ANKARA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

DOKTORA TEZİ

UÇTAN UCA YÜKSEK YÜKLÜ VE GÜVENLİ VERİ İLETİMİ İÇİN İSTATİSTİKİ ANALİZLERE DAYANIKLI İMAJ TABANLI STEGANOGRAFİK YÖNTEMLER

Timuçin KÖROĞLU

BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

ANKARA
2024

ÖZET

Doktora Tezi

UÇTAN UCA YÜKSEK YÜKLÜ VE GÜVENLİ VERİ İLETİMİ İÇİN İSTATİSTİKİ ANALİZLERE DAYANIKLI İMAJ TABANLI STEGANOGRAFİK YÖNTEMLER

Timuçin KÖROĞLU

Ankara Üniversitesi
Fen Bilimleri Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı

Danışman: Prof. Dr. Refik SAMET

Çalışmada, steganografik yöntemlerle veri iletiminde algınamamazlık-kapasite ve güvenlik-kapasite arasındaki ödünleşim problemini ortadan kaldırabilmek amacıyla iki yönlü hash fonksiyonu (İYHF) önerilmiştir. Tek yönlü hash fonksiyonları kullanarak orijinal veriyi elde etmek hesaplama açısından imkansızdır. Önerilen metot, keyfi uzunluktaki giriş verilerini hashlemeyi/şifrelemeyi ve şifresini çözmeyi amaçlamaktadır. Bu işlevler, verinin, boyutundan bağımsız olarak şifrelenmesine ve şifresinin çözülmesine olanak tanır. Bu özellik İYHF'nin tek yönlü hash fonksiyonlarının uygulandığı alanların yanı sıra steganografi gibi uygulanamadığı alanlarda da kullanılmasına imkan tanır. İYHF ile özeti alınan veri, örtü nesnesi içinde alıcı tarafına gönderildiğinde tersinir özelliğe sahip olduğu için alıcı orijinal veriyi yeniden elde etme imkanına sahip olabilir. Böylece hash fonksiyonlarının kendilerine özgü güvenlik özellikleri ile steganografinin gizlilik, direnç ve algılanamazlık özellikleri biraraya gelerek çok daha güvenli bir veri iletimi gerçekleştirilmiş olur. Önerilen yöntem, gizliliği sağlamak için yineleme anahtarını ve ikincil anahtarı kullanır. Bu çalışmada, İYHF'nin tek yönlü hash fonksiyonlarının güvenlik özelliklerini yerine getirdiğine dair güçlü ipuçları elde edilmiştir. İYHF, orijinal verileri üretilen bir ondalık sayı modelinde küçük bit kümeleri olarak saklar. Üretilen ondalık sayı örüntüsünün son ondalık sayısı hash kodu ve şifrelenmiş veridir. Bu çalışmanın katkısı, özgün bir değişim fonksiyonu, veri/blok sınıflandırması ve iş akışı ile literatürde henüz çalışılmamış işlevsel bir yöntem önermesidir. İYHF yönteminde şifreli hash kodu üretim sürecinin tamamen kararlı olduğu ve şifre çözme sürecinin ortalama %99,83 kararlı olduğu tespit edilmiştir.

Şubat 2024, 141 sayfa

Anahtar Kelimeler: Veri sıkıştırma, Veri gizleme, Güvenli veri iletimi, İmaj tabanlı steganografi, Hash fonksiyonları, Steganografi, Kriptoloji

ABSTRACT

PhD Thesis

IMAGE-BASED STEGANOGRAPHIC METHODS THAT WITHSTAND STATISTICAL ANALYSES FOR END-TO-END HIGH LOAD AND SECURE DATA TRANSMISSION

Timuçin KÖROĞLU

Ankara University
Graduate School of Natural and Applied Sciences
Department of Computer Engineering

Supervisor: Prof. Dr. Refik SAMET

In this work, a two-way hash function (TWHF) is proposed to overcome the trade-off problem between imperceptibility-capacity and security-capacity in data transmission using steganographic methods. It is computationally impossible to obtain original data using one-way hash functions. The proposed method aims to hash/encrypt and decrypt input data of arbitrary length. This allows data to be encrypted and decrypted regardless of its size. This feature allows TWHF to be used in areas where one-way hash functions can be applied as well as in areas where they cannot be applied, such as steganography. Since the data summarized with TWHF is reversible when sent to the receiver in the cover object, the receiver may be able to obtain the original data again. Thus, the specific security properties of two-way hash functions and the privacy, resistance and imperceptibility properties of steganography are combined to realize a much more secure data transmission. The proposed method uses iteration-key and the secondary key to ensure confidentiality. In this study, strong hints were obtained that TWHF fulfils the security properties of one-way hash functions. TWHF stores the original data as smallbit sets in a generated decimal number pattern. The last decimal number of the generated decimal number pattern is the hash code and the encrypted data. The contribution of this study is to propose an original change function, data/block classification and workflow, and a functional method that has not yet been studied in the literature. In the TWHF method, it was found that the encrypted hash code generation process was completely stable and the decryption process was 99.83% stable on average.

February 2024, 141 pages

Key Words: Data compression, Data hiding, Secure data transmission, Image-based steganography, Hash functions, Steganography, Cryptology

TEŞEKKÜR

Bu tez çalışmasının planlanmasında, araştırılmasında, yürütülmesinde ve oluşumunda ilgi ve desteğini esirgemeyen, engin bilgi ve tecrübelerinden yararlandığım, yönlendirme ve bilgilendirmeleriyle çalışmamı bilimsel temeller ışığında şekillendiren ve akademik ortamda olduğu kadar beşerî ilişkilerde de engin fikirleriyle yetişme ve gelişmeme katkıda bulunan sayın hocam Prof. Dr. Refik SAMET'e sonsuz teşekkürlerimi sunarım.

Bu tezin tamamlanmasında en büyük destekçilerim olan aileme teşekkür etmek isterim. Eşim Nihal İnci Köroğlu'na desteği ve sabrı için, çocuklarım Aygül Beyza Köroğlu, Mehmet Enes Köroğlu ve Erva Betül Köroğlu'na her zaman yanımda oldukları için müteşekkirim. Ayrıca bu yolda maddi manevi desteklerini esirgemeyen anne ve babama teşekkürlerimi sunarım. Ailem olmasaydı, bu süreçte aşılması güç zorlukları aşmak bu kadar kolay olmazdı.

Timuçin KÖROĞLU
Ankara, Şubat 2024

İÇİNDEKİLER

KISALTMALAR DİZİNİ

4G	4th Generation
AES	Advanced Encryption Standard
API	Application Programming Interface
Captcha	Completely Automated Public Turing test to tell Computers and Humans Apart
DES	Data Encryption Standard
DF	Değiştirme Fonksiyonu
DLP	Data Loss Prevention
DNA	Deoxyribonucleic Acid
DSA	Digital Signature Algorithm
EC	Elliptic Curve
ECC	Elliptic Curve Cryptography
ECCHH	Elliptic Curve Cryptography Homomorphic Hash
ECDLP	Elliptic Curve Discrete Logarithm Problem
eSTREAM	ECRYPT Stream Cipher Projec
EV	Eklenmiş Veri
FIPS	Federal Information Processing Standards
G	Gerçek
GÇ	Geç
HMAC	Hash-based Message Authentication Code
IDEA	International Data Encryption Algorithm
IIoT	Industrial Internet of Things
IP	Initial Permutation
IV	Initial Vector
İF	İndirgeme Fonksiyonu
İV	İndirgenmiş Veri
İYHF	İki Yönlü Hash Fonksiyonu
JSON	JavaScript Object Notation
KHV	Kripto Ham Veri
LFSR	Linear Feedback Shift Register
LSB	Least Significant Bit
LZW	Lempel-Ziv-Welch
MD5	Message Digest Algorithm
MDS	Maximum Distance Separable
MSB	Most Significant Bit
MSE	Mean Squared Error
NESSIE	New European Schemes for Signatures, Integrity, and Encryption
NIST	National Institute of Standards and Technology
OCE	Open Collaboration Environment
PGP	Pretty Good Privacy
PSNR	Peak Signal-To-Noise Ratio
PRGA	Pseudo-Random Generation Algorithm
RC4	Rivest Cipher 4
RLE	Run-Length Encoding

RNA	Ribonucleic Acid
RSA	Rivest–Shamir–Adleman
S	Sahte
S/MIME	Secure/Multipurpose Internet Mail Extensions
S-Box	Substitution Box
SHA	Secure Hash Algorithm
SPNs	Substitution Permutation Network
SQL	Structured Query Language
SSL	Secure Sockets Layer
TDF	Ters Değiştirme Fonksiyonu
TİF	Ters İndirgeme Fonksiyonu
TLS	Transport Layer Security
VLSI	Very Large Scale Integration
VT	Veri Toplama
WEP	Wired Equivalent Privacy
WPA	Wi-Fi Protected Access
XOR	Exclusive OR

ŞEKİLLER DİZİNİ

ÇİZELGELER DİZİNİ

1. GİRİŞ

Artan internet kullanıcı sayısı, internet teknolojisinin özellikle iletişim, veri depolama ve veri erişimi alanlarında küresel toplumun en temel ihtiyaçlarından biri haline geldiğini göstermektedir. Dünyadaki bireysel internet kullanıcı sayısının %66'ya ulaştığı günümüzde, buna paralel olarak üretilen veri miktarı da çok büyük boyutlara ulaşmıştır (Waseso ve Setiyanto 2023). Yapılan araştırmalar, her bir kişinin günlük 17 megabayt veri ürettiği dijital dünyada, üretilen günlük toplam veri miktarının 2,5 kentilyon baytın üzerinde olduğunu göstermektedir (Barman vd. 2021). Bu durum veri depolama ve iletimi ile ilgili sorunlara yol açmaktadır.

İlk Parkinson yasası, veri depolama ve iletim kapasitesindeki bir artışın veri depolama ve iletim ihtiyacını iki katına çıkarttığını belirtmektedir. Teknoloji ve cihazlar sürekli gelişim halindedir. Bu gelişime rağmen verinin büyüme hızına cevap verilememektedir. Bu nedenle büyük verinin yüksek doğrulukla depolanması ve iletilmesinde sorunlar yaşanmaktadır (Jayasankar vd. 2021). Bu problemi çözmek için veri sıkıştırma, hash fonksiyonları ve kriptoloji alanlarında çok sayıda çalışma yapılmıştır. Bu çalışma alanlarında mevcut problemlere kısmi çözümler bulunmuştur. Ancak, orijinal veriyi bit seviyelerine indirgemek için veri sıkıştırma, hash ve kripto algoritmalarının işlevlerini birleştiren tersinir bir kriptografik hash fonksiyonu rapor edilmemiştir. Tek bir algoritmanın bu işlevleri birleştirerek yerine getirebilmesi araştırmaya değer bir konudur. Tez çalışmasında, iki yönlü bir hash fonksiyonunun bu sorunları çözebilecek bir metodoloji olabileceği tartışılmaktadır. Veri sıkıştırma, hash fonksiyonları ve kriptografik algoritmaların literatürdeki mevcut durumu aşağıdaki gibi özetlenebilir:

Veri sıkıştırma algoritmaları, veriler arasında var olan örüntülerden faydalanarak verilerin boyutunu orijinal boyutlarına kıyasla azaltmayı amaçlar. Bu algoritmalar oldukça gelişmiş olmasına rağmen, orijinal veri boyutu ile sıkıştırılmış veri boyutu arasındaki indirgeme ilişkisi hash fonksiyonlarında olduğu kadar başarılı değildir. Gupta ve Nigam, popüler kayıpsız veri sıkıştırma algoritmaları üzerine yaptıkları araştırmada, bu algoritmaların ortalama %65,21'lik bir alan tasarrufu sağladığını tespit etmişlerdir (Gupta ve Nigam 2021). Bu istatistikler, kayıpsız sıkıştırma algoritmalarının çıktı uzunluklarının,

bir hash fonksiyonu tarafından sağlanan özet veri uzunluğundan önemli ölçüde daha büyük olduğunu göstermektedir. Hash fonksiyonlarında, giriş verisinin boyutu ne olursa olsun, çıkış verisi bit seviyesinde ve sabit uzunluktadır.

Bilginin sayısal veriye dönüştüğü günümüzde, iletilen verinin artması ve ağların küreselleşmesi nedeniyle veri güvenliği önemli bir sorun haline gelmiştir (William vd. 2022). İki yönlü olan şifreleme algoritmalarının aksine hash fonksiyonları tek yönlüdür ve bir hash kodunu ters yönde hashlemek teknik olarak mümkün olsa bile hesaplama gücü bunu imkânsız kılmaktadır (Verma vd. 2021). Bu nedenle verilerin şifrelenmesi ve deşifre edilmesi kriptografik algoritmalar kullanılarak gerçekleştirilir.

Çalışmada, steganografi biliminde var olan algınamamazlık-kapasite ve güvenlik-kapasite arasındaki ödünleşim problemini ortadan kaldırabilmek amacıyla keyfi uzunluktaki verilerin bit seviyesinde ve sabit uzunlukta şifrelenmiş verilere dönüştürülmesi amaçlanmaktadır. Bu bağlamda, hash fonksiyonları gibi veri özetleme fonksiyonlarına ve simetrik kriptografik algoritmalar gibi şifreleme fonksiyonlarına sahip olabilen İYHF önerilmiş ve İYHF'nin kavramları, fonksiyonları, işlemleri, sınırlamaları ve sonuçları sunulmuştur.

İYHF çalışma prensibi, değiştirme ve azaltma fonksiyonları kullanılarak yinelemeli olarak oluşturulan bir ondalık sayı örüntüsüne dayanır. Hashlenecek ve şifrelenecek tüm girdi verileri, sayı örüntüsündeki uygun ondalık sayılarda küçük bit kümeleri olarak saklanır. Sayı örüntüsünü oluşturan ondalık sayıların ikili eşdeğerlerinin uzunluğu 52 bittir. Veri uzunluğunun 52 bit seçilmesinin nedeni şu şekilde açıklanabilir.

Önerilen yöntemin iş akışını test etmek ve istatistiksel veri sağlamak amacıyla Python dilinde kodlanan ondalık sayıların ondalık kısmı en fazla 60 bit olabilmektedir. Sistemin kararlı çalışması için gerekli olan ondalık kısmın uzunluğu 28 bit olarak belirlenmiştir. Ondalık sayının tamsayı kısmının uzunluğu ile ondalık kısmının uzunluğu arasındaki ilişki Denklem 4.1 ile verilmiştir. Denklem 4.1'e göre, ondalık kısım 28 bit alındığında, tamsayı kısmı 24 bit uzunluğu verir. Bu durumda, ondalık verinin tamamı 24+28 = 52 bit uzunluğunda olmaktadır.

İYHF'de sayı örüntüsünün son ondalık sayısının ikili karşılığı hem hash kodu hem de şifreli metindir. Buna göre girdi uzunluğu ne olursa olsun çıktı her zaman 52 bittir. Son ondalık sayının başlangıç verisi olduğu şifre çözme sürecinde, şifreli hash kodu üretme sürecindeki fonksiyonların tersi alınarak örüntü yeniden elde edilir ve örüntüden gizli mesaj çıkarılır.

İYHF, veri sıkıştırma, veri depolama, kriptoloji, veri özetleme ve steganografi gibi birçok alanda kullanılabilme potansiyeline sahiptir. Yöntemin sadece görüntü steganografisindeki işlevi göz önünde bulundurulduğunda, örtü görüntüsünde olabilecek minimum bozulma ile çok büyük miktarlarda veri iletimi sağlayabileceği ve steganografik yöntemin başarısını arttırabileceği öngörülmektedir (Koroglu ve Samet 2023) .

1.1 Araştırmanın Temeli ve Önemi

Tek yönlü hash fonksiyonları, bilgi güvenliği ve kriptografi alanında temel bir role sahiptir. Bu matematiksel işlevler, herhangi bir veri kümesini belirli bir sabit uzunluktaki bir özete dönüştürür. Temel özellikleri, hızlı özet üretme, küçük değişikliklerin büyük değişikliklere yol açması ve üretilen özetin tersine dönüştürülememesidir. Şifreleme, dijital imza oluşturma, parola depolama ve bütünlük kontrolü gibi birçok uygulamada kullanılan bu fonksiyonlar, aynı girdiyi kullanarak her seferinde aynı özeti üretme, çarpışma direnci ve tek yönlü olma özelliklerine sahiptir. Bu özellikler sayesinde, tek yönlü hash fonksiyonları, verilerin güvenli depolanması, iletilmesi ve kimlik doğrulama süreçlerinde kullanılması gibi kritik güvenlik uygulamalarında önemli bir güvenlik katmanı sağlar.

Tek yönlü hash fonksiyonları çıkardıkları özeti kullanarak orijinal veriyi yeniden elde edemez. Bu özellikleri onlara bir çok alanda kullanılması için avantaj sağlar ve tercih sebebi olurlar. Ancak tek yönlü hash fonksiyonlarının sahip olduğu güvenlik özelliklerini sağlayabilen iki yönlü hash fonksiyonları daha fazla uygulama alanında daha fazla avantaj sunabilir. Örneğin, tek yönlü hash fonksiyonu kullanılarak özeti alınan bir veri, steganografik yöntemlerle bir örtü nesnesi içinde alıcıya gönderildiğinde alıcı tarafında

orijinal veri yeniden elde edilemez. Bunun sebebi özetin tersine dönüşü olmamasıdır. Ancak iki yönlü bir hash fonksiyonu ile özeti alınan veri örtü nesnesi içinde alıcı tarafına gönderildiğinde tersinir özelliğe sahip olduğu için alıcı, orijinal veriyi yeniden elde etme imkanına sahip olabilir. Böylece hash fonksiyonlarının kendilerine özgü güvenlik özellikleri ile steganografinin gizlilik, direnç ve algılanamazlık özellikleri biraraya gelerek çok daha güvenli bir veri iletimi gerçekleştirilmiş olur. İki yönlü hash fonksiyonları, bilgi iletiminin yoğun olduğu internet ya da türevi farklı ağlarda veri iletimi sürecinde ihtiyaç duyulan bant genişliğini azaltma potansiyeline sahip olabilir. Bunun nedeni, ağ üzerinde büyük boyutlu orijinal veriler yerine bu verilerin özetlerinin dolaşması olarak açıklanabilir. Özeti dolaşan veri, alıcısına ulaştığında alıcı tarafında orijinal veri yeniden elde edilebilir. Böylece bant genişliği çok daha az kullanılmış olur. İnternet gibi yoğun bilgi alışverişi olan ağlarda bant genişliği bağlamında iki yönlü hash fonksiyonlarının sağlayacağı fayda çok yüksek boyutlarda olabilir.

İki yönlü hash fonksiyonlarının kullanılabileceği ancak tek yönlü hash fonksiyonlarının kullanılamayacağı alanlardan birisi de veri depolamadır. Günümüzde veri depolama cihazları her ne kadar gelişmiş olsalar da bu gelişim depolama alanlarına duyulan ihtiyacı karşılayamamaktadır. Veri sıkıştırma algoritmaları bu ihtiyacı ancak kısmi bir biçimde karşılayabilmektedir. Oysa iki yönlü hash fonksiyonlarının bu alanda kullanılmasının faydaları inanılmaz olabilir.

1.2 Tezin Kapsamı

İYHF metodu keyfi uzunluktaki bir veriyi özetlemeyi ve özetinden orijinal veriyi yeniden elde etmeyi hedefler. Önerilen metodun aksine literatürde yer alan tüm hash fonksiyonları tek yönlüdür. Bu tez çalışması ile literatürde ilk kez, tek yönlü hash fonksiyonlarının iki yönlü olabilme durumu ele alınmıştır.

İki yönlü hash fonksiyonlarının hedefine ulaşabilmesi için nasıl bir metodolojiye sahip olması gerektiği çalışma kapsamındadır. Tek yönlü hash fonksiyonları metodolojisinde veri özetleme süreci; XOR, s-box ile bitlerin yerlerini değiştirme, bit kaydırma vb. ilkel işlemlerin belirli tur sayısınca yapılması ile gerçekleştirilir. Bu işlemler ile veri sabit bir

uzunluğa dönüştürülebilir. Ancak bu fonksiyonların orijinal veriyi yeniden elde etme hedefleri olmadığı için yapılan bu işlemler geri dönüşü mümkün hale getirecek niteliğe sahip değillerdir. Önerilen metot, veriyi özetlemenin yanısıra özetten orijinal veriyi yeniden elde etme süreci ile de ilgilenir ve bu hedefe yönelik metodolojiyi barındırır. Ayrıca tek yönlü hash fonksiyonlarının sağladığı güvenliği sağlamak önerilen metodolojinin kapsamındadır. Bu hedefleri yerine getirmek için İYHF metodu mevcut tek yönlü hash fonksiyonlarından çok farklı bir sürece sahiptir. Bu süreçte sabit uzunluğa sahip ondalık verilerden oluşan ve birbirine matematiksel fonksiyonlarla bağlı bir sayı örüntüsü üretilir. Sayı örüntüsü içinde yer alan veriler ve bu verilerin oluşturduğu bloklar sınıflandırılır. Sınıflandırılan bloklardaki bazı veriler Huffman kodlama ile indirgenmeye uygundur. İndirgenen veriler orijinal verinin bitlerini, indirgenme miktarlarına bağlı olarak küçük bit kümeleri halinde bünyesine katarlar. Böylece orijinal verinin bitleri üretilen sayı örüntüsüne dahil olur. Sayı örüntüsünü oluşturan ondalık veri uzunlukları sabittir. Bu yüzden sayı örüntüsünün son üretilen ondalık sayısı hash kodu olarak alındığında veri özetlenmiş olur. Şifre çözme sürecinde, hash kodu başlangıç verisi olarak kabul edilerek matematiksel olarak birbirine bağlı sayı örüntüsü ters matematiksel fonksiyonlarla yeniden oluşturulur ve örüntü içindeki orijinal veri, kurallar çerçevesinde çıkartılarak yeniden elde edilir.

Çalışma kapsamında iki yönlü hash fonksiyonlarının olası mevcut/farklı kullanım alanları ve sağlayabileceği faydalar detaylandırılarak açıklanmıştır. İYHF'nin farklı kullanım alanlarının başında steganografi ve veri depolama alanları gelmektedir. Yakın gelecekte kuantum bilgisayarların literatürdeki tüm kriptografik algoritmaların sunduğu güvenliği tehdit edeceği yönünde öngörüler mevcuttur. Bu nedenle steganografi yakın gelecekte sunduğu farklı güvenlik özellikleri ile ön plana çıkabilir. Çalışma kapsamında önerilen metot ile steganografinin birlikte kullanımı veri güvenliği bağlamında gelecek için umut verici çalışmalardan birisi olma potansiyeline sahiptir.

Ayrıca İYHF'nin tek yönlü hash fonksiyonlarının sahip olduğu ön görüntü direnci, ikinci öngörüntü direnci ve çarpışmaya karşı dayanıklılık özellikleri ele alınmış ve bu özellikleri ne kadar sağlayabileceği ile ilgili testler, sonuçları ile birlikte literatüre sunulmuştur.

1.3 Tezin Literatüre Katkısı

İYHF metodolojisi çeşitli kavramlar, fonksiyonlar ve işlemler içermektedir. İYHF'nin önemli bir unsuru olan veri değiştirme fonksiyonu bu çalışmada özgün olarak geliştirilmiştir. Veri değiştirme fonksiyonu sadece önerilen yöntemde değil, sözde rastgele sayı üretiminin önemli olduğu birçok alanda kullanılabilir.

Bu çalışmada ondalık veriler işlevlerine ve yapılarına göre dört farklı veri türüne ayrılmış ve her bir türe işlevlerine göre anlamlar yüklenmiştir. Veri sınıflandırması Huffman kodlamasını esas alır ve literatürde ilk kez tanımlanmıştır. Sürecin istikrarlı çalışmasını sağlamak için veri blokları da işlevlerine göre sınıflandırılmıştır. Veri sınıflandırmasının yanı sıra blok sınıflandırması bu çalışmanın literatüre bir diğer katkısıdır.

Tezin literatüre en büyük katkısı iki yönlü hash fonksiyonu metodolojisinin literatürde ilk kez önerilmiş olmasıdır. Literatürdeki mevcut hash fonksiyonlarının tamamı tek yönlüdür. Bu yönüyle tez çalışması özgün bir çalışmadır ve bu konuya ilgi duyan araştırmacılar için bir başlangıç çalışması niteliğindedir. İYHF metodolojisi sadece keyfi uzunluktaki verileri özetleme yönüyle değil tek yönlü hash fonksiyonlarının güvenlik özelliklerine sahip olabilme potansiyeli ile de öne çıkmaktadır. Yapılan çeşitli testler İYHF'nin tek yönlü hash fonksiyonlarının sahip olduğu güvenlik özelliklerine sahip olabileceği yönünde güçlü ipuçları vermektedir. Bu nedenlerden dolayı İYHF tek yönlü hash fonksiyonlarının kullanıldığı tüm alanlarda kullanılabilir. İYHF ayrıca çift yönlü özelliğinden dolayı steganografi ile veri iletimi ve veri depolama cihazlarında veriyi oldukça küçük boyutlarda saklayabilme imkanı sağlayabilecek potansiyele sahiptir.

1.4 Tezin Yapısı

Tez kapsamında önerilen metodoloji; özgün bir veri değiştirme fonksiyonu ve Huffman kodlaması ile veri-blok sınıflandırmalarının birlikte kullanıldığı özgün bir iş akışında detaylı bir biçimde açıklanmıştır.

Tez 8 ana bölümden oluşmaktadır. Çalışmanın geri kalanını şu şekilde bölümlendirilmiştir.

Bölüm 2, tez çalışmasının esas alındığı veri güvenliği üst başlığı altında literatürde yapılan ilgili çalışmaları içermektedir. Bu çalışmalar kriptografik algoritmaları, hash fonksiyonlarını, steganografik uygulamaları ve bunların çeşitli hedefler doğrultusunda hibrit kullanımlarını içerir. Bu çalışmalarda, literatürde var olan algoritmaların sağladığı güvenliğin ve verimliliğin arttırılması ön planda tutulmuştur. Bölüm 3'te kriptografik algoritmalar ve hash fonksiyonlarının tanımları, esasları, çalışma iş akışları ile hedefleri detaylandırılarak açıklanmıştır. Bölüm 4'te çalışma kapsamında önerilen iki yönlü hash fonksiyonu metodolojisi tüm detayları ile açıklanmıştır. Bu kapsamda metodolojide kullanılan fonksiyonlar, veri sınıflandırmaları, blok türleri, şifreli hash kodu üretme süreci ve şifre çözme süreci detaylı bir biçimde açıklanmıştır. Ayrıca İYHF'nin güvenlik özellikleri ve olası kullanım alanları da bu bölümde açıklanmıştır. Bölüm 5, önerilen metot uygulamasını bir örnek üzerinde, ekran çıktıları ve elde edilen veriler eşliğinde göstermektedir. Bölüm 6'da, İYHF güvenlik analiz testleri sonuçlarından elde edilen veri/grafikler ile birlikte NIST testleri sonuçları ve İYHF metodunun steganografi alanında kullanılması ile ilgili güvenlik ölçüm sonuçları sunulmuştur. Bölüm 7'de bir önceki bölümde elde edilen veri ve grafiklerin değerlendirilmesi yapılmıştır. Bölüm 8'de sonuç ve öneriler paylaşılmıştır.

2. KAYNAK ÖZETLERİ VE LİTERATÜR DEĞERLENDİRMESİ

Günümüzde siber güvenlik alanında çalışan araştırmacıların, internet üzerinden iletişim ve veri alışverişinin giderek yaygınlaşması nedeniyle artan siber saldırılara karşı verileri güvende tutmanın yollarını araştırarak çözümler sunması oldukça önemli hale gelmiştir. Saldırılara karşı birçok etkili yöntem önerilmiş ve uygulanmıştır. İlgili çalışmalar aşağıdaki gibi özetlenebilir.

Rajeshwaran ve Kumar çalışmalarında hücresel-otomata tabanlı bir hash algoritması önermişlerdir. Önerilen algoritma kullanılarak üretilen hash koduna difüzyon ve karışıklık sağlayarak güçlü bir kriptografik hash fonksiyonu tasarlamışlardır. Algoritmanın etkinliğini göstermek için hash fonksiyonunun iki temel özelliği test edilmiştir. Bunlar çığ etkisi ve çarpışma olasılığıdır. Bu çalışmada, bu iki temel özellik sadece bir farklı harf içeren iki cümle ve aynı cümle için iki farklı anahtar kullanılarak test edilmiştir (Rajeshwaran ve Kumar 2019). Elde edilen sonuçlara göre çığ etkisi özellikleri ve çarpışma olasılığı güçlenmiştir. Ancak, bu özelliklerin güçlendirildiğine dair kanıtlar ek veri kümelerinin kullanılmasıyla daha açıklayıcı olacaktır.

Kheshaifaty ve Gutub güçlü kimlik doğrulama için captcha, kriptografi ve hash fonksiyonlarını birleştiren çok katmanlı bir sistem sunmuştur. Captcha girişi onaylandıktan sonra, şifre AES algoritması kullanılarak şifrelenmiş ve hashlenmiştir. Giriş onayı için şifrenin hash kodunun sistemde depolanan kodla eşleşmesi gerekmektedir. Bu çalışma mevcut kriptografik algoritmalara ve hash fonksiyonlarına dayanmaktadır (Kheshaifaty ve Gutub 2020). Önerilen yöntem, kriptografik algoritmaların ve hash fonksiyonların hibrit kullanımını sunarak literatüre katkı sağlamaktadır. Ara katmanlar için özgün bir öneri sunulmamıştır.

Li ve Ge multimedya iletişim güvenliğini artırmak için çapraz bağlı harita kafeslerine dayalı bir kriptografik özet fonksiyonu önermişlerdir. Bu çalışmada, çapraz bağlı haritalara girilecek başlangıç değerleri doğrusal bir kaotik harita kullanılarak üretilmiş ve orijinal mesaj özellikleri arasındaki korelasyonu artırmak için bir matris kullanılarak genişletilmiştir. Mesaj bloklarının ara hash değerleri, çapraz bağlı haritaların girdisi olan

bir matris kullanılarak üretilmiştir. Nihai hash kodunu oluşturmak için tüm mesaj blokları paralel olarak işlenmiş ve ara hash değerleri mantıksal olarak işlenmiştir. Bu çalışmada, üretilen hash kodu, bir hash kodunun sahip olması gereken çarpışma, karışıklık ve yayılma özellikleri açısından test edilmiştir (Li ve Ge 2019). Elde edilen sonuçlar bu alanda yapılan çalışmalardan istatistiksel olarak daha iyi olup sadece mevcut çalışmaların geliştirilmesine katkı sağlamıştır

Ali ve Farhan, 128-2096 bit aralığında küçük çıkış uzunluğu nedeniyle çeşitli saldırı türlerine karşı başarısız olan MD5 algoritmasının çıkışını dinamik olarak genişleterek MD5 algoritmasını saldırılara karşı güçlendiren bir algoritma sunmuşlardır. Algoritma bu işlevi, RNA kodlaması için Doğrusal-Geri Beslemeli Kaydırma Kaydedicisi (LFSR) ve DES algoritmasının başlangıç permütasyon (IP) tablosu kullanılarak oluşturulan bir anahtar yardımıyla gerçekleştirir. Anahtar rastgele bir matris oluşturmak için kullanılır. Sonuçlar, MD5 algoritmasının çıkış uzunluğunu artırarak saldırılara karşı başarısını artırdığını göstermiştir (Ali ve Farhan 2020). Ancak Rivest tarafından 1992 yılında 128 bit uzunluğundaki MD5 algoritması sunulduktan sonra birçok hash fonksiyonu MD5'ten daha başarılı bir şekilde üretilmiştir. Yazarlar önerdikleri algoritmayı literatürdeki diğer hash fonksiyonları ile karşılaştırmamış, sadece orijinal MD5 algoritması ile karşılaştırmışlardır. Bu durum önerilen algoritmanın diğer algoritmalarla karşılaştırıldığında ne kadar başarılı olduğunu belirsiz hale getirmektedir.

William ve diğerleri AES, ECC ve SHA-256 algoritmalarını içeren hibrit bir algoritma kullanarak şifreleme ve şifre çözme verimliliğini artırmayı amaçlamıştır. Önerilen yöntemde mesaj, AES algoritması iş akışında, ECC algoritması ile şifrelenmiş bir anahtarın kullanımıyla şifrelenmiştir. Şifrelenen mesaj SHA-256 algoritması kullanılarak özetlenmiş ve alıcı için hazır hale getirilmiştir. Önerilen algoritma literatürdeki diğer algoritmalarla karşılaştırılarak verimliliğinin artırıldığı gösterilmiştir (William vd 2022). Bu çalışmada karşılaştırılan algoritmaların detaylarının verilmesi verimlilik konusunda daha doğru bir değerlendirme yapılmasını sağlayacaktır. Ayrıca algoritmanın çıktılarını doğrudan diğer algoritmalara beslenmesi yerine, özgün olarak geliştirilmiş ara katmanların kullanılması çalışmanın katkı düzeyini daha çok artırılabilirdi.

Abouchouar ve diğerleri klasik hash fonksiyonlarından farklı olarak yinelemesiz bir yapı kullanan bir hash fonksiyonu önermiştir. Yazarlar, literatürdeki klasik hash fonksiyonlarının aynı tasarım modeline sahip olduğunu ve savunmasız olduğunu savunmuşlardır. Önerilen yöntemin temel farklılıkları, bir başlangıç vektörüne sahip olmaması ve iç yapısındaki dönüşüm işlemlerinin sıkıştırma yerine genişletmeye dayalı olmasıdır. Bu yöntemde veriler paralel bloklar halinde genişletme fonksiyonuna girmiştir. Her bir fonksiyon çıktısı bir sonraki fonksiyonun çıktısı ile XOR işlemine tabi tutulmuştur. Son genişletme fonksiyonu ile bir önceki genişletme fonksiyonunun XOR sonucu nihai hash kodunu verir. Bu çalışmada, önerilen konseptin klasik hash fonksiyonlarına göre saldırılara karşı daha dirençli olduğuna dair teorik bilgiler verilmiştir (Abouchouar vd. 2020). Bir hash fonksiyonunun özellikleri ve saldırılara karşı sağlamlığı veri kümeleri kullanılarak test edilmelidir. Elde edilen sonuçlar test verilerine dayalı olarak literatürdeki hash fonksiyonları ile karşılaştırılmalı ve yöntemin etkinliği kanıtlanmalıdır.

Abroshan, bulut bilişim için Blowfish, EC ve MD5 algoritmalarını hızlı ve düşük bellek tüketimi ile birleştiren hibrit bir yapı önermiştir. Abroshan, MD5 ile veri bütünlüğünü sağlarken Blowfish ile veriyi ve EC ile anahtarı şifrelemiştir. Böylece çalışmalarında güvenlik ve performansı artırmışlardır (Abroshan 2021). Sonuçlar, veri boyutu arttığında önerilen hibrit yapının AES algoritmasına göre daha yavaş çalıştığını göstermektedir. Bununla birlikte, bulut bilişimde paylaşılan kaynaklar genellikle çok büyüktür. Bu bağlamda yazarın önerdiği hibrit yapının geliştirilmesi gerekmektedir.

Wang ve arkadaşları SHA-256 ve iteratif kaydırma kullanan kaotik bir görüntü şifreleme algoritması sunmuştur. Bu çalışmada girdi olarak W × H boyutunda düz gri tonlamalı bir görüntü kullanılmış ve çıktı olarak şifrelenmiş renkli bir görüntü elde edilmiştir. Orijinal görüntünün girdi olarak kullanıldığı SHA-256 algoritmasında, çıktı anahtar olarak kullanılır. Bu anahtar kullanılarak doğrusal kaotik haritaya girdi olarak altı başlangıç değeri üretilir. Görüntü, kaotik harita çıktısı ve diğer işlemlerden elde edilen rastgele bir dizi kullanılarak piksel düzeyinde şifrelenir (Wang vd. 2019). Yazarlar, deneyler ve güvenlik analizleri yaparak saldırılara karşı dayanıklı bir algoritma geliştirmişlerdir.

Ancak yapılan analizler, çalışmanın literatürdeki diğer algoritmalara yakın istatistiksel değerler ürettiğini göstermiştir.

Alotaibi ve diğerleri mobil cihazlarda kimlik doğrulama sistemleri için hash fonksiyonu, AES algoritması ve görüntü steganografi tekniklerini birleştiren hibrit bir yapı önermişlerdir. Bu yapıya göre şifrenin hash kodu, kullanıcı adının anahtar olarak kullanıldığı AES algoritması kullanılarak şifrelenmiştir. Şifrelenmiş parola hash kodu, LSB stego tekniği kullanılarak bir kapak görüntüsüne gömülmüştür. Önerilen yapının testleri, farklı hash fonksiyonları kullanılarak oluşturulan parola hash kodunun kapak görüntüsüne gömülme süresine dayanmaktadır (Alotaibi vd. 2019). Ancak, önerilen yapının bu alandaki diğer çalışmalarla karşılaştırılması, yapının güvenliği hakkında daha iyi bilgi sağlayacaktır.

Koptyra ve Ogiela farklı bir blok zinciri türü olarak görüntü zinciri yapısını sunmuşlardır. Görüntü zinciri yapısında, blok verileri görüntülerin içine gömülmüştür. Bloklar JSON formatlı alan ve değer çiftlerinden oluşmaktadır. Bir bloğun en önemli unsuru, zincirdeki görüntülerin hash kodudur. Her görüntü kendisinden önceki görüntünün hash kodunu saklar. Görüntü zinciri yapısındaki görüntülerden herhangi birinin zincirden çıkarılması veya değiştirilmesi durumunda sahtecilik tespit edilir (Koptyra ve Ogiela 2020).

Seok ve diğerleri çalışmalarında blockchain teknolojisinin, veri bütünlüğünü sağlamada etkin olan bir teknoloji olarak çeşitli alanlarda kullanımını incelemektedir. Özellikle, blockchain'in Endüstriyel Nesnelerin İnterneti (Industrial Internet of Things - IIoT) ile birleşmesinin yarattığı potansiyeli ele almaktadır. Kaynakları kısıtlı IIoT cihazlarından oluşan ağlarda, mevcut blok zinciri teknolojisinin uygulanmasındaki zorlukları aşmak amacıyla hafif blok zinciri tekniklerini incelemişlerdir. Araştırmacılar, hafif blok zinciri inşa etme tekniklerinin genellikle kriptografik performansı yeterince göz önüne almadığını belirtmişlerdir. Bu bağlamda, kaynak kısıtlı ortamlar için önerilen hafif hash fonksiyonlarının blok zinciri mimarilerinde nasıl kullanılabileceğini analiz etmişler ve yeni bir hafif hash tabanlı blok zinciri mimarisi önermişlerdir. Bu mimari madencilik için kullanılan hash algoritmasını ağ trafiğine göre dinamik olarak değiştirebilmektedir. Bu

öneri, kaynak kısıtlı IIoT ortamlarında daha etkili ve verimli bir blok zinciri uygulamasını mümkün kılabilmektedir (Seok vd. 2019).

Kheshaifaty ve Gutub yaptıkları çalışmada, çevrimiçi bilgisayar kullanıcılarının kimlik doğrulama sürecini ele almaktadır. Kimlik doğrulama, captcha'lar veya şifrelenmiş karma parolalar gibi yöntemler kullanılarak gerçekleştirilir ve temel amacı kullanıcı verilerini saldırganlardan korumak ve güvende tutmaktır. Yazarlar, bu bağlamda metin tabanlı captcha, güvenlik sistemi içinde kripto hash fonksiyonları ile entegre edilmiş bir kimlik doğrulama süreci önermektedir. Önerilen sistem, çoklu erişim saldırılarını önlemek için güvenlik sistemi içinde bir captcha modülü, giriş sistemi bileşeni ve SHA1 hash fonksiyonunu içermektedir. Bu modül, robot-otomasyonun kontrolü ve kimlik doğrulama sürecinin etkin güvenliği için SHA1 hash ve kriptografi ile birleştirilmiştir. Yazarlar, captcha kripto hash fonksiyonları sisteminin önerilen entegrasyonunun, eski sistemlere göre yaklaşık %30'luk etkili bir iyileştirmeye sahip olduğunu belirtmektedir (Kheshaifaty ve Gutub 2020).

Chen yaptığı çalışmada, bulut ortamlarında düz metinlerin güvenliğini artırmak amacıyla homomorfik hash algoritmalarının kullanımı incelenmiştir. Bu algoritmalar, her bir düz metni hash değeriyle temsil ederek, bulut ortamlarında sadece hash değerlerini depolayarak ve gelecekteki ihtiyaçlar için bu hash değerlerini hesaplayarak veri güvenliği sağlamaktadır. Ancak, daha güçlü güvenlik özelliklerine sahip homomorfik hash algoritmalarının, daha uzun hash değerleri üretme ve özetleme sürelerine ihtiyaç duyabileceği bilinmektedir. Homomorfik algoritmalar, belirli hesaplanabilir işlemleri gerçekleştirmeye olanak tanıyan bir şifreleme şemasıdır. Bu şema, genellikle üçüncü bir tarafın (örneğin, bulut hizmet sağlayıcısı) şifrelenmiş veri üzerinde işlemler gerçekleştirmesine izin verir. Homomorfik şifreleme, bu işlemlerin sonucunda elde edilen çıktıların özelliklerini ve şifrelenmiş verinin biçimini korur. Bu sayede, güvenlik korunurken veri manipülasyonu ve hesaplamalar üçüncü taraflarca gerçekleştirilebilir. Bu bağlamda, çalışma, eliptik eğri kriptografisi (ECC) temel alınarak geliştirilen homomorfik bir hash fonksiyonu olan ECC tabanlı homomorfik hash (ECCHH) önermektedir. Homomorfik hash algoritmasının özellikleri, önerilen ECC tabanlı yöntemde kullanılarak daha etkili bir çözüm sunulmaktadır. Ayrıca, önerilen yöntemi

desteklemek amacıyla matematiksel modeller ve pratik durumlar sunulmuştur (Chen 2023).

Adeniyi ve diğerleri yaptıkları çalışmada hedefleri, güvenliği artırmak ve hassas bilgilerin doğrulanmasını sağlamak amacıyla SHA-256 uygulaması ile RSA ve ElGamal kriptografik algoritmalarını kullanarak dijital imza oluşturmaktır. Günümüzde güvenlik, giderek karmaşık bir görev haline gelmiştir ve bu çalışma, paylaşılan verilerin kimlik doğrulamasını gerçekleştirebilmek adına SHA-256 fonksiyonunun kriptografik algoritmalara başarıyla uygulanmasını amaçlamaktadır. Yazarlar bu amacı gerçekleştirmek için kullanılan metodolojiyi C# programlama dilinde kodlamıştır. Kodlamada dijital imza oluşturmak amacıyla SHA-256 hash fonksiyonu, RSA ve ElGamal kriptografik algoritmaları birlikte kullanılmıştır. Yapılan deneysel sonuçlar, RSA algoritmasının şifreleme ve imza doğrulama süreçlerinde ElGamal'a göre daha iyi performans gösterdiğini, ancak ElGamal'ın şifre çözme ve imza oluşturma süreçlerinde RSA'dan daha iyi performans sergilediğini ortaya koymuştur. Bu bağlamda, çalışma, modern kriptografik algoritmaların etkin bir şekilde kullanılmasıyla dijital imza oluşturmanın güvenilirliğini ve veri paylaşımının güvenliğini artırmayı amaçlamaktadır (Adeniyi vd. 2022).

Piccolboni ve diğerleri yaptıkları çalışmada, kripto kötüye kullanımları dinamik olarak tespit eden açık kaynaklı bir araç olan CRYLOGGER'ı tanıtmaktadır. Kriptografik algoritmalar, güvenli sistemlerin temel bileşenleridir ve bu araç, geliştiricilerin sabit anahtarlar ve zayıf parolalar kullanarak kripto uygulama programlama arayüzlerini (API) kötüye kullanmasını önlemeyi amaçlamaktadır. CRYLOGGER, yürütme sırasında kripto API'lerine aktarılan parametreleri günlüğe kaydederek ve çevrimdışı olarak bir kripto kuralları listesi kullanarak bunların meşruiyetini kontrol ederek kripto kötüye kullanımlarını tespit eder. CryptoGuard ile karşılaştırıldığında, CRYLOGGER'ın CryptoGuard'ın sonuçlarını tamamladığı ve hem statik hem de dinamik yaklaşımları birleştirdiği gösterilmiştir. Çalışma, CRYLOGGER'ın dinamik ve otomatik olarak binlerce Android uygulamasındaki kripto kötüye kullanımlarını tespit edebildiğini göstermek için Google Play Store'dan indirilen 1780 popüler Android uygulamasını analiz etmektedir. Tersine mühendislik yapılan 28 Android uygulamasında

CRYLOGGER tarafından işaretlenen sorunlar doğrulanmış ve en kritik güvenlik açıkları uygulama geliştiricilerine bildirilerek geri bildirimler toplanmıştır. Yazarlar bu çalışmada, kripto güvenliğini artırmak ve kötüye kullanımları önlemek amacıyla etkili bir aracın potansiyelini vurgulamaktadır (Piccolboni vd. 2021).

