KOCAELİ ÜNİVERSİTESİ * FEN BİLİMLERİ ENSTİTÜSÜ

MD YAPILARI İÇİN TANIM DEĞERİ BELİRLEME

YÜKSEK LİSANS

Adem ATALAY

Anabilim Dalı: Matematik

Danışman: Yard. Doç. Dr. Hülya KODAL SEVİNDİR

KOCAELİ, 2012

KOCAELİ ÜNİVERSİTESİ * FEN BİLİMLERİ ENSTİTÜSÜ

MD YAPILARI İÇİN TANIM DEĞERİ BELİRLEME

YÜKSEK LİSANS TEZİ

Adem ATALAY

Tezin Enstitüye Verildiği Tarih: 04 Ocak 2012
Tezin Savunulduğu Tarih: 03 Şubat 2012

Tez Danışmanı
Yrd.Doç.Dr. Hülya KODAL SEVİNDİR
(.....)

Üye
Prof.Dr. Zahir MURADOĞLU
(.....)

Üye
Doç.Dr. Cemil ÖZ
(.....)

KOCAELİ, 2012

ÖNSÖZ ve TEŞEKKÜR

Tarihten bu yana insanlar bazı bilgileri gizli yollarla taşıma ihtiyacı duymuştur. 4000 yıl önce Mısırlı bir kâtibin efendisinin hayatını anlatırken hiç görülmemiş hiyeroglifler kullanması ile başlayan gizli yazı tekniği, milattan önce 5-6'ıncı yüzyıllarda askeri istihbaratta gizliliğin gerekmesi nedeniyle askeri alana girerek gelişmeye başlamıştır. Günümüz teknolojisinin baş döndürücü hızı göz önüne alındığında, bankacılık işlemlerinden haberleşmeye kadar birçok sayısal tabanlı uygulamaların kullanımının giderek vazgeçilemez olduğu fark edilebilmektedir. Bu kullanım sıklığı kötü amaçlı kullanılabileceği için sayısal olarak iletilen bilgilerin güvenliği her geçen gün daha fazla önem kazanmaktadır. Günümüz dijital dünyasının güvenlik ihtiyaçlarının karşılanmasında kriptografik sistemler kullanılır. Bu sistemler verilerin şifreli olarak güvensiz yollarda güvenli bir biçimde taşınmasını sağlamayı amaçlar. Dijital güvenlikli yapılarda şifreleme ve şifre çözme elemanlarının yanında, iletilen verinin bütünlüğünün doğrulanması gereklidir. Doğrulama yapmak için özet fonksiyonları kullanılır ve özet fonksiyonları temelde birer tek yönlü fonksiyonlardır. Bu fonksiyonlar kriptografik sistemlerin en hayati parçalarından birisidir. Tek yönlü fonksiyonların görüntülerinden başlangıç değerini bulma probleminin zorluğu tek yönlü fonksiyonları güvenli parçalar haline getirmektedir. 1980 yılında Hellman, Hellman tabloları yardımıyla deneme yanılma yönteminden daha verimli bir yol geliştirmiştir. 2003 yılında Oecslin, işlem karmaşıklığı bakımından Hellman tablolarından daha verimli çalışan Rainbow tablolarını geliştirmiş ve kullanmıştır. Bu yöntemler, verilen bir değerin fonksiyonun çıkışı olabilmesi için olası giriş değerini hesaplamada kullanılabilir. Merkle-Damgård (MD) güçlendirmesi kullanarak tasarlanan özet fonksiyonları girdi değerinin parçalar halinde ve sıralı olarak belli bir sıkıştırma fonksiyonundan geçirilerek kullanılması ile çalışır. Bu şekilde tasarlanan tek yönlü fonksiyonun giriş değeri Hellman ya da Rainbow yöntemi kullanılarak hesaplanabilir fakat bu yöntemler her adımda birden fazla sıkıştırma fonksiyonu kullanacağı için tercih edilmezler. Bu tezde, MD güçlendirmesi kullanarak tasarlanmış özet algoritmalarının sıkıştırma fonksiyonları kullanılarak Rainbow tablosu oluşturulmuş ve bu tablolar farklı bir yöntemle uygulanmıştır.

Kriptoloji hayatımda özet fonksiyonlarını ilgi alanım haline getirdiğinden, yönlendirmeleri ve yorumları ile akademik alandaki çalışmalarıma öncülük ettiğinden, başarıyı hedefleyen bakış açısıyla geniş yorumlar yapabilmemi sağladığından ve bu tezdeki önemli katkılarından dolayı Dr. Orhun KARA'ya, tez aşamasındaki yardımları ve değerli yönlendirmelerinden dolayı Yard. Doç. Dr. Hülya KODAL SEVİNDİR'e, üzerimde her zaman hissettiğim bana olan eksiksiz inancından dolayı değerli eşim Nilgün'e ve beni yetiştiren aileme sonsuz teşekkürlerimi ve şükranlarımı sunarım.

İÇİNDEKİLER

ÖNSÖZ i
İÇİNDEKİLER ii
ŞEKİLLER DİZİNİ iii
TABLOLAR DİZİNİ iv
ÖZET v
İNGİLİZCE ÖZET vi
1. GİRİŞ 1
1.1. Blok Şifreleyiciler 3
1.2. Özet Fonksiyonları 4
1.3. Tezin Yapısı ve Sonuçları 5
2. ÖZET FONKSİYONLARI ve MD YAPISI 7
2.1. Güvenlik Gereksinimleri 9
2.2. Merkle-Damgård(MD) Yapısı ve MD Güçlendirmesi 11
2.3. Çevrim Fonksiyonu ve Güvenlik İspatları 12
2.4. Bazı Örnekler: SHA–1 ve MD4 15
2.4.1. SHA–1 15
2.4.2. MD4 17
3. TEK YÖNLÜ FONKSİYONLARDA GİRİŞ DEĞERİ BULMA 19
3.1. Hellman Tablosu Yöntemi 19
3.2. Rainbow Tablosu Yöntemi 24
4. SIKIŞTIRMA FONKSİYONLARININ GÖRÜNTÜLERİNİ KULLANARAK ÖZET FONKSİYONLARININ GİRİŞ DEĞERLERİNİN ELDE EDİLMESİ 28
4.1. Temel Saldırı 29
4.2. Temel Saldırının Genişletilmesi ve Güvenlik İncelemeleri 32
4.2.1. Çıkış fonksiyonlu MD yapıları 33
4.2.2. Rastgele özet değeri oluşturma 34
4.2.3. Sıkıştırma fonksiyonu blok sayısını veya uzunluğunu giriş parametresi olarak kullanan MD yapıları 34
5. YENİ ÖNERİ: YAKIN İLK DEĞER 36
5.1. Güvenlik İhtiyaçları 37
6. BAŞARI ORANLARI VE DENEYSEL SONUÇLAR 39
7. SONUÇLAR 41
KAYNAKLAR 44
ÖZGEÇMİŞ 47

ŞEKİLLER DİZİNİ

Şekil 2.1: Kriptografik özet fonksiyonu 7
Şekil 2.2: Merkle-Damgård yapısı 11
Şekil 3.1: f altında görüntü matrisi 21
Şekil 3.2 : Solda $mt \times t$ büyüklüğünde Rainbow ve sağda $m \times t$ büyüklüğünde t tane klasik tablo yer almaktadır. 26
Şekil 4.1: Rainbow tablosu. Oklar sonraki zincir değerlerini göstermektedir. 31

TABLOLAR DİZİNİ

Tablo 2.1: SHA–1 için başlangıç vektörü 16
Tablo 2.2: SHA–1 adım fonksiyonları 16
Tablo 2.3: MD4 için başlangıç vektörü 17
Tablo 2.4: MD4 adım fonksiyonları 18
Tablo 6.1: Deneysel Sonuçlar. BD: Başarılı Denemeler. UD: Başarısız Denemeler.
TD: Toplam Denemeler. OZK: Ortalama Zaman Karmaşıklığı.
OYAS: Ortalama Yanlış Alarm Sayısı 40

ZAMAN-HAFIZA ÖDÜNLEŞİM TABLOLARI

Adem ATALAY

Anahtar Kelimeler: Kriptografik özet fonksiyonları, sıkıştırma fonksiyonu, ilk görüntü dayanıklılığı, ödünleşim, Hellman tablosu, Rainbow tablosu, tek yönlü fonksiyon.

Özet: Özet fonksiyonları dijital imza, veri bütünlüğü, asıllama protokolleri, mesaj asıllama kod (MAC) algoritmaları ve rastgele sayı üreteçleri (RNG'ler) gibi çeşitli uygulamalarda yaygın olarak kullanılan kriptografik yapılardan biridir. Özet fonksiyonlarının tek yönlü olması gereklidir, yani ilk değer dirençli. Bu fonksiyonların enteresan özelliklerinden biri farklı uzunluklarda mesajları sabit uzunluktaki mesajlara dönüştürmeleridir. Genellikle, bu işlem mesajı eşit uzunlukta bloklara bölüp her birini sırayla sıkıştırma fonksiyonundan geçirerek yapılır. Mesaj uzunluğu tamamlama bloğuna ek olarak özet değeri oluşmadan önce son blokla birleştirilir, bu yönteme Merkle-Damgård (MD) güçlendirmesi denir. Bu çalışmada, MD güçlendirmesi kullansa da özet fonksiyonlarının sıkıştırma fonksiyonlarından oluşturulan Rainbow tabloları ile ilk değer bulmaya yönelik bir yöntem önerilmiştir. MD güçlendirmesinin üstesinden gelmek için Rainbow veya Hellman yöntemlerinin klasik uygulanmasının aksine sütun fonksiyonları ilk değerler kümesinden oluşturulmuştur. Farklı bir yaklaşım olarak, ilk değeri bulmak için verilen değerin tablodaki pozisyonu kullanılmıştır. Zincir değerinin ve özet değerinin uzunluğu n olan bir fonksiyonda verilen bir özet değerinin ilk değerini bulmanın iş yükü $2^{2n/3}$ hafıza kullanıp $2^{2n/3}$ adım işlem yapmaktır. Çalışmada, çıkışta farklı fonksiyon kullanan, mesaj blok uzunluğunu girdi olarak alan veya rastgele tuz değeri kullanan bazı geliştirilmiş MD yapıları için ilk değer bulma yöntemine uzantılar ekleyerek yöntemin çalışmasının sağlanabileceği gösterilmiştir. Ayrıca, yakın ilk değer kavramı önerilmiş ve bu kavram için yakın ilk değer bulmaya yönelik yöntem sunulmuştur. Bu yöntem zincir ve özet değerinin uzunluğu eşit olmayan fonksiyonlar da göz önüne alınarak genelleştirilmiştir. Deneysel sonuçlar elde edilmiş ve 40 bit özet uzunluğuna sahip bir fonksiyona ait ilk değer deneme yanılma yöntemiyle yaklaşık bir hafta sürecekken bu yöntemle standart bir bilgisayar kullanılarak bir dakika içinde ele geçirilmiştir.

TIME-MEMORY TRADE-OFF TABLES

Adem ATALAY

Keywords : Cryptographic hash function; compression function; preimage resistance; trade-off; Hellman table; Rainbow table; one-way function.

Abstract : Hash functions are one of the ubiquitous cryptographic functions used widely for various applications such as digital signatures, data integrity, authentication protocols, message authentication code (MAC) algorithms, random number generators (RNGs), etc. Hash functions are supposed to be one-way, i.e., preimage resistant. One interesting property of hash functions is that they process arbitrary-length messages into fixed-length outputs. In general, this can be achieved mostly by applying compression functions onto the message blocks of fixed length, recursively. The length of the message is incorporated as padding in the last block prior to the hash, a procedure called the Merkle-Damgård strengthening. In this thesis, we introduce a new way to find preimages on a hash function by using a Rainbow table of its compression function even if the hash function utilizes the Merkle-Damgård (MD) strengthening as a padding procedure. To overcome the MD strengthening, we identify the column functions as representatives of certain set of preimages, unlike conventional usage of Rainbow tables or Hellman tables to invert one-way functions. As a different approach, we use the position of the given value in the table to invert it. The workload of finding a preimage of a given arbitrary digest value is $2^{2n/3}$ steps by using $2^{2n/3}$ memory, where n is both the digest size and the length of the chaining value. We give some extensions of the preimage attack on certain improved variants of MD constructions such as using output functions, incorporating the length of message blocks or using random salt values. Moreover, we introduce the notion of "near-preimage" and mount an attack to find near-preimages. We generalize the attack when the digest size is not equal to the length of chaining value. We have verified the results experimentally, in which we could find a preimage in one minute for the 40-bit hash function, whereas the exhaustive search took roughly one week on a standard PC.

1. GİRİŞ

Kriptografi yunanca gizli anlamına gelen kryptós ve yazmak anlamına gelen gráfo kelimesinin birleşiminden, kriptoloji ise yine gizli anlamına gelen kryptós ve bilim anlamına gelen logia kelimesinin birleşiminden oluşmuş kelimelerdir ve tanınmayan üçüncü kişilere karşı iletişim sistemlerini korumak için yapılan çalışmalara verilen isimlerdir. Temelde kriptoloji kavramı kriptografi ve kriptoanaliz olarak ikiye ayrılır. Kriptografi veri transferinin güvenliğini sağlayan parçaların oluşturulması ile ilgilenen bölümdür. Kriptoanaliz ise bu güvenli parçaların güvenlik analizleri ve bu tasarımların kırılması ile ilgilenen bölümdür. Kriptoloji, bu iki bölümün birbirine karşı direnişi ile yaşamakta ve gelişmektedir. Modern kriptografi matematiğin, bilgisayar ve elektronik bilimlerinin kesişiminden oluşmaktadır ve günlük hayatımızda para çekme makinelerinden bilgisayar şifrelerine kadar her alanda kullanılmaktadır.

Kriptolojinin tarihine bakarsak, yakın zamana kadar şifreleme ile eş anlamlı olarak ve hassas bilgilerin taşınmasında kullanıldığı görülür. Şifreleme ile bilginin başkası tarafından okunurken anlamsız bir hale gelmesi hedeflenmiştir. Bilgisayarlar hayatımızın önemli bir parçası haline gelmeden önce kriptoloji özellikle askeri alanlarda kullanılmış ve her çağda gelişme göstermiştir. Birinci ve İkinci Dünya Savaşı'nda kriptoloji birçok ülkenin kaderinin değişmesini sağlayan bir bilim olarak bilinir.

