T.C
FIRAT ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

MOBİL AYGITLAR İÇİN TEK KULLANIMLIK ŞİFRE ÜRETİLMESİ VE GÜVENLİĞİNİN SINANMASI

YÜKSEK LİSANS TEZİ

M.Şahin AÇIKKAPI

(091129104)

Tezin Enstitüye Verildiği Tarih : 15 Ağustos 2011

Tezin Savunulduğu Tarih : 06 Eylül 2011

Tez Danışmanı : Yrd. Doç. Dr. A. Bedri ÖZER

Diğer Jüri Üyeleri : Doç. Dr. İbrahim TÜRKOĞLU

Yrd. Doç. Dr. Ahmet ÇINAR

EYLÜL-2011

ÖNSÖZ

Bu çalışmanın hazırlanması esnasında her zaman iyi niyet, sabır ve hoşgörüsüyle çalışmama büyük katkı sağlayan danışmanım sayın Yrd. Doç. Dr A. Bedri ÖZER'e ve öğrenim hayatımda geldiğim noktada en büyük pay sahibi olan aileme teşekkürü bir borç bilirim.

M.Şahin AÇIKKAPI

ELAZIĞ-2011

İÇİNDEKİLER

Sayfa No:

ÖZET

Günümüzde hızla gelişen teknoloji, insan hayatını birçok yönden değiştirmektedir. Özellikle bilişim teknolojilerindeki gelişmeler, gerek bize kazandırdığı zaman gerekse etkinlikleri itibarıyla günümüz insanının iş, iletişim, bilgi edinme gibi birçok alışkanlığını temelden değiştiriyor. Gelişmiş bilgisayar sistemleri ve özellikle internet ağı, küresel bazda hemen her çeşit bağlantıya izin vermektedir. Gelişen bu teknolojide, özellikle internet tabanlı uygulamaların yaygınlaşmasıyla beraber bilgisayar sistemlerine erişen yetkili ve yetkisiz kullanıcıların birbirinden ayrılması konusu büyük önem kazanmıştır. Örneğin; İnternet bankacılığı kullanan kişiler, kullanıcı adı ve şifre bilgileriyle banka sistemine erişmektedirler. Bu kullanıcılar kullanıcı adı ve sabit şifre bilgisi kullandıklarında sistemlerine sızan bilgisayar korsanları bu bilgileri ele geçirip daha sonra bu bilgilerle kişinin banka hesabına erişip kanunsuz işlemler yapabilmektedirler. OTP (One Time Password) tek kullanımlık şifre sistemleri burada devreye girmekte ve kullanıcıya sisteme her erişimde farklı bir şifre kullanma imkanı vermektedir. Bu tez kapsamında Mobil aygıtlar için geliştirilen tek kullanımlık şifre yazılımı, haberleşme ortamında bahsedilen dezavantajları ortadan kaldırmakta ve kullanıcı güvenliğini maksimum seviyeye çıkarmaktadır.

Anahtar Kelimeler: Simetrik Şifreleme, Asimetrik Şifreleme, Özet Fonksiyon, DES, RSA, MD5, SHA-1, SHA-2, MAC, HMAC, OTP, HOTP

SUMMARY

Today fast-growing technology changes human life in many respects. Especially developments in information technologies greatly change many practices of today's human being such as business life, communication, knowledge acquisition by means of the speed and also efficiency it provides. Improved computer systems and especially the Internet network enables almost all sorts of connection on a global basis. In this developing technology, the subject of separation of authorized and unauthorized users who accessed to computer systems has gained great important especially with the spread of Internet-based applications. For example; people who are using internet banking, access to bank system by user name and password. When these users use user name and static password, the computer hackers may take these information and then they may make illegal transactions with these information by accessing personal bank account. OTP (One Time Password) one time password systems enter here and provides the possibility to use a different password by the user to access to the system for every time. In the scope of the thesis, one time password software which is generated for mobile devices, eliminates the above-mentioned disadvantages for communication environment, and produces the maximum level for the security of the user.

Key Words: Symmetric Encryption, Asymmetric Encryption, Hash Function, DES, RSA, MD5, SHA-1, SHA-2, MAC, HMAC, OTP, HOTP

ŞEKİLLER LİSTESİ

Sayfa No

Sayfa no:

SEMBOLLER LİSTESİ

KISALTMALAR

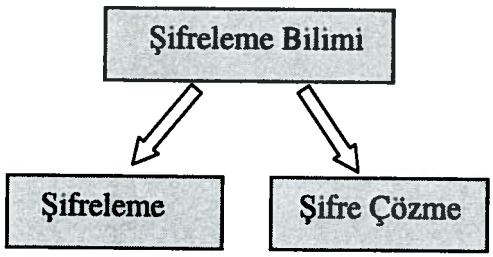
DES : Data Encryption Standard
AES : Advanced Encryption Standard
NIST : National Institute of Standards and Technology
NSA : National Security Agency
P : Plain Text
C : Cipher Text
D : Decryption Algorithm
E : Encryption Algorithm
K : Key
SAC : Strict Avalanche Criteria
FIPS : Federal Information Processing Standard
MAC : Message Authentication Code
HMAC : Hash Based Message Authentication Code
OTP : One Time Password
HOTP : Hash Based One Time Password
SHA : Secure Hash Algorithm
ECC : Elliptic Curve Cryptography

1. GİRİŞ

Çağımız artık bilgi çağı olarak adlandırılmakta, bilgi teknolojileri hayatımızın her alanında kullanılmaktadır. Bugün ülkeler askerlerden oluşan ordularının yanında sanal alem için "siber ordular" oluşturma gayreti içerisindelerdir. Teknoloji kullanılırken veri güvenliği konusu çok önemli bir etken olarak karşımıza çıkmaktadır. Günümüz Türkiyesinde kişiler, e-devlet kullanımı, internet bankacılığı gibi birçok sistemi evlerinden veya işyerlerinden kullanmaktadırlar, eskiden saatler/günler süren işlerini saniyeler içerisinde internet üzerinden yerine getiriyorlar. Ancak bu kullanımlar esnasında güvenlik konusu en büyük çekince olmaktadır. Örneğin internet bankacılığını güvenlik zaafiyetleri nedeniyle kullanmak istemeyen birçok insan bulunmaktadır. İşte bu güvenlik sıkıntıları özellikle son yıllarda şifreleme biliminin önemini çok büyük oranda artırmış bulunmaktadır. Şifreleme biliminde büyük bir önemi olan özetleme algoritmalarıda bize bir çok yönden güvenlik hizmeti sunmakta, bu hizmetlerden biride OTP (One Time Password) sistemine olan katkısıdır. OTP tek kullanımlık şifre yöntemleri internet üzerinden kullanılan sistemlerde güvenliğimize büyük katkı sağlamaktadır. Yakın zamanda yapılan düzenlemelerle tüm bankalar kullanıcılarına tek kullanımlık şifre hizmeti vermeye başlamıştır. Bu şifreler kimi zaman cep telefonlarına sms mesajı olarak gelmektedir. Bazı bankalar kullanıcısına OTP cihazları vermekte, yine bazı bankalar kullanıcı telefonunlarına tek kullanımlık şifre yazılımını yükletmektedirler. Mobil aygıtlarda kullanılabilecek OTP uygulaması bunlar arasında maliyet ve güvenlik açısından en avantajlısıdır.

1.1. Şifreleme Bilimi (Kriptoloji) Nedir?

Sözcük olarak kriptoloji Yunanca kryptos (gizli) ve logos (kelime) kelimelerinin bir araya getirilmesinden oluşmuştur. Kavram olarak şifreleme bilimi; Veri alış verişinde bulunan iki veya daha fazla tarafın veri alışverişlerini güvenli bir şekilde yapmalarını sağlayan ve bunu sağlamak içinde çözülmesi çok zor matematiksel işlemler ve problemler kullanan bilim dalıdır. Günümüzde ise şifreleme bilimi; matematik, elektronik, optik, bilgisayar bilimleri gibi birçok disiplinden faydalanan özelleşmiş bir bilim dalı olarak bilim dünyasında yerini almıştır. Şekil 1.1' de görüldüğü gibi şifreleme biliminin iki temel alt dalı vardır[1]:



Şekil 1.1 Şifreleme biliminin temel alt dalları

Şifreleme (Kriptografi), verinin şifrelenmesi süreci ile ilgilenen yöntemlere verilen genel addır.

Şifre Çözme (Kriptoanaliz), şifreleme sisteminin oluşturduğu veriyi inceler ve çözmeye çalışır. Şifreleme bilimi içerisinde şifre çözmenin önemi çok büyüktür, çünkü ortaya koyulan bir şifreleme sistemini inceleyerek, zayıf ve kuvvetli yönlerini ortaya koymak için şifre çözme kullanılır.

Son yıllarda elektronik bilgi sistemlerinin yaygınlaşması şifreleme biliminin önemini çok fazla arttırmıştır. Şifreleme biliminin önde gelen kullanım alanı, iletim halindeki veya saklanmış bilginin şifrelenmesi ve ihtiyaç duyulduğunda bu şifrenin çözülmesidir. Şifreleme biliminin ana malzemesi bilgi olduğundan ve günümüzde bilgi güvenliğinin önemi çok fazla arttığından, şifreleme biliminin birçok uygulamada kullanılması kaçınılmaz olmuştur[2].

1.2. Şifreleme Biliminin Tarihçesi

Şifreleme biliminin tarihi çok eski zamanlara kadar uzanmaktadır. Bu tarihçeye kısaca bakılacak olursa[2]:

- ✓ MÖ. 5. yüzyılda yaşayan Yunanlı tarihçi ve yazar Heredot' un anlattıklarına göre, yazarın zamanında gizli haberleşmeyi sağlayabilmek için bazı kölelerin saçları tamamen kesilir, gizli mesaj bu kölelerin başına yazılırdı. Ve köle, saçlarının uzamasından sonra mesajı götüreceği yere giderdi. Köle karşı tarafa ulaşınca saçları burada kesilir ve gizli mesaj alınırdı[3].

- ✓ MÖ.50' li yıllarda Julius Caesar alfabedeki harflerin yerlerini değiştirerek bir şifreleme sistemi kurdu. Bu sistemde her harf kendisinden üç sonraki harf ile ifade edilirdi. Örneğin 'A' harfi geçen yerlere 'D' harfi yazılırdı.

- ✓ Gaznelilere ait olup 1000-1200 yılları arasında yazıldığı tahmin edilen ve günümüze kadar ulaşan bazı dökümanlarda şifreli metinlere rastlanmıştır.

- ✓ 1523-1526 yılları arasında yaşayan Blaise de Vigenere 1586 yılında şifrelemeyle ilgili bir kitap kaleme almıştır. Vigenere oluşturduğu sistemde, kendisinin oluşturduğu ve daha sonraları Vigenere tablosu olarak adlandırılan bir tablo ve bir anahtar yardımıyla şifreleme işlemi yapabiliyordu. Bu günümüzde kullanılan bazı şifreleme yöntemlerininde temeli olmuştur.

- ✓ 1790' da Thomas Jefferson, Strip Cipher makinesini geliştirdi. Jefferson' un tasarladığı bu cihazda 36 disk vardı ve bu disklerin her birinde alfabenin harfleri yazılmıştı[4].

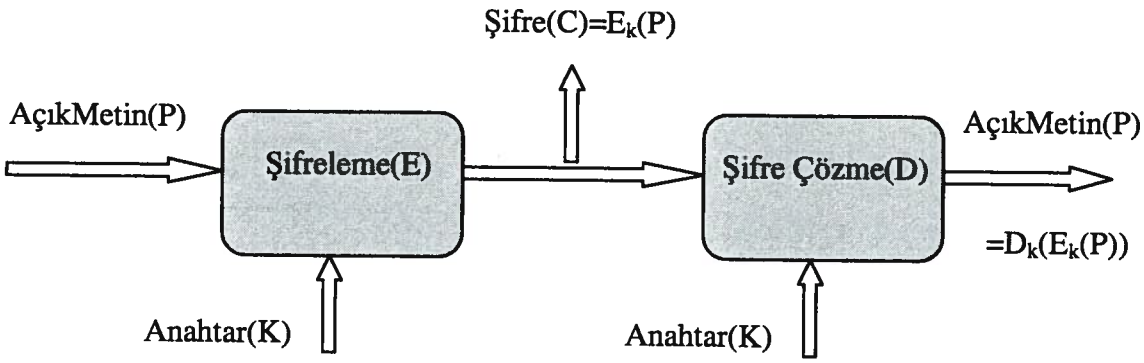
- ✓ Joseph Mauborgne ve Gilbert Vernam 1917 yılında Vernam şifreleme yöntemi olarak isimlendirilen 'Van Time Pad' i bulmuşlardır.

- ✓ 2. Dünya savaşında Almanlar Arthur Scherbius' un icat ettiği Enigma makinasını kullanmışlar, ancak Alan Matthison Turing ve arkadaşları bu şifreleri çözmeyi başarmışlar ve bu sayede Amerika ve İngiltere Almanyaya büyük kayıplar verdirmişlerdir.

- ✓ 1976' da DES (Data Encryption Standard) algoritması, ABD tarafından standard şifreleme yöntemi olarak kabul edildi ve FIPS 46 (Federal Information Processing Standard) standardı olarak açıklandı.

- ✓ 1990' lı yıllarda DES algoritmasının çözülmesi sonucu 1997' de ABD' nin NIST (National Institute of Standards and Technology) kurumu DES' in yerini alacak bir simetrik algoritma için yarışma açtı. 2001 yılında NIST' in yarışmasını Belçikalı Joan Daemen ve Vincent Rijmen' e ait Rijndael algoritması kazandı ve bu algoritma AES (Advanced Encryption Standard) adıyla standart haline getirildi[2].

1.3. Şifreleme Terminolojisi



Şekil 1.2 Şifreleme ve şifre çözme işleminin blok şeması

Temel şifreleme ve şifre çözme blok şeması Şekil 1.2' de görülmektedir. Temel terimler açıklanacak olursa:

Açık Metin (PlainText): Şifrelenecek açık metin veya data (ses,görüntü...).

Şifreli Metin (Cipher Text): Şifreleme algoritmasından sonra oluşacak olan metindir. İki farklı anahtar iki farklı sonuç üretir.

Şifreleme Algoritması (Encryption Algorithm): Açık metinden şifreli metine geçiş yapan algoritma.

Güvenli Anahtar (Secret Key): Şifreleme ve şifre çözme algoritmalarının giriş parametresidir, açık mesajdan ve algoritmadan bağımsızdır.

Şifre Çözme Algoritması (Decryption Algorithm): Şifreleme algoritmasını tersinden uygular. Giriş olarak anahtar ve şifreli metni alır, açık metin üretir.

Genel Anahtar (Puplic key): Herkesin görebileceği anahtar.

Gizli anahtar (Private key): Herkese açık değildir. Verinin şifrelenmiş halini açmak için kullanılan anahtardır.

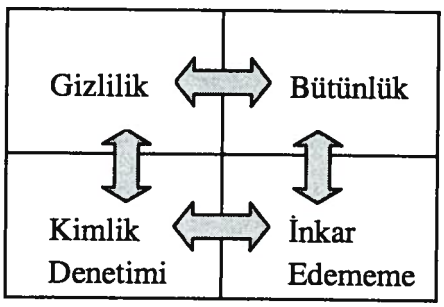
1.4. Şifreleme Yönteminde Aranan Özellikler

Şifreleme yönteminde aranan temel özellikler şunlardır:

- Şifreleme ve şifre çözme işlemlerinde çözümü zor algoritmalar kullanılmalıdır.
- Arzu edilen güvenlikle şifreleme algoritmasının zorluk derecesi doğru orantılı olmalıdır.
- Anahtarın seçimi ve şifreleme algoritması özel şartlara bağlı olmamalıdır. Yani her türlü veriye uygulanabilmelidir.
- Çok önemli bir özellik olarak; Şifreleme adımlarında yapılabilecek hatalar diğer adımlara yansımamalıdır.
- Algoritmaların karıştırma özelliği olmalı, açık mesaj ile şifreli mesaj arasında hiçbir şekilde ilişki kurulamamalıdır.
- Algoritmanın dağıtma özelliği olmalıdır. Algoritmanın şifrelediği veriler şifreli metinde karışık sıralarda yer almalıdır.

1.5. Şifrelemede Temel Amaçlar

Şifreleme genel olarak Şekil 1.3' te görülen şu temel konularla ilgilenir[5]:



Şekil 1.3 Şifrelemenin temel amaçları

1-Gizlilik (Veri Güvenliği-Confidentiality): Bilgi istenilmeyen kişiler tarafından anlaşılamamalıdır. Veri güvenliği sağlamanın; fiziksel korumalardan matematiksel algoritmalara kadar birçok yöntemi bulunmaktadır.

2-Veri Bütünlüğü (Data Integrity): Bilginin içeriğinin iletim esnasında değiştirilememesidir. Yani yetkisiz kişiler iletim esnasında veri kullanımında bulunursa, bu durum fark edilmelidir. Veri kullanımı; ekleme, silme ve değiştirme gibi işlemleri içerir.

3-Reddedilemezlik (İnkar Edememe-Non Repudiation): Bilgiyi gönderen veya işleyen kişinin yaptığı işi sonradan inkar edememesidir.

4-Kimlik Belirleme (Kanıtlama-Authentication): Bir iletişim içerisine giren kişilerin kimliklerinin doğruluğundan emin olunmasıdır. Kanıtlama işlemi, gönderilen bilginin doğruluğunu; bilginin kaynağı, başlangıç tarihi, veri içeriği, gönderme zamanı gibi özelliklerini dikkate alarak sağlamalıdır.

Elektronik tehditlere karşı kullanılan yöntemlerin şifrelemenin temel amaçlarını ne ölçüde karşıladığı Şekil 1.4' te sunulmuştur.

	Kimlik Doğrulama	**Gizlilik**	**Bütünlük**	**İnkar Edememe**
Antivirüs	--	--	Sağlar	--
Güvenlik Duvarları	Sağlar	Sağlar	--	--
Erişim Denetimi	Sağlar	Sağlar	--	--
Şifreleme	--	Sağlar	--	--
Açık Anahtar Alt Yapısı	Sağlar	Sağlar	Sağlar	Sağlar

Şekil 1.4 Elektronik tehditlere karşı kullanılan yöntemlerin karşılaştırılması

1.6. Saldırı Çeşitleri

Şifreli metne birçok saldırı çeşiti vardır. Bu saldırı çeşitlerini genel olarak açıklayacak olursak[6];

- **Sadece Şifreli Metin Saldırısı**: Bu saldırıda şifre çözen kişi yalnızca şifreli metin ve şifreleme algoritması bilgilerine sahiptir.
- **Bilinen Açık Metin Saldırısı**: Şifre çözen kişi; aynı gizli anahtarla şifrelenmiş açık ve şifreli metin çiftine, şifreleme algoritmasına ve kodu çözülecek şifreli metine sahiptir.
- **Seçilmiş Açık Metin Saldırısı**: Şifre çözen; şifreleme algoritmasına, kodu çözülecek şifreli metine ve kendisi tarafından seçilen açık metin ve bu metnin şifreli haline sahiptir.
- **Seçilmiş Şifreli Metin Saldırısı**: Şifre çözen; şifreleme algoritmasına, kodu çözülecek şifreli metine ve kendisi tarafından seçilen şifreli metin ve bu metnin düz haline sahiptir.
- **Seçilen Metin Saldırısı**: Şifre çözen; şifreleme algoritmasına, kodu çözülecek şifreli metine, kendisi tarafından seçilen açık metin ve bu metnin şifreli haline ve yine kendisi tarafından seçilen şifreli metin ve bu metnin düz haline sahiptir.
- **Kaba Kuvvet Saldırısı** (Brute Force Attack): mümkün olan tüm anahtar kombinasyonlarının denenmesi şeklinde uygulanır. Başarı için beklenen süre, tüm anahtar kombinasyonlarının yarısının denenmesi için gereken süredir.

1.6.1. Diferansiyel Saldırı

Bu saldırı yöntemi ilk kez 1990 yılında Eli Biham ve Adi Shamir tarafından yayınlanmıştır[7]. Eli Biham ve Adi Shamir, DES ve benzeri algoritmalarda iki farklı girişin farkının, çıkış değerlerinin farkı üzerindeki dağılımını inceleyerek gizli anahtarı elde etmeye çalışmışlardır. Aslında bu yöntem DES geliştirilirken fark edilmiş ve DES' in bu saldırıya karşı güvenli olması sağlanmıştır, ancak bu güvenlik açığı kullanılmaması için açıklanmamıştır.

Günümüzde bilinen en önemli saldırılardan birisidir çünkü bu yöntem sayesinde DES algoritmasının anahtarları, teorik olarak tüm anahtar uzayını denemeyle geçecek zamandan daha kısa sürede elde edilebilmektedir. Bu saldırı türü, şifreleme sistemlerinin yeniden gözden geçirilmesine, tasarım kriterlerinin tekrar ele alınmasına ve yeni sistemlerin şifre çözme saldırılarına karşı daha dayanıklı tasarlanmalarına neden olmuştur. Bu yönüyle şifrelemeye büyük katkısı olmuştur.

Bu saldırı yöntemi açık metin ikilileri farkının bunlara karşılık gelen şifreli metin ikilileri üzerindeki etkisini kullanarak analiz yapar. Bu farklar ihtimali en yüksek anahtarları belirlemek için kullanılır. Saldırı esnasında aynı farka sahip olan bir çok açık metin ikilisi ve bunlara karşılık gelen şifreli metin ikilileri kullanılır[8].

1.7. Şifreleme Algoritmalarının Performans Kriterleri

Bir şifreleme algoritmasının performansı aşağıdaki maddelerle doğrudan ilişkilidir.

- Şifrenin kırılabilme süresinin uzunluğu.
- Şifreleme ve şifre çözme işlemleri esnasında harcanan zaman.
- Şifreleme ve şifre çözme işlemi için ihtiyaç duyulan bellek miktarı.
- Algoritmalara dayalı şifreleme uygulamalarının esnekliği.
- Uygulamaların dağıtımındaki kolaylık veya algoritmaların standart hale getirilebilmesi.
- Algoritmaların kullanılacakları sistemlere uygunluğu.

1.8. Şifreleme Yöntemleri

Şifreleme algoritmalarında kullanılan yöntemleri iki kategoride inceleyebiliriz;

- Klasik Yöntemler (Sezar, Vigenere,...)
- Modern Yöntemler (DES, AES, RSA...)

1.8.1. Klasik Şifreleme Yöntemleri

Teknolojinin henüz çok fazla gelişmediği zamanlarda uygulanan şifreleme yöntemlerine genel olarak klasik şifreleme yöntemleri denilmektedir. Bu yöntemler şifrelemenin temeli ve tarihi açısından oldukça önemlidirler.

1.8.1.1. Sezar Şifreleme Yöntemi

Mono alfabetik bir şifreleme yöntemidir. M.Ö 50' li yıllarda Sezar tarafından uygulanmıştır. Alfabede her harfin kendisinden 3 sonraki harf ile yer değişmesi esasına dayanmaktadır[9,10].

Açık Metin : ABCDEFGHIJKLMNOPQRSTUVWXYZ

Şifreli Metin : DEFGHIJKLMNOPQRSTUVWXYZABC

Bu şifreleme yönteminin kırılması çok kolaydır. Frekans sayımı ile çok kolay kırılabilir. Örneğin: Türkçede frekansı en yüksek olan (en fazla kullanılan) harf 'A' harfidir. Şifreli metinde en fazla 'D' harfi olduğundan bu harfin 'A' harfi olduğu tahmin edilebilir. 25 farklı deneme ile açık metne ulaşılabilir.

1.8.1.2. Vigenere Şifreleme Yöntemi

16. yüzyılın sonlarında Blaise De Vigenere tarafından geliştirilmiştir. Bu yöntemde Şekil 1.5' teki tablodan yararlanılır. İngiliz alfabesi için bu tablo 26x26 boyutundadır[11].

Şifreleme İşlemi : Açık metin ile anahtarın kesişimi alınır.

Açık Metin : k r i p t o

Anahtar : i f g f i d

Şifreli Metin : s w o u b r

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

Şekil 1.5 Vigenere tablosu

Şifre Çözme: Anahtar ile kesişimi şifreli metni veren karakter açık metindir[12].

Şifreli Metin : s w o u b r

Anahtar : i f g f i d

Açık Metin : k r i p t o

Bu şifre bilgisayar yardımıyla frekans sayımı ile kolayca çözülebilmektedir.

1.8.1.3. Hill Şifreleme Yöntemi

Matris çarpımları ve modüler aritmetik kullanarak şifreleme yapan bir yöntemdir. Şekil 1.6' da görüldüğü gibi alfabedeki harfler sayısal değerlerle eşleştirilir.

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	Y	Z
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24

Şekil 1.6 Hill şifresi kod tablosu

Şifreleme İşlemi: Veri üçerli karakterler olarak şifrelenecekse anahtar 3x3 boyutunda seçilir. Açık metindeki harflerin kodu ile anahtar matrisin çarpımının mod 26' daki değeri bize şifreli metni verir.

