T.C.
PAMUKKALE ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

UZAKTAN ERİŞİMLİ KRİPTOGRAFİK GÜVENLİ HABERLEŞME PROTOKOLÜ

Alper UĞUR

Yüksek Lisans Tezi

DENİZLİ – 2005

UZAKTAN ERİŞİMLİ KRİPTOGRAFİK GÜVENLİ HABERLEŞME PROTOKOLÜ

Pamukkale Üniversitesi
Fen Bilimleri Enstitüsü
Tarafından Kabul Edilen
Bilgisayar Mühendisliği Anabilim Dalı
Yüksek Lisans Tezi

Alper UĞUR

Tez Savunma Tarihi: 08.07.2005

DENİZLİ – 2005

TEZ SINAV SONUÇ FORMU

Bu tez tarafımızdan okunmuş, kapsamı ve niteliği açısından Yüksek Lisans Tezi olarak kabul edilmiştir.

Yrd. Doç. Dr. Murat AYDOS
(Yönetici)

Yrd. Doç. Dr. A. Kadir YALDIR
(Jüri Üyesi)

Yrd. Doç. Dr. Serdar İPLİKÇİ
(Jüri Üyesi)

Pamukkale Üniversitesi Fen Bilimleri Enstitüsü Yönetim Kurulu'nun
...../...../..... tarih ve ..... sayılı kararıyla onaylanmıştır.

Prof. Dr. Mehmet Ali SARIGÖL
Müdür
Fen Bilimleri Enstitüsü

TEŞEKKÜR

Başta ileri görüşlülüğü ve fikirleriyle akademik hayatıma yön veren ve bana olan güvenini asla boşa çıkaramayacağım değerli danışman hocam Yard. Doç. Dr. Sayın Murat AYDOS olmak üzere, her zaman için benden değerli desteklerini ve güler yüzünü esirgemeyen başkanım Doç. Dr. Sayın Halil KUMSAR'a, kendisiyle çalışmaktan onur duyduğum hocam Yard. Doç. Dr. Sayın A. Kadir YALDIR'a, en yoğun anında bile bana ve sorularıma zaman ayıran Yard. Doç. Dr. Sayın Sezai TOKAT'a, bana akademisyenliği sevdiren ve kendilerinden çok şey öğrendiğim değerli hocalarım Yard. Doç. Dr. Sayın Necdet GÜNER'e, Yard. Doç. Dr. Sayın Ceyhun KARPUZ'a, Yard. Doç. Dr. Sayın Ahmet ÖZEK'e ve Yard. Doç. Dr. Sayın Serdar İPLİKÇİ'ye teşekkürü bir borç bilirim.

Ayrıca bu çalışmanın her anında bana varlığıyla destek olan değerli çalışma arkadaşım, "komşum" Şahin BAYZAN'a, askerden bile çalışmalarıma ivme sağlayan dostum Ali SAĞAT'a, çalışmalarımın en yoğun anlarında bile sabırlarını esirgemeyen değerli mesai arkadaşlarım Evgin GÖÇERİ ve Meriç ÇETİN'e, adlarını burada yazmaya kalksam tezi tamamlamaya fırsat bulamayacağım tüm araştırma görevlisi arkadaşlarıma, kendilerine her zaman borçlu kalacağım Sayın Bahtiyar MIHÇI'ya, Sayın Orhan AKKAŞ'a, Sayın Zihni DERİN'e ve bir bardak demli çayı benimle paylaşan herkese çok teşekkür ederim.

Son olarak bana ayakta durmayı değil uzak hedeflere koşmayı, doğru bildiklerimi söylemekten çekinmemeyi ve en önemlisi hayatın saklı bilgisini öğreten, benim diyebileceğim her şeyin asıl sahipleri, üzerimdeki haklarını asla tam olarak ödeyemeyeceğim canım anneme, babama ve varlıklarıyla beni dünyanın en şanslı kardeşi yapan, hakiki dostlarım ablam ve abime sonsuz teşekkürlerimi sunarım.

Alper UĞUR

ÖZET

Haberleşmeyi iki ya da daha çok taraf arasında birbirleriyle ilişkili iletilerin gönderilmesi olarak tanımlayabiliriz. İletilerin doğru şekilde aktarımı, taraflar arasındaki haberleşmenin belirlenmiş kurallar çerçevesinde yapılmasıyla sağlanabilir. Buna örnek olarak insanların birbirleriyle konuşmasını verebiliriz. Ancak ortak bir lisana sahip olan insanlar, bu lisana ait kurallar ve kelimeler sayesinde anlaşabilirler. Bu bağlamda, bir protokolü de haberleşmenin gerçekleşmesi için tanımlı kurallar bütünü olarak ele alabiliriz.

Bu çalışmada, uzak erişimli bir istemci ile yetkili istemcilere servis sağlayan bir sunucu arasında güvenli bir haberleşme protokolü tasarlanmıştır. Tasarım aşamasında dikkat edilmesi gereken güvenlik kriterleri ve bunları gerektiren sebepler ortaya konmuştur.

Güvenli bir haberleşme protokolü tasarlanırken bunun gerçeklenmesi için gerekebilecek bir kriptografik kütüphane de hazırlanmıştır. Tasarımda kullanılan kriptografik algoritmalar, standartlara uygun ve uygulamadan bağımsız olarak kodlanmış ve bu kütüphaneye dahil edilmiştir.

Genel olarak bütün kriptografik haberleşme protokollerinde kullanılan rasgele sayı serilerinin elde edilmesine yeni bir yaklaşım getirilmiştir.

Anahtar kelimeler: güvenli haberleşme protokolleri, sözde-rasgele sayı üreteçleri, hash kütüphanesi

Alper UĞUR

ABSTRACT

Communication can be defined as transportation of related messages between two or more parties. Exactness of message delivery may be provided with definite transportation rules between parties. This is same as human communication. Merely the people that know a common language can talk each other with words and rules of this language. At this point we can handle protocols as defined rules that a communication need to exist.

In this work, a secure communication protocol is designed between a server and a remote client. The security criterias that have importance in design phase and their cause of existence are clarified .

While studying on the design of a secure communication protocol, a cryptographic library was prepared for the simulation. The cryptographic algorithms used in design was implemented as in standards and embedded to the library.

A new approach was given to the input data of pseudo-random number generators that are in use of all cryptographic communication protocols, in general.

Keywords: secure communication protocols, pseudo-random number generators, hash library

Alper UĞUR

İÇİNDEKİLER

Sayfa

Birinci Bölüm

GİRİŞ

İkinci Bölüm

GÜVENLİ HABERLEŞME PROTOKOLÜ

Üçüncü Bölüm

ALGORİTMALAR VE STANDARTLAR

Dördüncü Bölüm

UYGULAMA

Beşinci Bölüm

SONUÇ VE ÖNERİLER

Ekler

TEST VEKTÖRLERİ

ŞEKİLLER DİZİNİ

Sayfa

ÇİZELGELER DİZİNİ

Sayfa

BİRİNCİ BÖLÜM
GİRİŞ

1. GİRİŞ

Tarihin başlangıcından günümüze, verilerin iletilmesinin yanında güvenliğinin sağlanması da haberleşme teknolojilerinin gelişim sürecinde önem taşımıştır. Özel mektuplarda zarf sistemi kullanılarak taşınan bilgi üçüncü taraflardan gizlenmeye çalışılmıştır. Ama bu sistem zarfın dikkatli açılıp kapatılması ile atlatılabilirdi. Sistemin bu açığına karşı gönderen tarafa ait özel işaret basılarak zarfta mühürleme kullanılmaya başlandı. Ne var ki bu işlem ancak zarfın yerine teslimatından emin olunduğu zaman işe yaramaktaydı. Zarfın üçüncü şahısların eline geçmesi durumunda alınan önlemlerin işe yaramaması ve özel bilgilerin açığa çıkması durumu söz konusuydu.

Bilgilerin türlü yöntemlerle kodlanıp gönderilmesi fikri taşınılan paket ele geçse ya da kaybolsa bile gizin korunmasını amaçlamak üzerine geliştirilmiştir. Bu koruma, kodlama yönteminin analize karşı dayanıklılığı ya da giz için ihtiyaç duyulan geçerlilik süresine göre farklılık göstermektedir. Bir takım bilgilerin kodlanarak uzun süreler saklanması gerekeceği gibi, haberleşme sürecinde aktarılan bir takım bilgilerinse o an için gizlenmesi yeterli olabilecektir.

Bu çalışmada bilginin matematiksel tabanlı yöntemler kullanılarak korunması ve gönderilebilmesinde iletinin güvenliğini sağlamasıyla var olan diğer problemler üzerine yoğunlaşılmıştır; iletinin doğru kişiden alındığının doğrulanması ve reddedilemez olması. Bunların sağlanması da iletilerin gönderici ve alıcılar tarafından ilişkilendirilmesi ile mümkün kılınmıştır.