Modern kriptoloji hesaplaması zor olan yaklaşımları temel alarak gizli algoritmalar tasarlar. Bu tasarımlarda temel hedef, kötü amaçlı kişilerin iletilen veriyi ele geçirememesini sağlamaktır. Tasarımlar teorik olarak belli bir çalışma sonunda kırılabilir fakat bu çalışma miktarı gerçek hayatta pratik olarak imkânsızdır. Bu kabul tasarımları hesaplama yöntemine göre güvenli kılar. Bazı yöntemlerin matematiksel olarak deneme yanılma yöntemi dışında kırılamayacağı ispat edilebilir. Buna rağmen bu yöntemlerin uygulanması teorik olarak kırılabilen fakat pratikte kırılması imkânsız olan bir algoritmaya göre çok zordur.

Bilgisayarların toplum hayatında önemli bir yer kaplamaya başlamasıyla hayatın dijitalleşmesi başlamış ve böylece kriptoloji, askeri amaçların dışında toplumun ihtiyaçları için de kullanılmaya başlanmıştır. Özel bilgilerin dijital hayatta güvenli iletilmesi amacıyla IBM tarafından Lucifer algoritması tasarlanmış ve bu algoritma daha sonra biraz değiştirilerek Amerikan Federal Bilgi İşleme Standardı (FIPS) olarak kabul edilmiştir. Kabul edilen bu sürüm veri şifreleme standardı (DES) olarak bilinmektedir. 1960 ve 2000 yılları arasında kriptografi ve kriptoanaliz bilgisayarların gelişimine paralel bir şekilde çok hızlı bir ilerleme kaydetmiştir. Bu ilerleme standart olarak kullanılan DES algoritmasını kullanılamaz hale getirmiş ve yeni bir standart zorunluluğunu doğurmuştur. 1998 yılında Rijmen ve Daemon tarafından Rijndael algoritması tasarlanmış ve bu algoritma 2001 yılında gelişmiş şifreleme standardı (AES) olarak kabul edilmiştir. Günümüzde bu algoritmanın güvenliğine karşı bir yöntem olmadığından halen yaygın olarak kullanılmaktadır.

Kriptografik algoritmaların sadece taraflar arasında bilinen gizli bir anahtar parametresi vardır. Bu parametre hem şifrelemede hem de şifre çözmede kullanılır. Bu simetrik yapısından dolayı bu algoritmalara simetrik anahtarlı yapılar denir. Yapıdaki aynı anahtarı kullanma zorluğu anahtar dağıtımında güvenlik ihtiyacı oluşturmuş ve buna karşılık 1976 yılında Diffie ve Hellman [1] herkesin kendine ait özel anahtarının kullanılabildiği asimetrik bir yapı önermiştir. Bu yapıda herkes kendisine gizli bir anahtar belirler ve bu anahtardan üretilen yeni bir anahtarı genel anahtar kabul edip dağıtır. Yapıdaki temel nokta gizli anahtardan genel anahtarın üretilme zorluğudur. Bu zorluk sonlu logaritma problemi gibi çözümü zor olan matematiksel problemlerle üretilir ve böylece temel noktanın güvenliği matematiksel olarak ispat edilebilir.

Asimetrik yapı ile oluşturulmuş ilk şifreleyici 1978 yılında Rivest, Shamir ve Adleman tarafından RSA [2] adında bir algoritma ile önerilmiştir. Bu algoritma matematikteki büyük sayıların çarpanlarına ayrılma zorluğunu kullanarak geliştirilmiştir. Asimetrik yöntemler son zamanlardaki dijital imza çalışmalarında yaygın olarak kullanılmakta ve bundan dolayı kriptoloji camiasında en sıcak konu olarak bilinmektedir.

Kriptolojinin temel yapılarından birisi ise özet fonksiyonlarıdır. Bu tez, özet fonksiyonlarının güvenlik gereksinimlerinden birisi olan ilk görüntü dayanıklılığı üzerine bir çalışmadır. Özet fonksiyonları çevrim fonksiyonu olarak blok şifreleyici tarzında fonksiyonlar kullandığı için 1.1 bölümünde blok şifreleyicilerden kısaca bahsedilecektir. 1.2 bölümünde ise özet fonksiyonlardan kısaca bahsedilecek ve özet fonksiyonları hakkında geniş bilgi 2. bölümde verilecektir. Çalışmanın yapısı ve çalışma sonunda elde edilen verilerin kısa açıklaması 1.3 bölümünde bulunabilir.

1.1. Blok Şifreleyiciler

Genel olarak blok şifreleyiciler n bit uzunluklu verileri k bit uzunluğunda anahtar kullanarak n bit uzunluklu şifreli veri haline getiren yapılardır. Günümüzde blok şifreleyicilerde kullanılan temel yapılar Feistel ve Substitution-Permutation-Networks (SPNs) şifreleyicileridir. Blok şifreleyicilerde gizli olan tek parça anahtardır [3]. Bilgiye ulaşması istenmeyen kişinin anahtar dışında şifreleyici hakkında her şeyi bildiği kabul edilir ve bu prensibe Kerkhoff prensibi denir. Kriptoanaliz işleminde bu kural esas alınır ve önemli olan gizli anahtarı bulma yöntemidir. Anahtar ele geçirildiğinde kripto sistem ne kadar karmaşık olursa olsun sistem kırılmış sayılır. Bu amaçla yapılan temel saldırı çeşitleri aşağıda verilmiştir:

1. Sadece Şifreli Metin ile Saldırı: Bu senaryoda saldırganın sadece şifreli metinleri bildiği ve açık metin hakkında yeterli bilgisinin olmadığı varsayılır. Şifreli metinden yola çıkılarak ilgili anahtara ulaşılmaya çalışılır.

2. Bilinen Açık Metin ile Saldırı: Bu senaryoda ise saldırgan yeterli sayıda açık metinleri ve bu açık metinlere karşılık gelen şifreli metinleri bilir. Bu metin çiftlerinden kullanılan anahtarı ele geçirmeye çalışır.

3. Seçili Açık Metin ile Saldırı: Bu senaryo, bilinen açık metin ile saldırı senaryosuna çok yakındır. Yapıda saldırganın şifreleme kutusuna erişebildiği kabul edilir ve istediği açık metinleri şifreleyerek karşılık gelen şifreli metinleri kullanır. Bu metin çiftleri kullanarak ilgili anahtara ulaşmaya çalışır.

4. Uygun Seçilmiş Açık ve Şifreli Metinler ile Saldırı: Bu senaryo da saldırganın hem şifreleme hem deşifre çözme kutularına erişiminin olduğu kabul edilir. Saldırgan istediği şekilde açık metinden şifreli metin ve şifreli metinden açık metin elde edebilir. Bu ikililer ile kullanılan anahtarı bulmaya çalışır.

5. İlişkili Anahtar Saldırısı: Bu senaryo diğer senaryolardan biraz farklıdır. Yine tek bilinmeyen veri anahtar kabul edilir fakat saldırganın elinde birden çok anahtar olduğu ve bu anahtarlar arasındaki ilişkiyi saldırganın bildiği varsayılır. Saldırgan bu bilgileri kullanarak doğru anahtarı bulmaya çalışır. Bu saldırının gerçeklenmesi durumu diğer senaryolarla birleştirilerek oluşur.

Blok şifreleyiciler kriptoloji dünyasının yapı taşlarından birisidir. Özet fonksiyonlarının sıkıştırma fonksiyonları blok şifreleyicilerin kullanım yerlerinden birisidir. Sıkıştırma fonksiyonlarında kabuller yukarıdakilerden farklı olmasına rağmen yine benzer senaryolar kullanılabilmektedir. Yukarıdaki senaryolar her zaman deneme yanılma saldırısından daha pratik bir yöntem geliştirmeyi hedefler. Deneme yanılma yöntemi, yeterli deneme sonucunda kesin olarak anahtarı bulabileceğinden kurulan senaryolarda karmaşıklığın üst sınırını deneme yanılma yöntemi için gerekli zaman ve hafıza karmaşıklığı belirler.

1.2. Özet Fonksiyonları

Özet fonksiyonları, herhangi bir verinin o veriye özel bir izini üretmek için kullanılırlar. Bu izin benzerinin çok zor bulunabildiği varsayıldığından, bu iz ile verinin gerçekten değiştirilmediğinin yani bütünlüğünün kontrolü yapılabilir. Hayatın her noktasında verinin bütünlüğü çok önemlidir. Dijital hayat, çok sayıda verinin dijital olarak hızlı bir şekilde taşınması üzerine inşa edilmiştir. Veriler dijital yöntemlerle ve gürültü dolu yollarla taşındığı için her verinin bütünlüğünün kontrol edilmesi kaçınılmazdır. Bu durum, özet fonksiyonlarında güvenlik yanında performans ihtiyacını da beraberinde getirmektedir.

Günümüz literatüründe birçok özet fonksiyonu mevcuttur. İlerleyen bölümlerde temel olması açısından MD4 ve SHA1 algoritmaları detaylı olarak incelenecektir. Bu

fonksiyonlar Rivest tarafından geliştirilen MD4 algoritmasını temel alan yapılardır. Bunların içinden SHA1 algoritması standart olarak kullanılmasına rağmen bu yapıların birçoğu çakışma bulunarak kırılmış, geri kalanlarının ise güvenlik açıkları tespit edilmiştir. Çakışma iki farklı verinin fonksiyon sonrasında aynı özet değerini vermesi anlamına gelir. Temelde dört farklı çakışma senaryosu vardır, bunlar:

1. Çakışma Saldırısı: Bu senaryoda hedef n bit özet değeri üreten bir fonksiyonda aynı özeti üretebilecek iki farklı mesajın $2^{n/2}$ işlemden daha az işlemle bulunmasını sağlamaktır.

2. Yakın-Çakışma Saldırısı: Bu senaryo çakışma saldırı ile benzerdir. Saldırgan aynı özeti veren çiftler yerine birbirine çok benzer özet değerlerini veren mesajlar bulmaya çalışır.

3. Sanki-Çakışma (Pseudo Collision) Saldırısı: Özet fonksiyonları genellikle kendilerine özel sabit bir değer ile başlar. Bu senaryoda saldırgan farklı başlangıç derleri ve farklı veriler kullanarak aynı özet değerini elde etmeye çalışır.

4. Sanki-Yakın-Çakışma Saldırısı: Bu senaryo ise yukarıdaki senaryoların birleşimi olarak kabul edilebilir. Saldırgan farklı başlangıç derleri ve farklı veriler kullanarak birbirine yakın özet değerleri elde etmeye çalışır.

Bu senaryolardan herhangi birinin gerçekleşmesi özet fonksiyonlarının kırılmış kabul edilmesi veya güvenliğinin sarsılması için yeterlidir [4]. Saldırgan için daha zor yöntemler ise ilk ve ikinci ilk değer saldırılarıdır. Bu saldırılar başarılı olursa özet fonksiyonu kullanılamaz hale gelir. Verilen bir özet değerini üreten veriyi bulmak n bit özet değeri üreten bir fonksiyonda 2^n işlemle yapılabilir. Daha az işlem gerektiren her yöntem özet fonksiyonunun kırılması anlamını taşır.

1.3. Tezin Yapısı ve Sonuçları

Bu çalışmada, temel olarak Merkle-Damgård (MD) yapısındaki özet fonksiyonlarının ilk değerlerini bulmaya yönelik bir yapı geliştirmek hedef alınmıştır. Akan

şifreleyicilerde tek yönlü fonksiyonların giriş değerlerinin bulunabilmesi için 1980 yılında Hellman [5] tarafından ödünleşim tabloları önerilmiş ve bu tablolar 2003 yılında Oecslin [6] tarafından geliştirilerek kullanılmıştır. Bu tez çalışmasında öncelikle bu tablolardan bahsedilecek ve bu tabloların Merkle-Damgård (MD) yapısındaki özet fonksiyonlarına uygulanması anlatılacaktır.

Verilen bir özet değerini üreten veriyi bulmak, diğer bir söyleyişle ilk değeri bulmak, n bit özet değeri üreten bir fonksiyonda 2^n işlemle yapılabilir. Bu çalışmadaki temel sonuç, Merkle-Damgård (MD) yapısındaki özet fonksiyonlarında bu sınırın 2^n yerine $2^{2n/3}$ olabileceğinin gösterilmesidir. Çalışmada çıkarılan sonucun pratik olarak gösterilmesi için iki farklı özet fonksiyonu tasarlanmış ve yöntem bu fonksiyonlar üzerinde başarıyla uygulanmıştır.

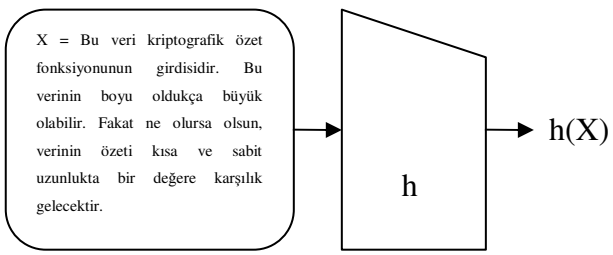
Tezin ikinci bölümünde özet fonksiyonları ve Merkle-Damgård (MD) yapısından bahsedilecektir. Aynı bölümde, özet fonksiyonlarının güvenlik gereksinimleri, çevrim fonksiyonları, bu fonksiyonların güvenlik ispatları ve bazı özet fonksiyon örnekleri incelenecektir. Üçüncü bölümde tek yönlü fonksiyonlarda giriş değerini bulma yöntemi anlatılacak ve bu bölüm için öncelikle Hellman ve Rainbow tablolarından bahsedilecektir. Dördüncü bölümde ise sıkıştırma fonksiyonlarının görüntülerinin kullanılarak özet fonksiyonlarının giriş değerlerinin bulunmasından bahsedilecektir. Bu bölüm temel saldırı ve bu saldırının genişletilmesi ile başlayacaktır. Yakın ilk değer kavramı önerisi ve güvenlik ihtiyaçları beşinci bölümde incelenecektir. Son bölüm elden edilen deneysel sonuçlardan ve bölümde yapılan önerilen bu saldırı yönteminin başarı oranları ve deneysel sonuçlarından oluşacaktır. Yedinci bölümde çalışmanın sonucundan bahsedilerek tez bitirilecektir.

