Fırat Üniv. Müh. Bil. Dergisi
29(1), 59-65, 2017

Science and Eng. J of Fırat Univ.
29(1),59-65, 2017

Mobil Aygıtlar İçin Tek Kullanımlık Şifre Üreteci

***Selman YAKUT[1], Ahmet Bedri ÖZER[1]**
[1]Bilgisayar Mühendisliği Bölümü, Fırat Üniversitesi, Elazığ, Türkiye
*syakut@firat.edu.tr

(Geliş/Received: 26.09.2016; Kabul/Accepted: 10.11.2016)

Özet

Bu çalışma da mobil aygıtlar için özet fonksiyonlarını temel alan Tek Kullanımlık Şifre Üreteci tasarlandı. Bu üreteç elektronik ortamlarda kimlik doğrulama amacıyla kullanılmak üzere Tek Kullanımlık Şifre üretir. Bu şifre bir defaya mahsus olarak kullanılan belirli uzunluktaki bir karakter dizisidir. Özet fonksiyonları rastgele uzunluktaki bir giriş mesajı için o mesaja özgü belirli uzunlukta bir çıktı üretir. Burada özet fonksiyonu olarak Keccak kullanıldı. Bu fonksiyon belirlenen son özet fonksiyonu standardıdır. Bu fonksiyon ve gizli anahtar değeri kullanılarak hash tabanlı mesaj doğrulama kodu yapısı tasarlandı ve bu yapının çıkış değeri üretildi. Bu değerden Tek Kullanımlık Şifre değeri sekiz karakter olarak çıkarıldı. Daha sonra kullanılan yöntem ve üretilen değer incelendi. Tek Kullanımlık Şifre üretiminde kullanılan çıkış değeri NIST 800.22 rastgelelik testleri ile test edildi. Güvenlik ve maliyet parametreleri dikkate alınarak üreteç mobil uygulama olarak gerçekleştirildi.

Anahtar Kelimeler: Özet Fonksiyonları, Keccak, Mobil Tek Kullanımlık Şifre Üreteci.

One Time Password Generator for Mobile Devices

Abstract

In this study One Time Password Generator based to hash functions was designed for mobile devices. This generator produces One Time password value for authentication in an electronic environment. This value which is fixed length strings is used as a one-time. Hash functions is produced a specific length output for a specific input message that is arbitrary length. In this, Keccak was used as the hash function. This function has been selected as the latest standards for hash algorithm. Secret key value and hash function is used to design hash based message authentication code and generate hash value. One-time password value is extract as eight character from this value. Then the method and values produced were examined. One time password that extract from the hash value is tested with NIST 800.22 Randomness tests. Considering safety and cost parameters, the generator was carried out as the mobile applications.

Keywords: Hash Functions, Keccak, Mobile One Time Password Generator.

1. Giriş

Teknolojinin gelişimiyle birlikte internet günlük hayatta daha önemli bir yer tutmaktadır. Bunun sonucu olarak E-devlet uygulamalarından internet bankacılığı kadar günlük hayatın bir parçası olan çok sayıda işlem internet yoluyla çözülmektedir. Bu sebeple bu tarz işlemler için güvenlik oldukça büyük önem kesp etmektedir. Dolayısıyla güvenlik noktasından ele alındığında bunun en kritik başlıklarından biri banka hesapları benzeri kritik kaynaklara yetkisiz kişilerin erişememesidir.

Genel olarak elektronik ortamlarda kritik öneme sahip kaynaklara erişimde güvenlik için şifreler kullanılır. Bu şifreler temel olarak sabit şifreler ve tek kullanımlık şifreler olmak üzere iki kısma ayrılabilir. Sabit şifreler önemli kaynaklara her erişimde değişmezken tek kullanımlık şifreler her erişimde değişmektedir.