Yukarıda değinilen durumlar ve çözümler kullanılacak ortamlara bağımlı olarak farklı güvenlik protokol şemalarını ortaya çıkarmıştır. Bu şemaların kullanımdaki geçerlilikleri yapılan analizler ve bu alanda geliştirilen standartların üzerine kurulmuş olmalarıyla doğru orantılıdır.

1.1 Problem Tanımı

Bu çalışmada; tanınmayan uzak bir bilgisayarın sunuculardaki kaynaklara erişimini amaçlayan bir protokol yapısında kimlik belirleme, doğrulama ve yetkilendirme problemleri ile trafik analizi ve yineleme gibi saldırılara karşı çözümler ele alınmıştır. Güvenli haberleşmenin sağlanabilmesi için gerekli ileti gizliliği ve veri bütünlüğü kavramları ön plana çıkarılmıştır. İletinin değiştirilmeden taraflar arasında taşınması için haberleşmede özel oturumlar veya korumalı tüneller oluşturulmaktadır. Böylece haberleşmenin üçüncü taraflara karşı gizliliği daha fazla artırılmaktadır. Bir protokolün teorik olarak sağlam temellere oturtulması kadar uygulamada da kullanılabilir olması önem arz etmektedir. Standartlarda yer alan algoritmalar güvenli bir şekilde kodlanmalı ve var olan yapıda kullanılabilmelidir. Bu da standartlara uyumlu ve modüler yapıda kriptografik bir kütüphane oluşturulmasını gerekli kılmıştır.

1.2 Amaç ve Yöntem

Bu çalışmada uzak sunucu erişimi ve yetkilendirme protokollerinin en gelişmişi olan ve yaygın biçimde kullanılan Kerberos protokolü incelenmiş, protokolün yapısı temele indirgenmiş, ele alınan kriterler kapsamında geliştirilip yeni bir protokol yapısının ortaya konulması amaçlanmıştır. Bu protokolde Kerberos'a ek olarak trafik analizi saldırısını etkisiz kılmak amacıyla trafik dolgulama tekniği rasgele port seçimi yöntemi ile birleştirilmiştir. Böylece haberleşme oturumları için özel bir tünel oluşturulmuştur. Protokol tasarımında standartlaşmış güvenlik algoritmaları kullanılmıştır.

Geçerliliğinin sınanabilmesi için protokol yazılım ortamına taşınmıştır. İlgili standartlardan hem bu çalışma hem de daha sonraki çalışmalarda kullanılmak üzere hash fonksiyonları, anahtarlı ileti denetim kodları ve gizli anahtarlı şifreleme modülleri içeren kriptografik bir kütüphane oluşturulmuştur. Bu çalışmada ayrıca kriptografik haberleşmenin genelinde kullanılan rasgele sayı üreteçleri için görüntü uzayından elde edilen verilerin kullanılması ile ilgili yeni bir yaklaşıma da yer verilmiştir.

Bu tez çalışmasıyla, uzaktan erişimli bir sunucu ile erişime yetkili istemciler arasında kurulacak haberleşme protokollerinin güvenliklerinin nasıl artırılabilineceğine çözüm bulmak

amaçlanmıştır. Bunu sağlarken yapılacak geliştirmelerin mevcut sistemlerde kullanılabilmesi de ulaşılmak istenen hedeflerden bir diğeridir.

1.3 Tezin Önemi ve Literatüre Katkısı

Yapılan literatür çalışması kriptografik güvenli haberleşme protokollerinin birbirlerinden özel problemlerin çözümlerinde farklılaştığını ortaya koymuştur. Bu çalışmada da trafik analizini temel alan saldırılara karşı trafik dolgulama ve tünelleme yöntemleri birleştirilmiş ve kriptografik haberleşme protokollerinde sınama cevap aşamalarında gerekli olan rasgele sayı serilerinin üretimine yeni bir yaklaşım getirilerek bu konudaki boşluklar doldurulmaya çalışılmıştır. Ayrıca, protokolün gerçeklenmesi sırasında kriptografik bir kütüphanenin oluşturulması bundan sonra yapılacak akademik çalışmalara temel teşkil etmesi bakımından önemlidir.

1.4 Tezin Organizasyonu

Tez çalışmasının ilk bölümünde haberleşme güvenliğinin gelişimi hakkında genel olarak bilgi verilmiş ve çalışma ile amaçlananlar anlatılmış, kullanılan yöntemlere değinilmiştir. İkinci bölümde haberleşme protokollerindeki güvenlik kriterlerine ve bu kriterleri sağlamak için geliştirilen yapılara, protokollere yapılan saldırılara yer verilmiş, ortaya konan kriptografik güvenli haberleşme protokolünün tasarımı, ulaşılması amaçlanan hedefler belirtilmiş ve bu hedeflerin sağlanması için kullanılan gerek kriptografik gerek tasarımsal yöntemler açıklanmıştır. Üçüncü bölümde güvenli haberleşme protokolünde kullanılan algoritmaların standartlarına yer verilmiştir. Dördüncü bölümde tasarlanan protokolün .NET ortamında gerçeklenmesi ve yapılan uygulama detaylandırılmıştır. Son bölümde ise bu çalışma ile elde edilen sonuçlar ortaya konmuş ve gelecekte yapılabilecek çalışmalardan bahsedilmiştir. Ekler bölümünde ise güvenli haberleşme protokolünde kullanılan algoritmaların test vektörlerine yer verilmiştir.

İKİNCİ BÖLÜM
GÜVENLİ HABERLEŞME PROTOKOLÜ

2. GÜVENLİ HABERLEŞME PROTOKOLÜ

Protokoller iki ya da daha fazla taraf arasında bir görevi yerine getirmek için tasarlanmış birbirini takip eden basamaklar serisi olarak tanımlanabilir (Schneier, 1996). Protokolün ve tüm aşamalarının kullanıcı taraflarca iyi bilinmesi, anlaşılmaz, açık olmayan hiçbir yanının bulunmaması, protokolün olası her duruma karşı verebilecek bir cevabı, yani eksiksiz olması gerekmektedir. Bunlarla birlikte, kullanıcı taraflar protokolün tüm aşamalarını yerine getirmeyi kabul etmelidir. Bu taraflar, bir haberleşme protokolünde görüşen taraflar olabildiği gibi protokolde yer alan hakem ya da gözlemci rolündeki güvenilir üçüncü taraflar da olabilmektedir. Bu tanımlamadaki "güvenilir" kelimesi; bu tarafın sunduğu verilerin protokoldeki diğer taraflarca tereddütsüz doğru kabul edilmesi, kimi protokollerde ise paylaşılan gizi açığa çıkarmayacağı konusunda emin olunması anlamına gelmektedir.

Bu bilgiler ışığında tanımlamaları biraz daha özelleştirirsek, bir kriptografik protokol, aşamalarında kriptografik algoritmalar bulunduran ve genel amacı gizlilik ve taraflar arası karşılıklı dürüstlüğü sağlayan protokoldür. Taraflar protokolde, bir gizi paylaşmak, diğer tarafın kimliğini belirlemek gibi tek yönlü işlemler yapabilecekleri gibi, paylaşılan giz parçalarını birleştirerek bir değeri ortaya çıkarmak, bir anlaşmayı imzalamak gibi ortak işlemler de yapabilirler.

Protokol kurulurken ve protokolün iç aşamalarında, taraflar karşılıklı öz-dürüstlüğe zorlanarak üçüncü bir taraf olmayacağı gibi, sadece protokolün doğru çalışmasını denetlemek, herhangi bir aksi durumda protokolü kurtarmak gibi protokol yapısına göre farklı görevleri üstlenecek güvenilir bir üçüncü taraf da protokole dahil olabilir.

2.1. Güvenlik Kriterleri

Bir haberleşme protokolünün temel amacı, taşınan bilginin güvenliğinin sağlanmasıdır. Bilgisayar ve ağ güvenliği araştırma ve üretimi üzerine yapılan çalışmalar, bilgi güvenliği için

belirlenmiş kriterler üzerine yoğunlaşarak bu güvenlik kriterlerini temel almaktadır (Menezes ve diğ.1997, Stallings 1998). Aşağıda bu güvenlik kriterlerine değinilmiştir.

Gizlilik: Bilgisayar sisteminde veya taşınılan bir bilginin yalnız yetkili kimse ya da programlar tarafından erişimini sağlar ve garanti eder. Erişimden kasıt; okuma, inceleme gibi bilgiyle ilgili işlemlerdir.

Kimlik doğrulama: Belgenin ya da iletinin asıl kaynağının belirlenmesini sağlar ve kimliğin gerçekliğini doğrular.

Bütünlük: Bilgisayar sisteminin özelliklerini veya iletiyi değiştirebilme hakkının sadece yetkili kişiler ya da programlarda olmasını sağlar. Değiştirmeden kasıt; yazma, durum değiştirme, silme, oluşturma, ve bekletme ya da gönderilen iletinin tekrarlanması işlemleridir.

Reddedilmezlik: Gönderici ve alıcının her ikisinin de bir ileti gönderiminde yapılan işlemi reddedemez olmasının sağlanmasıdır.

