T.C.
FIRAT ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

TEK KULLANIMLIK ŞİFRE ÜRETECİ

Selman YAKUT
(111129104)

Yüksek Lisans Tezi
Bilgisayar Mühendisliği Anabilim Dalı
Tez Danışmanı: Doç. Dr. A. Bedri ÖZER

OCAK, 2014

T.C.
FIRAT ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

TEK KULLANIMLIK ŞİFRE ÜRETECİ

YÜKSEK LİSANS TEZİ

Müh. Selman YAKUT
(111129104)

Anabilim Dalı: Bilgisayar Mühendisliği

Programı: Yazılım

Tezin Enstitüye Verildiği Tarih : 21.01.2014
Tezin Savunulduğu Tarih : 03.01.2014

Tez Danışmanı: Doç. Dr. A. Bedri ÖZER (F.Ü.)

Diğer Jüri Üyeleri: Doç. Dr. Bilal ALATAŞ (T.Ü.)

Yrd. Doç. Dr. Taner TUNCER (F.Ü.)

OCAK, 2014

ÖNSÖZ

Bilgi güvenliği insanlığın her dönemi için önemli bir konudur. İlk çağlarda ilkel yöntemlerle bilgi güvenliği sağlanırken gelişen teknoloji ile beraber bilgi güvenliği daha disiplinli ve düzenli yöntemlerle sağlanmaktadır. Özellikle bilgisayarın ve ağ yapılarının, dolayısıyla internetin yaygın kullanılmasıyla bilgi güvenliği konusu daha da önem kazanmaktadır.

Bilgi güvenliğinin en önemli parçalarından biri kimlik doğrulamadır. Kimlik doğrulama, herhangi bir işlem için bir kişinin iddia etiği kişi olduğunu ispatlamasıdır. Kimlik doğrulama için kullanılan yöntemlerin başında Tek Kullanımlık Şifre (TKŞ) gelmektedir. TKŞ kullanımındaki amaç erişimi kısıtlanmış kaynaklara yetkisiz erişimi daha da zor hale getirmektir. Alışıldık sabit şifreler yeterli deneme şansı ve zaman verildiği takdirde yetkisiz kişiler tarafından aşılabilir. Tek kullanımlık şifre uygulamasında şifre sürekli değiştiği için bu risk ortadan kalkar.

Bu tez çalışmasında değerli hocam sayın **Doç. Dr. A. Bedri ÖZER**'e katkılarından dolayı teşekkür ederim.

Selman YAKUT

OCAK-2014

İÇİNDEKİLER

ÖZET

Teknolojinin yaygınlaşmasıyla internet kullanımı çok fazla artmıştır. E-devlet uygulamalarından elektronik ticarete; fatura ödeme işlemlerinden internet bankacılığına kadar birçok uygulama artık günlük yaşantının bir parçası olmaktadır.

Elektronik ortamlarda yapılan işlemlerde kimlik doğrulama işlemi için çoğunlukla kullanıcı adı ve şifre kullanılmaktadır. Ancak gelişen siber saldırılar ile birlikte klasik kimlik doğrulama işlemi artık kritik verilerin güvenliği için yetersiz kalmaktadır. Bu yüzden son dönemde birçok uygulama iki seviyeli bir kimlik doğrulama işlemini tercih etmektedir. İki seviyeli kimlik doğrulama işleminde kullanıcı adı ve şifrenin doğrulanmasının ardından her bir giriş işleminde değişen rasgele bir karakter dizisinin girilmesi istenmektedir.

Tek Kullanımlık Şifre (TKŞ) elektronik ortamlarda kimlik doğrulama işlemini gerçekleştirmek için bir defaya mahsus olarak kullanılan belirli uzunluktaki bir karakter dizisidir. Bunlara atılabilir veya kullan at şifrelerde denilmektedir. Bu karakter dizisinin uzunluğu ve karakter türü uygulamalarda değişiklik göstermektedir.

Bu tez çalışmasında TKŞ üretimi incelendi. TKŞ çeşitli şekillerde üretilmekle birlikte en yaygın kullanım şekli özet fonksiyonlarını temel alan yaklaşımdır. Özet fonksiyonları rastgele uzunluktaki bir mesaj için belirli bir uzunlukta bir özet değeri üretir. Bu tez çalışmasında TKŞ üretimi için Keccak özet algoritması kullanıldı. Bu algoritma Ekim 2012'de NIST (National Instute of Standards and Technology) tarafından özet (Hash) algoritmaları için son standart olarak belilendi. Bu algoritma diğer aday algoritmalara göre daha hızlı, güvenli olduğundan tercih edildi.

Özet fonksiyonlarına dayanan TKŞ üretim yöntemleri, HMAC (Özetleme Tabanlı Mesaj Doğrulama Kodu) yapıları olarak adlandırılmaktadır. Bu yapılarda, özet fonksiyonu ve anahtar değeri beraber kullanılarak HMAC değeri üretilmektedir. Üretilen TKŞ değeri bu HMAC değerine dayanmaktadır. Bu uygulamada; TKŞ değerinin uzunluğu, uygulamada kullanışlı olabilmesi için, sekiz karakter olarak belirlendi.

Anahtar Kelimeler: Özet Fonksiyonları, Keccak, Tek Kullanımlık Şifreler, Özetleme Tabanlı Mesaj Doğrulama Kodu

ABSTRACT

One Time Password Generator

With the widespread of technology, use of internet has increased a lot. Many applications have been a part of our daily lives such as, e- government applications, electronic commerce, Internet banking and bill payment transactions.

The user name and password is used for authentication in electronic media process. However, along with evolving cyber-attacks, now classic authentication process is insufficient for the security of critical data. So, many applications are preferred a two-level authentication process in the last period. Two-level authentication process is required to be entered random string of characters after the input user name and his password in each the verification process.

One Time Password (OTP) which is fixed length strings to perform authentication in electronic media is used as a one-time. They are called in disposable or throwaway password. This character length and type is vary for applications.

OTP production was studied in this thesis. OTP produced in various ways but the most common approach is to use pattern based on the hash function. Hash functions for a message of arbitrary length value produces a summary of a certain length. In this thesis, Keccak digest algorithm was used for the production of OTP. This algorithm has been selected as the latest standards for hash algorithm in October 2012 by NIST (National Instute of Standards and Technology). This algorithm is preferred because it is faster and safer than the others.

OTP production methods based on hash functions is called HMAC (Hashing-Based Message Authentication Code) structure. In these structures, the key value is using with the hash function to generate the HMAC value. Produced OTP value is based on the HMAC value. In this application, the length of the value OTP was the eight characters to be useful in practice.

Keywords: Hash Function, Keccak, One Time Passwords, Hash-Based Message Authentication Code

ŞEKİLLER LİSTESİ

KISALTMALAR

AES	**:**	Advanced Encription Standart
DES	**:**	Data Encryption Standart
RSA	**:**	Rivest-Shamir-Adleman Algorithm
MD4	**:**	Message Digest Algorithm 4
MD5	**:**	Message Digest Algorithm 5
NIST	**:**	National Instute of Standart and Technology
FIPS	**:**	Federal Information Processing Standart
SHA-1	**:**	Secure Hash Algorithm-1
MDK	**:**	Mesaj Doğrulama Kodu
HMAC	**;**	Hash-Based Message Authentication Code
SHA-2	**:**	Secure Hash Algorithm-2
SHA-512	**:**	Secure Hash Algorithm-512
h	**:**	Özet Değeri
H	**:**	Özet Fonksiyonu
z	**:**	Sayısal İmza
$\boldsymbol{b_i}$	**:**	i. Mesaj Bloku
$\boldsymbol{x}$	**:**	Giriş mesajı
$\boldsymbol{b_i}$	**:**	i. Giriş Mesajı Bloğu
l	**:**	Mesaj Uzunluğu
E	**:**	Şifreleme
D	**:**	Şifre Çözme
k	**:**	Anahtar Değeri
P	**:**	Çakışmama Olasılığı
$\boldsymbol{\lambda}$	**:**	Çakışma Olasılığı
$\boldsymbol{k_{pub}}$	**:**	Genel Anahtar
$\boldsymbol{k_{pr}}$	**:**	Özel Anahtar
W	**:**	İfade (state)
s	**:**	Özet Fonksiyonu Hesaplamasında Kullanılan Gizli Değer

1. GİRİŞ

İlk çağlardan beri insanlar bilgi güvenliğiyle ilgilenmeye ihtiyaç duymuşlardır. Çağımızda teknolojinin hızlı gelişimi ve birçok bilgi kaynağın paylaşılması bu konuyu daha da önemli kılmaktadır. Dolayısıyla, askeri amaçlardan banka uygulamalarına, kamu kurumlarındaki uygulamalardan kişisel e-posta iletimlerine kadar birçok alanda bilginin güvenliği önem taşımaktadır [1].

Şifre bilimi olarak adlandırılan kriptoloji, şifreleme (kriptografi) ve şifre analiz (kriptoanaliz) olmak üzere iki ana başlıktan oluşmaktadır. Şifreleme bilgi güvenliği ile uğraşır yani bilginin gizlenmesini, saklanmasını amaçlar. Şifre Analizi ise şifrelemenin tam tersidir yani güvenli bilginin kırılması ve belirlenmesi için çalışılır [2]. Bilgi güvenliği, bilgileri izinsiz erişimlerden, kullanımından, ifşa edilmesinden, yok edilmesinden, değiştirilmesinden veya hasar verilmesinden koruma işlemidir [2]. Bilgi güvenliği ilk zamanlardan beri değişiklik uygulamaları olmakla beraber günümüzde genellikle matematiksel temellere ve işlemlere dayanan algoritmalarla sağlanmaktadır.

Bilgi güvenliği; temel olarak gizlilik, bütünlük, kimlik doğrulama ve inkâr edilemezlik gibi alt başlıklara ayrılmaktadır. Gizlilik genellikle şifreleme algoritmalarıyla, kimlik doğrulama ve inkâr edilemezlik sayısal imza yapısıyla sağlanmaktadır. Bütünlük, ise temel olarak özet (hash) fonksiyonlarıyla sağlanmaktadır. Bunun birlikte, diğer bir önemli güvenlik parametresi ise önemli kaynaklara yetkisiz kişilerce erişimin engellenmesidir.

Günümüzde banka hesap numaralarına erişim başta olmak üzere kimlik doğrulama gerektiren birçok uygulamada sabit şifre kullanımı yetersiz kalmaktadır. Bu sistemler artan ve gelişen güvenlik tehditlerine karşı yetersiz kalmaktadır. Özellikle alış veriş, havale gibi işlemlerin internet üzerinden yapılması ve bu ortamların güveli olmayışı kimlik doğrulama yöntemlerini daha da önemli kılmaktadır.

Genel olarak kimlik doğrulama; akıllı kart sistemleri, biyometrik sistemleri ve şifreleme sistemleri olmak üzere üç kısma ayrılır. Akıllı kart sistemleri, içerisinde kendine özel işlemcisi olan, özel şifreleme tekniği ile kopyalanmaya veya içeriğinin okunmasına izin vermeyen yonga içeren özel kartlardır. Biyometrik kimlik doğrulama sistemleri, kimlik onaylamak için kişiye ait benzersiz bir fiziksel veya davranışsal özelliği kullanan sistemlerdir. Şifreleme sistemlerinde ise kimlik doğrulama için şifre değerleri kullanılmaktadır.

Şifre tabanlı kimlik doğrulama sistemlerinde sunucu tarafında her kullanıcı için bir şifre değeri tutulmaktadır. Erişim esnasında kullanıcıdan alınan şifre değeri sistemde bulunan şifre değeriyle karşılaştırılarak yetkilendirme işlemi gerçekleştirilmektedir Ayrıca bu şifreler kimse tarafından görülmemeli, ağ üzerinde güvenli bir şekilde iletilmeli ve kullanıcı açısından ideal uzunlukta olmalıdır [3].

Kimlik doğrulama için kullanılan şifreler statik ve dinamik şifreler olmak üzere iki kısma ayrılır. Statik şifreler ilgili kaynağa her erişim için aynıdır. Şifre kullanıcıdan alınır ve daha önce kullanıcı için saklanan şifre ile karşılaştırılır eğer karşılaştırma sonucu pozitifse kimlik doğrulaması gerçekleştirilmiş olur. Dinamik şifreler ise ilgili kaynağa her erişimde değişir ve kullanıcı her defasında farklı bir şifre kullanılarak kimlik doğrulama işlemi gerçekleştirilir. Her kimlik doğrulama işleminde yenilenen bu şifrelere Tek Kullanımlık Şifreler (TKŞ) veya kullan at şifreler de denilmektedir. Bu yöntemde kullanılan şifreler tekrar kullanılmaz dolayısıyla herhangi bir oturumda kullanılan şifre elde edilse bile bu şifre değeri kullanılarak önemli kaynaklara erişim mümkün olmayacaktır [4].

TKŞ üretimi için çeşitli yöntemler mevcuttur. Bu tez çalışmasında HMAC tabanlı TKŞ üretimi incelendi çünkü iki seviyeli kimlik doğrulama için en yalın, kullanışlı ve yaygın sistem TKŞ kullanımıdır [5]. Bu yöntemin gerçekleştirilmesi için Keccak özet algoritması kullanıldı. Bu algoritma kullanılan en son özet algoritmaları standardıdır [6].

1.1. Tezin Amacı

Tezin amacı elektronik ortamlarda kimlik doğrulama işleminde kullanılabilecek TKŞ değerlerinin üretileceği bir uygulamasının gerçekleştirilmesidir. Yapılan tez çalışmasının ulusal yönden birçok katkı sağlaması amaçlanmıştır. Uygulama ile birlikte ihtiyaç duyulan bu yazılımların yurt içinden temin edilebilmesine katkı sağlanması hedeflenmektedir. İlk olarak bu yazılımların temin edilmesi için harcanan büyük miktardaki paraların yurt dışına çıkmaması hedeflendi. İkinci olarak ülkedeki bilgisayar güvenliği ve yazılım alanlarındaki eksikliğin doldurulması amaçlandı. Böylece ülkemizin bilgisayar güvenliği ve yazılım alanlarda ilerlemesine katkı sağlanması amaçlandı. Bunların yanında, bu tez çalışmasıyla ülke içinde bilgi güvenliği ve yazılım alanlarında bilgi birikimine katkı sağlanması hedeflendi.

Geliştirilen yazılımda kullanılan SHA-3 (Keccak) özet algoritması en son özet algoritması standardıdır [6]. Bu algoritmanın mevcut özet algoritmaları içinde en güvenli olduğu araştırmalarla kanıtlanmıştır ve tablolarla gösterilmiştir [7]. Önerilen yöntemde ise SHA-3 standardı kullanıldığı için herhangi bir şüpheye yer vermeden kişi ve kurumların uygulamayı güvenle kullanabilmesi sağlanmıştır [8].

1.2. Tezin Yapısı

Tez çalışması aşağıdaki kısımlardan oluşmaktadır.

İkinci kısımda, ilk olarak özet fonksiyonlarının, sayısal imza ve mesaj doğrulama kodu başta olmak üzere, kullanım alanları incelendi. Daha sonra ise sırasıyla; özet fonksiyonlarının çeşitleri, güvenli özet fonksiyonlarının yapısı, özellikle uzun zaman standart olarak kullanılan SHA-512 fonksiyonunun yapısı incelendi. Ayrıca, uygulama ve güvenlik açısından özet fonksiyonlarının sahip olması gereken özellikler ayrı ayrı ele alındı. Özellikle çakışma dayanıklılığında, etkili bir yaklaşım olan doğum günü çelişkisi matematiksel yaklaşımlarla ifade edildi. Son olarak da, özet fonksiyonlarına karşı yapılan saldırı türleri incelendi.

Üçüncü kısımda, özetleme algoritmaları için en son standart olan Keccak özet algoritması incelendi. Daha sonra bu algoritmanın blok yapısı ve çekirdek fonksiyonu ayrıntılı bir şekilde ele alındı. Son olarak Keccak özet algoritmasının güvenlik, maliyet ve esneklik gibi değerlendirme kriterleri incelendi.

Dördüncü kısımda, HMAC algoritması incelendi. İlk olarak bu algoritmanın tasarım hedefleri ve blok yapısı ele alındı. Daha sonra ise HMAC algoritması kullanarak anahtar değeri üretimi ve üretilen değerin test edilmesi incelendi.

Beşinci kısımda, öncelikle TKŞ yapılarının gerekliliği ve önemi üzerinde duruldu. Daha sonra TKŞ üretim yöntemleri olan zaman tabanlı ve matematiksel yöntemler incelendi. Son olarak, TKŞ değerlerinin dağıtımı ve üretilen değerlerin analizi ve değerlendirilmesi gerçekleştirildi.

Altıncı kısımda ise geliştirilen uygulamanın genel yapısı anlatıldı. Daha sonra uygulamada kullanılan yöntemin detayları incelendi ve iki örnek uygulama ara yüzü gösterildi.

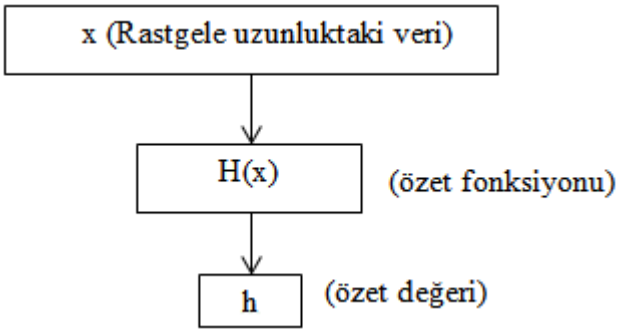
Yedinci kısımda ise sonuçların değerlendirmesi işlemi yapıldı.

2. ÖZET FONKSİYONLARI

Temel olarak özet fonksiyonları iki farklı kullanım alanına sahiptir. Bunların ilki veri tabanında veya dosya organizasyonunda kullanılan eşleştirme (hasing) yapan özet fonksiyonlarıdır [9]. Hash fonksiyonları, veri tabanında genellikle tabloda aranan bir veriyi hızlı bir şekilde bulmak veya veri karşılaştırma işlemlerini hızlandırmak, büyük bir dosyada aynı veya benzer kayıtları tespit etmek, DNA dizisinde benzer dizilimleri bulmak vb. işlemler için kullanılır [10]. Özet fonksiyonları diğer kullanım alanı ise şifrelemedir. Bu çalışmada şifrelemede kullanılan özet fonksiyonları incelendi.

Özet fonksiyonları önemli şifreleme yapıları olup protokollerde yaygın bir şekilde kullanılmaktadır. Bu fonksiyonların ürettiği değere mesajın özeti veya özet değeri denir. Genellikle bu değer mesaja göre kısadır ve kullanılan algoritmaya bağlı olarak belirli bir uzunluktadır. Belirli bir mesaj için bu değer o mesaja özgüdür o mesajın parmak izi olarak tanımlanabilir [11]. Şifrelemede özet fonksiyonları sayısal imza ve mesaj doğrulama kodu yapılarının temel parçasıdır. Ayrıca bu fonksiyonlar anahtar üretimi veya şifre özetlerinin saklanması gibi geniş bir alanda kullanılır [12].