Rahul ve diğerleri yaptıkları çalışmada, günümüzde yaygın olarak kullanılan dijital görüntülerin gizliliğini korumak amacıyla geliştirilen bir görüntü şifreleme şemasını tanıtmaktadır. Yazarlar, dinamik DNA kodlamasına ve kaotik haritalara dayanan bu şema ile güvenli, etkili ve güvenilir bir şifreleme tekniği sunmaktadır. Önerilen şema, Lojistik harita, Henon haritası ve Lorenz sistemi gibi basit yapılarla ilişkilendirilmiş kaotik haritaların kullanımını içerir. Ayrıca, SHA-256 hash tekniği ve zigzag geçişi gibi güçlendirici özelliklerle algoritmik bir yapıya sahiptir. Geliştirilmiş bir DNA kodlama şeması, her seferinde dört biti aynı anda kodlayabilme yeteneği ile öne çıkar. Makalede vurgulanan diğer önemli özellikler arasında düşük işlem maliyeti, yüksek rastgelelik, geniş anahtar uzayı, esnek parametre uzayı, yüksek hassasiyet ve hızlı yürütme hızı bulunmaktadır. Her şifreleme ve şifre çözme oturumu için benzersiz ve rastgele anahtarlar üretilir, bu da sistemdeki güvenliği artırır. Sonuç olarak, önerilen şema, hassas dijital görüntülerin çeşitli kriptografik saldırılara karşı etkili bir şekilde korunmasını sağlar (Rahul vd. 2023).

Pérez ve diğerleri yaptıkları çalışmada, mobil cihazlarda uygulandığında güvenli ve verimli olduğu kanıtlanmış hash fonksiyonlarını kullanan bir dijital imza protokolü geliştirilmiştir. Protokol, kullanıcılar arasında açık anahtar alışverişi için bir Sertifika Makamı modelini içermektedir. Yapılan deneysel araştırma, hash imza sisteminin mobil iletişim cihazlarında test edildiğini ve teorik bir araştırma olmaktan ziyade mevcut bir teoriyi veya performansı geliştirmeyi amaçladığını göstermektedir. Deneysel sonuçlar, hash imzanın ECC ile karşılaştırıldığında kriptografik anahtarların üretilmesi, imzalama ve doğrulama süreçlerinde verimliliği artırdığını göstermektedir. Hash imzanın, özellikle 2048 bit anahtar üretimi sırasında RSA'dan daha hızlı olduğu görülmüştür. Ayrıca, daha büyük RSA anahtarlarının oluşturulması daha fazla zaman alırken, hash anahtarlarının boyutunu artırmak gereksinimini ortadan kaldırmaktadır. Bu çalışma, elektronik ticarette dijital imzaların oluşturulmasında kullanılabilecek hash fonksiyonlarına dayalı yeni bir

yaklaşım sunmaktadır. Önerilen yöntem, hash fonksiyonlarının mobil ağ cihazları için uygun olmasını sağlayan yüksek hızı ve düşük donanım gereksinimleri nedeniyle öne çıkmaktadır. Ayrıca, yöntemin güvenliği, yaygın olarak bilinen ve tartışılan hash kriptografisinin güvenliğine dayanmaktadır (Pérez vd. 2019).

Shyla ve diğerleri yaptıkları steganografik çalışmada, bir genetik algoritma ile verinin gizleneceği uygun kapak görüntüsünün seçimini ve bu veriyi gizlemek için kapak nesnesinin hangi piksellerinin kullanılabilirliğini araştırmıştır. Böylece, uygun kapak görüntüsü ve pikselleri seçilerek, gizli verinin stego görüntüsünde çok az değişikliğe neden olması amaçlanmıştır. Gizli veri başka bir taşıma görüntüsü içindedir. Önerilen yöntem iki aşamadan oluşmaktadır. İlk aşamada, yüz resmi veritabanından seçilen yüz resimleri arasından sekiz tanesi kapak görüntüleri olarak belirlenir. Bu aşamada, taşıma ve kapak görüntülerinden istatistiksel veriler bu süreç için analiz edilir. İkinci aşamada ise, seçilen kapak görüntüsünün hangi piksellerine gizli verinin yerleştirilmesi gerektiği optimize edilir. Bu süreçte, verinin yerleştirilebileceği milyonlarca seçenek bulunmaktadır. Bu seçenekler arasında en uygun olanı, genetik algoritma tarafından belirlenir (Shyla vd. 2021).

Mohammed ve Al Saffar yaptıkları çalışmada, veri iletimini daha güvenli hale getirmek için görüntü steganografisi için McEliece şifreleme sistemini kullanmıştır. McEliece şifreleme sistemi, hata düzeltme kodlarına dayanan bir açık anahtarlı şifreleme sistemidir. Yazarlar çalışmalarında, önce metni, McEliece algoritması kullanılarak şifrelemiş sonrasında şifreli metni, LSB (Least Significant Bit) yöntemini kullanılarak kapak görüntüsünde gizlemiştir. Böylece, oluşturulan stego görüntü alıcıya gönderilerek veri iletimi daha güvenli hale getirilmiştir (Mohammed ve Al Saffar 2021).

Yapılan çalışmalar incelendiğinde önerilen yöntemlerin mevcut kriptografik hash fonksiyonlarının özelliklerini iyileştirmeye yönelik olduğu görülmüştür. Önerilen çalışmalarda hash fonksiyonları için çarpışma, karışıklık ve yayılma özelliklerinin iyileştirilmesi hedeflenirken, veri güvenliği algoritmaları için zaman verimliliği, düşük bellek tüketimi ve güvenlik artırımı hedeflenmiştir. Bu hedeflere ulaşmak için araştırmacılar, literatürdeki algoritmaları birleştiren hibrit sistemler önermişlerdir. Ayrıca

mevcut ilkellerin ara katmanlarında parametrik değişiklikler yapmışlardır. Araştırmacılar tarafından kriptografik hash fonksiyonları üzerine yapılan çalışmaların çoğu, yeni yöntemler önermekten ziyade mevcut yöntemleri iyileştirmeye yöneliktir.

İYHF ile literatürde henüz çalışılmamış özgün bir yöntem önerilmiştir. Önerilen yöntem, özgün bir fonksiyon, veri sınıflandırma kavramları ve ilkeleri ile bunları birleştiren bir iş akışı içermektedir. Bu yöntemin ana fikri, verinin hash kodunu üretmek ve bu hash kodundan veriyi tekrar elde etmektir. Bu özelliği ile İYHF steganografi, kriptoloji, hash fonksiyonları ve veri depolama gibi birçok alanda kullanıma uygundur. Önerilen yöntem, kriptografik özet fonksiyonlarına farklı bir bakış açısı ile literatürdeki diğer çalışmalar arasında özel bir yere sahip olma potansiyeline sahiptir. İYHF, literatürdeki mevcut algoritmaların istatistiksel değerlerini iyileştirmeyi amaçlayan çalışmaların aksine özgün bir çalışma olarak katkı sağlamaktadır.

3. VERİ GÜVENLİĞİ

Veri güvenliği, verileri yaşam döngüsü boyunca değiştirilmekten ve yetkisiz erişimden korumak için kullanılan süreçleri ifade eder. Veri güvenliği uygulamalarına örnek olarak, bir ağ üzerinden iletişim kurarken verilerin şifrelenmesi ve şifrelerinin çözülmesi, hassas verilerin farklı değerler veya sembollerle temsil edilerek korunması ve anahtar üretimi ve seçimi gibi anahtar yönetimi uygulamaları verilebilir (Alqad vd. 2019).

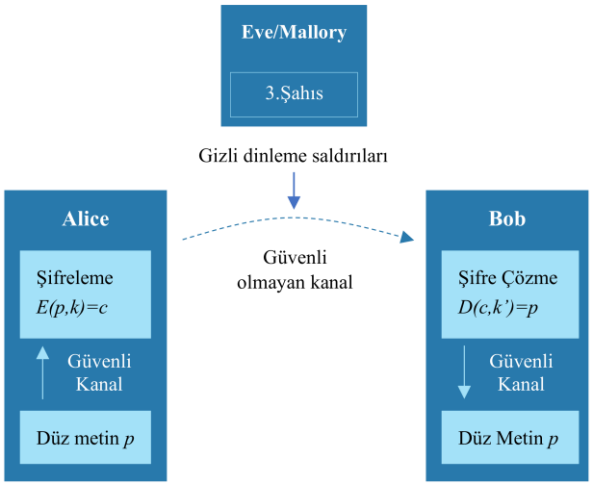
Güvenliği sağlamak için birçok teknoloji ve yöntem kullanılmaktadır. Bu teknolojilerin en güvenli olanlarından birisi kriptografidir (Anwar vd. 2019).

3.1 Kriptografi Nedir ?

Kriptografi, Roma döneminden günümüze kadar uzanan bir metin gizleme sanatıdır. Amacı bilgiyi güvende tutmaktır. Bunun için şifreleme ve şifre çözme işlevlerini kullanır. Şifreleme, basit bir mesajı şifreli metin adı verilen okunamaz bir forma dönüştürme işlemidir. Şifre çözme, şifreli metni tekrar orijinal metne dönüştürme işlemidir. Bu işlevlerin amacı, mesaj içeriğini yetkisiz kişilere karşı güvence altına almak ve mesaj içeriğine erişmelerini engellemektir (Maqsood vd. 2017).

Kriptografi bu amaçları yerine getirmek için bilgisayar bilimi ve matematiği kullanır. Şekil 3.1, üçüncü taraf saldırılarına karşı güvensiz bir kanal üzerinden güvenli bir mesaj iletiminin nasıl gerçekleşmesi gerektiğini göstermektedir. Alice ve Bob güvenli iletişim için mesaj içeriğini görme yetkisine sahiptir. Eve veya Mallory üçüncü taraflardır ve mesaj içeriğini görme yetkileri yoktur. Günümüzde kabul gören iki farklı şema vardır. Bunlar simetrik ve asimetrik şemalardır. Simetrik şemaya göre, Alice ve Bob mesajlarını şifrelemek için aynı anahtara sahip olmalıdır. Bu anahtar, iletişimden önce güvenli bir kanal üzerinden değiş tokuş edilmelidir. Asimetrik şema 1976 yılında Diffie Hellman'ın anahtar değişimi fikri ile ortaya çıkmıştır. Bu şemaya göre, Alice ve Bob'un bir özel ve bir açık anahtar olmak üzere iki anahtarı olmalıdır. Açık anahtar herkesle paylaşılabilir.

Bob, Alice'in gönderdiği mesajı açık anahtar ile şifreleyebilir. Ancak Alice, Bob'dan gelen mesajın şifresini sadece kendi özel anahtarı ile çözebilir (Barakat vd 2018).



Şekil 3.1 Kriptografi işleyiş blok diyagramı.

3.2 Temel Kriptografi İlkeleri

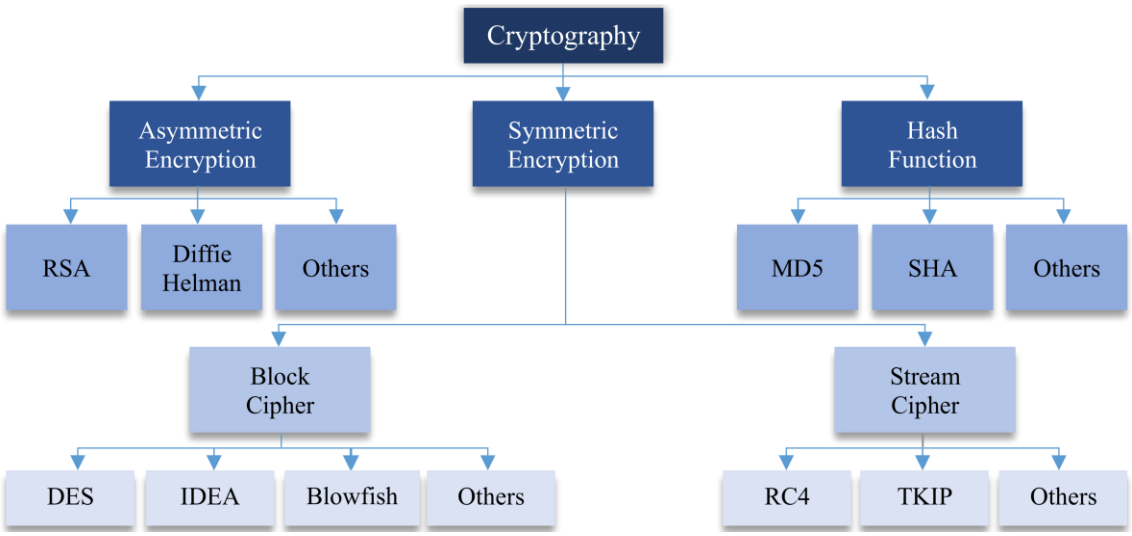
Kriptografi algoritmaları sadece veri gizliliğine çözüm bulmazlar. Temel kriptografi ilkeleri, bilgi güvenliği ve şifreleme konularındaki temel ilkeleri ifade eder. Bu ilkeler şunlardır.

- Kimlik Doğrulama: Bir anahtar kullanımı ile sadece yetkili kişilerin erişebileceği özel kaynaklar için kimlik sağlama işlemidir.

- Gizlilik: Kriptografinin ana amacıdır. Mesajın sadece anahtara sahip olan yetkili kişilere ulaşmasının garantisidir.

- Veri Bütünlüğü: Verilerin yalnızca verileri değiştirme erişimine sahip gruplar veya bireyler tarafından değiştirilebilmesini sağlama sürecidir. Bu durum sağlanarak, verilerin yaşam döngüsü boyunca tutarlılığı ve doğruluğu güvence altına alınacaktır.

- İnkar etmeme: Hem göndericinin hem de alıcının iletişim kurdukları mesajın kendileri tarafından iletildiğini ve teslim edildiğini teyit ettikleri süreçtir. Bu durumda gönderici, mesajı ilettiğini ve alıcının mesajı aldığını kabul etmekle yükümlüdür (Abood ve Guirguis 2018).



Şekil 3.2 Kriptografinin sınıflandırılması

Şifreleme algoritmaları, şifre çözme işlemini mümkün olduğunca zorlaştırmayı amaçlar. Şekil 3.2, kriptografinin sınıflandırılmasını göstermektedir (Alenezi vd. 2020). Sınıflandırmada 3 ana teknik bulunmaktadır. Bunlar simetrik, asimetrik ve hash kriptografik algoritmalara dayanmaktadır.

3.3 Simetrik Şifreleme Algoritmaları

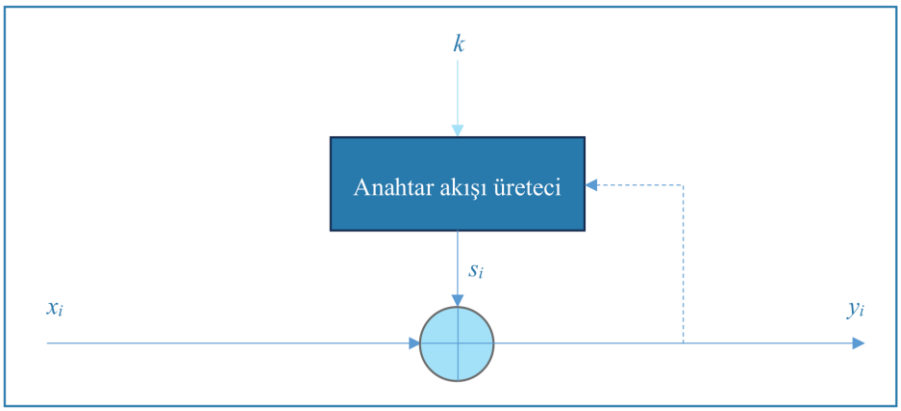
DES, RC4, Blowfish, ve AES gibi simetrik şifreleme algoritmaları en eski ve en basit olanlarıdır. Bu algoritmaların temel dezavantajı, gönderici ve alıcının hem şifreleme hem de şifre çözme için tek bir gizli anahtar kullanmasıdır. Bu dezavantajın nedeni, saldırganların iletişim kanalını dinleyebilmesi ve gizli anahtarı değiştirirken açık anahtarı elde etme olasılığının bulunmasıdır. Simetrik şifreleme, açık anahtarın değiş tokuş edilebilmesi için güvenli bir iletişim kanalı gerektirir. Ancak simetrik kriptografi

algoritmaları asimetrik kriptografi algoritmalarına göre hesaplama açısından daha verimlidir ve yaklaşık 1000 kat daha hızlıdır. Simetrik kriptografi algoritmaları aynı güvenlik seviyesini sağlarken daha küçük uzunlukta anahtar kullanır. Simetrik şifrelemede, iletişim kuran n kullanıcının toplam $n(n-1)/2n$ gizli anahtar değişimine ihtiyacı vardır. Bu, iletişim kuran her kullanıcının $n-1$ gizli anahtar saklamasını gerektirir ve birçok uygulama için pratik değildir. Asimetrik şifreleme algoritmalarında ise sadece bir gizli anahtarın saklanması yeterlidir (Domb 2019).

Simetrik kriptografi, blok ve akış şifreleme olmak üzere iki geniş kriptografi ailesini kapsar. Her ikisi de şifreleme şemalarında yaygın olarak kullanılmaktadır. Ancak, verileri şifreleme yöntemleri birbirinden oldukça farklıdır.

3.3.1 Akış şifreleme

Akış şifreleme algoritmalarında her bit ayrı ayrı şifrelenir. Senkron ve asenkron olmak üzere iki tür akış şifresi vardır. Senkron akış şifrelemede, anahtar akış üretecinin çıktısı ile düz metin verisi, şifreli metni üretmek için fonksiyonda işlenir. Şekil 3.3'te noktalı çizgilerle gösterilen çıktı, anahtar akışı üretecine girdi olarak geri verilir ve anahtar üretimine katılırsa, şifreleme türü asenkron olarak adlandırılır.



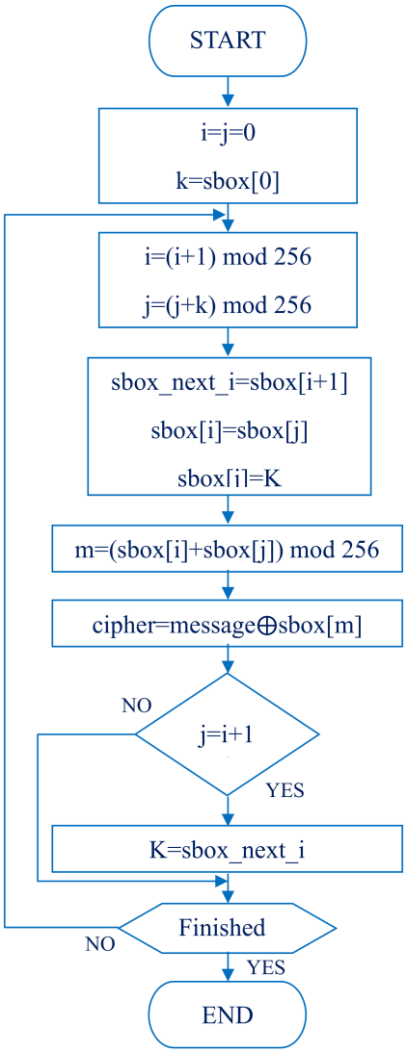
Şekil 3.3 Asenkron ve senkron akış şifreleme blok diyagramı

Günümüzde akış şifreleme algoritmaları uzun yıllar süren çalışmalar sonucunda daha güvenli hale gelmiştir. Bu algoritmalar bluetooth, mobil 4G, TLS vb. bağlantılarda kullanılmakta ve güvenli bir iletişim sağlamaktadır (Qadir ve Varol 2019). Akış şifreleme algoritmasına bir örnek RC4'tür. RC4, WEP ve TLS/SSL gibi farklı güvenlik protokollerinde yaygın olarak kullanılan bir akış şifreleme algoritmasıdır (Al-Badrei ve Alshawi 2021).

3.3.1.1 RC4

RC4 modern akış şifreleme algoritmaları arasında en etkili olanıdır. Algoritma 1994 yılında internet üzerinden genel olarak açıklanmış, tasarımındaki basitlik nedeniyle herkesin ilgisini çekmiş ve dünya çapında benimsenmiştir. Bu şifre, çeşitli yazılım ve web uygulamalarında geniş bir şekilde kullanılmaktadır. WEP (Kablosuz Eşdeğer Gizlilik), WPA (Wi-Fi Koruma Erişimi) ve SSL (Güvenli Soket Katmanı) gibi farklı ağ protokollerinde kullanılmaktadır. Ayrıca, Microsoft Windows, Apple OCE (Apple Açık İşbirliği Ortamı), güvenli SQL (veritabanı yönetimi ve veri depolama çözümü) gibi çeşitli uygulamalarda yaygın olarak kullanılmaktadır (Jindal ve Singh 2015).

RC4 algoritmasının çalışma prensibi aşağıda açıklanmıştır.



Şekil 3.4 RC4 şifreleme algoritması iş akış şeması

S-Box (Permütasyon Kutusu) Başlatma:

RC4 algoritması, 0'dan 255'e kadar olan sayıları içeren bir permütasyon kutusu olan S'yi başlatır. S[0] = 0, S[1] = 1, ..., S[255] = 255 şeklinde sıralı bir dizi oluşturulur. Ardından, genellikle 40 ila 256 bit uzunluğundaki bir anahtar kullanılarak S'nin başlangıç durumu belirlenir. Bu işlem, key-scheduling algorithm (anahtar zamanlama algoritması) olarak adlandırılır.

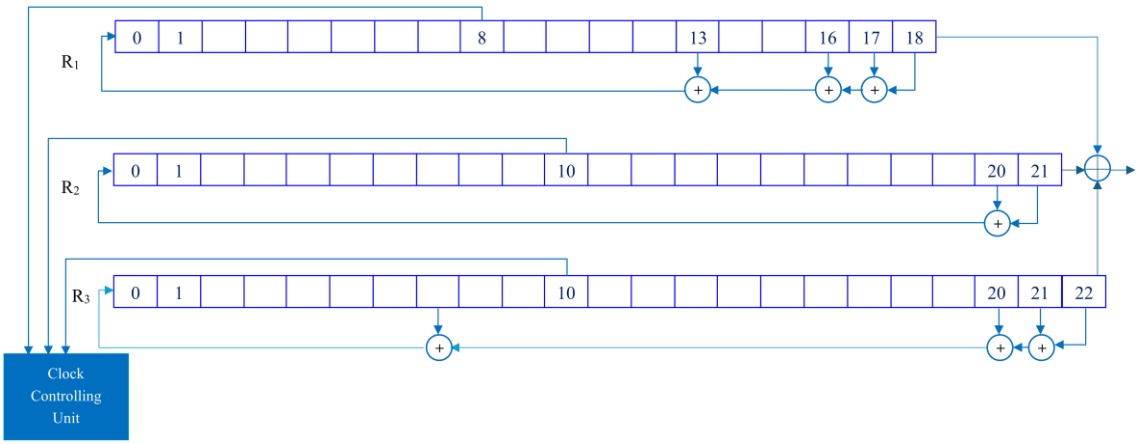
Anahtar Akışı Oluşturma (PRGA - Pseudo-Random Generation Algorithm):

Anahtar akışı (keystream) üretimi için PRGA kullanılır. Bir döngü içinde, S'nin elemanları arasında yer değiştirme (swap) işlemleri gerçekleştirilir. Bu işlemde i ve j adında iki sekiz bitlik işaretçi (pointer) kullanılır. S'deki elemanların yer değiştirmesi, i ve j'nin değerlerine bağlı olarak yapılır. Bu işlem, S'nin iç durumunu değiştirir ve anahtar akışı üretir.

Anahtar Akışı ile XOR İşlemi:

Anahtar akışı üretildikten sonra, bu akış kullanılarak şifreleme veya çözme işlemleri gerçekleştirilir. Düz metin (plaintext) ile anahtar akışı arasında XOR işlemi yapılır. Şifreli metin (ciphertext) elde edilir. RC4 şifreleme algoritmasının iş akış şeması Şekil 3.4'de verilmiştir (Sumartono vd. 2016).

3.3.1.2 A5/1, A5/2, A5/3 (GSM algoritmaları)



Şekil 3.5 A5/1 akış şifreleme algoritması

A5/1, GSM cep telefonlarındaki konuşmaların gizliliğini sağlamak için şu anda dünya çapında en çok ülke tarafından kullanılan akış şifreleme algoritmalarından biridir. A5/1, Şekil 3.5'de gösterildiği gibi çoğunluk saatleme yöntemiyle R1, R2 ve R3 adlı 3 kaydırma

kaydedicisinden oluşur. Kaydedicilerin başlatılması 64 bitlik KC ve 22 bitlik çerçeve numarası ile yapılır. Bunlar önce 3 kaydedicinin sol tarafına kaydırılır ve geri bildirimlerle XOR'lanır. Daha sonra A5/1, bitleri ilk karıştırmak için 100 döngü boyunca çoğunluk saatlemesi kullanılarak saatlenir. Daha sonra, A5/1'den gelen 114 bitlik çıktı şifrelemek/şifresini çözmek için düz metin ile XOR'lanır (Bakhtiari ve Maarof 2011).

3.3.1.3 Salsa20 algoritması

eSTREAM, bir dizi akış şifreleme algoritmasının değerlendirilmesi ve seçilmesi amacıyla düzenlenen bir yarışmadır. Salsa20, eSTREAM adayları arasında bulunan ve Daniel J. Bernstein tarafından önerilen bir eşzamanlı akış şifresidir. Bu şifreleme algoritması, basit aritmetik işlemleri (toplama, XOR, sabit mesafeli döndürme) temel alarak, çarpma veya s-box gibi karmaşık operasyonlardan kaçınma ilkesine dayanır. Bernstein, bu basit operasyonların kullanılmasının hem şifreleme sürecini hızlandırdığını hem de zamanlama saldırılarına karşı direnç sağladığını savunmuştur.

Salsa20'nin çekirdeği, 64 bayt giriş ve 64 bayt çıkışa sahip bir hash fonksiyonunu temsil eder. Bu hash fonksiyonu, sayaç modunda bir akış şifresi olarak kullanılır. Salsa20, 64 bayt uzunluğundaki bir düz metin bloğunu şifrelemek için anahtar, nonce (biricik sayı) ve blok numarasını kullanır. Bu bilgileri birleştirerek bir hash değeri elde eder ve ardından bu hash değerini düz metin bloğuyla XOR işlemine tabi tutarak şifreleme işlemini gerçekleştirir. Salsa20'nin bu tasarımı, güvenilir bir şifreleme mekanizması sunmak için matematiksel olarak sağlam bir temel oluşturur (Jolfaei ve Mirghadri 2010).

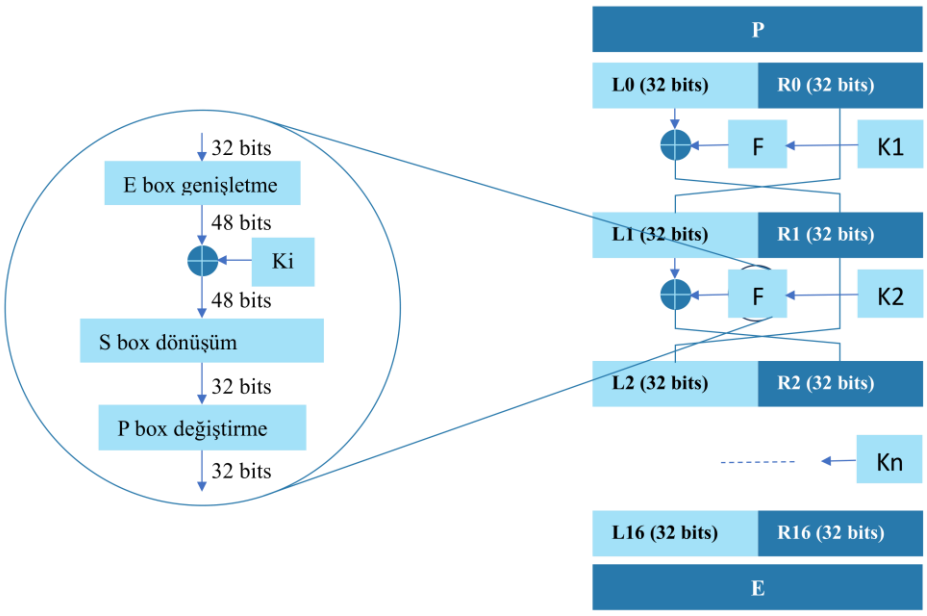
3.3.2 Blok şifreleme

Blok şifreleme, verileri n bitlik sabit uzunluklarda şifreleyen bir yöntemdir. Bloklar tipik olarak 64 bit, 128 bit ve 256 bit uzunluğundadır. Blok şifre algoritmaları, düz metin veri bloklarını sabit uzunlukta şifrelenmiş bloklara dönüştüren yinelemeli algoritmalardır. Blok şifreleme algoritmaları iç yapılarına göre iki gruba ayrılır: Substitution Permutation Networks (SPNs) ve Feistel networks (Shetty vd. 2020). Literatürde yer alan blok şifreleme algoritmaları aşağıdaki gibidir.

3.3.2.1 Data Encryption Standard (DES)

Veri Şifreleme Standardı (Data Encryption Standard-DES), 56 bit uzunluğunda anahtar kullanan blok simetrik şifreleme algoritmalarının tipik bir temsilcisidir. DES algoritmaları iki girdi alır. Bunlar düz metin ve anahtar verileridir. Düz metin ve anahtar uzunlukları 64 bittir. Ancak anahtarın 56 biti şifreleme fonksiyonlarında kullanılır. Kalan 8 bit ise eşlik bitleridir.

Şekil 3.6'da DES algoritmasının blok diyagramı gösterilmektedir.



Şekil 3.6 DES algoritması blok diyagramı

DES algoritması şifreleme işlemi aşağıdaki gibidir:

Adım 1: Düz metin 64 bitlik veri bloklarına bölünür. 64-bit veri bloğu P'nin veri bitleri, başlangıç permütasyon matrisi kullanılarak yeniden düzenlenir. Yeniden düzenleme, veri bitlerinin konumlarının permütasyon matrisi konumlarına göre değiştirilmesiyle gerçekleştirilir.

Adım 2: İlk permütasyon işlemi ile yeniden düzenlenen P bloğu, 16 turda iteratif dönüşümünü gerçekleştirmek için f fonksiyonunu kullanır. f fonksiyonunun iç yapısı aşağıdaki gibi çalışır. Her turda, verinin dönüşümü için farklı bir alt anahtar kullanılır. P bloğu her biri 32 bitlik iki alt bloğa bölünür. Bloklar sol ve sağ olarak adlandırılır. Sağ bloktaki 32 bitlik veri bir sonraki döngünün sol bloğu olur. 32 bit uzunluğundaki sol blok verisi E kutusu kullanılarak 48 bite genişletilir. Mevcut turun 48 bite indirgenmiş alt anahtarı ile xor işlemine giren genişletilmiş 48 bitlik veri çıkışı S kutuları ile dönüştürülerek tekrar 32 bite indirgenir. S kutusu çıkışı veri bitleri, P kutusu kullanılarak bu turdaki son dönüşümü tamamlar.

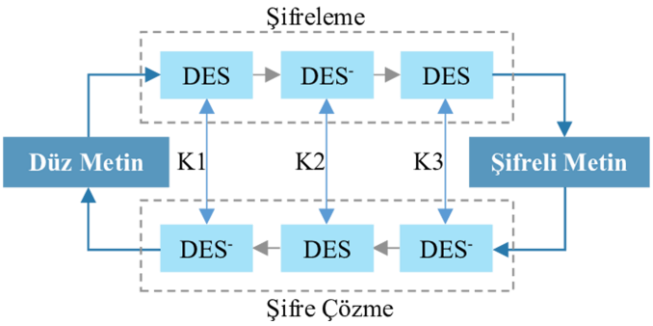
Adım 3: 16. Turun sonunda 64 bitlik bir şifreli metin çıktısı elde edilir.

DES, kısa anahtar uzunluğu nedeniyle güvenli kabul edilmez. DES şifreleme algoritmalarının yerini bir başka simetrik şifreleme algoritması olan Advanced Encryption Standard (AES) almıştır (Yihan ve Yongzhen 2021).

3.3.2.2 3-DES blok şifreleme algoritması

3-DES ilk olarak 1998 yılında yayınlanmıştır. 3-DES, DES şifreleme algoritmasını her veri bloğu için 3 kez kullandığı için bu şekilde adlandırılmıştır. Şekil 3.7'de, 3-DES algoritmasının blok diyagramı gösterilmiştir. Buna göre 3-DES K1, K2 ve K3 olmak üzere 3 anahtara ihtiyaç duyar. 3-DES şifreleme algoritmasında DES algoritması 3 kez kullanıldığı için DES algoritmasına göre daha güvenlidir. Ancak K1=K2 veya K2=K3

olursa 3-DES algoritması DES algoritması ile aynı olur. Bu durumdan kaçınmak için K1, K2 ve K3 arasındaki fark korunmalıdır (Wang ve Jiang 2019).

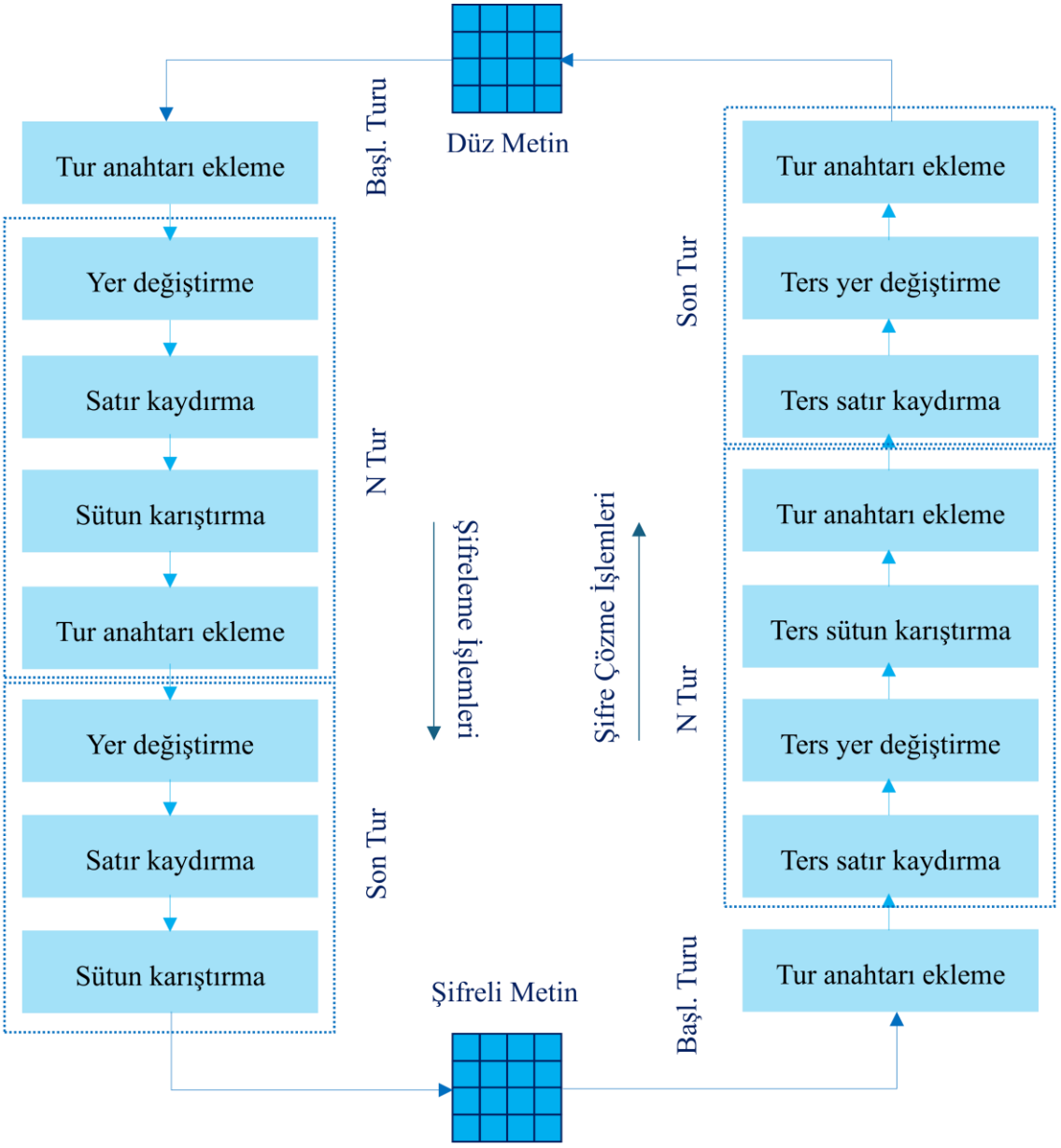


Şekil 3.7 3-DES blok şifreleme algoritması blok diyagramı

3.3.2.3 Advanced Encryption Standard (AES)

Sadece güvenli değil aynı zamanda çok hızlı olan AES algoritması hem donanımsal hem de yazılımsal olarak kullanılabilmektedir. NIST tarafından DES şifreleme algoritması yerine yeni şifreleme algoritması olarak kabul edilmiştir. Döngü sayısı kullanılan anahtarın uzunluğuna bağlıdır (Padmavathi ve Kumari 2013).

AES, 128, 192 ve 256 bit olmak üzere farklı anahtar uzunluklarıyla çalışabilen bir dizi anahtar turlarını kullanarak şifrelenmiş bir mesaj üretir. AES algoritması, 128-bit versiyonu için 4 x 4 boyutundaki bir durumu kullanır. Bu durum, şifrelenecek bilgiyi içeren bir matristir. Şifreleme ve şifre çözme aşamalarında her biri bir tur olarak kabul edilen adımlar eklenir. Toplamda 10 tur (128-bit için), 12 tur (192-bit için) veya 14 tur (256-bit için) yapılır. Bu tur sürecinde, her biri 4 temel dönüşüm sürecini içeren 9 tur üretilir. Bu dönüşümler yer değiştirme, satır kaydırmaları, sütun karıştırmaları ve tur anahtarı ekleme olmak üzere dört adettir.



Şekil 3.8 AES iş akış diyagramı

AES algoritmasının şifreleme ve şifre çözme süreci şematik olarak Şekil 3.8'te gösterilmiştir. Her adım, önceki adımların tersine kullanılarak gerçekleştirilir.

a. Yer değiştirme: Rijndael s-box olarak bilinen 8-bit yasası ikamesi kullanılır. Bu dönüşümde, 128-bit veri bloğu alınır ve her biri 16 bayt içeren bir veri matrisi ele alınır.

b. Satır kaydırma: Bu türde, veriler periyodik olarak kasanın geri kalan üç satırına göre değiştirilir. İkinci satır sola 1 dairesel kaydırılır ve üçüncü satırdaki iki bayt ve ardışık her biri üzerinde çalışılır.

c. Sütun karıştırmaları: Her sütunun çarpımı gerçekleştirilir ve bu baytlar çoklu isim olarak ele alınır.

d. Tur Anahtarı Ekleme Dönüşümü: Bu işlem, dairesel anahtarın 128-bit XOR'una benzer bir bit ve mevcut anahtarın 128-bitini kullanır (Salah vd. 2019).

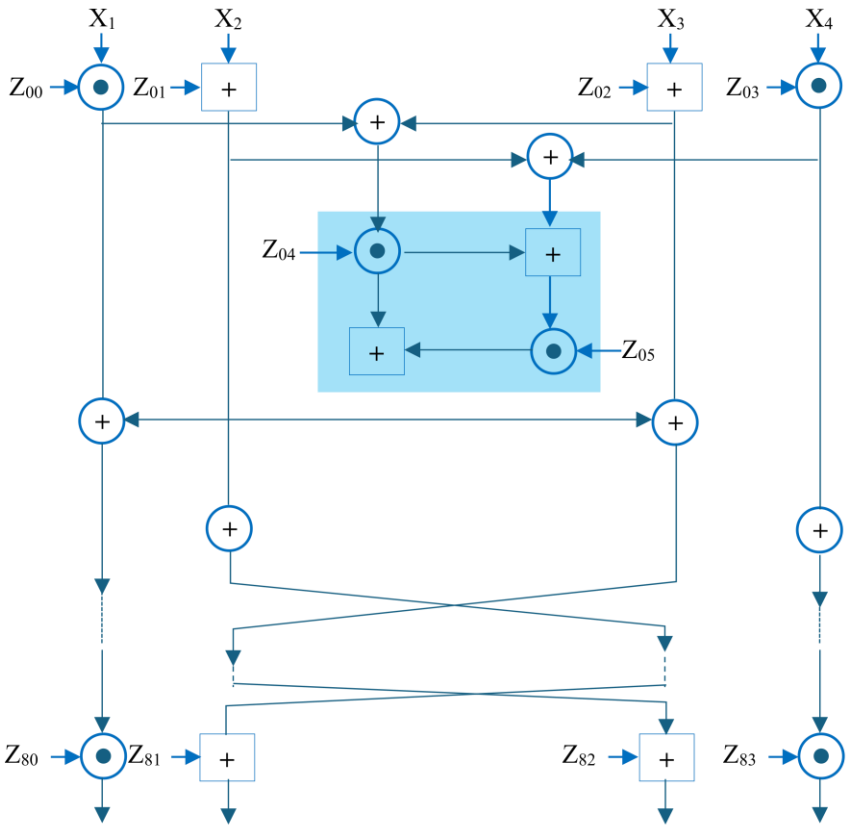
3.3.2.4 International Data Encryption Algorithm (IDEA)

64 bitlik veri giriş grubu, X1, X2, X3 ve X4 olmak üzere dört adet 16 bitlik alt gruba ayrılmıştır ve toplamda 8 tur bulunmaktadır. Her turda, dört alt grup, altı adet 16 bitlik alt anahtarla XOR işlemine tabi tutulur, ardından toplanır ve çarpılır. Tur geçişlerinde ise ikinci ve üçüncü alt bloklar yer değiştirir. Sekizinci turun tamamlanmasının ardından, dört alt blok toplanır ve çıkış dönüşümünde 52 alt anahtarla birleştirilir. Bu alt anahtarlar, alt anahtar üreteci tarafından üretilir. IDEA'nın genel yapısı Şekil 3.9'da gösterilmiştir (Yang vd. 2007). Bu şekilde kullanılan sembollerin anlamları Çizelge 3.1'de verilmiştir.

Çizelge 3.1 IDEA algoritması blok diyagramında kullanılan şekillerin anlamları

⊕	16 bitlik alt bloklar arasındaki XOR'lama
⊙	16 bit tamsayı mod $(2^{16}+1)$ çarpımını icra eder.
⊞	16 bit tamsayı mod 2^{16} toplama işlemini icra eder.

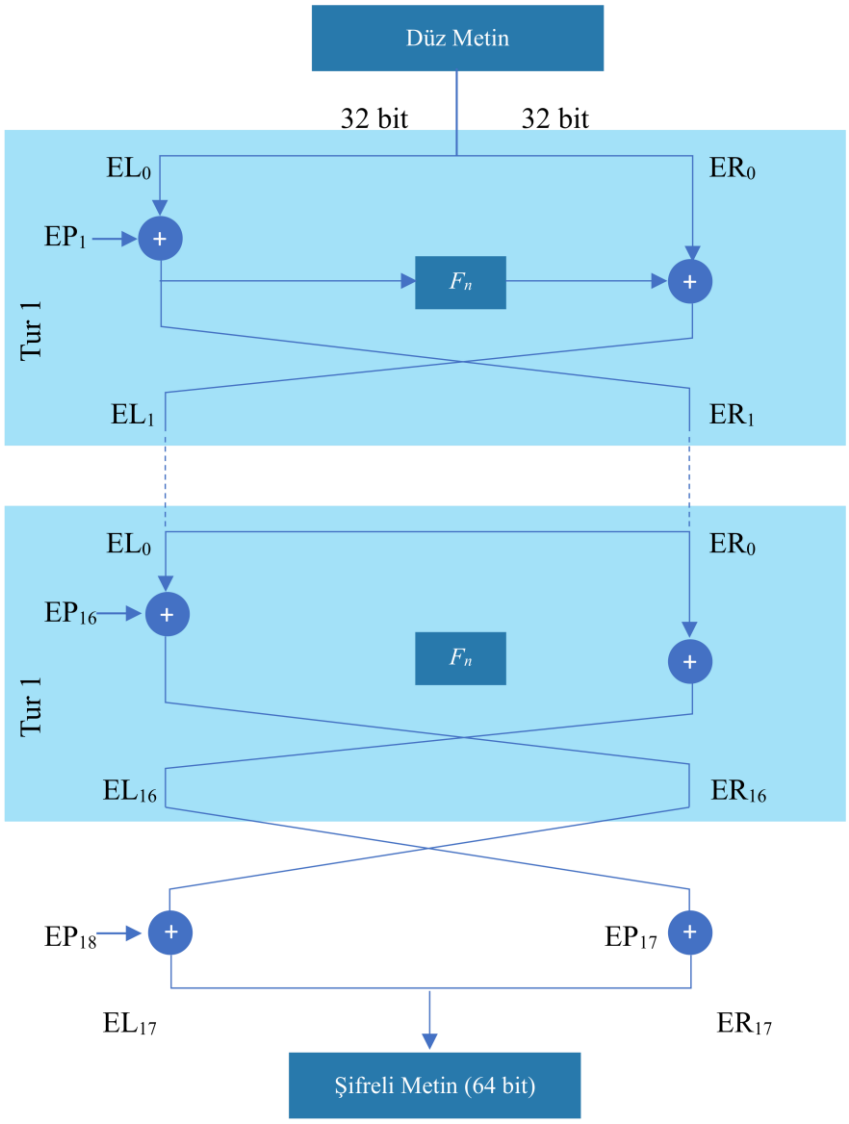
IDEA, güvenlik amacıyla yaygın olarak kullanılan şifreleme algoritmalarından birisi olmasına rağmen çok sayıda zayıf anahtara sahip olması IDEA algoritmasını dezavantajlı duruma getirmektedir. Buna ek olarak, IDEA'nın altıncı tur işlemleri sırasında bir saldırı tespit edilmiştir (Almasri ve Jani 2013).