Açık metin : PAY MOR (15,0,24,12,14,17)

17 17 5

Anahtar matris : K=[21 18 21] olsun.

2 2 19

15 17 17 5 375

Şifreli Metin = [0] x [21 18 21] = [819] mod 26

24 2 2 19 486

11

Şifreli Metin = [13] = L N S

18

'PAY' metninin şifrelenmiş hali 'LNS' olarak bulunur. Aynı şekilde diğer 3 karakterlik bloklarda şifrelenerek şifreli metne ulaşılmış olur[12].

1.8.1.4. Vernam (One Time Pad) Şifreleme Yöntemi

İlk olarak Vernam tarafından bulunmuştur. Bu şifreleme yönteminde rasgele üretilen bir karakter (harf veya rakam) dizisi kulanılarak şifreleme yapılır. Açık mesaj içinde yer alan

her karakter, üretilen dizide karşısına denk gelen karakterle XOR işlemine tabi tutularak şifreli mesaj elde edilir. Mesajı çözmek için rastgele dizinin bilinmesi gereklidir[2].

Açık Mesaj : BULUSMAYERIELAZIG

Rastgele Dizi : DEFRYPLCNMLJKHFGH

Şifreli Mesaj : RLDYDOY.....

Bu yöntemin güvenliği rasgele üretilen diziye bağlıdır. Bu dizi gerçekten rastgele üretilmelidir, eğer dizi bir kurala bağlı olarak üretilirse ve bu kural saldırgan tarafından bilinirse sistem kırılabilir. Bu tehdit dışında sistem mükemmel bir şifreleme sistemidir[6].

2. MODERN ŞİFRELEME ALGORİTMALARI

Günümüzde kullanılan şifreleme yöntemleri modern şifreleme algoritmaları olarak tanımlanırlar. Bu algoritmalar, kullandıkları anahtar biçimine göre simetrik veya asimetrik algoritmalar olarak adlandırılırlar. Modern şifreleme algoritmaları klasik şifreleme yöntemlerinin zayıf yönlerini ortadan kaldıran ve şifre çözmeye karşı dirençli olan algoritmalarla gerçeklenir. Bu yöntemler elektronik sistemlerde (bilgisayar, telekomünikasyon vb.) kulanılır ve ikili düzende (binary) saklanan veya taşınan bilgi üzerinde uygulanırlar. Bu nedenle anahtar olarak bit dizileri kullanırlar.

Modern Şifreleme Algoritmaları genel anlamda ikiye ayrılmıştır[6,13].

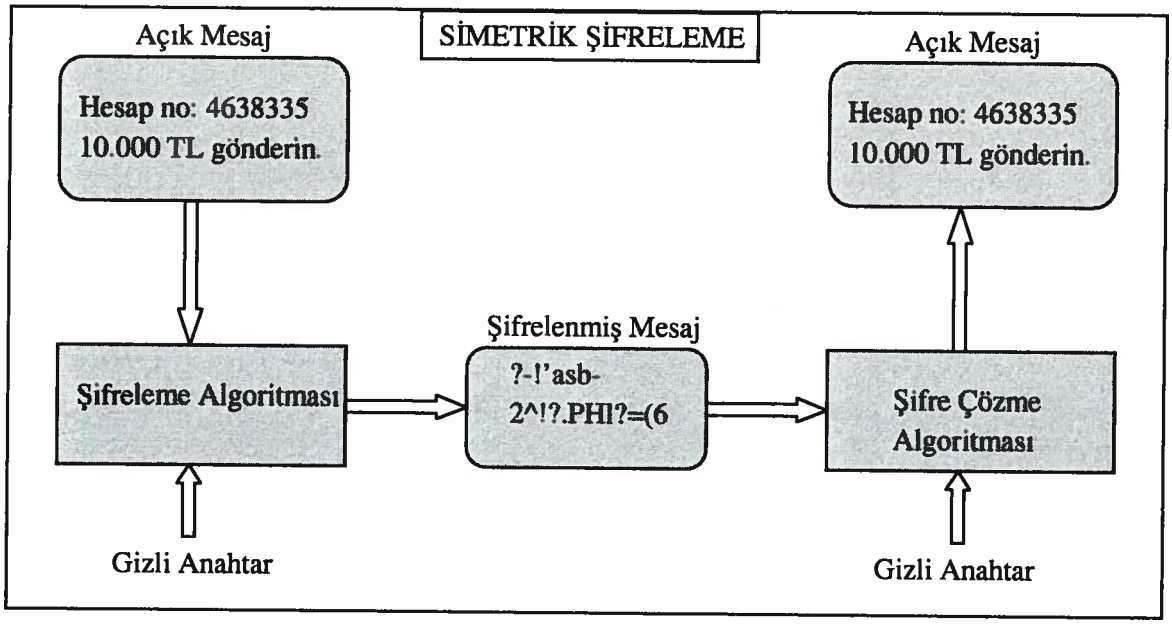
- Simetrik Şifreleme Algoritmaları
- Asimetrik Şifreleme Algoritmaları

2.1. Simetrik Şifreleme Algoritmaları

Simetrik şifrelemede, şifreleme ve şifre çözme işlemi için aynı anahtar kullanılır. Bu anahtar gizli anahtar (secret key) olarak adlandırılır. Gizli anahtar sadece gönderici ve alıcı tarafından bilinir. Bu tür algoritmalar da veri alışverişinde bulunan taraflar;

- Aynı şifreleme algoritmalarını kullanırlar.
- Anahtarlar aynıdır.

Simetrik şifreleme yöntemlerinin genel yapısı Şekil 2.1' de gösterilmiştir.



Şekil 2.1 Simetrik şifreleme yöntemi

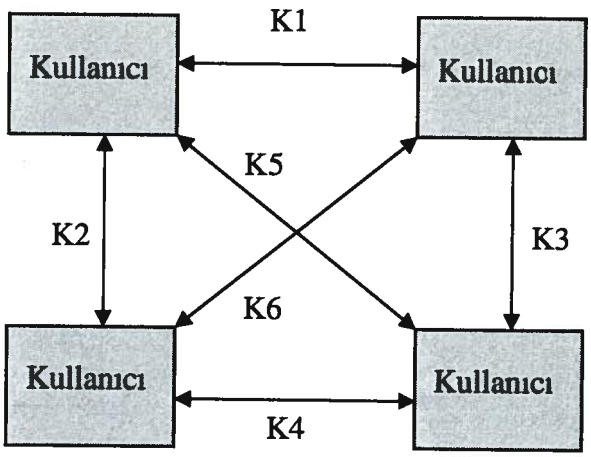
Simetrik şifrelemede mesaj açık kanaldan iletilirken, anahtar güvenli kanaldan iletilmelidir. Simetrik şifrelemede anahtar dağıtımı en büyük sorundur[14].

Simetrik şifrelemenin avantajları; Algoritmalar hızlı bir şekilde çalışır, algoritmaların donanımla gerçeklenmesi kolaydır. **'Gizlilik'** güvenlik hizmetini tam olarak yerine getirir. Simetrik şifrelemenin dezavantajları ise şunlardır; Gizli anahtarın emniyetli bir şekilde dağıtımı zordur, **'Veri Bütünlüğü'** ve **'Kimlik Doğrulama'** güvenlik hizmetlerini gerçekleştiremez, gizli anahtar sayısı çok fazladır.

Şekil 2.2' de görüldüğü gibi simetrik şifrelemede 4 kullanıcı için 6 anahtar, yani n adet uç arasında yapılacak şifreli iletişim için n * (n-1) / 2 adet anahtarın, iletişim öncesinde güvenli olarak karşı tarafa iletilmesine gereksinim vardır[1]. Kullanıcı sayısı arttıkça anahtar sayısı büyük boyutlara ulaşabilmektedir. Bu durum simetrik şifrelemenin en büyük dezavantajıdır.

Yukarıda bahsedilen dezavantajlar, özellikle anahtar sayısının çokluğu ve dağıtım zorluğu, internet altyapısı içinde geniş bir kullanıcı adedi düşünüldüğünde, simetrik şifreleme için anahtar paylaşımının ne kadar büyük bir sorun olduğunu göstermektedir.

Simetrik şifreleme sadece bilginin gizliliğini sağlamaktadır. Bu durum günümüzde çok önem kazanan sayısal imza, kimlik denetimi ve inkar edememe gibi özelliklerin gerekli olduğu alanlarda simetrik şifrelemenin en önemli eksiklerinden olmaktadır.



Şekil 2.2 Simetrik şifrelemede anahtar dağıtımı

Simetrik şifreleme algoritmalarında şifreleme ve şifre çözme işlemlerinde aynı gizli anahtar kullanılır ve blok şifreleme ve dizi (stream) şifreleme olmak üzere iki temel simetrik algoritma tipi vardır. Blok şifreleme, açık metni veya şifreli metni bloklara bölerek şifreleme/şifre çözme işlemini yapar. Dizi şifrelemede ise bir bit veya bir byte üzerinde şifreleme ve şifre çözme işlemleri yapılır[15].

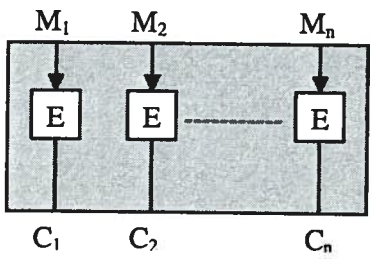
Blok şifreleme algoritmalarına en iyi örnekler olarak, DES[13,16] ve AES[17,18] verilebilir. Blok şifreleme algoritmalarının gücü incelenecek olduğunda, algoritmada kullanılan S kutuları, döngü sayısı, anahtarların XOR işlemine sokulması, blok uzunluğu, anahtarın uzunluğu ve anahtarın özelliğinin, algoritma güvenliğinde büyük etkisi olduğu görülür. Ayrıca kullanılacak olan anahtarın rastlantısal olması da gerekir. Diğer yandan blok uzunluğu algoritmaya yapılan saldırılara karşı dayanıklılıkta günümüz algoritmalarının gücünün ölçülmesinde bir ölçü olmuştur.

2.1.1. Dizi Şifreleme

Bu şifreleme yönteminde algoritmanın girdisi açık metin ve anahtardır. Algoritma anahtardan açık metin boyutunda, rastgele bir diziye çok benzeyen kayan anahtar dizisi üretir. Daha sonra kayan anahtar dizisinin elemanları ile açık metin veya kapalı metin dizisinin elemanları ikili tabanda toplanarak şifreleme veya şifre çözme işlemi yapılır. Dizi şifreleme mesajı bit bit işler (dizi olarak). En meşhur olanı **Vernam** şifreleyicisidir. 1917' de AT&T' de çalışan Vernam tarafından geliştirilmiştir. Basit olarak açık mesaj bitlerini rastgele anahtar bitlerine ekler. Açık mesaj biti kadar anahtar biti gerekir. Anahtar tamamen rastgele seçilirse koşulsuz güvenlik sağlanır. Açık mesaj uzunluğu kadar anahtar uzunluğu gerektirdiği için pratikte uygulanması çok zordur. Örneğin; 100 Megabyte açık mesaj için 100 Megabyte anahtar gerekir. Böyle büyük bir anahtarın dağıtımı güç olduğu için anahtar dizisi daha küçük bir anahtardan üretilebilir. Bunun için rastgele fonksiyonlar kullanılır. Her ne kadar bu çok uygun görünse de pratikte iyi bir rastgele fonksiyon bulmak zordur. Bu hala birçok araştırmacının üzerinde çalıştığı bir konudur.

2.1.2. Blok Şifreleme

Blok şifrelemenin en basit tanımı, açık metni bitişik bloklara bölme, her bloğu şifreleyerek şifreli metin bloklarına dönüştürme, bu şifreli blokları şifreli metin çıkışı olarak gruplamaktır. M1, M2,....., Mn açık metnin blokları, yani her biri K bitten oluşan ardışık parçaları, C1, C2,....., Cn bu bloklara karşılık gelen şifrelenmiş metinler ve E şifreleme işlemi olmak üzere, blok şifre sistemlerini Şekil 2.3' teki gibi gösterebiliriz[19].



Şekil 2.3 Blok şifre sistemlerinde şifreleme

Birçok simetrik blok şifreleme sisteminde blok uzunluğu 64 bittir. İşlemcilerin hızı arttıkça blok uzunluğuda artabilmektedir. Son yıllarda tasarlanan sistemlerde 128, 192, 256 bit blok uzunlukları kullanılmaya başlanmıştır. Bir blok şifre sisteminde, şifreli metin bloklarından birinin kaybolması, diğer blokların şifre çözme işleminde bir yanlışlığa neden olmaz. Bu blok şifre sistemlerinin en büyük avantajıdır.

Blok şifreler, karıştırma (confusion) ve yayılma (diffusion) tekniklerine dayanır[20]. **Karıştırma,** şifreli metin ve açık metin arasındaki ilişkiyi gizlemeyi amaçlarken, **yayılma,** açık metindeki izlerin şifreli metinde anlaşılmamasını sağlamak için kullanılır. Karıştırma ve yayılma, sırasıyla yerdeğiştirme ve lineer transformasyon işlemleri ile gerçekleştirilir.

Blok şifreler içerisinde en fazla kullanılan algoritma olan DES algoritmasıda büyük ölçüde karıştırma ve yayılma tekniklerine dayanmaktadır. **Karıştırma** özelliği yerdeğiştirme işlemleri ile yerine getirilir. Yerdeğiştirmede orijinal metindeki karakterler yerlerini şifreli metinde farklı karakterlere bırakırlar. Ancak bir karakter her zaman aynı karakter ile ifade edilmez, şifrelemenin farklı aşamalarında farklı karakterlerle yerdeğiştirir. **Yayılma** özelliği ise permutasyon işlemleri ile yerine getirilir. Farklı bölümlerin sırası yeniden düzenlenerek veri içerisindeki yerleri permute edilir. Bu permutasyonlar, yerdeğiştirmeye benzer şekilde anahtara ve açık metne bağlıdırlar.

2.1.3. Blok Şifreleme Algoritmalarının Önemli Özellikleri

Anahtar: Blok şifreleme algoritmalarında anahtarın uzunluğu en temel saldırı olan geniş anahtar arama saldırısına karşı güçlü olmalıdır. Örneğin DES algoritması 56-bit anahtar kullanırken AES algoritması DES' in bu zaafını örter niteliktedir ve 128, 192, 256 bit anahtar seçenekleri mevcuttur. Ayrıca anahtarın rastlantısal olmasıda gerekmektedir.

Döngü Sayısı: Blok şifreleme algoritmalarında döngü sayısı iyi seçilmek zorundadır. Çünkü lineer transformasyon ve yerdeğiştirmelerin bu seçilen değerle algoritmaya yeterli gücü vermesi gerekmektedir. Ayrıca yapılan saldırıların başarısız olması için en önemli şartlardan biridir.

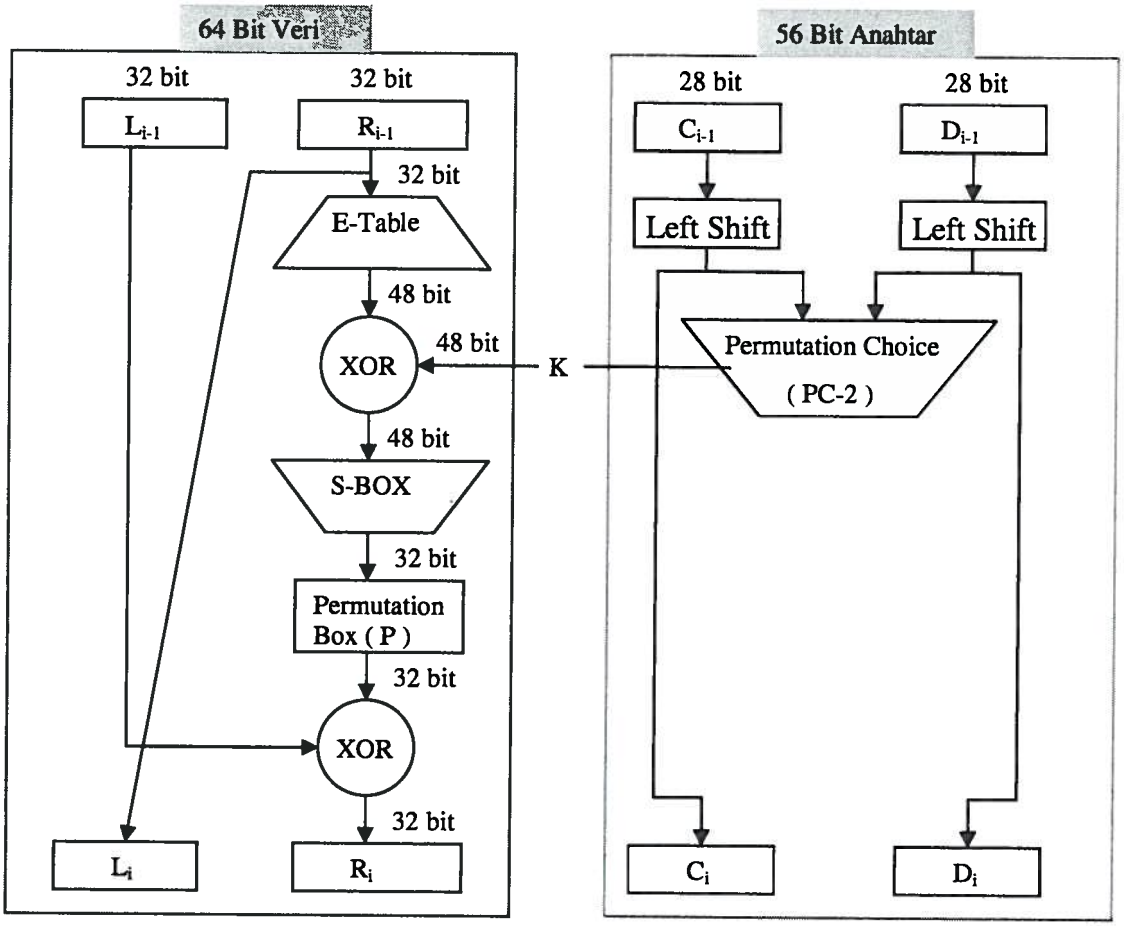
S kutuları: S kutuları bir blok şifreleme algoritmasının en önemli ana elemanıdır. Çünkü algoritmadaki tek non-lineer yapıdır ve dolayısıyla algoritmaya gücünü vermektedir. S kutuları için üç önemli nokta vardır. Bunların belirlenmesinde lineer kriptoanaliz, diferansiyel kriptoanaliz, Davies [21] saldırıları etkili olmuştur. Bunlar;

- **SAC** (Strict Avalanche Criteria-Çığ etkisi); 1 bit giriş değişimi sonucunda her çıkış bitinin değişme olasılığı ½ olur.
- **S kutularının genişliği** ; Kriptoanaliz saldırıları düşünüldüğünde büyük bir kutu küçüğüne oranla daha iyi olacaktır. Ayrıca diferansiyel saldırılardan korunmak için büyük sayıda çıkış bitleri ve lineer saldırılardan korunmak için büyük sayıda giriş bitleri gereklidir.
- **S kutusu gereksinimleri;** Çıkışların dağılımları Davies saldırısına karşı kontrol edilmeli, çıkışlar girişe göre lineer olmamalı, S kutusunun her sırasındaki değerler tek olmalıdır[22-25].

2.1.4. DES Algoritması

DES algoritması, 1970 yılında IBM tarafından geliştirilen Lucifer algoritmasının biraz daha geliştirilmiş halidir. 1974' te IBM' in NSA (National security Agency-Ulusal Güvenlik Ajansı) ile birlikte geliştirdiği algoritma olan DES' in yayınlanmasından itibaren DES algoritması üzerinde geniş ölçüde çalışmalar yapılmıştır. En yaygın blok şifreleme standardıdır. Şekil 2.4' te görüldüğü gibi blok boyutu 64 bit, anahtar boyutu 56 bittir. DES algoritmasını özel kılan F fonksiyonunun yapısı ve alt anahtarların ana anahtardan nasıl oluşturulduğudur.

İlk tasarlandığında donanım uygulamalarında kullanılması amaçlanmıştır. İletişim amaçlı kullanımda hem gönderen, hem de alıcı, şifreleme ve şifre çözmede kullanılan aynı gizli anahtar üzerinde anlaşmış olmalıdır. DES aynı zamanda sabit diskte veri saklamak gibi tek kullanıcılı şifreleme amaçlı da kullanılabilir. DES' in en büyük zayıflığı 56 bitlik anahtarıdır. Geliştirildiği zamanlarda çok iyi bir şifreleme algoritması olmasına rağmen modern bilgisayarlar tarafından yapılan anahtar saldırılarına karşı yetersiz kalmıştır. DES' in diger bir zayıflığıda yavaş olmasıdır[8].



Şekil 2.4 DES' in iç yapısı

2.1.4.1. DES Anahtar Üretimi

Ana anahtar 64 bittir. Anahtar üretiminde ana anahtar üzerinde bazı işlemler yapılarak DES' in 16 adımında kullanılacak 16 alt anahtar elde edilir. Öncelikle PC^1 permütasyonu kullanılarak anahtarın 56 biti seçilir. Sonuç 28 bitlik iki yarım parçaya ayrılır[16]. Her bir adımda:

- Her bir yarım parça, kaydırma tablosuna göre 1 veya 2 bit sola kaydırılır.
- İki yarım parça birleştirilir.
- 56 bitlik yeni sonuç permüte edilerek 48 bit seçilir.

K=0001001100110100010101110111100110011011101111001101111111110001 olsun. Anahtarımız başlangıçta 64 bittir. İlk olarak Şekil 2.5' teki PC^1 permüte tablosu kullanılarak 56 biti seçilir.

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

Şekil 2.5 PC^1 permutasyon tablosu

Anahtardan seçilen 56 bit: **'1111000 0110011 0010101 0101111 0101010 1011001 1001111 0001111'** dir. Alt anahtarlar oluşturulurken bu kısım esas alınır.

Ardından 56 bitlik anahtar iki parçaya bölünür.

C0 = 1111000 0110011 0010101 0101111

D0 = 0101010 1011001 1001111 0001111

Her bir yarım parça Şekil 2.6' daki sola kaydırma planına göre 1 veya 2 bit sola ötelenir.

Adım Numarası	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Sola Kaydırma Sayısı	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

Şekil 2.6 Anahtar sola kaydırma tablosu

İlk adım olduğu için anahtarın sol ve sağ yarımı 1 sola kaydırılır. Kaydırma işleminden sonra anahtar yarımları aşağıdaki gibi oluşur.

C1=11100001100110010101011111

D1=10101010110011001111100011110

İki parça birleştirilerek **K1** anahtarı elde edilir.

K1=11100001100110010101011111 10101010110011001111100011110

Sonra 56 bitlik K1 anahtarı Şekil 2.7' deki PC^2 tablosuna sokularak 48 bitlik bir seçim işlemi yapılır.

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

Şekil 2.7 PC^2 tablosu

Seçilen K1 anahtarı : **'000110110000001011101111111111000111000001110010'** dır.

Bu değer K1 alt anahtarının son halidir.

K2 alt anahtarının bulunması;

2. adım olduğu için her bir yarım (C0 ve R0) Şekil 2.6' daki sola kaydırma planına göre 2 defa sola kaydırılarak yeni anahtarın iki yarımı oluşturulur.

C2=11000011001100101010101111111

D2=01010101100110011111000111101

İki parça birleştirilerek K2 anahtarı elde edilir.

K2=11000011001100101010101111111010101011001100111110001111101

Sonra 56 bitlik K2 anahtarı Şekil 2.7' deki PC^2 tablosuna sokularak 48 bitlik bir seçim işlemi yapılır.

K2 alt anahtarının son hali **011110011010111011011001110110111100100111100101** olarak elde edilmiştir. Kalan 14 anahtarda aynı şekilde elde edilir. Ve her turda ilgili alt anahtar kullanılır (1.tur K1, 2.tur K2, .. , 16.tur K16).

2.1.4.2. DES Şifreleme İşlemi

64 bitlik verimiz; **'0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 1110 1111'** olsun. Bu veri Şekil 2.8' deki 64 bitlik IP tablosuna göre permute edilir.

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

Şekil 2.8 IP tablosu

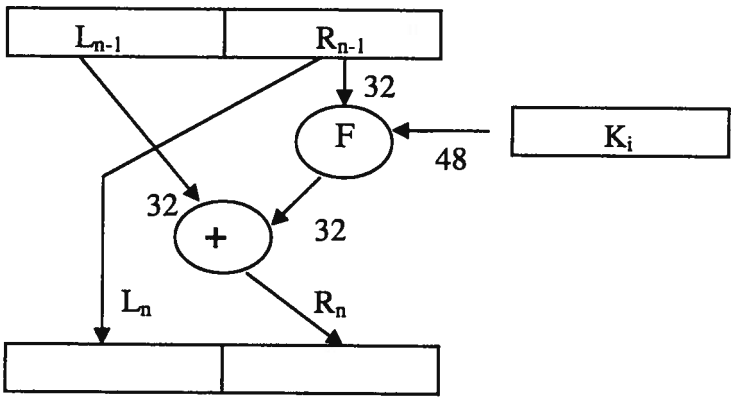
Verinin permute edilmiş hali **1100 1100 0000 0000 1100 1100 1111 1111 1111 0000 1010 1010 1111 0000 1010 1010** olarak elde edilir.