Özet fonksiyonları genel olarak herhangi bir veri üzerinde bütünlüğü kontrol etmek için kullanılır. Örneğin bu fonksiyonlarla göndericinin ve alıcının aynı mesajı paylaşıp paylaşmadığı, yani iletim veya depolama sırasında mesaj üzerinde bir değişiklik olup olmadığını belirlenebilir [12]. Bu fonksiyonların en önemli işlevi veri bütünlüğünü kontrol etmede kullanılmalarıdır [13].



Şekil 2.1. Özet fonksiyonlarının genel yapısı

Özet fonksiyonlarında değişik uzunluktaki veri bloklarını giriş olarak alınır. h özet değeri ve H özet fonksiyonu olmak üzere, herhangi uzunluktaki bir x giriş mesajı için $h = H(x)$ olacak şeklinde sabit uzunluktaki özet değeri üretilir. Şekil 2. 1.'de şifreleme

özet fonksiyonlarının genel yapısı verilmiştir. Genel olarak giriş mesajının blok uzunluğu, kullanılan algoritmaya bağlı olarak, eşit uzunluktaki bloklara ayrılır ve bu bloklar sırasıyla işlenir. Eğer gerekiyorsa son bloğun da aynı uzunluğa erişmesi için ekleme yapılır. Ayrıca orijinal mesajın uzunluğu da mesajın sonuna mesaj uzunluğunu göstermek için ayrılan alana yazılmaktadır. L sembolü ile gösterilen uzunluk alanı aynı özet değerine sahip alternatif mesajların üretilmesinin zorluğunu artırmak için kullanılan bir güvenlik önlemidir [12].

2.1. Özet Fonksiyonlarının Kullanım Alanları

Özet fonksiyonlar oldukça farklı güvenlik uygulamalarında ve internet protokollerinde kullanılırlar. Bu fonksiyonların daha iyi anlaşılması için özet fonksiyonlarının farklı alanlardaki kullanımı incelendi.

2.1.1. Mesaj Doğrulama Kodu

Mesaj doğrulama kodu (MDK), mesajın bütünlüğünü kontrol etmek için kullanılan yapılardır. Bu yapılar, mesaj iletimi sırasında mesaj üzerinde oluşan değişikliklerin fark edilebilmesini sağlar. Daha açık bir ifadeyle, bu yapılarla mesaj iletimi sırasında mesaj üzerinde herhangi bir ekleme, çıkarma, silme veya değiştirme işleminin yapılıp yapılmadığı kontrol edilebilir. Şekil 2. 2.'de özet fonksiyonlarının mesaj doğrulama kodundaki değişik kullanımları gösterilmiştir.

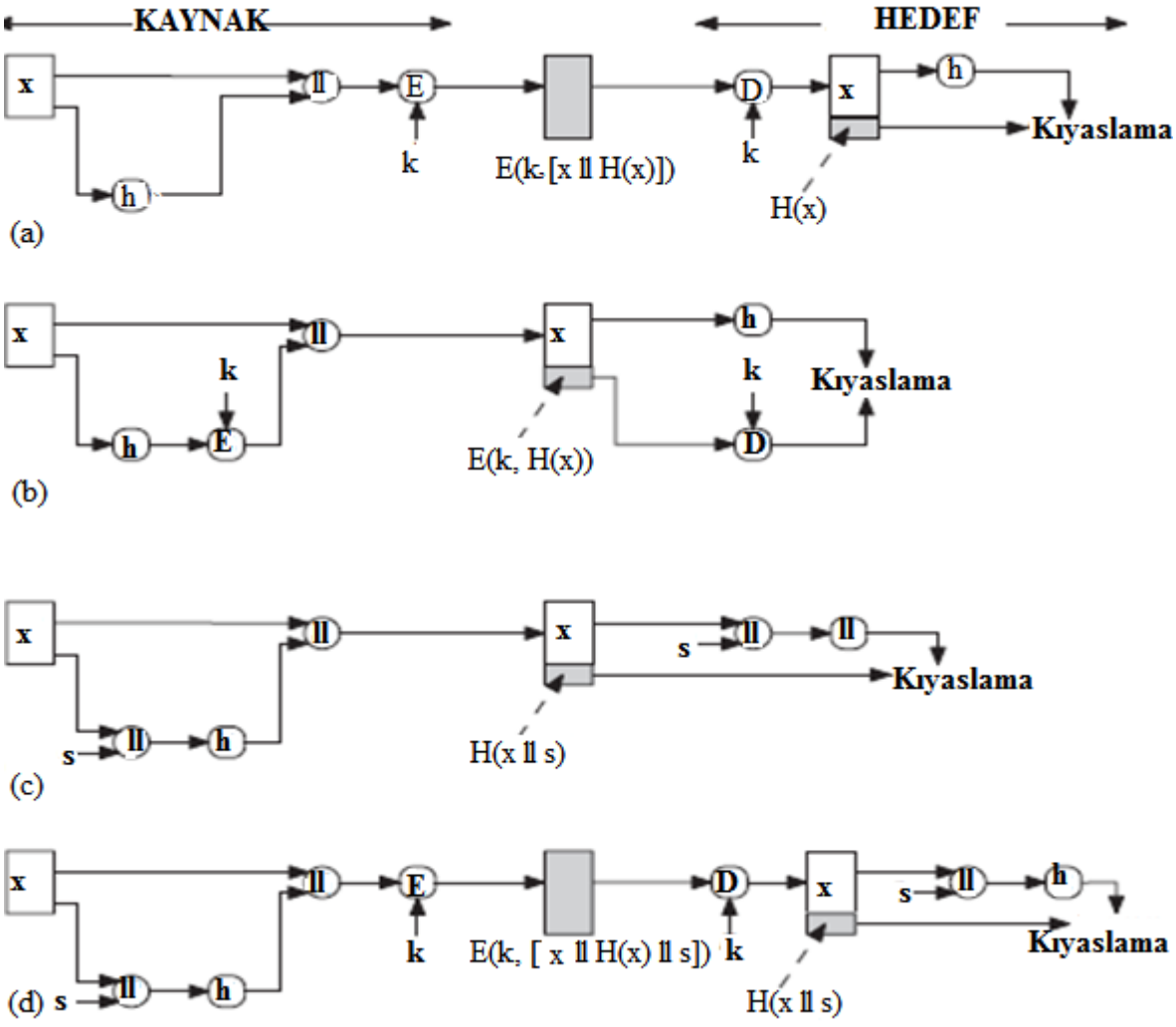
Şekil 2. 2. a'da özet kodu eklenmiş mesaj, simetrik şifreleme ile şifrelenmiştir. Sadece kaynak ve hedef kişi gizli anahtarı bildiği için kaynaktan gelen mesaj değiştirilemez. Özet fonksiyonuyla hem bütünlük kontrolü hem de gizlilik de sağlanmıştır. Özet değeri orijinal mesaja eklendiği için şifreleme işlemi bütün mesaj üzerinde yapılmıştır.

Şekil 2. 2. b'de simetrik şifre kullanılarak sadece özet değeri şifrelenmiştir. Veri gizliliğinin gerekli olmadığı uygulamalarda bu tarz kullanım işlem yükünü azaltır.

Şekil 2. 2. c'de mesaj doğrulama için özet fonksiyonu kullanıp şifreleme yapmamak mümkündür. Bu teknik iki tarafın s gizli değerini haberleşme ile paylaştıklarını varsayar. Kaynak x ve s değeriyle özet değerini hesaplar ve bu değeri mesaja ekler. Hedef, s değerine sahip olduğu için doğrulama işlemini yapmak amacıyla özet değerini yeniden

hesaplar. Gizli değerin kendisi gönderilmediği için iletim hattına üçüncü kişilerce mesaj değiştirilemez veya farklı bir özet değeri üretilemez.

Şekil 2. 2. d'de özet değeri eklenen bütün mesaj şifrelenerek c de sunulan yaklaşıma güvenlik eklendi.



Şekil 2.2. Mesaj doğrulama kodu örnekleri [13]

Eğer mesaj iletimi sırasında gizlilik gerekli değilse daha az hesaplama gerektiren Şekil 0. 0. (b)'deki problem modu, mesajın hepsini şifreleyen Şekil 0. 0. (a) ve Şekil 0. 0. (d)'deki türevlere göre daha avantajlıdır. Bu avantajlardan birkaçı aşağıda belirtilmiştir.

1. Şifreleme yazılımları oldukça yavaştır. Mesaj başına şifrelenecek veri miktarı küçük olmasına rağmen mesajların sisteme giriş çıkışı sürekli bir akım halinde olabilir.

2. Şifreleme donanımlarının maliyeti küçümsenmeyecek kadar yüksektir. Örneğin simetrik şifreleme algoritması olan DES in düşük maliyetli yonga uygulamaları

kullanılabilir fakat bu yongalar ağdaki bütün bilgisayarlara eklenmelidir, bu ise maliyeti artırır.

3. Şifreleme algoritmaları, algoritmayı tasarlayan kişiler tarafından saklanmaktadır ve lisanslı ürünlerin kullanımı maliyetlidir.

MDK, anahtarlı özet fonksiyonu olarak da bilinir. Bu fonksiyonlar iki taraf arasında bilgi değişimini doğrulamak için sadece iki tarafın paylaştığı gizli bir anahtar değeri kullanır. MDK fonksiyonları giriş olarak gizli anahtar değeri ve veri blokunu alır ve MDK olarak adlandırılan özet değerini üretir. Bu değer o mesajla birlikte iletilebilir veya saklanabilir. Eğer mesajın bütünlüğünü kontrol etmek gerekirse MDK fonksiyonu tekrar mesaja uygulanır ve sonuç daha önce elde edilen özet değeriyle kıyaslanır. Mesaj değiştirilebilir fakat gizli anahtarı bilinmediğinden aynı MDK değeri elde edilemez. Bu uygulamada göndericinin kimliği de doğrulanır çünkü gizli anahtar değeri sadece gönderici ve alıcı tarafından bilinmektedir.

Özetleme ve şifrelemenin beraber kullanılması sonucu genel yapılar oluşturulabilir. Bu yapılarla değişik uzunluktaki mesajlar ve gizli anahtar değeri alınır ve bu anahtar değerini bilmeyen saldırganlara karşı güvenli ve sabit uzunlukta bir çıkış üretilir. Pratikte, oldukça etkili olan MDK algoritmaları tasarlanabilir.

2.1.2. Sayısal İmza

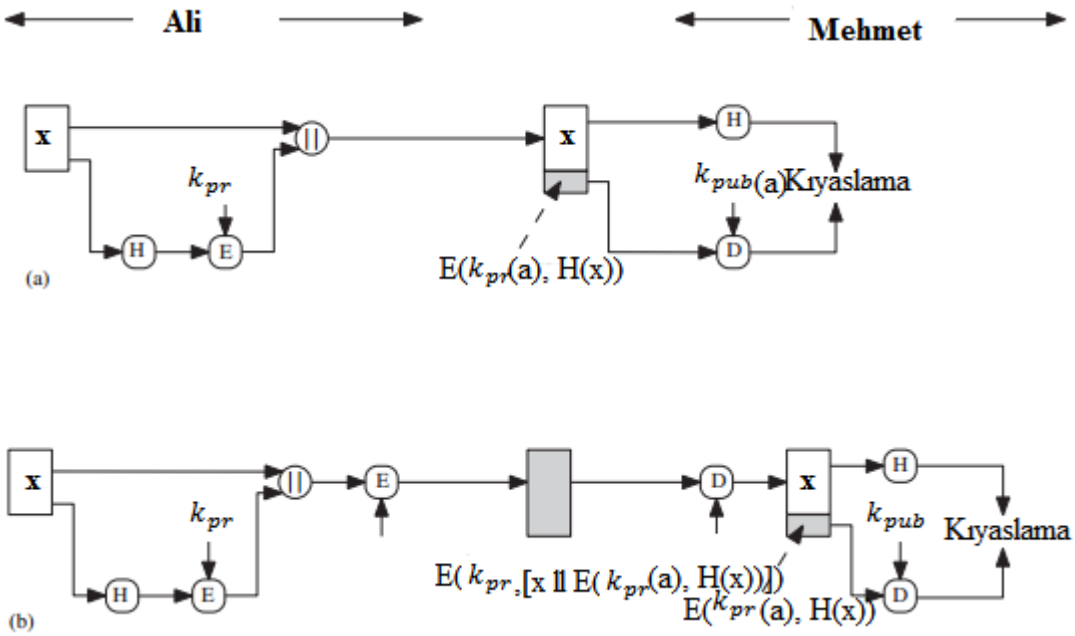
Modern şifrelemede özet fonksiyonları birçok uygulamada kullanılmalarına rağmen, belki de bu fonksiyonların en önemli kullanım alanları sayısal imza yapılarıdır. Sayısal imzanın temeli RSA (Rivest-Shamir-Adleman) gibi asimetrik algoritmalara dayanır. Fakat bütün bu yapılar için gerekli olan düz metnin uzunluğu sınırlıdır. Örneğin RSA da mesajın uzunluğu genellikle 1024 ve 3072 bitten daha uzun olamaz [12]. Acaba daha uzun bir mesajın imzalanması gerekirse imzalama işlemi için nasıl bir yol izlenecektir ve bu yolla etkili bir hesaplama yapılabilecek midir?

Mesaj doğrulama uygulamalarına benzeyen diğer önemli bir uygulama sayısal imza yapılarıdır. Sayısal imzada imzalanmak istenen mesajın özet değeri kullanıcının özel (private) anahtarıyla şifrelenir ve mesajla birlikte iletilir. Mesajı imzalayanın genel anahtarıyla sayısal imza yapısı kontrol edebilir. Bu durumda mesajın sonuna eklenen imza değeriyle alıcı tarafından üretilen özet değerinin aynı olduğu şekilde mesaj üzerinde

değişiklik yapılabilmesi için göndericinin özel anahtarının bilinmesi gerekir. Şekil 2. 3.'de özet değerinin sayısal imza yapısında kullanımı gösterilmiştir.

Şekil 2. 3. a'da göndericinin özel anahtarı kullanılarak özet değeri şifrelenmiştir. Bu kimlik doğrulama ve bütünlüğü sağlar çünkü sadece gönderici özet değerini üretebilir ve şifreleyebilir. Bu ise sayısal imzanın esasıdır.

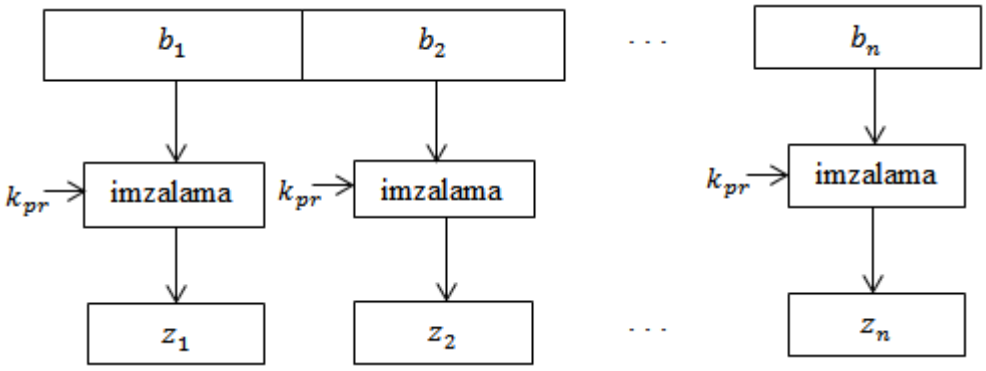
Şekil 2. 3. b sayısal imza ile beraber gizliliğin gerekli olduğu durumlarda kullanılan bir yöntemdir. Burada giriş mesajı ve imza değeri, simetrik şifre yapısı kullanılarak şifrelenir.



Şekil 2.3. Özet fonksiyonlarının sayısal imza yapısında kullanımı [13]

2.1.2.1. Uzun Mesajların İmzalanması

İmzalama işlemi mesajın özet değeri üzerinde yapılmadığı takdirde Şekil 2. 4. olduğu gibi yapılabilir. Burada mesaj, imzalama işlemi uygulanabilecek şekilde alt parçalara ayrılır ve her parça ayrı ayrı imzalanır. Bu yaklaşımda ise aşağıda belirtilen üç önemli problem ortaya çıkar.



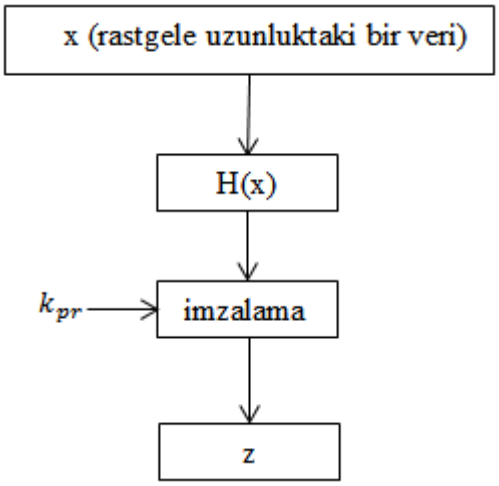
Şekil 2.4. Özet fonksiyonları kullanılmadan yapılabilecek sayısal imza yapısı

1. Sayısal imzalar büyük sayıların modüler üslerini hesaplamak gibi asimetrik işlemlere dayanmaktadır. Eğer sadece bir işlem için az bir zaman harcansa bile multimedya dosyaları gibi uzun masajlar için bu hesaplamaların yapması oldukça fazla bir hesaplama gerektirir. Üstelik bu hesaplama yükü sadece imzalama işlemi için değil aynı zamanda doğrulama işlemi için de gereklidir.

2. Bu yaklaşım mesaj yükünü iki katına çıkmaktadır. Çünkü mesajla beraber aynı uzunluktaki imza değerinin de iletilmesi gereklidir. Sonuç olarak ağ trafiği iki kat artar. Örneğin 1 MB'lık dosya için 1 MB da sayısal imzası değeriyle beraber yaklaşık 2 MB'lık dosya iletimi yapılır.

3. Eğer uzun mesaj bloklarını ayrı ayrı imzalanırsa güvenlik en önemli problem olur. Çünkü mesajın bütünü üzerinde bir kontrol işlemi yapılamaz. Örneğin; iletilen mesaj bloklarından biri veya mesaj blokuna karşılık gelen imza değeri silebilir, mesaj blokları yeniden imzalanabilir veya yeni mesaj blokları ve bu mesaj bloklarına karşılık gelen yeni imza değerleri eklenebilir. Birbirinden ayrı mesaj blokları üzerinde işlem yapılamadığı halde iletilen mesaj bir bütün olarak saldırılara karşı korunamaz.