Şekil 3.9 IDEA algoritması blok diyagramı

3.3.2.5 Blowfish

Blowfish simetrik şifreleme algoritması 1993 yılında Bruce Schneier tarafından DES ve IDEA algoritmalarının yerini almak üzere geliştirilmiştir. Blowfish algoritması, uzunluğu 32 bit ile 448 bit arasında değişen anahtarlar kullanır. DES algoritmasından daha hızlı şifreleme yapar (Sharma vd 2021). Blowfish gibi simetrik algoritma olduğu için şifreleme ve şifre çözme süreçlerinde aynı anahtarı kullanırlar. Bu durumda anahtar, mesajın göndericisi ve alıcısı dışındaki herkesten gizli tutulması gerekir.



Şekil 3.10 Blowfish şifreleme algoritması

Blowfish, hızlı, lisanssız, patentsiz ve serbestçe kullanılabilir bir şifreleme algoritmasıdır. Bu nedenle mevcut şifreleme algoritmalarına alternatif olabilir. Anahtar uzunluğu 32 ila 448 bit arasında değişebilir ve 64 bitlik bloklar kullanır. Blowfish algoritması, şifreleme işlemi için 16 tur kullanır.

Şekil 3.10'da gösterilen bir Feistel yapısına sahiptir. Bu algoritma, detaylı bir şekilde analiz edilmiş ve zaman içinde güvenilir bir blok şifre olarak popülerlik kazanmıştır. Diğer şifreler gibi, Blowfish algoritması VLSI donanımında etkili bir şekilde

kullanılabilir ve yazılım uygulamasında optimize edilebilir. Giriş verisi 64 bitlik bir bloğun E verisi olarak tanımlanmıştır.

Başlangıçta E verisi 32 bitlik iki parçaya ayrılır. 16 turluk bir süreçte EL ve ER değerleri aşağıda verilen algoritmaya göre işleme girer. Burada EP_i 18 adet 32 bitlik bir anahtar dizisidir. Her turun ayrı bir EP anahtarı vardır. Her turda, tura ait EP anahtarı E verisinin sol tarafındaki 32 bitlik veri ile XOR işlemine girer. XOR çıkışı EL'nin yeni değeridir. XOR işleminin çıktısında elde edilen yeni 32 bitlik veri (EL) Denklem (3.1) fonksiyonuna girdi olur. $F_n(EL)$ fonksiyonu çıktısı ile ER, XOR işlemine girer ve ER verisi yeniden elde edilir.

For i = 1 to 16:

EL = EL XOR EP_i

ER = F_n *(EL)* XOR ER

Swap EL ve ER

Next i

$$F_n(EL) = (S1, w + S2, x \ mod \ 2^{32}) \ XOR \ S3, y) + S4, z \ mod \ 2^{32} \qquad (3.1)$$

Denklem (3.1) ile verilen w, x, y ve z verileri 32 bitlik EL verisinin 8 bitlik parçalarıdır. 16. Tur sona erdikten sonra 17. Ve 18. EP anahtarlarının işlemleri aşağıdaki gibidir.

Swap EL and ER

ER = ER XOR EP17

EL = EL XOR EP18

Recombine EL and ER

Blowfish şifre çözme süreci EP1, EP2, ..., EP18'in ters sırada kullanılması dışında şifreleme işlemine benzer bir yaklaşımla ilerler. Blowfish algoritması dört adet S-kutusu kullanır. Bu tasarım, farklı baytlar arasındaki benzerliği engelleyerek güvenliği artırır. Blowfish algoritması her işlemde bir S-kutusu kullanır. Bu kullanım, her bir çıktının

önemsiz olmayan bir permütasyonu ile dört farklı çıktı üretilmesini sağlar. Dört S-kutusunun tasarımı, daha güvenli, daha hızlı ve programlaması kolaydır. Dört S-kutusunun çıktılarını birleştiren fonksiyon, dört çıktı değerini mod 2^32'nin karışık toplaması ile XOR yapacak kadar hızlıdır. Her turda toplama işleminin tekrarlanması ve tüm XOR işlemleri, nihai sonuç ER'ye XOR ile birleştirildiği için bir toplama ile sona erer (Mushtaq vd. 2017).

3.3.2.6 Twofish

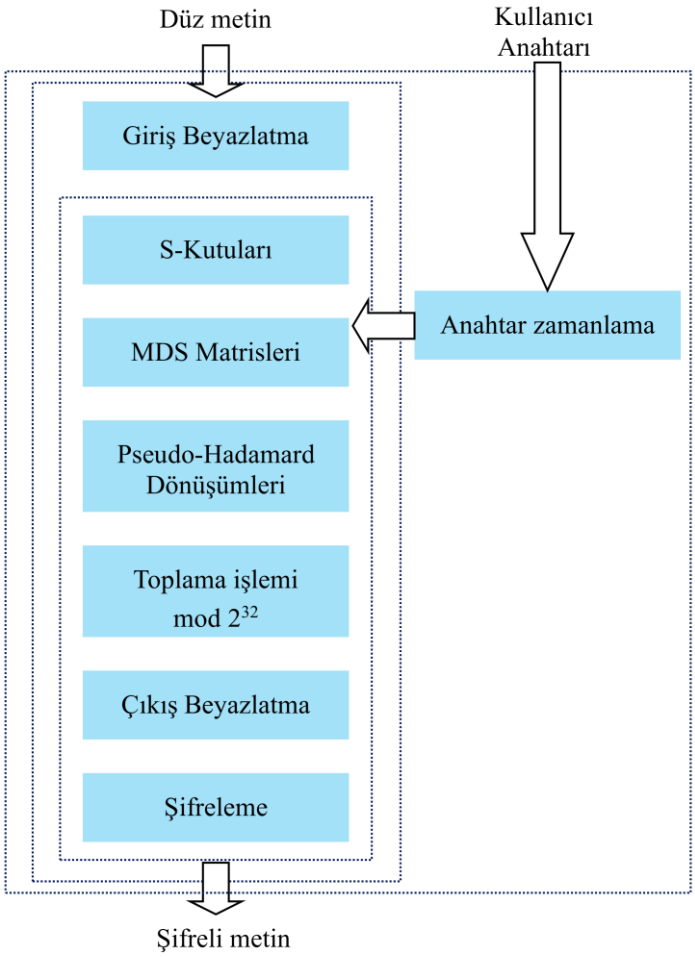
Twofish blok şifreleme algoritması Bruce Schneier tarafından, 1998 yılında tasarlanmıştır. Bu algoritma, "National Institute of Standards and Technology" (NIST) tarafından geliştirilen AES mimari standardının alternatifi olarak düşünülmüştür. Twofish, giriş mesajını 128-bit, 192-bit veya 256-bit anahtar kullanarak şifreler.

Twofish, sağlam anahtar yapısı ve uyarlanabilir tasarımı nedeniyle popüler bir tercih haline gelmiştir. Hem yazılım hem de donanım üzerinde hızlı ve etkili çalışabilen, çeşitli platformlarla uyumlu bir algoritmadır.

Twofish'in temeli, blowfish algoritmasına dayanmaktadır ve üçüncü tarafların algoritmayı ihlal etmesini zorlaştırmak amacıyla karmaşık matematiksel süreçler ve anahtar organizasyon tasarımlarını içermektedir. Algoritmanın tasarımı, Claude Shannon'ın 1949'da yayınlanan simetrik şifre oluşturma prensiplerine dayanmaktadır. Shannon, güvenli ve etkili şifreler oluşturmak için çoklu katmanlı bir "karıştırma dönüşümü" önermiştir. Bu dönüşüm, anahtar ve şifreli metin arasındaki ilişkiyi karmaşıklaştırmak için yer değiştirme tekniklerini kullanarak düz metin alımını kolaylaştırmak amacıyla kullanılan bir tekniktir. Difüzyon, şifreli metin ile düz metin arasındaki karmaşık istatistiksel bağlantıyı belirlemek için permütasyon sürecini içerir.

Twofish, Feistel yapısı üzerine kurulmuş bir algoritmadır. Bu yapı, bit bazında bağlantılı konum permütasyonları veya transpozisyonları yoluyla tanıtılan küçük yer değiştirmeler (S-kutuları olarak adlandırılır) koleksiyonundan oluşan bir ağ yapısını içerir. Ayrıca,

değiştirme-permutasyon ağları (SPN'ler) gibi, AES gibi diğer simetrik anahtar şifrelerini oluşturmak için kullanılan bir tür ağ yapısıdır. SPN'nin temel bileşenleri arasında difüzyon matrisleri, S-kutuları ve anahtar çizelgeleri yer alır. Difüzyon matrisleri, başlangıç noktası olarak Maksimum Mesafe Ayrılabilir (MDS) matrisi kullanılarak oluşturulur ve Twofish gibi Feistel Şifrelerinde de kullanılmaktadır.
Şekil 3.11'de Twofish algoritması aşamaları verilmiştir (Saieed ve Hattab 2023).



Şekil 3.11 Twofish algoritması aşamaları

3.4 Asimetrik Şifreleme Algoritmaları

Simetrik şifrelemenin aksine, DSA, RSA ve ECC gibi asimetrik şifreleme algoritmaları iki anahtar kullanır. Bunlar açık anahtar ve özel anahtardır. Açık anahtarın gizli tutulması

gerekmez. Herkes bir mesajı şifrelemek için açık anahtarı kullanabilir. Ancak şifrelenmiş mesajın şifresini çözmek için kullanılan özel anahtar gizli tutulmalıdır (Al-Shabi 2019).

Şifreleme işleminde, düz metin ve şifreli metin tam sayı olarak kodlanmalıdır. Benzer şekilde, şifre çözme işleminde de tamsayı bir mesaja dönüştürülmelidir.
Denklem (3.2) ve Denklem 3.3'de gösterilen matematiksel fonksiyonlar şifreleme ve şifre çözme için bu tamsayılara uygulanır.

$$\text{Şifreli metin, } C = f(\text{açık anahtar, } P) \tag{3.2}$$

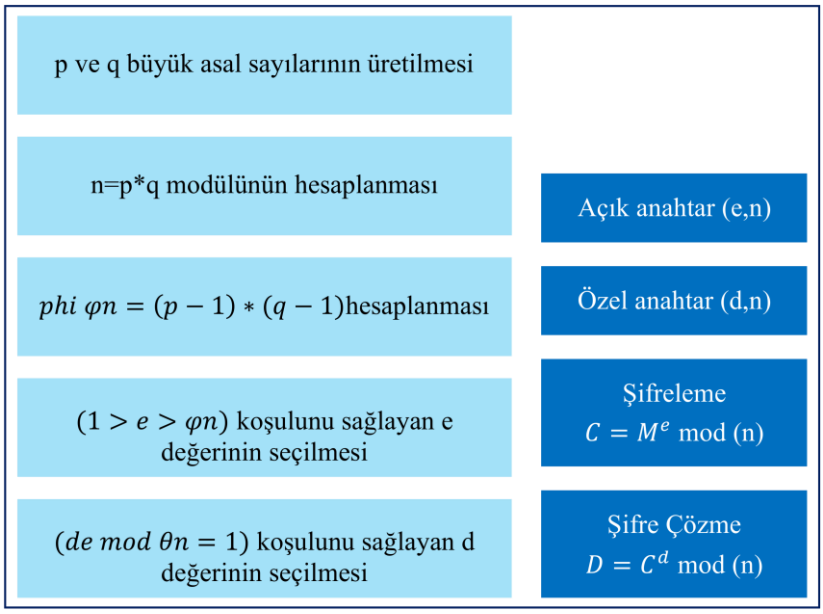
$$\text{Düz metin, } P = g(\text{özel anahtar, } C) \tag{3.3}$$

Şifreleme için f fonksiyonu ve şifre çözme için g fonksiyonu kullanılır. f fonksiyonu saldırganların şifreyi çözmesini engelleyen ve alıcının mesajı güvenli bir şekilde almasını sağlayan tek yönlü bir fonksiyondur (Sabitha ve Nair 2020).

Literatürde yer alan asimetrik şifreleme algoritmaları aşağıdaki gibidir.

3.4.1 Rivest-Shamir-Adleman (RSA)

En iyi bilinen asimetrik kriptosistem algoritmalarından biri olan RSA, 1978 yılında Ron Rivest, Adi Shamir ve Leonard Adleman tarafından tasarlanmıştır. RSA anahtar değişimi, dijital imzalama ve blok verilerin şifrelenmesi için kullanılır. Sayı teorisine dayanan RSA'nın şifreleme bloğu ve anahtarı farklı boyutlarda olabilir. RSA algoritmasının güçlü bir kriptografik sistem olarak gücü, seçilen p ve q sayılarına bağlıdır. Açık anahtar üretiminde kullanılan p ve q sayıları küçükse, şifreleme işleminin güvenliği zayıflar. Rastgele olasılık teorisi ve yan kanal saldırıları ile deşifre edilebilir. Diğer taraftan p ve q sayıları büyük seçilirse çok zaman harcar ve performansı oldukça düşer (Sood ve Kaur 2023).
RSA algoritması anahtar seçimi, mesaj şifreleme ve mesaj çözme iş akışı Şekil 3.12'de gösterilmiştir (Intila vd. 2019).



Şekil 3.12 RSA algoritması blok diyagramı

RSA algoritması anahtar seçimi, mesaj şifreleme ve mesaj çözme süreci ile ilgili bir örnek şu şekilde gerçekleşmektedir (Mallouli vd. 2019).

RSA anahtar üretimi

Adım 1. İki küçük asal sayı seçme:
p = 5 ve q = 7

Adım 2. n hesaplama: $n = p.q = 5.7 \rightarrow n = 35$

Adım 3. $\varphi(35) = (p-1)(q-1) = (5-1)(7-1) = 24$

Adım 4. Açık anahtar (e, n) seçme:
e, 1 ile $\varphi(n)$ arasında bir sayı olmalı ve $\varphi(n)$ ile aralarında asal olmalıdır. Bu örnekte, e 5 olarak seçilmiştir. Bu durumda açık anahtar : (e=5, n=35) olmuştur.

Adım 5. Özel anahtar (d,n) hesaplama
$d.e \equiv 1 \left(mod\ \varphi(n)\right) = d.5 \equiv 1\ (mod\ 24)$ olmalıdır. Bu denklemi sağlayacak d = 29 olmalıdır. Özel anahtar : (d = 29, n = 35)

Yukarıda verilen örnekte p = 5 ve q = 7 asal sayılarına göre açık anahtar (e=5, n=35) değer çifti ve özel anahtar (d = 29, n = 35) değer çifti olarak bulunmuştur. Bu anahtar

çiftlerine göre C=23 mesaj olarak seçildiğinde şifreli mesaj hesaplama süreci şu şekilde işler.

$$C \equiv 23^5 \ (mod\ 35)$$
$$C \equiv 6436343 (mod\ 35)$$
$$C \equiv 18 \ (mod\ 35)$$

C = 23 mesajı RSA blok şifreleme algoritmasına göre şifrelendiğinde 18 olarak hesaplanmıştır. Şifreli mesaj 18 değeri şifre çözme sürecinde şu şekilde çözülmektedir.

Özel anahtar : (29,35)
$$M \equiv C^d \ (mod\ n)$$
$$M \equiv 18^{29} \ (mod\ 35)$$
$$M \equiv 23 \ (mod\ 35)$$

3.4.2 Diffie-Hellman algoritması

Diffie-Helman algoritması ilk asimetrik şifreleme algoritmasıdır. Whitfield Diffie ve Martin Hellman tarafından 1976 yılında tasarlanmıştır. DH algoritması olarak ta bilinir. DH algoritması genellikle güvensiz iletişim ağlarında mesaj şifreleme ve şifre çözme için anahtar değişimi uygulamalarında kullanılır. Diffie-Hellman şifreleme algoritmasının güvenliği, çözülmesi zor olduğu düşünülen ayrık logaritma problemine dayanmaktadır (Yousif 2021).

Diffie-Hellman Algoritmasında yer alan denklemler ve adımlar şu şekildedir (Kumar vd. 2016).

1. Alice ve Bob bir p asal sayısı ve bir g tabanı üzerinde anlaşırlar.

2. Alice gizli bir a sayısı seçer ve Denklem (3.4)'e göre hesapladığı açık anahtarı Bob'a gönderir.

3. Bob gizli bir b sayısı seçer ve Denklem (3.5)'e göre hesapladığı açık anahtarı Alice'e gönderir.

4. Alice denklem (3.6)'ya göre gizli anahtarı elde eder.

5. Bob denklem (3.7)'ye göre gizli anahtarı elde eder.

$$(g^a \ mod \ p) \tag{3.4}$$

$$(g^b \ mod \ p) \tag{3.5}$$

$$((g^b \ mod \ p)^a \ mod \ p) \tag{3.6}$$

$$((g^a \ mod \ p)^b \ mod \ p) \tag{3.7}$$

Diffie-Hellman Algoritması p=23 ve g=5 değerlerine göre bir örnek şu şekildedir.

1. p = 23 ve g = 5.

2. Alice a = 6 değerini seçtiğinde açık anahtar $(5^6 \ mod \ 23) = 8$ olarak hesaplanır. Bu değer Bob tarafından alınır.

3. Bob b = 15 değerini seçtiğinde açık anahtar $(5^{15} \ mod \ 23) = 19$ olarak hesaplanır. Bu değer Alice tarafından alınır.

4. Alice aldığı açık anahtarı gizli anahtarı elde etmek için kullanır. Alice gizli anahtarı şu şekilde hesaplar. Gizli anahtar $(19^6 \ mod \ 23) = 2$ olarak bulunur.

5. Bob aldığı açık anahtarı gizli anahtarı elde etmek için kullanır. Bob gizli anahtarı şu şekilde hesaplar. Gizli anahtar $(8^{15} \ mod \ 23) = 2$ olarak bulunur.

3.4.3 ElGamal algortiması

Elgamal asimetrik şifreleme algoritması, Diffie-Hellman anahtar dağıtım şemasına dayanmaktadır. Elgamal algoritmasının çalışması şu şekilde gerçekleşir. ElGamal algoritması denklemleri gösteriminde özel anahtar K ile sembolize edilmiştir. Alice'in gizli anahtarı x_a ve Bob'un gizli anahtarı x_b'dir. p büyük bir asal sayı ve α bilinen ilkel elemandır. Alice, açık anahtarı Denklem (3.8)'e göre hesaplar ve Bob'a gönderir. Bob açık anahtarı Denklem 3.9'a göre hesaplar ve Alice'e gönderir.

$$y_a \equiv \alpha^{x_a} \, mod \, p \tag{3.8}$$

$$y_b \equiv \alpha^{x_b} \, mod \, p \tag{3.9}$$

Sonuç olarak, gizli K, Denklem (3.10)'a göre şu şekilde hesaplanır.

$$\begin{aligned} K &\equiv \alpha^{x_a x_b} \, mod \, p \\ &\equiv y_{a^{x_b}} \, mod \, p \\ &\equiv y_{b^{x_a}} \, mod \, p \end{aligned} \tag{3.10}$$

Alice ve Bob için gizli anahtar K'yı hesaplayabilmek kolaydır. Ancak kötü niyetli üçüncü şahıslar için K'yı hesaplamak oldukça zordur. ElGamal algoritması, Diffie-Hellman ve ayrık logaritma teorisine bağlı olarak tasarlanmıştır (Shen vd. 2021).

3.4.4 Digital Signature Algorithm (DSA)

Dijital imza, bir mesajın, yazılımın veya dijital belgenin gerçekliğini ve bütünlüğünü doğrulamak için kullanılan matematiksel bir tekniktir. Amerikan Federal Bilgi İşleme Standartları (Federal Information Processing Standards-FIPS) kapsamında belirlenmiş bir dijital imza algoritması olan Dijital İmza Algoritması (DSA), güvenliği asal alanların çarpımsal gruplarında ve bu grupların alt gruplarında Ayrık Logaritma Problemi'nin (DLP) varsayılan çözülemezliğine dayanmaktadır.

Dijital İmza Algoritması; anahtar üretimi, imza üretimi ve imza doğrulama olmak üzere üç aşamadan oluşur (Kumarı ve Roy 2018).

Anahtar Üretimi:

p bir asal sayı ve L 64'ün bir katıdır. $2^{L-1} < p < 2^{L}$ ve $512 \leq L \leq 1024$ biçiminde tanımlanırlar. q = p-1'in asal bir bölenidir ve $2^{150} < q < 2^{160}$ aralığındadır. h, $1 < h <$ $(p-1 >$ aralığında bir tamsayıdır. Bu parametrelere göre; özel anahtar x : $0 < x < q$ aralığında rassal bir tamsayı olur. Açık anahtar Denklem (3.11)'de verilmiştir.

$$g = h^{(p-1)/q} \, mod \, p$$
$$y = g^{x} \, mod \, p \tag{3.11}$$

İmza Üretimi: *M* mesaj, *k* rassal bir tamsayı olmak üzere anahtar *(r,s)* değer çiftidir. *r* ve *s* değerleri Denklem 3.12 ile hesaplanmaktadır.

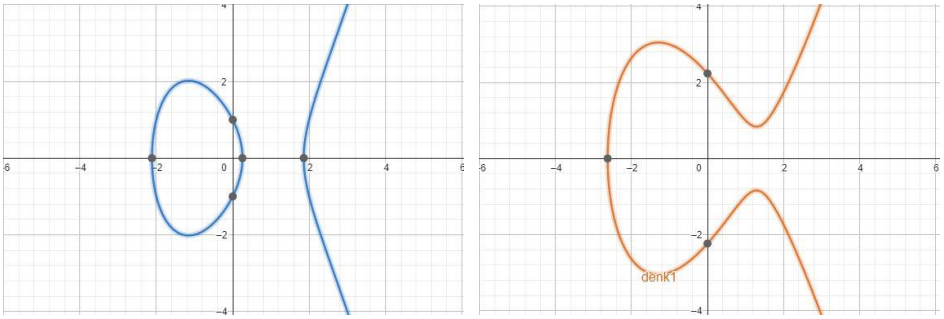
$$r = (g^{k} \, mod \, p) mod \, q$$
$$s = [k^{-1}(h(M) + xr)] \, mod \, q \tag{3.12}$$

İmza Doğrulama: *M', r', s'* sırasıyla *M, r, s* versiyonları olsun. Denklem (3.13)'e göre *v=r'* ise imza geçerlidir.

$$w = (s')^{-1} \, mod \, q$$
$$t_1 = [H(M')u] \, mod \, q$$
$$t_2 = (r'u) mod \, q$$
$$v = [(g^{t_1} y^{t_2}) mod \, p] \, mod \, q \tag{3.13}$$

3.4.5 Elliptic Curve Cryptography (ECC)

Şekil 3.13'de gösterilen ve sonlu bir alan üzerinde tanımlanan eliptik eğriler kriptografik şemaları uygulamak için bir grup yapısı sağlar. Grubun elemanları, "sonsuzdaki nokta" olarak nitelendirilen özel bir O noktası ile birlikte eliptik eğri üzerindeki rasyonel noktalardır.



Şekil 3.13 Eliptik eğri grafikleri sırasıyla $y^2 = x^3 - 4x + 1$ ve $y^2 = x^3 - 5x + 5$

Skaler nokta çarpımı tüm eliptik eğri kriptosistemlerinin ana yapı taşlarından birisidir ve k.P şeklinde bir işlemdir. Bu işlemde k pozitif bir tamsayı P ise eliptik eğri üzerinde bir noktadır. k.P işleminini sonucu eğri üzerinde başka bir Q noktasıdır. Bu işlem bir başka ifadeyle P noktasının kendisine tam olarak k - 1 kez eklenmesi anlamına gelir. Ters işlem ile yani k değeri elde edilir. Bunun için P ve Q = k.P noktaları bilinmelidir. Eliptik Eğri aynı zamanda Ayrık Logaritma Problemi (ECDLP) olarak nitelendirilir. Bugüne kadar, amacına uygun olarak seçilmiş bir eliptik eğri grubunda ECDLP'yi çözmek için üstel zamanın altında bir algoritma bilinmemektedir. Bu durum Eliptik Eğri Kriptografisini, günümüzde kullanılan diğer "geleneksel" DLP tabanlı şemalara benzer güvenliği, daha küçük anahtar boyutları ve bellek gereksinimleriyle sağlar. Bu durum Çizelge 3.2'de gösterilmiştir. Buna göre 1024 bit yerine 160 bitle güvenlik sunmak ECC algoritmasını açık anahtarlı kriptografinin umut verici bir dalı haline getirmektedir (Amara ve Siad 2011).

Çizelge 3.2 Açık anahtarlı ve simetrik anahtarlı kriptografi için anahtar uzunluğu

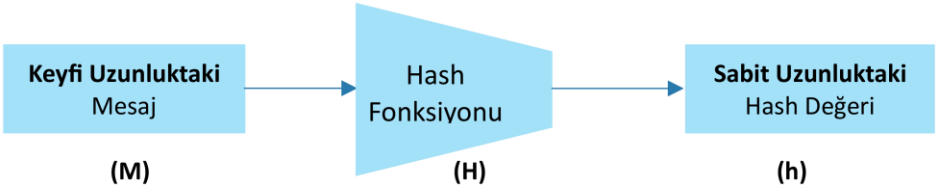
Symetric-key	ECC	RSA/DLP
64 bit	128 bit	700 bit
80 bit	160 bit	1024 bit
128 bit	256 bit	2048-3072 bit

3.4.6 McEliece algoritması

McEliece kriptosistemi, Robert McEliece tarafından 1978 yılında geliştirilen bir asimetrik şifreleme algoritmasıdır ve şifreleme sürecinde rastgeleleştirme kullanan ilk şemadır. Algoritma, genel bir doğrusal kodu çözmenin zorluğuna dayanır. Özel anahtar, t hatayı düzeltebilen bir hata düzeltme kodu seçilerek tanımlanır. Orijinal algoritma ikili Goppa kodlarını kullanır ve bu kodlar, verimli bir şekilde çözülebilen Patterson'a ait bir algoritma ile işlenir. Açık anahtar, seçilen kodun genel bir doğrusal kod olarak gizlenmesiyle özel anahtardan türetilir. McEliece algoritması, bir açık ve bir özel anahtar üreten olasılıksal bir anahtar üretme algoritması, olasılıksal bir şifreleme algoritması ve bir deterministik şifre çözme algoritması olmak üzere üç ana algoritmadan oluşur (Mohammed ve Al Saffar 2021).

3.5 Hash Fonksiyonları

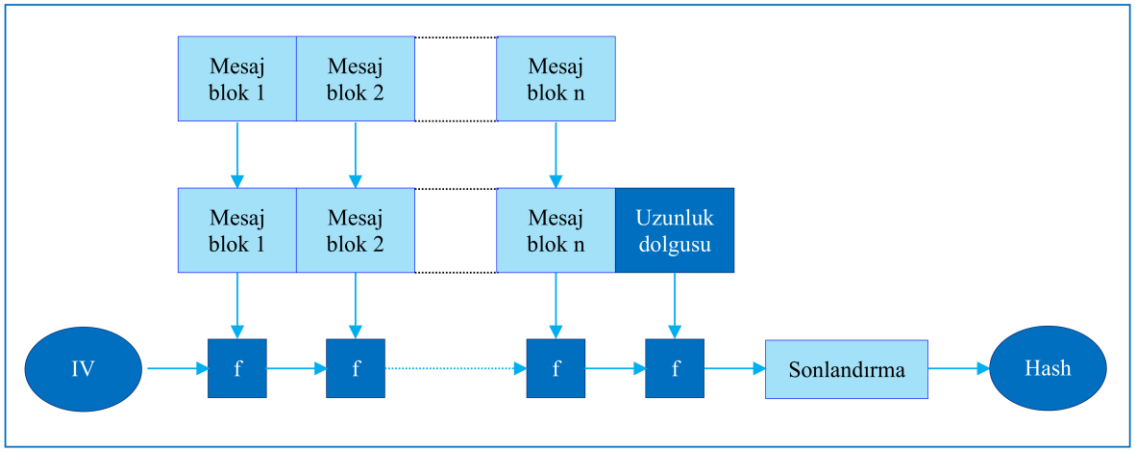
Hash fonksiyonları şifrelenmiş mesajı matematiksel işlemlerle sabit uzunlukta geri döndürülemez bir forma dönüştürür. Girdi olarak kullanılan mesaj, hash çıktıları kullanılarak tekrar elde edilemez (Babalola vd 2021). Bir kriptografik hash fonksiyonunun genel şeması Şekil 3.14'de gösterilmektedir (Anwar vd. 2021).



Şekil 3.14 Hash fonksiyonu blok diyagramı

Hash fonksiyonunun sabit uzunluktaki çıktısı, giriş verisinin parmak izi olarak görülebilir. Kriptografik hash fonksiyonları dijital imza, veri bütünlüğü, şifre koruması, rastgele sayı üretimi ve kimlik doğrulama protokolleri gibi birçok önemli uygulamada kullanılmaktadır. Kerberos kimlik doğrulama ve İnternet Protokol Güvenliği (IPSec) güvenli iletişim protokolleri hash fonksiyonlarını kullanır. Secure Socket Layer (SSL) protokolü için bir mesaj kimlik doğrulama kodu oluşturmak ve Pretty Good Privacy (PGP) ve Secure/Multipurpose Internet Mail Extensions (S/MIME) protokolleri için e-posta mesajlarının bütünlüğünü korumak için kullanılır. Birçok uygulamada kullanılan hash fonksiyonlarındaki bir hata, kullanıldıkları uygulamaları olumsuz etkileyecektir (Madhuravani ve Murthy 2013).

3.5.1 Merkel Damgard yapısı



Şekil 3.15 Merkle-Damgard Yapısı

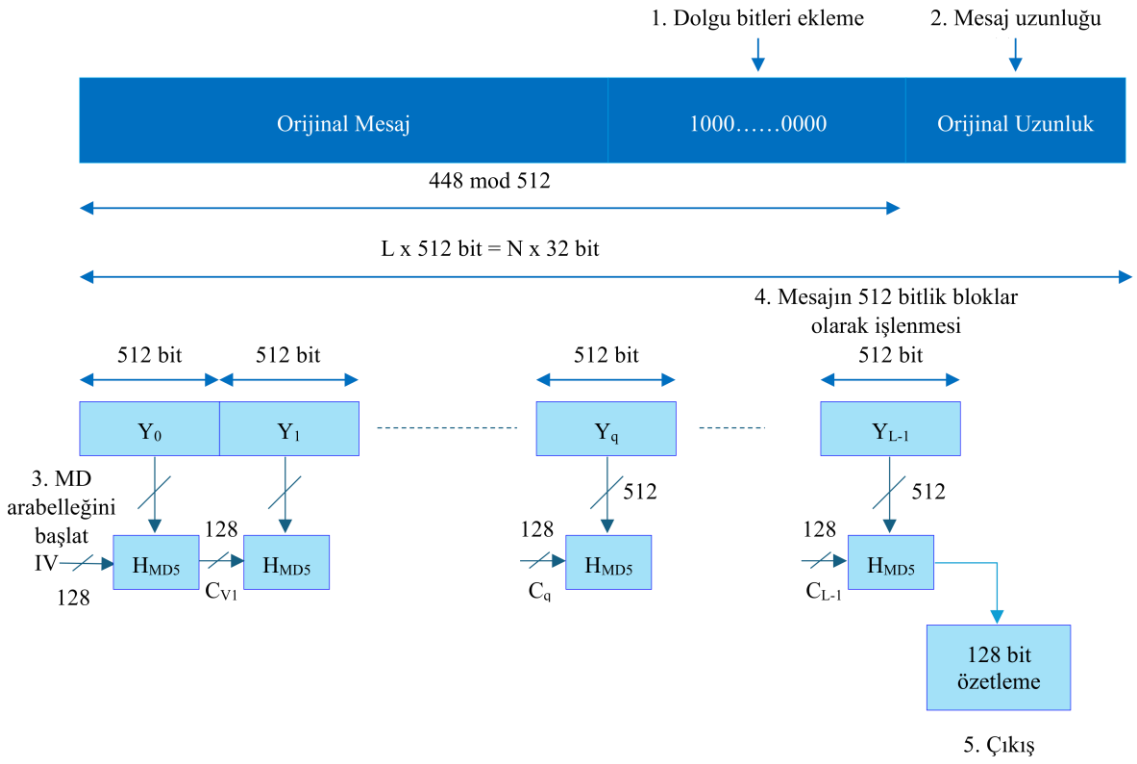
Merkle ve Damgård tarafından 1989 yılında tasarlanan Merkle Damgård yapısı, MD5, SHA-1, SHA-2 gibi yaygın olarak kullanılan hash fonksiyonları tarafından kullanılmaktadır. Yinelemeli bir sıkıştırma fonksiyonu kullanan Merkle Damgard yapısında mesaj sabit uzunlukta bloklara ayrılır. Bloklar f sıkıştırma fonksiyonu tarafından sırayla işlenir (Zellagui vd. 2019). Merkle Damgård yapısı başlangıç verisine (IV) ihtiyaç duyar. Başlangıç verisi ilk bloğu işlemek için kullanılır. Daha sonra her

bloğun çıktısı bir sonraki bloğu işler. Mesajın son bloğu işlendikten sonra nihai çıktı üretilir (Al-Odat ve Khan 2019).

Merkle Damgard yapısı Şekil 3.15'te gösterilmiştir (Chuah 2009).

3.5.2 Message Digest Algorithm 5 (MD5)

Message Digest 5 (MD5) algoritması 1992 yılında Profesör Ronald Rivest tarafından tasarlanmıştır. MD5 algoritmasının iş akışı Şekil 3.16'da gösterilmiştir (Landge ve Satopay 2018).



Şekil 3.16 MD5 Algoritması

128 bit uzunluğunda bir hash çıktısı üretir. MD5 yapısında mesajlar 512 bit uzunluğunda bloklara ayrılır. Her blok 32 bitlik 16 kelimeden oluşur. MD5 hash algoritmasında mesaj uzunluğu 448 mod 512 olarak doldurulur. Mesajın uzunluğu kalan 64 bite eklenir.

Algoritma yapısında A,B,C,D olarak adlandırılan ve her biri 32 bit uzunluğunda olan tampon alanlar bulunmaktadır. Tampon alanlara 16 kelimelik mesaj blokları girilir. Bu girdiler 4 tur boyunca işlenir. Çıktı yeni tampon değerleridir. Son tampon değer hash fonksiyonunun çıktısıdır (Debnath vd. 2017).

3.5.3 Secure Hash Algorithm-I (SHA-I) ve Secure Hash Algorithm-I (SHA-II)

SHA-1 hash algoritması 1995 yılında NIST tarafından tasarlanmıştır. Bu algoritmanın tasarlanmasındaki en büyük etken SHA-0 algoritması ile ilgili güvenlik endişeleridir. Ancak SHA-0 güvenlik açıkları NIST tarafından açıklanmamıştır. SHA-1, 160 bit uzunluğunda bir hash çıktısı verir. Bu çıktı, SHA-1 yapısındaki fonksiyonların 80 tur işlenmesinin sonucudur. SHA-1 algoritması Merkle Damgard yapısına dayanmaktadır.

SHA-2 hashing algoritmaları, farklı uzunluktaki kelimelerle çalışan SHA-256 ve SHA-512 varyantları olarak sınıflandırılır. SHA-256 algoritması 32 bit uzunluğundaki sözcükleri girdi olarak kabul ederken, SHA-512 algoritması 64 bit uzunluğundaki sözcükleri girdi olarak kabul eder. Başka farklılıklar da vardır. Bunlar: bazı sabit parametreler ve başlangıç değerleridir. SHA-2 ailesinde SHA-224, SHA-384, SHA 512/224 ve SHA 512/256 olmak üzere 4 versiyon daha bulunmaktadır. Bu hashing algoritmaları farklı başlangıç değerleri ve farklı çıktı uzunlukları ile birbirlerinden ayrılırlar. Ancak temel yapıları aynıdır. Bu bilgilere göre SHA-2 hash algoritma ailesinin SHA-256 ve SHA-512 algoritma yapılarına dayandığı söylenebilir (Martino ve Cilardo 2019).

SHA-2 ailesinde yaygın kullanım alanı bulan SHA 256 algoritmasının hesaplama süreci aşağıdaki gibi tanımlanmaktadır:

1. Mesaj doldurma:

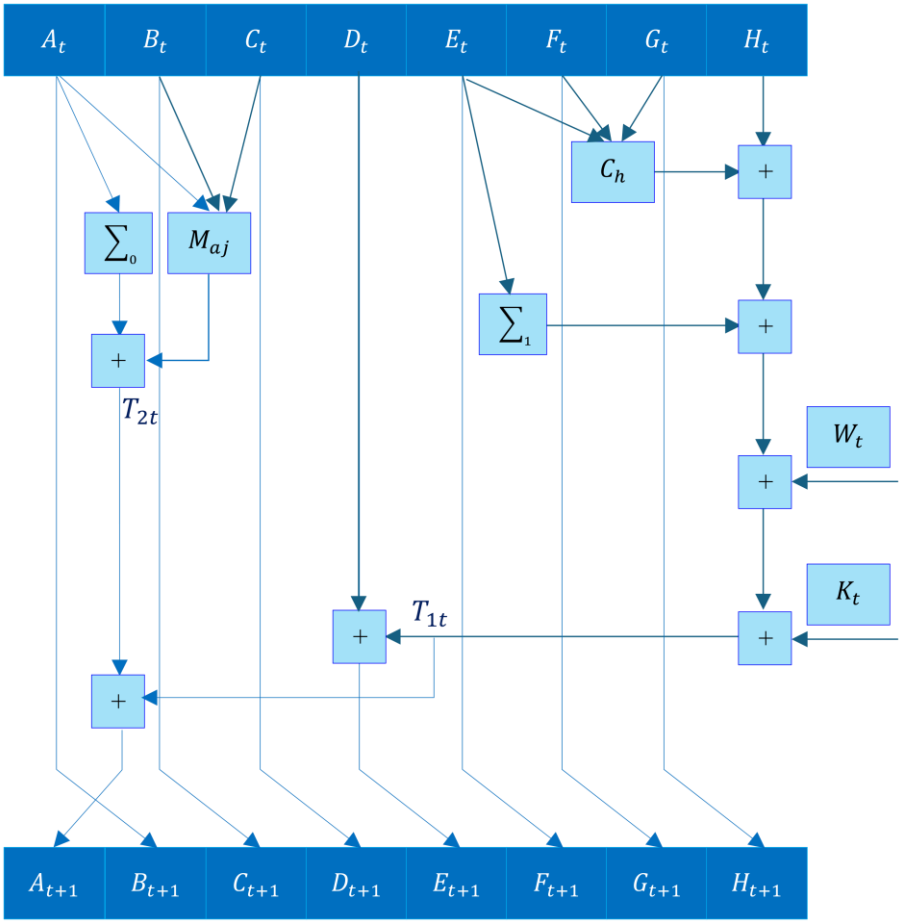
SHA-256 algoritması, mesajın bit uzunluğunu belirli bir formata getirir. Bu işlem, mesajın bit uzunluğunu 512'e bölündüğünde kalanın 448 olması için gereken '1' ve '0' bitlerini eklemeyi içerir. Eğer mesaj uzunluğu 448 bite ulaşmazsa geri kalan bitleri sadece ilki “1” diğerleri “0” ile doldurulur. Orijinal mesajın uzunluğu geri kalan 64 bitlik alana eklenir.

2. Değişken tanımlama

İlk bağlantı değişkeni değerleri, ara bağlantı değişkeni değerleri ve son hash değerleri, sekiz kayıtta A, B, C, D, E, F, G ve H olarak saklanır. Başlangıç bağlantı değişkenlerinin değerleri ise sırasıyla şu şekilde kabul edilebilir. H0 = 6a09e667, H1 = bb67ae85, H2 = 3c6ef372, H3 = a54ff53a, H4 = 510e527f, H5 = 9b05688c, H6 = 1f83d9ab, H7 = 5be0cd19. Bu veriler, SHA-256 algoritmasının içinde kullanılan başlangıç değerleri olabilir. Bu değerler, algoritmanın her adımında güncellenir ve hash işleminin güvenli ve etkili bir şekilde ilerlemesini sağlar. Başlangıçta sabit olan bu değerler, algoritmanın güvenlik özelliklerine katkıda bulunur.

3. Sıkıştırma fonksiyonunun çalışması

SHA-256 algoritmasının özetleme işlevi Şekil 3.17’de gösterilmiştir. Her bir mesaj bloğu üzerinde 64 tur döngüsel işlem gerçekleştirilir. Bu işlemlerde A, B, C, D, E, F, G, H kayıtlarının değerleri, her bir hesaplama turunun girdisi olarak kullanılır. Ayrıca, bu kayıtların mevcut değerleri, bir önceki turun çıktı bağlantı değişkeni değerlerinden türetilir. Bu süreç, SHA-256 algoritmasının güvenli ve etkili bir şekilde çalışmasını sağlamak üzere tasarlanmıştır.



Şekil 3.17 SHA-256 algoritması iş akışı

Her bir hesaplama turunda, 32 bitlik mesaj kelimesi W_t ve toplama sabiti K_t kullanılır. K_t , ilk 64 asal sayının küp kökünün ondalık kısmının ikili gösteriminin ilk 32 bitini içerir. Mesaj kelimesi W_t, şu şekilde gösterilir.

$$W_t = M_j[32.t + 31 : 32.t] \quad t < 16$$
$$W_t = \sigma_0(W_{t-2}) + W_{t-7} + \sigma_1(W_{t-15}) + W_{t-16} \quad t \geq 16 \qquad (3.14)$$

Denklem (3.14)'te, M_j j'inci mesaj bloğunu ve + 2^{32} modulo toplama işlemini gösterir. $\sigma_0(x)$ ve $\sigma_1(x)$ sırasıyla Denklem 3.15'de verilmiştir.

$$\sigma_0(x) = x \gg > r^7 \oplus x \gg > r^{18} \oplus x \ggg 3$$
$$\sigma_1(x) = x \gg > r^7 \oplus x \gg > r^{19} \oplus x \ggg 10 \qquad 3.15$$

Denklem (3.15)'de, $\ggg r^n$ döngüsel sağa kaydırma işlemini, $\ggg n$ sağa kaydırma işlemini ve $\oplus$ XOR işlemini temsil eder. Her döngüde aşağıdaki yineleme gerçekleştirilir.

$$A_{t+1} = T_{1t} + T_{2t} \quad B_{t+1} = A_t$$
$$C_{t+1} = B_t \quad D_{t+1} = C_t$$
$$E_{t+1} = D_t + T_{1t} \quad F_{t+1} = E_t$$
$$G_{t+1} = F_t \quad H_{t+1} = G_t \qquad (3.16)$$

Denklem 3.17'de verilen T_{1t} ve T_{2t} aşağıdaki gibi tanımlanmıştır.

$$T_{1t} = \sum\nolimits_1 (E_t) + Ch(E_t, F_t, G_t) + H_t + W_t + K_t$$
$$T_{2t} = \sum\nolimits_0 (A_t) + Maj(A_t, B_t, C_t) \qquad (3.17)$$

Denklem (3.18)'de, operasyonel fonksiyonlar $C_h, M_{aj}, \sum_0, \sum_1$ aşağıdaki gibi ifade edilir.

$$Ch(x, y, z) = (x \wedge z) \oplus (\neg x \wedge y)$$
$$Maj(x, y, z) = (x \wedge y) \oplus (x \wedge z) \oplus (y \wedge z)$$
$$\sum\nolimits_0 (x) = (x \ggg r^2) \oplus (x \ggg r^{13}) \oplus (x \ggg r^{22})$$
$$\sum\nolimits_1 (x) = (x \ggg r^6) \oplus (x \ggg r^{11}) \oplus (x \ggg r^{25}) \qquad (3.18)$$

4. Ara hash değerleri

Sıkıştırma döngüsünün 64 turunu tamamladıktan sonra, ara hash değerleri şu şekilde elde edilecektir. Kayıtların değerlerinin modüler toplama işlemi A, B,C, D, E, F, G, H ve (j-1) bloğunun hash değerleri sırasıyla Denklem (3.19)'da gösterilmiştir.

$$H_{0(j)} = A + H_{0(j-1)} \quad H_{1(j)} = B + H_{1(j-1)}$$
$$H_{2(j)} = C + H_{2(j-1)} \quad H_{3(j)} = D + H_{3(j-1)}$$
$$H_{4(j)} = E + H_{4(j-1)} \quad H_{5(j)} = F + H_{5(j-1)}$$
$$H_{6(j)} = G + H_{6(j-1)} \quad H_{7(j)} = A + H_{7(j-1)} \tag{3.19}$$

5. Son mesaj bloğu – Hash kodu

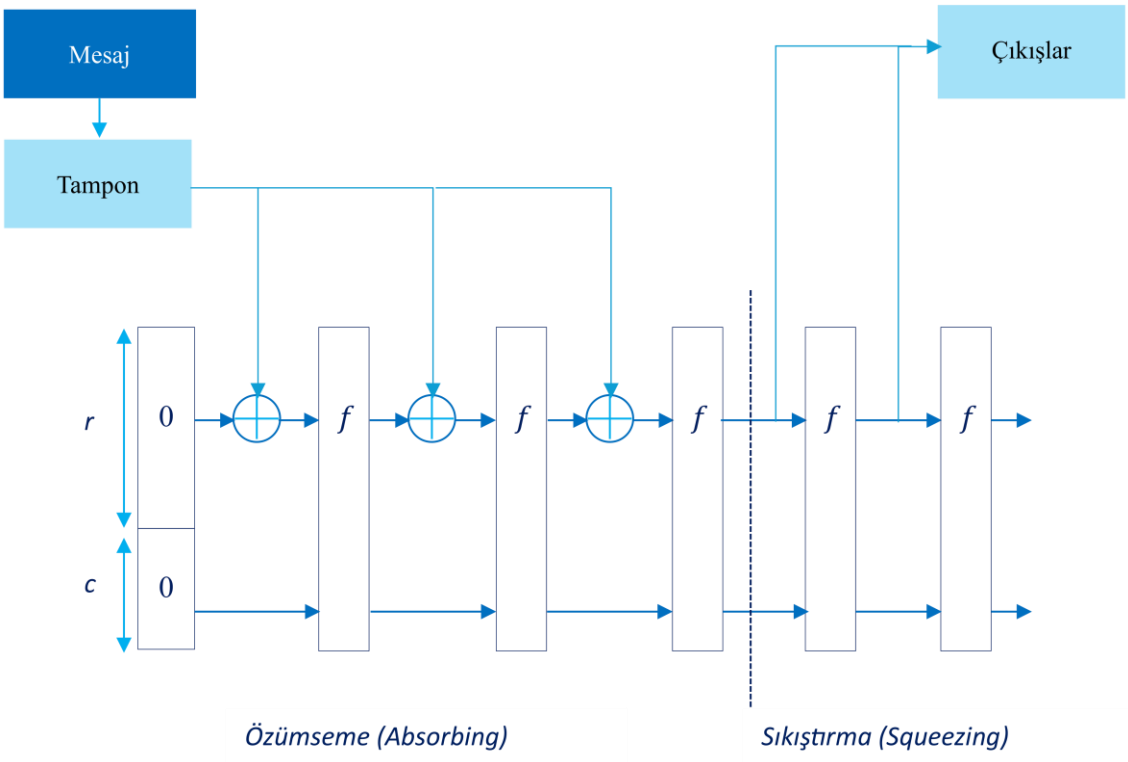
Son mesaj bloğunun iteratif işlemler ile elde edilen hash değeri SHA-256'nın çıkış sonucu olarak kullanılır. Çıkış, Denklem (3.20)'de gösterilmiştir (Wang vd. 2021).

$$Çıkış = H_0 \parallel H_1 \parallel H_2 \parallel H_3 \parallel H_4 \parallel H_5 \parallel H_6 \parallel H_7 \tag{3.20}$$

3.5.4 SHA-3 algoritması

SHA-3 hashing algoritması G. Bertoniet tarafından tasarlanmıştır. NIST, yeni bir yapı sunan bu algoritmayı 2015 yılında duyurmuştur. Keccak adı verilen bu algoritma, SHA ailesinin diğer üyelerinden farklı olarak Merkle Damgard yapısı yerine bir sünger yapısı kullanmaktadır. Bu yapı, "r" bit oranı ve "c" kapasite olmak üzere b=r+c bit durumuna dayanır. Yineleme başlamadan önce, mesajla birlikte durumun tüm b bitleri "0" bitleri eklenerek doldurulur ve r bit uzunluğunda bloklara bölünür. Bu işlemlerden sonra sırasıyla özümseme ve sıkıştırma aşamaları başlar. Özümseme aşamasında giriş, özet duruma "r" bit oranında emilir. Bunun için r bitlik veriye XOR ve blok permütasyon işlemleri uygulanır. Daha sonra aynı oranda bir hash çıktısı sıkıştırılır. Sıkıştırma işlemlerinde durumun ilk r biti çıktı olarak üretilir. Ayrıca ek çıktı istenirse blok permütasyon uygulanır (Sharma ve Mittal 2019).