Elde edilen veri sol ve sağ olmak üzere iki eşit parçaya bölünür.

- **L0**=11001100000000001100110011111111
- **R0**=11110000101010101111000010101010

1. Adım için;

K1 = 000110 110000 001011 101111 111111 000111 000001 110010 biliniyor.



Şekil 2.9 DES' in herhangi bir adımı

Ln=Rn-1 (Şekil 2.9' dan)

L1=R0= 1111 0000 1010 1010 1111 0000 1010 1010 olarak bulunur.

Rn=Ln-1 + F(Rn-1,Kn) (Şekil 2.9' dan)

R1 = L0 + F(R0,K1) olarak hesaplanacaktır.

$$F(R,K) = P\left(S\left(E(R) \oplus K\right)\right)$$

Şekil 2.10 DES' in F fonksiyonu

Şekil 2.10' da F fonksiyonunun içeriği görülmektedir. Uygulanacak olursa;

- Sağ yarım (R0) önce 48 bite genişletilir.[E(R0)]
- Ardından o adımın alt anahtarıyla XOR lanır. [E(R0) + K]
- 48 bitlik sonuç S-Box lara sokularak 32 bite indirgenir.[S(E(R0)+K)]
- Elde edilen sonuç permüte edilir. [P(S(E(R0)+K))]

F adım fonksiyonu elde edilmiş olur. Bu veri ile **L0** exorlanır ve **R1** değeri elde edilmiş olur.

Sağ yarımın genişleme fonksiyonuna sokularak 48 bite çıkarılması;

R0= 1111 0000 1010 1010 1111 0000 1010 1010 verisi Şekil 2.11' deki genişleme tablosuna (E-Table) sokularak 48 bite çıkarılır.

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

Şekil 2.11 E-Table (Genişleme tablosu)

Genişlemeden sonra ;

E(R0) : 011110100001010101010101011110100001010101010101 olarak elde edilir.

K1+E(R0) : Alt anahtar ile genişleyen veri XOR işlemine tabi tutulur.

K1: 000110 110000 001011 101111 111111 000111 000001 110010

E(R0): 011110 100001 010101 010101 011110 100001 010101 010101

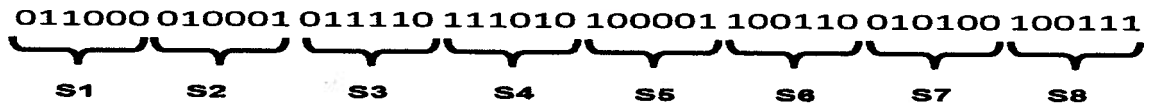
+----------

K1+E(R0): 011000 010001 011110 111010 100001 100110 010100 100111 olarak bulunur.

Bu veri gurubu 6 bitlik 8 gruba ayrılarak Şekil 2.12' deki S-Box' tan geçirilir.

S1	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13
S2	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9
S3	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12
S4	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14
S5	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3
S6	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
	9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13
S7	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12
S8	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

Şekil 2.12 S-Box (S1,S2….,S8)



Şekil 2.13 32 Bit verinin S-Box'a girmeden önce gruplanışı

Şekil 2.13' deki 6 bitlik S1 grubunun ilk ve son biti S-Box' ta bakacağımız satır numarasını, ortada kalan dört bit S-Box' ta bakacağımız sütun numarasını gösterir. Buna göre S1 grubunun sonucu S-Box' ta (S1' de) 0. satır ile 12. sütunun kesiştiği noktadaki 5 (0101) verisidir. Benzer Şekilde S2, S3..., S8' in S-Boxlardan geçmesiyle aşağıdaki değerler elde edilir.

0101 1100 1000 0010 1011 0101 1001 0111 (32 bit)

ardından S-Box çıkışındaki veri Şekil 2.14' teki P tablosuna sokularak veriler bir kez daha karıştırılır.

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

Şekil 2.14 P tablosu

P Tablosundan önce : 0101 1100 1000 0010 1011 0101 1001 0111

P Tablosundan sonra: 0010 0011 0100 1010 1010 1001 1011 1011

R1 = L0 + F(R0 , K1)' den

L0 : 1100 1100 0000 0000 1100 1100 1111 1111

F(R0 , K1) : 0010 0011 0100 1010 1010 1001 1011 1011

+----------

R1 : 1110 1111 0100 1010 0110 0101 0100 0100 elde edilir.

L0 ve R0' ın birleştirilmesiyle ;

11001100 00000000 11001100 11111111 11101111 01001010 01100101 01000100 verisi elde edilir. Son olarak bu 64 bitlik veri Şekil 2.15' deki IP^{-1} tablosu kullanılarak permute edilir.

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

Şekil 2.15 IP^{-1} tablosu

Şifrelemenin 1.adımı sonucunda **'10001001 10100001 11001111 11100101 00000001 10001001 11101111 11000101'** verisi elde edilir. Bu veri 2. adıma girdi olarak kabul edilir, sıraladığımız işlemler kalan 15 adım boyunca tekrarlanarak 64 bitlik verinin şifrelenmiş hali elde edilir. Son adımda şifrelenmiş verinin sol ve sağ yarımı yer değiştirilir.

1. adımda yapılan işlemler özetlenirse;

0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 1110 1111 şeklindeki 64 bitlik ilk veri;

- 64 bitlik IP tablosuna göre karıştırılıp 1100 1100 0000 0000 1100 1100 1111 1111 1111 0000 1010 1010 1111 0000 1010 1010 verisi elde edilir.

- Oluşan bu veri **L0** = 1100 1100 0000 0000 1100 1100 1111 1111,
  R0 =1111 0000 1010 1010 1111 0000 1010 1010 şeklinde iki yarıma ayrılır.

- **Ln = Rn-1**
  Rn = Ln-1 + F(Rn-1,Kn) formülünden **L1 = R0** olarak belirlenir.

  R1 = L0 + F(R0,K1) şeklinde hesaplanır. Burada ;

  F(R0,K1) = P (S (E(R0) + K1))' den elde edilir.

- E(R0) değerini elde etmek için R0 verisi E-Tablosundan geçirilerek 48 bite çıkarılır. Bu veri 48 bitlik adım anahtarı (K1) ile XOR lanır.(E(R0)+K1))

- Oluşan 48 bitlik veri S-Box tablosuna girdirilerek 32 bite indirgenir.

  (S (E(R0)+K1))

- Elde edilen 32 bitlik veri P tablosuna göre permute edilerek, R1 değeri elde edilir.

- Son olarak 64 bitlik veri (L1-R1) IP^{-1} tablosu kullanılarak permute edilir.

- Oluşan veri 2. adımın giriş verisi olarak ele alınıp kalan 15 adım aynı şekilde tamamlanarak 64 bitlik blok şifrelenmiş olur. Yalnız son adımda şifrelenmiş verinin sol ve sağ yarımı yer değiştirilir, bu işlem şifre çözme işleminin doğru çalışması için gereklidir[16].

2.1.4.3. DES Şifre Çözme

DES' in şifre çözme işlemi, şifreleme işlemi ile aynıdır. Kural şöyle ifade edilebilir: Şifreli metin, DES algoritmasına girdi olarak kullanılır, ancak K_i anahtarları ters sırada kullanılır. Yani ilk turda K16 anahtarı, ikinci turda K15 anahtarı kullanılır. Son tura gelindiğinde ise kullanılacak olan anahtar K1 olur. 16 adım sonucunda açık metin elde edilmiş olur[26].

2.1.4.4. DES' in Güvenliği

DES şifreleme algoritmasının en önemli parçası tek doğrusal olmayan parçası olan S kutularıdır. Doğrusal yapılara sahip şifreleme algoritmaları kolayca kırılabilir. DES şifreleme algoritması ilk olarak ortaya atıldığında bünyesinde bulundurduğu S kutularının gizli bir arka yapıya sahip olabileceği öne sürüldü. Çünkü o zamanlarda S kutularının özellikleri açıklanmamıştı. Ancak Biham ve Shamir 1990' lı yıllarda diferansiyel kriptoanalizi keşfettiklerinde S kutularının bu saldırıya benzer saldırı tipine karşı kullanıldığı anlaşıldı[7]. Aslında DES algoritması geliştirildiğinde IBM araştırmacıları tarafından diferansiyel kriptoanaliz biliniyordu ve Biham ile Shamir tarafından bağımsız olarak keşfedilene kadar 20 yıl bu bilgi gizlenmiştir.

DES algoritmasının en büyük zaafı 2^{56} olan anahtar uzayı genişliğidir. Bu gerçekten de güçlü bir şifreleme algoritması için oldukça küçük bir anahtar uzayı miktarıdır. 1970' lerde dahi bilinen açık metin saldırısının bu şifreleme algoritmasına karşı uygulanabileceği ve geniş anahtar arama saldırısı ile anahtarın elde edilebileceği fikri ortaya atıldı. Daha sonra DES algoritmasını kırabilecek çeşitli özel anahtar arama cihazları geliştirildi. Bunlardan 1998' de RSA labaratuarı tarafından geliştirilmiş olan 'DES Challange II-2' 56 saat içerisinde bir DES anahtarını başarı ile bulmuştur. Geniş anahtar arama saldırısı dışında DES algoritmasına yönelik olarak iki saldırı ön plana çıkmıştır. Bunlar sırasıyla doğrusal kriptoanaliz ve diferansiyel kriptoanalizdir. Doğrusal kriptoanaliz, Matsui tarafından geliştirilmiştir[27,28]. Doğrusal kriptoanaliz DES algoritması düşünüldüğünde daha başarılı olmuş bir saldırıdır diyebiliriz ve 2^{43} açık metin şifreli metin çifti gerektirmektedir. Bu saldırı için bu çiftleri üretmek 40 gün, anahtarı elde etmek ise 10 gün almıştır. Bu saldırı yinede DES üzerinde çok büyük bir etki yapmamaktadır çünkü saldırıyı gerçekleştirmek için gerekli açık metin şifreli metin çifti sayısı çok fazladır. Saldırganın aynı anahtarla üretilmiş açık metin ve şifreli metinleri toplaması pek de mümkün görünmemektedir[29].

2.2. Asimetrik Şifreleme Algoritmaları

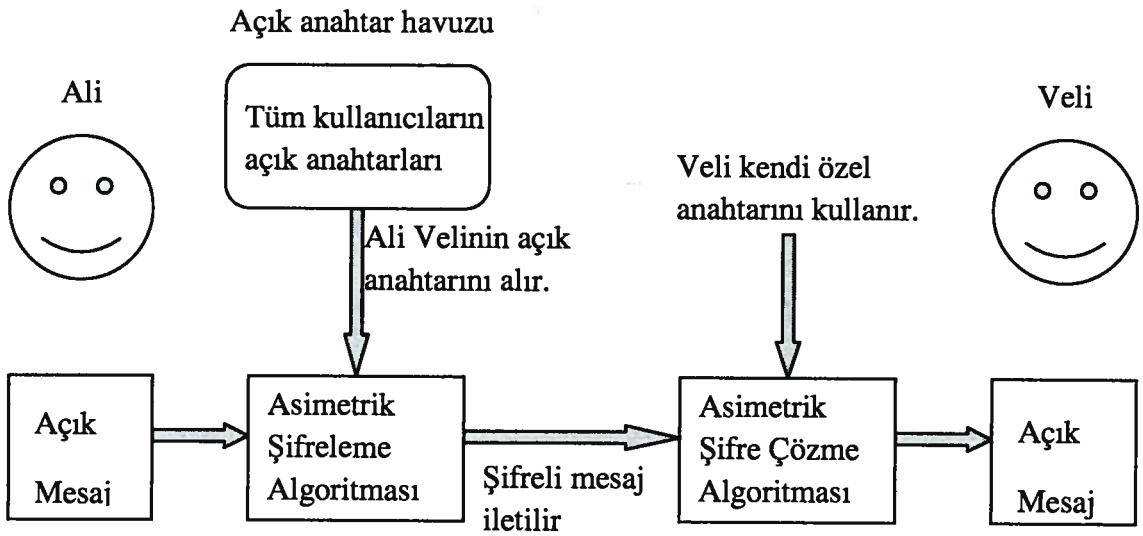
Asimetrik şifrelemeyi ilk ortaya koyanlar Stanford Üniversitesinden Whitfield Diffie ve Martin Helman' dır. Asimetrik şifrelemenin genel amacı simetrik şifrelemenin en büyük

iki problemine çözüm sağlamaktır. Bu problemlerden ilki anahtarların dağıtımıdır. İkincisi ise sayısal imza konusudur. Sayısal imza ile mesajı kimin gönderdiği kesinlikle bilinmiş olacaktır[30].

Asimetrik şifrelemede şifreleme ve şifre çözme işlemleri için farklı anahtarlar kullanılır. Bu anahtarlar **açık anahtar** (public key) ve **özel anahtar** (private key) olarak adlandırılır. Açık anahtarı mesajlaşmanın tüm tarafları görebilir. Mesaj gönderecek kişi mesajı karşı tarafın açık anahtarı ile şifreleyerek gönderir. Özel anahtar ise kişiye özeldir, paylaşılmaz. Mesajı alan taraf şifreli mesajı kendi özel anahtarını kullanarak çözer[14].

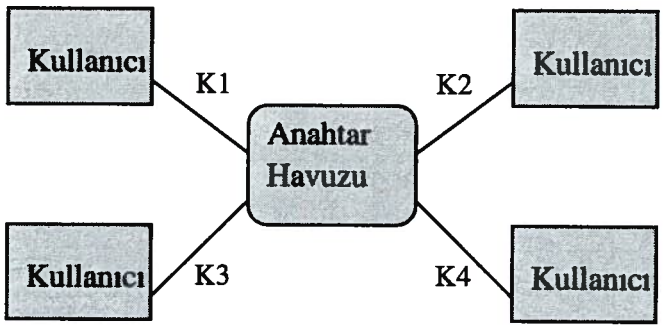
Asimetrik algoritmalar, simetrik algoritmalara göre daha güvenli ve kırılması zor algoritmalardır. Ancak buna karşılık permormans değerleri simetrik algoritmalara göre oldukça düşüktür. Asimetrik şifrelemenin en önemli özelliği; açık olan anahtarın ve beraberindeki şifreli metnin, herkese açık ve dolayısıyla güvenli olmayan bir kanaldan iletilebilmesidir.

Asimetrik algoritmaların genel yapısı aşağıdaki Şekil 2.16' da gösterilmiştir.



Şekil 2.16 Asimetrik anahtarlı sistemlerin genel yapısı

Asimetrik şifreleme yöntemini kullanarak veri alış verişinde bulunan kullanıcılar: Aynı şifreleme algoritmasını kullanırlar, birbirine uyumlu gerçeklemeler kullanırlar, ihtiyaç duydukları anahtarlara ulaşabilirler. Şekil 2.17' de görüldüğü gibi Asimetrik şifrelemede anahtar sayısı **n** kullanıcı için **n** adettir. Bu simetrik şifrelemeye göre çok önemli bir avantajdır.



Şekil 2.17 Asimetrik şifrelemede anahtar dağıtımı

Her kullanıcının bir anahtar çifti vardır. Dolayısı ile 1000 kullanıcı için sadece 1000 anahtara gereksinim bulunmaktadır. Açık anahtarlar ilan edilir. Anahtar paylaşım problemi yoktur. Asimetrik şifreleme yöntemlerinin örnekleri; RSA, Elgamal, ECC, Diffie-Hellman ve DSA dır. Asimetrik şifrelemenin avantaj ve dezavantajları aşağıda verilmiştir.

Asimetrik şifrelemenin avantajları şunlardır; Anahtar sayısı azdır, her kullanıcının sadece kendi özel anahtarını saklaması yeterlidir, şifre çözmeye karşı dirençlidir, **"Bütünlük, Gizlilik, Kimlik Denetimi, İnkar Edememe"** gibi özellikleri sağlar.

Asimetrik şifrelemenin en büyük dezavantajı ise algoritmasının simetrik şifreleme algoritmasına göre çok yavaş olmasıdır. Buda şifrelemede kullanılan anahtarın uzunluğundan ve algoritma adımlarının yavaşlığından kaynaklanmaktadır.

2.2.1. RSA Algoritması

En fazla kullanılan asimetrik algoritma örneğidir. 1977 yılında Ron Rivest, Adi Shamir ve Leonard Adleman tarafından bulunmuş ve 1978 yılında yayınlanmıştır. İsmini tasarımcılarının isimlerinden almıştır[31].

RSA Algoritmasının özelliklerini şöyle sıralayabiliriz[32-34];

- Çarpanlara ayırma problemi üzerine kurulmuştur.
- Tam sayı olan N' i oluşturan ve her ikisi asal sayı olan P ve Q değerleri bulunur, öyleki N=P.Q dur.
- Yeterince büyük bir N sayısı için kırılması çok zordur.
- Kök bulma problemindende faydalanır.
- Çok güvenli olmasına rağmen çok hızlı değildir.

Anahtarın oluşturulması şu şekildedir:

- P ve Q gibi çok büyük iki tane farklı ve rasgele asal sayı seçilir.
- Bu iki asal sayının çarpımı olan **N**=P.Q ve **Q(N)**=(P-1) . (Q-1) değerleri hesaplanır.
- 1' den büyük Q(N)' den küçük ve Q(N) ile aralarında asal bir E tamsayısı seçilir.
- D.E = 1 mod (Q(N)) olacak şekilde bir D tamsayısı seçilir.
- E ve N tamsayıları genel anahtarlar, D ve N tamsayıları ise özel anahtarlardır.

Özel ve genel anahtarlar elde edildikten sonra iletilmek istenen veri genel anahtar ile şifrelenir. C verinin şifreli hali, M şifrelenen veri olmak üzere; İşlem şu şekilde yapılır:

Şifreleme işlemi: $\mathbf{C = M^E \bmod N}$

Şifre çözme işlemi: $\mathbf{M = C^D \bmod N}$

Formülleri açıklayacak olursak: Şifrelenecek verinin sayısal karşılığının E' ninci kuvveti alınır ve bunun mod N' deki karşılığı şifrelenmiş veriyi oluşturur. Genel anahtar ile şifrelenmiş bir metin sadece özel anahtar ile açılabilir. Bundan dolayı yine aynı yöntemle,

şifrelenmiş verinin sayısal karşılığının D' ninci kuvvetinin mod N' deki karşılığının bulunması suretiyle şifrelenmiş veri çözülmüş olur.

RSA algoritmasına örnek verilecek olursa;

- İki asal sayı seçilir P=61 ve Q=53 olsun.
- N değeri N=P.Q olarak hesaplanır.
- N=61.53=3233 bulunur.
- Totient fonksiyonu Q(N)=(P-1) (Q-1) şeklinde hesaplanır.
- Q(N) = (61-1) . (53-1) = 60.52 = 3120 olarak bulunur.
- Totient fonksiyon sonucu olan Q(N) ile aralarında asal olan ve 1' den büyük bir sayı seçilir(E).
- E > 1 ve Q(N) ile aralarında asal olarak E = 17 olarak seçilsin (3120 ile aralarında asal).
- D.E=1 mod (N) olacak şekilde D sayısı bulunur. D=2753 (çünkü 17*2753= 46801= 1+15*3120) Bu sayının hesaplanması sırasında uzatılmış öklit yöntemi kullanılmıştır.
- Mesaj olarak 123 gönderilecekse;
- Şifreli metin 123^{17} mod 3233 = 855 olarak bulunur.
- Mesajı çözecek taraf için tersi işlem uygulanır;
- Açık mesaj 855^{2753} mod 3233 = 123 olarak elde edilir[35].

2.3. Simetrik Şifreleme ile Asimetrik Şifrelemenin Karşılaştırılması

Simetrik şifreleme; Şifrelemenin temel amaçlarından sadece **'Gizlilik'** güvenlik hizmetini sağlar, buna karşılık algoritması çok hızlıdır. Asimetrik şifreleme; Şifrelemenin temel amaçlarından **'Gizlilik, Bütünlük, Kimlik Doğrulama, İnkar Edememezlik'**

güvenlik hizmetlerini sağlar, ancak algoritması simetrik algoritmalara göre oldukça yavaştır.

Simetrik şifrelemenin çalışması için; Şifreleme ve şifre çözme işlemi için aynı algoritma aynı anahtarlarla birlikte kullanılır. Gönderici ve alıcı aynı algoritmayı paylaşır. **Asimetrik** şifrelemenin çalışması için; Şifreleme ve şifre çözme işlemi için aynı algoritma ve anahtarlardan birisi kullanılır. Şifreleme için kullanılan anahtar, şifre çözme için kullanılamaz. Gönderici ve alıcı, ilişkili anahtarlardan birine sahiptirler.

Simetrik şifrelemede güvenlik için; Anahtarlar gizli tutulmalıdır, diğer bilgiler saklandığında mesajı çözmek imkansız olmalıdır, algoritma ve şifreli metin örneklerini bilmek anahtarı bulmak için yetersiz olmalıdır. **Asimetrik** şifrelemede güvenlik için; Anahtarlardan biri gizli tutulmalıdır, diğer bilgiler saklandığında mesajı çözmek imkansız olmalıdır, algoritmayı, şifreli metin örneklerini bilmek ve anahtarlardan birine sahip olmak, diğer anahtarı bulmak için yetersiz olmalıdır[30].

3. TEK YÖNLÜ (HASH) FONKSİYONLAR ve HMAC

3.1. Tek Yönlü Fonksiyonlar

Tek yönlü fonksiyonlar, tek yönde çözüm hesaplamak kolay olmasına rağmen aksi yönde çözüm hesaplamak çok zor olan fonksiyonlardır. Normal yönde çözümü bulmak saniyeler sürerken, ters yönde çözüm hesaplamak bilgisayar sistemi hızına bağlı olarak aylar sürebilmektedir[36].

Tek yönlü fonksiyonlar günümüzde modern şifrelemede çok sık kullanılmaktadır. Bu fonksiyonlar sayesinde verilerin özetleri çıkarılır ve bu özetler haberleşme tarafına veriyle beraber yollanır. Karşı taraf kendisine gelen veriyi aynı tek yönlü fonksiyondan geçirerek özet veri elde eder ve kendisine gelen veri özeti ile karşılaştırır. Özetlerin aynı olması verinin haberleşme esnasında değişmediği anlamına gelir.

Tek yönlü fonksiyonlar **h=H(M)** şeklinde ifade edilir. Burada 'M' sabit uzunlukta olmayan mesajı, h ise bu mesajdan oluşturulan sabit uzunluktaki hash değeridir.

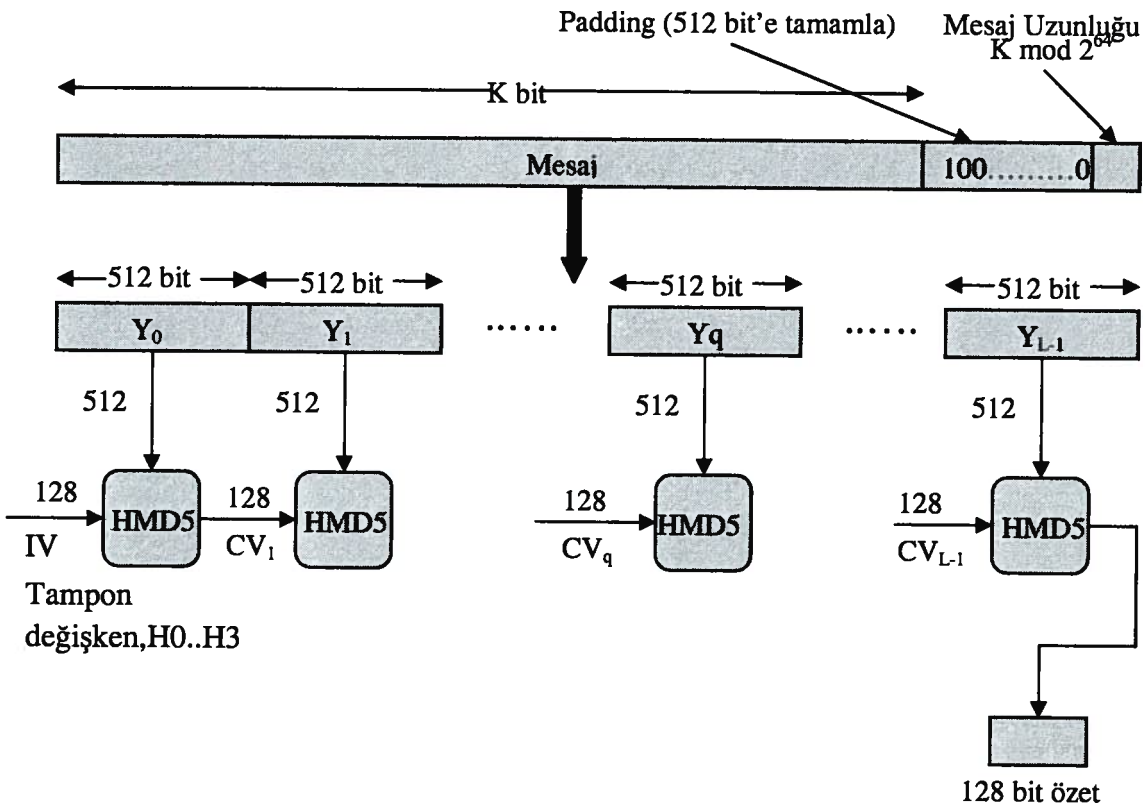
Tek yönlü fonksiyonların güvenli ve hızlı olabilmeleri için şu şartlara ihtiyacımız vardır[36];

- H fonksiyonu mesaj uzunluğuna bağlı olmamalıdır, yani her boyuttaki mesaja uygulanabilir olmalıdır.
- H fonksiyonu tüm mesaj boyutları için sabit uzunlukta bir çıkış vermelidir.
- Herhangi bir x girdisi için H(x) kolayca hesaplanabilir olmalı, ayrıca yazılımsal veya donanımsal olarak tasarlanabilmelidir.
- H(x) = y durumunda y biliniyorken x değerini bulmak çok zor olmalıdır. Bu duruma fonksiyonun tek yön özelliği denilmektedir. Tek yönlü fonksiyonların olmazsa olmaz özelliğidir.
- H(x) = H(y) olacak şekilde bir (x,y) çiftinin hesaplanması (collision-çarpışma) çok zor olmalıdır.