Önemli kaynaklara her erişimde aynı şifrenin kullanılması çok sayıda probleme sebep olabilir. Birincisi, bu şifreler güvenli olmayan bir ortamda iletildiğinde ortamın dinlenmesi benzeri değişik yöntemlerle şifreler elde edilebilir. İkincisi ise, yapılan araştırmalar gösteriyor ki çok sayıda kullanıcı şifre olarak 11112222, qwerty, ahmet57, abc123 benzeri tekdüze ve rahatlıkla tahmin edilebilen şifreler seçebilmektedir. Sonuç olarak kullanıcıların bu gibi tercihleri bu şifrelerin elde edilmesinde büyük bir zayıflık olabilmektedir. Bununla birlikte çalınan çerezler(Cookies) veya

internet tarayıcıları kullanıcı adı ve şifreleri hatırlayabilir. Daha da önemlisi bu şifreler Malware, Trojans, Keyloger gibi kötü amaçlı yazılımlar kullanılarak çalınabilir. Dahası ise sabit şifreler yeterli miktarda deneme sayısı ve zaman verildiği takdirde yetkisiz kişilerce de aşılabilir.

Sabit şifrelerin sebep olduğu bu problemler Tek Kullanımlık Şifre (TKŞ) kullanılarak rahatlıkla ortadan kaldırılabilir. TKŞ elektronik ortamlarda kimlik doğrulama amacıyla sadece bir defa kullanılan karakter dizisidir. Kullan at şifreler olarak da bilinirler. Bu şifrelerin uzunluğu kullanılan uygulamaya göre değişiklik gösterebilmekte ve büyük-küçük harf ve rakamlardan oluşabilmektedir.

TKŞ kullanımı ile daha önceki bir erişimde kullanılan şifre elde edilse bile bu şifre yeniden kullanılamaz. Çünkü ilgili kaynağa her erişimde kullanılan şifre güncellenir[1].

TKŞ üretimi oldukça farklı yöntemlerle yapılabilir. Bununla beraber bu yöntemler temelde iki sınıfa ayrılabilir. Bunlar zamansal tabanlı ve matematiksel tabanlı yaklaşımlardır. Zaman tabanlı yaklaşımda TKŞ üretiminde zaman algoritmanın önemli bir parçasıdır. Burada TKŞ kullanıcı ile kimlik doğrulama sunucusu üzerindeki saatin eş zamanlı çalışması söz konusudur [2]. Matematiksel yaklaşımda ise TKŞ kullanıcısı ile kimlik doğrulama sunucusu arasında eş zamanlı çalışan bir sayaç vardır [2]. Bu sayaç değeri yardımıyla hangi oturum için TKŞ değeri üretileceği belirlenir.

En önemli matematiksel yöntemler biri 1981'de Lamport tarafından önerilen TKŞ üretiminde tek yönlü özet fonksiyonlarının kullanımıdır [4]. Özet fonksiyonları rastgele uzunluktaki bir mesajdan o mesaja özgü bir değer üretir. Özet fonksiyonları birçok kriptografik uygulamanın temelini oluşturur. Ayrıca bu fonksiyonlar tek yönlü olmalarından TKŞ üretimi için çok daha önemlidir. Bu fonksiyonlar TKŞ üretiminde kullanılırken bir başlangıç değeri alınır daha sonra üretilen TKŞ değerleri bu değerinden türetilir. HMAC (Özet Tabanlı Mesaj Doğrulama Kodu) özet fonksiyonlarını temel alan önemli TKŞ üretim yöntemidir [5].

Bu çalışmada özet fonksiyonlarını temel alan HMAC yapısı mobil uygulama olarak gerçekleştirilerek TKŞ üreteci geliştirildi. Bu üreteç Android tabanlı mobil aygıtlara yüklenen bir uygulamadır. Burada özet fonksiyonu olarak Keccak özet fonksiyonu kullanıldı. Bu fonksiyon Ekim 2012'de NIST (National Institute of Standard and Technology) tarafından özet fonksiyonları için en son standart belirlendi [6]. Dahası bu fonksiyonun standart olarak belirlenme süreci dört yıl boyunca yüzlerce bilim adamı tarafından yapıldı. Burada aday algoritmaların değerlendirilme kıstasları güvenlik, hız gibi parametrelerdir.