Şekil 3.18'de sünger yapısı gösterilmektedir (Chen ve Ye 2022).



Şekil 3.18 Sünger (Sponge) yapısı

3.5.5 SHA hash algoritmaları karşılaştırılması

Bu bölümde, özetlenmiş mesajların uzunluğu ile hash fonksiyonlarının güvenlik parametreleri arasındaki ilişkiyi analiz etmenin sonuçlarını sunuyoruz. Bir kriptografik ilkenin güvenlik seviyesi bit cinsinden ifade edilir; bu bağlamda n-bit güvenlik, saldırganın bunu kırmak için 2 üzeri n işlem yapması gerektiği anlamına gelir. Bir kriptografik hash fonksiyonunun güvenlik seviyesi aşağıdaki özelliklerle tanımlanmıştır:

1. Pre-image saldırıları güvenlik bitleri,

2. İkinci ön görüntü direnci güvenlik bitleri,

3. Çarpışma direnci güvenlik bitleri,

Çarpışma direncini kaba kuvvet yöntemiyle aşmak için, saldırganın m mesajının bir dizi varyantını hash etmesi, ardından bu varyantları gözden geçirmesi ve eşit olan değerlere bakması gerekir. Örneğin, 160 bitlik bir hash çıktısında, saldırganın her iki listede de test etmek için 2 üzeri 80 girdiye ihtiyacı vardır. Bu nedenle, bu hash fonksiyonu için güvenlik bit sayısı, Doğum Günü Paradoksu nedeniyle 80'dir. Çizelge 3.3'te, NIST tarafından kabul edilen seçilmiş hash fonksiyonlarının güvenlik parametreleri sunulmuştur. Çizelge 3.3'te verilen *L(M)* fonksiyonu Denklem (3.21)'de şu şekilde tanımlanmıştır.

$$L(M) = \left\lceil \log_2 \frac{len(M)}{B} \right\rceil \tag{3.21}$$

Çizelge 3.3 NIST tarafından onaylanan hash fonksiyonları güvenlik parametreleri

Fonksiyon	**Çıkış büyüklüğü**	**Çarpışma**	**Ön-Görüntü**	**İkinci ön görüntü**
SHA-1	160	<80	160	160-$L(M)$
SHA-224	224	112	224	Min[224, 160-$L(M)$]
SHA-256	256	128	256	256- $L(M)$
SHA-384	384	192	384	384
SHA-512	512	256	512	512- $L(M)$
SHA3-224	224	112	224	224
SHA3-256	256	128	256	256
SHA3-384	384	192	384	384
SHA3-512	512	256	512	512

Burada M giriş mesajı, B hash fonksiyonunun blok boyutu ve ⌈.⌉ parantez içindeki argümandan kesinlikle daha küçük olmayan en küçük tamsayıyı ifade eder. Kaba kuvvet yöntemini kullanarak, her zaman sırasıyla $2^{n/2}$, 2^n ve 2^n adımdan oluşan genel bir saldırı stratejisi bulunmaktadır. Burada n hash uzunluğunu ifade eder. Bu, herhangi bir hash fonksiyonu için elde edilebilecek maksimum (ideal) güvenlik seviyesidir. Çizelge 3.3'de görülebileceği gibi, SHA-1 çarpışma saldırıları ve ikinci ön görüntü saldırıları açısından idealden daha düşük bir güvenlik seviyesine sahiptir. SHA-1, mesaj boyutu (bit

cinsinden) 160'a kadar olduğunda ikinci ön görüntü saldırıları açısından maksimum potansiyel gücü sunar (Tchórzewski ve Jakóbik 2019).

3.5.6 Whirlpool

Whirlpool, 2^{256} bit uzunluğundan daha kısa mesajlar üzerinde çalışan tek yönlü ve çarpışmaya dayanıklı bir 512 bitlik bir hash algoritmasıdır. Whirlpool, 512 bit anahtar kullanımına dayalı özel bir 512 bit blok şifresinin sıkıştırma işleminin yinelemeli bir uygulamasını içermektedir. Tur fonksiyonu ve anahtar programı, özel olarak tasarlanmış olan "Wide Trail" stratejisine göre yapılandırılmıştır. Bu strateji, çeşitli cihazlarda paralel işlemeyi kolaylaştırmak ve etkili bir performans elde etmek amacıyla geliştirilen kriptografik algoritmaların genişletilmiş bir stratejisidir.

Whirlpool uygulamaları, özellikle herhangi bir platforma bağlı olmayan bir fonksiyon yapısından faydalanarak 8-bit ve 64-bit işlemcilerde etkili bir şekilde çalışır. Bu, algoritmanın geniş bir kullanım yelpazesi için uygundur. Whirlpool, ilk olarak NESSIE projesi için sunulmuş olup, NESSIE projesi, Avrupa Birliği tarafından başlatılan ve yeni şifreleme standardı adaylarını değerlendirmeyi amaçlayan bir girişimdir. Whirlpool, iç yapısındaki eksiklik nedeniyle verimli donanım uygulamasını zorlaştıran rastgele oluşturulmuş bir ikame kutusu (S-kutusu) kullanır (Barreto ve Rijmen 2000).

3.5.7 Blake2

Aumasson, Henzen, Meier ve Phan tarafından tasarlanan BLAKE, beş SHA-3 finalistinden biridir. Farklılaştırılamazlık, bir hash fonksiyonunun güvenliği hakkında önemli bir kavramdır. Bu kavram, hash fonksiyonunun rastgele bir oracle gibi davranmaya ne kadar yakın olduğunu gösterir. "Oracle" terimi, bilgi sağlayan idealize bir varlığı temsil eder. Rastgele bir oracle ise her sorguya tamamen rastgele ve öngörülemez bir şekilde cevap veren bir ideal kaynaktır. Dolayısıyla, bir hash fonksiyonu ideal bir temel fonksiyonla benzer davranıyorsa, yapısal bir zayıflık olmadan saldırganın fonksiyonun iç yapısını anlamadan rastgele bir oracle gibi davranabilir.

Grøstl, JH, Keccak ve Skein dışındaki diğer dört SHA-3 adayı, farklılaştırılamaz güvenliklerini kanıtlamıştır. Bu durum, bu algoritmaların farklılaştırılamaz güvenlik sınırlarının zaten sağlam olduğunu gösterir. Ancak "sıkı" olarak ifade edilen bir kavram, bir hash fonksiyonunun bu güvenlik sınırını daha da iyileştirebileceği bir potansiyeli ifade eder. BLAKE algoritmasında, tasarımcılar, BLAKE'in sıkıştırma fonksiyonunun ideal olduğu varsayıldığında rastgele bir oracle'dan farksız olduğunu" belirtmişlerdir (Chang vd. 2011).

3.6 Hash Fonksiyonları Uygulama Alanları

Hash fonksiyonları, temel kriptografik algoritmalar arasında bulunmaktadır. Bu algoritmalar, iletişim, finans endüstrisi, gizli hesaplama gibi birçok alanda geniş bir uygulama alanına sahiptir (Yang vd. 2017). Hash fonksiyonları neredeyse tüm kriptografik protokollerin yanı sıra mesaj kimlik doğrulama kodları, veri bütünlüğü, şifre depolama ve rastgele sayı üretimi gibi birçok güvenlik uygulamasının temel bileşenleri olarak kabul edilir (Maetouq vd. 2018).

3.6.1 Veri bütünlüğü kontrolü

Veri bütünlüğü, bir dizi farklı hash algoritması kullanılarak sağlanabilir. Hash fonksiyonları, keyfi uzunluktaki bir girdiyi sabit bir çıktıya eşleyen ve tersine çevrilemez bir sıkıştırma fonksiyonu kullanan algoritmalardır. Bu özelliklerinden dolayı, veri bütünlüğünü güvence altına almak amacıyla kullanılır. Bu özellikleri, e-posta, dijital imzalar, elektronik oylama, e-ticaret ve çevrimiçi işlemler gibi uygulamalarda yaygın olarak kullanılmalarını sağlar. Hash fonksiyonları, simetrik ve asimetrik şifreleme gibi kriptografik ilkelerle karşılaştırıldığında, verimlilik açısından daha avantajlıdır (Sodhi ve Gaba 2018).

3.6.2 Dijital imzalar

Dijital imzalar, özgünlük, taklit edilemezlik ve inkar edilemezlik gibi güvenlik özellikleri sağlamak amacıyla kullanılan en temel kriptografik unsurlardan biridir (Srivastava vd.

2023). Dijital imza teknolojisi günümüzün e-ticaret ortamında çok önemlidir. Dijital imzaların güvenliğini sağlamak için, tek yönlü hash fonksiyonları son zamanlarda yaygın olarak kullanılmaktadır (Zhou vd. 2006).

Günümüzde yaygın olarak kullanılan çoğu dijital imza şeması, genellikle büyük tamsayıları çarpanlarına ayırmak için RSA ve ElGamal gibi ayrık logaritmaları hesaplamak üzerine kuruludur. Ancak, bu şemaların iki temel dezavantajı bulunmaktadır; kuantum bağışıklığına sahip değillerdir ve sınırlı hesaplama gücüne sahip küçük cihazlara uygundur. Tek yönlü hash fonksiyonlarına dayalı tek seferlik imza şemaları, bu iki soruna daha iyi bir çözüm sunar (Bunzel 2015).

3.6.3 Parola güvenliği

Bilgisayarlar, günlük faaliyetlerimizde yüksek hacimli kullanıcılara hizmet sağlamak üzere kullanılır. Tek faktörlü kimlik doğrulama, genellikle bir kullanıcı adı ve parola kombinasyonunu içerir ve web üzerinde kullanıcı kimliklerini doğrulamak için yaygın bir tercihtir. Ancak, zayıf parola yönetimi uygulamaları, saldırganlar tarafından istismar edilebilir. Bu istismarlar sonucunda, kullanıcıların kimlik bilgileri ifşa edilerek, hem kullanıcılara hem de hizmet sağlayıcılara zarar verilebilir (Hatzivasilis 2017). Bu yüzden parolalar, hash fonksiyonlarından geçirilerek özetler biçiminde veritabanlarında saklanırlar.

3.6.4 Veri tekilleştirme

Veri tekilleştirme teknolojisi, tekrar eden veri öğelerini tanımlamak, gereksiz tekrarları ortadan kaldırmak ve genel depolama veya aktarma ihtiyacını azaltmak amacıyla kullanılır. Tekrar eden veri öğelerini tespit etmek, bir dosya, blok veya biti, diğer bir dosya, blok veya bit ile karşılaştırarak gerçekleşir. Veri tekilleştirme teknolojisi, matematiksel hesaplamalar ve "hash" algoritmaları kullanarak her veri öğesi için benzersiz bir kod olan hash kimlik doğrulama numarası elde eder. Bu numaralar bir liste oluşturur ve genellikle hash indeksi olarak adlandırılır (He vd.2010).

3.6.5 HMAC (Hash-based Message Authentication Code)

HMAC standardı, güvenli olmayan bir iletişim kanalı üzerinden iletişim için özel bir mekanizma sunarak mesaj kimlik doğrulamasını sağlar. Bu mekanizma, MD5, SHA-1 gibi kriptografik özet fonksiyonlarını içerir. HMAC'in temel amacı, bir mesajın kaynağını ve bütünlüğünü doğrulamaktır. Bu doğrulama süreci için ana tasarım yaklaşımı, mesaj girdisi ve sadece mesajın kaynağı ile hedef alıcı tarafından bilinen gizli anahtarı içerir. HMAC, bu bilgilerin yoğunlaştırılmasıyla bir değer olan (MAC) Mesaj Kimlik Doğrulaması'nı üretir. Bu yöntem, iletişim güvenliği sağlamak amacıyla kullanılır ve anahtarın gizliliğini koruyarak mesajın doğruluğunu sağlar (Yiakoumis vd. 2006).

3.6.6 Blockchain ve kripto paralar

Blockchain, dağıtılmış veri depolama, uçtan uca iletim, mutabakat mekanizmaları, dijital şifreleme teknolojisi ve diğer bilgisayar teknolojilerini bir araya getiren yenilikçi bir uygulama modelidir. Bu model, merkezi olmayan, güvenli ve bilgi ifşasını engelleyen bir yapı sunar. Blok zincirinde, dijital şifreleme teknolojisi temel bir rol oynar. Kullanıcı bilgilerinin ve işlem verilerinin güvenliği, blok zincirinin etkili bir şekilde çalışabilmesi için önemli bir koşuldur.

Hash fonksiyonları genellikle veri bütünlüğünü sağlamak amacıyla bir başka ifade ile verinin yasa dışı olarak değiştirilmediğini doğrulamak için kullanılır. Test edilen veri değiştiğinde, hash değeri de buna bağlı olarak değişir. Bu nedenle, veri güvenli olmayan bir ortamda olsa bile, verinin bütünlüğü verinin hash değerine göre tespit edilebilir. Blok zinciri için hash fonksiyonları, blok ve işlem bütünlüğü doğrulaması yapmak için kullanılabilir. Blok zincirinde, bir önceki bloğun bilgilerinin hash değeri her bloğun başlığında saklanır ve herhangi bir kullanıcı hesaplanan hash değerini saklanan hash değeri ile karşılaştırabilir. Bu şekilde, önceki bloğun bilgilerinin bütünlüğü tespit edilir. Ayrıca, hash fonksiyonu genel-özel anahtar çiftleri oluşturmak için de kullanılabilir (Zhai vd. 2019).

3.6.7 Bulut veri güvenliği

Bulut tabanlı depolama, yerel kaynakların sınırlamaları olan durumlarda veri depolama sorununa etkili bir çözüm olarak yaygın olarak kullanılmaktadır. Ancak, genellikle genel bulut altyapısı üzerinde depolanan verilerin bütünlüğünü doğrulamak konusundaki kullanıcı endişeleri ve yasal sınırlamalar, önemli soru işaretleri oluşturmuştur. Özellikle depolanan dosyaların denetlenmesiyle elde edilecek geri getirilebilirlik kanıtının sağlanması yeni bir zorluk oluşturmaktadır. "Geri getirilebilirlik kanıtı" terimi, bir olayın veya ihlalin nedenlerini anlamak, soruşturmak veya yasal gereksinimlere uygun olarak belgelenmiş bir durumu doğrulamak için kullanılır. Güvenli bulut depolama sistemleri, gerekli güvenlik seviyelerini sağlamak için ek yüklere sahitir. Kullanıcı deneyimlerini optimize etmek için ise bulut ve yerel hesaplama kaynaklarının birleştirilmesi gerekmektedir. Büyük veri işleme paradigmalarındaki artan yerel işlem kapasiteleri, büyük veri uygulamaları için hash fonksiyonlarına olan ilgiyi artırmaktadır.

Hash fonksiyonlarının bu bağlamdaki rolü, veri bütünlüğünü sağlamak, özgünlük kontrolü yapmak ve bulut güvenliği açısından kritik bir güvenlik katmanı oluşturmak olarak özetlenebilir. Bu fonksiyonlar, depolanan verilerin değişmediğini doğrulamak, veri bütünlüğünü korumak ve yetkisiz erişimlere karşı koruma sağlamak amacıyla kullanılır (Doukas vd. 2019).

3.6.8 Salting (Tuzlama)

Hash fonksiyonlarının güvenliğini artırmak amacıyla kullanılan "tuzlama" tekniği, düz metin dizelerinin hash değerlerinin saldırılara karşı daha dayanıklı hale getirilmesini sağlar. Örneğin, basit bir kullanıcı senaryosunda, A kullanıcısının "12345" şifresinin hash değeri veritabanında depolandığında, aynı şifreyi kullanan başka bir kullanıcı B için aynı hash değeri üretilecektir. Bu durum, saldırganların önceden hesaplanmış tabloları (gökkuşağı tabloları) kullanarak çeşitli kombinasyonları deneyerek orijinal şifre değerini bulmalarına olanak tanır.

Bu sorunu aşmak için, düz metin değeri önce rasgele bir değerle birleştirilerek (tuzlama) hashlenmelidir, böylece her seferinde aynı hash değerini üretmez. Tuzlama tekniği, önceden hesaplanmış sözlük saldırılarına veya gökkuşağı tabloları saldırılarına karşı koruma sağlar. Tuz, parolaların hash işleminden sonra daha fazla rastgelelik eklemek amacıyla kullanılan bir güvenlik önlemidir. Tuz, rastgele karakterler, sayısal rakamlar veya özel karakter dizileri şeklinde parolayla birleştirilir. Veritabanı depolama sistemlerinde, her parola için benzersiz ve farklı tuz değerleri kullanılması önerilir.

Tuz, dizeye eklenerek mükemmel gizliliği sağlamaz; ancak parolayı kırmak için hesaplama zorluğu sağlar. Tuzlanmış hash değerleri, kaba kuvvet saldırılarına karşı direnci artırarak, hash'i kırmak için önemli miktarda zaman gerektirebilir. Tuzlar genellikle özeldir, ancak biri genel, diğeri özel olmak üzere iki farklı tuz kullanmak çevrimdışı parola tahmin saldırılarına karşı ek bir koruma sağlayabilir (Rathod vd. 2020).

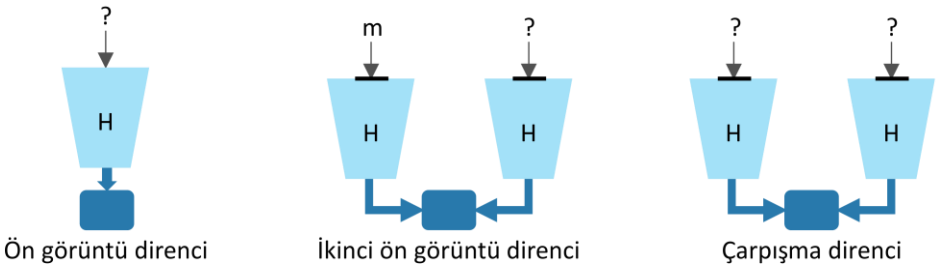
3.7 Hash Fonksiyonlarında Güvenlik

Günümüzde hash fonksiyonları güvenlik için geliştirilen uygulamalardaki ana araçlardan biridir. Uygulamalar hash fonksiyonlarının farklı özelliklerine bağlı olarak geliştirilmektedir. Bir hash fonksiyonunun sahip olması gereken üç temel özellik aşağıda verilmiştir.

- Ön görüntü direnci: Bir hash fonksiyonunun çıktısı olarak h kodu verildiğinde, H(x) hash fonksiyonunu kullanarak herhangi bir x girdisini bulmak hesaplama açısından imkansız olmalıdır.

- İkinci ön görüntü direnci: Bir hash fonksiyonuna girdi olarak m verildiğinde, H(y)=H(m) denklemi aynı çıktıyı veren m'den farklı bir y girdisi için hesaplanamamalıdır.

- Çarpışma direnci: (x,y) değer çiftleri girdi olarak kabul edildiğinde, H(x)=H(y) eşitliğini verebilecek hiçbir girdi çifti olmamalıdır (Alkandari vd. 2013). Hash fonksiyonu güvenlik özellikleri blok diyagramı Şekil 3.19'da verilmiştir.



Şekil 3.19 Hash fonksiyonu güvenlik özellikleri

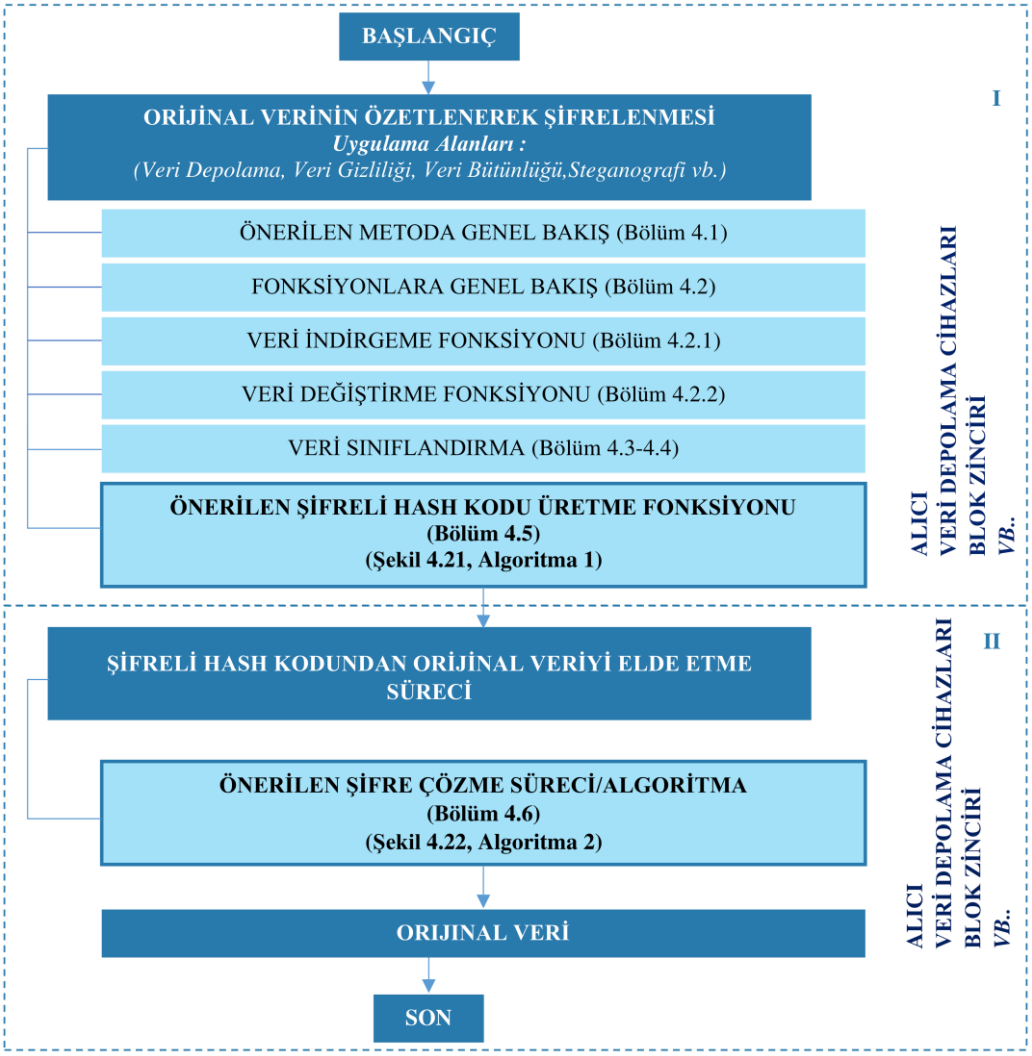
3.8 Steganografi

Steganografi, veri gizleme bilimini ifade eder. Bu bilim dalı, verinin bir taşıyıcı öğe üzerinde gizlenmesini amaçlar ve bu şekilde iletilerek üçüncü tarafların bu veriyi incelemesini ve ele geçirmesini engellemeyi hedefler (Li ve Lu 2018). Taşıyıcı öğe genellikle metin, ses, görüntü veya video gibi çeşitli medya türlerini içerebilir. Gizli verinin, taşıyıcı öğeden çıkarılabilir olup olamayacağına bağlı olarak, bu süreç tersine çevrilebilir veya tersine çevrilemez olarak sınıflandırılabilir (Jung 2018).

Steganografi başarısı, gizlenen verinin kapak nesnesinde algılanamaz olmasına bağlıdır. Belirsizlik, sağlamlık ve algılanamazlık, steganografinin temel özelliklerini oluşturur. Steganografide veri, uzaysal ya da frekans uzayında gizlenir. Uzaysal alanda veri gizleme, kapak görüntüsünün piksellerini değiştirerek gerçekleştirilir. Frekans uzayında ise veri, matematiksel yöntemler kullanılarak gizlenir (Subhedar ve Mankar 2018). Kapak görüntüsünün kalitesini bozmadan gizlenen veriyi maksimize etmek, steganografinin kilit konularından biridir. İnsan gözü, küçük miktarlardaki gizli veriyi tespit edemez. Ancak büyük miktarlarda gizli verinin varlığı, istatistiksel testlerle ortaya çıkarılabilir (Vanmathi ve Prabu 2018). Steganografinin kriptolojiye göre en önemli avantajı, tespit edilemez olmasıdır. Kriptoloji kullanılarak şifrelenen veriler genellikle dikkat çeker. Çünkü anlaşılmaz hale getirilen bilgiler saldırganların ilgisini çeker. Ancak, verinin saldırganların dikkatini çekmeden iletilmesi gerektiğinde, steganografik yöntemler kullanılarak veri bir örtü nesnesine gizlenir. Gizlenen veri, örtü nesnesi içinde alıcıya güvenli bir şekilde iletilir ve bu şekilde güvenlik sorunları aşılır (Bai vd. 2017).

4. ÖNERİLEN METODOLOJİ - İKİ YÖNLÜ HASH FONKSİYONU (İYHF)

4.1 İYHF Genel Bakış



Şekil 4.1 İYHF yönteminin açıklama dizisi diyagramı

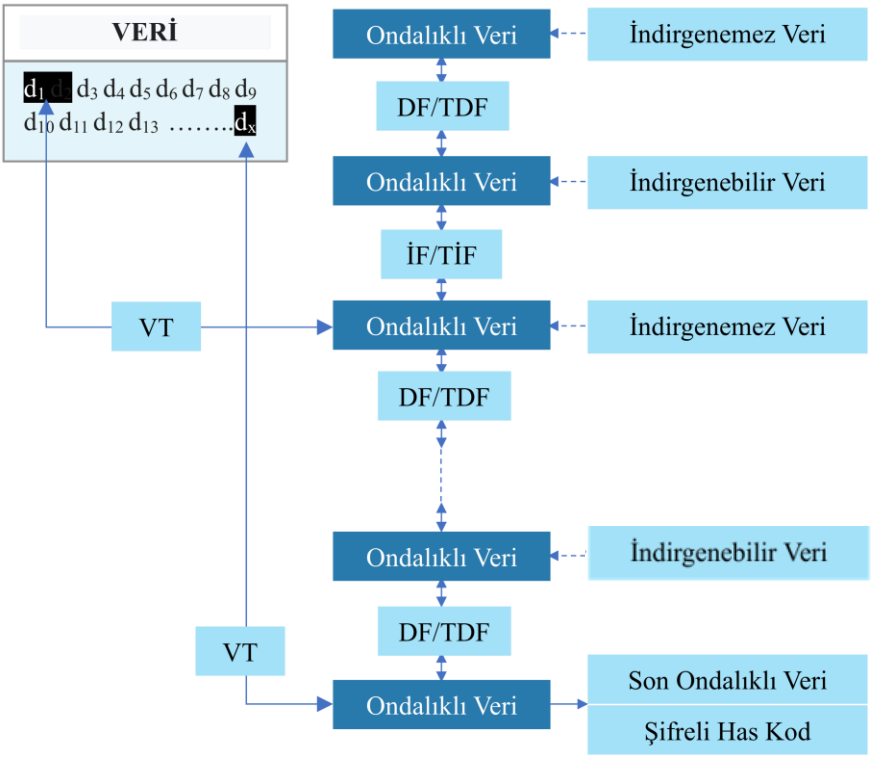
Bu bölümde, İYHF'nin temel kavramları ve yöntemin işleyişi blok diyagramlar ile algoritmalar kullanılarak açıklanmaktadır. Şekil 4.1, İYHF yönteminin açıklama dizisi diyagramını göstermektedir. Blok iki bölümden oluşmaktadır. Birincisi orijinal veriyi hashlemek ve şifrelemek, ikincisi ise şifrelenmiş hash kodundan orijinal veriyi elde etmektir. İYHF yöntemine ve işlevlerine genel bir bakış sırasıyla Bölüm 4.1'de

verilmiştir. Fonksiyonların ayrıntıları Bölüm 4.2, 4.2.1 ve 4.2.2'de açıklanmaktadır. İYHF'nin iş akışında önemli rol oynayan dört veri sınıfı ve detayları Bölüm 4.3 ve 4.4'te açıklanmıştır. Temel bilgiler verildikten sonra, hash/şifreleme iş akışı ve İYHF algoritması Bölüm 4.5'te detaylandırılmıştır. Şifrelenmiş hash kodundan orijinal verinin elde edilmesine yönelik iş akışı ve algoritma Bölüm 4.6'da sunulmuştur.

Şekil 4.2'de İYHF yönteminin genel çalışma diyagramında veri toplama (ekleme) (VT), değiştirme fonksiyonu (DF), ters veri değiştirme fonksiyonu (TDF), indirgeme fonksiyonu (İF) ve ters veri indirgeme fonksiyonu (TİF) gösterilmektedir. İYHF yöntemi iki yönde hareket eden bir ondalık sayı örüntüsü üretir ve bu sayı örüntüsü üzerinde işlemler gerçekleştirir. Bu bölümde ondalık sayının ikili karşılığı ondalık veri olarak tanımlanmıştır.

İYHF şifreleme ve hash kodu üretme süreci rastgele bir ondalık veri ile başlar. Sürecin bu aşamasında değişim fonksiyonu Huffman kodlamasına göre indirgenebilecek bir ondalık veri bulana kadar uygulanır. İYHF'nin bu kısmı bir sözde rastgele sayı üreteci gibi çalışır. Uygun bir ondalık veri bulunursa, Hufmann kodlamasına göre indirgenir ve orijinal verinin indirgenme miktarına eşit bir kısmı bu ondalık veriye birleştirme şeklinde eklenir. Böylece ondalık veri yeniden orijinal bit uzunluğuna ulaşır. Bu işlem, orijinal verinin tüm bitleri sayı örüntüsünde gizlenene kadar tekrarlanır. Örüntünün son ondalığı şifrelenmiş hash kodudur.

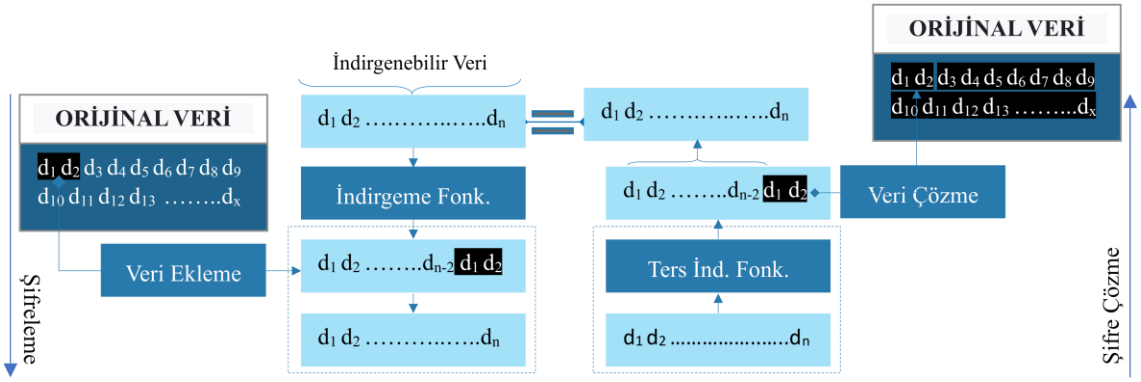
İYHF'nin şifre çözme işlemi, şifreli hash kodu oluşturma işleminde uygulanan fonksiyonların tersi alınarak gerçekleştirilir. Bu işlemin ilk ondalık verisi, şifreleme ve hash kodu üretme işlemlerinin son verisi olan hash kodudur. Böylece süreç, şifreli hash kodu üretme sürecinin tersi yönde ilerler. Amaç, sayı örüntüsünü geri almak ve örüntüde saklı olan orijinal verinin tüm bitlerini ortaya çıkarmaktır.



Şekil 4.2 İYHF genel çalışma prensibi.

4.2 İYHF Metodu Fonksiyonlarına Genel Bakış

Şekil 4.3 veri indirgeme, ters veri indirgeme fonksiyonlarını ve veri ekleme-çözme işlemlerini detaylandırmaktadır. Değişim fonksiyonu, diğer indirgenebilir verileri bulmak ve modelin devam ettiğini göstermek için şekle dahil edilmiştir. Şekilde, azaltma fonksiyonuna giren uygun ondalık verinin iki bit azaltıldığı varsayılmaktadır. Ortaya çıkan 2 bitlik boşluk orijinal verinin d_1d_2 bitleri ile doldurulur. Böylece veri toplama(ekleme) işlemi tamamlanmış olur. Veri kod çözme işlemi de şekilde yer almaktadır. Ondalık verinin içine gömülü olan d_1d_2 verisini elde etmek için ters değişim fonksiyonu uygulanır. Fonksiyonun çıkışında, eklenen veri ve indirgenebilir veri tekrar elde edilir.



Şekil 4.3 Fonksiyonlara genel bakış

4.2.1 Veri indirgeme fonksiyonu

Huffman kodlama, diğer veri sıkıştırma algoritmalarına göre daha iyi sonuçlar veren ve algoritma karmaşıklığı daha düşük olan kayıpsız bir veri sıkıştırma algoritmasıdır (Liu vd. 2022). RLE, Shannon-Fano, Huffman, LZW ve Aritmetik Kodlama kayıpsız sıkıştırma algoritmaları arasında yapılan testlerde verimlilik, kodlama-kod çözme süresi, sıkıştırma oranı vb. açısından en iyi performansın Huffman kodlama algoritması tarafından elde edildiği belirlenmiştir (Rahman ve Hamada 2019). Bu nedenlerden dolayı, İYHF metodolojisinde Huffman kodlaması kullanılmıştır.

Huffman kodlaması, Huffman kodlama ağacına göre oluşturulan bir frekans tablosuna dayanmaktadır. Önerilen yöntemde indirgenecek bit dizileri ikişerli gruplar olarak ele alınmaktadır. Bu durumda gruplar $2^2 = 4$ farklı değer alabilmektedir. Bunlar $(00)_2$, $(01)_2$, $(10)_2$ ve $(11)_2$'dir. Bu verilere semboller atamak Huffman kodlama ağacının kullanımını kolaylaştırır. Semboller harfleri oluşturur. İlgili kodlar ve semboller Çizelge 4.1'de sunulmuştur.

$(00\ 00\ 00\ 00\ 10\ 10\ 11\ 01)_2$ verisinin ikişerli gruplara ayrılmış sembolik gösterimi "aaaaccdb "dir. Huffman kodlaması yüksek frekanslı veriler için etkilidir (Arshad vd. 2016). Veri uzunluğunu azaltmadaki başarı, veri içindeki uzun veri bloklarının az sayıda sembolle, kısa veri bloklarının ise çok sayıda sembolle tanımlanmasına bağlıdır.

Çizelge 4.1 Kodlar ve harf sembolleri

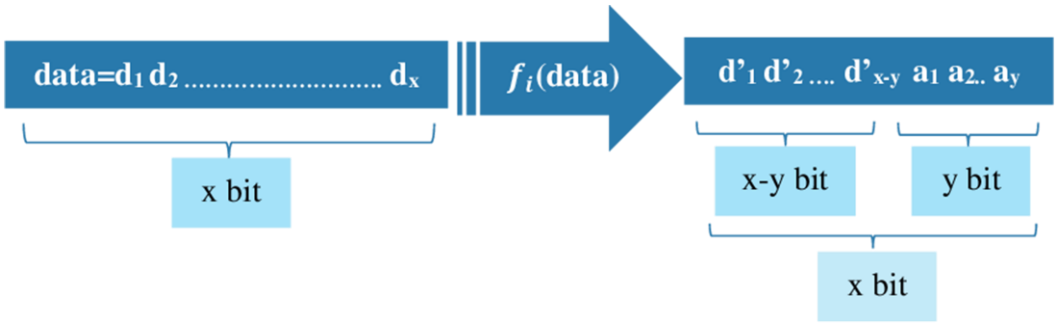
Code	Symbol
00	a
01	b
10	c
11	d

Huffman kodlama ağacı ile kodlama öncesi ve sonrası elde edilen değerlerin eşleştirildiği frekans tablosu Çizelge 4.2'de gösterilmiştir. Bu frekans tablosuna göre verideki iki bit uzunluğundaki $(00)_2$ bitlerinin frekansı arttıkça azalma miktarı da artmaktadır.

Çizelge 4.2 Frekans Tablosu

	a	b	c	d
	00	01	10	11
Frekans Tablosu =	[“1”, “00”,	“010”,	“011”]	

4.2.1.1 Veri ekleme

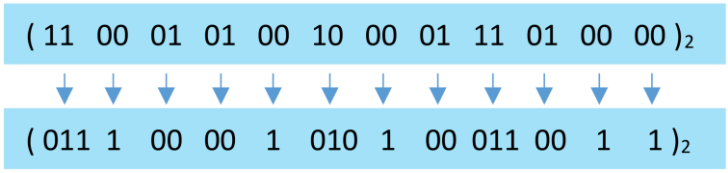


Şekil 4.4 Veri ekleme işleminin blok diyagramı

Bu işlem, Huffmann kodlaması kullanılarak x bitten x-y bite indirgenen veriye y bit uzunluğunda veri eklenmesini içerir. Bu durumda x-y uzunluğundaki veri tekrar x bit uzunluğuna ulaşır. Veri ekleme işleminin blok diyagramı Şekil 4.4’te gösterilmektedir.

Veri azaltma ve ekleme işlevinin uygulanmasına ilişkin bir örnek Şekil 4.5'te aşağıdaki gibidir:

Örnek veri : 24 bit $(110001010010000111010000)_2 = (12919248)_{10}$



Şekil 4.5 İndirgeme fonksiyonu uygulama örneği

Örnek veri, frekans tablosuna göre her biri iki bit uzunluğunda 12 gruba ayrılmıştır. Her gruptaki iki bitlik veriler frekans tablosundaki kod değerleri ile eşleştirilmiştir. Eşleştirme sonucunda elde edilen yeni 12'li grubun bit uzunluğu 24 bitten küçük olmalıdır. Eşleştirme sonrası elde edilen 22 bit uzunluğundaki indirgenmiş veri, Şekil 4.5'te gösterildiği gibi $(0111000010101000110011)_2$ şeklindedir. Bu durumda iki bit veri eklenmiştir. Bu örnekte eklenecek verinin $(00)_2$ olduğu varsayılmıştır. Veri LSB tarafına eklenir. Böylece 24 bit uzunluğundaki yeni veri $(011100001010100011001100)_2 = (7383244)_{10}$ olur.

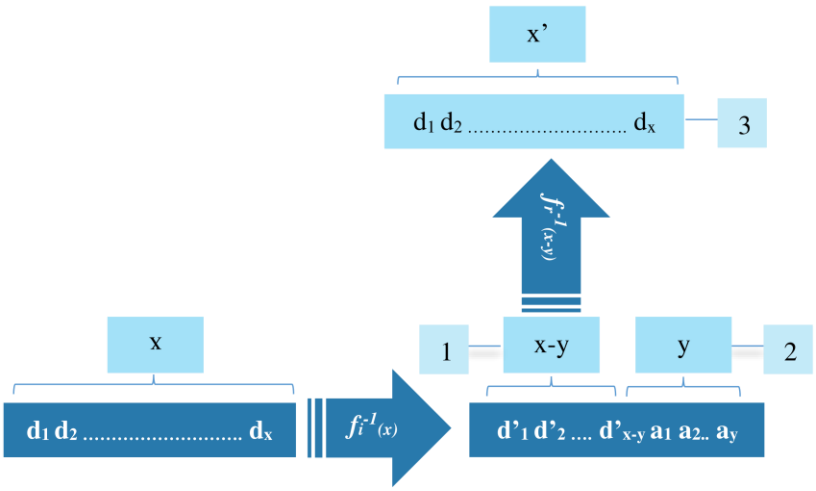
4.2.1.2 Veri çözme

Verilerin indirgenmiş veri + toplanmış veri olarak ayrıştırılması işlemidir. Bu işlem ters Huffmann indirgeme fonksiyonu $f_{i-1}(x)$ kullanılarak gerçekleştirilir. Veri çözme sürecinin blok diyagramı Şekil 4.6'da gösterilmektedir. Çıkarılacak veriler:

1) x-y bit uzunluğunda indirgenmiş veri,

2) y bit eklenmiş veri,

3) x-y uzunluğundaki indirgenmiş verilerin indirgendiği orijinal veri x'.



Şekil 4.6 Veri çözme süreci blok diyagramı

Ters indirgeme fonksiyonu ve veri çözme uygulamasının bir örneği aşağıdaki gibidir.

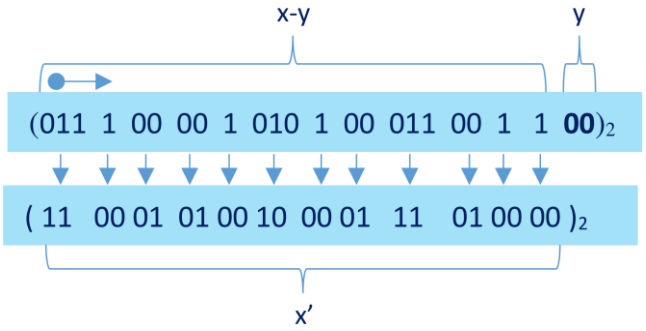
Veri çözümüne bir örnek Şekil 4.7'de $(011100001010100011001100)_2 = (7383244)_{10}$ verisi için azaltılmış veri + eklenmiş veri formatında gösterilmiştir. Ters fonksiyon uygulanırken, veriler frekans tablosundaki $(1)_2$, $(00)_2$, $(010)_2$, ve $(011)_2$ bitlerine göre gruplandırılmıştır. Verilerin "çözülebilir" olarak kabul edilebilmesi için 12 gruptaki toplam bit uzunluğunun 24 bitten az olması gerekmektedir. Buna göre, eğer $f_i^{-1}(x)$ uygulanırsa,

1) $x\text{-}y=(011100001010100011001 1)_2$,

2) $y=(00)_2$,

3) $x'=(110001010010000111010000)_2 = (12919248)_{10}$

olarak bulunur.



Şekil 4.7 Veri çözme örneği

4.2.2 Veri değiştirme fonksiyonu

Bu yöntemde kullanılan bir diğer fonksiyon ise veri değişim fonksiyonudur. Bu yöntem özgün olarak bu çalışmada geliştirilmiştir. Amacı verilerin değerini değiştirmektir. Ek önlemler olmadan, ondalıklı işlemlere dayalı değişim fonksiyonunun çıktısının tersi alındığında, fonksiyonun giriş verileri doğru bir şekilde elde edilemez. Bunun nedeni yuvarlamadır. Yuvarlama hataları her bilgisayar programcısının çok dikkat etmesi gereken sıkıntılı bir problemdir (Paterson ve Glasbey 1985). Değişim fonksiyonunun çıktısı ters değişim fonksiyonun girdisi olduğunda giriş verilerinin yeniden elde edilmesi gerekir. Aksi takdirde şifreli hash kodu sürecinde üretilen sayı örüntüsü şifre çözme sürecinde yeniden elde edilemez. Bu durum şifrelenmiş veriyi çözülemez hale getirir.