Erişim kontrolü: Sistem tarafından bilgi kaynaklarına erişimin kontrol altında tutulmasının sağlanmasıdır.

Erişilebilirlik: İhtiyaç halinde sistem özellik ve yeteneklerinin yetkili kişi ya da programlar için kullanılabilir olmasının sağlanmasıdır.

2.2. Protokol Saldırıları

Protokolü tanımlarken temel hedefin gizliliğin korunması olduğu belirtilmişti. Protokolün bu hedefini engelleyen ya da boşa çıkarmayı amaçlayan karşı işlemler protokole yapılan saldırılar olarak nitelendirilebilir.

Kriptografik saldırılar, doğrudan protokole olabileceği gibi protokolde kullanılan kriptografik algoritmalara, protokol ya da algoritmalar uygulamaya konulurken kullanılan tekniklere karşı da olabilmektedir.

Sistem üzerindeki etkilerine göre saldırılar, iki tür altında toplanabilir. Bunlardan birincisi sistem işleyişine doğrudan yönelik olan aktif saldırılar, diğeri ise sistem akışını etkilemeyen, genelde ileride kullanılmak üzere bilgi toplamaya yönelik olan pasif saldırılardır. Aşağıda bu iki türe dahil olan saldırı tipleri belirtilmiştir.

Engelleme: Sisteme ait bir işlev ya da gönderilen iletinin yok edilmesi, erişilemez ya da kullanılamaz hale getirilmesi. Bu tip bir saldırı aktif saldırıdır.

Yakalama: Yetkisiz program ya da şahısların erişim hakkı elde etmesidir. Yakalama saldırısı, gizlilik ilkesine yapılmış pasif türde bir saldırıdır.

Değiştirme: Bu tip bir saldırı, yetkisiz tarafın sadece erişim hakkı elde etmesiyle kalmayıp ileti ya da sistem işlevinin içeriğine müdahale ettiği bir aktif saldırıdır.

Uydurma: Yetkisiz program ya da şahısların sisteme taklit iletiler göndermesi ile gerçekleşen iletinin geçekliğine ve doğruluğuna yapılan aktif saldırıdır.

Bu saldırı türlerine protokol çerçevesinde bakıldığında; protokol dışından protokolün belirli bir kısmının ya da tamamının bilgi elde etmek amacıyla dinlenmesi protokolün işleyişini etkilemediğinden pasif saldırı olarak nitelendirilebilir. Eğer bir saldırgan iletileri değiştirerek, başkasıymış gibi davranarak, haberleşme kanalını keserek, protokolü kendi lehine olacak şekilde değiştirmeye çalışırsa, bu tip saldırılar protokolün üzerinde işlediği ağa bağlı olup, aktif saldırılar olarak nitelendirilir.

Protokole karşı yapılan saldırılar, kriptografik bir protokol tasarlanırken dikkat edilmesi gereken temel unsurları oluşturur. Tasarımın başlangıcında bunların belirlenmesi ve etkin önlemler alınması protokolün geleceği açısından önemlidir. Bu tespitler, maddeler halinde özetlenirse: (Lee, 1999)

- Korunacak varlıkların belirlenmesi
- Sistemin zayıflıklarının belirlenmesi
- Sistemde açığa yol açabilecek tehditlerin belirlenmesi
- Sisteme uygulanabilecek güvenlik düzenlemelerin belirlenmesi

- Kriptografik güvenlik gereksinimlerinin belirlenmesi
- Yukarıda sıralanan tespitleri adresleyen güvenlik servislerinin tanımlanmasıdır.

Yapılacak tespitler, seçilecek kriptografik servis ve yöntemler, var olan sistemlerin geliştirilmesi ya da yeni sistemlerin uygulanması hakkında bilgi kaynağı teşkil edecektir.

2.3 Güvenlik Sağlayıcı ve Saldırılara Karşı Geliştirilen Yöntemler

Belirlenen tespitler doğrultusunda sistemin gereksinim duyduğu yöntem ve mekanizmalar protokole dahil edilmelidir. Aşağıda haberleşme güvenliği için geliştirilen yöntem ve modellere yer verilmiştir.

2.3.1 Ağ Güvenlik Modeli

Daha önce değinilen güvenlik kriterlerinin yanında tüm güvenlik teknikleri iki bileşen içerir. Bunlardan biri ileti içeriğinin şifrelenmesi ve ileti gönderen ile ileti arasındaki ilişkiyi kuran ek bilgi benzeri iletim güvenliğini sağlayan bilgilerin gönderilmesi, diğeri de başkaları tarafından bilinmediği umulan, iletişim yapan taraflar arasında paylaşılan gizli bilgidir. Kimi durumlarda bunlara ek olarak daha önce de belirtildiği gibi güvenilir bir üçüncü taraf da güvenlik modeline dahil olup iletişim güvenliğini sağlamaktadır. Güvenilir üçüncü taraf, taraflar arası paylaşılan gizin dağıtımı veya kimlik doğrulama görevlerini üstlenmektedir.

Güvenlik modeli tasarımı dört temel parçaya ayrılmaktadır. (Stallings, 1998) Bunlar:

1. Güvenli iletişimi sağlayabilecek algoritmanın tasarlanması
2. İlgili algoritma ile kullanılacak gizli bilginin üretilmesi
3. Gizli bilginin dağıtımı ve paylaşımı için yöntemlerin üretilmesi
4. Bu güvenlik modelinin kullanılarak taraflar arası güvenli iletişimi sağlayacak protokolün belirlenmesi.

Bir protokolün bu tasarım modeli doğrultusunda hazırlanması işleyiş esnasında ortaya çıkacak tehditleri azaltacaktır.

2.3.2 Geleneksel Şifreleme Modeli

Şifreleme, gizliliğe yapılan saldırılar için kullanılan bir çözümdür. Paylaşılan gizin ya da iletilen verinin sadece taraflar için anlamlı olabilecek şekilde kodlanması esasına dayanır. Şifrelemenin nerede yapılacağı ve nelerin şifrelenmesi gerektiği kurulan protokol tasarımında verilmesi gereken en önemli kararlardandır.

Geleneksel Şifreleme Modeli; anlamlı ileti metninin kullanılacak bir algoritma ve ileti metninden bağımsız uzunluk ve biçimde bir anahtar ile rasgele gözüken anlamsız şifreli metin karşılığına dönüştürülmesi, şifreli metnin gönderildiği yerde şifreleme algoritması ile bağlantılı (ya da aynı) bir algoritma ve anahtar ile tekrar anlamlı düz bir metine çevrilmesi işlemidir.

Bu modelde önemli olan, algoritmanın elde edilmiş şifreli metnin düz metin karşılığına çevrilebilmesi için tek başına yeterli olmayacak biçimde dayanıklı olmasıdır. Bununla birlikte geleneksel şifreleme modeli algoritmanın değil anahtarın gizliliğine dayanmaktadır. Bir başka deyişle, sadece anahtarın gizli tutulması güvenliği sağlamak için yeterli olmalıdır.

Bu şekilde, matematiksel olarak zorluğu/geçerliliği ispat edilip öne sürülen şifreleme algoritmalarının açık olması, hem bir meydan okumadır, hem de akademik bilim çevrelerince geliştirilip algoritmaların zayıf taraflarının kapanması için iyi bir yoldur.

Kriptografik sistemler genelde üç bağımsız boyutta sınıflandırılır.

1. Düz metni şifreli metne dönüştürmede kullanılan yöntemler: Tüm şifreleme algoritmaları iki genel yönteme dayanır. Bunlardan biri ikame (yerine koyma) yöntemi, diğeri ise yer değiştirme yöntemidir. İkame yönteminde düz metindeki harf, ikili kelime bloğu gibi algoritmik elemanların yerine bunlara karşılık getirilen elemanlar konulur. Yer değiştirme yönteminde ise yapılan kaydırmalar ile düz metin tekrar düzenlenir. Dikkat edilmesi gereken en önemli nokta bu işlemler esnasında hiçbir bilginin kaybolmamasıdır. Böylece yapılan tüm işlemler tersinir özelliklerini koruyacaktır.

2. Kullanılan anahtar sayısı: Bu sınıflandırma, sistemin çalışma ilkesi ile ilgilidir. Eğer kullanılan anahtar her iki tarafta aynı ise bu sistem, simetrik, gizli anahtarlı ya da tek anahtarlı

şifreleme olarak adlandırılır. Eğer her iki taraf iletinin dönüştürülmesi ve tekrar çevrilmesi için farklı anahtarlar kullanıyorsa bu sistem asimetrik, açık-anahtarlı ya da iki-anahtarlı şifreleme olarak adlandırılır.

3. Düz metnin işlenmesi: Sisteme giren iletinin işlenme şekli yeni bir sınıflandırma ortaya koymaktadır. Eğer düz metin belirli boyutlardaki bloklara ayrılıp sistemde bir anda bir blok işlem görüp her bir blok girdisi bir blok olarak çıktı veriyorsa bu işleme modeli blok şifreleme, eğer düz metin sürekli olarak işlenip bir tek çıktı veriyorsa bu işleme modeli de akış şifreleme olarak sınıflandırılır.