TKŞ üretimi gibi dağıtımı da oldukça farklı yöntemlerle yapılabilir [2]. Mobil aygıtların yaygın kullanılmasıyla bu yöntemler için en kullanışlı olanlardan biri mobil aygıtlar kullanılarak TKŞ üretimi ve dağıtımıdır. Burada kullanılan dağıtım yönteminde uygulama iki parçadan oluşur. Uygulamanın birinci kısmı mobil cihaz üzerine yüklü kullanıcıya yönelik uygulama ikinci kısım ise kimlik doğrulama merkezinin bulunduğu kısımdır. Bu yaklaşımda kullanıcı uygulama tarafından üretilen TKŞ merkezde üretilen TKŞ değeriyle karşılaştırılır ve bu işlemin sonucuna göre erişim reddedilir veya kabul edilir [3].

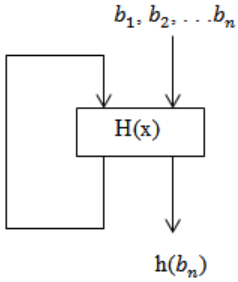
TKŞ üreteçlerinin genel yapıları daha önce belirtildiği gibi olmasına rağmen bu üreteçler uygulamada oldukça farklı ayrıntılara sahip olabilir. Bunun en önemli sebebi TKŞ üreten firmaların çıkarlarını zedeleyebilen ve ticari sır olarak ifade edilebilen ayrıntıları vermekten kaçınmasıdır. Çünkü bu üreteçler bankacılık başta olmak üzere kritik kaynakların kullanımında güvenliğin en önemli parçasını oluşturmaktadır.

TKŞ üreteçlerinin güvenli kabul edilebilmesi için bazı uluslararası testlerden geçmesi gerekmektedir. Bu testlerin başında NIST 800.22 rastgelelik testleri gelmektedir. Bu testler temel olarak üretilen TKŞ'lerin tahmin edilemezliği rastgeleliği gibi parametrelerini göz önüne almaktadır.

2. Özet Fonksiyonları

Özet fonksiyonları herhangi uzunlukta bir giriş mesajı alır ve bu mesajı bazı işlemlerden geçirdikten sonra o mesaja özgü ve o mesajın parmak izi olarak ifade edilen bir çıkış değeri üretir [7]. Şekil 1'de bu fonksiyonlar için belirlenen genel hatlar verildi. Bu yapıda öncelikle giriş mesajı kullanılan algoritmaya

bağlı olarak belli uzunluktaki bloklara ayrılır. Daha sonra Bu blokların özet fonksiyonunca sırasıyla işleme alınır. En son blokun işlenmesi ile mesajın çıkış değeri üretilir ve değer özet değeri olarak ifade edilir.



Şekil 1. Özet fonksiyonlarının genel yapısı.

Özet fonksiyonlar mesaj doğrulama kodu, sayısal imza, rastgele sayı üretimi gibi farklı güvenlik uygulamalarında ve internet protokollerinde kullanılır.

Özet fonksiyonlarının güvenli olarak kabul edilebilmesi için üç temel özellik sağlanmalıdır. Birincisi belli bir mesaj için üretilen özet değerinden mesajın kendisi elde edilmemesi gerekir. Bu özellik bu fonksiyonlarının en önemli fonksiyonudur. İkinci olarak belirli bir mesaj ve bu mesaja ait özet değeri için, aynı özet değerine sahip ikinci bir mesajın hesaplanması mümkün olmamalıdır. Sonuncusu ise rastgele iki mesaj seçilir ve mesajların özet değerinin aynı olup olmadığına bakılır. Burada iki metninde seçimi serbest olmaktadır.

Özet fonksiyonlarını hedef alan saldırılar temel olarak iki sınıfa ayrılır. Bunlardan ilki kaba kuvvet saldırısıdır. Diğeri ise algoritmanın zayıflıklarına dayanan sezgisel saldırılarıdır.

Kaba kuvvet saldırısı olası bütün uzayın taranmasına dayanır. Özet fonksiyonları için bu olası uzay özet değerinin uzunluğudur. Bu sebeple bu fonksiyonların çıkış uzunluğu bu saldırılara karşı güvenli olacak şekilde belirlenmelidir. Bunun en önemli sebebi ise doğum günü çelişkisi benzeri yaklaşımlarla güvenlik seviyesi özet değerinin yarısı kadar olur [7].