3.1.1. MD5 Algoritması (Message Digest Algorithm)

Ron Rivest tarafından 1991 yılında geliştirilen tek yönlü bir şifreleme algoritmasıdır. Giriş mesaj boyutuna bağlı olmadan her durumda 128 bitlik özet çıkış üretir. MD5' te girilen her mesajdan farklı sonuç üretilmesi mümkün değildir. Çünkü çıkış boyutu 128 bit olup, giriş mesaj boyutu ise teorik olarak sonsuzdur[37].

Algoritma yapısı Şekil 3.1' de görülmektedir.

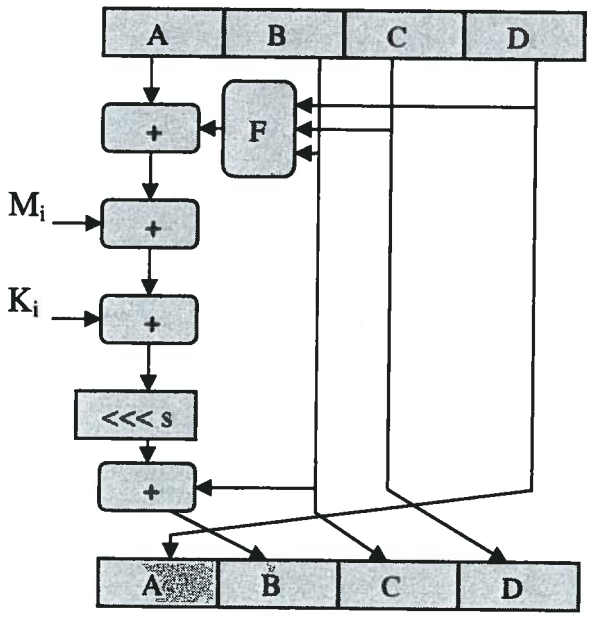


Şekil 3.1 MD5' in genel yapısı

MD5, veriyi 512 bitlik bloklara ayırır ve her bir blok üzerinde aynı işlem uygulanır. Üzerinde işlem yapılacak veri boyutunun 512 bitin katları olması gerekmektedir, fakat

gerçek verimiz bu uzunlukta olmayabilir. Bu sorunu çözmek için ekleme (padding) işlemi uygulanır[38]. Ekleme işleminde şu kural gözetilir: Verinin uzunluğu 512 bitin en yakın katından 64 bit eksik olacak şekilde, verinin sonuna bir adet **1** ve geri kalanlar için ise **0** eklenir. Son 64 bitlik kısım ise verinin uzunluğunu belirtmekte kullanılır.

Bir örnekle açıklanacak olursa; veri 200 bit uzunlukta olsun. Bunun 448 bite (512-64) tamamlanması gerekmektedir. Dolayısı ile 201. bit olarak **1** ve geri kalan 247 bit için **0** eklenir. Şu anda veri 448 bit oldu. Gerçek verinin uzunluğu 200 bit idi ve buda ikilik tabanda 64 bit ile ifade edilip 448 bitlik veriye eklenir. Böylece 512-bitlik yeni oluşturulan veri üzerinde MD5 algoritması uygulanabilir. Ekleme işleminden sonra, MD5 veriyi herbir blok için Şekil 3.2' de görüldüğü gibi işlemeye başlar.



Şekil 3.2 MD5' te 512 bitlik bloğun işlenişi

Döngünün başlangıcında H_0=67452301, H_1=EFCDAB89, H_2=98BADCFE, H_3= 10325476 olmak üzere, 32 bitlik dört tane değişken bulunur. t=0 dan t=63' e kadar toplam 64 adımlık döngüde bu değişkenler üzerinde bir dizi işlem adımları uygulanarak geçici A, B, C, D değişkenler değerleri güncellenir.

MD5' in paddinglenmiş 512 bitlik bir blok için işlem adımları açıklanacak olursa;

1. t=0 dan t=63' e kadar olan 64 adımlık döngü başlamadan önce H_0, H_1, H_2, H_3 değişkenleri öncelikle A, B, C, D değişkenlerine aktarılır.

2. Her bir adımda A=D (önceki adımın D değişkeni), C=B (önceki adımın B değişkeni, D=C (önceki adımın C değişkeni) olarak hemen güncellenir. B değişkeninin yeni değeri ise **B=B+leftrotate((A+F_t(B,C,D)+K[t]+W[g]), R[t])** eşitliğinden elde edilir.

Eşitlik açıklanacak olursa;

F_t(B,C,D) B, C, D değişkenlerini parametre olarak alır ve bu değişkenler üzerinde bazı mantıksal fonksiyonlar uygulayarak 32 bitlik geri dönüş değeri üretir.

0 <= t <=15 aralığında F_t(B,C,D)=(X **AND** Y) **OR** (^X **AND** Z) ,

16<= t <=31 aralığında F_t(B,C,D)=(X **AND** Z) **OR** (Y **AND** ^Z),

32<= t <=47 aralığında F_t(B,C,D)= X **XOR** Y **XOR** Z ,

48<= t <=63 aralığında F_t(B,C,D)= Y **XOR** (X **OR** ^Z) eşitlikleri kullanılır.

K[t] değeri K[t]=2^{32} **x abs(sin(t+1))** eşitliğinden elde edilir. Böylece 64 adımda kullanılacak 64 adet rastgele 32 bit değer sağlanmış olur.

W[g] değeri başlangıçta 32 bitlik 16 kelimeye ayrılan 512 bitlik bloğun 16 kelimesinden birini ifade eder. ***g*** değerleri;

0 <= t <=15 aralığında g=i,

16<= t <=31 aralığında g=(5*i+1) mod 16,

32<= t <=47 aralığında g=(3*i+5) mod 16,

48<= t <=63 aralığında g=(7*i) mod 16 eşitliklerinden elde edilir.

R[t] değeri **(A+F_t(B,C,D)+K[t]+W[g])** değerine kaç defa sola kaydırma işlemi uygulanacağını belirtir. R[t] değerleri;

0 <= t <=15 aralığında **r[0..15]= {7, 12, 17, 22, 7, 12, 17, 22, 7, 12, 17, 22, 7, 12, 17, 22}**
16<= t <=31 aralığında **r[16..31]={5, 9, 14, 20, 5, 9, 14, 20, 5, 9, 14, 20, 5, 9, 14, 20}**
32<= t <=47 aralığında **r[32..47]={4, 11, 16, 23, 4, 11, 16, 23, 4, 11, 16, 23, 4, 11, 16, 23}**
48<= t <=63 aralığında **r[48.63]={6, 10, 15, 21, 6, 10, 15, 21, 6, 10, 15, 21, 6, 10, 15, 21}**

olarak kullanılır. Tekrar eşitliğe dönülecek olursa ;

B=B+leftrotate((A+F_t(B,C,D)+K[t]+W[g]),R[t]) eşitliğinde **A+F_t(B,C,D)+K[t]+W[g]** değeri R[t] kadar sola ötelenerek önceki adımın B değeri ile toplanır ve yeni B değeri elde edilmiş olur.

Son olarak her bir 512 bitlik bloğun son adımından sonra $H_0=H_0+A$, $H_1=H_1+B$, $H_2=H_2+C$, $H_3=H_3+D$ şeklinde H_0, H_1, H_2, H_3 değerleri elde edilir ve bu değerler 128 bitlik MD5 özet değerini ifade etmektedir.

3.1.2. SHA-1 Algoritması

İlk olarak 1993 yılında Amerika Ulusal Güvenlik Kurumu NSA tarafından FIPS PUB-180 standardı olarak olarak yayınlanmıştır. SHA (Secure Hash Algorithm)' nın bu ilk sürümüne SHA-0 denilmektedir. Daha sonra algoritmanın sıkıştırma fonksiyonunda bir kısım değişiklikler yapılarak yine NSA tarafından SHA-1 ismiyle FIPS 180-1 standardı olarak yayınlanmıştır. Giriş verisini 512 bit bloklar olarak işleme alıp çıkışta 160 bit özet değer üretir. MD tek yönlü algoritmalarına benzerdir. Her bir blok 4 döngü ve her döngüde 20 adımlık işlemlere, yani toplamda 80 adımlık işleme tabi tutulur. SHA-1 algoritması MD5 algoritmasından daha güvenlidir. Ancak MD5 algoritmasına göre daha yavaştır[38].

SHA-1' in işlem adımları[39]:

1. Ekleme (Padding) İşlemi: Veri ikili sistemde ifade edilir ve bu verinin sonuna 512 bitin en yakın katından 64 bit eksik olacak şekilde, verinin sonuna bir adet 1 ve geri kalanlar için ise 0 eklenir. Son 64 bitlik kısım ise verinin uzunluğunu belirtmekte kullanılır. İkili sisteme çevrilmiş olan bu veri, her dört bitlik grup bir hexadecimal sayıya karşılık gelecek şekilde hexadecimal sayı sistemine çevrilir.

2. 32 bitlik H_0, H_1, H_2, H_3, H_4 değişkenlerinden oluşan 160 bitlik buffer' a başlangıç değerleri verilir. Bu değerler H_0=67452301, H_1=EFCDAB89, H_2=98BADCFE, H_3=10325476, H_4=C3D2E1F0 şeklindedir.

3. 512 bit olan blok, 32 bitlik 16 kelimeye ayrılır. Bu kelimeler W0, W1...W15 olarak gösterilir. Bu 16 kelimeden 80 adımlık döngüde kullanılacak kalan 64 kelime;

t=16' dan t=79' a Wt = S^1 (W_{t-3} XOR W_{t-8} XOR W_{t-14} XOR W_{t-16}) şeklinde elde edilir. S^1 bir defa dairesel sola kaydırma işlemini ifade eder.

4. İlk adım için A=H_0, B=H_1, C=H_2, D=H_3, E=H_4 şeklinde belirlenir. 80 adımlık döngünün her adımında TEMP=S^5(A)+f_t(B,C,D)+E+W_t+K_t olacak şekilde TEMP geçici değişkeni hesaplanır. Burada **S^5(A)** A' nın 5 bit dairesel sola kaydırılması işlemini ifade ederken, **K_t** döngülerde kullanılacak sabit değerleri ifade eder. **t=0'** dan **t=79** değerine kadar sürecek döngüde kullanılacak olan **f_t(B,C,D)** fonksiyonları Şekil 3.3' te, yine bu 80 adımda kullanılacak 80 tane K_t sabit değeri Şekil 3.4' de görülmektedir. Her döngüde geçerli 512 bitlik blok alınır ve A, B, C, D, E geçici bellek değerleri **E=D** (bir önceki adımın D değeri), **D=C** (bir önceki adımın C değeri), **C=S^{30}(B)**, **B=A** (bir önceki adımın A değeri), **A=TEMP** olacak şekilde güncellenir[38,39].

Adım	Fonksiyon Adı	Fonksiyon Değeri
($0 \leq t \leq 19$)	ft(B,C,D)	(B and C) or (B' and D)
($20 \leq t \leq 39$)	ft(B,C,D)	B $\oplus$ C $\oplus$ D
($40 \leq t \leq 59$)	ft(B,C,D)	(B and C) or (B and D) or (C and D)
($60 \leq t \leq 79$)	ft(B,C,D)	B $\oplus$ C $\oplus$ D

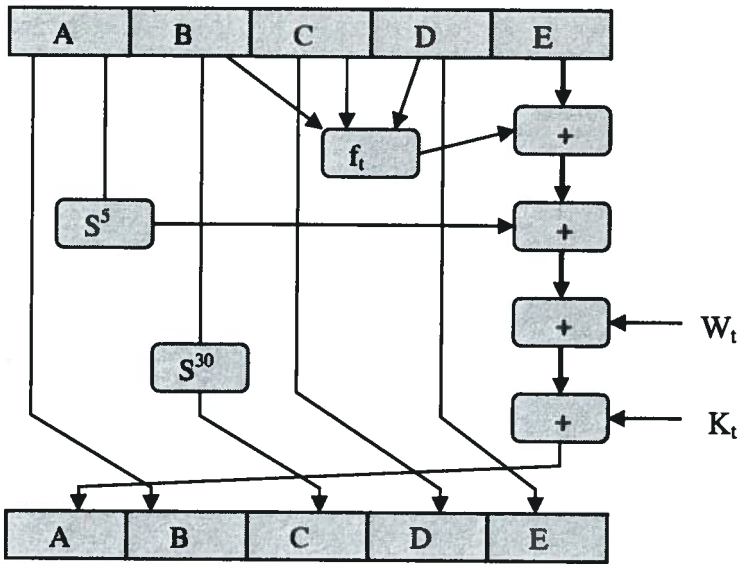
Şekil 3.3 Üç tane 32 bit değişkenden bir tane 32 bit değişken üreten fonksiyonlar

Adım	Onaltılık gösterim
$0 \leq t \leq 19$	Kt= 5A827999
$20 \leq t \leq 39$	Kt= 6ED9EBA1
$30 \leq t \leq 59$	Kt = 8F1BBCDC
$60 \leq t \leq 79$	Kt = CA62C1D6

Şekil 3.4 SHA-1 döngülerinde eklenen sabitler

80. adım sonunda elde edilen A, B, C, D, E değişkenleri H_0, H_1, H_2, H_3, H_4 değişkenlerine $H_0=H_0+A$, $H_1=H_1+B$, $H_2=H_2+C$, $H_3=H_3+D$, $H_4=H_4+E$ olacak şekilde eklenir, elde edilen H_0, H_1, H_2, H_3, H_4 değişkenleri SHA-1' in oluşturduğu 160 bitlik özet veriyi oluşturmaktadır.

SHA-1' in tek bir adımının işleyişi Şekil 3.5' de görülmektedir.



Şekil 3.5 SHA-1' de bir adımın işleyişi

SHA-1 fonksiyonu örnek bir mesaj üzerinde uygulanacak olursa;

Mesaj '01100001 01100010 01100011' olarak seçilsin. İlk önce bu veriye padding işlemi uygulanır yani verinin sonuna bir adet **1** ve 512 bitten 64 bit eksik olana kadar (448. bite kadar) **0** eklenir. Ve veri hexadecimal sayı sisteminde ifade edilir. Son 64 bit, yani son iki kelime verinin ilk uzunluğunu saklayacaktır. Burada orjinal veri uzunluğu 24 bittir ve hexadecimal olarak 18' e karşılık gelir.

Padding işleminden sonra veri hexadecimal olarak;

61626380 00000000 00000000 00000000

00000000 00000000 00000000 00000000

00000000 00000000 00000000 00000000

00000000 00000000 00000000 00000018 şeklinde elde edilir. Bu veri;

W[0] = 61626380,
W[1] = 00000000,
W[2] = 00000000,
W[3] = 00000000,
W[4] = 00000000,
W[5] = 00000000,
W[6] = 00000000,
W[7] = 00000000,
W[8] = 00000000,
W[9] = 00000000,
W[10] = 00000000,
W[11] = 00000000,
W[12] = 00000000,
W[13] = 00000000,
W[14] = 00000000,
W[15] = 00000018

olmak üzere 32 bitlik 16 kelimeye ayrılır. Bu 16 kelimeden 80 adımlık döngüde kullanılacak olan kalan 64 tane W_t değeri;
t=16' dan **t=79'** a $W_t = S^1(W_{t-3}$ XOR W_{t-8} XOR W_{t-14} XOR $W_{t-16})$ eşitliğinden elde edilir. Her adımda ilgili W_t değeri kullanılacaktır. $\mathbf{S^1}$ bir defa dairesel sola kaydırma işlemini ifade eder.

32 bitlik H_0, H_1, H_2, H_3, H_4 değişkenlerinin başlangıç değerleri H_0=67452301, H_1=EFCDAB89, H_2=98BADCFE, H_3=10325476, H_4=C3D2E1F0 şeklindedir.

İlk adım uygulanacak olursa(t=0);

$TEMP=S^5(A)+f_t(B,C,D)+E+W_t+K_t$, E=D, D=C, $C=S^{30}(B)$, B=A, A=TEMP

Eşitliklerinden E=D, D=C, B=A olacak şekilde yeni E, D, B değerleri eski D, C, A değerlerinden hemen elde edilir. Yeni C değeri için B değişkeni 30 bit sola dairesel olarak kaydırılır. Kaydırmadan sonra en sağdaki bit en soldaki bite eklenir ve yeni C değeri elde

edilmiş olur. A değişkeninin yeni değerini elde etmek için öncelikle TEMP geçici değişken değeri elde edilir.

TEMP değerinin elde edilmesi;

S^5**(A)** ifadesiyle A değişkeni 5 defa dairesel sola kaydırılır, f_t**(B,C,D)** fonksiyonunda t=0 olduğu için Şekil 3.3' ten f_t(B,C,D)= (B and C) or (B' and D) eşitliği kullanılarak B, C, D değişkenlerinden bir tane 32 bitlik değer elde edilir. Ardından S^5**(A) , ft(B,C,D), E, Wt, Kt** değerleri toplanarak TEMP geçici değişkeni hesaplanmış olur. Burada t=0 olduğundan Wt=W[0]=61626380 ve yine t=0 olduğundan Şekil 3.4' den K_t= K_0= 5A827999 değerleri kullanılır.

Son olarak TEMP geçici değişken değerinin A değişkenine kopyalanmasıyla A, B, C, D, E değişkenlerinin son değerleri; **A=0116FC33, B=67452301, C=7BF36AE2, D=98BADCFE, E=10325476** şeklinde elde edilmiş olur. Aynı işlemler t=0' dan t=79 olana kadar 80 adım için uygulanırsa A, B, C, D, E değişken değerleri;

	A	B	C	D	E
t = 0:	0116FC33	67452301	7BF36AE2	98BADCFE	10325476
t = 1:	8990536D	0116FC33	59D148C0	7BF36AE2	98BADCFE
t = 2:	A1390F08	8990536D	C045BF0C	59D148C0	7BF36AE2
t = 3:	CDD8E11B	A1390F08	626414DB	C045BF0C	59D148C0
t = 4:	CFD499DE	CDD8E11B	284E43C2	626414DB	C045BF0C
t = 5:	3FC7CA40	CFD499DE	F3763846	284E43C2	626414DB
t = 6:	993E30C1	3FC7CA40	B3F52677	F3763846	284E43C2
t = 7:	9E8C07D4	993E30C1	0FF1F290	B3F52677	F3763846
t = 8:	4B6AE328	9E8C07D4	664F8C30	0FF1F290	B3F52677
t = 9:	8351F929	4B6AE328	27A301F5	664F8C30	0FF1F290
t = 10:	FBDA9E89	8351F929	12DAB8CA	27A301F5	664F8C30

t = 11: 63188FE4 FBDA9E89 60D47E4A 12DAB8CA 27A301F5

t = 12: 4607B664 63188FE4 7EF6A7A2 60D47E4A 12DAB8CA

t = 13: 9128F695 4607B664 18C623F9 7EF6A7A2 60D47E4A

t = 14: 196BEE77 9128F695 1181ED99 18C623F9 7EF6A7A2

t = 15: 20BDD62F 196BEE77 644A3DA5 1181ED99 18C623F9

t = 16: 4E925823 20BDD62F C65AFB9D 644A3DA5 1181ED99

t = 17: 82AA6728 4E925823 C82F758B C65AFB9D 644A3DA5

t = 18: DC64901D 82AA6728 D3A49608 C82F758B C65AFB9D

t = 19: FD9E1D7D DC64901D 20AA99CA D3A49608 C82F758B

t = 20: 1A37B0CA FD9E1D7D 77192407 20AA99CA D3A49608

t = 21: 33A23BFC 1A37B0CA 7F67875F 77192407 20AA99CA

t = 22: 21283486 33A23BFC 868DEC32 7F67875F 77192407

t = 23: D541F12D 21283486 0CE88EFF 868DEC32 7F67875F

t = 24: C7567DC6 D541F12D 884A0D21 0CE88EFF 868DEC32

t = 25: 48413BA4 C7567DC6 75507C4B 884A0D21 0CE88EFF

t = 26: BE35FBD5 48413BA4 B1D59F71 75507C4B 884A0D21

t = 27: 4AA84D97 BE35FBD5 12104EE9 B1D59F71 75507C4B

t = 28: 8370B52E 4AA84D97 6F8D7EF5 12104EE9 B1D59F71

t = 29: C5FBAF5D 8370B52E D2AA1365 6F8D7EF5 12104EE9

t = 30: 1267B407 C5FBAF5D A0DC2D4B D2AA1365 6F8D7EF5

t = 31: 3B845D33 1267B407 717EEBD7 A0DC2D4B D2AA1365

t = 32: 046FAA0A 3B845D33 C499ED01 717EEBD7 A0DC2D4B

t = 33: 2C0EBC11 046FAA0A CEE1174C C499ED01 717EEBD7

t = 34: 21796AD4 2C0EBC11 811BEA82 CEE1174C C499ED01

t = 35: DCBBB0CB 21796AD4 4B03AF04 811BEA82 CEE1174C

t = 36: 0F511FD8 DCBBB0CB 085E5AB5 4B03AF04 811BEA82

t = 37: DC63973F 0F511FD8 F72EEC32 085E5AB5 4B03AF04

t = 38: 4C986405 DC63973F 03D447F6 F72EEC32 085E5AB5

t = 39: 32DE1CBA 4C986405 F718E5CF 03D447F6 F72EEC32

t = 40: FC87DEDF 32DE1CBA 53261901 F718E5CF 03D447F6

t = 41: 970A0D5C FC87DEDF 8CB7872E 53261901 F718E5CF

t = 42: 7F193DC5 970A0D5C FF21F7B7 8CB7872E 53261901

t = 43: EE1B1AAF 7F193DC5 25C28357 FF21F7B7 8CB7872E

t = 44: 40F28E09 EE1B1AAF 5FC64F71 25C28357 FF21F7B7

t = 45: 1C51E1F2 40F28E09 FB86C6AB 5FC64F71 25C28357

t = 46: A01B846C 1C51E1F2 503CA382 FB86C6AB 5FC64F71

t = 47: BEAD02CA A01B846C 8714787C 503CA382 FB86C6AB

t = 48: BAF39337 BEAD02CA 2806E11B 8714787C 503CA382

t = 49: 120731C5 BAF39337 AFAB40B2 2806E11B 8714787C

t = 50: 641DB2CE 120731C5 EEBCE4CD AFAB40B2 2806E11B

t = 51: 3847AD66 641DB2CE 4481CC71 EEBCE4CD AFAB40B2

t = 52: E490436D 3847AD66 99076CB3 4481CC71 EEBCE4CD

t = 53: 27E9F1D8 E490436D 8E11EB59 99076CB3 4481CC71

t = 54: 7B71F76D 27E9F1D8 792410DB 8E11EB59 99076CB3

t = 55: 5E6456AF 7B71F76D 09FA7C76 792410DB 8E11EB59

t = 56: C846093F 5E6456AF 5EDC7DDB 09FA7C76 792410DB

t = 57: D262FF50 C846093F D79915AB 5EDC7DDB 09FA7C76

t = 58: 09D785FD D262FF50 F211824F D79915AB 5EDC7DDB

t = 59: 3F52DE5A 09D785FD 3498BFD4 F211824F D79915AB

t = 60: D756C147 3F52DE5A 4275E17F 3498BFD4 F211824F

t = 61: 548C9CB2 D756C147 8FD4B796 4275E17F 3498BFD4

t = 62: B66C020B 548C9CB2 F5D5B051 8FD4B796 4275E17F

t = 63: 6B61C9E1 B66C020B 9523272C F5D5B051 8FD4B796

t = 64: 19DFA7AC 6B61C9E1 ED9B0082 9523272C F5D5B051

t = 65: 101655F9 19DFA7AC 5AD87278 ED9B0082 9523272C

t = 66: 0C3DF2B4 101655F9 0677E9EB 5AD87278 ED9B0082

t = 67: 78DD4D2B 0C3DF2B4 4405957E 0677E9EB 5AD87278

t = 68: 497093C0 78DD4D2B 030F7CAD 4405957E 0677E9EB

t = 69: 3F2588C2 497093C0 DE37534A 030F7CAD 4405957E

t = 70: C199F8C7 3F2588C2 125C24F0 DE37534A 030F7CAD

t = 71: 39859DE7 C199F8C7 8FC96230 125C24F0 DE37534A

t = 72: EDB42DE4 39859DE7 F0667E31 8FC96230 125C24F0

t = 73: 11793F6F EDB42DE4 CE616779 F0667E31 8FC96230

t = 74: 5EE76897 11793F6F 3B6D0B79 CE616779 F0667E31

t = 75: 63F7DAB7 5EE76897 C45E4FDB 3B6D0B79 CE616779

t = 76: A079B7D9 63F7DAB7 D7B9DA25 C45E4FDB 3B6D0B79

t = 77: 860D21CC A079B7D9 D8FDF6AD D7B9DA25 C45E4FDB

t = 78: 5738D5E1 860D21CC 681E6DF6 D8FDF6AD D7B9DA25

t = 79: 42541B35 5738D5E1 21834873 681E6DF6 D8FDF6AD

şeklinde elde edilir. 80. adım sonunda ise elde edilen A, B, C, D, E değişken değerleri H_0, H_1, H_2, H_3, H_4 değişkenlerine eklenir.