4.2.2.1 Veri değiştirme fonksiyonunun matematiksel ifadesi

Ondalık verilerin tamsayı kısmının basamak sayısını bulan fonksiyon Denklem (4.1)'de gösterilmiştir. Veri değiştirme fonksiyonu için girdi olarak kullanılacak ondalık veriler Denklem (4.2)'de verilen normalizasyon fonksiyonu ile 0-1 arasında normalize edilmiştir. Hata miktarı fonksiyonu Denklem (4.3) ile verilmiştir ve hata miktarını tamsayıya dönüştürmek için kullanılır. Ters değişim fonksiyonunun çıkışında elde edilen veri, yuvarlama hatası nedeniyle değişim fonksiyonunun girişi olan ondalıklı veriye eşit değildir. Bu fark hatanın miktarını gösterir. Veri değiştirme fonksiyonu, hata ve

normalizasyon fonksiyonlarının toplamıdır. Değiştirme fonksiyonu Denklem (4.4) ile verilmiştir.

d veri ve d ∈ ℝ

$\boldsymbol{d_x}$: n elemanlı bir kümede x. veri $\boldsymbol{n} \in \boldsymbol{Z^+}$

$\boldsymbol{d_{int}}$: Ondalık verinin tamsayı kısmının bit uzunluğu;

$\boldsymbol{d_{int}} = \{\boldsymbol{d_{int}} | \mathbf{24} \leq \boldsymbol{d_{int}} \leq \mathbf{52}, \mathbf{x} \in \mathbb{Z}^+\}$

$\boldsymbol{d_{pre}}$: Ondalık verilerin ondalık kısımlarının hassasiyeti.

$\boldsymbol{f_{norm}(d_x)}$: Verileri 0-1 aralığına normalleştirme işlevi.

$\boldsymbol{f_{err}(x)}$: Hata miktarı fonksiyonu.

$\boldsymbol{f_{change}(x)}$: Veri değiştirme fonksiyonu.

$$d_{pre} = \left\lfloor log_{10^{2^{d_{int}-1}}} \right\rfloor + 2 \tag{4.1}$$

$$f_{norm}(d_x) = \frac{x}{(2^{d_{int}-1})} \tag{4.2}$$

$$f_{err}(d_x) = \left[(f_{norm}(x)(2^{d_{int}} - 1) - x)\left(10^{d_{pre}}\right)\right] + 2^{d_{int}-1} \tag{4.3}$$

$$f_{change}(d_x) = f_{err}(x) + f_{norm}(x) \tag{4.4}$$

4.2.2.2 Ters veri değişim fonksiyonunun matematiksel ifadesi

Ters değişim fonksiyonu (4.5) ile verilmektedir. Bu fonksiyonun girdisi olan ondalıklı verinin tamsayı kısmı hata miktarını, ondalıklı kısmı ise normalize edilmiş sayıyı vermektedir. İlk olarak verilerin ondalık kısımlarına ters normalizasyon uygulanmıştır.

Ancak, bu işlemden elde edilen çıktı yuvarlama nedeniyle hatalıdır. Hatanın saklandığı ondalık verinin tamsayı kısmı kullanılarak orijinal veri hatasız olarak tekrar elde edilmiştir.

$$f_{change^{-1}}(d_{x'}) = (d_x{}')\,[(x - \lfloor x \rfloor)(2^{d_{int}} - 1)] - \left(\frac{\lfloor x \rfloor - 2^{d_{int}-1}}{10^{d_{pre}}}\right) \quad (4.5)$$

Ters değişim fonksiyonu için bir örnek aşağıda gösterilmiştir. Bu örnekte d=195.00000000 ondalık verisinin tamsayı kısmının uzunluğu 24 bit, ondalık kısmının uzunluğu ise 28 bit olarak alınmıştır. Bu fonksiyonların parametreleri ve çıktıları sırasıyla Çizelge 4.3 - Çizelge 4.4 ile verilmiştir.

Çizelge 4.3 Veri değiştirme fonksiyonu örneği ve sonuçları

Veri değişim süreci	Açıklama	Sonuç
d_{int}	Ondalık verinin tamsayı kısmının bit uzunluğu	24
d_{pre}	Ondalık verilerin ondalık kısmının hassasiyeti	$\lfloor log_{10^{(2^{24}-1)}} \rfloor + 2 = 8$
d_x	Değiştirilecek ondalık veri	195,00000000
$f_{norm}(195{,}00000000)$	[195,00000000/(2^{24} -1)]	0,00001162
$f_{err}(195{,}00000000)$	[(0,00001162)(2^{24}-1)]-[(3512438-(2^{24} /2)]/(10^8)	3512438
$f_{change}(195{,}00000000$	3512438+0,00001162	3512438,00001162

Çizelge 4.4 Ters veri değişim fonksiyonu örneği ve sonuçları

Veri Çözme Süreci	Açıklama
$f_{change}{}^{-1}(3512438{,}00001162)$	(0,00001162x(2^{24}-1))-(3512438-(2^{24-1}))/(10^8)
$f_{change}{}^{-1}(3512438{,}00001162)$	195,00000000

4.3 Verilerin sınıflandırılması

İYHF'de şifreleme ve şifre çözme işlemlerinde ondalıklı sayı örüntüsü üretimi belirli kurallara göre gerçekleştirilir. Bu kurallar kullanılan ondalık verinin türüne göre

uygulanmaktadır. Veri tipleri ondalık verinin tamsayı ve ondalık kısımları için ayrı ayrı belirlenmektedir. Bu çalışmada belirlenen veri tipleri dört sınıfa ayrılmıştır. Bunlar:

1. İndirgenebilir ve Çözülebilir,

2. İndirgenebilir ve Çözülemez,

3. İndirgenemez ve Çözülebilir,

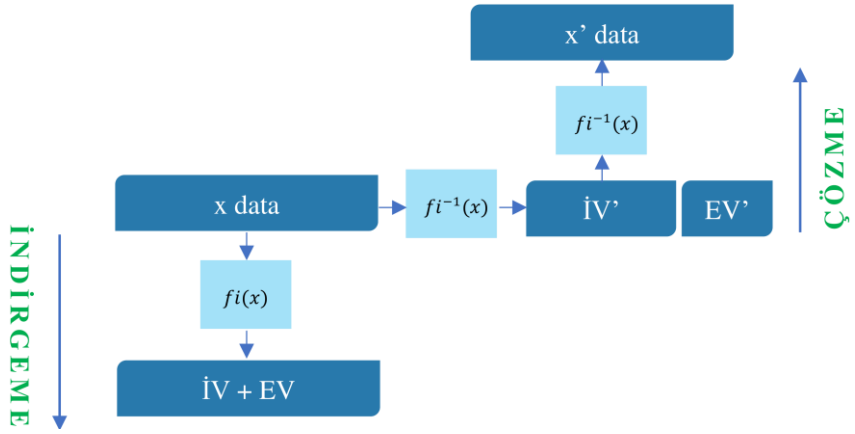
4. İndirgenemez ve Çözülemez.

Veri türünü belirlemek için iki fonksiyon kullanılmıştır. Bunlar indirgeme ve ters-indirgeme fonksiyonlarıdır. İndirgeme fonksiyonu ilk olarak veri türünü belirlemek için uygulanır. Verinin indirgenebilir olarak kabul edilebilmesi için en az "1" bit indirgenmiş olması gerekir. Bu koşul sağlandıktan sonra, verilere bir ters indirgeme fonksiyonu uygulanır. Verinin çözülebilir olarak kabul edilebilmesi için iki koşulun sağlanması gerekir. Birincisi, ters indirgeme fonksiyonunun çıktısının veri bit uzunluğu/2 kadar gruba bölünebilmesidir. İkinci koşul ise grubun toplam bit uzunluğunun verinin bit uzunluğundan daha az olmasıdır. Veri bit uzunluğu ile indirgenmiş veri bit uzunluğu arasındaki fark "eklenen veri" miktarıdır. Bu işlemler yalnızca veri türünü belirlemek için kullanılır. Gerçekte indirgenmiş veriye herhangi bir veri ekleme işlemi uygulanmaz. Benzer şekilde, şifre çözme işleminde veri türü tespit edilirken, eklendiği tespit edilen tüm veriler yok sayılır. Bu çalışma kapsamında tanımlanan veri tiplerine ilişkin detaylar aşağıdaki gibidir.

4.3.1 Tür 1

Bu veriler indirgenebilir ve çözülebilir verilerdir. Veri türünü belirlemek için indirgeme ve ters indirgeme fonksiyonları ayrı ayrı uygulanır. Veriler indirgenip çözülebiliyorsa Tür 1 olarak sınıflandırılır. Blok diyagramı Şekil 4.8'de gösterilmektedir. Şifrelenecek ve hashlenecek verinin bitleri tür-1 verisine eklenir. Bunun nedeni verinin

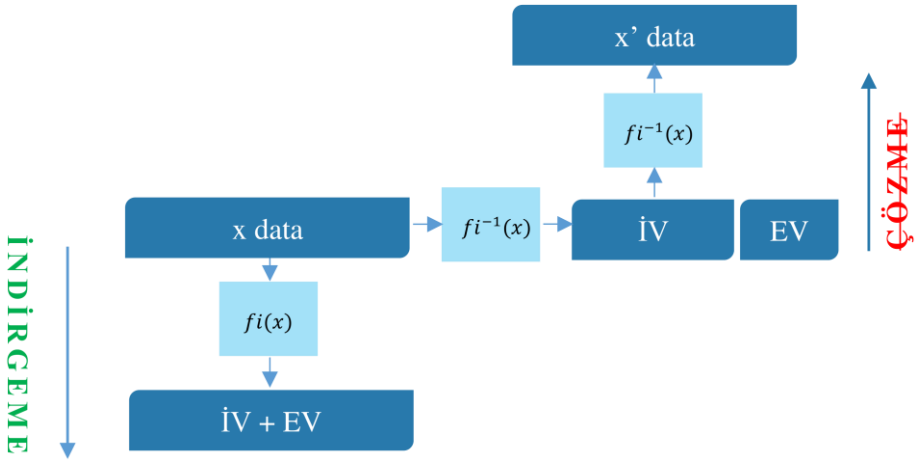
indirgenebilmesidir. Veriler indirgeme işlemi ile orijinal bit uzunluğundan daha küçük bit uzunluklarına indirgenir. Böylece indirgenmiş veriye indirgeme miktarına göre düz metin bitleri eklenebilir. Şekilde x verisinin İV ve EV birleşiminden meydana geldiği görülmektedir.



Şekil 4.8 Veri türü 1'in blok diyagramı

4.3.2 Tür 2

Bu veriler indirgenebilir ve çözülemez özellikler gösterir. Blok diyagramı Şekil 4.9'da gösterilmektedir. Sayıları diğer veri türlerine kıyasla çok azdır. Bu tür veriler indirgenebildiğinden dolayı düz metin veri bitleri bu verilere eklenebilir. Ancak sayıları çok az olduğu için veri eklenemeyen veri türleri olarak kategorize edilmiştir. Kriptografik süreçte bu tür verilerle karşılaşıldığında indirgeme yerine değiştirme fonksiyonu uygulanır. Bu yüzden şifre çözme sürecinde ters değişim fonksiyonu uygulanır.



Şekil 4.9 Veri türü 2'nin blok diyagramı

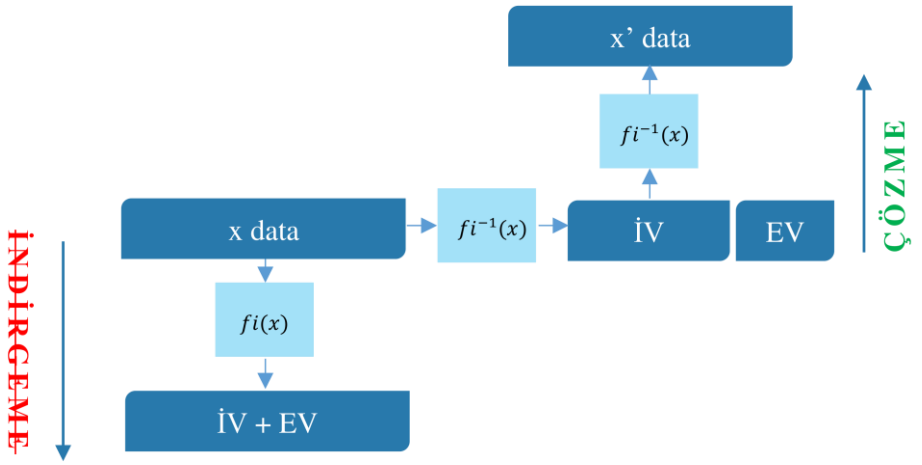
4.3.3 Tür 3

İndirgenemeyen ve çözülebilen verilerdir. Blok diyagramı Şekil 4.10'da gösterilmiştir. Bu çalışmada kullanılan veri setinde en çok bulunan veri türüdür. Bu türdeki verilere, veri bitleri eklenemez. Bunun nedeni boyutlarının indirgenememesidir. Bu nedenle indirgeme fonksiyonu yerine değiştirme fonksiyonu uygulanır. Tür 3 veriler çözülebilir oldukları için indirgenmiş ve eklenmiş veri parçalarına ayrılabilirler. Bu özellikler şifre çözme sürecinde belirsizliğe yol açabilmektedir.

Belirsizlik kavramı, Şekil 4.11'deki blok diyagramda gösterilen senaryo ile açıklanabilir. Farklı veri türlerine farklı fonksiyonlar uygulanabilir. Ancak, fonksiyonun çıkışındaki veriler aynı türde olabilir.

Şekil 4.11'de gösterilen ilk durumda, Tür 1 verileri $f_i(x)$ fonksiyonu kullanılarak indirgenmiştir. İndirgenme miktarına eşit bit veya bitler fonksiyon çıkışında 1 türündeki veriye eklenmiştir. Aynı senaryonun ikinci durumunda, değişim fonksiyonu Tür 3 verisine uygulanmıştır. Her iki durumda da fonksiyon çıkışlarından elde edilen veri türünün Tür 3 olduğu varsayılmıştır. Şifre çözme süreci şifreli hash kodu üretme sürecinin tersi yönünde ilerlediğinden dolayı şifre çözme süreci için senaryonun her iki durumunda da, fonksiyonların çıkışındaki Tür 3 veri ile karşılaşılmıştır. Tür 3 verinin çözülebilir

doğası nedeniyle, her iki senaryoda da Tür 3 veri içinde eklenmiş veri algısı oluşmaktadır. İlk senaryoda, Tür 3 veri bitlerinin LSB tarafı orijinal metin bitleri içermektedir. Bu eklenen veriler çalışma kapsamında "gerçek" olarak nitelendirilmiştir. İkinci durumda, aslında hiçbir veri eklenmemiştir. Bunun nedeni Tür 3 verilerinin değiştirme işlevi kullanılarak elde edilmiş olmasıdır. Ancak, indirgeme fonksiyonu kullanılarak elde edilmiş durumu sergilemiştir. Bu durumda, çalışma kapsamında "sahte" olarak nitelendirilen veriler içerebilirler. Şifre çözme sürecinde, ek kurallar oluşturulmadığı sürece "gerçek" ve "sahte" veriler birbirinden ayırt edilemez. Bu çalışmada ondalık verinin ondalık kısmına kontrol verisi eklenerek "sahte" veri miktarı önemli ölçüde azaltılmıştır. Ancak bu durum tamamen ortadan kaldırılamamıştır. Bu nedenle hem "gerçek" hem de "sahte" verilerin eşleştirildiği ekstra bir veri bloğu oluşturulmuştur. Bu veri bloğu "kripto ham veri" olarak adlandırılır

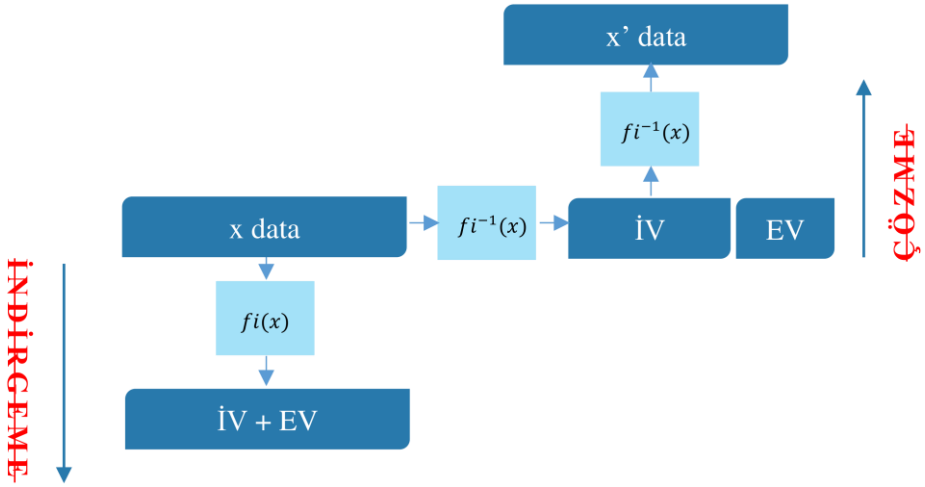


Şekil 4.10 Veri tipi 3'ün blok diyagramı

Şekil 4.11 Veri çözme sürecinde karşılaşılabilecek olasılıklar

4.3.4 Tür 4

İndirgenemez ve çözülemez olan bu veriler Tür 4 olarak nitelendirilir. Blok diyagramı Şekil 4.12'de gösterilmektedir. Bu tür verilere indirgeme fonksiyonu ve veri çözme işlemleri uygulandığında hiçbir çıktı üretilmez. Şifreli hash kodu üretme sürecinde bu tür bir veri ile karşılaşıldığında sadece değişim fonksiyonu uygulanır. Tür 4 veriler şifre çözme sürecinde ortaya çıkarsa, belirsizlik ihtimali bulunmaz.



Şekil 4.12 Veri türü 4'ün blok diyagramı

4.4 Veri Türleri İstatistikleri

İstatistiksel veriler 24 bitlik bir veri kümesi kullanılarak elde edilmiştir. Bu durumda, veri kümesi 0-16777215 aralığındaki tam sayıları kapsamaktadır. Uygulama Python'da geliştirilmiştir. 0 - (2^{24} -1)] aralığındaki tamsayılar parametrik olarak veri türü belirleme fonksiyonuna gönderilmiş ve veri türü geri dönüş değeri olarak elde edilmiştir. Veri türü sayısı kullanılan frekans tablosuna bağlı olarak değişmektedir. Kullanılan frekans tablosu ve elde edilen veri türü sayısı Çizelge 4.5'te listelenmiştir.

Çizelge 4.5 24-bit uzunluğundaki verilere dayalı olarak elde edilen veri türlerinin sınıflandırılmış sayıları

00 01 10 11 Frekans Tablosu = [“0” “11” “101” “100”]		
Tür no	Açıklama	Veri Türlerinin Sayısı
1	Küçültülebilir ve Çözülebilir	1907370
2	Küçültülebilir ve Çözülemez	12773
3	Küçültülemez ve Çözülebilir	12949703
4	Küçültülemez ve Çözülemez	1907370
Tür 3 verisinin Tür 1’e oranı		12949703/1907370=**6,8**
Toplam		2^{24}=16.777.216

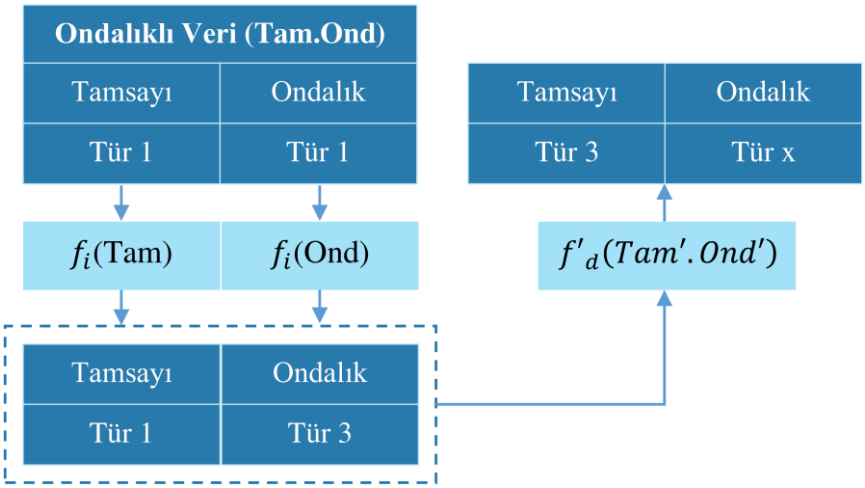
4.5 Şifreli Hash Kodu Üretme Süreci

Şifreli hash kodu üretme süreci öncesi rassal olarak bir başlagıç verisi ve olası frekans tablolarından birisinin seçilmesi gerekmektedir.

Şifreli hash kodu üretme süreci bazı koşullara bağlıdır. Koşullar veri türleri ve blok türleri ile ilişkilidir. Koşulların sağlanma/sağlanmama durumu hangi fonksiyonun uygulanacağına karar verilmesinde etkilidir. Uygulanacak fonksiyonlar veri indirgeme ve veri değiştirme fonksiyonlarıdır. Süreç tüm detayları ile bu bölümde açıklanmaktadır.

4.5.1 Veri yükleme fonksiyon ve koşulları

Verileri şifrelerken ve hash kodları üretirken, verilerin eklenebileceği uygun ondalık sayıları bulmak sürecin en önemli aşamasıdır. Bu işlem için gerekli fonksiyonlar ve koşullar Şekil 4.13'te gösterilmiştir.



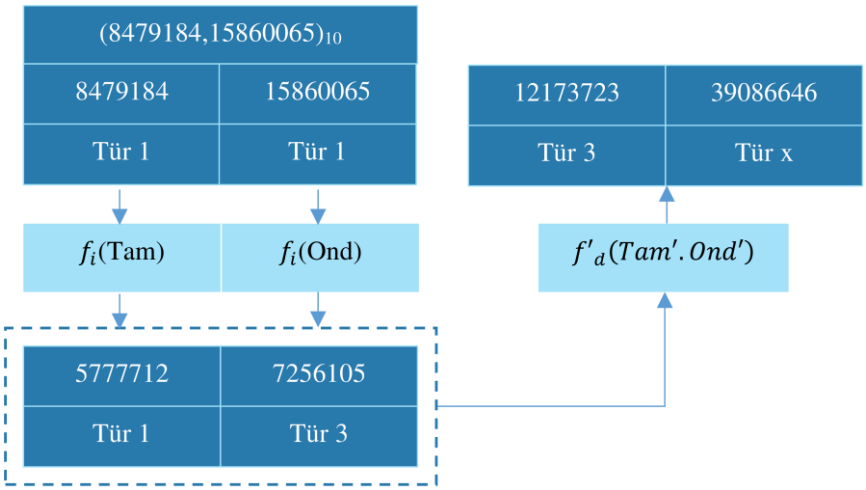
Şekil 4.13 Veri yükleme bloğunda kullanılan fonksiyonlar ve koşullar

Düz metin bitleri ondalık verinin tam sayı kısmına eklenirken, kontrol verileri ondalık kısma eklenir. Veri eklemenin ilk koşulu, ondalık verinin hem tamsayı hem de ondalık kısımlarının tür 1 olmasıdır. Bunun nedeni tür 1 verilerinin indirgenebilir nitelikte olmasıdır. İkinci koşul şu şekildedir. İndirgeme işlevi, ondalık verilerin hem tamsayı hem de ondalık kısımlarına uygulanır. İndirgeme uzunluğuna eşit bit uzunluğundaki veriler ondalık verilerin hem tamsayı hem de ondalık kısmına eklenir. Eklenen veriler bit seviyesinde ve LSB tarafınadır. Sonuç olarak, ondalık verinin hem tamsayı hem de ondalık kısımları orijinal bit uzunluğuna ulaşır. Bu işlemler ondalık verinin sayısal değerlerini değiştirir. İkinci koşul, değişen ondalık verinin tamsayı kısmının tür 1 ve ondalık kısmının tür 3 olmasını gerektirir. Tür 2 ve 4'e ait veriler ikinci koşula dahil edilmezler. Tür 3 verileri tür 1 olanlardan yaklaşık 6,8 kat daha fazladır. Oranlar, Çizelge 4.5'te listelenmiştir. Bu orana göre, verilerin tamsayı kısmının tür 1 yerine tür 3 ile koşullandırılması veri yüklenmediği halde yüklenmiş olarak görülen ve "sahte" olarak nitelendirilen veri bloklarının sayısında önemli bir artışa neden olmaktadır. Bu nedenle tamsayı kısmının tür 1 olması tercih edilmiştir. Ondalıklı kısımların koşullandırılması için ise tür 3 tercih edilmiştir. Bu tercih ikinci koşulun yerine getirilmesini kolaylaştırır. Böylece ondalık verinin her iki kısmı için koşul dengesi korunmuş olur. Üçüncü koşul, ikinci koşulu karşılayan ondalık verilere uygulanır. Bu amaçla koşulu karşılayan veriye ters indirgeme fonksiyonu uygulanır. Fonksiyonun çıktısı ondalık veridir. Bu verinin tamsayı kısmı tür 3 olmalıdır. Ondalık kısım ise herhangi bir türde olabilir. Son koşul,

"sahte" verilerin en aza indirilmesine katkıda bulunmayı amaçlar. Üç koşul da yerine getirildiğinde, veri ekleme işlemi tamamlanmış olur. Koşullardan birinin karşılanmadığı durumda, bloğun başlangıç verileri olan ondalık verilere indirgeme fonksiyonu yerine değiştirme fonksiyonu uygulanır. Sayı örüntüsü bir sonraki blok için devam eder.

4.5.1.1 Veri yükleme örneği

Tüm veri yükleme koşullarını yerine getiren ondalık veri $(8479184,15860065)_{10}$ ve fonksiyon çıkışlarındaki değerleri Şekil 4.14'deki blok diyagramında verilmiştir.



Şekil 4.14 $(8479184,15860065)_{10}$ ondalıklı verisi için veri yükleme bloğu

İşlemin ilk adımında ondalıklı verinin tamsayı $(8479184)_{10}$ ve ondalıklı $(15860065)_{10}$ kısımları için tür belirleme işlemi gerçekleştirilmiştir. Bu verinin türü 1.1 olduğundan dolayı ilk koşul yerine getirilmiştir. Bir sonraki adımda, ondalık verinin hem tamsayı hem de ondalık kısımlarına indirgeme fonksiyonu uygulanmıştır. Ondalıklı verinin tamsayı kısmı indirgeme fonksiyonu ile dört bit uzunluğunda indirgendiğinden dolayı düz metnin sıradaki dört bitlik verisi ondalıklı verinin tamsayı kısmına eklenmiştir. Örnekte eklenecek verinin $(0000)_2$ olduğu varsayılmıştır. Veri eklendikten sonra tamsayı kısmının bit uzunluğu 24 bit ve sayısal karşılığı $(5777712)_{10}$ olmuştur. Ondalık verinin ondalık kısmına kontrol verisi $((1)_2)$ eklendiğinde $(15860065)_{10}$ değeri $(7256105)_{10}$ değerine dönüşmüştür. İndirgeme fonksiyonu ve veri ekleme işlemi kullanılarak hesaplanan yeni

ondalık veri $(5777712,7256105)_{10}$ olmuştur. İkinci koşula göre, üretilen ondalık verinin tamsayı kısmı 1 tipinde ve ondalık kısmı 3 tipinde olmalıdır. $(5777712,7256105)_{10}$ bu koşulları yerine getirmiştir. Son olarak, verilere ters veri değiştirme fonksiyonu uygulanmıştır. Fonksiyon uygulandığında $(12173723,39086646)_{10}$ ondalıklı verisi elde edilmiştir. $(12173723)_{10}$ verisi tür 3 olmalıdır. Bu koşul da sağlanmıştır. Verinin $(3908664646)_{10}$ ondalık kısmı herhangi bir türde olabildiğinden koşula bir etkisi yoktur. Böylece $(8479184,15860065)_{10}$ ondalıklı verisi için tüm koşullar yerine getirilmiş olur. Veri ekleme işlemi, verinin tamsayı kısmına 4 bit uzunluğunda $(0000)_2$ verisinin ve verinin ondalık kısmına 1 bit uzunluğunda $(1)_2$ kontrol verisinin eklenmesiyle sona ermiştir.

4.5.2 Blok türleri

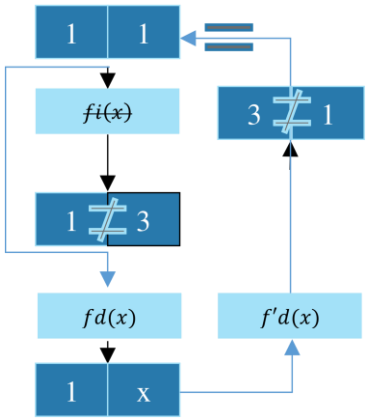
Önerilen yapıdaki tüm olası blok türleri bu bölümde sunulmuştur. Blok türleri, bloğun başlangıç ve bitiş ondalık verilerinin tamsayı ve ondalık kısımlarının türlerine göre belirlenir. Veri tiplerinin gösterimi "tamsayı kısım türü.ondalık veri türü" formatındadır. Örnek formatlar 1.1, 1.3, 3.1, 3.3, 3.4 vb.dir. Blok tipleri aşağıdaki gibi sınıflandırılır.

4.5.2.1 Blok başlangıç türü : 1.1 – Blok bitiş türü : 1.3

Bu bloklar veri yükleme bloklarıdır. Blok 1.1 ile başlar ve 1.3 ile biter. Bu iki blok arasında farklı bir veri türü yoktur. Blok başlangıç verilerine (1.1) indirgeme fonksiyonu ve veri ekleme işlemleri uygulanır. Bu fonksiyon ve işlemler sonucu elde edilen blok sonu verisi 1.3 türünde olmalıdır. Üçüncü koşul olarak blok sonu verisine (1.3) ters değişim fonksiyonu uygulanmalı ve fonksiyon çıkışı veri türü 3.x formatında olmalıdır. Veri yükleme fonksiyonlarını ve koşullarını gösteren Şekil 4.13'deki blok diyagramı bu blok yapısına aittir.

Tür 1.1-1.3 bloklarında meydana gelebilecek durumlar aşağıda listelenmiştir.

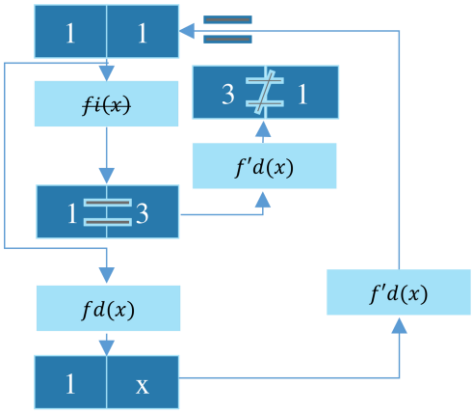
Koşul 2'nin yerine getirilmemesi: Koşullardan biri yerine getirilmezse, tür 1.1 blok başlangıç verilerine hiçbir veri eklenemez. Yerine getirilmemesi olası koşullardan ilki, indirgeme fonksiyonu uygulanarak veri eklendikten sonra elde edilen verinin gerekli tür koşulunu (1.3) sağlamamasıdır. Bu durumda uygulanacak fonksiyon bir değişim fonksiyonudur. Değişim fonksiyonunun çıkışında elde edilecek ondalıklı verinin türü 1.x olabilir. 1.x türündeki veri "sahte" veri sorununa yol açmaz. Bu veri türlerinden 1.1/1.2/1.4, ikinci koşul olan 1.3'ü sağlamadığından "sahte" veri olasılığı yoktur. 1.3 türündeki veriler ikinci koşulu sağlar ve eklenmiş veri içerdiği mesajını verir. Ancak gerçekte durum böyle değildir. Çünkü ondalık veri, değiştirme fonksiyonunun bir çıktısıdır. Bu durumda üçüncü koşul "sahte" verilerin ortaya çıkmasını engeller. Çünkü 1.3 türündeki bu veriye ters değişim fonksiyonu uygulandığında çıktı verisi 1.1 türünde olacağından 3.x koşulu sağlanmaz. Koşulun sağlanmaması “sahte” / “gerçek” veri bulunmadığını açıklar. Tüm bu durumlar Şekil 4.15'te gösterilmektedir.



Şekil 4.15 Blok tipi 1.1-1.x verilerinin veri yükleme kuralını karşılamadığı durum için blok şeması – I

Koşul 2'nin yerine getirilmesi, koşul 3'ün yerine getirilmemesi: Tür 1.1'in ondalık verilerine bir indirgeme fonksiyonu ve veri ekleme işlemi uygulandığında, elde edilen veriler 1.3 türünde olabilir. Bu durumda, ikinci koşul sağlanmış olur. Ancak, üçüncü koşul yerine getirilmezse, 1.1 türündeki ondalık verilere uygulanan indirgeme fonksiyonu iptal edilir ve değiştirme işlevi uygulanır. Veri değiştirme fonksiyonunun çıkışındaki ondalıklı veri 1.x tipinde olması durumunda "sahte" veri olasılığı yoktur. Bunun nedeni, ters veri

değiştirme fonksiyonu uygulandığında fonksiyonun çıkışının 1.1 türünde olmasıdır. Bu durum Şekil 4.16'da gösterilmektedir.



Şekil 4.16 Blok tipi 1.1-1.x verilerinin veri yükleme kuralını karşılamadığı durum için blok diyagramı – II

Blok başlangıç türü 1.1 olan verilere uygulanan ilk veri değiştirme fonksiyonunun çıktısının 2.x/3.x/4.x türlerinden biri olması durumu, blok başlangıç tipi 1.1, ara türler 2.x/3.x/4.x ve bitiş türü 1.x başlığı altında incelenmiştir.

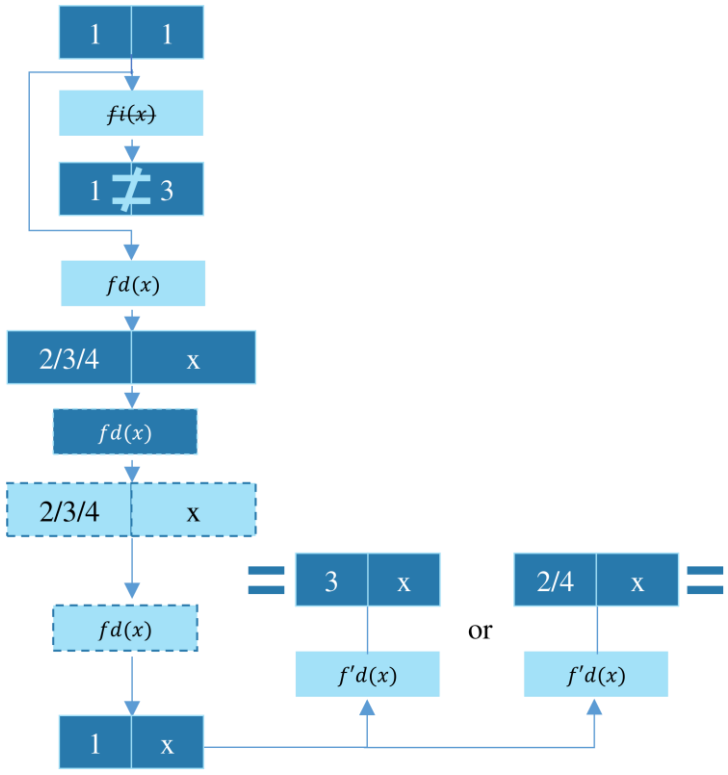
4.5.2.2 Blok başlangıç türü : 1.1 – Ara türler : 2/3/4.x - Blok bitiş türü : 1.x

Bu tür bloklar "sahte" verilerin oluşabileceği bloklardır. Bu bloğun oluşması için ilk koşul, başlangıç blok türünün 1.1 olması ve veri yükleme koşullarından birinin yerine getirilmemesidir. Veri yüklemesi mümkün olmadığından, başlangıç blok verilerine veri değiştirme fonksiyonu uygulanır. Bloğun oluşması için bir diğer koşul, uygulanan ilk veri değiştirme fonksiyonunun çıktısının 1.x türünden farklı bir veri türünde olmasıdır. Bunlar 2.x/3.x/4.x türleridir. Bu tür veriler bloğu sonlandırmaz. Blok, 1.x türünde veri elde edildiğinde sonlanır. Blok, 1.1, 1.2 veya 1.4 tiplerinden biriyle sonlandırılırsa, "sahte" veri çıkışı olmaz. Ancak bloğun 1.3 tipinde sonlandırılması "sahte" verilere neden olabilir. Bu verilere ters veri değiştirme fonksiyonu uygulandığında, 3.x türünde veri elde edilir ve kontrol verisi/verileri koşulu sağlanırsa, blok bir “veri yükleme bloğu” olarak kabul edilir. Bu düşünce, şifre çözme sürecinde ters veri değiştirme fonksiyonu yerine ters

indirgeme fonksiyonunun uygulanmasına yol açar. Yanlış fonksiyonun uygulanması, örüntünün bu bloktan yukarı doğru yanlış üretilmesine ve şifre çözme sürecinin başarısız olmasına neden olur.

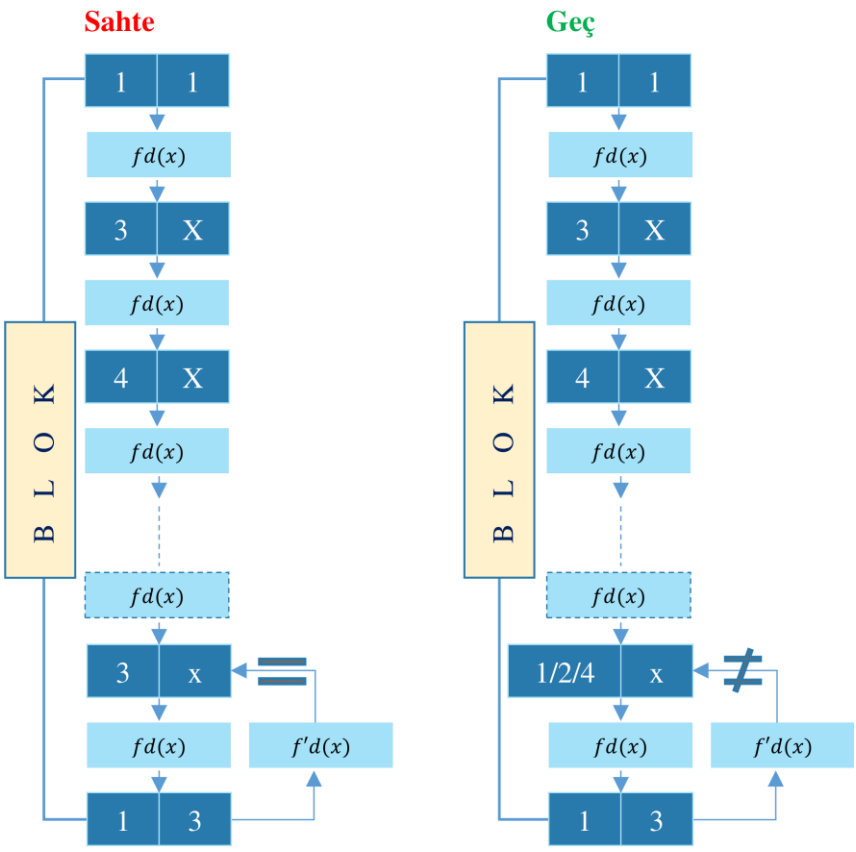
İYHF ideal çalışma durumuna ulaşana kadar bu soruna geçici çözüm "kripto ham verisi" ile şu şekilde sağlanır. Şifreleme sürecinde, "sahte" veri içeren bloklar $(0)_2$ ve "gerçek" veri içeren bloklar $(1)_2$ ile haritalanır (eşlenir). Bu harita, şifre çözme sürecinde kullanılır. Böylece ters fonksiyonlarda herhangi bir hata yapılmamış ve kriptografik veriler çözülmüş olur.

Şekil 4.17, başlangıç türü 1.1, ara türler 2/3/4.x ve bitiş türü 1.3 olan blokların davranışını göstermektedir. Bu bloklarda veri yükleme koşulları yerine getirilmemiştir. Bu nedenle şifreleme ve hash kodu oluşturma sürecinde bloklarda veri değiştirme fonksiyonu kullanılmıştır. Blok, iteratif olarak uygulanan veri değiştirme fonksiyonunun çıkışındaki 1.x türündeki ilk veri bulunduğunda sona erer.



Şekil 4.17 Tür 1.1-2/3/4.x-1.x blokları için kuralların ve davranışların genel gösterimi

Şekil 4.18, her iki bloğun son verilerinin 1.3 türünde olduğunu ve iki bloktan birinin "sahte" veri içeremeyeceğini, diğer bloğun ise "sahte" veri içerebileceğini göstermektedir.

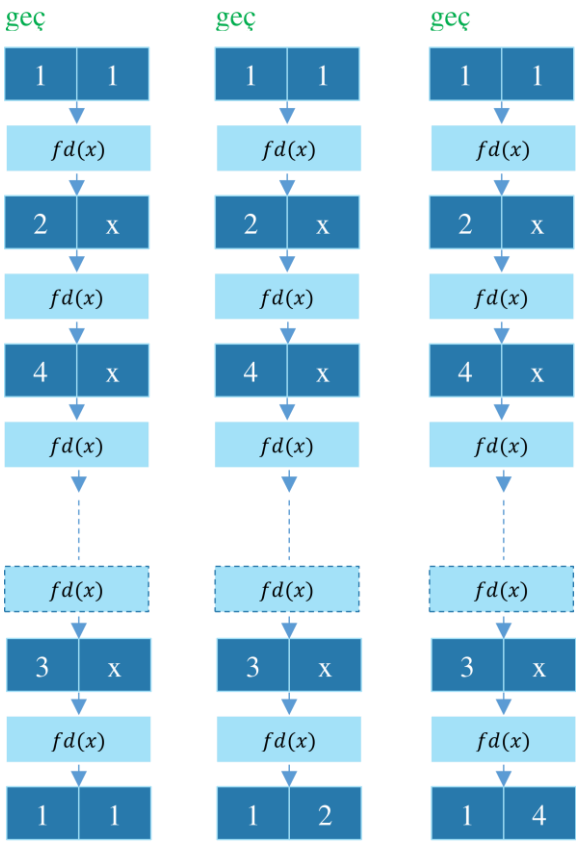


Şekil 4.18 Tür 1.1-2/3/4.x-1.3 bloklarının davranışına ilişkin örnekler

"Sahte" veri içermeyen bloğun son verisinden önce 1/2/4.x veri türlerinden birine sahip olduğu görülebilir. Son veriye ters değişim fonksiyonu uygulandığında 1/2/4.x türü 3.x türüne eşit olamaz, dolayısıyla bu blokta "sahte" veri yoktur. Şekil 4.18'de soldaki bloğun "sahte" veri olması mümkündür. Bu blok, üç koşuldan ilk ikisini yerine getirmektedir. Birincisi bloğun son verisinin 1.3 türünde olması, ikincisi ise ters değişim fonksiyonu bloğun son verisine uygulandığında fonksiyonun çıkışındaki verinin 3.x türünde olmasıdır. Eğer 1.3 türündeki verinin ondalık kısmına yüklenen kontrol verisi de koşulu sağlıyorsa, 1.3 türündeki veri "sahte" niteliğinde olur. Kontrol verisi ondalık verinin ondalık kısmına indirgenme miktarına göre $(1)_2$ bit dizileri şeklinde eklenir. Şifre çözme sürecinde elde edilen tüm kontrol bitleri $(1)_2$ bitlerinden oluşuyorsa kontrol bitlerine

ilişkin koşul sağlanmış olur. Bu durumda veri “gerçek” olmadığı halde "gerçek" olarak değerlendirilir. Örneğin, 1.3 türündeki verinin ondalık kısmının indirgenme miktarı 3 bit ise, kontrol verisi $(111)_2$ olmalıdır. Bunu sağlamayan 3 bitlik kontrol verileri $(000)_2$, $(001)_2$, $(010)_2$, $(011)_2$, $(100)_2$, $(101)_2$, ve $(011)_2$ biçimlerinde olabilir. Şifre çözme sürecinde bu kontrol verilerinden herhangi birine erişilirse, verilerin "gerçek" olmadığı ortaya çıkar. Ancak, kontrol verisi olarak $(111)_2$ değeri elde edilirse, tür 1.3 “sahte” olarak kabul edilir.

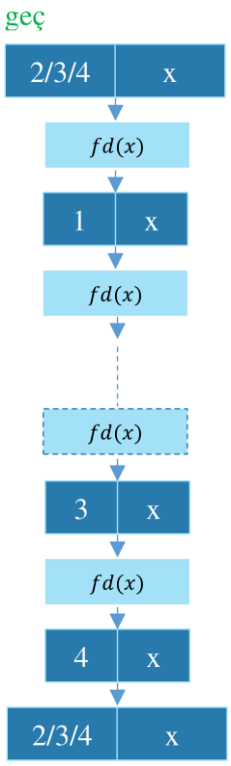
Şekil 4.19'da gösterilen blokların son veri tipleri 1.1, 1.2 ve 1.4 olarak verilmiştir. Bu blokların son veri türleri 1.3 türünde olmadığı için ikinci koşul karşılanmaz. Dolayısıyla bu blok türünde "sahte" veri oluşmaz. İkinci koşul yerine getirilmediği için üçüncü koşulun doğrulanmasına gerek yoktur.