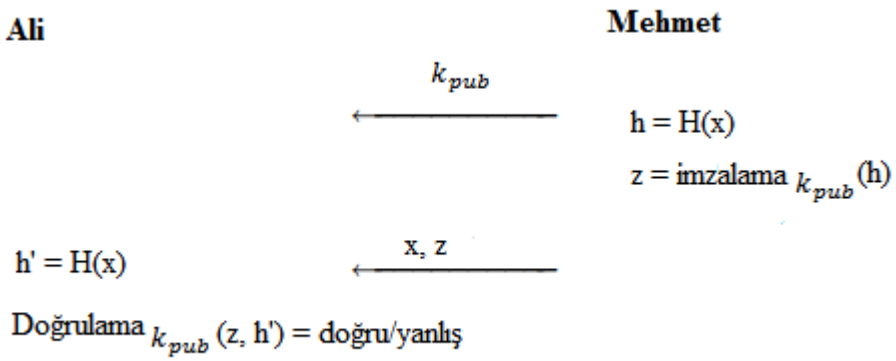
Bu sebeplerden dolayı, herhangi bir mesaj için küçük boyutlu ama bütün mesajı temsil edebilen bir imzalama işlemi olması istenir. Bu küçük boyutlu ama bütün mesajı temsil etmeyi sağlayan yapılar özet fonksiyonlarıdır. Eğer mesajın karakteristiğini hesaplayan bir özet fonksiyonu kullanılırsa sayısal imzalama işlemi şekil 2. 5.'de olduğu gibi hesaplanabilir.



Şekil 2.5. Özet fonksiyonlarıyla mesaj imzalama yapısı

Özet fonksiyonuyla gerçekleştirilen sayısal imza yapısının bütünlük kontrolü ve kimlik doğrulamada kullanımı Şekil 2. 6.'da verildi. Mehmet ismindeki gönderici sayısal imza yapısı kullanılarak imzalanmış metni, mesajı alacak kişi olan Ali'ye gönderiyor.

Mehmet x mesajının özeti olan h özet değerini kendi özel anahtar değeriyle imzalar. Alıcı tarafta, Ali alınan x mesajının h' özet değerini hesaplar daha sonra Mehmet'in genel anahtarıyla mesajı doğrular. Bu noktada şu belirtmelidir ki imzalama ve doğrulama işlemleri mesajın kendisinden ziyade özet değeri üzerinde yapılır. Çünkü özet değeri bütün mesajı temsil etmektedir ve her mesaj için özel ve tek olmaktadır. Bundan dolayı, özet değeri bazen mesajın parmak izi olarak da ifade edilir.



Şekil 2.6. Özet fonksiyonu ile sayısal imza için basit bir gösterim

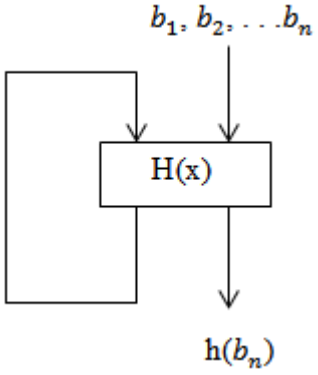
2.1.3. Diğer Uygulamalar

Özet fonksiyonları tek yönlü şifre dosyalarının oluşturulmasında yaygın olarak kullanılır. İşletim sistemlerinde şifrelerin kendilerinden ziyade özet değerleri saklanır. Böylece, şifre dosyalarına erişilse bile gerçek şifreler tekrar üretilemez.

Özet fonksiyonları izinsiz giriş algılama ve virüs algılamada da kullanılır. Bir sistemde bütünlük kontrolü için özet fonksiyonlarının kullanımı ele alınırsa şöyle bir uygulama yapılabilir. Sistemdeki her dosyanın özet değeri hesaplanır ve güvenli şekilde saklanır. Eğer dosya değiştirilmişse özet değeri yeniden hesaplanarak bu değişim fark edilebilir. Saldırganın özet değerini değiştirmeden dosyaları değiştirmesi gerekmektedir [13].Özet fonksiyonlarının diğer bir kullanım alanı ise sözde rastgele sayı üretimidir [13].

2.2. Özet Fonksiyonlarına Genel Bakış

Daha önce de ifade edildiği gibi, özet fonksiyonları rastgele uzunluktaki bir mesajı giriş olarak alır ve sabit uzunlukta bir çıkış değeri üretir. Bu fonksiyonların uygulanabilmesi için ilk olarak giriş verisi eşit uzunluktaki bloklara ayrılır. Bu bloklar temelinde sıkıştırma fonksiyonlarının bulunduğu özet fonksiyonları tarafından sırasıyla işlenir. Tekrarlı özet fonksiyonları olarak da adlandırılan bu yapılar Merkle tarafından önerilmiştir ve Merkle-Damgard yapısı olarak bilinir [14]. Günümüzde yaygın olarak kullanılan, SHA dâhil, birçok özet fonksiyonu bu yapıya dayanmaktadır. Giriş mesajının değeri sıkıştırma fonksiyonunun son döngüdeki çıkış değeri olarak tanımlanır. Bu yapı Şekil 2. 7.'deki gibi tanımlanabilir.



Şekil 2.7. Merkle-Damgard fonksiyon yapısı

Tekrarlı bir kullanıma dayanan Merkle-Damgard yapıları temel olarak iki sınıfa ayrılır.

1. Adanmış (Dedicated) özet fonksiyonları, özetleme işlemi yapmaları için özel olarak tasarlanmış algoritmalardır.
2. Blok şifre temelli özet fonksiyonları, bu özet fonksiyonları AES gibi blok şifre yapılarının kullanımına dayanır.

2.2.1. Adanmış Özet Fonksiyonları

Bu fonksiyonlar genel olarak özetleme işlemi yapmak için özel olarak tasarlanmış algoritmalardır. MD4 bu türün en temel fonksiyonudur. Oldukça yenilikçi bir fikir olan bu fonksiyon Ronald Rivest tarafından geliştirilmiştir. Ayrıca bu fonksiyon orijinal, etkin ve etkili yazılım uygulamalarına olanak tanıyan bir yapıya sahiptir [12]. Bu fonksiyonda bütün işlemler AND, OR, XOR gibi bit düzeyindeki Boolean işlemlerine dayanır [12].

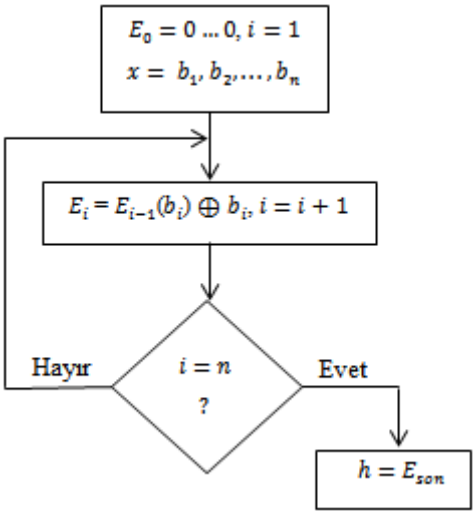
MD5, SHA ailesi ve RIPEMD ailesi gibi birçok özet fonksiyonu MD4 fonksiyonundan türetilmiştir. Dolayısıyla bu fonksiyonlar benzer yapılar içerir ve aynı temellere dayanır [15]. MD4 fonksiyonundan türetilen fonksiyonlar içinde en çok kullanılanlar SHA ailesidir. Adanmış özet fonksiyonlarının daha iyi anlaşılabilmesi için SHA ailesi bölüm 2. 3.'de ayrıntılı bir şekilde incelenmiştir.

2.2.2. Blok Şifreleme Özet Fonksiyonları

Blok şifreleme özetleme fonksiyonları, blok şifreleme yapılarının, özet değeri üretecek şekilde kullanımına dayanan yapılardır. Genellikle blok şifreleme özet fonksiyonları kullanılarak oluşturulan özet değeri boyutu, bu fonksiyonlar kullanılarak oluşturulan şifreli metin bloğuyla aynıdır. Örneğin 128 bitlik metin şifreleyen AES uygulamasında 128 bitlik özet değeri üretilir. Bu yapılarda da adanmış özet fonksiyonlarında olduğu gibi giriş mesajı bloklara ayrılır. Daha sonra mesaj blokları şifreleme fonksiyonu tarafından sırasıyla işlenir. En son bloğun işlenmesiyle özet değeri oluşturulur.

Blok şifre özet fonksiyonu gerçekleştirmenin oldukça değişik yolları bulunmaktadır. Örnek bir blok şifreleme özet fonksiyonu Şekil 2. 8.'de verilmiştir. Bu değişik yapıların her biri için temel olarak E_0 gibi bir başlangıç değerinin atanması

gerekmektedir. Bu değer sıfır gibi genel bir değer olabileceği gibi daha karmaşık değerlerde olabilir. Bu yapıda giriş mesajı b blok boyutunda alt parçalara ayrılır. E şifreleme yapısının başlangıç değeri sıfır olarak alınır. Daha sonra b_i blok parçaları E blok şifreleme yapısının bir önceki değeriyle XOR'lanır ve yeni elde edilen değer E şifreleme yapısı tarafından işlenir. Bütün bloklar işlendikten sonra en son blok için hesaplanan çıkış değeri bütün mesajın özetini oluşturur.



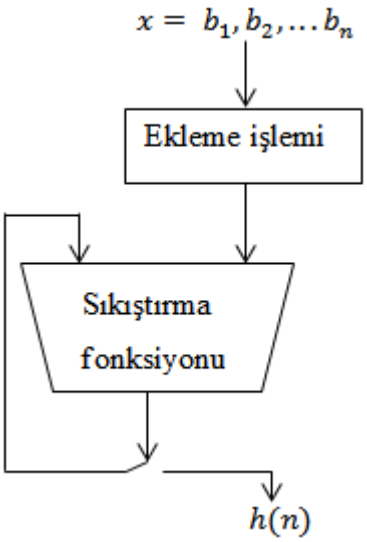
Şekil 2.8. Blok şifreleme yapısıyla özet fonksiyonu tasarımı

Uygulamalarda farklı seviyede güvenlik gerektiği için ihtiyaç duyulan seviyede güvenlik sağlayan blok özet fonksiyonları kullanılır. Örneğin sadece tek yönlülük ve zayıf çakışma dayanıklılığının gerektiği durumlarda AES'in 128 bitlik blok şifreleme fonksiyonu kullanılabilir. Bu fonksiyon 128 bitlik bir özet değeri üreteceği için bu iki tehdide karşı yeterli seviyede güvenlik sağlar. Bununla birlikte, çakışma dayanıklılığına karşı güvenlik gerektiren birçok blok şifreleme yapısında bu seviyede bir güvenlik yeterli olmamaktadır. Çünkü doğum günü çelişkisi problemi olarak ifade edilen ve bölüm 2. 4. 2. 3.'de açıklanan matematiksel bir yaklaşımla 128 bitlik güvenlik seviyesi 64 bite inmektedir. Bu yöntemle özet fonksiyonunun kaba kuvvet saldırılarına karşı sağladığı güvenlik seviyesi özet değerinin yarısı kadar olduğu ispatlanmıştır [12]. Bu seviyede bir güvenlik ise, gelişen teknolojiyle, saldırganlar için hesaplanabilir bir seviyedir. Dolayısıyla çakışma dayanıklılığına karşı güvenlik gerektiren uygulamalarda ya AES'in 192 veya 256 bitlik çıkışları kullanılabilir yada Hirrose yapısı olarak bilinen ve birden fazla blok özet fonksiyonunun beraber kullanımına dayanan yapılar kullanılabilir [12].

2.3. Güvenli Özetleme Algoritmaları

Güvenli Özetleme Algoritmaları (SHA), MD4 mesaj özetleme algoritmalarının son zamanlarda en yaygın kullanılan türevleridir. Bu algoritmalara karşı çeşitli saldırılar yapılmasına karşın özellikle SHA-2 gibi son türevleri hala yaygın olarak kullanılmaktadırlar. SHA algoritmaları temelde Merkle-Demgard algoritmasına dayanmaktadır [15].

SHA algoritmalardaki sıkıştırma fonksiyonunun blok şifreleme yapısı gibi çalışmaktadır. Şöyle ki bir önceki fonksiyonun özet değeri olan h_{i-1} ve anahtar değeri algoritmanın giriş değerini oluşturmaktadır. Şekil 2. 9.'da SHA algoritmasının yapısı gösterilmiştir.



Şekil 2.9. SHA algoritmalarının genel yapısı

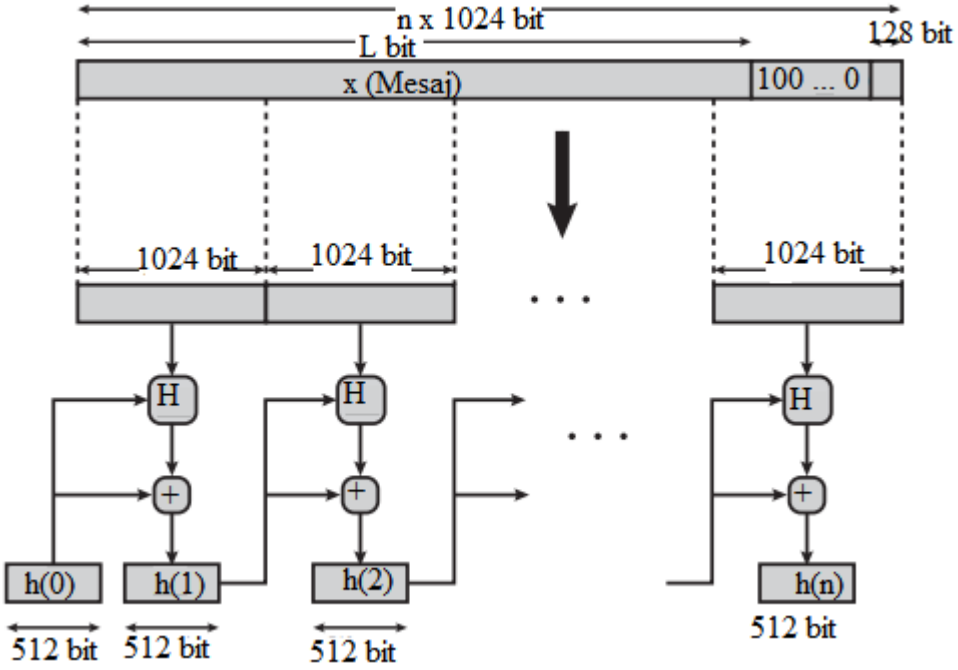
SHA, NIST (National Institute of Standard and Technology) ve FIBS (Federal Information Processing Standart) tarafından 1993 te yayınlanmıştır. SHA-0 olarak bilinen algoritmada zayıflıklar bulununca algoritma yeniden gözden geçirilmiştir ve SHA-1 olarak adlandırılan algoritma 1995'te SHA-0 yerine kullanılmaya başlanmıştır. SHA-1 160 bitlik özet değeri üretir. Daha sonra gelişen saldırılarla birlikte SHA-1 birçok uygulamada yeterli güvenlik sağlayamadı. Yapılan araştırmalar sonucunda SHA-1 üzerinde yapılan saldırılarla algoritmanın dayanıklılığının 2^{80} den çok az olan 2^{69} olduğunu bulunmuştur [16]. Böylece NIST tarafından yeni standart değerleri üretilmiştir. Bunlar FIPS tarafından SHA-256,

SHA-384 ve SHA-512 olarak bilinen ve sırasıyla 256, 384 ve 512 bitlik özet değeri üreten yeni standartlar olarak yayınlanmıştır. Bu üç fonksiyon birlikte SHA-2 olarak adlandırılmıştır [15].

NIST, 2005 te SHA-2 algoritmasının 2010'a kadar güvenli olacağı tahmin edilmıştır. Bununla birlikte yapılan saldırılar sonucunda SHA-2 algoritmasının beklenen seviyede güvenlik sağlamadığı saplanmıştır [17]. Ayrıca SHA-2 standardı, başta SHA-1 olmak üzere MD ve RİPEMD aileleriyle benzer yapılara sahiptir [12]. Dolayısıyla adanmış özet fonksiyonlarının daha iyi anlaşılması için SHA-512 yapısı incelenmiştir.

2.3.1. SHA-512

SHA-512 algoritması MD4 ailesinin en çok kullanılan mesaj özetleme algoritmasıdır. SHA-0, SHA-1 gibi Merkle-Damgard yapısına dayanan algoritmalar, yapı ve tasarım olarak benzerlik gösterdikleri için bu ailenin en gelişmiş ve standart halini almış SHA-512 yapısı incelenmiştir. Bu yapıda giriş değeri bir önceki özet değerinin çıkışından elde edilir.



Şekil 2.10. SHA-512 algoritmasını kullanarak özet değeri üretme [13]

$+ = 2^{64}$ formunda kelime kelime toplama işlemi

SHA-512 algoritması giriş olarak 2^{128} den daha küçük bir mesaj yapılarını alır ve çıktı olarak 512 bitlik bir mesaj özet çıktısı üretir. Giriş verisi 1024 bitlik bloklara ayrılmış mesaj blokları olarak alınır ve daha sonra bu blokları sırasıyla işlenir. Şekil 2. 9.'da fonksiyonun genel yapısı gösterildi. Bu fonksiyonda gerçekleşen işlemler aşağıda adımlar halinde verildi.

1. Doldurma bitlerinin eklenmesi: Mesaj boyutunun 1024 ün katı olması için mesaja ekleme yapılmasıdır. Giriş mesajı 1024'ün katı olmadığı sürece ekleme işlemi her zaman yapılır. Bundan dolayı eklenecek bit sayısı 1 ile 1024 bit arasında değişir [13]. Ekleme yapısı bir tane '1' değeri ve bunu takip eden yeterli sayıda '0' oluşur. Bu yapı şekil 2. 10.'da gösterilmiştir.

2. Uzunluğun eklenmesi: Giriş mesajına orijinal mesajın boyutunu içeren 128 bitlik bir blok eklenir. Şekil 2. 10.'da L ile gösterilen orijinal mesajın uzunluğu giriş mesajına eklenir. İlk iki adım sonucunda giriş mesajının uzunluğu 1024 ün katı olur.