H_0 = 67452301 + 42541B35 = A9993E36

H_1 = EFCDAB89 + 5738D5E1 = 4706816A

H_2 = 98BADCFE + 21834873 = BA3E2571

H_3 = 10325476 + 681E6DF6 = 7850C26C

H_4 = C3D2E1F0 + D8FDF6AD = 9CD0D89D elde edilir.

SHA-1 fonksiyonu sonucunda oluşan 160 bit özet değer:

"A9993E36 4706816A BA3E2571 7850C26C 9CD0D89D" olarak elde edilir[39].

3.1.3. SHA-2 Ailesi

SHA-1 algoritmasının kırılmasının verdiği güvensizlik nedeniyle SHA-2 algoritmaları geliştirilmiştir. SHA-224, SHA-256, SHA-384 ve SHA-512 algoritmaları SHA-2 ailesi olarak adlandırılır. Bu algoritmalar SHA-1' den farklı olarak sırasıyla 224, 256, 384, 512 bit özet çıktısı üreterek daha güvenli bir şifreleme imkanı sunarlar. Şekil 3.6 SHA-2 ailesinin özelliklerini sunmaktadır.

Algoritma	Mesaj Boyutu(Bit)	Block Boyutu(Bit)	Kelime Boyutu(Bit)	Mesaj Özet Uzunluğu(Bit)
SHA-1	$<2^{64}$	512	32	160
SHA-224	$<2^{64}$	512	32	224
SHA-256	$<2^{64}$	512	32	256
SHA-384	$<2^{128}$	1024	64	384
SHA-512	$<2^{128}$	1024	64	512

Şekil 3.6 SHA-2 ailesinin özellikleri

3.1.3.1. SHA-256 Algoritması

SHA-2 ailesi üyelerinden olan SHA-256 algoritması veriyi 512 bitlik bloklar olarak işler. İşlem adımları açıklanacak olursa;

1. Ekleme (Padding) İşlemi: Veri ikili sistemde ifade edilir ve bu verinin sonuna 512 bitin en yakın katından 64 bit eksik olacak şekilde, verinin sonuna bir adet 1 ve geri kalanlar için ise 0 eklenir. Son 64 bitlik kısım ise verinin uzunluğunu belirtmekte kullanılır. İkili sisteme çevrilmiş olan bu veri, her dört bitlik grup bir hexadecimal sayıya karşılık gelecek şekilde hexadecimal sayı sistemine çevrilir.

2. 32 bitlik H_0, H_1, H_2, H_3, H_4, H_5, H_6, H_7 değişkenlerinden oluşan 256 bitlik buffer' a başlangıç değerleri verilir. Bu değerler H_0=6a09e667, H_1=bb67ae85, H_2=3c6ef372, H_3=a54ff53a, H_4=510e527f, H_5=9b05688c, H_6=1f83d9ab, H_7=5be0cd19 şeklindedir.

3. 512 bit olan blok, 32 bitlik 16 kelimeye ayrılır. Bu kelimeler W0, W1...W15 olarak gösterilir. Bu 16 kelimeden 64 adımlık döngüde kullanılacak kalan 48 kelime, t=16 dan t=63' e kadar $Wt= \sigma_1 (W_{t-2})+ W_{t-7} + \sigma_0 (W_{t-15})+ W_{t-16}$ şeklinde hesaplanır. Toplam 64 Wt değeri elde edilmiş olur. Burada;
   $\sigma_0 = ROTR^7(Wt)$ XOR $ROTR^{18}(Wt) + SHR^3(Wt)$,
   $\sigma_1 = ROTR^{17}(Wt)$ XOR $ROTR^{19}(Wt) + SHR^{10}(Wt)$ şeklindedir.
   SHR sağa kaydırmayı, ROTR ise dairesel sağa kaydırmayı ifade eder.
4. İlk adım için A=H_0, B=H_1, C=H_2, D=H_3, E=H_4, F=H_5, G=H_6, H=H_7 şeklinde belirlenir.
5. 64 adımlık döngünün her bir adımında;
   $T_1=h+\Sigma_1(E)+Ch(E,F,G)+K_t+W_t$ ve $T_2= \Sigma_0 (A)+Maj(A,B,C)$ olacak şekilde T geçici değişkenleri hesaplanır. Burada $\Sigma_0=ROTR^2(x)$ XOR $ROTR^{13}(x)$ XOR $ROTR^{22}(x)$,
   $\Sigma_1=ROTR^6(x)$ XOR $ROTR^{11}(x)$ XOR $ROTR^{25}(x)$,
   $Ch(E,F,G)$=(E AND F) XOR (^E AND G) ve
   $Maj(A,B,C)$=(A AND B) XOR (A AND C) XOR (B AND C) şeklinde hesaplanır.
   Yine burada kullanılan Wt değeri 3. adımda hesaplanan W değerlerinden ilgili adımın W değerini ifade eder. SHA-256 için 64 adet sabit Kt değerleri ise soldan sağa sırasıyla;

428a2f98 71374491 b5c0fbcf e9b5dba5 3956c25b 59f111f1 923f82a4 ab1c5ed5 d807aa98 12835b01 243185be 550c7dc3 72be5d74 80deb1fe 9bdc06a7 c19bf174 e49b69c1 efbe4786 0fc19dc6 240ca1cc 2de92c6f 4a7484aa 5cb0a9dc 76f988da 983e5152 a831c66d b00327c8 bf597fc7 c6e00bf3 d5a79147 06ca6351 14292967 27b70a85 2e1b2138 4d2c6dfc 53380d13 650a7354 766a0abb 81c2c92e 92722c85 a2bfe8a1 a81a664b c24b8b70 c76c51a3 d192e819 d6990624 f40e3585 106aa070 19a4c116 1e376c08 2748774c 34b0bcb5 391c0cb3 4ed8aa4a 5b9cca4f 682e6ff3 748f82ee 78a5636f 84c87814 8cc70208 90befffa a4506ceb bef9a3f7 c67178f2

şeklindedir. 64 adımın her birinde geçici T_1, T_2 değerlerinin hesaplanmasının ardından her biri 32 bit olan 8 değişken **H=G**, **G=F**, **F=E**, **E=D+T1**, **D=C**, **C=B**, **B=A** ve $\mathbf{A=T_1+T_2}$ olacak şekilde güncellenir.

6. Döngünün bitmesiyle son adımda elde edilen A, B, C, D, E, F, G, H değişken değerleri başlangıçtaki sabit H_0, H_1, H_2, H_3, H_4, H_5, H_6, H_7 değişkenlerine $H_0=A+H_0$, $H_1=B+H_1$, $H_2=C+H_2$, $H_3=D+H_3$, $H_4=E+H_4$, $H_5=F+H_5$, $H_6=G+H_6$, $H_7=H+H_7$ olacak şekilde eklenir. SHA-256 algoritmasının 256 bitlik özet değeri **H0||H1||H2||H3||H4||H5||H6||H7** olarak elde edilmiş olur.

SHA-256 fonksiyonu örnek bir mesaj üzerinde uygulanacak olursa;

Mesaj '01100001 01100010 01100011' olarak seçilsin. İlk önce bu veriye padding işlemi uygulanır yani verinin sonuna bir adet **1** ve 512 bitten 64 bit eksik olana kadar (448. bite kadar) **0** eklenir. Ve veri hexadecimal sayı sisteminde ifade edilir. Son 64 bit, yani son iki kelime verinin ilk uzunluğunu saklayacaktır. Burada orjinal veri uzunluğu 24 bittir ve hexadecimal olarak 18' e karşılık gelir.

Padding işleminden sonra veri hexadecimal olarak;

61626380 00000000 00000000 00000000

00000000 00000000 00000000 00000000

00000000 00000000 00000000 00000000

00000000 00000000 00000000 00000018 şeklinde elde edilir. Bu veri;

W[0] = 61626380,

W[1] = 00000000,

W[2] = 00000000,

W[3] = 00000000,

W[4] = 00000000,

W[5] = 00000000,

W[6] = 00000000,

W[7] = 00000000,

W[8] = 00000000,

W[9] = 00000000,

W[10] = 00000000,

W[11] = 00000000,

W[12] = 00000000,

W[13] = 00000000,

W[14] = 00000000,

W[15] = 00000018

olmak üzere 32 bitlik 16 kelimeye ayrılır. Bu 16 kelimeden diğer 48 tane W_t değeri;

$Wt= \sigma_1 (W_{t-2}) + W_{t-7} + \sigma_0 (W_{t-15})+ W_{t-16}$ şeklinde elde edilir. Toplam 64 Wt değeri elde edilmiş olur. Burada σ_0 ve σ_1 değerleri;

$\sigma_0 = ROTR^7(Wt)$ XOR $ROTR^{18}(Wt) + SHR^3(Wt)$,

$\sigma_1 = ROTR^{17}(Wt)$ XOR $ROTR^{19}(Wt) + SHR^{10}(Wt)$ olacak şekilde hesaplanır. SHR sağa kaydırmayı, ROTR ise dairesel sağa kaydırmayı ifade eder.

32 bitlik H_0, H_1, H_2, H_3, H_4, H_5, H_6, H_7 değişkenlerinden oluşan 256 bitlik buffer' a başlangıç değerleri verilir. Bu değerler H_0=6a09e667, H_1=bb67ae85, H_2=3c6ef372, H_3=a54ff53a, H_4=510e527f, H_5=9b05688c, H_6=1f83d9ab, H_7=5be0cd19 şeklindedir.

64 adımlık döngünün her bir adımında;

$T_1=h+\Sigma_1(E)+Ch(E,F,G)+K_t+W_t$ ve $T_2=\Sigma_0(A)+Maj(A,B,C)$ olacak şekilde T geçici değişkenleri hesaplanır. Burada;

$\Sigma_0=ROTR^2(x)$ XOR $ROTR^{13}(x)$ XOR $ROTR^{22}(x)$,

Σ_1=ROTR^6 (x) XOR ROTR^{11}(x) XOR ROTR^{25}(x),

Ch(E,F,G)=(E AND F) XOR (^E AND Z) ve

Maj(A,B,C)=(A AND B) XOR (A AND C) XOR (B AND C) şeklinde hesaplanır.

64 adımın her birinde geçici T_1, T_2 değerlerinin hesaplanmasının ardından her biri 32 bit olan 8 değişken **H=G**, **G=F**, **F=E**, **E=D+T_1**, **D=C**, **B=A** ve **A=T_1+T_2** olacak şekilde güncellenir. 64 adımdaki A, B, C, D, E, F, G, H değerleri;

	A	*B*	*C*	*D*	*E*	*F*	*G*	*H*
t = 0 :	5d6aebcd	6a09e667	bb67ae85	3c6ef372	fa2a4622	510e527f	9b05688c	1f83d9ab
t = 1 :	5a6ad9ad	5d6aebcd	6a09e667	bb67ae85	78ce7989	fa2a4622	510e527f	9b05688c
t = 2 :	c8c347a7	5a6ad9ad	5d6aebcd	6a09e667	f92939eb	78ce7989	fa2a4622	510e527f
t = 3 :	d550f666	c8c347a7	5a6ad9ad	5d6aebcd	24e00850	f92939eb	78ce7989	fa2a4622
t = 4 :	04409a6a	d550f666	c8c347a7	5a6ad9ad	43ada245	24e00850	f92939eb	78ce7989
t = 5 :	2b4209f5	04409a6a	d550f666	c8c347a7	714260ad	43ada245	24e00850	f92939eb
t = 6 :	e5030380	2b4209f5	04409a6a	d550f666	9b27a401	714260ad	43ada245	24e00850
t = 7 :	85a07b5f	e5030380	2b4209f5	04409a6a	0c657a79	9b27a401	714260ad	43ada245
t = 8 :	8e04ecb9	85a07b5f	e5030380	2b4209f5	32ca2d8c	0c657a79	9b27a401	714260ad
t = 9 :	8c87346b	8e04ecb9	85a07b5f	e5030380	1cc92596	32ca2d8c	0c657a79	9b27a401
t = 10 :	4798a3f4	8c87346b	8e04ecb9	85a07b5f	436b23e8	1cc92596	32ca2d8c	0c657a79
t = 11 :	f71fc5a9	4798a3f4	8c87346b	8e04ecb9	816fd6e9	436b23e8	1cc92596	32ca2d8c
t = 12 :	87912990	f71fc5a9	4798a3f4	8c87346b	1e578218	816fd6e9	436b23e8	1cc92596
t = 13 :	d932eb16	87912990	f71fc5a9	4798a3f4	745a48de	1e578218	816fd6e9	436b23e8
t = 14 :	c0645fde	d932eb16	87912990	f71fc5a9	0b92f20c	745a48de	1e578218	816fd6e9
t = 15 :	b0fa238e	c0645fde	d932eb16	87912990	07590dcd	0b92f20c	745a48de	1e578218
t = 16 :	21da9a9b	b0fa238e	c0645fde	d932eb16	8034229c	07590dcd	0b92f20c	745a48de
t = 17 :	c2fbd9d1	21da9a9b	b0fa238e	c0645fde	846ee454	8034229c	07590dcd	0b92f20c
t = 18 :	fe777bbf	c2fbd9d1	21da9a9b	b0fa238e	cc899961	846ee454	8034229c	07590dcd
t = 19 :	e1f20c33	fe777bbf	c2fbd9d1	21da9a9b	b0638179	cc899961	846ee454	8034229c
t = 20 :	9dc68b63	e1f20c33	fe777bbf	c2fbd9d1	8ada8930	b0638179	cc899961	846ee454
t = 21 :	c2606d6d	9dc68b63	e1f20c33	fe777bbf	e1257970	8ada8930	b0638179	cc899961
t = 22 :	a7a3623f	c2606d6d	9dc68b63	e1f20c33	49f5114a	e1257970	8ada8930	b0638179
t = 23 :	c5d53d8d	a7a3623f	c2606d6d	9dc68b63	aa47c347	49f5114a	e1257970	8ada8930
t = 24 :	1c2c2838	c5d53d8d	a7a3623f	c2606d6d	2823ef91	aa47c347	49f5114a	e1257970

t = 25 : cde8037d 1c2c2838 c5d53d8d a7a3623f 14383d8e 2823ef91 aa47c347 49f5114a

t = 26 : b62ec4bc cde8037d 1c2c2838 c5d53d8d c74c6516 14383d8e 2823ef91 aa47c347

t = 27 : 77d37528 b62ec4bc cde8037d 1c2c2838 edffbff8 c74c6516 14383d8e 2823ef91

t = 28 : 363482c9 77d37528 b62ec4bc cde8037d 6112a3b7 edffbff8 c74c6516 14383d8e

t = 29 : a0060b30 363482c9 77d37528 b62ec4bc ade79437 6112a3b7 edffbff8 c74c6516

t = 30 : ea992a22 a0060b30 363482c9 77d37528 0109ab3a ade79437 6112a3b7 edffbff8

t = 31 : 73b33bf5 ea992a22 a0060b30 363482c9 ba591112 0109ab3a ade79437 6112a3b7

t = 32 : 98e12507 73b33bf5 ea992a22 a0060b30 9cd9f5f6 ba591112 0109ab3a ade79437

t = 33 : fe604df5 98e12507 73b33bf5 ea992a22 59249dd3 9cd9f5f6 ba591112 0109ab3a

t = 34 : a9a7738c fe604df5 98e12507 73b33bf5 085f3833 59249dd3 9cd9f5f6 ba591112

t = 35 : 65a0cfe4 a9a7738c fe604df5 98e12507 f4b002d6 085f3833 59249dd3 9cd9f5f6

t = 36 : 41a65cb1 65a0cfe4 a9a7738c fe604df5 0772a26b f4b002d6 085f3833 59249dd3

t = 37 : 34df1604 41a65cb1 65a0cfe4 a9a7738c a507a53d 0772a26b f4b002d6 085f3833

t = 38 : 6dc57a8a 34df1604 41a65cb1 65a0cfe4 f0781bc8 a507a53d 0772a26b f4b002d6

t = 39 : 79ea687a 6dc57a8a 34df1604 41a65cb1 1efbc0a0 f0781bc8 a507a53d 0772a26b

t = 40 : d6670766 79ea687a 6dc57a8a 34df1604 26352d63 1efbc0a0 f0781bc8 a507a53d

t = 41 : df46652f d6670766 79ea687a 6dc57a8a 838b2711 26352d63 1efbc0a0 f0781bc8

t = 42 : 17aa0dfe df46652f d6670766 79ea687a decd4715 838b2711 26352d63 1efbc0a0

t = 43 : 9d4baf93 17aa0dfe df46652f d6670766 fda24c2e decd4715 838b2711 26352d63

t = 44 : 26628815 9d4baf93 17aa0dfe df46652f a80f11f0 fda24c2e decd4715 838b2711

t = 45 : 72ab4b91 26628815 9d4baf93 17aa0dfe b7755da1 a80f11f0 fda24c2e decd4715

t = 46 : a14c14b0 72ab4b91 26628815 9d4baf93 d57b94a9 b7755da1 a80f11f0 fda24c2e

t = 47 : 4172328d a14c14b0 72ab4b91 26628815 fecf0bc6 d57b94a9 b7755da1 a80f11f0

t = 48 : 05757ceb 4172328d a14c14b0 72ab4b91 bd714038 fecf0bc6 d57b94a9 b7755da1

t = 49 : f11bfaa8 05757ceb 4172328d a14c14b0 6e5c390c bd714038 fecf0bc6 d57b94a9

t = 50 : 7a0508a1 f11bfaa8 05757ceb 4172328d 52f1ccf7 6e5c390c bd714038 fecf0bc6

t = 51 : 886e7a22 7a0508a1 f11bfaa8 05757ceb 49231c1e 52f1ccf7 6e5c390c bd714038

t = 52 : 101fd28f 886e7a22 7a0508a1 f11bfaa8 529e7d00 49231c1e 52f1ccf7 6e5c390c

t = 53 : f5702fdb 101fd28f 886e7a22 7a0508a1 9f4787c3 529e7d00 49231c1e 52f1ccf7

t = 54 : 3ec45cdb f5702fdb 101fd28f 886e7a22 e50e1b4f 9f4787c3 529e7d00 49231c1e

t = 55 : 38cc9913 3ec45cdb f5702fdb 101fd28f 54cb266b e50e1b4f 9f4787c3 529e7d00

t = 56 : fcd1887b 38cc9913 3ec45cdb f5702fdb 9b5e906c 54cb266b e50e1b4f 9f4787c3

t = 57 : c062d46f fcd1887b 38cc9913 3ec45cdb 7e44008e 9b5e906c 54cb266b e50e1b4f

t = 58 : ffb70472 c062d46f fcd1887b 38cc9913 6d83bfc6 7e44008e 9b5e906c 54cb266b

t = 59 : b6ae8fff ffb70472 c062d46f fcd1887b b21bad3d 6d83bfc6 7e44008e 9b5e906c

t = 60 : b85e2ce9 b6ae8fff ffb70472 c062d46f 961f4894 b21bad3d 6d83bfc6 7e44008e

t = 61 : 04d24d6c b85e2ce9 b6ae8fff ffb70472 948d25b6 961f4894 b21bad3d 6d83bfc6

t = 62 : d39a2165 04d24d6c b85e2ce9 b6ae8fff fb121210 948d25b6 961f4894 b21bad3d

t = 63 : 506e3058 d39a2165 04d24d6c b85e2ce9 5ef50f24 fb121210 948d25b6 961f4894

şeklinde hesaplanır. Son adımın ardından H_0, H_1, H_2, H_3, H_4, H_5, H_6, H_7 değişkenleri;

$H_0=H_0+A=$ 6A09E667 + 506E3058 = **BA7816BF,**

$H_1=H_1+B=$ BB67AE85 + D39A2165 = **8F01CFEA,**

$H_2=H_2+C=$ 3C6EF372 + 04D24D6C = **414140DE,**

$H_3=H_3+D=$ A54FF53A + B85E2CE9 = **5DAE2223,**

$H_4=H_4+E=$ 510E527F + 5EF50F24 = **B00361A3,**

$H_5=H_5+F=$ 9B05688C + FB121210 = **96177A9C,**

$H_6=H_6+G=$ 1F83D9AB + 948D25B6 = **B410FF61,**

$H_7=H_7+F=$ 5BE0CD19 + 961F4894 = **F20015AD** olarak elde edilir. 256 bit özet değer:
BA7816BF8F01CFEA414140DE5DAE2223B00361A396177A9CB410FF61F20015AD olarak elde edilmiş olur.

SHA-2 ailesinin özet mesaj uzunluğu SHA-1' e göre daha uzun olsada yapısı SHA-1' e benzediğinden bu algoritmaların kırılma ihtimallarine karşı araştırmacılar daha güvenli bir bir algoritma arayışına girmişlerdir.

3.1.3.2. SHA-512 Algoritması

SHA-2 ailesi üyelerinden olan SHA-512 algoritması veriyi 1024 bitlik bloklar olarak işler. İşlem adımları açıklanacak olursa;

1. Ekleme (Padding) İşlemi: Veri ikili sistemde ifade edilir ve bu verinin sonuna 1024 bitin en yakın katından 128 bit eksik olacak şekilde, verinin sonuna bir adet **1** ve geri kalanlar için ise **0** eklenir. Son 128 bitlik kısım ise verinin uzunluğunu belirtmekte kullanılır. İkili sisteme çevrilmiş olan bu veri, her dört bitlik grup bir hexadecimal sayıya karşılık gelecek şekilde hexadecimal sayı sistemine çevrilir.

2. 64 bitlik H_0, H_1, H_2, H_3, H_4, H_5, H_6, H_7 değişkenlerinden oluşan 512 bitlik buffer' a başlangıç değerleri verilir. Bu değerler H_0=6a09e667f3bcc908, H_1=bb67ae8584caa73b, H_2=3c6ef372fe94f82b, H_3=a54ff53a5f1d36f1, H_4=510e527fade682d1, H_5=9b05688c2b3e6c1f, H_6=1f83d9abfb41bd6b, H_7=5be0cd19137e2179 şeklindedir.

3. 1024 bit olan blok, 64 bitlik 16 kelimeye ayrılır. Bu kelimeler W0, W1...W15 olarak gösterilir. Bu 16 kelimeden 80 adımlık döngüde kullanılacak kalan 64 kelime, t=16' dan t=79' a kadar $Wt= \sigma_1 (W_{t-2})+ W_{t-7} + \sigma_0 (W_{t-15})+ W_{t-16}$ şeklinde hesaplanır. Toplam 80 tane Wt değeri elde edilmiş olur. Burada;

   $\sigma_0 = ROTR^1(Wt)$ XOR $ROTR^8(Wt) + SHR^7(Wt)$ ve

   $\sigma_1 = ROTR^{19}(Wt)$ XOR $ROTR^{61}(Wt) + SHR^6(Wt)$ şeklindedir. SHR sağa kaydırmayı, ROTR ise dairesel sağa kaydırmayı ifade eder.