2. ÖZET FONKSİYONLARI ve MD YAPISI

Özet fonksiyonları kriptolojinin birçok uygulamasında kullanılan en hayati parçalarından birisidir. Bu uygulamalara, dijital imza, asıllama protokolleri, bilgi doğrulama protokolleri, veri bütünlüğü, rastgele sayı üreteçleri ve kimlik doğrulama protokolleri örnek olarak gösterilebilir.

Kriptolojik özet fonksiyonları değişik uzunluklarda verileri belli uzunlukta bir veriye çeviren sabit fonksiyonlardır. Bu özellikte birçok fonksiyon bulmak mümkündür. Özet fonksiyonları ekstra özelliklere sahip olduğu zaman asıllama ve bilgi güvenliğinde çok değerli bir araç haline gelir [7]. Anahtarlı özet fonksiyonlarında ise girdi tek değişkenden oluşmaz. Değişik boyda bir veri ve belirli boyda bir anahtarı girdi olarak kabul edip sabit boyda bir çıktı üretirler.

Özet fonksiyonları yakın zamanda çok değer kazanmış önemli bir araçtır. Tam olarak güvenli bir özet fonksiyonunun nasıl tasarlanması gerektiği ve hangi parçaları içermesi gerektiği standart olarak bilinmediğinden kriptografik özet fonksiyonlarının tasarım ve analizleri son zamanların en çok araştırılan konularındandır. National Institute of Information and Standarts (NIST)'ın 2007 yılında başlattığı SHA–3 yarışması standart olarak kullanılabilecek güvenli, hızlı ve donanım üzerinde az yer kaplayan özet fonksiyonunu bulmak amacını taşımaktadır [8].



Şekil 2.1: Kriptografik özet fonksiyonu

Tanım (Anahtarsız özet fonksiyonu) : n pozitif bir sayı olmak üzere. h, n-bit çıktı boyu olan bir özet fonksiyonu, farklı uzunlukta verileri gerekirci (deterministic) yollarla sabit n-bit uzunluğunda bir veriye dönüştüren bir fonksiyondur. Aşağıdaki gibi gösterilebilir:

$$h:\{0,1\}^{*} \rightarrow \{0,1\}^{n} \tag{2.1}$$

Tanım (Anahtarlı özet fonksiyonu) : l, n pozitif sayılar olsun. h, n-bit çıktı boyu ve l-bit anahtar boyu olan bir özet fonksiyonu, farklı uzunlukta verileri l-bit uzunluğunda bir anahtarla birlikte gerekirci yollarla sabit n-bit uzunluğunda bir veriye dönüştüren bir fonksiyondur. Aşağıdaki gibi gösterilebilir:

$$h:\{0,1\}^{*} \times \{0,1\}^{l} \rightarrow \{0,1\}^{n} \tag{2.2}$$

Yukarıda tanımı verilen özet fonksiyonlarının kriptografik anlamda kullanılabilmesi için herhangi bir x değeri alındığında $h(x)$ değeri çok kolay bir şekilde hesaplanırken verilen bir $h(x)$ değeri için x orijinal verisinin bulunmasının pratik olarak çok zor olması gerekmektedir.

Özet fonksiyonlarının tanımı gereği aynı $h(x)$ değerini üreten birden fazla x değerleri vardır. Bu özellikte göz önüne alınırsa kriptografik özet fonksiyonundan aşağıdaki üç özelliği sağlaması beklenir.

- İlk görüntü dayanıklılığı: Verilen bir $h(x)$ özet değerine karşılık, bu değeri veren x değerini bulmak pratik olarak zor olmalıdır.

- İkinci görüntü dayanıklılığı: Verilen x ve $h(x)$ değerleri için aynı $h(x)$ değerini veren farklı bir x' değerini bulmak pratik olarak zor olmalıdır.

- Çakışma dayanıklılığı: Herhangi bir x değeri için aynı $h(x)$ değerini veren ikinci bir x' değerini bulmak pratik olarak zor olmalıdır.

Çakışma dayanıklılığı, sağlanması en zor olan özelliktir ve bu özelliğe en sağlam özellik denir. Bu durum, çakışma dayanıklılığının en kolay aşılabilen özellik olduğu anlamına gelmektedir [9]. Bundan dolayı genellikle saldırılar ilk olarak çakışma bulmaya yönelik yapılır. Çakışmaya dayanıklı bir özet fonksiyonu aynı zamanda ikinci görüntü dayanıklılığına da sahiptir.

2.1. Güvenlik Gereksinimleri

Tek yönlü bir fonksiyon, ilk görüntü dayanıklılığını, ikinci görüntü dayanıklılığını ve çakışma dayanıklılığını sağladığında bu fonksiyon özet fonksiyonu olarak kullanılabilir. Bu özelliklerden emin olabilmek için kullanılan fonksiyonun yapı taşlarının incelenmesi gerekmektedir [10]. Genel özet fonksiyonu yapısında, özeti alınacak veri eşit uzunlukta parçalara bölünüp bu parçalar peş peşe çevrim fonksiyonu denen bir f fonksiyonundan geçirilir. Çevrim fonksiyonuna sıkıştırma fonksiyonu da denebilir. Bu fonksiyon sabit uzunlukta giriş ve çıkış boylarına sahiptir.

Genel özet fonksiyonu yapısında, X verisinin özeti alınırken bu veri $X = X_0 \| X_1 \| \cdots \| X_{n-2} \| X_{n-1}$ olmak üzere eşit uzunlukta n parçaya ayrılır. Son parça uzunluğu tam istenen uzunlukta olmaması durumunda bu parçaya tamamlama yapılır. Tamamlama uygulamaya göre değişebilir. En çok kullanılan özet fonksiyonları göz önüne alınırsa, verinin sonunda veri bitişini bilmek adına bir tane '1' biti eklenir, daha sonra veri boyu parça boyunun katı olana kadar $0\cdots0l$ bitleri eklenir. Burada l orijinal verinin boyunu, $0\cdots0$ ise gerektiği kadar sıfır bitini göstermektedir.

Çevrim fonksiyonu f olan bir h özet fonksiyonunda işlem akışı aşağıdaki gibi gösterilebilir:

$$
\begin{aligned}
&H_0 = IV \\
&H_i = f(X_i, H_{i-1}) \qquad i = 1, 2, \ldots, n \\
&h(X) = H_n
\end{aligned}
$$

(2.3)

Yukarıda ilk değer olan $H_0 = IV$ değerine başlangıç değeri denir. Özet fonksiyonunun ilk ve ikinci görüntü dayanıklılığını sağlaması f çevrim fonksiyonunun doğrusal bir özelliğe sahip olmaması ile doğrudan ilişkilidir. Fonksiyonun girdi bitleri ve çıktı bitleri birbirleri ile bağlantılı olmamalıdır. Bu varsayımlar altında güvenlik gereksinimleri aşağıdaki gibi sıralanabilir:

- Bir özet fonksiyonunun, ilk veya ikinci görüntü dayanıklılığına karşı uygulanabilecek en basit saldırı f fonksiyonunun ters görüntüsünü hesaplamaktır. $X_i' \neq X_i$ ve $f(X_i', H_{i-1}) = H_i$ olacak şekilde bir X_i' bulunması pratik olarak zor olmalıdır. Bunun için f fonksiyonu H_i altında bire-bir fonksiyon olması gereklidir. Aynı durum çakışma dayanıklılığı için de düşünülebilir. Çevrim fonksiyonu f, H_i altında bire-bir değilse $f(X_i', H_{i-1}) = f(X_i, H_{i-1})$ eşitliği sağlayan iki farklı $X_i' \neq X_i$ değerleri bulunabilir. Bu durum $X^1 = X_0 \| X_1 \| \cdots \| X_i \| \cdots \| X_{n-2} \| X_{n-1}$ ve $X^2 = X_0 \| X_1 \| \cdots \| X_i' \| \cdots \| X_{n-2} \| X_{n-1}$ şeklinde iki mesajın özet değerlerinin aynı olması yani çakışması anlamına gelir.

- İkinci görüntü dayanıklılığına karşı farklı bir mesaj bulma problemi, saldırganın mesajın X_j'inci parçasını X_j' ile değiştirip, $i > j$ olacak şekilde i'inci adımda H_{i-1}, H_{i-1}' değerlerini yeni bir mesaj parçası ile aynı noktada buluşturması ile sağlanabilir. Bir özet fonksiyonunda $f(X_i', H_{i-1}') = f(X_i, H_{i-1}) = H_i$ koşulunu sağlayan X_i', X_i değerlerinin bulunması pratik olarak zor olması gerekir. Bu senaryo çakışma içinde kullanılabilir. Bundan dolayı yukarıdaki gereksinim çakışma dayanıklılığı için de önemlidir.

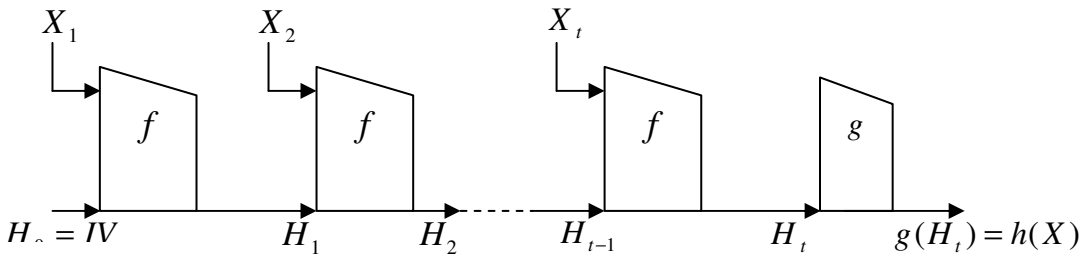
- $f(X_i, H_{i-1}) = H_{i-1}$, olacak şekilde bulunan (X_i, H_{i-1}) ikilisine f fonksiyonunun sabit noktası denir. Sabit nokta veren ikili, X_i seçildiği zaman H_{i-1}'in hesaplanması veya rastgele (X_i, H_{i-1}) ikilisinin bulunması ile elde edilebilir. H_{i-1} değerleri, sabit bir X_i değeri altında düzgün dağıtılmışsa gerekli değerin

bulunması deneme yanılma saldırısı kadar vakit alır. Bu özellik kullanılarak ilk veya ikinci görüntü dayanıklılığına karşı iyi bir sonuç elde edilemez. En iyi sonuç çakışma dayanıklılığına karşı elde edilebilir. Bu zayıflık ise mesajın özet değeri hesaplanmadan önce sonuna mesaj uzunluğunun eklenmesi ile çözülebilir.

Yukarıda anlatılan gereksinimler, bir özet fonksiyonunun ilk görüntü saldırısı, ikinci görüntü saldırısı, sanki ilk görüntü saldırısı, sanki ikinci görüntü saldırısı, çakışma saldırısı, farklı başlangıç değerleri ile çakışma saldırısı ve sanki çakışma saldırısı gibi genel saldırılara karşı dayanıklı olması için gereklidir. Bu gereksinimler sağlanması durumunda yine de, özet fonksiyonunun güvenliği hakkında kesin bir bilgiye sahip olunamaz.

2.2. Merkle-Damgård(MD) Yapısı ve MD Güçlendirmesi

Merkle-Damgård yapısı Ralph Merkle [11] ve Ivan Damgård [12] tarafından 1989 yılında bağımsız bir şekilde sunulmuştur. MD yapısı, Şekil 2.2'de gösterildiği gibi f çevrim fonksiyonunun mesaj parçaları ve ara durum değerleri kullanılarak artarda çalıştırılması ile oluşan bir yapıdır.



Şekil 2.2: Merkle-Damgård yapısı

Özet değeri alınacak X verisi, her biri l-bit boyunda $X = X_0 \| X_1 \| \cdots \| X_{t-2} \| X_{t-1}$ olacak şekilde t parçaya ayrılır. Başlangıç değeri $H_0 = IV$ olarak ayarlanır ve $f(X_i, H_{i-1}) = H_i, i = 1, 2, \ldots, t$ işlemi t defa tekrarlanır. En son oluşan H_t değeri özet

değeri olarak kabul edilebilir veya bu değer farklı bir g bitiş fonksiyonundan geçirilerek $g(H_t) = h(X)$ özet değeri elde edilir.

Bu yapıda mesaj boyu en son blokta işlendiği için, MD yapısı akan mesajların özetini almakta kullanılabilir. Mesajın sonu gelene kadar mesaj uzunluğunun bilinmesine gerek yoktur [13]. Veri uzunluğunun en son bloktaki kullanım yapısına MD güçlendirmesi denir. Verinin bittiğini belirtmek için en son bitten sonra bir tane '1' biti eklenir. Her X_i bloğunun uzunluğu l-bit olacağından, son blok uzunluğu $l-64$ olana kadar '0' biti ile doldurulur. Son olarak veri boyunun 64 bitlik ifadesi eklenerek son blok uzunluğu da l-bite tamamlanmış olur. Bu işleme tamamlama denir. Her zaman son blokta tüm verinin uzunluğu yer almalıdır, bundan dolayı uzunluk l değerinin bir tamsayı katının $65 < k < l$ eksiğine eşitse yeni bir blok oluşturulması gerekmez. Veri uzunluğunun l'in bir tamsayı katından eksiği $k < 65$ olması durumunda ise '1' biti eklendikten sonra tüm veri boyunun 64 bitlik gösteriminin ekleneceği kadar yer kalmadığı için yeni bir blok eklenir ve bu blok uzunluğu $l-64$ olana kadar '0' biti ile doldurulur. Geriye 64 bitlik yer kaldığından, tüm veri boyu buraya yazılır ve verinin bloklara ayrılması işlemi bitmiş olur. Burada kullanılan 64 değeri, sabit bir değer değildir. Kullanılan fonksiyona göre değişiklik gösterebilir.

2.3. Çevrim Fonksiyonu ve Güvenlik İspatları

Merkle-Damgård yapısında çakışmaya dayanıklı çevrim fonksiyonu kullanılıyorsa, özet fonksiyonunu ihtiyaç duyulan çakışma dayanıklılık seviyesini sağlar. Dayanıklı çevrim fonksiyonlarının kullanımı aynı zamanda ilk görüntü dayanıklılığı sağlar. Bu bölümde sıkıştırma fonksiyonu bir kara kutu olarak kabul edilecek ve güvenlik seviyeleri hakkında teoriler verilecektir.

Teori 1: H, MD yapısında çevrim fonksiyonu f olsun. Çevrim fonksiyonu f çakışmaya dayanıklı ise H da çakışmaya dayanıklıdır [12].