3. Eklenmiş mesajın bölünmesi: Sıkıştırma fonksiyonu uygulanmadan önce giriş mesajı 1024 bitlik alt bloklara ayrılır. Daha sonra her 1024 bitlik bloklar 64 bitlik 16 alt parçaya ayrılır. Örneğin, b mesajının i. Bloku aşağıdaki şekilde ayrılır.

$$b_i = (b_i^{(0)}, b_i^{(1)} \dots b_i^{(15)}) \qquad (2.\ 1)$$

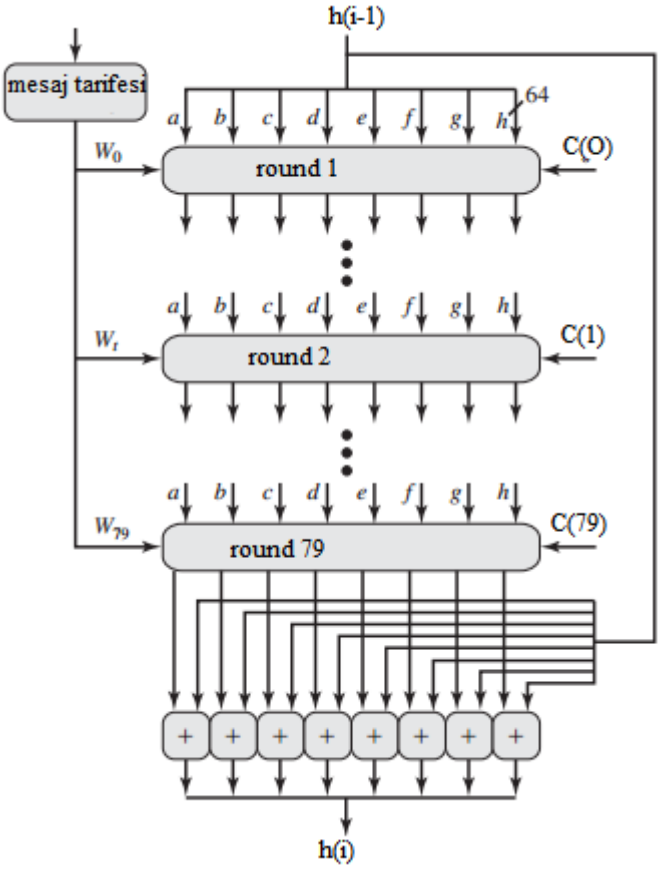
4. Başlangıç özet değerleri: SHA-512 özet fonksiyonunda ara ve sonuç değerleri Şekil 2. 10.'da görüldüğü gibi 512 bittir. Bu değerler 64 bitlik sekiz kaydedici tarafından saklanır. Bu kaydedicilere aşağıdaki başlangıç değerleri atanır. Bu değerler kelimenin en anlamlı baytının en sola yerleştirmeye dayanan big-endian formatında saklanır [13].

Tablo 2.1. SHA-512 algoritmasında başlangıç değerleri

a =	6A09E667F3BCC908	e =	510E527FADE682D1
b =	BB67AE8584CAA73B	f =	9B05688C2B3E6C1F
c =	3C6EF372FE94F82B	g =	1F83D9ABFB41BD6B
d =	A54FF53A5F1D36F1	h =	5BE0CD19137E2179

5. 1024 (128 kelime) blokların işlenmesi: Algoritmanın temeli 80 turdan oluşan modüler bir yapıdır. Ve bu yapı şekil 2. 11.'de verilmiştir.

Her bir tur hafızadaki 512 bitlik, $abcdefgh$, giriş olarak alır ve daha sonra hafızadaki bu değerleri günceller. İlk turda (round) hafızada önceden tanımlanmış değerler mevcuttur. Her t turunda 1024 bitlik mesaj bloklarından türetilen 64 bitlik W_t değerleri kullanılır. Bu değerler tanımlanmış mesaj tarifesi kullanılarak sırasıyla üretilir. Ayrıca her turda ek sabit c_t değerlerini kullanır. Bu sabitler 80 tane asal sayının küp köklerinin kesirli kısmını temsil eder. Bu değerler rastgeleliği artırır ve giriş değerinin elde edilme ihtimalini azaltır [13].



Şekil 2.11. SHA-512 algoritmasında 1024 bir bloktaki işlemler [13]

h_{i} çıkış değerinin üretilmesi için başlangıçtaki h_{i-1} hafıza değerleri ile son turdaki hafıza çıkış değerleri toplanır. Bu toplama değerleri birbirine denk düşen hafıza değerleri arasında birbirinden bağımsız şekilde gerçekleştirilir.

6. Sonuç değeri: 1024 bitlik bütün bloklar işlendikten sonra n. blokun çıkışındaki 512 bitlik değer bütün mesajın özetini oluşturur. SHA-512 algoritmasında yapılan işlemler aşağıdaki gibi özetlenebilir.

$h_0 = IV$

$h_i = SUM_{64}(h_{i-1}, abcdefgh_i)$

$h = h_n$ (2. 2)

$IV = abcdefgh$ değerlerinin başlangıç değerleridir.

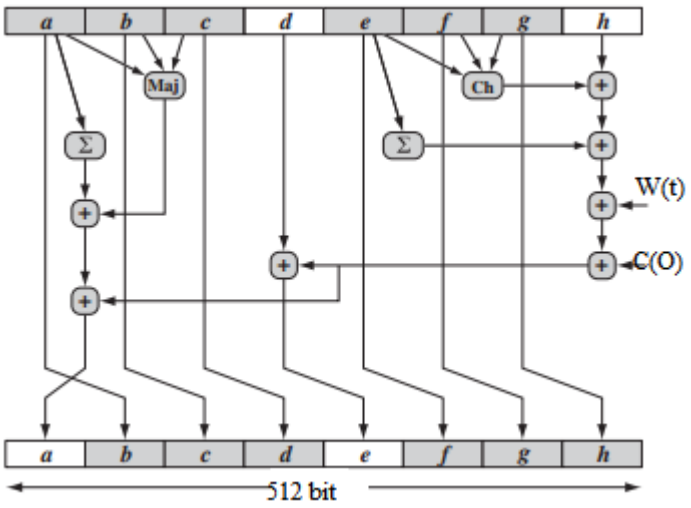
$abcdefgh_i = i$. mesaj blokunda işlenen son turun çıkış değeridir.

N = mesajdaki blok sayısıdır.

SUM_{64} = giriş çifti kelimelerinden her biri için 2^{64} katında yapılmış toplama işlemidir.

2.3.1.1. SHA-512 Tur Fonksiyonu

Her turda gerçekleşen işlemler aşağıdaki şekil 2. 11.'de gösterildi. Burada SHA-512 özet fonksiyonun her turunda gerçekleşen işlemler detaylı olarak incelendi. Bu işlemler kümesi özet fonksiyonları için çekirdek fonksiyonu ve sıkıştırma fonksiyonu olarak adlandırılır. Ayrıca her bir turda gerçekleşen işlemler aşağıdaki denklem kümesi yardımıyla tanımlandı.



Şekil 2.12. SHA-512 algoritmasında tur fonksiyonunun içyapısı [13]

$T_1 = h + Ch(e, f, g) + (\sum_1^{512}\mathrm{e}) + W_t + c_t$

$T_2 = (\sum_0^{512}\mathrm{e}) + Maj(a, b, c)$

$h = g$

$g = f$

$f = e$

$e = d + T_1$ (2. 3)

$d = c$

$c = b$

$b = a$

$a = T_1 + T_2$

Burada

t = adım sayısı; $0 \leq t \leq 79$

$Ch(e, f, g) = (e\ AND\ f) \oplus (NOT\ e\ AND\ g)$

şartlı fonksiyon eğer e doğruysa f değilse g

$Maj(a, b, c) = (a\ AND\ b) \oplus (a\ AND\ c) \oplus (b\ AND\ c)$

işlenen değerlerin çoğu (iki veya üçü) doğruysa fonksiyon doğrudur

$(\sum_0^{512}\text{a}) = ROTR^{28}(\text{a}) \oplus ROTR^{34}(\text{a}) \oplus ROTR^{39}(\text{a})$

$(\sum_0^{512}\text{e}) = ROTR^{14}(\text{e}) \oplus ROTR^{18}(\text{e}) \oplus ROTR^{41}(\text{e})$

$ROTR^n$(X) = işlenen 64 bitlik x kelimesinin n bit dairesel sağa kaydırma (döndürme)

W_t = 512 bitlik giriş blokundan türetilmiş 64 bitlik kelimeler

c_t = 64 bitlik eklenecek sabitler

+ = 2^{64} formunda toplama

Tur fonksiyonu hakkında iki gözlem yapılabilir.

1. Formül 2. 12.'de görüldüğü gibi tur fonksiyonunun çıkışındaki sekiz kelimeden altı tanesi sadece yer değiştirme işlemine tabi tutuldu.

2. Sadece iki kelime (a, e) yeniden üretilir. e kelimesi, w_t tur kelimesinin ve c_t sabitinin giriş değişkenlerinin (d, e, f, g, h) fonksiyonundan oluşur. Yine a kelimesi, w_t tur kelimesi ve c_t round sabitinin d dışındaki bütün değişkenlerinin fonksiyonu olarak elde edilir.

Burada 64 bitlik w_t kelime değerlerinin 1024 bitlik mesajdan nasıl türetildiği gösterildi. w_t tur kelimelerinin ilk 16 tanesi direk mesajdan gelen kelime değerleridir. Kalan değerlerin üretilmesi ise aşağıdaki gibidir.

$W_t = \sigma_1^{512}(W_{t-2}) + W_{t-7} + \sigma_0^{512}(W_{t-15}) + W_{t-16}$ (2. 4)

$\sigma_0^{512} = ROTR^1(X) \oplus ROTR^8(X) \oplus ROTR^7(X)$

$\sigma_0^{512} = ROTR^{19}(X) \oplus ROTR^{61}(X) \oplus ROTR^6(X)$

$ROTR^N(X)$ = işlenen 64 bitlik x kelimesinin n bit dairesel sağa kaydırma (döndürme)

$SHR^n(X)$ = işlenen 64 bitlik x kelimesinin sağına sıfır ekleyerek n bit sola döndürme

$+ = 2^{64}$ formunda toplama

Böylece ilk 16 turda kullanılan W_t değerleri mesaj blokundaki kelime değerlerine karşı düşer. Geriye kalan 64 turda ise daha önceki dört W_t değerinin kaydırma ve döndürme işlemlerine tabi tutularak XOR'lanmasıyla elde edilir. Sıkıştırılmış bloklar mesaj içerisinde bağımlılığa sebep olur şöyle ki aynı sıkıştırma fonksiyonu çıkışı verecek iki farklı mesajların bulunmasını daha da zorlaştırır.

SHA-512 algoritmasının şöyle bir özelliği vardır; özet değerindeki her bir bit değeri giriş değerindeki her bir bit değerinin fonksiyonudur. F fonksiyonunun karmaşık yapısı iyi şekilde karışmış sonuçlar üretir yani aynı düzenliliği gösterseler bile iki farklı mesajın aynı özet değerini üretmesi çok küçük bir ihtimaldir [13]. SHA-512 özet algoritmasında herhangi bir için mesaj özetinden orijinal mesajın bulunması 2^{512} işlem gerektirir. Dahası rastgele alınmış iki mesajın aynı özet değerini üretme ihtimali ise 2^{256}'dir.

2.4. Gereksinimler ve Güvenlik

2.4.1. Özet Fonksiyonlarında Olması Gereken Uygulamaya Yönelik Özellikler

Özet fonksiyonlarının güvenlik özelliklerine geçilmeden bu fonksiyonlar için olması gereken giriş çıkış değerleri incelendi. Burada ifade edilen özellikler özet fonksiyonlarının uygulamada sahip olmaları gereken özelliklerdir. Bu pratik özellikler aşağıda maddeler halinde belirtildi.

- Özet fonksiyonları rastgele uzunluktaki bir mesaja uygulanabilmelidir. Özet değeri hesaplanacak giriş verisi, günlük hayatta olduğu gibi, birkaç bayt uzunluğunda olabileceği gibi kilobayt düzeyinde de olabilmelidir. Özet fonksiyonları bu değişik boyuttaki veriler için herhangi bir gereksinime ihtiyaç duymadan uygulanabilir olmalıdır.

- Özet fonksiyonunun çıkışı sabit uzunlukta ve giriş mesajının uzunluğundan bağımsız olmalıdır. Günümüz için kullanışlı özet fonksiyonları 128 ile 512 bit arasında değişen özet değeri uzunluğuna sahiptirler. Giriş verisinin boyutu ne olursa olsun hesaplanan özet değeri, kullanılan algoritmaya bağlı olarak, belirtilen aralıkta olmalıdır.
- Özet fonksiyonlarının etkili bir hesaplama yapması gerekir. Eğer yüzlerce megabaytlık uzun bir mesajın özeti çıkarılacaksa bunu hesaplamak göreceli olarak daha hızlı olmalıdır.
- Hesaplanan özet değeri büyük ölçüde bütün giriş bitlerine duyarlı olmalıdır. Yani; çıkıştaki bir bit girişteki bitlerin birçoğunun işlenmesi sonucunda oluşmalıdır. Örneğin; eğer giriş mesajında ufak bir değişiklik yapılırsa özet değeri bu değişiklikten oldukça fazla etkilenmelidir [13].

2.4.2. Özet Fonksiyonlarının Güvenlik Gereksinimleri

Daha önce ifade edildiği gibi, diğer bütün şifreleme algoritmalarının aksine özetleme fonksiyonlarında anahtar değerleri yoktur. Acaba bir özet fonksiyonlarının güvenli olabilmesi için bazı özel nitelikler gerekli midir? Aslında özet fonksiyonlarının uygulamalar üzerinde herhangi bir etkisi olup olmadığını düşünülebilir, çünkü bu yapılar şifreleme işlemi yapmıyor ve bir anahtar değeri de yoktur. Şifrelemede olduğu gibi, böyle şeyler aldatıcı olabilir ve saldırganlar özet fonksiyonlarının zayıflıklarını kullanabilir. Bundan dolayı, özet fonksiyonlarının güvenliğinin iyi bir şekilde değerlendirilebilmesi için bu fonksiyonlarla ilgili incelenmesi gereken üç önemli özellik aşağıda verilmiştir.

1. Öngörme dayanıklılığı veya Tek yönlülük (preimage resistance or one wayness)
2. Zayıf çakışma dayanıklılığı (second preimage resistance or weak collision resistance)
3. Çakışma dayanıklılığı (collision resistance or strong collusion resistance)

Özet fonksiyonlarının iyi anlaşılabilmesi için bu üç özellik detaylarıyla incelenmelidir. Genellikle şifreleme özet fonksiyonlarının gerekli özellikleri arasında yer almamasına rağmen sözde rasgelelikte (pseudorandomnes) diğer bir güvenlik parametresidir. Özet fonksiyonları genellikle anahtar üretiminde ve yalancı rastgele sayı üretiminde kullanılmaktadır. Ayrıca mesaj bütünlüğü uygulamaları ve yukarıda belirtilen üç güvenlik özelliği mesajın özet değerinin rastgele görünmesine bağlıdır.

2.4.2.1. Tek Yönlülük

Özet fonksiyonlarının tek yönlü olması gereken yapılardır yani verilen bir h özet değerinden fonksiyonun uygulandığı giriş mesajı değerinin hesaplanması mümkün olmamalıdır. Başka bir deyişle, verilen özet değerinden bu özet değerini oluşturan mesajın çıkartılamaması gerekir. Bu özellik özet fonksiyonlarında olması gereken en önemli güvenlik parametresidir çünkü sayısal imza gibi birçok yapı temel olarak özet fonksiyonlarına dayanmaktadır. Bundan dolayı, eğer özet fonksiyonları tek yönlü yapılar olmazsa özet değerinden orijinal mesaj çıkarılabilir. Buda sayısal imza, mesaj doğrulma kodu gibi özet fonksiyonlarını temel alan bütün yapılarda önemli bir güvenlik zafiyetidir.

Özet fonksiyonlarında tek yönlülüğün önemi bir örnekle daha iyi anlaşılabilir. Örneğin, gönderilecek mesaj, imza kısmı hariç, şifrelensin daha sonra şifreli metin ve sayısal imza ikilisi iletilsin.

$$(E_k(\mathrm{x}), imza_{k\ prB}(h)) \tag{2. 5}$$

Burada $E_k()$ AES gibi iki tarafında aynı anahtarı paylaştığı bir simetrik şifreleme yapısıdır. Göndericinin RSA sayısal imzalama yapısını kullandığını farz edilirse sayısal imza değeri şu şekilde hesaplanır.

$$z = imza_{k\ prB}(h) \equiv h^D \ mod\ n \tag{2. 6}$$

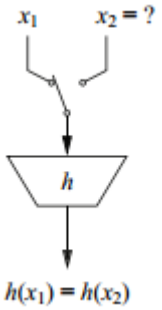
Saldırgan bu hesaplamayı yapmak için göndericinin genel anahtarını kullanabilir.

$$s^E \equiv z\ mod\ n \tag{2. 7}$$

Eğer saldırgan sayısal imzadan giriş metnini çıkarabiliyorsa metnin şifrelenmiş olmasının da çok anlamı yoktur çünkü mesaj sayısal imzadan üretilebilmektedir. Görüldüğü gibi özet fonksiyonlarının en önemli güvenlik parametresi tek yönlülüktür. Bununla birlikte, anahtar üretimi gibi özet fonksiyonlarının kullanıldığı birçok uygulamada özet fonksiyonlarının tek yönlülük özelliği daha çok öne çıkmaktadır.

2.4.2.2. Zayıf Çakışma Dayanıklılığı

Özet fonksiyonlarının diğer bir güvenlik parametresi zayıf çakışma dayanıklılığıdır. Bu yapılarda iki farklı mesajın aynı özet değerini vermemesi gerekir. Yani belirli bir x_1 mesajı için üretilen h_1 değeri için $x_1 \neq x_2$ olacak şekilde seçilen farklı bir mesajın $h_1 = H(x_1) = H(x_2) = h_2$ olacak şekilde aynı özet değerini üretmesinin hesaplama olarak imkânsız olması istenir. Bu yapı şekil 2. 13.'te gösterildi.

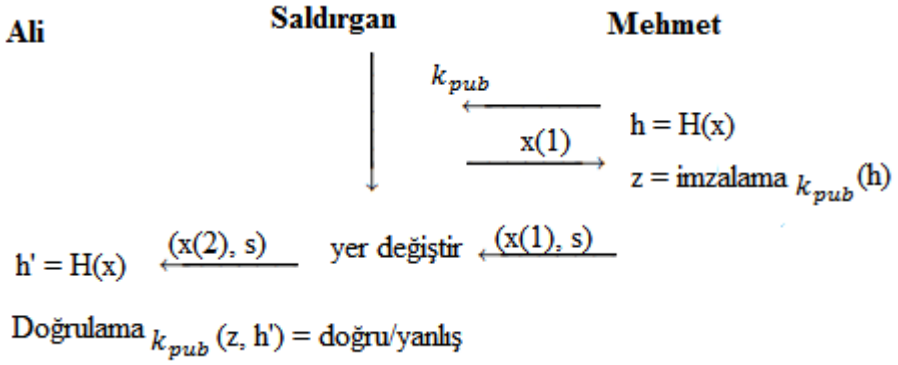


Şekil 2.13. Zayıf çakışma dayanıklılığı

Özet fonksiyonlarında olan iki farklı çakışma türünü birbirinden ayırmalıdır. Zayıf çakışma dayanıklılığı olarak ifade edilen bu çakışmada x_1 giriş verisi ve bu verinin özet değeri hesaplanır daha sonra aynı özet değerini veren x_2 mesajı bulunmaya çalışılır. Çakışma dayanıklılığında ise x_1 ve x_2 gibi iki farklı mesaj alınır ve bunların çakışmasıyla ilgilenilir.

Sayısal imza gibi özet fonksiyonlarının kullanıldığı yapılarda zayıf çakışma dayanıklılığı oldukça önemlidir. Örneğin, eğer saldırgan aynı özet değerine sahip başka bir mesaj elde ederse şekil 2. 14.'de gösterildiği gibi bir saldırı yapabilir.

