müdahalelerde hekim kusuru (1593 Sy. K. m.10)

Bilirkişi rapor ve görüşlerinin ilke olarak bağlayıcılığı yoktur. Bilirkişi mütalaası bir delil değil, delillerin değerlendirilme vasıtasıdır. Yargılama süreci içerisinde yargıç karar veren kişiyken, bilirkişi özel ve teknik konularda delillerin değerlendirilmesinde yargıca yardımcı olan uzmandır.

Yargıcın, tamamen hukuk bilimi dışığında olan teknik incelemeler neticesinde elde edilmesi mümkün olan ve yargıcın geçerliliğini kontrol edemediği sonuçları kabul etmemesi kolay değildir. DNA analizi sonuçları gibi bilişim alanındaki bir analiz veya araştırmanın sonuçları da yüksek teknik bilgiyi gerektiren ve çoğunlukla doğrudan kabul gören veriler olmaktadır. Fakat tüm bunlar yargıç veya Cumhuriyet savcısının bu konuda ikinci bir uzman görüşüne başvurmasına engel değildir, bilginin incelenmesinde veya rapor haline getirilmesinde beşeri hatalar olması mümkün olabilmektedir. Yargıç bilimsel raporu kabule mecbur olmamakla birlikte, kabul etmeme gerekçesini bilimsel olarak izaha ve kararın gerekçesine yansıtmaya mecburdur.

1.4.1. Dijital Deliller

Sayısal deliller, delil olabilme şartlarına haiz olmasıyla birlikte veri saklayabilen elektronik ortam materyalleridir. Bunlar disk, disket, CD, DVD ve Flash Bellek gibi bilgi saklama yeteneğine sahip olan bütün cihazlar olarak tanımlanabilir. Bu tanımdan da anlaşılabileceği gibi akıllı beyaz eşyalar[9] içerisinde hafıza alanları olabilmektedir. Kısaca herhangi bir nesnenin sayısal delil olma özelliğine sahip olması için elektronik alanda, herhangi bir yöntem ile veri-bilgi bulundurabilmesi yeterli olacaktır.

Dijital delillerin incelenmesi ve yorumlanması ancak çok yoğun teknik bilgiye sahip uzmanlarca yapılabilecekken toplanması aşamasını içeren olay yeri incelemesi bile özel uzmanlık gerektirmektedir. Bu delillerin tespiti, toplanması, laboratuara nakli ve muhafazası, delilin türüne göre farklılıklar içermektedir.

Dijital deliller çok kolay bozulabilmekte ve delil olma yeterliliklerini kaybedebilmektedir. Bu yapıları sebebiyle, delil tespitinden bilirkişi raporunun hazırlanmasına kadar geçen bütün süreçler kayıt altına alınmakta ve sonuç rapora yansıtılmaktadır. Bu tür delillerin, diğer dillerin elde edilmesinden daha katı kuralları vardır.

Bütün sürecin yansıtıldığı raporun vazgeçilmez bir parçası da hash değerleridir. Hash değerleri her ne kadar "…dünyada daha çok adli bilişim alanında delil bütünlüğünün… amacıyla kullanılan hash algoritması…" [10] tanımlamalarıyla delil bütünlüğü ispatına indirgense de, aslında kriptografik alanda temel bir yapıyı oluşturan ve daha çok sayısal imza alanında kullanılan hash algoritmalarının delil bütünlüğü ispatı amacıyla kullanımı ek bir faydadır. Bu algoritmalar incelenen veri kaynağındaki bütün

9 Televizyonlar, klimalar, buzdolapları, çamaşır makineleri v.b.

10 Başar, Y.:'Siber Suç Soruşturmalarında Adli bilişim İncelemeleri', Yüksek Lisans Tezi, Eylül 2015, Afyon Kocatepe Üniversitesi, Fen bilimleri Enstitüsü

bit değerlerini belirlenmiş bir fonksiyon iterasyonunda işleyerek sabit uzunlukta bir sonuç üretir. Veri kaynağındaki yalnızca bir bit değerinin değişmesi bile elde edilecek hash değerinin kaotik farklılık göstermesiyle sonuçlanır.

Tespiti yapılan dijital veri kaynağının ilk aşamada hash değerinin alınması ve fiili laboratuar incelemesinin sonuçlandığı son aşamada yeniden hash değerinin alınması gerekir. Elde edilen her hash değerinin eş olması delil üzerinde hiçbir değişiklik yapılmadığı ispatı olmaktayken, farklı değerler elde edilmesi delil üzerinde veri farklılıkları olduğunu ve delil bütünlüğünün bozulduğunu işaret eder.

Delil bütünlüğünün bozulması ceza yargılamasının temel prensibini "*şüpheden sanık yararlanır*" ilkesini etkinleştirecektir, bu durumda da kesin bir kanaatin varlığında dâhil sanığın cezalandırılamaması sonucunu ortaya çıkarabilecektir.

Bu çalışmanın yazılmakta olduğu tarihte hash değerleri yazılı hukuk kurallarında yer almamaktadır. Ancak normsal zorunluluğu olmasa da hash değerleri defacto olarak dijital delillerin bütünlüğünü ispat aracı olarak kullanılmaktadır.

1.4.2. Adli bilişim ve Kriptografi

Adli bilişim ve kriptografi ayrılmaz bir birliktelik oluştururlar. Dijital delillerin bütünlüğünün ispatında hash fonksiyonlarının kullanılması, şifrelenmiş verilerin ve/veya haberleşmenin çözümlenmesi en çok bilinen örnekleri oluştururken hemen hemen araştırmanın içeriğinde yer kaplayan konular olmaktadır.

Kriptografinin özel bir ilgi ve eğitimi gerektiren ileri düzey bilgi gereksinimi adli bilişim alanında uzmanlaşanların gerekli yeterlilik gösteremedikleri bir alanı oluşturmaktadır. Literatürde yer alan lisansüstü bitirme tezleri incelendiğinde hash

algoritmalarını adli bilimlerle sınırlayan, eksik ve hatta hatalı bilgiler yer aldığı görülmektedir.

Bu çalışmanın yapıldığı tarihte Ceza Mahkemesi Kanununun 134/1-2 Maddesi elde edilen dijital delilin şifrelenmiş olması durumunda da şifrenin çözülmesine dair bir yetkiden bahsetmektedir. Ancak şifrelenmiş dijital verilerin çözümü kolay bir işlem değildir. 4096-bit düzeyinde yapılabilen bir şifreleme hiçbir teknik bilgisi olmayan kişilerce, internette yayınlanan ücretsiz yazılımlarla yapılabilmektedir. Çoğu zaman güçlü bir algoritmaya sahip olan şifreleme yöntemlerine karşı tek çözüm yolu deneme-yanılma yöntemi olmaktadır. Bu yöntemin uygulanması için bilgisayar kullanıcıları tarafından doğrudan yada dolaylı olarak kullanılabilen 2^8 ASCII [11] karakterin her birisinin 8 haneli bir şifre kullanıldığı varsayılan dosya şifresi üzerinde denenmesi gerekir. Böyle bir durumda $(2^8)^8$ yani 18.446.744.073.709.551.616 olasılığın denenmesi gerekir.

Şifrelenmiş verilerin çözümlenmesinde cluster isimli ağ yapıları üzerinde çalışan bilgisayar yığınları kullanılması bir seçenek olarak yer almaktadır. Burada birçok bilgisayarın bir araya gelerek olasılık evreninin sistematik paylaşımıyla anahtarın aranması süreci işlenir. Bu yöntem siyah şapkalı bilgisayar korsanları [12] tarafından, internet üzerinden kötü amaçlı yazılımlar aracılığıyla bilgisayarları zombi haline getirilen mağdurlar aracılığı ile gerçekleştirilmektedir.

Bir diğer çözüm kümesi de kriptanaliz yöntemidir. Bu yöntem yüksek seviyeli matematik bilgisi ve yeteneği gerektirir. Burada önce şifreleme yönteminin bilinmesi veya tespiti gerekir, ancak bu birçok adli bilişimcinin eğitimleri müfredatında yer

11 American National Standard Code for Information Interchange - Veri degisimi için, Amerikan Ulusal Kod Standardı

12 Siyah Şapkalı Bilgisayar Korsan: Bilgisi ve/veya yeteneğini zarar vermek, kendisine veya başkasına çıkar sağlamak amacıyla kullanan kişi.

almaz. Yöntemin teşhisi sonrasında verinin bulunduğu alan bilinmelidir, CD v.b. sabit alandaki veri ile internet gibi iletişim kanalı üzerindeki şifrelenmiş verinin çözümlenmesi ayrı yöntemler gerektirecektir. Devamında veriye kriptanaliz uygulaması yapılabilecektir.[13]

Burada anlatılmak istenen adli bilişim ve kriptografinin birbirlerinin ayrılmaz birer parçası ancak ezeli düşmanları olduğudur.

1.4.3. Şüpheli / Sanık Tarafında Delillere Erişim Hakkı

CMK. 134. Maddesi bilgisayarlarda, bilgisayar programlarında ve kütüklerinde arama, kopyalama ve el koyma ilişkin hususları açık şekilde belirtmiştir.

İlgili kanun metni yapılan son değişiklikler ile birlikte şu durumdadır;

Bilgisayarlarda, bilgisayar programlarında ve kütüklerinde arama, kopyalama ve el koyma

MADDE 134- (1) (21.02.2014 – 6526 sayılı Kanunla değişik) Bir suç dolayısıyla yapılan soruşturmada, somut delillere dayanan kuvvetli şüphe sebeplerinin barlığı ve başka surette delil elde etme imkânının bulunmaması halinde, Cumhuriyet savcısının istemi üzerine şüphelinin kullandığı bilgisayar ve bilgisayar programları ile bilgisayar kütüklerinde arama yapılmasına, bilgisayar kayıtlarından kopya çıkarılmasına, bu kayıtların çözülerek metin haline getirilmesine hâkim tarafından karar verilir.

(2) Bilgisayar, bilgisayar programları ve bilgisayar kütüklerine şifrenin çözülmemesinden dolayı girilememesi veya gizlenmiş bilgilere ulaşılamaması halinde el konulabilir. Şifrenin çözümünün yapılması ve gerekli kopyaların alınması halinde, el

13 Kriptanaliz yöntemleri Bölüm 5. KRİPTOGRAFİK SALDIRILAR altında detaylı olarak ele alınmıştır.

konulan cihazlar gecikme olmaksızın iade edilir.

(3) Bilgisayar veya bilgisayar kütüklerine el koyma işlemi sonrasında, sistemdeki bütün verilerin yedeklemesi yapılır.

(4) (Değişik, 21.02.2014/6526) Üçüncü fıkraya göre alınan, yedekten bir kopya çıkarılarak şüpheliye veya vekiline verilir ve bu husus tutanağa geçirilerek imza altına alınır.

(5) Bilgisayar veya bilgisayar kütüklerine el koymaksızın da, sistemdeki verilerin tamamının veya bir kısmının kopyası alınabilir. Kopyası alınan veriler kâğıda yazdırılarak, bu husus tutanağa kaydedilir ve ilgililer tarafından imza altına alınır.

Yukarıda da açıklandığı üzere C.M.K.'nın 134/4. Maddesi gereği "..alınan yedekten bir kopya çıkarılarak şüpheliye veya vekiline verilir…" emredici norm şeklinde yerini almıştır. Bu durumda normsal olarak el konulan dijital delillerin bir kopyasının şüpheliye / sanığa verilmesi zorunlu kılmıştır. Bu ceza yargılamasında silahların eşitliği, delillerin müşterek olması açısından gerekliliği açık bir durumdur.

1.4.4. Mağdur Hak ve Mahremiyeti

Önceki bölüm 1.4.3. Şüpheli / Sanık Tarafında Delillere Erişim Hakkı ile açıklanan şüpheli / sanık tarafında delillere erişim hakkı delillerin müşterekliği ilkesinin bir sonucudur. Ancak yukarıda görülen bu durum ceza yargılamalarında örneğin özel hayatın ihlali içerikli bir yargılamada mağdurun daha da mağdur olmasına sebep olabilecek sonuçları ortaya çıkartabilecek sonuçları ortaya çıkartmaktadır.

Burada örnek özel hayatın ihlali açısından işlenmiş bir suçun ele alındığını varsaydığımızda, suçun delili olan mağdura ait müstehcen fotoğrafların şüpheli/sanık bilgisayarında veya depolama ortamlarında ele geçirildiği bir durum oluştuğunda, ilgili C.M.K. 134/4 maddesinin emredici kuralı gereği bu delile ol konulması karşısında

delilin bir kopyası yeniden şüpheli/sanık tarafına tam mevcudiyeti ile verilecektir. Bu ceza yargılamasının tüm ilkeleri açısından da gerekli bir durumdur. Ancak eşit yargılamanın, silahların eşitliğinin, delillerin müşterekliğinin ve/veya şüpheli/sanık haklarının korunduğu bu ilkeler ışığında mağdur açısından ortaya çıkacak sonuçların çözülmesi, bilişim ile iç içe gelmiş devlet ve yargı sistemi içerisinde mümkündür.

Yukarıda anlatılan örneğin görsel(resim) olduğu ele alındığında bu görselin doğrudan CD veya benzeri veri saklama ortamları içerisinde şüpheli/sanığa verilmesi yerine belli prosedürler içerisinde şüpheli/sanığın belli kurallar ile ulaşabileceği UYAP üzerinde açılacak bir alanda erişimine açılması ve asıl verilerin başka bir alanda saklanması mümkündür.

- Bu durum mağdur açısından ne gibi bir fayda sağlayacaktır ?
  - ✓ UYAP üzerinden bilişim sistemleri vasıtası ile erişim imkânı verilen resim verisi içerisine steganografi veya watermark araçları ile imza bırakılması mümkündür.
  - ✓ UYAP üzerinde yapılacak bir düzenleme ile şüpheli/sanık ilgili resim verisini kayıt etmek istediğinde bunun çeşitli yazılım kodları aracılığı ile engellenmesi mümkündür

- Bu durumun şüpheli/sanık haklarında açısından ileri sürülebilecek sorunlar nelerdir?.
  - ✓ Şüpheli/Sanık bilişim ortamının olmadığı bir alanda bu delillere ulaşamayacaktır.
  - ✓ Ancak bilişim/teknoloji dünyası ile iç içe gelmiş günümüzde UYAP üzerinde geliştirilecek ortamın doğru hazırlanması ile ilgili

verilere cep telefonları veya akıllı telefonlarla erişimi mümkün olabilecektir.

- ✓ Şüpheli/Sanık bu şekilde elinde fiziksel kâğıt/dosya ile delil taşımak yerine istediği an/istediği yerden bu delillere ulaşabilecektir.
- ✓ Veriler düzenli bir hal içerisinde olacaktır bu sayede şüpheli/sanık delilleri karışık fiziksel dosyalama arşiv içerisinde aramak yerine daha senkronize bir yapıda arayıp bulabilecektir.

- o Delillerin müşterekliği ilkesi zedelenecektir.
  - ✓ Delillerin asıl yapısına zarar verilmemiş sadece belli imzaların yerleştirilmiş hali anlık erişime açılacak ve asıl halleri yine UYAP ortamında ve dosya içerisinde var olacaktır.
  - ✓ İmzalı verilerde imzasız delillere hakim/savcı doğrudan erişebilecektir.
  - ✓ İmzalı olan verilere şüpheli/sanık her zaman, her yerden erişebilecektir.

- o Deliler üzerine imza verileri bırakılarak, deliller bozulacaktır.
  - ✓ Delillerin asılları ayrıca dosya ve UYAP ortamı üzerinde saklanacaktır.
  - ✓ Dijital ses, video veya resim üzerinde imza işlemleri sırasında kullanılacak olan steganografi ve/veya damgalama algoritmaları veri üzerinde işitsel ve/veya insanın işitsel algı seviyesinin çok

altında ve/veya üzerinde kodlama yapmaktadır ve deliller üzerinde asıl amaç olan savunma amaçlı erişim ve tetkik işlemini şüpheli/sanık yapabilecektir.

- Bu imzalama yöntemi sayesinde elde edilmek istenen fayda ve sonuç ne olacaktır ?
  - ✓ Mağdura ait ses, video veya resmin yeniden şüpheli/sanık mevcudiyetinde olup usulsüz ve mağdurun daha fazla mağdur olmasına(internet ortamında paylaşımı, tehdit vs. devamı gibi…) karşı kullanımı engellenebilecektir.
  - ✓ İlgili delillerin bilişim sisteminin imkânlarının kodlama yeteneklerinden faydalanması suretiyle şüpheli/sanık tarafından yeniden bir veri saklama ortamına alınması engellenebilecektir.
  - ✓ İlgi delillerin bir şekilde yeniden şüpheli/sanık tarafından hazırlanacak sistemden elde edilmesi durumunda ilgi verinin, kim tarafından ne zaman nerede ve hangi ortamda elde edildiği delil içerisinde bırakılan imza sayesinde tespit edilebilecektir.
  - ✓ Şüpheli/sanık delillere herhangi bir zamanda, herhangi bir zamanda erişim sağlayabilecek ve dosya/fotokopi v.s. aşamalarında bulunması gerekmeyecektir.
  - ✓ Deliller belirli bir sistem içerisinde saklanabilecek ve şüpheli/sanık dosya prosedürleri içeren karmaşası yaşamadan delillere gerçek anlamda doğrudan erişebilecektir.

Burada ortaya çıkan sonuç şüpheli/sanık haklarının, mağdur haklarının çatışması durumu günümüz teknoloji imkânlarından faydalanması suretiyle ortadan kalkabilecektir. Bu yönde yapılacak uygulama delillere erişimin kısıtlanması değil bu imkânın genişletilmesi ve düzenlenmedir. Ceza yargılamasında şüpheli/sanık haklarının korunması suretiyle mağdur haklarının engellenmesi mümkün değildir. Bir kısım teknik çalışmalarla mağdur haklarının korunması mümkündür.

Bu çalışmada yer alan Cube, ICDS bu yönde çalışmaların yapılabilmesi ve geliştirilmesi için tasarlanmıştır.

1.5. Bilgi Güvenliği

Yaygın literatüre bakıldığında bilgi güvenliğinin internetin gelişimine ilişik tutulduğu görülür. Bu bilgi güvenliğinin daha geniş çevrelerce daha sık tartışılması yönünden doğru bir yaklaşımdır. Ancak bilgi internetten çok daha eski tarihlerde güvenlik altına alınmak istenmiştir. En çok bilinen Sezar şifresi M.Ö. 60-50 yıllarında Julius CEASAR tarafından kullanılmaya başlanmıştır. İnternet sadece bilginin daha kolay yayılmasını sağlamış ve veri yığınlarını oluşturmuştur. Bu veri yığını korunması gereken daha fazla bilgiyi meydana getirmiştir. Burada ele alınması gereken asıl durum gerekli tedbirlerin her zaman gecikmelerle gerçekleştirilmesidir.

Bugün gsm şebekelerinden telsiz haberleşmesine, el yazısı mektuplardan e-posta sunucularına uzanan her alan bilgi güvenliği ihtiyacındadır.

Bireysel kapsamda genel olarak mahremiyet ile ilişkilendiriliyor olsa da durum daha vahimdir. Bireylerin rutinlerini elektronik ortamda saklayıp-işleyen kurumlar tarafından bakıldığında bireyin sağlık-hastalık kayıtları, ilaç kullanım rutinleri gibi doğrudan hayati unsurların yine bu ortamlarda yer aldığı görülür. Benzeri bir durdum

bireylerin yaşamlarını kolaylaştıran her tür araçta görülebilmektedir. Bu çalışmanın hazırlandığı tarihte otonom sisteme geçiş çalışmalarının devam ettiği ulaşım araçları veya nesnelerin internetine konu olan akıllı ev sistemleri bireylerin soyut alan dışığında somut tehditlerle karşı karşıya gelebileceği örneklerden sadece birkaçıdır.

Kurumsal kapsamda, kurumların varlığını tehdit edebilecek durumların ele alındığı görülür. Bu genel olarak enformasyon ile ilişkilendirilirken SCADA[14] sistemleri kurumların askeri veya diğer şebeke alt yapılarındaki asıl tehdit unsurlarını oluşturur. İran Natanz nükleer tesislerinde yaşanan patlamalara, bu tesise özel tasarlanmış stuxnet bilgisayar virüsünün yol açması bilinen bir örneği oluşturur. Bu olayda stuxnet belirlenmiş bir hedefe saldırmıştı. Bu hedefte olduğunu-ulaştığını tespit etmesi için hazırlanan betik ise tesisin daha öne basına yansıyan fotoğraflarından yola çıkılarak santrifüj sayısını ve yapısını arıyordu. Bu örnekte sadece saldırı değil, bilgi güvenliği istismarının ortaya çıktığı görülür.

Bugün bilgi güvenliği daha tedbirli yaklaşımları gerektirir. Bunun sebebi hiç kuşkusuz internetin yaygınlaşmasıdır. Ancak bu tehdidi oluşturanın internet değil onu kullanan ve geliştiren insan olduğu unutulmamalıdır. İnternet istenenler kadar istenmeyen bilgilerinde ışık hızıyla dünyaya yayıldığı alt yapıyı sunar.

14 SCADA: Supervisory Control And Data Acquisition

2. KRİPTOLOJİYE GİRİŞ

2.1. Giriş

Kriptoloji bir veri veya mesajın şifrelenmesini ve çözülmesini inceleyen bilim dalıdır. İsmini Yunanca gizli anlamına gelen "krptòs"'dan alır. Bu bilim dalı kendi içerisinde iki ana alt dala sahiptir, ilki şifreleme yöntemlerini inceleyen ve geliştiren kriptografi, ikincisi bu yöntemlere saldırılar düzenleyen ve gizli verileri çözümlemeyi içeren kriptanalizdir.

Kriptolama – şifreleme veri içerisindeki bilgileri hedef dışındaki kişilerden gizlemeyi amaçlar. Ancak burada gizlenen veri değil, veri içerisinde yer alan bilgilerdir. Veri şifrelenmiş ve anahtarı olmadığında anlamsızlaştırılmış olarak açıktan gönderilebilmektedir. Bu verinin değil içeriğinin gizlenmesiyken, verinin kendisinin de gizlendiği durumlar steganografi denilen bir başla alanı oluşturmaktadır.

Kriptografi alt sınıfı olan hash fonksiyonlarıyla e-imza, kimlik doğrulama, veri bütünlüğü ispatı gibi alanlarda da birçok soruna çözümler üretmektedir. Bu fonksiyonlar kullanıcılar tarafında farkında olmaksızın hergün başvurulan yöntemlerdir.

Bugün erişim denetimi, bilgi gizliliği, ağ güvenliği, ATM-kredi kartları, şifre kodlama-gizleme, e-ticaret, e-imza, kimlik ispatı, veri bütünlüğü ispatı, virüs tarama sistemleri, dijital delilin bütünlüğü, simetrik-asimetrik şifrelemede anahtar oluşturma gibi birçok alanda hash algoritmaları kullanılır.

Şifreleme yöntemleriyse askeri telsiz-haberleşme sistemleri, web siteleri, e-postalar, dijital ortam dosyaları, telefon-gsm sistemleri gibi sayısız temel alanda kullanılırken sınırları bu alanda belirtmek mümkün değildir. Şifreleme her birey ve her

kurumun günlük rutinlerine dâhil olan bir alandır ve kriptografi bu alanın canlı ve güvende olmasını sağlayan bilim dalıdır.

2.2. Tarihi

Örnekli bilgi saklama-gizleme tarihine baktığımızda Mısırda yer alan sıra dışı hiyograflara kadar uzanabilen bir listeyle karşılaşabiliriz. Ancak tarihsel olarak gizli haberleşmenin en çok bilineni M.Ö. 60'da Julius CAESAR'ın bugünde halen kullanılan, metindeki karakterlerin alfabetik sıradaki 3 harf sonraki karakterle değiştirilmesine dayalı Sezar yöntemi en çok bilinen yöntem olacaktır.

Rönesans döneminde Johannes TRITHEMIUS'un şifreleme üzerine yazdığı 3 kitabı ve stego metodu bir başka örneği oluşturur. Burada sıralı sütunlara yerleştirilen kelimelerin ilk harflerinin birleştirilmesiyle mesajın ortaya çıkarılması yöntemi söz konusudur. Bugün edebi yazımlarda rastlanılan akrostiş uygulaması ismini bu yöntemden alır. 1500 yılında yazdığı "*Stegonographia*" isimli eserinde ilk defa "*Akrostiş*" terimini kullanan Johannes TRITHEMIUS bu uygulamanın hem fikir, hem isim sahibi olarak bilinen kişidir (7).

Şifreleme yöntemleri oldukça eskilere dayanır. Hatta insanoğlunun iletişim kurmaya başladığı zamana kadar genişletilebilecek bir zaman dilimi söz konusudur. Bilinen şifreli haberleşme yöntemleri eski Yunanlılara dayandırılır. M.Ö. 7. Yüzyılda Yunanlı şair Archilochus, gizli haberleşme için, aşağıda Şekil 1 ile verilen "*scytale/skıtéli*" isimli bir çubuk kullanmıştır. Gönderici, şerit halinde ki kâğıdı çubuğa sarar ve mesajını bu çubuk üzerine boylamasına yazar, daha sonra kâğıdı bu çubuktan ayırarak alıcıya gönderirdi. Çubuğun çapını (burada bu anahtar oluyor) bilmeyen

kimseler mesajı okuyamazdı. Ancak çubuğa yada çubuğun oran bilgilerine sahip olan alıcı kâğıdı çubuğa sararak mesajı elde edebilirdi (7).

Şekil 1: Acrchilochus'un skytale çubuğu[15]

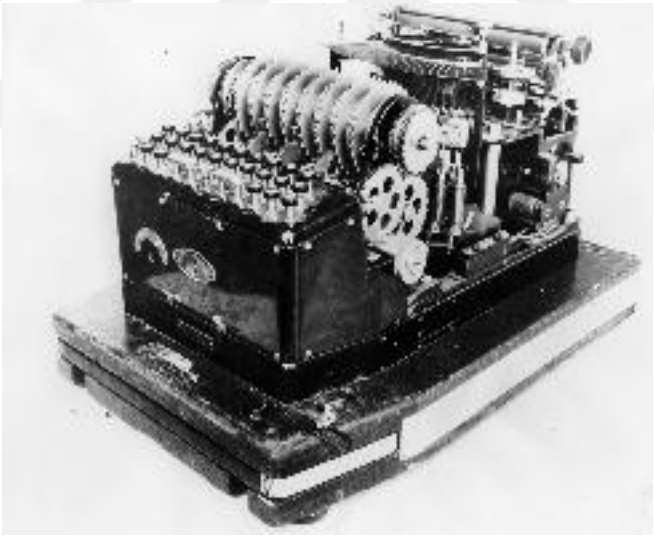
Yunanlı tarihçi ve bilim insanı olan Polybius'un Polybius tablosu da yine tarihte bilinen bir örneği oluşturur. Burada alfabedeki karakterlerin bir tabloya yerleştirilmesi ve iletilmek istenen mesajın bu tablonun satır-sütun sırasına göre kodlanması izlenir. Aşağıda yer alan Tablo I. **Polybius tablosu** örneğine göre "*ÜSKÜDAR ÜNİVERSİTESİ-BABE*" kelimesi kodlandığında sonuç "*26 22 13 26 04 00 21 99 26 16 11 27 05 21 22 11 24 05 22 11 24 05 22 11 46 01 00 01 05*" olur.

Tablo I. Polybius tablosu

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
0	A	B	C	Ç	D	E	F	G	Ğ	H
1	I	İ	J	K	L	M	N	O	Ö	P
2	Q	R	S	Ş	T	U	Ü	V	W	X
3	Y	Z	0	1	2	3	4	5	6	7
4	8	9	!	?	.	+	-	/	*	a
5	b	c	ç	d	e	f	g	ğ	h	ı
6	i	j	k	l	m	n	o	ö	p	q
7	r	s	ş	t	u	ü	v	w	x	y
8	z	=	(	)	%	&	^	"	{	}
9	[	]	@	;	,	:	\	$	#	

[15] 23.08.2019 tarihinde http://upload.wikimedia.org/wikipedia/commons/5/51/Skytale.png linkinden alınmıştır.

2. Dünya savaşına gelindiğinde kriptografinin önemi iyice açığa çıkmıştır. Bugünde devam eden uluslar arası enformasyon savaşlarının ivme kazanması 2. Dünya Savaşında gerçekleşmiştir. Almanların Arthur SCHERBIUS ile geliştirdiği Enigma makinesi, bir board ve üç adet birbiriyle kombine edilebilir rotor parçasından oluşmaktaydı. Bu rotor parçaları her algoritma işleminden önce sökülerek farklı kombinasyonlarda yeniden takılırdı. Her mesaj öncesinde cihazın altında yer alan soket dizilimleri de değiştirilir ve şifreleme daha güçlü hale getirilirdi. Bu cihazın anahtar uzunluğu 116-bit değerindeydi[16.] Bu cihazın algoritması, İngilizlerin Alan TURING ile birlikte Londra Bletchley Parkta gerçekleştirdiği ve Colossus adlı tüplü bir bilgisayar yardımıyla Ultra Kod adlı çalışmayla kırılmıştır.



Şekil 2: Enigma cihazı

16 Bu anahtar uzunluğu bugün en güçlü simetrik şifreleme algoritmalarının kullandığı anahtar uzunluğuna çok yakındır.

Yukarıda anlatılanlar dışığında kriptografide en çok bilinenlere dair kronografik sıralama aşağıda verildiği gibidir;

- M.Ö. 1990-1800: Mısırlı bir kâtibin yazdığı sıra dışı hiyeroglifler içeren yazıtlar.
- M.Ö. 60-50: Julius CAESAR'ın kelimelerin karakterlerini/harflerini alfabetik sıralamada 3 sonrasında yer alan karakterle değiştiren Sezar Yöntemi.
- M.S. 725-790: Abu al-Rahman'ın Bizans imparatoru için, Yunanca yazılmış şifreli metni çözmesinden aldığı ilhamla yazdığı kitabı.
- M.S. 1000-1200: Gazeliler tarafından yazılan şifrelenmiş metinler bulunmuştur. Bu metinlerde üst düzey devlet görevlilerinin yazışmaları yer almaktadır. Bu görevliler görev yerlerine giderken şahıslarına özel şifreleme bilgileri de kendilerine verilmekteydi.
- M.S. 1585: Blaise de VIGENERE (1523-1596) şifreleme üzerine yazdığı kitabında açık metin (plaintext) ve şifreli metin (ciphertext) için otomatik anahtar (cipher) yöntemini ilk kez ortaya atmıştır. Bugün bu yöntem kapalı metin zincirleme ve şifreyi geri besleme yöntemlerinin temelini oluşturur.
- M.S. 1623: Sır Francis BACON, 5-bitlik ikili kodlamayla karakter değişimine dayalı yeni bir yöntem buldu.
- M.S. 1970: Thomas JEFFERSON, geliştirdiği Strip Cipher makinesiyle ABD'nin 2. Dünya savaşında kullandığı M138-A cihazının temelini oluşturdu.

- M.S. 1883: Auguste KERCHOFF yazdığı 62 sayfalık "*La Cryptographie Militaire*" [17] adlı kitabıyla günümüzde de kullanılan birçok uygulama ve prensibe temel oluşturmuştur.
- M.S. 1917: Joseph MAUBORGNE ve Glibert VERNAM one-time-pad şifreleme yöntemini bulmuşlardır.
- M.S. 1920-1930: İçki kaçakçılarıyla mücadele etmek isteyen FBI, haberleşme yöntemlerinin araştırılması için bir birim kurmuştur. William Frederick FRIEDMAN bu birimden doğan Riverbank Laboratuarlarını kurmuş ve ABD için kriptanaliz çalışmaları yapmıştır. 2. Dünya savaşında da Japonlara ait Purple Machine isimli şifreleme sistemi bu laboratuarlarda yapılan çalışmalar ile çözülmüştür.
- M.S. 190-1940: 2. Dünya savaşı sırasında Almanlar Arthur SCHERBIUS ile birlikte Enigma isimli şifreleme cihazını geliştirmiştir. Bu cihazın şifreleri Almanların bütün mesajlarının sonuna "*Hail Hitler*" metnini koymalarından oluşan hataları sonucu İngilizlerle çalışan Alan TURING ve ekibi tarafından kırılmıştır.
- M.S. 1952: ABD, NSA ismiyle yeni bir kurum oluşturdu. Bu kurum bugünde dünyadaki haberleşme sistemlerinin önemli bir bölümünü izlemekte ve arşivlemektedir. Sivillere ait haberleşmeleri yasalara aykırı yollarla takip ettikleri ve şirketleri bu bilgileri vermeye teşvik ettikleri bilinmektedir. Bu kurum ayrıca halen birçok ülkenin kullandığı SHA hash - mesaj özetleme ailesinin mimarıdır. Türkiye Cumhuriyeti'nin de dahil olduğu bu ülkeler e-

17 La Cryptographie Militaire: Askeri Kriptografi.

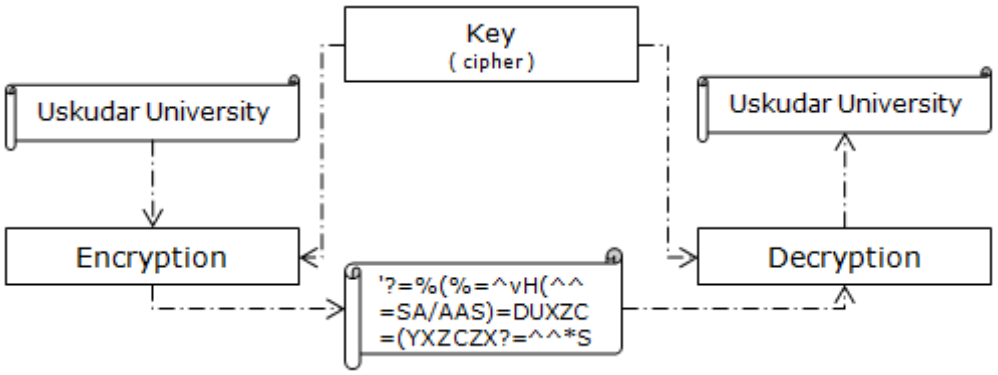
imza ve diğer kriptografik altyapılarında bu SHA ailesini kullanarak bilgi güvenliklerini temin etmeye çalışmaktadırlar.

- M.S. 1970: IBM'de görevli olan Horst FEISTEL geliştirdiği Lucifer isimli şifreleme yöntemiyle DES ve 3DES'in temelini oluşturmuştur. Ayrıca kendi ismi ile anılan Feistel Ağları bugün blok şifreleme yöntemlerinin temelini oluşturmaktadır.
- M.S. 1976: ABD, FIPS-46 koduyla standartlaştırdığı DES algoritmasını açıkladı. Bir dönem hükümet kurumlarında kullanılan bu algoritmaya karşı düzenlenen başarılı diferansiyel saldırılar sonucu anahtar değerlerini üç kat arttıran 3DES açıklanmıştır. Devamında 3DES'inde kullanılmasından vazgeçilerek yerine Rijndael isimli algoritma kullanılmaya başlanmıştır.
- M.S. 1976: Whitfield DIFFIE ve Martin HELLMAN açık (public) – gizli (private) anahtar sistemine dair makalelerini yayınladılar. Bu makale aslında bir anahtar paylaşım protokolünü açıklıyordu.
- M.S. 1978: Ronald L.RIVEST, A. SHAMIR ve Leonard M. ADLERMAN asal sayıların çarpanlarına ayrılması zorluğuna dayalı ve RSA olarak bilinen açık (public) – kapalı (private) anahtar yöntemini yayınladılar.
- M.S. 1985: Neal KOBLITZ ve Victor S. MILLER çalışmalarında Eliptik Eğri Şifresini açıklamışlardır.
- M.S. 1990: Xuejla LAI ve James MASSEY, IDEA isimli şifreleme algoritmasını geliştirdiler.
- M.S. 1991: Phil ZIMMERMAN, PGP ismini verdiği şifreleme sistemini geliştirmiş ve ücretsiz olarak dağıtmıştır.

- M.S. 1995: NIST 160-bitlik değerler üreten SHA1 isimli mesaj özetleme – hash fonksiyonunu standart olarak duyurmuştur.
- M.S. 2001: NIST'in 1997'de başlattığı bir yarışmayla DES'in yerini alacak bir şifreleme algoritması araması sonucu Belçikalı Joan DAEMEN ve Vincent RIJMEN'e ait Rijndael algoritması AES adıyla standartlaşan yeni şifreleme algoritması olmuştur. Bu algoritma 128, 192 ve 256-bitlik anahtar bloklarında iteratif yapıda kodlama yapar.
- M.S. 2005: Çinli bir ekip SHA1 ve MD5 olarak bilinen mesaj özetleme algoritmalarında çakışmalar üretebildiklerini – kırdıklarını açıkladılar. Bu bir kısım ülkelerin ve şirketlerin bu algoritmaları kullanmaktan vazgeçtiklerini duyurmasıyla devam eden bir süreci başlattı.
- M.S. 2009: Belçika'da Kathalieke Üniversitesi 25-28 Şubat 2009 tarihleri arasında ön elemelerini gerçekleştirdiği kriptografi olimpiyatlarını başlattı. Bu olimpiyatlarda SHA1 yerine yeni bir mesaj özetleme algoritması arandı. SHA2 olarak adlandırılan fonksiyonun atası bu olimpiyatlar olmuştur.

2.3. Terminoloji

Bir kriptosistem herhangi bir verinin yetkilisi olanlar dışığında elde edilmesini engelleyen matematiksel bir dönüşümü ifade eder.



Şekil 3: Kriptolama / Şifreleme

Kriptoloji terminolojisinde bir kısım temel terimler şöyledir:

- Açık / şifrelenmemiş veri veya mesaj; düz metin (plaintext) veya temiz/açık metin (cleartext) olarak adlandırılır.
- Şifreleme işleminde kullanılacak şifreye anahtar (cipher veya key) denilir.
- Düz metnin veya düz verinin, anahtar ile şifrelenmiş haline şifreli metin / şifreli veri (ciphertext veya cipherdata) denir.
- Düz metinden / veriden, şifrelenmiş metin / veri elde etme işlemine şifreleme (encryption) denir.
- Şifrelenmiş metinden / veriden, düz metin / veri elde etme işlemine şifreleme (decryption) denir.

Bir şifreleme ve şifre çözme işlemi çoğu algoritmada bir anahtar (cipher) kullanılarak yapılır. Teoride doğru anahtarın bilinmemesi halinde düz metin / veri elde etmek mümkün olmamalıdır.

Şifreleme bilimi ile ilgilenen kişilere kriptograf denir. Şifreleme algoritmalarını ve / veya şifreli metinleri çözmekle ilgilenenlere kriptanalist denir ve bu alan kriptanaliz olarak adlandırılır.

Simetrik şifre, şifrelemede kullanılan anahtarın şifrenin çözümündeki anahtar ile eş olmasını belirtir. Bu durumda göndericide alıcıda aynı anahtara sahiptirler.

Asimetrik şifre, bugünkü literatüre göre açık (public) ve gizli (private) anahtar yapılarından oluşur. Burada açık anahtar herkesçe bilinen ve yalnızca şifreleme yapan anahtardır. Gizli anahtar ise yalnızca şifreyi çözmeye yetkili kişilerce bilinen çözme işlemini yapan anahtardır. Bu yöntemlerin güvenliği bazı matematiksel problemlere dayanır[18].

Simetrik şifreleme yöntemlerinin bugün en çok bilinenleri blok ve feistel ağları üzerine kuruludur. Blok yapısı verinin bloklar halinde işlenerek şifrelenmesine dayalıdır. Blok yapısını kullanan algoritmalar genelde iteratif fonksiyonlar olarak çalışırlar[19].

2.4. Kriptoloji Matematiği

Kriptoloji matematik temelli bir bilim dalıdır. Ancak kullanım alanlarının getirdiği zorunluluklar nedeniyle bilgisayar ve elektronik bilim alanlarıyla iç içedir.

Kriptoloji matematiği özel bir ilgi ve eğitimi gerektiren bir alanı oluşturur. Her geçen gün alanında yeni tezlerin ve makalelerin sunulduğu alanın anlaşılması için geçmişte yapılan çalışmaların çok iyi irdelenmesi ve takip edilmesi gerekir.

Bu bölümde alt başlıklar halinde temel kavramlara yer verilmiştir.

18 Bu matematiksel problemler, çalışma devamında 3. KRİPTOGRAFİ ve KRİPTOSİSTEMLER ile verilmiştir.

19 Blok yapıları çalışma devamında 3. KRİPTOGRAFİ ve KRİPTOSİSTEMLER alt başlığı 3.3. Blok Kriptosistemler ile verilmiştir.

2.4.1. Asal Sayılar

Kendisi ve 1'den başka hiçbir sayıya bölünemeyen sayılara asal sayı denir. Asal sayılar asimetrik şifre algoritmalarının temelini oluşturur.

Asal sayılar ile ilgili, bu sayıların oluşturulması ve bir sayının asallığının sınanmasına dair fonksiyonlar en önemli konular olarak görülür.

$xy^{-1} \equiv 1 \, mod \, y$, $1 < x < y$ ve $x, y \in \mathbb{Z}^+$ olsun, burada y sayısına x tabanına göre sanki asal (pseudoprime) sayı denir.

Euler Totient fonksiyonu $x \in \mathbb{Z}^+$ için $e(x)$, x'den daha küçük ve x ile aralarında asal bütün tam sayıların sayısını verir.

- p asal ise $e(p) = p - 1$'dir.
- $x = p.q, p, q$ asal $\rightarrow e(x) = e(p).e(q) = (p-1).(q-1)$ olur.

Fermat teoremine göre x bir asal sayı ve $OBEB(y, x) = 1$ ise x, y tabanına göre sanki asal (pseudoprime) sayı olur.

2.4.2. Modulo

Modüler aritmetik asimetrik ve simetrik kriptosistemlerde anahtarların oluşturulmasında da kullanılan bir kısım sayısal işlemleri içerir. Ayrıca günümüzde sıkça kullanılan 3DES ve AES gibi simetrik kriptosistemlerin kullandığı Galois Field işlemlerinde veya stream tipi kriptosistemlerinde başvurulan işlemdir.

$x, y, z \in \mathbb{Z}$, $n \neq 0$ ve $x - y = z.t$, $x \equiv b \, mod \, z$ şeklinde tanımlanabilecek modulo işlemi ayrıca $a, b, c \in \mathbb{Z}$, $c \neq 0$ ve $\Delta = (+ \, veya \, *)$ olduğunda da $(a \, \Delta \, b) \, mod \, c \equiv [(a \, mod \, c) \, \Delta \, (b \, mod \, c)] \, mod \, c$ denkliğini sağlar.

Bir $x = y \, mod \, z$ eşitliği, x ve y aynı z ile bölünürse aynı kalanı verdiğini gösterir.

$x \equiv y \; mod \; z$ çoğu durumda $0 \leq y \leq z - 1$ olur.

$x \equiv y \; mod \; z$ için y, $x \; mod \; z$'nin kalanıdır.

$x \in \mathbb{Z}$ ise x ile ytapılan bütün mod değerleri 0 ile x arasında olur.

Tablo II. Mod alma örnekleri

1.	11 mod 3 = 2
2.	257 mod 256 = 1
3.	8 mod 2 = 0

Modüler aritmetik kendi içerisinde bir kısım işlemsel özellikler barındırır;

- Toplama için; $(x \; mod \; y) + \; (z \; mod \; y) = (x + z) \; mod \; y$
- Çıkartma için; $(x \; mod \; y \;) - (z \; mod \; y) = [x + (-z)] \; mod \; y$
- Çarpma için; $(x \; mod \; y).(z \; mod \; y) = (x.z) \; mod \; y$
    - $x, z \; \neq 0$ iken $x.z = 0$ olabilir.
        - *örnek:* $4.5 \; mod \; 20$
- Bölme için; $\frac{x}{y} \; mod \; z$, y'nin tersiyle çarpımı gibidir, $\frac{x}{y} = x.y^{-1} \; mod \; z$
    - Eğer z asal sayıysa $y^{-1} \; mod \; z$ vardır ve $y.y^{-1} \; mod \; z$ olur.
        - örnek: $2.3 \; mod \; 5$ ve $\frac{4}{2} = 4.3 = 2 \; mod \; 5$
- Birleşme için; $[(x + y) + z] \; mod \; t = [x + (y + z)] \; mod \; t$
- Geçişlilik için; $(x + y) \; mod \; z = (y + x) \; mod \; z$
- Dağılma için; $[(x + y).z] \; mod \; t = [(x.z) + (y.z)] \; mod \; t$
    - Ayrıca, indirgeme tamsayılar halkasında tamsayı modula z'lerin halkasına bir homomorfizm olduğundan, herhangi bir işlemden sonra $modula \; z$'yi indirgeyebilmesini veya devamında yapılacak işlem seçilebilir.
        - $(x \pm y) \; mod \; z = [(x \; mod \; z) \pm (y \; mod \; z)] \; mod \; z$

- $(x.y) mod\ z = [(x\ mod\ z).(y\ mod\ z)]\ mod\ z$

- Eğer x, y doğal sayısı olmaya zorlanırsa bu yapı Galois Field $modula\ y$ olur ve $GF(y)$ olarak gösterilir.
  - Bu sınırlı alan içerisinde de tam sayılar aritmetiğindeki kurallar geçerlidir.

2.4.3. Olasılık

Olasılık herhangi bir durumun yada olayın gerçekleşme şansını ifade eden bir sayı olarak tanımlanır ve şansa bağlı olayın sonucuna ilişkin bir çıkarımda bulunma aracı olarak kullanılabilir. 0 ve 1 dahil olmak üzere 0 ile 1 kapalı aralığında değerler alan ve bir olayın ortaya çıkma şansını belirten sayıya olasılık denir (8).

- Örnek: İki madeni para ile yazı-tura atma deneyinde 4 örneklem noktası vardır: $P = \{YY, YT, TT, TY\}$

Rastsal bir deneye ait örneklem uzayının olası her bir alt kümesine rastsal olay denir.

- Örnek: Bir yazı ve bir tura gelmesi olayı: $P = \{YT, TY\}$

Bir olayın gerçekleşmesi diğer olayın gerçekleşme olasılığı 0'a eşitliyorsa, bu iki olaya karşılıklı dışlamalı olaylar denir.

- Örnek: $P = \{YY, YT, TY\}$ ve $\{TT\}$ karşılıklı dışlamalı olaylardır.

Değerleri rastlantısal bir deneyle belirlenen değişkenlere rastsal değişken denir ve kısaca *"ro"* (*rv*) denir.

- ❖ Rastsal değişkenler X, Y, Z şeklinde büyük harflerle ve aldıkları değerler x, y, z şeklinde küçük harflerle temsil edilirler.
- ❖ Bir rastsal değişken kesikli (*discrete*) veya sürekli (*continuous*) formda olur.
- ❖ Kesikli rastsal değişken sadece sonlu sayıda farklı değerler alabilir.
  - o Örnek: Zar
- ❖ Sürekli rastsal değişken belirli bir aralıkta herhangi bir değeri alabilir.
  - o Örnek: Rastlantısal seçilmiş iki nokta arasındaki uzaklık.

X örneklem uzayında bir olay ve rastsal deney sürekli yenilenmeliyse, X olayının gerçekleşmesini belirten sıklık oranına X olayının gerçekleşme olasılığı denir. Bu durum $P(X)$ veya $Prob(X)$ ile temsil edilir.

- $P(X)$ gerçek değerli bir fonksiyondur.
- Her X için $0 \leq P(X) \leq 1$'dir.
- Örneklem uzayı $X_1, X_2, X_{\dots}, X_n$ ile oluşuyorsa $P(X_1, X_2, X_{\dots}, X_n) = 1$ olur.
- X_1, X_2 ve X_3 karşılıklı dışlamalı olaylar ise $P(X_1, X_2, X_{\dots}, X_n) = \big(P(X_1) + P(X_2) + P(X_3)\big)$ olur.

Sonlu $X = \{x_1, x_2, x_{\dots}, x_n\}$ örneklem uzayı, $P(\,)$ olasılık fonksiyonu ve $P(x_1) = P(x_2) = P(x_{\dots}) = P(x_n)$ olduğunda X örneklem uzayına eş olumlu (olasılı) denir.

- X eş olumlu örneklem uzay ve $s(X) = n$ olduğunda $x_1 \in X$ olma olasılığı $P(x_1) = \frac{1}{n}$ olur.
- X eş olumlu örneklem uzay ve $Y \subset X$ olduğunda Y olayının gerçekleşme olasılığı $\frac{s(Y)}{s(x)}$ olur.
    - Örnek: Bir torbada eşit boy, ağırlık ve yüzey yapılarına sahip 4 mavi, 2 kırmızı ve 2 yeşil bilye vardır. Bu torbadan rastsal olarak seçilen bir bilyenin mavi olma olasılığı için örneklem uzayı $X = \{m_1, m_2, m_3, m_4, k_1, k_2, y_1, y_2\}$ ve $s(X) = 8$ 'dir. Mavi bilye seçilmesi $= Y = \{m_1, m_2, m_3, m_4\}$ ve $s(Y) = 4$ 'tür. Bu durumda seçilen bilyenin mavi olma olasılığı $= P(Y) = \frac{s(Y)}{s(X)} = \frac{4}{8} = \frac{1}{2}$ olur.
    - Örnek: 2 adet 6 yüzlü hilesiz bir zar hilesiz olarak atıldığında her iki zarında aynı sayıda gelmesi olasılığı ne olur ? Burada $s(X) = 36$, $Y = \{(1,1), (2,2), (3,3), (4,4), (5,5), (6,6)\}$ ve $s(Y) = 6$ 'dır. Bu durumda $P(Y) = \frac{s(Y)}{s(X)} = \frac{6}{36} = \frac{1}{6}$ olur. Bu örnekteki 2 adet zara ait örneklem uzayı aşağıda Tablo III. 2 Ayrı Zar İçin Örneklem Uzayı ile verilmiştir.

Tablo III. 2 Ayrı Zar İçin Örneklem Uzayı

		2. zar					
		1	**2**	**3**	**4**	**5**	**6**
1.zar	**1**	1,1	1,2	1,3	1,4	1,5	1,6
	2	1,1	2,2	2,3	2,4	1,5	2,6
	3	3,1	3,2	3,3	3,4	1,5	3,6
	4	4,1	4,2	4,3	4,4	1,5	4,6
	5	5,1	5,2	5,3	5,4	1,5	5,6
	6	6,1	6,2	6,3	6,4	1,5	6,6

2.4.4. Sayı Sistemleri

Kriptoloji günlük yaşantıda kullanılan onluk sayı sistemiyle birlikte ikilik, sekizlik ve onaltılık sayı sistemlerini kullanır. Bunlar arasında en sık başvurulanlarını ikilik ve onaltılık sayı sistemleri oluşturur.

- İkilik sayı sistemi 0 ve 1 karakterlerini içerir. Bu karakterlere bit[20] denir. Bit değerleri elektronik devre elemanlarının da temelini oluştururlar. Bilgisayarlar bu sayı sistemi üzerine kurulu 0 (false) ve 1 (true) ikili durum makineleridir denilebilir.

  Aşağıda yer alan fonksiyon ile onlu sayı sistemindeki bir sayının ikilik sayı sistemine dönüşümü temsil edilmiştir.

$$\begin{cases} out = \{\emptyset\} \\ i = 0 \\ while\ 0 < x \\ \quad out_i = x\ mod\ 2 \\ \quad x = \frac{x-(x\ mod\ 2)}{2} \\ \quad i = i + 1 \end{cases} \qquad (1)$$

  Aşağıda yaralan fonksiyon ile ikili sayı sistemindeki bir sayının onluk sayı sistemine dönüşümü gösterilmiştir. Burada x ikilik sayı sisteminde bir sayıyı ve i, x'in basamağını temsil eder. Ayrıca i ile elde edilen basamak değerinin rakamsal karşılığı kullanılır ($x_i\ mod\ 10$) burada x'in n basamaklı olduğu varsayılmıştır.

$$\sum_{i=1}^{n}\left(x_i . 2^{i-1}\right) \qquad (2)$$

20 Bit: **Bi**nary Digi**t**

İkilik sayı sisteminin kullanılması çalışmanın devamında açıklanan bit düzeyinde mantıksal işlemlerde kolaylık sağlar. Mantıksal işlemler AND, OR, XOR ve NOT işlemleri olup, 0 ve 1'ler ile temsil edilen değerlerin bit basamaklarını kolayca işletebilmektedir. Bu işlemlerden XOR işlemi kriptografide şıkça başvurulan bir işlemdir.

- Sekizlik sayı sistemi 0, 1, 2, 3, 4, 5, 6 ve 7 karakterlerini içerir. Bir kısım elektronik şemalarda ve eski nesil şifreleme yöntemlerinde kullanılmıştır. Ayrıca günümüzde küresel anlamda en çok kullanılan sunucu işletim sistemleri olan Unix/BSD ve Linux aileleri dosya sistemlerinde yetkilendirme için (*chmod*) bu sayı sistemini kullanırlar.

Aşağıda yer alan fonksiyon ile onlu sayı sistemindeki bir sayının sekizlik sayı sistemine dönüşümü temsil edilmiştir.

$$\begin{cases} out = \{\emptyset\} \\ i = 0 \\ while\ 0 < x \\ \quad out_i = x\ mod\ 8 \\ \quad x = \frac{x-(x\ mod\ 8)}{8} \\ \quad i = i + 1 \end{cases} \quad (3)$$

Aşağıda yer alan fonksiyon ile sekizli sayı sistemindeki bir sayının onluk sayı sistemine dönüşümü gösterilmiştir. Burada x

sekizlik sayı sisteminde bir sayıyı ve i, x'in basamağını temsil eder. Ayrıca i ile elde edilen basamak değerinin rakamsal karşılığı kullanılır ($x_i \, mod \, 10$) burada x'in n basamaklı olduğu varsayılmıştır.

$$\sum_{i=1}^{n} \left(x_i . 8^{i-1} \right) \quad (4)$$

- . Onaltılık sayı sistemi $0, 1, 2, 3, 4, 5, 6, 7, 8, 9, a, b, c, d, e$ ve f karakterlerini içerir. Burada $a = 10, b = 11, c = 12, d = 13, e = 14$ ve $f = 15$ değerlerini temsil ederler. Onaltılık sayıları temsil eden bu karakterlere hex[21] değerleri de denilmektedir. Bilgisayar ve elektronik bilimlerinde en sık başvurulan sayı sistemidir. Temel/Alt düzey makine programlamasında bu sayı sistemi kullanılır. İkilik sayı sisteminde toplam 4 basamağı, yani 4-biti temsil edebilir. Bu yapısı alan ve hesaplama maliyetini düşük tutar. Kriptografik mesaj özetleme (hash) fonksiyonlarının ürettiği değerler bu sayı sistemi ile temsil edilirler.

  Aşağıda yer alan fonksiyon ile onlu sayı sistemindeki bir sayının onaltılık sayı sistemine dönüşümü temsil edilmiştir.

21 Hex: Taban olarak 16'yı kabul eden sayı sistemi. Bu sistemde, 0-9 arası sayılara ilave olarak A, B, C, D, E ve F harfleri sayı olarak kullanılır.

$$\begin{cases} out = \{\emptyset\} \\ i = 0 \\ while\ 0 < x \\ \quad out_i = x\ mod\ 16 \\ \quad x = \frac{x-(x\ mod\ 16)}{16} \\ \quad i = i + 1 \end{cases} \quad (5)$$

Aşağıda yer alan fonksiyon ile onaltılık sayı sistemindeki bir sayının onluk sayı sistemine dönüşümü gösterilmiştir. Burada x sekizlik sayı sisteminde bir sayıyı ve i, x'in basamağını temsil eder. Ayrıca i ile elde edilen basamak değerinin rakamsal karşılığı kullanılır ($x_i\ mod\ 10$) burada x'in n basamaklı olduğu varsayılmıştır.

$$\sum_{i=1}^{n}\left(x_i . 16^{i-1}\right) \quad (6)$$

Aşağıda yer alan Tablo IV ile ikilik, sekizlik, onluk ve onaltılık sayı sistemlerindeki ilk 32 değerin karşılıkları verilmiştir.

Tablo IV. 0-31 aralığında sayma sistemleri

Onlu	İkilik	Sekizli	Onaltılık	Onlu	İkili	Sekizli	Onaltılık
0	00000000	00	00	16	00010000	20	10
1	00000001	01	01	17	00010001	21	11
2	00000010	02	02	18	00010010	22	12
3	00000011	03	03	19	00010011	23	13
4	00000100	04	04	20	00010100	24	14
5	00000101	05	05	21	00010101	25	15
6	00000110	06	06	22	00010110	26	16
7	00000111	07	07	23	00010111	27	17
8	00001000	10	08	24	00011000	30	18
9	00001001	11	09	25	00011001	31	19
10	00001010	12	0A	26	00011010	32	1A
11	00001011	13	0B	27	00011011	33	1B
12	00001100	14	0C	28	00011100	34	1C
13	00001101	15	0D	29	00011101	35	1D
14	00001110	16	0E	30	00011110	36	1E
15	00001111	17	0F	31	00011111	37	1F

2.4.5. Elektronik Ortam Değerleri

Bilgisayar ve mikro denetleyicili elektronik devrelerin kendilerine has lisanı vardır. Bu sistemler giriş birimlerinden aldıkları verileri önce kendilerinin anlamlandırabileceği bir dile çevirerek kayıt ederler yada işlerler. Devamında saklanmış yada işlenmiş olan veriler bizlerin anlamlandırabileceği formdaki karakterler ile çıkış birimlerine gönderilir.

Bilgisayar sistemlerinde verilerin saklanmasında ve işlenmesinde kullanılan temel yapı ikilik sayı sistemidir. Bu yapı bole cebri olarak ele alınır ve 0 (yanlış-false) – 1 (doğru-true) değerlerine sahiptir.

0 ve 1 ile temsil edilen değerlere BIT denir. Bit kavramı **Bi**nary Digi**t**'ten türetilmiştir.

Bit değerlerinin bir araya gelmesiyle oluşan bit topluluklarının genel kabul görmüş tanımları mevcuttur. Bu değerler bilgisayarlarda değişkenler olarak ele alınır ve CTS[22] kurallarına göre oluşturulurlar.

Değişkenler bit temeliyle oluşturulduklarından dolayı değerleri 2^x boyutlarında olabilir. Terminolojide bu değişkenlere kelime (word) ve kapladığı bit alanına göre kelime uzunluğu (word length) denilir.

İkilik sayı sistemiyle saklanan bir değişkenin negatif değer olup olmayacağı işaretli veya işaretsiz olmasıyla belirlenir. Bir değerin işaretsiz olması doğrudan ikilik tabandaki karşılığıyla, işaretli olması eksi sayıları da kapsayan karşılığıyla saklanmasını sağlar. İşaretli değişkenler iki farklı yapıda olabilir; İlkinde eksi sayılar 2'ye tümleyen aritmetiğiyle elde edilir yani bire tümleyene 1 eklemeyle yapılır.

- **Örnek:** $(0000\ 0001 = +1)$ 'in 1'e tümlenişi aşağıdaki şekilde olur:

$$\begin{array}{r} 1111\ 1110 \\ \qquad\quad 1 \\ +\_\_\_\_\_ \\ 1111\ 1111 = -1 \end{array}$$

22 Common Type System

İkincisinde değişkelinin en soldaki (en anlamlı) biti işaret biti olarak kullanılır, diğer bitlerse değerin ikili tabandaki karşılığıdır.

Aşağıda yer alan Tablo V**.** Bilgisayar Sistemlerinde Değişken Özellikleri ile genel kabul görmüş değişkenlerin özellikleri verilmiştir.

Tablo V. Bilgisayar Sistemlerinde Değişken Özellikleri

Değişken Adı	**Bit Uzunluğu**	**İşaretli**	**Değer Aralığı**
byte	8-Bit	-	$0 \leq \cdots \leq (2^8 - 1) \in \mathbb{Z}^+$
sbyte	8-Bit	+	$-(2^7) \leq \cdots \leq (2^7 - 1) \in \mathbb{Z}$
short	16-Bit	+	$-(2^{15}) \leq \cdots \leq (2^{15} - 1) \in \mathbb{Z}$
int16	16-Bit	+	$-(2^{15}) \leq \cdots \leq (2^{15} - 1) \in \mathbb{Z}$
int32	32-Bit	+	$-(2^{31}) \leq \cdots \leq (2^{31} - 1) \in \mathbb{Z}$
long	64-Bit	+	$-(2^{63}) \leq \cdots \leq (2^{63} - 1) \in \mathbb{Z}$
float	32-Bit	+	$\pm(1{,}5)10^{-45} \leq \cdots \leq \pm(3{,}4)10^{38} \in \mathbb{R}^{\pm}$
double	64-Bit	+	$\pm(5{,}0)10^{-384} \leq \cdots \leq \pm(1{,}7)10^{308} \in \mathbb{R}^{\pm}$

Yukarıda yer alan tabloda işaretli (+) olarak verilen short, int16, int32 ve long değişken tipleri işaretsiz olarak da kullanılabilmektedir. İşaretli değişkenin, işaretsiz olarak kullanılması durumunda x değişkenin bit uzunluğuyken, değişken tipinin değer aralığı $0 \leq \cdots \leq (2^x - 1) \in \mathbb{Z}^+$ olur.

Ayrıca büyük verilerin tanımlanmasında bazı kısaltmalar kullanılmaktadır. KiloByte(KB), MegaByte(MB), şeklinde tanımlanan bu kısaltmalarda bit ile byte değerlerinin karıştırılması söz konusu olabilmektedir. Bu sebeple bit tanımlamalarında "b", byte tanımlamalarında "B" kullanılmaktadır (9).

Yukarıda açıklandığı gibi bilgisayarlar ikili durum makineleridir ve ikilik sayı sistemiyle çalışırlar. Günlük kullanımda bir değerin 1000 katı anlamına gelen Kilo (K) kavramı elektronik ortamda farklılaşır. 1K (1000) değeri ikilik sayı sistemindeki değişkenlerde $2^{10} = 1024$ olarak görülür, yani "K" karakterinin sayısal karşılığı 1024 olur. Bu bilgiden hareketle büyük verilerin ölçülmesinde kullanılan tanımalar aşağıda yer alan Tablo VI**.** Veri Ölçü Değerleri ile verilmiştir.

Tablo VI. Veri Ölçü Değerleri

Değer	**Sembol**	**Bit Değeri**	**Değer**	**Sembol**	**Bit Değeri**	**Üs Değeri**
Bit	bit		Byte	B	2^8	8-Bit
Kilobit	Kb(it)	2^{10}	Kilobyte	KB	2^{18}	1024-Byte
Megabit	Mb(it)	2^{20}	Megabyte	MB	2^{28}	1024-Kilobyte
Gigabit	Gb(it)	2^{30}	Gigabyte	GB	2^{38}	1024-Megabyte
Terabit	Tb(it)	2^{40}	Terabyte	TB	2^{48}	1024-Gigabyte
Petabit	Pb(it)	2^{50}	Petabyte	PB	2^{58}	1024-Terabyte
Exabit	Eb(it)	2^{60}	Exabyte	EB	2^{68}	1024-Petabyte
Zettabit	Zb(it)	2^{70}	Zetabyte	ZB	2^{78}	1024-Exabyte
Yettabit	Yb(it)	2^{80}	Yettabyte	YB	2^{88}	1024-Zetabyte

Bilgisayar bilimlerinde her karakter bilgisayarların anlayabilmesi için sayılar ile temsil edilerek kodlanır. Bu karakterlerin oluşturulmasında ASCII, EBCID gibi birçok farklı başvuru tablosu mevcuttur. Ancak günümüzde en sık başvurulan tablo ASCII tablosudur. ASCII tablosu 0-255 aralığında toplam 256 ayrı karakteri temsil eder. Bu karakterlerden 0-32 aralığındaki karakterler işlevsel amaçlarla ayrılmış ve görüntüsüzdürler.

Aşağıda yer alan Şekil 4**:** ASCII Kodlama-Karakter Listesi ile 0-255 aralığındaki 256 adet ASCII karakterinin listesi verilmiştir.

000	NUL	033	!	066	B	099	c	132	ä	165	Ñ	198	ã	231	þ
001	Start Of Header (SOH)	034	"	067	C	100	d	133	à	166	ª	199	Ã	232	Þ
002	Start Of Text (STX)	035	#	068	D	101	e	134	å	167	º	200	╚	233	Ú
003	End Of Text (ETX)	036	$	069	E	102	f	135	ç	168	¿	201	╔	234	Û
004	End Of Transmission (EOT)	037	%	070	F	103	g	136	ê	169	®	202	╩	235	Ù
005	Enquiry	038	&	071	G	104	h	137	ë	170	¬	203	╦	236	ý
006	Acknowledge (ACK)	039	'	072	H	105	i	138	è	171	½	204	╠	237	Ý
007	Bell	040	(	073	I	106	j	139	ï	172	¼	205	═	238	¯
008	Backspace (BS)	041	)	074	J	107	k	140	î	173	¡	206	╬	239	´
009	Horizontal Tab	042	*	075	K	108	l	141	ì	174	«	207	¤	240	-
010	Line Feed **(LF)**	043	+	076	L	109	m	142	Ä	175	»	208	ð	241	±
011	Vertical Tab	044	,	077	M	110	n	143	Å	176	░	209	Ð	242	‗
012	Form Feed **(FF)**	045	-	078	N	111	o	144	É	177	▒	210	Ê	243	¾
013	Carriage Return **(CR)**	046	.	079	O	112	p	145	æ	178	▓	211	Ë	244	¶
014	Shift Out	047	/	080	P	113	q	146	Æ	179	│	212	È	245	§
015	Shift In	048	0	081	Q	114	r	147	ô	180	┤	213	ı	246	÷
016	Dataline Escape (DLE)	049	1	082	R	115	s	148	ö	181	Á	214	Í	247	¸
017	DC 1 (XON)	050	2	083	S	116	t	149	ò	182	Â	215	Î	248	°
018	DC 2	051	3	084	T	117	u	150	û	183	À	216	Ï	249	¨
019	DC 3 (XOFF)	052	4	085	U	118	v	151	ù	184	©	217	┘	250	·
020	DC 4	053	5	086	V	119	w	152	ÿ	185	╣	218	┌	251	¹
021	Negative Acknowledge (NAK)	054	6	087	W	120	x	153	Ö	186	║	219	█	252	³
022	Synchronous Idle	055	7	088	X	121	y	154	Ü	187	╗	220	▄	253	²
023	End Of Transmission Block	056	8	089	Y	122	z	155	ø	188	╝	221	¦	254	■
024	Cancel	057	9	090	Z	123	{	156	£	189	¢	222	Ì	255	
025	End Of Medium	058	:	091	[	124	|	157	Ø	190	¥	223	▀		
026	Substitude	059	;	092	\	125	}	158	×	191	┐	224	Ó		
027	Escape **(ESC)**	060	<	093	]	126	~	159	ƒ	192	└	225	ß		
028	File Seperator	061	=	094	^	127 (DEL)	⌂	160	á	193	┴	226	Ô		
029	Group Seperator	062	>	095	_	128	Ç	161	í	194	┬	227	Ò		
030	Record Seperator	063	?	096	`	129	ü	162	ó	195	├	228	õ		
031	Unit Seperator	064	@	097	a	130	é	163	ú	196	─	229	Õ		
032	*SPACE* (SP)	065	A	098	b	131	â	164	ñ	197	┼	230	µ		

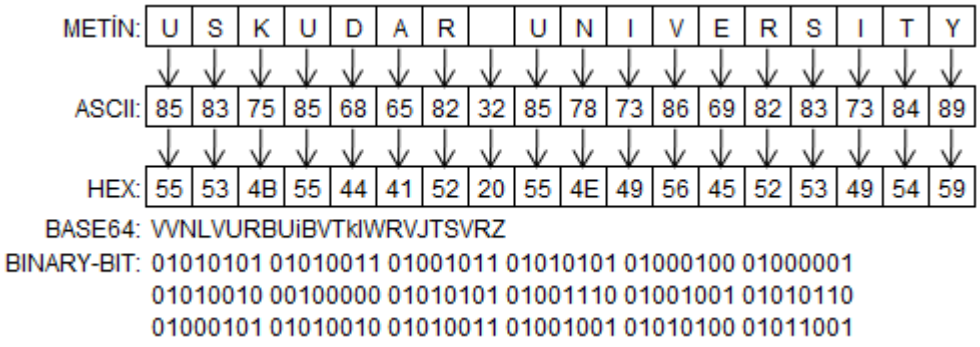
Şekil 4: ASCII Kodlama-Karakter Listesi

Ağ tabanlı haberleşme sistemlerinde kullanılmak üzere hazırlanan Base64 kodlama da ayrıca bu çalışma içerisinde başvurulan bir kaynaktır. Base64 ile ağ tabanıyla iletilecek verilerde görsel ve el ile kodlanabilecek karakterlerin standartlaştırılması sağlanmıştır. Base64 kodlama değerlerinin her biri 6-bitlik bir alan işgal eder. Buradan da anlaşıldığı gibi Base64 64 adet karakter ile temsil sağlar. 0-63 aralığındaki 64 adet Base64 karakteri aşağıda yer alan Tablo VII**.** Base64 Kodlama-Karakter Tablosu ile verilmiştir.