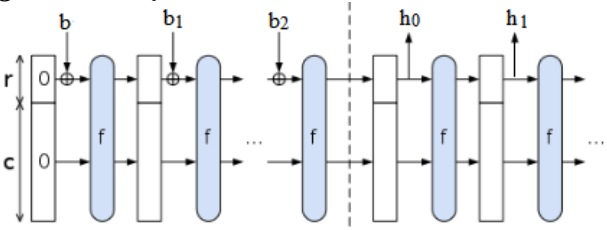
Sezgisel saldırılar özet algoritmasının zayıflıklarından yararlanmaya dayanır. Bu saldırılar başarılı olarak kabul edilmesi için kaba kuvvet saldırılarına oranla daha iyi sonuç vermelidir [8]. Örneğin SHA-1 algoritması için özet değeri 160 bittir. Bu sebeple kaba kuvvet saldırılarına karşı güvenlik seviyesi 2^80'dir. Bununla birlikte sezgisel saldırılar sonucu bu seviye 2^69 kadar inmiştir [8].

3. Keccak Özetleme Algoritması

Keccak özet fonksiyonu 2012 de özet fonksiyonları için yeni standart olarak belirlendi. Bu fonksiyon sünger (Sponge) yapılarına dayanır [9]. Bu yapılar giriş olarak herhangi bir uzunlukta bir giriş bit dizisi alır ve istenilen uzunlukta özet değeri üretebilir [10]. Bu yapıda giriş mesajının blokları öncelikle başlangıç değeriyle XOR işlemine tabi tutulur. Burada ilk blok için başlangıç değeri sıfırdır, diğer bloklar için ise başlangıç değeri bir önceki f fonksiyonu bloğunun çıkışıdır. Burada f Keccak fonksiyonunun çekirdeğidir.

4. Blok Yapısı

Blok yapısında r ve c parametreleri başlangıç vektörünü oluşturur. r her defasında işlenen mesaj blokunu gösterir. c ise güvenlik amacıyla kullanılan bir kıstastır. Mesaj blokları f fonksiyonu tarafından sırasıyla işleme alındıktan sonra özet değeri üretilir. Kecak fonksiyonunun genel hatları şekil 2'de verildi.



Şekil 2. Keccak özet algoritmasının genel yapısı.

Keccak özet fonksiyonu iki temel parçadan oluşur.

1. Giriş mesajının bloklarının alındığı parça
2. Özet değerlerinin üretildiği parça

Keccak fonksiyonunun birinci parçası aşağıdaki işlemlerden oluşur.

1. ilk olarak başlangıç vektörü sıfır alınır Daha sonra giriş mesajı eşit uzunluktaki bloklara ayrılır ve eğer gerekirse son blok için ekleme yapılabilir. Ekleme işlemi eksik bitlerin yerine 10…1 bit dizisi yazılarak yapılır.
2. Birinci bloktan itibaren bütün bloklar sırasıyla giriş vektörünün ilk r biti ile XOR işlemine tabi tutulur. Bu işlemin sonucu giriş vektörünün yeni r bitlik parçası oluşturulur.

3. Giriş vektörü f fonksiyonunca işlenir ve bu işlem sonucunda yeni giriş vektörü olur. Bütün giriş blokları işlendikten sonra fonksiyonun birinci parçası tamamlanır.

Keccak fonksiyonunun ikinci parçası şu şekilde ifade edilebilir.

Birinci parçanın tamamlanmasıyla üretilen giriş vektörünün ilk r biti alınır. Eğer gerekiyorsa giriş vektörü f fonksiyonunca yeniden işleme alınır ve yeni oluşan giriş vektörünün yine r bitlik parçası alınır. Bu işlem istenen uzunlukta özet değeri üretilene kadar devam edebilir.