Şekil 2. 14.'de görüldüğü gibi alıcının hesapladığı doğrulama işlemi 'doğru' ifadesini verdiği için alıcı x_2'yi doğru mesaj olarak kabul edecektir. Aslıda farklı mesajlar olmalarına rağmen özet değerleri aynı olduğundan doğrulama işlemi ikinci mesajı da doğru olarak kabul eder. Bundan dolayı saldırganın aynı özet değerini veren ikinci bir mesaj bulmaya çalışması bu saldırıya iyi bir örnek olarak gösterilebilir.



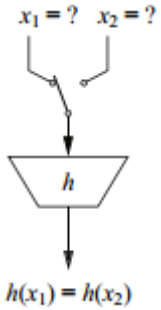
Şekil 2.14. Özet fonksiyonlarında zayıf çakışma dayanıklılığı saldırı örneği

Şimdi ele alınacak bir diğer konu saldırganın aynı özet değerine sahip ikinci mesajı bulması nasıl önlenebilir. İdeal durumda zayıf çakışmanın olmadığı bir özet fonksiyonunun olması istenir. Fakat uygulamada, idealde olması beklenen durum olmamaktadır. Şöyle ki özet fonksiyonlarını çıkışı belli uzunluktadır yani sınırlı bir uzaydır oysa sonsuz tane giriş verimiz olabilmektedir. Bu durumda özet değerlerinde kesinlikle çakışma olacağı söylenebilir. Bu durumu daha somut olarak güvercin deliği ilkesiyle açıklanabilir. Güvercin deliği ilkesi şu şekilde ifade edilebilir. Eğer 100 güvercinin 99 deliğe girmesi gerekiyorsa en ez bir deliğe iki güvercin girmiş olmalıdır. Tabi ki en iyi ihtimalle olması beklenen bu durum her zaman gerçekleşmeyebilir. Daha açık bir ifade ile bazı yuvalar boş kalırken bazı yuvalarda birden fazla güvercin bulunabilir. Bu yaklaşım özet fonksiyonları içinde geçerlidir. Şöyle ki, her özet fonksiyonunun çıkışı sabit ve belli bir uzunlukta olduğu için n bitlik çıkışa sahip bir özet fonksiyonu 2^n farklı çıkış değeri olabilir. Giriş değerleri sonsuz sayıda olduğundan çakışma olacaktır. Pratikte her çıkış değeri rastgele bir giriş değerine denk gelir. Dolayısıyla birden çok giriş değeri aynı çıkış değerini verir [12].

Bununla birlikte bu çakışmanın sadece teoride olduğu, pratikte olmadığı garanti edilebilir. Şöyle ki, güçlü özet fonksiyonları öyle düzenlenmelidir ki verilen bir x_1 ve bu mesaj için üretilen h_1 değeri için $h_1 = h_2$ olacak şekilde ikinci bir x_2 mesajının oluşturulması imkânsız olmalıdır. Bu durumda Şekil 2. 14.'tekine benzer herhangi bir saldırı yapılamaz. Ancak rasgele mesajlar alıp bunların özet değerlerini hesaplanır ve bunların $h(x_1)$'le eşitliğine bakabilir. Günümüz bilgisayarlarıyla bu saldırıları engellemek için özet fonksiyonu çıkışının n=80 bit olması yeterlidir. Fakat ileride ifade edildiği gibi daha güçlü saldırılardan dolayı bu uzayı daha büyük tutmak gerekmektedir.

2.4.2.3. Çakışma Dayanıklılığı

Özet fonksiyonlarının diğer bir güvenlik parametresi çakışma dayanıklılığıdır. Burada rastgele seçilen $x_1 \neq x_2$ gibi iki farklı giriş mesajı için $h_1 = h_2$ olacak şekilde aynı özet değeri üretilmemelidir. Burada iki metninde seçimi serbest olduğundan çakışma ihtimali daha zor gibi gözükmektedir. Çakışma dayanıklılığını, zayıf çakışma dayanıklılığından ayıran en önemli özellik aynı özet değerlerinin bulunması için iki mesajında değişebilir olmasıdır. Bu yapı Şekil 2. 15.'de verildi.



Şekil 2.15. Çakışma dayanıklılığı

Eğer aynı özet değerine sahip iki farklı mesaj elde edilirse Şekil 2. 14.'teki gibi bir saldırı yapılabilir. Burada alıcının hesapladığı doğrulama işlemi 'doğru' ifadesini verdiği için alıcı x_2'yi doğru mesaj olarak kabul eder. Aslıda farklı mesajlar olmalarına rağmen aynı özet değerlerini ürettikleri için doğrulama işlemi ikinci mesajı da doğru olarak kabul eder. Bundan dolayı aynı özet değerini veren iki farklı mesaj için böyle bir saldırı yapılabilir.

Güvercin deliği probleminde de belirtildiği gibi çakışma her zaman olur önemli olan bu ihtimalin ne kadar zor olduğudur. Bu saldırının zayıf çakışma dayanıklılığından daha zor olduğunu düşünülebilir. Yani özet fonksiyonunun çıkışı 80 bitse 2^{80} mesajı kontrol etmek gerektiği gibi. Fakat yapılan çalışmalar gösteriyor çakışmanın bulunması için 2^{40} mesajın kontrol edilmesi yeterlidir. Bu şaşırtıcı sonuç doğum günü saldırısı olarak ifade edilen matematiksel bir yaklaşımla temellendirilmektedir. Bu saldırı aslında şifre analizinde kullanılan güçlü bir yöntem olan 'doğum günü çelişkisi' yöntemine dayanmaktadır [12].

Özet fonksiyonlarındaki bu tür çakışmayı daha somut hale getirmek için, bu fonksiyonlarındaki çakışmaya oldukça benzer bir problem ele alındı. Örneğin bir partide en

az iki insanın aynı günde doğmuş olma ihtimalinin makul bir değerde olması için en az kaç insana ihtiyaç vardır? Bir yılda 365 gün olduğundan bir çakışmanın olabilmesi için sezgisel olarak 183 kişi gerekli olduğunu sanılabilir. Fakat çok daha az kişinin yeterli olduğu şu şekilde bir yaklaşımla kolaylıkla görülebilir. Öncelikle bu kişilerin farklı günlerde doğma ihtimali bulunur daha sonra bu değer 1 den çıkartıldığında çakışma olasılığı bulunmuş olur. Bir kişi için çakışmama ihtimali 1'dir. İkinci kişi için 365 üzerinden 364 tür. Bu durum aşağıdaki şekilde formüle edilebilir.

$$\text{P(iki kişi arasında çakışma yok)} = \left(1 - \frac{1}{365}\right) \tag{2. 8}$$

Üç kişi için çakışma olmama ihtimali:

$$\text{P(üç kişi arasında çakışma yok)} = \left(1 - \frac{1}{365}\right) . \left(1 - \frac{2}{365}\right) \tag{2. 9}$$

t kişi için:

$$\text{P}(t \text{ kişi arasında çakışma yok}) = \left(1 - \frac{1}{365}\right) . \left(1 - \frac{2}{365}\right) \ldots \left(1 - \frac{t}{365}\right) \tag{2. 10}$$

$t = 366$ olduğunda çakışma ihtimali 1 olacaktır çünkü sadece 365 gün vardır. Şimdi tekrar başlangıçtaki soru incelenecek olursa: çakışma ihtimalinin 50% olması için kaç insana ihtiyaç duyulur. Denklemden de görüleceği gibi çakışma ihtimalinin %50 olması için 23 kişi yeterlidir. Bu yaklaşım denklemlerle aşağıdaki gibi ifade edilebilir.

$$\text{P(en az bir çakışma)} = 1 - \text{P(çakışma yok)} \tag{2. 11}$$
$$= \left(1 - \frac{1}{365}\right) . \left(1 - \frac{2}{365}\right)$$
$$= 0.507 \approx 50\%.$$

Denklem yardımıyla görülür ki 40 kişi için bu ihtimal 90%'dır. Bu örnek denemenin şaşırtıcı sonuçları doğum günü çelişkisi olarak adlandırılmaktadır.

Özet fonksiyonlarındaki çakışma partiye katılan kişiler arasında doğum günlerinin çakışmasını bulma problemine oldukça benzerdir. Fakat özet fonksiyonlarında işlem uzayı 365 değil, n bit çıkışlı bir özet fonksiyonu için 2^n'dir. Özet fonksiyonlarının güvenliği için

n değeri önemli bir parametredir. Peki, saldırganın $H(x_i) = H(x_j)$ olan aynı özet değerine sahip iki farklı mesajın çakışma ihtimalinin makul bir değerde olması için kaç tane mesajın özetini almalı? t adet özet değeri arasında çakışmanın olmama ihtimali aşağıdaki gibidir.

P(üç kişi arasında çakışma yok) = $(1 - \frac{1}{2^n})$. $(1 - \frac{2}{2^n})$ . . . $(1 - \frac{2}{2^n})$

$= \prod_{i=1}^{t-1} (1 - \frac{i}{2^n})$ (2. 12)

Aşağıdaki yaklaşımı matematiksel olarak ifade edilebilir:

$$e^{-y} = 1 - y \quad (2.\ 13)$$

i / 2^n << 1 olmaktadır. Olasılık değeri şuna yaklaşmaktadır.

P(çakışma yok) = $\prod_{i=1}^{t-1} e^{-\frac{i}{2^n}}$

$= e^{-\frac{1+2+3+\ldots+t-1}{2^n}}$ (2. 14)

Aritmetik seri şuna denktir: 1+2+…+t-1= t.(t-1)

P(çakışma yok) = $e^{-\frac{t(t-1)}{2.2^n}}$ (2. 15)

Burada amaç çakışma bulabilmek için ne kadar mesajın gerektiğini bulmaktır. Bundan dolayı denklem t için çözülmelidir. λ = 1 – P(çakışma yok) denklemi ile en azından bir çakışma olduğu gösterilirse.

$\lambda \approx 1 - e^{-\frac{t(t-1)}{2^{n+1}}}$

$\ln(1 - \lambda) \approx -\frac{t(t-1)}{2^{n+1}}$

$t\,(t - 1) \approx 2^{n+1} \ln(\frac{1}{1-\lambda})$ (2. 16)

Pratikte t >>1 olduğu için $t^2 = t\,(t - 1)$ olarak alınır.

$t \approx \sqrt{2^{n+1} \ln(\frac{1}{1-\lambda})}$

$t \approx 2^{(n+1)/2} \sqrt{\ln(\frac{1}{1-\lambda})}$ (2. 17)

Yukarıdaki denklem oldukça önemlidir çünkü bu denklem özet çıkış uzunluğu n bit olan ve çakışma olasılığı λ olan arasında bir çakışma için ne kadar mesaj özetine ihtiyaç duyulduğunu gösterir. Doğum günü saldırısından çıkarılacak en önemli sonuç bir çakışma bulmak için ihtiyaç duyulan özet sayısı yaklaşık olarak mümkün özet çıkışı sayısının karekökünе eşittir. Buda şöyle ifade edilebilir: $\sqrt{2^n} = 2^{n/2}$. Bundan dolayı x bitlik güvenlik seviyesi için özet fonksiyonu $2x$ bitlik çıkış değerine ihtiyaç duyar. Örnek olarak; 80 bitlik çıkışı olan özet fonksiyonu için çakışma değerini bulmak istediğimizi varsayalım. 50% lik başarı ihtimali için özet fonksiyonu $t = 2^{81/2}\sqrt{\ln(1/(1-0{,}5))} \approx 2^{40{,}2}$ adet giriş gereklidir [12]. Çakışma için yaklaşık olarak 2^{40} özet değerini hesaplamak ve kontrol etmek günümüz bilgisayarlarıyla mümkündür. Doğum günü çelişkisine dayanan çakışma saldırılarını önlemek için özet fonksiyonlarının çıkış uzunluğu sadece zayıf çakışma saldırılarına karşı koruma sağlayan çıkış değerlerinin iki katı olmalıdır. Bu sebeple yeni özet fonksiyonları daha uzun çıksa sahip olmakla birlikte bütün özet fonksiyonları en az 128 bitlik çıkış uzunluğuna sahip olmalıdır. İlginçtir ki, matematiksel çıkarımlarla başarı olasılığı 0.5 ve 0.9 arasında küçük fark bulunmaktadır [12].

Doğum günü saldırısı, özet fonksiyonları için genel bir kaba kuvvet saldırısıdır çünkü algoritmanın zayıflıklarına dayanmamaktadır ve kaba kuvvet saldırısında olduğu gibi sadece olası değerleri denemektedir. Dolayısıyla, bu saldırının herhangi bir özet fonksiyonuna karşı yapılabileceği anlamına gelir. Diğer yandan verilen bir özet fonksiyonu için uygun olan en güçlü saldırının bu saldırı olduğu aşikârdır çünkü deneme sayısı en azdır. Bununla birlikte popüler bazı özet fonksiyonları için, özellikle MD5 ve SHA-1 te, doğum günü saldırılarından daha hızlı olan matematiksel çakışma saldırıları vardır [16].

Diğer yandan mesaj saklama gibi birçok özet uygulaması için sadece öngörme dayanıklılığının gerektiği belirtilmelidir. Bundan dolayı 80 bit gibi göreceli küçük özet fonksiyonu çıkışı değeri çakışma saldırısının olmadığı durumlarda yeterli olabilir.

2.5. Özet Fonksiyonlarına Karşı Yapılan Saldırılar

Temel olarak özet fonksiyonlarına da düzenlenen iki tür saldırı vardır. Bunlar kaba kuvvet saldırıları ve algoritma zayıflıklarına dayanan sezgisel saldırılardır. Özet fonksiyonlarında kaba kuvvet saldırısı algoritmanın özelliklerine bağlı değildir sadece özet değerinin bit uzunluğuna bağlıdır. Fakat sezgisel yöntemler ise kullanılan özetleme algoritmasının zayıflıklarına dayanır.

Kaba kuvvet saldırıları genel olarak bütün olası uzayın araştırılmasına dayanmaktadır. Özet fonksiyonları için bu uzay özet değerinin boyutu olmaktadır. Dolayısıyla özet fonksiyonlarının bu saldırılara karşı güvenli olabilmesi için çıkış değerinin uzunluğu uygun bir şekilde belirlenmelidir çünkü doğum günü çelişkisi gibi yaklaşımlarla güvenlik seviyesi özet fonksiyonu çıkışının yarısı kadar olmaktadır. Güvercin deliği ilkesi ve doğum günü çelişkisi yöntemlerinde kaba kuvvet saldırıları detaylı bir şekilde incelendi. Bu yöntemlerle kaba kuvvet saldırıları matematiksel bir temele dayandırıldı.

Sezgisel yöntemler ise bütün anahtarlar uzayını denenmesinden ziyade kullanılan özet algoritmanın bazı zayıflıkları kullanmaya dayanır. Bu yöntemlerin başarılı olarak kabul edilebilmeleri için kaba kuvvet saldırılarına göre daha iyi sonuç vermeleri gerekmektedir. Dolayısıyla, ideal bir özet fonksiyonunda için sezgisel yöntemlerinin gücü kaba kuvvet saldırılarının gücüne eşit veya daha büyük olmalıdır [13]. Örneğin SHA-1 algoritmasının özet değeri 160 bittir. Dolayısıyla çakışma dayanıklılığına karşı sağladığı güvenlik seviyesi 2^{80}'dir. Fakat sezgisel yöntemlerle yapılan saldırılar sonucu bu seviye 2^{69} olark bulunmuştur [16]. Benzer şekilde MD5 özet fonksiyonu 128 bitlik özet değeri üretir yani çakışma saldırılarına karşı 2^{64} seviyesinde güvenlik sağlaması gerekir. Fakat yapılan saldırılarla bu seviye 2^{33} alarak belirlenmiştir [18].

3. KECCAK ÖZETLEME ALGORİTMASI

2013 yılı itibariyle, SHA-1 algoritması bile birçok uygulamada hala kullanılmaktadır. Dolayısıyla çoğu uygulama için yeterli güvenlik seviyesine sahiptir. Dahası, SHA-2 algoritması SHA-1 algoritmasından çok daha iyi bir güvenlik sağlamaktadır ve herhangi bir yöntemle hala kırılamamıştır. Temel olarak SHA-2 ve SHA-1 özetleme algoritmaları, MD4 ve SHA-0 algoritmalarına benzer yapılar ve temel matematik işlemler kullanmaktadır [12]. MD4 ve SHA-0 algoritmaları kırıldığından benzer yapılar içeren diğer özetleme algoritmaların da güvenli olmadığı göz önüne alınmaktadır. Dolayısıyla bu algoritmalara alternatif yeni bir standardın belirlenmesi kararlaştırılmıştır. 12 Ekim 2012'de Keccak algoritması NIST tarafından yeni standart olarak belirlenmiştir [19].

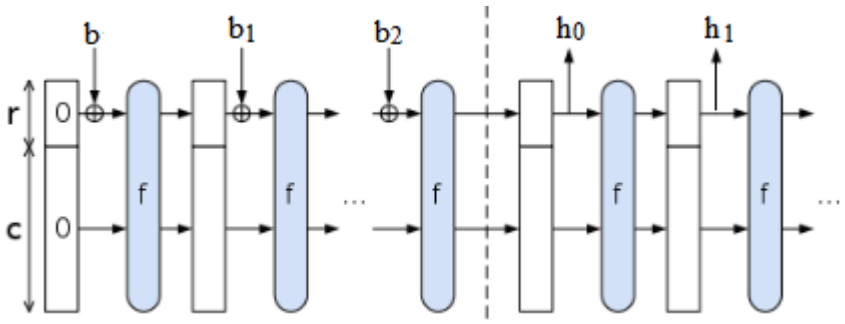
Keccak algoritması Guido Bertoni, Joan Daemen, Michaël Peeters, ve Gilles Van Assche tarafından tasarlandı ve Ekim 2012 de yeni özet algoritmaları standardı olarak ilan edildi. Bu algoritma yaklaşık dört yıl süren ve NIST tarafından düzenlenen bir yarışma sonucunda belirlendi. Keccak algoritması yarışmaya katılan 51 algoritma arsında üçüncü tur itibariyle son beş algoritma arasında yer almıştır [7]. Bu algoritmaların arasında değerlendirilme kriterleri güvenlik, esneklik ve hız gibi parametrelerdir. Son beş algoritma arasında ise en hızlı olanı Keccak algoritması olarak belirlenmiştir [20].

Keccak özetleme algoritması sünger (sponge) yapılarına dayanmaktadır [21]. Bu yapılarda giriş mesaj blokları önce başlangıç değeriyle XOR'lanır. İlk blok için başlangıç değeri sıfır olarak alınır, diğer bloklar için bu değer bir önceki f fonksiyonunun çıkışıdır. f fonksiyonu Keccak yapısının çekirdek fonksiyonudur. Şifrelemede, sünger işlevi veya sünger yapıları giriş olarak herhangi bir uzunlukta bir giriş bit dizisi alır ve istenilen uzunlukta bir çıkış üretmek için kullanılır [22]. Bu yapılar sonlu alanların bir sınıfıdır. Sünger fonksiyonları kimlik doğrulama kodları, akış şifreleme, blok şifrelere, rasgele sayı üreteçleri gibi farklı kullanım alanlarına sahiptir.