Tablo VII. Base64 Kodlama-Karakter Tablosu

Değer	Karakter	Değer	Karakter	Değer	Karakter	Değer	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	+

Yukarıda yer alan bilgilerle bilgisayar alanına işlenen bir metin örneği aşağıda verilmiştir;



2.4.6. Galois Field

Galois Field sonlu cisimlerin bir alt kümesini oluşturmaktadır ve kriptografi alanında yaygın olarak kullanılan bir yapıdır. Galois Field'ı tanımlamak için sırasıyla grup, halka, cisim ve sonlu cisimlerin tanımlanması gerekir.

- **Tanım 1 – Grubun düzeni (order):** G bir grup olduğunda bu gruba ait elemanların sayısı G grubunun düzeni olur, bu durum $ord(G) = |G|$ ile temsil edilir.

- **Tanım 2 – Çarpmalı grup:** H bir grup ve G grubunun alt grubu olduğunda $|H|$ değeri $|G|$ değerini böler. Bu durumda eğer G grubunun düzeni bir asal sayıysa G'nin tek alt grubu kendisi olur ve G grubu çarpmalı olarak yazılır.
    - G grubu çarpmalı olarak yazılabildiğinde $g \in G$ ve g sayısı G grubunun düzeniyse g sayısı $i \in \mathbb{N} \cup \{\infty\}$ ve $g^{'} = 1$ şartını sağlayan en küçük i değeri olur. Burada $\forall j, i \in \mathbb{Z} : g^{j} = g^{i} \leftrightarrow j \equiv i \; mod \; ord(g)$'dir (10).

- **Tanım 3 – Devirli grup (cyclic group):** G grubunun altındaki bütün gruplar g elemanın bir üssü olur ve $< g >$ ile temsil edilirler. Burada $< g >= G$ olduğunda g sayısı G grubunun üreteci olur. Bu tür gruplara devirli grup (cyclic group) denir.
    - G grubunun düzeni p asal sayısıysa, grupta yer alan 1 harici bütün sayılar G grubunun üreteci olur. Burada $< g >$'nin düzeni p veya 1 olur

- Fermat teoremine göre p bir asal sayı olduğunda $\forall x \in \mathbb{Z}$ için $x \equiv x \, mod \, p$ denkliği p ile bölünemeyen $\forall x \in \mathbb{Z}$ için $x^{p-1} \equiv 1 \, mod \, p$ denkliği her zaman doğrudur (11).

- **Tanım 4 - Değişmeli grup (Abel grubu):** Abel grubu, $< \mathbb{G}, + >$, bir $\mathbb{G}$ kümesi ve bu kümenin elemanları üzerinde tanımlanan bir '+' işleminden oluşmaktadır ve $+: \mathbb{G} \, \boldsymbol{x} \, \mathbb{G} \rightarrow \mathbb{G}: (\boldsymbol{a}, \boldsymbol{b}) \rightarrow \boldsymbol{a} + \boldsymbol{b}$ 'dir.

  $< \mathbb{G}, + >$ grubunun bir Abel grubu olabilmesi için ' + ' işleminin sağlaması gereken koşullar vardır:

  - **Kapalılık Özelliği**: $\forall \, a, b \in \mathbb{G}: (a + b) \in \mathbb{G}$
  - **Birleşme Özelliği**: $\forall \, a, b, c \in \mathbb{G}: (a + b) + c = a + (b + c)$
  - **Değişme Özelliği**: $\forall \, a, b \in \mathbb{G}: a + b = b + a$
  - **Etkisiz Eleman Özelliği** : $\exists \, 0 \in \mathbb{G}, \forall \, a \in \mathbb{G}: a + 0 = a$
  - **Ters Eleman Özelliği**: $\forall \, a \in \mathbb{G}, \exists \, b \in \mathbb{G}: a + b = 0$
    - **Örnek:** Tamsayılar kümesi ve toplama işlemi olan $< Z, + >$, bir Abel grubu oluşturur. Aynı şekilde 0'dan $n - 1$ ' e kadar olan tamsayıların oluşturduğu küme ve 'mod n toplama' işlemide, $< \mathbb{Z}_n, + >$, de bir Abel grubu oluşturur.

- **Tanım 5 - Halka:** $< \mathbb{R}, +, * >$ bir $\mathbb{R}$ kümesinden ve bu kümeye ait elemanlar üzerinde tanımlanmış olan iki adet işlemden (' + ' ve ' $*$ ') oluşmaktadır.

  $\mathbb{R}$ kümesiyle bir ' $halka$ ' oluşturulabilmesi için ' + ' ve ' $*$ ' işlemlerinin sağlaması gereken koşullar vardır;

- $< \mathbb{R}, + >$ bir Abel grubu oluşturabilmelidir
- ' $*$ ' işlemi, $\mathbb{R}$ üzerinde, kapalılık ve birleşme özelliklerine sahip olmalı ve ' $*$ ' işleminde de bir etkisiz eleman (genellikle '1' ile gösterilir) tanımlanabilmelidir
- ' $*$ ' işleminin, ' $+$ ' işlemi üzerinde dağılma özelliği olmalıdır

$$\forall\, a, b, c \in \mathbb{R} : (a + b) * c = (a * c) + (b * c) \quad \textbf{(1)}$$

Burada ' $*$ ' işleminin değişme özelliği olduğunda, $< R, +, * >$ halkası 'Değişmeli Halka' olarak adlandırılır.

- ❖ **Örnek:** Tamsayılar kümesi, 'toplama' ve 'çarpma' işlemleriyle birlikte, $< Z, +, * >$, bir halka oluşturmaktadır.

- **Tanım 6 - Cisim:** Bir sayı kümesi, '$\mathbb{F}$', üzerinde tanımlanmış ' $+$ ' ve ' $*$ ' işlemleri ile bir arada aşağıda belirtilen şartları sağladığında bir 'cisim' oluşturur
  1. $< F, + >$ ve $.< F, * >$ birer Abel grubu olmalıdırlar,
  2. Yalnızca toplama işlemi etkisiz elemanı ('0') için bir ters eleman olması zorunlu değildir,
  3. $< F, +, * >$ bir halka olmalıdır, bu yukarıda yer alan şartlar haricinde ' $*$ ' işleminin ' $+$ ' üzerinde dağılma özelliğinin sağlanması ile sağlanır.

  ❖ **Örnek:** Gerçel sayılar kümesi, 'toplama' ve 'çarpma' işlemleriyle bir aradayken bir cisim oluşturur.

- **Tanım 7 – Sonlu Cisim:** Sonlu cisim, sonlu sayıda elemana sahip olan 'cisim'dir. Bu cismin sahip olduğu eleman sayısı, sonlu cismin düzenini (order) belirler. Eş sayıda elemanı olan sonlu cisimle eş yapılıdır, tamamen aynı matematiksel yapıyı işaret ederler, yalnızca elemanlarının gösterilişinde farklılıklar bulunur.

Karakteristiği 2 olan $GF(2^n)$ sonlu alanı kriptografide en sık başvurulan alanı temsil eder. Bit (*Binary Digit*) değerleri $GF(2^n)$ sonlu alanında değerler olarak ele alınırlar. Bit değerlerinin $GF(2^n)$ sonlu alan bağlılığı bit değerlerine bağlı 8-bitlik byte değerlerini ve diğer değişken tiplerini $GF(2^n)$ sonlu alanına bağlar. Buradan hareketle 8-bitlik bir byte değeri $GF(2^8)$ sonlu alanında bir değer olarak temsil edilir.

$GF(2^n)$ sonlu alanı n adet eleman içerir. Bu elemanlar yan yana yazılmış bit (*0, 1*) değerleridir. Bu alanın polinomsal gösterimi, $GF(2^n)$ için $\{x^{n-1}, x^{n-2}, x^{n-\cdots}, x^2, x, 1\}$ kümesinden meydana gelir.

- ❖ **Örnek 1:** $GF(2^8)$ sonlu alanındaki $a_{10} = 34$ onluk sayı değerinin, ikilik sayı sistemindeki değerleri $a_2 = \{00100010\}$ olur. a değerinin polinomsal temsili $a(x) = x^5 + x$ olur.
- ❖ **Örnek 2:** $GF(2^8)$ sonlu alanındaki $a_{10} = 26$ onluk sayı değerinin, ikilik sayı sistemindeki değerleri $a_2 = \{00011010\}$ olur. a değerinin polinomsal temsili $a(x) = x^4 + x^3 + x$ olur.

Bit ve byte değerleri $GF(2^n)$ sonlu alanı değerleri olarak ele alındığından dolayı bu alan üzerinde bit düzeyinde işlem yapılması zaman ve güç maliyetini düşürür.

$GF(2^n)$ alanının özellikleri kullanılarak bit değerlerinin taşınması, mantıksal operatörlerde işletilmesi veya tümleyenin alınması kolayca gerçekleşebilir.

Çalışma devamında $GF(2^n)$ alanındaki işlemler ele alınmıştır.

2.4.6.1. Mantıksal operatörler – AND, OR, XOR, NOT

$GF(2^n)$ alanındaki bitler ile işlem yapmak bir sayı değerinin bütünü üzerinde değil, doğrudan bit değerlerini sınamak değiştirmek öteleme yapmak anlamındadır. Bu alan üzerinde işlem yapan 6 operatör vardır.

- **AND (∧) Operatörü:** $GF(2^n)$ sonlu alanındaki en az iki değerin farklı üssel değere sahip polinom elemanlarından oluşan kesişim elemanlarının toplamıdır.

$$(a \wedge b) = (a \cap b) \quad (7)$$

❖ **ÖRNEK:** $(a_{10} = 34) = \{x^5 + x\}$ ve $(b_{10} = 26) = \{x^4 + x^3 + x\}$ için $a \wedge b = 2 = \{x\}$ olur.

	x^7	x^6	x^5	x^4	x^3	x^2	x^1	x^0	
	128	64	32	16	8	4	2	1	
34 =	0	0	1	0	0	0	1	0	
26 =	0	0	0	1	1	0	1	0	
34 ^ 26 =	0	0	0	0	0	0	1	0	= 2

Şekil 5: AND Operatörü Örneği

- **OR (V) Operatörü:** $GF(2^n)$ sonlu alanındaki en az iki değerin, sahip oldukları polinom elemanlarından oluşan birleşim elemanlarının toplamıdır.

$$(a \vee b) = (a \cup b) \quad (8)$$

❖ **ÖRNEK:** $(a_{10} = 34) = \{x^5 + x\}$ ve $(b_{10} = 26) = \{x^4 + x^3 + x\}$ için $a \vee b = 58 = \{x^5 + x^4 + x^3 + x\}$ olur.

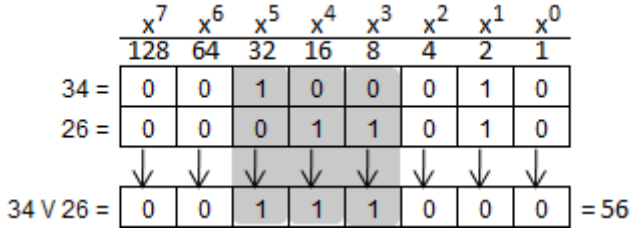
	x^7	x^6	x^5	x^4	x^3	x^2	x^1	x^0	
	128	64	32	16	8	4	2	1	
34 =	0	0	1	0	0	0	1	0	
26 =	0	0	0	1	1	0	1	0	
34 V 26 =	0	0	1	1	1	0	1	0	= 58

Şekil 6: OR Operatör Örneği

- **XOR ($\oplus$) Operatörü:** $GF(2^n)$ sonlu alanındaki en az iki değerin, farklı üssel değere sahip polinom elemanlarının toplamıdır.

$$(a \oplus b) = \big((a \cup b) - (a \cap b)\big) \quad (9)$$

❖ **ÖRNEK:** $(a_{10} = 34) = \{x^5 + x\}$ ve $(b_{10} = 26) = \{x^4 + x^3 + x\}$ için $a \vee b = 56 = \{x^5 + x^4 + x^3\}$ olur.



Şekil 7: XOR Operatör Örneği

- **LShift (≪) Operatörü:** $GF(2^n)$ sonlu alanındaki bir değerin polinom elemanlarının üssel değerlerinin y sayısınca arttırılmasıdır. Ancak burada x^{i+y} değerinin 2^{n-1} değerini aşması durumunda ilgili bit $GF(2^n)$ sonlu alanını aşmış olacaktır. Bu durumda alanı aşan bit değeri sıfırlanacaktır (ezilecektir).

$$(a \ll y) = \left(\{x^{(n-1)+y}, x^{(n-\cdots)+y}, x^{0+y}\} \wedge \{z^{n-1} + z^{n-2} + z^{n-\cdots} + z^{0}\}\right) \quad (10)$$

❖ **ÖRNEK:** $(a_{10} = 34) = \{x^5 + x\} = (a_2 = 00100010)$ ve $y = 4$ için $GF(2^8)$ sonlu alanında $a \ll y = 32 = \{x^5\}$ olur.

$$34 = \{x^5 + x^1\} = 00100010$$
$$\ll 4 = \{x^{5+4} + x^{1+4}\} = 001000100000$$
$$= \{x^9 + x^5\} = 001000100000$$
$$GF(2^8) = \{\cancel{x^9} + x^5\} = \cancel{0010}00100000$$
$$= \{x^5\} = 00100000$$

- **RShift (≫) Operatörü:** $GF(2^n)$ sonlu alanındaki bir değerin polinom elemanlarının üssel değerlerinin y sayısınca eksiltilmesidir. Ancak burada

$x^{i-y} < 0$ olması durumunda ilgili bit $GF(2^n)$ sonlu alanını aşmış olacaktır. Bu durumda alanı aşan bit değeri sıfırlanacaktır (ezilecektir).

$$(a \gg y) = \left(\{x^{(n-1)-y}, x^{(n-\cdots)-y}, x^{0-y}\} \wedge \{z^{n-1} + z^{n-2} + z^{n-\cdots} + z^{0}\}\right) \quad (11)$$

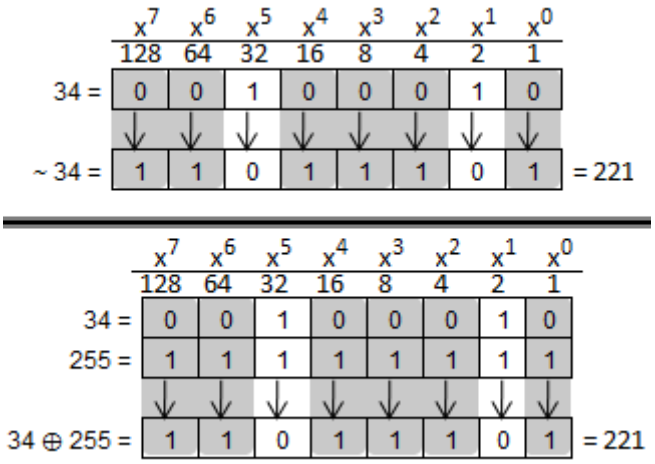
- **ÖRNEK:** $(a_{10} = 34) = \{x^5 + x\} = (a_2 = 00100010)$ ve $y = 3$ için $GF(2^8)$ sonlu alanında $a \gg y = 3 = \{x^2\}$ olur.

$$34 = \{x^5 + x^1\} = 00100010$$
$$\gg 3 = \{x^{5-3} + x^{1-3}\} = 00000100010$$
$$= \{x^2 + x^{-2}\} = 00000100010$$
$$GF(2^8) = \{x^2 + \cancel{x^{-2}}\} = 00000100\cancel{010}$$
$$= \{x^2\} = 00000100$$

- **One's Complement (~) Operatörü:** $GF(2^n)$ sonlu alanındaki bir değerin polinom elemanlarının bire tümlenmesidir. Burada $GF(2^n)$ için x 'in tümleyeni $(2^n - 1) - x$ 'dir. x değerinin $GF(2^n)$ sonlu alanındaki ikilik sayı sistemindeki 0'ların 1, 1'lerin yapılması bu operatörün basit tanımıdır.

- **ÖRNEK:** $(a_{10} = 34) = \{x^5 + x\} = (a_2 = 00100010)$ için $GF(2^8)$ sonlu alanında bir $a\sim = 221 = \{x^7 + x^6 + x^4 + x^3 + x^2 + x^0\}$ olur.

$$34 = \{x^5 + x\} = 00100010$$
$$\sim 34 = \left((2^8 - 1) - 34\right) = (255 - 34) = 221 \Rightarrow$$
$$= 11011101$$



Şekil 8: ~ ve XOR Operatör Örneği

2.4.6.2. Cebir – Toplama, çıkarma, çarpma ve bölme

Toplama; İki polinomun toplanması aynı üssel değere sahip x değerlerinin katsayılarının toplanmasıyla gerçekleşir. Ancak burada sonuç polinomun bazı katsayıları 0 veya 1'den farklı değerler alırsa sonuç $GF(2^n)$ alanını aşmış olur, bu sebeple $GF(2^n)$ alanı ikilik sayı sisteminde olduğundan dolayı $GF(2^n)$ alanındaki yx^z için $yx^z = x^{z+y}$ eşitliği elde edildikten sonra $GF(2^n)$ alanını aşan değerler göz ardı edilir.

XOR ($\oplus$) operatörü en sağ /değersiz bit değerinden en sol/değerli bit değerine sıralaması ile yapılır. Burada alt alta iki bit değeri 1 ise alta sıfır gelir ancak sonraki değer 1 değerini alır, sonraki değerin 1 olması durumunda o değer 0 olur ve sonraki değer 1 olur, işlem her bit değerinin hesaplanmasında göz önüne alınır.

Yukarıda yer alan işlem dışığında $(a + b)\ mod\ 2^n$ de $GF(2^n)$ sonlu alanı içerisinde sonuç değeri verir.

	x^7	x^6	x^5	x^4	x^3	x^2	x^1	x^0	
	128	64	32	16	8	4	2	1	
34 =	0	0	1	0	0	0	1	0	$= \{x^5 + x^1\}$
26 =	0	0	0	1	1	0	1	0	$= \{x^4 + x^3 + x^1\}$

Şekil 9: GF(2) Sonlu Alanında Toplama

$$p(34) = \{x^5 + x^1\}$$
$$p(26) = \{x^4 + x^3 + x^1\}$$
$$p(34) + p(26) = \{x^5 + x^4 + x^3 + 2x^1\} = \{x^5 + x^4 + x^3 + x^2\}$$

($2x^1 \rightarrow x^2$)

$$0_7 0_6 1_5 1_4 1_3 1_2 0_1 0_0$$

Çarpma; İki polinomun çarpımı aynı üssel değere sahip x değerlerinin katsayılarının toplanmasıyla gerçekleşir. Ancak burada işlemin klasik polinom çarpımı şeklinde yapılması katsayıların 0 veya 1'den farklı değerler almasıyla sonuçlanır ve bu değerler $GF(2^n)$ sonlu alanını aşarlar. Aşıma uğrayan değerler göz ardı edilirler.

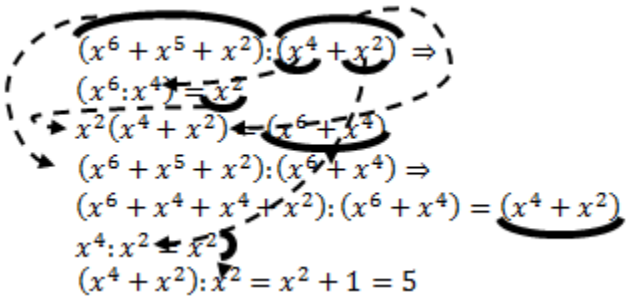
❖ **ÖRNEK:** $a = 34 = \{34 + 5\}$ ve $b = 26 = \{x^4 + x^3 + x\}$ klasik polinom çarpımı ve $GF(2^8)$ sonlu alanına zorlanması gösterilmiştir.

$$(x^5 + x).(x^4 + x^3 + x)$$
$$x^9 + x^8 + x^6$$
$$+$$
$$x^5 + x^4 + x^2$$
$$x^9 + x^8 + x^6 + x^5 + x^4 + x^2 = a.b$$
$$\cancel{x^9 + x^8 +} x^6 + x^5 + x^4 + x^2 = GF(2^8)[a.b]$$
$$x^6 + x^5 + x^4 + x^2 = 116$$

Bölme; Burada bölünecek polinomun en büyük üstel değeri, bölenin en büyük üstel değeriyle bölünür ve sonuç elde edilir. Devamında elde edilen bu sonuç bölenin tüm değerleriyle çarpılır ve bölünen polinomdan çıkarılarak yeni bir bölünen polinomu elde edilir. Aynı işlemler bölünen polinomun en büyük üstel değeri, bölenin en büyük üstel değerinden küçük değere ulaşıncaya kadar tekrarlanır. Tekrarlar sonlandığında sonda kalan bölünen, işlem sonucu kalan polinomu verir.

❖ **ÖRNEK:** Aşağıda bölünen $a = 100 = \{x^6 + x^5 + x^2\}$ ve bölen $b = 20 = \{x^4 + x^2\}$ için polinom bölümü gösterilmiştir.

$$(x^6 + x^5 + x^2) : (x^4 + x^2) \Rightarrow$$
$$(x^6 : x^4) = x^2$$
$$x^2(x^4 + x^2) = (x^6 + x^4)$$
$$(x^6 + x^5 + x^2) : (x^6 + x^4) \Rightarrow$$
$$(x^6 + x^4 + x^4 + x^2) : (x^6 + x^4) = (x^4 + x^2)$$
$$x^4 : x^2 = x^2$$
$$(x^4 + x^2) : x^2 = x^2 + 1 = 5$$



Şekil 10: GF – Bölme

3. KRİPTOGRAFİ VE KRİPTOSİSTEMLER

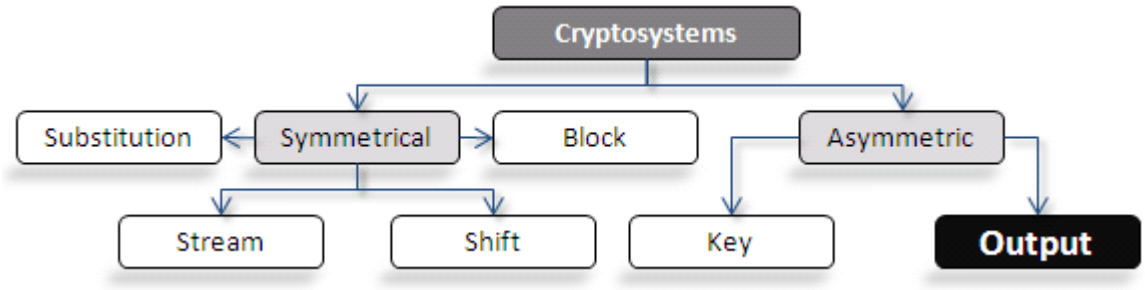
3.1. Giriş

Kriptografi her ne kadar şifrelemeyi çağrıştıran bir anlam ifade etse de, bu çok doğru bir tanımlama olmaz. Şifreleme (encryption) kriptografi için bir alt alana işarete eder. Şifreleme işlemi 3 ayrı veriyi içerir, birincisi şifrelenmemiş / açık veriyi temsil eden $\mathcal{P}$ (Plaintext) veya $\mathcal{M}$ (Message), ikincisi şifre / anahtar verisini temsil eden $\mathcal{K}$ (Cipher / Key) ve son olarak anahtarla şifrelenmiş veriyi temsil eden $\mathcal{C}$ (ciphertext). Burada anahtarın bilinmesi durumunda şifrelenmiş veriden her zaman Plaintext/Message elde edilebilmelidir. Bilinen ve bu çalışmaya kadar olan mevcut literatür ciphertextin her zaman sabit olması üzerine kuruludur[23].

Ancak mesaj özetleme (hash) fonksiyonları veya e-imza (digital signature) fonksiyonları da kriptografik yapıdadır ve *teoride* bu fonksiyonların ürettiği değerlerden giriş verisine dönmek mümkün değildir.

Hash fonksiyonlarıyla, sayısal imza algoritmaları çoğu kez birbirleriyle karıştırılıyor olsa da aslında aynı temel yapıları kullanırken farklı sonuçlar üretirler. Sayısal imza algoritmaları değer (imza) üretebilmek için hash algoritmalarını kullanırlar. Sayısal imza algoritmaları asimetrik kriptosistemlerle de karıştırılabilmektedir. Ancak sayısal imza algoritmalarında sonuç verisinden (mesaj özeti imza), giriş verisine (imza, mesaj) dönmek *teoride* mümkün değildir.

23 Bu çalışma içerisinde yer alan Cube bilinen yöntemler dışında, Ciphertextin asimetrik olduğu bir yapıyı sunmaktadır.



Şekil 11: Kriptosistemler

Mevcut literatürde asimetrik kriptosistemler, anahtarın asimetrisi üzerine kurulmuştur. Burada açık (public) / gizli (private) anahtarlar olup, şifreleme anahtarı ile şifre çözme anahtarının farklı olması söz konusudur.

Asimetri terimi gereği, asimetrik kriptosistem sadece anahtarı değil, şifreleme işlemine ait anahtarı (cipher), şifreleme (encryption) algoritmasını ve şifrelenmiş veriyi (ciphertext) kapsamına alabilecek yapıdadır. Ancak bu çalışmanın hazırlandığı tarihte anahtar dışığında diğer sınıflar üyelerini bulamamışlardır.

Bu çalışma devamında açıklanan Cube anahtar değil, şifrelenmiş verinin (ciphertext) asimetrisini sağlayan ve yeni bir alt alana işaret eden yeni nesil bir kriptosistemdir. Ayrıca asimetrik değer üretimine bağlı asimetrik alt fonksiyon çağırımı[24] da literatür için bir yeni alan olması yanında, çalışma odağının şifrelenmiş verinin asimetrisi olması tercih edilen durum olmuştur.

3.2. Simetrik Kriptosistemler

Simetrik kriptosistemler, düz metnin (plaintext), sabit anahtar (cipher) ile şifrelenmesi (encryption) sonucu her zaman aynı şifrelenmiş metnin (ciphertext) elde

24 Burada asimetrik fonksiyon çağırımı, fonksiyon değer ve değişkenleri için değil, çağırılacak fonksiyonun seçimini işaret eder.

edilmesini işaret eder. Doğru ve tek olan anahtarın (cipher) bilinmesi durumunda sabit olan şifreli metinden (ciphertext), düz metne (plaintext) ulaşılır.

Bugün için simetrik kriptosistem literatürü akan (stream), kaydırma (shift), değiştirme (substitution), permütasyon ve yapıları ile evrimine devam etmektedir.

3.2.1. Akan (Stream) Kriptosistem

Akan (stream) kriptosistem bir $\mathcal{K} = \{\mathcal{K}_1, \mathcal{K}_{...}, \mathcal{K}_n\}$ anahtar dizisi oluşturmak ve $\mathcal{P} = \{\mathcal{P}_1, \mathcal{P}_{...}, \mathcal{P}_n\}$ düz metin dizisini $C = \{\mathcal{C}_1, \mathcal{C}_{...}, \mathcal{C}_n\} = \left(B = \left\{\left(B_{\mathcal{K}_1}(\mathcal{P}_1)\right), \left(B_{\mathcal{K}_{...}}(\mathcal{P}_{...})\right), \left(B_{\mathcal{K}_n}(\mathcal{P}_n)\right)\right\}\right)$ denkliğinde kodlamak olarak gösterile bilir.

Burada anahtar $\mathcal{K} \in \mathbb{K}$, $a_1, a_{...}, a_n$ düz metin $k, \mathcal{K}_{...}, \mathcal{K}_n$ bir anahtar akımı değeri, $f_i(\,)$ ise anahtar akımının i. elemanını oluşturan fonksiyon olsun; bu durumda ilk $i-1$ düz veri $\mathcal{C}_i = f_i(\mathcal{K}, a_i, a_{i-1})$ olur.

c_i anahtar akımı değeri, a_i değerini şifreler. Bu durumda $a = a_1, a_{...}, a_n$ düz metninin şifrelenmesi için $c_1, b_1, c_{...}, b_{...}, c_n, a_n$ hesaplanır.

Akan (stream) kriptosistem için aşağıdaki $\mathcal{P}$, $\mathcal{C}$, $\mathcal{K}$, S, F temsilleri sağlanmalıdır.

- $\mathcal{P}$ (Plaintext): Düz/Açık metni temsil eden sonlu küme.
- $\mathcal{C}$ (Ciphertext): Şifreli metni temsil eden sonlu küme.
- $\mathcal{K}$ (Cipher): Anahtarı temsil eden sonlu küme.
- S (Stream – Keystream): Anahtar akım alfabesini temsil eden sonlu küme.
- F: $(f_1, f_{...}, f_n)$ anahtar üretim fonksiyonu olup burada $i \in \mathbb{Z}^+, i > 0$ için $f_i\left(\mathcal{K}.\mathcal{P}^{i-1}\right) = L$ olur.

Ayrıca $\forall c \in S \Rightarrow e_c \in E$ olmak üzere bir şifreleme fonksiyonu ve $d_c \in D$ olmak üzere bir şifre çözüm fonksiyonu sağlanır. Burada $ec(\mathcal{P}) = \mathcal{C}$ ve dc($\mathcal{C}$)=$\mathcal{P}$ fonksiyonları $\forall a \in \mathcal{P}: d_c(e_c(a) = a)$ eşitliğini sağlar.

Çalışma devamında ele alınan blok kriptosistemde Akan (stream) kriptosistemin özelleştirilmiş bir halidir denilebilir, bu $i > 0, \forall i \in \mathbb{Z}^+: c_i = \mathcal{K}$ ile sağlanır.

Bir akan şifre anahtar akımı dizisinden bağımsızsa eş zamanlı olarak tanımlanır. Burada $i > 0_i \forall i \in \mathbb{Z}^+$ için $c_{i+d} = c_i$ periyodu ile d periyodik durumdadır.

Akan (stream) kriptosistemler genelde ikili alfabelerle temsil edilir. Burada $\mathcal{P} = \mathcal{C} = S = \mathcal{C}_2$ olsun, bu durumda şifreleme;

$$\mathrm{e}_2(\mathrm{a}) = (\mathrm{a} + \mathrm{c}) \bmod 2 \qquad (12)$$

ve şifre çözme;

$$\mathrm{d}_2(\mathrm{b}) = (\mathrm{b} + \mathrm{c}) \bmod 2 \qquad (13)$$

olarak gösterilebilir.

$g = 3$ ve anahtar akımı $c_i + 3 = c_i + c_{i+1}\, mod\ 2, (i > 0)$ eşitliğinde oluşturulmuş bir şifreleme ele alınsın. Burada anahtar akım $(0,0,0)$'dan farklı olursa, bu durumda elde edilen anahtar akımının periyodu 8 olur. Burada $(1,0,0)$ başlangıçlı bir anahtar akımı $(1,0,0,1,0,1,1,1, \ldots)$ olur.

Bir anahtar kaydırma g aşamada kullanılır. Bu kaydırma için $(k_1, k_{\ldots}, k_g)$ dizisi kullanılır. Her işlemde;

- k_1 sonraki anahtar akım bitine işaret eder.
- $k_1, k_{\dots}, k_g$ sola doğru kaydırılır.
- k_g'nin yeni değeri $\sum_{i=0}^{g-1}(c_i k_i + 1)$ ile bulunur.

❖ **Örnek:** Anahtar (cipher) $k = 26$ ve düz metin (plaintext) $USKUDAR$ olsun.

1. Düz metin Şekil 4 yararlanılarak ASCII formatında tam sayılara dönüştürülür.

$$(\mathcal{P}_{CHAR} = \{U, S, K, U, D, A, R\})$$
$$= (\mathcal{P}_{ASCII} = \{85,83,75,85,68,65,82\})$$

1. Anahtar akımı için ilk eleman yerine anahtar konulur, devamında düz metin boyutu ile eşleştirme için düz metnin elemanları sırayla anahtar akımına eklenir.

2. $\mathcal{P}_n$ ve $\mathcal{K}_n$ elemanları toplanarak $mod\ 2^8$ (*ASCII tablosundaki 256 değer sebebiyle...*) değerleri alınmıştır.

$$(\mathcal{C}_{ASCII} = \{111,168,158,168,153,133,147\}) = \mathcal{C}_{CHAR}$$
$$= \{o, ¿, Ş, á, Ö, à, ô\}$$

3. Elde edilen şifreli metin (ciphertext) şifresinin çözülmesi için önce $a_1, a_{\dots}, a_n$ hesaplanır

$$a_1 = d_{256}(111) = (111 - 26)\ mod\ 256 = \mathbf{85}$$
$$a_2 = d_{256}(168) = (168 - \mathbf{85})\ mod\ 256 = \mathbf{83}$$
$$a_3 = d_{256}(158) = (158 - \mathbf{83})\ mod\ 256 = \mathbf{75}$$
$$a_4 = d_{256}(160) = (160 - \mathbf{75})\ mod\ 256 = \mathbf{85}$$
$$a_5 = d_{256}(153) = (153 - \mathbf{85})\ mod\ 256 = \mathbf{68}$$
$$a_6 = d_{256}(133) = (133 - \mathbf{68})\ mod\ 256 = \mathbf{65}$$
$$a_7 = d_{256}(147) = (147 - \mathbf{65})\ mod\ 256 = \mathbf{82}$$

Burada sırasıyla çözülen her bir şifreli metin (*ASCII formatındaki sayısal karşılığı*) karakteri bir sonraki şifreli metin karakterinin şifre çözümünde kullanılır.

- $\mathcal{P} = \mathcal{C} = \mathcal{K} = S = \mathbb{Z}_{256}^{+}$, $c_1 = \mathbb{K}$ ve $c_i = a_{i-1}[(i > 1), i \in \mathbb{Z}^{+}]$ olsun. Bu durumda $0 \leq c \leq 255$ ve $(a, b, c \in \mathbb{Z}_{256}$) için şifreleme;

$$e_a(a) = a + c\ mod\ 256 \quad (14)$$

ve şifre çözme;

$$d_a(b) = b - c\ mod\ 256 \quad (15)$$

tanımlanır.

3.2.2. Değiştirme (Substitution) Kriptosistem

Değiştirme (Substitution) kriptosistem $\mathcal{P} = \mathcal{C} = \mathcal{K} = S = \mathbb{Z}_{26}^{+}$ ve $\forall f(\mathcal{P}) \in \mathcal{K}$ için $f(\mathcal{P})^{-1}$ için $f(\mathcal{P})$' nin tersi olmak üzere şifreleme $\mathcal{C}_{\mathcal{K}}(\mathcal{P}) = f(\mathcal{P})$ ve şifre çözme $\mathcal{P}_f(\mathcal{P}) = f^{-1}(\mathcal{C})$ olarak tanımlanabilir.

Diğer bir tanımlama ile $\mathcal{P} = \mathcal{C} = \mathcal{K} = S = \mathbb{Z}_{256}^{+}$ olsun, her $a \in \mathbb{Z}_{26}^{+}$ permütasyonu için, a^{-1} , $a(\,)$ 'nın ters permütasyonu olmak üzere $e_a(b) = a(b)$ $d_a(b) = a^{-1}(c)$ olur.

Burada düz metin (plaintext) küçük ve şifreli metin (ciphertext) karakterleri büyük harflerle yazılarak aşağıdaki Tablo VIII. Değiştirme tablosuile verilen a fonksiyonu verilmiştir.

Tablo VIII. Değiştirme tablosu

a	**b**	**c**	**d**	**e**	**f**	**g**	**h**	**i**	**j**	**k**	**l**	**m**
B	C	R	F	K	G	E	O	T	M	P	V	W

n	**o**	**p**	**q**	**r**	**s**	**t**	**u**	**v**	**w**	**x**	**y**	**z**
Q	U	Z	A	S	N	I	H	L	D	Y	X	J

Tablo VIII ile verilen $a(\,)$ şifreleme fonksiyonunun tersi bu tablonun tersidir. a fonksiyonunun tersi aşağıda verilen Tablo IX**.** Değiştirme tablosu tersi ile verilen şeklindedir.

Tablo IX. Değiştirme tablosu tersi

A	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**J**	**K**	**L**	**M**
q	a	b	w	h	d	f	u	t	z	e	v	J

N	**O**	**P**	**Q**	**R**	**S**	**T**	**U**	**V**	**W**	**X**	**Y**	**Z**

❖ **Örnek:** $a()$ şifreleme fonksiyonu ile " $uskudaruniversitesi$ " plaintextin şifrelenmiş hali;

"$HNPHFBSHQLKSNTIENT$"

olur ve bu şifreli metnin (ciphertext) tablo 8'e göre

$$d_{a-1}(\boldsymbol{H}) = \boldsymbol{u}, d_{a-1}(\boldsymbol{N}) = \boldsymbol{s}, d_{a-1}(\boldsymbol{P}) = \boldsymbol{k}, d_{a-1}(\boldsymbol{H}) = u, d_{a-1}(\boldsymbol{F}) = d, d_{a-1}(\boldsymbol{B}) = \boldsymbol{a}, d_{a-1}(\boldsymbol{S}) = \boldsymbol{r}, d_{a-1}(\boldsymbol{H}) = \boldsymbol{u}, d_{a-1}(\boldsymbol{Q}) = n, d_{a-1}(\boldsymbol{T}) = i, d_{a-1}(\dots) = \cdots$$

şeklinde deşifre edilmiş hali;

"$uskudaruniversitesi$"

olarak elde edilir.

Yaygın literatür bu yöntemi 26 alfabetik karakter kümesiyle kullanır ve yukarıda yer alan örnek bu şekilde oluşan bir anahtar ve bunun permütasyonuyla kullanılmıştır. Bu duruma göre 26 harflik olasılık durumu ortaya çıkar. Bu da ilk karakter için $\frac{1}{26}$, ikincisi $\frac{1}{25}$ olarak devam eden ve $\frac{1}{26}, \frac{1}{25}, \frac{1}{\dots}, \frac{1}{26-n}$ olasılık evrenini ortaya ortaya çıkarır, bu permütasyonların toplamı 26! olarak gösterilebilir. Bu deneme yanılma saldırısının bir

bilgisayarla yapılmasını zorlaştırır. Ancak kriptanaliz yöntemleri kullanıldığında bu metot ile yapılan şifrelemeler çok kolay çözülebilmektedir. Buradan da anlaşılacağı gibi kriptanaliz ile deneme-yanılma saldırıları farklı yöntemlerdir.

3.2.3. Affine Kriptosistem

Affine kriptosistem $a, b \in \mathbb{Z}_{26}^{+}$ ve $e(x) = (ax + b)\ mod\ 26$ olarak tanımlanabilir. Yapısı akan (stream) kriptosisteme benzese de aynı değildir. Modüler aritmetik asimetrik ve simetrik kriptosistemlerde anahtarların oluşturulmasında

Afffine ile şifrelenmiş bir şifreli metnin (ciphertext) çözülebilmesi için bir affine fonksiyonun hangi durumda bire bir olacağını belirlemek gerekir, bu;

$$y \in \mathbb{Z}_{26}^{+}: ax + b \equiv y\ (mod\ 26) \quad (16)$$

denkliğinde x için tek bir çözümün olmasıyla mümkün olur bu da;

$$ax \equiv y - b\ (mod\ 26) \qquad (17)$$

ile eş olacaktır. Bu durumda $y \in \mathbb{Z}_{26}^{+}$ olarak farklı değerler aldıkça $y - b$'de $(y - b) \in \mathbb{Z}_{26}^{+}$ içerisinde farklı değerler alır. Bu her y için sadece ve sadece $OBEB(a, 26) = 1$ olursa tek bir çözüme sahip olur.

Önce $OBEB(a, 26) = d > 1$ olduğu varsayılsın, bu durumda $ax \equiv 0\ (mod\ 26)$ denkliği $\mathbb{Z}_{26}^{+}$ için $x = 0$ ve $x = 26/d$ olarak en az iki farklı çözüme sahip olur. Buna göre $e(x) = ax + b\ (mod\ 26)$ bire bir fonksiyon olmaz ve bu bir şifreleme fonksiyonu olamaz.

- **Örnek:** $OBEB(14,26) = 2$ olsun, bu durumda herhangi bir $x \in \mathbb{Z}_{26}^{+}$ için x ve $x + 13$ aynı değeri vereceğinden $14x + 5$ doğru bir şifreleme fonksiyonu olamaz.

- **Örnek:** $OBEB(a, 26) = 1$ olsun ve herhangi x_1 ve x_2 için $ax_1 \equiv ax_2 \ (mod\ 26)$ olsun, bu durumda $a(x_1 - x_2) \equiv 0 \ (mod\ 26)$ olur ve $26/a(x_1 - x_2)$olur. Burada şu durum ortaya çıkar;

$$OBEB(a, b) = 1,\ a/bc \Rightarrow a/c \qquad (18)$$

buradan hareketle örneğin;

$$\begin{aligned} &26/a(x_1 - x_2),\ \gcd(a, 26) = 1 \ \Rightarrow \\ &\quad 26 \mid (x_1 - x_2) \ \Rightarrow \\ &\quad x_1 \equiv x_2 \ (mod\ 26) \end{aligned} \qquad (19)$$

elde edilecektir.

Eğer $OBEB(a, 26) = 1$ ise $ax \equiv y \ (mod\ 26)$ denkliği $\mathbb{Z}_{26}^{+}$ için en çok bir çözüme sahip olur. Burada x'in $\mathbb{Z}_{26}^{+}$ içinde değişimi gerçekleşirse $ax \ mod\ 26$, 26 farklı değer alabilir ve bu her değeri yalnızca bir defa alması anlamına gelir. Bu herhangi $y \in \mathbb{Z}_{26}^{+}$ için $ax \equiv y \ (mod\ 26)$ denkliğinde y'nin tek bir çözümü olduğunu gösterir.

- $ax \equiv b\ (mod\ m)$ denkliği $\forall b \in \mathbb{Z}_m^+$ için, yalnızca $OBEB(a,m) = 1$ olduğunda $26 = 2.13$ olduğu için $OBEB(a,26) = 1$ için $\forall \in \mathbb{Z}_{26}^+$ değerleri $a = \{1,3,5,7,9,11,13,17,19,21,23,25\}$ olur. b ise $\mathbb{Z}_{26}^+$ 'da herhangi bir değerdir. Burada affine kriptosistemin $12.26 = 312$ anahtarı olur.

- $a \geq 1, m \geq 2; a, m \in \mathbb{Z}^+$ olsun. $\gcd(a,m) = 1$ ise a ve m aralarında asal olur. m ile aralarında asal olan $\mathbb{Z}_m^+$ 'deki değerlerin sayısı $\phi(m)$ olur. ϕ simgesi ile temsil edilen fonksiyona $Euler(phi)$ fonksiyonu denir. Bu fonksiyon çalışma devamında ele alınan asimetrik kriptosistem algoritmalarında da kullanılır.

- p_i bir asal sayılar dizisi ve dizi değerlerinin her biri farklı değerlerde ise $(e_i > 0), (1 \leq i \leq n)$ olduğunda;

$$A = \prod_{i=1}^{n} \left(p_i^{e_i}\right) \quad (20)$$

olur, bu durumda da;

$$\phi(m) = \prod_{i=1}^{n} \left(p_i^{e_i} - p_i^{e_i - 1}\right) \quad (21)$$

olur.

$\mathbb{Z}_m^+$'de affine kriptosistemdeki anahtarların sayısı yukarıda verilen $\phi(m)$ ile temsil edilir. Ancak şifreleme fonksiyonu $e(x) = ax + b$ olması durumunda b'nin sahip olabileceği değerlerin sayısı m olur ve a için $\phi(m)$ olur.

- **Örnek:** $m = 40 = 2^2.2.5$ olursa;

$$\phi(40) = \phi(2^2.2.5) = (2^2 - 2).(2 - 1).(5 - 1) = 2.1.4 = 8$$

olur ve bu durumda affine kriptosistemdeki anahtarları sayısı $40.8 = 320$ olarak bulunur.

Affine kriptosistemde $m = 26$ olarak şifre çözme işlemi için $OBEB(a, 26) = 1$ olsun. Bu durumda x için $y \equiv ax + b \ (mod\ 26)$ denkliği çözülmelidir, bu denklik $\mathbb{Z}_{26}^+$ için tek bir çözüme sahiptir. Tek bir çözümün olması şifrenin çözülmesi için yeterli değildir. Bu çözümün yapılabilmesi için modüler aritmetiğe ait bazı algoritmalar kullanılır.

- $a \in \mathbb{Z}_m^+$ olduğunda $a.a^{-1} \equiv a^{-1}.a \equiv 1 \ (mod\ m)$ olan bir $a^{-1} \in \mathbb{Z}_m^+$ olur
- $\mathbb{Z}_{26}^+$ içinde değerlerin çarpmaya göre tersleri sırasıyla $1^{-1} \equiv 1$, $3^{-1} \equiv 9$, $5^{-1} \equiv 21$, $7^{-1} \equiv 15$, $11^{-1} \equiv 19$, $17^{-1} \equiv 23$ ve $25^{-1} \equiv 25$ olur. Burada bir örnek olarak $3.9 = 27 \equiv 1 \ (mod\ 26)$ olur, bu durumda $3^{-1} = 9$ olur.

- $\mathcal{P} = \mathcal{C} = \mathbb{Z}_{26}^{+}$ ve $\mathbb{K}\{(a,b) \in \mathbb{Z}_{26}^{+} \times \mathbb{Z}_{26}^{+};\ \gcd(a,26) = 13\}$ ve $\mathcal{K} = (a,b) \in \mathbb{K}$ ise $e_{\mathcal{K}}(x) = ax + b\ mod\ 26$ ve $d_{\mathcal{K}}(y) = a^{-1}(y - b)\ mod\ 26$ olur.

- $y = ax + b\ (mod\ 26)$ olduğunda, bu $ax = (y - b)\ (mod\ 26)$'ya eşit olur, bu durumda $OBEB(a, 26) = 1$ olacağından bu denkliğin her iki tarafı da a^{-1} ile çarpılabilecektir;

$$a^{-1}(ax) = a^{-1}(y - b)\ (mod\ 26) \quad (22)$$

devamında $mod\ 26$ 'ya göre birleşme özelliği denklemin soluna uygulanırsa;

$$a^{-1}(ax) = (a^{-1}.a)x = 1x = x \quad (23)$$

bulunur, bu durumda da;

$$x = [a^{-1}(y - b)]\ (mod\ 26) \quad (2)$$

olur ve bu x'in açık formülü olur. Sonuç olarak burada şifre çözme fonksiyonu;

$$d(y) = [a^{-1}(y - b)]\ (mod\ 26) \quad (3)$$

olur.

❖ **Örnek <u>Anahtar</u>:** Yukarıda yer alan açıklamalara göre anahtar $\mathcal{K} = (17,11)$ olsun, bu durumda;

$$17^{-1}\ mod\ 26 = 26$$

olur, burada $\mathbb{Z}_{26}^{+}$ olduğu varsayılırsa şifreleme için kullanılacak fonksiyon

$$e_{\mathcal{K}}(x) = 17x + 11$$

olur. Şifre çözme için kullanılacak fonksiyon ise;

$$d_{\mathcal{K}}(y) = 23(y - 11) = 23y - (23.11\ mod\ 26) = 23y - 19$$

olur. Burada $x \in \mathbb{Z}_{26}^{+}$ için $d_{\mathcal{K}}\big(e_{\mathcal{K}}(x)\big) = x$ eşitliğini sağlayan bütün x değerleri için karşılaştırmak önemlidir. $\mathbb{Z}_{26}^{+}$ içerisindeki değerler ile yapılacak işlemlerle;

$$\begin{gathered} d_{\mathcal{K}}\big(e_{\mathcal{K}}(x)\big) = d_{\mathcal{K}}(17x + 11) \\ = (23(17x + 11) - 19)\ mod\ 26 \\ = (361x + 253 - 19)\ mod\ 26 \\ = \Big(\big((26.13) + 23\big)x + 234\Big)\ mod\ 26 \\ = x \end{gathered}$$

bulunur.

❖ **Örnek <u>Şifreleme</u>:** Düz metnimiz (Plaintext) $\mathcal{P} =$ "$uskudar$" olsun, önce düz metnin bütün harfleri $\mathbb{Z}_{26}^{+}$'ya göre sayısal karşılıklarına dönüştürülür;

$$uskudar = 21, 19, 11, 21, 4, 1, 17$$

devamında bir anahtar belirlenir, burada anahtarın yukarıda verilen örnekte yer alan $\mathcal{K} = \{17,11\}$ olduğu varsayılsın. "uskudar" düz metninin sayısal karşılıkları $\{21, 19, 11, 21, 4, 1, 17\}$ bu anahtara göre sırasıyla aşağıdaki şekilde şifrelenir;

$$u = 21 = \big((17.21) + 11\big)\, mod\ 26 = 4$$
$$s = 19 = \big((17.19) + 11\big)\, mod\ 26 = 22$$
$$k = 11 = \big((17.11) + 11\big)\, mod\ 26 = 16$$
$$u = 21 = \big((17.21) + 11\big)\, mod\ 26 = 4$$
$$d = 4 = \big((17.4) + 11\big)\, mod\ 26 = 1$$
$$a = 1 = \big((17.1) + 11\big)\, mod\ 26 = 2$$
$$r = 17 = \big((17.17) + 11\big)\, mod\ 26 = 14$$

$\{4, 22, 16, 4, 1, 2, 14\}$ olarak elde edilen bu değerlerin $\mathbb{Z}_{26}^{+}$ içerisindeki karakter karşılığı ile değişimi sonrasında;

$$\mathcal{C} = \text{dvpdabn}$$

şifreli metni elde edilir.

3.2.4. Permütasyon Kriptosistem

Permütasyon kriptosistem, düz metin (platintext) değerinin veya karakterlerinin değiştirilmesi üzerine kuruludur.

Burada $m \in \mathbb{Z}^+$, $\mathcal{P} = \mathcal{C} = \mathbb{Z}_{26}^m$ ve $\mathcal{K}$, $\{1, 2, \dots, m\}$'nin bütün permütasyonlarını temsil etsin. Bir a anahtarı için şifreleme;

$$\mathcal{E}_a(x_1, x_\dots, x_m) = x\left(x_{a(1)}, x_{a(\dots)}, x_{a(m)}\right) \quad (4)$$

ve şifre çözme;

$$\mathcal{D}_a(y_1, y_\dots, y_m) = y\left(y_{a(1)}^{-1}, y_{a(\dots)}^{-1}, y_{a(m)}^{-1}\right) \quad (5)$$

olur.

- **Örnek:** $m = 4$ ve anahtar aşağıdaki Tablo X. $\mathcal{E}$ Permütasyon tablosu ile verilen $\mathcal{E}$ permütasyonu, $\mathcal{E}$'nin ters permütasyonu $\mathcal{E}' = \mathcal{D}$ ise Tablo XI. $\mathcal{D}$ Permütasyon tablosu ile verilen şekilde ve düz metin (plaintext) "üsküdar-üniversitesi" olsun,

Tablo X. $\mathcal{E}$ Permütasyon tablosu

1	2	3	4	5
3	5	2	1	4

Tablo XI. $\mathcal{D}$ Permütasyon tablosu

1	2	3	4	5
4	3	1	5	2

öncelikle düz metin beş harflik gruplar haline getirilir;

üsküd | ar-ün | $ivers$ | $itesi$

elde edilen bu gruplar a permütasyon tablosundaki sıralamaya göre yeniden düzenlenir;

üküds | ü-anr | $reisv$ | $seiit$

şifrelenmiş metnin çözümü için metin yeniden beş harflik gruplara bölünür ve $\mathcal{D}$ permütasyonuna göre yeniden düzenlenir ve birleştirilerek düz metne ulaşılır.

Permütasyon kriptosistem, çalışma devamında ele alınan Hill Kriptosistemin özelleştirilmiş halidir denilebilir. Burada $\{1, \dots, m\}$ kümesinin bir ε permütasyonu ile:

$$k_{i,j} = \begin{cases} 1, & j = \varepsilon(i) \\ 0, & j \neq \varepsilon(i) \end{cases} \quad (6)$$

fonksiyonuyla bağlantılı bir $m \times m$ olan bir $K_{\varepsilon}(k_{i,j})$ permütasyon matrisi tanımlanabilir.

Bir Permütasyon matrisinin bütün satır ve sütunlarında mutlaka bir adet 1 değeri bulunur ve geri kalan değerleri 0 olur. Permütasyon matrisi birim matrisinin satır ve sütunlarının permütasyonundan elde edilir.

3.2.5. Kayan (Shift) Kriptosistem

Kayan (Shift) kriptosistem, modüler aritmetiğe dayalı bir şifreleme yöntemidir. Burada $\mathcal{P} = \mathcal{C} = \mathcal{K} = \mathbb{Z}_{26}^{+}; 0 \leq \mathcal{K} \leq 25$ ve $x, y \in \mathbb{Z}_{26}^{+}$ olduğunda şifreleme işlemi;

$$\mathcal{E}_{\mathcal{K}}(x) = (x + \mathcal{K}) \; mod \; 26 \quad (7)$$

ve şifre çözme işlemi;

$$\mathcal{D}_{\mathcal{K}}(y) = \big((y + 26) - \mathcal{K}\big) \; mod \; 26 \quad (8)$$

olarak tanımlanır.

- **Bilgi:** Eğer $\mathcal{K} = 3$ olarak seçilirse, en çok bilinen Caesar kriptosistemi elde edilmiş olunur.

Kaydırma kriptosistemi için aşağıda yer alan Tablo XII ile verilen İngiliz alfabesinin sıralaması kullanılabilir. Burada harflerin sıralaması $mod\,26$ 'ya göre $a \to 0, b \to 1, c \to 2, \ldots \to \cdots, z \to 25$ sıralamasında kullanılır.

Tablo XII. İngiliz alfabesinin mod 26'ya göre 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:** Kaydırma anahtarı $\mathcal{K} = 8$ ve düz metin (plaintext) $\mathcal{P} =$ uskudarsecret olsun. İlk aşamada düz metin Tablo XII. İngiliz alfabesinin mod 26'ya göre karşılıklarına göre sayısal karşılıklarına dönüştürülür;

Plaintext:	**u**	**s**	**k**	**u**	**d**	**a**	**r**	**s**	**e**	**c**	**r**	**e**	**t**
	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
	20	18	10	20	3	0	17	18	4	2	17	4	19

Devamında her değere $mod\ 26$'ya göre 8 eklenir;

Plaintext:	**20**	**18**	**10**	**20**	**3**	**0**	**17**	**18**	**4**	**2**	**17**	**4**	**19**
	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
$\mathcal{P}_i + 8\ mod\ 26$:	2	0	18	2	11	8	25	0	12	10	25	12	1

Son olarak elde edilen bu sayısal değerler Tablo XII. İngiliz alfabesinin mod 26'ya göre karşılıklarından alfabetik karakterlere dönüştürülerek şifreli metin (ciphertext) elde edilmiş olunur;

$\mathcal{C}_{0-\infty}$:	**2**	**0**	**18**	**2**	**11**	**8**	**25**	**0**	**12**	**10**	**25**	**12**	**1**
	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
$\mathcal{C}_{A-Z}$:	C	A	S	C	C	L	Z	A	M	K	Z	M	B

Elde edilen şifreli metinden şifresiz metnin elde edilmesi için yukarıdaki sürecin tersine takip edilmesi yeterli olur.

Ancak $\mathcal{C} \to 2$ değerinin 8'e göre tersine $mod\ 26$ karşılığının bulunması için $\big((2 + 26) - 8\big)\ mod\ 26$ formülünün kullanılması daha uygun olur. Aksi durumda $(2 - 8 = -6)\ mod\ 26$ işlemi bir kısım bilgisayar kodlama sistemlerinde karışıklıklara sebep olabilecektir.

Bu yöntemde tek bir anahtar seçilir ve düz metne ait bütün karakterler bu anahtara göre değiştirilir. Bu metoda ***monoalfabetik*** şifreleme de denilmektedir.

3.2.6. Vigenere Kriptosistem

Vigenere kriptosistemde $m \in \mathbb{Z}^+$ ve $\mathcal{P}, \mathcal{C}, \mathcal{K} \in \mathbb{Z}_{26}^m$ için olduğunda bir $\mathcal{K} = (\mathcal{K}_1, \mathcal{K}_2, \mathcal{K}_{\dots}, \mathcal{K}_m)$ anahtarı için şifreleme;

$$\mathcal{E}_{\mathcal{K}} = (\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_{\dots}, \mathcal{C}_m) = (\mathcal{P}_1 + \mathcal{K}_1, \mathcal{P}_2 + \mathcal{K}_2, \mathcal{P}_{\dots} + \mathcal{K}_{\dots}, \mathcal{P}_m + \mathcal{K}_m) \quad (9)$$

ve şifre çözme;

$$\mathcal{D}_{\mathcal{K}} = (\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_{\dots}, \mathcal{C}_m) = (\mathcal{C}_1 - \mathcal{K}_1, \mathcal{C}_2 - \mathcal{K}_2, \mathcal{C}_{\dots} - \mathcal{K}_{\dots}, \mathcal{C}_m - \mathcal{K}_m) \quad (10)$$

olarak temsil edilir.

Tablo XII**.** İngiliz alfabesinin mod 26'ya göre karşılıkları ile $a \to 0, b \to 1, \dots, z \to 25$ bağıntısına göre m uzunlukta oluşturulan bir anahtar ve aynı uzunlukta bir düz metin mevcut olduğunda Vigenere yöntemi ile şifreleme yapılabilir.

- **Örnek:** Düz metin $\mathcal{P} =$ "$uubabe$" ve anahtar (Cipher) $\mathcal{K} =$ "$adlbil$" olsun. Burada $m = 6$ olur. Düz metnin Tablo XII'ye göre sayısal karşılığı;

Plaintext:	**U**	**U**	**B**	**A**	**B**	**E**
	↓	↓	↓	↓	↓	↓
	20	20	1	0	1	4

ve anahtarın karşılığı;

Cipher:	**A**	**D**	**L**	**B**	**I**	**L**
	↓	↓	↓	↓	↓	↓
	0	3	11	1	8	11

olur, devamında düz metin ve anahtar $mod\ 26$'ya göre toplanır;

Plaintext:	**20**	**20**	**1**	**0**	**1**	**4**
Cipher:	**0**	**3**	**11**	**1**	**8**	**11**
$\mathcal{P}_i + \mathcal{K}_i\ mod\ 26$:	↓	↓	↓	↓	↓	↓
	20	**23**	**12**	**1**	**9**	**15**

elde edilen bu sayısal değerlerin alfabetik karşılıkları yerine yerleştirildiğinde;

Ciphertext: $\mathcal{C} = UXMBJP$

şifreli metni elde edilir.

Aynı $\mathcal{K} =$ "$adlbil$" anahtarı ile $\mathcal{P} = adlibilimler$ düz metnini şifrelemek için yine 6 elemanlı "$adlbil$" anahtarı kullanılabilir, ancak burada $\mathcal{P}$ eleman sayısı $\mathcal{K}$ eleman sayısından büyük olduğu için 6'lı gruplar halinde şifreleme yapılır. Önce $\mathcal{P} = adlibilimler$ düz metni Tablo XII. İngiliz alfabesinin mod 26'ya göre karşılıklarına göre sayısal karşılıklarına dönüştürülür;

Plaintext:	**A**	**D**	**L**	**I**	**B**	**I**	**L**	**I**	**M**	**L**	**E**	**R**
	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
Plaintext:	**0**	**3**	**11**	**8**	**1**	**8**	**11**	**8**	**12**	**11**	**4**	**17**

düz metinden elde edilen sayısal değerler 6'lı gruplar halinde anahtarın sayısal karşılıkları ile $mod\ 26$'ya göre toplanır;

Plaintext:	**0**	**3**	**11**	**8**	**1**	**8**	**11**	**8**	**12**	**11**	**4**	**17**
Cipher:	**0**	**3**	**11**	**1**	**8**	**11**	**0**	**3**	**11**	**1**	**8**	**11**
	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
Ciphertext:	0	6	22	9	9	19	11	11	23	12	12	2

son olarak elde edilen sayısal değerlerin Tablo XII**.** İngiliz alfabesinin mod 26'ya göre karşılıkları'ye göre alfabetik karakter karşılıkları yerleştirilerek;

AGWJJTLLXMMC

şifreli metni elde edilir.

Elde edilen şifreli metinden (ciphertext) düz metne (plaintext) ulaşmak için aynı süreç uygulanacaktır. Sadece $mod\ 26$ 'ya göre toplama değil, çıkarma işlemi yapılacaktır.

Vigenere kriptosistemde anahtar (cipher) uzunluğu $m > 1$ olduğunda bu tür yöntemlere ***polialfabetik*** şifre sistemi denir ve Kaydırma kriptosistem gibi monoalfabetik şifrelerden daha güvenlidir denilebilir. Burada $mod\ 26$'ya göre işlem yapıldığı varsayılırsa m uzunlukta anahtarların sayısı 26^m olur. $m > 1$ uzunlukta anahtarlara; anahtar kelime (keyword)'de denilmektedir.

3.2.7. Hill Kriptosistem

Hill kriptosistem Lester S. HILL tarafından 1929 yılında bulunmuştur. Vigenere gibi bu yöntemde polialfabetik şifreleme yöntemlerindendir.

$m \in \mathbb{Z}^+$, $\mathcal{P}, \mathcal{C} \in (\mathbb{Z}_{29})^m$ olduğunda, düz metne ait m değerinin, m lineer kombinasyonu alınarak şifreleme yapılmasına dayanan bu yöntemde, şifreli metin elemanı içerisinde m değeri üretilir.

❖ **Örnek:** $m = 4$ olsun, bu durumda düz metin $\mathcal{P} = (\mathcal{P}_1, \mathcal{P}_2)$ ve şifreli metin $\mathcal{C} = (\mathcal{C}_1, \mathcal{C}_2)$ olarak temsil edilebilir. Burada $\mathcal{C}_1$, $\mathcal{P}_1$ ve $\mathcal{P}_2$ 'nin lineer bir kombinasyonu olacaktır. Aynı kombinasyon $\mathcal{C}_2$ içinde geçerli olur.

Burada;

$$\mathcal{C}_1 = 13\mathcal{P}_1 + 5\mathcal{P}_2 \text{ ve } \mathcal{C}_2 = 12\mathcal{P}_1 + 11\mathcal{P}_2$$

olsun. Bu durumda aşağıdaki matris elde edilir;

$$(\mathcal{C}_1, \mathcal{C}_2) = (\mathcal{P}_1, \mathcal{P}_2)\begin{bmatrix} 13 & 12 \\ 5 & 11 \end{bmatrix}$$

$\mathcal{K}$ anahtarı için yaygın olarak $m \times m$ olan bir matris kullanılır. $\mathcal{P} = (\mathcal{P}_1, \mathcal{P}_{\dots}, \mathcal{P}_m) \in \mathcal{P}$ ve $\mathcal{K} \in \mathbb{K}: \mathcal{C} = \mathcal{E}_{\mathcal{K}}(\mathcal{P}) = (\mathcal{C}_1, \mathcal{C}_{\dots}, \mathcal{C}_m)$ *şu şekilde bulunur.*

$$(\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_{\dots}, \mathcal{C}_m) = (\mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_{\dots}, \mathcal{P}_m)\begin{bmatrix} \mathcal{K}_{11} & \mathcal{K}_{12} & \cdots & \mathcal{K}_{1m} \\ \mathcal{K}_{21} & \mathcal{K}_{22} & \cdots & \mathcal{K}_{2m} \\ \vdots & \vdots & \vdots & \vdots \\ \mathcal{K}_{m1} & \mathcal{K}_{m2} & \cdots & \mathcal{K}_{mm} \end{bmatrix} \quad (11)$$

Bu durumda $\mathcal{C} = \mathcal{P}\mathcal{K}$ olur. Düz metinden şifreli metin elde edebilmek için linner dönüşüm gerekmektedir. Burada şifreli metnin çözümü de ele alınmalıdır ve $\mathcal{K}^{-1}$ test matrisi tanımlanmalıdır. Şifreli metnin çözümü ;

$$\mathcal{P} = \mathcal{C}\mathcal{K}^{-1} \quad (12)$$

olur.

Eğer $A = (a_i, j)$, $l \times m$ matrisi ve $B = \left(b_j, k\right)$ $m \times n$ matrisi olursa, burada matris çarpımı olan $AB = (c_i, k)$;

$$C_{i,k} = \sum_{j=1}^{m}\left(a_{ij} \,.\, b_{jk}\right),\ 1 \le i \le 1\ ,\ 1 \le k \le n \quad (13)$$

olur, bu durumda $m \times m$ birim matrisi lm ile temsil edilir. l matrisi 2×2 matrisi ise;

$$l_2 = \begin{bmatrix} 13 & 12 \\ 5 & 11 \end{bmatrix} \quad (14)$$

olarak temsil edilir.