5. Keccak Çekirdek Fonksiyonu

Çekirdek fonksiyonu özet fonksiyonunun en temel parçasıdır. Bu yapıda kullanılan f çekirdek fonksiyonu da yine sünger yapısının en önemli kısmıdır. Burada kullanılan başlangıç vektörü ve ara giriş vektörlerinin boyutu değişebilir. Bu değişim temel olarak v = 5x5xw yapısıyla ve w = 2^l, l = 1, 2, …, 6 aralığı ile ifade edilebilir. Bu sebeple giriş ve ara vektörlerinin boyutu 25,50,…,1600 aralığında alabilir.

Keccak özet fonksiyonunda vektör boyutuna göre fonksiyonun tur sayısı değişir. Bu değişim 12+2l denklemiyle ifade edilir. Örneğin 1600 bit uzunluklu giriş vektörü ele alındığında l değeri 6 olur ve tur sayısı ise 12+2l formülüyle 24 olur. Güvenlik sebebiyle vektör boyutunun büyük olması istenir.

f çekirdek fonksiyonunda her turda için aşağıda verilen beş temel işlem sırayla gerçekleştirilir.

Θ: 5 bitlik sütunlar için eşitlik (parity) hesaplar ve komşu sütunlar XOR işlemine tabi tutar.

P: üçgen sayısı değerlerine göre 25 kelime değerlerinden her biri için bit düzeyinde döndürme işlemi yapılır.

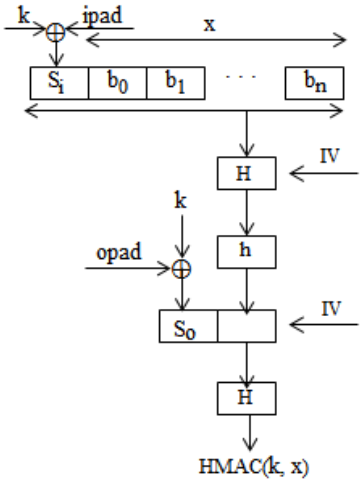
π: 25 kelime sabit bir yapıda yer değiştirme işlemi yapılır.

X: $a = a \oplus (\neg b \ \& \ c)$ denkleminden yararlanılarak bit düzeyinde birleşme işlemi yapılır.

ι: Giriş vektör içerisindeki değerler XOR işlemine tabi tutulur.

6. Özet Tabanlı Mesaj Doğrulama Sistemi

HMAC yapısının genel hatları Şekil 3'te verildi. Burada x giriş mesajı, H özet fonksiyonu (MD5, Keccak vb.), k gizli anahtar, opad hex (0x36) değerinin tekrarı ile oluşturulan byte dizisi, ipad hex (0x5C) değerinin tekrarı ile oluşturulan bayt dizisi, || birleştirme işlemi , ⊕ XOR operatörü, b bir mesaj bloku olarak alındı.



Şekil 3. HMAC algoritmasının yapısı.

HMAC yapıları, bu yapıları oluşturmak için kullanılan özetleme fonksiyonlarına göre daha güvenlidir [8]. HMAC yapılarının gücü iki parametreye bağlıdır. Bunlardan ilki HMAC yapısının temelini oluşturan özet fonksiyonudur. Daha açık bir ifade ile özet fonksiyonu ne nispette güvenli ise HMAC yapısı da o nispette güvenlidir. Diğeri ise HMAC yapısının çıkış değerini üretmek amacıyla kullanılan gizli anahtardır. Burada olarak kullanılan gizli anahtar değerinin güvenli olması HMAC yapısının da güvenli olmasını sağlar.

7. Mobil Uygulama

İnternetin ve teknolojinin gelişimi ve yaygın kullanımı ile günlük hayatın önemli bir parçasını akıllı telefonlar, tablet bilgisayarlar gibi mobil aygıtlar oluşturur. Bu aygıtlar üzerinde çalışması için geliştirilen yazılımlara "Mobil Uygulama" denilir [12]. Bu uygulamalar mobil aygıtların kullandığı "Mobil İşletim Sistemi" ne uygun olarak geliştirilir ve bu işletim sistemine sağlanan uygulamalar bu sisteme ait marketlerinden edinilebilir.