Kriptografik sistemleri oluşturan diğer temel taşlar ilk defa Claude Shannon (1970) tarafından ortaya konan ve kabul gören dağılım ve düzensizlik terimleridir. Bu temeller sistemi istatistiksel analizlere karşı geliştirilmiştir. Dağılım özelliği düz metin ve şifreli metin arasındaki ilişkinin istatistiksel araştırmalara uzun süreli dayanmasını sağlar ve bu özellik düz metindeki tek bir ikilin değişmesinin karşılık gelen şifreli metinde birçok ikili etkilemesi ile sağlanır. Bu özellik aynı zamanda çığ efekti olarak da adlandırılır. Bir diğer özellik olan düzensizlik özelliği ise anahtar ve şifreli metin arasındaki ilişkiye dayanan istatistiksel testlere dayanıklılığını amaçlar. Bu özellik ise karmaşık yerine koyma algoritmaları ile sağlanır.

2.3.3 İleti Kimlik Denetleme ve Hash Fonksiyonları

Şifreleme, gizliliğe yönelik pasif ataklara karşı koruma sağlar. Uydurma iletiler için farklı yöntemler kullanmak gerekmektedir. İletilerin gerçekliğine yapılan bu tip aktif saldırılara karşı kullanılan yöntem ileti kimlik denetlemedir.

Bir ileti, dosya ya da komut geldiği iddia edilen yerin gerçekliliğinin ispatlanabilirliği ölçüsünde hakikidir, özgündür. İleti kimlik denetimi bunu doğrulamak için kullanılır. Yapılan işlem, iletinin değiştirilmemiş olduğunun ve geldiği adresin doğruluğunun teyididir. Bunlarla birlikte, herhangi bir tekrarlama aktif saldırısına karşı iletinin zamana veya önceki ileti alışverişlerine bağlılığının da kontrol edilmesi gereklidir.

Geleneksel şifreleme kullanılarak da ileti doğruluğu kontrol edilebilir. Eğer gizli anahtarın sadece taraflar arasında bilindiği varsayılırsa iletiyi uygun şifreli olarak gönderen tarafın gerçekliği ispat edilmiş olur. Bununla birlikte iletiye hata denetim kodu eklenerek veya

haberleşme esnasında o anki iletiye ait bir sıra numarası eklenerek değişimin olmadığı ve akışın doğruluğu da kontrol edilebilir. Eğer ileti, zaman-pulu denilen ileti gönderim zamanını gösterir bir ek bilgi ile donatılırsa iletideki gecikmenin tespiti de yapılmış olur.

Genelde açık olarak ileti ile birlikte gönderilen şifrelemeye benzer yapıda olan ileti kimlik denetim kodu bu işlevleri sağlamak için düşünülmüş bir yöntemdir. Veri paketlerinin genelde uçtan uca şifreli gönderilmesi, bazı trafik yükü ağır olan sistemlerde ise kimlik denetimi için gerekli çözme işlemi zamanından tasarruf sağlanması amacıyla bu denetim kodu açıktan gönderilmekte ve ayrı bir şifreleme işlemine tabii tutulmamaktadır.

Kullanılan ileti kimlik denetimi yöntemlerinden biri, iki taraf arasında paylaşılan bir gizli anahtar ve tek yönlü bir fonksiyona sokularak iletiden elde edilen özetin küçük bir veri bloğu oluşturularak gönderilen iletiye eklenmesidir. Karşı taraf bu veri bloğunu açtığında paylaşılan anahtar tutuyorsa gönderen tarafın kimliği denetlenmiş olur. Gelen iletiye aynı işlem uygulanarak elde edilen özet eğer gelen bloktaki özet ile birebir uyuşuyorsa iletinin değiştirilmemiş olduğu da kontrol edilmiş olur. Yapılan ileti özeti çıkarma işlemi şifreleme işlemine benzemekle birlikte tersinir değildir, yani özetten tekrar iletinin kendisi elde edilemez. Şifrelemede daha önce de değinildiği gibi tekrar çözme ve şifrelenen metin ya da dosyanın elde edilmesi için veri kaybı olmaması esastır. Oysa bir metnin ya da dosyanın özeti çıkarılırken o metne ya da dosyaya özel sıkıştırılmış kısaltılmış bir bilgi oluşturulması söz konusudur.

2.3.4 Trafik Gizliliği ve Trafik Dolgulama Yöntemi

Bir protokolde taraflar arasında kurulan haberleşme esnasında veri trafiğinin analizi de haberleşme güvenliğini tehdit eden etkenlerdendir. Trafik analizlerinden elde edilebilecek bilgiler aşağıda sıralanmıştır.

- Tarafların kimlikleri
- Görüşme sıklıkları
- İletiye ait model, uzunluk gibi özellikler ve önemli bir bilgi değişimi için yapılan görüşme miktarı
- Taraflar arası yapılan özel görüşmelerle ilgili olaylar

Bağlı şifreleme yapılması ya da paket başlıklarının şifreli gönderilmesi trafik analizinin etkisini azaltmaya yönelik çalışmalardandır. Ama bu düzenekler sisteme giren ve çıkan veri miktarına yönelik sıklık ve çokluk analizlerini engellememektedir.

Bu tip trafik analizlerini etkisiz hale getirmenin bir yolu da trafik dolgulamadır. Trafik dolgulama düz metnin yokluğunda sürekli şifreli metin üretme metodudur.

Düz metnin varlığında metin şifrelenir ve gönderilir, eğer girdi olarak düz metin yoksa rasgele üretilen veriler şifrelenerek trafik akışı devam ettirilir. Böylece pasif saldırganın giden verilerin bilgi mi yoksa gürültü mü olduğuna dair tespit yapması imkansızlaşır.

2.3.5. Oturum Gizliliği ve Oturum Anahtarları Yaşam-Süreci

Taraflar arasında kurulan her bir oturumun farklı anahtarla şifrelenmesi, protokole karşı yapılan şifre kırma analizlerini faydasızlaştıracaktır. Oturum anahtarları ne kadar sıklıkla değiştirilirse yapılan haberleşme o kadar güvenlidir. Çünkü ilgili oturum anahtarı ile ilgili istatistiksel ve diğer analiz türleri için çok fazla bağlantılı şifreli metin bulunmayacaktır.

2.4. Protokol Tasarımında Standartların Kullanımı

Standartlar, var olan kullanımları, yöntemleri ve ölçümleri tanımlar. Herhangi bir ürünün inşasında standartların kullanımı güvenilirliği ve ürün etkililiğini artıracağı gibi ürünün belirli bir kalite seviyesinde üretildiğini gösterir.

Standartlar, geniş bir topluluk tarafından kabul gören ve ilgili konunun uzmanları tarafından değerlendirilen çözümleri içerir. Bununla birlikte standartlar bilişim teknolojilerinde birlikte çalışabilirlik, güvenlik ve bütünlüğü sağlar (Lee, 1999).

Bir ürün, aynı standardı sağlayan diğer ürünlerle uyumlu çalışabilmelidir. Örneğin, herhangi bir üretimci tarafından standarda uygun kriptografik algoritma ile yapılmış bir şifreleme programı kullanılarak yapılan bir dönüşümün, aynı standarda uyan başka bir üretici tarafından üretilmiş program ile geri dönüşümü sağlanabilmelidir.

Standartlaşan veya kabul gören bir algoritmanın, bu aşamaya gelene kadar, belirli bir dönem boyunca akademik, araştırmacı ve uzman toplulukların analiz ve yorumlarından geçmesi; kriptografik standartların belirli bir güvenlik seviyesi sağladığını ortaya koymak için yeterlidir.

2.5. Protokol

Bu çalışmada ortaya konulan protokol, uzaktan erişimli bir sunucu ile erişime yetkili istemciler arasındaki iletişimin kriptografik olarak güvenliğini sağlamayı amaçlayan bir haberleşme protokolüdür.

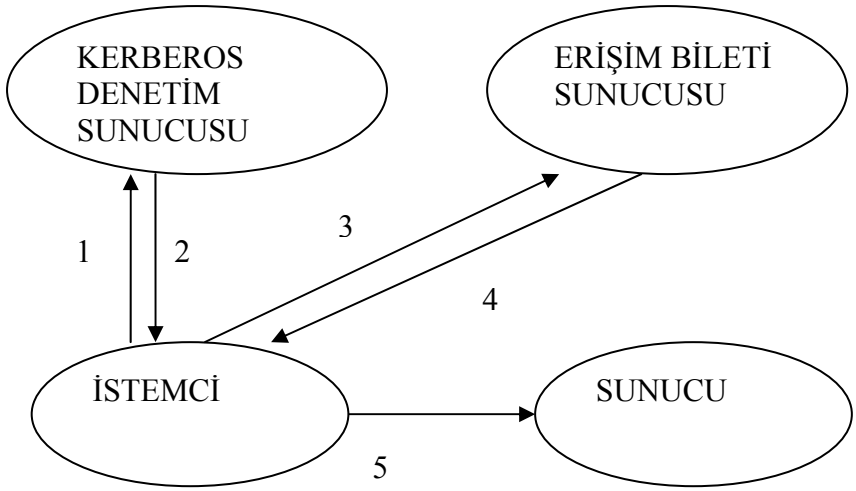
Bu tip haberleşme protokollerinden en gelişmişi ve yaygın olarak kullanılanı Kerberos protokolüdür. Çalışmada bu protokol üzerine yoğunlaşılmış ve belirlenen kriterler kapsamında protokolün geliştirilmesi hedeflenmiştir.