$l \times m$ A matrisi ve $m \times n$ B matrisi ise $Al_m = A$ ve $Bl_m = B$ olur. Burada $m \times m$ A matrisinin tersi A^{-1} matrisi $A.A^{-1} = A^{-1}.A = lm$ olur. Ancak her matrisin tersi olmaz, var olması durumunda da yalnızca bir tersi bulur.

Bu durumda $\mathcal{P} = \mathcal{C}\mathcal{K}^{-1}$ şifreleme fonksiyonu için $\mathcal{C} = \mathcal{P}\mathcal{K}$ denkleminde her iki taraf $\mathcal{K}^{-1}$ ile çarpılır ve

$$\mathcal{C}\mathcal{K}^{-1} = (\mathcal{P}\mathcal{K})\mathcal{K}^{-1} = \mathcal{P}(\mathcal{K}.\mathcal{K}^{-1}) = \mathcal{P}lm = \mathcal{P} \quad (15)$$

bulunur. Yukarıda verilen $\begin{bmatrix} 13 & 12 \\ 5 & 11 \end{bmatrix}$ şifreleme matrisinin $\mathbb{Z}_{29}^{+}$'daki tersi aşağıda verildiği şekilde olur;

$$\begin{bmatrix} 13 & 12 \\ 5 & 11 \end{bmatrix}^{-1}$$
$$\equiv (\det \mathcal{K})^{-1} \begin{bmatrix} 11 & -12 \\ -5 & 13 \end{bmatrix} \bmod (29)$$
$$\equiv 7 \begin{bmatrix} 11 & 17 \\ 24 & 13 \end{bmatrix} \bmod (29)$$
$$\equiv \begin{bmatrix} 7.11 & 7.17 \\ 7.24 & 7.13 \end{bmatrix} \bmod (29)$$
$$\equiv \begin{bmatrix} 77 & 119 \\ 168 & 91 \end{bmatrix} \bmod (29)$$
$$\equiv \begin{bmatrix} 19 & 3 \\ 23 & 4 \end{bmatrix} \bmod (29)$$

Elde edilen bu şifreleme ve şifre matrisleriyle "$dört$" düz metnini şifrelemek istediğimizde, düz metin öncelikle sayısal[25] karşılıklarına çevrilir;

D	**Ö**	**R**	**T**
↓	↓	↓	↓
4	**18**	**20**	**23**

ve elde edilen bu değerler dö $\rightarrow (4,18)$ ve $rt \rightarrow (20,23)$ ikilileri olarak 2 ayrı gruba ayrıldıktan sonra şifreleme matrisine göre;

25 Burada Türk Alfabesinin 29 harflik sıralaması kullanılmıştır.

$$dö \rightarrow (4,18)\begin{bmatrix}13 & 12\\ 5 & 11\end{bmatrix} \equiv (4.13+18.5, \quad 4.12+18.11) \pmod{29}$$
$$rt \rightarrow (20,23)\begin{bmatrix}13 & 12\\ 5 & 11\end{bmatrix} \equiv (20.13+23.5, \quad 20.12+23.11) \pmod{29}$$

bulunur, burada sayısal değerlerin alfabetik karakter karşılıkları yerleştirildiğinde;

26	**14**	**27**	**0**
↓	↓	↓	↓
V	**L**	**Y**	**A**

elde edilir. Elde edilen şifreli metinden, düz metni elde etmek için şifre çözme matrisi kullanıldığında;

$$vl \rightarrow (26,14)\begin{bmatrix}19 & 3\\ 23 & 4\end{bmatrix} \equiv (26.19+14.23, \quad 26.3+14.4) \pmod{29}$$
$$ya \rightarrow (27,0)\begin{bmatrix}19 & 3\\ 23 & 4\end{bmatrix} \equiv (27.19+0.23, \quad 27.3+0.4) \pmod{29}$$

4	**18**	**20**	**23**
↓	↓	↓	↓
D	**Ö**	**R**	**T**

düz metin kolay bir şekilde elde edilmektedir (7).

$\mathcal{K}$ anahtarı için yaygın olarak $m \times m$ olan bir matris kullanılır. $\mathcal{P} = (\mathcal{P}_1, \mathcal{P}_{\dots}, \mathcal{P}_m) \in \mathcal{P}$ ve $\mathcal{K} \in \mathbb{K}: \mathcal{C} = \mathcal{E}_{\mathcal{K}}(\mathcal{P}) = (\mathcal{C}_1, \mathcal{C}_{\dots}, \mathcal{C}_m)$ *şu şekilde bulunur.*

3.3. Blok Kriptosistemler

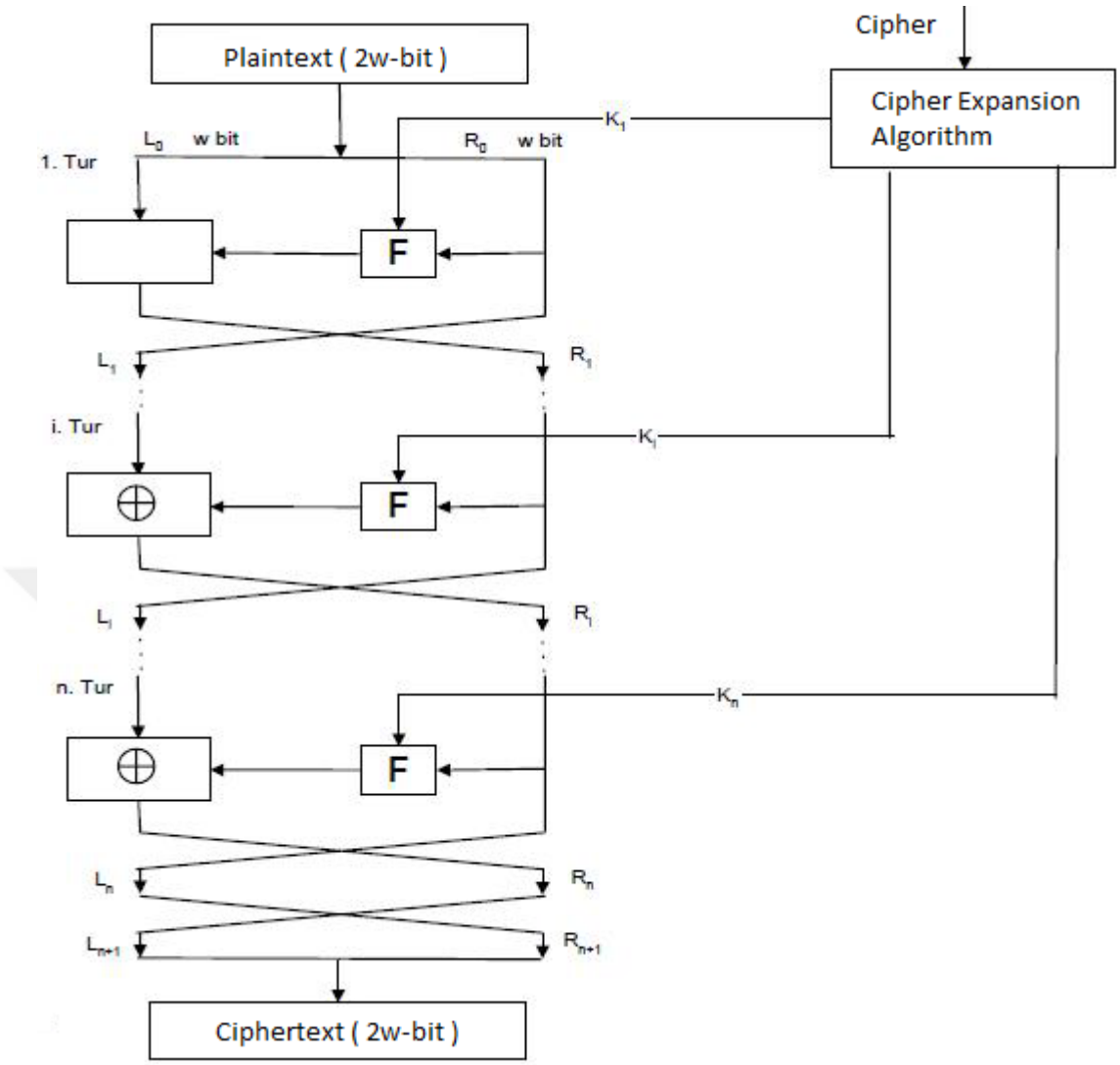
Blok kriptosistemlerde şifreleme (encryption) ve şifre çözmede (decryption) düz metinler (plaintext) sabit uzunlukta dizinlere bölünür ve bu bloklar üzerinden işleme

alınırlar. Bu yöntemi kullanan algoritmalar incelendiğinde bütün algoritmalarda 8 ve katları değerinde blok boyları kullanıldığı görülür. 8 ve katları olan blok boyutlarının kullanım gerekçeleri incelendiğinde, algoritmaların $GF(2^8)$ sonlu alanında çalışmaları ortak gerekçe olarak ortaya çıkmaktadır. $GF(2^8)$ sonlu alanı 8-bitlik değerler için kullanılmaktadır, bu da 1-byte olarak temsil edilen değere karşılık gelir.

Blok kriptosistemler için en çok bilinen örnek 1976 yılında IBM'in FIPS-46 kodu ile standartlaşan DE[26] algoritması olacaktır. Bu algoritma Horst FEISTEL'in tasarımını temel almıştır. A.B.D.'inde NIST tarafından güvenirliği onaylanarak hükümetin hassas verilerinin saklanması için kullanılmaktaydı. DES düz metni 64-bitlik bloklar halinde şifrelemektedir ve anahtar boyutu da 64-bit uzunluğundadır. Ancak anahtardaki işaret bitlerinin ayrılması durumunda anahtar boyutu 56-bite düşmektedir. Nitekim 56-bitlik alana 1999 yılında *Electronic Frontier Foundation* tarafından yapılan diferansiyel saldırı ile herhangi bir şifrelenmiş metnin (ciphertext) 22 saat 15 dakikada çözülebileceği pratik olarak ispat edilmiştir. Bu durum üzerine DES'in Blok boyunun üç katı olarak düzenlenmiş 192-bitlik yeni hali duyurulmuştur. Daha doğru bir tanımlama ile 168-bitlik tek bir blok alanı yerine üç ayrı 56-bitlik blok kullanılarak 168-bitlik anahtar alanı elde edilmiştir, bu yapıya 3DES ismi verilmiştir.

DES ve 3DES'in güvenilmez kabul edilmesinin ardından NIST'in 1997'de başlattığı bir yarışmayla Belçikalı Joan DAEMEN ve Vincent RIJMEN'e ait Rijndael kriptosistemi AES adıyla standartlaşan yeni şifreleme algoritması olmuştur. AES 128-bit, 192-bit ve 256-bit anahtar ve tek bir 128-bitlik düz metin bloğu boyutuna sahiptir. Blok halindeki düz metin ve anahtarı iteratif olarak işler.

26 Data Encryption Standard – Veri Şifreleme Standardı



Şekil 12: Feistel Ağı

Bugün için en çok bilinen blok kriptosistemler DES, 3DES, AES, IDEA, BlowFish, RC5, ve CAST-128 olarak sıralanabilir. DES kriptosistemi gibi birçok blok kriptosistem IBM'de çalışan Horst FEISTEL'in 1973 yılında geliştirdiği Feistel Ağını temel alır. Şekil 12: Feistel Ağı ile verilen bu yapı $2w - bit$ uzunluğa sahip şifresiz metni iki eşit parça olarak bölerek bunları algoritmaya sağ ve sol taraftan kabul eder. Burada her turda anahtardan türetilen tura özgü anahtar ile sağ taraf F fonksiyonundan geçirilir. Sol tarafta bu işlem sonucu XOR $(\oplus)$ işleminden geçirilmiş olur. Devamında

solda yer alan değer ile sağda yer alan değer yer değiştirir. Bu iterasyon n sayısınca devam eder. Burada F fonksiyonu ilk anahtardan türetilen alt anahtarlarla beslenir.

Feistel ağını temel alan blok kriptosistem yöntemlerinin, bu yapıdan miras aldığı en önemli özellikleri şöyledir;

1. **Alt Anahtar / Anahtar Genişletme:** İteratif yapıdaki kriptosistemler turlara özgü anahtarlara ihtiyaç duyar. Permütasyon içermeyen bir şifreleme algoritmasında aynı anahtarın her turda, kullanımı anahtarın $XOR\ (\oplus)$ işlemince şifrelenmiş veri üzerindeki etkisini sıfırlama sonucunu doğurur. Ayrıca anahtarın blok boyu kadar genişletilmesi her kriptosistemde yer almasa da uygulayıcı tarafında önemli faydalar sağlar. Uygulayıcı anahtar seçimini blok boyuna göre seçmeye zorlanması yerine alt algoritma ile bu işlemin anahtar genişletme olarak ele alınması mümkün olabilmektedir.

2. **Anahtar Uzunluğu:** Teoride anahtar uzunluğu güvenliği doğrudan etkilemektedir. Fakat düz verinin yerleştirildiği blok alanının boyutundan daha büyük anahtar boyutunun kullanılması kriptosistemin hızında olumsuz etkiye sebep olur.

3. **Blok Uzunluğu:** Anahtar uzunluğu gibi, blok uzunluğu da teoride güvenliği doğrudan etkiler. Düz veya şifrelenmiş metnin blok alanına hangi yapıda aktarıldığı ve blok üzerinde işlem yapan fonksiyonların yapısı kriptosistemin hızını olumsuz yönde etkileyebilecektir.

4. **On-The-Fly[27] Şifreleme / Şifre Çözme:** Kriptosistemlerin donanımsal gerçeklenmesine On-The-Fly denir. Bu gerçeklenmenin mümkün olabilmesi için kriptosistemin uygun yapıda olması gerekir.

5. **Tur Fonksiyonu:** Tur fonksiyonunun döngü içerisinde sonraki tura bıraktığı miras değişkenleri ve değişkenlerin bağıl parametreleri kriptanalize karşı direnç sağlar. Burada diferansiyel hesaplama göz önüne alınarak değişkenlerin fonksiyonca etkilenmesi önemlidir.

6. **Tur Sayısı:** Teoride tur sayısı güvenliği olumlu yönde etkiler. Bu teoriler diferansiyel saldırılara karşı yüksek değerli yayılım ve dağılım sağlanması üzerine kuruludur. Tur sayısının çok yüksek tutulması dışındaki yöntemlerle de diferansiyel direnci arttırılabilmektedir. Bunun için iterasyondaki değişkenlerin kaotik yapıda sonuçlar vermesi sağlanması gerekir.

7. **Uygulama Kolaylığı:** Kriptosistemlerin karmaşık yapıda tasarlanması gerekir. Ancak burada karışıklık ile uygulayıcıların kriptosistemi anlaması zorlaştırılmamalıdır. Asıl amaç sade algoritma ve kaotik işlem yapısı olmalıdır.

27 On-The-Fly: Donanım temelli anında gerçeklenme.

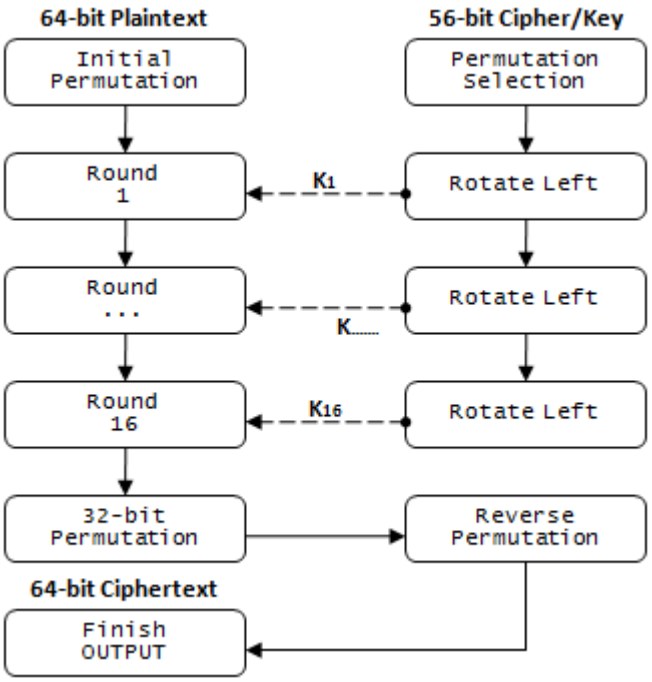
3.3.1. DES – Data Encryption Standart

DES 1974 yılında IBM tarafından geliştirilmeye başlanmış ve 1977 yılında NIST tarafından standart olarak tanımlanmıştır.

DES'in algoritma yapısı Feistel Ağına dayanır. Düz metin (plaintext) Blok boyu 64-bittir. Düz metin boyutu ne olursa olsun şifrelenmiş metin (ciphertext) her zaman 64-bit olur.

Şifreleme (encryption) ve şifre çözme (decryption) işlemlerinde kullandığı algoritma ve anahtar (cipher) aynıdır.

DES için anahtar uzunluğunun 64-bit olduğu duyurulmuştur. Ancak $GF(2^8)$ alanında 8-bitlik 8-byte değerinden oluşturulan 64-bitlik anahtar uzunluğu 56-bit olarak kullanılır. 8 ayrı byte değerinin her birisinin 8. bit değeri işaret biti olarak ihmal edilir. Burada 64-bitlik değer içerisinde 8., 16., 24., 32., 40., 48., 56. ve 64. bit değerleri ihmal edilmektedir.



Şekil 13: Des İterasyonu

Algoritma önce düz metni (plaintext) iki parçaya böler, metnin yarısını sağ, diğer yarısını sola yerleştirir. Devamında anahtar ile metin birleştirilerek bir sonraki adıma geçilir, bu iterasyon 16 tur boyunca devam eder. 16. adım bittiğinde sol ve sağdaki parçalar birleştirilerek şifrelenmiş metin (ciphertext) elde edilir. Bu tur yapısı aşağıda şekil Şekil 13**:** Des İterasyonu ile verilmiştir.

Des kriptosistem algoritması her adımda anahtarın bitlerini değiştirir. Burada 56-bitlik anahtarın 48-biti seçilir. Ayrıca verinin 32-bitlik sağ yarısı genişletme yapılarak 48-bite genişletilir. Bu genişletilen alan 48-biti seçilen anahtarla XOR $(\oplus)$ işleminden geçirilir. Devamında elde edilen sonuç S-Box isimli bir tabloya gönderilir ve farklı bir değer alır. Bu işlemlerin bir araya geldiği yapı $f(\,)$ fonksiyonu olarak anılır.

$f(\,)$ fonksiyonu ile elde edilen değer solda yer alan 32-bitlik yarım ile XOR $(\oplus)$ işleminden geçirilir. Son olarak elde edilen sağ yarım yeni bir değer almış olur ve sol yarım sağ yarımın eski değerinde olur. DES'in bu fonksiyonu, L sol yarımı, R sağ yarımı ve K anahtarı temsil ettiğinde, aşağıda gösterildiği şekilde temsil edilebilir.

$$L_i = R_{i-1}, \quad L_i = R_{i-1} \oplus f(R_{i-1}, K_i) \qquad (24)$$

3.3.2. Blowfish

Blowfish kriptosistem Bruce SCHEINER tarafından 1993 tarafından geliştirilmiştir. DES'in güvenilirliğinin tartışıldığı tarihten itibaren de en çok kullanılan alternatif haline gelmiştir. Halen çoğu Linux işletim sistemi dağıtımlarında standart olarak yer almaktadır. Bu kriptosistemin bu kadar yaygınlaşmasının altında yatan asıl sebep hiç kuşkusuz ki basit ve hızlı çalışan algoritmasıdır.

Blowfishin kaotik hesaplama yeteneğine karşın hesaplama algoritması basitçe anlaşılabilir durumdadır. Bu yapısı sadece 5-Kbitlik (5120-bit) bellek alanı ile çalışabilmektedir.

Kriptosistemin anahtar uzunluğu değişkendir ve en fazla 448-bite kadar genişleyebilmektedir. Ayrıca DES ve ileride ele alınan AES gibi S-Boxı kullanır. Ancak Blowfish için S-Box sabit bir yapıda değildir, anahtar değerlerine göre dinamik bir yapıya sahip olan S-Box değerleri kullanılır. Blowfish, S-Box ve alt anahtarların üretilmesi için 512 tekrarlı iterasyon uygular.

3.3.3. RC5

RC5, 1994 yılında Ron RIVEST tarafından *RSA* Laboratories'da geliştirilen bu kod *Ron's Code 5* olarak da bilinmektedir. Bu kriptosistemin ayrıca RC2 – *Ron's Code 2*, RC4 – *Ron's Code 4* ve RC6 – *Ron's Code 6* olarak bilinen üç sürümü daha bulunmaktadır. Ron RIVEST kriptolojide en çok bilinen çalışmalara katkı sağlayan yada imza atan kişi olarak bilinmektedir. Ayrıca en çok kullanılan asimetrik kriptosistem olan RSA'nın geliştiricilerinden biridir.

RC5'in temel özellikleri şöyledir;

1. **Basit yapı:** Algoritması kolay programlanabilir ve anlaşılabilir yapıya sahiptir.

2. **Değişken Anahtar Uzunluğu:** RC5 anahtar boyutunu parametre olarak kabul eder. İstenilen uzunlukta anahtarlar kullanabilmektedir. Anahtar uzunluğu ile güvenlik ilişkilidir. Bu sebeple anahtar uzunluğunun azaltılması hızlanmayı sağlarken güvenlik tarafında azalmaya sebep olur.

3. **Değişken Tur Sayacı:** RC5 tur sayısını da parametre olarak kabul eder. Tur sayısının düşürülmesi hızda artışa sebep olurken güvenlik tarafında azalmaya sebep olur.

4. **Değişik Kelime Uzunlukları:** RC5 kelime uzunluklarını değişken olarak kabul eder. Bu farklı mikro-işlemci mimarileriyle uyum sorununa karşı yerleştirilen parametredir.

5. **Donanım ve Yazılım ile Uyum:** RC5 bütün mikro işlemci mimarileriyle uyumlu primitif hesaplama operatörlerini kullanır.

6. **Düşük Bellek Kullanımı:** Düşük bellek gereksinimi sebebiyle alt düzey cihazlar veya akıllı kartlarda kullanılabilmektedir.

7. **Hız:** Basit ve kelime düzeyinde bit operatörleri kullanır. Ayrıca kelime üzerinde yapılacak düzenlemeler tek fonksiyonla yapılır. Bu yapısı RC5'in hızlı yanıt süresine sahip olmasını sağlar.

8. **Veriye Göre İşlem:** Verinin içeriği tur fonksiyonlarını etkiler. Bu değişkenlik kriptanalize karşı direnç gösterme yeteneğinde artış sağlar.

9. **Yüksek Güvenlik:** RC5'in parametrik anahtar, Blok boyu ve tur sayısı değişkenleri yüksek tutulduğunda yüksek seviyeli güvenlik sağlanabilir (12).

3.3.4. CAST-128

CAST-128, 1997 yılında Entrust Technology'den Carlise ADAMS ve Stafford TAVARES'in geliştirdiği yapıdır. Bu kriptosistem anahtar uzunluğunu 40-bitten 128-bite farklılaşan değerlerde parametre olarak kabul eder. CAST-128 algoritması da S-Box'ı kullanır. Ancak CAST-128'in kullandığı S-Boxların boyutları DES'ten daha büyüktür. Blowfish S-Boxları dinamik olarak kullanırken, CAST-128 için S-Box değerleri sabittir ve alt anahtar üretiminde bu S-Box değerlerini kullanır. Ayrıca işlem fonksiyonu $f()$ her turda değişen veriye göre farklı işlemler yapma yeteneğindedir (12).

3.3.5. AES - Rijndael

AES[28], NIST'in 1997 yılında DES kriptosisteminin yerine geçecek yeni bir yöntem bulmak için duyurusunu yaptığı yarışma ile ortaya çıkmıştır. Yarışmaya Rijndael kriptosistemiyle katılan Joan DAEMEN ve Vincent RIJMEN'den oluşan Belçikalı takımın kriptosistemi 2000 yılında birinci olarak açıklanmıştır. Bugün AES adıyla bilinen ve standartlaştırılan bu kriptosistemin asıl adı Rijndael'dir.

AES algoritmasında düz metin (plaintext) 128-bitlik bloklara, anahtar (cipher) ise 128-bit, 192-bit veya 256-bit bloklara sahiptir. AES kullanılan anahtar bloğu boyutuna göre AES-128, AES-192 ve AES-256 isimleriyle çağırılır.

28 AES: Advanced Encryption Standard – Gelişmiş Şifreleme Standardı

AES aşağıda verilen özelliklere sahiptir:

1. **Anahtar Uzunluğu:** Kullanılan anahtar uzunluğu ile seçilen anahtar blok uzunluğu aynı olmak zorundadır.

2. **Anahtar Türetme:** AES verili anahtardan tur sayısınca yeni anahtar *türetir*[29].

3. **Sütun Karıştırma:** AES veri bloğunu bir tablo yapısında ele alır. Sütun karıştırma işlemi için aşağıda verilen yapı kullanılır. Burada her bir sütun ile $GF(2^8)$ sonlu alanında $x^4 + 1$ 'e göre mod denkliğinde çarpımı alınarak işlem yapır.

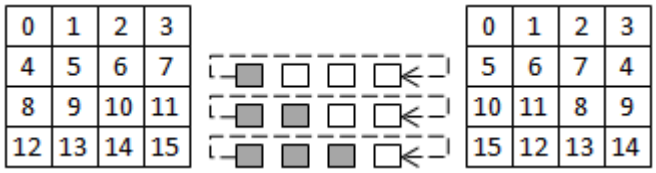
$$a(x) = \{03\}x^3 + \{01\}x^2 + \{01\}x^1 + \{02\}$$

$$\begin{bmatrix} S_0, c \\ S_1, c \\ S_2, c \\ S_3, c \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} S_0, c \\ S_1, c \\ S_2, c \\ S_3, c \end{bmatrix} \quad for\ 0 \leq c \leq Nb \qquad (16)$$

4. **Satır Öteleme:** Verilerin yerleştirildiği tablodaki alttaki 3 satırın birer artırımlı olarak sola kaydırılmasını temsil eder. Bu kaydırma aslında 8-bitlik bir döndürme işlemidir. Burada kaydırılacak satırın $GF(2^8)$ alanındaki temsili $\{a^{31} + b^{23} + c^{15} + d^7\}$ olsun, burada tek öteleme sonucu yeni satır değeri $\{b^{31} + c^{23} + d^{15} + a^7\}$ olur. Bu öteleme, tabloda alt satırlara indikçe artar. Öteleme ilk satırda 0, ikinci satırda 1,

29 Literatürde yer alan çalışmalar incelendiğinde AES'in anahtar genişletmesi uyguladığı belirtilir ancak AES sadece verili anahtardan tur sayısınca yeni anahtar türetir, verili anahtarı genişletmez, bu sebeple anahtar genişletme teriminin burada kullanılması doğru bir anlatım olmayacaktır.

üçüncü satırda 2 ve dördüncü satırda 3 adet 8-bitlik değeri kaydırır. Bu kaydırma işlemi aşağıda yer alan Şekil 14 ile temsil edilmiştir.



Şekil 14: AES - Satır Öteleme

5. **S-Box Başvurusu:** AES, S-Boxları kullanır. Burada 8-bitlik (1-byte) verilerin 16'lık sayı sistemi karşılığında yer alan değerler kullanılır. Bu 16'lık sayının solundaki değer S-Box tablosunun x-satır karşılığı, sağındaki değer S-Box tablosunun y-sütun karşılığıdır. Bu S-Box değerleri için $SBox[0-255]$ başvuru dizisini ve $SBox^{-1}[0-255]$ geri dönüş dizisini temsil etmek üzere, dizisel değerlerin elde edilmesi için aşağıda yer alan fonksiyon kullanılabilir;

$$\begin{aligned} &SBox[0] = 99 \\ &SBox^{-1}[99] = 0 \\ &for\ i = 1\ to\ 255 \\ &\begin{cases} x = (255 - \log(i))^2 \\ x = x; \quad y = (y \ll 1)|(y \gg 7) \\ x = x \oplus y; \quad y = (y \ll 1)|(y \gg 7) \\ x = x \oplus y; \quad y = (y \ll 1)|(y \gg 7) \\ x = x \oplus y; \quad y = y \oplus 99 \\ SBox[i] = x \\ SBox^{-1}[x] = i \end{cases} \end{aligned} \qquad (17)$$

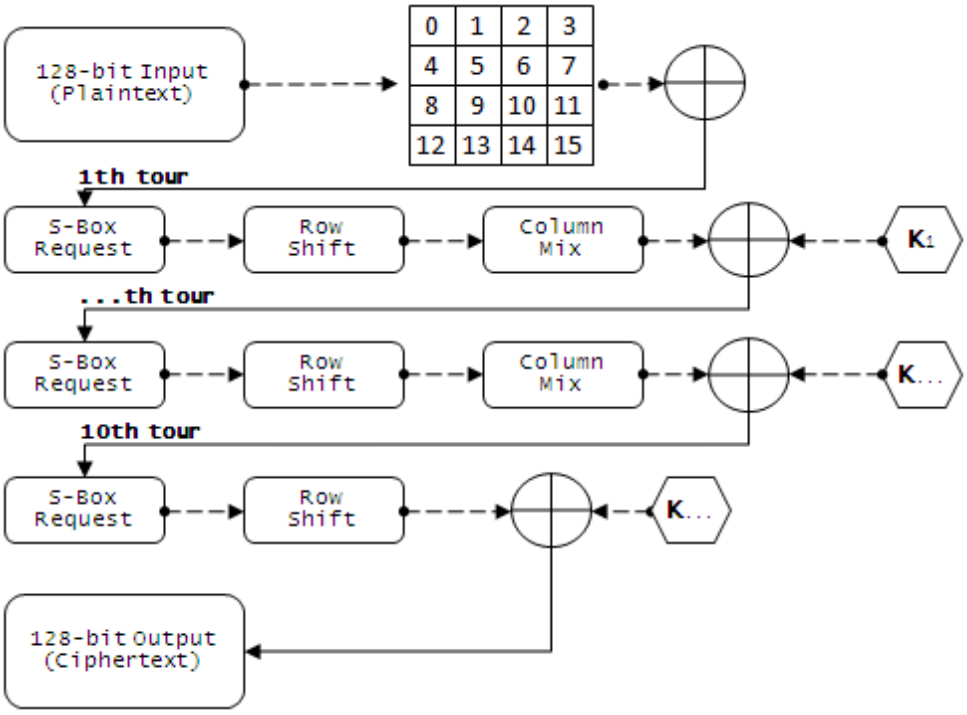
SBox başvuru tablosu aşağıda yer alan ile verilmiştir. AES'in kullandığı S-Box tablosu aşağıda yer alan Tablo XVIII ile verilmiştir. Bu tabloya göre

$be_{hex} = ae_{hex}$ $(46_{dec} = 30_{dec})$ ve $b2_{hex} = 37_{hex}(34_{dec} = 55_{dec})$ örnekleri verilebilir.

Tablo XIII. AES - SBox Tablosu

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**A**	**B**	**C**	**D**	**E**	**F**
0	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
3	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
4	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
5	53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
6	D0	EF	AA	FB	43	4D	33	85	45	F9	02	7F	50	3C	9F	A8
7	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
8	CD	0C	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73
9	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB
A	E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	91	95	E4	79
B	E7	C8	37	6D	8D	D5	4E	A9	6C	56	F4	EA	65	7A	AE	08
C	BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	8A
D	70	3E	B5	66	48	03	F6	0E	61	35	57	B9	86	C1	1D	9E
E	E1	F8	98	11	69	D9	8E	94	9B	1E	87	E9	CE	55	28	DF
F	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16

6. **Tur Sayısı:** AES 3 farklı anahtar uzunluğuna sahiptir ve bu anahtar uzunluklarına bağlı olarak 3 farklı tur sayısı olarak 128-bit için 10 tur, 192-bit için 12 tur ve 256-bit için 14 tur kullanır. AES Turları aşağıda verilen Şekil 15: AES - Kriptosistem Tur Yapısı ile temsil edilmiştir.



Şekil 15: AES - Kriptosistem Tur Yapısı

3.4. Asimetrik Kriptosistemler

Asimetrik kriptosistemlerin kullanımı 1970'li yıllarda başlamıştır. Asimetri adın da anlaşılacağı gibi şifrelemenin simetrik bir yapıya sahip olmasını anlatır. Literatürde bilinen asimetrik kriptosistemler anahtarın (cipher) asimetrisi üzerine kuruludur. Anahtar odaklı olarak yayınlanmış çalışmalar, şifreleme (encryption) için kullanılan açık (public) anahtar ve şifre çözme (decryption) için kullanılan gizli (private) anahtarı temel almaktadır. Ancak asimetrinin terimsel anlamı şifrelenmiş metnin (ciphertext) ve şifreleme yöntemini (encryption) içerisine alabilecek durumdayken bu çalışmanın yapıldığı tarihe kadar bu sınıflar yerini bulamamıştır.

Bilinen literatürdeki asimetrik kriptosistemlerin 1976 yılında Diffie ve Hellman'ın çalışması ile başladığı söylenebilir. Devamında 1977 yılında Rivest, Shamir ve Adlerman'ın RSA'sı yada El-Gamalın kendisi ile aynı ismi taşıyan yöntemi bilinen

yöntemler olmuştur. Bu yöntemler asal sayıların çarpanlarına ayrılması zorluğunu temel alırken, farklı temeldeki çalışmalarda yayınlanmaya başlamıştır.

Eliptik eğri şifresi (ECC) ve Kafes – NTRU kriptosistemi bu çalışmanın yayınlandığı tarihte asimetrik kriptosistemlerin yeni üyeleri olarak yerlerini almıştır. Ancak bu yöntemlerinde anahtar (cipher) odaklı olması alana yeni bir alt sınıfı değil, üyeleri kazandırmıştır.

Bilinen asimetrik anahtar [30] yöntemlerinde göndericisi düz metnini alıcıya göndermek için, alıcı tarafın herkesçe bilinen açık anahtarıyla düz metnini şifreler. Alıcı taraf şifreli metni sadece kendisinin bildiği gizli anahtarı ile açarak düz metni elde eder.

Asimetrik anahtar yönteminin simetrik kriptosistemlere katkı sağladığını söylemek daha doğru olur. Genel anlamda asimetrik anahtar yöntemleri yalnız başına bir şifreleme yöntemi olarak kullanılması yerine, simetrik kriptosistemler için anahtar üretilmesi ve paylaşılması için kullanılmaktadır.

Asimetrik anahtar yönteminin faydası şu şekilde özetlenebilir; eğer n kişilik bir grup söz konusuysa ve her bir kişinin diğerinin şifrelenmiş verisini çözememesi sağlanması gerekiyorsa bu durumda toplam $n-1$ ayrı anahtar gerekir. Bu grup yüzlerce veya binlerce kişiden oluştuğunda bu anahtarların üretimi için harcanması gereken güç ve zaman çok yüksek olacaktır. Burada asimetrik anahtar yöntemlerinin açık (public) anahtarı ile şifrelenmiş metin ancak gizli (private) anahtar ile çözülebilecektir. Bu da herkes için ayrı – ayrı anahtar üretilmesi ihtiyacını ortadan kaldıracaktır.

30 Bu çalışma içerisinde yer alan CUBE, şifrelenmiş metnin (ciphertext) asimetrisini sağladığından çalışmanın devamında asimetri terimi anahtar asimetrisi veya şifrelenmiş metnin asimetrisi ayrımıyla kullanılacaktır.

Asimetrik anahtar yöntemleriyle yapılan şifreleme (encryption) ve şifre çözme (decryption) işlemleri simetrik yöntemlere göre daha yavaştır. Bu durumda, bu yöntemlerin simetrik, yöntemler için anahtar paylaşım amaçlı kullanılma gerekçelerindendir.

Asimetrik anahtar kriptosistemler son derece basit algoritmik yapılarına karşın bu algoritmaların güvenliği teorik anlamda ispat edilememiştir. Güvenli iletişim ihtiyacının çözümü olarak öne sürülen bu yapının ispatsız güvenirliği son derece ciddi tehditler oluşturur. Nitekim Shor'un kuantum bilgisayarlarda kullanılmak üzere geliştirdiği çarpanlarına ayırma algoritması bu sorunu saniyeler içerisinde çözümleyebilmektedir (13).

3.4.1. Diffie-Hellman Kriptosistemi

Asimetrik anahtar alanında duyurulan ilk kriptosistemdir. Diffie ve Hellman tarafından ilk defa 1976 yılında "*New Directions in Cryptography*" adlı makalelerinde yer aldı. Bu çalışmada yer alan yöntem aslında bir anahtar değişim protokolüydü. Bu yöntem, makalenin yayınlanmasını takiben birçok alanda hemen talep görmüş ve yaygın bir kullanım alanı bulmuştur.

Yukarıda da değinildiği gibi bu yöntem aslında bir anahtar değişim protokolüdür. Doğrudan yayınlanmış bir açık (public) anahtar yapısı yerine alıcı ve verici arasında belirli sayısal değerlerin, aradaki dinleyicilerin anlamayacağı şekilde iletilerek iki tarafta da aynı değerde olan yeni bir anahtarın üretilebilmesini amaçlar.

Diffie-Hellman ayrık logaritma problemi üzerine kuruludur. Güvenirliği ancak çok büyük asal sayıların kullanılması ile mümkün olur.

Diffie-Hellman, $GF(2^n)$ sonlu alanında bir asal sayının tam sayı modu veya polinomsal alanda üstelleştirilmesini temel alır. Burada nb üstelleştirme $O([\log n]^3)$ mertebesinde olur. Güvenliği alanındaki logaritmik hesaplama zorluğunda dayalıdır. Bu nb ayrık logaritması $O\left(e^{\log n \log n \log n}\right)$ mertebesindedir.

Bu algoritmanın işleyişi temsili A ve B kişileri ele alınarak aşağıdaki şekilde açıklanmıştır;

1. A ve B takip edilme-izlenme riski olan bir iletişim kanalı üzerinden şifreleme için ortak bir anahtar (cipher) oluşturmak isterler.
2. Herkesin bilebileceği büyük bir asal sayı p[31] seçilir.
3. Herkesin bilebileceği $mod\ p$ primitif elemanı olan bir a seçilir.
4. A kişisi $x_A < p$ olan ve sadece kendisinin bildiği bir x_A seçer.
5. B kişisi $x_B < p$ olan ve sadece kendisinin bildiği bir x_B seçer.
6. A ve B birbirleriyle güvensiz iletişim kanalından paylaşacakları y_A ve y_B değerlerini aşağıdaki şekilde hesaplarlar;

$$y_A = a^{x_A}\ mod\ p \quad (18)$$

$$y_B = a^{x_B}\ mod\ p \quad (19)$$

7. A kişisi y_A değerini B kişisine, B kişisi de y_B değerini A kişisine güvensiz iletişim kanalından gönderirler. Burada gizli anahtar aşağıda verilen şekildedir.

31 Bu asal sayının yaklaşık 200 basamaklı olması önerilir.

$$K_{AB} = a^{x_A(x_B)} \, mod \, p \qquad (20)$$

8. A kişisi bilgilerle gizli anahtarı hesaplar;

$$(y_A)^{x_B} \, mod \, p \qquad (21)$$

9. B kişisi bilgilerle gizli anahtarı hesaplar;

$$(y_B)^{x_A} \, mod \, p \qquad (22)$$

10. A kişisi ve B kişisi güvensiz iletişim kanalı üzerinden ortak bir anahtar (cipher) üzerinde anlaşmış olurlar. Bu anahtarı şifreleme için kullanabilirler.

3.4.2. RSA

En çok bilinen ve kullanılan asimetrik anahtar kriptosistemidir. 1977 yılında Rivest, Shamir ve Adlerman tarafından geliştirilmiştir. *RSA* ismi geliştiricilerinin soyadlarının baş harflerinin bir araya getirilmesiyle türetilmiştir. Bu algoritmanın duyurulmasının ardından asimetrik anahtar kriptosistemine uyarlanmıştır. Bugün halen şifreleme, sayısal imza yöntemlerinde kullanılmaktadır.

Bu çalışmanın yapıldığı tarihte RSA halen güvenilir olarak kabul edilmekteydi. RSA'nın güvenirliği modüler aritmetik üzerine kurulu ve analizi asal sayıları çarpanlarına ayırmaya dayalı zorluğu üzerinedir.

RSA'nın çözülebilmesi birkaç farklı yöntem ile gerçekleşebilir. İlk yöntem iletişim kanalını izleyen bir kişinin belirlenen açık anahtara (public) uygun gizli anahtarı (private) bulmasıdır. Bu durumda iletişimi izleyen kişi bütün şifrelenmiş metinleri (ciphertext) görebilir veya sayısal imzaları taklit edebilir. Bunu yapmanın yolu algoritmada kullanılan p ve q asal sayılarının hesaplanması için n değerinin asal çarpanlara ayrılmasıdır. RSA'nın güvenirliği p ve q asallarının çok büyük değerlerde olması ile ilişkili olduğundan bu yöntem bilinen mikro-işlemci mimarisi ve algoritmalarla *zaman ve güç maliyeti* bakımından başarısız olur.

Diğer yönden RSA'nın temelini oluşturan asal sayıların çarpanlarına ayrılması zorluğu ispatlanmış bir konu değildir. Geçmişte Fermat ve Legendre gibi matematikçiler bu yönde çalışmalar yapmıştır (14). Ayrıca bu çalışmanın yapıldığı tarih için çokta uzak olması beklenmeyen bir gelecekte kuantum işlemci teknolojisinin tutarlılık kazanması halinde RSA'nın güvenirliği tamamen ortadan kalkmış olacaktır.

RSA'nın işleyişi temsili A ve B kişileri ele alınarak aşağıdaki şekilde açıklanmıştır;

1. A kişisi p ve q ile temsil edilen iki asal sayı seçer.
2. A kişisi p ve q değerlerine göre N ve Z değerlerini hesaplar.

$$N = p.q \qquad (23)$$

$$Z = (p-1)(q-1) \qquad (24)$$

3. A kişisi Z ile ortak böleni 1 olacak şekilde bir E sayısı bulur ve açık anahtar (public) değerleri $\{E, N\}$ olarak belirlenir.

4. A Kişisi gizli anahtar (private) değerleri olan $\{D, N\}$ için D değerini hesaplar.

$$D = E^{-1}\ mod\ Z \qquad (25)$$

5. A kişisi güvenliği önemsiz iletişim kanalı üzerinden $\{E, N\}$ açık anahtarını (public) B kişisine gönderir.

6. B Kişisi $\{E, N\}$ değerlerini aldıktan sonra $\mathcal{P}$ açık metnini $\mathcal{P} < N$ koşuluyla A kişisine göndermek için açık metnini şifreler (encryption).

$$\mathcal{C} \equiv \mathcal{P}^E\ mod\ N \qquad (26)$$

7. Şifreli metni (ciphertext) alan A tarafı aşağıdaki şekilde şifre çözme işlemini (decryption) gerçekleştirerek düz metne olaşır.

$$\mathcal{P} = \mathcal{C}^D\ mod\ N \qquad (27)$$

3.4.3. El-Gamal

El-Gamal kriptosistemi ile Diffie-Hellman kriptosistemiyle matematiksel bağlantıya sahiptir. Bu yöntemin güvenliği $p \in \mathbb{Z}^+$ olmak üzere Diffie-Hellman probleminin çözüm zorluğuna dayanır.

El-Gamal'ın işlem temsili A ve B kişileri ele alınarak aşağıdaki şekilde açıklanmıştır.

1. Anahtar üretimi (cipher);

a. Büyük bir p asal sayısı seçilir.

b. $mod\ p$ primitif elemanı olan bir a değeri seçilir.

c. A kişisi x_A ile temsil edilen ve $x_A < p$ koşulunda gizli bir değer belirler.

d. B kişisi x_B ile temsil edilen ve $x_B < p$ koşulunda gizli bir değer belirler.

e. A ve B kişileri açık anahtarları (public) için y_A ve y_B değerlerini belirlerler;

$$y_A = a^{x_A}\ mod\ p \quad (28)$$

$$y_B = a^{x_B}\ mod\ p \quad (29)$$

f. $0 \leq k \leq p-1$ ve $k \in \mathbb{Z}^+$ olmak üzere rastlantısal olarak bir k değeri seçilir.

g. $\mathcal{K}$ bir anahtar (cipher) olmak üzere hesaplanır;

$$\mathcal{K} = y_B^k\ mod\ p \quad \textbf{(30)}$$

2. Şifreleme (encryption): $\mathcal{P}$ düz metin (plaintext) ve $\mathcal{C}$ şifreli metin (ciphertext) olmak üzere şifreleme işlemi için;

a. Şifreli metin çifti $\mathcal{C} = \{c_1, c_2\}$ hesaplanır;

$$c_1 = a^k \ mod \ p \qquad (31)$$

$$c_2 = (\mathcal{K})\mathcal{P} \ mod \ p \quad (32)$$

3. **Şifre çözme (decryption)**;

   a. $\mathcal{K}$ anahtarı hesaplanır;

$$\mathcal{K} = c_1^{x_B} \ mod \ p = a^{(k)x_B} \ mod \ p \qquad (33)$$

   b. $\mathcal{P}$ düz metni (plaintext) aşağıdaki şekilde elde edilir.

$$c_2 = (\mathcal{K})\mathcal{P} \ mod \ p \quad (34)$$

3.4.4. Eliptik Eğri Şifre Kriptosistemi

Eliptik eğri şifre kriptosistemi sonlu alanlar temelinde matematiksel işlemlere dayanır. RSA ve benzeri diğer asimetrik anahtar kriptosistemlere göre esas avantajı kullandığı daha kısa anahtarlarıyla (cipher) alan ve zaman kazanımıyken diğer kriptosistemler ile eşit derecede güvenlik sağlayabilmektedir. Bir örnek ile RSA'nın 372-bitlik açık (public) anahtarının sağladığı güvenliği Eliptik Eğri Şifreleme 256-bit anahtar ile sağlayabilmektedir (10).

Aşağıda verilen ile NIST'in karşılaştırmalı anahtar boyutlarına dair önerileri yer almaktadır. Bu tabloda NIST asimetrik anahtarları simetrik kriptosistemleriyle birlikte değerlendirmiştir (15).

Tablo XIV. NIST Eliptik Eğri Şifre Tablosu

Tarih	En Düşük Dayanıklılık	Simetrik Algoritma	Çarpan Modları	Çarpanlara Ayırma Anahtar Grubu	Eliptik Eğri
2010	80	2 TDEA	1024	160 – 1024	160
2011-2030	112	3 TDEA	2048	224 – 2048	224
≫ 2030	128	AES-128	3072	256 – 3072	256
≫ 2030	192	AES-192	7680	384 - 7680	384

Eliptik Eğriler sonlu bir $\mathbb{F}_p$ cismindeki $E_{a,b}(\mathbb{F})$ eliptik eğrisi, $a, b \in \mathbb{F}_p$ ve $4a^3 + 27b^2 \not\equiv 0 \ (mod\ p)$ koşulunu sağlayan

$$y^2 \times^3 + ax + b \ (35)$$

eşitliği için $P = (x, y)$ noktalarına sonsuzluktaki $\mathcal{O}$ noktasının eklenmesi ile oluşturulan değerler kümesidir.

Bir eliptik eğri üzerindeki iki nokta için toplama işlemi teğet ve kiriş kuralı ile anılan grup operasyonuyla yapılır. Burada eliptik eğri komutatif bir grup oluşturur.

Eliptik eğrilerde sabitle çarpma işlemi ise $a \in \mathbb{Z}$ ve $P \in E_{a,b}\left(\mathbb{F}_p\right)$ için $a.P$ işlemi a adet P noktasının toplamı olur bu durumda $0.P = \mathcal{O}_E$ ve $1.P = P$ olarak tanımlanabilir.

Aşağıda yer alan örnek ile bir Eliptik Eğri Şifre uygulaması gösterilmiştir.

1. p asal sayısı $p = 11$ olarak seçilir.
2. $p = 11$ için $y^2 = x^3 + x + 6$ eliptik eğrisi üzerinde $P(10,2)$ ve $Q(5,9)$ noktaları için Q değerinin P tabanında ayrık logaritması bulunur. Bunun

için $k.P = Q$ eşitliğini sağlayan k değeri hesaplanır. P, Q elde edilene kadar kendisiyle toplanır.

$$P + P = 2P \quad (36)$$

3. $s \equiv \frac{(3.10^2)+1}{2.2} \ (mod\ 11)$, P noktasından geçen teğetin eğimi olmak üzere;

$$s \equiv \frac{(3.10^2)+1}{2.2} \ (mod\ 11)$$
$$= (3+1)(4)^{-1} \ (mod\ 11)$$
$$= 4.3 \ (mod\ 11) \equiv 1$$

ve;

$$x_{2P} \equiv (1 - 2.10) \ (mod\ 11) \equiv -19 \ (mod\ 11) \equiv 3$$

$$y_{2P} \equiv \big(-2 + 1(10-3)\big) \ (mod\ 11) \equiv 7$$

olur, bu durumda $2P = (3,5)$ olur.

4. $P + 2P = 3P$ ise;

$$s \equiv (5-2)(3-10)\ (mod\ 11)$$

$$s \equiv 3.(-7)^{-1}\ (mod\ 11) \equiv 3.4^{-1}\ (mod\ 11) \equiv 3.3\ (mod\ 11) \equiv 9$$

$$x_{3P} \equiv (9^2 - 10 - 3)\ (mod\ 11) \equiv -9\ (mod\ 11) \equiv 2$$

$$y_{3P} \equiv \big(-2 + 9(10-2)\big)\ (mod\ 11) \equiv 4$$

$$3P = (2,4)$$

olarak bulunur.

5. $P + 3P = 4P$ ise;

$$s \equiv (4-2)(2-10)^{-1}\ (mod\ 11) \equiv 2(-8)^{-1}\ (mod\ 11)$$

$$s \equiv 2.3^{-1}\ (mod\ 11) \equiv 2.4\ (mod\ 11) \equiv 8$$

$$y_{4P} \equiv (8^2 - 10 - 2)\ (mod\ 11) \equiv 8\ (mod\ 11) \equiv 8$$

$$y_{4P} \equiv \big(-2 + 8(10-8)\big)\ (mod\ 11) \equiv 3$$

olup, $4P = (8,3)$, $5P(7,2)$, $6P = 0$ olarak bulunur. Bu durumda Q değerinin P tabanındaki ayrıl logaritması 6 olur

Eliptik Eğri Şifresinde genelde izlenen protokol Diffie-Hellman anahtar paylaşım protokolü olmaktadır.

Eliptik eğrinin tanımlandığı asal cisim $\mathbb{F}$, başlangıç noktası P, skaler çarpım sonucu elde edilen Q noktası ve E eliptik açık anahtar (public) olup, k skaleri gizli

(private) anahtardır. $n.p = 0$ koşulunu sağlayan en büyük n olacak şekilde başlangıç noktası seçilir. Anahtar değişimi yapacak A ve B kişileri aşağıdaki adımları izler.

1. A kişisi, $k_n < n$ koşulunda bir $k_A \in Z$ sayısı seçer ve $k_A.P$ değerini hesaplar. Burada k_A değeri A kişisinin gizli anahtarı (private)., $k_A.P = P_A$ değeride A kişisinin açık anahtarıdır (public).
2. B kişisi, $k_n < n$ koşulunda bir $k_B \in Z$ sayısı seçer ve $k_B.P$ değerini hesaplar. Burada k_B değeri B'nın gizli anahtarı (private), $k_B.P = P_B$ değeride B'nın açık anahtarıdır (public).
3. A kişisi, B kişisinin açık anahtarı (public) ile $k_A.P_B$ değerini hesaplar.
4. B kişisi, A kişisinin açık anahtarı (public) ile $k_B.P_A$ değerini hesaplar.
5. Bu durumda;

$$k_A.P_B = k_A(k_B.P) = k_B.(k_A.P) = k_B.P_A \quad (37)$$

olarak bulunur.

Eliptik eğrilerde şifreleme için mesajın eliptik eğri üzerinde bir nokta olarak temsil edilmesi gerekir. Ancak burada dikkat edilmesi gereken her nokta eliptik eğri üzerinde olmayabilir.

Temsili bir A kişisi m mesajını B kişisine göndermek istesin. Burada P, E eliptik eğrisi üzerinde bir başlangıç noktasını temsil ettiğinde bu şifreleme işlemi aşağıdaki şekilde gerçekleşir (10):

1. A kişisi rastlantısal bir $n \in \mathbb{Z}$ değeri seçer ve P_m düz metninden şifreli metni C_m hesaplayarak B kişisine gönderir.

$$\mathcal{C}_m = \left(n_A.P.P_m + (n_A.P_B)\right) \quad (38)$$

2. B kişisi şifreli metin ($\mathcal{C}_m$)'den, açık metni (m) aşağıdaki şekilde elde eder.

$$P_m = P_m + (n_A B_A) - \left(P_B(n_A P)\right) \quad (39)$$

Bu işlemi bir örnekle vermek gerekirse $p = 17$ ve $P = (1,2)$ değerleri $y^2 = x^3 - x + 4$ eliptik eğrisi üzerinde bir başlangıç noktası olsun. B kişisinin açık anahtarı (public) $P_B = 4P = (21)$ olsun ve A kişisi $P_m = (11,10)$ düz metnini şifrelemek istesin. gerçekleşir (10);

1. A kişisi rastlantısal bir $n \in \mathbb{Z}$ değeri olarak $n = 5$ seçer.
2. A kişisi düz metnini;

$$\mathcal{C}_m\left(5(1,2), (11,10) + 5(12,13)\right)$$
$$= \mathcal{C}_m\left((15,13), (11,10) + (13,4)\right)$$
$$= \mathcal{C}_m\left((15,13), (11,7)\right)$$

olarak şifreler (encryption) ve B kişisine gönderir.

3. B kişisi şifreli metni;

$$P_m = P_m + n_A P_A - n_{B(n_A.P)} =$$
$$= (11,7) - 4\left(5(1,2)\right)$$
$$= (11,7) - 4(15,13)$$

$$= (11,10)$$

işlemiyle çözerek (decryption), düz metni elde eder.

3.4.5. Kafes – NTRU Kriptosistem

NTRU halka tabanlı asimetrik anahtar kriptosistemidir. İlk olarak J.Hoffstein ve J.H.Silverman tarafından geliştirilmiştir (16). İlk versiyonlarından sonra birkaç kez değiştirilmiş ve optimize edilmiştir, fakat temel ilkeler aynı kalmıştır (17).

NTRU, belirlenmiş polinom halkalarının cebirsel yapılarını temel alarak geliştirilmiş bir kriptosistemdir. Bilinen diğer asimetrik anahtar kriptosistemlerden farklı olarak güvenliğini en kısa kafes vektörünü bulma zorluğundan alır.

NTRU'nun diğer asimetrik anahtar kriptosistemlere karşı önemli avantajları vardır bunlar;

- Açık (public) ve gizli (private) anahtar çiftlerini hesaplama kolay ve hızlıdır.
- Şifreleme (encryption) ve şifre çözme (decryption) işlemleri güç ve zaman maliyeti yönünden hızlıdır.
- Diğer asimetrik anahtar kriptosistemlerin güvenliğini daha kısa anahtar uzunluklarıyla sağlayabilmektedir.

NTRU kriptosisteminin kullandığı temel elemanlar $n \in \mathbb{Z}^+$ olmak üzere $\mathcal{R} = \mathbb{Z}[x]/_{< x-1 >}$ halkasında yer alan polinomlardır. Bu polinomlar şifreleme ve şifre çözme işleminde aralarında asal olan iki tam sayıya indirgenir. Bu tam sayı

parametreleri p ve q ile temsil edilir *ancak asal olmaları zorunlu değildir*. Burada dikkat edilmesi gereken q değerinin p değerinden çok büyük olmasıdır.

NTRU, kriptosisteminde yapılacak hesaplamalar $\mathcal{R}$ halksında yapılır. Buradan hareketle bir $a \in \mathcal{R}$ değeri bir polinom veya vektör olarak temsil edilebilir;

$$a = \sum_{i=0}^{n-1}(a_i . x^i) = [a_0, a_1, a_{\dots}, a_{n-1}] \qquad (40)$$

Burada bir $\mathcal{R}$ halkasında bii çarpma işlemi;

$$a = a_0 + (a_1 . x^1) + (a_2 . x^2) + (a_{\dots .} x^{\cdots}) + (a_{n-1} . x^{n-1}) \in \mathcal{R} \quad (41)$$

$$b = b_0 + (b_1 . x^1) + (b_2 . x^2) + (b_{\dots .} x^{\cdots}) + (b_{n-1} . x^{n-1}) \in \mathcal{R} \quad (42)$$

olduğunda $\forall k = \{0,1, \dots, n-1\}$ için;

$$\mathcal{C}_k = \sum_{i=0}^{k}(a_i . b_{k-i}) + \sum_{j=k+10}^{n-1}(a_j . b_{n-j}) \quad (43)$$

olmak üzere;

$$a.b = c = \left(c_0 + (c_1 . x) + (c_2 . x^2) + (c_{\dots} . x^{\cdots}) + (c_{n-1} . x^{n-1})\right) \qquad (44)$$

olarak tanımlanır. Bu işlem devirli evrişim çarpımı (cyclic convolution product) olarak bilinir.

Şifreleme (encryption) ve şifre çözme (decryption) işleminde bu çarpma işlemi yapılırken katsayılar $mod\ q$ 'ya indirgenir. $mod\ q$ ile işlem yapılan halka $\mathcal{R} =$

$\mathbb{Z}[x]/(x^n - 1)$ ile temsil edilir. Buna göre $n.$ dereceden iki polinomun çarpımı için n^2 adet çarpma işlemi yapılması gerekir. Yüksek işlem sayısını gerektiren bu durum sebebiyle NTRU kriptosisteminde küçük katsayılı polinomlar kullanılır.

Şifreleme için bilinmesi gereken bir değer işlem polinomların tersinin alınmasıdır. $\mathcal{R}_q$ halkasında bir a polinomunun tersi $a.a^{-1} = a^{-1}.a = 1$ olan $a^{-1} \in \mathcal{R}_q$ polinomudur. Ancak $\mathcal{R}_q$ halksında her polinomun tersi bulunmaz. $f(1) = 1$ koşulunu sağlayan rastlantısal seçilen bir $f \in \mathcal{R}_q$ polinomunun yüksek olasılık ile var olduğu Silverman'ın çalışmasıyla ispat edilmiştir (18).

NTRU kriptosisteminde şifreleme yapmak için üç adım gerçeklenmelidir, ilki anahtar üretimi, ikincisi şifreleme ve üçüncüsü şifre çözme adımlarıdır.

Anahtar üretiminde $\mathcal{R}$ halkasının bir alt kümesi olan L_f kümesinin rastlantısal küçük bir $mod\ p$'de tersi olduğu doğrulanır. Eğer f polinomunun tersi yoksa (tersini değilse) bir başka rastlantısal polinom seçilir. Tekrar tersinir olduğu doğrulanır.

Tersinir polinom bulunduktan sonra yine $\mathcal{R}$ halkasından bir alt küme olan L_g kümesinden rastlantısal küçük değerli bir g polinomu seçilir ve

$$h \equiv p f_q^{-1}.g\ (mod\ q) \qquad (45)$$

hesaplanır. Böylece f_q^{-1} , f_p^{-1} ve g polinomları gizlenerek h açık (public) anahtarı ve f gizli anahtarı elde edilmiş olunur.

- **Örnek:** $n = 11$, $q = 32$, $p = 3$, $d_f = 4$ ve $d_g = 3$ olsun. Bu değerlerde bir NTRU kriptosistemi için anahtar çifti üretilecektir.

Burada f polinomu 10. Dereceden olmalıdır ve dört katsayısı 1, üç katsasayısı -1 ve diğer katsayıları 0 olmalıdır.

g ise 10. Dereceden üç katsayısı 1, üç kat sayısı -1 ve geri kalan katsayıları 0 olan bir polinom olmalıdır;

$$f = -1 + x + x^1 - x^4 + x^6 + x^9 - x^{10} \quad (46)$$

$$g = -1 + x^2 + x^3 + x^5 - x^8 - x^{10} \quad (47)$$

f ve g polinomları yukarıda verilenler olsun. Devamında f polinomunun $mod\ p$ ve q'da tersi bulunur;

$$f_p^{-1} = 1 + 2x + 2x^3 + 2x^4 + x^5 + 2x^7 + x^8 + x^9 \quad (48)$$

$$\begin{aligned} f_q^{-1} = 5 + 9x + 6x^2 + 16x^3 + 4x^4 \\ + 15x^5 + 16x^6 + 22x^7 \\ + 20x^8 + 18x^9 + 30x^{10} \end{aligned} \quad (49)$$

Bu değerler belirlendikten sonra anahtar üretimi aşağıdaki adımla sonuçlanır;

$$h \equiv \left(p f_q^{-1}.g\right)(mod\ q) \quad (50)$$

$$h \equiv \big(3(5 + 9x + 6x^2 + 16x^3 + 4x^4 + 15x^5 + 16x^6 + 22x^7 + 20x^8 + 18x^9 + 30x^{10}).(-1 + x^2 + x^3 + x^5 - x^8 - x^{10})\big) \ (mod\ 32) \quad (51)$$

$$h \equiv 8 + 25x + 22x^2 + 20x^3 + 12x^4 + 24x^5 + 15x^6 + 19x^7 + 12x^8 + 19x^9 + 16x^{10} \quad (52)$$

Buradan elde edilen

açık anahtar (public);

$$h \equiv 8 + 25x + 22x^2 + 20x^3 + 12x^4 + 24x^5 + 15x^6 + 19x^7 + 12x^8 + 19x^9 + 16x^{10} \quad (53)$$

Gizli anahtar (private):

$$f = -1 + x + x^1 - x^4 + x^6 + x^9 - x^{10} \quad (54)$$

olur. Yukarıda yer alan anahtar üretim işlemleri iki adet kafes tabanı hesaplanmasına imkân verir. Buradaki gizli anahtara ait $3n$ boyutlu kafese ait alt tabanın matrisi;

$$L_{(private)}\begin{bmatrix} I & M_f & I \\ 0 & qI & qM_{f_p^{-1}} \\ 0 & 0 & pI \end{bmatrix} \quad (55)$$

olur. Diğer yandan açık anahtar (public) ait $2n$ boyutlu matris;

$$L_{(public)}\begin{bmatrix} I & M_f \\ 0 & qI \end{bmatrix} \quad (56)$$

olur.

Şifreleme için $\mathcal{R}$ halkasının bir alt kümesi L_M düz metin kümesinden bir M seçilir. $\mathcal{R}$'de bir alt küme olan L_r kümesinden rastlantısal küçük bir r polinomu seçilir. Bu polinoma M düz metnini (plaintext) gizlemek için körleştirme değeri[32] polinomu denir. Burada $\mathcal{C}$ şifreli metnini elde etmek için

$$\mathcal{C} \equiv (r.h) + M\ (mod\ q) \quad (57)$$

hesaplanır. Bu durumda $n = 11$, $q = 32$, $p = 3$, $d_f = 4$, $d_g = 3$ ve $d_r = 3$ için değerler ve şifreleme aşağıdaki şekilde olur;

Açık anahtar (public-cipher):

$$h \equiv 8 + 25x + 22x^2 + 20x^3 + 12x^4 + 24x^5 + 15x^6 + 19x^7 + 12x^8 + 19x^9 + 16x^{10} \quad (58)$$

32 Körleştirme Değeri: Blinding-Value

Düz Metin (plaintext):

$$m = -1 + x^3 - x^4 - x^8 + x^9 + x^{10} \qquad (59)$$

r değeri:

$$r = -1 + x^2 + x^3 + x^4 - x^5 - x^7 \qquad (60)$$

Şifreli metin (ciphertext):

$$\mathcal{C} \equiv (\mathrm{r.h}) + \mathrm{M} \pmod{\mathrm{q}} \qquad (61)$$

$$\begin{aligned}\mathcal{C} \equiv ([-1 + x^2 + x^3 + x^4 - x^5 - x^7].[8 \\ + 25x + 22x^2 + 20x^3 \\ + 12x^4 + 24x^5 + 15x^6 \\ + 19x^7 + 12x^8 + 19x^9 \\ + 16x^{10}]) \\ + (-1 + x^3 - x^4 - x^8 \\ + x^9 + x^{10}) \pmod{32}\end{aligned} \qquad (62)$$

$$\begin{aligned}\mathcal{C} \equiv 14 + 11x + 26x^2 + 24x^3 + 14x^4 \\ + 16x^5 + 30x^6 + 7x^7 \\ + 25x^8 + 6x^9 + 19x^{10}\end{aligned} \qquad (63)$$

$\mathcal{C}$ Şifreli metni (ciphertext) elde edilmiş olunur. Bu şifreleme işlemi $[0, \mathcal{C}] = [r, \psi]\begin{bmatrix} I & M_f \\ 0 & qI \end{bmatrix} + [-r, m]$ olarak da temsil edilebilir. Burada ψ vektörü, $\mathcal{C}$ polinomunun katsayılarını $mod\ p$'ye göre belirlenen aralıkta tutacak şekilde seçilir.

Şifre çözme işlemi için f gizli anahtarı (private cipher) ve f_p^{-1} kullanılarak düz metne ulaşılır. Burada şifre çözme (decryption) işlemi aşağıdaki şekilde olur;

$$a \equiv (f.\mathcal{C})\ (mod\ q) \quad (64)$$

$$a' \equiv a\ (mod\ p) \quad (65)$$

$$d \equiv \left(f_p^{-1}\right)\ (mod\ p) \quad (66)$$

olmak üzere, burada d polinomu yüksek olasılıkta M düz metnine eşit olacaktır. *Ancak bazı parametre değerleri hesaplamanın başarısız olmasına sebep olabilir. Bu sebeple her metin bloğuna birkaç kontrol biti eklenir.* Burada şifre çözme işlemindeki (decryption) başarısızlığın sebebi polinomların doğru şekilde ortalanmamasıdır.