Şekil 4.19 1.1-2/3/4.x- 1.1/2/4 olan bloklar için tüm davranışların örnek gösterimleri

4.5.2.3 Blok başlangıç türü : 2.x/3.x/4.x – Blok bitiş türü : 2.x/3.x/4.x

Blok başlangıç türü 2/3/4.x olan bloklar en çok üretilen bloklardır. Koşulları karşılamadıkları için "sahte" veya "gerçek" veri içermezler. Bu tür bloklarda uygulanan tüm fonksiyonlar veri değiştirme fonksiyonlarıdır. Benzer şekilde şifre çözme sürecindeki tüm fonksiyonlar da ters veri değiştirme fonksiyonlarıdır. Fonksiyonlarda belirsizlik olmadığı için bloğun tamamı hatasız çözülür. Başlangıç türü 2/3/4.x olan bir blok 2/3/4.x türünde bir veri ile sonlanmalıdır. Bu blok türüne bir örnek Şekil 4.20'de gösterilmiştir.



Şekil 4.20 2/3/4.x-2/3/4.x blok türleri için kural ve davranışların örnek gösterimi

4.5.3 Şifreli hash kodu üretme süreci iş akış diyagramı ve açıklamaları

İYHF'nin şifreleme ve hash kodu oluşturma sürecinin blok diyagramı Şekil 4.21'de gösterilmiş olup kullanılan algoritma Algoritma 1'de verilmiştir. Süreç, rastgele seçilen ondalık başlangıç verisine bir değiştirme fonksiyonunun uygulanması ile başlar. Fonksiyonun çıkışındaki veriler, oluşturulacak ilk bloğun başlangıç verileridir. Başlangıç

verisinin türü, verinin bloğa eklenip eklenemeyeceğini, uygulanacak fonksiyonu, blok uzunluğunu ve bloğun hangi veri türüyle sonlanacağını belirler. Başlangıç verisi üç farklı biçimde olabilir: Bunlar 1.1, 1.2/1.3/1.4 ve 2.x/3.x/4.x veri türleridir. Bu türlerden sadece başlangıç veri türü 1.1 olan bloklara veri eklenebilir. Modelin şifreleme ve hash kodu üretme sürecinde oluşabilecek tüm durumları, Şekil 4.21'deki blok diyagrama göre açıklanmıştır.

4.5.3.1 Akış 1

Blok başlangıç veri türü 1.1 ve tüm koşulların sağlanması: Kontrol edilecek ilk veri türü 1.1'dir. Blok başlangıç verisi 1.1 türündeyse, indirgeme fonksiyonu başlangıç verisinin hem tamsayı hem de ondalık kısımlarına uygulanır. Düz metin bitleri ondalık verinin tamsayı kısmına eklenirken, kontrol verileri ondalık kısma eklenir. Süreç ikinci ve üçüncü veri yükleme koşullarının doğrulanmasıyla devam eder. Bu koşullar sağlanırsa, blok bir haritalama işlemi ile sonlandırılır. Blok veri eklenmesiyle sonlandığı için "gerçek" olarak kabul edilir ve $(1)_2$ verisi haritaya eklenir. Daha sonra, iş akışı Bağlantı 1'e yönlendirilir. Bir sonraki bloğun başlangıç verilerini elde etmek için bloğun bitiş verilerine veri değiştirme fonksiyonu uygulanır.

4.5.3.2 Akış 2 ve akış 2.1

Bu iş akışları, blok başlangıç veri türü 1.1 ve koşullara uymama durumunu tanımlar. İlk olarak, blok başlangıç verisine indirgeme fonksiyonu uygulanır. Ardından veri yükleme koşulları kontrol edilir. Koşullardan biri yerine getirilmezse, ilk blok verilerine uygulanan indirgeme fonksiyonu iptal edilir ve blok başlangıç verisine değiştirme fonksiyonu uygulanır (Akış 2). Veri değiştirme fonksiyonunun çıktısı 1.x türünde ise blok sonlandırılır. Akış 1 numaralı bağlantıdan devam eder. Veri değiştirme fonksiyonunun çıktısı 1.x türünde değilse blok sonlandırılmaz ve akış bağlantı 2'ye yönlendirilir (Akış 2.1).

Veri değiştirme fonksiyonunun uygulanması 1.x türünde veri elde edilene kadar devam eder. Eğer 1.x türünde veri elde edilirse blok sonlandırılır. Bir blok bu şekilde sonlandırılırsa, iki farklı durum ortaya çıkar. Blok 1.1, 1.2 veya 1.4 türlerinden biriyle sonlanırsa, iş akışı bir sonraki bloğun başlangıç verilerini almak için bağlantı 1'e yönlendirilir ve bu noktadan devam eder. Blok 1.3 türü ile sonlanırsa, blok bitiş verisinin "sahte" olup olmadığı kontrol edilir. Tüm koşullar yerine getirilirse, blok "sahte" olarak nitelendirilir ve haritaya $(0)_2$ verisi eklenir. "Sahte veri" koşulları yerine getirilmezse, iş akışı bağlantı 1'den devam eder.

4.5.3.3 Akış 3 ve akış 3.1

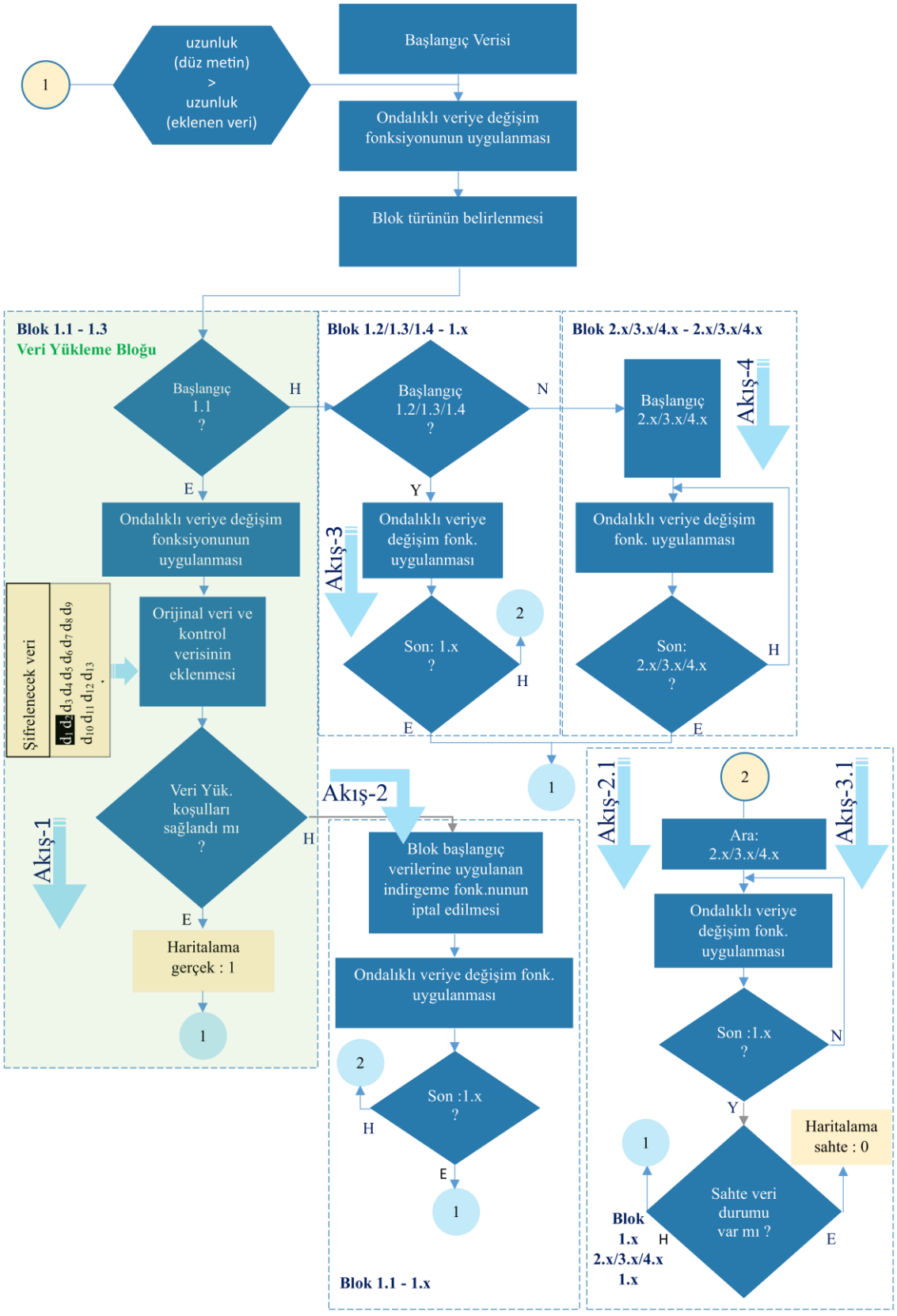
Akış-3 ve akış 3.1 - Blok başlangıç veri türü 1.2/1.3/1.4 : Bloğun başlangıç verileri 1.2/1.3/1.4 türündeyse, verilerin yüklenmesi için ilk koşul yerine getirilmemiş olur. Akış 2 numaralı bağlantı ile devam eder. Bu durumda, blok başlangıç verisine veri değiştirme fonksiyonu uygulanır. İlk uygulanan veri değiştirme fonksiyonunun çıktısı 1.x türünde olursa blok sonlandırılır (Akış 3). Blok bu şekilde sonlanırsa, "sahte veri" yoktur. Bunun nedeni üçüncü koşulun yerine getirilmemesidir. Blok sonlandırıldıktan sonra akış Bağlantı 1'e yönlendirilir ve bu noktadan devam eder.

Bloğa uygulanan ilk veri değiştirme fonksiyonunun çıktısı 2.x/3.x/4.x türünde olduğunda blok sonlandırılmaz. Bloğu sonlandırmak için veri değiştirme fonksiyonu 1.x türünde bir veri elde edilene kadar iteratif olarak uygulanır. Fonksiyon çıkışında 1.1/1.2/1.4 veri tiplerinden biri elde edilirse blok sonlandırılır. Bu durumda "sahte" veri oluşmaz. Ancak iterasyon sonunda 1.3 tipi elde edilirse "sahte" veri durumu kontrol edilir. Eğer "sahte veri" koşulları sağlanıyorsa, süreç haritalama ile devam eder. Bu aşamada haritaya "sahte" durumunu gösteren $(0)_2$ verisi eklenir. "Sahte" veri koşulu yerine getirilmezse, akış bağlantı 1'den sonra devam eder.

4.5.3.4 Akış 4

Blok başlangıç veri türü 2.x/3.x/4.x ve blok bitiş veri türü 2.x/3.x/4.x durumu meydana geldiğinde akış 4 yürütülür. Blok başlangıç türü 1.x değilse, süreç akış 4'e yönlendirilir. Bu durumda, bloğun başlangıç veri türü 2.x/3.x veya 4.x veri türlerinden biridir. Süreç, veri değiştirme fonksiyonunun iteratif olarak uygulanmasıyla devam eder. Bloğu sonlandırmanın koşulu, 2.x/3.x veya 4.x veri türlerinden biriyle ilk kez karşılaşılmasıdır. Bu bloklarda "sahte" veri olasılığı yoktur. Blok sona erdiğinde, akış Bağlantı 1'den devam eder.

İYHF süreci, orijinal metnin son bitinin oluşturulan sayı örüntüsüne yüklenmesiyle tamamlanır. Oluşturulan sayı örüntüsünün son ondalık verisi şifrelenmiş veri ve hash kodudur.



Şekil 4.21 İYHF şifreleme ve hash kodu oluşturma sürecinin blok akış diyagramı

Algoritma 1 İYHF şifreleme ve hash kodu oluşturma süreci

***Input*:** *Idd : Initial decimal data*

pt: Plain text

Output: *ch: Chiphertext and hashcode*

Definitions:

f_r: *Reduction function*

f_c: *Change function*

f_t: *Type determination function*

rd: Reduced data

cd: Changed data

cdt: Control data

dlc: data loading conditions

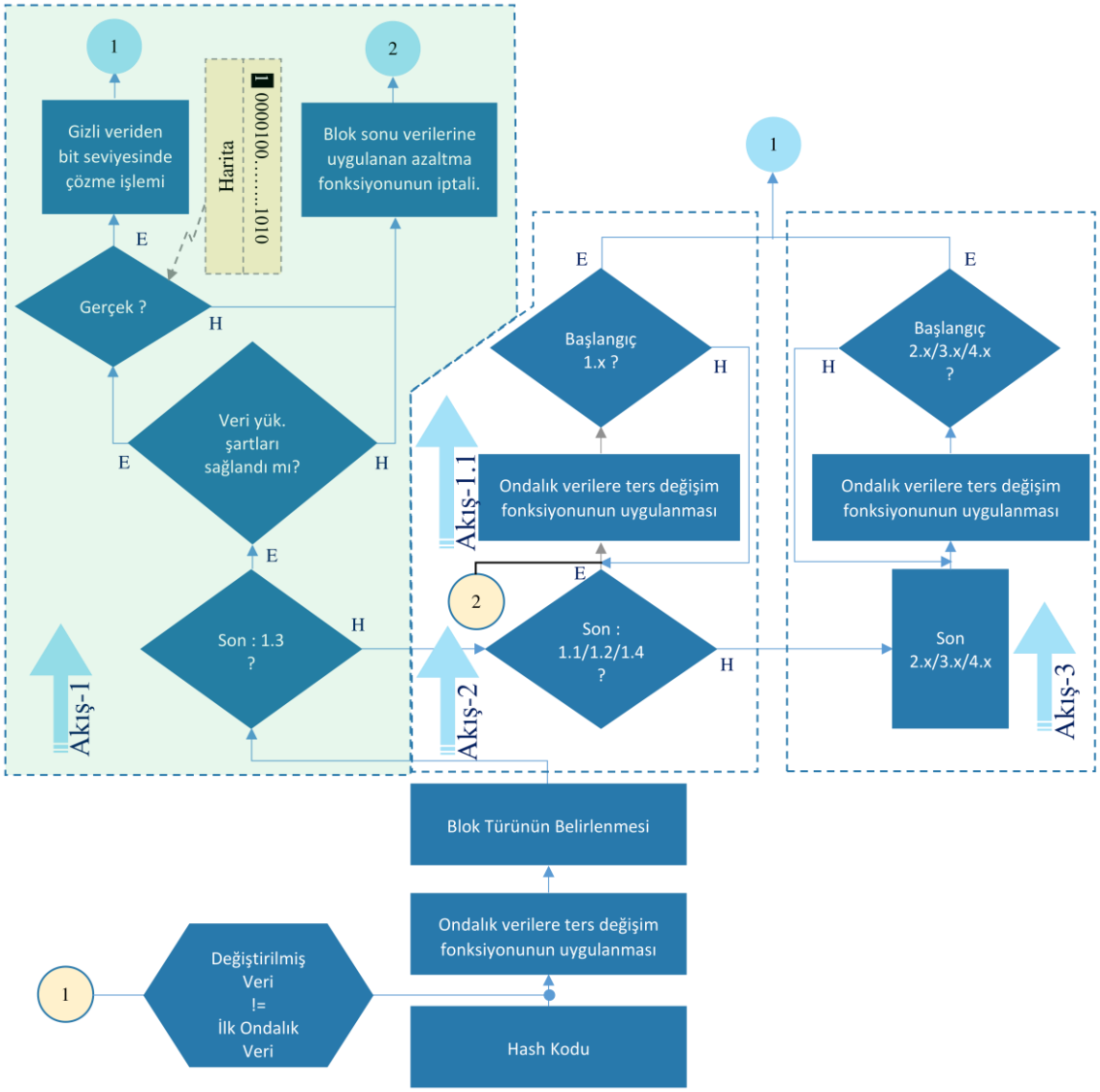
fds:fake data status

1. Set Initial Parameters : idd, pt
2. while(length(pt)>length(added data)) do
3. cd ← $f_c(cd)$
4. Flow1 operations
5. if $f_t(cd)$==1.1 then
6. rd ← $f_r(cd)$
7. Add pt and ctd bits in rd
8. if (dlc==True) then
9. Mapping process “real”
10. Else
11. Flow2 and Flow 2.1 operations
12. cd ← $f_c(cd)$
13. if ($f_t(cd)! = 1.x$) then
14. while($f_t(cd)$!=1.x) do
15. cd ← $f_c(cd)$
16. End
17. if (fds==True) then
18. Mapping process “fake”
19. End

20. End
21. End
22. Flow3 operations
23. else if ($f_t(cd)$==1.2 or $f_t(cd)$==1.3 or $f_t(cd)$==1.4) then
24. cd ← $f_c(cd)$
25. if ($f_t(cd)$!=1.x) then
26. Flow3.1 operations
27. while(cd!=1.x) do
28. cd ← $f_c(cd)$
29. End
30. if (fds==True) then
31. Mapping process “fake”
32. End
33. End
34. Flow4 operations
35. else if ($f_t(cd)$==2.x or $f_t(cd)$==3.x or $f_t(cd)$==4.x) then
36. cd ← $f_c(cd)$
37. while($f_t(cd)$!=2.x or $f_t(cd)$!=3.x or $f_t(cd)$!=4.x) do
38. cd ← $f_c(cd)$
39. End
40. End
41. End
42. ch←cd

4.6 Şifre Çözme Süreci

Şifre çözme sürecinin akış şeması Şekil 4.22'de ve algoritması, Algoritma 2'de gösterilmiştir.



Şekil 4.22 İYHF şifre çözme işleminin blok akış diyagramı

Akış yönü yukarı doğrudur. Şifre çözme sürecinde işlenen ilk ondalık veri, şifreleme sırasında oluşturulan sayı örüntüsünün son ondalık verisidir. Bu süreçte, değiştirme ve azaltma fonksiyonlarının tersi kullanılır. Amaç, şifreleme sürecinde oluşturulan sayı örüntüsünü yeniden elde etmektir.

Algoritma 2 TWHF şifre çözme işlemi

***Input*:** e*dd : End decimal data*

Idd : Initial decimal data

Output: *pt: Plain Text*

Definitions:

f_r^{-1}: *Inverse reduction function*

f_c^{-1}: *Inverse change function*

f_t*: Type determination function*

cd: Changed data

dlc: data loading conditions

rs: Real Status

1. Set Initial Parameters : edd
2. Idd←edd
3. cd ← $f_c^{-1}(edd)$
4. while(cd!=Idd)) do
5. cd ← $f_c^{-1}(edd)$
6. Flow1 and Flow 1.1 operations
7. if ($f_t(cd)$==1.3) then
8. if (dlc==True) then
9. if (rs==True) then
10. Add crypto data bit to pt
11. Else
12. cd ← $f_c^{-1}(cd)$
13. while ($f_t(cd)$!=1.x)
14. cd ← $f_c^{-1}(cd)$
15. End

End
Else
cd ← $f_c^{-1}(cd)$
while ($f_t(cd)$!=1.x)
cd ← $f_c^{-1}(cd)$
End
end
Flow2 operations
else if ($f_t(cd)$==1.1 or $f_t(cd)$==1.2 or $f_t(cd)$==1.4) then
cd ← $f_c^{-1}(cd)$
while ($f_t(cd)$!=1.x)
cd ← $f_c^{-1}(cd)$
End
Flow3 operations
else if ($f_t(cd)$==2.x or $f_t(cd)$==3.x or $f_t(cd)$==4.x) then
cd ← $f_c^{-1}(cd)$
while ($f_t(cd)$!=2.x or $f_t(cd)$!=3.x or $f_t(cd)$!=4.x)
cd ← $f_c^{-1}(cd)$
End
End
End
write pt

4.6.1 İş akış diyagramı ve açıklamaları

Önerilen modelin şifre çözme süreci blok diyagramında oluşabilecek tüm durumlar ve açıklamaları aşağıda listelenmiştir.

4.6.1.1 Akış 1 (Blok 1.x-1.3) ya da Akış 1.1 (Blok 1.x-1.3/Blok 1.x-2.x/3.x/4.x-1.3)

Şifre çözme süreci, şifreleme ve hash kodu oluşturma işlemlerinin son verilerini başlangıç verisi olarak kullanır. Süreç bu veriye ilk ters veri değiştirme fonksiyonunun uygulanmasıyla başlar. Değişen verinin türünün 1.3 olup olmadığı kontrol edilir. Bu koşul sağlanmışsa diğer koşulların da sağlanıp sağlanmadığı kontrol edilir. İlk koşul, ters indirgeme fonksiyonu 1.3 türündeki verilere uygulandığında, ortaya çıkan başlangıç blok verilerinin 1.1 türünde olmasıdır. Diğer koşul ise, ters değişim fonksiyonu 1.3 türündeki veriye uygulandığında, çıktının 3.x türünde olması gerektiğidir. Ayrıca, tür 1.3 verisinin ondalık kısmındaki kontrol verisi tamamen $(1)_2$ bit dizisinden oluşmalıdır. Eğer tüm koşullar sağlanıyorsa, verinin "gerçek/sahte" durumu haritaya bakılarak belirlenir. Haritada, sıradaki veri $(1)_2$ ise veri "gerçek" olarak değerlendirilir ve tür 1.3 verisinin tamsayı kısmına yüklenen düz metin bitleri ayrıştırılır. Böylece şifreli hash kodu üretme sürecinde bloğa yüklenen düz metinden birkaç bitlik veri elde edilmiş olur.

Ancak, haritada sıradaki veri $(0)_2$ ise, veriler "sahte" olarak kabul edilir. Tür 1.3 verisine uygulanan ters indirgeme fonksiyonu iptal edilir ve süreç bağlantı 2'den devam eder (Akış-1.1). Bu noktadan itibaren, ters veri değiştirme fonksiyonu 1.x türü verilere ulaşana kadar iteratif olarak uygulanır. 1.x türünden veriler elde edilirse, blok sonlandırılır. Bir sonraki işlem bloğu için süreç bağlantı 1'den devam eder.

Şifre çözme yönüne göre, bloğun son verisi veri yükleme koşullarından birini karşılamıyorsa, işlem 2. bağlantıdan devam eder (Akış-1.1). Tür 1.3 verileri için, ters indirgeme fonksiyonu yerine ters veri değiştirme fonksiyonu iteratif olarak uygulanır. Tür 1.x verisine ulaşıldığında iterasyon sona erer. Bir sonraki işlem bloğu için süreç bağlantı 1'den devam eder.

4.6.1.2 Akış 2 (Blok 1.x-1.1/1.2/1.4 ya da Blok 1.x-2.x/3.x/4.x-1.1/1.2/1.4)

Şifre çözme yönüne göre, bloğun son verisi 1.3 türünde değilse, verinin 1.1, 1.2 veya 1.4 türünde olup olmadığı kontrol edilir. Bu türlerden biri için koşul yerine getirilirse, ters veri değiştirme fonksiyonunun uygulandığı iterasyon başlar. Amaç blok başlangıç verisine ulaşmaktır. İterasyon, 1.x türünde veri bulunduğunda sona erer ve şifre çözme yönüne göre blok sonlandırılır. Süreç Bağlantı 1'den devam eder.

4.6.1.3 Akış 3 (Blok 2.x/3.x/4.x – 2.x/3.x/4.x)

Şifre çözme yönüne göre bloğun son verisi 2.x/3.x/4.x türünde ise işlem ters veri değiştirme fonksiyonunun uygulandığı iterasyon ile devam eder. Ters veri değiştirme fonksiyonlarının çıkışında elde edilecek 2.x/3.x/4.x türündeki ilk veri bloğu sonlandırır. Blok bitiminden sonra iş akışı, 1 numaralı bağlantıdan devam eder.

4.7 İYHF Yöntemi Blok Türlerine Dayalı Bir Örnek

Çizelge 4.6'da İYHF ile $(0011)_2$ verisinin şifreli hash kodunun oluşturulması ve çözülmesine ilişkin bir örnek blok türlerine göre gösterilmektedir.

Çizelge 4.6 İYHF ile örnek bir verinin şifrelenmiş hash kodunun üretilmesi ve çözülmesi örneği

<table>
<tr><th>İterasyon</th><th>Akış</th><th>Blok Türleri</th><th>S/G/GÇ</th><th>EV</th><th>KHV</th><th>İterasyon</th><th>Akış</th><th>Blok Türleri</th><th>S\G\GÇ</th><th>EV</th><th>KHV</th></tr>
<tr><td rowspan="2">1.
2.</td><td rowspan="2">Akış 1
VY</td><td>1.1 (İF)
(TDF)
1.3 (DF)
(TİF)</td><td rowspan="2">G</td><td rowspan="2">0</td><td rowspan="2">1</td><td>28.</td><td rowspan="3">Akış 4</td><td>2.4 (DF)
(TDF)</td><td rowspan="3">GÇ</td><td rowspan="3">-</td><td rowspan="3">-</td></tr>
<tr><td>29.</td><td>1.3 (DF)
(TDF)</td></tr>
<tr><td>3.</td><td>Akış 2</td><td>1.1(~~İF~~)
(DF)
(TDF)</td><td>GÇ</td><td>-</td><td>-</td><td>30.</td><td>4.2 (DF)
(TDF)</td></tr>
</table>

Çizelge 4.6 İYHF ile örnek bir verinin şifrelenmiş hash kodunun üretilmesi ve çözülmesi örneği (devam)

İterasyon	Akış	Blok Türleri	S/G/GÇ	EV	KHV	İterasyon	Akış	Blok Türleri	S\G\GÇ	EV	KHV
4.		1.3(DF) (TDF)				**31.**	**Akış 4**	3.3 (DF) (TDF)	GÇ	-	-
5.	**Akış 2.1** Durum 1	1.1(~~İF~~) (DF) (TDF)	GÇ	-	-	**32.**		2.4 (DF) (TDF)			
6.		3.1(DF) (TDF)				**33.**	**Akış 3.1** Durum 2	1.3 (DF) (TDF)	**S/** GÇ	-	**0**
7.		1.1(DF) (TDF)				**34.**		2.3 (DF) (TDF)			
8.	**Akış 2.1** Durum 2	1.1(~~İF~~) (DF) (TDF)	S/ **GÇ**	-	-	**35.**		3.1 (DF) (TDF)			
9.		3.3 (DF) (TDF)				**36.**		1.3 (DF) (TDF)			
10.		2.4 (DF) (TDF)				**37.**	**Akış 2.1** Durum 1	1.1(~~İF~~) (DF) (TDF)	GÇ	-	-
11.		4.1 (DF) (TDF)				**38.**		2.3 (DF) (TDF)			
12.		1.3 (DF) (TDF)				**39.**		3.1 (DF) (TDF)			
13.	**Akış 3**	1.2 (DF) (TDF)	GÇ	-	-	**40.**		3.4 (DF) (TDF)			
14.		1.3 (DF) (TDF)				**41.**		3.2 (DF) (TDF)			
15.	**Akış 3.1** Durum 1	1.3 (DF) (TDF)	GÇ	-	-	**42.**		4.1 (DF) (TDF)			
16.		4.2 (DF) (TDF)				**43.**		2.1 (DF) (TDF)			
17.		3.1 (DF) (TDF)				**44.**		1.1 (DF) (TDF)			

Çizelge 4.6 İYHF ile örnek bir verinin şifrelenmiş hash kodunun üretilmesi ve çözülmesi örneği (devam)

İterasyon	Akış	Blok Türleri	S/G/GÇ	EV	KHV	İterasyon	Akış	Blok Türleri	S\G\GÇ		
18.		2.3 (DF) (TDF)				**45.**	**Akış 1 VY**	1.1 (İF) (TDF)	**G**	**01**	**1**
19.		4.1 (DF) (TDF)				**46.**		**1.3 (DF) (TİF)**			
20.		1.2 (DF) (TDF)				**47.**	**Akış 2**	1.1(~~İF~~) (DF) (TDF)	GÇ	-	-
21.		1.4 (DF) (TDF)	**S/** GÇ	-	**0**	**48.**		1.4(DF) (TDF)			
22.		2.1 (DF) (TDF)				**49.**	**Akış 3**	1.4 (DF) (TDF)	GÇ	-	-
23.		3.4 (DF)				**50.**		1.1 (DF) (TDF)			
24.	**Akış 3.1** Durum 2	3.1(DF)				**51.**	Akış 4	4.3 (DF) (TDF)	GÇ	-	-
25.		2.3 (DF)				**52.**		3.1 (DF) (TDF)			
26.		3.2 (DF)				**53.**	**Akış 1 VY**	1.1 (İF) (TDF)	**G**	**1**	**1**
27.		1.3 (DF)				**54.**		**1.3 (DF) (TİF)**			

Çizelgede kullanılan İF/DF şifreli hash kodu oluşturma işleminde, TİF/TDF ise şifre çözme işleminde kullanılmaktadır. Önerilen yöntem iki yönlü olduğu için bu fonksiyonlar birlikte verilmiştir. Örnek incelenirken şifreli hash kod üretimi için İF/DF, şifre çözme için TİF/TDF dikkate alınmıştır.

Örüntüyü oluşturan ondalık sayılar "tamsayı kısmı türü.ondalıklı kısmı türü" formatında verilmiştir. Böylece örüntüyü ve akışı takip etmek daha kolay olmuştur. Tablodaki her akış aynı zamanda bir blok tipini temsil etmektedir. Blokların başlangıcı ve bitişi

arasındaki ondalıklı veri türleri, çalışma kapsamında sınıflandırılan blok türlerine uygun biçimde rassal olarak seçilmiştir. Örneğin, 5-7 arası iterasyonlar 1.1-3.1-1.1 "tür sırası" ile verilmiştir. Ancak "tür sırası" 1.1-4.2-1.1 veya 1.1-2.3-1.1 vb. olabilmektedir. Çizelgedeki akışlar, yöntemin anlaşılmasını kolaylaştıracak bir sırada seçilmiştir. Aşağıda sadece önemli iterasyon aralıkları açıklanmıştır.

1-2 iterasyonlarında, orijinal $(0011)_2$ verisinin MSB tarafındaki $(0)_2$ verisi 1.3 türündeki verinin tamsayı kısmına yüklenmiştir. Bu blokta tüm koşulların sağlandığı ve indirgeme miktarının bir bit olduğu anlaşılmaktadır. Ayrıca "kripto ham verisi" için "1" değeri ile haritalama işlemi yapılmıştır.

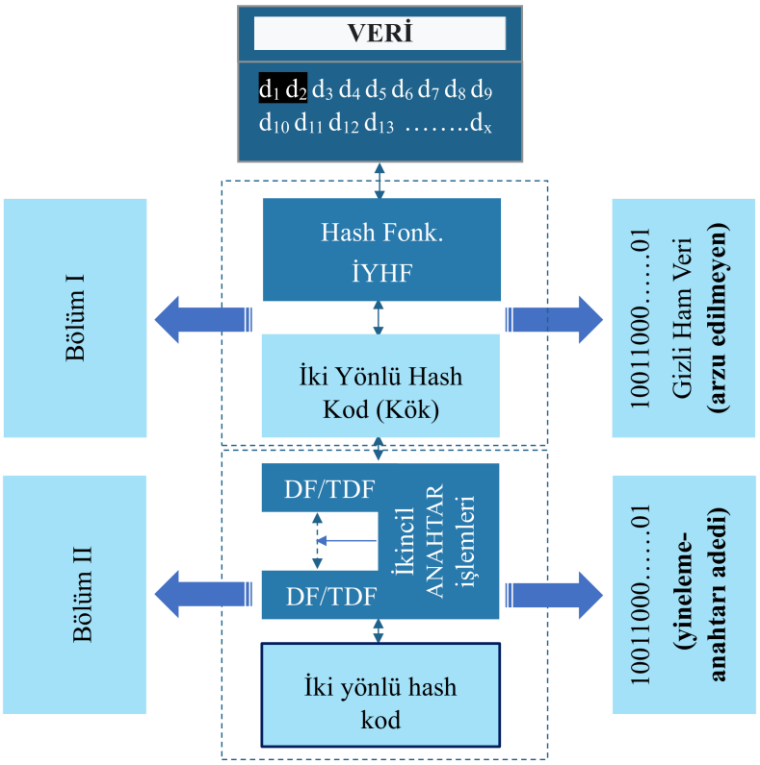
3. ve 4. iterasyonlarda tür 1.1 ile başlayan blokta koşullar sağlanmadığı için indirgeme fonksiyonu yerine değiştirme fonksiyonu uygulanmıştır. 8-12 iterasyon aralığında, 12. iterasyondaki ondalık verinin tür 1.3 olması nedeniyle "sahte" veri olma ihtimali bulunmaktadır. Bu örnekte "sahte" veri koşullarının karşılanmadığı kabul edilmiştir. Bu nedenle blok "geç" olarak nitelendirilmiş ve "kripto ham veri" bit dizisine $(0)_2$ eklenmemiştir. 21-27. iterasyonlar aralığında, 27. iterasyon 1.3 türündedir. Bu blokta "sahte" veri olma ihtimali vardır. Burada "sahte" veri olma koşullarının sağlandığı kabul edilmiş ve "kripto ham veri" için $(0)_2$ verisi ile haritalama yapılmıştır. Benzer şekilde, 33-36. iterasyon aralığında, 36. iterasyondaki 1.3 türündeki veri "sahte" olarak kabul edilmiş ve $(0)_2$ ile haritalanmıştır. 45. ve 46. iterasyonlardaki verilerin koşulları sağladığı ve indirgeme miktarının iki bit olduğu varsayılmıştır. Bu nedenle, orijinal verinin bir sonraki verisi $(01)_2$ bu bloğa yüklenmiş ve $(1)_2$ ile haritalanmıştır. Örnekteki son veri yükleme bloğu 53. ve 54. iterasyonlarda yer almaktadır. Bu blok için indirgeme miktarı bir bit olarak kabul edilmiştir. Dolayısıyla orijinal verinin son biti olan $(1)_2$ buraya yüklenmiş ve $(1)_2$ ile haritalanmıştır. Son iterasyonla birlikte "kripto ham verisi" $(11001)_2$ olmuştur. Açıklanan iterasyonlar dışındaki akışlarda veri gömme ve "sahte" veri olasılığı yoktur.

Çizelgenin 1. iterasyonundan 54. iterasyonuna kadar olan akış, orijinal $(0011)_2$ verisinin oluşturulan sayı modeline yüklenmesini kapsar. 54'üncü iterasyondaki ondalık veri şifrelenmiş hash kodu (kök) olarak kabul edilir. Bu süreçte istenmeyen "kripto ham veri" bit dizisi de ortaya çıkmıştır.

Şifre çözme süreci de Çizelge 4.6 kullanılarak açıklanabilir. Şifre çözme süreci son iterasyondan (iterasyon 54) başlar. İterasyon yönü şifreli hash kodu oluşturma sürecinin tersidir. Süreç boyunca değiştirme ve indirgeme fonksiyonlarının tersi alınır ve her zaman bir önceki iterasyondaki ondalık sayı elde edilir. Bu süreçte, haritalanan "kripto ham veri" takip edilerek tüm bloklar doğru bir şekilde çözülür. 1. iterasyona ulaşıldığında orijinal veri elde edilmiş olur.

Çizelge 4.6'daki verilere göre, örnek bir şifre çözme sürecinin birkaç iterasyonu aşağıdaki gibi açıklanabilir. Şifre çözme süreci 54. iterasyonda 1.3 türündeki veri ile başlar. Bu veri 1.3 türünde olduğundan dolayı "gerçek" veri içerme olasılığı vardır. Bunu belirlemek için KHV değeri kontrol edilir. KHV değerinin $(1)_2$ olması, 1.3 türündeki verinin orijinal verinin birkaç bitini içerdiğini gösterir. Bu durumda orijinal veri ayrıştırılır. Örüntüde, 1.3 türündeki veriden önce gelen veriyi bulmak için TİF uygulanır. Fonksiyonun çıktısı 53. iterasyondaki 1.1 türündeki veridir. Erişilen verinin 1.1 türünde olması bir bloğun başlangıcı olduğunu gösterir. Süreç 1.1 türündeki verilere TDF uygulanması ile devam eder. Fonksiyonun çıktısı 52. iterasyonda 3.1 türündeki veridir. Süreç, 1. iterasyondaki veriye ulaşana kadar devam eder.

4.8 İYHF ve Tek Yönlü Hash Fonksiyonları Güvenlik Özellikleri



Şekil 4.23 İYHF'nin tek yönlü fonksiyonların özelliklerine sahip olabilmesi için gerekli model

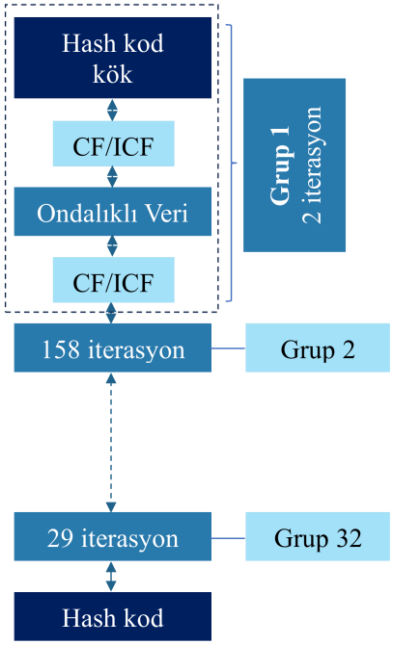
İYHF metodolojisinin veri çözme aşaması ortalama %0,17'lik bir belirsizlikle sonuçlanmıştır. Belirsizlik, örüntünün bir önceki ondalık verisine ulaşmak için aktif ondalık veriye hangi fonksiyonun uygulanacağına karar verememe durumudur. Bunlar ters indirgeme ve ters veri değiştirme fonksiyonlarıdır. Bu duruma geçici bir çözüm olarak "kripto ham veri" adı verilen bir veri önerilmiştir. Bu veri bir rehber olarak kullanıldığında, veri şifre çözme süreci başarıyla tamamlanır. Bir anlamda, belirsizliği ifade eden "kripto ham veri" çalışmada istem dışı ortaya çıkan arzu edilmeyen bir durumdur. İYHF'nin genel işleyişini gösteren Şekil 4.23'deki blok diyagramda bu durum "Bölüm I" olarak verilmiştir.

İYHF tek yönlü fonksiyonların güvenlik özelliklerine de sahip olmalıdır. Bu amaçla, Bölüm II olarak Şekil 4.23'de belirtilen süreçte "yineleme anahtarı" ve “ikincil anahtar”

kullanılabilir. "Yineleme anahtarı" yapısı Şekil 4.24'de gösterilmiştir. Bu yapıya ilişkin bir örnek ve açıklamalar aşağıdaki gibidir.

Bu çalışmada anahtar 256 bit uzunluğunda seçilmiş ve 32 gruba bölünmüştür. Yöntemin kullanıldığı uygulamanın ihtiyaçlarına göre anahtar uzunluğu ve grup sayısı değiştirilebilir.

Şekil 4.24'e göre grupların aldığı örnek veriler (2 158 168 178 255 0 254 32 128 130 1 52 45 248 12 30 56 21 78 20 25 128 136 0 0 1 24 12 178 13 17 29)$_{10}$ olarak kabul edilmiştir. Bu değerler iterasyon sayılarıdır. Bu anahtarı bilen bir gönderici kök hash kodundan başlayarak veri değiştirme fonksiyonunu uygular. Fonksiyon, "iteration-key" içindeki gruplarda verilen sayılar kadar tekrar eder. Örnekteki iterasyon sayısına göre kök hash kodundan başlayarak 2431 kez veri değiştirme fonksiyonu uygulanarak iki yönlü hash kodu elde edilir. İterasyon sayısının düşük tutulması metodolojinin uygulama süresini de kısaltacaktır. Kötü niyetli kişiler iterasyon sayılarını tek tek denemeden kaba kuvvet saldırısı ile anahtardaki toplam iterasyon sayısını deneyerek şifrelenmiş hash kodunu çözebilir. Bu durum, her grup iterasyonu sonunda üretilen hash kodu ve aynı uzunluğa sahip ikincil anahtarın XOR işlemine tabi tutulmasıyla engellenebilir. Bu ifade XOR işleminin 32 kez uygulanacağını açıklar. Anahtarın 32 gruba bölünmesi ile toplam iterasyon sayısı düşük tutulmuş ancak kaba kuvvet saldırıları için olasılık sayısı oldukça artırılmıştır. Kök hash kodundan uygulanan bu anahtar mantığı farklı şekillerde de uygulanabilir. Hali hazırda önerilen anahtar İYHF'nin tek yönlü hash fonksiyonlarının güvenlik özelliklerine sahip olmasını sağlamak için yeterlidir.



Şekil 4.24 İterasyon anahtarı ve ikincil anahtar iş akışı blok diyagramı.

Bölüm I'de verilen İYHF uygulandığında ve örüntünün son katmanındaki ondalıklı veri hash kodu olarak kabul edildiğinde, bu veriden orijinal verinin elde edilmesinin önünde hiçbir engel bulunmamaktadır. Ancak çalışmada, İYHF'nin tek yönlü hash fonksiyonlarının güvenlik özelliklerine sahip olması da amaçlandığından dolayı orijinal verinin kök hash kodu elde edildikten sonra örüntünün, yineleme anahtarında tanımlanan iterasyon sayıları kadar devam ettirilmesi gerekmektedir. Bu durumda kök hash kodunun güvenliği sağlanmış olur. Bunun sebebi şifre çözme sürecinin ilerleyebilmesi için kök hash koduna ulaşılması zorunluluğudur. Bu işlemin gerçekleşmesi anahtarların kullanımı gerektirmektedir. Anahtarlar alıcıya önceden güvenli bir iletişim kanalıyla ulaştırılmalıdır.

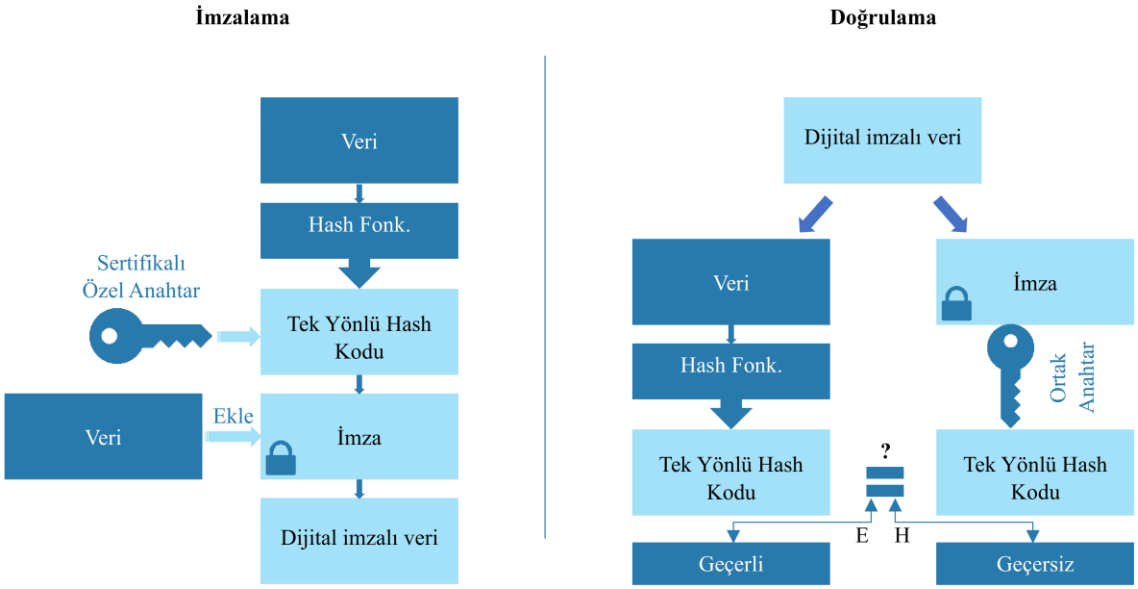
4.9 İYHF Olası Kullanım Alanları

Önerilen metodun olası kullanım alanlarından bazıları aşağıda listelenmiştir. Bu alanlar dijital imza, blok zinciri uygulamaları, steganografi ve veri depolama olarak belirlenmiş ve İYHF'nin bu alanlarla birlikte nasıl kullanılabieceği açıklanmıştır.

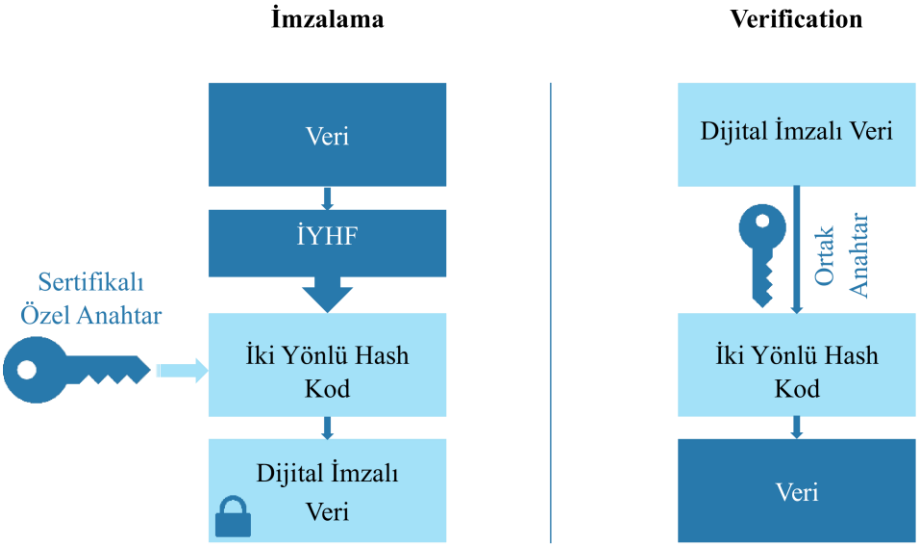
4.9.1 Dijital imza

Şekil 4.25, dijital imza iş akışının tek yönlü hash fonksiyonları ile mevcut kullanımını göstermektedir (Singh vd. 2015). Dijital imzanın amacı veri bütünlüğünü ve kimlik doğrulamasını garanti etmektir. Dijital imza uygulamalarında, orijinal verinin şifrelenmiş hash kodu, orijinal veri ile birlikte alıcıya gönderilir. Alıcı tarafta orijinal veri tekrar hashlenir. Şifrelenmiş hash kodu bir açık anahtar kullanılarak deşifre edilerek hash kodu tekrar elde edilir. Hash kodlarının karşılaştırma sonuçları aynı ise verinin gerçekliği ve bütünlüğü doğrulanmış olur. Kimlik doğrulamayı sağlayan ana işlem, hash kodunun göndericinin özel anahtarı ile şifrelenmesidir. Bu nedenle şifreleme, kimlik doğrulamada önemli bir rol oynar. Hash fonksiyonu daha çok verinin bütünlüğüne odaklanır. Orijinal verinin hash kodu ile alıcıya iletilmesinin bazı dezavantajları olabilir. Bunlardan biri internet bant genişliği kullanımının artması, diğeri ise orijinal verinin gizliliğinin tehlikeye girmesidir. Bu nedenlerden dolayı alıcıya sadece hash kodunun gönderilmesi daha uygundur.