3.1. Blok Yapısı

Keccak özet algoritmasının genel yapısı Şekil 3. 1.'de gösterilmiştir. Burada r ve c değerleri v giriş vektörünü oluşturmaktadır. r, f fonksiyonunda kullanılan blok boyutunu göstermektedir ve çıkış değerine bağlıdır. Sünger yapılarında kullanılan c değeri kapasite

olarak ifade edilir ve güvenlik seviyesini belirlemeyi sağlayan parametresidir. Güvenlik gerekçesiyle özet algoritmasında kullanılacak c güvenlik parametresinin uzunluğu özet değerinin uzunluğunun iki katı olmalıdır [6]. Giriş vektörünün f fonksiyonu tarafından işlenebilmesi için 5x5xw şeklinde bir ifadeye (state) dönüştürülmesi gerekmektedir. $w = 2^l$, l = 1,2,…,6 şeklinde ifade edilen bir değerdir. b_i giriş mesajının bloklarını göstermektedir. h_i, f fonksiyonu sonucunda elde edilen özet fonksiyonu parçalarını göstermektedir.



Şekil 3.1. Keccak özet algoritmasının genel yapısı [6]

Sünger yapıları genel olarak iki ana kısımdan oluşmaktadır.

1. Giriş veri bloklarının alındığı ve işlendiği kısım
2. Çıkış değerlerinin üretildiği kısım

Algoritmanın 1. kısmı mesaj bloklarının sırayla alınır ve f çekirdek fonksiyonu tarafından işlenir. Algoritmanın ayrıntılı yapısı aşağıda adım adım verilmiştir.

1. r ve c değerlerinden oluşan başlangıç vektörü sıfıra eşitlenir.
2. Giriş verisi bloklara ayrılır eğer gerekiyorsa son blokta ekleme işlemi gerçekleştirilir. Eksik bitlerin yerine 10…1 bit dizisi yanı eksi bitlerin ilki ve sonuncusu 1, diğerleri 0 olacak şekilde doldurulur.
3. Giriş vektörünün r'lik parçası aynı boyuttaki 1. blok parçasıyla XOR'lanır.
4. Oluşan değer giriş vektörünün yeni r' bitlik parçasını oluşturur.
5. Giriş vektörü f fonksiyonuna giriş verisi olarak verilir.
6. f fonksiyonunun çıkışı yeni giriş verisini oluşturur. Eğer bütün giriş blokları işlenmişse algoritmanın birinci kısmı sonlandırılır aksi halde 3. adıma geçilir.

Bu kısımda özet değeri üretilir. Algoritmanın ayrıntılı yapısı aşağıda adım adım verilmiştir.

1. En son uygulanan f fonksiyonunun çıkış vektörünün ilk r biti çıkış olarak alınır.
2. En son çıkış vektörüne tekrar f fonksiyonu uygulanır.
3. Yeni çıkış vektörünün ilk r biti çıkış olarak alınır. Eğer istenilen uzunlukta çıkış değeri elde edilmişse işlem sonlandırılır aksi halde 2. adıma geçilir.

3.2. f Fonksiyonu

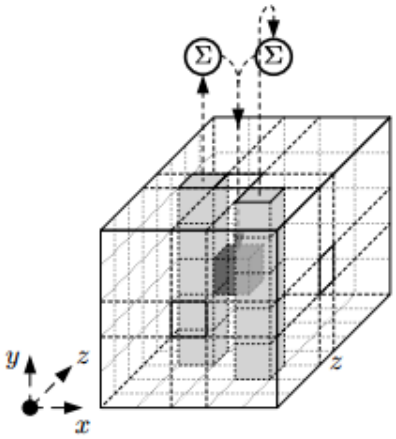
Çok güvenli ve esnek bir yapıya sahip olan f çekirdek fonksiyonu sünger yapılarının temel parçasını oluşturmaktadır.

Fonksiyonda kullanılan başlangıç ve ara giriş vektör değerleri farklı olabilmektedir. Bu değişiklik temel olarak v = 5x5xw yapısından kaynaklanmaktadır. Çünkü $w = 2^l$, l = 1,2,…,6 ifadesinde görüldüğü gibi w değeri değişebilmektedir. Dolayısıyla vektör değerinin boyutu 25,50,…,1600 değerlerini alabilmektedir.

Keccak özet algoritmasında kullanılan vektör boyutuna göre fonksiyonda kullanılan tur sayısı değişmektedir. Bu değişiklik 12+2l ifadesiyle gösterilmektedir. Örneğin 1600 bit uzunluklu giriş vektörü için l değeri 6 olur ve tur sayısı ise 12+2l formülünden 24 olmaktadır. Özellikle güvenlik açısından vektör değerinin boyutunun büyük olması istenmektedir.

f fonksiyonu her turda aşağıda anlatılan beş temel işlemi gerçekleştirmektedir. Bu işlemler her tur için sırayla tekrar edilmektedir.

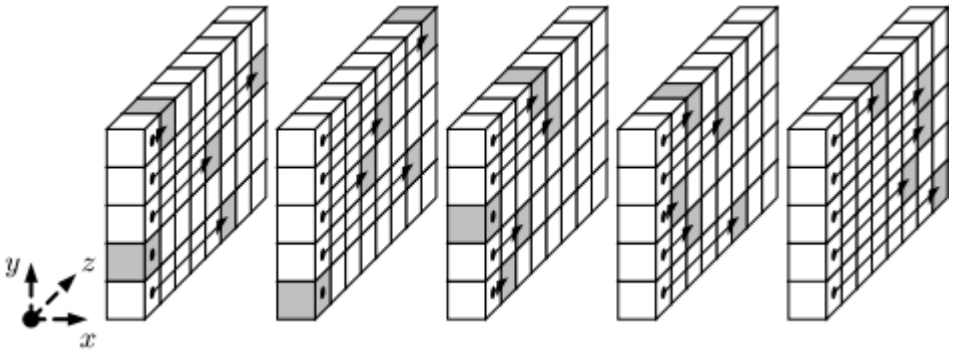
Θ: Her 5 bitlik sütunlar için eşitlik (parity) hesaplanır ve komşu sütunlar XOR işlemine tabi tutulur. Daha açık bir ifade ile, satır, sütun ve w indis değeri sırasıyla i, j ve k olarak alınırsa eşitlik değeri a[i][j][k] $\oplus$ = parity(a[0..4][j−1][k]) $\oplus$ parity(a[0..4][j+1][k−1]) şeklinde hesaplanabilir. Burada ifade edilen yapı Şekil 3. 2.'de verilmiştir.



Şekil 3.2. Θ işleminin genel yapısı [7]

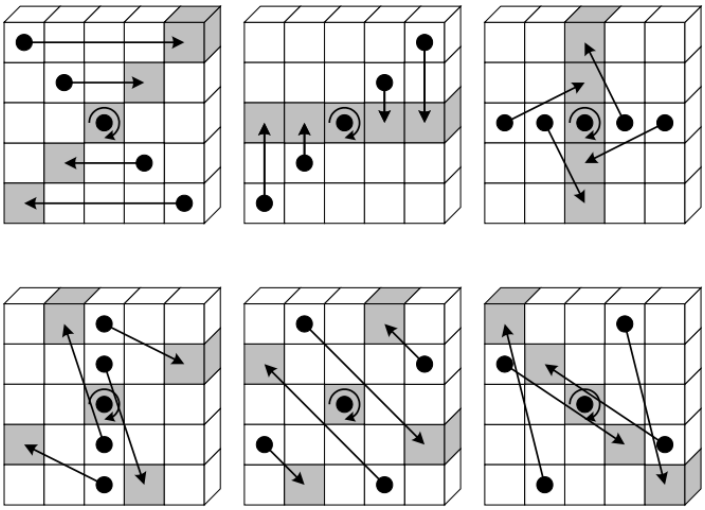
$\boldsymbol{p}$**:** üçgen sayısı değerlerine göre 25 kelime değerlerinden her biri için bit düzeyinde döndürme işlemi gerçekleştirilir. Şekil 3. 3.'de ilgili yapı gösterilmiştir. Üçgen sayıları üçgen oluşturmada kullanılan nokta sayısıyla temsil edilen ve 0, 1, 3, 6, 10, 15, …. şeklinde artış gösteren sayı dizisidir. Daha açık bir ifade ile, a[0][0] için döndürme işlemi gerçekleştirilmez ve 0≤t<24 aralığındaki her t a[i][j][k] = a[i][j][k−(t+1)(t+2)/2], burada i ve j değerleri aşağıdaki formül 3. 1.'ye göre yapılmaktadır.

$$\begin{pmatrix} i \\ j \end{pmatrix} = \begin{pmatrix} 3 & 2 \\ 1 & 0 \end{pmatrix}^{t} \begin{pmatrix} 0 \\ 1 \end{pmatrix} \qquad (3.1.)$$



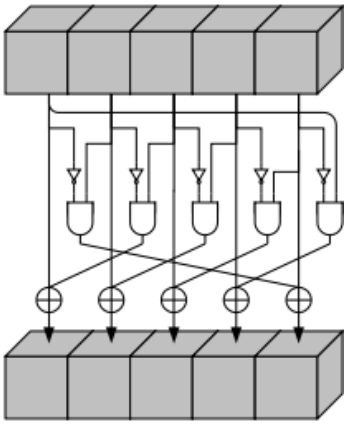
Şekil 3.3. p işleminin genel yapısı [7]

$\boldsymbol{\pi}$**:** 25 kelime sabit bir yapıda yer değiştirme işlemi yapar. Yani, $a[j][2i+3j] = a[i][j]$ şeklinde yer değiştirme işlemi gerçekleştirilir. İfade edilen yapı Şekil 3. 2.'de gösterilmiştir.



Şekil 3.4. π işleminin genel yapısı [7]

***X*:** a = a $\oplus$ (¬b & c) yapısı kullanılarak bit düzeyinde birleştirme işlemi yapılır. Daha açık bir ifade ile, $a[i][j][k]$ $\oplus$= $\neg a[i][j+1][k]$ & $a[i][j+2][k]$ olacak şekilde birleştirme işlemi gerçekleştirilir. Şekil 3. 5.'de *X* işleminde gerçekleştirilen işlemler verilmiştir. Bu işlem Keccak yapısında doğrusal olmayan tek yapıdır.



Şekil 3.5. *X* işleminin genel yapısı [7]

***ι*:** Giriş vektör içerisindeki değerler XOR işlemine tabi tutulur. Yani, *n*. turda $0 \le m \le \ell$, için $a[0][0][2^m-1]$ değeri 8. dereceden $m+7n$ değeri ile XOR'lanır.

3.3. Keccak Algoritmasının Analizi

Keccak algoritması daha önce kullanılan özetleme algoritmalarına göre çok daha esnek ve güvenli bir yapıya sahiptir.

Esneklik: Herhangi bir uygulamada basit bir yer değiştirme ile SHA-2 özetleme algoritmasını SHA-3 ile yer değiştirilmesi mümkündür. Bu yüzden SHA-3 224, 256, 384 ve 512 veya başka özetleme algoritmalarıyla kolaylıkla değiştirilebilmektedir çünkü bu algoritmalar için gereken bit uzunluğunu desteklenmektedir.

Keccak algoritması online uygulamalarda kolaylıkla kullanılabilecek bir yapıya sahiptir çünkü bu algoritmada giriş mesajının hepsinin hafızaya yerleştirmesine gerek yoktur. Giriş mesajı 256 veya 512 bitlik gibi küçük blokları şeklinde sırayla alınabilir ve çıkış değeri de sırayla üretilebilir.

Keccak özetleme algoritmasıyla, Merkle-Demgard yapılarına dayanan algoritmalardan farklı olarak istenilen uzunlukta çıkış değeri üretilebilmektedir. Dolayısıyla bu özellik Keccak özetleme algoritmasının akış şifreleme ve rastgele sayı üretimi gibi birçok alanda kullanımına olanak tanımaktadır [6].

Güvenlik: Keccak algoritmasının güvenlik düzeyi teorik olarak ihtiyaç duyulan farklı boyuttaki özet değerlerini karşılamaktadır. Ayrıca bu fonksiyon ön görme dayanıklılığı ve çakışma dayanıklılığına karşı yeterli seviyede güvenlik sağlamaktadır. Dahası bu fonksiyon SHA-2 fonksiyonuna karşı yapılabilecek muhtemel başarılı saldırılara karşı dayanıklıdır çünkü bu fonksiyonun yapısı ve işlemleri diğer fonksiyonlardan farklıdır [6].

Keccak algoritmasıyla büyük boyutta özet değerleri üretilebildiği için bu algoritma güvercin deliği ilkesi veya doğum günü çelişkisi gibi kaba kuvvet saldırılarına karşı oldukça dayanıklıdır. Ayrıca istenilen boyutta özet değeri üretimine olanak tanıdığından özet değerinin tahmin edilebilme ihtimalini azaltmaktadır.

Blok boyutuna göre tur sayısının değişmesi Keccak algoritmasının güvenliğini artırmaktadır. Çünkü bu durumda giriş mesajının ne kadar işlemden geçtiği belirlenemez. Dolayısıyla giriş mesajının elde edilme ihtimali azalır. Ayrıca başlangıç ve ara vektör değerlerinin mesaj bloklarından büyük olması Keccak algoritmasının güvenliğini artıran önemli bir parametredir [6].

Maliyet: Keccak algoritmasının diğer özetleme algoritmalarından hızlı olduğu dolayısıyla daha az hesaplama kaynağı gerektirmektedir [23]. Ayrıca algoritmanın

gerçekleştirilmesi için küçük boyutta hafıza yeterli olduğundan mobil cihaz gibi sınırlı kaynağa sahip donanımlarda kolaylıkla gerçekleştirilebilmektedir. Dolayısıyla bu algoritma donanım platformları üzerinde hem zaman hem de maliyet bakımından daha etkilidir [24].

Keccak özetleme algoritması güvenlik, esneklik ve maliyet gibi değerlendirme paramereleri göz önüne alınarak değerlendirildiğinde diğer aday algoritmalara karşı oldukça iyi olduğu görülmektedir.

4. ÖZET TABANLI MESAJ DOĞRULAMA KODU (HMAC)

Şifrelemede MDK yapıları, kimlik doğrulama ve bütünlük kontrolü için kullanılır. Bölüm 2. 1. 1.'de MDK yapıları ele alınmıştır. Genel olarak bu yapılar iki farklı şekilde gerçekleştirilir.

1. Blok şifreleme fonksiyonlarına dayanan MDK yapılarıdır. Burada AES, DES gibi blok şifreleme fonksiyonları MDK yapılarını gerçekleştirecek şekilde kullanılır [13].
2. Özet fonksiyonları yapısına dayanan mesaj doğrulma kodlarıdır [25].

Özet fonksiyonlarını temel alan yapılar HMAC olarak adlandırılır ve MD5, SHA–1 gibi özet fonksiyonları kullanılarak gerçekleştirilir. Bu yapılar blok şifreleme fonksiyonlarına dayanan MDK yapılarından daha fazla kullanışlıdır. Çünkü bu yapılar hem daha fazla hazır kütüphane imkânları sunmaktadır hem de blok şifreleme yapılarına dayanan MDK yapılarından daha hızlıdır [13].

4.1. HMAC Tasarım Hedefleri

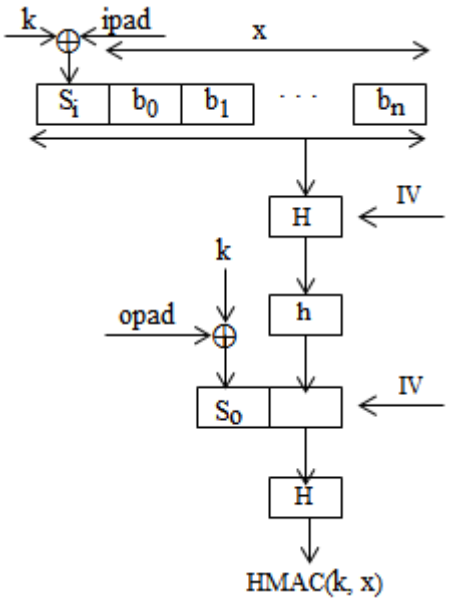
HMAC tasarlamasında istenen hedefler RFC 2104 te aşağıdaki gibi belirtilmiştir [26].

1. Herhangi bir değişiklik yapmadan, mevcut özet fonksiyonlarını kullanmak.
2. Özet fonksiyonun orijinal performansını, önemli bir bozulma olmadan, kullanmak.
3. Özet fonksiyonun altındaki makul varsayımlara dayalı kimlik doğrulama mekanizmasının gücünün kriptografik analizini iyi anlamak.
4. Gerektiğinde veya daha etkili ve güvenli özet fonksiyonlarının bulunması durumunda bu fonksiyonların kullanımına veya bunlarla yer değiştirmeye olanak tanımak.

İlk iki özellik HMAC yapılarının kullanımı için oldukça önemlidir. Ayrıca bu yapılar özet fonksiyonlarını "kara kutu" gibi görür. Böylece hem HMAC kodları paketlenebilir ve herhangi bir değişiklik yapılmadan kullanılabilir hem de bu yapılardaki özet fonksiyonları yeni türevleri ile yer değiştirilebilir. Çünkü daha etkili özet fonksiyonları ortaya çıktıkça bu gerekebilir. Özellikle mevcut fonksiyonlar güvenlik kısıtlamalarına sebep olduğunda daha güvenli fonksiyonlarla yer değiştirilebilir.

4.2. HMAC Algoritması

HMAC algoritmasının genel yapısı Şekil 4. 1.'de gösterilmiştir. Bu şekilde x giriş mesajı, H kullanılacak özet fonksiyonu (MD5, Keccak vb.) , k gizli anahtar, $opad$ hex(0x36) [ascii olarak "6" karakteri] değerinin tekrarlanmasıyla oluşan byte dizisi, $ipad$ hex(0x5C)[ascii olarak "\" karakteri] değerinin tekrarlanmasıyla oluşan byte dizisi, || birleştirme işlemi , $\oplus$ XOR operatörü, b mesaj bloku olarak alınmıştır.



Şekil 4.1. HMAC algoritmasının yapısı

HMAC (k,x) = H[(k+opad) | | H [(k+ipad) | | x]]

1. k (key)'nın uzunluğu b (bloksize)'den büyükse H (hash) özet fonksiyonu ile özetlenerek küçültülür.
2. k'nın uzunluğu b'den küçükse b'ye eşit olana kadar hex(0x00) ile doldurulur.
3. Yine b'nin uzunluğunda $ipad$ (tamamen hex(0x36) ile doldurularak) ve $opad$ (tamamen hex(0x5C) ile doldurularak) oluşturulur.
4. $k \oplus ipad$ işlemi ile S_i oluşturulur.
5. $k \oplus opad$ işlemi ile S_o oluşturulur.
6. k_ipad 'e x (mesaj) eklenir.
7. 6.Adım da üretilen akışa H uygulanır.
8. 7.Adım da oluşturulan hash değerine 5.adımda üretilen k_opad değeri eklenir.

9. 8.Adımda oluşan ifadeye H fonksiyonu uygulanır ve HMAC değeri elde edilir.