4. İlk adım için $A=H_0$, $B=H_1$, $C=H_2$, $D=H_3$, $E=H_4$, $F=H_5$, $G=H_6$, $H=H_7$ şeklinde belirlenir.

5. 80 adımlık döngünün her bir adımında;

   $T_1=h+\Sigma_1(E)+Ch(E,F,G)+K_t+W_t$ ve $T_2= \Sigma_0 (A)+Maj(A,B,C)$ olacak şekilde T geçici değişkenleri hesaplanır. Burada $\Sigma_0=ROTR^{28}$ (x) XOR $ROTR^{34}$(x) XOR $ROTR^{39}$(x),
   $\Sigma_1=ROTR^{14}$ (x) XOR $ROTR^{18}$(x) XOR $ROTR^{41}$(x),
   Ch(E,F,G)=(E AND F) XOR (^E AND G) ve
   Maj(A,B,C)=(A AND B) XOR (A AND C) XOR (B AND C) şeklinde hesaplanır. Yine burada kullanılan Wt değeri 3. adımda hesaplanan W değerlerinden ilgili adımın W değerini ifade eder. SHA-512 için 80 adet 64 bitlik sabit Kt değerleri ise soldan sağa sırasıyla;

   428a2f98d728ae22 7137449123ef65cd b5c0fbcfec4d3b2f e9b5dba58189dbbc 3956c25bf348b538 59f111f1b605d019 923f82a4af194f9b ab1c5ed5da6d8118 d807aa98a3030242 12835b0145706fbe 243185be4ee4b28c 550c7dc3d5ffb4e2 72be5d74f27b896f 80deb1fe3b1696b1 9bdc06a725c71235 c19bf174cf692694 e49b69c19ef14ad2 efbe4786384f25e3 0fc19dc68b8cd5b5 240ca1cc77ac9c65 2de92c6f592b0275 4a7484aa6ea6e483 5cb0a9dcbd41fbd4 76f988da831153b5 983e5152ee66dfab a831c66d2db43210 b00327c898fb213f bf597fc7beef0ee4 c6e00bf33da88fc2 d5a79147930aa725 06ca6351e003826f 142929670a0e6e70 27b70a8546d22ffc 2e1b21385c26c926 4d2c6dfc5ac42aed 53380d139d95b3df

650a73548baf63de 766a0abb3c77b2a8 81c2c92e47edaee6 92722c851482353b a2bfe8a14cf10364 a81a664bbc423001 c24b8b70d0f89791 c76c51a30654be30 d192e819d6ef5218 d69906245565a910 f40e35855771202a 106aa07032bbd1b8 19a4c116b8d2d0c8 1e376c085141ab53 2748774cdf8eeb99 34b0bcb5e19b48a8 391c0cb3c5c95a63 4ed8aa4ae3418acb 5b9cca4f7763e373 682e6ff3d6b2b8a3 748f82ee5defb2fc 78a5636f43172f60 84c87814a1f0ab72 8cc702081a6439ec 90befffa23631e28 a4506cebde82bde9 bef9a3f7b2c67915 c67178f2e372532b ca273eceea26619c d186b8c721c0c207 eada7dd6cde0eb1e f57d4f7fee6ed178 06f067aa72176fba 0a637dc5a2c898a6 113f9804bef90dae 1b710b35131c471b 28db77f523047d84 32caab7b40c72493 3c9ebe0a15c9bebc 431d67c49c100d4c 4cc5d4becb3e42b6 597f299cfc657e2a 5fcb6fab3ad6faec 6c44198c4a475817

şeklindedir. 80 adımın her birinde geçici T_1, T_2 değerlerinin hesaplanmasının ardından her biri 64 bit olan 8 değişken **H=G**, **G=F**, **F=E**, **E=D+T1**, **D=C**, **C=B**, **B=A** ve $\mathbf{A=T_1+T_2}$ olacak şekilde güncellenir.

6. Döngünün bitmesiyle son adımda elde edilen A, B, C, D, E, F, G, H değişken değerleri başlangıçtaki sabit H0, H1, H2, H3, H4, H5, H6, H7 değişkenlerine $H_0=A+H_0$, $H_1=B+H_1$, $H_2=C+H_2$, $H_3=D+H_3$, $H_4=E+H_4$, $H_5=F+H_5$, $H_6=G+H_6$, $H_7=H+H_7$ olacak şekilde eklenir. SHA-512 algoritmasının 512 bitlik özet değeri **H0||H1||H2||H3||H4||H5||H6||H7** olarak elde edilmiş olur.

SHA-512 fonksiyonu örnek bir mesaj üzerinde uygulanacak olursa;

Mesaj '01100001 01100010 01100011' olarak seçilsin. İlk önce bu veriye padding işlemi uygulanır yani verinin sonuna bir adet **'1'** ve 1024 bitten 128 bit eksik olana kadar (896. bite kadar) **0** eklenir. Ve veri hexadecimal sayı sisteminde ifade edilir. Son 128 bit, yani son dört kelime verinin ilk uzunluğunu saklayacaktır. Burada orjinal veri uzunluğu 24 bittir ve hexadecimal olarak 18' e karşılık gelir. Padding işleminden sonra veri hexadecimal olarak;

6162638000000000 0000000000000000 0000000000000000 0000000000000000 0000000000000000 0000000000000000 0000000000000000 0000000000000000 0000000000000000 0000000000000000 0000000000000000 0000000000000000 0000000000000000 0000000000000000 0000000000000000 0000000000000018

şeklinde elde edilir. Bu veri;

W[0] = 6162638000000000,

W[1] = 0000000000000000,

W[2] = 0000000000000000,

W[3] = 0000000000000000,

W[4] = 0000000000000000,

W[5] = 0000000000000000,

W[6] = 0000000000000000,

W[7] = 0000000000000000,

W[8] = 0000000000000000,

W[9] = 0000000000000000,

W[10] = 0000000000000000,

W[11] = 0000000000000000,

W[12] = 0000000000000000,

W[13] = 0000000000000000,

W[14] = 0000000000000000,

W[15] = 0000000000000018

olmak üzere 64 bitlik 16 kelimeye ayrılır. Bu 16 kelimeden diğer 64 tane W_t değeri; t=16' dan t=79' a $Wt= \sigma_1 (W_{t-2}) + W_{t-7} + \sigma_0 (W_{t-15})+ W_{t-16}$ şeklinde elde edilir.

Burada σ_0 ve σ_1 değerleri;

$\sigma_0 = ROTR^1(Wt)$ XOR $ROTR^8(Wt) + SHR^7(Wt)$ ve

$\sigma_1 = ROTR^{19}(Wt)$ XOR $ROTR^{61}(Wt) + SHR^6(Wt)$ olacak şekilde hesaplanır. SHR sağa kaydırmayı, ROTR ise dairesel sağa kaydırmayı ifade eder.

64 bitlik H_0, H_1, H_2, H_3, H_4, H_5, H_6, H_7 değişkenlerinden oluşan 512 bitlik buffer' a başlangıç değerleri verilir. Bu değerler H_0=6a09e667f3bcc908, H_1=bb67ae8584caa73b, H_2=3c6ef372fe94f82b, H_3=a54ff53a5f1d36f1, H_4=510e527fade682d1, H_5=9b05688c2b3e6c1f, H_6=1f83d9abfb41bd6b, H_7=5be0cd19137e2179 şeklindedir.

80 adımlık döngünün her bir adımında;

$T_1=h+\Sigma_1(E)+Ch(E,F,G)+K_t+W_t$ ve $T_2=\Sigma_0(A)+Maj(A,B,C)$ olacak şekilde T geçici değişkenleri hesaplanır. Burada;

$\Sigma_0=ROTR^{28}(x)$ XOR $ROTR^{34}(x)$ XOR $ROTR^{39}(x)$,

Σ_1=ROTR^{14}(x) XOR ROTR^{18}(x) XOR ROTR^{41}(x),

Ch(E,F,G)=(E AND F) XOR (^E AND G) ve

Maj(A,B,C)=(A AND B) XOR (A AND C) XOR (B AND C) şeklinde hesaplanır.

80 adımın her birinde geçici T_1, T_2 değerlerinin hesaplanmasının ardından her biri 64 bit olan 8 değişken **H=G**, **G=F**, **F=E**, **E=D+T_1**, **D=C**, **B=A** ve **A=T_1+T_2** olacak şekilde güncellenir. 80 adımdaki A, B, C, D, E, F, G, H değerleri;

	A / E	**B/F**	**C /G**	**D/H**
t = 0 :	f6afceb8bcfcddf5	6a09e667f3bcc908	bb67ae8584caa73b	3c6ef372fe94f82b
	58cb02347ab51f91	510e527fade682d1	9b05688c2b3e6c1f	1f83d9abfb41bd6b
t = 1 :	1320f8c9fb872cc0	f6afceb8bcfcddf5	6a09e667f3bcc908	bb67ae8584caa73b
	c3d4ebfd48650ffa	58cb02347ab51f91	510e527fade682d1	9b05688c2b3e6c1f
t = 2 :	ebcffc07203d91f3	1320f8c9fb872cc0	f6afceb8bcfcddf5	6a09e667f3bcc908
	dfa9b239f2697812	c3d4ebfd48650ffa	58cb02347ab51f91	510e527fade682d1
t = 3 :	5a83cb3e80050e82	ebcffc07203d91f3	1320f8c9fb872cc0	f6afceb8bcfcddf5
	0b47b4bb1928990e	dfa9b239f2697812	c3d4ebfd48650ffa	58cb02347ab51f91
t = 4 :	b680953951604860	5a83cb3e80050e82	ebcffc07203d91f3	1320f8c9fb872cc0
	745aca4a342ed2e2	0b47b4bb1928990e	dfa9b239f2697812	c3d4ebfd48650ffa
t = 5 :	af573b02403e89cd	b680953951604860	5a83cb3e80050e82	ebcffc07203d91f3
	96f60209b6dc35ba	745aca4a342ed2e2	0b47b4bb1928990e	dfa9b239f2697812
t = 6 :	c4875b0c7abc076b	af573b02403e89cd	b680953951604860	5a83cb3e80050e82
	5a6c781f54dcc00c	96f60209b6dc35ba	745aca4a342ed2e2	0b47b4bb1928990e
t = 7 :	8093d195e0054fa3	c4875b0c7abc076b	af573b02403e89cd	b680953951604860
	86f67263a0f0ec0a	5a6c781f54dcc00c	96f60209b6dc35ba	745aca4a342ed2e2
t = 8 :	f1eca5544cb89225	8093d195e0054fa3	c4875b0c7abc076b	af573b02403e89cd

d0403c398fc40002 86f67263a0f0ec0a 5a6c781f54dcc00c 96f60209b6dc35ba

t = 9 : 81782d4a5db48f03 f1eca5544cb89225 8093d195e0054fa3 c4875b0c7abc076b

00091f460be46c52 d0403c398fc40002 86f67263a0f0ec0a 5a6c781f54dcc00c

t = 10 : 69854c4aa0f25b59 81782d4a5db48f03 f1eca5544cb89225 8093d195e0054fa3

d375471bde1ba3f4 00091f460be46c52 d0403c398fc40002 86f67263a0f0ec0a

t = 11 : db0a9963f80c2eaa 69854c4aa0f25b59 81782d4a5db48f03 f1eca5544cb89225

475975b91a7a462c d375471bde1ba3f4 00091f460be46c52 d0403c398fc40002

t = 12 : 5e41214388186c14 db0a9963f80c2eaa 69854c4aa0f25b59 81782d4a5db48f03

cdf3bff2883fc9d9 475975b91a7a462c d375471bde1ba3f4 00091f460be46c52

t = 13 : 44249631255d2ca0 5e41214388186c14 db0a9963f80c2eaa 69854c4aa0f25b59

860acf9effba6f61 cdf3bff2883fc9d9 475975b91a7a462c d375471bde1ba3f4

t = 14 : fa967eed85a08028 44249631255d2ca0 5e41214388186c14 db0a9963f80c2eaa

874bfe5f6aae9f2f 860acf9effba6f61 cdf3bff2883fc9d9 475975b91a7a462c

t = 15 : 0ae07c86b1181c75 fa967eed85a08028 44249631255d2ca0 5e41214388186c14

a77b7c035dd4c161 874bfe5f6aae9f2f 860acf9effba6f61 cdf3bff2883fc9d9

t = 16 : caf81a425d800537 0ae07c86b1181c75 fa967eed85a08028 44249631255d2ca0

2deecc6b39d64d78 a77b7c035dd4c161 874bfe5f6aae9f2f 860acf9effba6f61

t = 17 : 4725be249ad19e6b caf81a425d800537 0ae07c86b1181c75 fa967eed85a08028

f47e8353f8047455 2deecc6b39d64d78 a77b7c035dd4c161 874bfe5f6aae9f2f

t = 18 : 3c4b4104168e3edb 4725be249ad19e6b caf81a425d800537 0ae07c86b1181c75

29695fd88d81dbd0 f47e8353f8047455 2deecc6b39d64d78 a77b7c035dd4c161

t = 19 : 9a3fb4d38ab6cf06 3c4b4104168e3edb 4725be249ad19e6b caf81a425d800537

f14998dd5f70767e 29695fd88d81dbd0 f47e8353f8047455 2deecc6b39d64d78

t = 20 : 8dc5ae65569d3855 9a3fb4d38ab6cf06 3c4b4104168e3edb 4725be249ad19e6b

4bb9e66d1145bfdc f14998dd5f70767e 29695fd88d81dbd0 f47e8353f8047455

t = 21 : da34d6673d452dcf 8dc5ae65569d3855 9a3fb4d38ab6cf06 3c4b4104168e3edb

8e30ff09ad488753 4bb9e66d1145bfdc f14998dd5f70767e 29695fd88d81dbd0

t = 22 : 3e2644567b709a78 da34d6673d452dcf 8dc5ae65569d3855 9a3fb4d38ab6cf06

0ac2b11da8f571c6 8e30ff09ad488753 4bb9e66d1145bfdc f14998dd5f70767e

t = 23 : 4f6877b58fe55484 3e2644567b709a78 da34d6673d452dcf 8dc5ae65569d3855

c66005f87db55233 0ac2b11da8f571c6 8e30ff09ad488753 4bb9e66d1145bfdc

t = 24 : 9aff71163fa3a940 4f6877b58fe55484 3e2644567b709a78 da34d6673d452dcf

d3ecf13769180e6f c66005f87db55233 0ac2b11da8f571c6 8e30ff09ad488753

t = 25 : 0bc5f791f8e6816b 9aff71163fa3a940 4f6877b58fe55484 3e2644567b709a78

6ddf1fd7edcce336 d3ecf13769180e6f c66005f87db55233 0ac2b11da8f571c6

t = 26 : 884c3bc27bc4f941 0bc5f791f8e6816b 9aff71163fa3a940 4f6877b58fe55484

e6e48c9a8e948365 6ddf1fd7edcce336 d3ecf13769180e6f c66005f87db55233

t = 27 : eab4a9e5771b8d09 884c3bc27bc4f941 0bc5f791f8e6816b 9aff71163fa3a940

09068a4e255a0dac e6e48c9a8e948365 6ddf1fd7edcce336 d3ecf13769180e6f

t = 28 : e62349090f47d30a eab4a9e5771b8d09 884c3bc27bc4f941 0bc5f791f8e6816b

0fcdf99710f21584 09068a4e255a0dac e6e48c9a8e948365 6ddf1fd7edcce336

t = 29 : 74bf40f869094c63 e62349090f47d30a eab4a9e5771b8d09 884c3bc27bc4f941

f0aec2fe1437f085 0fcdf99710f21584 09068a4e255a0dac e6e48c9a8e948365

t = 30 : 4c4fbbb75f1873a6 74bf40f869094c63 e62349090f47d30a eab4a9e5771b8d09

73e025d91b9efea3 f0aec2fe1437f085 0fcdf99710f21584 09068a4e255a0dac

t = 31 : ff4d3f1f0d46a736 4c4fbbb75f1873a6 74bf40f869094c63 e62349090f47d30a

3cd388e119e8162e 73e025d91b9efea3 f0aec2fe1437f085 0fcdf99710f21584

t = 32 : a0509015ca08c8d4 ff4d3f1f0d46a736 4c4fbbb75f1873a6 74bf40f869094c63

e1034573654a106f 3cd388e119e8162e 73e025d91b9efea3 f0aec2fe1437f085

t = 33 : 60d4e6995ed91fe6 a0509015ca08c8d4 ff4d3f1f0d46a736 4c4fbbb75f1873a6

efabbd8bf47c041a e1034573654a106f 3cd388e119e8162e 73e025d91b9efea3

t = 34 : 2c59ec7743632621 60d4e6995ed91fe6 a0509015ca08c8d4 ff4d3f1f0d46a736

0fbae670fa780fd3 efabbd8bf47c041a e1034573654a106f 3cd388e119e8162e

t = 35 : 1a081afc59fdbc2c 2c59ec7743632621 60d4e6995ed91fe6 a0509015ca08c8d4

f098082f502b44cd 0fbae670fa780fd3 efabbd8bf47c041a e1034573654a106f

t = 36 : 88df85b0bbe77514 1a081afc59fdbc2c 2c59ec7743632621 60d4e6995ed91fe6

8fbfd0162bbf4675 f098082f502b44cd 0fbae670fa780fd3 efabbd8bf47c041a

t = 37 : 002bb8e4cd989567 88df85b0bbe77514 1a081afc59fdbc2c 2c59ec7743632621

66adcfa249ac7bbd 8fbfd0162bbf4675 f098082f502b44cd 0fbae670fa780fd3

t = 38 : b3bb8542b3376de5 002bb8e4cd989567 88df85b0bbe77514 1a081afc59fdbc2c

b49596c20feba7de 66adcfa249ac7bbd 8fbfd0162bbf4675 f098082f502b44cd

t = 39 : 8e01e125b855d225 b3bb8542b3376de5 002bb8e4cd989567 88df85b0bbe77514

0c710a47ba6a567b b49596c20feba7de 66adcfa249ac7bbd 8fbfd0162bbf4675

t = 40 : b01521dd6a6be12c 8e01e125b855d225 b3bb8542b3376de5 002bb8e4cd989567

169008b3a4bb170b 0c710a47ba6a567b b49596c20feba7de 66adcfa249ac7bbd

t = 41 : e96f89dd48cbd851 b01521dd6a6be12c 8e01e125b855d225 b3bb8542b3376de5

f0996439e7b50cb1 169008b3a4bb170b 0c710a47ba6a567b b49596c20feba7de

t = 42 : bc05ba8de5d3c480 e96f89dd48cbd851 b01521dd6a6be12c 8e01e125b855d225

639cb938e14dc190 f0996439e7b50cb1 169008b3a4bb170b 0c710a47ba6a567b

t = 43 : 35d7e7f41defcbd5 bc05ba8de5d3c480 e96f89dd48cbd851 b01521dd6a6be12c

cc5100997f5710f2 639cb938e14dc190 f0996439e7b50cb1 169008b3a4bb170b

t = 44 : c47c9d5c7ea8a234 35d7e7f41defcbd5 bc05ba8de5d3c480 e96f89dd48cbd851

858d832ae0e8911c cc5100997f5710f2 639cb938e14dc190 f0996439e7b50cb1

t = 45 : 021fbadbabab5ac6 c47c9d5c7ea8a234 35d7e7f41defcbd5 bc05ba8de5d3c480

e95c2a57572d64d9 858d832ae0e8911c cc5100997f5710f2 639cb938e14dc190

t = 46 : f61e672694de2d67 021fbadbabab5ac6 c47c9d5c7ea8a234 35d7e7f41defcbd5

c6bc35740d8daa9a e95c2a57572d64d9 858d832ae0e8911c cc5100997f5710f2

t = 47 : 6b69fc1bb482feac f61e672694de2d67 021fbadbabab5ac6 c47c9d5c7ea8a234

35264334c03ac8ad c6bc35740d8daa9a e95c2a57572d64d9 858d832ae0e8911c

t = 48 : 571f323d96b3a047 6b69fc1bb482feac f61e672694de2d67 021fbadbabab5ac6

271580ed6c3e5650 35264334c03ac8ad c6bc35740d8daa9a e95c2a57572d64d9

t = 49 : ca9bd862c5050918 571f323d96b3a047 6b69fc1bb482feac f61e672694de2d67

dfe091dab182e645 271580ed6c3e5650 35264334c03ac8ad c6bc35740d8daa9a

t = 50 : 813a43dd2c502043 ca9bd862c5050918 571f323d96b3a047 6b69fc1bb482feac

07a0d8ef821c5e1a dfe091dab182e645 271580ed6c3e5650 35264334c03ac8ad

t = 51 : d43f83727325dd77 813a43dd2c502043 ca9bd862c5050918 571f323d96b3a047

483f80a82eaee23e 07a0d8ef821c5e1a dfe091dab182e645 271580ed6c3e5650

t = 52 : 03df11b32d42e203 d43f83727325dd77 813a43dd2c502043 ca9bd862c5050918

504f94e40591cffa 483f80a82eaee23e 07a0d8ef821c5e1a dfe091dab182e645

t = 53 : d63f68037ddf06aa 03df11b32d42e203 d43f83727325dd77 813a43dd2c502043

a6781efe1aa1ce02 504f94e40591cffa 483f80a82eaee23e 07a0d8ef821c5e1a

t = 54 : f650857b5babda4d d63f68037ddf06aa 03df11b32d42e203 d43f83727325dd77

9ccfb31a86df0f86 a6781efe1aa1ce02 504f94e40591cffa 483f80a82eaee23e

t = 55 : 63b460e42748817e f650857b5babda4d d63f68037ddf06aa 03df11b32d42e203

c6b4dd2a9931c509 9ccfb31a86df0f86 a6781efe1aa1ce02 504f94e40591cffa

t = 56 : 7a52912943d52b05 63b460e42748817e f650857b5babda4d d63f68037ddf06aa

d2e89bbd91e00be0 c6b4dd2a9931c509 9ccfb31a86df0f86 a6781efe1aa1ce02

t = 57 : 4b81c3aec976ea4b 7a52912943d52b05 63b460e42748817e f650857b5babda4d

70505988124351ac d2e89bbd91e00be0 c6b4dd2a9931c509 9ccfb31a86df0f86

t = 58 : 581ecb3355dcd9b8 4b81c3aec976ea4b 7a52912943d52b05 63b460e42748817e

6a3c9b0f71c8bf36 70505988124351ac d2e89bbd91e00be0 c6b4dd2a9931c509

t = 59 : 2c074484ef1eac8c 581ecb3355dcd9b8 4b81c3aec976ea4b 7a52912943d52b05

4797cde4ed370692 6a3c9b0f71c8bf36 70505988124351ac d2e89bbd91e00be0

t = 60 : 3857dfd2fc37d3ba 2c074484ef1eac8c 581ecb3355dcd9b8 4b81c3aec976ea4b

a6af4e9c9f807e51 4797cde4ed370692 6a3c9b0f71c8bf36 70505988124351ac

t = 61 : cfcd928c5424e2b6 3857dfd2fc37d3ba 2c074484ef1eac8c 581ecb3355dcd9b8

09aee5bda1644de5 a6af4e9c9f807e51 4797cde4ed370692 6a3c9b0f71c8bf36

t = 62 : a81dedbb9f19e643 cfcd928c5424e2b6 3857dfd2fc37d3ba 2c074484ef1eac8c

84058865d60a05fa 09aee5bda1644de5 a6af4e9c9f807e51 4797cde4ed370692

t = 63 : ab44e86276478d85 a81dedbb9f19e643 cfcd928c5424e2b6 3857dfd2fc37d3ba

cd881ee59ca6bc53 84058865d60a05fa 09aee5bda1644de5 a6af4e9c9f807e51

t = 64 : 5a806d7e9821a501 ab44e86276478d85 a81dedbb9f19e643 cfcd928c5424e2b6

aa84b086688a5c45 cd881ee59ca6bc53 84058865d60a05fa 09aee5bda1644de5

t = 65 : eeb9c21bb0102598 5a806d7e9821a501 ab44e86276478d85 a81dedbb9f19e643

3b5fed0d6a1f96e1 aa84b086688a5c45 cd881ee59ca6bc53 84058865d60a05fa

t = 66 : 46c4210ab2cc155d eeb9c21bb0102598 5a806d7e9821a501 ab44e86276478d85

29fab5a7bff53366 3b5fed0d6a1f96e1 aa84b086688a5c45 cd881ee59ca6bc53

t = 67 : 54ba35cf56a0340e 46c4210ab2cc155d eeb9c21bb0102598 5a806d7e9821a501

1c66f46d95690bcf 29fab5a7bff53366 3b5fed0d6a1f96e1 aa84b086688a5c45

t = 68 : 181839d609c79748 54ba35cf56a0340e 46c4210ab2cc155d eeb9c21bb0102598

0ada78ba2d446140 1c66f46d95690bcf 29fab5a7bff53366 3b5fed0d6a1f96e1

t = 69 : fb6aaae5d0b6a447 181839d609c79748 54ba35cf56a0340e 46c4210ab2cc155d

e3711cb6564d112d 0ada78ba2d446140 1c66f46d95690bcf 29fab5a7bff53366

t = 70 : 7652c579cb60f19c fb6aaae5d0b6a447 181839d609c79748 54ba35cf56a0340e

aff62c9665ff80fa e3711cb6564d112d 0ada78ba2d446140 1c66f46d95690bcf

t = 71 : f15e9664b2803575 7652c579cb60f19c fb6aaae5d0b6a447 181839d609c79748

947c3dfafee570ef aff62c9665ff80fa e3711cb6564d112d 0ada78ba2d446140

t = 72 : 358406d165aee9ab f15e9664b2803575 7652c579cb60f19c fb6aaae5d0b6a447

8c7b5fd91a794ca0 947c3dfafee570ef aff62c9665ff80fa e3711cb6564d112d

t = 73 : 20878dcd29cdfaf5 358406d165aee9ab f15e9664b2803575 7652c579cb60f19c

054d3536539948d0 8c7b5fd91a794ca0 947c3dfafee570ef aff62c9665ff80fa

t = 74 : 33d48dabb5521de2 20878dcd29cdfaf5 358406d165aee9ab f15e9664b2803575

2ba18245b50de4cf 054d3536539948d0 8c7b5fd91a794ca0 947c3dfafee570ef

t = 75 : c8960e6be864b916 33d48dabb5521de2 20878dcd29cdfaf5 358406d165aee9ab

995019a6ff3ba3de 2ba18245b50de4cf 054d3536539948d0 8c7b5fd91a794ca0

t = 76 : 654ef9abec389ca9 c8960e6be864b916 33d48dabb5521de2 20878dcd29cdfaf5

ceb9fc3691ce8326 995019a6ff3ba3de 2ba18245b50de4cf 054d3536539948d0

t = 77 : d67806db8b148677 654ef9abec389ca9 c8960e6be864b916 33d48dabb5521de2

25c96a7768fb2aa3 ceb9fc3691ce8326 995019a6ff3ba3de 2ba18245b50de4cf

t = 78 : 10d9c4c4295599f6 d67806db8b148677 654ef9abec389ca9 c8960e6be864b916

9bb4d39778c07f9e 25c96a7768fb2aa3 ceb9fc3691ce8326 995019a6ff3ba3de

t = 79 : 73a54f399fa4b1b2 10d9c4c4295599f6 d67806db8b148677 654ef9abec389ca9

d08446aa79693ed7 9bb4d39778c07f9e 25c96a7768fb2aa3 ceb9fc3691ce8326

şeklinde hesaplanır. Son adımın ardından H_0, H_1, H_2, H_3, H_4, H_5, H_6, H_7 değişkenleri;