İspat: H, MD yapısında çevrim fonksiyonu f ve özet boyu t olan bir özet fonksiyonu olsun. Çevrim fonksiyonunun girdi uzunluğunun m bit olduğunu kabul edersek, mesaj bloğunun uzunluğu $l > 0$ olacaktır. Bu durumda $l = m - t + 1$ olarak kabul edilirse, n bit uzunluğunda bir mesajın özet değeri en fazla $n/(m-t+1)+1$ defa çevrim fonksiyonu çağrılarak hesaplanabilir. Bu ispatta a ve b gibi iki değerin birleşimi $a \| b$ olarak gösterilecektir.

Teoriyi ispatlamak için iki ayrı durumu incelemek gerekmektedir. İlk olarak $m - t > 1$ durumu ve daha sonra ise $m - t = 1$ durumu incelenecektir.

i. $m - t > 1$ durumu: Uzun bir $x \in \{0,1\}^*$ mesajının özeti hesaplanırken mesaj $m - t - 1$ bit uzunluğunda bloklara ayrılır. Son blok eksik kaldıysa 0 bitleri ile tamamlanır. Bu durumda tamamlama sonunda uzunluğu $n = |x|$ bit olan mesaj $x = x_1 \| x_2 \cdots \| x_{n/(m-t-1)}$ şeklinde gösterilebilir. Son olarak mesaja $m - t - 1$ bit uzunluğunda $x_{n/(m-t-1)+1}$ bloğu eklenir. Bu blok 0 bitleri ile başlayan ve mesaj uzunluğunun ikili sistemdeki gösterimi ile biten tamamlama bloğudur.

Özet değeri hesaplanırken t bit uzunluklu $h_1, h_2, \ldots.$ zincir değerleri aşağıdaki gibi gösterilebilir.

$$\begin{aligned} h_1 &= f(0^{t+1} \| x_1) \\ h_{i+1} &= f(h_i \| 1 \| x_{i+1}) \end{aligned} \qquad (2.4)$$

Özet değeri ise $H(x) = h_{n/(m-t)+1}$ olacaktır. Çakışma olup olmadığını kontrol etmek için iki farklı $x \neq x'$ mesajlarının $H(x) = H(x')$ şeklinde aynı özet değerini verdiğini varsayalım.

Eğer $|x| \neq |x'| \bmod (m-t)$ ise mesajların son tamamlama blokları $x_{n/(m-t)+1} \neq x'_{n/(m-t)+1}$ olmalıdır. Bu durumda $H(x) = H(x')$ eşitliğinin sağlanması çevrim fonksiyonunda

çakışma olduğunu gösterir ve bu ilk varsayımın tersine olur. Bundan dolayı $|x| = |x'| \bmod(m-t)$ ve $|x'| > |x|$ olduğu kabul edilsin.

Bu durumda aşağıdaki eşitlik doğru olabilir.

$$H(x) = f(h_{n/(m-t)} \| x_{n/(m-t)+1}) = f(h'_{n'/(m-t)} \| x'_{n'/(m-t)+1}) = H(x') \tag{2.5}$$

Eğer $f(h_{n/(m-t)} \| x_{n/(m-t)+1}) = f(h'_{n'/(m-t)} \| x'_{n'/(m-t)+1})$ ve $h_{n/(m-t)} \| x_{n/(m-t)+1} \neq h'_{n'/(m-t)} \| x'_{n'/(m-t)+1}$ durumu sağlanıyorsa, bu mesajlar yine çevrim fonksiyonunda çakışma olduğunu gösterir ve bu ilk varsayımın tersine olur. Mesaj bloklarının eşit olduğunun varsayıldığı durumda bir önceki bloklarda aynı durumla karşılaşılacaktır. Bu şekilde devam edilirse ya çevrim fonksiyonunda çakışma olduğu kabul edilerek durulacak ya da ilk blok olan $0^{t+1} \| x_1$ bloğunun diğer mesajın $h'_i \| 1 \| x'_{i+1}$ olan bir bloğuna eşit olduğu kabul edilmelidir ki bu çelişkidir.

ii. $m-t=1$ durumu: Son olarak $m-t=1$ durumunda yukarıdaki ile aynı kurulum varsayılabilir. Önceki durumdan farklı olarak sadece zincir değerlerini aşağıdaki gibi kabul etmek gerekmektedir.

$$\begin{aligned} h_1 &= f(0^t \| x_1) \\ h_{i+1} &= f(h_i \| x_{i+1}) \end{aligned} \tag{2.6}$$

İlk durumdaki ispat bu durum için de geçerlidir. Buradaki blok değerleri bir bitlik değerlerdir ve ilk zincir değerinin çeşitliliği açısından t bit uzunluğunda y_0 başlangıç değeri alınırsa, zincir değerleri aşağıdaki gibi olacaktır.

$$\begin{aligned} h_1 &= f(y_0 \| x_1) \\ h_{i+1} &= f(h_i \| x_{i+1}) \end{aligned} \tag{2.7}$$

Aynı yöntem uygulanırsa son olarak ortaya çıkan eşitlik bize ya çevrim fonksiyonunda çakışma olduğunu ya da y_0 değerine ait bir ilk değerin

bulunabildiğini gösterecektir. Bu durum ise özet fonksiyonlarının güvenlik gereksinimlerine ters bir durumdur. Böylece çakışmaya dayanıklı çevrim fonksiyonlarından MD yapısı kullanılarak çakışmaya dayanıklı özet fonksiyonları oluşturulur teorisi ispatlanmış olur.

Teori 2: H, MD yapısında çevrim fonksiyonu h ve özet boyu n olan bir özet fonksiyonu olsun. Blok sayısı l olan bir mesaj verildiğinde ikinci görüntüsünü q sorgu ile bulma ihtimali $q \cdot l \cdot 2^{-n}$ ile yukarıdan sınırlıdır.

İspat: Mesajın blok sayısı l olduğu için, özet değeri hesaplanmadan önce l tane ara değer görülür. Rastgele seçilen bir mesaj bloğunun çevrim fonksiyonundan geçtiğinde bu ara değerlerden birisiyle çakışması ikinci görüntü elde etmemizi sağlar. Herhangi bir denemenin l tane ara değerden birisine eşit olma ihtimali $l \cdot 2^{-n}$'e eşittir. Saldırgan q sorgu kullanırsa ikinci görüntü için bir çakışma elde etme ihtimali en fazla $q \cdot l \cdot 2^{-n}$ olur.

2.4. Bazı Örnekler: SHA–1 ve MD4

MD4, MD yapısı ile geliştirilmiş ve birçok özet algoritmasının temelini oluşturan bir özet fonksiyonudur. MD5, SHA–0, SHA–1 ve RIPEMD, MD4'ün kırılmasından sonra güçlendirilerek önerilmiş MD4'e benzer yapıdaki özet fonksiyonlarıdır. Bu bölümde SHA–1 ve MD4 kısaca anlatılacaktır.

2.4.1. SHA–1

SHA, diğer bir ifadeyle SHA–0, özet fonksiyonu NIST tarafından geliştirilmiş ve federal standart olarak (FIPS 180) yayınlanmıştır. 1995 yılında küçük bir değişiklik yapılarak FIPS 180–1 sürümü yayınlanmıştır. Bu sürüm SHA–1 olarak bilinmektedir.

SHA–1, tamamlama yapısında MD güçlendirmesi kullanır ve veri uzunluğunun 64 bit ifadesini verinin sonuna ekler. Bundan dolayı en fazla 2^{64} bit uzunluğundaki verilerin özetini hesaplayabilir. Fonksiyonun ürettiği özet boyu 160 bittir ve veriyi

$l = 512$ bitlik bloklar halinde işleme tabi tutar. SHA–0 ve SHA–1 Merkle-Damgård yapısını ve MD güçlendirmesini kullanan bir özet fonksiyonudur. SHA-1'in başlangıç vektörü beş tane 32-bit sayı ile Tablo 2.1'de gösterilmiştir.

Tablo 2.1: SHA–1 için başlangıç vektörü

A	B	C	D	E
0x67452301	0xEFCDAB89	0x98BADCFE	0x10325476	0xC3D2E1F0

Algoritma, 512 bitlik bir mesaj bloğunu işlerken, bu bloğu mesaj genişletme fonksiyonundan geçirip her adım için kullanılabilecek bir mesaj parçası oluşturur. 80 adımdan oluşan algoritma, her adımda Tablo 2.1'deki A, B, C, D ve E değerlerini günceller. Adımların sonunda oluşan bu değerlerin, ilk değerleri ile D-Ya işlemi sonucu çevrim fonksiyonunun çıkışını verir. En son oluşan çevrim fonksiyon çıktısı özet değeri olarak kabul edilir.

SHA–1 algoritmasının her 20 adımında kullanılan doğrusal olmayan fonksiyon değişir. Kullanılan fonksiyonlar Tablo 2.2'de verilmiştir.

Tablo 2.2: SHA–1 adım fonksiyonları

Adım	**Fonksiyon**
$0 \le j \le 19$	$(B \wedge C) \vee (\overline{B} \wedge D)$
$20 \le j \le 39$	$B \oplus C \oplus D$
$40 \le j \le 59$	$(B \wedge C) \vee (B \wedge D) \vee (C \wedge D)$
$60 \le j \le 79$	$B \oplus C \oplus D$

2.4.2. MD4

MD4, R.Rivest tarafından geliştirilmiş MD yapısı ve MD güçlendirmesi kullanan bir özet alma fonksiyonudur. İlk defa Eurocrypt'90 konferansının kısa bölümünde sunulmuş ve Crypto'90 [14] konferansında yayınlanmıştır.

MD4, 128 bit özet değerini, $l = 512$ bitlik mesaj blokları kullanarak üretir. MD4, tamamlama yapısında MD güçlendirmesi kullanır ve veri uzunluğunun 64 bit ifadesini verinin sonuna ekler. Bundan dolayı en fazla 2^{64} bit uzunluğundaki verilerin özetini hesaplayabilir. Başlangıç değeri 32 bitlik dört sayı ile Tablo 2.3'de verilmiştir.

Tablo 2.3: MD4 için başlangıç vektörü

A	B	C	D
0x67452301	0xEFCDAB89	0x98BADCFE	0x10325476

Algoritma, 512 bitlik bir mesaj bloğunu işlerken, bu bloğu 16 tane 32 bitlik sayı olarak kabul eder ve her adım için bu bir sayı kullanır. 48 adımdan oluşan algoritma, her adımda Tablo 2.3'deki A, B, C ve D değerlerini günceller. Adımların sonunda oluşan bu değerlerin, ilk değerleri ile D-Ya işlemi sonucu çevrim fonksiyonunun çıkışını verir. En son oluşan çevrim fonksiyon çıktısı özet değeri olarak kabul edilir.

MD4 algoritmasının her 16 adımında kullanılan doğrusal olmayan fonksiyon değişir. Kullanılan fonksiyonlar Tablo 2.4'de verilmiştir.

Tablo 2.4: MD4 adım fonksiyonları

Adım	**Fonksiyon**
$0 \le j \le 15$	$(B \wedge C) \vee (\overline{B} \wedge D)$
$16 \le j \le 31$	$(B \wedge C) \vee (B \wedge D) \vee (C \wedge D)$
$32 \le j \le 47$	$B \oplus C \oplus D$

3. TEK YÖNLÜ FONKSİYONLARDA GİRİŞ DEĞERİ BULMA

Tek yönlü bir fonksiyona giriş değeri verildiğinde çıkış değeri kolayca hesaplanabilirken, çıkış değeri kullanılarak giriş değerine hesaplayarak ulaşılamaz. Taranacak N değişik çözüm varsa, çözüme T işlem ve M hafıza kullanarak ulaşılabilir. Kullanılabilecek bir yöntem direk aramadır ($T = N$, $M = 1$). Bu yöntemde olası giriş değerlerinin hepsi denenerek hafıza kullanılmadan doğru giriş değeri bulunabilir. Tablo tarama yönteminde ($T = 1$, $M = N$) ise bütün olası giriş değerlerine karşılık gelen çıkış değerleri N boyutunda bir tabloda sıralı bir şekilde tutulur. Bir çıkış değerini veren giriş değeri, bu tablodan bakılarak bulunabilir. İşlem ile hafıza arasındaki $TM = N$ eşitliğine zaman-hafıza ödünleşimi denir.

1980 yılında Hellman [5], zaman-hafıza ödünleşim yöntemini Hellman tabloları ile yapmış ve bu tablolar yardımıyla deneme yanılma yönteminden daha verimli bir yol geliştirmiştir. 2003 yılında Oecslin [6], işlem karmaşıklığı bakımından Hellman tablolarından daha verimli çalışan Rainbow tablolarını geliştirmiştir.

3.1. Hellman Tablosu Yöntemi

Tek yönlü ve bire bir özelliğe sahip bir f fonksiyonu verildiğinde, bu fonksiyona ait bir görüntüden yola çıkarak giriş değerini bulmak en fazla görüntü kümesinin büyüklüğü kadar zordur. Tanım kümesindeki her değerin tek tek denenmesi yönteminin hafıza gerektirmeyen bir işlem olması gibi bütün değerlerin bir tabloda tutulup görüntünün karşılığını bulmakta işlem gerektirmeyip hafıza gerektiren ve teorik olarak kullanılabilir bir yöntemdir. Bu yöntemler görüntü kümesinin boyu yeterince büyük olduğu durumlarda anlamını yitirir. Kabul ettiğimiz f fonksiyonunu aşağıdaki gibi tanımlayabiliriz:

$$f : \{0,1\}^n \rightarrow \{0,1\}^n \tag{3.1}$$

Bu tanımda f fonksiyonunun $N = 2^n$ tane farklı görüntüsü vardır. Bu görüntüleri hafızada tutmak 2^n hafıza gerektireceği gibi deneyerek bulma yöntemi de 2^n işlem gerektirir. Hellman, 2^n tane bilgi içeren fakat daha az hafıza gerektiren tablolar kullanarak yerden kazanç sağlamıştır. Bu tablolar kullanılarak tablodan okuma yöntemi ile aranan sonuca ulaşılabilir. Tabloda herhangi bir bilgiye ulaşmak belli sayıda işlem ile yapılabileceğinden bu yöntem tablodan okuma yönteminden daha fazla işlem ile yapılır. Bahsedilen hafıza ve işlem arasındaki bağıntıda bulunan uygun değer kullanılarak aranan sonuca daha kolay varılabilir.