4.3. HMAC Algoritmasında kullanılan Anahtar Değeri

HMAC algoritması için anahtar uzunluğunun belli bir uzunlukta olması gerekir çünkü edilen bu uzunluk değeri bu algoritmalar için önemli bir güvenlik parametresidir. Bununla beraber belli bir değerden sonra anahtar değerinin uzunluğu güvenlik seviyesini artırmaz. HMAC anahtar üretim yöntemlerinde blok boyutundan küçük anahtar değerlerine 0 eklenerek anahtar uzunluğu blok uzunluğuna eşit hale getirilir [27]. Blok değerinden uzun anahtar değerlerinin özet değeri hesaplanarak anahtar değeri olarak belirlenir.

4.4. HMAC Algoritmalarının Güvenliği

HMAC yapıları, bu yapıları oluşturmak için kullanılan özetleme algoritmasına göre daha güvenlidir [12]. HMAC algoritmalarının gücü iki parametreye bağlıdır. Bunlardan ilki HMAC yapısını oluşturan özet fonksiyonudur ikincisi ise HMAC değeri üretmek için kullanılan gizli bir anahtar değeridir. Öncelikle kullanılan özet fonksiyonu ne kadar güvenli ise HMAC yapısı da o nispette güvenli olur. İkinci olarak kullanılan gizli anahtar değerinin güvenli olması HMAC yapısının da güvenli olmasını sağlar. Ayrıca bu yapıların çıkış değerlerinin uzunluğu önemli bir diğer güvenlik parametresidir [11].

HMAC algoritmalarından olması gereken bir diğer özellik ise kullanılan özet algoritmasının güvenliği ile özet fonksiyonunun gömüldüğü HMAC algoritmasının güvenliği arasında doğru bir ilişki olmasıdır. Diğer taraftan, HMAC algoritmaları, çakışmalara bu algoritmaları oluşturmak için kullanılan özet fonksiyonlarından daha dayanıklıdır [28]. Örneğin MD5'te bulunan zayıflıklar HMAC-MD5 algoritmasında bulunmamaktadır [13].

5. TEK KULLANIMLIK ŞİFRE

Teknolojinin yaygınlaşmasıyla internetin kullanımı çok fazla artmıştır. Bunun sonucunda alışverişten fatura ödemeye veya havale işlemlerine kadar günlük hayattaki birçok iş internet üzerinden yapılmaktadır. Bu durumda banka hesapları gibi önemli kaynaklara yetkisiz kişilerin erişememesi istenmemektedir. Ayrıca Ülkemizde bankaların internet şubelerinde TKŞ kullanımı BDDK'nın 14.09.2007 tarih ve 26643 sayılı tebliği gereğince 1 Ocak 2010 tarihi itibariyle yasal olarak zorunlu hale getirildi. Ülkemizdeki tüm bankaları kapsayan bu uygulamalar İnternet Şubesi'ne girişinizde müşteri numarası, parola ve şifre bilgilerinize ek olarak TKŞ girilmesi istenmektedir. Dolayısıyla bu tarihten itibaren TKŞ üretimi be bu hizmetin müşterilere sunulması bir zorunluluk haline gelmiştir [29].

Genel olarak bu tür erişimlerin engellenmesi kullanıcıya özel şifrelerin kullanılmasıyla sağlanmaktadır. Bu özel şifreler, sabit ve dinamik şifreler olmak üzere iki sınıfa ayrılır.

Sabit şifreler; bu önemli kaynaklara erişimden önce kullanıcıdan alınır ve sistemde o kullanıcı için tutulan şifreyle karşılaştırılır. Eğer şifreler uyuşuyorsa erişime izin verilir aksi halde erişim reddedilir. Her erişimde aynı şifre kullanıldığı için şifreyi elde eden kişi bu önemli kaynaklara erişebilecektir, çünkü kaynağa her erişimde aynı şifre kullanılmaktadır.

Sabit şifrelerin kullanımı birçok probleme sebep olmaktadır. İlk olarak, bu şifreler güvenli olmayan bir ağ üzerinden iletildiği için ağ dinlenmesi gibi çeşitli yöntemlerle şifreler elde edilebilir. İkinci olarak, yapılan araştırmalarda birçok kullanıcının şifre olarak 11112222, qwerty, ahmet57, abc123 gibi basit ve kolayca tahmin edilebilecek şifreler seçtiği görülmüştür. Dolayısıyla müşterilerin şifre alışkanlıkları sabit şifrelerin elde edilmesine sebep olabilir. Bunun yanında çalınan çerezler (cookies) veya internet tarayıcıları kullanıcı adı ve şifre değerlerini hatırlayabiliyor. Dahası bu veriler; Malware, Trojansi Keyloger gibi kötü amaçlı yazılımlar ile elde edilebilir. Bununla beraber sabit şifreler yeterli deneme sayısı ve zaman verildiği takdirde yetkisiz kişiler tarafından aşılabilir.

Sonuç olarak sabit şifre kullanımından kaynaklanan bu problemleri ortadan kaldırmak için TKŞ (Tek Kullanımlık Şifre) kullanılması gerekmektedir. Eğer TKŞ kullanılırsa herhangi bir erişim için kullanılan şifre elde edilse bile kaynağa yeniden erişim

mümkün olmayacaktır çünkü ilgili kaynağa her erişimde yeni bir şifre kullanılmaktadır. Ayrıca TKŞ algoritmaları sonraki TKŞ değerinin tahmin edilmesini engellemek için rastgelelik veya sözde rastgelelik kavramlarından faydalanır.

5.1. TKŞ Üretim Yöntemleri

TKŞ üretim algoritmalarının detayları büyük farklılıklar göstermekle beraber genel olarak iki farklı yaklaşım mevcuttur. Bunlardan ilki kullanıcı ile doğrulama merkezi arasında zaman senkronizasyonuna dayanan sistemlerdir. İkincisi ise matematiksel algoritmalara dayanan sistemlerdir. Matematiksel yöntemlerden bir kısmı önceki TKŞ değerlerini kullanırken bir kısmı ise rastgeleliğe dayanan sistemleri kullanır.

5.1.1. Zaman Tabanlı Sistemler

İlk yaklaşım zaman tabanlı sistemlerdir. Zaman tabanlı TKŞ genellikle 'token' adı verilen bir donanım parçaları ile üretilmektedir. Bu donanımın içinde gerçek zamanlı ve kimlik doğrulama sunucusu üzerindeki saat ile senkronize bir saat bulunur yani eş zamanlı çalışma söz konusudur. Bu algoritmalarda üretilen yeni TKŞ değerleri önceki bir parola veya gizli anahtardan ziyade o anki zaman ile alakalıdır ve zaman, algoritmanın önemli bir parçasıdır.

TKŞ değerleri token denilen cihazlarla üretildiği gibi yazılım programı veya mobil cihaz gibi farklı yollarla da üretilebilir. Bu sistemde üretilen şifreler belirli bir zaman aralığı için üretilir ve sadece o zaman aralığında geçerli olur. Bununla beraber, bu belirlenen zaman aralığı geniş tutulursa, aynı şifrenin tekrar kullanımı söz konusu olur [30]. Ayrıca, mekanizma istemciden sunucuya erişme zamanını da göz önünde bulundurarak bir zaman sapma değeri de hesaplamak zorundadır [31].

5.1.2. Matematiksel Tabanlı Sistemler

Bu sistemler olay tabanlıdır yani sisteme yapılan başarılı erişimde şifre değişir [32]. Bu tasarımda, hem kullanıcı hem de şifre üretim merkezi tarafında eş zamanlı çalışan bir sayaç mevcuttur [33]. Bu algoritmaların bir örneği Leslie Lamport tarafından önerilmiştir. Lamport 1981'de kimlik doğrulama mekanizması olan tek kullanımlık şifre tasarımında tek

yönlü özet fonksiyonlarının kullanımını önermiştir [34]. Bu fonksiyonlarda başlangıç değeri için bir tohum değeri belirlenir daha sonra üretilen TKŞ değerleri bu tohum değerinden türetilir. Bu değerler $f(s), f(f(s)), f\left(f(f(s))\right)\ldots$ şeklinde gösterilebilir. Eğer belirsiz bir şifre dizisi elde edilmek istenirse belli bir süre sonra tohum değeri değiştirilmelidir. Bu şekilde TKŞ üretimi Lamport yapısına dayanmaktadır.

Tasarımlarda özet fonksiyonu kullanılmasının temel sebebi bu yapıların tek yönlü olmasıdır yani elde edilen şifre değerinden bir önceki şifre değeri elde edilemez. Ayrıca güvenli özet fonksiyonlarında üretilecek yeni şifrenin tahmin edilmesi mümkün değildir. Bu yüzden herhangi bir erişimde kullanılan TKŞ değeri bir sonraki erişim için kullanılamaz ve yeni TKŞ değerleri için bir tehdit oluşturmaz.

5.2. TKŞ Dağıtım Yöntemleri

TKŞ değerlerinin üretimi farklı şekillerde olabilmektedir. Bunlardan ilki daha önce bahsedilen token denilen cihazların kullanılmasıdır. Bu cihazlarda TKŞ üretimi için zaman tabanlı veya matematiksel sistemler kullanılabilir. Bu cihazların kullanımı basittir fakat çalınma gibi durumlarda yetkisiz kişilerde kullanabilir. Ayrıca bu cihazların batarya problemi olmaktadır ve belli bir maliyet gerektirir. İkinci bir yol ise şifrelerin bir kâğıt üzerine yazılı olarak üretilmesidir. Bu yöntem ise kullanışlıdır ve basittir fakat güvenli değildir. Üçüncü bir yöntem ise mobil cihazlarla TKŞ değerlerinin üretimi yapılmaktadır. Mobil cihazların token gibi kullanımına dayanır. Kullanıcıya erişim kolaydır çünkü hemen her kullanıcı mobil telefon sahibidir. Ayrıca bu uygulamalar genellikle fazla bir depolama alanı ve hesaplama yükü gerektirmez. Bununla birlikte bu yöntem çalınma ve kaybolma gibi durumlarda yetkisiz kişilerce de kullanılabilir. Bir diğer yöntem ise SMS olarak bilinen metin mesajlaşma kavramıdır. Burada herhangi bir cep veya sabit telefonda metin mesajı yoluyla şifre iletilir. Bu yöntem oldukça yaygın bir iletişim kanalına sahip olduğundan iletişim hattının oluşturulması için bir maliyet gerektirmez. Dolayısıyla bu yöntem tüm tüketicilere ulaşmak için büyük bir potansiyele sahiptir [35]. Bu yöntemlerde verinin güvenli iletimi için A/x olarak adlandırılan kanal şifreleme standartları kullanılmaktadır. fakat bu standartların yeterli seviyede güvenlik sağlayamadığı belirlenmiştir [36]. Ayrıca bu yöntemde mobil vericinin aktif olmadığı durumlarda kullanıcıya erişim mümkün olmamaktadır. Dahası bu sistemlerde sağlanan güvenlik mobil operatörler tarafından sağlanan güvenlik ile sınırlıdır.

TKŞ değerlerinin dağıtılmasında kullanılan bütün bu sistemlerde göz önünde bulundurulan çeşitli parametreler bulunmaktadır. Bunlar güvenlik, maliyet, kullanılabilirlik, erişim vb. şeklinde sıralanabilir. Kullanılacak sistemin ihtiyaçlarına ve kalitesine göre bu parametreler belirlenmektedir.

5.3. TKŞ Değerinin Analizi

TKŞ değerinin güvenliği test edilmelidir. Yani bu şifrelerin uluslararası standartlara uyması gerekmektedir. Bu güvenlik testleri anahtar değerinin rastgele olması ve tahmin edilememesi gibi parametrelerden oluşmaktadır. Bu testler NIST'in SP800-22b, sp800-90a, sp800-90b sp800-90c testleri başta olmak üzere uluslararası testlerdir. Bu standartlar genellikle NIST, FIPS (Federal Information Processing Standart) uluslararası etkinliği olan ve kabul görmüş kuruluşlar tarafından belirlenmektedir. Bu standartlardan birkaçı rfc6560-otp, rfc6238-totp, rfc4226-hotp'dir[8]. Bununla birlikte, temel yapıları yukarıda belirtildiği gibi olan TKŞ üreten sistemler uygulamada farklı ayrıntılara sahip olmaktadır. Dolayısıyla TKŞ üreten firmalar kendi çıkarlarını zedeleyecek ticari sır sayılabilecek ayrıntıları açıklamaktan kaçınmaktadır. Çünkü bu yazılımlar bankalar başta olmak üzere, birçok alanda ticari ürün olarak kullanılmaktadır.

6. TEK KULLANIMLIK ŞİFRE ÜRETECİ

Bu tez çalışmasında Tek Kullanımlık Şifre değerlerinin üretimi incelendi. Bu değerlerin üretilmesi için Leslie Lamport tarafından önerilen yaklaşım kullanıldı. Bu yaklaşımla özet fonksiyonları ve bir anahtar değeri kullanılarak TKŞ değerleri üretilmektedir. Bu yöntem HMAC olarak adlandırılır. Dolayısıyla üretilen TKŞ değerlerinin güvenliği kullanılan özet algoritmasının güvenliğine ve anahtar değerine bağlıdır. Üretilen şifre değerlerinin güvenliğinin en üst düzeyde olması için en son özetleme fonksiyonu standardı olan Keccak algoritması kullanıldı. Bu fonksiyonda TKŞ değerinin üretilmesi için tohum değeri olarak kullanıcı ve doğrulayıcı arasında paylaşılan gizli bir anahtar değeri kullanıldı. Bu anahtar değerine ve sayıcıya bağlı olarak TKŞ değeri üretildi.

6.1. Tek Kullanımlık Şifre Üretecinde Kullanılan Yapı

TKŞ üretmek için özet algoritmasıyla birlikte güvenli bir gizli anahtar değeri ve sayıcı değeri kullanılarak HMAC algoritması gerçekleştirildi. TKŞ üretimi için kullanılan HMAC yapısı bölüm 4. 2. verildi. Algoritmada kullanılan $\boldsymbol{k}$ gizli anahtar değeri sadece kullanıcı ve doğrulayıcı mekanizma arasında paylaşılır. Sayıcı değeri ise her erişimde güncellenen ve kullanıcı ile doğrulayıcı mekanizma arasında senkronizasyonu sağlayan bir değişkendir. Formül 6. 1.'de bu değer $\boldsymbol{c}$ olarak ifade edildi.

$$\text{TKŞ} = \text{kırpma}\ (\text{HMAC-Keccak}(k, c)) \qquad (6.\ 1.)$$

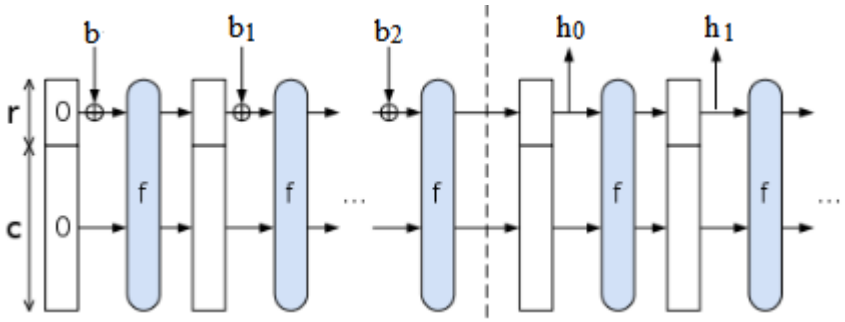
Bölüm 4. 2.'de verilen HMAC algoritması sonucunda üretilen HMAC değeri kırpma işlemine tabi tutularak TKŞ değeri üretildi. Kırpma işlemi dinamik olarak yapılır. Burada üretilen altmış dört karakterlik HMAC değerinden sekiz karakterlik TKŞ değeri üretilir. Bu işlemde HMAC değerinin son karakteri alınır ve bu karakterin sayı değeri hesaplanır. Daha sonra bu değerden başlanarak ilk sekiz karakter alınır.

6.2. Kullanılan Özetleme Algoritmasının Blok Yapısı

Bu tez çalışması projesinde özet fonksiyonu olarak Keccak kullanıldı. Keccak özetleme algoritması sünger yapılarına dayandığı için giriş mesaj blokları önce başlangıç

değeriyle XOR'landı. İlk blok için başlangıç değeri sıfır olarak alındı, diğer bloklar için bu değer bir önceki f fonksiyonunun çıkışı olarak alındı. f fonksiyonu Keccak yapısının çekirdek fonksiyonudur. Projede kullanılan sünger yapısıyla herhangi bir uzunlukta bir giriş mesajı alınabilir ve istenilen uzunlukta bir özet değeri üretilebilir.

Kullanılan özet algoritmasının genel yapısı Şekil 6. 1.'de gösterildi. Burada r ve c değerleri v giriş vektörünü oluşturmaktadır. r, f fonksiyonunda kullanılan blok boyutuna bağlı olarak belirlenmektedir ve bu değere bağlı olarak 1024 bit olarak alınmıştır. Giriş vektörünün f fonksiyonu tarafından işlenebilmesi için 5x5xw şeklinde bir ifade değerlerine dönüştürüldü. $w = 2^l$, şeklinde belirtildiği için l değeri 6 olarak alındı. b_i giriş mesajının bloklarını göstermektedir ve 1024 olarak alındı. h_i, f fonksiyonu sonucunda elde edilen özet fonksiyonu parçalarını göstermektedir ve kullanılan algoritmaya bağlı olarak bu değer de 1024 olarak belirlenmiştir.



Şekil 6.1. Geliştirilen uygulama için kullanılan yapı [6]

Sünger yapıları genel olarak iki ana kısımdan oluşmaktadır.

1. Giriş veri bloklarının alındığı ve işlendiği kısım
2. Çıkış değerlerinin üretildiği kısım

Algoritmanın 1. kısmının ayrıntılı yapısı aşağıda adım adım gösterilmiştir.

1. r ve c değerlerinden oluşan başlangıç vektörü sıfıra eşitlendi.
2. Giriş verisi bloklara ayrılır eğer gerekiyorsa son blokta ekleme işlemi yapıldı. Ekleme işlemi son blok boyutunu 1024'de tamamlayacak şekilde 100 . . . 1 olarak başlangıç ve bitiş değerleri sıfır olacak şekilde yeterli sayıda sıfır eklenerek gerçekleştirildi.
3. Giriş vektörünün r'lik parçası aynı boyuttaki 1. blok parçasıyla XOR'landı.
4. Oluşan değer giriş vektörünün yeni r' bitlik parçasını oluşturdu.
5. Oluşan bu yeni giriş vektörü f fonksiyonuna giriş verisi olarak verildi.

6. f fonksiyonunun çıkışı yeni giriş verisini oluşturur. Eğer bütün giriş blokları işlenmişse algoritmanın birirnci kısmı sonlandırılır aksi halde 3. adıma geçildi.

Algoritmanın 2. kısmının ayrıntılı yapısı ise aşağıda adım adım gösterilmiştir.