- **Örnek:** Bir önceki örnekteki $n = 11$, $q = 32$, $p = 3$, $d_f = 4$, $d_g = 3$ ve $d_r = 3$ parametreleri ile elde edilen;

$$\begin{aligned} \mathcal{C} \equiv 14 + 11x + 26x^2 + 24x^3 + 14x^4 \\ + 16x^5 + 30x^6 + 7x^7 \\ + 25x^8 + 6x^9 + 19x^{10} \end{aligned} \quad (67)$$

şifreli metninin çözümü için önce a polinomu hesaplanarak katsayıları $\left(\left[-q/2\right],[q/2]\right)$ aralağına indirgenirek,

$$a \equiv (f.\mathcal{C})\ (mod\ q) \qquad (68)$$

$$\begin{aligned} a \equiv (&-1 + x + x^1 - x^4 + x^6 + x^9 \\ &- x^{10}).(14 + 11x + 26x^2 \\ &+ 24x^3 + 14x^4 + 16x^5 \\ &+ 30x^6 + 7x^7 + 25x^8 \\ &+ 6x^9 + 19x^{10})\ (mod\ 32) \end{aligned} \qquad (69)$$

$$\begin{aligned} a \equiv 3 &- 7x - 10x^2 - 11x^3 + 10x^4 \\ &+ 7x^5 + 6x^6 + 7x^7 + 5x^8 \\ &- 3x^9 - 7x^{10}\ (mod\ 32) \end{aligned} \qquad (70)$$

$$\begin{aligned} a^{-1} \equiv &-x - x^2 + x^3 + x^4 + x^5 + x^7 \\ &- x^8 - x^{10}\ (mod\ 32) \end{aligned} \qquad (71)$$

bulunur. Devamında $d \equiv \left(f_p^{-1}\right)(mod\ p)$ denkliği bulunarak M düz metnine (plaintext) ait polinoma ulaşılır,

$$d \equiv (1 + 2x - 2x^3 + 2x^4 + x^5 + 2x^7 + x^8 + x^9).(-x - x^2 + x^3 + x^4 + x^5 + x^7 - x^8 - x^{10}) \ (mod\ 3) \quad (72)$$

$$d \equiv -1 + x^3 - x^4 - x^8 + x^9 + x^{10} \ (mod\ 3) \quad (73)$$

Aslında şifre çözme işlemini (decryption) daha kolay gerçeklendirmek mümkündür. Bunun için $[\mathcal{C}, 0,0]$ vektörüne en yakın olan kafes vektörünü bulmak gerekir;

$$[\mathcal{C}, a, b] = [\mathcal{C}\lambda\mu]\begin{bmatrix} I & M_f & I \\ 0 & qI & qM_{f_p^{-1}} \\ 0 & 0 & pI \end{bmatrix} \quad (74)$$

Yukarıda verilen işlemde λ vektörü $a = (f.\mathcal{C}) + (q.\lambda)$ değeri için seçilen en küçük λ polinomunun katsayı vektörüdür. Burada,

$$a = f.\mathcal{C} + (q\lambda) = (p.r)q + (f.m) \quad (75)$$

eşitliği sağlandıktan sonra μ vektörü

$$b = \left(\mathcal{C} + \left(qf_p^{-1}.\lambda\right) + (p.\mu)\right) \quad (76)$$

eşitliğini minimize edecek şekilde seçildiyse $b = m$ olur. Bu durumda da şifre çözme işlemi (decryption) tamamlanmış olur.

4. MESAJ ÖZETLEME / HASH ALGORİTMALARI

Mesaj özetleme algoritmaları giriş verisi olan mesaja ve algoritmaya bağlı olarak matematiksel yöntemlerle sabit uzunlukta ve kısaltılmış sayısal değerler üretirler. Bu değerler anlamsız olup, bu değerlerden giriş olan orijinal mesajın yeniden hesaplanması mümkün olmamalıdır. Özetleme fonksiyonları kriptosistemler, sayısal imza yöntemleri, sayısal deliller, şifre güvenliği, kimlik ispatı ve veri bütünlüğü ispatı gibi birçok alanda kullanılır. Ayrıca büyük veri (big-data) için veri ile tekil değerler üretilmesi sayesinde arşivleme, işleme ve anlamlandırma uygulamaları bu fonksiyonlar için başvurulan önemli alanlardır.

Ek olarak bu fonksiyonlar sayısal delillerin arşivlenmesinde ve/veya sayısal delillerin bütünlüğünü ispat amacıyla de-facto yada de-jure olarak ulusal yargı mekanizmalarının başvuru kaynağıdır. Buradan anlaşılacağı gibi bu fonksiyonlar insanlar farkında olmadan birçok temel hak ve özgürlüklere etki etmektedir.

Bugün en çok kullanılan ve bilinen mesaj özetleme algoritmaları MD (MD2 (19), MD3, MD5 (20), MD6 (21)) ve SHA (SHA0, SHA1, SHA2) (22) aileleriyken HAVAL (23), Whirlpool (24), ECOH (25) ve Fuque (26) diğer bilinen örnekleri oluştururlar. Yerel kaynaklarla geliştirilip NIST'e öneri olarak gönderilmesi amaçlanan Kerem VARICI'nın Sarmal (27) fonksiyonu da bu çalışma kapsamında özellikle belirtilmesi gereken fonksiyonlardandır.

Yukarıda açıklanan fonksiyonların hepsi Galois Field sonlu alanında sonlanmış elemanları ve çarpım, toplama, çıkarma işlemlerini kullanırlar. Bu fonksiyonlar incelendiğinde hepsinin bilinen yapı temelleri üzerine kurulduğu ve hemen-hemen hiçbirisinin yenilikçi bir yapıda olmadığı görülür. Bu fonksiyonların dağılım ve yayılım işlemleri için aynı s-box yapıları aynı eleman değerleriyle kullandığı görülmektedir.

Bu çalışmanın yazıldığı tarihteki mevcut literatürde yer alan mesaj özetleme algoritmaları gelişen teknoloji ile birlikte güvenirliklerini yavaş – yavaş yitirmektedir. Wang ve diğerleri tarafından MD5CRK projesi ile önce 17 Ağustos 2004'te sadece bir IBM P690 bilgisayar ile MD5'e düzenledikleri ve veri genişletme fonksiyonunun yayılım özelliğinin GF(2n) düşük olmasını kullanan analitik saldırı ile 1 saatlik bir zaman diliminde başarıya ulaştıklarını duyurmuşlardır (28). Devamında 10 Mart 2006 tarihinde Vlastimil KLIMA bir laptop ile 60 saniye içerisinde MD5 çakışmaları türetebilen tünel adını verdiği bir algoritmayı yayınlamıştır (29). Bu ailelere yapılan saldırılar bu fonksiyonların güvenilirliğini zedelemiş, daha uzun ve değişken özet boyutlarına sahip fonksiyonların uygulanması gerekliliği ortaya çıkmıştır.

4.1. Kullanım Alanları

Mesaj özetleme (*HASH*) terimi ilk defa 1953 yılında Birleşmiş Milletlerde bilgisayar bilim araştırmacısı olan Hans Peter LUHN tarafından kullanılmıştır. Ancak literatürde teknik bir terim olarak kullanımdan resmi bir terime dönüşümü Robert MORRIS tarafından CACM[33] dergisinde yayınlanan bir makale ile olmuştur.

Mesaj özetleme fonksiyonlarının kullanım alanları aslında uygulayıcı ve amaca bağlı olduğundan kullanım alanları konusunda keskin sınırlar koymak mümkün değildir. Ancak bu fonksiyonların kullanıcılar tarafında farkında olmaksızın sıklıkla kullanıldığı bir gerçektir.

Kullanıcılar tarafında özellikle kamu kurumu personelinin kullandığı e-imza sistemleri bu fonksiyonlar temelinde kuruludur. Örnek verilebilecek birkaç alan

33 CACM: Communications of ACM

içerisine web uygulamalarında sıklıkla kullanılan SSL[34] veya veritabanına kayıt olan kullanıcı şifrelerini anlamsızlaştırma uygulamaları da girebilecektir. Bütün bu uygulamaların mesaj özetleme algoritmaları üzerine kurulu olduğunu belirtmek, kullanım sıklığını örneklendirmek açısından önemlidir.

Hash fonksiyonlarının bilinen kullanım alanları şu şekilde ana başlıklar halinde sunulabilir;

- **Sayısal İmza (Digital Signature):** Burada $\mathcal{H}$ mesaj özeti değeri, $\mathcal{M}$ imzalanacak mesajı, f_e şifreleme işlemini, f_d şifre çözme işlemini, k_x açık anahtarı (public cipher), k_y gizli anahtarı (private cipher) ve S sayısal imza değeri olduğunda $\mathcal{M}$ mesajının imzalanması;

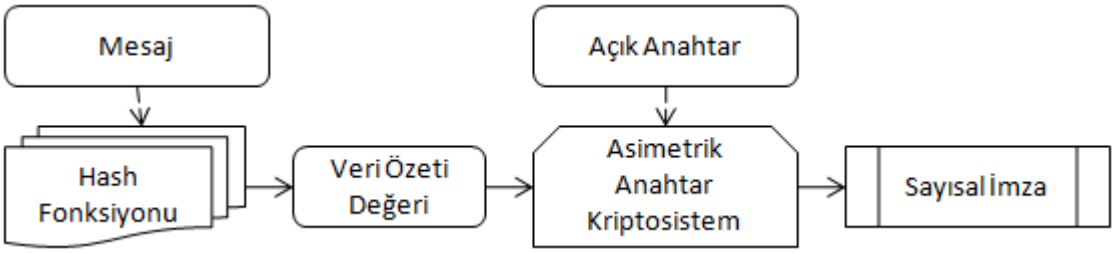
$$S^1 = f_e\big(\mathcal{H}(\mathcal{M}, k_x)\big) \qquad (77)$$

ve mesajın imza doğrulaması

$$S^2 = f\_d\big(S, k_y\big) \qquad (78)$$

olur. Burada $S^1 = S^2$ olduğunda imza kaynağı doğrulanmış olur.

34 SSL (Secure Socket Layer): Güvenli Yuva Katmanı



Şekil 16: Sayısal imza

- **Dosya Bütünlüğü (File Integrity):** Burada herhangi bir kaynaktan gelen / alınan dosyanın iletişim kanalı üzerinde herhangi bir değişime uğramadan elde edilmesi veya mevcut dosyaların değişime uğrayıp uğramadığının doğrulanması sağlanır.

- **Kimlik Doğrulama Protokolleri (Authencation Protocols):** Bu alan iletişim kanalları uçlarındaki tarafların kimlik ispatı amacıyla kullanılır. Burada uç tarafların kendilerine özgül oluşturdukları imza kodları doğrulanarak iletişim güvenliği amaçlanır. Bu uç taraflar şahıslar olabildiği gibi çoğunlukla güvenli ağ katmanları tarafında otomatikleştirilmiş bir yapıda kullanılan ve arka planda çalışan hizmet sistemleridir.

- **Şifre Koruması (Out-of-the-box Passwords):** Kullanıcılara yönelik hizmet sağlayan web, e-posta veya dosya aktarım protokolleri gibi internet sistemleri ve kullanıcı odaklı uygulamalar-yazılımlar tarafında kullanılır. Burada kullanıcı şifreleri olduğu gibi doğrudan kayıt altına alınmaz, bunun yerine şifrenin mesaj özet değeri kayıt

edilir. Teoride mesaj özet değerlerinden mesaja ulaşılması mümkün olmadığından, kullanıcıya ait şifre veritabanının çalınmasına veya sistem yetkililerine karşı korunmuş olur. Aşağıda yer alan Şekil 17: Kullanıcı Bilgilerinde Veri Özetleme Algoritması ile şifreleri mesaj özet değeri ile anlamsızlaştırılmış temsili bir kullanıcı veritabanı gösterilmiştir.

name	lastna	uname	pass_icds	email
Burak	BAYSAN	bbaysan	0b685a93e6353778670e2ba3	burak.baysan@st.uskudar.edu.tr
Serhat	ÖZEKES	sozekes	eb14294201fc7db313f58651	serhat.ozekes@uskudar.edu.tr
Mehmet	KIRAY	mkiray	c80d5bb2a5a4564ba894754c	mehmetkiray@gmail.com
Gülderen	ERTAŞ	ravioli	a1f16684d0f53a2872fc449f	gulderen.erta@gmail.com
Murat	ETİZ	arshe	7d2095fe1c246d0a61124ff5	metiz@9eylul.edu.tr
Harun	KADIOĞLU	fbli	1b288f507e213973bef32ca2	odd-digit@even-digit.com
Aaron	WENDEL	cisco_burger	fc0651786d82cbf754b85857	aaron@wholesaleinternet.net
James	ZELLER	victimofunix	94606e0593108f32700e0e61	james@wholesaleinternet.net
Göktuğ	PEHLİVAN	crazy_driver10	44acd0895c17fcb8bfc0fa0e	goktugce@hotmail.com
Kamil	KOCATEPE	bleach_sniffer	08db17d51636fdcf7ae34af2	kocatepe.37@icloud.com
Erdem	YILDIZ	screwdriver	35cc0a3d8698da242894b85f	erdemyildizz@gmail.com

Şekil 17: Kullanıcı Bilgilerinde Veri Özetleme Algoritması

- **Kötü Amaçlı Yazılım Tespiti (Malware Detection):** Antivirüs yazılımları başta olmak üzere bilgisayar güvenliği ve kötü amaçlı yazılımlarla mücadele eden uygulamalar bu fonksiyonlarca üretilen mesaj özet değerlerini kullanırlar. Burada ilk seçenek kötü amaçlı yazılımların mesaj özet değerlerinden oluşturulan bir veri tabanının her bir dosyanın mesaj özet değeriyle karşılaştırılması yoluyla kötü amaçlı yazılımların tespit edilmesidir. İkinci seçenekse mevcut bütün sistem dosyalarının mesaj özet değerlerinden bir veri tabanı

oluşturulması ve bu dosyalara herhangi bir müdahale olup olmadığının sınanmasıdır[35.]

- **Kaotik Değer Üreteci (Chaotic Value Generator):** Bu yapısı çoğunlukla kriptosistemlerin ihtiyaç duyduğu öngörülemez sayı üretimi için kullanılır. Bir mesaj özetleme algoritmasına giren mesajdaki bir bitlik farklılık bile fonksiyonun üreteceği değerde kaotik farklılıklar üretir. Bu özelliği sadece kriptosistemler için değil rastlantısal sayı üretimi gerektiren matematiksel birçok alanda kullanılır.

4.1. Özellikleri

Mesaj özetleme fonksiyonlarının sahip olması gereken beş temel özellik vardır bunlar sırasıyla;

1. Mesaj uzunluğu herhangi bir değerde olabilir.
2. Özet sabit bir uzunlukta olmalıdır veya sabit uzunluk belirlenebilmelidir.
3. Özet hesaplama işlemi kolayca gerçeklenebilmelidir ve algoritma anlaşılır olmalıdır.
4. Herhangi bir özet değerinden, orijinal mesaja ulaşılamamalıdır36.
5. Özet fonksiyonunun sonlanmış alanında, iki ayrı mesajın aynı özet değeri olmamalıdır[37].

35 Bu yöntem rootkit denilen ve sistem dosyalarının manipülasyonuna dayanan kötücül yazılımların tespitinde kullanılır.

36 Buna tek-yön (one-way) prensibi de denilmektedir.

37 Buna çakışmasızlık (collision-free) prensibi de denilmektedir

Mesaj özetleme fonksiyonlarının bu temel özellikleri dışığında detaylandırılmış diğer özellikleri alt başlıklar ahalinde verilmiştir.

4.1.1. Öngörüntü Direnci (Preimage Resistance)

$\mathcal{M}_1$ ve $\mathcal{M}_2$ mesaj, h_1 ve h_2 mesaj özet değerleri ve $\mathcal{H}$ anahtarsız mesaj özet değeri olduğunda önceden belirlenen bütün mesaj özet değerleri için aynı değeri veren bir $\mathcal{M}$ mesajı hesaplanabilir olmamalıdır (herhangi bir $\mathcal{M}'$ değeri bulmak kolay olmamalıdır).

4.1.2. İkinci Öngörüntü Direnci (2nd Preimage Resistance)

$\mathcal{M}_1$ mesaj, h_1 mesaj özet değeri ve $\mathcal{H}$ anahtarsız mesaj özet fonksiyonu olduğunda $\mathcal{H}(\mathcal{M}_1) = h_1$ değeri için $\mathcal{M}_1$ ile aynı h_1 değerini veren ikinci bir $\mathcal{M}_2$ bulmak kolay olmamalıdır. Burada $\mathcal{M}_1 \neq \mathcal{M}_2$ koşulundayken, $\mathcal{H}(\mathcal{M}_1) = \mathcal{H}(\mathcal{M}_2)$ koşuluna karşı direnç sağlanması gerekir.

4.1.3. Çakışma Direnci (Collision Resistance)

Çakışma direnci iki farklı türde ele alınır. İlki $\mathcal{H}_1$ mesaj özet değerinin bütünü olarak ele alınır. Burada aynı $\mathcal{H}_1$ mesaj özeti değerini veren iki farklı $\mathcal{M}_1$ ve $\mathcal{M}_2$ mesajın bulunması mümkün olmamalıdır. Bu durum $\mathcal{H}(m_1) = \mathcal{H}(m_2)$ olacak iki ayrı değerde $\mathcal{M}$ değerinin bulunamaması olarak açıklanır.

İkincisi mesaj özet değerini oluşturan bitlerin aynı konuma ait olanları olarak ele alınır. Burada bilinen en düşük 128-bit uzunluğundaki bir mesaj özetinde iki farklı $\mathcal{M}_1$ ve $\mathcal{M}_2$ mesajında çakışan bit değerleri olması beklenen bir sonuçtur. Ancak çakışan bit

değerleri sayısının az olması amaçlanan sonuçtur. Bu çakışmalarda karşılaştırılması yapılacak ölçüm değerleri 1-bit, 4-bit (hexadecimal karakter) veya 8-bit (byte) olarak ele alınabilmektedir.

Mesaj özetleme fonksiyonlarının çakışma direncine dair istatistiksel ölçümleri birbirinden sadece bir bitlik farklılık gösteren iki ayrı $\mathcal{M}_1$ ve $\mathcal{M}_2$ mesajlarından elde edilen mesaj özeti değerlerinin çakışan bitlerinin sayımı ile yapılır. Burada ölçümler için çeşitli kombinasyonlar üretilir, aksi durumda bilinen en düşük 128-bitlik uzunluğa sahip mesaj özetleme algoritması için 2^{128} ayrı $\mathcal{M}$ değerinin denenerek çakışmanın ölçümü gerekecektir.

Burada $\mathcal{M}_1 \neq \mathcal{M}_2$ koşulundayken, $\mathcal{H}(\mathcal{M}_1) = \mathcal{H}(\mathcal{M}_2)$ koşuluna karşı direnç sağlanması gerekir.

4.1.4. Hızlı Hesaplama (Rapid Compute)

Bir mesaj özetleme algoritması $\mathcal{M}$ mesajı için $\mathcal{H}$ mesaj özeti değerini hızlı bir şekilde hesaplayabilmelidir. Bu büyük boyutlu mesajlar için veya yoğun iş-görev yüklü sistemlerin yanıt süreleri için gereklidir. Yanıt süresinin gecikmesi aynı zamanda işlemci-iş gücü maliyetine de sebep olur.

4.1.5. Uygulanabilirlik (Implementation)

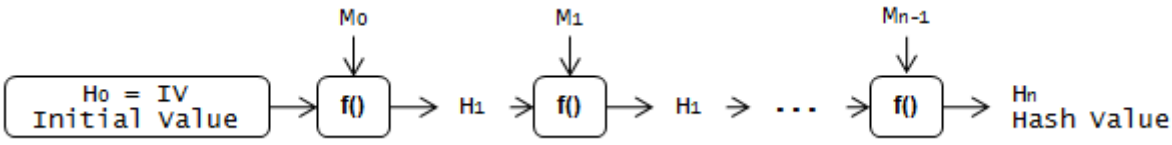
Bir mesaj özetleme algoritması donanımsal veya yazılımsal olarak kolayca uygulanabilir ve anlaşılabilir algoritmik yapıya sahip olmalıdır. Kısa ve anlaşılır yapısı güvenliğine zarar vermemeli ve direnç gereksinimlerini karşılayabilmelidir.

4.1.6. Kaotik Değerler (Chaotic Values)

$\mathcal{M}$ mesaj verisinin uzunluğundan bağımsız olarak mesaj içerisindeki en küçük boyutlu bir farklılığın $\mathcal{H}$ mesaj özet değerinde kaotik farklılıklara yol açması gerekir. Bu kaotik farklılıklar $\mathcal{M}$ mesajının sadece değer farklılığında değil, uzunluğunda veya ikilik, sekizlik, onaltılık veya 2^n katları olan diğer sayı sistemlerindeki basamak değerlerinin konumsal farklılığında da aynı kaotik değerli sonuç özelliğini vermelidir.

4.2. Merkle-Damgard

En çok bilinen SHA ve MD ailelerinin de üzerine kurulu olduğu yapı Merkle-Damgard modelidir. Bu modelde bir sıkıştırma fonksiyonu kullanılır ve iteratif olarak çalıştırılır.



Şekil 18: Merkle-Damgard Hash Modeli

Burada $\mathcal{H}:\{0,1\}^* \rightarrow \{0,1\}^n$ Merkle-Damgard yapısı üzerine kurulu bir özetleme algoritması olduğunda, $\mathcal{H}$ bir $\mathcal{M}$ mesajını b uzunluğunda $\mathcal{M}_0, \mathcal{M}_1, \mathcal{M}_{...}, \mathcal{M}_{n-1}$ bloklarına yerleştirerek alır ve $\mathcal{H}(\mathcal{M})$ mesaj özet değerini iteratif olarak çalıştıracağı f fonksiyonu ile hesaplar. f sıkıştırma fonksiyonu i adımında n-bit $\mathcal{H}_{i-1}$ özet bit dizisi ve b-bit m_{i-1} mesaj bit dizisi alır ve sonraki adım n-bit $\mathcal{H}_i$ bit dizisi verir (Şekil 18: Merkle-Damgard Hash Modeli). Burada ilk değer $\mathcal{H}_0$ başlangıç değeridir ve buna IV (Initial value) denir.

Bu yapı aşağıdaki fonksiyon ile temsil edilebilir ve bu açık bir iterasyonu işaret eder;

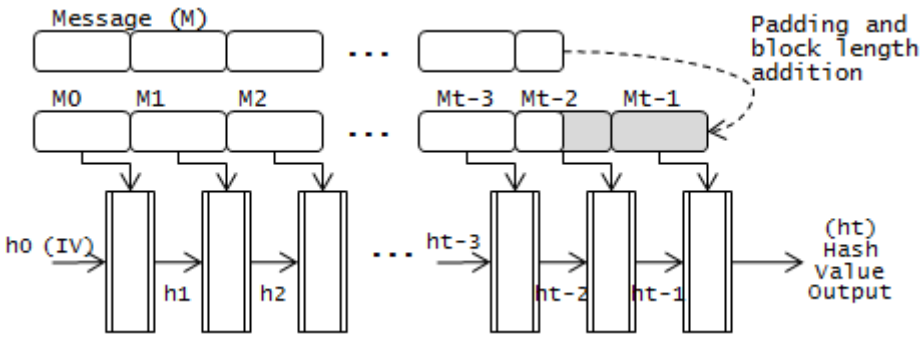
$$\mathcal{H}_i \coloneqq \begin{cases} IV, & for\ i = 0, \\ f(\mathcal{H}_{i-1}, \mathcal{M}_{i-1}) & for\ \text{İ} = 1, \dots, n-1 \end{cases} \quad (79)$$

Ayrıca $f: \{0,1\}^n \times \{0,1\}^b \to \{0,1\}^n$ bir sıkıştırma fonksiyonu olarak bu yapıda temsil edilir. Genel olarak f fonksiyonu, mesaj bloklarını alt adımlar içerisinde döngülerle çağırılan alt fonksiyonlarda sıkıştırır.

f_i tur fonksiyonu olmak üzere $\mathcal{M}_i$ bloğunun bitleri ile $\mathcal{H}_i$ bu fonksiyonun turları ile karıştırılır. Bu fonksiyonun her bir turunda:

- **Lineer yapıda**; XOR işlemlerini, bit rotasyonlarını, permütasyonları veya özgül matrisleri kullanır.
- **Linear olmayan yapıda**; AND işlemlerini veya S-Box tablolarını kullanır.

✓ **Teorem 1:** Eğer $h(x)$ çakışma dirençli bir sıkıştırma fonksiyonuysa $\mathcal{H}(x)$ mesaj özetleme algoritması çakışma dirençlidir.



Şekil 19: Merkle-Damgard Yapısı (Güçlendirme)

✓ **İspat 1:** Bu teoreme ait ispat Magnus DAUM tarafından yapılan çalışmada yer almaktadır. Bu çalışmaya kaynakçada (30). sırada yer verilmiştir.

Teoremin güvenlik tarafına getirdiği yaklaşım sonucu, çakışma dirençli sıkıştırma fonksiyonlarının popülerliği artmıştır. Günümüz mesaj özetleme fonksiyonlarının birçoğu bu yapı üzerine kurulmuştur.

4.3. HAIFA: Hash Iterative Framework

HAIFA, Biham ve Dukelman tarafından Merkle-Damgard şemasının güçlendirilmesi ve genelleştirilmesi için önerilmiştir (31). HAIFA, Merkle-Damgard yapısının bütün iyi özelliklerini almayı, bu özellikleri geliştirerek güvenlik özellikleri eklemeyi ve değişken mesaj özeti değerlerinde kullanmayı amaçlar.

Merkle-Damgard yapısı, sıkıştırma fonksiyonu n boyutunda değerlerin zincirleme aktarılmasını / devrini kullanır. HAIFA ise, zincirleme olarak aktarılan bu blok değerlerine, b bit sayacını ve s mesaj veya blok boyunu dâhil eder.

Burada HAIFA aşağıdaki şekilde tanımlanır;

$$f: [0,1]^n \times [0,1]^n \times [0,1]^b \times [0,1]^s \rightarrow [0,1]^n \qquad (80)$$

bh mesaj özeti alınan bit sayısını ve s salt boyut değerini temsil ettiğinde;

$$h_i = f(h_{i-1}, m_i bh_i, s_i) \qquad (81)$$

olur.

Bir mesaj özeti hesaplanmasında HAIFA'ya ait ilk üç ana adım ve bir seçenek aşağıda verilmiştir;

1. Mesaj dolgulama.
2. Başlangıç değeri (IV) hesaplama.
3. *f* sıkıştırma fonksiyonu iterasyonu.
4. Zincirdeki son değerin budanması.

4.4. Yapılar

Önceki bölümlerde veri özetleme fonksiyonlarının temelleri ele alınmıştır. Bu bölümde bu fonksiyonları oluşturan yapılar ve özellikleri açıklanmıştır.

Bugün veri özetleme algoritmaları için önerilmiş birçok yapı bulunmaktadır. Bu yapıların hepsinde ortak özellik çakışmaya karşı dirençli olma iddiasıdır.

Bu yapıların büyük bir bölümü tamamen matematik problemlerine, bir bölümü en çok bilinen kriptografi modellerine, bir bölümüyse kaotik fonksiyonlara dayanırken bu yapıların tamamen dışığında farklılaşmış fonksiyonlarda yer alır.

Bu bölümde en çok bilinen yapılar ele alınarak detayları açıklanmıştır. Bunlar blok şifre temelli (Block Cipher), Sünger fonksiyon temelli (Sponge Function Based) ve Provably Secure yapılar olarak üç gruptan oluşmaktadır.

4.4.1. Blok Şifre Temelli (Block Cipher Based)

2000 yılı sonrasında veri şifreleme fonksiyonları üzerine yapılan çalışmaların birçoğu blok şifre temeli üzerine kurulmuştur. Ancak blok şifre temelinin veri özetleme fonksiyonlarında kullanılmasında bir sorun vardır. Bu sorun blok şifrenin tek yönlü

(Bijectivenes) olmamasıdır. Bu sorunun aşılması için blok şifre yapılarına ek işlemler ve fonksiyonlar eklenerek hesaplamanın başında veya sonunda çağırılmaktadır.

Blok şifre temelinde özetleme algoritmalarının geliştirilmesinde bir kısım önemli gerekçeler öne sürülmektedir. Bunların en başında bu yöntemin hesaplama yeteneğinin kullanılmasının amaçlanması, düşük donanım ihtiyacıyla yüksek performanslı hesaplama yeteneğine ulaşılması gelmektedir. Ancak bu tasarımların yetenekleri üzerine kurulu veri özetleme algoritmaları, bu yeteneklerle birlikte başka bir şeyi de almaktadır, o da güvenlik durumudur. Bugün en çok bilinen ve güvenilir sayılarak yapıları örnek alınan mesaj özetleme algoritmaları, güvenlikleri en büyük zaafa uğramış olanlardır.

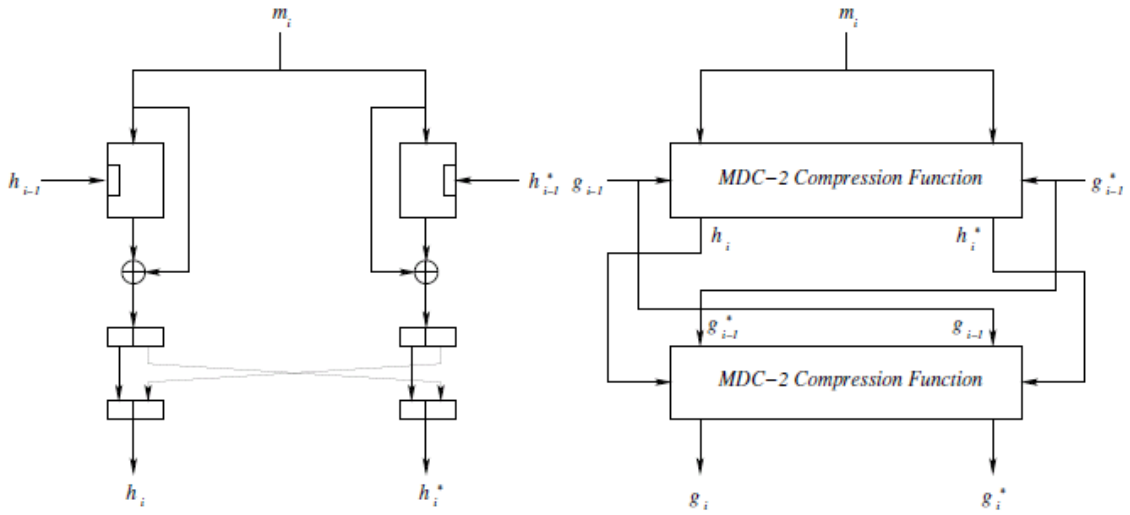
Blok şifreleme temelinde bir sıkıştırma fonksiyonu kullanılır. Bu sıkıştırma fonksiyonu iteratif olarak mesaj özetini hesaplayarak bir sonraki tura aktarır. Merkle ve Damgard sıkıştırma fonksiyonunun çakışma dirençli olması durumunda, mesaj özetleme algoritmasının da çakışma dirençli olduğunu göstermesi sonrasında blok şifre temeli daha sık başvurulan yapı durumuna gelmiştir. Bugün çoğu mesaj özetleme algoritmasında kullanılan ve standartlaşan mesaj özetleme algoritmalarında bu yapı kullanılmıştır ve çok benzer şifreleme algoritmaları kullanırlar.

Ayrıca blok şifre yapılı mesaj özetleme algoritmaları mesaj özet değeri uzunluğunun tek blok uzunluğuna, çiftli blok uzunluğuna veya farklı sayıdaki blok uzunluğuna bağlı olmasına göre gruplara sahiptir.

Küçük uzunlukta mesaj özetleme algoritmaları ile tek uzunluk, en çok tercih edilen seviyelerdir. Büyük uzunlukta mesaj özetleme algoritmaları güvenlik tarafında daha etkinken, performans-hız açısından verimli değildir. Bir kısım çift blok uzunluklu mesaj özetleme algoritmaları performans sorununa çözüm getirmek üzere tasarlanmıştır

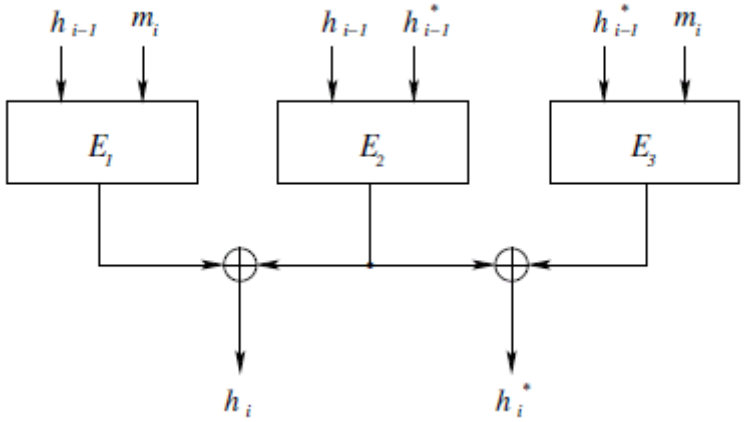
ancak bu uygulayıcılar tarafında tercih edilmemiş ve soruna etkin bir çözüm getirmemiştir.

- Tek Blok Uzunluklu Mesaj Özeti: Bilinen bütün tek blok uzunluklu tasarımlar rate-1 olarak adlandırılır. Bu tasarımlar sırasıyla verilmiştir:
  - Davies-Meyer
  - Matyas-Meyer-Oseas (32)
  - Miyaguichi-Prencel
- Çift Blok Uzunluklu Mesaj Özeti: Eğer blok şifre temelli mesaj özetleme algoritması çift blok uzunluğuna eşitse, bu fonksiyona çift blok uzunluklu özetleme denir. Bu yapıda en çok bilinenler MDC-2 (33) ve MDC-4 (33) olarak literatürde yer almaktadır. Bu algoritmaların çift uzunluklu özet üretimi en çok bilinen kriptosistemlerden DES gibidir. ½ ve ¼ uzunluklu bu algoritmalar Şekil 20 (27) ile verilmiştir.



Şekil 20: Çift Uzunluklu Mesaj Özetleme

Bir başka örnek olan Nandi ve diğerlerinin çift uzunluklu ve 2/3 seviyeli çalışması Şekil 21 (27) ile verilmiştir. Bu yapıya ek olarak Abreast-DM (34), Hirose ailesi (35), Parallel-DM (36) çalışmaları literatürdeki yerlerini almıştır.



Şekil 21: Çift Uzunluklu Mesaj Özetleme

4.4.2. Sünger Fonksiyonlar Temelli (Sponge Based)

Sünger (Sponge) fonksiyonları Bertoni ve diğerleri tarafından tanımlanmıştır (37). Bu fonksiyonlar sonlu durum iterasyonudur. Bir sünger fonksiyon girdi olan bir değişken-uzunluk ile sonsuz-uzunlukta olası çıkış üretebilir. Süngerler veri özetleme algoritmaları ve akan kriptosistemlerin ikisinde de kullanılabilir özelliklere sahiptirler.

4.4.3. Akan Şifre Temelli

Daha etkili mesaj özetleme fonksiyonlarına dair bazı çalışmalar yeni özetleme tekniklerini de getirmiştir. Bu yapı metodu matematiksel problemler veya bir blok şifre yöntemi temelinde değildir. Bu yapı akan şifre yapısının senkronizasyonuyla ilgilidir. Burada simetrik şifre yönteminin önemli bölümleri kullanılarak yüksek hızlı hesaplama

amacıyla bir yapı oluşturulur. Akan şifre yapısının düşük donanım ihtiyacı aynı zamanda yüksek verimliliği de sağlamaktadır.

Bu yapıda çalışan ve en çok bilinen mesaj özetleme algoritması Panama'dır (38). Panama Daemen ve diğerleri tarafından geliştirilmiştir ancak bu fonksiyonun çakışmaları bulunmuştur ve güvenilmez olarak kabul edilir (39).

Panamadan sonra daha gelişmiş olan Radiogatun 2007 yılında duyurulmuştur (40). Bu algoritma RC4 (41) kriptosistemi temelinde kurulmuş ancak Indesteege ve Areneel tarafından kırılarak bu çalışmada güvenilmez duruma gelmiştir (42).

Akan şifre temelli özetleme algoritmaları performans ihtiyacı tarafında çözüm üretebilmekteyken, güvenlik tarafında herhangi bir matematiksel veya teorik ispat sunamamışlardır.

4.5. En Çok Bilinen Mesaj Özetleme Algoritmaları

Bu bölümde literatürde yer alan ve en çok bilinen mesaj özetleme algoritmalarının incelemeleri ve alt bölüm başlıkları halinde verilmiştir.

4.5.1. SHA Ailesi

Bu ailenin ilk üyesi SHA-0 algoritmasıdır. 1993 yılında NIST tarafından yayınlanmış ve standart olarak duyurulmuştur. Ancak sha-0 için bulunan bir zafiyet sebebiyle bu algoritmaya sadece bit kaydırma fonksiyonu eklenerek SHA-1 adıyla güncellemesi duyurulmuştur.

SHA-1 bugün en çok bilinen mesaj özetleme algoritmasıdır ancak 2005 yılında bu algoritmada bir zafiyet bulunması üzerine bu yapı da değişerek SHA-2 adıyla bilinen yeni alt aile ortaya çıkmıştır (43-45).

SHA-1 isimli veri özetleme fonksiyon 1995 yılında Amerika Birleşik Devletlerinin NSA adlı kurumu tarafından tasarlanmıştır. Bu çalışma FIPS PUB 180-4 kodu ile NIST tarafından standartlaştırılarak yayınlanmıştır.

SHA-1 en fazla $2^{64} - 1$ bit uzunlukta mesajların veri özetini hesaplayabilmekte ve 160-bit uzunukta veri özet değeri vermektedir. Burada SHA-1'in kullandığı blok boyutu 512 bit olup Merkle-Damgard yapısı üzerine kurulu iteratif blok şifre yapısını kullanır.

SHA-1 ailesi veri özetlemesini üç adımda gerçekleştirir. İlk adımda M mesajına bit dizisi dolgulama işlemi gerçekleştirilerek dolgulanmış M mesajı elde edilir. Dolgulama işlemi sonrasında M' mesajı 512 bit eş uzunluklu parçalara bölünür. İkinci adımda özet değeri için 160-bitlik başlangıç değerleri/durumu (IV – Initial Values) atanır. Üçüncü adımda sıkıştırma fonksiyonu f() 512-bitlik M değerlerine göre 160-bitlik H değerini hesaplar. Burada *for i = 0, 1…, n-1* olmak üzere her tur sonunda elde edilen *H* değeri bir sonraki tura devredilir. *f()* fonksiyonu toplam 80 alt çevirime sahiptir. Bütün *M* değerleri bu *f()* sıkıştırma fonksiyonundan geçirildikten sonra elde edilen H değeri SHA-1 için mesaj özeti çıkışını oluşturur.

SHA-2, 224, 256, 384 ve 512-bit uzunlukta mesaj özetleri üretebilmek için SHA-1 üzerinde yapılan düzenlemeler ile geliştirilmiştir. SHA-2'de 384 ve 512 bit uzunluklu değerlerin blok boyutları 1024 bit olarak genişletilmiştir.

4.5.2. MD Ailesi

MD (Message Digest) ailesi Ron RIVEST (20) tarafından geliştirilmiştir. Bu fonksiyonun bilinen dört ayrı türü vardır bunlar MD2, MD4 (46), MD5 ve NIST yarışması için tasarlanan MD6'dır (21).

MD2 1982 yılında açıklanmıştır. 128-bit özet uzunluğuna sahip olup çakışmalar üretilmesinin yöntemi bulunmuştur. Bugün bu algoritma kullanılmamaktadır.

MD4 1990 yılında açıklanmıştır. 128-bit özet uzunluğuna ve 512 bit blok boyutuna sahiptir özet hesaplaması blok yapısının iterasyonu ile üç ayrı iç döngü fonksiyonunda yapılır. burada aynı özet değerine sahip olan iki ayrı mesaj bulmak için 2^{64}, belirli bir özet değerini veren bir mesaj bulmak için 2^{128} olası değerin sınanması gerekir. Bu algoritma için 2^{20} sıkıştırma fonksiyonunun çağırılması sonucu çakışma üretilmesinin yolu bulunmuş ve güvenilmez olarak kabul edilmiştir.

MD5, MD4'ün geliştirilmiş halidir. Temel yapısı MD4 ile aynı olmasına karşı MD4'den daha karışık hesaplama yapar. Algoritmada çakışmaların nasıl hesaplanacağını detaylarıyla veren çalışmalar literatüre girmiştir. Güvenilmezliği bilimsel olarak ispat edilmiş olmasına karşın halen pek çok uygulamada kullanılmaktadır. Algoritma 512-bitlik bloklarda, 32-bit kelime değeri ve en fazla 2^{64} uzunlukta mesaj ile 128-bit uzunluğunda mesaj özeti üretir.

4.5.3. RIPE-MD

Bu mesaj özetleme algoritması RIPE.NET için tasarlanmıştır. Yapısı MD4 algoritması temeli üzerine geliştirme yapılması ve diferansiyel saldırılara dayanıklılık amaçlı hazırlanmış olup 128-bit uzunlukta özet değeri üretmektedir. Geliştirme MD4'ün tur fonksiyonlarının ve kelime sıralamasının değiştirilmesini de kapsamıştır. Ayrıca tur fonksiyonu sayısı ikiye yükseltilmiş olup, her iki fonksiyona da farklı sabitler verilerek her blok hesaplaması sonunda iki fonksiyonun ürettiği değerler birbirlerinin iteratif değişkenlerine eklenir.

4.5.4. Haval

Haval değişken uzunlukta mesaj özet değerleri üretebilen ve parametrik fonksiyon yapısına sahip bir mesaj özetleme algoritmasıdır. MD5'in yapısı temelinde tasarlanan HAVAL, MD5'in 512 bitlik blok uzunluğunu 1024-bit olarak arttırmış ve yine MD5'in 4 adet 32-bitlik zincir değişkeni sayısı 8'e yükseltilmiştir. Haval, 128, 160, 192, 224 ve 256 bit uzunluklarda mesaj özeti değerleri üretebilmektedir. Bu değerlerin üretilmesinde non-lineer yedi değişken bir sonrakini etkileyerek yüksek etki oluşturan oluşturulması amaçlanmıştır. Fonksiyon iki temel döngüye sahip olup, değişken sayıda döngü ve değişken uzunlukta mesaj özetleme değeri seçenekleri, Havalın 15 ayrı yapıda çalıştırılabilmesini sağlar.

5. KRİPTOGRAFİK SALDIRILAR

5.1. Giriş

Kriptografik saldırılarda amaç kriptografi sistemlerindeki zaafların bulunmasıdır. Bu uygulama sisteme karşı yıkıcı bir amaçla olabileceği gibi sistemin geliştirilmesini ve güvenirliğini ölçme amaçlı da olabilmektedir. Bu yöntemler kriptografik metot araştırmacıları ve geliştiricileri tarafından, önerecekleri sistemlerin – yöntemlerin güvenlik seviyelerinin ölçüm ve ispatlarında da kullanılır.

İkinci Dünya Savaşı sırasında Bell'de görevli olan Claude SHANNON, blok kriptosistemlerin bir çeşidi olan tek zamanlı sistemlerin kırılamaz olduğuna dair ispatını sunmuştur. Bu ispatında anahtarın tek kullanımlık olması ve blok boyunun düz metin ile eş ya da daha büyük boyutlu olduğu durumu varsayılmıştır.

Bugün için çoğu şifreli mesaj deneme yanılma yöntemi ile çözülebilmektedir. Ancak burada sonuca ulaşmak için sarf edilen zaman ve güç maliyeti, sonuçtan elde edilecek faydadan daha fazla olmamalıdır. Nitekim düz metinden elde edilecek bilgi, çözümleme sonuçlanmadan değerini kaybedecekse veya bilgi için sarf edilecek güç maliyeti bilgiden daha kıymetliyle bir çözümün getirmesi amaçlanan fayda elde edilemeyecektir.

5.2. Saldırı metotları

Kriptografik Bugün kriptoanaliz birçok farklı alt alana sahip olup literatürde birçok metot yer almaktadır. Ancak yaygın olarak genel bir sınıflandırmada, analistin elinde olan bilgi türü kullanılmaktadır. Bu saldırı türleri aşağıda verilmiştir.

1. **Sadece şifreli metin saldırısı – Ciphertext only attack:** Kriptanalistin elinde sadece şifreli metinler vardır. Ancak günümüz kriptosistemleri bu

tür saldırılara karşı dirençlidir. Özellikle düz metin ve / veya anahtar üzerindeki etkilerin şifreli metne yüksek dağınım ve karışıklık sağlayan[38] kriptosistemler karşısında sadece şifreli metin saldırısı etkili sonuçlar vermeyecektir.

2. **Bilinen düz metin saldırısı – Known plaintext attack:** Kriptanalist hem şifreli metne hem de şifreli metnin elde edildiği düz metne erişebilmektedir.
3. **Seçilmiş düz metin saldırısı – Chosen plaintext attack:** Kriptanalist seçeceği düz metinden, şifreli metin elde edebilmektedir.
4. **Seçilmiş şifreli metin – Chosen ciphertext attack:** Kriptanalist seçeceği şifreli metinlerden, açık metni elde edebilmektedir.

5.3. Saldırı amaçları

Kriptografik Bugün Kriptanaliz saldırılarının belirlenmiş amaçları olur ve saldırı yöntemi bu amaca göre seçilir, bu amaçlar aşağıda verilmiştir.

1. **Ayırt etme saldırıları – Distinguishing attacks:** Kriptosistem çıkışının rastlantısal bir permütasyon çıkışından ayırt edilebilmesini amaçlar.

2. **Kısmi düz metin bilgisi – Partial knowledge of the plaintext:** Düz metnin bir kısmının elde edilebilmesini amaçlar.

3. **Şifre çözme – Decryption:** Şifrelenmiş metnin en az bir bölümünün şifresinin çözülebilmesini amaçlar.

38 Dağınım / Karışıklık: Literatürde bu etki Diffusion / Confusion olarak tanımlanır.

4. **Şifreleme – Encryption:** Anahtarın bilinmedi durumlarda, herhangi bir şifreli metnin anahtarıyla bir başka düz metnin şifrelenmesindeki sonucun elde edilebilmesini amaçlar.

5. **Kısmı anahtar kurtarma – Partial key recovery:** Anahtarın bir kısmının elde edilebilmesini amaçlar. Burada anahtarın bir kısmının elde edilmesi çoğu durumda anahtarın diğer bölümlerinin elde edilmesinde bir adım olarak kullanılır.

6. **Tam anahtar kurtarma – Total key recovery:** Bu durum bir kriptosistem için en kötü senaryoyu açıklar. Burada kullanılan anahtarın tamamen bulunabilmesi ve kriptosistemin etkisizleştirilmesi amaçlanır.

5.4. Kriptanaliz yöntemleri

Kriptanaliz yöntemleri her ne kadar bir bilgisayar yardımı ile kimi istatistiksel sonuçları elde etme zorunluluğunda olsa da bunların yorumlanması analizcinin yeteneklerine bağlıdır. Analistin bir kriptosistemi çözebildiğini duyurması bu sistemin değiştirilmesi sonucunu getirir. Bu değişim DES örneğinde olduğu gibi blok boyunun genişletilmesi veya fonksiyon işlevlerinin değişikliği şeklinde olabilmektedir.

Kriptanaliz yöntemleri aşağıda verilmiştir:

1. **Etraflı arama – Exhaustive search:** Kriptosistemlere yönelik en açık ve doğrudan uygulanabilen bir yöntemdir. Olası bütün anahtarlar kısa düz metin ve şifreli metin örneklerinde denenir.

Doğru anahtarın bulunması burada seçili düz metnin, hedef şifreli metne dönüşümünü ve seçili şifreli metnin düz metne dönüşümünü sağlar. Bugün AES ve benzeri kriptosistemler 128-bit ve üzerinde anahtar uzunlukları sağlayarak bu saldırı türüne karşı direnç sağlamayı amaçlarlar. DES'in 56-bitlik kısa sayılabilecek anahtar uzunluğu bu tür saldırıya karşı en önemli zaafını oluşturmaktadır.

2. **Sözlük saldırısı – Dictionary attack:** Bu yöntem basit ancak blok kriptosistemlere karşı etkin bir saldırı yöntemidir. Burada şifreli metinlerin kısa olması durumunda bunlar toplanarak tekrarlanma oranları ölçülür. Bu atağa karşı en zayıf yapı değiştirmeli kriptosistemdir[39].

3. **Eşitlik tanımlama – Equivalent description:** Kriptosistem algoritmasının eş değerler üretmesi ve bu çıkışı vermesi diferansiyel saldırılarına hedef olmaktadır.

4. **Değişmezler için devirlilik veya arama – Preiodicity or search for invariables:** Değişmezler kriptosistemin hesaplama sürecinde sabit kalan yapı ve değerleridir. Bunlar istenilmeyen özelliklerdir. Kriptanalist bu değişmezlerden herhangi birisini tespit edebilirse diferansiyel saldırı için değer tespit etmiş olur. Sadece değişmezlerin

39 Değiştirmeli Kriptosistem: Substitution Cipher

değil farklı düz metin ve / veya şifreli metin arasındaki korelâsyonda istenilmeyen ve engellenmesi gereken bir sonucu doğurur.

5. **Ortada buluşma saldırısı – Meet-in-the-middle attack:** Kriptosistemin iki eşit parça olarak ele alınmasını ve iki uçtan orta noktaya kadar şifre çözme işlemi yapılmasını temsil eder. Ortada buluşma sonucunda elde edilen sonuçlar uyumluysa anahtar bulunmuştur.

6. **İstatistiksel yaklaşımlar – Statistical approaches:** Düz ve şifreli metinler arasındaki ilişkilerin arandığı istatistiksel yöntemi temsil eder. Aslında burada yapılan, uygulanacak diğer atakların seçimi için ön hazırlıktır.

7. **Özel bir atakla ilgili zayıf anahtarlar –Weak keys with respect to a particular attack:** Kriptosistemin herhangi bir anahtar için, aynı sonucu veren daha kısa veya uzun anahtarlara karşı dirençsizliğini ifade eder. DES zayıflığını da oluşturan bu durum, DES'te dört zayıf ve on iki kısmi-zayıf anahtar olmasına yol açar.
    - ❖ **Örnek:** Anahtar $\mathcal{K}$ ve şifreleme $\mathcal{E}_{\mathcal{K}}$ olduğunda dört zayıf anahtar $\mathcal{E}_{\mathcal{K}}\left(\mathcal{E}_{\mathcal{K}}(m)\right) = m$ ve on iki kısmi zayıf anahtar $\mathcal{E}_{\mathcal{K}1}\left(\mathcal{E}_{\mathcal{K}2}(m)\right) = m$ eşitliğine sebep olur.
    - ❖ **Örnek:** IDEA kriptosistemi 2^{128} anahtar uzayında 2^{63} adet zayıf anahtara sahiptir.

5.5. Sadece şifreli metin saldırısı

Beker ve Piper'ın yaptıkları bir çalışma ile İngilizcede en çok kullanılan tekli, ikili ve üçlü harf dizilimlerini belirlenmişlerdir (47)[40]. Bu çalışma ile belirlenen 26 İngiliz alfabesi harfinin kullanım olasılığı aşağıda yer alan Tablo XV**.** 26 İngiliz harfinin kullanım olasılıkları ile verilmiştir.

Tablo XV. 26 İngiliz harfinin kullanım olasılıkları

Harf	**Olasılık**	**Harf**	**Olasılık**
a	0.082	m	0.067
b	0.015	o	0.075
c	0.028	p	0.019
d	0.043	q	0.001
e	0.127	r	0.060
f	0.022	s	0.063
g	0.020	t	0.091
h	0.061	u	0.028
i	0.070	v	0.010
j	0.002	w	0.023
k	0.008	x	0.001
l	0.040	y	0.020
m	0.024	z	0.001

Bu tabloda yer alan harf kullanım olasılıkları beş ayrı gruba dağıtıldığında aşağıda verilen durum oluşur.

1. 0,120 olasılığı olan; e
2. 0,06 ile 0,09 aralığında olasılığı olanlar; t, a, o, i, n, s, h, ve r

40 Beker ve Piper'ın bu çalışmalarından elde ettikleri sonuçlar başvuru kaynağı olarak kullanılmıştır.

3. 0,40 ile 0,043 aralığında olasılığı olanlar; d ve l
4. 0,015 ile 0,023 aralığında olasılığı olanlar; c, u, m, w, f, g, y, p ve b
5. 0,001 ile 0,010 aralığında olasılığı olanlar; v, k, j, x, q ve z

Bunlarla birlikte Beker ve Piper en çok ikili ve üçlü harf dizilimlerini aşağıda verilen şekilde açıklamışlardır (47).

1. En çok kullanılan ikili harf dizilimleri; AN, AR, AS, AT, EA, ED, EN, ER, ES, ET, HA, HE, HI, IN, IS, IT, ND, NG, NT, OF, ON, OR, OU, RE, SE, ST, TE, TH, TI ve TO
2. En çok kullanılan üçlü harf dizilimleri; AND, DTH, ENT, ERE, ETH, FOR, HER, ING, NTH, THA, THE ve WAS

❖ **Örnek:** Burada sadece şifreli metin saldırısı, affine kriptosisteme yönelik bir saldırı örneği örnek ile ele alınacaktır. Affine kriptosistemiyle elde edilen şifreli metin aşağıda verilen şekilde olduğu varsayılsın:

dtfdwhoduvibotvyxvut

yvydybzghwwvryvzuhuw

gzobutvrtrvburbtdtbv

yzomztbvytbrobyfbxvw

Bu şifreli metindeki harf kullanım frekansları aşağıda yer alan Tablo XVI**.** Şifreli metin harflerinin frekansları ile verildiği değerlerdedir.

Tablo XVI. Şifreli metin harflerinin frekansları

Harf	Frekans	Harf	Frekans
a	0	m	0
b	10	o	5
c	0	p	0
d	5	q	0
e	0	r	5
f	2	s	0
g	2	t	9
h	3	u	6
i	1	v	11
j	0	w	5
k	0	x	2
l	0	y	7
m	1	z	5

Bu frekans değerleri ve şifreli metinden şu bilgiler elde edilebilmektedir:

1. Şifreli metin 80 karakterden oluşmaktadır.
2. Şifreli metinde, sırasıyla en çok kullanılan karakterler ve olasılık değerleri aşağıdaki değerlerdedir:

- v ; 11 defa (0,137)
- b ; 10 defa (0,125)
- t ; 9 defa (0,112)

Burada Tablo XII**.** İngiliz alfabesinin mod 26'ya göre karşılıkları ve Tablo XVI**.** Şifreli metin harflerinin frekansları ile verilen şifreli metne ait istatistikî bilgiler karşılaştırılarak anahtara ulaşılmaya çalışılır.

1. Tablo XII'ye göre İngiliz alfabesinde en çok kullanılan harfler 0,127 olasılıkla e ve 0,91 olasılıkla t olmaktadır
2. frekanslarına göre şifreli metinde en çok kullanılan harfler 0,137 olasılıkla *v* ve 0,125 olasılıkla *b* harfleri olmuştur.
3. İstatistikî sonuçlar doğrultusunda önce şifreli metindeki *v* harfinin *e* harfine, *b* harfinin *t* harfine denk geldiği varsayılacaktır.

a. frekanslarına göre *v=21, e=4, b=1 ve t=19* ile temsil edilir. Bu durumda $\mathcal{K} = \{(a,b) \in \mathbb{Z}_{26}^{+} \times \mathbb{Z}_{26}^{+} : (a,26) = 1\}$ olmak üzere $\mathcal{E}_{\mathcal{K}}(ax+b)$ denklemi aşağıdaki iki bilinmeyenli eşitlik olarak ortaya çıkar:

$$4a + b \equiv 21 \pmod{26}$$
$$19a + b \equiv 1 \pmod{26}$$
$$4a + b \equiv 21 \pmod{26}$$
$$4a^{-1} - 26b \equiv 21 \pmod{26}$$
$$26 \equiv 5.1 + 21 \pmod{26}$$
$$5 \equiv 5.21 + 4 \pmod{26}$$
$$1 \equiv 5.21 \pmod{26}$$
$$21 \equiv 26 - 21.1 \pmod{26}$$
$$21 \equiv 1.(-21) - 26.(-21) \pmod{26}$$
$$a^1 \equiv -21 \equiv -21 + 26 \equiv 5 \pmod{26}$$
$$if\ a = 21 \Rightarrow 4a = 84$$
$$84 + b \equiv 6 + b \equiv 21 \pmod{26}$$
$$b = 15$$

Bulunan a=21 ve b=15 için bölüm 3.2.3. Affine Kriptosistem ile verilen denklik sağlanamayacağından farklı değerler seçilir. Bu defa e için b karakter karşılığı 1 seçilir ve $t_p = t_c$ [41] sebebiyle t için y karakter karşılığı 24 seçilir:

41 Şifreli metinde karşılığı aranan harf t olduğundan bir sonraki en yakın frekansa sahip karakter karşılığı seçilir.

$$4a + b \equiv 1 \pmod{26}$$
$$19a + b \equiv 24 \pmod{26}$$
$$4a + b \equiv 1 \pmod{26}$$
$$4a - 26b = 1 \Rightarrow$$
$$\begin{pmatrix} 26 = 5.5 + 1 \Rightarrow \\ 5 = 5.1 + 0 \end{pmatrix} \text{ //öklid}$$
$$\begin{pmatrix} 1 = 26 - 5.5 \\ 1 = 5(-5). -26(-1) \end{pmatrix} \text{ //ters öklid}$$
$$a \equiv -5 \equiv -5 + 26 \equiv 21 \pmod{26}$$
$$a = 5 \Rightarrow$$
$$4a = 20 \Rightarrow$$
$$20 + b \equiv 1 \pmod{26} \Rightarrow$$
$$b = 7 \Rightarrow$$
$$a = 5, \quad b = 7 \Rightarrow$$
$$4.5 + 7 \equiv 1 \pmod{26}$$
$$19.5 + 7 \equiv 24 \pmod{26}$$

b. Bulunan a=5, ve b=7 için yukarıdaki denklik sağlanabilmiş olduğundan $\mathcal{K}(5x + 7)$ şifreleme anahtarına ait şifre çözme anahtarı hesaplanır:

$$aa^{-1} \equiv 1 \pmod{26} \Rightarrow$$
$$aa^{-1} - 26y = 1 \Rightarrow 5a^{-1} - 26y = 1 \Rightarrow$$
$$\begin{pmatrix} 26 = 5.5 + 1 \\ 5 = 5.1 + 0 \end{pmatrix} \text{ //öklid}$$
$$\begin{pmatrix} 1 = 26 - 5.5 \\ 1 = 5(-5) - 26(-1) \end{pmatrix} \text{ //ters öklid}$$
$$y \equiv 5x + 7 \pmod{26} \Rightarrow$$
$$y - 7 \equiv 5x \pmod{26} \Rightarrow$$
$$\mathcal{D}_{\mathcal{K}} \equiv 21(y - 7) \equiv 21y - 147 \equiv 21y - 17 \pmod{26} \Rightarrow$$
Şifreleme: $\mathcal{E}_{\mathcal{K}}(x) = 5x + 7 \; mod \; 26$
Deşifreleme: $\mathcal{D}_{\mathcal{K}}(x) = 21x - 17 \; mod \; 26$

c. Elde edilen $\mathcal{D}_{\mathcal{K}}(x)$ şifre çözme anahtarı ile şifreli metnin bütün karakterleri Tablo XII. İngiliz alfabesinin mod 26'ya göre karşılıkları karşılıklarıyla yeniden hesaplanır. Elde edilen ve aşağıda verilen sonuç anlamlı bir metin olduğundan affine kriptosistemin anahtar ikinci denemede kırılmıştır.

	d	**t**	**f**	**d**	**w**	**h**	**o**	**d**	**u**	**v**	**i**	**b**	**o**	**t**	**v**	**y**	**x**	**v**	**u**	**t**
	3	19	5	3	22	7	14	3	20	21	8	1	14	19	21	24	23	21	20	19
$\mathcal{D}_{\mathcal{K}}(x)$:	20	18	10	20	3	0	17	20	13	8	21	4	17	18	8	19	24	8	13	18
	u	s	k	u	d	a	r	u	n	i	v	e	r	s	i	t	y	i	n	s

	y	**v**	**y**	**d**	**y**	**b**	**z**	**g**	**h**	**w**	**w**	**v**	**r**	**y**	**v**	**z**	**u**	**h**	**u**	**w**
	24	21	24	3	24	1	25	6	7	22	22	21	17	24	21	25	20	7	20	22
$\mathcal{D}_{\mathcal{K}}(x)$:	19	8	19	20	19	4	14	5	0	3	3	8	2	19	8	14	13	0	13	3
	t	i	t	u	t	e	o	f	a	d	d	i	c	t	i	o	n	a	n	d

	g	**z**	**o**	**b**	**u**	**t**	**v**	**r**	**t**	**r**	**v**	**b**	**u**	**r**	**b**	**t**	**d**	**t**	**b**	**v**
	6	25	14	1	20	19	21	17	19	17	21	1	20	17	1	19	3	19	1	21
$\mathcal{D}_{\mathcal{K}}(x)$:	5	14	17	4	13	18	8	2	18	2	8	4	13	2	4	18	20	18	4	8
	f	o	r	e	n	s	i	c	s	c	i	e	n	c	e	s	u	s	e	i

	y	**z**	**o**	**m**	**z**	**t**	**b**	**v**	**y**	**t**	**b**	**r**	**o**	**b**	**y**	**f**	**b**	**x**	**v**	**w**
	24	25	14	12	25	19	1	21	24	19	1	17	14	1	24	5	1	23	21	22
$\mathcal{D}_{\mathcal{K}}(x)$:	19	14	17	11	14	18	4	8	19	18	4	2	17	4	19	10	4	24	8	3
	t	o	r	l	o	s	e	i	t	s	e	c	r	e	t	k	e	y	i	d

Düz Metin:
$$\begin{bmatrix} \text{uskudaruniversityins} \\ \text{tituteofaddictionand} \\ \text{forensicsciencesusei} \\ \text{torloseitsecretkeyid} \end{bmatrix}$$

6. ICDS

6.1. Giriş

Dördüncü Bölüm – Mesaj Özetleme / Hash Algoritmaları başlığı altında özetleme algoritmaları ve Beşinci Bölüm – Kriptografik Saldırılar Başlığı altında bu fonksiyonların saldıra açık yönleri detaylı ve uygulama olarak incelenmiştir.

Mesaj özetleme algoritmaları temelinde kriptografik ve matematiksel yapıda olsalar da kendi alanlarını aşkın olarak birçok alan için başvurulan metotlardır.

Bu çalışma kapsamında tasarlanan ICDS(Inconvertible Data Signature), adli bilimlerde kullanılan mesaj özetleme fonksiyonlarının zayıflıklarından arındırılmış ve adli alanda kullanımı da senkronize edilebilecek yapı motivasyonuyla geliştirilmiştir.

Tasarım motivasyonu adli bilimler / adli bilişim olmasıyla birlikte, elde edilen fonksiyon ile en çok kullanılan ve bilinen mesaj özetleme fonksiyonlarından daha güvenilir ve esnek bir yapı elde edilmesi amaçlanmıştır. Bu amaç dâhilinde NIST gibi kurumlarca uluslar arası alana kullanımı önerilen ve adli bilimler alanında delillerin müşterekliğinin ispatı amacıyla kullanılan mesaj özetleme fonksiyonlarına bir alternatif geliştirilmesi amaçlanmıştır.

Bugün en sık kullanılan mesaj özetleme algoritmaları MD ailesini temel almıştır. Bu Fonksiyonlar iteratif olan bir sıkıştırma fonksiyonu, başlangıç durum değerleri, sabit uzunlukta blok yapıları kullanır ve çıkış (hash) değerleri olarak ayrıca sabit uzunlukta bir blok kullanırlar.

Literatür yaygın olarak National Institute of Standards and Technology (NIST)'nin öneri ve belirlediği standartları takip etmektedir. Bu kapsamda NIST'in

2007 yılında belirlediği ve bugünde kullanılan SHA ailesinin standardını oluşturduğu öneri ve standartları aşağıda verilen gereksinimlerdedir (22):

1. Algoritma yayınlanabilir ve herkesçe bilinebilir olmalıdır

2. Algoritmanın uygulanabileceği platformlar geniş aralıkta olmalıdır

3. Algoritma 224, 256, 384 ve 512 bit mesaj özetlemeyi ve en az $2^{64} - 1$ bit uzunluğunda mesaj (girdi) uzunluğunu destekliyor olmalıdır

NIST, yukarıda sayılanlar dışığında özetleme fonksiyonlarının güvenliği, hesaplama verimliliği, hafıza-bellek gereksinimleri, donanımsal ve yazılımsal uygulama elverişliliği ve lisanslama gereksinimlerini değerlendirme parametreleri olarak kabul etmektedir.

Bu çalışma kapsamında tasarlanan ICDS ile NIST'in bütün gereksinimlerini sağlayan ve adli bilimler alanında kullanıma uygun bir yapı oluşturulması amaçlanmıştır.

Beşinci Bölüm – Kriptografik Saldırılar Başlığı altında zayıflıkları ele alınan mesaj özetleme fonksiyonlarından daha güvenilir, geliştirilmiş, hızlı ve esnetilebilir bir yapı elde edilmesi amacıyla:

1. Mevcut mesaj özetleme fonksiyonlarının kullanım alanları ve bu fonksiyonlar incelenmiştir (Dördüncü Bölüm – Mesaj Özetleme Algoritmaları / Hash Algoritmaları).
2. En çok bilinen mesaj özetleme fonksiyonlarının zayıf yönleri ve saldırı yöntemleri, incelenmiştir (Beşinci Bölüm – Kriptografik Saldırılar).

Bu İncelemeler neticesinde bu bölümde;

1. Adli bilimlerde delil bütünlüğü ispatında, delil müşterekliği ve delillerin saklanması motivasyonunda gereksinimler tespit edilmiştir.
2. Geliştirilecek mesaj özetleme algoritmasının esnek yapıda olması ve farklı alanlara da kaynak olabilmesi için gereksinimler belirlenmiştir.
3. Güvenlik, hız ve donanım gereksinimleri incelenmiştir.
4. Yukarıda yer alan inceleme tespitler neticesinde modüler olarak matematiksel algoritma ve yapılar tasarlanmıştır.
5. Geliştirilen algoritmaların her birisi bilinen kriptografik saldırılar ve daha güçlü saldırılar altında sınanmıştır.
6. Modüler olarak ayrı ayrı geliştirilen yapıları, adli bilimler alanına ve diğer ilgili alanlarına esnetilebilecek bir algoritma ile bir araya getirilmiştir.
7. Geliştirilen mesaj özetleme algoritmasının bir bütün olarak alan uygulanabilirliği incelenmiştir.
8. Geliştirilen mesaj özetleme algoritmasının alan kullanımındaki güvenlik ve uygulanabilirliği incelenmiştir.
9. Geliştirilen mesaj özetleme algoritmasının ve NIST standartlarının birbirleriyle *karşılıklı* gereksinim durumları incelenmiştir.