Kapladığı hafızadan daha fazla bilgi içeren, Hellman'ın kullandığı tablolar peş peşe çalıştırılan f fonksiyonu ile üretilir. Bu fonksiyon t defa ardı ardına çalıştırıldığında t farklı değer elde edilir. Bu değerlerden ilkine $X_0 = \dot{I}D$ (İlk Değer), sonuncusuna ise $X_{t-1} = SD$ (Son Değer) adını verip bu değerleri aşağıdaki gibi ifade edebiliriz.

$$\dot{I}D = X_0 \xrightarrow{f} X_1 \xrightarrow{f} X_2 \xrightarrow{f} \cdots \xrightarrow{f} X_{t-2} \xrightarrow{f} X_{t-1} = SD \tag{3.2}$$

Yukarıdaki gösterimde her X_k değerine aşağıdaki gibi ulaşılabilir.

$$X_k = f^k(X_0) \tag{3.3}$$

Bu durumda, ilk değerin bilinmesi ile her ara en fazla t işlem sonunda ulaşılabileceğinden ara değerlerin hafızada tutulmasına gerek yoktur. Sadece $X_0 = \dot{I}D$ ve $X_{t-1} = SD$'in hafızada tutulması yeterlidir. Böylece 2 değerlik yer kullanarak t farklı değer içeren bir tablo hazırlanmış olur.

Yukarıdaki t değer içeren satırdan m tane farklı ilk değer için satırlar üretildiği düşünülürse, $2m$ hafıza kullanarak mt tane değer tutan bir tablo hazırlanmış olur. Bu tablo Şekil 3.1'de verilmiştir.

$$
\begin{aligned}
\dot{I}D_1 &= X_{10} \xrightarrow{f} X_{11} \xrightarrow{f} X_{12} \xrightarrow{f} \cdots \xrightarrow{f} X_{1t-2} \xrightarrow{f} X_{1t-1} = SD_1 \\
\dot{I}D_2 &= X_{20} \xrightarrow{f} X_{21} \xrightarrow{f} X_{22} \xrightarrow{f} \cdots \xrightarrow{f} X_{2t-2} \xrightarrow{f} X_{2t-1} = SD_2 \\
&\vdots \qquad\quad \vdots \qquad\quad \vdots \qquad\quad \vdots \qquad\quad \vdots \\
\dot{I}D_m &= X_{m0} \xrightarrow{f} X_{m1} \xrightarrow{f} X_{m2} \xrightarrow{f} \cdots \xrightarrow{f} X_{mt-2} \xrightarrow{f} X_{mt-1} = SD_m
\end{aligned}
$$

Şekil 3.1: f altında görüntü matrisi

Görüntüsü $X_{i,j}$ olan değer tabloda $f(X_{i,j-1}) = X_{i,j}$ şeklinde görünmektedir. Sadece $\dot{I}D_1$ ve SD_1 değerlerinin bilindiği varsayılırsa, $X_{1,j}$ değerini görüntüsüne sahip değeri bulmak için $X_{1,j}$ değeri devamlı f fonksiyonundan geçirilip sonucu SD_1'e eşit olup olmadığı kontrol edilebilir. Bu değer p adımda bulunursa,

$$f^p(X_{1,j}) = SD_1 \Rightarrow f^{t-p-1}(\dot{I}D_1) = X_{1,j-1} \tag{3.4}$$

Böylece sadece $\dot{I}D_1$ ve SD_1 değerleri bilinerek t işlem ile $X_{1,j}$ görüntüsüne sahip değer hesaplanabilmektedir. Tabloda $mt = 2^n$ tane farklı değerin depolandığı varsayılırsa, $X_{i,j}$ görüntüsüne sahip değer bulunması için aşağıdaki sorgunun sağlanması gerekmektedir.

$$f^l(X_{i,j}) \in \{SD_1, SD_2, \ldots, SD_{m-1}, SD_m\} \qquad l = 0,1,2,\ldots,t-1$$

(3.5)

Yukarıdaki koşulu sağlayan bir l değeri bulunduğunda, ilgili $\dot{I}D$ değeri kullanılarak $X_{i,j}$ görüntüsüne sahip değer bulunabilmektedir. Burada her $f^l(X_{i,j})$ değerini $\{SD_1, SD_2, \ldots, SD_{m-1}, SD_m\}$ kümesinde en az işlemle aramak için bu kümenin sıralanmış olması gerekmektedir [15]. Arama işleminin ilk adımında f fonksiyonu bir defa kullanılırken, ikinci adımda f^2 hesaplanacağı için iki defa kullanılır. Son adımda ise $t-1$ defa f fonksiyonu hesaplanır. Toplamda yaklaşık yapılan işlem sayısı $T = t^2$ olacaktır. Tabloda $mt = 2^n$ nokta bilgisi bulunmasına rağmen kullanılan

hafıza yaklaşık $M = m$ kadardır. Hafıza ve işlem eğrisi $MT = mt^2 = N$ üzerindeki en uygun nokta $m = t = N^{1/3}$ olarak alınabilir. Tablonun oluşturulması bir defaya mahsus yapıldığından arama işlemine dâhil edilmesine gerek yoktur.

Tablonun içerdiği mt elemanın birbirinden farklı olması durumunda başarının sağlanma olasılığı $P(S) = mt / N$ olur. Tabloda, $X_{i,j} = X_{k,l}$, $i \neq k$ olmak üzere iki noktanın aynı olması tablonun yapısı gereği daha sonraki noktalarından aynı sırada devam etmesi anlamına gelmektedir. Tablo f fonksiyonunun ardı ardına kullanılması ile oluşacağından $X_{i,j} = X_{k,l} \Rightarrow X_{i,j+1} = X_{k,l+1}, X_{i,j+2} = X_{k,l+2} \cdots$ olacaktır.

Teori 3: f, $\{1,2,\ldots,N\}$ kümesinden aynı kümeye rastgele, bire bir ve örten bir fonksiyon olsun. Herhangi bir $Y = f(X)$ değerini veren X'i bulma olasılığı aşağıdaki eşitlikle gösterilebilir.

$$P(S) \geq (1/N) \sum_{i=1}^{m} \sum_{j=0}^{t-1} \left[(N - it) / N \right]^{j+1} \tag{3.6}$$

İspat: Yukarıda bahsedildiği gibi tablodaki mt elemanın birbirinden farklı olması durumunda başarının sağlanma olasılığı $P(S) = mt / N$ olur. Aslında bu olasılık tablodaki birbirinden farklı eleman sayısının N'ye oranına eşittir. Teorideki f fonksiyonunun görüntülerinden oluşan tablo rastgele değerler içerecek ve bu değerler belli bir ihtimalle birbirinden farklı olacaktır. Her satırında t elemanın olduğu bir tabloda ilk satırdaki ilk iki elemanın birbirinden farklı olma olasılığı

$$P(X_{0,0} \neq X_{0,1}) = 1 \cdot \frac{N-1}{N} \tag{3.7}$$

olur. Üç elemanın birbirinden farklı olma olasılığı ise

$$P(X_{0,0} \neq X_{0,1} \neq X_{0,2}) = 1 \cdot \frac{N-1}{N} + 1 \cdot \frac{N-1}{N} \cdot \frac{N-2}{N}$$

(3.8)

olacaktır. Satırdaki t eleman için yukarıdaki gibi olasılık hesabı yazılabilir ve sonuç olarak her eleman $(N-t)/N$'den büyüktür. Aynı şekilde diğer satılar içinde buna benzer olasılık hesapları yapıldığında $i.$ satır için her elemanın $(N-it)/N$'den küçük olduğu görülür. Bu durumda tablodaki mt elemanın birbirinden farklı olması olasılığı $\sum_{i=1}^{m}\sum_{j=0}^{t-1}\left[(N-it)/N\right]^{j+1}$'den büyüktür. Başarı olasılığı, birbirinden farklı eleman sayısının N'ye oranına eşit olacağından bu olasılık

$$\frac{1}{N} \cdot \sum_{i=1}^{m}\sum_{j=0}^{t-1}\left[(N-it)/N\right]^{j+1} \tag{3.9}$$

'den büyük olacaktır.

Yukarıda bahsedilen en iyi uygun noktanın $m = t = N^{1/3}$ olması için her tabloda $P(S) \approx N^{-1/3}$ olmalıdır. Bu durumda farklı fonksiyonlar kullanan birden fazla tablo kullanılması bu olasılığı artıracaktır. Sonuç olarak her biri $m = N^{1/3}$ eleman barındıran $N^{1/3}$ tane tablo kullanılırsa toplamda $M = N^{2/3}$ hafıza ve $T = N^{2/3}$ işlem ile başarı oranı oldukça yükseltilebilir.

Bu işlemler yapılırken bazı değerler doğru gibi davranabilirler. Bu değerlere yanlış alarm denir. Kurulum yukarıdaki gibi yapıldığında yanlış alarmların çok fazla olmayacağını göstermek başarı oranının yükseleceğini ispatlar.

Teori 4: Her tabloda beklenilen yanlış alarm sayısı,

$$E(F) \leq mt(t+1)/2N \tag{3.10}$$

ile sınırlıdır.

İspat: Bir tabloda yanlış alarmlar her satırdaki bitiş noktalarıyla tablonun içinde de karşılaşılması sebebiyle oluşur. F_{ij}, $Y_j = SD_i$ olmasıyla oluşan yanlış alarmları gösterirse olasılık

$$E(F) \le \sum_{i=1}^{m} \sum_{j=1}^{t} \Pr(F_{ij}) \tag{3.11}$$

olarak gösterilebilir. F_{ij}, $f(K)$'nın i inci satırdaki bir değerle birleşmesinden dolayı kolon sayısı kadar yolla gerçekleşebilir. Örneğin, $f(K) = f^{i-j+1}(\dot{I}D_i)$ ise ya da bir adım sonra birleşiyorsa yani $f(K)$ tablonun i inci satırında değilse fakat $f^2(K) = f^{i-j+2}(\dot{I}D_i)$ ise, $f(K)$'nın i inci satırdaki bir değerle birleşmiştir. Her j farklı yolun gerçekleme ihtimali $K, f(K)$ değerlerinin rastgele $\{1, 2, \cdots, N\}$ kümesinden seçilmiş olmasından dolayı en fazla $1/N$'dir. Bu durumda

$$E(F) \le \sum_{i=1}^{m} \sum_{j=1}^{t} j / N = mt(t+1) / 2N \tag{3.12}$$

olur.

3.2. Rainbow Tablosu Yöntemi

Deneme yanılma tabanlı yapılan kriptanalitik aramalar her ihtimali deneyerek yapıldığından dolayı çok fazla zaman ve enerji harcar. Her değeri hafızada tutarak bu zaman anlık hale getirilebilir fakat bu durumda da çok fazla hafıza ihtiyacı doğar. Önceki bölümde Hellman'ın bu probleme yaklaşımı anlatılmıştır. Tek bir Hellman tablosu doğum günü paradoksu nedeniyle tüm uzayı içeremez. Bu yüzden Hellman her tablonun kendine özel öteleme fonksiyonu olacak şekilde birden fazla tablo kullanımı ile uygulanır. Bu şekilde yukarıda anlatılan yanlış alarm sayısı azaltılmış olur. Rainbow tablosu yönteminde bir tablonun her sütununda farklı bir fonksiyon kullanılarak tablo içindeki çakışma ve birleşme ihtimali azaltılır [16]. Böylece iki farklı satırda çakışma olsa bile tablonun o noktalardan sonrasındaki değerler farklı olacaktır.

Rainbow tablosunda t sütun olduğu düşünülürse, öteleme fonksiyonları 1inci fonksiyon ile başlar ve $t-1$ inci fonksiyon ile sona erer. Böylece iki farklı zincir çakışsa bile çakışma noktası farklı satırın aynı sütununda olmadıktan sonra birleşmezler. Böylece t uzunluktaki zincirlerde çakışma olursa bunların birleşme ihtimali $1/t$ olur. Bundan dolayı $m \times t$ boyunda bir tablodaki başarı oranı aşağıdaki gibi gösterilebilir:

$$P_{tablo} = 1 - \prod_{i=1}^{t} (1 - \frac{m_i}{N})$$

(3.13)

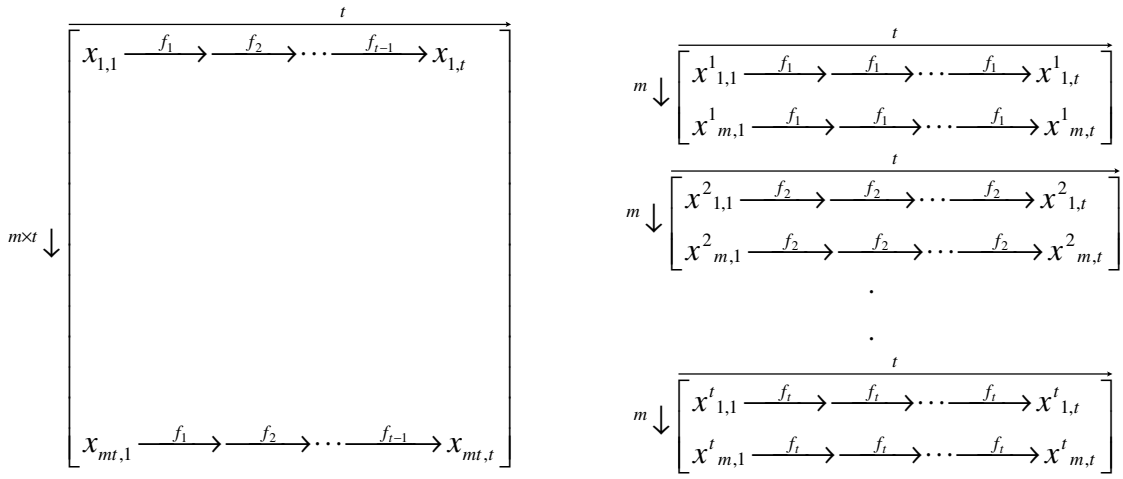
Burada $m_1 = m$ ve $m_{n+1} = N(1 - e^{-\frac{m_n}{N}})$ 'dır.

Her sütundaki elemanlar birbirinden farklı olursa başarı olasılığı $P_{tablo} \approx 1$ olacaktır. Yukarıdaki olasılık hesabı da her sütundaki elemanların birbirinden farklı olması ihtimali hesaplanarak yapılabilir. Birinci sütunda $m_1 = m$ tane farklı eleman olduğu düşünülürse bir sonraki sütun N elemanlı kümede rastgele seçilen değerlerden oluşacağından bir sonraki sütunda m_2 tane farklı eleman

$$m_2 = N(1 - \left(1 - \frac{1}{N}\right)^{m_1}) \approx N(1 - e^{-\frac{m_1}{N}}) \qquad (3.14)$$

olacaktır.