Şekil 4.26, dijital imza iş akışında İYHF yönteminin olası kullanımını göstermektedir. İYHF kullanan bir dijital imza uygulamasında orijinal veriler gönderilmez. Alıcı tarafa yalnızca şifrelenmiş hash kodu gönderilir. Önerilen hash fonksiyonu iki yönlü olduğu için orijinal veri hash kodundan geri alınabilir. Böylece orijinal verilerin gizliliği tamamen garanti altına alınmış olur. Kimlik doğrulama, mevcut uygulamalarda olduğu gibi, hash kodunun gönderenin özel anahtarıyla şifrelenmesi ve açık anahtarla şifresinin çözülmesiyle gerçekleştirilir. İYHF yöntemindeki iterasyon anahtarı ve ikincil anahtar, İYHF'nin kullanıldığı uygulamanın ihtiyaçlarına bağlı olarak isteğe bağlı olarak kullanılabilir ya da kullanılmayabilir.



Şekil 4.25 Dijital imza iş akışında mevcut hash fonksiyonlarının kullanımı

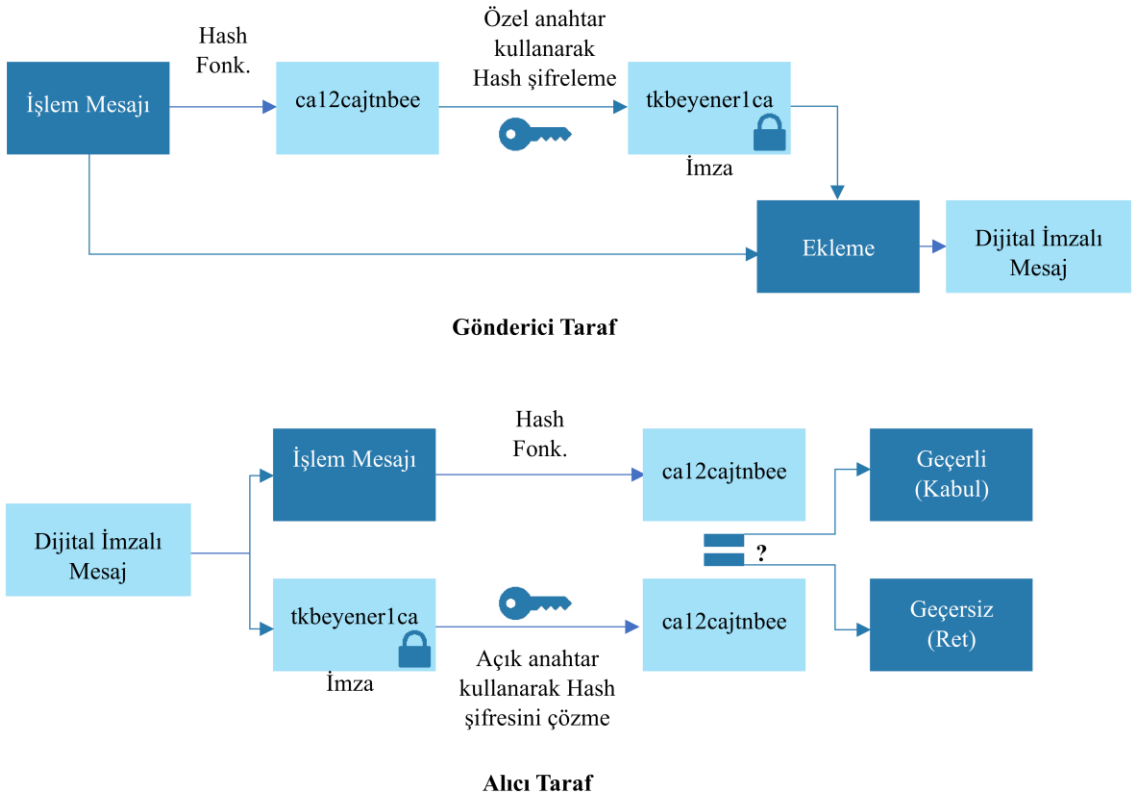


Şekil 4.26 Dijital imza iş akışında İYHF kullanımı

Verinin herhangi bir nedenle değiştirilmiş olarak alıcıya ulaşması durumunda, İYHF'nin matematiksel olarak birbirine bağlı sayı örüntüsü nedeniyle şifresi çözülemez ve veri bütünlüğünün bozulduğu tespit edilir. İYHF'nin veri alışverişinin sık olduğu ve kimlik doğrulamanın gerekli olduğu uygulamalarda ağların daha az bant genişliği kullanmasına katkı sağlayacağı söylenebilir.

4.9.2 Blok zinciri uygulamaları

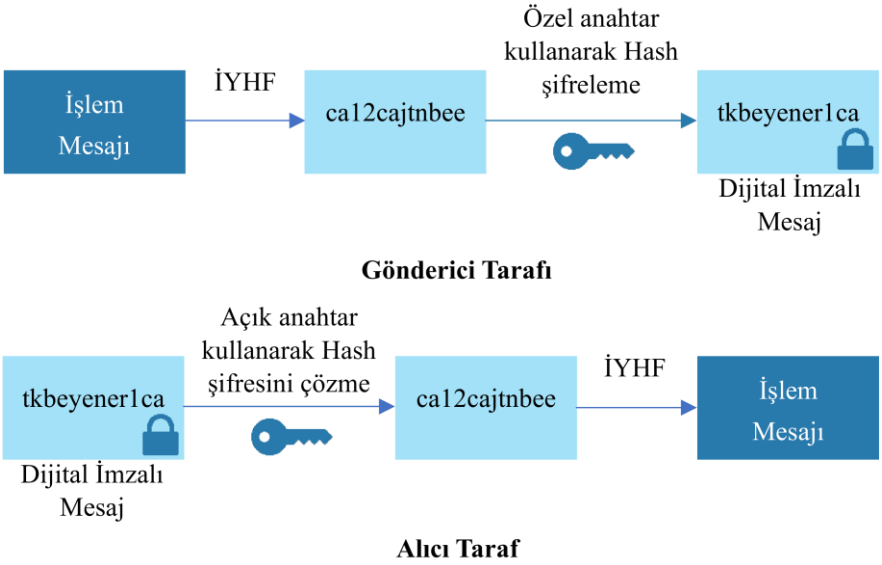
Şekil 4.27, tek yönlü hash fonksiyonlarının bir blok zinciri iş akışında kullanımını göstermektedir (Rajasekaran vd. 2022).



Şekil 4.27 Blok zinciri iş akışında mevcut hash fonksiyonlarının kullanımı

İşlem mesajları blok sistemindeki en küçük yapılardır. Rolleri bilgi ve kayıtları muhafaza etmektir. Blok zinciri uygulamalarında şifrelenmiş hash fonksiyonlarının kullanılmasının amacı, dijital imza uygulamalarında olduğu gibi veri bütünlüğünü ve kimlik doğrulamasını sağlamaktır. Hash fonksiyonu kullanılarak gönderici taraftaki işlem mesajlarının hash kodları elde edilir ve göndericinin özel anahtarı ile şifrelenir. Süreç, orijinal işlem mesajlarının şifrelenmiş hash kodlarına eklenmesi ve alıcıya gönderilmesi ile devam eder. Böylece işlem mesajları dijital olarak imzalanmış olur. Alıcı tarafta açık anahtar kullanılarak şifrelenmiş işlem mesajlarının hash kodları elde edilir. Alıcı tarafa gelen orijinal işlem mesajlarının hashlenmesi ile kimlik doğrulama işlemi devam eder.

İki farklı işlemle elde edilen hash kodları karşılaştırıldığında birbiriyle eşleşiyorsa veri bütünlüğü ve kimlik doğrulama sağlanmış olur. Kimlik doğrulama işleminin ardından işlem mesajları yeni bir blok olarak zincir yapısına eklenir.



Şekil 4.28 İYHF-Blockchain kullanımı

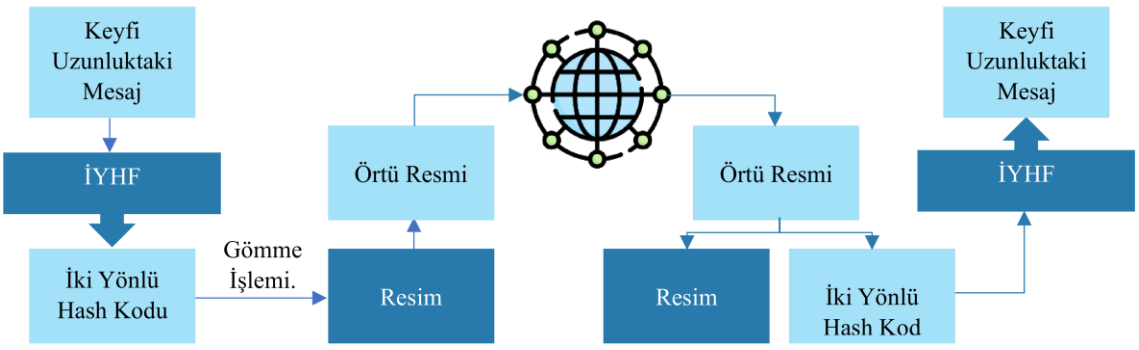
Şekil 4.28, blok zinciri iş akışının İYHF ile olası kullanımını göstermektedir. İYHF kullanan bir blok zinciri uygulamasında süreç şu şekildedir. İşlem mesajlarının şifrelenmiş hash kodları İYHF kullanılarak elde edilir. Mevcut uygulamalarda olduğu gibi orijinal işlem mesajları alıcı tarafa gönderilmez. Bu aşamada sadece işlem mesajlarının şifrelenmiş hash kodları alıcıya gönderilir. Bu, iletişim ağının bant genişliğinin ve alıcı taraftaki depolama alanının daha verimli kullanılmasını sağlar. Dahası, orijinal işlem mesajlarının gizliliği geleneksel uygulamalara göre daha güvenlidir. Bunun nedeni, orijinal işlem mesajlarının alıcı tarafa ek olarak gönderilmemesidir. Alıcı tarafa gelen şifrelenmiş hash kodları bir açık anahtar kullanılarak çıkarılır ve verilerin kimliği doğrulanır.

İYHF'nin örüntülü yapısı veri bütünlüğünü sağlar. İYHF yönteminin olası blok zinciri uygulamasında, bir yineleme anahtarının ve ikincil bir anahtarın kullanılması isteğe bağlıdır.

4.9.3 İYHF metodunun steganografi alanında uygulanması

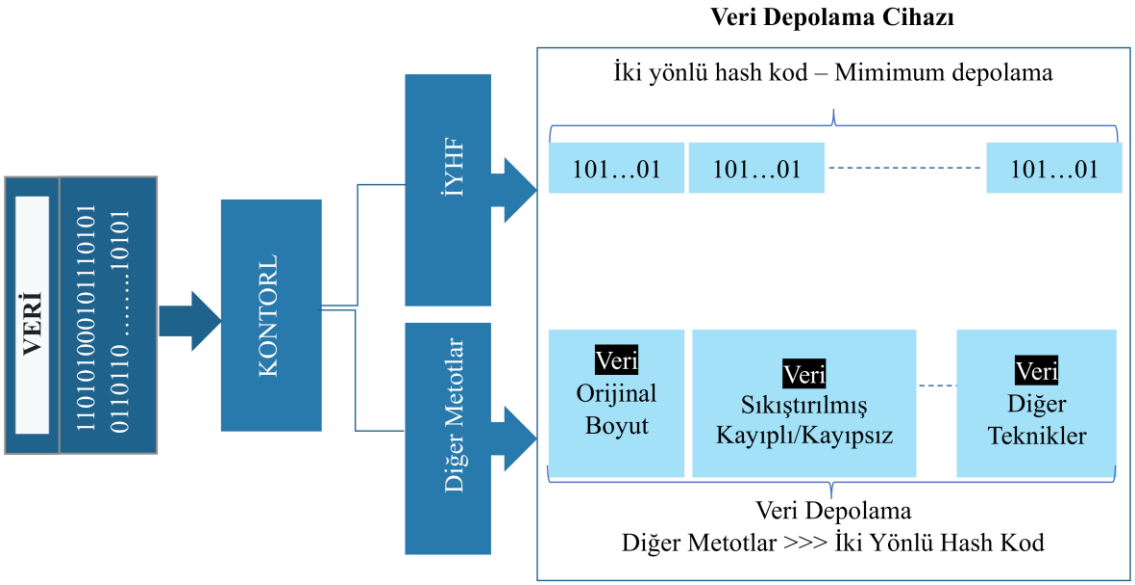
Görüntü steganografisinde alıcıya gönderilecek mesaj, örtü nesnesi olarak adlandırılan görüntünün içine gömülür. Görüntüye gömülen veri, görüntünün piksellerinde bir değişikliğe neden olur. Başka bir deyişle, orijinal görüntü değiştirilir ve bozulur. Görüntü steganografisi çalışmalarının amacı bu bozulmayı en aza indirmektir. Dolayısıyla örtü görüntüsü orijinal görüntüye ne kadar benzerse üçüncü şahısların dikkatini çekme ihtimali de o kadar azalır. Bu durum gerçekleştiğinde mesaj güvenli bir şekilde alıcısına ulaşır. İYHF kullanımı ile steganografinin amaçlanan hedefine optimize edilmiş bir şekilde ulaşılır. Bunun nedeni, büyük miktarda bit uzunluğundaki verinin özetlenmesi ve sabit uzunlukta veriye dönüştürülmesidir. Bazı istisnalar dışında, verilerin özetlenmesi, veri sıkıştırma algoritmaları kullanılarak elde edilen çıktıdan daha iyidir. Bu şekilde kapak görüntüsü mümkün olan en az bozulma ile elde edilir ve orijinal görüntüye daha çok benzer. Özellikle bit uzunluğundaki artışla doğru orantılı olarak daha iyi sonuçlar elde edilir. Bunun nedeni orijinal veri bit uzunluğu artsa bile görüntüdeki bozulmanın değişmemesidir.

Şekil 4.29, İYHF'nin görüntü steganografisinde olası kullanımını göstermektedir. İYHF kullanan görüntü steganografisinde, örtü nesnesi alıcıya ulaştığında görüntü ve iki yönlü hash kodu ayrıştırılır. Orijinal mesaj daha sonra alıcı tarafında iki yönlü hash kodu kullanılarak tekrar elde edilir. Bu işlemlerin gerçekleştirilmesi mevcut hash fonksiyonları ile mümkün değildir.



Şekil 4.29 Görüntü steganografisinde İYHF kullanımı.

4.9.4 İYHF ve veri depolama



Şekil 4.30 İYHF ve Veri Depolama

İYHF'nin en çok fayda sağlayacağı alanlardan biri, verilerin veri depolama cihazlarında çok az yer kaplayacak şekilde depolanmasını sağlayarak donanım maliyetlerini önemli ölçüde azaltacak olmasıdır.

Şekil 4.30'da İYHF'nin veri depolama cihazlarında kullanımına ilişkin bir senaryo gösterilmektedir. Bu senaryoya göre, tüm verilerin İYHF kullanılarak depolanması gerekmemektedir. Kontrol ünitesi tarafından yapılan seçime bağlı olarak, ayrıştırılan veriler İYHF uygulanarak hash kodları şeklinde saklanabilir. Verilerin bu şekilde saklanması donanım maliyetlerini önemli ölçüde azaltacaktır. Ayrıca donanım cihazlarına olan ihtiyacın azalması elektrik tüketiminin de azalmasına katkıda bulunur. Dahası, verilerin yoğun bir şekilde değiş tokuş edildiği internet ortamında bant genişliği ihtiyacı önemli ölçüde düşebilir. Bunun nedeni, verilerin orijinal bit uzunlukları ya da diğer tekniklerle biraz azaltılmış bit uzunlukları yerine hash kodlarıyla ağda dolaşması fikridir.

5. ÖNERİLEN METODUN UYGULANMASI

Önerilen metodun iş akışı ve güvenliği ile ilgili bazı uygulamalar yapılmıştır. Önerilen yöntemin iş akışında ortaya çıkan “gerçek”, “sahte” ve “kripto ham veri” miktarının tespiti için “A” karakteri için şifreli hash kodu üretilmiş ve bununla ilgili istatistiki değerler elde edilmiştir.

5.1 "A" Karakteri için Şifreleme/Hash Kodu Oluşturma Sürecinde Elde Edilen Sonuçlar

Bu çalışmada, "sahte" veri sayısı, "gerçek" veri sayısı, şifrelenmiş veri uzunluğu, toplam iterasyon sayısı ve diğer istatistiksel veriler Python'da bir uygulama ile elde edilmiştir. İstatistiksel veriler için, şifrelenecek ve hashlenecek düz metin olarak "A" karakteri ve başlangıç verisi olarak $(1000000,00000000)_{10}$ ondalık verisi seçilmiştir.

"Gerçek" ve "sahte" ondalık verilerin sayısal değerleri Çizelge 5.1'de listelenmiştir. İkili karşılığı $(01000001)_2$ olan "A" karakterini şifrelemek ve hash kodu üretmek için oluşturulan sayı örüntüsünde 4 "gerçek" ve 12 "sahte" veri gözlemlenmiştir. İlk "gerçek" verinin tür 1.3 $(463078,30547297)_{10}$ olduğu tespit edilmiştir. Bu veri için indirgeme miktarı bir bittir. "A" karakterinin ilk biti $(0)_2$ olduğu için bu veri biti, değeri $(463078,30547297)_{10}$ olan "gerçek" verinin içine kaydedilmiştir. $(12652818,13963791)_{10}$ ondalıklı verisi bir sonraki "gerçek" veri olarak belirlenmiştir. Bu veri için indirgeme miktarı 2 bittir. "A" karakterinin sıradaki bitleri $(10)_2$ olduğundan, "gerçek" veriye yüklenen bitler $(10)_2$ olmalıdır.

$(4199888,77380527)_{10}$ ondalık verinin 3. "gerçek" veri olduğu bulunmuştur. Bu "gerçek" veri için azaltma miktarı bir bit olarak bulunmuş ve "A" karakterinin sıradaki verisi olan $(0)_2$ biti bu veriye yüklenmiştir.

$(5621825,95322159)_{10}$ son "gerçek" veri olarak belirlenmiştir. Depolanacak son veri $(0001)_2$ olmalıdır. Son "gerçek" verinin azaltılma miktarı dört bittir. Bu nedenle, "A" karakterinin kalan dört biti son "gerçek" veriye yüklenmiştir.

"Geçer" olarak nitelendirilen veriler Çizelge 5.1'e dahil edilmemiştir. Bu nedenle, "gerçek" ve "sahte" veriler çizelgede ardışık olarak gösterilmiştir. Ancak sayı örüntüsünde bu şekilde sıralanmamışlardır. Gerçekte bu veriler "geç" olarak nitelendirilen veriler arasındadır. Çizelge 5.1'e göre şifrelenmiş veri/hash kodu $(10011100000100010111100100011111111101001101010110000)_2$ ve kripto ham verisi $(1010001001000000)_2$ olarak bulunmuştur.

"A" karakteri için şifreleme ve hash kodu oluşturma sürecinde elde edilen ek istatistiksel veriler Çizelge 5.2'de gösterilmiştir Çizelgede gösterildiği gibi, "A" karakterini şifrelemek için 7680 iterasyon gerçekleştirilmiştir. Bu sayı aynı zamanda sayı örüntüsünde 7680 ondalık veri üretildiğini göstermektedir. Bu verilerin dördünün "gerçek", on ikisinin ise "sahte" olduğu tespit edilmiştir. Bu durumda "gerçek" ve "sahte" ondalık sayıların toplamı, yani kripto ham verisinin bit uzunluğu 16 olarak hesaplanmıştır. "Kripto ham verisinin" boyutu bir indirgeme fonksiyonu (Huffman) kullanılarak 15 bite düşürülmüştür. Dört "gerçek" ondalık sayıya toplam 17 bit veri eklenmiştir. Eklenen 17 bitin sekizinin ondalık verinin tamsayı kısmına eklenen düz metne ait olduğu, kalan dokuz bitin ise ondalık verinin ondalık kısımlarına eklenen kontrol bitleri olduğu tespit edilmiştir. Kontrol bitleri "sahte" veri sayısını yaklaşık 7,4 kat azaltarak 89'dan 12'ye düşürmüştür.

Çizelge 5.1 "A" karakterinin şifrelenmesi sırasında elde edilen istatistikler

Şifrelenecek Veri => "A" $(65)_{10} = (01000001)_2$ Başlangıç Verisi : 1000000,00000000

Frekans Tablosu : 00 01 10 11

["1" "00" "010" "011"]

Sıra No	Sahte/Gerçek	Ondalıklı Veri	İndirgeme Miktarı (Bit)	Eklenen Veri	Eklenen Kontrol Verisi	Kripto Ham Veri
1	Sahte	14 024 721.46273759	-	-		0
2	Sahte	8 462 144.31595595	-	-		0
3	Sahte	72 540.25308223	-	-		0
4	Sahte	7 708 704.72681047	-	-		0
5	Sahte	5 292 350.10560543	-	-		0
6	Sahte	460 893.48845001	-	-		0
7	**Gerçek**	**463 078.30547297**	**1**	**0**	**1**	**1**
8	Sahte	9 522 640.89161647	-	-		0
9	Sahte	14 495 746.91813303	-	-		0
10	**Gerçek**	**12 652 818.13963791**	**2**	**10**	**11**	**1**
11	Sahte	399 390.61647479	-	-		0
12	Sahte	248 325.18498111	-	-		0
13	Sahte	97 358.95540271	-	-		0
14	**Gerçek**	**4 199 888.77380527**	**1**	**0**	**1111**	**1**
15	Sahte	8 701 196.27951145	-	-		0
16	**Gerçek**	**5 621 825.95322159**	**4**	**0001**	**11**	**1**

KRİPTO HAM VERİ OLUŞTURMA YÖNÜ

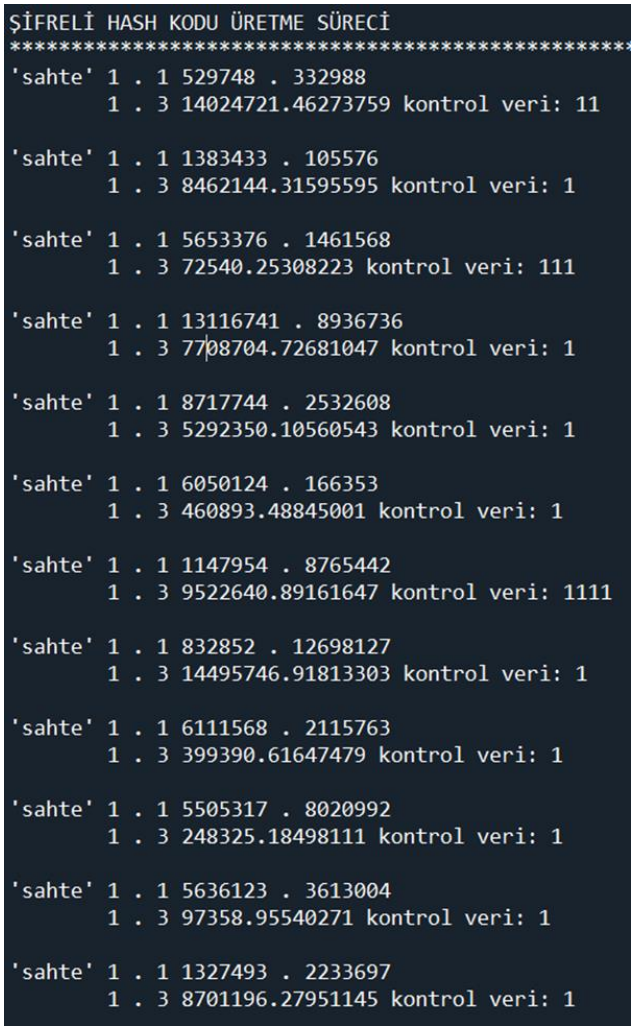
Ondalık sayı örüntüsü son verisi (şifreli metin ve hash kodu)	$(10011100000100010111100100011111111100110101011000)_2$
	$(10228089.33508696)_{10}$

Çizelge 5.2 "A" karakterlerinin şifrelenmesi sürecinde elde edilen diğer istatistikler

Toplam iterasyon sayısı	7680
Kripto ham veri (16 bit)	$(1010001001000000)_2$
Kripto ham verilerinin indirgeme fonksiyonu çıktısı (15 bits)	$(010010101000111)_2$
"Gerçek" veri sayısı	4
"Sahte" veri sayısı	12
Gerçek verinin tamsayı kısmına yüklenen düz metnin toplam bit sayısı.	8 bit
Gerçek verinin ondalık kısmına yüklenen kontrol verisinin toplam bit uzunluğu.	9 bit
Gerçek veriye yüklenen toplam bit sayısı Düz metin veri bitleri + kontrol verileri	8+9= 17 bit
Sahte veri sayısı-I (kontrol verileri uygulandığında)	12
Sahte veri sayısı - II (kontrol verileri uygulanmadığında)	89
Sahte veri sayısı - II / Sahte veri sayısı - I	89/16=7.41

```
**********************************
ŞİFRELİ HASH KODU ÜRETME SÜRECİ
**********************************

 1 . 1 6055696.17461765
 1 . 3 463078.30547297
gerçek:  Eklenen Veri: 0


 1 . 1 364945.00561431
 1 . 3 12652818.13963791
gerçek:  Eklenen Veri: 10


 1 . 1 9783305.68162354
 1 . 3 4199888.77380527
gerçek:  Eklenen Veri: 0


 1 . 1 8914001.85992787
 1 . 3 5621825.95322159
gerçek:  Eklenen Veri: 0001
```

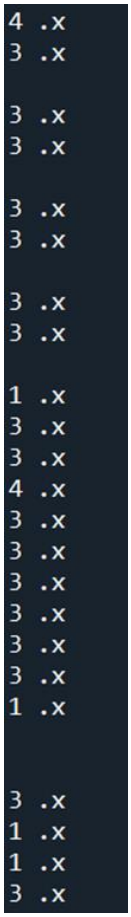
Şekil 5.1 "A" karakteri "gerçek" veri blokları değerleri



Şekil 5.2 “A” karakteri şifreli hash kodu üretme süreci “sahte” veri blokları değerleri

Şekil 5.1’de “A” karakteri şifreli hash kodu üretme sürecinde sadece “gerçek” veri bloklarının aldıkları değerler, python kodları ekran görüntü kesiti ile gösterilmiştir. Şekil 5.2’de ise sadece “sahte” veri blokları ve aldıkları değerlerin python kodları ekran görüntü kesiti verilmiştir. Şifreli hash kodu çözme süreci yönüne göre önce bloğun son verileri olan 1.3 türündeki veriler değerlendirilmektedir. Buna göre Şekil 5.2’de verilen son blok ele alındığında 1.3 türündeki $(8701196.27951145)_{10}$ ondalık verisinin içinde gerçekte veri olmamasına rağmen veri sakladığı davranışı sergilediği görülmektedir. Bunun sebebi öncelikle bitiş blok veri türünün 1.3 olması ve kontrol veri şartını sağlamasıdır. Bu verinin ondalık kısmı bir bit indirgenebildiği için kontrol verisi $(1)_2$ olmakta ve bu şartın sağlandığı görülmektedir. Bu koşullardan başka diğer koşullardan ters indirgeme

fonksiyonu uygulandığında 1.1 türüne ulaşması ve ters değiştirme fonksiyonu uygulandığında 3.x tür koşulunu sağlanması bu bloğun “sahte” türde olduğunu gösterir.



Şekil 5.3 "A" karakterlerinin şifrelenmesi süreci son 6 blok türleri

Şekil 5.3’te “A” karakteri şifreleme sürecinde elde edilen bloklardan son 6’sının python kodları ekran çıktı görüntü kesiti verilmiştir. Buna göre son 6 bloğun veri yükleme ve “sahte” veri bloklarına sahip olmadıkları görülmektedir. Bu blokların tamamı “geç” türündedir. Son blok değerlendirildiğinde bloğun başlangıç türünün 3.x bitiş türünün yine 3.x türünde olduğu görülmektedir. Blok başlangıç türü 3.x türünde olduğu için bitiş türününde 2/3/4.x türlerinden birisi olması gerekmektedir. Ancak blok başlangıç türünden sonra üretilen veri türleri 1.x ve 1.x olmuş ve blok sonlandırılmamıştır. Bloğun dördüncü verisi 3.x türünden olduğu için blok sonlandırılmıştır.

6. BULGULAR

Çizelge 6.1'de, önerilen metod kullanılarak elli farklı metin için elde edilen istatistiksel değerler gösterilmektedir. İş akışı ile ilgili yapılan bir diğer istatistiki uygulamada koşulların önerilen metot üzerindeki etkisi incelenmiştir. İYHF'nin tek yönlü hash fonksiyonlarının sahip olduğu güvenliği sağlayıp sağlayamayacağına yönelik çeşitli testler yapılmış ve önemli veriler elde edilmiştir.

Çizelge 6.1 Farklı metinler için istatistikler

Metin	Real	Fake	İterasyon Sayısı	Fake/İterasyon Oranı (%)
Steganografi, bilgiyi gizleme sanatıdır ve farklı veri türleriyle kullanılabilir.	402	2016	1.140.700	0.177
Gizli mesajlar, resim dosyalarının piksellerine yerleştirilerek taşınabilir.	371	1779	1.033.898	0.172
Metin steganografisi, yazılı metinlerin içine gizli mesajların saklanmasıdır.	386	1892	1.065.830	0.178
Ses steganografisi, ses dosyalarında gizli bilgilerin taşınmasını sağlar.	350	1844	1.050.156	0.176
Video steganografisi, video dosyalarında gizli bilgilerin gizlenmesini sağlar.	398	1863	1.082.994	0.172
Renk steganografisi, renk kodlaması kullanılarak gizli mesajların saklanmasını ifade eder.	434	2061	1.239.850	0.166
Dosya steganografisi, dosya içlerine gizli verilerin yerleştirilmesidir.	347	1742	985.584	0.177
Alfanumerik steganografi, metin, sayı ve sembollerle gizli veri aktarımıdır.	360	1627	999.224	0.163
Hafif steganografi, orijinal veriyi en az etkileyen gizleme yöntemlerini kullanır.	379	2013	1.205.130	0.167
Ağ steganografisi, ağ trafiği içinde gizli mesajların taşınmasını sağlar.	355	1730	1.004.538	0.172
Harita steganografisi, harita verilerinde gizli bilgilerin saklanmasını sağlar.	384	1979	1.101.062	0.18
Dosya sistemi steganografisi, dosya sistemlerinde gizli verilerin saklanmasını ifade eder.	427	2289	1.320.440	0.173
Steganografi, kriptografiyle birlikte kullanılarak daha güvenli iletişim sağlar.	382	1901	1.102.496	0.172

Çizelge 6.1 Farklı metinler için istatistikler (devam)

Metin	**Real**	**Fake**	**İterasyon Sayısı**	**Fake/İterasyon Oranı (%)**
Bilgisayar programları, steganografi kullanarak gizli mesajlar taşıyabilir.	360	1858	1.087.778	0.171
Steganografi, antik çağlardan beri gizli bilgi iletiminde kullanılmıştır.	359	1728	978.970	0.177
Birçok dijital medya türü, steganografi için uygun bir ortam sağlar.	321	1661	968.114	0.172
Steganografik araçlar, kullanıcılara gizli mesajları oluşturma ve çözme imkanı sağlar.	430	2269	1.322.814	0.172
Gizli mesajların taşınması için steganografi, alternatif bir güvenlik yöntemi sunar.	411	1971	1.136.650	0.173
Biyometrik veriler, steganografi ile güvenli bir şekilde saklanabilir.	342	1907	1.120.148	0.17
Steganografi, veri gizlemenin yanı sıra veri bozulmasını da önleyebilir.	347	1621	938.852	0.173
Steganografi, istihbarat toplama ve kriptoanalizde önemli bir rol oynar.	347	1648	934.930	0.176
Steganografi, gizli mesajların tespit edilmesini zorlaştırabilir.	309	1591	919.918	0.173
Modern steganografi teknikleri, yüksek oranda gizlilik sağlayabilir.	325	1436	823.620	0.174
Steganografi, bilgisayar güvenliği alanında önemli bir konudur.	304	1341	768.186	0.175
Kuantum steganografi, kuantum mekaniği temelli gizli bilgi iletimi sağlar.	371	1924	1.139.488	0.169
Steganografik yöntemler, siber saldırılar sırasında kullanılabilir.	331	1599	940.770	0.17
Steganografi, iletişim gizliliğini korumak için kullanılabilir.	318	1578	922.024	0.171
Steganografi, dosya boyutunu artırmadan gizli mesajların saklanmasına olanak tanır.	420	2000	1.161.452	0.172
Steganografi, veri sıkıştırma teknikleriyle birlikte kullanılabilir.	328	1736	1.010.648	0.172
Steganografi, veri güvenliği ve mahremiyetinde önemli bir rol oynar.	339	1712	984.500	0.174
Steganografi, görünürde zararsız dosyalarda gizli bilgiler saklayabilir.	363	1782	994.766	0.179

Çizelge 6.1 Farklı metinler için istatistikler (devam)

Metin	**Real**	**Fake**	**İterasyon Sayısı**	**Fake/İterasyon Oranı (%)**
Steganografi, sinyal işleme ve gömülü sistemlerle ilişkilidir.	303	1561	890.252	0.175
Steganografi, dijital su işaretleme sistemlerinde kullanılabilir.	303	1668	933.576	0.179
Steganografi, güvenli elektronik ticaret uygulamalarında kullanılabilir.	353	1669	942.956	0.177
Steganografi, mesajın sızdırılmasını önlemek için kullanılabilir.	310	1393	781.210	0.178
Steganografi, hukuki belgelerde bilgi saklamak için kullanılabilir.	327	1745	1.000.854	0.174
Steganografi, tıbbi görüntülerde gizli bilgilerin saklanmasına olanak tanır.	375	1844	1.102.820	0.167
Steganografi, ses dosyalarında gizli mesajların taşınmasını sağlar.	335	1590	933.220	0.17
Steganografi, e-posta iletişiminde gizlilik sağlamak için kullanılabilir.	360	1773	1.033.788	0.172
Steganografi, ağ trafiğinde gizli mesajların taşınmasına izin verir.	342	1611	885.790	0.182
Steganografi, görsel içeriklerde gizli bilgilerin saklanmasına izin verir.	360	1617	927.452	0.174
Steganografi, dijital imza sistemlerinde kullanılabilir.	278	1279	747.554	0.171
Steganografi, sanal özel ağ (VPN) iletişiminde kullanılabilir.	309	1376	803.402	0.171
Steganografi, veri saklama sistemlerinde kullanılabilir.	281	1295	755.296	0.171
Steganografi, biyometrik tanıma sistemlerinde kullanılabilir.	297	1429	827.724	0.173
Steganografi, veri hırsızlığını önlemek için kullanılabilir.	307	1503	851.620	0.176
Steganografi, yüksek güvenlik gerektiren ortamlarda kullanılabilir.	320	1509	884.962	0.171
Steganografi, veri sızıntılarını önlemek için kullanılabilir.	306	1549	904.808	0.171
Steganografi, dijital hak yönetimi (DRM) sistemlerinde kullanılabilir.	336	1525	888.590	0.172
Steganografi, savunma ve güvenlik alanlarında kullanılabilir.	293	1375	781.816	0.176

6.1 Koşulların İYHF Üzerindeki Etkisi ve Değerlendirilmesi

İYHF metodolojisinde, belirli koşullar değiştirildiğinde veya kaldırıldığında "sahte" veri sayısı değişir. İYHF'nin ideal çalışma durumuna ulaşması, "sahte" verilerin tamamen ortadan kaldırılmasına bağlıdır. Bu nedenle, belirli koşulların İYHF metodolojisi üzerindeki etkisi analiz edilmiştir. Bu amaçla, "A" karakteri için şifreli hash kodu üretme sürecinde bazı koşullar kaldırılmış ve değiştirilmiştir. Sonuçlar Çizelge 6.2'te gösterilmiştir.

Çizelge 6.2 Koşul 2 ve koşul 3'ün İYHF üzerindeki etkisi

	Mevcut Durum	1.1-1.3 bloğunun 1.1-1.1 olması durumu (Koşul 2)	3. koşulun kaldırılması durumu
Sahte veri sayısı-I (kontrol verileri uygulandığında)	12	47	27
Sahte veri sayısı - II (kontrol verileri uygulanmadığında)	89	225	201

Koşul 2, bir veri yükleme bloğunun 1.3 türündeki veri ile sonlanmasını gerektirir. Koşul 2, bloğun 1.3 yerine 1.1 türü ile sonlandırılması gerektiği ile değiştirilmiştir. Süreç çalıştırıldığında, "sahte" verilerin sayısı kontrol verileriyle 47 ve kontrol verileri olmadan 225 olarak elde edilmiştir. Mevcut durumla karşılaştırıldığında "sahte" veri sayısında kontrol verili duruma göre 3,9 kat, kontrol verisiz duruma göre ise 2,5 kat artış söz konusudur.

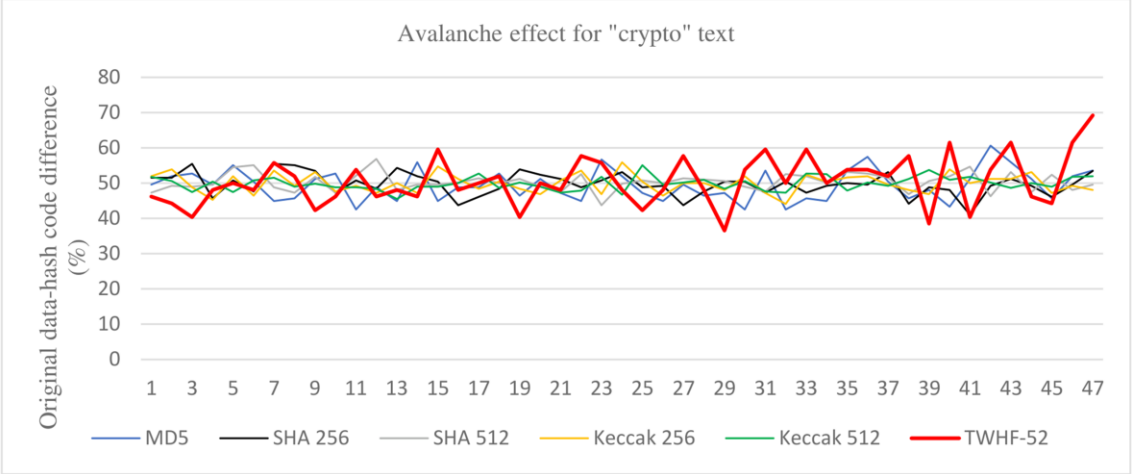
İYHF süreci Koşul 3'ün kaldırılması ile uygulandığında, "sahte" veri sayısı kontrol verileri ile 27, kontrol verileri olmadan 201 olarak bulunmuştur. Her iki durumda da mevcut duruma kıyasla yaklaşık 2,25'lik bir artış gözlemlenmiştir. Çizelge 6.2'den elde edilen veriler, mevcut durumun daha iyi sonuçlar verdiğini göstermektedir. Bu sonuçlar, İYHF ile ilgili gelecekte yapılacak çalışmalarda "sahte" veri sayısının azaltılabileceğine veya tamamen ortadan kaldırılabileceğine dair güçlü ipuçları vermektedir.

6.2 İYHF Güvenlik Analizleri

İYHF ve bazı tek yönlü hash fonksiyonlarının güvenliğini analiz etmek için testler yapılmıştır. İlk testte, "crypto" kelimesi için üretilen hash kodu orijinal hash kodu olarak kabul edilmiştir. Sürecin kalan aşamalarında 48 bit uzunluğundaki "crypto" kelimesinin her bir biti MSB tarafından başlanarak değiştirilmiş ve 48 farklı hash kodu üretilmiştir. Bitlerin değiştirilme işlemi "0" için "1" ve "1" için "0" şeklindedir. Üretilen 48 hash kodu orijinal hash kodu ile bit seviyesinde karşılaştırılmıştır. Her bir hash kodunun orijinal hash koduna göre yüzde oranı cinsinden farkları bulunmuştur. Bu değerler, orijinal verinin bir bitindeki değişikliğin üretilen hash kodlarındaki yansımasını vermektedir.

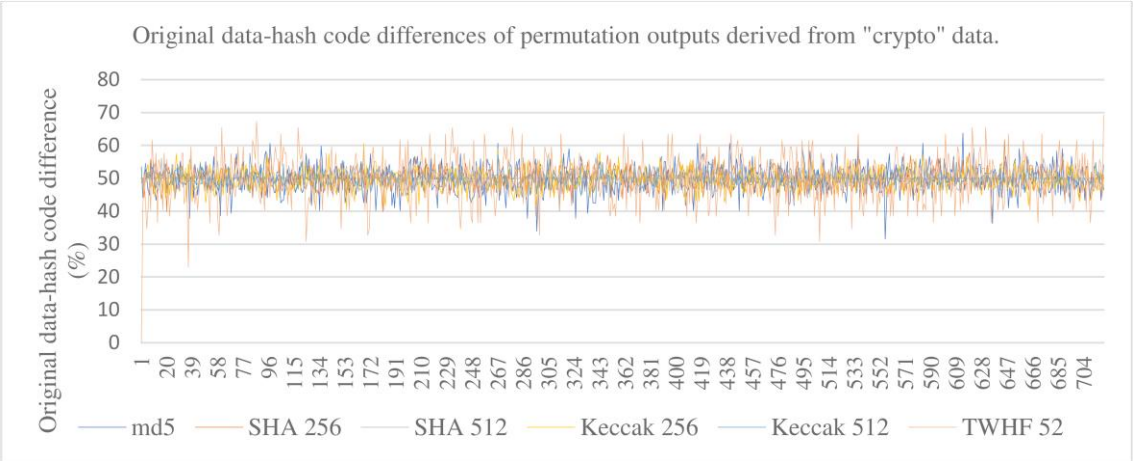
Şekil 6.1'deki grafik hash fonksiyonlarının çığ etkisini göstermektedir.

İYHF'nin 52 bitlik çıktısı için ortalama fark %50,57'dir. Diğer hash fonksiyonları için bulunan değerler MD5 %49,42, SHA 256 %49,70, SHA 512 %50,31, Keccak 256 %49,90 ve Keccak 512 %49,87'dir. Bu testte İYHF, tek yönlü özet fonksiyonlarından daha iyi performans göstermiştir.



Şekil 6.1. "crypto" metni için çığ etkisi

Güvenlik analizi için bir başka test de "crypto" kelimesi için permütasyon testidir. Bu amaçla ilk olarak "crypto" kelimesi için bir hash kodu üretilmiştir. Üretilen bu kod orijinal hash kodu olarak kabul edilmiştir. Daha sonra "crypto" kelimesinin harfleri değiştirilerek 719 farklı kelime türetilmiştir. Bu kelimelerden bazıları "crypot", "crytpo", "crytop" vb. olarak örneklendirilebilir. Türetilen 719 kelimenin her biri için hash kodu üretilmiştir. Bu hash kodları orijinal hash kodu ile bit seviyesinde karşılaştırılmıştır. Karşılaştırma sonucunda türetilen kelimelerin hash kodlarının orijinal hash kodundan yüzde olarak ne kadar farklı olduğu elde edilmiştir. Bu değerlere göre çizilen bir grafik Şekil 6.2'de gösterilmiştir. Ayrıca ortalama fark değerleri İYHF %50,07, md5 %49,77, SHA 256 %49,87, SHA 512 %49,91, Keccak 256 %49,63 ve Keccak 512 %49,98 olarak elde edilmiştir. İYHF, bu testte de daha iyi performans göstermiştir.



Şekil 6.2 "crypto" verilerinden türetilen permütasyon çıktılarının orijinal veri-hash kodu farklılıkları

6.2.1 NIST testleri

İYHF'nin ve bazı tek yönlü hash fonksiyonlarının rassallığı NIST testleri ile ölçülmüştür. Çığ etkisi testi için üretilen 48 farklı hash kodu NIST testlerinde de kullanılmıştır. Sonuçlar Çizelge 6.3'te gösterilmiştir. Çizelgeye göre, tüm hash fonksiyonları Maurer'in Evrensel İstatistik testinde başarısız olmuştur. İYHF, rastgelelik testlerinin 14'ünden geçmiştir.