6.2. Gereksinimler

6.2.3. Adli Bilimler

Yukarıda ele alınan bilgiler doğrultusunda, tasarlanan özetleme algoritmasının adli bilimlerde sağlaması gereken temel özellikler aşağıda verilmiştir.

1. Delil bütünlüğü ispatında kullanılabilmelidir.

a. Sayısal delile yapılacak en küçük müdahalede özet değeri büyük farklılıklar göstermelidir.
b. Mevcut sayısal delil üstünde tasarlanarak bir veri eklenmesi, silinmesi veya değiştirilmesiyle ilk özet değerinin elde edilmesi hesaplanabilir olmamalıdır.
c. Mevcut sayısal delilin özet değerinde özet değeri sonucu verecek farklı veriler hesaplanabilir olmamalıdır.

2. Donanımsal ve yazılımsal elverişli olmalıdır.
a. Olay yeri inceleme birimlerinin veya olay yeriyle ilk temasta bulunan yetkililerin temel teknik bilgisiyle kullanabileceği donanımların geliştirilmesine uygun olmalıdır.
b. Donanımsal uygulama ile kurulum ve hazırlık gerektirmeyen cihazlar geliştirilebilmeli ve acil gereksinimlerde uygulanabilmelidir.
c. Donanımsal erişimin mümkün olmadığı veya farklı gereksinimler için yazılımsal uygulamalar geliştirilebilmelidir.
d. Delil donanımsal müdahalenin uyarlanamayacağı herhangi bir sayısal ortamda bulunması durumunda, yazılımsal uygulama ile özetlenebilmelidir.

3. Delil inceleme aşamasında birim veya personel özgül inceleme yapabilmeli ve bütünlüğünü koruyan delil kendilerine özgül özet değerinde özetlenebilmelidir.

a. Delile erişen yetkili birim veya personel erişim öncesi ve sonrasında kendilerine özgül anahtarla özet değerleri alabilmeli ve delil bütünlüğü birden fazla anahtarla ispat edilebilmelidir.
b. Delile ve bütün anahtarlara herkes sahip olsa da bu parametrelere özgül mesaj özet değerleri farklı parametrelerle elde edilememelidir.
c. Delil birden fazla birim ve personelce, birden fazla anahtarla özetlenebilmeli ve ayrı-ayrı bütünlük ispatı yapılabilmelidir. Delil, anahtar ve özet değerlerinin herkesçe bilinebilir olması durumunda delil bütünlüğü ispatının müşterekliği sağlanabilmelidir.
d. Mevcut sayısal delilin özet değerinde özet değeri sonucu verecek farklı veriler hesaplanabilir olmamalıdır.

4. Kolay uygulanabilir, anlaşılabilir ve geliştirilebilir algoritmaya sahip olmalıdır.

a. Uygulama ve anlaşılabilir algoritmayla delile müdahalede olası hataların önüne geçilmesi, birim ve personel tarafında etkin tepki sürelerinin elde edilmesi sağlanmalıdır.
b. Algoritma sadece teknik bilgiye sahip birim ve yetkililerce uygulanabilir değil, kamu tarafınca da uygulanabilir olmalı ve delil müşterekliği sağlanabilmelidir.

c. Algoritma farklı platformlarda uygulamaya ve açık kaynak olarak gelişen teknolojik imkânlar doğrultusunda geliştirilmeye uygun olmalıdır.

d. Algoritmanın uygulandığı yazılım ve donanımların teknik yeterliliğe sahip olmayan kişilerce erişilebilir olması için uyarlama geliştirmeler mümkün olmalıdır.

5. Delile veya amaca özgül parametrik yapıda olmalıdır.

a. Sayısal delilin veri boyutuna veya olası kanuni-normsal zorunluluklara göre geniş aralıkta özet değeri uzunluğu seçilebilir olmalıdır.

b. Sayısal delilden elde edilecek özet değerinin ayrıca bir anahtara bağımlı olup olmaması seçilebilir olmalıdır.

c. Özet değeri sayısal deliller haricinde, adli bilimler ile ilişkili sayısal verilerin imzalanabilmesi, kaynak ispatı gereksinimlerine uygun parametre seçeneklerine sahip olmalıdır.

d. Özet değeri sayısal delil ve adli verilerin kaynak takibini sağlamaya uygun olmalıdır.

6. Güvenirliği bilimsel yöntemlerle ispatlanmış ve bu ispatlar tekrar gözlemlenebilir olmalıdır.

a. Sayısal delilde yapılacak en küçük değişimin, özet değerinde yüksek değerde değişimlere sebep olduğu bilimsel olarak ispatlanmalıdır.

b. Sayısal delile kontrollü bir müdahale ile istenilen değerde veya aynı değerde özet değerinin hesaplanabilir olamayacağı bilimsel olarak ispatlanmalıdır.

c. Mevcut sayısal delil ile aynı özet değerini veren başka veri yapılarının hesaplanabilir olamayacağı bilimsel olarak ispatlanmalıdır.

d. Bilimsel ispatların hepsi gözlemlenebilir, tekrarlanabilir, güvenilirlik ispatı amacına özgül analiz ve yöntemlerle gerçekleştirilmiş olup bütün olarak kamu ile paylaşılmış olmalıdır.

6.2.2. Esnek Yapı

Yukarıda ele alınan bilgiler doğrultusunda, tasarlanan özetleme algoritmasınun Adli bilimler başta olmak üzere esnek kullanım alanlarına sahip olabilmesi için sağlanması gereken temel özellikler aşağıda verilmiştir.

1. Özetleme yapılabilecek platform seçenekleri geniş bir aralıkta olmalıdır.

a. Özetleme algoritması donanımsal, yazılımsal ve her iki platformu içerir hibrit uygulamaya elverişli olmalıdır.

b. Yazılımsal tarafta, masaüstü, aplikasyon (mobil platformlar dâhil), bulut ve internet gibi sayısal iletişim sistemleri üzerinde erişime uygun uygulamalar geliştirilebilmesi mümkün olmalıdır.

c. Donanımsal tarafta, düşük donanım gereksiniminde olması veya donanıma uygun özet değeri uzunlukları, parametreleri ve değişkenleri olmalıdır,

2. MAC[42] veya MDC[43] yapısında uygulanabilir olmalıdır.

a. MAC yapısında çalışabilmesi için özet değerinde etkili farklılıklara yol açan ayrı bir anahtar giriş parametresi olmalıdır.

b. Anahtar parametresinin boş olması durumunda MDC aksi durumda MAC yapısında olan bir yapıda olmalıdır.

3. DSA[44] uygulamalarında kullanılabilir olmalıdır.

a. Farklı sayısal imza algoritmalarına uygun algoritmik yapıya sahip olmalıdır.

b. Geniş aralıktaki özet değeri uzunluğu seçeneği sayısal imza algoritmalarının gereksinimlerini karşılayabilmelidir.

c. Asimetrik anahtar kriptosistemleriyle hesaplanan açık (public) ve gizli (private) anahtarlar mesaj veya anahtar değeri olarak kullanılabilmelidir.

d. Asimetrik anahtar kriptosistemlerinin üretebildiği anahtar uzunluklarını tek blok ile özetlenebilmeli ve anahtarın uzunluğu ile sağladığı güvenliği azaltılmamalıdır.

42 MAC: Message Authentication Code – Mesaj Yetkilendirme Kodu

43 MDC: Modification Detection Code – Değişiklik Denetleme Kodu

44 DSA: Digital Signature Algorithm – Sayısal İmza Algoritması

e. Anahtar parametresinin boş olması durumunda MDC aksi durumda MAC yapısında olan bir yapıda olmalıdır.

f. Donanımsal tarafta, düşük donanım gereksiniminde olması veya donanıma uygun özet değeri uzunlukları, parametreleri ve değişkenleri olmalıdır.

6.2.3. Güvenlik, Hız ve Donanım

Yukarıda ele alınan bilgiler doğrultusunda, tasarlanan özetleme algoritmasında sağlanması gereken güvenlik, hız ve bellek gereksinimleri aşağıda verilmiştir.

1. Algoritma yapısına bağlı olan ve algoritma yapısına bağlı olmayan saldırı yöntemlerine karşı dirençli olmalıdır.

a. Algoritma yapısına bağlı olan MITM[45], Fixed Point[46] ve diferansiyel[47] saldırılarına dirençli olmalıdır.

b. Algoritma yapısına bağlı olmayan doğum günü, ön görüntü, ikinci ön görüntü saldırılarına dirençli olmalıdır.

c. Bir özet değerinden mesaj değerinin bulunabilmesi mümkün olmamalı ve algoritma tek yönlü bir fonksiyon olarak çalışmalıdır.

d. Seçilen bir özet değerine sahip mesaj hesaplanamamalıdır.

e. Aynı özet değerini veren-çakışan iki farklı mesaj değeri hesaplanamamalıdır.

45 MITM: Meet in the Middle – Ortada buluşma (Bknz: Beşinci Bölüm Kriptografik Saldırılar)

46 Fixed Point: Sabit Nokta (Bknz: Beşinci Bölüm Kriptografik Saldırılar)

47 Diferansiyel Saldırılar: (Bknz: Beşinci Bölüm Kriptografik Saldırılar)

f. Mesaj değerindeki en küçük değerin özet değerinde kaotik etkiye sahip olması sağlanmalıdır.

g. Algoritma yapısına bağlı saldırılara direnç sağlanması için mesajın sadece özet değerine etki eden bir değer değil ayrıca fonksiyon parametrelerine etki etmesi sağlanmalıdır.

2. Özet değerinin elde edilmesi düşük zaman maliyetinde olmalıdır.

a) Özet değerinin hesaplanması kısa sürede sonuçlanmalıdır.

b) Zaman maliyetinde elde edilecek fayda güvenlik ve bellek gereksinimi maliyetine sebep olmamalıdır.

c) Zaman maliyetindeki fayda farklı donanım yapılarında da orantılı sonuçlar vermelidir.

d) Zaman maliyetindeki fayda, aynı donanım üzerinde ve aynı anda birden fazla özetleme işleminde de fayda sağlamalıdır.

3. Düşük bellek ve donanım gereksinimlerine sahip olmalıdır.

a) Algoritma düşük bellek-hafıza alanına gereksinim duymalıdır.

b) Algoritma düşük işlem ve mikro işlemci alanına ihtiyaç duymalıdır.

c) Algoritma aritmetik hesaplama ile eş işlemleri mantıksal hesaplama ile elde edebilmeli ve böylelikle donanım-hız maliyetinde fayda sağlanabilmelidir.

6.3. Matematiksel Yapı ve Algoritmalar

6.3.1. Matematiksel Altyapı

6.3.1.1. Blok Yapısı

Algoritma yapısı temel olarak blok üzerinde işlem yapılmasına dayalıdır. Blok yapısına bağlı olarak:

1. Özet değerinin alabileceği en kısa ve en uzun değerlerin aralığı bellidir.

2. Algoritmanın kullanacağı kelime uzunluğu bellidir.

3. Gereksinim duyulan donanımsal hafıza bellek ve işlemci mimarisi bellidir.

Yukarıdan anlaşılacağı üzere, algoritmanın güvenlik, esneklik, hız, donanım gereksinimi blok yapısının özelliklerine bağlıdır. Bu durumda ICDS ile elde edilmek istenen gelişmiş yapı için blok yapısının özel olarak tasarlanması gerekmektedir.

ICDS blok yapısından elde edilmek istenen faydalar ve bu amaçlarla yapı tasarımı aşağıdaki şekilde oluşturulmuştur:

1. Blok yapısının en küçük değerleri bit değerleridir. Bu bit değerlerinin 8 ve katları sayısınca bir araya gelmesiyle kelime (word) değerleri ortaya çıkar.

2. Blok yapısının oluşturulmasında kelime değerleri kullanılmıştır. Bu kelime değerlerinin bit uzunluğu donanımsal hafıza-bellek ve işlemci gereksinimine de temel oluşturduğundan en geniş aralıkta platformda çalışabilecek kelime uzunluğu olarak **32-bit kelime** uzunluğu kullanılması tercih edilmiştir. $\boldsymbol{\beta}$ bloğu, $\boldsymbol{c(\beta)}$ β

bloğundaki kelime sayısını ettiğinde, β bloğunun bit uzunluğunu temsil eden, ***len(ß)*** değeri aşağıdaki eşitliktedir.

$$len(\beta) = 32.c(\beta) \quad (104)$$

3. Blok uzunluğu algoritmanın güvenliğini doğrudan etkileyen yapı değeridir. Ancak blok uzunluğunun çok büyük olması algoritmanın işlem ve tepki süresini doğrudan düşüren bir etki sunar. Literatürde MD4 (46)**,** MD5 (20), SHA (48), algoritmalarında 512-bitlik ve 1024-bitlik sabit bloklar kullanmıştır. ICDS için istenilen özet değerine göre değişen değerlerde blok uzunluğu kullanılmıştır.

Burada $\mathcal{H}$ özet değeri ve ***HASHLEN*** özet değerinin uzunluğu olduğunda, blok uzunluğunun özet değeri uzunluğunun iki katı olması tercih edilmiştir.

$$len(\beta) = 2.HASHLEN \quad (105)$$

4. Bilinen mesaj özetleme algoritmaları özet değerlerini lineer fonksiyonlar ile hesaplar. ICDS'in hesaplama ve sıkıştırma fonksiyonlarında bu hesaplama iki boyutlu tablolar üzerinde yapılır. Bu yapıyla blok uzunluğunu oluşturan 32-bitlik kelimelerin 8-bitlik değerler olarak işlenmesi sağlanır. 8-bite indirgenmiş bu sanal yapı ICDS'in donanım ihtiyaçlarının en düşük seviyede tutulmasını sağlar.

ICDS'in 256-bit mesaj özet değerinde kullanılan 512-bitlik blok yapısı aşağıda Şekil 22**:** ICDS - 512 bitlik blok yapısı temsil edilmiştir.

w0	w1	w2	w3	w...	w...	w...	w...	w...	w...	w...	w...	w...	w...	w...	w15
8bit	8bit	8bit	8bit	8bit	8bit	8bit	8bit	8bit	8bit	8bit	8bit	8bit	8bit	8bit	8bit
8bit	8bit	8bit	8bit	8bit	8bit	8bit	8bit	8bit	8bit	8bit	8bit	8bit	8bit	8bit	8bit
8bit	8bit	8bit	8bit	8bit	8bit	8bit	8bit	8bit	8bit	8bit	8bit	8bit	8bit	8bit	8bit
8bit	8bit	8bit	8bit	8bit	8bit	8bit	8bit	8bit	8bit	8bit	8bit	8bit	8bit	8bit	8bit
= 32bit	= 32bit	= 32bit	= 32bit	= 32bit	= 32bit	= 32bit	= 32bit	= 32bit	= 32bit	= 32bit	= 32bit	= 32bit	= 32bit	= 32bit	= 32bit

Şekil 22: ICDS - 512 bitlik blok yapısı

5. 1, 2, 3 ve 4 ile açıklanan ICDS'in blok özellikleri şu şekildedir;

Kelime; $\beta = \beta_0, \beta_{\dots}, \beta_{n-1}$ dizisi elemanları olarak temsil edilirler ve her bir kelime 32-bit uzunluğundadır.

Blok; β ile temsil edilir, uzunluğu *len(β)* ile temsil edilir. 32-bitlik kelimelerden oluşur.

Blok Uzunluğu; *len(β)* ile temsil edilir, enaz 512-bit olmak üzere mesaj özeti uzunluğunun iki katı uzunluktadır $len(\beta) = 2.HASHLEN$.

$$len(\beta) = \begin{cases} 512, & 512 > 2.HASHLEN \\ \left\{ \begin{array}{l} len(\beta) = 512 \\ while\ len(\beta) < 2.HASHLEN\ do \\ \quad \{len(\beta) = 2.len(\beta)\} \end{array} \right\}, & 2.HASHLEN \geq 512 \end{cases} \qquad \textbf{(82)}$$

128 ≤ HASHLEN ≤ 65536 eşitliği sebebiyle *len(β)* için olası değer aralığı aşağıdaki eşitliktedir.

$$512 \leq HASHLEN \leq 131072 \quad (83)$$

Blok Uzunluğu Tablosu; ICDS en kısa 128-bit özet değeri uzunluğu için hesaplama yapar. 128-65536 bit özet değeri aralığına göre kullanılacak blok uzunluklarının özellikleri tablo ile verildiği değerlerdedir[48.]

Tablo XVII. ICDS - Blok uzunlukları değerleri

Özet Uzunluğu ***HASHSIZE***	**Blok Uzunluğu** ***len(β)***	**Kelime Sayısı** ***c(в)***
128	512 *	8
160	512 *	10
192	512 *	12
224	512 *	14
256	512 *	16
384	768	24
512	1024	32
1024	2048	64
2048	4096	128
4096	8192	256

* 256-bit mesaj özeti değerine kadar 512-bitlik blok uzunlukları kullanılır, bu değerin üzerindeki özet değerlerinde hesaplama için özet değerinin iki katı uzunluğunda blok uzunlukları kullanılır…

6.3.1.2. S-Box Tablosu

Algoritma 32-bit uzunluktaki kelimelerden oluşur ve bu kelimelerin her birisi 8-bitlik 4 ayrı parça olarak değerlendirilir *(Bkz: Bölüm 7.3.1.1. Şekil 22: ICDS - 512 bitlik blok yapısı)*. ICDS 32- bitlik kelimelerden oluşan blok yapısını iteratif sıkıştırma

48 65536-bit özet değeri burada bir üst sınır olmayıp çalışma yapıldığında bilinen en yüksek özet değeri 1024-bittir.

fonksiyonlarından geçirir. Bu fonksiyonlarda anahtar hesaplamasında ve bloğun ilk değerlerinin atanmasında s-box değişim tablosu kullanılır.

S-Box değişim tablosu AES (49) başta olmak üzere birçok kriptografik algoritmada kullanılan ve etkili kullanımında yapıya güç kazandıran bir yöntemdir.

ICDS, bilinen s-box tablosu değerlerinden farklı değerlere sahip bir tablo kullanır. ICDS için kullanılan bu tablo daha karmaşık ve korelâsyonu değeri daha düşük değerler sıralamasına sahiptir.

Bu tablonun oluşturulmasında aşağıdaki fonksiyon kullanılır.

$$
\begin{aligned}
&a = 203 \\
&for\ i = 0\ to\ 255\ do \\
&\begin{cases}
sbox[i] = \big((a.3)\ mod\ 256\big) \\
sbox[i] = \begin{pmatrix}
\big((sbox[i] \ll 4)\ \wedge 192\big)\ \vee \wr \\
\big((sbox[i] \gg 2)\ \wedge 48\big)\ \vee \wr \\
\big((sbox[i] \ll 2)\ \wedge 12\big)\ \vee \wr \\
\big((sbox[i] \gg 4)\ \wedge 3\big)
\end{pmatrix} \\
a = (a+1)\ mod\ 256
\end{cases} \\
&for\ i = 0\ to\ 1023\ do \\
&\begin{cases}
sbox[i\ mod\ 256] = \\
\begin{pmatrix}
\Big(\big((17.sbox[i\ mod\ 256]\ mod\ 256) \ll 4\big)\ \wedge 192\Big) \vee \wr \\
\Big(\big((17.sbox[i\ mod\ 256]\ mod\ 256) \gg 2\big)\ \wedge 48\Big) \vee \wr \\
\Big(\big((17.sbox[i\ mod\ 256]\ mod\ 256) \ll 2\big)\ \wedge 12\Big) \vee \wr \\
\Big(\big((17.sbox[i\ mod\ 256]\ mod\ 256) \gg 4\big)\ \wedge 3\Big)
\end{pmatrix}
\end{cases}
\end{aligned}
\qquad (84)
$$

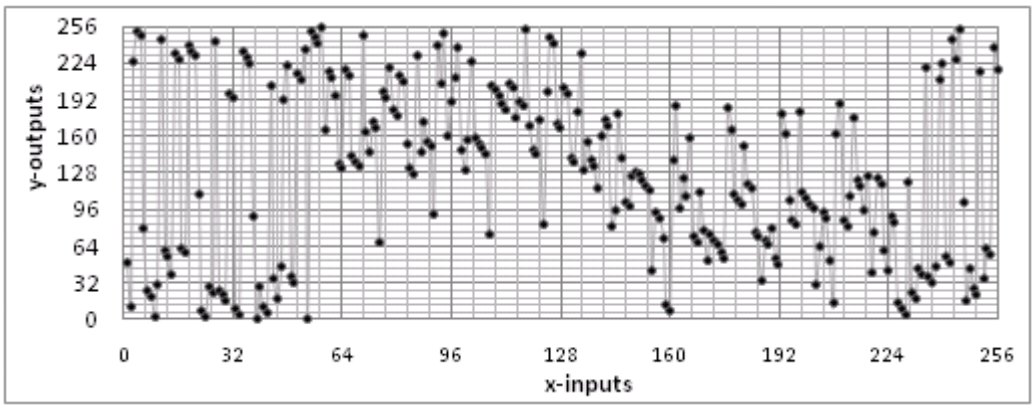
Yukarıdaki formül ile elde edilen s-box tablosu aşağıda tablo Tablo XVIII**.** ICDS - SBox Tablosu ile verilmiştir. Bu tabloda yer alan değerler 16'lık sayı sistemi temsilinde olup toplam 256 ayrı değer içerir.

Bu 256 ayrı değer 8-bitlik bir değerin alabileceği değerleri temsil eder. 8-bitlik bir değer 16'lık sayı sisteminde iki karakter ile gösterilebilir. Burada *xy* sayısı için *x* tablonun satırlarını, *y* tablonun sütunlarını temsil eder.

Tablo XVIII. ICDS - SBox Tablosu

		y														
		0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**a**	**b**	**c**	**d**	**e**	**f**
x	**0**	31	0c	e2	fc	f8	50	19	14	3	1e	f5	3d	38	27	e9	e4
	1	3f	3a	ef	eb	e6	6d	7	2	1d	18	f3	1a	15	10	c6	c1
	2	9	4	ea	e5	e0	5b	1	1c	0b	6	cc	24	13	2e	c0	de
	3	26	21	d7	d2	ec	0	fb	f6	f1	ff	a6	d9	d4	c3	88	84
	4	db	d6	8f	8a	86	f9	a4	93	ad	a8	43	c7	c2	dc	b7	b2
	5	d5	d0	99	85	80	e7	92	ac	9b	97	5c	f0	ce	fa	a0	be
	6	d3	ed	94	83	9d	e1	9f	9a	96	91	4a	cd	c9	c4	bd	b8
	7	cf	cb	b1	bf	ba	fe	a9	95	90	ae	54	c8	f7	f2	ab	a7
	8	ca	c5	8e	89	b5	e8	82	9c	8b	87	72	a1	af	aa	51	5f
	9	b3	8d	67	63	7d	81	7f	7a	75	71	2a	5d	58	47	0d	8
	a	8c	bb	62	7c	6b	9e	49	44	70	4e	34	4b	46	42	3b	36
	b	b9	a5	6e	69	64	98	77	73	4d	48	22	45	41	4f	35	30
	c	b4	a3	68	57	53	b6	6f	6a	65	61	1f	40	5e	59	33	0e
	d	a2	bc	56	52	6c	b0	79	74	60	7e	29	4c	7b	76	3c	2b
	e	5a	55	0f	0a	5	78	17	12	2d	28	dd	25	20	2f	d1	df
	f	37	32	f4	e3	fd	66	11	2c	1b	16	d8	23	3e	39	ee	da

4. **Örnek:** $243_{10} = f3_{16}$ olur. Bu değerin s-box tablosundaki karşılığı için *x*(satır) bölümünde f ve *y*(sütün) bölümünde *3* karşılığına bakılır. Bu durumda Tablo XVIII**.** ICDS - SBox Tablosu'e göre *sbox[f3] = e3* olarak bulunur.



Şekil 23: SBOX dağılım grafiği

Bu sbox tablosunun dağılım grafiği Şekil 23 ile verilmiştir. Bu grafikle de görüldüğü gibi, kullanılan sbox tablosu *x* girişlerinde yüksek dağılım gösteren sonuç karşılıkları döndürmektedir.

6.3.1.3. Mesaj Dolgulama

Literatürde yer alan mesaj özetleme algoritmaları mesajı, bloğun uzunluğunun katları kadar dolgular.

ICDS'de blok uzunluğu üretilecek özet değeri uzunluğuna göre değişmektedir. Bu yapısı sebebiyle dolgulama uzunluğu ve değerleri özet değeri uzunluğuna göre değişir.

Dolgulama işleminde, *len(β)* blok uzunluğu için, *len(*$\mathcal{M}$*)* mesaj uzunluğunun len(β)'nin katları olması sağlanır.

ICDS'de dolgulama işlemi yaygın literatürden farklı gerçekleşir. Yaygın literatür mesajın son bit değerinden sonra bir bitlik 1 değerini ekler ve devamını 0 bitleriyle doldurup son 64 veya 128 bitlik bölümüne mesaj uzunluğunu ekler.

ICDS $\mathcal{M} = \mathcal{M}_0, \ldots$ mesaj dizisini 8-bitlik parçalar halinde okur. $\mathcal{M}$ mesaj dizisinin son elemanı da kullanıldıktan sonra, β dizisine alınacak sıradaki değer sbox

tablosundan *sbox[i mod 256]* olarak çağırılarak β dizisine yerleştirilir. Bu dolgulama işlemi blok doldurulunca kadar devam eder.

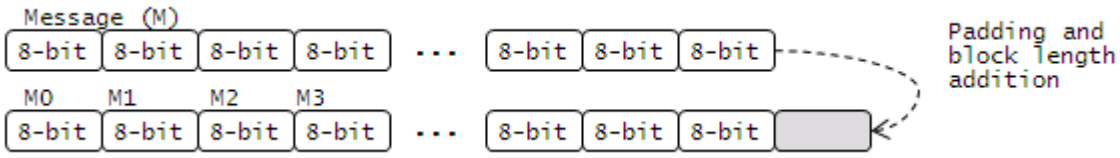
Farklılaştırılmış dolgulama işlemi *0* değerli bit yerleşimine odaklanan diferansiyel saldırılarına direnç sağlamayı amaçlar.

Dolgulama işleminde kullanılan bit uzunluğu *PAD*, mesaj uzunluğu $len(\mathcal{M})$ ve blok uzunluğu $len(\beta)$ olduğunda *PAD* aşağıda verilen değerde olur.

$$PAD = \begin{cases} \text{len}(\beta) \bmod \text{len}(\mathcal{M}), & \text{len}(\beta) > len(\mathcal{M}) \\ \text{len}(\beta) \bmod \text{len}(\mathcal{M}), & \text{len} < len(\mathcal{M}) \end{cases} \quad (85)$$

Mesaj uzunluğu olan $len(\mathcal{M})$ değerinin dolgulamada kullanılmaması ve uzunluk değeri yerine *sbox[i]* değerlerinin kullanılması, özetlenecek mesaj uzunluğunun sınırsız olmasını sağlar. Literatürde yer alan çalışmalar kullandıkları kelime uzunlukları ve dolgulama ile ilişkili olarak mesaj boyutlarını 64-bit veya 128-bit ile gösterilebilecek sınırlarda kullanırlar.

ICDS'in dolgulama işlemi aşağıda Şekil 24 ile temsil edilmiştir.



Şekil 24: ICDS - Mesaj dolgulama

$\mathcal{M}_0||\mathcal{M}_1||\mathcal{M}_{....}$ 8'er bitlik mesaj değerleri ve $sbox[i\ mod\ 256]$ $i\ mod\ 256$ değerine göre sbox tablosundan çağırılan 8-bitlik değerler olduğunda mesaj dolgulama, aşağıda yer alan eşitlik 5 ile temsil edilebilir;

$$\mathcal{M}_i \coloneqq \mathcal{M}_0||\mathcal{M}_1||\mathcal{M}_{....}||sbox[i\ mod\ 256] \qquad (86)$$

6.3.2. Alt Fonksiyonlar

Bu bölümde ICDS'in mesaj özetleme işlemi için tasarlanan fonksiyonlar verilmiştir. Alt fonksiyonlara ait algoritmalarda kullanılan hexadeximal sayılar, sollarına yerleştirilen $\mathbf{0x}$ ... ile temsil edilmiştir.

6.3.2.1. blockmix fonksiyonu

Bu fonksiyon 32-bitlik kelimelerden oluşan bloğun 8-bitlik parçalar halinde karıştırılmasını ve ICDS'in daha etkili diffusion ve confusion oluşturmasını sağlar. Bu sayede mesajın herhangi bir bitindeki değişimin bütün blok içerisinde etki oluşturması sağlanır.

ICDS, bölüm 7.5.1.1'de Şekil 22**:** ICDS - 512 bitlik blok yapısı ile gösterildiği gibi mesaj özet uzunluğunun iki katı uzunluğa sahip blokları kullanır ve bu bloklar AES'de (49) olduğu gibi iki boyutlu matrisler olarak ele alınır.

blockmix fonksiyonu önce 32-bitlik kelimelerin blok içerisindeki sıralamasını değiştirir. Kelimelerin sıralama değişiminden sonra kelimeleri oluşturan 32-bitlik değerler 8-bitlik parçalar halinde yer değişimine tutulur.

Buradaki permutasyon işlemi iskambil destesinin karıştırılmasına benzer. Ancak blok yapısı iki boyutlu olduğundan bu karıştırma işlemi iki boyutlu olarak gerçekleşir.

Bu fonksiyon bloğu oluşturan kelime sayısına bağlı olarak iki farklı tür karıştırma işlemi yapar. Burada β bloğundaki kelime sayısı *c(β)* 'nın *c(β) mod 2 = 1* veya *c(β) mod 2 = 0* olması fonksiyonun iki farklı permütasyona yönelmesini işaret eder.

Fonksiyonun içerisindeki *a* ve *b* değişkenleri aşağıdaki şekilde hesaplanır.

$$\begin{cases} a = (c(\beta) - 1)/2 \\ b = (c(\beta) + 1)/2 \end{cases} \quad , c(\beta)\ mod\ 2 = 1$$
$$\begin{cases} a = c(\beta)/2 \\ b = c(\beta)/2 \end{cases} \quad , c(\beta)\ mod\ 2 = 0 \qquad (87)$$

a ve *b* değerleri belirlendikten sonra fonksiyon bu değerlere göre kelimelerin blok içerisindeki yerini ve kelimelerin bitlerini 8'erli parçalar halinde yeniden konumlandırır.

Bloktaki kelimelerin yeniden konumlandırılması *for i = 0 to (a - 1)* döngüsünde gerçekleşir.

Kelimeye ait bitlerin yeniden konumlandırılması için kelime değerinin bit değerleri taşınır, bu işlem aşağıda gösterilmiştir.

$$\beta[i] = cper\begin{pmatrix} \beta[i] \wedge 0x0000ff00 \ll 16 \vee \wr \\ \beta[i] \wedge 0xff000000 \gg 8 \vee \wr \\ \beta[i] \wedge 0x000000ff \ll 8 \vee \wr \\ \beta[i] \wedge 0x00ff0000 \gg 16 \end{pmatrix} \qquad (88)$$

Bu fonksiyon Algoritma 1. ICDS – blockmix fonksiyonu ile verilmiştir.

Algoritma 1. ICDS – blockmix fonksiyonu

Inputs:	$\boldsymbol{\beta}[\boldsymbol{i}]$;// 32-bit uzunluğunda kelimelerden oluşan blok dizisi.
Variables:	$\boldsymbol{a}, \boldsymbol{b}$;// β[i] kelime dizisinin permütasyonunda kullanılacak değişkenler.
	$\boldsymbol{c}(\boldsymbol{\beta})$;// β[i] dizisindeki kelime sayısı, dizinin eleman sayısı.
Outputs:	$\mathbb{O}_{\boldsymbol{i}}$;// Kelimeler ve kelimelere ait bit değerlerinin yeniden konumlandırıldığı $\beta[i]$ dizisi.

```
01: if (c(β) mod 2) = 1) then
02:   a = ((c(β) – 1 ) / 2);    b = ((c(β) + 1 ) / 2);
03: else if (c(β) mod 2) = 0) then
04:   a = (c(β) / 2);          b = (c(β) / 2);
05: end if
06: for i = 0 to (a – 1)
07:   ⦙𝕆[2.i] = cper( ((β[b + i] ∧ 0x0000ff00) ≪ 16) ⋁ ⤸
      ⦙                ((β[b + i] ∧ 0xff000000) ≫ 8) ⋁ ⤸
      ⦙                ((β[b + i] ∧ 0x000000ff) ≪ 8) ⋁ ⤸
      ⦙                ((β[b + i] ∧ 0x00ff0000) ≫ 16) );
08:   ⦙𝕆[(2.i)+1] = cper( ((β[i] ∧ 0x0000ff00) ≪ 16) ⋁ ⤸
      ⦙                  ((β[i] ∧ 0xff000000) ≫ 8) ⋁ ⤸
      ⦙                  ((β[i] ∧ 0x000000ff) ≪ 8) ⋁ ⤸
      ⦙                  ((β[i] ∧ 0x00ff0000) ≫ 16) );
09: end for
10: if (c(β) mod 2 = 1) then
      ⦙𝕆[c(β)-1] = cper( ((β[a] ∧ 0x0000ff00) ≪ 16) ⋁ ⤸
      ⦙    ((β[a] ∧ 0xff000000) ≫ 8) ⋁ ⤸
      ⦙    ((β[a] ∧ 0x000000ff) ≪ 8) ⋁ ⤸
      ⦙    ((β[a] ∧ 0x00ff0000) ≫ 16) );
11: end if
12: return 𝕆
```

Bu fonksiyona gönderilen ve bloğu oluşturan kelimelerin permütasyonu sonucu Şekil 25 ile temsil edilmiştir.

(A)

	w0	w1	w2	w3
0	a0	b0	c0	d0
1	a1	b1	c1	d1
2	a2	b2	c2	d2
3	a3	b3	c3	d3

	w2	w0	w3	w1
2	c0	a0	d0	b0
0	c1	a1	d1	b1
3	c2	a2	d2	b2
1	c3	a3	d3	b3

	w2	w0	w3	w1
2	c2	a2	d2	b2
0	c0	a0	d0	b0
3	c3	a3	d3	b3
1	c1	a1	d1	b1

(B)

	w0	w1	w2	w3	w4
0	a0	b0	c0	d0	e0
1	a1	b1	c1	d1	e1
2	a2	b2	c2	d2	e2
3	a3	b3	c3	d3	e3

	w3	w0	w4	w1	w2
0	d0	a0	e0	b0	c0
1	d1	a1	e1	b1	c1
2	d2	a2	e2	b2	c2
3	d3	a3	e3	b3	c3

	w3	w0	w4	w1	w2
2	d2	a2	e2	b2	c2
0	d0	a0	e0	b0	c0
3	d3	a3	e3	b3	c3
1	d1	a1	e1	b1	c1

Şekil 25: ICDS - blockmix fonksiyonu
(A) c(β) mod 2 = 0 için yerleşim, **(B)** (β) mod 2 = 1 için yerleşim

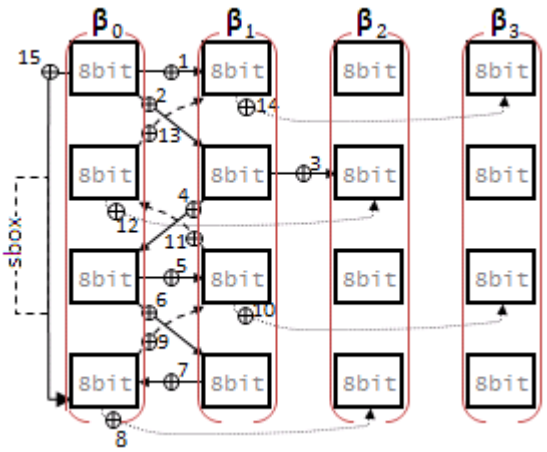
6.3.2.2. zigzag fonksiyonu

zigzag fonksiyonu, girdi olan bloktaki kelimeleri 8-bitlik parçalar halinde ve zigzag pozisyonunda XOR işleminden geçirir. Bu fonksiyon blok yerleşimi sonrasında başlatılan sıkıştırma ve hesaplama döngülerinde çağırılır.

Fonksiyonun çalışması turlar halinde gerçekleşir ve blokta bulunan kelime sayısınca *c(β)* turdan oluşur. Her turda iki kelime, dört kelimeye yaygın olarak işletilir. Buradaki her turda toplam 128-bitlik alanda işlem yapılmış olunur.

zigzag pozisyonunda XOR işlemi Şekil 26**:** ICDS - zigzag fonksiyonu ile gösterildiği gibi gerçekleşir, bu şekil üzerinde 128-bitlik bir blok uzunluğu örnek alınmıştır ve sadece ilk tura ait XOR işlemi gösterilmiştir. Burada her işlemde, iki kelime toplam 4 kelime değerine göre değer alır. Tur içerisinde $\beta[3]$ kelimesine ulaşıldığında (dolayısıyla 4. turda) devamında üç kelime bulunmaması sebebiyle, XOR işlemine alınacak kelime sırası modula ile belirlenir. Bu durumda XOR işlemine alınacak kelimelerin sıra değeri $\beta[i+1]\ mod\ c(\beta)$, $\beta[i+2]\ mod\ c(\beta)$ ve $\beta[i+$

3] $mod\ c(\beta)$ olarak bulunur. Bu modula sıra değeri sayesinde herhangi bir kelimedeki bit değeri değişikliği bütün blok içerisine yayılarak kelebek etkisi oluşturulmuş olur



Şekil 26: ICDS - zigzag fonksiyonu

Şekil 26**:** ICDS - zigzag fonksiyonu ile, her bir turda toplam 15 ayrı değerin etkileşimi görülmektedir. ICDS en az 256-bitlik mesaj uzunluğu ile çalıştığından burada *len(β)=32.c(β)* ⟶ *256 = 32.8* eşitliği ortaya çıkar. *zigzag* fonksiyonunun her kelimede 15 etki oluşturması eşitlikte kullanıldığında, *zigzag* fonksiyonu en az *8.15=120* etki oluşturur.

Bu fonksiyon Algoritma 2. ICDS - zigzag fonksiyonu ile verilmiştir.

Algoritma 2. ICDS - zigzag fonksiyonu

Inputs:	$\boldsymbol{\beta}[\boldsymbol{i}]$ **;//** 32-bit uzunluğunda kelimelerden oluşan blok dizisi.
Variables:	$\boldsymbol{c}(\boldsymbol{\beta})$ **;//** β dizisindeki kelime sayısı, dizinin eleman sayısı.
Outputs:	$\boldsymbol{\beta}[\boldsymbol{i}]$ **;//** zigzag fonksiyonundan geçirilmiş $\boldsymbol{\beta}[\boldsymbol{i}]$ dizisi.

```
01: for i = 0 to (c(β)-1) do
02: ¦β[i] ⊕=                    (β [(i+1) mod c(β)] ∧ 0xff)
03: ¦β[i] ⊕=                    ((β [(i+1) mod c(β)] ≫ 8) ∧ 0xff)
04: ¦β[(i+1) mod c(β)] ⊕=       (β[(i+2) mod c(β)] ∧ 0xff00)
05: ¦β[(i+1) mod c(β)] ⊕=       ((β[(i+1) mod c(β)] ≫ 8) ∧ 0xff00)
06: ¦β[i] ⊕=                    (β[(i+1) mod c(β)] ∧ 0xff0000)
07: ¦β[i] ⊕=                    ((β[(i+1) mod c(β)] ≫ 8) ∧ 0xff0000)
08: ¦β[(i+1) mod c(β)] ⊕=       (β[i] ∧ 0xff000000)
09: ¦β[i] ⊕=                    (β[(i+2) mod c(β)] ∧ 0xff000000)
10: ¦β[i] ⊕=                    ((β[(i+1) mod c(β)] ≪ 8) ∧ 0xff0000)
11: ¦β[(i+1) mod c(β)] ⊕=       (β[(i+3) mod c(β)] ∧ 0xff0000)
12: ¦β[(i+1) mod c(β)] ⊕=       (β[i] ≪ 8) ∧ 0xff0000))
13: ¦β[i] ⊕=                    (β[(i+2) mod c(β)] 0xff00)
14: ¦β[i] ⊕=                    ((β[(i+1) mod c(β)] ≪ 8) ∧ 0xff00)
15: ¦β[(i+1) mod c(β)] ⊕=       (β[(i+3) mod c(β)] ∧ 0xff)
16: ¦β[i] ⊕=                    sbox[((β[i] ≫ 24) ∧ 0xff)]
17: ¦end for
18: return β
```

6.3.2.3. bitshift fonksiyonu

Bu fonksiyon 32-bitlik kelimelerden oluşan blok ile çalışır. Bloğa ait bit topluluğunu sağdan sola olacak şekilde 15-bit kaydırır.

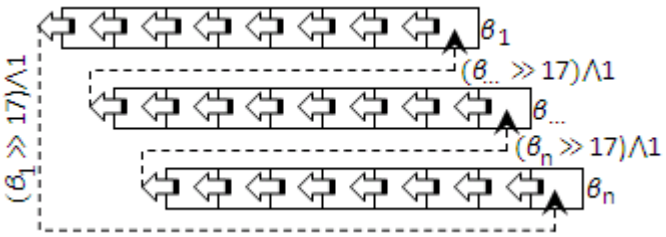
Burada *bitshift* fonksiyonu, β bit bloğu, *c(β)* bloğu oluşturan kelimelerin sayısını, *r* fonksiyonun tur sayısını temsil eder.

1. Önce β bloğunu oluşturan ilk kelime *β[0]*'ın 31., 30., …, 17. bitleri, 17-bit sağa kaydırılarak *a* değişkeninde saklanır.

2. Devamında $\beta[0]$ kelimesinden başlanarak, sırasıyla bütün kelimelerin 16., 15., …, 0. bitleri 15-bit sola kaydırılır ve bir sonraki değerleri bu kelimenin 16., 15., …, 0. bit aralığına taşınır.
3. Bütün kelimelerin bit taşıma işlemi sonuçlandığında ilk $\beta[0]$ değerinin 17-bit sağa kaydırılarak, a değerinde 16., 15., …, 0. bitler olarak saklanan bitler son kelime olan $\beta[c(\beta)-1]$ 'ye *eklenir.*

Bu dönüşüm sonucu blok içerisinde yer alan bitler kelimelerden bağımsız bir sonlu alan içerisinde kaydırılmış olur. i kelimenin blok içerisindeki sıra değerini, j kelimeyi oluşturan bitlerin sıra değerini ve $\beta[i,j]$ bloğu temsil ettiğinde bir tur sonucunda oluşan blok yapısı $\beta =$ $\{\{\beta[i,16],\dots,\beta[i,0]\},\{\beta[i+1,31],\dots,\beta[i,17]\},\dots,\{\beta[c(\beta)-1,16],\dots,\beta[c(\beta)-1,0]\},\ \{\beta[0,31],\dots,\beta[0,17]\}\}$ sıralamasına sahip olur.

Blok içerisindeki kelimelere ait bitlerin kaydırılması Şekil 27**:** ICDS - bitshift fonksiyonu ile temsil edilmiştir.



Şekil 27: ICDS - bitshift fonksiyonu

bitshift fonksiyonu Algoritma 3**:** ICDS - bitshift fonksiyonu ile verilmiştir.

Algoritma 3: ICDS - bitshift fonksiyonu

Inputs:	$\boldsymbol{\beta}[\boldsymbol{i}]$;// 32-bit uzunluğunda kelimelerden oluşan blok dizisi.
Variables:	$\boldsymbol{c}(\boldsymbol{\beta})$;// β dizisindeki kelime sayısı, dizinin eleman sayısı.
Outputs:	$\boldsymbol{\beta}[\boldsymbol{i}]$;// zigzag fonksiyonundan geçirilmiş $\boldsymbol{\beta}[\boldsymbol{i}]$ dizisi.

```
for i = 0 to (r - 1) do
    a = β[0] ≫ 17
    for j =  0 to (c(β) – 1) do
        β[j] = ( (β[j] ≪ 15)  ∨ (β[(j + 1) mod c(β)] ≫ 17) )
    end for
    β[c(β) - 1] = (β[c(β - 1)] ≪ 15)  ∨ a
end for
return β
```

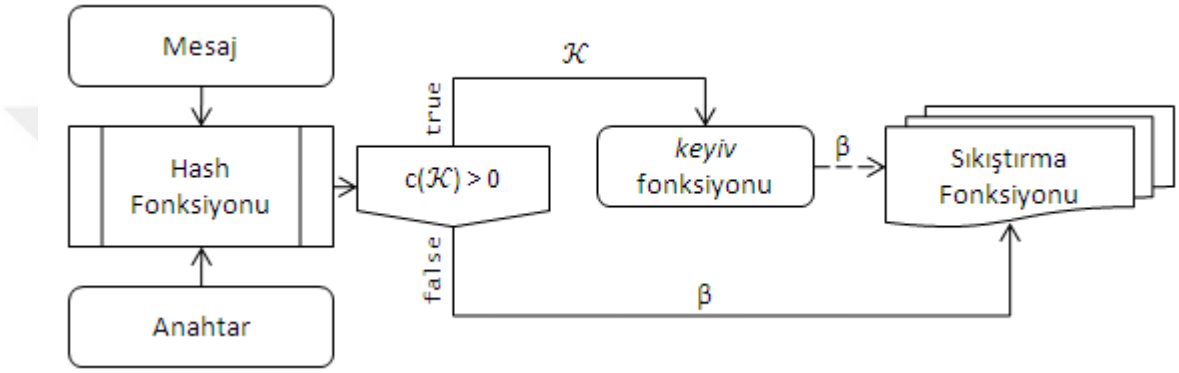
7.3.2.4. keyiv fonksiyonu

ICDS, MAC ve MDC yapısında çalışma özelliğine sahiptir. Bu özelliğini, mesaj özetinin hesaplandığı bloğun başlangıç değerlerini *keyiv* fonksiyonu ile değiştirebilmesiyle sağlar.

Özet değerlerinde bu özelliğin kullanılabiliyor olması ile aşağıda verilen yeterlilikler elde edilmiştir.

1. Adli bilimler alanında sayısal delilin özgül özet değerleriyle saklanabilir ve birden fazla kişi veya kurumca doğrulama kodu alınabilir.
2. Büyük veri için sınıflandırma ve imzalamada özgül özet değerleri üretilebilir.
3. Kötü amaçlı yazılım tespiti uygulamalarında kullanılabilir.
4. Sayısal imza algoritmalarında kullanılabilir.
5. Belirlenmiş mesaj değerlerinin özet değeri karşılıklarının yer aldığı saldırı amaçlı veri havuzlarına direnç sağlar.

Başlangıç değerleri bir anahtar dizisi olan $\mathcal{K}$ ile ana fonksiyona gönderilir ve ilk başlangıç değerleri yerine $\mathcal{K}$ dizisinden oluşturulan değerler β kelime dizisine atanır. Mesaj özetleme işleminde bu aşamadan sonra başlayarak farklılaştırılmış β bloğu ve $\mathcal{H}$ mesaj özet değeri hesaplanır.



Şekil 28: ICDS - Anahtar girişi

$\mathcal{K}$ anahtar dizisi 8-bitlik byte elemanlarından oluşur. *keyiv* fonksiyonuna herhangi bir uzunlukta giren 8-bitlik elemanlı $\mathcal{K}$ dizisi, 32-bitlik toplam *c(β)* elemanlı β dizisi olarak çıkış yapar. Burada $\mathcal{K}$ dizisi β dizisine işlenirken aynı zamanda anahtar genişletme işlemi de uygulanmış olur.

Algoritma 4: ICDS - keyiv fonksiyonuile verilmiştir.

Algoritma 4: ICDS - keyiv fonksiyonu

Inputs: $\boldsymbol{\beta}[\boldsymbol{i}]$;// 32-bit uzunluğunda kelimelerden oluşan blok dizisi.

$\boldsymbol{\mathcal{K}}[\boldsymbol{i}]$;// 8-bit uzunluğunda kelimelerden oluşan anahtar dizisi.

Variables: ***CA[i]*** ; // 8-bit uzunluğunda kelimelerden oluşan geçici dizi.

$\boldsymbol{a}$;// $\mathcal{K}$ dizisinin CA dizisine aktarımında kullanılacak tur sayısı.

$\boldsymbol{c}(\boldsymbol{\mathcal{K}})$;// $\mathcal{K}$ dizisinde bulunan 8-bitlik değerlerin sayısı.

$\boldsymbol{c}(\boldsymbol{\beta})$;// β dizisininde bulunan 8-bitlik değerlerin sayısı.

Outputs: $\boldsymbol{\beta}[\boldsymbol{i}]$;// $\mathcal{K}$ anahtar dizisine gore yeniden hesaplanmış blok.

```
01: a = 0;
02: if c(𝒦) > c(β) then
03: ¦ a = c(𝒦)
04: else if c(𝒦) ≤ c(β) then
05: ¦ a = c(β)
06: end if
07: CA[0] = c(𝒦) ∧ 255;
08: CA[1] = c(𝒦) ≫ 8;
09: CA[2] = sbox[CA[0]];
10: CA[3] = sbox[CA[1]];
11: CA[4] = sbox[CA[2] ⊕ CA[3]];
12: CA[5] = sbox[c(β) ∧ 255) ⊕ CA[4]];
13: for i = 0 to (a - 1) do
14: ¦ CA[i mod (4.c(β))] ⊕= 𝒦[i mod c(𝒦)]
15: end for
16: for i = 6 to ((4.c(β)) - 1) do
17: ¦ CA[i] ⊕= sbox[CA[i – 1]]
18: end for
19: for i = 6 to (c(β) - 1) do
20: ¦ β[i] = ( CA[(i.4)] ∨ (CA[(i.4)+1] ≪ 8 ) ∨ ⤶
    ¦ CA[(i.4)+2] ≪ 16 ) ∨ (CA[(i.4)+3] ≪ 24 ) )
21: end for
22: for i = 6 to (c(β) - 1) do
23: ¦ zigzag(β)
24: ¦ blockmix(β)
25: ¦ if (i mod 2 > 0) then
26: ¦ ¦ bitshift(β, ((β[0] mod 4)⊕(β[1] mod 4)⊕(β[2] mod 4)⊕(β[3] mod 4)) ⤶
    ¦ ¦ +1)
27: ¦ end if
28: end for
29: return β
```

6.3.2.5. cper fonksiyonu

Bu fonksiyon 32-bitlik bir kelime girişi *I'yı* halka olarak sola doğru 17-bit döndürme işleminden geçirir. Devamında, bitleri kaydırılan *I* kelimesinin bitleri 8'erli 4 ayrı grup olarak sbox tablosundaki karşılıklarıyla değiştirilir.

Bu işlem aşağıdaki formül ile verilmiştir:

$$cper(I) = \begin{cases} I = (I \ll 17) \vee (I \gg 15) \\ I = (sbox[\, I \wedge 0xff] \vee \wr \\ \quad (sbox[\, (I \gg 8) \wedge 0xff] \ll 8) \vee \wr \\ \quad (sbox[\, (I \gg 16) \wedge 0xff] \ll 16) \vee \wr \\ \quad (sbox[\, (I \gg 24) \wedge 0xff] \ll 24)\,) \end{cases} \quad (89)$$

6.3.2.6. Özet Hesaplama / Sıkıştırma Fonksiyonu

ICDS ile mesaj özetleme için üç giriş değeri vardır. Bu giriş değerleri ve özellikleri aşağıda Tablo XIX**:** ICDS - Özetleme için giriş parametreleri ile verilmiştir

Tablo XIX: ICDS - Özetleme için giriş parametreleri

Parametre	**Zorunlu**	**Tür**	**Açıklama**
HASHLEN	+	Değer	$HASHLEN \in \mathbb{Z}_{\infty}^{+}; 128 \leq HASHLEN < \infty;$ $HASHLEN \equiv 0\ (mod\ 32)$ olan ve mesaj özeti boyutunu temsil eden değer
$\mathcal{M}[i]$	+	Dizi	$\forall\, \mathcal{M}[i] \in \mathbb{Z}_{255}^{+}; 1 \leq i < \infty$ eşitliğinde ve elemanları 8-bitlik değerler olan mesaj dizisidir.
$\mathcal{K}[i]$	-	Dizi	$\forall\, \mathcal{K}[i] \in \mathbb{Z}_{255}^{+}; 1 \leq i < \infty$ eşitliğinde ve elemanları 8-bitlik değerler olan anahtar dizisidir.

Özetleme işleminde $\mathcal{M}$ mesaj dizisi ve *HASHLEN* özet uzunluğu değeri belirtilmesi zorunlu alanlarken $\mathcal{K}$ dizisinin boş olması fonksiyonun MDC yapısında çalışmasını sağlar.

ICDS için önerilen en kısa özet uzunluğu 256-bittir ve ön tanımlı değerdir. Ancak önerilmemesine karşın 128-bit uzunluğuna kadar kısaltılabilir uzunluk değerlerinde özetleme yapılabilir.

Özetleme işleminde öncelikle *HASHLEN* değerine bağlı olarak *len(β)* blok uzunluğu ve blok içerisinde yer alan 32-bitlik kelimelerin sayısı bulunur.

$$len(\beta) = \begin{cases} 512, & 512 > 2.HASHLEN \\ \left\{ \begin{matrix} len(\beta) = 512 \\ while\ len(\beta) < 2.HASHLEN\ do \\ \quad \{len(\beta) = 2.len(\beta) \end{matrix} \right\}, & 2.HASHLEN \geq 512 \end{cases} \quad (90)$$

$$c(\beta) = len(\beta)/32$$

Mesaj uzunluğu *len(𝓜)*, blok uzunluğu *len(β)* ve blok sayısı *BCOUNT* ile temsil edilir. mesaj dolgulama işleminde 8-bitlik parçalar halinde sbox tablosundan mod 256 denkliğinde değerler çağırılır.

ICDS, $\mathcal{M}$ mesajını 8-bitlik byte parçaları olarak bloğa yerleştirir, bu durumda $\mathcal{M}$ mesajının bit uzunluğunu temsil eden *len(𝓜)* için 8-bitlik elemanların sayısını temsil eden *c(𝓜)* aşağıdaki şekilde bulunur.

$$\begin{cases} c(\mathcal{M}) = len(\mathcal{M})/8 \\ n = c(\mathcal{M}) - 1 \\ M = M[0], \dots, M[n] \end{cases} \quad (91)$$

Mesaj özetinin hesaplanması mesajın, *len(β)*-bit uzunluğunda parçalar halinde, β bloğunda sıkıştırma fonksiyonlarından geçirilmesiyle gerçekleşir. Burada $len(\mathcal{M})$ mesaj uzunluğu için, β bloğunun kullanılacağı *BCOUNT* sayısı aşağıdaki şekilde bulunur.

```
if len(β) ≥ len(M) then
    BCOUNT = 1
else if len(β) < len(M) then
    BCOUNT = (len(M) – (len(M) mod len(β)) / len(β)
    if (len(M) mod len(β)) > 0) then
        BCOUNT = BCOUNT + 1
    end if
end if
```
(92)

ICDS, en kısa 128-bit uzunlukta mesaj özeti hesaplar ve bu 512-bitlik β bloğunda gerçekleşir. ICDS için mesaj uzunluğu ve özet uzunluğundan bağımsız olarak, toplamda 128-bit olan 4 ayrı 32-bitlik kelime değeri ön tanımlı olarak β bloğunun ilk 4 kelimesi olarak tanımlıdır.

```
β[0] = 0x61746f7a;      β[1] = 0x73657268;
β[2] = 0x6b626179;      β[3] = 0x62757261;
```
(93)

Yukarıda β bloğunun ilk kelime değerlerinin atanması veya varsa $\mathcal{K}$ anahtar dizisine göre değerler verilmesi sonunda ön işlem aşaması tamamlanmış olur.

Devam eden aşamada *BCOUNT* sayısınca tur ile mesaj bloklara bölünerek özet değeri $\mathcal{H}$ hesaplaması gerçekleştirilecektir.

Özet değeri $\mathcal{H}$, *BCOUNT* blok sayısınca tur ile hesaplanırken her döngü sonunda $\mathcal{H}$ özet değerini oluşturacak β blok değerleri iki ayrı tur döngüsüyle alt fonksiyonlardan geçirilir.

icds fonksiyonu Algoritma 5 ile verilmiştir.

Algoritma 5: ICDS - Özetleme Algoritması

Inputs:	$\boldsymbol{\mathcal{M}[i]}$ **;//** 8-bit uzunluğunda değerlerden oluşan, özet değeri hesaplanacak mesaj dizisi.
	$\boldsymbol{\mathcal{K}[i]}$ **;//** 8-bit uzunluğunda değerlerden oluşan, özet MAC yapısında özet hesaplanması için kullanılacak anahtar dizisi.
	$\boldsymbol{HASHLEN}$ **;//** $128 < HASHLEN < \infty$ ve 32'nin katları olmak üzere özet değeri uzunluğunu temsil eden değer.

Variables:	$\boldsymbol{c, b}$ **;//** β[i] kelime dizisinin permütasyonunda kullanılacak değişkenler.
	$\boldsymbol{BCOUNT}$ **;//** $\mathcal{H}$ özet değerinin hesaplanmasında kullanılacak blok sayısı.
	$\boldsymbol{len(\beta)}$ **;//** β bloğunun uzunluğu.
	$\boldsymbol{len(\mathcal{M})}$ **;//** $\mathcal{M}$ mesajının uzunluğu.
	$\boldsymbol{c(\beta)}$ **;//** β[i] dizisindeki 32-bitlik kelimelerin sayısı.
	$\boldsymbol{c(\mathcal{H})}$ **;//**$\mathcal{H}$[i]özet değeri dizisindeki 32-bitlik kelimelerin sayısı.
	$\boldsymbol{c(\mathcal{K})}$ **;//** $\mathcal{K}$[i] dizisindeki 8-bitlik kelimelerin sayısı.
	$\boldsymbol{c(\mathcal{M})}$ **;//** $\mathcal{M}$[i] dizisindeki 8-bitlik kelimelerin sayısı.
	$\boldsymbol{a,\ b}$ **;//** *a* $\mathcal{M}$ mesaj dizisi için sayaç, *b* bitshift fonksiyonu için geçici değeri temsil eden değerdir.
	$\boldsymbol{B[i]}$ **;//** β mesajın yerleştirileceği ve gerekirse dolgulama işleminin yapılacağı bloğu ve $\mathcal{H}$ özet değerinin hesaplanacağı diziyi temsil eder.

Outputs:	$\boldsymbol{\mathcal{H}}$ **;//** $\mathcal{M}$ mesajının özet değeri.

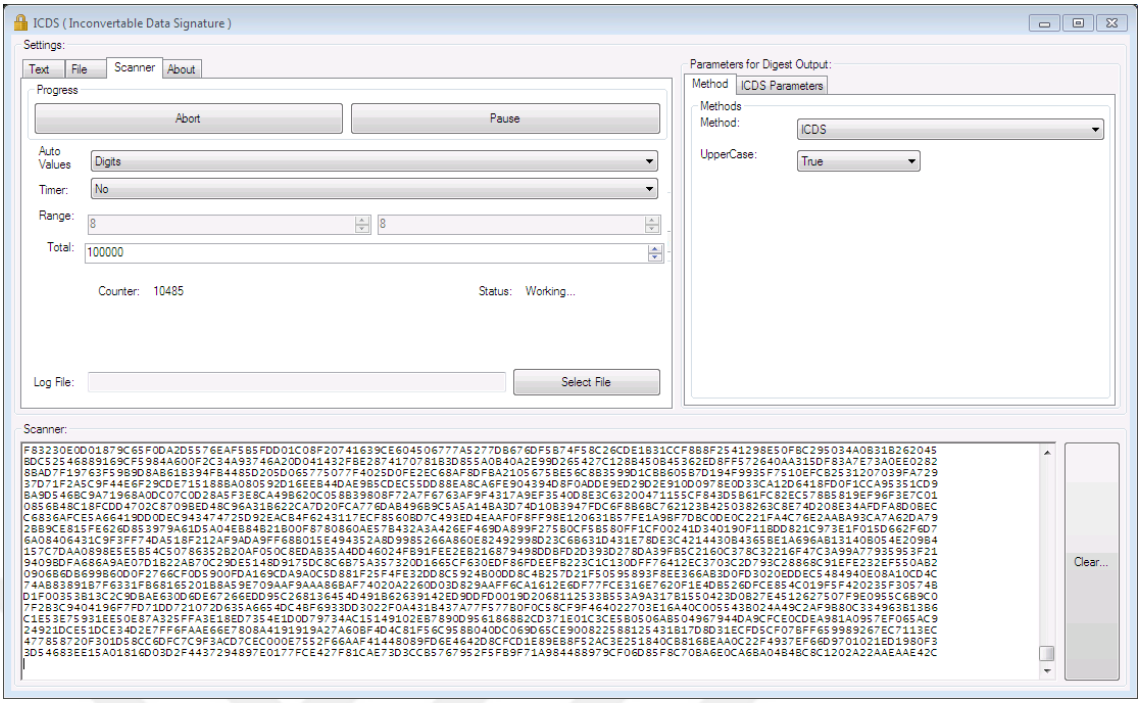
```
01: c(ℋ) = HASHLEN / 32;        c(ℳ) = len(ℳ)/8;
02: if (512 ≥ (2.HASHLEN)) then
03:     len(β) = 512
04: else if (512 < (2.HASHLEN)) then
05:     lenB = 512
06:     while (len(β) < (2.HASHLEN))
07:         lenB = 2* len(β)
08:     end while
09: cB = len(β) / 32
10: if (len(β) ≥ len(ℳ)) then
```

```
11:  ¦BCOUNT = 1
12:  else if (len(β) < len(𝓜)) then
13:  ¦BCOUNT = (len(𝓜) – (len(𝓜) mod len(β)) / len(β)
14:  ¦if (len(𝓜) mod len(β)) > 0) then
15:  ¦ ¦BCOUNT = BCOUNT + 1
16:  ¦end if
17:  end if
18:  β[0] = 0x61746f7a;        β[1] = 0x73657268;
19:  β[2] = 0x6b626179;        β[3] = 0x62757261;
20:  for i = 0 to (c(β) - 1) do
21:  ¦zigzag(β)
22:  ¦blockmix(β)
23:  end for
24:  if c(𝓚) > 0 then
25:  ¦keyiv(β, 𝓚);
26:  end if
27:  for i = 0 to (BCOUNT-1) do
28:  ¦for j = 0 to (c(β) - 1) do
29:  ¦ ¦β[j] ⊕= ( (c(𝓜) > a?𝓜[a]      : sbox[a mod 256])  ∨
     ¦ ¦        ((c(𝓜) > a+1 ? 𝓜[a+1]   : sbox[a+1 mod 256]) ≪ 8)   ∨
     ¦ ¦        ((c(𝓜) > a+2 ? 𝓜[a+2]   : sbox[a+2 mod 256]) ≪ 16) ∨
     ¦ ¦        ((c(𝓜) > a+3 ? 𝓜[a+3]   : sbox[a+3 mod 256]) ≪ 24) )
30:  ¦ ¦a = a + 4
31:  ¦end for
32:  ¦for j = 0 to 15 do
33:  ¦ ¦zigzag(β)
34:  ¦ ¦blockmix(β)
35:  ¦ ¦if (j mod 2) > 0 then
36:  ¦ ¦ ¦bitshift(β, ((β[0] mod 4) ⊕ (β[1] mod 4)  ⊕  (β[2] mod 4) ⊕
                          (β[3] mod 4)) + 1)
37:  ¦ ¦end if
38:  ¦ end for
39:  ¦b = 0;
40:  ¦for j = 0 to (c(𝓗) - 1)  do
41:  ¦ ¦H[j] ⊕= (β[b] ⊕ β[b+1])
42:  ¦ ¦b = b + 2
43:  ¦end for
44:  end for
45:  return 𝓗
```

6.4. Uygulama

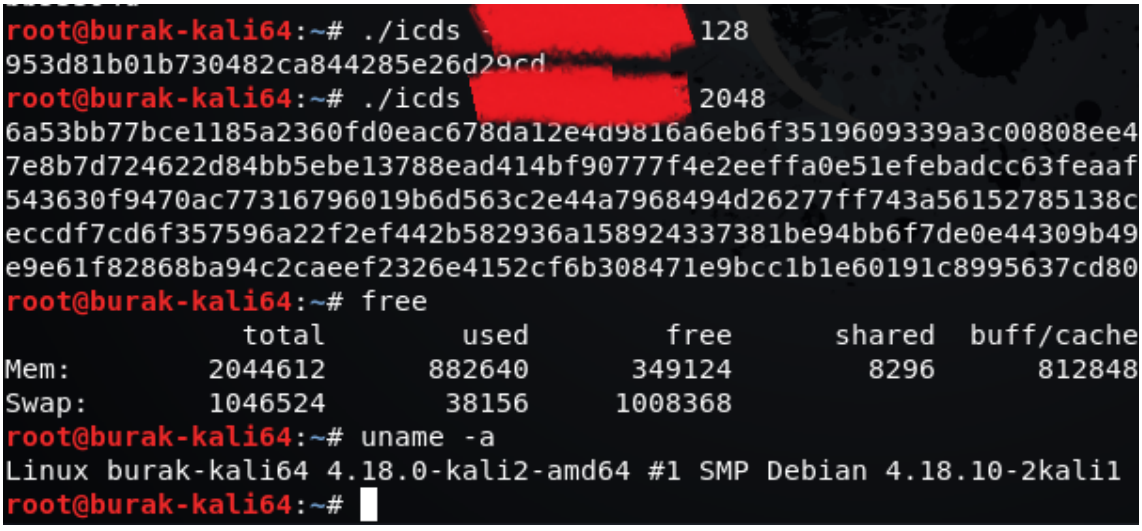
ICDS'in özet hesaplama yapısı farklı alt fonksiyonlara dağıltılmıştır. Bu fonksiyonlar sbox tablosunu kullanırlar ve tek boyutlu 32-bitlik kelime değerlerini 8-bitlik değerler olarak kullanmak için sanal bir ikinci boyut oluştururlar. Ancak bu ikinci boyut alanı 32-bit değerlerin 8-bitlik parçalara bölünmesiyle değil, 8-bitlik işlemlerden geçirilmesiyle sağlanır. Bu sanal hesaplama yöntemi ICDS'in her 8-bitlik değer için ayrı bir değişken ve işlemci kayıt alanı gereksinimi ortadan kaldırır. Bu $\frac{32}{8} = 4$ işlem sayesinde kayıt alanı maliyeti $1/4$ oranında azaltılmıştır.