$H_0=H_0+A=$ 6A09E667F3BCC908 + 73A54F399FA4B1B2 = **DDAF35A193617ABA**,
$H_1=H_1+B=$ BB67AE8584CAA73B + 10D9C4C4295599F6 = **CC417349AE204131**,
$H_2=H_2+C=$ 3C6EF372FE94F82B + D67806DB8B148677 = **12E6FA4E89A97EA2**,
$H_3=H_3+D=$ A54FF53A5F1D36F1 + 654EF9ABEC389CA9 = **0A9EEEE64B55D39A**,
$H_4=H_4+E=$ 510E527FADE682D1 + D08446AA79693ED7 = **2192992A274FC1A8**,
$H_5=H_5+F=$ 9B05688C2B3E6C1F + 9BB4D39778C07F9E = **36BA3C23A3FEEBBD**,
$H_6=H_6+G=$ 1F83D9ABFB41BD6B + 25C96A7768FB2AA3 = **454D4423643CE80E**,
$H_7=H_7+F=$ 5BE0CD19137E2179 + CEB9FC3691CE8326 = **2A9AC94FA54CA49F**
olarak elde edilir. 512 bit özet değer:

DDAF35A193617ABACC417349AE20413112E6FA4E89A
97EA20A9EEEE64B55D39A2192992A274FC1A836BA3C
23A3FEEBBD454D4423643CE80E2A9AC94FA54CA49F
olarak elde edilmiş olur.

3.1.3.4. SHA-224 Algoritması

SHA-224 algoritması SHA-256 ile aynı özelliklere sahiptir. Aralarında iki fark vardır;

- $H_0, H_1, H_2, H_3, H_4, H_5, H_6, H_7$ değişkenlerinin başlangıç değerleri farklıdır.
- SHA-224' te özet değer; sonuçta oluşan 256 bit çıktı değerinin sol 224 bitidir.

3.1.3.5. SHA-384 Algoritması

SHA-384 algoritması SHA-512 ile aynı özelliklere sahiptir. Aralarında iki fark vardır;

- $H_0, H_1, H_2, H_3, H_4, H_5, H_6, H_7$ değişkenlerinin başlangıç değerleri farklıdır.
- SHA-384' te özet değer; sonuçta oluşan 512 bit çıktı değerinin sol 384 bitidir.

3.1.4. SHA-3 Algoritması

SHA-1 algoritmasındaki güvenlik zaaflarından sonra, temel yapısı aynı olduğundan dolayı SHA-2 algoritmasınında güvenlik sorunu oluşturabileceğini düşünen NIST, 2007 yılının Kasım ayında yeni bir özet algoritması için uluslar arası bir yarışma açtığını duyurdu. Kasım 2007' de başlayan yarışma 31 Ekim 2009' da sona erdi. Toplam 64 tane yeni geliştirilmiş algoritma sunuldu. 13 tanesi kurallara uygun bulunmadı ve yarışma 52 algoritma ile devam etti[40].

23-29 Şubat 2010 tarihinde birinci SHA-3 Adayları Konferansında sunumlar gerçekleştirildi. Bu konferansta çakışma bulunma durumları veya tam kırılmasa bile güvenli olmadığı için geri çekilen algoritmalar ile birlikte ikici turda 14 algoritma kalmıştır[40]. İkinci konferansta olmasa bile üçüncü konferansta adayların sayısı 5 veya 6' ya inecektir ve sonunda SHA-3 algoritması kullanıma sunulacaktır. SHA-1 ve SHA-2 algoritmalarında olmasa da SHA-3 algoritmasında, yarışmacılar arasından Çetin KAYA KOÇ **'Spectral Hash'** algoritmasıyla, Orhun KARA **'Shamata'** algoritmasıyla, Kerem VARICI **'Sarmal'** algoritmasıyla ve Özgül KÜÇÜK **'Hamsi'** algoritmasıyla ilk 51

algoritma arasına girmişlerdir. İkinci tura ise sadece, **'Hamsi'** adlı algoritmasıyla Özgül KÜÇÜK kalmıştır.

3.2. Anahtarlı Hash Fonksiyonları (MAC)

Anahtarlı hash fonksiyonlarının amacı mesajın bütünlüğünü garantilemek yani mesajın doğru mesaj olduğunu kanıtlamaktır. Bu fonksiyonlar MAC (Message Authentication Code-Mesaj Doğrulama Kodu) algoritmaları olarak isimlendirilirler.

Anahtarsız hash fonksiyonlarının fazlalığına karşı çok fazla MAC algoritması bulunmamaktadır. MAC algoritmaları sayısal imzadan daha zayıf güvenlik sağlarlar. Bu nedenle tarafların birbirine güvendiği simetrik ortamlarda tercih edilmelidir. Gönderici ve alıcı MAC algoritmasını kullanmak için önceden bir ***K*** anahtarını paylaşırlar. ***K*** anahtarı k bit rastgele bir veri olup boyutu genelde 56-128 bit arasıdır. Gönderici mesajı korumak için m-bit uzunluklu verinin MAC değerini hesaplar ve bu veriyi mesaja ekler. MAC, mesajın ve anahtarın her bitinin karmaşık bir fonksiyonudur. Alıcı aldığı veri üzerinde MAC' i tekrar hesaplar ve kendisine gelen MAC değerine eşit olup olmadığını kontrol eder. Böylece mesajı doğrulamış olur[41].

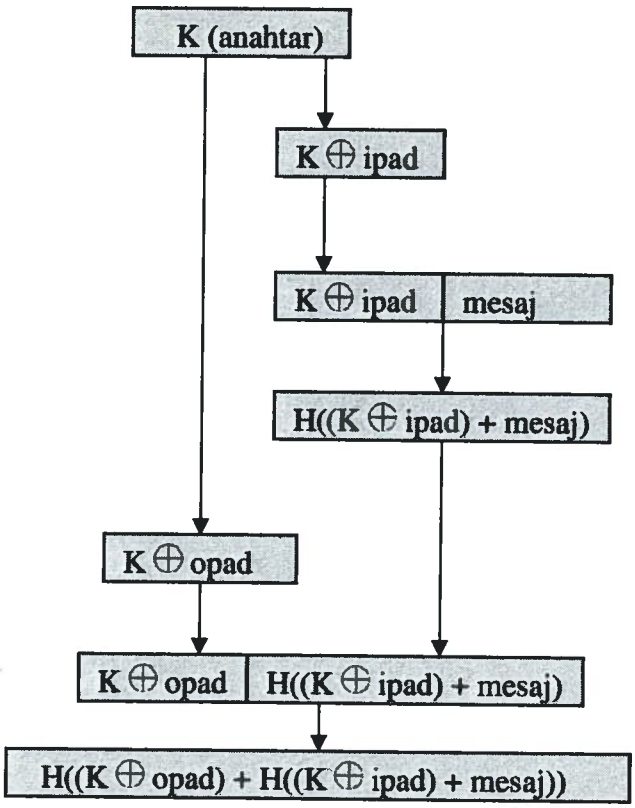
3.2.1. HASH' a Dayalı MAC (HMAC)

Güvenilmeyen bir ortamda iletilen veya depolanan bilginin bütünlüğünü kontrol etmek için bir yol sağlanması bilgisayar ve iletişim dünyasında bir zorunluluktur. Gizli bir anahtara dayalı böyle bütünlük kontrolünü sağlayan mekanizmalar, genellikle MAC olarak adlandırılır. Genellikle mesaj doğrulama kodları taraflar arasında iletilen bilgiyi doğrulamak için gizli bir anahtarı paylaşan iki taraf arasında kullanılır. Şifrelemede HMAC (Hash Based Message Authentication Code-Hash tabanlı mesaj doğrulama kodu) ise gizli bir şifre ile birlikte kriptografik bir hash fonksiyonunu kapsayan mesaj doğrulama kodunu hesaplamak için özel bir yapıdır. Tek kullanımlık şifre üretim algoritmalarının büyük çoğunluğunda HMAC kullanılır. HMAC herhangi bir MAC gibi, hem veri bütünlüğünü hem de mesajın güvenilirliğini doğrulamak için kullanılabilir. HMAC' in hesaplanmasında MD5 veya SHA-1, SHA-2 gibi herhangi bir kriptografik hash fonksiyonu kullanılabilir. Çıkan MAC algoritmasına göre buna HMAC-MD5, HMAC-SHA-1 veya HMAC-SHA-2

denilir ve en fazla kullanılan algoritmalar bunlardır. HMAC' in şifreleme gücü; kullandığı hash algoritmasının gücüne ve bit cinsinden hash çıktısının uzunluğuna bağlıdır[42].

3.2.1.1. HMAC' in Tanımı

HMAC bir gizli anahtar **K** ve **H** ile gösterilen bir kriptografik hash fonksiyonu gerektirir. HMAC' in çalışma şeması Şekil 3.7' de görüldüğü gibidir.



Şekil 3.7 HMAC çalışma şeması

HMAC' in formülü;

HMAC (M,K) = H [(K ⊕ opad) + H [(K ⊕ ipad) + M]] şeklindedir[42].

Burada;

M = Mesaj,

H[ ] = Kullanılan hash fonksiyonu,

K = Paylaşılan gizli anahtar,

opad= 0x5C' lerden oluşan veri dizisi,

ipad= 0x36' lardan oluşan veri dizisi,

$\oplus$ = XOR işlemi,

+ ise String birleştirme işlemini ifade eder.

4. TEK KULLANIMLIK ŞİFRE ÜRETİMİ

4.1. OTP Tek Kullanımlık Şifre

Güvenilir olmayan iletişim ortamlarında, taraflar arasındaki veri alış verişine yapılması muhtemel saldırılardan biri de ortamın saldırgan tarafından dinlenilmesi ve kullanıcı adı ve parola bilgilerinin saldırgan tarafından elde edilmesidir. Bu bilgiler elde edildiği takdirde saldırgan sonraki zamanlarda sisteme rahat bir şekilde sızabilecektir. İşte tek kullanımlık şifre OTP burada devreye girer. Haberleşme hattının dinlenmesi sonucu bazı bilgiler saldırganın eline geçmişse dahi kullanılan şifre bir defaya mahsus olduğu için saldırganın sonraki sızma girişimleri başarısız olacaktır[36].

OTP iki farklı yöntemle uygulanmaktadır. Zaman senkronizasyonlu OTP ve sayaç kullanan OTP. Zamana dayalı modelde kullanıcının saati server saatiyle aynı olmalıdır. Sayaca bağlı modelde ise her iki tarafın sayaçları eşit olmalıdır. Zamana bağlı şifre üretiminde senkronizasyon sorunu olduğu için çok fazla tercih edilmezler. Sayaca dayalı OTP sisteminde sayaç, en son doğru şifreye göre otomatik artırılır. Sayacın başlangıç değeri ve paylaşılan bir gizli anahtar her iki tarafça bilinir. Daha sonra girilen şifre sayacın n kadar artırılmasıyla yeni sayaca göre kontrol edilir. Sayaca göre üretilen şifre yanlışsa sayacın değeri x kadar artırılarak tekrar denenir (Burada her iki taraf **n** ve **x** değerleri üzerinde önceden anlaşmışlardır). Eğer bu denemeler de olumsuz sonuçlanırsa şifre yanlış kabul edilir veya kullanıcı şifrenin doğru olduğundan emin ise sayaçların senkronize edilmesi gerekir. Sisteme her başarılı girişten sonra sayaç, kullanıcı ve server tarafından güncellenir.

Hash tabanlı tek kullanımlık şifreleme sistemlerine HOTP (HMAC-based One Time Password) adı verilmektedir. HOTP için HMAC fonksiyonunun gerçeklenmesi gerekmektedir. HMAC bir nevi karıştırma yapan bir algoritmadır. Temelinde ise çoğu zaman SHA algoritmaları kullanılmaktadır.

K; Gizli anahtar,

C; Sayaç,

Truncate; HMAC sonucunun 8 byte' ı,

M; Şifrelenecek mesaj,

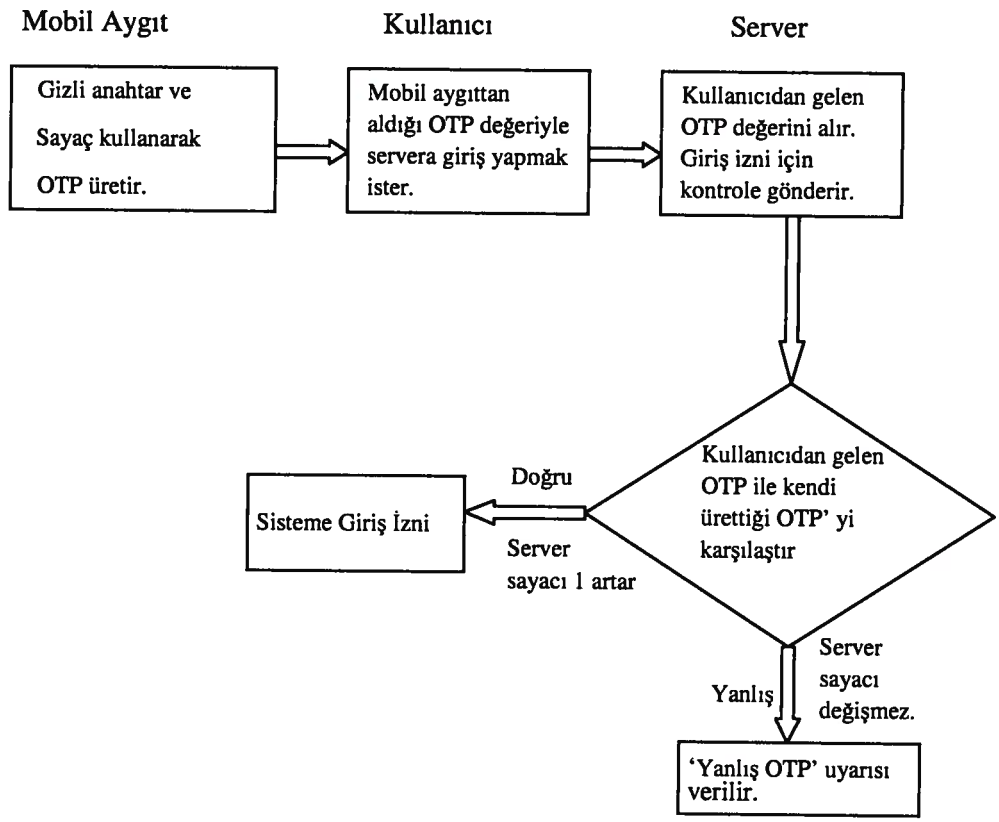
HMAC (K,M)= H [(K $\oplus$ opad) + H [(K $\oplus$ ipad) + M]],

olmak üzere HOTP değeri;

HOTP(K,C) = Truncate(HMAC(K,C)) & 0x7FFFFFFF

şeklinde elde edilir[45].

Truncate işlemi ile HMAC sonucundan alınan 8 byte ve 0x7FFFFFFF ile lojik AND işlemine tabi tutularak (MSB biti bu şekilde sıfırlanır, sonucun negatif çıkmaması için) HOTP elde edilir. OTP şifre sistemlerinin genel şeması Şekil 4.1' de görülmektedir.



Şekil 4.1 OTP şifre sisteminin genel yapısı

5. GELİŞTİRİLEN YAZILIM

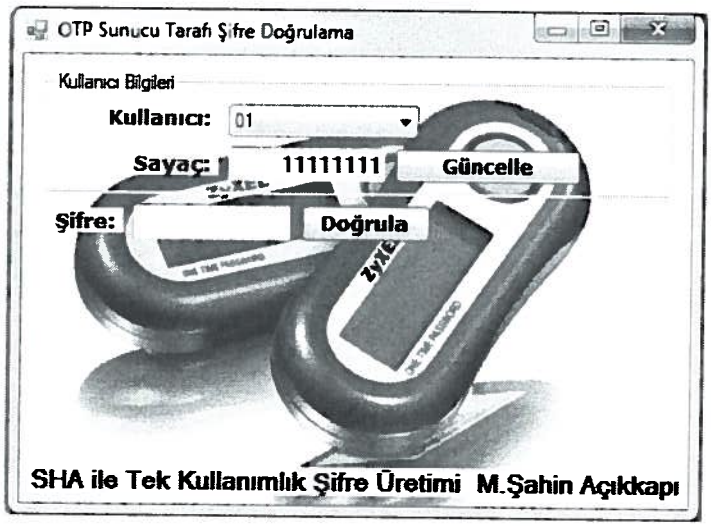
5.1. Sunucu İçin Geliştirilen Yazılım

Oluşturulan OTP yazılım sisteminde sunucu tarafında Microsoft .NET Framework 4.0 sürümü ve bu sürüm üzerinde çalışan C# programlama dili kullanılmıştır. Kullanıcı ve server, öncelikli olarak bir gizli anahtar değerini ve bir sayaç değerini paylaşırlar. Ardından her iki taraf bu anahtar ve sayaç değerleri üzerinden HOTP değerini ;

HOTP(K,C) = Truncate(HMAC(K,C)) & 0x7FFFFFFF şeklinde hesaplar.

HMAC değeri;

HMAC(K,M) = H [(K ⊕ opad) + H [(K ⊕ ipad) + M]] formülü üzerinden hesaplanır. Burada **K** sayaç değeri, **M** paylaşılan gizli anahtar değeri ve **H** SHA-512 fonksiyonudur. SHA-512 fonksiyonu için server tarafında Microsoft .NET Framework' de bulunan ***System.Security.Cryptography*** kütüphanesinin SHA-512 değeri hesaplayan metotları kullanılırken, mobil aygıtta Java' nın ***java.security.MessageDigest*** kütüphanesinden faydalanılmıştır. Sunucuda çalışacak olan programın ekran görüntüsü Şekil 5.1' de görülmektedir.



Şekil 5.1 Sunucuda çalışan doğrulama programı

5.2. Mobil Aygıt Yazılımı

Kullanıcı tarafında programlama dili olarak, mobil aygıtlarda en fazla kullanılan program olan Java programlama dili kullanılmıştır. Geliştirme ortamı olarak Netbeans geliştirme ortamının 6.9.1 sürümü kullanılmış, mobil aygıt için yazılım geliştirilirken NetBeans' ın kendi içerisinde bulunan simulatör kullanılmıştır. Ayrıca yazılan program Nokia 3600 cep telefonunda test edilip, başarılı sonuçlar alınmıştır. Mobil aygıt simulatörü Şekil 5.2' de, Nokia 3600 cep telefonu ise Şekil 5.3' te görülmektedir.

Şekil 5.2 Mobil aygıt simulatörü

Şekil 5.3 Nokia 3600' da programın çalışma görüntüsü

5.3. OTP Şifre Programının Çalıştırılması

Kullanıcı sisteme giriş yapmak istediğinde telefonundaki OTP programını çalıştırır ve Şekil 5.4' te görüldüğü gibi tek kullanımlık şifreyi üretir. Bu işlemden sonra telefondaki sayaç 1 artırılır.

Şekil 5.4 Mobil aygıtta şifre üretimi

Kullanıcı aldığı geçici şifre ile servera ulaşır ve şifre doğru ise Şekil 5.5' te görüldüğü gibi ***'şifre doğrulandı'*** mesajı alınır ve sisteme güvenli giriş yapılmış olur. Doğrulama işlemi gerçekleşince serverdaki sayaç 1 artırılır.

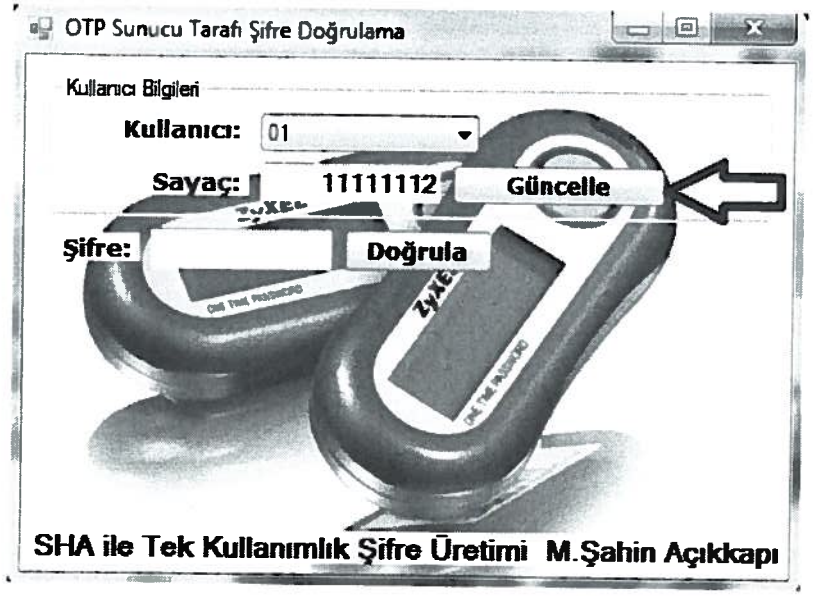
Şekil 5.5 OTP şifresinin kabul aşaması

Şifre yanlış ise Şekil 5.6' da olduğu gibi ***'şifre doğrulanamadı'*** mesajı alınır. Serverdaki sayaç değişmez.

Şekil 5.6 OTP şifresinin red aşaması

5.4. OTP Sayaç Güncelleme

OTP sistemlerinde server ve kullanıcı öncelikli olarak bir gizli anahtar ve sayaç değerini paylaşırlar. Bu gizli anahtar değeri sabit kalmakla beraber sayaç değeri ise mobil aygıtta her şifre üretiminden sonra, server tarafında ise her doğru şifre girişinden sonra 1 artar. Mobil aygıtta şifre üretildiği halde servera giriş yapılmazsa iki tarafın sayaç değerleri farklı olacağı için sisteme girişlerde sorun yaşanacaktır. Bu sorun şu şekilde giderilmeye çalışılmaktadır; Server şifre doğrulama aşamasında şifre doğrulanamazsa ileriye doğru sayacı artırarak doğrulama işlemini tekrar dener. Ancak bu sayacı ileri doğru artırma işlemi belirli bir sayıya kadar olmak zorundadır. Tezde önerilen yazılım sisteminde bu sayı 5' tir. Yani server, sayacı ileriye doğru 5 defa ilerleterek şifre doğrulama işlemini gerçekleştirmeye çalışır. Şayet bu şekildede doğrulama işlemi gerçekleştirilemezse (şifre doğru olduğu halde) her iki tarafın sayaç değerlerinin aynı değere güncellenmesi gerekir. Tezde önerilen programda server ve mobil aygıtın sayaç güncelleme işlemleri, ilgili programlarda sayaç alanına sayaç değeri yazıldıktan sonra **'sayaç güncelle'** butonuyla sağlanır. Sayaç güncelleme işlemlerinin ekran görüntüleri Şekil 5.7 ve Şekil 5.8' de görülmektedir.