2.5.1 Kerberos Protokolü

Kerberos, Massachusets Teknoloji Enstitüsü'nün (MIT) Athena bilgisayar sistemi projelerinin güvenliğini sağlamak amacıyla geliştirilmiş olup, uygulama seviyesindeki en önemli kimlik doğrulama servislerinden biridir. Kerberos sistemi, dağıtık bir ağda dış istemcilerdeki kullanıcıların sunuculara erişimi ile ilgilenir. Erişim isteklerinin sınırlandırılması ve yetkili istemcilerin kimlik doğrulaması servislerini sağlar. Dağıtık sistemlerde sistemdeki istemci bilgisayarların erişim için doğrudan yetkilendirilmesi söz konusu değildir. Çünkü, herhangi bir kullanıcı istemcilerden birini kullanıp doğrudan erişim isteyebilir ya da ağ adresi değişikliği ile sunuculara yetkili görünmeye çalışabilir. Dolayısıyla bu istemciler ve sunucular arası kimlik doğrulama protokolü kurulması gereklidir. Ama bu şekilde her sunucu istemci arasına protokol kurulması yerine Kerberos sistemi sunucular ve istemciler arasında merkezi bir kimlik doğrulama servisi sağlar.

Protokole baktığımızda temel senaryo, A kullanıcısının sisteme erişim isteğiyle başlar. Kullanıcı iş istasyonunda oturum açar ve sunucudan servis isteğinde bulunur. Denetim sunucusu kullanıcının veritabanına erişim haklarını doğrular, erişim bileti için izin verir ve

oturum anahtarını kullanıcıya iletir. Sonuçlar kullanıcının parolasından elde edilen anahtarla şifrelenerek gönderilir.



Şekil 2.1: Kerberos Protokolü Şeması (Schneier, 1999)

İş istasyonu şifreli gönderilmiş iletinin açılması için kullanıcıdan parolasını ister ve doğru girildiği takdirde erişim izni ve kimlik belirleyiciyi gönderir. Denetim sunucusu tarafından gönderilen kimlik belirleyicisi, kullanıcının ismini, ağ adresini ve erişim bileti sunucusuna bağlantı zamanını içerir.

Erişim bileti sunucusuna servis için istekte bulunulduğunda sunucu erişim bileti iznini ve kimlik belirleyiciyi çözer, isteği doğrular ve erişim isteğinde bulunulan sunucu için bir bilet oluşturur.

İş istasyonu bu bileti ve kimlik belirleyiciyi sunucuya gönderir. Sunucu bileti ve kimlik belirleyiciyi doğrular ve hizmete erişim izni verir. Eğer karşılıklı kimlik doğrulama gerekiyorsa sunucu da kendi kimlik belirleyicisini gönderir.

2.5.2 Protokolün Tasarımı

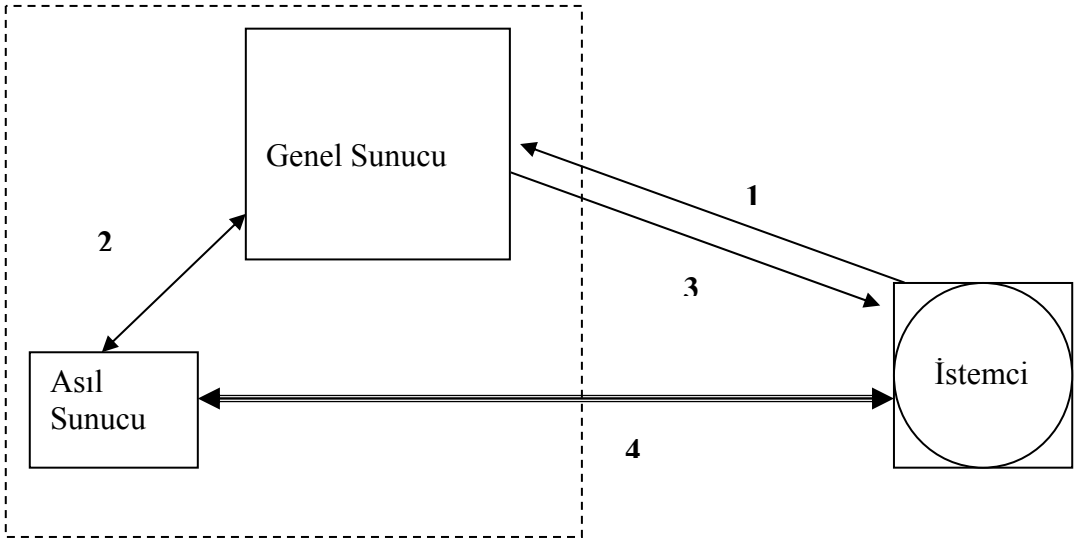
Bu çalışmada tasarlanan uzaktan erişimli kriptografik güvenli haberleşme protokolü Kerberos protokolünün kimlik doğrulama şeması çekirdek olarak alınmış, istemci ve sunucu arasında özel bir tünelleme modeli geliştirilerek trafik analizlerine karşı protokolün

dayanıklılığı sağlanmıştır. Bu modelde taraflar arasında kurulan tünelde trafik dolgulama yapılmış ve oturum anahtarı yaşam süreci ile zorlanan şifreli metne karşı yapılacak analizler;

- asıl veri iletiminin yapıldığı kanala da yaşam süreci uygulaması getirilerek ve
- taşınan veri paketinde protokole özel, ileti bloğu değişikliği yapılarak

sunucu istemci arasında kurulan bağlantı aşamasında karmaşıklık arttırılmıştır.

Protokolde taraflar arasında karşılıklı taşınan veriler simetrik anahtarlı şifreleme algoritmaları ile gizlenmiş olup kimlik denetimi süreci sıfır-bilgi ispat yöntemine bağlı kalınarak tarafların gizli anahtarları açığa çıkmadan sınama-yanıt aşamalarıyla sağlanmıştır. Bu aşamalarda kullanılan rasgele sayı serilerinin üretimine Bölüm 2.7'de yer alan, görüntü uzayından veri elde etme yöntemiyle yeni bir yaklaşım getirilmiştir.



Şekil 2.2: Protokol genel şeması

Şekil 2.2'deki şema, protokolün işleyiş aşamalarını göstermektedir. Bu aşamaları özetlemek gerekirse; istemci, genel sunucu ile bağlantı kurup asıl sunucuya erişim isteğinde bulunur. Genel sunucu istemciyi kimlik doğrulama işlemine tabi tutar. Daha sonra genel sunucu asıl sunucudan kimliği belirlenmiş istemci için yetkilendirme onayı ister. Asıl sunucu ilgili istemci ile bağlantı için özel bir tünel ve oturum anahtarı sunar. Genel sunucu bu bilgileri istemciye iletir. İstemcinin asıl sunucuya erişim isteğiyle protokol tamamlanır.

Aşağıda bu protokolde bir seviye daha derine inilerek her adımda gerçekleşen işlemler ayrıntılandırılmıştır. Protokolün gösteriminde kullanılan kısaltmalar:

İ: istemci

Gs: Genel sunucu

As: Asıl sunucu

IP_i : i istemcisinin IP numarası (kimliği)

Ka_b: a ve b taraflarının paylaştığı ortak anahtar

Kot: oturum anahtarı

t: zaman pulu

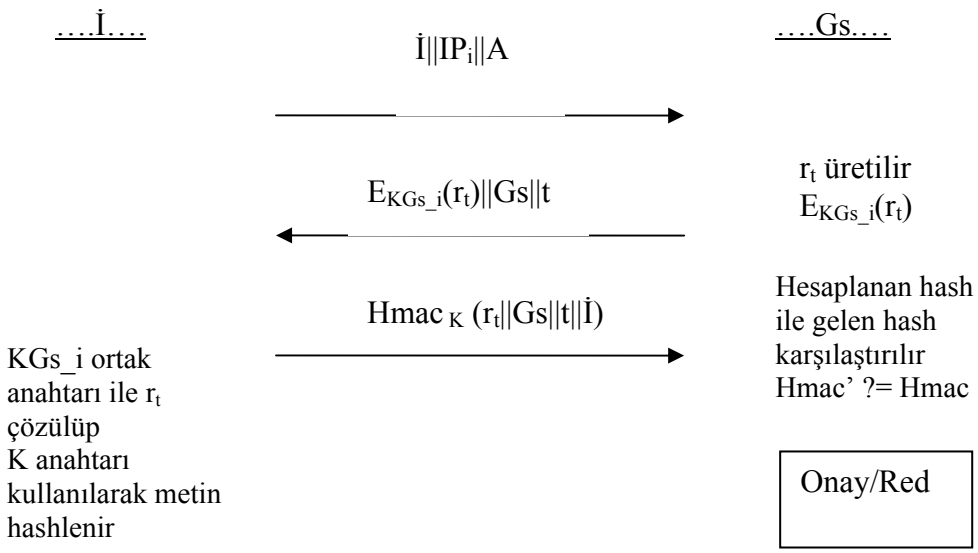
r_t: t zamanına ait rasgele sayı

$P_{i...j}$: tünel için açılan portlar

x: haberleşme için kullanılacak asıl port

Basamak 1:

İ ve Gs arasında geçer. Protokolün bu basamağında yapılan işlemler; İ nin Gs den A ile bağlantı kurulmasında gerekli bilgilerin temini için istekte bulunması ve Gs nin İ nin kimliğini doğrulamak için cevap-yanıt sürecini devreye sokmasıdır.