Mobil uygulamalar hedeflenen platforma göre tasarım ve kodlama yapılır. Kodlama o platformun kabul ettiği dil kullanılarak yapılır [13]. Eğer uygulama için yalnızca Android tabanlı aygıtlar hedefleniyorsa kullanılacak cihazın boyutlarına göre tasarlanır ve Java programlama diliyle kodlanır. Eğer bu uygulama Windows veya Apple tabanlı cihazlarda çalışması istenirse cihaza göre tasarım ve platforma uygun programlama dili kullanılır.

Çok sayıda mobil işletim sistemi bulunmakla beraber bunlardan bir kaçı yaygın bir kullanıma sahiptir. Bunlardan biri Google firması tarafından geliştirilen ve birçok mobil aygıt üreticisi, mobil aygıt kullanıcı tarafından kullanılan Android işletim sistemidir. Bu işletim sistemini sahip cihazlar Google Play Store'dan mobil uygulama edinebilir. Mobil uygulamalar bu marketleri yoluyla ile mobil cihazınıza indirilebilir ve kurulabilir. Google Play Store'da uygulama sayısı 2 milyonun üzerinde olduğundan aranan uygulamanın kolayca bulunabilmesi için bu uygulamalar kategorize edilir. Bu kategorilere; haberler, oyunlar, müzik, bankacılık, fotoğrafçılık, spor, sosyal ağlar, eğlence gibi pek çok örnek verilebilir.

Uygulama marketlerindeki uygulamalar ücretli veya ücretsiz olabilir. Ücretsiz uygulamalar bu marketinden mobil cihaza indirilebilir, kurulumu yapılabilir veya kaldırılabilir. Eğer mobil uygulama ücretli ise sisteme tanımlamış olan kredi kartı ile satın alındıktan sonra cihaza yüklenebilir. İnternet bağlantısı üzerinden mesajlaşma ve konuşma imkânı sunan bazı mobil uygulamalarda dönemlik ücretlendirme olabilir [12].

Mobil uygulamalar başlangıçta standart gereksinimler için geliştirilse de genellikle hayatı kolaylaştırmaya yöneliktir. Mobil cihazlar uygulama marketlerinde bulunan oyunlar sayesinde bir eğlence merkezine dönüşebilir. Ayrıca mobil uygulamalar, internetteki bir web sitesine göre çok daha az veri kullanarak aynı bilgiye ulaşmayı sağlar. Çünkü mobil cihazlarında her ne kadar yaygınlaşsa da kotalı internet bağlantısına sahip kullanıcılar çoğunluktadır ve onlar için de daha bu yapı avantaj yaratır. Mobil uygulamalar, bu özellikleri sayesinde insanların hayatında çok önemli bir rol oynama yolunda ilerlemektedir.

8. Materyal ve Metot

TKŞ üreteci mobil uygulama olarak tasarlandı. Bu üreteç Android tabanlı mobil cihazlar için geliştirilen bir uygulamadır. Android Tabanlı sistemlerde yazılımın geliştirme dili Java olduğundan bu uygulamada da Java programlama dili ile kodlama işlemi yapıldı. Böylece geliştirilen yazılımın daha güvenli ve az maliyetli olması amaçlandı.

Geliştirilen TKŞ üreteci iki kısımdan oluşur. Bunlardan ilki kullanıcı tarafında olan yazılım (mobil uygulama) ve diğeri TKŞ üretim merkezindeki (Sistem merkezi) yazılımdır. Kullanıcı tarafında üretilen TKŞ değeri ile banka hesabı gibi kaynaklara erişim talep edilir ve bu talep TKŞ merkezinde üretilen değerle karşılaştırıldıktan sonra sonuca göre kabul edilir veya reddedilir.

TKŞ üretimi için Leslie Lamport tarafından önerilen yaklaşım kullanıldı. Bu yaklaşımla özet fonksiyonları ve bir anahtar değeri kullanılarak TKŞ değerleri üretilir. Bu yöntem HMAC olarak adlandırılır. Dolayısıyla üretilen TKŞ değerlerinin güvenliği kullanılan özet fonksiyonunun güvenliğine ve anahtar değerine bağlıdır. Üretilen şifre değerlerinin güvenliğinin en üst düzeyde olması için en son özet fonksiyonu standardı olan Keccak özet fonksiyonu kullanıldı. Bu yapıda TKŞ değerinin üretimi için tohum değeri olarak kullanıcı ve kimlik doğrulama merkezi arasında paylaşılan gizli bir anahtar değeri kullanıldı. Bu anahtar değerine ve sayıcıya bağlı olarak TKŞ değeri üretildi.