Şekil 5.7 Server programında sayaç güncelleme

Şekil 5.8 Mobil aygıtta sayaç güncelleme

5.5. Geliştirilen Yazılımın Güvenliği

Tek kullanımlık şifre algoritmalarının temelinde hash fonksiyonları olduğundan güvenlikleri doğrudan kullandıkları hash fonksiyonlarının güvenliğiyle orantılıdır. Bu sebeple hash fonksiyonlarının güvenlik analizi büyük önem taşımaktadır. Hash

fonksiyonlarına yapılan saldırılar, saldırıda kullanılan temel parametrelerin çeşitlerine göre sınıflandırılırlar. Örnek verilecek olursa bazı saldırılar sadece özet değer uzunluğuna bağlı iken diğer saldırılar, özet fonksiyonunun sıkıştırma fonksiyonlarına bağlıdırlar. Bu noktada hash fonksiyonuna yapılan saldırılar; **genel** saldırılar ve **spesifik** saldırılar olarak kategorize edilebilir.

Genel saldırılar çoğunlukla tüm hash fonksiyonlarına uygulanabilirken, spesifik saldırılar özel hash fonksiyonları için kriptoanaliz metotlarıdırlar ve sınırlı hash fonksiyonlarına uygulanabilirler. Tez kapsamında hash fonksiyonlarının güvenliği incelenirken daha çok genel saldırılar ve bu saldırılar içerisinde en fazla kullanılan birthday saldırıları üzerinden fonksiyonların güvenliklerine değinilmiştir. Birthday saldırıları detaylı açıklanacak olursa;

Birthday saldırısı tüm hash fonksiyonlarına uygulanabilen temel saldırı yöntemidir. Burada amaç **Çakışma (Collision)** bulmaktır. Çakışma, az sayıdaki insanların doğum günü problemlerine dayalıdır ve böylelikle aynı doğum gününe sahip olan en az iki kişinin olasılığı ½ dir. Problemi özet fonksiyonlarına uyarlamak için aynı doğum gününe sahip olan iki insan bir çakışma olarak adlandırılabilir. Doğum günü çakışmasındaki temel nokta, çakışma arayan kişi özellikli bir çakışmadan çok herhangi bir çakışmayı aramaktadır. Şayet, çakışma arayan kişi, **N** kişileri arasında özellikli bir çakışma arıyor olsaydı, en azından onunla aynı doğum gününe sahip olan bir kişi olduğu söylenebilir.

$$P_{collision}(N)=1-(364/365)^N$$

½ ' den daha büyük olan bu olasılıkta, **N** 254' den daha büyük olmalıdır. Bu düşünce özet fonksiyonuna uygulanırsa, çakışma arayan kişi, ikinci bir öngörüntü araştırması olan bütün olası mesajları sürekli inceleyerek özellikli bir mesajla karma bir değere sahip olan bir mesaj bulmaya çalışacaktır. Fakat, eğer çakışma arayan kişi, N doğum günleri arasında herhangi bir çakışma saptamaya çalışırsa, olasılık,

$$P_{collision}(N)=1-(365/365) \cdot (364/365) \cdot (363/365) \ldots(365-N+1/365)$$

olacaktır.

Burada **N'** i çözmek için, N=23 kişinin ½ den daha büyük olasılıklı doğum günlerine eşleştirmek yeterli olacaktır.

Özet fonksiyonu **H** için 'doğum günü durumu' çakışma olarak genellenebilir. Eşit olmayan X ve Y değerleri için H(X) = H(Y) olacak şekilde iki farklı eleman bulma olasılığı;

$$P_{collision}(t) = 1-1-e^{-t(t-1)/2x}$$

$$P_{collision}(t) = 1-(x/x) \cdot (x-1/x) \cdot (x-2/x) \ldots (x-t+1/x)$$

$$= 1- \prod_{i=1}^{t-1}(1-i/x)$$

$$=1- \prod_{i=1}^{t-1} e^{-i/x} \text{ since } 1-i/x \approx e^{-i/x} \text{ for } i<< x$$

$$=1-e^{-\sum_{i=1}^{t-1} i/x}$$

$$= 1-e^{-\frac{1}{x}\sum_{i=1}^{t-1} i}$$

$$= 1-e^{-\frac{t(t-1)}{2x}}$$

$$P_{collision}(t)= \frac{1}{2} = 1-e^{-\frac{t(t-1)}{2x}}$$

$$\frac{1}{2} = e^{-\frac{t(t-1)}{2x}}$$

$$\ln(\frac{1}{2}) = \ln(e^{-\frac{t(t-1)}{2x}})$$

$$-\ln(2) = -\frac{t(t-1)}{2x}$$

$$2x\ln(2) \approx t^2$$

$$\sqrt{2x\ln(2)} \approx t$$

$$1{,}17\sqrt{x} \approx t$$

Bu nedenle, eğer çakışma arayan kişi, *n* bit özet fonksiyonu için rastlantısal bir çatışma bulmak isterse, yaklaşık olarak $2^{n/2}$ mesajın karma değerini oluşturur. ½ olasılıklı bir çakışma oluşacaktır. $2^{n/2}$ ' den daha düşük bir olasılıkla çakışma bulunursa özet fonksiyonu kırılmış sayılacaktır[44].

Tek kullanımlık şifre üretiminde yoğun bir şekilde özet fonksiyonlar kullanılmaktadır ve güvenliği bu fonksiyonların güvenliğine bağlıdır. HMAC-SHA-1 ve HMAC-SHA-2 en fazla kullanılan algoritmalardır. Tez yazılımında geliştirilen OTP algoritmasında HMAC-SHA-2 algoritması kullanıldı. Bundan dolayı SHA-2 algoritmasının güvenliği önem taşımaktadır. Ancak SHA-2' nin güvenliğinin anlaşılması için SHA-1' in güvenlik durumuda büyük önem taşımaktadır.

5.5.1. SHA-1 Algoritmasının Güvenliği

Özetleme fonksiyonlarını kullanışlı yapan özelliklerden en önemlisi mesaj özetlerinin birbirinden farklı olmasıdır. Mesaj özetleri aynı olan iki farklı mesaj olması durumuna ***çakışma*** adı verilmektedir. Matematiksel olarak X elemanlı bir kümede, bir çakışma gerçekleşebilmesi için ortalama $\sqrt{x}$ tane rastgele örnekleme yapılması beklenmektedir. Bundan dolayı bir algoritma $2^{n/2}$ karmaşıklıktan daha az işlemle çakışma buluyorsa özet fonksiyonunu kırmış sayılmaktadır. 2005 yılında SHA-1' in indirgenmiş versiyonu üzerinde 2^{80} işlemden daha az işlem ile çakışma bulundu. Şubat 2005' te bir grup araştırmacı tarafından SHA-1' in tam sürümü üzerinde 2^{69} işlemden daha az işlem ile

çakışma bulan saldırılar duyuruldu. Ağustos 2005' te, CRYPTO konferansında, çakışma oluşturmak için gereken saldırı sayısının 2^{63}' e düşürüldüğü duyuruldu[45].

Genellikle sayısal imzalarda kullanılan bir algoritmanın bu derecede çakışmaya açık olması önemli bir güvenlik açığı sayılabilir. Saldırı işlem sayısı fazladır ancak günümüzde bunları gerçekleştirebilecek bilgisayar sistemleri bulunmaktadır. 160 bitlik çıktı üreten SHA-1 algoritmasının çakışmalara karşı 80 bitlik güvenlik sağlaması gerekir. Ancak SHA-1 için bu değer 39 bite kadar düşürülmüştür[46].

SHA-1' e yapılan başarılı saldırılardan dolayı daha güvenli olan SHA-2 algoritması kullanılmaya başlanmıştır.

5.5.2. SHA-2 Algoritmasının Güvenliği

SHA-2 ailesine karşı yapılan saldırılardan şimdiye kadar başarılı bir sonuç alınamamıştır. SHA-2 yüksek standartta güvenlik sağlamakta olup, sayısal imza, veri bütünlüğünü sağlama, OTP şifre üretimi gibi birçok alanda güvenle kullanılmaktadır. SHA-2' ye yapılan saldırılar, SHA-1 ailesinin kırılma sürecinde olduğu gibi özellikle çarpışma saldırıları yani iki farklı mesaj için aynı hash değerini bulma amaçlı saldırılar şeklindedir ve SHA-2' nin azaltılmış adım şekline yapılmaktadır. SHA-1 ailesine yapılan saldırılar da öncelikle SHA-1' in azaltılmış adımına yapılıp lokal çarpışmalar bulunmuş ve son olarak SHA-1' in tam sürümünde çarpışmalar bulunana kadar bu saldırılar genişletilmiş ve sonunda SHA-1 algoritması kırılmıştır. SHA-2' nin tam sürümünde herhangi bir çakışma bulunamamakla beraber, azaltılmış adımlı SHA-2 ailesinde lokal çarpışmalar bulunmuştur.

İlk olarak 2003 yılında Gilbert ve Handschuh[47] tarafından SHA-2' nin 9 adıma azaltılmış sürümüne saldırılar düzenlenmiş ve 2^{66} olasılık ile lokal çarpışmalar bulunmuştur. Daha sonra Hawkes ve arkadaşları[48] bu saldırıyı modüler farklılıkları gözeterek genişlettiler ve 2^{39} olasılık ile lokal çarpışmalar bulmayı başardılar. XOR farklılıkları kullanılarak Hölbl ve arkadaşları[49] tarafından 2^{38}' e kadar olasılıkla lokal çarpışmalara ulaşılmıştır. Sanadhya ve Sarkar[50] tarafından daha düşük olasılıkla ancak başka özelliklerle birlikte lokal çarpışmalar incelenmiştir.

Nikolic ve Biryukov[51] farklı bir teknik kullanarak 21 adıma kadar çarpışmalar elde etmiştir. Yakın bir tarihte ise Sanadhya ve Sarkar[52] bu çalışmayı genişletmiş ve SHA-256' nın 22 adımı için bir çarpışma örneği göstermiştir. Son olarak Sebastian INDESTEEGE ve arkadaşları SHA-2 ailesinin üyelerinden SHA-256 ve SHA-512 algoritmalarının azaltılmış adım şekillerinden 23 adım ve 24 adım sürümlerinde çakışmalar tespit etmişlerdir[53]. Tespit edilen çakışma olasılığı 23 ve 24 adım SHA-256 için sırasıyla 2^{18} ve $2^{28.5}$, 23 ve 24 adım SHA-512 için sırasıyla $2^{44.9}$ ve $2^{53.}$ tür. buda SHA-2 ailesinin ileri bir zamanda kırılabileceği yönündeki ihtimalleri büyük ölçüde arttırmıştır.

6. SONUÇLAR

Bu tezde düşük bir maliyetle tek kullanımlık OTP şifre üretimi sistemi tasarımı amaçlanmış olup şifre üretimi java destekli mobil aygıtlarda gerçekleştirilmiştir. OTP şifre üretiminde HMAC-SHA-2 algoritması tercih edilmiştir. Çünkü HMAC-SHA-1 algoritması SHA-1' in kırılması nedeniyle risk altında bulunmaktadır. SHA-2 algoritmasının azaltılmış adım şekline bazı çarpışma saldırıları düzenlensede henüz SHA-2 tam sürümüne karşı başarılı bir saldırı kaydedilmemiştir. SHA-2 ailesinin iki önemli üyesi bulunmaktadır. Bunlar SHA-256 ve SHA-512 dir. SHA-256 mesajı 512 bitlik bloklar olarak işleyerek 256 bit özet değer üretirken, SHA-512 mesajı 1024 bitlik bloklar olarak işleyerek 512 bit özet değer üretir. Her ne kadar SHA-512 daha büyük boyutta çıktı üretsede, iki algoritmanın sıkıştırma fonksiyonları çok benzer olduğundan SHA-256 ya yapılabilecek başarılı bir saldırı SHA-512 algoritmasınıda etkileyecektir. Tez çalışmamızda üretilen OTP yazılımında SHA-2 ailesinden SHA-512 algoritması kullanıldı. SHA-3 algoritması için seçmeler devam etmekle beraber SHA-512 algoritması şu anda en güvenilir özetleme algoritması olarak kabul edilmekte ve sayısal imza, bütünlük doğrulama, güvenlik sertifikası gibi uygulamaların büyük bölümünde başarıyla kullanılmaktadır. Tez yazılımında SHA-512' nin bu gücünden yararlanılarak tek kullanımlık şifre üretme hedeflenmiş ve başarılı sonuçlar alınmıştır. Mobil aygıt olarak java destekli telefonlar hedef alınmış ve yazılım bu doğrultuda geliştirilmiştir. Yazılımın server tarafında ise Microsoft .NET 4.0 destekli doğrulama programı kullanılmıştır. OTP sistemlerinden önce şifreleme bilimi ve tarihi detaylı bir şekilde incelenmiş ve şifreleme biliminde önemli köşe taşları olan DES, RSA gibi şifreleme algoritmaları ve yine OTP sistemlerinin temel esası olan tek yönlü fonksiyonlar incelenmiş bunlardan tez programında kullanılan SHA ailesinin şifreleme algoritmaları detaylı olarak anlatılmıştır.

Tez programı Netbeans 6.9.1 geliştirme ortamında ve Microsoft Visual Studio 2010 programında çalıştırılıp başarıyla uygulanmıştır. Ayrıca java destekli olan Nokia 3600 cep telefonu modelinde client programı denenmiş ve başarılı sonuçlar alınmıştır.

Ülkemizde son yıllarda yapılan düzenlemelerle internet bankacılığı sistemlerinde tek kullanımlık şifre sistemlerini kullanmak zorunlu hale getirilmiştir. Tezde önerilen programla bu şifre üretimi dışa bağımlı olmadan ve düşük bir maliyetle güvenli bir şekilde

internet bankacılığı sistemlerinde veya güvenliğin çok önemli olduğu web tabanlı sistemlerde kullanılabilecektir.

KAYNAKLAR

[1] **Pawlıw B.**, 2006. Cryptography, Prentice Hall, London.

[2] **www.kamusm.gov.tr/** Açık Anahtar Yapısı Eğitim Kitabı, 1 Haziran 2011.

[3] **www.genbilim.com/** Şifrelemenin Tarihi, 1 Haziran 2011.

[4] **Cohen F.**, 1995. A Short History of Cryptography, ASP Press, New York.

[5] **Kodaz H.**, 2003. RSA Şifreleme Algoritmasının Uygulaması, *Akademik Bilişim 2003*, Çukurova Üniversitesi, Adana. 3-5 Şubat.

[6] **Stallings W.**, 1997. Cryptography and Network Security, Second Edition, Prentice Hall, London.

[7] **Biham E.**, **Shamir A.**, 1991. Differential Cryptanalysis of DES-like Cryptosystems, *Journal of Cryptology*, Vol 4, No 1 pp.3-72.

[8] **Altan K.**, **Kaşkaloğlu K.**, 2004. Kriptolojiye Giriş Ders Notları, Uygulamalı Matematik Enstitüsü, ODTÜ, Ankara.

[9] **Kak A.**, 2009. Computer and Network Security, Purdue University, Indiana.

[10] **http://faculty.ksu.edu.sa/mdahshan/default.aspx/** Course Notes-Security and Internet Protocols, 1 Haziran 2011.

[11] **Dalkılıç G.**, **Akın O.**, 2005. Anahtar Tabanlı Gelişmiş Rotor Makinesi, *Akademik Bilişim 2005*, Gaziantep Üniversitesi, Gaziantep. 2-4 Şubat.

[12] **Stallings W.**, 2005. Crytography and Network Security Principles and Practices, Fifth Edition, Prentice Hall, London.

[13] **Schneier B.**, 1996. Applied Cryptology, Second Edition: Protocols, Algorithms and Source Code in C, Wiley Publishing. New York.

[14] **Yıldırım K.**, 2006. Veri Şifrelemesinde Simetrik ve Asimetrik Anahtarlama Algoritmalarının Uygulanması, *Yüksek Lisans Tezi*, Kocaeli Üniversitesi Fen Bilimleri Enstitüsü, Kocaeli.

[15] **Sakallı T.**, **Şahin A.**, **Buluş E.**, 2005. Modern Blok Şifreleme Algoritmalarının gücünün İncelenmesi, *2. Mühendislik Bilimleri Genç Araştırmacılar Kongresi MBGAK*, İstanbul. 17-19 Kasım.

[16] **FIPS 46-3**, 1999. Data Ecryption Standard, *Federal Information Processing Standard (FIPS)*, Publication 46-3, National Bureau of Standards, U.S. Department of Commerce, Washington D.C.

[17] **FIPS 197**, 2001. Advanced Encryption Standard, *Federal Information Processing Standard (FIPS)*, Publication 197, National Bureau of Standards, U.S.

Department of Commerce, Washington D.C.

[18] **Raphael C. , Phan W.** , 2004. Impossible Differential Cryptanalysis of 7-round Advanced Encryption Standard (AES), Information Processing Letters, *Elsevier Science*, vol. 91, number 1, 2004, pp. 33-38.

[19] **Saygı Z. , Yıldırım E.** ,2004. Kriptolojiye Giriş Ders Notları,Uygulamalı Matematik Enstitüsü, ODTÜ, Ankara.

[20] **Kelıher L.**, 2002. Linear Cryptanalysis of Substitution-Permutation Networks, *Ph. D. Thesis*, Queen's University, Canada.

[21] **Biham E. , Bıryukov A.**, 1997. An Improvement of Davies' Attack on DES, *Journal Of Cryptology 10*, no. 3, 195-205.

[22] **Adams C., Tavares S.**, 1990. Adams C., Tavares S., The Use of Bent Sequences to Achieve Higher-Order Strict Avalanche Criterion in S-Box Design, *Technical Report*, Ontario, Canada.

[23] **Biham E., Bıryukov A.**, 1994. How to Strengthen DES using Existing Hardware, ASIACRYPT '94 4th *International Conference on the Theory and Application of Cryptology*. Wollongong, Australia.

[24] **Kwangjo K.**, 1991. Construction of DES-like S-boxes based on Boolean Functions Satisfying the SAC, *ASIACRYPT'91*. Japan.

[25] **Adams C., Tavares S.**, 1993. Designing S-boxes, Conclusions.

[26] **Yerlikaya T.**, 2002. Şifreleme Teknikleri ve Güncel Uygulama Olanakları, *Yüksek Lisans Tezi*, Trakya Üniversitesi Fen Bilimleri Enstitüsü, Edirne.

[27] **Matsuı M.**, 1994. The First Experimental Cryptanalysis of the Data Encryption Standard, *Advanced in Cryptology, Crypto'94, Lecturer Notes in Computer Science, Springer-Verlag, pp. 1-11.*

[28] **Matsuı M.,** 1994. Linear Crptanalysis Method for DES Cipher, *Advances in Cryptology-Eurocrypt'93, Springer-Verlag, pp.386-397.*

[29] **Sakallı T.** , 2006. Blok ve Akış Şifreleri, *Yüksek lisans Tezi*, Trakya Üniversitesi Fen Bilimleri Enstitüsü, Edirne.

[30] **Avaroğlu E.**, 2010. Asimetrik Şifreleme Algoritmaları, *Doktora Semineri*, Fırat Üniversitesi Fen Bilimleri Enstitüsü, Elazığ.

[31] **Rivest, R., L.,Shamir, A.and Adleman,** 1978. A Method for Obtaining Digital Signatures and PublicKey Crytosystems, *Comminicationa of the ACM.*

[32] **http://world.std.com/~franl/crypto/rsa-guts.html** , The Mathematical Guts of RSA Encryption, 1 Mayıs 2011.

[33] **http://www.cryptographyworld.com/rsa.htm** , The Cryptography Guide-Rsa, 1 Haziran 2011.

[34] **Yerlikaya T. , Buluş E. , Arda D. ,** 2005. Asimetrik Kripto Sistemler ve Uygulamaları, *II.Mühendislik Bilimleri Genç Araştırmacılar Kongresi MBGAK 2005* , İstanbul 17–19 Kasım.

[35] **www.sadievrenseker.com/kriptosem.pdf**, Şifreleme Sistemlerine Giriş ve Açık Anahtar Şifreleme, 1 Haziran 2011.

[36] **Yüksel A.** , 2006. Tek kullanımlık kimlik doğrulamalı anahtar değişim yazılım Uygulamasının geliştirilmesi, *Yüksek Lisans Tezi*, Hacettepe Üniversitesi Fen Bilimleri Enstitüsü, Ankara.

[37] **RFC 1321**, 1992. MD5 Message Digest Algorithm, *MIT Laboratory for Computer Science and RSA Data Security, Inc.*

[38] **Kırkıl K.**, 2010. Kriptografik Özetleme Algoritmaları, *Yüksek Lisans Semineri*, Fırat Üniversitesi Fen Bilimleri Enstitüsü, Elazığ.

[39] **FIPS 180-1,** 1995. Announcing the Standard for Secure Hash Standard, *Federal Information Processing Standards.*

[40] **http://csrc.nist.gov/groups/ST/hash/sha-3/Round1/submissions_rnd1.html** , First Round Candidates, 1 Haziran 2011.

[41] **Soyalıç S.** , 2005. Kriptografik Hash Fonksiyonları ve Uygulamaları, *Yüksek Lisans Tezi* , Erciyes Üniversitesi Fen Bilimleri Enstitüsü, Kayseri.

[42] **RFC 2104**, 1997. HMAC: Keyed-Hashing for Message Authentication. *Request for Comments.*

[43] **RFC 4226,** 2005. HOTP, An Hmac-Based One-Time Password Algorithm. *Request for Comments.*

[44] **Koçak O.** , 2009. Hash Fonksiyonlarının Dizayn ve Analizi, *Yüksek Lisans Tezi,* Ortadoğu Teknik Üniversitesi Fen Bilimleri Enstitüsü, Ankara.

[45] **Wang X. , Yin Y. L. , Yu H. ,** 2005. Finding Collisions in the Full SHA-1, *Crypto-2005*, Santa Barbara, California, USA.

[46] **Wang X. , Yu H. , Yin Y. L. ,** 2005. Efficient Collision Search Attacks on SHA-0, *Crypto-2005*, Santa Barbara, California, USA.

[47] **Gilbert H. and Handschuh H.** , 2003. Security analysis of SHA-256 and sisters. In Mitsuru Matsui and Robert Zuccherato, editors, Selected Areas in Cryptography, *10th Annual International Workshop, SAC 2003*, Ottawa, Canada.

[48] **Hawkes P. , Paddon M. and Gregory G. Rose.**, 2004. On corrective patterns for the SHA-2 family. *Cryptology ePrint Archive,* Report 2004/207, August.

[49] **Hölbl M. , Rechberger C. and Welzer T.** , 2008. Searching for messages conforming to arbitrary sets of conditions in SHA-256. *In Proceedings of WEWORC 2007*, volume 4945 of LNCS.

[50] **Kumar S. , Sanadhya and Palash Sarkar.** , 2007. New Local Collisions for the SHA-2 Hash Family. *In Kil-Hyun Nam and Gwangsoo Rhee, editors, ICISC,* volume 4817 of LNCS, pages 193–205.

[51] **Nikolic I. and Biryukov A.** , 2008. Collisions for Step-Reduced SHA-256. In Kaisa Nyber, editor, Fast Software Encryption, *15th International Workshop,FSE 2008*, Lausanne, Switzerland, March 26-28, 2008, LNCS.

[52] **Kumar S. , Sanadhya and Palash Sarkar.** , 2008. Non-Linear Reduced Round Attacks Against SHA2 Hash Family. *In Proceedings of ACISP.*

[53] **Indesteege S. , Mendel F., Preneel B. and Rechberger C.** , 2008. Collisions and other Non-Random Properties for Step-Reduced SHA-256. *Cryptology ePrint Archive Report 2008/131.*

ÖZGEÇMİŞ

1982 yılında Elazığ' da doğdu. İlk, orta ve lise öğrenimini Elazığ' da tamamladı. Fırat Üniversitesi Mühendislik Fakültesi Bilgisayar Mühendisliği Bölümünden 2005 yılında mezun oldu.

2009 yılında Fırat Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Bölümünde yüksek lisans eğitimine başladı.

2009 yılından itibaren Tunceli Üniversitesi Tunceli Meslek Yüksek Okulunda öğretim görevlisi olarak çalışmaktadır. Evli olup iki çocuk babasıdır.

EKLER

Ek-1: Bilgisayar Programı