Çizelge 6.3 "crypto" metni için İYHF ve tek yönlü fonksiyonların NIST testleri

KECCAK 512		KECCAK 256		SHA 512		SHA256		MD5		Hash Fonksiyonları
0.021	R	0.759	R	0.527	R	0.690	R	0.28	R	Frequency Test (Monobit)
0.580	R	0.798	R	0.680	R	0.626	R	0.671	R	Frequency Test within a Block
0.746	R	0.348	R	0.157	R	0.537	R	0.628	R	Run Test
0.188	R	0.344	R	0.422	R	0.203	R	0.019	R	Longest Run of Ones in a Block
0.810	R	0.107	R	0.805	R	0.118	R	0.161	R	Binary Matrix Rank Test
0.584	R	0.100	R	0.989	R	0.235	R	0.649	R	Discrete Fourier Transform (Spectral) Test
0.768	R	0.121	R	0.152	R	0.950	R	0.695	R	Non-Overlapping Template Matching Test
0.805	R	0.641	R	0.535	R	0.768	R	0.085	R	Overlapping Template Matching Test
-1	N-R	-1	N-R	-1	N-R	-1	N-R	-1	N-R	Maurer's Universal Statistical test
0.430	R	0.766	R	0.534	R	0.353	N-R	$1.048e^{-5}$	N-R	Linear Complexity Test
0.809	R	0.699	R	0.075	R	0.557	R	0.250	R	Serial test:
0.420	R	0.092	R	0.231	R	0.010	R	0.011	R	Approximate Entropy Test
0.017	R	0.981	R	0.871	R	0.063	R	0.163	R	Cummulative Sums (Forward) Test
0.011	R	0.769	R	0.344	R	0.136	R	0.573	R	Cummulative Sums (Reverse) Test
0.367	R	0.136	R	0.053	R	0.161	R	0.156	R	Random Excursions Test – State : +1
0.779	R	0.666	R	0.010	N-R	0.004	N-R	0.317	R	Random Excursions Variant Test – State : -1

Çizelge 6.3 "crypto" metni için İYHF ve tek yönlü fonksiyonların NIST testleri (devam)

TWHF-52	0.606	0.518	0.582	0.776	0.481	0.956	0.557	0.254	-1	0.985	0.758	$2.33e^{-7}$	0.225	0.566	0.382	0.570
	R	R	R	R	R	R	R	R	N-R	R	R	N-R	R	R	R	R

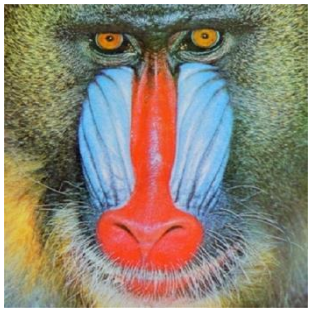
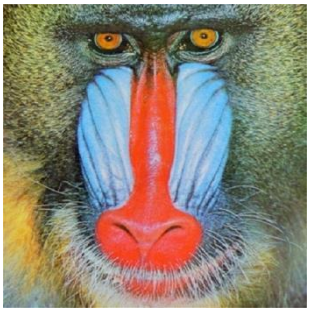
6.2.2 İYHF metodunun imaj steganografisi üzerine etkileri

Çalışmada öne sürülen İYHF metodunun steganografi veri gizleme bilimi üzerine olan olumlu etkileri, Çizelge 6.4, Çizelge 6.5, Çizelge 6.6 ve Çizelge 6.7 gösterilen bulgularla ifade edilmektedir. Bu çizelgelerde MSE ve PSNR temel ölçüt olarak alınmıştır. Bir görüntü üzerinde yapılan herhangi bir işlem, görüntü kalitesinde önemli bir kayba yol açabilir. Görüntü kalitesini değerlendirmenin iki ana yöntemi bulunmaktadır: öznel ve nesnel. Öznel yöntemler, belirli kriterlere dayanmayan ve insanların duyusal değerlendirmesine dayanan bir değerlendirme türüdür. Nesnel yöntemler ise istatistiksel parametreler ve testler aracılığıyla ifade edilen ve bilgiye ve gerçeklere dayanan objektif değerlendirme türleridir. Görsel kaliteyi ölçmek için kullanılan MSE ve PSNR gibi parametreler objektif yöntemlere örnektir. MSE (ortalama kare hatası) sıfıra yaklaştıkça PSNR sonsuza yaklaşır. Daha yüksek PSNR değerleri, daha yüksek görüntü kalitesini temsil etmektedir. Başka bir deyişle, çok düşük bir PSNR değeri iki görüntü arasında büyük sayısal farklılıklar olduğunu gösterir. MSE formülü denklem 6.1'de, PSNR formülü ise denklem 6.2'de verilmiştir. Hore & Ziou (2010).

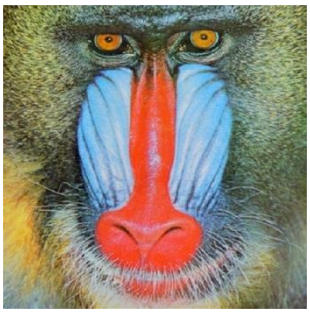
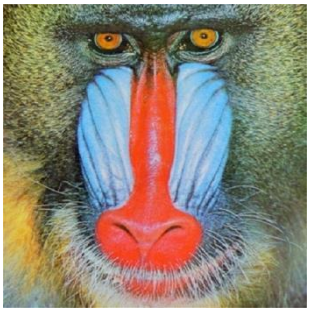
$$MSE(f,g) = \left(\frac{1}{M.N}\right)\sum_{i=1}^{M}\sum_{j=1}^{N}\left(f_{ij} - g_{ij}\right)^2 \tag{6.1}$$

$$PSNR(f,g) = 10\log_{10}\left(\frac{255^2}{MSE(f,g)}\right) \tag{6.2}$$

Çizelge 6.4 6432 bit uzunluğundaki orijinal verinin steganografik ölçüm değerleri

ORİJİNAL METİN					
Steganography is the art of hiding information and can be used with different types of data. Hidden messages can be carried by embedding them in pixels of image files. Text steganography is the hiding of secret messages in written text. Audio steganography allows hidden information to be carried in audio files. Video steganography allows hidden information to be hidden in video files. Color steganography refers to the hiding of secret messages using color coding. File steganography is the insertion of secret data into files. Alphanumeric steganography is the transmission of secret data using text, numbers and symbols. Light steganography uses obfuscation methods that minimize the impact on the original data. Network steganography enables the transport of secret messages within network traffic.					
Veri (Hash Kodu)	**Veri Binary (6432 bit)**	**Orijinal Resim**	**Stego Resim**	**MSE**	**PSNR**
	0101001101110100011001010110110111011000010110 ……..01100011001011110			9.1189e-03	68.5314 dB

Çizelge 6.5 6432 bit uzunluğundaki orijinal verinin hash kodu steganografik ölçüm değerleri

Veri (Hash Kodu)	Veri Binary (52 bit)	Orijinal Resim	Stego Resim	MSE	PSNR
	0000001010100001101111111100 0001111000000000010111111110			6.6440e-05	88.7950 dB

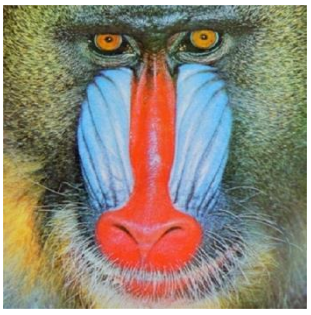
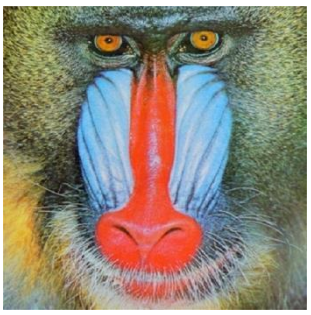
Çizelge 6.4'te, 6432 bit uzunluğunda metin türündeki bir veri, jpg türünde RGB kanalına sahip 347x347 boyutlarındaki bir resme steganografik LSB tekniği ile yerleştirilmiştir. Elde edilen stego resminde MSE değeri 9.1189e-03, PSNR değeri 68.5314 dB olarak ölçülmüştür. 6432 bit uzunluğundaki orijinal veri, steganografi alanındaki ödünleşmeyi ortadan kaldırmayı hedefleyen İYHF metodu ile hash işleminden geçirilerek 52 bit uzunluğundaki $(0000001010100001101\ 1111100000111100000\ 00000101111110)_2$ hash kodu elde edilmiştir. Elde edilen hash kodu orijinal verinin yerleştirildiği orijinal resme ayrıca yüklenerek ikinci bir stego resim elde edilmiştir. Bu stego resim için MSE değeri 6.6440e-05 ve PSNR değeri 88.7950 dB olarak ölçülmüştür. Çizelge 6.5'te ölçüm değerleri ile birlikte orijinal resim ve stego resim verilmiştir.

Çizelge 6.4 ve Çizelge 6.5 veri ve ölçüm sonuçları MSE özelinde değerlendirildiğinde hash kodu yüklenen stego resimdeki ölçümün orijinal verinin yüklendiği stego resme göre yaklaşık 137 kat daha düşük olduğu görülmektedir. MSE değerinin çok küçük olması stego resmin orijinal resme çok daha yakın olduğunu gösterir.

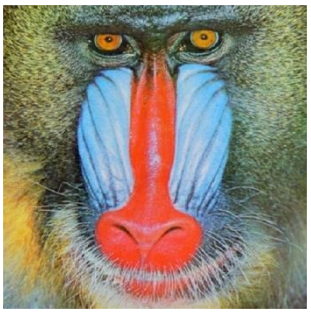
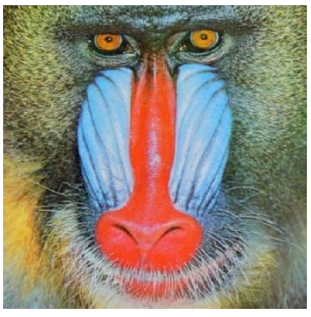
Çizelge 6.6 18984 bit uzunluğundaki orijinal verinin steganografik ölçüm değerleri

Steganography, an ancient art of secret communication, has evolved into a sophisticated technique used in modern digital environments. It involves concealing messages within seemingly innocuous carriers, such as images, audio files, or even text, to transmit information covertly. Unlike cryptography, which focuses on encryption to secure data, steganography aims to hide the existence of the message itself. The origins of steganography can be traced back to ancient civilizations, where messages were hidden within wax tablets or tattooed onto the shaved heads of messengers. These covert methods allowed sensitive information to be transmitted across enemy lines without detection. Over time, steganography has adapted to the digital age, finding new forms of concealment within the vast expanse of digital data. In digital steganography, information is embedded into digital media through subtle modifications that are imperceptible to the human eye or ear. For example, in image steganography, data can be inserted into the least significant bits of pixel values without visibly altering the image. Similarly, in audio steganography, hidden messages can be encoded within the frequency spectrum of audio files. The proliferation of digital communication channels and the increasing volumes of data exchanged online have made steganography an invaluable tool for various purposes. From covert communication in espionage and intelligence operations to digital watermarking for copyright protection, steganography serves a wide range of applications. However, steganography also poses challenges for security professionals and law enforcement agencies. Its use by criminals and terrorists to conceal malicious activities or coordinate illegal operations presents a significant threat to national security and public safety. Detecting and decrypting hidden messages embedded within digital media require advanced forensic techniques and sophisticated algorithms. Despite the challenges it presents, steganography continues to intrigue researchers and practitioners alike. The ongoing arms race between steganographers and those tasked with detecting hidden messages fuels innovation in both fields. As technology advances, steganography will remain a powerful tool for clandestine communication, shaping the landscape of digital security and privacy in the years to come.

Çizelge 6.6 18984 bit uzunluğundaki orijinal verinin steganografik ölçüm değerleri (devam)

Veri (Hash Kodu)	Veri Binary (18984 bit)	Orijinal Resim	Stego Resim	MSE	PSNR
	0101001101110100011001011011001 1101100001011O.....1011010110010100101110			2.6360e-02	63.9213 dB

Çizelge 6.7 18984 bit uzunluğundaki orijinal verinin hash kodu steganografik ölçüm değerleri

Veri (Hash Kodu)	Veri Binary (52 bit)	Orijinal Resim	Stego Resim	MSE	PSNR
	1001111111100100100010011100 0110000101000010001011110			8.3050e-05	88.9374 dB

İYHF metodunun kullanımı ile orijinal veri yerine hash kodunun yüklenmesi ile steganografideki ödünleşmenin orijinal veri miktarı arttıkça veriminin daha çok arttığını göstermek amacıyla Çizelge 6.6'da gösterildiği üzere orijinal veri miktarı arttırılarak 18984 bite çıkarılmıştır. 18894 bit uzunluğundaki veri, Çizelge 6.4 ve Çizelge 6.5'teki veriler için kullanılan orijinal resme yüklenerek MSE ve PSNR değerleri bu orijinal veri için MSE değeri 2.6360e-02 ve PSNR değeri 63.9213 dB olarak ölçülmüştür. Orijinal verinin hash kodu İYHF ile elde edilerek orijinal resmin içine yerleştirilmiş ve MSE 8.3050e-05, PSNR 88.9374 dB olarak ölçülerek ve Çizelge 6.7'de gösterilmiştir.

7. TARTIŞMA

Tez çalışmasında, literatürde tek yönlü olarak yer alan hash fonksiyonlarının iki yönlü olup olamayacağı araştırılmış bu bağlamda steganografik yöntemlerle veri iletiminde algınamamazlık-kapasite ve güvenlik-kapasite arasındaki ödünleşim problemini ortadan kaldırabileceği öne sürülmüştür. Bu kapsamda, iki yönlü hash fonksiyonlarının faydaları ve tek yönlü hash fonksiyonlarının sunduğu güvenlik özelliklerini yerine getirme kabiliyetleri tartışılmıştır. Bu amaçla, orijinal veriden şifreli bir özet kodu üretmeyi ve orijinal veriyi şifreli özet kodundan geri almayı içeren İYHF metodolojisi geliştirilmiştir.

Şifreli hash kodu üretme sürecinde İYHF, aralarında matematiksel bir ilişki bulunan ondalık sayılardan oluşan bir örüntü oluşturur ve orijinal veriyi küçük bitler şeklinde bu örüntüde saklar. Şifre çözme sürecinde, sayı örüntüsünün son ondalık sayısı hash kodu olarak alınır ve ondalık sayılar arasındaki matematiksel ilişkilerden yararlanılarak orijinal veriye ulaşılır.

Tez çalışmasında bazı araştırma sorularına cevaplar elde edilmiştir. Bu sorular ve sonuçlara göre verilen cevaplar aşağıdaki gibi özetlenebilir.

Günümüzde tek yönlü olan hash fonksiyonları çift yönlü olabilir mi? Bu çalışma, tek yönlü hash fonksiyonları için hesaplama zamanı açısından imkansız olan orijinal veriye ulaşmayı mümkün kılmayı amaçlamaktadır. Önerilen metodolojinin şifreli hash kodu oluşturma süreci kararlıdır. Ancak şifre çözme işlemi sırasında Çizelge 6.1'deki verilere göre ortalama %0,17 oranında "sahte" veri gözlemlenmiştir. Başka bir deyişle, bu veriler sayı örüntüsünün %0,17'sinin kodunun çözülmesinde belirsizlik olduğunu göstermektedir. Belirsizlik, çalışmanın mevcut durumunda "kripto ham veri" olarak nitelendirilen ek veriler aracılığıyla geçici olarak giderilmiştir.

Hash fonksiyonlarına iki yönlü özellik kazandırmanın amacı nedir? Hash fonksiyonlarının tek yönlü olması bazı uygulamalar için güvenlik açısından avantaj sağlasa da dezavantajları da olabilmektedir. Bu dezavantajlar, hash kodunun yanı sıra orijinal verinin de alıcıya gönderilmesi ya da alıcının orijinal veriye önceden sahip olması gerekliliği ile açıklanabilir. Ancak, tek yönlü hash fonksiyonlarının güvenlik özelliklerine sahip iki yönlü bir hash fonksiyonunda, orijinal verilerin alıcıya gönderilmesine gerek yoktur. Bu da daha yüksek bir güvenlik seviyesi tanımlar.

İki yönlü hash fonksiyonlarının olası kullanım alanları nelerdir? Tek yönlü hash fonksiyonlarına kıyasla iki yönlü hash fonksiyonlarının bu alanlarda kullanılmasının faydaları nelerdir? İki yönlü hash fonksiyonları, tek yönlü hash fonksiyonlarının kullanıldığı dijital imza ve blok zinciri uygulamalarında kullanılabilir. Bu alanlarda kullanıldığında orijinal verinin alıcı tarafa ulaşması gerekmez ve bu nedenle daha yüksek bir güvenlik seviyesi sunar. Ayrıca hash fonksiyonlarının iki yönlü olması, steganografi ve veri depolama için de kullanılmalarına olanak tanır. Steganografik uygulamalarda, örtü nesnesinin bozulması en aza indirilebilir. Hash fonksiyonlarının veriyi bir özet şeklinde sunması ve orijinal verinin bit uzunluğu artsa bile çıktı uzunluğunun değişmemesi steganografide çok daha etkili olmasını sağlar. Verinin depolama alanlarında özet halinde tutulması ve ihtiyaç duyulduğunda orijinal veriye kayıpsız olarak ulaşılmasının faydaları inanılmazdır.

İki yönlü hash fonksiyonları tek yönlü hash fonksiyonlarının güvenlik özelliklerini sağlayabilir mi? İYHF yönteminin 52 bitlik düşük uzunluklu çıktısının girdi verisi ile anlamlı bir ilişkisi olup olmadığı çalışma kapsamında yapılan testlerle ortaya konulmaya çalışılmıştır. Bu amaçla gerçekleştirilen testlere ait grafikler Şekil 6.1 ve Şekil 6.2'de gösterilmiştir. Sonuçları Şekil 6.1'de görselleştirilen ve İYHF'nin çığ etkisini gösteren testlerin ilkinde ortalama %50,57'lik bir fark oranı elde edilmiştir. Bu sonuç, bazı yaygın tek yönlü fonksiyonlar kullanılarak elde edilen sonuçlardan daha iyidir.

Şekil 6.2'de görselleştirilen ikinci testte ise "crypto" metni için permütasyon ile üretilen 719 farklı metnin hash kodlarının orijinal metnin hash koduna göre fark oranı ortalama %50,07 olarak bulunmuştur. Bu ortalama fark yüzdesi, tek yönlü hash fonksiyonlarının

ortalama fark yüzdesinden biraz daha iyidir. İYHF'nin son güvenlik testi, sonuçları Çizelge 6.3'te listelenen NIST rastgelelik testidir. Bu test, 48 bitlik "kripto" metninin her bir bitinin sırayla değiştirilmesiyle elde edilen 48 farklı veri için 48 farklı hash kodunun rastgeleliğini içermektedir. Bu teste göre, İYHF dahil tüm hash fonksiyonları Maurer'in Evrensel İstatistik testinde başarısız olmuştur. İYHF ayrıca yaklaşık entropi testinde de başarısız olmuş ancak diğer 14 rastgelelik testini geçmiştir. Test sonuçları, İYHF'nin tek yönlü hash fonksiyonlarının güvenlik özelliklerini sağlayabileceğini güçlü bir şekilde göstermektedir. Ancak, iki yönlü doğası nedeniyle, İYHF metodolojisine "anahtar" yapısı dahil edilmelidir. Bu anahtar ile öncelikle "kök hash koduna" erişilmeli ve orijinal verinin deşifre edilme süreci bu noktadan başlamalıdır. Aksi takdirde bu güvenlik aşaması olmadan şifre çözme işlemine başlamak İYHF yöntemini güvenilmez hale getirebilir. Bu durumun görsel bir temsili Şekil 4.23'te gösterilmektedir.

Çalışmanın ana konusu olan, yüksek yüklü verilerin steganografik yöntemlerle yüksek verimlilik ile güvenli olmayan ortamlar üzerinden güvenli bir biçimde gönderilmesine yönelik yapılan ölçümler ve İYHF metodunun bu çalışmadaki yerine göre yapılan değerlendirmeler şu şekildedir. Çizelge 6.4 ile Çizelge 6.5'te verilen ölçümlerde kullanılan orijinal veri miktarının yaklaşık 3 kat artmasına rağmen Çizelge 6.6 ve Çizelge 6.7'te verilen veriler özelinde iki stego resmi karşılaştırıldığında verimliliğin steganografi bilimini göre oldukça arttığı gözlemlenmiştir. Buna göre Çizelge 6.6 ve Çizelge 6.7'deki iki stego resmi için ölçülen MSE değerleri oranladığında oranın yaklaşık 317 kat olduğu belirlenmiştir. Bir diğer ifade ile orijinal metnin gömüldüğü stego resmin MSE değeri hash kodunun gömüldüğü stego resmin MSE değerinden 317 kat büyüktür. MSE değerinin çok küçük olması stego resmin kalitesininin arttığının göstergesi olmasından dolayı Çizelge 6.4'te verilen orijinal veri büyüklüğünün artmasına rağmen MSE değerinin küçülmesi İYHF metodunun steganografik alanda kullanılmasının önemini göstermektedir.

7.1 Açık Sorular

İYHF'nin mevcut metodolojisinde özel bir anahtar sağlayabileceği ve sadece bu anahtarın sahiplerinin kök hash koduna erişebileceği söylenebilir. Ancak kök hash kodundan sonraki süreçte ortaya çıkan ortalama %0,17'lik belirsizlik için bir çözüm bulunması gerekmektedir. İYHF'nin mevcut durumunda belirsizlik oranı çeşitli koşullar altında önemli ölçüde azaltılmıştır; ancak tamamen ortadan kaldırılamamıştır. Koşulların İYHF yöntemi üzerindeki etkileri Çizelge 6.2'de listelenmiştir. Mevcut durumda belirsizliğin ortadan kaldırılamaması, geçici olarak "kripto ham veri" olarak nitelendirilen verilerin varlığını ortaya çıkarmıştır. Bu verilere olan ihtiyacın ortadan kalkması durumunda İYHF ideal çalışma seviyesine ulaşabilir.

İYHF'nin güvenlik sonuçları mevcut tek yönlü hash fonksiyonlarının verdiği sonuçlarla benzerlik göstermektedir. Ancak İYHF'nin mevcut durumunda çalışma hızı tek yönlü hash fonksiyonlarına göre düşüktür. Bunun nedeni İYHF'nin orijinal veriyi özetlemenin yanı sıra orijinal veriyi gizleme işlemlerini de içermesidir. Dolayısıyla bu konuyla ilgilenen araştırmacıların İYHF hızını artırmaya yönelik çalışmalar da yapması gerekmektedir.

8. SONUÇ

Literatürdeki tüm hash fonksiyonları tek yönlüdür. Bu çalışmanın literatüre katkısı, steganografik iletim yöntemlerinde yüksek yük ile birlikte güvenli veri iletiminde kullanmak üzere iki yönlü hash fonksiyonları kavramını tartışması ve bunun için bir metodoloji sunmasıdır. Bu çalışmada iki yönlü hash fonksiyonlarının kullanılabileceği diğer alanlar, tek yönlü fonksiyonların güvenlik özelliklerine sahip olup olamayacakları ve uygulanabilirlikleri üzerine çalışmalar yapılmıştır.

İYHF, özgün olarak geliştirilen fonksiyonlar, kavramlar ve iş akışı ile veri sıkıştırma, hash fonksiyonları ve kriptografik algoritmaların faydalarını tek bir metodolojide birleştirmektedir. İYHF, rastgele uzunluktaki verileri bir hash kodu ile ifade etmeyi ve gerektiğinde hash kodundan orijinal veriyi elde etmeyi amaçlamaktadır. Bu özelliği ile İYHF, veri depolama, veri gizliliği, veri bütünlüğü, sayısal imza uygulamaları, blokzincir ve steganografi gibi birçok alanda kullanılma potansiyeline sahiptir.

İYHF'nin güçlü yönleri metodolojisinden kaynaklanmaktadır. Bu bağlamda, yeni bir değişim fonksiyonu, veri/blok sınıflandırmaları ve bunları Huffman kodlama ile entegre eden özgün bir iş akışı sunulmuştur. Değişim fonksiyonu, Huffman kodlamasına uygun verileri bulmak için kullanılmıştır. Veriler, literatüre katkı sağlayan dört ana sınıfa ayrılmıştır. Bu çalışmanın bir diğer önemli katkısı da verilerin oluşturduğu blokların sınıflandırılması ve bunların önerilen yöntemin iş akışına entegre edilmesidir.

İYHF'nin analizi ve elde edilen sonuçlar şu şekilde özetlenebilir.

Yaygın olarak kullanılan tek yönlü hash fonksiyonlarının güvenlik test sonuçları İYHF ile karşılaştırıldığında sonuçların tatmin edici olduğu söylenebilir.

İYHF'nin iki yönlü özelliği sayesinde tek yönlü fonksiyonların kullanıldığı alanlarda daha etkin kullanılabileceği düşünülmektedir. Ayrıca İYHF yönteminin iki yönlü olması, tek yönlü hash fonksiyonlarının kullanılamadığı alanlarda da kullanımını mümkün kılabilir.

İYHF şifreli hash kodu üretim sürecinde herhangi bir sorun yaşanmamaktadır. Şifre çözme işlemi sırasında bazı verilerde belirsizlik tespit edilmiş ve bu veriler "sahte" olarak nitelendirilmiştir. Belirsizlik, bazı koşullar ve kontrol verileri kullanılarak yaklaşık 7,4 kat azaltılmıştır.

Bu çalışma çözülmemiş bir problem türündedir ve daha da geliştirilmeye açıktır. Bu çalışma, iki yönlü hash fonksiyonlarının tüm yönlerini, bu fonksiyonun varlığını tartışmakta ve geliştirilmesi için rehberlik sağlamayı amaçlamaktadır. Gelecekte, bu çalışmada sunulan metodolojinin daha fazla araştırmacının katılımıyla geliştirilmesi son derece önemlidir. Ayrıca çalışmanın bu haliyle yayınlanmasının konuyla ilgilenen araştırmacıların yeni araştırmalar başlatmasına ve böylece metodolojiye ilişkin daha etkin sonuçlar elde edilmesine neden olacağı öngörülmektedir.

KAYNAKLAR

Abood, O. G., & Guirguis, S. K. (2018). A survey on cryptography algorithms. International Journal of Scientific and Research Publications, 8(7), 495-516.

Abouchouar, A., Omary, F., & Achkoun, K. (2020). New concept for cryptographic construction design based on noniterative behavior. IAES International Journal of Artificial Intelligence, 9(2), 229-235.

Abroshan, H. (2021). A hybrid encryption solution to improve cloud computing security using symmetric and asymmetric cryptography algorithms. International Journal of Advanced Computer Science and Applications, 12(6), 31-37.

Acar, A., Aksu, H., Uluagac, A. S., & Conti, M. (2018). A survey on homomorphic encryption schemes: Theory and implementation. ACM Computing Surveys (Csur), 51(4), 1-35.

Adeniyi, E. A., Falola, P. B., Maashi, M. S., Aljebreen, M., & Bharany, S. (2022). Secure sensitive data sharing using RSA and ElGamal cryptographic algorithms with hash functions. Information, 13(10), 442.

Al-badrei, H. H., & Alshawi, I. S. (2021). Improvement of RC4 security algorithm. Adv. Mech, 9(3), 1467-1476.

Alenezi, M. N., Alabdulrazzaq, H., & Mohammad, N. Q. (2020). Symmetric encryption algorithms: Review and evaluation study. International Journal of Communication Networks and Information Security, 12(2), 256-272.

Ali, A. M., & Farhan, A. K. (2020). A novel improvement with an effective expansion to enhance the MD5 hash function for verification of a secure E-document. IEEE Access, 8, 80290-80304.

Alkandari, A. A., Al-Shaikhli, I. F., & Alahmad, M. A. (2013, September). Cryptographic hash function: A high level view. In 2013 International Conference on Informatics and Creative Multimedia (pp. 128-134). IEEE.

Almasri, O., & Jani, H. M. (2013). Introducing an Encryption Algorithm based on IDEA. International Journal of Science and Research (IJSR), India, 2(9), 334-339.

Al-Odat, Z., & Khan, S. (2019, December). Constructions and attacks on hash functions. In 2019 International Conference on Computational Science and Computational Intelligence (CSCI) (pp. 139-144). IEEE.

Alotaibi, M., Al-hendi, D., Alroithy, B., AlGhamdi, M., & Gutub, A. (2019). Secure mobile computing authentication utilizing hash, cryptography and steganography combination. Journal of Information Security and Cybercrimes Research, 2(1), 73-82.

Alqad, Z., Oraiqat, M., Almujafet, H., Al-Saleh, S., Al Husban, H., & Al-Rimawi, S. (2019). A new approach for data cryptography. International Journal of Computer Science and Mobile Computing, 8(9), 30-48.

Al-Shabi, M. A. (2019). A survey on symmetric and asymmetric cryptography algorithms in information security. International Journal of Scientific and Research Publications (IJSRP), 9(3), 576-589.

Amara, M., & Siad, A. (2011, May). Elliptic curve cryptography and its applications. In International workshop on systems, signal processing and their applications, WOSSPA (pp. 247-250). IEEE.

Anwar, M. N. B., Hasan, M., Hasan, M. M., Loren, J. Z., & Hossain, S. T. (2019). Comparative study of cryptography algorithms and its' applications. International Journal of Computer Networks and Communications Security, 7(5), 96-103.

Anwar, M. R., Apriani, D., & Adianita, I. R. (2021). Hash Algorithm In Verification Of Certificate Data Integrity And Security. Aptisi Transactions on Technopreneurship (ATT), 3(2), 181-188.

Arshad, R., Saleem, A., & Khan, D. (2016, August). Performance comparison of Huffman coding and double Huffman coding. In 2016 Sixth International Conference on Innovative Computing Technology (INTECH) (pp. 361-364). IEEE.

Babalola, A., Bamidele, E., & Esther, O. (2021). Cryptography: A Review, 3(10), 1385-1391.

Bai, J., Chang, C.C., Nguyen, T.S., Zhu, C., & Liu, Y. (2017). A high payload steganographic algorithm based on edge detection. Displays, 46, 42-51.

Bakhtiari, M., & Maarof, M. A. (2011). An efficient stream cipher algorithm for data encryption. International Journal of Computer Science Issues (IJCSI), 8(3), 247.

Barakat, M., Eder, C., & Hanke, T. (2018). An introduction to cryptography. Timo Hanke at RWTH Aachen University, 1-145.

Barman, R., Deshpande, S., Kulkarni, N., Agarwal, S., & Badade, S. (2021). A review on lossless data compression techniques. Int J Sci Res Eng Trends, 7(1), 143-148.

Barreto, P. S. L. M., & Rijmen, V. (2000, November). The Whirlpool hashing function. In First open NESSIE Workshop, Leuven, Belgium (Vol. 13, p. 14).

Bunzel, A. F. (2015). Hash Based Digital Signature Schemes.

Chang, D., Nandi, M., & Yung, M. (2011). Indifferentiability of the hash algorithm BLAKE. Cryptology ePrint Archive. https://ia.cr/2011/623

Chen, A. C. (2023, July). Using Elliptic Curve Cryptography for Homomorphic Hashing. In 2023 International Conference on Smart Systems for applications in Electrical Sciences (ICSSES) (pp. 1-5). IEEE.

Chen, Z., & Ye, G. (2022). An asymmetric image encryption scheme based on hash SHA-3, RSA and compressive sensing. Optik, 267, 169676. https://doi.org/10.1016/j.ijleo.2022.169676.

Chuah, C. W. (2009). Omega network hash construction (Doctoral dissertation, Universiti Sains Malaysia), http://eprints.uthm.edu.my/id/eprint/3706.

Debnath, S., Chattopadhyay, A., & Dutta, S. (2017, November). Brief review on journey of secured hash algorithms. In 2017 4th International Conference on Opto-Electronics and Applied Optics (Optronix) (pp. 1-5). IEEE.

Domb, M. (Ed.). (2019). Modern Cryptography: Current Challenges and Solutions. BoD–Books on Demand.

Doukas, N., Markovskyi, O. P., & Bardis, N. G. (2019). Hash function design for cloud storage data auditing. Theoretical Computer Science, 800, 42-51.

Gupta, A., & Nigam, S. (2021). A review on different types of lossless data compression techniques. International Journal of Scientific Research in Computer Science, Engineering and Information Technology, 7(1), 50-56.

Hatzivasilis, G. (2017). Password-hashing status. Cryptography, 1(2), https://doi.org/10.3390/cryptography1020010

He, Q., Li, Z., & Zhang, X. (2010, October). Data deduplication techniques. In 2010 international conference on future information technology and management engineering (Vol. 1, pp. 430-433). IEEE.

Hore, A., & Ziou, D. (2010). Image quality metrics PSNR vs. SSIM. In 2010 20th International Conference on Pattern Recognition, 2366-2369.

Intila, C., Gerardo, B., & Medina, R. (2019, February). A study of public key ‘e’in RSA algorithm. In IOP Conference Series: Materials Science and Engineering (Vol. 482, No. 1, p. 012016). IOP Publishing.

Jayasankar, U., Thirumal, V., & Ponnurangam, D. (2021). A survey on data compression techniques: From the perspective of data quality, coding schemes, data type and applications. Journal of King Saud University-Computer and Information Sciences, 33(2), 119-140.

Jindal, P., & Singh, B. (2015). A survey on RC4 stream cipher. International Journal of Computer Network and Information Security, 7(7), 37-45.

Jolfaei, A., & Mirghadri, A. (2010). Survey: image encryption using Salsa20. International Journal of Computer Science Issues (IJCSI), 7(5), 213.

Jung, K.H. (2018). A survey of interpolation-based reversible data hiding methods. Multimedia Tools and Applications, 77(7), 7795-7810.

Kheshaifaty, N., & Gutub, A. (2020). Preventing multiple accessing attacks via efficient integration of captcha crypto hash functions. Int. J. Comput. Sci. Netw. Secur.(IJCSNS), 20(9), 16-28.

Kheshaifaty, N., & Gutub, A. (2020). Preventing multiple accessing attacks via efficient integration of captcha crypto hash functions. Int. J. Comput. Sci. Netw. Secur.(IJCSNS), 20(9), 16-28.

Koptyra, K., & Ogiela, M. R. (2020). Imagechain—application of blockchain technology for images. Sensors, 21(1), 82.

Koroglu, T., & Samet, R. (2023). Secure steganographıc data transmission method for periodically updated data. Journal of Modern Technology & Engineering, 7(3), 153-171.

Kumar, S., Prabhu, S. A., Durga, R., & Jeevitha, S. (2016). A survey on various asymmetric algorithms. International Research 7. Journal of Advanced Engineering and Science, 1(4), 117-119.

Kumarı A., & Roy, B. (2018). A Survey of Lattice Attack on Digital Signature Algorithm, Conference on Internet of Things and Connected Technology, 942-946.

Landge, I. A., & Satopay, H. (2018, February). Secured IoT through hashing using MD5. In 2018 fourth international conference on advances in electrical, electronics, information, communication and bio-informatics (AEEICB) (pp. 1-5). IEEE.

Li, P., & Lu, A. (2018). LSB-based steganography using reflected gray code for color quantum images. International Journal of Theoretical Physics, 57(5), 1516-1548.

Li, Y., & Ge, G. (2019). Cryptographic and parallel hash function based on cross coupled map lattices suitable for multimedia communication security. Multimedia Tools and Applications, 78(13), 17973-17994.

Liu, X., An, P., Chen, Y., & Huang, X. (2022). An improved lossless image compression algorithm based on Huffman coding. Multimedia Tools and Applications, 81(4), 4781-4795.

Lizama-Pérez, L. A., Montiel-Arrieta, L. J., Hernández-Mendoza, F. S., Lizama-Servín, L. A., & Simancas-Acevedo, E. (2019). Public hash signature for mobile network devices. Ingeniería, investigación y tecnología, 20(2).

Madhuravani, B., & Murthy, D. S. R. (2013). Cryptographic hash functions: SHA family. International Journal of Innovative Technology and Exploring Engineering, 2(4), 326-329.

Maetouq, A., Daud, S. M., Ahmad, N. A., Maarop, N., Sjarif, N. N. A., & Abas, H. (2018). Comparison of hash function algorithms against attacks: A review. International Journal of Advanced Computer Science and Applications, 9(8), 98-103.

Mallouli, F., Hellal, A., Saeed, N. S., & Alzahrani, F. A. (2019, June). A survey on cryptography: comparative study between RSA vs ECC algorithms, and RSA vs El-Gamal algorithms. In 2019 6th IEEE International Conference on Cyber Security and Cloud Computing (CSCloud)/2019 5th IEEE International Conference on Edge Computing and Scalable Cloud (EdgeCom) (pp. 173-176). IEEE.

Maqsood, F., Ahmed, M., Ali, M. M., & Shah, M. A. (2017). Cryptography: a comparative analysis for modern techniques. International Journal of Advanced Computer Science and Applications, 8(6), 442-448.

Martino, R., & Cilardo, A. (2019). A flexible framework for exploring, evaluating, and comparing SHA-2 designs. IEEE Access, 7, 72443-72456.

Mohammed, H. A., & Al Saffar, N. F. H. (2021). LSB based image steganography using McEliece cryptosystem. Materials Today: Proceedings. https://doi.org/10.1016/j.matpr.2021.07.182

Mohammed, H.A., Al Saffar, N.F.H. (2021). LSB based image steganography using McEliece cryptosystem. Materials Today: Proceedings. https://doi.org/10.1016/j.matpr.2021.07.182.

Mushtaq, M. F., Jamel, S., Disina, A. H., Pindar, Z. A., Shakir, N. S. A., & Deris, M. M. (2017). A survey on the cryptographic encryption algorithms. International Journal of Advanced Computer Science and Applications, 8(11), 333-344.

Padmavathi, B., & Kumari, S. R. (2013). A survey on performance analysis of DES, AES and RSA algorithm along with LSB substitution. IJSR, India, 2, 2319-7064.

Paterson, L. J., & Glasbey, C. A. (1985). 69.21 An illustration of rounding error on computers. The Mathematical Gazette, 69(448), 128-131.

Piccolboni, L., Di Guglielmo, G., Carloni, L. P., & Sethumadhavan, S. (2021, May). Crylogger: Detecting crypto misuses dynamically. In 2021 IEEE Symposium on Security and Privacy (SP) (pp. 1972-1989). IEEE.

Qadir, A. M., & Varol, N. (2019, June). A review paper on cryptography. In 2019 7th international symposium on digital forensics and security (ISDFS) (pp. 1-6). IEEE. 10.1109/ISDFS.2019.8757514

Rahman, M. A., & Hamada, M. (2019). Lossless image compression techniques: A state-of-the-art survey. Symmetry, 11(10), 1274.

Rahul, B., Kuppusamy, K., & Senthilrajan, A. (2023). Dynamic DNA cryptography-based image encryption scheme using multiple chaotic maps and SHA-256 hash function. Optik, 289, 171253.

Rajasekaran, A. S., Azees, M., & Al-Turjman, F. (2022). A comprehensive survey on blockchain technology. Sustainable Energy Technologies and Assessments, 52, 102039.

Rajeshwaran, K., & Kumar, K. A. (2019, February). Cellular automata based hashing algorithm (CABHA) for strong cryptographic hash function. In 2019 IEEE International Conference on Electrical, Computer and Communication Technologies (ICECCT) (pp. 1-6). IEEE.

Rathod, U., Sonkar, M., & Chandavarkar, B. R. (2020, July). An experimental evaluation on the dependency between one-way hash functions and salt. In 2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT) (pp. 1-7). IEEE.

Sabitha, S., & Nair, B. V. (2020). Survey on Asymmetric Key Cryptographic Algorithms, International Journal of Scientific Research in Science, 7(2), 404-407.

Saieed, A. H., & Hattab, A. A. (2023, December). Modifications and improvements to the two fish encryption algorithm: A review. In AIP Conference Proceedings (Vol. 2834, No. 1). AIP Publishing.

Salah, S. K., Humood, W. R., Khalaf, A. O., & Abdalrdha, Z. K. (2019). Subject Review: Comparison Between 3DES, AES and HiSea Algorithms. Int. J. Sci. Res. Sci. Eng. Technol, 6(6), 97-103.

Seok, B., Park, J., & Park, J. H. (2019). A lightweight hash-based blockchain architecture for industrial IoT. Applied Sciences, 9(18), 3740.

Sharma, A. K., & Mittal, S. K. (2019, January). Cryptography & network security hash function applications, attacks and advances: A review. In 2019 Third International Conference on Inventive Systems and Control (ICISC) (pp. 177-188). IEEE.

Sharma, S., Patel, K. N., & Jha, A. S. (2021, December). Cryptography Using Blowfish Algorithm. In 2021 3rd International Conference on Advances in Computing, Communication Control and Networking (ICAC3N) (pp. 1375-1377). IEEE.

Shen, Y., Sun, Z., & Zhou, T. (2021, September). Survey on asymmetric cryptography algorithms. In 2021 International Conference on Electronic Information Engineering and Computer Science (EIECS) (pp. 464-469). IEEE.

Shetty, V. S., Anusha, R., MJ, D. K., & Hegde, P. (2020, February). A survey on performance analysis of block cipher algorithms. In 2020 International Conference on Inventive Computation Technologies (ICICT) (pp. 167-174). IEEE.

Shyla, M.K., Kumar, K.S., & Das, R.K. (2021). Image steganography using genetic algorithm for cover image selection and embedding. Soft Computing Letters, 3, 100021.

Singh, S., Iqbal, M. S., & Jaiswal, A. (2015). Survey on techniques developed using digital signature: public key cryptography. International Journal of Computer Applications, 117(16).

Sodhi, G. K., & Gaba, G. S. (2018). An efficient hash algorithm to preserve data integrity. Journal of Engineering Science and Technology, 13(3), 778-789.

Sood, R., & Kaur, H. (2023). A literature review on RSA, DES and AES encryption algorithms. Emerging Trends in Engineering and Management, 57-63.

Srivastava, V., Baksi, A., & Debnath, S. K. (2023). An Overview of Hash Based Signatures. Cryptology ePrint Archive. https://ia.cr/2023/411

Subhedar, M. S., & Mankar, V. H. (2018). Curvelet transform and cover selection for secure steganography. Multimedia Tools and Applications, 77(7), 8115-8138.

Sumartono, I., Siahaan, A. P. U., & Mayasari, N. (2016). An overview of the RC4 algorithm. IOSR J. Comput. Eng, 18(6), 67-73.

Tchórzewski, J., & Jakóbik, A. (2019). Theoretical and experimental analysis of cryptographic hash functions. Journal of Telecommunications and Information Technology, (1), 125-133.

Vanmathi, C., & Prabu, S. (2018). Image steganography using fuzzy logic and chaotic for large payload and high imperceptibility. International Journal of Fuzzy Systems, 20(2), 460-473.

Verma, J., Shahrukh, M., Krishna, M., & Goel, R. (2021). A critical review on cryptography and hashing algorithm SHA-512. International Research Journal of Modernizationin Engineering, 3(12), 1760-1764.

Wang, J., Liu, G., Chen, Y., & Wang, S. (2021). Construction and analysis of SHA-256 compression function based on chaos S-box. IEEE Access, 9, 61768-61777.

Wang, L., & Jiang, G. (2019, January). The design of 3-DES encryption system using optimizing keys. In 2019 China-Qatar International Workshop on Artificial Intelligence and Applications to Intelligent Manufacturing (AIAIM) (pp. 56-58). IEEE.

Wang, X., Wang, S., Wei, N., & Zhang, Y. (2019). A novel chaotic image encryption scheme based on hash function and cyclic shift. IETE Technical Review, 36(1), 39-48.

Waseso, B. M. P., & Setiyanto, N. A. (2023). Web Phishing Classification using Combined Machine Learning Methods. Journal of Computing Theories and Applications, 1(1), 11-18.

William, P., Choubey, A., Chhabra, G. S., Bhattacharya, R., Vengatesan, K., & Choubey, S. (2022, March). Assessment of hybrid cryptographic algorithm for secure sharing of textual and pictorial content. In 2022 International conference on electronics and renewable systems (ICEARS) (pp. 918-922). IEEE.

William, P., Choubey, A., Chhabra, G. S., Bhattacharya, R., Vengatesan, K., & Choubey, S. (2022, March). Assessment of hybrid cryptographic algorithm for secure sharing of textual and pictorial content. In 2022 International conference on electronics and renewable systems (ICEARS) (pp. 918-922). IEEE.

Yang, S., Piao, H., Zhang, L., & Zheng, X. (2007, August). An improved idea algorithm based on usb security key. In Third International Conference on Natural Computation (ICNC 2007) (Vol. 3, pp. 184-188). IEEE.

Yang, Y., Chen, F., Zhang, X., Yu, J., & Zhang, P. (2017). Research on the hash function structures and its application. Wireless Personal Communications, 94, 2969-2985.

Yiakoumis, I. I., Papadonikolakis, M. E., Michail, H. E., Kakarountas, A. P., & Goutis, C. E. (2006). Maximizing the hash function of authentication codes. IEEE Potentials, 25(2), 9-12.

Yihan, W., & Yongzhen, L. (2021, January). Improved design of DES algorithm based on symmetric encryption algorithm. In 2021 IEEE International Conference on Power Electronics, Computer Applications (ICPECA) (pp. 220-223). IEEE.

Yousif, S. F. (2021, February). Secure voice cryptography based on Diffie-Hellman algorithm. In IOP Conference Series: Materials Science and Engineering (Vol. 1076, No. 1, p. 012057). IOP Publishing

Zellagui, A., Hadj-Said, N., & Ali-Pacha, A. (2019). Comparative Study Between Merkle-Damgård And Other Alternative Hashes Construction. In Second conference on informatics and Applied Mathematics IAM (Vol. 180, pp. 30-34).

Zhai, S., Yang, Y., Li, J., Qiu, C., & Zhao, J. (2019, February). Research on the Application of Cryptography on the Blockchain. In Journal of Physics: Conference Series (Vol. 1168, p. 032077). IOP Publishing.

Zhou, C., Zhu, G., Zhao, B., & Wei, W. (2006, November). Study of one-way hash function to digital signature technology. In 2006 International Conference on Computational Intelligence and Security (Vol. 2, pp. 1503-1506). IEEE.