TKŞ üreteci özet fonksiyonuyla birlikte anahtar değeri ve sayıcı kullanılarak tasarlandı. TKŞ üretimi için kullanılan ve HMAC olarak adlandırılan bu yapı bölüm 3. 1. verildi. Algoritmada kullanılan k gizli anahtar değeri sadece kullanıcı ve doğrulayıcı mekanizma arasında paylaşılır. Sayıcı değeri ise her erişimde güncellenen ve kullanıcı ile doğrulayıcı mekanizma arasında eş zamanlı çalışmayı sağlayan bir değerdir. Formül 1'de bu değer c olarak verildi.

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

HMAC algoritması sonucunda üretilen altmış dört karakterlik çıkış değeri kırpma

işlemine tabi tutularak sekiz karaktere indirildi. Bu işlemde HMAC değerinin son karakteri alınır ve bu karakterin sayı değeri hesaplanır. Daha sonra bu değerden başlanarak ilk sekiz karakter alınır. Tablo 1'de geliştirilen sistemin 6 örnek çalışması verilmiştir. İkinci sütunda ise anahtar değerindeki bir bitlik değişime karşılık TKŞ değerinde gözlemlenen değişim verildi.

Tablo 1. Geliştirilen sistemin örnek çıktısı.

Anahtar Değeri	0EBF65AC017…	1EBF65AC017...
1. TKŞ Değeri	22f6a253	f2937c6c
2. TKŞ Değeri	1d297ff2	060941d4
3. TKŞ Değeri	8919e718	a8df8bd2
4. TKŞ Değeri	0956d112	71ec3d4d
5. TKŞ Değeri	52682f1d	1894bf4b
6. TKŞ Değeri	6b0c608d	fedcc7e9

9. Sonuçlar

Bu çalışmada kullanılan yöntem Leslie Lamport tarafından önerilen TKŞ üretim yöntemine dayanır. Bu yöntemde TKŞ üretimi için özet fonksiyonuyla birlikte güvenli anahtar değeri kullanılarak HMAC algoritması gerçekleştirildi. Bu algoritmanın gücü temel olarak kullanılan özet algoritmasının gücüne bağlıdır çünkü bu yapısının temelinde özet fonksiyonları bulunur. Bu amaçla bu fonksiyonların en son standardı olan Keccak kullanıldı. Bu fonksiyonun kodlanması yapıldı ve online çıktılarla karşılaştırılarak doğruluğu tespit edildi. Bu fonksiyonun kullanımının temel amaç güvenli, rastgele ve tahmin edilemeyen TKŞ üretimidir.

Bu çalışmasında kullanılan Keccak özet fonksiyonu Ekim 2012'de NIST tarafından özet fonksiyonları için yeni standart olarak belirlendi. Bu fonksiyon yaklaşık dört yıl boyunca yüzlerce bilim adamı tarafından incelendi ve herhangi bir zayıflığı belirlenemedi. Bu yarışmada aday algoritmaların değerlendirilmesinde dikkate alınan ölçütler temel olarak güvenlik, hız gibi parametrelerdir. Dolaysıyla yapılan yarışmada özet fonksiyonları içinde en güvenli ve en hızlı algoritma olarak belirlendi Bu sebeplerden dolayı bu çalışmada Keccak özet algoritması kullanıldı.