ICDS'in geliştirilmesi 32-bitlik mikro işlemci mimarisinde büyük boyutlu özet değerlerinin hesaplanabilmesi için tasarlanmış ve oluşturulan algoritmanın donanım mimarisinden bağımsızlık oranının yüksek tutulması sağlanmıştır. ICDS kodlaması, bu çalışmanın yapıldığı tarihte birçok mikro işlemci mimarisinde ve mikro denetleyici kitlerinde standart olarak kullanılan ANSI C programlama diliyle yapılmış ve başarıyla uygulanmıştır. Oluşturulan ANSI C kodu işletim sistemi platformundan bağımsız olarak, windows ve linux platformlarında başarıyla çalıştırılmıştır. Aynı algoritmanın masaüstü uygulaması için Microsoft Visual Studio 2012 ve Microsoft .NET Framework 4.5.5 ile C# dilinde hazırlanan kodu da başarıyla çalışmış ve devam eden bölümlerdeki analiz uygulamaları bu kod ile gerçekleştirilmiştir. Şekil 29 ile ICDS algoritma ve analiz sürecinde kullanılan ara yüz görülmektedir.



Şekil 29: ICDS – Windows platformu uygulaması

Şekil 30 ile ANSI C dilinde kodlanan ICDS algoritmasının linux işletim sistemi üzerinde çalıştığı görülmektedir.



Şekil 30: ICDS - Linux plantformunda uygulama

6.5. Analizler

6.5.1. Güvenlik

ICDS diffusion etkilerini alt fonksiyonları ile gerçekleştirir. Burada *blockmix* blok içerisindeki değerleri etkili bir şekilde blok alanına dağıtır, *zigzag* blok içerisindeki verileri iki boyutlu sanal bir alanda XOR işleminden geçirir, *bitshift* bloğu bir bütün olarak en az 17-bit sola kaydırır ve *cper* 32-bitlik bir kelimenin bit sıralamasını değiştirerek sbox tablosu karşılıklarıyla değiştirir. Bu fonksiyonların bir bütün olarak oluşturduğu yapı blok içerisindeki değerlerin etkili olarak yayılmasını sağlarken aynı zamanda her değerin birbirine etki etmesini sağlar.

ICDS'in diffusion ve confusion analizi için detaylı ve istatistiksel yöntemler kullanılmıştır. Literatürde yaygın olarak kabul gören ve güvenilir kabul edilen analiz yöntemleri, burada ICDS ve referans olarak SHA ailesi için uygulanmıştır.

6.5.1.1. Diffusion ve Confusion

Diffusion ve confusion mesajın özet değerinde ortaya çıkardığı değişim sonucunu temsil eder. Burada mesaj değerindeki bir bitlik değişimin özet değerinde ortaya çıkardığı değişim analiz edilmiştir.

Bu analiz için toplam özet değeri uzunluğu L, test tekrar sayısı n, değişen bit sayısı $X = \sum_{i=0}^{n-1} X_i$, değişen bit sayısının aritmetik ortalaması $\overline{X} = X/n$, $\overline{X}$'in L'deki değişim oranı $\overline{X}P = \overline{X}/L$, değer kareli ortalaması $KO = \sqrt{\sum Xi^2 / n}$, KO'nın L'deki değişim oranı KP=KO/L olarak temsil edilmiştir.

$\mathcal{M}$ mesajındaki verilerin mesaj özetine etkisinin test edilmesi için aşağıdaki adımlar takip edilmiştir:

1. $8 \leq \mathcal{M} \leq 2048$ aralığında rastlantısal uzunlukta ve rastlantısal değerlerde $\mathcal{M}$ mesajı oluşturulmuş ve özet değeri hesaplanmıştır.
2. Orijinal mesajın rastlantısal seçilen bir konumuna 0 veya 1 bir biti eklenerek özet değeri hesaplanmıştır.
3. İki özet değeri karşılaştırılmış ve değişen bit değerleri X_i olarak temsil edilmiştir.
4. 1'den 3'e kadar olan adımlar 2048 defa tekrar edilmiştir.

Bu testten elde edilen veriler aşağıda Tablo XX ile verilmiştir.

Tablo XX: ICDS - Diffusion ve confusion testi

Algoritma	**L**	$\overline{X}$	$\overline{X}P$	**KO**	**KP**
ICDS32	128	60,977051	0,476383	61,612127	0,481345
ICDS32	160	77,368164	0,483551	79,414508	0,496341
SHA1	**160**	**76,460938**	**0,477881**	**78,429805**	**0,490186**
ICDS32	192	93,911621	0,489123	94,248251	0,490876
ICDS32	224	110,044922	0,491272	110,356317	0,492662
SHA	**224**	**108,615723**	**0,484892**	**110,539972**	**0,493482**
ICDS32	256	126,540527	0,494299	126,836753	0,495456
SHA	**256**	**123,550781**	**0,482620**	**126,008425**	**0,492220**
ICDS32	288	142,758301	0,495689	143,033245	0,496643
ICDS32	320	159,259277	0,497685	159,529611	0,498530
ICDS32	352	175,458496	0,498462	175,733192	0,499242
ICDS32	384	191,211426	0,497946	191,478177	0,498641
SHA	**384**	**140,264648**	**0,365273**	**164,281416**	**0,427816**
ICDS32	416	207,707520	0,499297	207,707520	0,499297
ICDS32	448	224,194824	0,500435	224,304951	0,500681
ICDS32	512	255,722168	0,499457	255,980957	0,499963

Algoritma	L	$\overline{X}$	$\overline{X}P$	KO	KP
SHA	**512**	**186,971191**	**0,365178**	**218,984300**	**0,427704**
ICDS32	1024	512,077148	0,500075	512,324252	0,500317
ICDS32	2048	1023,776367	0,499891	1024,033000	0,500016
ICDS32	4096	2047,124512	0,499786	2047,369854	0,499846

Tablo XX ile verilen $\overline{X}$ ve *KO* değerleri mesaj bitinde bir bitlik değer değişimi sonucunda mesaj özetinde değişen bit sayısının ortalamalarını temsil eder. $\overline{X}$P ve KP değerleri değişimi gözlemlenen bitlerin, toplam L uzunluğuna sahip mesaj özeti değerindeki değişimin yüzdelik oranını temsil eder.

Analiz sonucu elde edilen verilere göre ICDS mesaj verisindeki tek bitlik değişim sonucu özet değerinde büyük farklılıklar oluşturur.

Referans olarak SHA ailesine aynı koşullar ve mesajlar ile aynı analizler gerçekleştirilmiş ve SHA ailesi üyelerinin de benzer değerler elde ettiği görülmüştür.

SHA ailesi için alınan sonuçlar ve ICDS için alınan sonuçlar karşılaştırıldığında ICDS'in yüksek değerler elde ettiği görülmüştür.

Yapılan bu analiz ICDS'in mesajdaki en küçük değişiklikte özet değerinde büyük değişimler oluşturduğunu göstermiştir. Referans olarak aynı testlerin uygulandığı SHA ailesinin eş mesaj boyutlu algoritmalarından bulunan değerleri, ICDS için bulunan değerlerden düşük sonuçlar vermiştir. Bu değerlere göre ICDS, güvenlik kriteri olan diffusion ve confusion için yüksek yeterlilik sağlamaktadır.

6.5.1.2. Hassasiyet

Bu bölümde ICDS'in mesajdaki ekleme, azaltma, arttırma ve eksiltme değişimlerinin hepsini kapsayan bir test yapılarak, mesaj değerlerine gösterdiği hassasiyet analiz edilmiştir.

Bu analiz için toplam özet değeri uzunluğu L, test tekrar sayısı n, değişen bit sayısı $X = \sum_{i=1}^{4} \sum_{j=0}^{n-1} X_{i,j}$, değişen bit sayısının aritmetik ortalaması $\overline{X} = X/(4.n)$, $\overline{X}$ 'in L'deki değişim oranı $\overline{X}P = \overline{X}/L$, değer kareli ortalaması $KO = \sqrt{\sum Xi,j^2 \ / \ 4.n}$, KO'nın L'deki değişim oranı KP=KO/L olarak temsil edilmiştir.

$\mathcal{M}$ mesajındaki verilerin mesaj özetine etkisinin test edilmesi için aşağıdaki adımlar takip edilmiştir:

1. $8 \leq \mathcal{M} \leq 2048$ aralığında rastlantısal uzunlukta ve rastlantısal değerlerde $\mathcal{M}$ mesajı oluşturulmuş ve özet değeri hesaplanmıştır.
2. Orijinal mesajın rastlantısal seçilen bir herhangi bir 0 biti 1 olarak değiştirilmiş ve özet değeri hesaplanmıştır.
3. Orijinal mesajın rastlantısal seçilen bir herhangi bir 1 biti 0 olarak değiştirilmiş ve özet değeri hesaplanmıştır.
4. Orijinal mesajın ilk bit değeri silinerek özet değeri hesaplanmıştır.
5. Orijinal mesajın rastlantısal seçilen bir konumuna, rastlantısal olarak seçilen 1 veya 0 biti eklenerek özet değeri hesaplanmıştır.
6. Özet değerleri karşılaştırılmış ve değişen bit değerleri sayılmıştır.
7. 1'den 6'e kadar olan adımlar 2048 defa tekrar edilmiştir.

Bu testten elde edilen veriler aşağıda Tablo XXI ile verilmiştir.

Tablo XXI: ICDS – Hassasiyet testi

Algoritma	L	$\overline{X}$	$\overline{X}P$	KO	KP
ICDS32	128	61,658447	0,060213	61,802449	0,060354
ICDS32	160	78,005371	0,060942	78,116620	0,061029
SHA1	**160**	**77,494141**	**0,060542**	**77,868805**	**0,060835**
ICDS32	192	94,359985	0,061432	94,455723	0,061495
ICDS32	224	110,402466	0,061609	110,494808	0,061660
SHA	**224**	**109,167358**	**0,060919**	**109,585914**	**0,061153**
ICDS32	256	126,659180	0,061845	126,748888	0,061889
SHA	**256**	**124,308350**	**0,060697**	**124,870429**	**0,060972**
ICDS32	288	142,840454	0,061997	142,928364	0,062035
ICDS32	320	159,066284	0,062135	159,151036	0,062168
ICDS32	352	175,325928	0,062261	175,409818	0,062290
ICDS32	384	191,385742	0,062300	191,463742	0,062325
SHA	**384**	**153,013184**	**0,049809**	**157,844004**	**0,051382**
ICDS32	416	207,552246	0,062365	207,628530	0,062388
ICDS32	448	223,960854	0,062489	223,998513	0,062500
ICDS32	512	255,326294	0,062336	255,417767	0,062358
SHA	**512**	**205,645874**	**0,050207**	**211,944101**	**0,051744**
ICDS32	1024	511,761841	0,062471	511,862613	0,062483
ICDS32	2048	1023,452148	0,062467	1023,565883	0,062474
ICDS32	4096	2044,480713	0,062393	2044,980202	0,062408

Tablo XXI ile verilen $\overline{X}$ ve *KO* değerleri mesaj bitinde bir bitlik değer değişimi sonucunda mesaj özetinde değişen bit sayısının ortalamalarını temsil eder. $\overline{X}$P ve KP değerleri değişimi gözlemlenen bitlerin, toplam L uzunluğuna sahip mesaj özeti değerindeki değişimin yüzdelik oranını temsil eder.

Analiz sonucu elde edilen verilere göre ICDS mesaj verisindeki tek bitlik değişim sonucu özet değerinde büyük farklılıklar oluşturur.

Referans olarak SHA ailesine aynı koşullar ve mesajlar ile aynı analizler gerçekleştirilmiş ve SHA ailesi üyelerinin de benzer değerler elde ettiği görülmüştür.

Yapılan bu analiz ICDS'in mesajdaki en küçük değişiklikte özet değerinde büyük değişimler oluşturduğunu göstermiştir. Referans olarak aynı testlerin uygulandığı SHA ailesinin eş mesaj boyutlu algoritmalarından bulunan değerleri, ICDS için bulunan değerlerden düşük sonuçlar vermiştir. Bu değerlere göre ICDS, güvenlik kriteri olan diffusion ve confusion için yüksek yeterlilik sağlamaktadır.

6.5.1.3. Çakışma Analizi

Çakışmafarklı mesajların aynı özet değerini vermesidir. Matematiksel teoride bu zorluk, özetleme algoritmasınun sağladığı çakışma direnci kapasitesi anlamındadır. Çakışma direncini doğrudan ve tam olarak ölçebilen bir analiz aracı bulunmamaktadır. Ancak literatürde bu direncin analiz edildiği sıkça başvurulan bir yöntem burada da kullanılmıştır.

Bu analiz için toplam özet değeri uzunluğu L, test tekrar sayısı n, çakışan byte sayısı $X = \sum_{i=0}^{n-1} X_i$, çakışan byte sayısının aritmetik ortalaması $\overline{X} = X/n$, $\overline{X}$'in L'deki değişim oranı $\overline{X}P = \overline{X}/L$, kareli ortalaması $KO = \sqrt{\sum Xi^2 \,/\, n}$, KO'nın L'deki değişim oranı KP=KO/L olarak temsil edilmiştir.

Bu test için aşağıdaki adımlar takip edilmiştir:

1. $8 \leq \mathcal{M} \leq 2048$ aralığında rastlantısal uzunlukta ve rastlantısal değerlerde $\mathcal{M}$ mesajı oluşturulmuş ve özet değeri hesaplanmıştır.
2. Orijinal mesajın rastlantısal seçilen bir konumuna, rastlantısal olarak seçilen 1 veya 0 biti eklenerek özet değeri hesaplanmıştır.

3. Elde edilen iki özet değerindeki byte karakterlerden aynı konum ve değere sahip olanların sayısı X_i olarak temsil edilmiştir
4. 1'den 3'e kadar olan adımlar 2048 defa tekrar edilmiştir.

Bu testten elde edilen veriler aşağıda Tablo XXII ile verilmiştir.

Tablo XXII: ICDS – Çakışma testi

Algoritma	**L**	$\overline{X}$	$\overline{X}P$	**KO**	**KP**
ICDS32	128	11,488281	0,089752	16,965498	0,132543
ICDS32	160	12,576172	0,078601	16,031463	0,100197
SHA1	**160**	**16,507813**	**0,103174**	**35,113432**	**0,219459**
ICDS32	192	13,974609	0,072784	16,222814	0,084494
ICDS32	224	15,769531	0,070400	17,731945	0,079160
SHA	**224**	**20,230469**	**0,090315**	**41,641326**	**0,185899**
ICDS32	256	17,417969	0,068039	19,408439	0,075814
SHA	**256**	**24,412109**	**0,095360**	**50,801160**	**0,198442**
ICDS32	288	18,919922	0,065694	20,815597	0,072276
ICDS32	320	20,976563	0,065552	22,817551	0,071305
ICDS32	352	23,199219	0,065907	25,112870	0,071343
ICDS32	384	24,759766	0,064479	26,664437	0,069439
SHA	**384**	**120,730469**	**0,314402**	**200,375429**	**0,521811**
ICDS32	416	26,644531	0,064049	28,489856	0,068485
ICDS32	448	27,980347	0,062456	28,822673	0,064336
ICDS32	512	32,173828	0,062840	34,005859	0,066418
SHA	**512**	**161,367188**	**0,315170**	**267,385285**	**0,522237**
ICDS32	1024	63,714844	0,062222	65,554843	0,064018
ICDS32	2048	128,324219	0,062658	130,227925	0,063588
ICDS32	4096	256,121094	0,062530	257,918137	0,062968

Tablo XXII ile verilen $\overline{X}$ ve *KO* değerleri mesaj bitine bir bitlik değer eklenmesi sonucunda mesaj özetinde çakışan byteların sayısının ortalamalarını temsil eder. $\overline{X}$P ve KP değerleri çakışmaları gözlemlenen byteların, toplam L uzunluğuna sahip mesaj özeti değerindeki değişimin yüzdelik oranını temsil eder.

Burada $\overline{X}$ ve KO değerlerinin düşük olması daha az çakışma gerçekleştiğini gösteren ve istenen durumdur. $\overline{X}$P ve KP değerlerinin yüksek olması daha fazla çakışma gerçekleştiğini gösteren ve istenmeyen durumdur.

Analiz sonucu elde edilen verilere göre ICDS mesaj verisinde tek bitlik değişim sonucu özet değerinde büyük farklılıklar oluşturur ve daha az çakışan değerlere sebep olur.

Referans olarak SHA ailesine aynı koşullar ve mesajlar ile aynı analizler gerçekleştirilmiş ve SHA ailesi üyelerinin de benzer değerler elde ettiği görülmüştür.

Yapılan bu analiz ICDS'in mesajdaki en küçük değişiklikte özet değerinde büyük değişimler oluşturduğunu ve daha az çakışan değer oluşturduğunu göstermiştir. Referans olarak aynı testlerin uygulandığı SHA ailesinin eş mesaj boyutlu algoritmalarından bulunan değerleri, ICDS için bulunan değerlerden daha fazla çakışmayı gösteren sonuçlar vermiştir.

6.5.1.4. UACB – Unified Average Changing Bytes

Literatürde sıkça başvurulan UACI, test yöntemi image kriptosistem yöntemlerinin analizinde kullanılan ve 8-bitlik pixel değerlerinin ölçümüne dayalı bir yöntemdir (36-38). Ancak bu test yönteminin 8-bitlik pixellere odaklı olması aynı testlerin aynı bit uzunluğuna sahip byte ölçümleri içinde kullanılmasını mümkün kılar. UACB değerinin yüksek olması değişim oranının yüksek olduğunu gösterir ve istenilen

sonuçtur. Burada a orijinal ve b düzenlenmiş mesaja ait özet değeri, i özet değerindeki 8-bitlik elemanları temsil ettiğinde UACB $\frac{1}{L/8}\left(\sum_{i=0}^{L-1}\frac{|a(i)-b(i)|}{255}\right)$ olarak hesaplanır.

Bu test için için aşağıdaki adımlar takip edilmiştir:

1. $8 \leq \mathcal{M} \leq 2048$ aralığında rastlantısal uzunlukta ve rastlantısal değerlerde $\mathcal{M}$ mesajı oluşturulmuş ve özet değeri hesaplanmıştır.
2. Orijinal mesajın rastlantısal seçilen bir bit değeri değiştirilerek özet değeri hesaplanmıştır.
3. 1. ve 2. adımlar 2048 defa tekrar edilmiştir.

Bu testten elde edilen veriler aşağıda Tablo XXII ile verilmiştir.

Tablo XXIII: ICDS – UACB testi

Algoritma	**L**	**UACB**	**Algoritma**	**L**	**UACB**
ICDS32	128	0,237793	ICDS32	352	0,249868
ICDS32	160	0,240671	ICDS32	384	0,249196
SHA1	**160**	**0,239676**	**SHA**	**384**	**0,182260**
ICDS32	192	0,245183	ICDS32	416	0,250169
ICDS32	224	0,246262	ICDS32	448	0,250132
SHA	**224**	**0,242791**	ICDS32	512	0,249738
ICDS32	256	0,246515	**SHA**	**512**	**0,182680**
SHA	**256**	**0,241892**	ICDS32	1024	0,249890
ICDS32	288	0,247708	ICDS32	2048	0,249930
ICDS32	320	0,249449	ICDS32	4096	0,250108

Tablo XXIII ile verilen UACB değerleri mesaj bitinde bir bitlik değer değişiminin mesaj özetinde oluşturduğu değişimi temsil eden değerlerdir.

Analiz sonucu elde edilen verilere göre ICDS mesaj verisinde tek bitlik değişim sonucu özet değerinde yüksek değerli UACB değeri oluşturmaktadır. Referans olarak SHA ailesine aynı koşullar ve mesajlar ile aynı analiz gerçekleştirilmiş ve SHA ailesi üyelerinin de benzer değerler elde ettiği görülmüştür.

Yapılan bu analiz ICDS'in mesajdaki en küçük değişiklikte özet değerindeki yüksek değişim değerine sahip olduğunu göstermiştir. Referans olarak aynı testlerin uygulandığı SHA ailesinin eş mesaj boyutlu algoritmalarından bulunan değerleri, ICDS için bulunan değerlerden daha küçük UACB sonuçları vermiştir.

6.5.1.5. Korelâsyon

Korelâsyon analizi, en genel anlamıyla değişkenler arasındaki ilişkilendirme derecesini ölçme yöntemidir. Kriptografide korelâsyon analizinden elde edilen sonuçlar, kriptografik yönteme yapılacak analiz saldırıları için çok önemli bilgiler sağlayabilmektedir. Bu sebeple kriptosistem yönteminin bütün giriş değerlerinin çıkış değerleri üzerindeki etkisinin saklanması önemlidir (10, 30, 50-52).

Bu analizin yapılması için Pearson Korelâsyon yöntemi kullanılmıştır. a orijinal ve b düzenlenmiş mesaja ait özet değeri olduğunda, s_a a'nın standart sapma değeri, s_b b'nin standart sapma değeri, m_a a'nın aritmetik ortalama değeri ve m_b b'nin aritmetik ortalama değeri olmak üzere r korelâsyonu $r = \frac{\sum_{i=0}^{L-1}(a_i - m_a)(b_i - m_b)}{L.s_a.s_b}$ formülü ile hesaplanmıştır.

Bu test için aşağıdaki adımlar takip edilmiştir:

1. $8 \leq \mathcal{M} \leq 2048$ aralığında rastlantısal uzunlukta ve rastlantısal değerlerde $\mathcal{M}$ mesajı oluşturulmuş ve özet değeri hesaplanmıştır.

2. Orijinal mesajın rastlantısal seçilen bir bit değeri değiştirilerek özet değeri hesaplanmıştır.
3. 1. ve 2 . adımlar 2048 defa tekrar edilmiştir.

Bu testten elde edilen veriler aşağıda Tablo XXIV ile verilmiştir.

Tablo XXIV: ICDS – Korelâsyon testi

Algoritma	**L**	**r**	**Algoritma**	**L**	**r**
ICDS32	128	0,045000	ICDS32	352	0,002998
ICDS32	160	0,031550	ICDS32	384	0,004164
SHA1	**160**	**0,044248**	**SHA**	**384**	**0,269490**
ICDS32	192	0,020868	ICDS32	416	0,001312
ICDS32	224	0,016917	ICDS32	448	-0,000853
SHA	**224**	**0,030449**	ICDS32	512	0,001069
ICDS32	256	0,010957	**SHA**	**512**	**0,269669**
SHA	**256**	**0,034846**	ICDS32	1024	-0,000117
ICDS32	288	0,008407	ICDS32	2048	0,000211
ICDS32	320	0,004409	ICDS32	4096	0,000426

Tablo XXIV ile verilen r değerleri mesaj bitinde bir bitlik değer değişiminin mesaj özetinde oluşturduğu değişimi temsil eden değerlerdir. Analiz sonucu elde edilen verilere göre ICDS mesaj verisinde tek bitlik değişim sonucu elde edilen iki özet değerindeki ilişkilendirme değeri anlamsız olmaktadır. Referans olarak SHA ailesine aynı koşullar ve mesajlar ile aynı analiz gerçekleştirilmiş ve SHA ailesi üyelerinin de benzer değerler elde ettiği, ancak SHA 512 ve SHA1 için r korelâsyon değerinin anlamlandırılmaya yatkın yükselişleri görülmüştür.

Yapılan bu analiz ICDS'in mesajdaki en küçük değişiklikte özet değerindeki yüksek değişim değerine sahip olduğunu ve elde edilen özet değerlerinin istatistiksel olarak ilişkilendirilemeyeceğini göstermiştir. Referans olarak aynı testlerin uygulandığı SHA ailesinin eş mesaj boyutlu algoritmalarından bulunan değerleri, ICDS için bulunan değerlerden daha büyük ilişkilendirme değerleri vermiştir.

6.4.2. Performans

Performans testlerinde ICDS için Microsoft Visual Studio 2012 ve Microsoft .NET Framework 4.5.5 ile C# dilinde hazırlanan kodu kullanılmıştır. Burada ölçü birimi olarak bilgisayar bilimlerinde sıkça başvurulan ticks değeri kullanılmıştır. Bu değer, bir görevin tamamlanması sürecinde oluşmaktadır.

Bu test için aşağıdaki adımlar takip edilmiştir:

1. $8 \leq \mathcal{M} \leq 2048$ aralığında rastlantısal uzunlukta ve rastlantısal değerlerde $\mathcal{M}$ mesajı oluşturulmuş ve özet değeri hesaplanmıştır.
2. 1. adım 2048 defa tekrar edilmiştir.

Bu testten elde edilen veriler aşağıda Tablo XXV ile verilmiştir.

Tablo XXV. ICDS – Performans testi

Algoritma	**L**	**Ticks**	**Algoritma**	**L**	**Ticks**
ICDS32	128	61713	ICDS32	352	57122
ICDS32	160	58232	ICDS32	384	59377
SHA1	**160**	**62866**	**SHA**	**384**	**61247**
ICDS32	192	61918	ICDS32	416	58671
ICDS32	224	58602	ICDS32	448	58974
SHA	**224**	**58627**	ICDS32	512	61781
ICDS32	256	59558	**SHA**	**512**	**61572**
SHA	**256**	**61924**	ICDS32	1024	60396
ICDS32	288	57621	ICDS32	2048	62479
ICDS32	320	57566	ICDS32	4096	58783

Tablo XXV ile verilen ticks değerleri, ICDS'in hesaplama performansının yüksek olduğunu ve hızlı yanıtlar verebildiğini göstermektedir. Referans olarak SHA ailesine aynı koşullar ve mesajlar ile aynı analiz gerçekleştirilmiş ve SHA ailesi üyelerinin de benzer değerler elde ettiği, yalnızca 512-bit L değerinde ICDS'den daha iyi performans gösterdiği görülmüştür.

Yapılan bu analiz ICDS'in yüksek zaman performansı ve düşük donanım yükü ile hesaplama işlemini gerçekleştirdiğini göstermektedir.

6.4.3. Tartışma

SHA ailesi yaygın literatür ve kurumlarca başvurulan en güvenilir olarak kabul edilen mesaj özetleme algoritmalarından oluşur. Genel kabulde ve yayınlarda sahip

olduğu bu durum sebebiyle SHA ailesi bu testler için referans alınan algoritmalar olmuştur.

Mesajdaki en küçük değişimlerin kaotik özetler ile sonuçlanması kripto saldırılarında analistin bu verileri anlamlandırmasına karşı direnç sağlar. Ayrıca saldırganın mesaja hesaplanabilir eklemeler yaparak aynı özet değerini elde etmesine karşı direnç sağlanması gerekir. Burada gerçekleştirilen analizlerde elde edilen sonuçlar mesajdaki değişimlerin özet değerine etkisinin gizlenebildiğini ve saldırılara karşı dayanıklılık için kriterlerin yerine getirildiğini göstermiştir.

ICDS'e referans olarak aynı mesajlar ve koşullarda analizlerin yapıldığı SHA ailesinin büyük değerlerde sonuçlar verdiği, ICDS'in ise bütün mesaj uzunluklarında SHA ailesinden daha büyük değişimler sağladığı ve daha iyi sonuçlar verdiği görülmüştür. Bu test sonuçları ICDS'in genel kabul görmüş ve en güvenilir kabul edilen SHA ailesinden daha etkili güvenlik ve özetleme sağladığını göstermiştir.

Diffusion ve confusion testlerinde elde edilen değerlere çok yakın değerler elde edilen hassasiyet testlerinde ICDS'in yüksek değerler elde ettiği görülmüştür. Aynı testlerin SHA ailesine, aynı koşul ve mesaj değerleriyle uygulanması sonucu elde edilen değerler de ICDS'in halen daha yüksek değerler elde ettiği görülmüştür.

Çakışma direnci ile ilgili literatürde kesin sonuçlar veren bir yöntem ve araç bulunmamaktadır. Bu durum sebebiyle *"...çakışma direnci vardır..."* çıkarımının bu test ile elde edilebilmesi bilimsel dayanaktan uzak olacaktır. Ancak literatürde sıkça başvurulan analiz yöntemleri, bu araştırmadan yararlanacak olanlar için faydalı bilgiler sunabileceği gerekçesiyle gerçekleştirilmiştir. Bu testin uygulanması sonucu ICDS düşük çakışma sonuçları gösterirken aynı testin SHA ailesine, aynı koşul ve mesaj

değerleriyle uygulanması sonucu elde edilen değerler ICDS'in daha az çakışmaya sebep olduğu görülmüştür.

Korelâsyon analizi bir yöntem hakkında birçok bilgi edinilmesini sağlar. Burada ICDS için elde edilen korelâsyon değerlerinin çok düşük olduğu görülmüştür. Bu iki farklı mesaj değeri oluşturularak, mesaj özetinde kontrollü bir değer elde edilmesini ve bu özet değerlerinden mesaja veya yönteme dair bilgi edilmesine direnç sağlar. Diğer testler gibi aynı testin SHA ailesine, aynı koşul ve mesaj değerleriyle uygulanması sonucu elde edilen değerler ICDS'in güvenliğini gösteren daha düşük anlamlandırma değerleri elde ettiğini göstermiştir

Algoritmaların performans başarımları bilgisayar programları olarak kodlanmaları aşamasında büyük oranda etkilenmektedir. Bu sebeple analizlerde performans analizlerinin algoritma haricinde programlama metodu ve derleme yöntemine bağlı olduğu gerçeğiyle, açık bir performans yorumunun bilimsel dayanaklar ile yapılması mümkün olmamaktadır.

7. CUBE

7.1. Giriş

Bölüm 3. KRİPTOGRAFİ ve KRİPTOSİSTEMLER başlığı altında kriptosistemler detaylı ve uygulamalı olarak incelenmiştir.

Kriptosistemler temel olarak matematiksel yapıda olmalarıyla birlikte alanlarını aşkın olarak birçok farklı alana kaynak oluşturmaktadırlar.

Asimetrik şifreleme literatürü bu güne kadar anahtarın asimetrik olmasını temel almıştır. Ancak Cube asimetrisini şifreli metinde gerçekleştiren yeni bir yapı sunar. Cube yapısı düz metni iki boyutlu matris veya tek boyutlu dizi yerine üç boyutlu alandaki altı yüzlü cubie yüzlerine yerleştirir ve şifreleme işlemini bu cubieler ile oluşturduğu blok yapısıyla gerçekleştirir.

Bu çalışma kapsamında tasarlanan Cube, Rubik küpünün yapısından esinlenerek geliştirilmiştir. Elde edilen yapı kriptografide asimetrik ve simetrik alanlarının ikisini de temsil etmektedir.

Cube yönteminde düz metnin sayısal değerleri her şifreleme işlemi sonunda farklı sonuçlarda olur. Bu aynı düz metin ve aynı anahtar için her şifreleme tekrarında farklı şifreli metin çıkışı olarak gerçekleşir.

Cube'ün asimetrik-simetrik olarak sınıflandırılan kriptografi alanında iki alanında özelliklerini taşıması sebebiyle, asimetrik alanda asimetrik şifreli metin ile yeni bir alt alanda ve simetrik alanda çok boyutlu blok yapısıyla blok yapı alanında yer alır.

Cube, adli bilimlerde kullanılabilmesini sağlayacak şekilde tasarlanmıştır. Bu tasarımın geliştirilmesinde bölüm 1.4.3. Şüpheli / Sanık Tarafında Delillere Erişim Hakkı ile anlatılan Türk Ceza Mahkemesi Kanunun 134. maddesinde yer alan "*(4)*

Üçüncü fıkraya göre alınan, yedekten bir kopya çıkarılarak şüpheliye veya vekiline verilir ve bu husus tutanağa geçirilerek imza altına alınır."[49] hükmü ve bölüm 1.4.4. Mağdur Hak ve Mahremiyeti altında yer alan açıklamalar temel alınmıştır. Özel hayatın istismarı başta olmak üzere, bir yargılamaya esas sayısal delilin şüpheli-sanığa doğrudan bir kopyası verilmesi halinde suç ve mağduriyetin tekrarlanması riski karşısında, delillerin müşterekliği ve savunma haklarının evrensel hukukun temel zorunluluklarıdır. Türk Ceza Mahkemesi Kanunun ilgili maddesiyle karşılığını bulan bu durum evrensel haklar gereği bütün ülkeler için aynı durumdadır. Bu delillerin şüpheli-sanığa verilmesinin engellenemeyeceği durumlarda bu sayısal delillerin kaynak takibini sağlayabilecek ve hukuk dışı kullanımları durumunda failin tespitine yönelik yapılarak dönüştürülmesi Cube'ün ilk tasarım amacını oluşturmuştur. Yine adli alan için sayısal delillerin saklanması veya incelenmesinde, ilgili kurum ve/veya personelin bu delili kendilerine özgü anahtarlar ile imzalamalarına yönelik öznel kaynakların olmaması diğer bir çalışma amacını oluşturmuştur. Sayısal delilin her inceleme sonrasında yada kuruma-personele, aslı bozulmaksızın özgül yapıya dönüştürülerek saklanması alan için önemli bir gereksinim olarak varlığını sürdürmektedir. Bu yapıyı sağlayan Cube verinin aslı bozulmaksızın özgül bir yapıya dönüştürülmesini ve ilgilisine teslimine imkân sağlar.

Cube tek bir veri yığını için seçilebilir 256 ayrı yapıda dönüşüm sağlar. Bu 256 ayrı yapı Cube özgül şifreleme yöntemiyle sağlanır. Bu yapıların ilk hali her zaman aynı yapıya dönüşmesi sebebiyle mesaj özetleme fonksiyonlarında aynı mesaj özet değeri elde edilebilirken Cube ile elde edilen 256 ayrı yapı tamamen farklı 256 ayrı mesaj özet değerine sahip olur.

49 Değişik, 21.02.2014 / 6526

Cube ile elde edilen yapı sadece adli bilimler alanına değil, bilgi güvenliği alanına işaret eden yeni bir şifreleme yöntemidir. Çalışma devamında analizler ile gözlemleneceği gibi bilgi güvenliği alanında büyük blok boyu ve yeni saldırı direnci yöntemlerini sağlar.

Cube'ün bu yapısının geliştirilmesi aşağıdaki incelemeler yapılmıştır:

1. Mevcut şifreleme yöntemleri kullanım alanları ve algoritmaları incelenmiştir (3. KRİPTOGRAFİ ve KRİPTOSİSTEMLER).
2. En çok bilinen şifreleme yöntemlerinin zayıf yönleri ve saldırı yöntemleri incelenmiştir.
3. Kriptografinin adli bilimlerdeki kullanım alanları ve yorumlanması incelenmiştir.
4. Delil Kavramının ve Sayısal Delillerin yorumlanması incelenmiştir (1.4. Ceza Yargılamasında Deliller ve Bilirkişilik).

Bu incelemeler neticesinde bu bölümde aşağıdaki anlatımlar gerçekleştirilmiştir.

1. Adli bilimlerde delil bütünlüğünün ve müşterekliğinin bozulmaksızın mağdur haklarının korunmasına yönelik gereksinimler tespit edilmiştir.
2. Adli bilimlerde ilgili kurum ve personel gereksinimleri ve şifreleme yöntemiyle bu yönde sağlanabilecek gereksinimler tespit edilmiştir.
3. Cube'ün esnek yapıda olması ve farklı alanlara kaynak olabilmesi için gereksinimler tespit edilmiştir.
4. Güvenlik, hız ve donanım gereksinimleri incelenmiştir.
5. Yukarıda yer alan gereksinimlerin tespiti sonrası modüler olarak matematiksel algoritma ve yapılar tasarlanmıştır.

6. Tasarlanan algoritmalar bilinen kriptografik saldırılar ve daha güçlü saldırılar altında sınanmıştır.
7. Modüler algoritmalar adli bilimler ve diğer ilgili alanlara esnetilebilecek bir algoritma ile bir araya getirilmiştir.
8. Geliştirilen kriptosistemin bir bütün olarak alan uygulanabilirliği sınanmıştır.
9. Geliştirilen kriptosistemin bilinen kriptosistemlerle karşılıklı incelemesi yapılmıştır.

7.2. Gereksinimler

7.2.1. Adli Bilimler

Bölüm 7. CUBE, alt bölümü 7.1. Giriş bölümünde yer alan bilgiler doğrultusunda tasarlanan kriptosistemin adli bilimlerde sağlaması gereken temel özellikler aşağıda verilmiştir.

1. Delil bütünlüğünü ve müşterekliğini bozmadan delil özgülleştirilmesinde ve saklanmasında kullanılabilmelidir.
    a. Sayısal delilin şüpheliye sanığa bir kopyası verildiğinde delilin ilgilisine özel yapıya kavuşturulması ve kaynak takibi mümkün olmalıdır.
    b. Sayısal delilin hukuka aykırı paylaşımında ve / veya kullanımında delilin hangi kaynaktan geldiği tespit edilebilmelidir.
    c. Sayısal delilin birden fazla kaynağa özgül yapıda oluşturulabilmesi mümkün olmalıdır.

d. Yetkili kurum ve / veya personelce delil üzerinde yapılacak incelemelerde, incelemeyi yapan tarafın kendisine özgül olarak delili muhafaza etmesi ve delil incelemelerinde delilin takip ettiği süreç izlenebilmelidir.

2. Donanımsal ve yazılımsal uygulamaya elverişli olmalıdır.

a. Olay yeri inceleme birimleri – personelinin temel teknik bilgileriyle kullanabilecekleri donanımların ve yazılımların geliştirilebilmesine uygun olmalıdır

b. Olay yerinde veya delile ilk müdahalede / erişimde şüpheliye-sanığa delilin bir kopyası verilecekse, delilin özgül yapıya [50] dönüştürülmesi kolay olmalıdır.

3. Delil inceleme aşamasında birim veya personel özgül inceleme yapabilmeli ve bütünlüğünü koruyan delili, incelemeyi yapan tarafa özgül olarak muhafaza edebilmelidir.

a. Delile erişen ve / veya inceleme yapan yetkili için özgül yapıda delil muhafazası ve delil inceleme aşamalarının takibi mümkün olmalıdır.

b. Delilin birden fazla birim veya personele özgül olarak işaretlenmesi mümkün olmalıdır.

c. Özgülleştirilmiş delilin her ilgilisi için ayrı bir anahtar ve / veya mesaj özet değeriyle imzalanması mümkün olmalıdır.

50 İmzalanmış ve kaynağı takip edilebilir.

4. Kolay uygulanabilir, anlaşılabilir ve geliştirilebilir algoritmaya sahip olmalıdır..
    a. Uygulama ve anlaşılabilir algoritmayla delile müdahalede olası hataların önüne geçilmesi birim ve personel tarafında etkin tepki sürelerinin elde edilmesi sağlanmalıdır.
    b. Kriptosistem sadece teknik bilgiye sahip birim ve personelce uygulanabilir değil, kamu tarafınca da uygulanabilir olmalıdır.
    c. Kriptosistem farklı platformlarda uygulamaya ve açık kaynak olarak, gelişen teknolojik imkânlar doğrultusunda geliştirilmeye uygun olmalıdır.
    d. Kriptosistemin uygulandığı yazılım ve donanımların teknik yeterliliğe sahip olmayanlarca erişilebilir olması için uyarlama ve geliştirmeler mümkün olmalıdır.

5. Delile veya amaca özgül parametrelerle kullanılabilmelidir.
    a. Sayısal delilin veri boyutuna veya farklı gereksinimlere yanıt verebilecek farklı blok ve anahtar boyutlarına sahip olmalıdır.
    b. Sayısal delil birden fazla farklı yapıya dönüştürülebilmelidir.
    c. Sayısal delilin özgül halinin hangi birim ve / veya personelce ilgili olduğunun gösterimi için kriptosistemde kullanılabilecek bir mesaj özetleme algoritması geliştirilmelidir[51].

51 Bu amaç ile geliştirilen ICDS, Bölüm 7. ICDS ile verilmiştir.

6. Güvenilirliği bilimsel yöntemlerle ispatlanmış ve bu ispatlar tekrar gözlemlenebilir olmalıdır.
    a. Sayısal delilin özgül olarak şifrelenmesinden elde edilecek verinin kırılması mümkün olmamalıdır.
    b. Şifreleme yönteminin güvenirliği analitik ve geçerliliği kabul olmuş yöntemlerle ispat edilmelidir.
    c. Mevcut sayısal delilin sadece ilgilisine özgül yapıda oluşturulabileceği ispatlanmalıdır.
    d. Bilimsel ispatların hepsi gözlemlenebilir, tekrarlanabilir, güvenilirlik ispatı amacına özgül analiz ve yöntemlerle gerçekleştirilmiş oluş bütün olarak kamu ile paylaşılabilir olmalıdır.

7.2.2. Esnek Yapı

1. Geniş bir platform aralığında uygulanabilirliği mümkün olmalıdır.
    a. Şifreleme algoritması donanımsal, yazılımsal ve her iki platformu içerir hibrit uygulamaya elverişli olmalıdır.
    b. Yazılımsal tarafta, masaüstü, aplikasyon (mobil platformlar dahil), bulut ve internet gibi sayısal iletişim sistemleri üzerinde erişime uygun uygulamalar geliştirilebilmesi mümkün olmalıdır.
    c. Donanımsal tarafta, düşük donanım gereksiniminde olması veya donanıma uygun özet değeri uzunlukları, parametreleri ve değişkenleri olmalıdır,

2. İmzalı verilerin oluşturulması ve özgül yapıda saklanması mümkün olmalıdır.
    a. Cube sadece bir şifreleme algoritması olarak değil aynı zamanda özgül yapıda verilerin saklanması için bir dosya formatı olarak kullanılabilmelidir.
    b. Aynı değerlere sahip verilerin birden fazla özgül yapıda saklanması uygulayıcı tarafından seçilebilmelidir.

3. Ayrıca bir şifreleme algoritması olarak kullanılabilmesi mümkün olmalıdır.
    a. Cube sadece dosya saklama formatı olarak değil, şifreleme algoritması olarak kullanılabilmelidir.
    b. Özgül yapıda dosyaların oluşturulmasında kullanılan algoritma rastlantısal şifrelenmiş veriler üreten bir kripto sistem olarak kullanılabilmelidir.

7.2.3. Güvenlik, Hız ve Donanım

Yukarıda ele alınan bilgiler doğrultusunda, tasarlanan özetleme algoritmasında sağlanması gereken güvenlik, hız ve donanım gereksinimleri aşağıda verilmiştir.

1. Algoritma yapısına bağlı olan ve algoritma yapısına bağlı olmayan saldırı yöntemlerine karşı dirençli olmalıdır.

a. Algoritma yapısına bağlı olan MITM [52], Fixed Point [53] ve diferansiyel [54] saldırılarına dirençli olmalıdır ve algoritma yapısına bağlı olmayan brute-force [55], yöntemlerine karşı dirençli olmalıdır.
b. Düz metin ile şifrelemenin arasında çok düşük korelâsyon olmalıdır.
c. Düz metin, anahtar ve/veya Özgülleştirilmiş şifreli metin parametreleri şifreli metinde öngörülemez sonuçlar vermelidir.
d. Her şifreleme işleminden farklı şifrelenmiş metin elde edilebilmelidir.
e. Düz metin şifreli metne geçerken her bir bit değerinin bloktaki konumu değişme özelliğinde olmalıdır.
f. Algoritma fonksiyonları düz metne ait bit değerlerini karıştırırken anahtar, düz metin ve diğer parametrelere göre işlem yapmalı, fonksiyonlar sabit işlemlere sahip olmamalıdır.

2. Şifrelenmiş verilerin oluşturulması ve çözülmesi düşük zaman maliyetinde olmalıdır.

a) Şifreleme ve şifre çözme kısa sürede sonuçlanmalıdır.
b) Zaman maliyetinde elde edilecek fayda güvenlik ve bellek gereksinimi maliyetine sebep olmamalıdır.

52 MITM: Meet in the Middle – Ortada buluşma (Bknz: Beşinci Bölüm Kriptografik Saldırılar)

53 Fixed Point: Sabit Nokta (Bknz: Beşinci Bölüm Kriptografik Saldırılar)

54 Diferansiyel Saldırılar: (Bknz: Beşinci Bölüm Kriptografik Saldırılar).

55 Literatürde eksiksiz arama olarak da geçen bu yöntem olası bütün anahtarların denenmesine dayanır.

c) Zaman maliyetindeki fayda farklı donanım yapılarında da orantılı sonuçlar vermelidir.

d) Zaman maliyetindeki fayda, aynı donanım üzerinde ve aynı anda birden fazla özetleme işleminde de fayda sağlamalıdır.

3. Düşük bellek ve donanım gereksinimlerine sahip olmalıdır.

a) Algoritma düşük bellek-hafıza alanına gereksinim duymalıdır.

b) Algoritma düşük işlem ve mikro işlemci alanına ihtiyaç duymalıdır.

c) Algoritma aritmetik hesaplama ile eş işlemleri mantıksal hesaplama ile elde edebilmeli ve böylelikle donanım-hız maliyetinde fayda sağlanabilmelidir.

7.3. Matematiksel Yapı ve Algoritmalar

7.3.1. Matematiksel Altyapı

7.3.1.1. Yapı

Şifreleme algoritmaları bölüm 3. KRİPTOGRAFİ ve KRİPTOSİSTEMLER Algoritma ile açıklandığı gibi farklı yapı ve sınıflara sahiptir.

Cube'un temel yapısı blok temelinde tasarlanmıştır. Ancak blok yapısı bilinen tek boyutlu diziler veya çift boyutlu yapılardan farklı olarak 3-boyutlu yapıda tasarlanmıştır. Bu blok yapısına bağlı olarak:

1. Şifreleme işleminde tek seferde şifrelenebilecek düz metin boyu bellidir.

2. Algoritmanın kullanılabileceği anahtar boyu ve kelime uzunluğu bellidir.

3. Gereksinim duyulan donanımsal hafıza/bellek ve işlemci mimarisi bellidir.

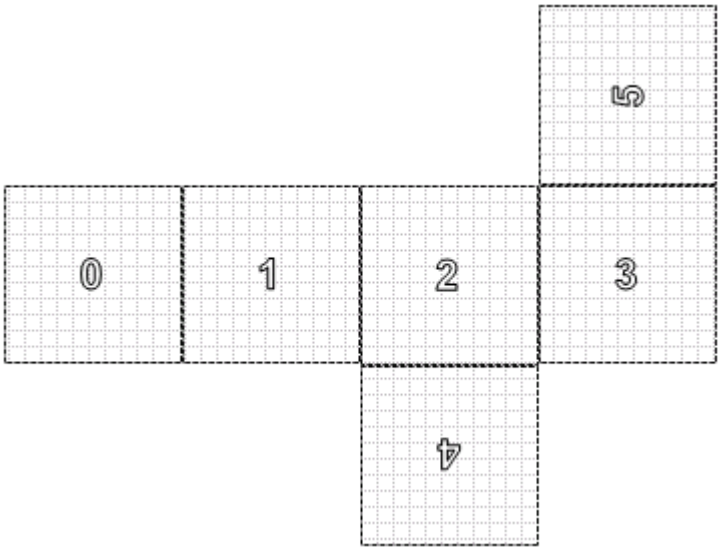
Yukarıdan anlaşıldığı gibi, algoritmanın güvenlik, esneklik, hız ve donanım gereksinimi blok yapısının özelliklerine bağlıdır. Bu sebeple Cube'ün blok yapısı bilinen şifreleme yöntemlerinden farklılıklarla tasarlanmıştır. Bu yapının tasarımı aşağıdaki şekilde oluşturulmuştur:

1. Sbox tablosu olarak bölüm 6.3.1.2. S-Box Tablosu ile verilen sbox tablosunu kullanılmıştır.

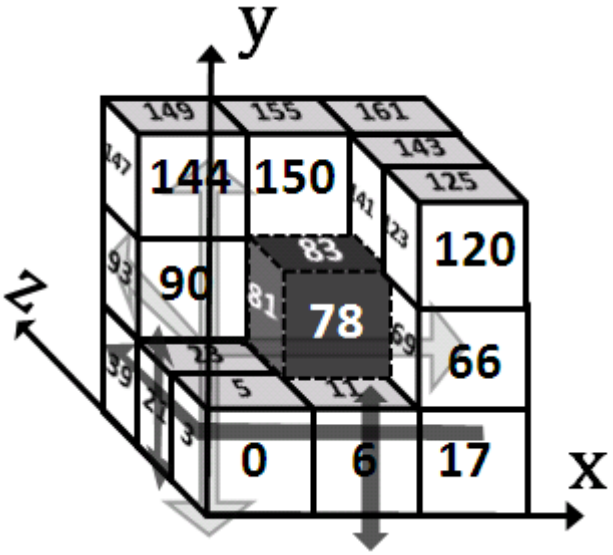
2. Blok yapısının en küçük değerleri bit değerleridir. Bu bit değerlerinin 8 ve katları sayısınca bir araya gelmesiyle kelime (word) değerleri ortaya çıkar

3. Cube için blok yapısının oluşturulmasında Rubik küpünden esinlenilmiş ve ℓ katman değerince *x, y* ve *z* koordinatları oluşturulmuştur.

4. Cube blok yapısında 6 yüzü bulunan cubieleri kullanacak ve her cubie yüzüne 8-bit (1 byte) kelime yerleştirilecek şekilde tasarlanmıştır. Şekil 31**:** Cube - açık ve 3-boyutlu cubienin temsili ile gösterilmiştir.



Şekil 31: Cube - açık ve 3-boyutlu cubienin temsili



Şekil 32: Cube - ℓ=3 katman değerine sahip bir küp bloğu kesiti

5. Cubielerin bir araya getirilmesiyle bir küp bloğu oluşturulmuştur.

6. Küp bloğunun oluşumunda aşağıdaki eşitliğe göre cubie yüz sayısı ve blok boyu oluşturulur:

- ℓ katman değeri, 6 bir cubienin yüz sayısı, *x*,*y* ve *z* Cube'ün 3-boyut koordinatlarıdır. Bu durumda her bir koordinata ait katman değerleri ele alınırsa, cubie sayısı:

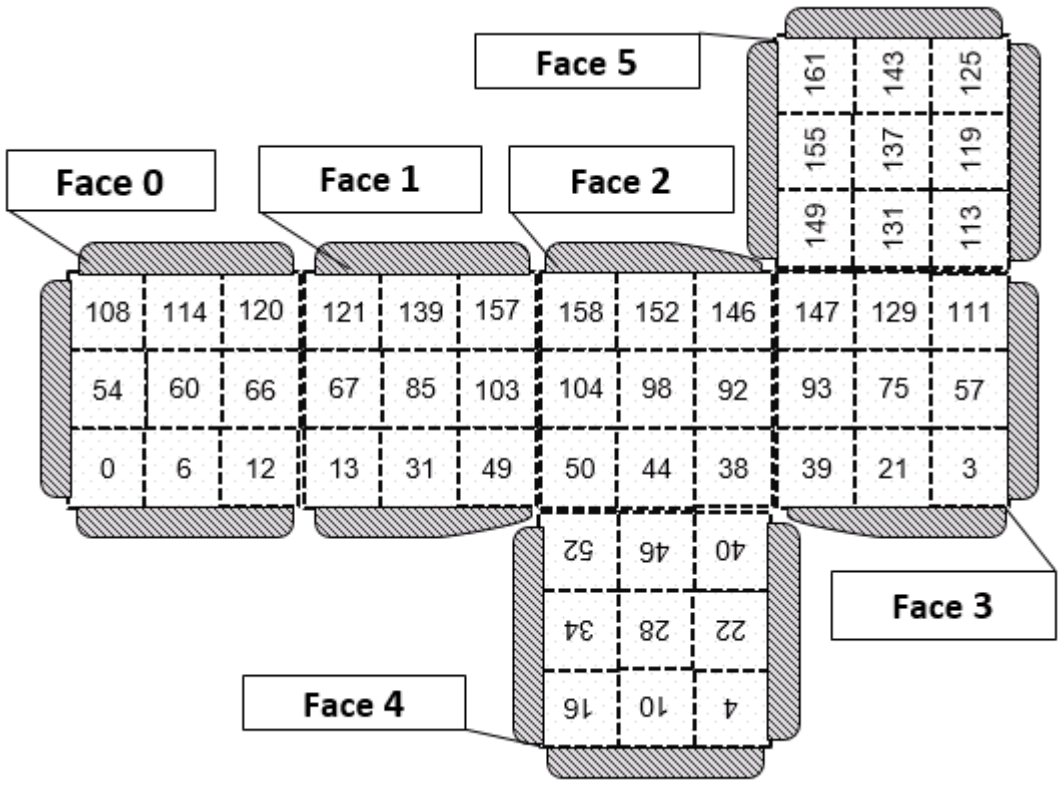
$$\ell = x = y = z \Rightarrow \\ x.y.z = \ell^3 \qquad (111)$$

olarak bulunur. Her bir cubienin 6 yüzü olması sebebiyle, toplam cubie yüzü sayısı:

$$6\ell^3 \qquad (112)$$

olarak bulunur.

7. Cube bloğunda, her cubie yüzüne 8-bit yerleşimi ile $48\ell^3$-bitlik bir Blok boyu elde edilir. Bu bloğun açılmış halinin ön yüzleri Şekil 33 ile gösterilmiştir.



Şekil 33: Cube - Küp bloğunun açılmış hali

7.3.1.2. 3-Boyutlu Cubieler

Blok üzerinde veri yerleşimi kombinasyonu, cubilere göre gerçekleşir. Blok yapısını, oluşturan cubieler, blokta yer alan diğer cubieler ile, bağlantılıdır. Bir cubienin *x, y* veya *z* koordinatlarındaki yer değişimi, ortak koordinattaki diğer cubielerinde taşınmasıyla sonuçlanır.

7.3.1.3. Cube Blokları

Cube, şifrelemede lineer transformasyon ve permütasyon yöntemlerinden farklı karıştırma, diffusion turları ve ℓ katman değerine bağlı anahtar genişletme yöntemleri kullanır.

Cube, $\mathcal{P}$ düz metin yerleşiminde 3-boyutlu alanda kübik blok yapısını referans alır. Geometrik dönme ve cubie yüzlerine yerleştirilen verilerin karıştırılmasında / diffusion için kullanılır.

Bir blok yapısı oluşturulması ℓ katman değerine bağlıdır. Bu değer *x, y* ve *z* koordinatlarındaki cubielerin sayısını da temsil eder. Bir küp bloğundaki cubie sayısı $x.y.z$ olup $\ell = x = y = z$ olduğunda, kısaca ℓ^3 olarak bulunur. Cubielerin 6 yüzünün her birisinin 8-bit değerinde olması sebebiyle bir küp bloğu boyutu $8(6\ell^3) = 48\ell^3$-bit olur.

7.3.1.4. Merkez Cubie

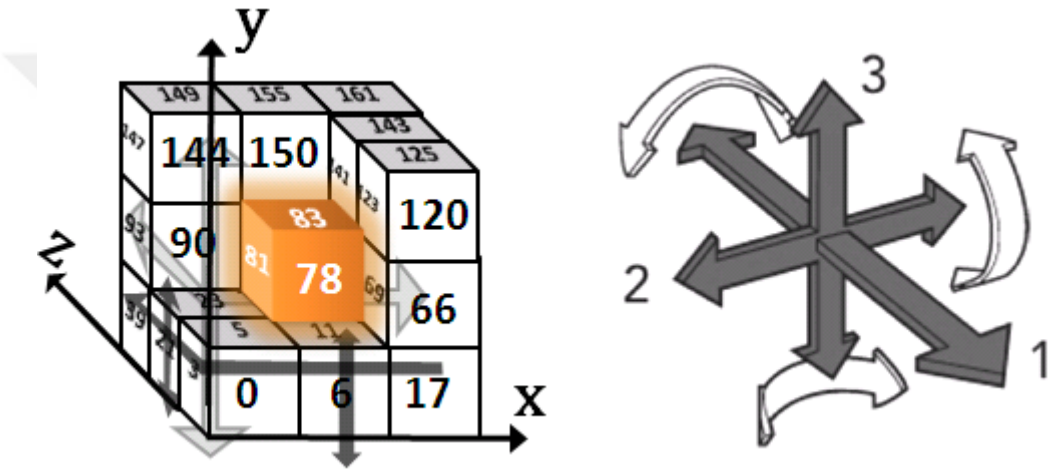
Cube, blok merkezinde yer alan cubieyi asimetrik şifreleme için dönme merkezi olarak kullanır. Dönme merkezinde bir merkez cubienin olması için *x, y* ve *z* ve dolayısıyla ℓ değeri tek sayı olmalıdır. 3-boyutlu kübik blok yapısının oluşturulması aşağıdaki eşitlikle sağlanır:

$$\ell > 1,\ \ell \in \mathbb{Z}^+, \ell = \mathrm{x} = \mathrm{y} = \mathrm{z}, \ell\ mod\ 2 = 1 \quad (113)$$

bu eşitliğe göre en küçük ℓ katman değeri $\ell = 3$ ve en küçük blok boyu $48(3^3) = 1296$-bit olur.

Dönme merkezindeki merkez cubienin koordinatları ℓ katman değerine bağlıdır. Merkez cubienin x, y ve z koordinatları $x = (\ell - 1)/2$, $y = (\ell - 1)/2$ ve $z = (\ell - 1)/2$ olarak hesaplanır.

Koordinatlardaki herhangi bir dönme işlemi merkez cubienin yüz yönlerini döndürebilir ancak merkez cubienin blok içerisindeki konumu dönme işlemi ile değiştirilemez.



Şekil 34: Cube - Merkez Cubie / Dönme Merkezi

Cube – Merkez Cubie / Dönme Merkezi ile ℓ=3 katmanlı bir Cube bloğunun dönme merkezinde yer alan merkez cubie turuncu renk ile temsil edilmiştir.

$\mathcal{P}$ düz metni veya $\mathcal{C}$ şifreli metni küp bloğuna yerleştirilmek istendiğinde bu veriler blok boyunca parçalara bölünür. Ancak, bloğa yerleştirilen verinin boyu ve rastlantısal sayı değeri de dönme merkezinde farklı cubie yüzlerine yerleştirilir. Veri boyu için en fazla 2 cubie yüzü kullanılır. En fazla 2 cubie yüzünün veri boyuna ayrılması ve her yüzün 8-bit olması sebebiyle en fazla 2^{16}-bit boyunda en büüyk blok boyu 55566-byte olur ve en büyük katman değeri $\ell = x = y = z = 21$ olur, bu

durumda da $\ell = 21 \rightarrow (6\ell^3 \rightarrow 55566) < 2^{16}$ olur. Bu eşitliğe göre eğer ℓ=23 olsaydı $(\ell = 23) \rightarrow (6\ell^3 = 73002) \nleq 2^{16}$ olur, bu durumda da bu veri boyutu 2 cubie yüzünden elde edilen 16-bitlik alan ile temsil edilebilecek değeri aşmaktadır

Tablo XXVI: Cube - Desteklenen blok boyları

Katman Değeri (ℓ değeri)	**Blok Boyu (byte olarak)**	**Düz Metin Boyu (byte olarak)**	**Rezerve Yüz**
3	162	160	2
5	750	747	3
7	2058	2055	3
9	4374	4371	3
11	7986	7983	3
13	13182	13179	3
15	20250	20247	3
17	29478	29475	3
19	41154	41151	3
21	55566	55563	3

Tablo XXVI ile ℓ katmanları için blok boyları verilmiştir. Burada verilen blok boyları ayrıca anahtar boyları olarak kullanılmaktadır.

Merkez cubieye yerleştirilecek değerler ℓ katman değerine bağlıdır. Merkez cubienin yerleşim düzeninde rezerve yüzler ve diğer yüzlerin kullanımı Tablo XXVII ile verilmiştir.

Tablo XXVII: Cube - Merkez cubie yüz değerleri

Katman Değeri (ℓ değeri)	Merkez Cubie Numarası	Blok konumu başlangıç değeri	Rezerve Edilen Yüzleri	Veri Girilebilen Yüzleri
3	13	78	0R, 1S	2-5 Data
5	62	372	0R, 1-2S	3-5 Data
7	171	1026	0R, 1-2S	3-5 Data
9	364	2184	0R, 1-2S	3-5 Data
11	665	3390	0R, 1-2S	3-5 Data
13	1098	6588	0R, 1-2S	3-5 Data
15	1687	10122	0R, 1-2S	3-5 Data
17	2456	14736	0R, 1-2S	3-5 Data
19	3429	20574	0R, 1-2S	3-5 Data
21	4630	27780	0R, 1-2S	3-5 Data

*** R:** Rastlantısal Sayı, **S:** Veri Boyu

Tablo XXVII ile görüldüğü gibi 0. Yüz her zaman rastlantısal sayı, 1. Yüz her zaman veri boyu, 2. Yüz ℓ katman değerine bağlı olarak veri ve 3-5 aralığındaki yüzler her zaman veri için kullanılır.

7.3.1.5. Özgül Şifreli Metin

Cube yapısı kaynak paylaşımında takip ve kontrolü sağlayan özgül şifrelenmiş metinler üretebilir. Bu şifrelenmiş metinler, kriptografik amaçlar dışığında açık veriler içinde kullanılabilir.

Özgül şifreli metin randval fonksiyonuyla *AC=0* ve *SC* değerlerine bağlı olarak belirlenir. *SC* 8-bitlik GF(2^n) sonlu alanında ve $0 \leq SC \leq 255$ aralığında bir değerdir. *SC*'nin bu değer aralığı aynı düz metin ve anahtar ile 256 ayrı şifreli metin çıkışı için kullanılır.

AC ve *SC* değerleri encryption fonksiyonuyla belirlenerek putcubies ve randval alt fonksiyonlarında kullanılır. Bu alt fonksiyonlar $AC > 0$ olması durumunda rastlantısal şifreli metin çıkışını temsil eder ve randval fonksiyonu rastlantısal değer üretir. $AC = 0$ olması durumunda randval fonksiyonu rastlantısal değer üretmez ve *SC* ile belirtilen değer göre değerler üretir. $AC = 0$ için $0 \leq SC \leq 255$ aralığında özgül şifrelenmiş metin sağlanır.

Bu yapı aşağıdaki gereksinimleri karşılamak için geliştirilmiştir:

1. Hedefe özgül veri üretimi ve veri kaynak takibi.
2. Aynı veri üzerindeki çoklu çalışmalarda kaynak değişim ve düzenlemelerinin izlenebilmesi.
3. Veri saklanmasında özgül konumlandırma işaretlenmesi.

7.3.2. Algoritmalar

7.3.2.1. Rastlantısal değer üretim fonksiyonu – randval

randval fonksiyonu $AC > 0$ için rastlantısal $AC = 0$ için *SC* değerine göre 8-bitlik değerler üretir.

Literatürde farklı kaotik sayı üretim algoritmaları olmasına karşın bu çalışmada randval için basit bir algoritma kullanılarak $AC = 1$ değerine tanımlanmıştır. Burada mikro işlemciden alınan tick değerinin rakamları *sbox* karşılıklarıyla XOR işleminden geçirilerek kullanılır.

randval fonksiyonu $AC = 1$ için *toSequencedSeperation* fonksiyonunu kullanır. Bu fonksiyon, $i, j, a \in \mathbb{Z}_{\infty}^{+}; \mathbb{O} = \mathbb{O}_0, \dots, \mathbb{O}_{n-1}$ olduğunda a giriş değerinin

basamaklarındaki rakamları $\mathbb{O}$ dizisine atar. Örnek olarak *a=123* ise çıkış $\mathbb{O}_0 = 3, \mathbb{O}_1 = 2, \mathbb{O}_2 = 1$ olur. *toSequencedSeperation* fonksiyonu aşağıda verilmiştir.

$$
\begin{aligned}
&i, j, a \in Z_{\infty}^{+}; i = 0; j = \boldsymbol{log}_{10}\ a\text{-}(\boldsymbol{log}_{10}\ a\ \boldsymbol{mod}\ 1) \\
&\boldsymbol{for}\ i = 0\ \boldsymbol{to}\ j\ \boldsymbol{do} \\
&\mathbb{O}_i = (a\ \boldsymbol{mod} 10^{i+1}) \,/\, 10^i \qquad (114) \\
&a = a - (\mathbb{O}_i)\ .10^i \\
&\boldsymbol{end\ for}
\end{aligned}
$$

AC için 0 ve 1 değerleri tanımlanarak randval fonksiyonunda kullanılmıştır. Farklı rastlantısal sayı algoritmaları da, *AC > 1* için randval fonksiyonuyla kullanılabilir. randval fonksiyonu aşağıda verilmiştir.

Algoritma 6: CUBE - randval fonksiyonu

Variables:	$\boldsymbol{tick_i}$; // The temporal tick value is taken from the microprocessor.
Inputs:	$\boldsymbol{RC}$;// Takes values between 0-255 and effects the output of the function.
	$\boldsymbol{AC}$;//When AC = 0, specific ciphertext output is obtained depending on the SC value. When AC≥1, the ciphertext is generated randomly based on the *randval* function.
	$\boldsymbol{SC}$;// is a value effecting$\mathbb{O}$ when AC= 0.
Outputs:	$\mathbb{O}$;//is the random or non-random output value.

```
01: RC = RC mod 256;
02:  if(AC = 0)then
03:  𝕆 = sbox[(sbox[RC] ⊕sbox[SC]]
04: else if(AC = 1 )then
05:  𝕆 = 𝕆 + 1
06:  tick = toSequencedSeperation(tick)
07:  for i = 0 to (len(tick)- 1) do
08:  𝕆 ⊕= sbox[tick[i]] ⊕RC]
09:  RC= (RC+ 1) mod 256
10:  end for
11: end if
12: return 𝕆
```

7.3.2.2. Anahtar ve XOR – key_xor fonksiyonu

β bir blok, $\mathcal{K}$ bir anahtar dizisi ve key_xor bir fonksiyon olduğunda, keyxor β bloğunu $\mathcal{K}$ dizisi değerleriyle x-y-z katmanları ve komşu katmanları düzeninde katmanlara etki ederek XOR işleminden geçirir. key_xor fonksiyonu aşağıda verilmiştir.