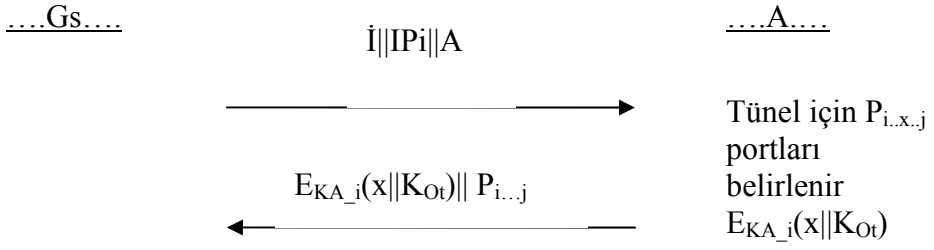
Şekil 2.3: Protokolün birinci basamağı

Birinci basamakta şifreli gönderilemesi gereken tek veri istemci kimliğinin doğrulanmasında kullanılacak olan r_t rasgele sayısıdır. r_t rasgele sayısı t zamanı ile birlikte Gs

sunucusunda cevaplanma süresince tutulur. Bu basamakta eğer hesaplanan hash sonuçları eşleşirse; t değeri iletinin doğru oturuma ait olduğunu ve tekrar iletisi olmadığını belirlemekte ve r_t rasgele sayısının doğru gizli anahtar ile çözümlenip hashlendiğini ortaya koymaktadır. Böylece oturum ve istemci kimliği denetlenmiş olacaktır.

Basamak 2:

Gs ve A arasında geçer. Protokolün bu basamağında Gs A ya istekte bulunan istemci ile ilgili bilgileri verir. A Gs ye istemci ile kendi arasındaki oturum için belirlediği oturum anahtarını ve tünel özelliklerini İ ile A arasındaki ortak anahtarla şifrelenmiş olarak iletir.

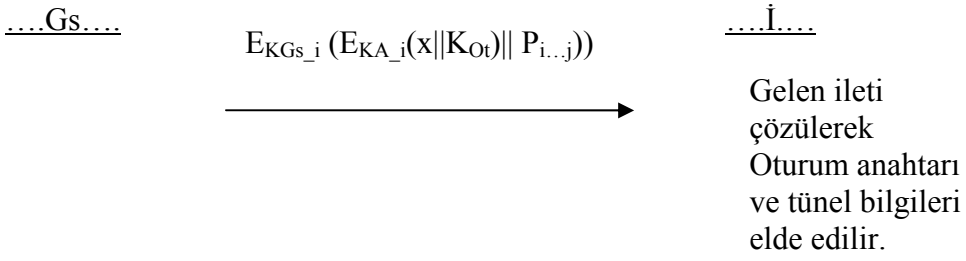


Şekil 2.4: Protokolün ikinci basamağı

Gs ve A sunucularından her biri aynı sisteme dahil olduklarından bu iki sunucu arasında veri şifrelemesi uygulanmamıştır. Böylece A ve Gs arasında gizli anahtar gereksinimi ortadan kaldırılmıştır. Ama A sunucusu ile yetkili İ istemcisi arasında kurulacak oturuma ait (Kot, x) oturum anahtarı ve haberleşme başlangıç portu çifti Gs sunucusundan gizlenmiştir. Bunun yapılmasındaki hedeflerden biri Gs genel sunucusunun bile açık verebileceği hesaplanarak yapılacak görüşmenin üst seviyede olası güvenliğinin sağlanması, diğeri ise genel sunucu Gs nin sistemde bulunabilecek birçok A sunucusu için bu veri çiftinin hesaplaması görevinin A sunucularına devredilerek genel sunucudaki işlem yükünün azaltılmasıdır. Zaten şifreleme işlemi doğrudan istemci ve asıl sunucunun ortak gizli anahtarı ile yapılmaktadır.

Basamak 3:

Gs ve İ arasında geçer. Protokolün bu basamağında Gs İ ye A tarafından belirlenen şifreli parametreleri Gs ve İ ortak anahtarı ile şifrelenmiş olarak gönderir.

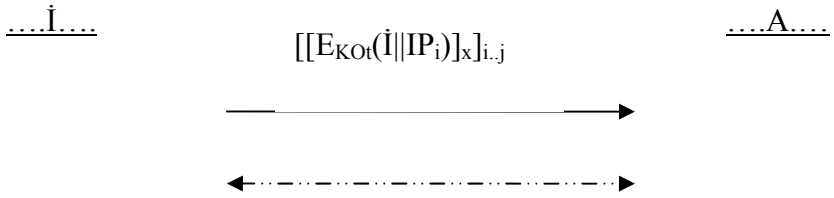


Şekil 2.5: Protokolün üçüncü basamağı

Bu basamak protokolde A ve İ arasında kurulacak oturum hakkındaki önemli verileri içerdiğinden tüm iletinin şifrelenmesi tercih edilmiştir.

Basamak 4:

İ ve A arasında geçer. Protokolün bu basamağı asıl haberleşmenin yapıldığı kısımdır. Oturum için açılmış portlarla oluşturulan tünelde belirlenmiş oturum anahtarı ve ana (x) portu ile haberleşme başlatılır. Diğer portlardan trafik analizleri ve kriptografik analizlerde karmaşıklığı sağlayacak şekilde şifreli rasgele veriler gönderilir. Ana haberleşme portu oturum süresince tünel içindeki diğer portlar arasından bir başkasıyla değiştirilerek karmaşıklık seviyesi artırılır.



Şekil 2.6: Protokolün son basamağı

Bu basamakta, pakette taşınan ek bilgi bloğu ile tüm portların kesikli kullanımı trafik dolgulama ve karmaşıklığı sağlayan port değiştirme yöntemine alternatif olarak seçilebilir. Bu yöntemde alınan paket bir sonraki paketin geleceği port bilgisini veri bloğunda taşıyacaktır.

OOOO	

Ek port bilgisi Kullanılacak veri bloğu

Şekil 2.7: Değiştirilen veri bloğu yapısı

……..				
		3	…	
1			…	n
	2		…	
…….				

Şekil 2.8: Tünelde veri paketlerinin rasgele akışı

Taşınılan veri paketleri 2^4 farklı porta rasgele dağıtılarak karmaşıklığın tüm portlara yayılması ve rasgele port seçimi sonucu yapılacak saldırılar da engellenmiş olacaktır.

Protokol tasarımlarında ve kriptografik yöntemlerde güvenlik seviyesinin artırılması çoğunlukla performans düşüşüne sebep olmaktadır. Bu yöntemde de göz önünde bulundurulması gereken, yöntemin taraflara tüm portları dinleme, anlamlı paketleri dağıtma/işleme gibi bir görevi yüklemesi ve haberleşme esnasında gönderilecek paket sayısının artmasıdır.

2.6 Rasgele Sayı Üretiminde Görüntü Uzayının Kullanılması

Bu bölümde gerek protokol kurulmadan yapılan sınama-doğrulama aşamaları gerekse veri bütünlüğünün sağlanmasında kullanılan rasgele sayı üretimi ve oturumun güvenliğini sağlayan oturum anahtarı üretimi için kullanılabilinecek güvenli haberleşmede görüntü uzayının kullanılması ve elde edilen verilerin işlenmesi yöntemi (Uğur ve Aydos, 2005) açıklanacaktır .

2.6.1 Sözde (Pseudo) Rasgele Sayılar

Sistemlerde ve hemen hemen her derleyicide hali hazırda bulunan rasgele sayı üreteçlerinin geçerli rasgele sayılar üretememesi ve bu sayıların kriptografik olarak kesinlikle güvenli olmadığı bilinmektedir. Bilgisayarların deterministik yapısı sebebiyle üretilecek sayının gerçek bir rasgele sayı olması beklenemez. Bunun yerine kriptografi, ihtiyacını sözde rasgele sayılardan karşılamaktadır.