Rainbow ve Hellman yöntemlerinde başarı oranları yaklaşık aynıdır fakat bu yöntem ile hesaplama klasik Hellman tablosunun aksine tek bir tabloda yapılabilir. Klasik yöntemde t farklı $m \times t$ büyüklüğündeki tablolar toplamda mt^2 farklı nokta içerirken bu yöntemde $mt \times t$ büyüklüğünde tek bir tablo ile mt^2 farklı nokta ifade edilebilir. Tablolar arasındaki benzerlik aşağıdaki şekilde gösterilmiştir.



Şekil 3.2 : Solda $mt \times t$ büyüklüğünde Rainbow ve sağda $m \times t$ büyüklüğünde t tane klasik tablo yer almaktadır.

Rainbow tablosunda tarama işlemi Hellman tablosundan biraz farklı şekilde yapılır. Verilen bir x'in tanımını bulmak için bu değere f_{t-1} fonksiyonu uygulanır ve tablodaki bitiş değerleri arasında bu değerin varlığı kontrol edilir. Değer bulunursa görüntüsü verilen değer olan tanım değeri Hellman yöntemindeki gibi ilk değerden başlayıp f_{t-2}'ye kadar fonksiyonlar uygulandığında çıkacak değer olarak hesaplanır. Bitiş değerleri içinde $f_{t-1}(x)$ yok ise, $f_{t-1}(f_{t-2}(x))$'in bitiş değerleri içinde varlığı kontrol edilir ve bu şekilde bitiş değerlerinde x'in fonksiyonlardan geçmiş hali bulunana kadar tarama yapılır. Burada her adımda fonksiyonlar tekrar kullanıldığı için toplam yapılan iş $\frac{t(t-1)}{2}$ olacaktır. Bu t tane klasik tablo kullanarak arama yapmaya göre yaklaşık iki kat daha az işlem anlamına gelir.

Rainbow tablolarında çakışan noktalar aynı zamanda birleşiyorsa bu satırların bitiş noktaları da aynı olmalıdır. Bu şekilde tablodaki birleşmeler kolayca görülebilir. Tablolar bitiş değerlerine göre sıralanırsa birleşen zincirler elenebilir. N elemanlı uzayın her elemanını içeren ve aynı zamanda birleşme noktaları olmayan tablolara kusursuz tablo denir. Bu tabloları oluşturmak için ilk hedef t uzunluklu birleşme noktası olmayan kaç tane zincir elde edilebilir sorusuna cevap bulmaktır. Rainbow tabloları için soru kusursuz olmayan zincirler için yukarıda gösterdiğimiz başarı oranı hesabına benzer bir yöntemle cevaplanabilir. Başarı oranı hesabında her

sütunda farklı eleman olma olasılığına bakılmıştı. Burada da farklı zincir sayısını bulmak için son sütundaki farklı eleman sayısına bakılması gerekir. Bu durumda,

$$P_{tablo} = 1 - e^{-t\frac{m_t}{N}} \tag{3.15}$$

burada $m_1 = N$ ve $m_{n+1} = N(1 - e^{-\frac{m_n}{N}})$ 'dır.

Tablonun son sütununda ki farklı eleman sayısı yani birleşmeyen zincir sayısı sınırlı olduğundan kusursuz tablo elde etmek zor bir işlemdir. Kusursuz tablo inşa etmek için harcanacak enerji yerine kusursuz tablolara yakın tablolar kullanmak başarı oranında çok büyük değişikliğe yol açmaz.

4. SIKIŞTIRMA FONKSİYONLARININ GÖRÜNTÜLERİNİ KULLANARAK ÖZET FONKSİYONLARININ GİRİŞ DEĞERLERİNİN ELDE EDİLMESİ

Özet fonksiyonları başlıca dijital imzalarda, veri bütünlüğünde, asıllama protokollerinde, MAC algoritmalarında ve RNG'lerde olmak üzere çeşitli yerlerde kullanılan kriptografik yapılardır. Temel özelliklerinden en ilginç olanı farklı boydaki fonksiyon girdilerini n bitlik sınırlı bir kümeye götürmeleridir. Birçok özet fonksiyonları girdisinin bir parçasını ve belirli boydaki bir önceki zincir değerini kullanarak sıralı bir şekilde ilerleyen yapılardır. Bu bölümde bahsedilecek yöntem [17], MD yapısına sahip bir özet fonksiyonunun zincir değerlerini kullanarak giriş değerini bulmayı amaçlamaktadır. Çalışmada içinde geçerli özet değerleri içeren Rainbow tabloları [5,6] kullanılmaktadır. Tablolardaki her özet değeri MD güçlendirmesi kullanılan sıkıştırma fonksiyonları ile diğerlerine bağlıdır. Her noktadaki değerler geçerli bir özet değeri olacağından tabloda görülen her değer için kolayca giriş verisi bulunabilir.

Hellman ve Rainbow gibi tablo ile yapılan taramalarda olduğu gibi tablo hazırlamak için gerekli işlem yükü deneme yanılma yapmak kadar fazladır. Tablo hazır olduğunda ise herhangi bir özet değeri verildiğinde n bit zincir ve özet uzunlupuna sahip bir özet fonksiyonu için giriş verisi $2^{2n/3}$ hafıza gereksinimi ile $2^{2n/3}$ adımda hesaplanabilir.

Alt bölümlerde bu yöntemin bazı uzantıları gösterilecektir. Örneğin, bitiş fonksiyonu kullanan özet fonksiyonlarına karşı, mesaj bloğunun uzunluğunu da girdi olarak alan sıkıştırma fonksiyonlarına veya rastgele tuzlama [18] değeri kullanımına karşı bu yöntemin davranışları anlatılacaktır. Ayrıca ileriki bölümlerde yakın ilk değer kavramı sunulacak ve bu değerin bulunması üzerine çıkarımlar yapılacaktır. Yakın ilk değer dayanıklılığı özet fonksiyonunun ilk değere karşı dayanıklılığından daha azdır. Bundan dolayı Rainbow tablolarıyla yakın ilk değere karşı yapılan saldırılar pratik tarama işlemleri olabilir.

Çalışmayı doğrulamak için deneysel sonuçlar elde edilmiştir ve bölümün son kısmında bu çalışmalar ve başarı oranlarından bahsedilecektir. Deneysel çalışma yapmak için MD5 fonksiyonunda değişiklikler yaparak 32 ve 40 bitlik özet fonksiyonları elde edilmiş ve bu fonksiyonlara ilk değer bulmaya yönelik bahsi geçen saldırı yapılmıştır. Saldırı için fonksiyon değerlerini kullanarak iki tablo hazırlanmıştır. Standart bir bilgisayarda 40 bit bir fonksiyonunun ilk değerini bulmak yaklaşık bir hafta sürerken bu tablolar kullanılarak 40 bit fonksiyonda bir dakika içinde ilk değer bulunabilmiştir.

4.1. Temel Saldırı

H, MD yapısında oluşturulmuş bir özet fonksiyonu ve CF bu fonksiyonun sıkıştırma fonksiyonu olsun. Bir M mesajı $M = M^1 \| M^2 \| \cdots \| M^l$ olacak şekilde b bitlik bloklara ayrılsın. Burada M^i mesajın i inci bloğunu, M^l mesajın tamamlama bloğunu ve $\|$ birleştirme işlemini göstersin. Mesaj tamamlama MD güçlendirme ile yapıldığı varsayılırsa: mesajın sonuna '1' biti ve sonrasında yeteri kadar '0' biti ve son olarak mesaj uzunluğu ikili sistemde yazılarak eklensin. Her blok ilgili zincir değeri ile CF fonksiyonunun girdilerini oluşturarak yeni zincir değeri oluşturur. Bu işlem h_0 ilk zincir değeri olmak üzere aşağıdaki gibi gösterilebilir

$$h_i = CF(h_{i-1}, M^i) \qquad i = 1, \cdots, l \tag{4.1}$$

Özet değeri ise $H(M) = h_l = h$ olmak üzere son zincir değeridir.

Burada H fonksiyonu üzerine bir Rainbow tablosu ile bir saldırı düzenlenecek ve sütunların fonksiyonları özel bir tipte olacaktır: i inci sütun i inci tamamlama bloğu olacak ve böylece sütun fonksiyonları zincir değerlerinin fonksiyonları olacaktır.

Tek blok uzunluğunda ve her biri birbirinden farklı $S_1, S_2, \cdots, S_m$ mesajları alınsın. Zincir değeri $1 \le j \le m$ için $h_j^0 = CF(h_0, S_j)$ alınsın. P_i, uzunluğu i blok ve her blokta b bit olan i inci tamamlama mesajı olarak tanımlansın. Bu durumda

$P_i = 1\|00\cdots0\|ib$ olur. Son olarak j inci satırın i inci zincir değeri aşağıdaki gibi tanımlanabilir.

$$h_i^j = CF(h_{i-1}^j, P_i) \tag{4.2}$$

Rainbow tablosu h_i^j zincir değerlerinden oluşacaktır. Tablonun ilk sütunu $h_1^1, \cdots, h_1^m$ ve son sütun $h_t^1, \cdots, h_t^m$ şeklinde olacaktır. Bu arada her h_i^j değeri aynı zamanda gerçek bir özet değeridir. Her h_i^j için ilk değer aşağıdaki gibi gösterilebilir.

Çıkarım 1: h_i^j zincir değerlerini hepsi H özet fonksiyonu tarafından oluşturulmuş gerçek birer özet değeridir. $2 \le i \le t$ ve $1 \le j \le m$ için $H(S_j \|P_1\|P_2\|\cdots\|P_{i-1}) = h_i^j$ ve $i = 1$ durumunda $H(S_j) = h_1^j$'dir.

İspat : Bu ispat $i = 1, \cdots, t$ için i üzerinde tüme varım yöntemiyle gösterilebilir. $i = 1$ durumunda

$$H(S_j) = CF(CF(h_0, S_j), P_1) = CF(h_0^j, P_1) = h_1^j \tag{4.3}$$

olur. Bu durumda h_1^j geçerli bir özet değeridir ve S_j değeri h_1^j , $j = 1, \cdots, m$ için bir ilk değerdir.

Yukarıdaki çıkarımın $1 \le k \le t$ gibi bir k değeri için doğru olduğu varsayılsın. Bu durumda
$j = 1, \cdots, m$ için $H(S_j \|P_1\|P_2\|\cdots\|P_{k-1}) = h_k^j$ olur.

Böylece, h_k^j, $S_j \|P_1\|P_2\|\cdots\|P_k$ değerinin son zincir değeridir ve P_k, $S_j \|P_1\|P_2\|\cdots\|P_{k-1}$ mesajının tamamlama bloğudur. Son olarak

$$H(S_j \|P_1\|P_2\|\cdots\|P_k) = CF(h_k^j, P_{k+1}) \tag{4.4}$$

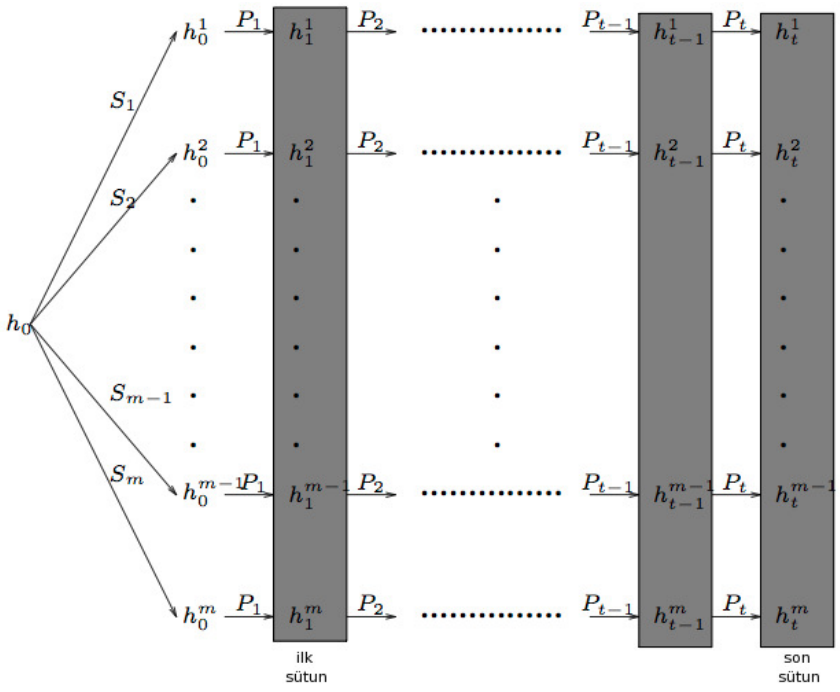
olur.

Diğer taraftan, $CF(h_k^j, P_{k+1}) = h_{k+1}^j$ olacaktır. Böylece, h_{k+1}^j, $S_j \| P_1 \| P_2 \| \cdots \| P_k$ mesajının özet değeri olur. Sonuç olarak çıkarım $k+1$ değeri içinde doğrudur.

Tablonun (i, j)'inci verisi h_i^j değeridir. Rainbow tablosu Şekil 4.1'de gösterilmiştir. i inci sütun fonksiyonu aşağıdaki gibi gösterilebilir.

$$f_i(h) = CF(h, P_i) \tag{4.5}$$

Rainbow tablosunu kullanmak için temel fikir i'inci sütun fonksiyonunu inşa etmektir. Bu tablolar genellikle tek yönlü fonksiyonları geri çevirmek için kullanılır fakat bu çalışmada f_i fonksiyonlarının geri çevrilmeleri önemsizdir. Bu fonksiyonların temel özelliği giriş ve çıkış değerlerinin gerçek birer özet değeri olmasıdır. Böylece bir özet değerinin ilk değerini bulmak için tablodaki konumunu bilmek yeterlidir.



Şekil 4.1: Rainbow tablosu. Oklar sonraki zincir değerlerini göstermektedir.

Tablo $m \cdot t = N = 2^n$ olacak ise, bu değeri Şekil 4.1'de m satır ve n sütun içermektedir. Öte yandan kullanılan toplam hafıza $M \approx m$ ve gerekli zaman $T \approx t^2$

olacaktır. Böylece $M^2 \cdot T = N^2$ eğrisi elde edilir. Bu eğri üzerindeki en iyi nokta $M = T = 2^{2n/3}$ olacaktır.