Algoritma 7: CUBE - key_xor fonksiyonu

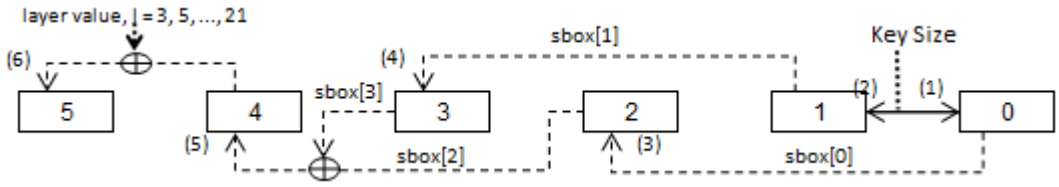
Variables:	**c;**//indicates the order of the value to be called from the $\mathcal{K}$ byte array. It takes values between 0-255.
Inputs:	$\boldsymbol{\beta_{x,y,z,f}}$**;**// is the multidimensional data array used to XOR cubies with $\mathcal{K}$ byte array values, when ℓ=x=y=z and f=$f_0, \ldots, f_5$. $\boldsymbol{\mathcal{K}_n}$**;**//is the key array. $\boldsymbol{XORCount}$**;**// is the value representing the number of times the function is called and the value to be called from the sbox. **XORType;**// represents encryption or decryption.
Outputs:	$\boldsymbol{\beta_{x,y,z,f}}$**;**//is a data block that has been XORed with $\mathcal{K}$ key values.

```
01: c = 0
02: for i = 0 to (ℓ - 1)do
03:     for j = 0 to (ℓ - 1) do
04:         for k = 0 to (ℓ-1) do
05:             for f = 0 to 5 do
06:                 if XORType = encryption
07:                     β[k][i][j][f] ⊕= ( β[(k + 1) modℓ][i][j][f] ⊕ β[k][(i + 1) modℓ][j][f] ⊕
00:                     β[k][i][ (j + 1) modℓ][f] ⊕ sbox[XORCount mod 256] ⊕ 𝒦[c] )
08:                 else if  XORType = decryption
09:                     β[(ℓ-1)-k][(ℓ-1)-i][(ℓ-1)-j][f] ⊕= ( β[(ℓ-k) modℓ][(ℓ-1)-i][(ℓ-1)-j][f] ⊕
00:                     β[(ℓ-1)-k][ (ℓ-i) modℓ][(ℓ-1)-j][f] ⊕β[(ℓ-1)-k][(ℓ-1)-i][(ℓ-j) modℓ][f] ⊕
00:                     sbox[((ℓ² − 1) – XORCount) mod 256] ⊕𝒦[(6ℓ³ − 1) – c] )
10:                 end if
11:                 c = c + 1
12:             end for
13:         end for
```

14: ¦ end for
15: end for
16: return β

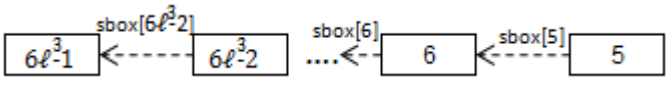
7.3.2.3. Anahtar genişletme – keyexpansion fonksiyonu

keyexpansion fonksiyonu girdi olan $\mathcal{K}$ anahtar dizisini $48\ell^3$*-bit* uzunluğunda genişletir. keyexpansion bir fonksiyon, $\mathcal{K}$ anahtar dizisi *len(*$\mathcal{K}$*)* bu dizideki 8 bitlik elemanların sayısı, $\mathcal{K}_0^1$ genişletilmiş anahtar dizisi olsun. Once *len(*$\mathcal{K}$*)* nin ilk 8-bitlik değeri $\mathcal{K}_0^1$ dizi elemanı ve son 8-bitlik değeri $\mathcal{K}_1^1$ dizi elemanı olarak aktarılır. Devamında sırasıyla bu değerlerin xbox dizisindeki karşılıkları alınarak $\mathcal{K}_0^1$ dizisinin ilk 6 elemanı atanır. Bu işlem Şekil 35 ile gösterilmiştir.



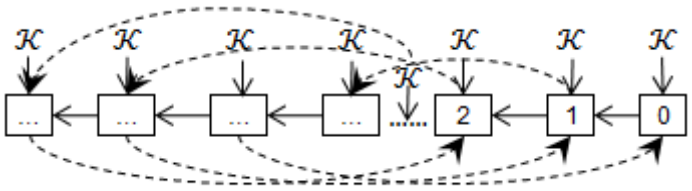
Şekil 35: Cube - Anahtar genişletme, adım 1

İlk 6 değerin tanımlanmasından sonra K_0^1 dizi elemanından başlayarak son dizi elemanında kadar bütün elemanlar bir önceki dizi elemanının *sbox* karşılığıyla değiştirilir. Bu işlem Şekil 36 ile gösterilmiştir



Şekil 36: Cube - Anahtar genişletme, adım 2

Bu şekilde anahtar genişletme için bir anahtar dizilimi oluşturulmuş olunur. $\mathcal{K}_i$ dizisindeki anahtarlar, değerleri hazırlanan diziye aktarılırken boyu büyük olan dizinin boy değeri tur sayısı olarak kullanılır. Her turda $\mathcal{K}_i$ dizisinin 3 ayrı eleman değeri hesaplanır, bir tur temsili Şekil 37 ile gösterilmiştir.



Şekil 37: Cube - Anahtar genişletme, adım 3

Yukarıdaki tanım ile sağlanan anahtar genişletme işlemi her bir anahtar bitinin genişletilmiş alana etkili dağıtılmasını sağlar. keyexpansion fonksiyonuna ait algoritma aşağıda verilmiştir.

Algoritma 8: CUBE - keyexpansion

Variables:	c;// The number of rounds to use in calculating the key array and the counter value.
	len($\mathcal{K}$) ;// Represents the number of 8-bit (byte) values in the key array.
Inputs:	$\mathcal{K}_n$;// is the key array used to calculate the expanded key output array values in $\mathbb{O}_n$.
Outputs:	$\mathbb{O}_n$;// is the $\mathcal{K}_n$ array expanded to $6\ell^3$.

```
01: c = 0;
02: if len(𝒦) = 0 then
03: ¦ for i = 0 to (6ℓ^3 − 1) do
04: ¦ ¦ 𝒦[i] = 0
05: ¦ end for
06: end if
07: 𝕆[0] = len(𝒦) ∧ 255;    𝕆[1] = len(𝒦) ≫ 8;
08: 𝕆[2] = sbox[𝕆[0]];          𝕆[3] = sbox[𝕆[1]] ;
09: 𝕆[4] = sbox[𝕆[2] ⊕ 𝕆[3]];
10: 𝕆[5] = sbox[ℓ] ⊕ sbox[𝕆[4]];
11: for i = 6 to (6ℓ^3 - 1) do
12: ¦ 𝕆[i] = sbox[𝕆[i- 1]]
13: end for
14: for i = 0 to (c − 1) do
15: ¦ 𝕆[i mod 6ℓ^3] ⊕=𝒦[i mod len(𝒦)]
16: ¦ 𝕆[(i+ 1) mod 6ℓ^3] ⊕=sbox[𝕆[imod6ℓ^3]]
17: ¦ 𝕆[(3ℓ^3+ 1)  mod 6ℓ^3] ⊕=sbox[𝕆[(i+ 1) mod6ℓ^3]]
18: ¦ end for
19: return 𝕆
```

7.3.2.4. Kübik formda veri yerleşimi – putcubies fonksiyonu

8-bitlik $6\ell^3$ sayısınca cubie yüzüne very yerleşimi özgül bir şekilde gerçekleşir. Önce merkez cubienin rastlantısal (*AC > 0* için) veya özgül *(AC = 0* için) değeri olarak 0. yüzüne ait değerler atanır. Devamında da $\mathcal{P}$ veri boyuna ait değerin yerleşimi gerçekleştirilir.

Düzmetin veya şifreli metin için bloğa yerleşim cubie yüzlerini (f) referans alan ve x,y,z koordinatlarına göre işleyen sıralama ile gerçekleşir.

Burada ℓ=x=y=z olduğundan ℓ tur fonksiyonunun üst limiti olur ve $\circlearrowleft_{y=0}^{\ell-1}\left(\circlearrowleft_{z=0}^{\ell-1}\left(\circlearrowleft_{x=0}^{\ell-1}\left(\circlearrowleft_{f=0}^{5}(\cdots)\right)\right)\right)$ kombinasyonu uygulanır. putcubies fonksiyonu aşağıda verilmiştir.

Algoritma 9: CUBE - putcubies

Variables:	$\boldsymbol{c}$ **;**/ /is the counter value representing the 8-bit (byte) elements of the $\mathcal{P}_n$ array. **mc ;**// represents the core cubie with $(\ell - 1)$ / 2 value. **rc ;**// is the counter variable to be used in the call of *randval* function.

Inputs:	$\boldsymbol{\mathcal{P}_n}$ **;**// represents the plaintext block having size of $6\ell^3 - rf$ and *rf* represents the number of cubie faces allocated for the data size and random number value in the core cubie.
	AC ; // is the value that represents the Pseudorandom Number Generator algorithm used for AsymmetricCiphertext. If it is 0, the ciphertext is symmetric and the ciphertext output is obtained according to the SC value representing the specific ciphertext.
	SC ;// is the value in range 0-255 and represent the specific ciphertext if AC = 0.

Outputs:	$\mathbb{O}_{\boldsymbol{x,y,z,f}}$ **;**// The multidimensional array in which the $\boldsymbol{\mathcal{P}_n}$ array values are placed, with $\ell = x = y = z$ and $f = \{f_5, f_{\dots}, f_0\}$.

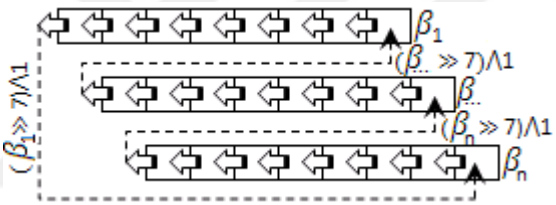
```
01: rc = 0;  c= 0;  mc=(ℓ - 1) / 2;  𝕆[mc][mc][mc][0]=randval(rc, AC, SC);  rc = rc + 1;
02: if (ℓ = 3) then
03: ¦rf = 2
04: else if (ℓ > 3)then
05: ¦rf = 3
06: end if
07: for y = to (ℓ - 1) do
08: ¦for z = 0 to (ℓ - 1) do
09: ¦ ¦for x = 0 to (ℓ-1) do
```

```
10: ¦ ¦ ¦for f = 0 to 5 do
11: ¦ ¦ ¦ ¦if (x = y = z = mc) then
12: ¦ ¦ ¦ ¦ ¦if (f > (rf - 1)) then
13: ¦ ¦ ¦ ¦ ¦ ¦if(len(𝒫) > c) then
14: ¦ ¦ ¦ ¦ ¦ ¦ ¦𝕆[x][y][z][f] = 𝒫[c] ⊕ 𝕆[mc][mc][mc][0]
15: ¦ ¦ ¦ ¦ ¦ ¦ ¦rc = rc + 1;
13: ¦ ¦ ¦ ¦ ¦ ¦else
14: ¦ ¦ ¦ ¦ ¦ ¦ ¦𝕆[x][y][z][f] = 𝒫[c] ⊕ 𝕆[mc][mc][mc][0]
15: ¦ ¦ ¦ ¦ ¦ ¦ ¦rc = rc + 1;
16: ¦ ¦ ¦ ¦ ¦ ¦end if
17: ¦ ¦ ¦ ¦ ¦end if
18: ¦ ¦ ¦ ¦else
13: ¦ ¦ ¦ ¦ ¦if(len(𝒫) > c) then
14: ¦ ¦ ¦ ¦ ¦ ¦𝕆[x][y][z][f] = 𝒫[c] ⊕ 𝕆[mc][mc][mc][0]
15: ¦ ¦ ¦ ¦ ¦ ¦rc = rc + 1;
13: ¦ ¦ ¦ ¦ ¦else
14: ¦ ¦ ¦ ¦ ¦ ¦𝕆[x][y][z][f] = 𝒫[c] ⊕ 𝕆[mc][mc][mc][0]
15: ¦ ¦ ¦ ¦ ¦ ¦rc = rc + 1;
16: ¦ ¦ ¦ ¦ ¦end if
24: ¦ ¦ ¦ ¦end if
25: ¦ ¦ ¦end for
26: ¦ ¦end for
27: ¦end for
28: end for
29: if ℓ = 3 then
29: 𝕆[mc][mc][mc][1] = len(𝒫) ⊕ 𝕆[mc][mc][mc][0]
30: else
31: 𝕆[mc][mc][mc][1] = (len(𝒫) ≫ 8) ⊕ 𝕆[mc][mc][mc][0]
32: 𝕆[mc][mc][mc][2] = (len(𝒫) ∧ 255) ⊕ 𝕆[mc][mc][mc][0]
33: end if
34: return 𝕆
```

7.3.2.5. Blok boyunda bit döndürme – bitshift fonksiyonu

bitshift fonksiyonu, $cubemixer()$ fonksiyonunca uygulanan tur tarafından kullanılır. Burada $bitshift(\mathcal{V}, v)$ bir fonksiyon, β bir bloğu, RV fonksiyonun tur sayısını temsil ettiğinde, β bloğundaki 8-bitlik her değerin 8. bit değeri bir önceki 8-bitlik değerin 1. bitine aktarılır ve $1, \dots, 7$ aralığındaki her bit değeri bir sonraki bit değeri basamağına taşınır.

Şekil 38 ile gösterildiği gibi $\beta_n = \{\{0_8, 0_{\dots}, 0_1\}, \dots, \{n_8, n_{\dots}, n_1\}\}$ olduğunda bitshift fonksiyonundaki bir tur sonucu $\beta_n^2 = \{\beta_{[i][7]}^1, \beta_{[i][\dots]}^1, \beta_{[i][1]}^1, \beta_{[i+1][8]}^1\}$ elde edilir.



Şekil 38: Cube – bitshift

bitshift fonksiyonu aşağıda verilmiştir.

Algoritma 10: CUBE - bitshift

Variables:	$ca_{6\ell^3}$ **;//** is a temporary array to be used when transferring values in a multidimensional $\beta_{x,y,z,f}$ array to a one-dimensional array.
	cv **;//** is a temporary array that hosts the first value of the ca array by processing $ca_0 \gg 7$.
	c ;// is the counter used to access the values in the ca_i array

Inputs:	$\beta_{x,y,z,f}$ **;//** is the data array whose bits are shifted on the cafrom right-to-left depending on RV, given that ℓ=x=y=z and f=$f_0, \dots, f_5$.

***RV* ;//** is the value representing the shift amount of bit values in $\beta_{x,y,z,f}$ array moved to the ca array.

***ShiftType* ;//** indicates the purpose of encryption or decryption in function call.

Outputs: $\mathbb{O}_{x,y,z,f}$**;//** is the left shifted version of $\beta_{x,y,z,f}$ by the RV value.

01: cv = 0; c= 0; RV = RV mod ℓ;
02: for y=0 to (ℓ - 1) do
03: ¦for z=0 to (ℓ- 1) do
04: ¦ ¦for x=0 to(ℓ-1) do
05: ¦ ¦ ¦for f=0 to 5do
06: ¦ ¦ ¦ ¦ca[c] = β[x][y][z][f]
07: ¦ ¦ ¦ ¦c=c+ 1
08: ¦ ¦ ¦end for
09: ¦ ¦end for
10: ¦end for
11: end for
12: if (ShiftType = encryption)
13: ¦for i = 0 to (RV – 1)
14: ¦ ¦cv = (ca[0] ≫ 7) ∧1
15: ¦ ¦ ¦for j = 0 to ($6\ell^3$-2)do
16: ¦ ¦ ¦ ¦ca[j] = ((ca[j] ≪ 1) ∧254) ∨ ((ca[(j + 1) mod $6\ell^3$] ≫ 7) ∧ 1)
17: ¦ ¦ ¦end for
18: ¦ ¦ca[$6\ell^3$ - 1] = (ca[$6\ell^3$- 1] ≪ 1) ∨cv
19: ¦end for
20: end if
21: if (ShiftType = decryption)
22: ¦for i = 0 to (RV – 1)
23: ¦ ¦cv = (ca[$6\ell^3$ - 1] ≪ 7) ∧ 254
24: ¦ ¦ ¦for j = 0 to ($6\ell^3$-2)do
25: ¦ ¦ ¦ ¦ca[($6\ell^3$ - 1) - j] = ((ca[($6\ell^3$ -1) - j]≫ 1) ∧ 127) ∨ ((ca[($6\ell^3$ - 2) -j] ≪ 7) ∧ 128)
26: ¦ ¦ ¦end for
27: ¦ ¦ca[0] = ((ca[0] ≫ 1) ∧ 127) ∨ cv
28: ¦end for
29: end if
30: c = 0

```
31: for y=0 to (ℓ - 1) do
32: ¦for z=0 to (ℓ- 1) do
33: ¦ ¦for x=0 to(ℓ-1) do
34: ¦ ¦ ¦for f=0 to 5do
35: ¦ ¦ ¦ ¦𝕆[x][y][z][f]=CA[c]
36: ¦ ¦ ¦ ¦c=c+ 1
37: ¦ ¦ ¦end for
38: ¦ ¦end for
39: ¦end for
40: end for
41: return 𝕆
```

7.3.3. Geometrik algoritmalar

x_move, *y_move* ve *z_move* fonksiyonları, bloğun (β) düzlem sayıldığı alanda dönme merkezi haricindeki bütün cubielerin noktalarını değiştiren dönme dönüşümünü gerçekleştirirler. Bu dönme dönüşümlerinde α dönme açısı *90°, 180°* ve *270°* değerlerinde olur.

x_move dönme dönüşümü (y, z) düzleminde gerçekleşirken *x* herzaman sabit, *y_move* dönme dönüşümü (x, z) düzleminde gerçekleşirken y herzaman sabit ve *z_move* dönme dönüşümü (x, y) düzleminde gerçekleşirken *z* herzaman sabit olur.

Bu dönme işlemleri x_move, $y_move$ve z_move fonksiyonlarıyla gerçekleşirken, $RV = \{0, 1, 2, 3\}$ değeri β bloğundaki $\{0°, 90°, 180°, 270°\}$ dönüşlerini temsil eder.

7.3.3.1. x ekseninde dönme - x_move fonksiyonu

x_move fonksiyonu cubielerin *y* ve *z* koordinatlarında değişim sağlarken *x* koordinatı sabit kalır.

Algoritma 11: CUBE - x_move

Inputs: $\boldsymbol{\beta_{x,y,z,f}}$ **;//** is the data block to be rotated on layer *x*, given that ℓ=x=y=z and f=$f_0, \ldots, f_5$.

RV ;// is the amount of rotation on x coordinate, given that {0,1,2,3 } → {0°,90°,180°,270°}.

x ;// is the data block layer to be rotated.

Outputs: $\mathbb{O}_{\boldsymbol{x,y,z,f}}$ **;//** is the rotated version of *x* layer of $\beta_{x,y,z,f}$ based on *RV* value.

01: $\mathbb{O} = \beta$; x = x mod ℓ; RV = RV mod 4;
02: for i = 0 to (RV - 1) do
03: ¦ for j = 0 to (ℓ - 1) do
04: ¦ ¦ for k = 0 to (ℓ - 1) do
05: ¦ ¦ ¦ $\mathbb{O}$[x][k][j][0] = β[x][j][(ℓ * k) + ℓ – k - 1 mod ℓ][4]
06: ¦ ¦ ¦ $\mathbb{O}$[x][k][j][1] = β[x][j][(ℓ * k) + ℓ – k - 1 mod ℓ][1]
07: ¦ ¦ ¦ $\mathbb{O}$[x][k][j][2] = β[x][j][(ℓ * k) + ℓ – k - 1 mod ℓ][5]
08: ¦ ¦ ¦ $\mathbb{O}$[x][k][j][3] = β[x][j][(ℓ * k) + ℓ – k - 1 mod ℓ][3]
09: ¦ ¦ ¦ $\mathbb{O}$[x][k][j][4] = β[x][j][(ℓ * k) + ℓ – k - 1 mod ℓ][2]
10: ¦ ¦ ¦ $\mathbb{O}$[x][k][j][5] = β[x][j][(ℓ * k) + ℓ – k - 1 mod ℓ][0]
11: ¦ ¦ end for
12: ¦ end for
13: ¦ β = $\mathbb{O}$
14: end for
15: return $\mathbb{O}$

7.3.3.2. y ekseninde dönme - y_move fonksiyonu

y_move fonksiyonu cubielerin *x* ve *z* koordinatlarında değişim sağlarken *y* koordinatı sabit kalır.

Algoritma 12: CUBE - y_move

Inputs: $\boldsymbol{\beta_{x,y,z,f}}$ **;//** is the data block to be rotated on layer *y*, given that ℓ=x=y=z and f=$f_0, \ldots, f_5$.
RV ;// is the amount of rotation on y coordinate, given that {0,1,2,3 } → {0°,90°,180°,270°}.
y ;// is the data block layer to be rotated.

Outputs: $\mathbb{O}_{x,y,z,f}$ **;//** is the rotated version of *y* layer of $\beta_{x,y,z,f}$ based on *RV* value.

```
01: 𝕆 = β;  y = y mod ℓ;  RV = RV mod 4;
02: for i = 0 to (RV - 1) do
03: ¦for j = 0 to (ℓ - 1) do
04: ¦ ¦for k = 0 to (ℓ - 1) do
05: ¦ ¦ ¦for f = 0 to 3 do
06: ¦ ¦ ¦ ¦ 𝕆[k][y][j][f] = β[j][y][(ℓ * k) + ℓ – k - 1 mod ℓ][f + 3 mod 4]
07: ¦ ¦ ¦end for
08: ¦ ¦end for
09: ¦end for
10: ¦β = 𝕆
11: end for
12: return 𝕆
```

7.3.3.3. z ekseninde dönme - z_move fonksiyonu

z_move fonksiyonu cubielerin *x* ve *y* koordinatlarında değişim sağlarken *z* koordinatı sabit kalır

Algoritma 13: CUBE - z_move

Inputs:	$\boldsymbol{\beta_{x,y,z,f}}$ **;//** is the data block to be rotated on layer *z*, given that ℓ=x=y=z and f=$f_0, \dots, f_5$.
	RV ;// is the amount of rotation on z coordinate, given that {0,1,2,3 } → {0°,90°,180°,270°}.
	z ;// is the data block layer to be rotated.

Outputs:	$\mathbb{O}_{\boldsymbol{x,y,z,f}}$ **;//** is the rotated version of *z* layer of $\beta_{x,y,z,f}$ based on *RV* value.

```
01: 𝕆 = β;  z = z mod ℓ;  RV = RV mod 4;
02: for i = 0 to (RV - 1) do
03:     for j = 0 to (ℓ - 1) do
04:         for k = 0 to (ℓ - 1) do
05:             𝕆[j][k][z][0] = β[(ℓ * k) + ℓ – k - 1 mod ℓ][j][z][0]
06:             𝕆[j][k][z][1] = β[(ℓ * k) + ℓ – k - 1 mod ℓ][j][z][5]
07:             𝕆[j][k][z][2] = β[(ℓ * k) + ℓ – k - 1 mod ℓ][j][z][2]
08:             𝕆[j][k][z][3] = β[(ℓ * k) + ℓ – k - 1 mod ℓ][j][z][4]
09:             𝕆[j][k][z][4] = β[(ℓ * k) + ℓ – k - 1 mod ℓ][j][z][1]
10:             𝕆[j][k][z][5] = β[(ℓ * k) + ℓ – k - 1 mod ℓ][j][z][3]
11:         end for
12:     end for
13:     β = 𝕆
14: end for
15: return 𝕆
```

7.3.3.4. Karıştırma, cubemixer ve cubedemixer fonksiyonları

Çalışmanın önceki bölümlerinde dönme merkezinde yer alan birimin (◎) simgesiyle temsil edildiğini *x*, *y* ve *z* koordinatlarının $\frac{\ell-1}{2}$ olduğunu ve blokta $3\ell^3$ –

$3, \dots, 3\ell^3 + 2$ aralığındaki 8-bitlik 6 değeri barındırdığı yapı açıklanmıştır. Dönme merkezi olan birimde yer alan değerler ve $keyexpansion$ fonksiyonu ile oluşturulan $\mathcal{K}$ dizisinin $n = \{3\ell^3 - 3, \dots, 3\ell^3 + 2\}$ aralığındaki değerleri cube bloğunun karıştırılmasında kullanılır.

Burada β bloğu içerisinden alınan değerler, $\beta_{\left[\frac{(\ell-1)}{2}\right]\left[\frac{(\ell-1)}{2}\right]\left[\frac{(\ell-1)}{2}\right][0]}$ yüzeyinde bulunan rastlantısal veya isteğe bağlı değere göre farklılık gösterir. $\mathcal{K}_n$ dizisinde yer alan değerler ise anahtar verisine $(\mathcal{K}_n)$ ve ℓ değerine bağlı olarak sabittir.

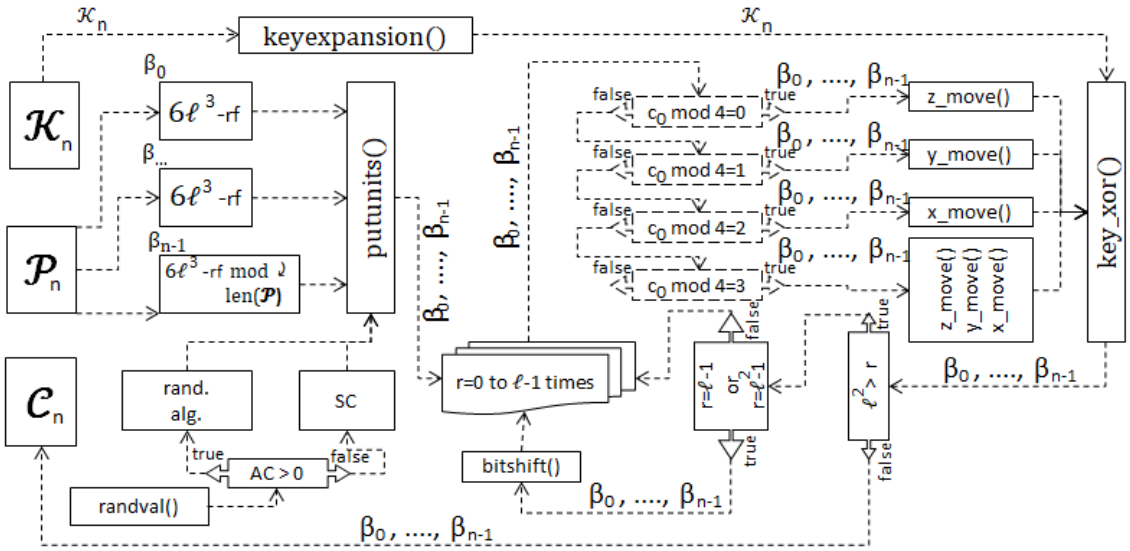
Karıştırma işlemi ℓ^2 tur döngüsü sayısınca tekrarlanır. i o anki döngü ve ca_n karıştırma işlemi için kullanılacak 6 adet değeri barındıran dizi olduğu var sayılsın.

$i \; mod \; 2 = 0$ olması durumunda $\beta_{\left[\frac{(\ell-1)}{2}\right]\left[\frac{(\ell-1)}{2}\right]\left[\frac{(\ell-1)}{2}\right]}$ küp birimi ca_n dizisine kopyalanır, β blok ca_n dizisinin değerlerine göre bu bölüm başlığında verilen *x_move, y_move* veya *z_move* fonksiyonlarına gönderilerek karıştırılır, ve son olarak ca_n dizisi $\beta_{\left[\frac{(\ell-1)}{2}\right]\left[\frac{(\ell-1)}{2}\right]\left[\frac{(\ell-1)}{2}\right]}$ birimine kopyalanır.

$i \; mod \; 2 = 1$ olması durumunda $\mathcal{K}_{[3\ell^3-3]}, \dots, \mathcal{K}_{[3\ell^3+2]}$ aralığı c_n dizisine kopyalanır, β blok ca_n dizisinin değerlerine *x_move, y_move* veya *z_move* fonksiyonlarına gönderilerek karıştırılır, ancak ca_n dizisi β bloğuna kopyalanmaz.

Her tur sonunda β bloğu ve $\mathcal{K}$ dizisi *key_xor* fonksiyonuna gönderilirler. Ancak tur sayaç değeri $i = \ell - 1$ veya $i = \ell^2 - \ell$ olduğunda, β bloğu *key_xor* fonksiyonundan sonar ayrıca $RV = \mathcal{K}_{[i]}$ giriş değeriyle bitshift fonksiyonuna gönderilir.

Bu aşamalı tur döngüsü ile verilerin β blokta rastlantısal ve $\mathcal{K}$ dizisine bağlı olarak geri dönüştürülebilir karışımı sağlanır. Tur döngülerine ait bir şema Şekil 39 ile verilmiştir.



Şekil 39: Cube - Diyagram

cubemixer algoritması aşağıda verilmiştir.

Algoritma 14: CUBE - cubemixer

Variables: ca_n ;// is a six-element temporary array in which the core cubie or key values are copied during cyclic mixing.
mc ;// is the value representing the x, y, z coordinates of the core cubie.
mo ;// represents the beginning of the six values in the middle of the $\mathcal{K}_n$ array.

Inputs: $\boldsymbol{\beta_{x,y,z,f}}$;// is the data block whose layers x, y, and z are geometrically repositioned (mixed), bit values are shifted, and XORed with $\mathcal{K}_n$, given thatℓ=x=y=z and f=$f_0, \ldots, f_5$.
$\boldsymbol{\mathcal{K}_n}$;// is the key array with size n = $6\ell^3$ used for encryption of the data block $\beta_{x,y,z,f}$.

Outputs: $\mathbb{O}_{x,y,z,f}$;// is a mixed and encrypted version of the $\beta_{x,y,z,f}$ block.

```
01: mc=(ℓ − 1)/2;  mo=3ℓ³ − 3;    𝕆=β;
02: for i =0 to (ℓ²- 1) do
03:   if(i mod 2 = 0) then
04:     ca = 𝕆[mc][mc][mc]
05:   else if (i mod 2 = 1) then
06:     for j = 0 to 5 do
```

```
ca[j] = 𝒦[mo +j]
end for
end if
if(ca[0] mod 4 = 0 )
    𝕆 = z_move(𝕆, ca[1], ca[2])
else if(ca[0] mod 4 = 1 )
    𝕆 = y_move(𝕆, ca[1], ca[2])
else if(ca[0] mod 4 = 2 )
    𝕆 = x_move(𝕆, ca[1], ca[2])
else if(ca[0] mod 4 = 3 )
    𝕆 = z_move(𝕆, ca[1], ca[2])
    𝕆 = y_move(𝕆, ca[1], ca[2])
    𝕆 = x_move(𝕆, ca[1], ca[2])
end if
if (i mod 2 = 0)
    𝕆[mo][mo][mo] = ca
end if
𝕆 = key_xor(𝕆, 𝒦, (imod 256), encryption)
if (i = (ℓ−1) or i = (ℓ²−ℓ))
    𝕆 = bitshift(𝕆, 𝒦[i], encryption)
end if
end for
return 𝕆
```

Yukarıda verilen *cubemixer* fonksiyonunun tersi $cubemixer^{-1}$ fonksiyonuna *cubedemixer* dersek, bu fonksiyon aşağıda verilmiştir.

Algoritma 15: CUBE - cubedemixer

Variables: ca_n **;//** is a six-element temporary array in which the core cubie or key values are copied during cyclic mixing.
mc ;// is the value representing the x, y, z coordinates of the core cubie.
mo ;// represents the beginning of the six values in the middle of the $\mathcal{K}_n$ array.

Inputs: $\boldsymbol{\beta_{x,y,z,f}}$;// is the data block whose layers x, y, and z are geometrically repositioned (mixed), bit values are shifted, and XORed with $\mathcal{K}_n$, given thatℓ=x=y=z and f=$f_0, \ldots, f_5$.

$\boldsymbol{\mathcal{K}_n}$;// is the key array with size n = $6\ell^3$ used for encryption of the data block $\beta_{x,y,z,f}$.

Outputs: $\mathbb{O}_{x,y,z,f}$;// is a mixed and encrypted version of the $\beta_{x,y,z,f}$ block.

---

mc $= (\ell - 1)/2$; mo= $3\ell^3 - 3$; $\mathbb{O} = \beta$;
for i=0 to (ℓ^2- 1) do
  if (i = (ℓ–1) or i =(ℓ^2–ℓ))
    $\mathbb{O}$ = bitshift($\mathbb{O}$, $\mathcal{K}[(\ell^2 - 1) - i]$, decryption)
  end if
  $\mathbb{O}$ = key_xor($\mathbb{O}$, $\mathcal{K}$, (i mod 256), decryption)
  if (i mod 2 = 0)
    ca = $\mathbb{O}$[mc][mc][mc]
  else
    for j = 0 to 5 do
      ca[j] = $\mathcal{K}[mo+ j]$
    end for
  end if
  if (ca[0] mod 4 = 3)
    $\mathbb{O}$ = x_move($\mathbb{O}$, 4 – (ca[1] mod 4), ca[2])
    $\mathbb{O}$ = y_move($\mathbb{O}$, 4 – (ca[1] mod 4), ca[2])
    $\mathbb{O}$ = z_move($\mathbb{O}$, 4 – (ca[1] mod 4), ca[2])
  else if(ca[0] mod 4 = 2)
    $\mathbb{O}$ = x_move($\mathbb{O}$, 4 – (ca[1] mod 4), ca[2])
  else if(ca[0] mod 4 = 1)
    $\mathbb{O}$ = y_move($\mathbb{O}$, ca[1], ca[2])
  else if(ca[0] mod 4 = 0)
    $\mathbb{O}$ = z_move($\mathbb{O}$, ca[1], ca[2])
  end if
  if (i mod 2 = 0)
    $\mathbb{O}$[mo][mo][mo] = ca
  end if
end for

29: return 𝕆

7.3.4. Şifreleme ve Deşifreleme

7.3.4.1. Şifreleme fonksiyonu - encryption

$\mathcal{P}_n$ açık veri dizisi, $\mathcal{K}_n$ anahtar veri dizisi, ℓ $\{3, 5, \dots, 21\}$ aralığında küp katmanları değeri, AC asimetrik çıktıda kullanılacak algoritmayı işaret eden değer ve SC ise AC değerinin 0 olması durumunda özelleştirilmiş çıktı için $\{0, \dots, 255\}$ aralığında bir değerdir.

Öncelikle, $\mathcal{P}_n$ veri dizisi, ℓ küp katman değerine gore oluşturulacak parçalara bölüneceğinden, şifreleme işlemi sırasında oluşturulacak β blokların sayısını temsil eden tb değeri belirlenir. $\ell = 3 \Rightarrow 2$ veya $3 < \ell \Rightarrow 3$ olmak üzere, veri boyutunu ve $randval$ fonksiyonuyla üretilecek rastlantısal sayıyı yerleştirmek için her β bloğunda alan ayrılacaktır, bu alan fonksiyon içerisinde rf ile temsil edilecektir. Bu durumda $\text{len}(\mathcal{P}) \leq (6\ell^3 - \text{rf})$ şartında 1 adet $\mathcal{V}$ bloğu yeterli olacaktır, ancak $(6\ell^3 - \text{rf}) < len(\mathcal{P})$ şartında daha fazla blok kullanılması gerekecektir.

Devamında anahtar verisi dizisi $keyexpansion$ fonksiyonuna gönderilerek $48\ell^3$ - bitlik anahtar üretilecektir. Bu yeni anahtar dizisi $\mathcal{K}_n$ ile temsil edilecektir.

Yukarıda yer alan fonksiyon ile tb değeri belirlendikten sonra $\mathcal{P}_n$ dizisinin değerleri $6\ell^3 - rf$ sayısınca veri içeren gruplar halinde oluşturulan β bloklarına yerleştirilecek ve $cubemixer$ fonksiyonundan geçirildikten sonra $\mathcal{C}_n$ dizisine aktarılacaktır.

AC ve SC değerleri, $encryption$ fonksiyonunca çağırılan $putunits$ ve $putunits$ fonksiyonunca çağırılan $randval$ fonksiyonunca şifreleme işlemine dahil edilir. rc

değeri $randval$ fonksiyonu için bir giriş değeridir ve *randval* fonksiyonunun her çağırımında bir önceki değerden farklı bir değer gönderebilmesi için kullanılır.

Şifreleme işlemi için kullanılacak *encryption* fonksiyonu aşağıda verilmiştir.

Algoritma 16: CUBE - encryption

Variables: **rf ;**// takes 2 for ℓ=3 and takes 3 for $\ell > 3$ to represent the plaintext size and the number of cubie surfaces allocated to store the random or SC value.

$\boldsymbol{\beta_{x,y,z,f}}$ **;**// is the multidimensional block array in which to place the plaintext array $\mathcal{P}_n$, given that $\ell = x = y = z$and f={$f_0, \ldots, f_5$}.

len($\mathcal{P}$) ;// is the value representing the size of the $\mathcal{P}_n$array.

tb ;//is the number of blocks required to encrypt the $\mathcal{P}_n$ array.

bs ;// is the value representing the size of tb blocks.

$\boldsymbol{ca_n}$ **;**//is the array where $\mathcal{P}_n$ partitions will be stored temporarily by the number of tb and size of BS.

pc, cc, rc ;//pc is counter value for $\mathcal{P}_n$ array values that have been processed, cc is counter value for processed $\mathcal{C}_n$ array values and rc is counter value for randval calls.

bpl ;// represents the length of the $\mathcal{P}_n$array in the block.

Inputs: $\boldsymbol{\mathcal{P}_n}$ **;**// is the plaintext array whose elements are 8-bit values.

$\boldsymbol{\mathcal{K}_n}$ **;**// is the key array whose elements are 8-bit values.

AC ;// If AC = 0, the ciphertext output is obtained depending on the SC value. If AC ≥ 1, the ciphertext output is obtained according to the random number algorithm represented in the randval by AC.

SC ;// is the value representing 256 specific ciphertexts when AC = 0.

$\boldsymbol{\ell}$ **;**// represents the number of x, y, and z layers of the cube block to be used in encryption, given that $3 \leq \ell \leq 21$; ℓ mod 2 = 1 and$\ell \in Z^+$.

Outputs: $\boldsymbol{\mathcal{C}_n}$ **;**// is the array representing ciphertext.

```
01: 𝒦 = keyexpansion(𝒦_n);  tb= 0;  bs = 0;  pc = 0;  cc = 0;  rc = 0;  bpl = 0;
02: if ( ℓ = 3 )
03:     rf = 2
04: else
05:     rf = 3
```

```
06: end if
07: if (len(𝒫) ≤ 6ℓ^3 - rf)
08: ¦tb = 1
09: else
10: ¦tb = (len(𝒫)  - (len(𝒫) mod (6ℓ^3 - rf))) / (6ℓ^3 - rf)
11: ¦if (len(𝒫) mod (6ℓ^3 - rf) > 0)
12: ¦ ¦b = tb + 1
13: ¦end if
14: end if
15: for i = 0 to (tb – 1) do
16: ¦if ((len(𝒫) – pc) ≥ (6ℓ^3 - rf))
17: ¦ ¦bs = 6ℓ^3 - rf
18: ¦else
19: ¦ ¦bs = len(𝒫) – pc
20: ¦end if
21: ¦for j = 0 to (bs - 1) do
22: ¦ ¦if (len(𝒫) > pc)
23: ¦ ¦ ¦ca[j] = 𝒫[pc];    pc = pc + 1;
24: ¦ ¦else
25: ¦ ¦ ¦ca[j] = randval(rc, AC, SC);    rc = (rc+1) mod 256
26: ¦ ¦end if
27: ¦end for
28: ¦β = putcubies(ca, AC, SC);    β = cubemixer(β, 𝒦)
29: ¦for y = 0 to (ℓ- 1) do
30: ¦ ¦for z = 0 to (ℓ-1) do
31: ¦ ¦ ¦for x = 0 to (ℓ-1) do
32: ¦ ¦ ¦ ¦for f = 0 to 5 do
33: ¦ ¦ ¦ ¦ ¦𝒞[cc] = β[x][y][z][f];    cc = cc + 1
34: ¦ ¦ ¦ ¦end for
35: ¦ ¦ ¦end for
36: ¦ ¦end for
37: ¦end for
38: end for
39: return 𝒞
```

7.3.4.2. Şifre çözme fonksiyonu – decryption

Şifreleme fonksiyonu olan $encryption$ ile deşifreleme fonksiyonu olan $decryption$ arasında ortak değerler olan $\mathcal{K}_n$ ve ℓ deşifreleme için gerekli olan verilerdir. Ancak şifreleme işleminde kullanılan ve çıktı sonuca etki eden AC ve SC değerleri deşifreleme işlemi için gerekli değildir. AC ve SC değerlerinin şifreli veri üzerindeki değişimi, anahtar veri dizisi olan $\mathcal{K}_n$ dizisinin bilinmesi durumunda $decryption$ fonksiyonu ve alt fonksiyonlarca geri alınır. Rastlantısal veya seçimli olarak oluşturulan $randval$ fonksiyonu değerleri $\mathcal{C}_n$ şifreli veri dizisi içerisine ilk şifreleme aşamasında dahil edilir.

$decryption$ fonksiyonunun çalışması için ℓ değerine bağlı olarak bir rf değeri belirlenir. Devamında $\mathcal{K}_n$ anahtar veri dizisi $keyexpansion$ fonksiyonuna gönderilerek genişletilir. Toplam blok sayısını temsil eden tb değeri $len(\mathcal{C})$ değerine bağlı olarak bulunur. $tb = len(\mathcal{C})/6\ell^3$ olmak üzere $\mathcal{C}_n$ dizisi $6\ell^3$ veriden oluşan tc sayısınca parçalar halinde deşifre edilir. Deşifreleme işleminde key_xor ve $cubedemixer$ fonksiyonlarından faydalanılır. $decryption$ fonksiyonu aşağıda verilmiştir.

Algoritma 17: CUBE - decryption

Variables:
rf ;//takes 2 for ℓ=3 and takes 3 for $\ell > 3$ to represent the plaintext size and the number of cubie surfaces allocated to store the random or SC value.
$\boldsymbol{\mathcal{K}_n}$ **;//** is the $6\ell^3$ sized array representing the expanded $\mathcal{K}_n$ array.
$\boldsymbol{\beta_{x,y,z,f}}$ **;//** is the multidimensional block array in which to place the plaintext array $\mathcal{P}_n$, given that $\ell = x = y = z$and f=$\{f_0, \dots, f_5\}$.
len($\mathcal{C}$) ;// is the value representing the size of the $\mathcal{C}_n$ array.
tb ;//is the number of blocks in $\mathcal{C}_n$ depending on ℓ.
bpc ;// represents the length of decrypted$\mathcal{P}_n$array.
mc ;// is the x, y, z coordinate values of core cubie
tpc, cc ;//tpc is the counter representing the number of values inserted to the $\mathcal{P}_n$

array and the position of the next value in the $\mathcal{P}_n$ array. cc is the counter used for the processed $\mathcal{C}_n$ array values.

bpl ;// represents the length of the $\mathcal{P}_n$ array in the block.

Inputs: $\boldsymbol{\mathcal{C}_n}$ **;**// is the ciphertext array whose elements are 8-bit values.

$\boldsymbol{\mathcal{K}_n}$ **;**// is the key array whose elements are 8-bit values.

$\boldsymbol{\ell}$ **;**// represents the number of x, y, and z layers of the cube block to be used in encryption, given that $3 \leq \ell \leq 21$; ℓ mod 2 = 1 and $\ell \in Z^+$.

Outputs: $\boldsymbol{\mathcal{P}_n}$**;**// represents the decrypted plaintext data block.

01: $\mathcal{K}$ = keyexpansion($\mathcal{K}_n$); tb = len($\mathcal{C}$) / $6\ell^3$;
02: mc = (ℓ-1)/2; bpc = 0; tpc = 0; cc = 0; bpl = 0;
03: if ($\ell = 3$)
04: ¦ rf = 2
05: else
06: ¦ rf = 3
07: end if
08: tb = len($\mathcal{C}$) / $6\ell^3$
09: for i = 0 to (tb – 1) do
10: ¦ for y = 0 to (ℓ- 1) do
11: ¦ ¦ for z = 0 to (ℓ-1) do
12: ¦ ¦ ¦ for x = 0 to (ℓ-1) do
13: ¦ ¦ ¦ ¦ for f = 0 to 5 do
14: ¦ ¦ ¦ ¦ ¦ β[cc] = $\mathcal{C}$[x][y][z][f]; cc = cc + 1
15: ¦ ¦ ¦ ¦ end for
16: ¦ ¦ ¦ end for
17: ¦ ¦ end for
18: ¦ end for
19: ¦ β=cubedemixer(β, $\mathcal{K}$)
20: ¦ bpl= β[mc][mc][mc][1] ⊕ β[mc][mc][mc][0]
21: ¦ if $\ell > 3$ then
22: ¦ ¦ bpl= (bpl ≪ 8) ∨ (β[mc][mc][mc][2] ⊕ β[mc][mc][mc][0])
23: ¦ end if
24: ¦ BPC = 0
25: ¦ for y = 0 to (ℓ- 1) do
26: ¦ ¦ for z = 0 to (ℓ-1) do

```
for x = 0 to (ℓ-1) do
    for f = 0 to 5 do
        if(bpl > bpc) then
            if(x=y=z=mc) then
                if(f > (rf - 1) then
                    P[tpc] = β[x][y][z][f] ⊕ β[mc][mc][mc][0];  tpc = tpc + 1;  bpc = bpc+ 1
                end if
            else
                P[tpc] = β[x][y][z][f] ⊕ β[mc][mc][mc][0];  tpc = tpc + 1;  bpc = bpc+1
            end if
        end if
    end for
end for
end for
return P
```

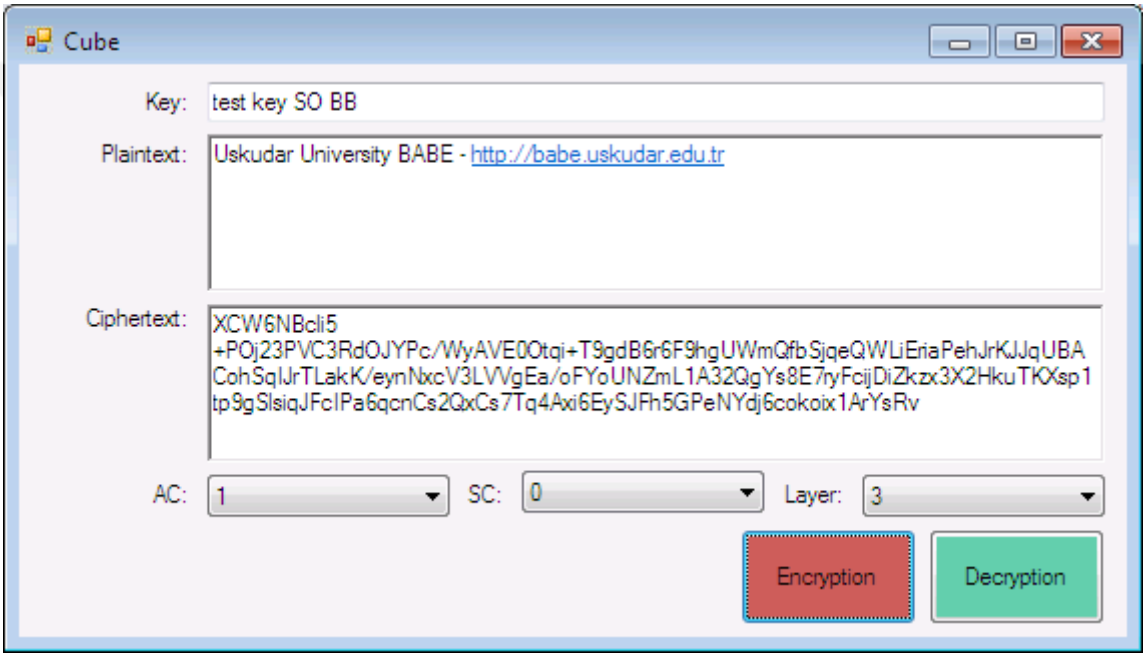
7.4. Uygulama

Önceki bölümlerde Cube tasarımı tanımlanmış ve bu bölümde Cube tasarımın uygulanması deneylenmiştir.

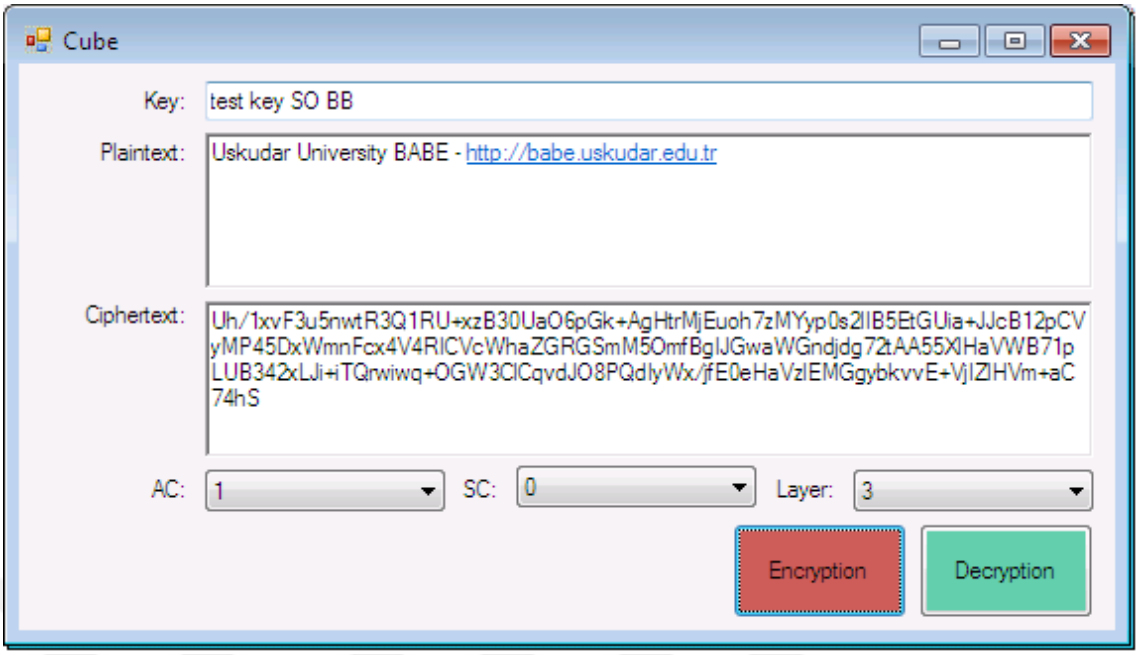
Cube'ün geniş anahtar boyu ve birden fazla fonksiyonla oluşturulan şifreleme tasarımı kişisel bilgisayarlardan daha küçük işlemci ve bellek donanıma sahip PIC ve diğer mikro denetleyici kitleriyle uygulanabilir değişkenlere sahiptir.

Bu çalışmanın yapıldığı ve tarihte birçok mikro işlemci mimarisinde ve mikro denetleyici kitlerinde standart olarak kullanılan ANSI C programlama dilinde Cube algoritmaları kodlanmış ve başarıyla uygulanmıştır. Oluşturulan ANSI C kodu işletim sistemi platformundan bağımsız olarak, Windows ve linux platformlarında başarıyla

çalıştırılmıştır. Aynı algoritmanın masaüstü uygulaması için Microsoft Visual Studio 2012 ve Microsoft .NET Framework 4.5.5 ile C# dilinde hazırlanan kodun da başarıyla çalışmış ve devam eden bölümlerdeki analiz uygulamaları bu kod ile gerçekleştirilmiştir. Şekil 40 ile CUBE algoritma ve analiz sürecinde kullanılan ara yüz ve Şekil 41 ile şifreleme aynı düz metin ve anahtar ile yapılan şifreleme işleminde farklı ciphertext elde edildiği görülmektedir.



Şekil 40: CUBE – Windows platformu uygulaması, 1



Şekil 41: CUBE – Windows platformu uygulaması, 2

7.5. Analizler

Buradaki güvenlik ve performans analizleri Intel Core i5 – 7400 3.00 GHz 64-bit işlemci, 8.00 GB of RAM (useable: 7.89 GB) donanımı ve Microsoft Windows 10 Pro 64-Bit (release: 10.0.1.17134) işletim sistemi ile gerçekleştirilmiştir. Bu platform üzerinde C# programlama dili kodu Microsoft Visual Studio 2012 arayüzü ve derlemesi ile kullanılmıştır.

7.5.1. Güvenlik

Bu bölümde Cube için performans analizleri gerçekleştirilmiştir. Bu analizlerden elde edilen sonuçların tekrar gözlemlenebilmesi için π ve e sayısının ilk 444528-biti Mathematica düz metin ve anahtar için kullanılmıştır ($\ell = 21 \Longrightarrow 48\ell^3 = 444528$). Burada bit dizileri $\pi = \pi_0, \pi_1, \ldots, \pi_{n-1}$ ve $e = e_0, e_1, \ldots, e_{n-1}$ olarak temsil edilmiştir.

Burada Cube güvenliği için ölçülecek kriterler confusion ve diffusion başarımları olacaktır.

Confusion, ciphertextin her bir bitinin parçalar halinde ve farklı yollardan anahtara bağımlı olmasını ve ikisi arasındaki bağıntının saklanmasını/gizlenmesini ifade eder. Aynı anahtar ile şifrelenmiş büyük boyutlu ve çok sayıda şifreli metin elde edildiğinde bile, bu veri yığınından anahtarın elde edilmesini engellemek confusion ile sağlanır.

Diffusion, düz metnin sadece tek bir bit değerinin değişmesinin, şifreli metin bitlerinin yarısını değiştireceğini ifade eder. Benzer olarak şifreli metinde yapılan tek bir değişim, aynı şekilde düz metin bitlerinin yarısının değişmesi anlamına gelir.

Confusion şifreli metinden anahtara olabildiğince complex dağılım oluşturmalıdır. Diffusion, düz metnin şifreli metin ciphertext üzerinden istatistiksel bilgi alınabilecek yapılarını engellemelidir.

Burada aynı sırada ve konumda bulunan bit değerlerinin değişenleri (changed) ve çakışanları (collision) analiz edilmiştir. Analizler için ciphertext difference rate (CDR) ve unified average changing bytes (UACB) testlerinden elde edilen değerler hesaplanmıştır, bu testler literatürde kriptosistemlerin testlerinde kullanılan yöntemler arasındadır (36-38). Ek olarak Byte Collision Mean (BCM) değeri çakışmaları test etmek için kullanılmıştır.

CDR yöntemi şifreli metin üzerindeki değişimlerin ölçümü için kullanılan bir yöntemdir ve değerin büyük olması istenilen sonuçtur (53). UACB [0, 1] aralında bir değeri verir ancak literatürde ideal bir ortalama verilmemiştir.

Literatürde sıkça başvurulan UACI, test yöntemi image kriptosistem yöntemlerinin analizinde kullanılan ve 8-bitlik pixel değerlerinin ölçümüne dayalı bir

yöntemdir (36-38). Ancak bu test yönteminin 8-bitlik pixellere odaklı olması aynı testlerin aynı bit uzunluğuna sahip byte ölçümleri içinde kullanılmasını mümkün kılar. UACB değerinin yüksek olması değişim oranının yüksek olduğunu gösterir ve istenilen sonuçtur.

Burada C orjinal düz metin ve anahtardan elde edilen şifreli metni, C_1 ve C_2 ise anahtar veya düz metin üzerinde kontrollü düzenlemeler sonucu elde edilen şifreli metinleri, $\mathcal{P}$ düz metin ve enc şifreleme işlemini temsil ettiğinde, UACB, CDR ve BCM değerleri aşağıdaki formüller ile elde edilir:

$$C = enc(\mathcal{P}, K) \quad (115)$$

$$C_1 = enc(\mathcal{P}^1, \mathcal{K}^1) \quad (116)$$

$$C_2 = enc(\mathcal{P}^2, \mathcal{K}^2) \quad (117)$$

$$Diffp\big(A(i), B(i)\big) = \begin{cases} 1, A(i) \neq B(i) \\ 0, A(i) = B(i) \end{cases} \quad (118)$$

$$Diff(A, B) = \sum_{i=0}^{6\ell^3-1} Diffp(A(i), B(i)) \quad (119)$$

$$Colp\big(A(i), B(i)\big) = \begin{cases} 0, A(i) \neq B(i) \\ 1, A(i) = B(i) \end{cases} \quad (120)$$

$$Col(A, B) = \sum_{i=0}^{6\ell^3-1} Colp(A(i), B(i)) \quad (121)$$

$$CDR = \frac{Diff\ (C,C_1)+Diff\ (C,C_2)}{2.6\ell^3} \quad (122)$$

$$\mathrm{B}CM = \frac{Col\,(C,C_1)}{6\ell^3} \quad (123)$$

$$\mathrm{UACB} = \frac{1}{6\ell^3}\left(\sum_{i=0}^{6\ell^3-1} \frac{|C(i)-C_1(i)|}{255}\right) \quad (124)$$

Korelâsyon analizi, en genel anlamıyla değişkenler arasındaki ilişkilendirme derecesini ölçme yöntemidir. Kriptografide korelâsyon analizinden elde edilen sonuçlar, kriptografik yönteme yapılacak analiz saldırıları için çok önemli bilgiler sağlayabilmektedir. Bu sebeple kriptosistem yönteminin bütün giriş değerlerinin çıkış değerleri üzerindeki etkisinin saklanması önemlidir (10, 30, 50-52).

Bu analizin yapılması için Pearson Korelâsyon yöntemi kullanılmıştır. Düzmetin $\mathcal{P}$ ve şifreli metin $\mathcal{C}$ olduğunda, $s_{\mathcal{P}}$ a'nın standart sapma değeri, $s_{\mathcal{C}}$ $\mathcal{C}$'nin standart sapma değeri, $m_{\mathcal{C}}$ $\mathcal{C}$'nın aritmetik ortalama değeri ve $m_{\mathcal{P}}$ $\mathcal{P}$'nin aritmetik ortalama değeri olmak üzere r korelâsyonu $r = \frac{\sum_{i=0}^{6\ell^3-1}(\mathcal{P}_i - m_{\mathcal{P}})(\mathcal{C}_i - m_{\mathcal{C}})}{(6\ell^3-2)s_{\mathcal{P}}s_{\mathcal{C}}}$ formülü ile hesaplanmıştır.

7.5.1.1. Kriptoanaliz direnci

Kriptanaliz, sadece şifreli veri, bilinen açık veri, seçilmiş açık veri ve seçilmiş şifreli veri sınıflarında yapılır. Sadece şifreli veride saldırgan verisini kontrol eder. Bilinen açık veride saldırgan hem açık veriyi hem şifrelenmiş veriyi kontrol eder. Seçilmiş şifreli veride saldırgan şifreleme algoritmasına sahip olup istediği şifrelenmiş verileri ve düz veri çıkışlarını seçer. Seçilmiş açık veride saldırgan şifreleme algoritmasına sahip olup istediği açık verileri ve şifreli verileri seçer.

Bir başka kriptanaliz yöntemi de Vigenere için kullanılan Kasiski testidir. Bu yöntemde 3 karakterli özdeş bölüm çiftleri bulunur. Bulunan iki bölümün(parts) başlangıç(initial) durumları tespit edilir. Farklı(different) başlangıç değerleri bulunursa, Anahtar uzunluğu için bu farklı başlangıç değerlerin $EBOB$ unu böldüğü var sayılır.

Burada örneklendirildiği gibi kriptanaliz yöntemleri açık verinin, şifreli veri üzerindeki etkileri ve istatistiksel analizlere dayanır. Belirli tekrarların gözlemlenmesi

ve giriş değerleri şifreli veri üzerindeki etkilerinin gözlemlenmesi bütün kriptanaliz yöntemlerinin temelini oluşturur.

Cube için bilinen kriptanaliz ve diferansiyel saldırıları anlamlı zaman bir süre içerisinde sonuç vermez. Cube asimetrik şifreli veri üretimiyle, saldırganın analiz yöntemlerine karşı direnç sağlar. Asimetrik şifreli veri, saldırganın analiz ettiği verinin hangi kümeye veya açık veriye ait olduğunu bulmasına karşı direnç sağlar. Ayrıca birden fazla tur döngüsü, verilerin her seferinde yeniden konumlandırılması ve turlara özgü anahtar verisi yüksek yayılım(diffusion) ve karıştırmayı(mix) sağlayarak diferansiyel saldırılarını etkisiz kılmayı amaçlar.

7.5.1.2. Confusion ve Diffusion:

Cube, karmaşıklığı kendisine özgül *x-y-x move*, *bitshift* ve *cubie_xor* fonksiyonlarını, *cubemixer* fonksiyonuyla çağırarak sağlar. *cubemixer* fonksiyonunun blok halindeki verileri iterative olarak işleme alması ve parametrelerini önceki turdan aktarılan düzmetin bloğundan veya anahtardan alması Cube'ün karmaşıklığını sağlar.

Substitution bit değerlerinin başka bit değerleriyle yeniden konumlandırılması ve permütasyon bit değerleri konumlarının bazı algoritmalara göre farklılaşmasıdır. Burada algoritmaya bağımlılık ve parametrik değer bağımlılığı olmayan permütasyon uygulaması şifrelemenin statik yapıya sahip bölümünü de oluşturur. Bu durumda substitution ve permutation, ancak sabit bir değer değişimi ve bu değerlerin konumlarının sabit bir yapıyla değişimini sağlar.

Cube doğrudan substitution ya da permutation olarak adlandırılmayacak yeni bir yöntem kullanır. Düz metne ait veri bloğu, iterative olarak *cubemixer* fonksiyonunda işlenir. Burada *i* tur sayacı olduğunda, *i* çift sayıysa (0 dahil) plaintexte bağımlı bloktan

alınan değerlerle, i tek sayıysa cipher değerlerine göre mix fonksiyonları olan *x-y-z move* çağırılır. $i = \ell - 1$ ve $i = \ell^2 - \ell$ olduğunda bitshift fonksiyonu çağırılır. Cubeun iterative yapıda ve bir önceki tura ait blok değerlerine bağımlı bu yapısı, permutation ve substitution olarak bilinen yapılardan daha etkin karmaşıklık yapısı sağlar.

Confusion ve diffusion değerlerinin ölçümleri için, analiz edilecek parametreye ($\mathcal{P}$-Düz metin, $\mathcal{K}$-Anahtar, AC-Asimetrik Ciphertext, SC-Specific Ciphertext, ℓ -Katman) özgü değerler seçilerek bir referans şifreli metin elde edilmiştir. Devamında ilk değerler üzerinde kontrollü düzenlemeler yapılarak yeni şifreli metinler elde edilmiştir. Elde edilen şifreli metin bir önceki düzenleme sonucuyla karşılaştırılarak parametrelerin ciphertext üzerindeki etkileri ölçülmüştür.

7.5.1.3. İstatistiksel Rastlantısallık Analizi

Rastlantısallık Pseudorandom Number Generators (PNRG) ve şifreleme yöntemleri için vazgeçilmez bir unsurdur. Kriptoanalize karşı direncin en önemli unsuru ciphertextin rastlantısal yapıda üretilmesidir. Rastlantısallık kriteri genel kabul görmüş bir dizi istatistiksel test ile ispat edilmelidir. National Institute of Standards and Technology (NIST) nin bu deneylerin gerçeklenmesi için yayınladığı test suiti en sık başvurulan test ortamını sunmaktadır. Bu bölümde Cubeün bu test suitinden elde edilen analiz sonuçları ele alınmıştır.

Bu test için NIST Test Suitinin ön tanımlı Frequency, Block Frequency, Cusum-Forward, Cusum-Reverse, Runs, Long Runs of Ones, Rand, Spectral DFT, Non-overlapping Templates (m = 9, B = 000000001), Overlapping Templates ($m = 9$), Universal, Approximate Entropy ($m = 10$), Random Excursions (x= +1), Random

Excursions Variant ($x = -1$), Linear Complexity ($M = 500$) ve Serial ($m = 16, \nabla\Psi_m^2$) parametreleri kullanılmıştır.

Bu test için π sayısının ilk $24\ell^3$ bit bit değeri seçilmiş ve AC=0 SC=0,..,255 parametreleriyle 256 ayrı şifreleme yapılarak sonuçları test süitinde analiz edilmiştir.

Tablo XXVIII: CUBE - NIST test süit

		P-value for each ℓ									
#	**Test (T)**	**3**	**5**	**7**	**9**	**11**	**13**	**15**	**17**	**19**	**21**
1	Frequency	0,60	0,67	0,74	0,50	0,51	0,50	0,55	0,62	0,56	0,57
2	Block Frequency	0,43	0,50	0,57	0,33	0,34	0,33	0,38	0,48	0,39	0,40
3	Cusum-Forward	0,64	0,71	0,78	0,54	0,55	0,54	0,59	0,66	0,60	0,61
4	Cusum-Reverse	0,67	0,74	0,81	0,57	0,58	0,57	0,62	0,68	0,63	0,64
5	Runs	0,46	0,53	0,61	0,36	0,37	0,36	0,41	0,51	0,43	0,43
6	Long Runs of Ones	0,13	0,20	0,27	0,13	0,14	0,13	0,07	0,23	0,09	0,10
7	Rand	0,18	0,25	0,32	0,18	0,19	0,18	0,12	0,27	0,14	0,15
8	Spectral DFT	0,11	0,19	0,26	0,12	0,12	0,11	0,06	0,22	0,08	0,08
9	Non-overlapping Templates (m = 9, B = 000000001)	0,25	0,32	0,39	0,15	0,16	0,15	0,19	0,33	0,21	0,22
10	Overlapping Templates (m = 9)	0,36	0,43	0,50	0,26	0,27	0,26	0,31	0,42	0,32	0,33
11	Universal	0,68	0,75	0,82	0,58	0,59	0,58	0,62	0,69	0,64	0,65
12	Approximate Entropy (m = 10)	0,41	0,49	0,56	0,32	0,32	0,31	0,36	0,47	0,38	0,38
13	Random Excursions (x= +1)	0,83	0,90	0,97	0,73	0,73	0,73	0,77	0,81	0,79	0,80
14	Random Excursions Variant (x= -1)	0,76	0,83	0,90	0,66	0,66	0,65	0,70	0,75	0,72	0,72
15	Linear Complexity (M = 500)	0,32	0,39	0,47	0,23	0,23	0,22	0,27	0,39	0,29	0,29
16	Serial (m = 16, $\nabla\Psi_m^2$)	0,23	0,30	0,37	0,13	0,14	0,13	0,18	0,31	0,19	0,20

Tablo XXVIII'de yer alan her bir test için olasılık değerinin (p value) en az $\geq 00{,}1$ olması, o test için rastlantısallık kriterinin sağlandığını gösterir. Tablo XXVIII'de yer alan test sonuçlarında 0,06 - 0,97 aralığında değerler alındığı görülmektedir, bu değerler Cube'ün rastlantısallık kriterini yerine getirdiğini göstermektedir.

7.5.1.4. Anahtar hassasiyeti ve confusion etkisi

Anahtarın şifreli metin üzerindeki confusion etkisinin ölçülmesi için π sayısının ilk $48\ell^3 - (8.rf)$ biti düzmetin ve e sayısının ilk $48\ell^3$-biti anahtar olarak seçilmiştir. Asimetrik şifreli metin (AC=1) seçilmiş ve 256 ayrı şifreleme yapılmıştır, devamında aynı $\mathcal{P}$ ve $\mathcal{K}$ ile AC=0 olarak seçilmiş ve SC=0,…,255 aralığı için 256 ayrı şifreleme yapılmıştır.

Bu test için aşağıdaki adımlar izlenmiştir.

Tur sayacı i=0 olarak tanımlanmıştır. π sayısının ilk $48\ell^3 - (8.rf)$ biti düzmetin ve e sayısının ilk $48\ell^3$-biti anahtar olarak seçilmiştir.

1. Orijinal düz metin ve anahtar ile şifreleme yapılmıştır

2. Şifreleme sonrası anahtarın i. biti bire tümlenmiş ve yeniden şifreleme yapılmıştır.