Bir sayı serisinin sözde rasgele sayı serisi olması için

1- Sayının rasgele görünümünde olması
2- Bir sonraki sayının tahmin edilemez olması
3- Tekrar üretilemez olması

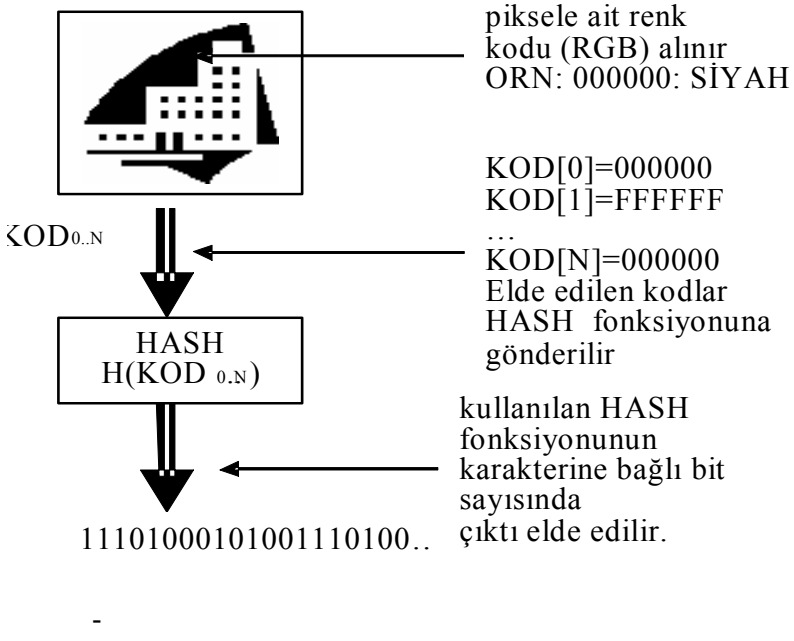
özelliklerini taşıması gerekir (Schneier, 1996).

Hash fonksiyonlarının büyük olasılıkla farklı çıkış vermeleri bu fonksiyonların rasgele sayı üretiminde kullanılabileceğini göstermiştir (FIPS PUB 180-2, 2002).

Bu fonksiyonlar için kullanılacak verinin görüntü serilerinden eldesi yukarıda sıralanan özelliklerin sağlanmasına katkıda bulunacaktır.

2.6.2 Rasgele Sayı Üretiminde Kullanılan Yöntem

Bu yöntem, görüntü uzayından veriyi, görüntü yüzeyinin piksellere doğrusal operatör matrisi ile ayrılarak her bir pikselde bulunan renk kodlarının işlenmesi ile sağlamaktadır. Daha sonra elde edilen bu veriler standartlaşmış hash fonksiyonları için (yapılan çalışmada hash fonksiyonu olarak FIPS 180-2'ye uygun olarak kodlanan SHA-256 kullanılmıştır) girdi oluşturarak sözde rasgele sayı serisini meydana getirmektedir. Bu yöntem Şekil-2.9'da şematik olarak gösterilmiştir.



Şekil 2.9: Görüntüden rasgele sayı üretimi

2.6.3 Görüntülerin Çeşitliliği Ve Optik Farklılık

Görüntünün elde edilmesi kullanılan donanım ve yazılıma bağlı olarak veri farklılığı oluşturduğu gibi, içinde bulunduğu ortamın da görüntünün özelliklerini değiştirdiği bir gerçektir. Işığın yüzeyden yansıyarak görüntüyü oluşturması, donanımlar arasındaki çözünürlük farkları, ortamın aydınlığı, optik açı farkları görüntüyü, dolayısıyla görüntüden elde edilecek veriyi etkileyen, değiştiren faktörlerdendir.

Kuramsal olarak fiziksel görüntüler sadece sonlu zaman aralıklarında gözlemlenebilirler. Buna bağlı olarak görüntü ışık fonksiyonu C(x,y,t,λ) gibi dört değişkenli bir fonksiyondur. Göreli kırmızı, yeşil, mavi (RGB) koordinat sistemi:

(x,y): uzaysal koordinat sistemi,

t: zaman

λ: dalga boyu

$R(x,y,t)= \int^{\infty}_{0} C(x,y,t,\lambda)\, Rs(\lambda) d\lambda$

$G(x,y,t)= \int^{\infty}_{0} C(x,y,t,\lambda)\, Gs(\lambda) d\lambda$

$B(x,y,t)= \int^{\infty}_{0} C(x,y,t,\lambda)\, Bs(\lambda) d\lambda$

değerlerinden oluşur. (Pratt, 2001)

Dolayısıyla görüntülerin zamana (t) ve ışık spektrumuna (Xs(λ)) bağlı değişken yapıda olduğu görülmektedir. Görüntüyü oluşturan denklem değişkenlerindeki farklılık, görüntünün de farklılaşmasına yol açacaktır. Bu durum farklı zaman dilimlerinde aynı plandan alınan görüntülerin farklı verileri taşıdıklarını göstermektedir.

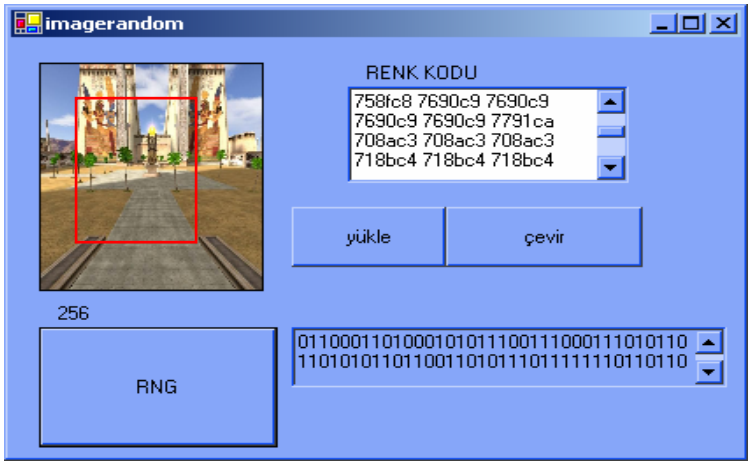
Bunlara ek olarak cihazın bakış açısındaki küçük değişiklikler, görüntünün kısmi ya da bütünsel olarak işlem görmesi tek kaynaktan elde edilecek veri çeşitliliğini artırmaktadır.

Tüm bu bilgilerin ışığında, varolan bu veri çeşitliliği görüntü uzayının rasgele sayı serisi üretiminde kullanılmasını cazip kılmıştır. Aynı yöntem ile oturum için gerekli anahtarın üretilmesi de uygulamaya dahil edilmiştir.

2.6.4 Rasgele Sayı ve Oturum Anahtarı Üreteci

Protokol esnasında, sunucu ve istemcinin kullandığı rasgele sayı serileri, arka planda çalışan rasgele sayı üreteci tarafından bahsedilen yöntem kullanılarak görüntü havuzlarından otomatik olarak üretilmektedir. Bu seriler, protokoldeki kimlik denetimi ve trafik dolgulama mekanizmalarında kullanılmaktadır.

Şekil 2.10 rasgele sayı serisi üretecinin, uygulamanın geliştirilmesi esnasında test amacıyla hazırlanmış olan arayüzü göstermektedir. Bu test modülü üretecin daha açık bir şekilde anlatılması için bu kısma dahil edilmiştir.



Şekil 2.10: Rasgele sayı üreteci

Görüntü havuzundan elde edilen görüntünün, sol üst köşede çerçeve ile belirtilen kısım işlenir ve renk kodları matrisine dönüştürülür. Bu matris oluşturulan kriptografik kütüphanedeki SHA-256 güvenli hash fonksiyonuna gönderilir ve 256 ikillik rasgele sayı üretilir.

Bu örnekte belirtilen çerçevenin yatay ve dikey eksenlerde hareketi, daraltılıp genişletilmesi ya da seçilen dikdörtgen alan yerine matematiksel eğriler ve bu eğrilerin altlarında kalan alanlar kullanılarak, bir görüntüden elde edilecek farklı veri bloğu sayısını artıracaktır.

Rasgele sayıların oturum başlamadan üretilip kullanılması uygulama esnasında performans sağlayacağı gibi kullanılacak oturum anahtarlarının da bu şekilde önceden hesaplanması mümkündür. Tarafların kullanılacak oturum anahtarını doğrudan belirtmek yerine haberleşme esnasında kullanılacak görüntü kimliği ve çerçeve özelliklerini göndermesi oturum anahtarının açığa çıkmasını engelleyici başka bir güvenlik unsuru olacaktır.

Haberleşme protokolünün tasarımında önemli güvenlik kriterleri göz önünde bulundurulmuş, kullanılan tünel şeması ile trafik analizlerine karşı protokolün dayanıklılığı artırılmıştır.

Geliştirilen rasgele sayı üreteci yöntemi ile üretilen geçerli rasgele sayıların haberleşme protokolü uygulamalarındaki kimlik denetimi ve yetkilendirme aşamalarında kullanılması, deterministik yöntemlerle üretilen rasgele sayılara nazaran güvenlik seviyesini artırıcı rol oynayacaktır.