Klasik Rainbow tablolarının aksine bu tabloda ilk sütunun saklanmasına gerek yoktur. Sadece son sütun değerlerine göre sıralı şekilde son sütun $h_t^1, \cdots, h_t^m$ değerlerini ve S_j mesajlarını hafızada saklamak yeterlidir. Verilen bir h özet değeri için, öncelikle bu değerin son sütunda olduğu kontrol edilir. Değer bulunamıyorsa $CF(h, P_t)$ değerinin son sütunda varlığı kontrol edilir. Değer bulunamıyorsa $CF(CF(h, P_{t-1}), P_t)$ değerinin son sütunda varlığı kontrol edilir ve değerlerden bir tanesi bulunana kadar bu şekilde devam edilir. Değer tabloda bulunduğu zaman ilk değeri bulmak için h özet değerinin konumunu belirlemek yeterli olacaktır.

Bazı i ve j değerleri için $h = h_i^j$ ise bu değeri bulmak için sıkıştırma fonksiyonunu $(t-i)(t-i-1)/2$ defa çalıştırmak yeterlidir. İlk değer aşağıdaki gibi bulunabilir.

$$H(S_j \| P_1 \| P_2 \| \cdots \| P_{i-1}) = h$$

(4.6)

Bu durumda en kötü ihtimalde yaklaşık $t^2/2$ defa CF fonksiyonu çalıştırılarak h özet değerinin tabloda varlığı kontrol edilebilir. Böylece ilk değerin bulunması için ortalama masraf yaklaşık $2^{2n/3}$ işlem ve $2^{2n/3}$ hafıza kullanımı olacaktır.

4.2. Temel Saldırının Genişletilmesi ve Güvenlik İncelemeleri

Önceki bölümde verilen temel saldırının bazı çeşitleri MD yapılarındaki gelişmiş fonksiyonlara da uygulanabilir. Bu gelişmiş çeşitleri üç gruba ayırabiliriz:

- Son zincir değerine farklı bir bitirme fonksiyonu uygulayarak özet değeri üreten fonksiyonlar.

- Tuzlama değeri kullanarak rastgele özet değeri üreten fonksiyonlar. Bu çeşitlerde sıkıştırma fonksiyonları giriş değerine ek olarak rastgele bir değer alırlar.

- Sıkıştırma fonksiyonunda mesaj bloğunun uzunluğunu bit veya bayt olarak ek bir girdi şeklinde kullanan fonksiyonlar.

Bu çeşitlere uygulanabilecek temel saldırılar aşağıdaki bölümlerde anlatılacaktır.

4.2.1. Çıkış fonksiyonlu MD yapıları

Temel saldırı bölümünün başında anlatıldığı gibi sıkıştırma fonksiyonu CF olan MD yapısına sahip bir H özet fonksiyonu kabul edilsin. Bu fonksiyondan aşağıdaki gibi bir H' özet fonksiyonu türetilsin

$$H'(M) = G(H(M)) \tag{4.7}$$

ve

$$G : GF(2)^n \rightarrow GF(2)^{n'}. \tag{4.8}$$

Bu tanıma göre H' özet fonksiyonu G çıkış fonksiyonuna ve n' uzunluğuna sahiptir. Temel saldırıyı uygulamak için H' özet fonksiyonuyla üretilmiş bir özet h' değerinin verildiği varsayılsın. İlk değeri bulmak için öncelikle $G^{-1}(h')$ değerine bakılmalıdır. Bu değer için $G(h) = h'$ olduğu varsayılabilir. Bu durumda h değeri H özet fonksiyonu için hazırlanmış olan tabloda bulunabiliyorsa, bu yöntemle bulunan H fonksiyonuna göre h ilk değeri aynı zamanda H' özet fonksiyonuna göre h' özetinin ilk değeridir. Bu H' özet fonksiyonu için ilk değer bulmaya yönelik basit bir saldırı tipi olarak düşünülebilir.

Saldırı, bir kümeye ait herhangi bir elemanı tabloda arayarak ilerletilebilir. Çıkış fonksiyonu G olan h' özetinin ilk değerini bulmak için gerekli iş yükü C olsun. Eleman sayısı 2^k olan ve G fonksiyonu ile üretilen h' değerlerini içeren bir $P_{h'}$ kümesi alınsın. Bu kümedeki elemanların ilk değerlerini bulmak için gerekli iş yükü $2^k C$ olacaktır. Öte yandan $P_{h'}$ kümesi içindeki herhangi bir elemanın H özet fonksiyonu için hazırlanmış olan tabloda bulunması aynı zamanda H' özet fonksiyonuna göre h' özetinin ilk değerinin bulunması anlamına gelir. Böylece

eleman sayısı 2^{n-k} olan bir tablo kullanılması yeterli olacaktır. Bu tablo m satır ve t sütundan oluşacak ve bunların çarpımı $m \times t = 2^{n-k}$ olacaktır. Bu durumda doğum günü paradoksuna göre $P_{h'}$ kümesinin herhangi bir elemanı belli bir ihtimalle (bu ihtimal $1-\exp(-1) \approx 0.63$ olur) bu tablonun içinde olacaktır.

Gerekli hafızanın $M \approx m$ olduğu bu tabloda gerekli zaman $T \approx t^2 2^k$ olacaktır. Böylece $m \times t = 2^{n-k}$ olması bize $M^2 T = 2^{2n-k}$ eğrisini verir. Sonuç olarak, bu eğriye bakarak bir ilk değerin bulunması $2^{(2n-k)/3} + 2^k C$ sürede $2^{(2n-k)/3}$ hafıza kullanarak gerçekleştirilebilir.

4.2.2. Rastgele özet değeri oluşturma

Özet fonksiyonundan oluşturulan yeni bir H' özet fonksiyonu aşağıdaki gibi tanımlanabilir

$$H'(M, r) = H(r \| M^1 \oplus r \| \cdots r \| M^\ell \oplus r) \tag{4.9}$$

Yukarıdaki tanım Halevi ve Krawczyk [19] tarafından önerilen RMX şeklindedir. Burada r değeri rastgele seçilmiş tuzlama değeri olarak bilinir. Rastgele oluşturulan bir özet değeri farklı yollarla da elde edilebilir. Örneğin, rastgele özet fonksiyonu Bihan ve Dunkelman [20] tarafından önerilen HAIFA yapısını kullanarak da oluşturulabilir. Bu durumda, verilen bir özet değerinin ilk değerini bulma problemi r_0 değerini sabitleyip temel saldırıyı uygulayarak çözülebilir.

4.2.3. Sıkıştırma fonksiyonu blok sayısını veya uzunluğunu giriş parametresi olarak kullanan MD yapıları

Blok sayısını veya mesaj uzunluğunu girdi olarak alan CF' fonksiyonu, H' özet fonksiyonunun sıkıştırma fonksiyonu olsun. Bu durumda CF' blok sayısını veya mesaj uzunluğunu özet değerini hesaplamak için üçüncü bir parametre olarak kabul etmektedir.

Temel saldırı, bu tipteki H' özet fonksiyonunun son sıkıştırma işlemini yapan CF' fonksiyonu önceki adımlardaki ile aynı şekilde çalışıyorsa uygulanabilir. Kısaca son bloğa (tamamlama bloğuna) uygulanan CF' daha önceki bloklara uygulandığı gibi çalışıyorsa temel saldırı gerçekleştirilebilir. Bu varsayımla, bir blok uzunluğunda birbirinden farklı $S_1,...,S_m$ mesajları alınsın. Özet değeri $1 \le j \le m$ için $h_0^j = CF'(h_0, S_j, 1)$ şeklinde hesaplanabilir. Burada sıkıştırma fonksiyonu blok sayısını aldığı varsayıldığı için 1 kullanılmıştır. Mesaj uzunluğunun kullanıldığı durumda $h_0^j = CF'(h_0, S_j, b)$ olmalıdır. Mesaj blok sayısı i ve her blok uzunluğunun b bit olduğu yapıda P_i tamamlama bloğu $P_i = 1\|00\cdots\|ib$ şeklinde oluşturulabilir. Bu durumda tablodaki j inci satırın i inci zincir değeri $h_i^j = CF'(h_{i-1}^j, P_i, i+1)$ (mesaj uzunluğu kullanıldığı durumda ise $h_i^j = CF'(h_{i-1}^j, P_i, b(i+1))$ olacaktır. Bu durumda temel saldırı söz konusu özet fonksiyonu için gerçeklenebilir olmaktadır.

5. YENİ ÖNERİ: YAKIN İLK DEĞER

Verilen bir özet değerini veren ilk değerin bulunması gibi, bu değere yaklaşık bir değer veren bir ilk değer bulmak da zor bir problemdir. Verilen bir h özet değerine karşılık bir $d(H(M),h) \leq \in$ olacak şekilde bir M mesajı bulmak için gerekli iş gücü en az 2^{n-e} lur. Burada kullanılan $d()$ fonksiyonu Hamming uzaklığını ve 2^e değeri ise h özetine $\in$-komşuluğundaki değerlerin bulunduğu kümenin boyutunu göstermektedir. Bu tanım yakın çakışma durumu için de geçerli bir tanımdır ve burada yakın ilk değer için kullanılmaktadır.

Rainbow tablosu ile yakın ilk değer bulma yöntemi temel saldırı ile çok benzerdir. Tabloda h özetine $\in$-komşuluğunda bir h' değeri varsa tablo kullanılarak h' özet değerinin ilk değeri bulunabilir demektir. Doğum günü paradoksuna göre h özetinin $\in$-komşuluğundaki elemanlarla boş olmayan bir kesişim kümesi elde etmek için tabloda $m \times t = 2^{n-e}$ eleman bulunmalıdır. Tablo bu komşuluktaki her eleman için taranır. Bu durumda zaman karmaşıklığı $T \approx t^2 2^e$ olacaktır. Böylece $M^2 T = 2^{2n-e}$ eğrisi elde edilir. Eğrinin en uygun noktası $M = T = 2^{(2n-e)/3}$ kullanılmalıdır.

Değerlerde kullanılan alfabenin ikili sistemden olması durumunda $\in$-komşuluğundaki eleman sayısı aşağıdaki gibi gösterilebilir.

$$e = \log_2 \left(\sum_{i=0}^{e} \binom{n}{i} \right) \tag{5.1}$$

Genel olarak alfabenin z eleman içerdiği düşünülebilir. Aynı zamanda ilk f basamağın aynı olduğu düşünülürse $\in$-komşuluğunda ki eleman sayısı $e = e(z, f, \in)$ şeklinde yazılabilir.

$$e = e(z, f, \in) = \log_2 \left(\sum_{i=0}^{\in} (z-1)^i \binom{n-f}{i} \right) \tag{5.2}$$

Örneğin, 128 bit çıktısı olan bir özet fonksiyonu alındığı varsayılsın. Bu 32 basamaklı 16'lık tabandaki sayıların insan gözüyle kontrol edildiği düşünülsün. Birçok insan ilk dört basamağı aynı olan ve kalan basamaklarda en fazla altı basamağı farklı olan değerleri karşılaştırırken bu değerlerin aynı olduğuna kanaat getirebilir. Bu durumda verilen bir h özet değeri için, birisi sistemin güvenliğini Hamming uzaklığı $d(H(M),h) \leq 6$ ve ilk dört basamağı h özet değeri ile aynı olacak şekilde bir M mesajı bularak kırabilir. Bu mesajla kontrol eden kişinin $H(M)$ ve h değerinin aynı olduğunu düşünmesini sağlayabilir. Bu durumda h değerinin $\in$-komşuluğunda ki eleman sayısı $e = e(16,4,6)$ aşağıdaki gibi hesaplanabilir.

$$e = e(16,4,6) = \log_2 \left(\sum_{i=0}^{6} 15^i \cdot \binom{28}{i} \right) \approx 42 \tag{5.3}$$

Bu durumda tablo 2^{86} eleman içerir ve bu tabloyu tarama maliyeti ve hafıza gereksinimi birbirine eşit ve 2^{71}'dir. Bu örnek insan davranışları ve kriptografiyi karşılaştırmak için güzel bir örnek olarak düşünülebilir.

5.1. Güvenlik İhtiyaçları

Önceki bölümlerde temel ilk değer saldırısının değişik MD yapılarına nasıl uygulanabileceği gösterildi. Saldırının uygulanabilir olması bu yöntemin deneme yanılma yönteminden daha başarılı bir yol olduğunu göstermektedir.

Özet değerini sıkıştırma fonksiyonundan farklı bir fonksiyonla n bit uzunluktan n' bit uzunluğuna kısaltarak çalışan özet fonksiyonlarında bu yöntemin uygulanma maliyeti $2^{(n+n')/3}$ zaman ve $2^{(n+n')/3}$ hafızadır. Bu durumda $n < 2n'$ olması ilk değer saldırısı yönteminin deneme yanılma yönteminden daha başarılı olması anlamına gelir. Böylece ilk değer saldırısından korunmak için bir yöntem önerilebilir: zincir değerinin uzunluğu özet değerinin en az iki katı kadar olmalıdır. Bu sonuç aslında şaşırtıcı bir sonuçtur. Bu sonuç akan şifreleyicilerin ödünleşim saldırılarına karşı

korunması için yapılan bir öneridir [21,22] ve bu şifreleyicilerin tasarım ölçütü olarak kabul edilir.

6. BAŞARI ORANLARI VE DENEYSEL SONUÇLAR

Temel saldırı için elde edilen teorik sonuçları gösterebilmek için MD yapısında 32 ve 40 bit çıktıya sahip iki farklı özet fonksiyonu tasarlanmıştır. Her iki tasarımda da mesaj bloklarının uzunluğu 128 bit olarak belirlenmiştir.

Bir tablonun başarı oranı (herhangi bir değerin bu tabloda bulunması ihtimali) aşağıdaki gibi gösterilebilir.

$$P_s(m,t) = 1 - \prod_{i=1}^{t} \left(1 - \frac{m_i}{2^n}\right) \tag{6.1}$$

Burada m_i değeri i sütunundaki birbirinden farklı elemanların sayısını göstermektedir. Bu olasılık tüm $i = 1,...,t$ değerleri için $m_i = m$ alındığında $1-(1-m/2^n)^t$ tarafından yukarıdan sınırlıdır. Bu koşullardaki tablo kusursuz tablo olarak bilinir [6]. Diğer yandan,

$$1 - \left(1 - \frac{m}{2^n}\right)^t \approx 1 - \exp\left(-\frac{mt}{2^n}\right) \text{'dir.} \tag{6.2}$$

ve bu değer $mt \approx 2^n$ olduğunda yaklaşık $1-\exp(-1) \approx 0.63$ 'dür. Deneylerde başarı oranları 32 bit fonksiyon için 15698 denemede 0.556 ve 40 bit fonksiyon için 1414 denemede 0.565 olarak görülmüştür.