3. Şifreleme sonrası anahtarın sondan i. biti bire tümlenmiş ve yeniden şifreleme yapılmıştır

4. 1 – 3 arası adımlardan elde edilen 3 şifreli metinden elde edilen CDR ve UACB değerleri karşılaştırılmıştır.

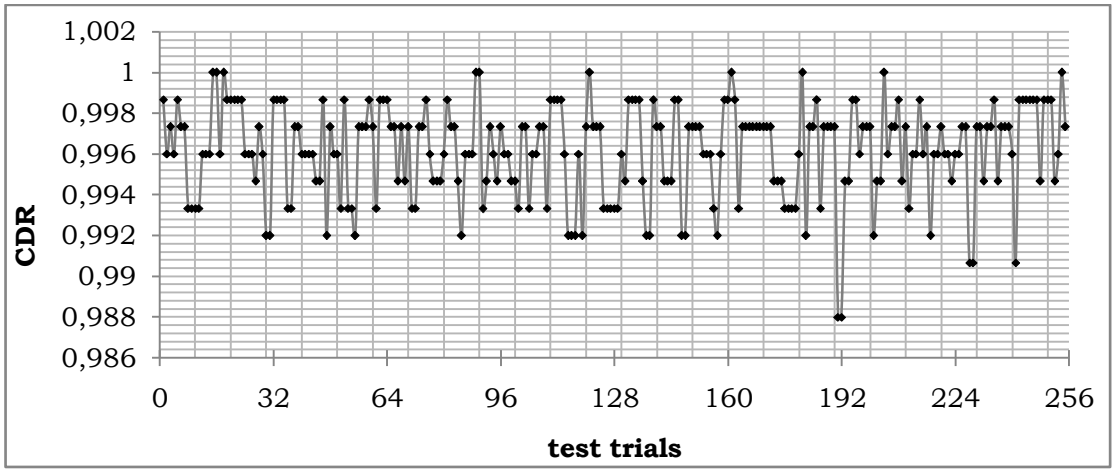
5. 1-4 aralığındaki adımlar 255 defa tekrarlanmıştır.

Bu test ile elde edilen 255 şifreleme değerinden CDR ve UACB sonuçları aşağıda verilmiştir

Tablo XXIX: CUBE - Anahtar confusion testi - CDR ve UACB

ℓ	**Asimetrik Şifreli Metin AC= 1**		**Özgülleştirilmiş Şifreli Metin AC=0 SC=0,…,255**	
	CDR	**UACB**	**CDR**	**UACB**
3	0,993871	0,339524	0,991115	0,290803
5	0,996083	0,332171	0,996156	0,321503
7	0,995922	0,348717	0,995922	0,331775
9	0,996140	0,339872	0,996001	0,339198
11	0,996126	0,342281	0,996238	0,334496
13	0,996119	0,335712	0,996841	0,323708
15	0,996113	0,332144	0,996131	0,338806
17	0,996132	0,341254	0,996014	0,332981
19	0,996110	0,321474	0,996148	0,332046
21	0,996093	0,335287	0,996171	0,338973

Cube'ün yüksek CDR ortalaması ve yüksek UACB değerleri verdiği Tablo XXIX ile görülmektedir. Bu tabloda Cube için en yüksek CDR ortalaması olan 0.996156 sonucu ℓ = 5 ve AC = 0 için elde edilmiştir, ve en yüksek UACB ortalaması olan 0.348717 sonucu ℓ = 7 ve AC = 1 için elde edilmiştir. Bu şifreleme tekrarlarında ℓ = 5, AC = 0 için elde edilen CDR dağılım grafiği Şekil 42 ile verilmiştir, burada CDR değerinin 15, 16, 18, 89, 90, 121, 161, 181, 204 ve 254 sayılı tekrarlarda olabilecek en üst değer 1'e ulaştığı görülmüştür. Bu veri bloğunun tamamen değiştiğini göstermektedir.



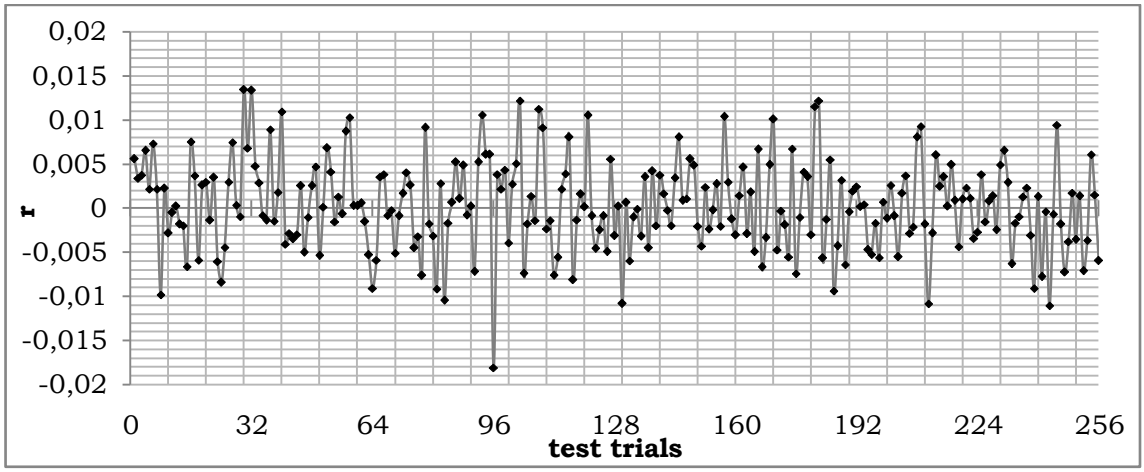
Şekil 42: CUBE - Anahtar testi, ℓ=5 ve AC=0 için CDR

Tablo XXX buradaki şifreleme tekrarları sırasında elde edilen korelâsyon analizi sonuçlarını vermektedir.

Tablo XXX: CUBE - Anahtar confusion test – Korelâsyon

ℓ	**Asymmetric Ciphertext AC= 1**	**Specific Ciphertext AC=0 SC=0,…,255**
3	-0,006554058340291090	-0,01781991021971461
5	0,001477486196922329	-0,00330903448441828
7	-0,001553027189816170	-0,00261553226662424
9	-0,001028495287000000	0,00060538633089747
11	-0,001137480071573750	-0,00223117501649412
13	-0,000407586549393169	0,00027565524806754
15	-0,001028495287000000	0,00060538633089747
17	-0,001867005885191420	-0,00055100059703884
19	0,000263624028574943	0,00035930793714124
21	-0,000294333877866922	-0,00030870579033093

Burada 0'a en yakın korelâsyon değeri 0.000263624028574943 ℓ= 19 ve AC = 1 için elde edilmiştir, bu değerin şifreleme tekrarları sırasında elde ettiği korelâsyon dağılım grafiği Şekil 43 ile verilmiştir.



Şekil 43: CUBE - Düz metin - Şifreli metin korelâsyonları, ℓ=19 ve AC= 1 için

Bu analiz sonuçlarında UACB ve CDR Cube'ün anahtar girişlerinde yüksek hassasiyet ile confusion ve diffusion sağladığını göstermektedir. Ayrıca korelâsyon analizinden elde edilen değerler anahtar girişindeki değişikliklerin düz metin ve şifreli metin arasında anlamlandırılamaz değerler verdiğini göstermiştir, bu sonuçlar anahtar saldırılarında korelâsyon analizine karşı yüksek direnç ve güvenlik sağlandığını göstermektedir.

7.5.1.5. Düz metin hassasiyeti ve Diffusion etkisi.

Düz metnin şifreli metin üzerindeki diffusion etkisinin ölçülmesi için π sayısının ilk $48\ell^3 - (8.rf)$ biti düzmetin ve e sayısının ilk $48\ell^3$-biti anahtar olarak seçilmiştir. Asimetrik şifreli metin (AC=1) seçilmiş ve 256 ayrı şifreleme yapılmıştır, devamında aynı $\mathcal{P}$ ve $\mathcal{K}$ ile AC=0 olarak seçilmiş ve SC=0,…,255 aralığı için 256 ayrı şifreleme yapılmıştır.

Bu test için aşağıdaki adımlar izlenmiştir.

Tur sayacı i=0 olarak tanımlanmıştır. π sayısının ilk $48\ell^3 - (8.rf)$ biti düzmetin ve e sayısının ilk $48\ell^3$-biti anahtar olarak seçilmiştir.

1. Orijinal düz metin ve anahtar ile şifreleme yapılmıştır

2. Şifreleme sonrası düz metnin i. biti bire tümlenmiş ve yeniden şifreleme yapılmıştır.

3. Şifreleme sonrası düz metnin sondan i. biti bire tümlenmiş ve yeniden şifreleme yapılmıştır

4. 1 – 3 arası adımlardan elde edilen 3 şifreli metinden elde edilen CDR UACB değerleri karşılaştırılmıştır.

5. 1-4 aralığındaki adımlar 255 defa tekrarlanmıştır.

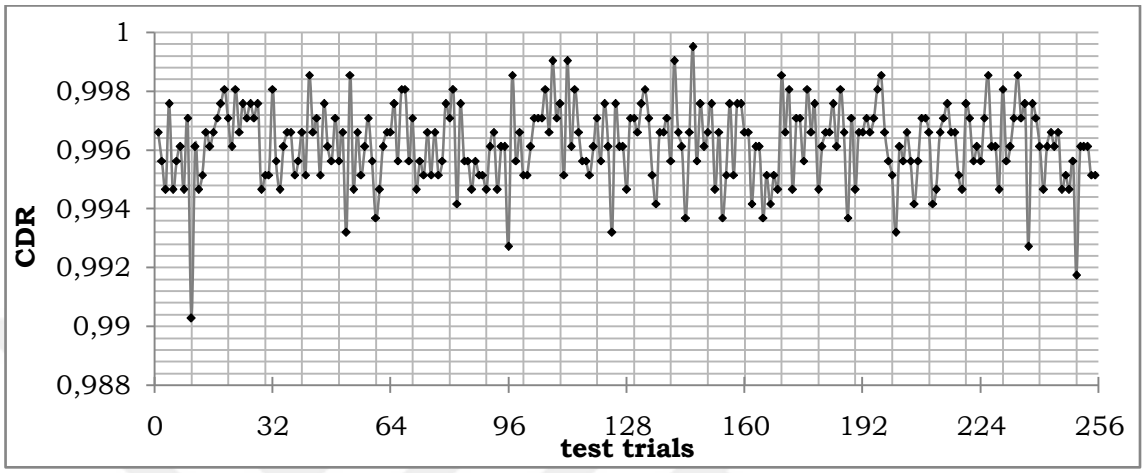
Bu test ile elde edilen 255 şifreleme değerinden CDR ve UACB sonuçları aşağıda verilmiştir.

Tablo XXXI: CUBE – Düz metin diffusion testi - CDR ve UACB

ℓ	**Asimetrik Şifreli Metin AC= 1**		**Özgülleştirilmiş Şifreli Metin AC=0 SC=0,…,255**	
	CDR	**UACB**	**CDR**	**UACB**
3	0,994504	0,392156	0,988211	0,275761
5	0,996156	0,397385	0,996135	0,396339
7	0,996188	0,379389	0,996164	0,391394
9	0,996099	0,384356	0,992196	0,386687
11	0,996096	0,390241	0,988298	0,379880
13	0,996094	0,389925	0,996083	0,392394
15	0,996045	0,392253	0,992208	0,389368
17	0,996093	0,391764	0,996086	0,390021
19	0,996134	0,390079	0,992228	0,392175
21	0,994504	0,392156	0,988211	0,275761

Cube'ün yüksek CDR ortalaması ve yüksek UACB değerleri verdiği Tablo XXXI ile görülmektedir. Bu tabloda Cube için en yüksek CDR ortalaması olan 0.996188

sonucu ℓ = 7 ve AC = 1 için elde edilmiştir, ve en yüksek UACB ortalaması olan 0.397385 sonucu ℓ = 5 ve AC = 1 için elde edilmiştir. Bu şifreleme tekrarlarında ℓ = 7 ve AC = 1 için elde edilen CDR dağılım grafiği Şekil 44 ile verilmiştir,



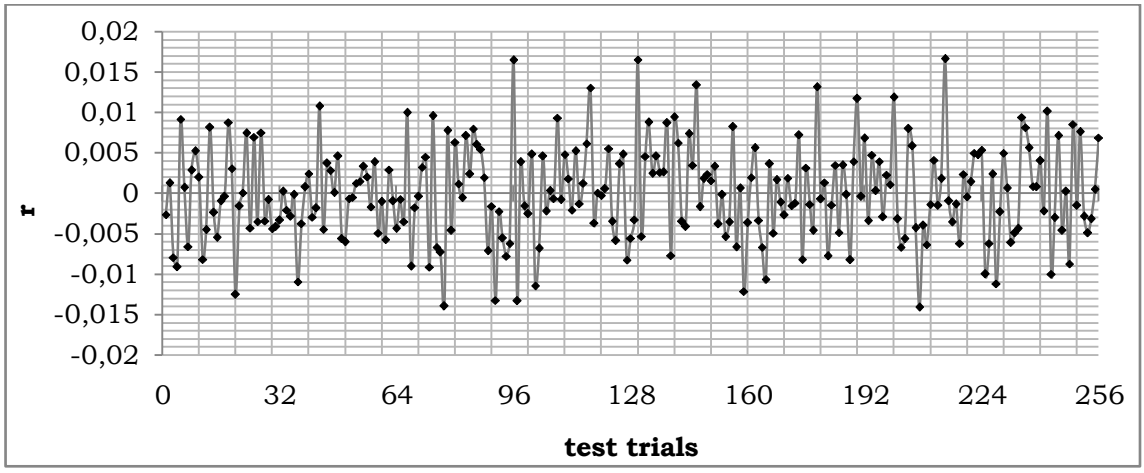
Şekil 44: CUBE - Düzmetin testi, ℓ=7 ve AC=1 için CDR

Tablo XXXII buradaki şifreleme tekrarları sırasında elde edilen korelâsyon analizi sonuçlarını vermektedir.

Tablo XXXII: CUBE - Anahtar confusion testi – Korelâsyon

ℓ	**Asymmetric Ciphertext AC= 1**	**Specific Ciphertext AC=0 SC=0,…,255**
3	-0,00552075176563079	-0,0158519475500013
5	-0,00109256504590659	-0,00208972844725515
7	-0,00131360986738595	0,000923229766396604
9	0,000666366579023829	-0,00182283811544155
11	-0,000509170401470381	-0,00189054619804284
13	-0,000287915986861504	0,000242351513525198
15	-0,000349411483853276	0,000268778362949565
17	0,000109185426130903	-0,00383621112627845
19	0,000231525720267845	-0,00018915882567173
21	-0,000291921737596026	-0,00038664642400475

Burada 0'a en yakın korelâsyon değeri 0.000109185426130903 ℓ = 17 ve AC = 1 için elde edilmiştir, bu değerin şifreleme tekrarları sırasında elde ettiği korelâsyon dağılım grafiği Şekil 45 ile verilmiştir.



Şekil 45: CUBE - Düz metin - Şifreli metin korelâsyonları, ℓ=19 ve AC= 1 için

Bu analiz sonuçlarında UACB ve CDR Cube'ün düz metin girişlerinde yüksek hassasiyet ile diffusion sağladığını göstermektedir. Ayrıca korelâsyon analizinden elde edilen değerler, düz metin girişindeki değişikliklerin düz metin ve şifreli metin arasında anlamlandırılamaz değerler verdiğini göstermiştir, bu sonuçlar düz metin ile gerçekleştirilecek saldırılarında korelâsyon analizine karşı yüksek direnç ve güvenlik sağlandığını göstermektedir.

7.5.1.6. Özgülleştirilmiş şifreli metin ve confusion etkisi

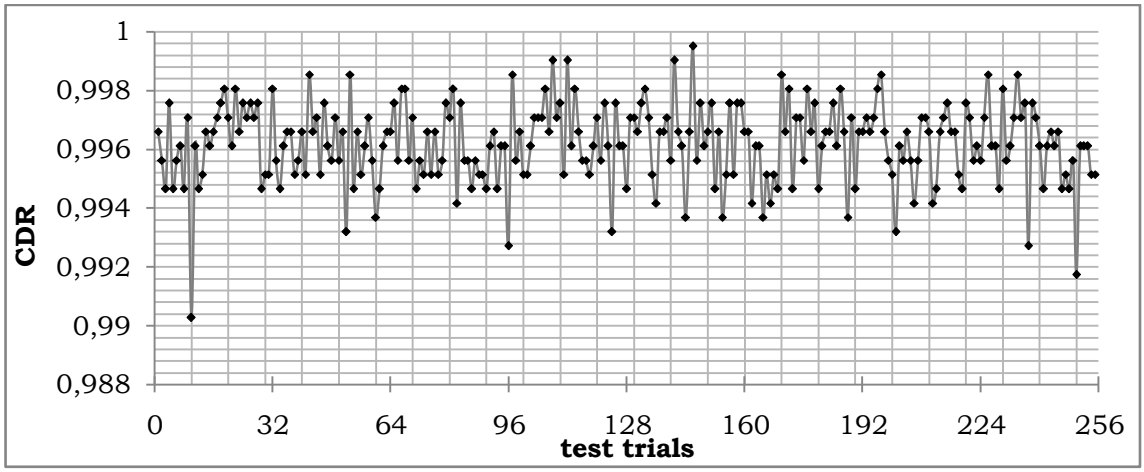
Asimetrik veya özgülleştirilmiş şifreli metin temsil değeri olan AC'nin şifreli metin üzerindeki diffusion etkisinin ölçülmesi için π sayısının ilk $24\ell^3$ biti düzmetin ve $\mathcal{K} = 0_{48\ell^3,\ldots,48\ell^3-1}$ anahtar olarak seçilmiştir. Asimetrik şifreli metin temsil değeri AC=0 olarak seçilmiş ve SC=0,…,255 aralığı için 256 ayrı şifreleme yapılmıştır.

Bu test için, tur sayacı i=0 olarak tanımlanmış, π sayısının ilk $24\ell^3$ biti düzmetin ve $\mathcal{K} = 0_{48\ell^3,\ldots,48\ell^3-1}$ anahtar olarak seçilmiştir. Elde edilen 255 şifreleme değerinden CDR ve UACB sonuçları aşağıda verilmiştir.

Tablo XXXIII. CUBE – AC diffusion testi - CDR ve UACB

ℓ	CDR	UACB
3	0,996054	0,450254
5	0,996062	0,396339
7	0,995998	0,392156
9	0,996078	0,398074
11	0,996117	0,396478
13	0,996087	0,395994
15	0,996069	0,391788
17	0,996093	0,388591
19	0,996100	0,389517
21	0,996095	0,389686

Cube'ün yüksek CDR ortalaması ve yüksek UACB değerleri verdiği Tablo XXXIII ile görülmektedir. Bu tabloda Cube için en yüksek CDR ortalaması olan 0.996117 sonucu ℓ = 11 için elde edilmiştir ve en yüksek UACB ortalaması olan 0.388591 sonucu ℓ = 3 için elde edilmiştir. Bu şifreleme tekrarlarında ℓ = 11 için elde edilen CDR dağılım grafiği Şekil 46 ile verilmiştir,



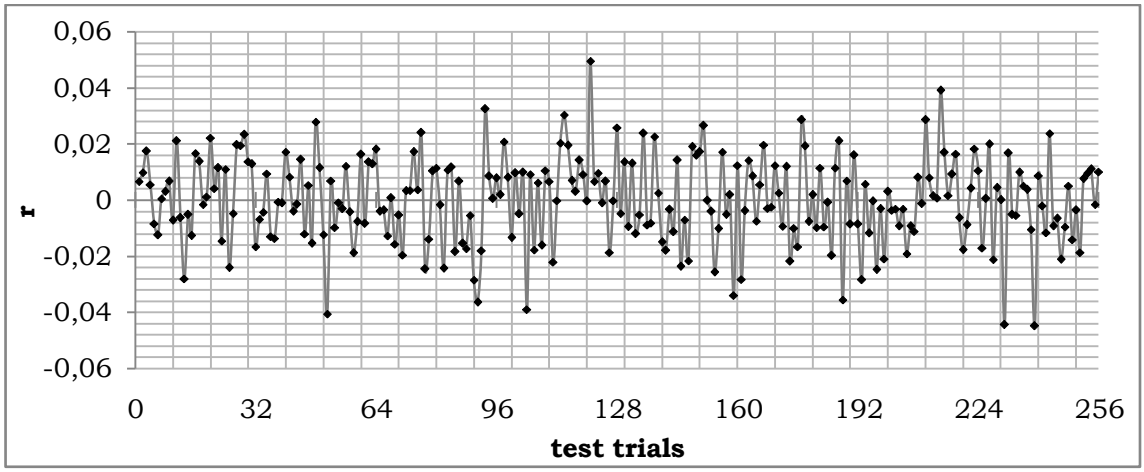
Şekil 46: CUBE – Özgülleştirilmiş şifremi metin testi, ℓ=11 için CDR

Tablo XXXIV buradaki şifreleme tekrarları sırasında elde edilen korelâsyon analizi sonuçlarını vermektedir.

Tablo XXXIV: CUBE – Özgülleştirilmiş şifreli metin confusion testi – Korelâsyon

ℓ	Asymmetric Ciphertext AC= 1	Specific Ciphertext AC=0 SC=0,…,255
3	-0,0561250615742265	-0,000823989602691925
5	0,013526445185905	-0,00079113106756002
7	-0,000797508017674197	-0,000797508017674197
9	0,00690193230005878	-0,000102780694103529
11	-0,0153360441121235	0,000927319545366361
13	0,00689959771838286	-0,00032654253450797
15	-0,00570877685423579	-0,000232486491057775
17	0,00750668376718097	-0,000245465636874993
19	-0,0040915109020048	-0,002717303597648235
21	-0,000286512955791687	-0,000382661036815918

Burada 0'a en yakın korelâsyon değeri -0.000102780694103529 ℓ = 9 için elde edilmiştir, bu değerin şifreleme tekrarları sırasında elde ettiği korelâsyon dağılım grafiği Şekil 47 ile verilmiştir.



Şekil 47: CUBE - Düz metin - Şifreli metin korelâsyonları, ℓ=9 için

Bu analiz sonuçlarında UACB ve CDR Cube'ün özgülleştirilmiş şifreli metin çıkışı olan *SC* değeri girişlerinde yüksek hassasiyet ile confusion ve diffusion sağladığını göstermektedir. Ayrıca korelâsyon analizinden elde edilen değerler *SC* girişindeki değişikliklerin düz metin ve şifreli metin arasında anlamlandırılamaz değerler verdiğini göstermiştir, bu sonuçlar *SC* saldırılarında korelâsyon analizine karşı yüksek direnç ve güvenlik sağlandığını göstermektedir.

7.5.1.7. Çakışma

Burada çakışma saldırılarına karşı direnç test edilmiştir. Düşük çakışma değeri istenilen sonuçtur ve bu kriptografik fonksiyonlarının yerine getirmesi gereken önemli bir güvenlik kriteridir. Literatür kriptografik fonksiyonların çakışma analizlerinde 8-bitlik byte değerlerini kullanmaktadır (10, 54-56).

Bu çalışmada çakışma testi için Cube'ün $\mathcal{K}$ anahtar, $\mathcal{P}$ düzmetin ve SC özgülleştirilmiş şifremi metin parametreleri sırasıyla analiz edilmiştir. Her seferinde bir bitlik değişiklikler yapılan aşağıdaki testler gerçekleştirilmiştir.

Düz metin ve anahtar için aşağıdaki adımlar izlenmiştir:

1. π sayısının ilk $48\ell^3 - (8RF)$ bit değeri düzmetin ve e sayısının ilk $48\ell^3$ bit değeri anahtar olarak seçilmiştir.
2. Orijinal düz metin ve anahtar ile şifreleme yapılmıştır.
3. Anahtar testi için $\mathcal{K}$, düz metin testi için $\mathcal{P}$ olmak üzere orijinal parametrede rastlantısal bir bit değeri değiştirilmiştir ve şifreleme yapılmıştır.
4. Elde edilen iki şifreli metin tekrarlayan byte değerlerinin tespiti için karşılaştırılmıştır.
5. 3-5 arası adımlar 255 defa tekrarlanmıştır.

Özgülleştirilmiş şifreli metin testi için aşağıdaki adımlar izlenmiştir:

1. π sayısının ilk $48\ell^3 - (8RF)$ bit değeri düzmetin ve e sayısının ilk $48\ell^3$ bit değeri anahtar, AC = 0 ve SC=0 olarak seçilmiştir.
2. Şifreleme yapılmıştır
3. SC değeri bir arttırılarak şifreleme yapılmıştır.
4. 3. Adım 255 defa tekrarlanmıştır

ℓ = 3,5,…, 21 aralığındaki bütün Cube katman değerleri için 255 şifreleme sonucu elde edilen collision test sonuçları Tablo XXXV ile verilmiştir.

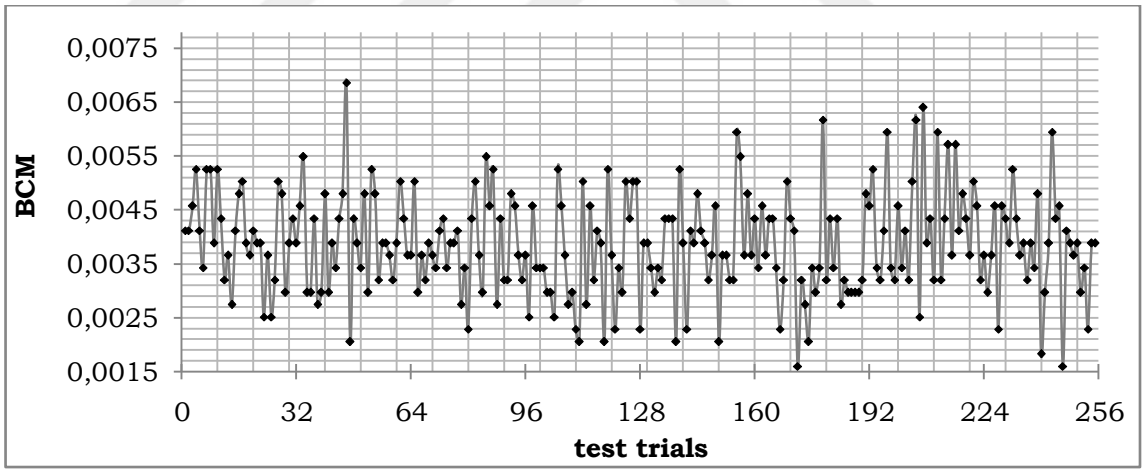
Tablo XXXV. Çakışma Testi - BCM

ℓ	Test Parameter		
	$\mathcal{K}$ **Değişimleri**	$\mathcal{P}$ **Değişimleri**	**SC Değişimleri**
3	0, 003861316872428	0, 003895037521181	0, 003945775841200
5	0, 003916339869281	0, 003843137254901	0, 003937254901960
7	0, 003875836048705	0, 003811048228815	0, 004001600640256

9	0, 003859705747868	0, 003900051104117	0, 003921568627450
11	0, 003873936251184	0, 003903399576710	0, 003882284193416
13	0, 003880811921187	0, 003905503940310	0, 003912346307055
15	0, 003886516581941	0, 003954103122730	0, 003930670539820
17	0, 003867157827239	0, 003906535814678	0, 003906535814678
19	0, 003889837025348	0, 003865633340861	0, 003899556615181
21	0, 003906818459306	0, 003901454761798	0,003904912935191

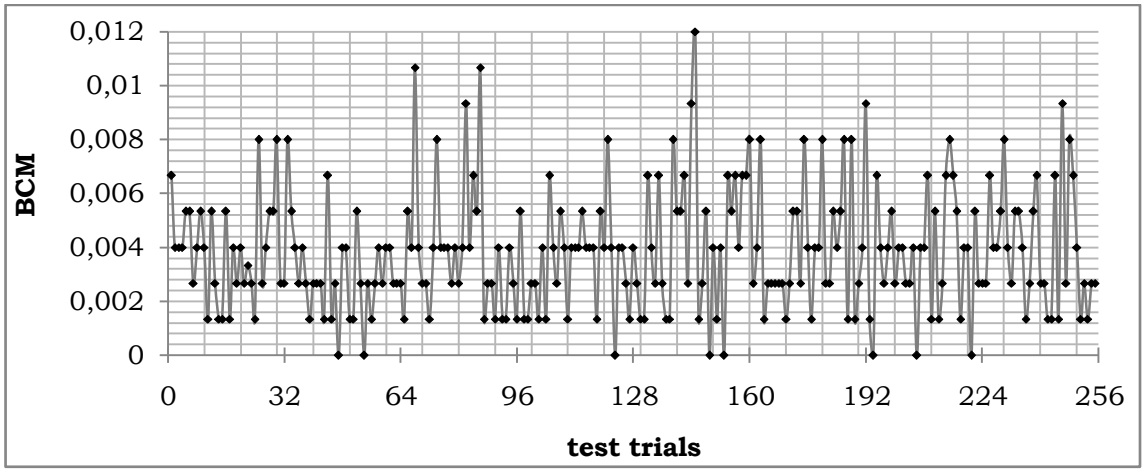
Tablo XXXV anahtar, düz metin ve özgülleştirilmiş şifreli metin parametrelerinde yapılan değişimler sonucu oluşan çakışma değerlerini vermektedir.

Tablo XXXV'e göre $\mathcal{K}$ için ile en düşük çakışma değerini 0.003859705747868 ile ℓ=9 katmanı elde etmiştir, bu katmanın şifreleme tekrarlarında oluşturduğu çakışma dağılım grafiği Şekil 48 ile verilmiştir.



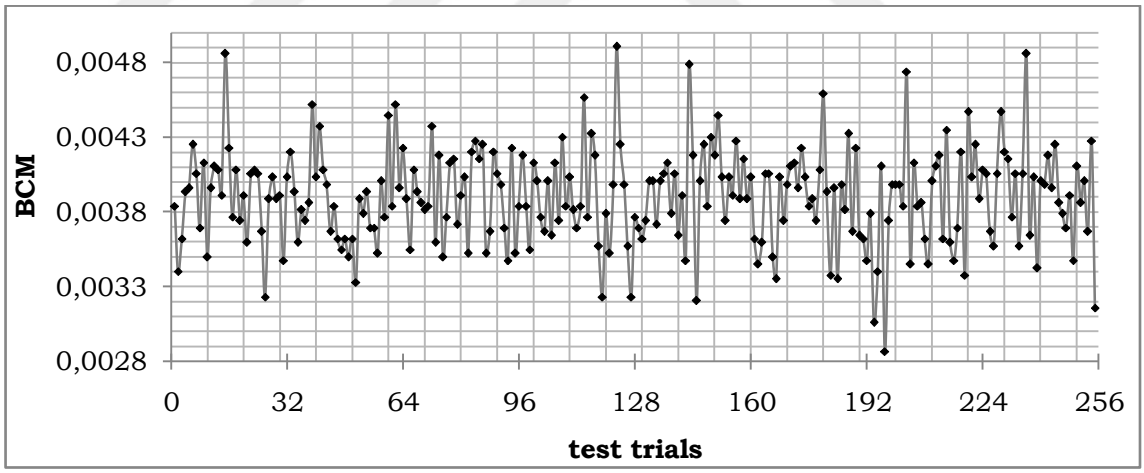
Şekil 48: CUBE - Çakışma Dağılım Grafiği, $\mathcal{K}$ düzenlemesi ve ℓ=9

Tablo XXXV'e göre $\mathcal{P}$ için ile en düşük çakışma değerini 0,003843137254901 ile ℓ=5 katmanı elde etmiştir, bu katmanın şifreleme tekrarlarında oluşturduğu çakışma dağılım grafiği Şekil 49 ile verilmiştir.



Şekil 49: CUBE - Çakışma Dağılım Grafiği, $\mathcal{P}$ düzenlemesi ve ℓ=5

Tablo XXXV'e göre SC için ile en düşük çakışma değerini 0,003899556615181 ile ℓ=19 katmanı elde etmiştir, bu katmanın şifreleme tekrarlarında oluşturduğu çakışma dağılım grafiği Şekil 50 ile verilmiştir.



Şekil 50: CUBE - Çakışma Dağılım Grafiği, SC düzenlemesi ve ℓ=19

Bu analiz sonuçlarında elde edilen BCM değerleri, Cube'ün düz metin, anahtar ve *SC* girişlerinde yüksek hassasiyet ile diffusion sağladığını göstermektedir. Bu sonuçlar Cube'e karşı gerçekleştirilecek analiz saldırılarına yüksek direnç ve güvenlik sağlandığını göstermektedir.

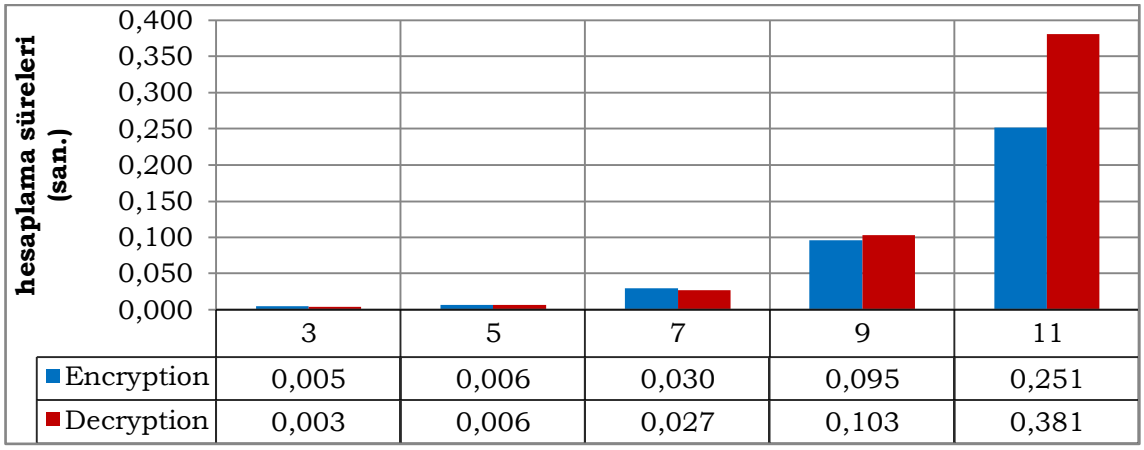
7.5.2. Performans

Cube yönteminin performans testi $48\ell^3 - (8.RF)$ bit uzunlukta düz metnin şifrelenmesi ve şifresinin çözülmesi ile gerçekleştirilmiştir. Bu rada $48\ell^3$ bit uzunlukta anahtar ve $\ell = 3, 5, 7, 9, 11, 13, 15, 17, 19, 21$ katman değerleri ve asimetrik şifreli metin çıkışını temsil eden AC=1 değeri kullanılmıştır. Her bir ℓ değeri ile tekrarlanan 100 şifreleme ve şifre çözme işlemi sonucu elde edilen aritmetik ortalamalardan elde edilen sonuçlar Tablo XXXVI ile görülmektedir.

Tablo XXXVI: CUBE - Performans Analizi

ℓ	Blok Boyu (bit)	Düz Metin Boyu (bit)	Anahtar Boyu (bit)	Şifreleme Zamanı (sn.)	Şifreleme MByte/s	Şifre Çözme Zamanı (sn.)	Şifre Çözme MByte/s
3	1.296	1.280	1.296	0,005	2,654	0,003	2,710
5	6.000	5.976	6.000	0,006	1,862	0,006	2,096
7	16.464	16.440	16.464	0,030	2,988	0,027	3,899
9	34.992	34.968	34.992	0,095	3,083	0,103	3,350
11	63.888	63.864	63.888	0,251	4,790	0,381	3,540
13	105.456	105.432	105.456	0,569	5,210	0,672	4,593
15	162.000	161.976	162.000	1,309	7,038	1,555	5,465
17	235.824	235.800	235.824	2,086	6,485	2,917	7,893
19	329.232	329.208	329.232	4,005	9,373	4,551	11,054
21	444.528	444.504	444.528	7,349	11,788	10,708	13,874

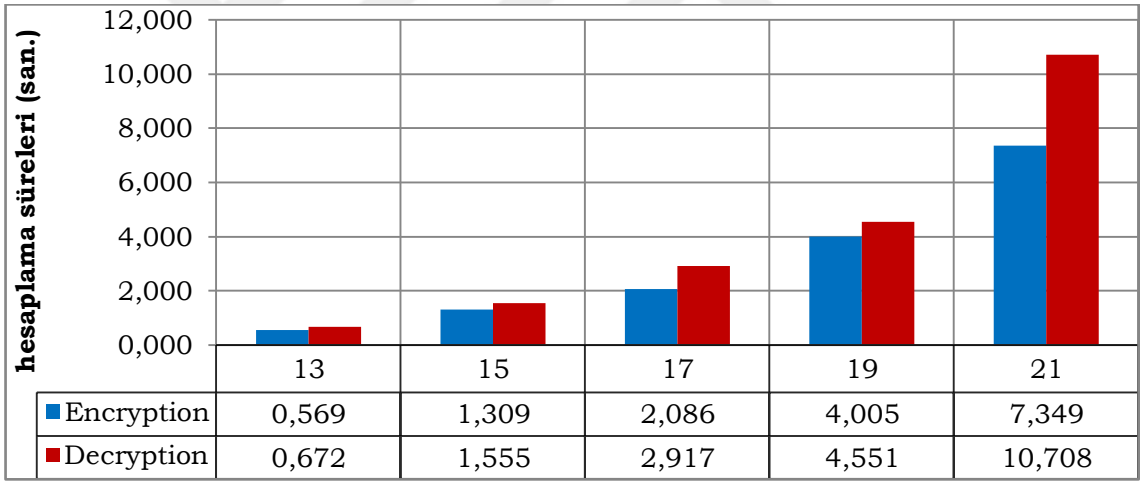
3, 5, 7, 9 ve 11 katman değerleri için hesaplama zamanı performansı değerleri Şekil 51 ile görülmektedir.



	3	5	7	9	11
Encryption	0,005	0,006	0,030	0,095	0,251
Decryption	0,003	0,006	0,027	0,103	0,381

Şekil 51: CUBE - Performans, Süreleri, 3, ...,11 arası

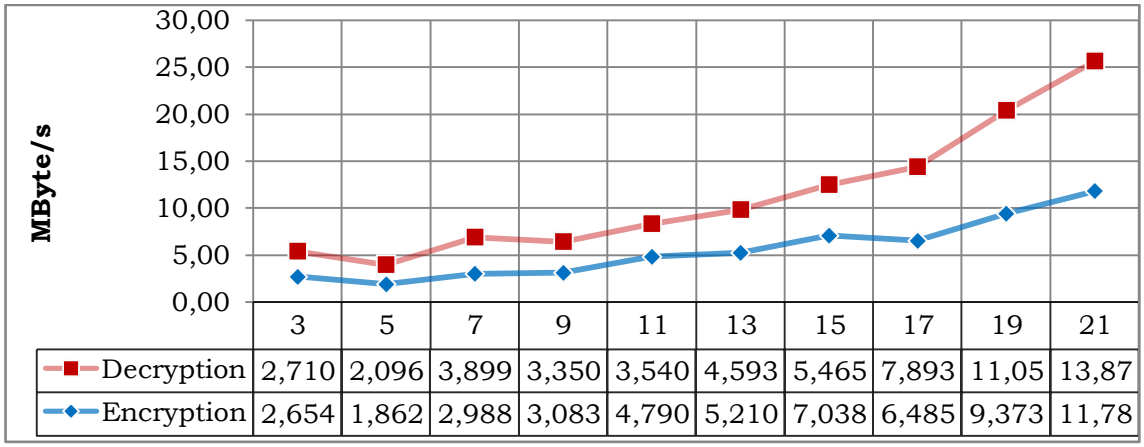
13, 15, 17, 19 ve 21 katman değerleri için hesaplama zamanı performansı değerleri Şekil 52 ile görülmektedir



	13	15	17	19	21
Encryption	0,569	1,309	2,086	4,005	7,349
Decryption	0,672	1,555	2,917	4,551	10,708

Şekil 52: CUBE - Performans, Süreleri, 13, ...,21 arası

3, …, 21 katman değerleri için MByte/s performans değerleri Şekil 53 ile görülmektedir



	3	5	7	9	11	13	15	17	19	21
Decryption	2,710	2,096	3,899	3,350	3,540	4,593	5,465	7,893	11,05	13,87
Encryption	2,654	1,862	2,988	3,083	4,790	5,210	7,038	6,485	9,373	11,78

Şekil 53: CUBE - Performans, MByte/s, 3, ...,21 arası

ℓ katman değerinin 11'i geçmesi durumunda hesaplama sürelerinde anlamlı artışlar gösterdiği, ancak bu artışın hesaplanan veri boyu ile paralel bir artış gösterdiği gözlemlenmiştir. Bu blok boyunun ($6\ell^3$) kübik olarak büyümesi ve şifreleme işleminde gerçekleştirilen ℓ^2 tur sayısının bu değere göre artış göstermesi sonucu gelişmektedir. Bu analizlerde Cube için en iyi şifreleme performans değeri ℓ=21 değeriyle elde edilmiştir, ancak burada $6\ell^3 - rf$ düz metin ve $6\ell^3$ anahtar kullanılmıştır. Cube için seçilen düz metin boyu büyüklüğünün, blok boyu olan $6\ell^3 - rf$ değerinden büyük olması performans değerlerinin azalmasıyla sonuçlanır, Bu sebeple şifrelenecek düz metin ile blok boyu arasında oranlı değerlerin kullanılması gerekir. Burada iletişim sistemleri üzerinde veri iletimi için ℓ = 3, ℓ = 5 ve ℓ = 7 katman değerlerinin, yüksek seviyeli ve hassasiyetli veri için ℓ > 7 katman değerlerinin uygun performans değerlerini vereceği görülmüştür.

8. TARTIŞMA

Bu çalışma *sayısal veri ve deliller için küp tabanlı çok boyutlu asimetrik şifreleme yöntemi* tez konusu altında iki ayrı başlıkta kriptografik yapının tasarımını gerektirmiştir. Hem adli bilimler alanına hem bilgi güvenliği alanlarına dâhil bir çalışmanın yapılmasında, adli bilimler alanına ya da bilgi güvenliği alanına yönelim yerine iki alan sentezinde araştırma sürecinin geliştirilmesi, çalışma içeriğinde geniş bir incelemenin de yer almasını zorunlu kılmıştır.

Çalışma kapsamında tasarlanan Cube ile adli bilimler alanında bölüm 1.4.3. Şüpheli / Sanık Tarafında Delillere Erişim Hakkı ile verilen sayısal delile şüpheli/sanığın erişiminin hukuki zorunsallığı konusunda başvurulabilecek bir kriptosistem elde edilmesi amaçlanmıştır. Elde edilecek yeni yöntemin veri özetlemesini ve imzalaması için, ayrıca bir mesaj özetleme algoritması kullanılmalıyıydı. Bu gerekçe ile Cube kriptosisteminin tasarımı öncesinde literatürde yer alan mesaj özetleme algoritmaları incelendiğinde:

1. En çok bilinen ve güvenilir kabul edilen mesaj özetleme algoritmalarının, uluslararası bir kurum niteliği olmayan NIST tarafından değerlendirildiği ve geliştirildiği görülmüştür.
2. En çok bilinenlerin hepsinde Merkle-Damgard modelini kullandıkları görülmüştür (19-21, 46, 48, 57-58).
3. En çok bilinen ve güvenilir kabul edilen mesaj özetleme fonksiyonlarının önemli bir bölümü için güvenlik zafiyetleri bulunmuş ve birçoğu için doğrudan diferansiyel saldırı yöntemlerinin literatüre girdiği görülmüştür (28-30, 43-44, 59-62).

4. Çakışma elde edilebilecek yöntemlerin tespit edildiği MD5 ve SHA ailesinin bugün birçok kurum ve adli bilişim uzmanınca delil bütünlüğü ispatında kullanıldığı görülmüştür.

Bu tespitler devamında Cube için yeni bir mesaj özetleme algoritması tasarlanması gerekmiş ve bu çalışma da ICDS olarak verilmiştir. ICDS diğer mesaj özetleme fonksiyonlarını değerlendirip yayınlayan NIST'in analiz araçlarıyla ve ek test yöntemleriyle analiz edildiğinde, referans alınan SHA mesaj özetleme algoritmasından daha başarılı sonuçlar elde etmiştir.

ICDS için bilinen Merkle-Damgard modeli veya HAIFA modeli kullanılmamıştır. Sabit bir mesaj özet değeri uzunluğu yerine değişken aralıkta uzunluklar kullanılmasına imkân veren bir algoritma tasarımı geliştirilmiştir. Bilinen mesaj dolgulama ve iteratif fonksiyonlar kullanılmamıştır. Alışılmış ve en çok bilinen mesaj özetleme fonksiyonlarından farklı yapısı, araştırmacı ve uygulamacılar tarafında çözümleme-uyarlama-anlaşılma aşamasında karışıklığa yol açabileceğinden, bu çalışma içerisinde detaylı ve bilinebilecek bilgilerin tekrar vurgulanması gerekmiştir.

ICDS analiz ve uygulama testleri çalışmada yer alan algoritmalar ile C# ve ANSI C programlama dillerinde kodlanarak gerçeklenmiştir. Uygulama için bu programlama kodları çalışma içerisinde verilmemiş ancak algoritmaların yazılmasında farklı programlama dilleri için uyarlanabilir temsiller kullanılmıştır. Araştırmacı ve uygulayıcıların verilen algoritmalarla, ICDS modeline uygun farklı betikler geliştirmesi ICDS için ek çalışma konuları olacaktır. ICDS için:

1. Bu çalışmada yer alan detaylı anlatımın sadeleştirilmiş hali için çalışma gerçekleştirilmeli ve açık olarak yayınlanmalıdır.

2. Farklı platformlarda uygulanabilecek uygulamalar / yazılımlar üretilmeli ve açık kaynaklardan erişime açılmalıdır.
3. Çalışmaya dair gereken kaynakların standart bir platform üzerinden yayınlanması ve araştırmacılara bilinen bir kaynak sunulmalıdır.

Bu gereksinimlerin sağlanması ve çalışma üzerinde standartların duyurulabileceği bir otoritenin oluşturulması gerekliliği sebepleri şöyledir:

1. Algoritmalardaki en küçük değişkenin farklılaştırılması ICDS'in çok farklı sonuçlar üretmesine yol açar ve bu sonuçların güvenliği-güvenirliği öngörülemez olur.
2. Araştırmacı ve uygulayıcıların ICDS'e ile ilgili çalışmaları takip edebilecekleri ve gerçekleştirebilecekleri tek kaynağın olması gereklidir.
3. Alana hakim olmayanlar için ICDS algoritmasıyla geliştirilen bilgisayar yazılımlarının kullanılabilir şekilde paylaşımı gereklidir.

Bu çalışma konusu olan CUBE kriptosistemi, ICDS tasarım ve analizleri sonrasında geliştirilmiştir. Tasarımda Rubik küpünden esinlenilmiştir ve geometrik bir konu olan *dönme merkezine* dayalı bir model oluşturulmuştur. Turlu hesaplama ve geometrik dönme merkezi şifreleme / şifre çözme için geri besleme olarak kullanılmıştır. Bu sayede kriptosistemler literatürüne yeni bir asimetrik alan olabilecek asimetrik şifreli veri elde edilmiştir.

Hızla gelişmekte olan teknolojik imkânlar sebebiyle, Cube blok ve anahtar boyutları büyük değerlerde tasarlanarak çalışmada verilmiştir. Literatür anahtar boyunu güvenlik için önemli bir parametre olarak kabul etmektedir. Çalışmanın yapıldığı tarihte

en çok bilinen ve en güvenilir kabul edilen blok kriptosistemi AES (49) için *en büyük anahtar boyu 256-bit*tir. Cube için *en küçük anahtar boyu 1296-bit* ve *en büyük anahtar boyu 444528-bit*tir.

Cube ile veri yerleşimi 8-bitlik byte değerleri ile gerçekleştirilmiştir. Bu büyük blok ve anahtar değerleri elde edilmesini sağlamıştır. Çok daha küçük blok ve anahtar değerleri kullanılması bu çalışma ile ele alınmamıştır ancak çalışma içerisinde yer alan model izlenerek, cubie yüzlerine 8-bit yerine 1-bit yerleştirilmesi mümkündür ve araştırmacılar için ek bir konu sağlamaktadır. Burada Cube üzerinde araştırma yapacak araştırmacıların merkez cubiede rastlantısal sayı yerleşimi ve düz metin boyunun yerleşimi için ayrılan bölümlerine dikkat etmeleri gerekir. Merkez cubieye ait yüzlerin 8-bit olarak bırakılması uyarlamaları için önerilir. Rastlantısal sayının 8-bitten küçük olması 256 özgül şifrelenmiş metninde küçülmesi sonucunu ortaya çıkaracağı ve $\ell^3 - rf$ olarak verilen blok değerininde merkez cubiede iki face ile temsil edilebilmesi gerekir. Ayrıca mevcut bilgisayar mimarisi işlemci ve hafıza kayıt alanlarında 8-bitlik dizilimler kullandığından oluşturulacak küp bloğundaki toplam bit sayısının sekizin katları olacak şekilde tasarlanması olası sorunların önüne geçilmesi için önemlidir.

Cube kriptosisteminin asimetrik yapısı şifreli metinden kaynaklanmaktadır, ancak şifreli metin akış diyagramının üzerinde yapılacak çalışmalarla bu modelin asimetrik anahtar sınıfı için kullanılabilirliğinin araştırılması alan için faydalı olabilecek bir diğer çalışma konusudur. Cube üzerinde bu amaçla yapılacak çalışmalarda kafes indirgeme problemlerinin ve NTRU kriptosistemlerinin incelenmesi faydalı olacaktır (18, 55).

Cube ve ICDS için güvenlik ve uygulama testlerinde en güvenilir kabul edilen kriptografik fonksiyonlar için uygulanan testler uygulanmıştır. Bunlar dışığında çalışma kapsamında farklı güvenlik ve performans testleri de gerçekleştirilmiştir. Aynı test

sonuçlarının uygulandığı en çok bilinen ve güvenilir kabul edilen algoritmaların testlerde elde edilen başarıları Cube ve ICDS'den az olmuştur.

Cube ve ICDS alt yapı sunan iki ayrı yöntemdir. Bu çalışmada kaynak takibini mümkün kılacak ve veriyle gerçekleştirilen mağduriyetlerin engellenmesinde kullanılabilecek Cube ve ICDS geliştirilmiştir, ancak bu iki yöntemle elde edilen alt yapıların kullanılacağı uygulama ve tasarımlar ayrıca geliştirilmelidir. Cube'ün özelliklerini kullanacak dosyalama sistemi ve format modelinin geliştirilmesi bu konuda yapılacak en önemli çalışmadır. Cube bu yöndeki çalışmalara gerekli alt yapıyı sağlamak için geliştirilmiştir, ancak bu alt yapının kullanılmadığı durumlarda Cube sadece gelişmiş bir kriptosistem algoritması olarak kullanılabilir.

Cube için geliştirilen ICDS mesaj özetleme algoritması mevcut durumuyla adli bilimler alanında sayısal veri delilinin bütünlüğünün ispatı ve steganografik yöntemlerde damga değerleri için gerekenleri karşılamaktadır.

9. SONUÇ

Bu çalışma kapsamında kriptografik yöntemler öncesinde gerçekleştirilen incelemeler ile adli bilimler ve kriptoloji arasındaki ilişki, kriptoloji ve kriptografi hakkında detaylı bilgiler sağlanmış, devamında adli bilimlerde ve bilgi güvenliği alanlarında kullanılabilecek kriptografik Cube kriptosistemi ve ICDS mesaj özetleme algoritmaları geliştirilmiştir.

Cube kriptosistemi asimetrik şifreli metin çıkışı yapabilen yapısıyla yeni bir kriptografi sınıfındadır. Cube'ün asimetrik özelliği adli bilimlerde dijital delillerin kaynak kontrolü ve ispatı için kullanılabilecek yöntemler için alt yapı sağlamıştır. Cube'ün özgülleştirilmiş şifreli metin çıkış özelliği, düz metin ve aynı anahtarla seçilebilir veya rastlantısal 256 ayrı şifreli metin çıkışını sağlar.

ICDS mesaj özetleme algoritması sayısal imza algoritmalarında, veri bütünlüğü ispatında, kaynak kontrolünde, veri işaretlemede, kimlik ispatında, şifre örtmede ve rastlantısal sayı üreteci (PRNG) için kullanılabilen, 128-65536 aralığında seçilebilir uzunlukta özet değerleri üretebilmekte ve donanımsal ve yazılımsal olarak kullanılabilir algoritmaya sahiptir.

Cube ve ICDS için gerçekleştirilen detaylı analizlerde her iki yöntem için yüksek güvenlik ve performans sonuçları elde edilmiştir. ICDS ile aynı kriptografik yöntem sınıfında olan, en çok bilinen ve güvenilir kabul edilen SHA referansı ile gerçekleştirilen analizlerin hepsinde ICDS daha başarılı sonuçlar elde etmiştir.

KAYNAKLAR

1. Diffie-hellman key agreement method. RFC2631 1999.

2. A method for obtaining digital signatures and public-key cryptosystems. Rives, R.L., Adlerman, L. 21, s.l. : Commun.ACM, 1978, s. 120-126.

3. Centel, N. ve Zafer, H. Ceza Muhakemesi Hukuku. İstanbul : s.n., 2010.

4. Ceza Muhakemesinde İspat, CMK ve Uygulamamız. Ünver, Y. 5, Ceza Hukuku Dergisi, Cilt 2, s. 5.

5. C.G.K. 18.11.2003/262-277. 262-277, s.l. : Ceza Genel Kurulu, 18 11 2003.

6. C.G.K. 28.09.2010/109-177. 109-177, s.l. : Ceza Genel Kurulu, 28 09 2010.

7. Özekes, Serhat. Veri Gizleme Yöntemleri. Veri Gizleme Yöntemleri. İstanbul, Üsküdar, Türkiye : Üsküdar Üniversitesi, 2018.

8. Özmen, A, et al. İstatistik. 1. Baskı. Eskişehir : Anadolu Üniversitesi Basımevi, 2012. Anadolu Üniversitesi Yayını No: 2590.

9. Özekes, Serhat. Sayısal Delil İle İlgili Temel Kavramlar. Sayısal Delil İle İlgili Temel Kavramlar. [Ders Sunumu]. İstanbul, Türkiye : Üsküdar Üniversitesi, Bağımlılık ve Adli Blimler Enstitüsu, 2018.

10. Koblitz, N. A Course in Number Theory and Cryptography. 2nd ed. New York : Springer-Verlag, 1994. 0-387-94293-9.

11. İşpınar. [Çevrimiçi] 2004. http://www.bilmuh.gyte.edu.tr/~ispinar/BM550/SECURITYCRS9-13.pdf.

12. Stalling, W. Cryptography and Network Security Principles and Practices. 4. Sürüm. s.l. : Prentice Hall, 2005.

13. 35.Annual Symposium on Foundations of Computer Science. Shor, P.S. Los Alamitos : IEEE Press, 1995. Algorithms for Quantum Computation: Discrete Logarithms and Factoring.

14. Standard Specifications For Public−Key Cryptography. s.l. : IEEE, Ekim 1998. P1398.

15. Key Length. [Çevrimiçi] [Alıntı Tarihi: 2018 Nisan 24.] http://www.keylength.com.

16. Piper, J., Silverman, J.H. ve Hoffstein, J. NTRU: A ring-based public key cryptosystem. Lecture Notes in Computer Science. 1998, 1423, s. 267-288.

17. Tekin, Sevcan. Ntru. İstanbul : Yıldız Teknik Üniversitesi, Fen bilimleri Enstitüsü, 2011. Matematik Anabilim Dalı Yüksek Lisans Tezi.

18. Silverman, J.H.,. Invertibility in Truncated Polynomial Rings. NTRU Cryptosystems Technical Report 9. [Çevrimiçi] [Alıntı Tarihi: 12 Aralık 2018.] http://www.securityinnovation.com/cryptolab/tech_notes.shtml.

19. Kaliski, B. The MD2 Message-Digest Algorithm. RFC. 1319, 1992.

20. R., Rivest. The MD5 Message-Digest Algorithm. RFC1321 1992. Request for Comments.

21. Rivest R. I., Agre B., Bailey D.V., Crutchfield C., Dodis Y., Ellioti K., Khan F.A., Krisnamurthy J., Lin Y., Reyzin L., Shen E., Sukha J., Sutherland D., Tromer E., Yin Y.L. The MD6 hash function. NIST's SHA-3 Competition. 2008.

22. NIST. SHA-3 standardization Tech. rep. National Institute of Standards and Technology. [Çevrimiçi] 2013-2015. [Alıntı Tarihi: 25 09 2018.] http://csrc.nist.gov/groups/ST/hash/sha-3/sha-3_standardization.html.

23. HAVAL - a one-way hashing algorithm with variable length of output. ZhengY., Pieprzyk J., Seberry J. 718, Heidelberg : Springer, 1993, Advances in Cryptology, s. 83-104.

24. Baretto P., Rijmen V. The WHIRLPOOL hashing function. [Çevrimiçi] 2003. [Alıntı Tarihi: 25 09 2018.] http://www.larc.usp.br/pbaretto/whirlpool.zip.

25. ECOH: the Elliptic Curve Only Hash. D. R. L. Brown, A. Antipa, M. Campagna, and R. Struik. s.l. : Submission to NIST, 2008.

26. Halevi S., Hall W., Jutla C. The hash function "Fugue". NIST's SHA-3 Competition. 2009.

27. Varıcı, Kerem. Sarmal: A Cryptographic Hash Function. The Graduate School of Applied Mathematics of Middle Fast Technical University. Ağustos 2008. Thesis for the Degree of Master of Science.

28. Wang X., Feng D., Lai X. and Yu H. Collisions for Hash Functions MD4, MD5,. [Çevrimiçi] 2004. http://eprint.iacr.org/2004/199.pdf.

29. V., Klima. MD5 Collisions Within a Minute, International Association for Cryptologic Research. [Çevrimiçi] 2006. http://eprint.iacr.org/2006/105.

30. Daum, Magnus. Cryptanalysis of Hash Functions of the MD4-Family. Ruhr Universitat. 2005. PhD thesis.

31. Biham, Eli ve Orr, Dunkelman. A Framework for Iterative Hash Functions - HAIFA. Cryptology ePrint Archive. [Çevrimiçi] 24.07.2018 2007. http://eprint.iacr.org. Report 2007/278.

32. Generating strong one-way functions with. S.M. Matyas, C.H. Meyer, J. Oseas. 27(10A):5658-5659, s.l. : IBM Technical Disclosure Bulletin, 1985.

33. Digital signatures, an update. D. Davies, W.L. Price. s.l. : In 5th International Conference, 1994.

34. Hash Function Based on Block Ciphers. Xuejia Lai, James L. Massey. s.l. : In EUROCRYPT, 1992.

35. Collision-free Hash Functions Based on Block Cipher Algorithms. Vandewalle, Bart Preneel and Antoon Bosselaers and Ren´e Govaertsi Joos. s.l. : In International Carnahan Conference on Security Technology, 1989.

36. Security of Iterated Hash Functions Based on Block Ciphers. Walter Hohl, Xuejia Lai, Thomas Meier, Christian Waldvoge. s.l. : In Stinson.

37. Sponge Functions. Bertoni, Guido, Joan, Daemen ve Micha¨el Peeters, Gill. s.l. : ECRYPT Hash Worksop, 2007.

38. Fast Hashing and Stream Encryption with PANAMA. Clapp, Joan Daemeni Craig S. K. 1998. : Springer, Lecture Notes in Computer Science, s. 60–74.

39. Producing Collisions for Panama. Vincent Rijmen, Bart Van Rompay, Bart Preneel, Joos Vandewalle. 2355, s.l. : Springer,, 2001, Lecture Notes inComputer Science, s. 37–51.

40. Radio-Gat´un A Belt and Mill Hash Function. Guido Bertoni, Joan Daemen , Gilles VanAssche , Micha¨el Peeters. s.l. : Springer, 2007.

41. RC4-Hash: A New Hash Function Based on RC4. Donghoon Chang, Kishan Chand Gupta, Mridul Nandi. s.l. : INDOCRYPT,, 2006.

42. Sebastiaan Indesteege, Bart Preneel. Collisions for RC4-Hash. ISC 2008.

43. Herding Hash Functions and the Nostradamus Attack. Kelsey, John ve Kohno, Tadayoshi. 4004, Heidelberg : Springer, 2006, Advances in Cryptology - EUROCRYPT ’06, s. 183-200. Lecture Notes in Computer Science.

44. H., Wang, et al. Advanced Meet-in-the-Middle Preimage Attacks: First Results on Full Tiger, and Improved Results on MD4 and SHA2, International Association for Cryptologic Research. [Çevrimiçi] 2010. http://eprint.iacr.org/2010/016.pdf.

45. Di. Florent Chabaud, Antoine Joux. London : Springer, 1998., In CRYPTO '98:Proceedings of the 18th Annual International Cryptology Conference on Advances in Cryptology.

46. The MD4 Message Digest Algorithm. Rivest, R. L. 537, Heidelberg : Springer, 1990, CRYPTO, s. 303–311.

47. Stinson, D. R. Theory and Practice. New York : CRC Press, 1995. 0-8493-8521.

48. Technology, National Institute of Standards and. Secure Hash Standard. FIPS PUB 180-1 USA, 1995.

49. The Design of Rijndael AES-The Advanced. Rijmen, V. ve Daemen, J. Germany : Springer, 2002, s. 10-17, 31-50.

50. Differential Cryptanalysis of the Data Encryption Standard. Advances in Cryptology. Biham, E., Shamir, A. Verlag : Springer, 1990, CRYPTO'90, s. 2-21.

51. Collision free hash functions and pubic key signature schemes. Damgård, I. B. 1988, Advances in, s. 203–216. LNCS 304.

52. Cryptographic hash-function basics: definitions, implications, and separations for preimage resistance, second-preimage resistance, and collision resistance. Rogayway P., Shrimpton T. 3017, Heidelberg : Springer, 2004, s. 371-388.

53. A novel chaotic image encryption algorithm based on content-sensitive dynamic function switching scheme. Yavuz, E. s.l. : Optics and Laser Technology, 2019.

54. The data Encryption Standard (DES) and strength against attacks. Coppersmith, D. 38, s.l. : IBM Journal of Research and Development, 1994, s. 243-250.

55. Introduction to Cryptography with Coding Theory. Trappe, W. s.l. : Prentice Hall, 2002.

56. NPCR and UACI randomness tests for image encryption, cyber journals: multidisciplinary journals in science and technology. Wu, Y., Noonan, J.P., Agaian, S. 1, s.l. : J. Select. Areas Telecommun, 2011.

57. RIPEMD-160: a strengthened version of RIPEMD. Dobbertin H., Bosselsers A., Preneel B. 1039, Heidelberg : Springer, 1996, Fast Software Encryption '96, s. 71-82. Lecture Notes in Computer Science.

58. H., Wu. The Hash Function JH. NIST's SHA-3 Competition. 2011.

59. M., Stevens. Fast Collision Attack on MD5. [Çevrimiçi] 2006. http://eprint.iacr.org/2006/104.pdf.

60. Manuel, S. Classification and Generation of Disturbance Vectors for Collision Attacks. International Association for Cryptologic Research. [Çevrimiçi] 2008. http://eprint.iacr.org/2008/469.pdf.

61. Finding collisions in the full SHA-1. Wang X., Yin Y., Yu H. 3621, Heidelberg : Springer, 2005, Advances in Cryptology- CRYPTO, s. 17-36.

62. Xiaoyun, Wang ve Hongbo, Yu. How to Break MD5 and Other Hash Functions. EUROCRYPT. 2005, s. 19-35.

63. Sahinoglu, M. AES Algoritmasinin FPGA Üzerinde Gerceklemesine Elektromanyetik Alan Saldirisi. Istanbul : Istanbul Technical University - Institute of Science and Technology, 2008.

64. On the power of memory in the design of collision resistant hash functions. Preneel B., Govaerts R., Vandewalle J. 718, Heidelberg : Springer, 1992, s. 105-121.

65. Constructing cryptographic hash functions from fixed-key blockciphers. P., Rogaway. 5157, Heidelberg : Springer, 2008, Advances in Cryptology-CRYPTO, s. 443-450.

66. NIST. The Keyed-Hash Message Authentication Code (HMAC). FIPS 198-1 July 2008.

67. DRAFT Randomized Hashing Digital Signatures. SP 800-106 July 2008.

68. Digital Signature Standard (+change notice). s.l. : National Institute for Standards and Technology, 2000. FIPS 186-2.

69. Digital Signature Standard. s.l. : National Institute for Standards and Technology, 1994. FIPS 186,.

70. Advanced Encryption Standard. FIPS 197 November 2001.

71. On the XOR of multiple random permutations. Mennink B., Prenell B. Heidelberg : Springer, 2015, Applied Cryptography and Network Security - ACNS.

72. L'Ecuyer, P., Simard, R.J. TestU01: A C library for empirical testing of random number generators. ACM Trans. Math. Softw. [Çevrimiçi] 2007. [Alıntı Tarihi: 2018 09 25.] http://doi.acm.org/10.1145/1268776.126877 .

73. Strengthening digital signatures via randomized hashing. Halevi S., Krawczyk H. 4117, Heidelberg : Springer, 2006, Advances in Cryptology-CRYPTO, s. 41-59. Lecture Notes in Computer Science.

74. A design principle for hash functions. Damgard, I. 435, Heidelberg : Springer, 1990, Advances in Cryptology-CRYPTO, s. 416-427. Lecture Notes in Computer Science.

75. One way hash functions and DES. 1990, Advances in Cryptology-Crypto'89, s. 428–446. LNCS 435.

76. An image encryption scheme using self-adaptive selective permutation and inter-intra-block feedback diffusion. Liu, D., Zhang, W., Yu, H., Zhu, Z.L. 151, s.l. : Signal Proccess, 2018.

77.Chaotic keyed hash function based on feedforward-feedback nonlinear digital filter. Zhang, J., Wang, X., Zhang, W. A 362, s.l. : Phsy.Lett, 2007.

ÖZGEÇMİŞ

Burak BAYSAN, 1987 yılında Eskişehir'de doğdu. 2012 yılında Eskişehir Anadolu Üniversitesi İktisat Fakültesinde başladığı lisans öğrenimini 2016 yılında yüksek onur derecesiyle tamamlamıştır. 2017 yılında Üsküdar Üniversitesi Bağımlılık ve Adlim Bilimler Enstitüsünde yüksek lisans öğrenimine başlamıştır. Adli bilişim, siber güvenlik ve kriptoloji alanlarına dâhil olan çalışmalarla ilgilenmektedir.

e-posta: burak.baysan@st.uskudar.edu.tr