TKŞ değerinin güvenliği NIST'in SP800-22b, testleri ile test edildi. Bu güvenlik testleri rastgelelik ve tahmin edilemezlik gibi parametrelerden oluşur. Dolayısıyla üretilen TKŞ değerlerinin güvenli kabul olması için bu testlerden başarılı olması gerekir. Bu amaçla TKŞ değerlerinin üretildiği HMAC çıkış değerleri bu testlere uygulandı. Bu testlerin uygulanabilmesi için test edilecek veri beli bir uzunlukta olmalıdır. Test verisini bu uzunluğa eriştirmek amacıyla algoritma defalarca çalıştırıldı ve çıkış değerleri kaydedilerek bu değerler üzerinde test işlemi yapıldı. Tekrar işleminde çıkış değeri sonraki seferde giriş olarak alındı. TABLO 1 de görüldüğü gibi test sonuçları başarılıdır. Dolayısıyla bu değerlerden üretilen TKŞ değerleri de güvenli olur. Ayrıca TKŞ değerlerinin rfc6560-otp, rfc4226-hotp standartlarına uygunluğu sağlandı.

Mobil cihazların yaygın kullanımından dolayı TKŞ üreteci uygulama mobil olarak gerçekleştirildi. Mobil cihazlar günlük hayatta oldukça yaygın bir kullanıma sahip olduğundan ayrıca bir maliyet gerektirmez. Uygulamanın mobil cihaza yüklenmesiyle kolaylıkla kullanılabilir. Ayrıca sms iletimi söz konusu olmadığından böyle bir maliyet içermez. TKŞ değerlerinin güvensiz hat üzerinden iletimi söz konusu olmadığından ortadaki Adam saldırısına karşı dayanıklıdır. Güvenlik için ayrıca ihtiyaç duyulursa kullanım şifresi belirlenebilir. Böylece geliştirilen yazılım daha güvenli ve az maliyetli olur.

Tablo 2. Geliştirilen sistem tarafından üretilen sayıların NIST 800.22 testi kullanılarak bulunan test sonuçları

NIST 800.22 Testi	P_{Value}	Sonuç
Frequency Monobit Test	0,547	Başarılı
Frequency Test with a Block	0,036	Başarılı
Runs Test	0,948	Başarılı
Longest Run of Ones in a Block	0,240	Başarılı
Binary Matrix Rank	0,641	Başarılı
Discrete Fourier Transform	0,651	Başarılı
Non Overlapping Template Matching	0,684	Başarılı
Overlapping Template Matching	0,270	Başarılı
Universal Test	0,696	Başarılı
Linear Complexity	0,808	Başarılı
Serial Test	0,567	Başarılı
Approximate Entropy	0,049	Başarılı
Cumulative Sum	0,525	Başarılı

10. Kaynaklar

1. Eldefrawy M. H., Khan M. K., Alghathbar K., Kim T. H., Elkamchouchi H. (2012). Mobile one-time passwords: two-factor authentication using mobile phones, *Securıty and Communıcatıon Networks*, **Vol 5**, p 5508–516.
2. Kua Y., Choia O., Kima K., Shona T., (2013).

Two-factor authentication system based on extended OTP mechanism, International Journal of Computer Mathematics, **Vol 90**, Issue 12.

3. Zhang Q., Chen C., Dai Y., Liao S., (2009). A Unidirectional One-Time Password Authentication Scheme without Counter Desynchronization, in *ISECS International Colloquium on Computing, Communication, Control, and Management*, Sanya. *p* 361 - 364.

4. Lamport L. (1981). Password Authentication with Insecure Communication, *Communications of the ACM*, ***Vol.* 11**. No: 24, *p* 770-772.

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

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

7. W. Stallings. (2011). *Cryptography and Network Security Principles and Practices, Chapter 11 Cryptographic Hash Functions*, Fifth Edition ed.: Prentice Hall.

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

9. Daemen J., Peeters M., Assche G. V., Bertoni G. (2007). Sponge Functions, in *Ecrypt Hash Workshop*.

10. Daemen J., Peeters M., Assche G. V. Bertoni G. (2008) On the Indifferentiability of the Sponge Construction, in *EuroCrypt*.

11. Pelzl J., Paar C. (2010). Hash Functions, in Understanding of Criptography. Berlin Heidelberg: Springer Verlang, pp. 293-317.

12. İncearık M. E., Paksoy M. (2014). Tasarımdan Programlamaya Mobil Uygulama Geliştirme, Kodlab.

13. Uslu B. (2015) Android Tabanlı Mobil Uygulama Geliştirme, Kodlab.