Zaman karmaşıklığı için yaklaşık değer sabitler ve yanlış alarmlardan kaynaklanan fazlalıkların yok sayılması durumunda $T \approx t^2$ olur. Son zamanların çalışmalarında, bir Rainbow tablosu için ortalama zaman karmaşıklığını veren formül aşağıdaki gibi bulunmuştur [21].

$$T_{av} = \frac{m}{2^n}\sum_{k=1}^{t}\left(1-\frac{m}{2^n}\right)^{k-1}\cdot\left(\frac{k(k-1)}{2}+\sum_{i=t-k+1}^{t} i\left(1-\frac{m}{2^n}-\frac{i(i-1)}{t(t-1)}\right)\right) + \left(1-\frac{m}{2^n}\right)^{t}\cdot\left(\frac{t(t-1)}{2}+\sum_{i=1}^{t} i\left(1-\frac{m}{2^n}-\frac{i(i-1)}{t(t-1)}\right)\right) \quad (6.3)$$

Yapılan deneylerde iki durum söz konusudur: $(n,m,t)=(32,2^{21},2^{11})$ ve $(n,m,t)=(40,2^{26},2^{14})$. Bu durumlar için teorik zaman karmaşıklığı yukarıdaki formüle göre $2^{19.25}$ ve $2^{25.25}$'dir. Deneysel sonuçlar aşağıdaki tabloda gösterildiği gibi teorik sonuçları doğrulamaktadır.

Tablo 6.1: Deneysel Sonuçlar. BD: Başarılı Denemeler. UD: Başarısız Denemeler. TD: Toplam Denemeler. OZK: Ortalama Zaman Karmaşıklığı. OYAS: Ortalama Yanlış Alarm Sayısı

		BD	**UD**	**TD**
OZK	2^{32}	$2^{18.6}$	$2^{19.9}$	$2^{19.3}$
	2^{40}	$2^{24.8}$	$2^{26.1}$	$2^{25.5}$
OYAS	2^{32}	$2^{8.3}$	2^{10}	$2^{9.3}$
	2^{40}	$2^{11.3}$	2^{13}	$2^{12.3}$

7. SONUÇLAR

Bu tezde Rainbow tabloları kullanılarak bir özet değerinin ilk değerini bulmak için doğal bir yöntem önerilmiştir. Bu yöntemle, güvenlik seviyesi 2^n olarak kabul edilen zincir uzunluğu ve özet uzunluğu n bit olan MD yapısındaki özet fonksiyonlarının güvenlik seviyesinin $2^{2n/3}$'e indirilebileceği gösterilmiştir. Öte yandan, bu yöntemin en önemli sakıncası ön işlemin deneme yanılma yöntemi kadar iş gücü gerektirmesidir fakat bu ön işlem çevrimdışı ve sadece bir defa yapılır.

Ayrıca, bu tezde MD güçlendirmesinin üstesinden gelebilmek için yer tabanlı geri çevirme tekniği kullanılmıştır. Bu yöntem için Rainbow tablosu hazırlanırken sıkıştırma fonksiyonundan çıkan her değerin gerçek bir özet değeri olması sağlanmıştır. Bu yöntemle, tablodaki her özet değeri birbirine sıkıştırma fonksiyonları ile bağlıdır. Bir noktanın pozisyonu ve onun ilk değeri arasındaki bağlantı Çıkarım 1'de verilmiştir. Klasik Rainbow yönteminde sadece pozisyonun bilinmesi fonksiyonun ilk görüntüsünün bulunmasına yardımcı olmaz. Çıkarım 1'deki bu özellik, MD güçlendirmesinin aşılmasını sağlamakla birlikte, bu tezdeki yöntemin ve klasik Rainbow tablosu yöntemi veya Hellman yönteminin arasındaki temel farktır.

Önerilen bu yöntem MD güçlendirmesinin önemli bazı uzantıları için genişletilmiştir. Farklı çıkış fonksiyonu kullanan, sıkıştırma fonksiyonuna mesaj uzunluğunu veya blok sayısını parametre olarak alan ve özet değeri hesaplarken rastgele tuzlama değeri kullanan fonksiyonlar MD güçlendirmesinin önemli bazı uzantılarıdır. Bunun yanında, bu tezde yakın ilk değer kavramı sunulmuş ve bu değeri bulmak için bir saldırı yöntemi önerilmiştir. Ayrıca, bu kavramın kullanılabilir olduğunu sağlayan senaryo örnekleri verilmiştir. Yakın ilk değer için kullanılacak güvenlik limiti ilk değer güvenliğine göre oldukça azdır. Bundan dolayı yakın ilk değer için yapılabilecek saldırı küçük çıktı boyundaki özet fonksiyonları kullanan sistemler için pratik bir tehlike oluşturur.

Verilen yöntemin, küçük çıktı boyları üreten fonksiyonlara karşı pratik olduğunu göstermek için MD5 algoritmasından 32 ve 40 bitlik değerler üreten iki özet fonksiyonu türetilmiştir. Bu özet fonksiyonları için iki Rainbow tablosu oluşturulmuş ve kullanılmıştır.

Baştan kabul etmek gerekir ki büyük özet uzunluklarında Rainbow tablosu yöntemi pratik bir yöntem değildir. Örneğin özet uzunluğu 2^{512} olan bir özet fonksiyonu için tablo hazırlamak imkânsızdır. Öte yandan, bütün özet fonksiyonu uygulamaları büyük özet uzunlukları gerektirmezler. Rainbow tablosu yöntemi bu tarz uygulamalarda kullanılan özet fonksiyonları ve bu fonksiyonları kullanan sistemler için ciddi tehlike oluşturur.

Son zamanlarda, literatürde ve endüstride kısa özet değerleri kullanan birçok kriptografik sistem ortaya çıkmıştır. Bazı elektronik ödeme yöntemleri ve özellikle küçük ödeme sistemleri, örneğin "PayWorld" ve "MicroMint" [22], "Millicent" [23] ve "NetBill" [24], verimlilik amacıyla küçük uzunluktaki özet değerlerini kullanan sistemlerdir. Ayrıca, insan gücüyle yapılan karşılaştırma tabanlı asıllama protokolleri genellikle kısa uzunluklu özet değerleri üreten fonksiyonlar kullanırlar.

RFID etiketi kullanan cihazlar kısa uzunluklu özet fonksiyonları için yaygın bir uygulama alanıdır. Bazı RFID sistemleri asıllama ve gizlilik sağlamak için küçük boylu özet değeri üreten fonksiyonlar kullanır [27, 28]. Bunun yanında, etiket tabanlı birçok uygulamanın güvenlik protokolleri çakışmaya karşı dirence ihtiyaç duymaz [25]. Güvenlikleri sadece tek yönlülük özelliğine dayalıdır. Örneğin, bazı RFID sistemlerinde [26] kullanıcı gizliliği özet zincir mekanizmasıyla korunur ki buda kısmen ilk değer direnci gerektirir. DM-PRESENT-128 [27] bazı protokoller için son zamanlarda tasarlanmış özet fonksiyonlarından biridir. Bu tasarım PRESENT blok şifreleyici tabanlı bir Davies-Meyer yapısıdır. 128-bit blok uzunluğu ve 64-bit özet uzunluğuna sahiptir. Bundan dolayı 2^{43} PRESENT fonksiyonu çalıştırarak ve 2^{47} bayt hafıza kullanılarak bu özet fonksiyonu için ilk değer bulmak mümkündür.

Sunulan bu yöntemi etkisiz kılmak için temel olarak iki yöntem önerilebilir:

i. İlk yöntem zincir uzunluğunu özet değerinin uzunluğunun en az iki katı kadar yapmaktır. Lucks [28] bu argümanı makalesinde desteklemektedir. Aslında bu kıstas verilen bir mesajla aynı özet değerini veren ikinci değeri bulmaya karşı tam güvenlik sağlamak için önemlidir.

ii. Yöntemin çalışmasını önlemek için ikinci yol ise sıkıştırma fonksiyonuna mesaj verilmeden önce rastgele bir tuz değeri almak ve bu değerle birlikte başlangıç zincir değerini belirlemektir. Buna rağmen, rastgele özet fonksiyonu oluşturmak girişi tek parametreli olan protokoller için uygun olmayabilir.

KAYNAKLAR

[1] Diffie W., Hellman M., "New Directions in Cryptograpy", ***IT-22*** , p. 644-654, (1976).

[2] Rivest R.L, Shamir A., Adleman L., "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems", ***Communications of the ACM 21*** , p. 120-126, (1978).

[3] Özen O., "On The Security of Tiger Hash Function", Ankara, ***ODTÜ Matematik***, (2008).

[4] Joux A., "Multicollisions in Iterated Hash Functions", ***Advances in Cryptography***, p. 306-316, (2004).

[5] Hellman M.E., "A cryptanalytic time-memory trade-off", ***IEEE Trans. on Information Theory*** , p. 401-406, (1980).

[6] Oechslin P., "Making a faster cryptanalytic time-memory trade-off", ***Advances in Cryptology***, p. 617-630, (2003).

[7] Preneel B., "Analysis and Design of Cryptographic Hash Functions", ***Phd Thesis*** (1999).

[8] Bertoni G., Daemen J., Peeters M., Van Assche G., "Sponge Functions", ***Third NIST Cryptographic Hash Workshop*** , (2007).

[9] Kelsey J., Schneier B., "Second-Preimages on n-bir Hash Functions for Much Less Than 2^n Work", ***Advances in Cryptography***, p. 474-490, (2005).

[10] Menezes A.J, Vanstone S.A, Van Oorschot P.C., "Handbook of Applied Cryptography" FL, USA, ***CRC Press***, (1996).

[11] Merkle R., "One way hash function and DES", ***Advances in Cryptology***, p. 428-446, (1989).

[12] Damgård I.B., "A design principle for hash functions", ***Advances in Cryptology***, p. 416-427, (1989).

[13] Kelsey J., Kohno T., "Herding Hash Functions and the Nostradamus Attack", ***Advances in Cryptography***, p. 183-200, (2006).

[14] Rivest R.L., "The MD4 message digest algoritm", ***Advances in Cryptology***, p. 303-311, (1990).

[15] Knuth D.E., "Sorting and Searching", ***The art of Computer Programming*** , p. 304 - 309, (1973).

[16] Barkan E., Biham E., Shamir A., "Rigorous Bounds on Cryptographic Time/Memory Tradeoffs", ***Advances in Cryptography***, (2006).

[17] Kara O., Atalay A., "Preimages of Hash Functions Through Rainbow Tables", ***Computer and Information Sciences, 2009. ISCIS 2009. 24th International Symposium*** , p. 304 - 309, (2009).

[18] Wikipedia., Salt, One Usage Of Random Bits in Cryptography, [Online]. http://en.wikipedia.org/wiki/Salt_%28cryptography%29. **(Ziyaret tarihi: 2 Eylül 2011)**

[19] S. Halevi H.K., "Strengthening Digital Signatures Via Randomized Hashing", ***Advances in Cryptology***, p. 41-49, (2006).

[20] E. Biham O.D., "A Framework for Iterative Hash Functions: HAIFA", ***Second NIST Cryptographic Hash Workshop*** , (2006).

[21] G. Avoine P.JaPO., "Characterization and Improvement of Time-Memory Trade-Offs Based on Perfect Tables", ***ACM Transactions on Information and Systems Security*** , 11 (4)(17)p. 17:1-17:22, (2008).

[22] Rivest R.L, Shamir A., Payworld and Micromint: Two simple micropayment schemes, [Online].; 2001. http://people.csail.mit.edu/rivest/RivestShamir-mpay.pdf. **(Ziyaret tarihi: 23 Kasım 2011)**

[23] Manasse M.S., Millicent (electronik microcommerce), [Online].; 1995. http://www.research.digital.com/SRC/personal/Mark Manasse/uncommon/ucom.html. **(Ziyaret tarihi: 22 Kasım 2011)**

[24] NETBILL., The Netbill electronic commerce project, [Online].; 1995. http://www.ini.cmu/NETBILL/home.html. **(Ziyaret tarihi: 20 Kasım 2011)**

[25] Dean R.D., "Formal Aspects of Mobile Code Security", ***Princeston University*** (1999).

[26] Suzuki K., Ohkubo M., Kinoshita S., "Cryptographic approach to "privacy-friendly" tags.", ***RFID Privacy Workshop*** , USA, MIT, (2003).

[27] Bogdanov A., Leander G., Paar C., Poschmann A., Robshaw M.JB, Seurin Y., "Hash Functions and RFID Tags: Mind the Gap.", ***Advances of Cryptology***, p. 283-299, (2008).

[28] Lucks S., "A failure-Friendly Design Principle for Hash Functions", ***Advances in Cryptography***, p. 474-494, (2005).

[29] Golic J., "Cryptanalysis of Alleged A5 Stream Cipher", Springer, p. 239-255, (1997).

[30] Dysli E., Avoigne G., Oechslin P., "Reducing time complexity in RFID systems.", ***Selected Areas in Cryptography***, p. 291-306, (2006).

[31] Babbage S., "Improved Exhaustive Search Attacks on Stream Ciphers", ***IEEE Conference publication*** , p. 161-166, (1995).

ÖZGEÇMİŞ

1983 yılında İstanbul'da doğdu. İlk, orta ve lise öğrenimini İstanbul'da tamamladı. 2001 yılında Ortadoğu Teknik Üniversitesi Matematik Bölümüne başladı ve 2006 yılında Matematikçi olarak mezun oldu. 2008 yılında TÜBİTAK UEKAE Kriptoloji bölümünde çalışmaya ve İstanbul Teknik Üniversitesi Matematik bölümünde yüksek lisans öğrenimine başladı. 2009 yılında yüksek lisans öğrenimine Kocaeli Üniversitesi Matematik bölümünde devam etmeye karar verdi. 2010 yılında yüksek lisans öğrenimine ara verip askerliğini tamamladı. Askerlik görevini tamamladıktan sonra özel bir iş kurdu. 2011 yılından beri özel işi ile beraber öğrenimine kaldığı yerden devam etmektedir.