1. En son uygulanan f fonksiyonunun çıkış vektörünün ilk r biti çıkış olarak alındı.
2. En son çıkış vektörüne tekrar f fonksiyonu uygulandı.
3. Yeni çıkış vektörünün ilk r biti çıkış olarak alınır. Eğer istenilen uzunlukta çıkış değeri elde edilmişse işlem sonlandırıldı aksi halde 2. adıma geçildi.

6.3. Uygulamada Kullanılan Çekirdek Fonksiyonu

Çok güvenli ve esnek bir yapıya sahip olan f çekirdek fonksiyonu sünger yapılarının temel parçasını oluşturmaktadır. Bu fonksiyonda kullanılan başlangıç ve ara giriş vektör değerleri v = 5x5xw yapısında oluşturuldu ve bunun sonucunda. w değeri 2^l ifadesine bağlı olarak 6 alındı. Dolayısıyla vektör değerinin boyutu 1600 oldu. Ayrıca Keccak özet algoritmasında kullanılan vektör boyutuna göre fonksiyonda kullanılan tur sayısı değişmektedir. Bu değişiklik 12+2l ifadesiyle belirlendiği için 24 olarak alındı. Vektör değeri boyutunun özellikle güvenlik için tercih edildi.

f fonksiyonu her turda aşağıda anlatılan beş temel işlemi gerçekleştirmektedir. Bu işlemler her tur için sırayla tekrar edilmektedir.

Θ: Her 5 bitlik sütunlar için eşitlik (parity) hesaplanır ve komşu sütunlar XOR işlemine tabi tutulur. Daha açık bir ifade ile, satır, sütun ve w indis değeri sırasıyla i, j ve k olarak alınırsa eşitlik değeri a[i][j][k] $\oplus$ = parity(a[0..4][j−1][k]) $\oplus$ parity(a[0..4][j+1][k−1]) şeklinde hesaplandı. Algoritmanın kaba kodu aşağıda verilmiştir.

for (i sıfırdan dörde kadar)

c[i] = $a[i][0]$

for (j sıfırdan dörde kadar)

$c[i][j][k] = c[i][j][k]\ XOR\ a[i][j][k]$

for (i sıfırdan dörde kadar)

$d[\ i] = c[\ i - 1]\ XOR\ (ROT(c[\ i + 1],\ 1))$

for (j sıfırdan dörde kadar)

$a[\ i, j] = a[\ i, j]\ XOR\ d[x]$

***p*:** üçgen sayısı değerlerine göre 25 kelime değerlerinden her biri için bit düzeyinde döndürme işlemi gerçekleştirilir. Üçgen sayıları üçgen oluşturmada kullanılan nokta sayısıyla temsil edilen ve 0, 1, 3, 6, 10, 15, …. şeklinde artış gösteren sayı dizisidir. Daha açık bir ifade ile, a[0][0] için döndürme işlemi gerçekleştirilmez ve 0≤t<24 aralığındaki her t a[i][j][k] = a[i][j][k−(t+1)(t+2)/2], burada i ve j değerleri aşağıdaki formüle göre yapılmaktadır. p yapısının kaba kodu aşağıda verilmiştir.

$$\begin{pmatrix} i \\ j \end{pmatrix} = \begin{pmatrix} 3 & 2 \\ 1 & 0 \end{pmatrix}^{t} \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

for (t sıfırdan yirmi üçe kadar)

A[x, y] = ROT(a[x, y], (t + 1)(t + 2)/2)

$$\begin{pmatrix} i \\ j \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 2 & 3 \end{pmatrix} \begin{pmatrix} i \\ j \end{pmatrix}$$

***π*:** 25 kelime sabit bir yapıda yer değiştirme işlemi yapar. Yani, *a*[*j*][2*i*+3*j*] = *a*[*i*][*j*] şeklinde yer değiştirme işlemi gerçekleştirildi. Burada yapılan işlemlerin kaba kodu aşağıda verilmiştir.

for (i sıfırdan dörde kadar)

for (i sıfırdan dörde kadar)

$$\begin{pmatrix} X \\ Y \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 2 & 3 \end{pmatrix} \begin{pmatrix} i \\ j \end{pmatrix}$$

A[X][Y] = a[x][y]

***X*:** f fonksiyonundaki liner olmayan tek yapıdır. Satırlar üzerinde işlem yapılır ve a = a ⊕ (¬b & c) yapısı kullanılarak bit düzeyinde birleştirme işlemi yapılır. Daha açık bir ifade ile, *a*[*i*][*j*][*k*] ⊕= ¬*a*[*i*][*j*+*1*][*k*] & *a*[*i*][*j*+*2*][*k*] olacak şekilde birleştirme işlemi gerçekleştirildi. *X* yapısının kaba kodu aşağıda verilmiştir.

for (j sıfırdan dörde kadar)

for (i sıfırdan dörde kadar)

$a[i][j][k] = a\,[i][j][k] XOR\ ((NOT\ a[i][j+1][k])\ \ AND\ a[i][j+2][k])$

***ι*:** Giriş vektör içerisindeki değerler XOR işlemine tabi tutulur. Yani, *n*. turda 0≤*m*≤ℓ, için *a*[0][0][2^m−1] değeri 8. dereceden *m*+7*n* değeri ile XOR'landı.

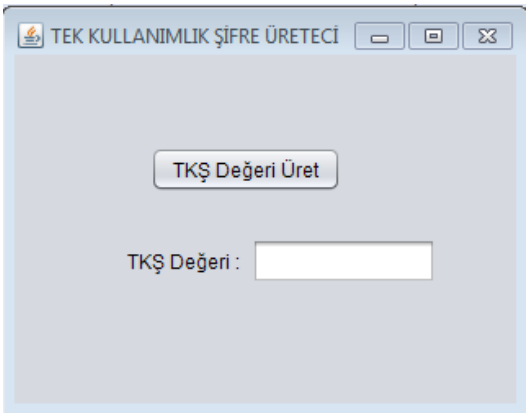
Tasarımlarda özet fonksiyonu kullanılmasının temel sebebi bu yapıların tek yönlü olmasıdır yani elde edilen şifre değerinden bir önceki şifre değeri elde edilemez. Ayrıca güvenli özet fonksiyonlarında üretilecek yeni şifrenin tahmin edilmesi mümkün değildir. Bu yüzden herhangi bir erişimde kullanılan TKŞ değeri bir sonraki erişim için kullanılamaz ve yeni TKŞ değerleri için bir tehdit oluşturmaz.

Fonksiyonda kullanılan anahtar değeri kullanıcıya bırakıldı. Böylece kullanıcının istenilen uzunlukta ve karakterde anahtar değeri üretebilmesi sağlandı. Anahtar değerinin kullanıcıya bağlı ve ona özel olması kullanılan yapıyı daha güvenli hale getirir. Böylece kullanılan giriş değeri bilince bile üretilecek TKŞ değerinin tahmin dilme ihtimalini azaltır.

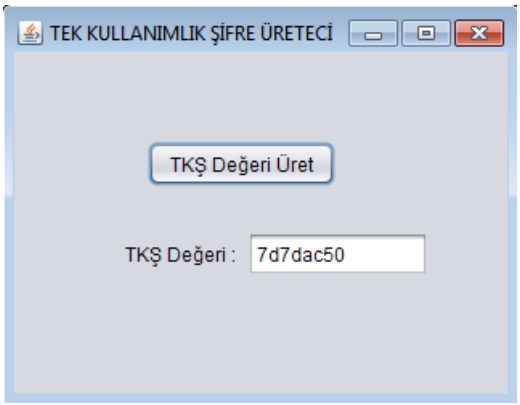
1024 (124 karakter) bitten oluşan özet değerinden 8 karakterlik TKŞ değerinin elde edilmesi için bu karakterlerden 16'nın katı olanlar alındı.

TKŞ üreteci projesi Netbeans platformunda ve Java programlama dilinde gerçekleştirildi. Bu seçimde Java'nın platformdan bağımsızlığı ve esnek yapısı etkili oldu.

Şekil 6. 3. ve Şekil 6. 4.'de TKŞ üretimi için geliştirilen uygulamanın iki farklı sonucu verilmiştir.



Şekil 6.2. Tek Kullanımlık Şifre Üreteci Uygulama Arayüzü



Şekil 6.3. Tek Kullanımlık Şifre Üreteci Uygulaması

7. SONUÇ

TKŞ değerleri internet bankacılığı başta olmak üzere kablosuz ağ erişimi, web sitesi güvelik uygulamaları, çağrı merkezi güvenlik uygulamaları, yazılım güvenlik uygulamaları, hatta bina giriş çıkış güvenlik uygulamaları gibi birçok alanda kullanılmaktadır. Özellikle internet bankacılığı uygulamalarında TKŞ kullanımı BDDK'nın 14.09.2007 tarih ve 26643 sayılı tebliği gereğince 1 Ocak 2010 tarihi itibariyle yasal olarak zorunlu hale geldi. Ülkemizde yaklaşık olarak 20 milyon internet bankacılığı hesabı ve 9 milyonu aşkın da internet bankacılığını kullanan aktif müşteri bulunmaktadır. Dolayısıyla TKŞ için oldukça büyük bir pazar bulunmaktadır.

Bu tez çalışmasında kullanılan yöntem Leslie Lamport tarafından önerilen TKŞ üretim yöntemi dayanmaktadır. Bu yöntemde kimlik doğrulama mekanizması olan tek kullanımlık şifre üretimi için tek yönlü özet fonksiyonları kullanılmaktadır.

TKŞ üretmek için özet algoritmasıyla birlikte güvenli bir anahtar değeri kullanılarak HMAC algoritması gerçekleştirildi. HMAC algoritması bir anahtar değeri ve özet algoritması kullanarak 128 bitlik karakter dizisi üretildi. HMAC algoritmasının gücü kullanılan özet algoritmasının gücüne bağlıdır. Bu tez çalışmasında kullanılan Keccak özet algoritması Ekim 2012'de NIST tarafından özet algoritmaları için yeni standart olarak belirlenmiştir. Bu algoritma yaklaşık dört yıl boyunca yüzlerce bilim adamı tarafından incelenmiştir ve herhangi bir zayıflığı belirlenememiştir. Bu sebeplerden dolayı bu tez çalışmasında Keccak özet algoritması kullanıldı. Bu algoritma sonucu üretilen 64 veya daha uzun karakter dizisi daha kullanışlı olan 8 karakter gibi makul bir seviyeye indirgendi.

KAYNAKÇA

[1] **Bhole A. T., Chaudhari S.,** 2013. Web Based Security using Online Password Authentication in Mobile Application, *International Journal of Science and Research (IJSR)*, vol. 2319-7064, no. 2, pp. 74-77, Haziran.

[2] (Kasım, 2013) TÜBİTAK UEKAE Açık Anahtar Altyapısı Eğitim Kitabı, Şifreleme. [Online]. http://www.kamusm.gov.tr/dosyalar/kitaplar/aaa/

[3] **Tsai C.S., Lee C.C., Hwang M.S.,** 2005. Password Authentication Scheme: Current Status and Key Issues, *International Journal of Network Security, Vol. 3, No 2, Sayfa 101-115*, Eylül.

[4] **Yinxiang L., Li X., Zhong L., Jing Y.**, 2010. One-time Password Authentication Scheme Authentication System and Apllication in Banking Financial System One-Time Password Authentication Scheme Authentication System and Apllication in Banking Financial System, *Networked Computing and Advanced Information Management (NCM), 2010 Sixth International Conference on*, Seoul, pp. 172 - 175.

[5] **Raihi D. M., Bellare M., Hoornaert F., Naccache D., Ranen O.,** 2005. An HMAC-Based One-Time Password Algorithm, Network Working Group, Request for Comments 4226,.

[6] **Bertoni G., Daemen J., Peeters M. ve Assche G. V.** (2013, Aralık) The Keccak sponge function family. [Online]. http://keccak.noekeon.org/

[7] (2013, Nisan) SHA-3 Competition (2007-2012). [Online]. http://csrc.nist.gov/groups/ST/hash/sha-3/index.html

[8] **Guo X., Huang S., Nazhandali L., Schaumont P.**, 2010. Fair and Comprehensive Performance Evaluation of 14 Second Round SHA-3 ASIC Implementations, *NIST 2nd SHA-3 Candidate Conference*.

[9] **Morris J.,** *Data Structures and Algorithms, 8.3 Hash Tables.*, 1998.

[10] **Knuth D. ,** *Sorting and Searching.*: Adisson-Wesley, 1973.

[11] **Shyamala C. K., Harini N., Padmanabhan T. R.,** *Cryptography and Security*. New Delhi, Hindistan: A Jhon Willey and Sons, 2013.

[12] **Paar C., Pelzl J.,** *Understanding Cryptography, Chapter 11*. Bochum, Almanya: Springer, 2010.

[13] **Stallings W. ,** *Cryptography and Network Security Principles and Practices, Chapter*

11 Cryptographic Hash Functions, Fifth Edition ed.: Prentice Hall, 2011.

[14] **Merkle R. C.,** *Secrecy, Authentication and Public Key Systems*. Standford, Amerika Birleşik Devletleri: Standford University, 1979.

[15] **Menezes A. J., Oorschot P. C., Vanstone S. A.**, *Handbook of Applied Cryptography, Chapter 9, 343-351*. Amerike Birleşik Deleti: Amazon Books , 2001.

[16] **Wang X., Yin Y. L., Yu H.,** 2005. Finding Collisions in the Full SHA-1, *Advances in Cryptology – CRYPTO 2005, Lecture Notes in Computer Science*, no. 3621, pp. 17-36.

[17] **Itai D., Orr, S. Adi,** *New Attacks on Keccak-224 and Keccak-256*, Anne Canteaut, Ed. Washington, DC, USA: Springer Berlin Heidelberg, 2012.

[18] **Liang J., Lai X.-J.,** 2007. Improved Collision Attack on Hash Function MD5, *Journal of Computer Science and Technology* , vol. 1, no. 22, pp. 79-87, Ocak.

[19] 2012-10-02. NIST Selects Winner of Secure Hash Algorithm (SHA-3) Competition, NIST, Retrieved.

[20] **Bertoni G., Daemen J., Peeters M., Assche G.V., Keer R.V,** 2012. Keccak Implemtation Overwiev, National Institute of Standards and Technology, Yarışma Sonucu Version 3.2,.

[21] **Bertoni G., Daemen J., Peeters M., Assche G. V.**, 2007. Sponge Functions, *Ecrypt Hash Workshop*.

[22] **Bertoni G., Daemen J., Peeters M. ve Assche G. V.**, 2008. On the Indifferentiability of the Sponge Construction, *EuroCrypt*.

[23] **Kavun E. B., Yalcin T.**, 2012. "On the Suitability of SHA-3 Finalists for Lightweight Applications, *The Third SHA-3 Candidate Conference,* Washington, D.C.

[24] **Gaj K., Homsirikamol E., Rogawski M., Shahid R., Sharif M. U.,** 2012. Comprehensive Evaluation of High-Speed and Medium-Speed Implementations of Five SHA-3 Finalists Using Xilinx and Altera FPGAs, IACR Cryptology ePrint Archive, 368,.

[25] **Bellare M., Canetti R., Krawczyk H.,** 1996. Keying Hash Functions for Message Authentication, *Advances in Cryptology — CRYPTO '96* , pp. 1-15.

[26] **Krawczyk H., Bellare M., Canneti R.,** 1997. HMAC Keyed-Hashing for Message Authentication, Network Working Group, Request for Comments 2104,.

[27] 2002. The Keyed Hash Message Authentication Code (HMAC), Federal Information Processing Standart Publication, FIPS PUB 198,.

[28] **Bellare M.,** *New Proofs for NMAC and HMAC: Security Without Collision-Resistance*, Cynthia Dwork, Ed. Santa Barbara, California, Amerika Birleşik Devletleri: Springer Berlin Heidelberg, 2006.

[29] 2007. Bankalarda Bilgi Sistemleri Yönetiminde Esas Alınacak İlkelere İlişkin Tebliğ, Banka Düzenleme ve Denetleme Kurumu, Resmî Gazete 26643,.

[30] **Lin I., Chang C.,** 2009. A countable and time-bound password-based user authentication scheme for the applications of electronic commerce, *Information Sciences*, no. 179, pp. 1269–1277.

[31] **Kim H. C., Lee H. W., Lee K. S., Jun M. S.**, 2008. A Design of One-Time Password Mechanism Using Public Key Infrastructure, *Fourth International Conference on Networked Computing and Advanced Information Management*, Gyeongju, pp. 18-24.

[32] **Lee Y. S., Lim H. T., Lee H. J.**, 2010. A Study on Efficient OTP Generationusing Stream Cipher with Random Digit, *Advanced CommunicationTechnology (ICACT)*, Phoenix Park, pp. 1670-1675.

[33] **Liao S., Zhang Q., Chen C., Dai Y.**, 2009. A Unidirectional One-Time Password Authentication Scheme without Counter Desynchronization, *ISECS International Colloquium on Computing, Communication, Control, and Management*, Sanya, pp. 361 - 364.

[34] **L. Lamport ,** 1981. Password Authentication with Insecure Communication., , *Communications of the ACM*, vol. 11, no. 24, pp. 770-772.

[35] **Indu S., Sathya T.N., Saravana Kumar V.**, 2013. A Stand-Alone and SMS-Based Approach for Authentication Using Mobile Phone, *Information Communication and Embedded Systems (ICICES), 2013 International Conference on* , Chennai, pp. 140 - 145.

[36] **Barkan E., Biham E., Keller N.**, 2003. Instant Ciphertext-Only Cryptanalysis of GSM Encrypted Communication, *Advances in Cryptology - CRYPTO 2003* , Santa Barbara, California, pp. 600-616.

[37] **Jia S. Z., Lin J., Feng X. R.,** 2013. An Identity Authentication Scheme Based on Dynamic Password Technology, *Applied Mechanics and Materials*, no. 411 - 414, pp.

166-171, Eylül.

[38] **Cha B. R., Kim N. H., Kim J. W.**, 2011. Prototype Analysis of OTP Key-Generation Based on Mobile Device Using Voice Characteristics, *Information Science and Applications (ICISA), 2011 International Conference on* , Jeju Island , pp. 1-5.

[39] **Tao F. Y., Ping S. G.**, 2009. Design of Two-Way One-Time-Password Authentication Scheme Based On True Random Numbers, *Computer Science and Engineering, WCSE '09. Second International Workshop on*, Qingdao, pp. 11-14, Ekim.

ÖZGEÇMİŞ

1987 Adıyaman merkez doğumluyum. İlkokul eğitimini Olgunlar Köyü İlköğretim Okulunda okudum. Ortaokulu Çelikhan YİBO'da okudum. Lise eğitimimi Rekabet Kurumu Lisesinde okudum. Lisans eğitimimi Karadeniz Teknik Üniversitesi Bilgisayar Mühendisliği bölümünde tamamladım. 2011 yılında lisans eğitimimi tamamladım. 2011 yılında Fırat Üniversitesi Bilgisayar Mühendisliği bölümünde Yüksek Lisans eğitimine başladım. 2013 yılından itibaren Fırat Üniversitesinde Araştırma Görevlisi olarak çalışmaktayım.