ÜÇÜNCÜ BÖLÜM
ALGORİTMALAR VE STANDARTLAR

3. ALGORİTMALAR VE STANDARTLAR

Bu bölümde protokolde ve uygulamanın güvenliğini sağlamak için kullanılan SHA, HMAC ve RC6 algoritmalarına yer verilmiştir. Standartlaşmış olan SHA ve HMAC algoritmaları ilgili standartlardan metinler korunarak bu bölüme aktarılmıştır. Kriptografik kütüphanede yer alan algoritmalar için test örneklerine ekler kısmında yer verilmiştir.

3.1 SHA Güvenli Hash Standardı

Standart bilgileri:
1-Standardın Adı: Güvenli Hash İmza Standardı (SHS) (FIPS PUB 180-2)
2-Standart Kategorisi: Bilgisayar Güvenliği Standardı, Kriptografi
3-Açıklama: Bu standart elektronik verilerin (iletilerin) yoğunlaştırılmış gösterimini hesaplamak için kullanılan SHA-1, SHA-256, SHA-384, SHA-512 adlı dört güvenli hash algoritmasını tanımlar. SHA-1 ve SHA-256 için uzunluğu 2^{64} bitten az ya da SHA-384 ve SHA-512 için uzunluğu 2^{128} bitten az iletiler algoritma için girdi oluşturduğunda sonuç ileti özeti denilen çıktıdır. İleti özetleri 160 ila 512 bit uzunlukları arasında kullanılan algoritmaya bağlı olarak değişir. Güvenli hash algoritmaları genelde sayısal imza algoritmaları, anahtarlı hash ileti denetim kodları veya rasgele sayı (ikili) üretimi gibi diğer kriptografik algoritmalarda kullanılır.

Bu standartta belirtilen dört hash algoritması da 1) verilen ileti özete karşılık gelen iletiyi bulmanın ya da 2) aynı ileti özetini üreten iki farklı ileti bulmanın hesaplamalara göre mümkün olmaması sebebiyle güvenlidir. İletideki herhangi bir değişiklik büyük bir olasılıkla farklı ileti özeti olarak sonuçlanacaktır. Bu da güvenli hash algoritmasının kullanıldığı bir anahtarlı-hash ileti kimlik denetimi algoritmasında veya bir sayısal imza algoritmasında doğrulama hatası ile sonuçlanacaktır.

Bu standart daha uzun ileti özeti üretimine açık üç algoritmanın eklenmesiyle FIPS 180-1 in yerini almıştır. Belirtilen SHA-1 algoritması önceki FIBS 180-1'de belirtilen algoritmanın aynısı olmakla beraber bazı yazım gösterimleri SHA-256, SHA-384 ve SHA-512 algoritmalarına uyacak şekilde değiştirilmiştir.

Standartta yer alan 4-7 sayılı başlıkların burada belirtilmesine gerek duyulmamıştır.

8.Uygulamalar: Bu standartta açıklanan güvenli hash algoritmaları, yazılım, bellenim, donanım veya bunların birleşimi şeklinde uygulanabilir. Sadece bu standarda uyan algoritmalar NIST tarafından onaylanmaktadır.

3.1.1 Standart Hakkında Ön Bilgi

Bu standart SHA-1, SHA-256, SHA-384 ve SHA-512 adlı dört hash algoritmasını içerir. Bu dört algoritmanın hepsi de ileti özeti denilen bir iletinin yoğunlaştırılmış karşılığını üreten, yinelemeli tek yönlü hash fonksiyonlarıdır. Bu algoritmalar bir ileti bütünlüğünün tespitine imkan tanır. Öyle ki iletideki herhangi bir değişiklik büyük bir olasılıkla farklı bir ileti özeti ile sonuçlanacaktır. Bu özellik sayısal imza doğrulaması, ileti denetim kodları ve rasgele sayı (ikili) üretimi için elverişlidir.

Her bir algoritma, önişlem ve hash hesaplaması şeklinde iki aşamayla tanımlanabilir. Önişlem iletinin dolgulanmasını, dolgulu iletinin m-ikillik bloklara ayrıştırılmasını ve hash hesaplamasında kullanılacak başlangıç değerlerinin oluşturulmasını içerir. Hash hesaplaması dolgulu iletiden bir ileti çizelgesi üretir ve bu çizelgeyi fonksiyonlar, sabitler ve kelime blok işlemleri ile birlikte kullanarak bir dizi hash değerini yinelemeli olarak oluşturur. Hash hesaplaması tarafından üretilen en son hash değeri ileti özetini belirler.

Dört algoritma, ileti özeti uzunluğuyla doğrudan ilişkili olan hash işlemine tabi tutulan veri için sağlanan ikili sayısı güvenliği kısmında birbirinden ayrılır. Güvenli bir hash algoritması başka bir algoritma ile birlikte kullanıldığında ikili sayısınca güvenlik seviyesi gerekliliği belirlenen şartlar arasında yer almalıdır. Örneğin, eğer bir ileti 128 ikili güvenlik seviyesinde bir sayısal imza algoritması ile imzalanmışsa, o imza algoritması 128 ikili güvenlik seviyesi

sağlayan güvenli hash algoritması gerektirmektedir (örn. SHA-256) . Çizelge 3.1'de güvenli hash algoritmalarının özellikleri özetlenmiştir.

Çizelge 3.1 Güvenli Hash Algoritmalarının Özellikleri

Algoritma	İleti Uzunluğu (ikili)	Blok uzunluğu (ikili)	Kelime bloğu uzunluğu (ikili)	İleti özet uzunluğu (ikili)	Güvenlik (ikili)
SHA-1	$<2^{64}$	512	32	160	80
SHA-256	$<2^{64}$	512	32	256	128
SHA-384	$<2^{128}$	1024	64	384	192
SHA-512	$<2^{128}$	1024	64	512	256

3.1.2. Tanımlar

Bu kısımda algoritma açıklanırken kullanılan terimlere ve kısaltmalara yer verilmiştir.

3.1.2.1 Terminoloji ve Kısaltmalar

İkili: 0 ya da 1 değerine sahip ikili sayı

Bayt: 8 ikili-sayı grubu

FIPS: Federal Bilgi İşlem Standardı

Kelime Bloğu: Güvenli hash algoritmasına bağlı 32 ikilik (4 bayt) ya da 64 ikilik (8 bayt) veri grupları

3.1.2.2 Algoritma Parametreleri, Semboller ve Terimler

Bu kısımda standartta kullanılan parametre, sembol ve terimlere yer verilmiştir.

3.1.2.2.1 Parametreler

Bu standartta belirtilen güvenli hash algoritmalarınca kullanılan parametreler aşağıda sıralanmıştır.

a,b,c, ... , h	$H^{(i)}$ hash değerlerinin hesaplanmasında kullanılan w-ikili kelime blok değişkenleri
$H^{(i)}$	*i* nci hash değeri. $H^{(0)}$ başlangıç hash değeri; $H^{(N)}$ ileti özetini belirlemekte kullanılan en son hash değeridir.
$H^{(i)}_j$	i nci hash değerinin j nci kelime bloğu, $H^{(i)}_0$ i nci hash değerinin en sol kelime bloğu.
K_t	Hash hesaplamasının t nci yinelemesinde kullanılacak sabit değer.
k	Dolgulama aşamasında iletiye eklenen sıfır sayısı
l	M iletisinin uzunluğu (ikili)
m	$M^{(i)}$ ileti bloğundaki ikili sayısı
M	hash işlemi uygulanacak ileti
$M^{(i)}$	m ikili uzunluğunda i ileti bloğu
$M^{(i)}_j$	i nci ileti bloğunun j nci kelime bloğu, $M^{(i)}_0$ i nci ileti bloğunun en sol kelime bloğu.
n	işlenen kelime bloğunda döndürülecek veya ötelenecek ikili sayısı
N	dolgulu iletideki blok sayısı
T	hash hesaplamasında kullanılan geçici w-ikili kelime bloğu
w	bir kelime bloğundaki ikili sayısı
W_t	ileti çizelgesindeki t nci w-ikili kelime bloğu

3.1.2.2.2 Semboller

Aşağıda her biri w-ikili kelime bloklarıyla çalışan ve güvenli hash algoritmalarının belirtiminde kullanılan semboller verilmiştir.

$\wedge$ ikili sistemde AND işlemi

$\vee$ ikili sistemde OR işlemi

$\oplus$ ikili sistemde XOR işlemi

$\neg$ ikili sistemde tümleyen işlemi

$+$ mod 2^w de toplama işlemi

<< sola öteleme işlemi, x<<n : x kelime bloğunun soldan n adet ikilin atılarak sonuca soldan n sıfır dolgulanarak elde edilir.

>> sağa öteleme işlemi, x>>n : x kelime bloğunun sağdan n adet ikilin atılarak sonuca sağdan n sıfır dolgulanarak elde edilir.

3.1.3. Gösterimler ve Yöntemler

Bu kısımda standartta yer alan gösterimler ve uygulanan yöntemlere yer verilmiştir.

3.1.3.1 İkili Dizgeler ve Tamsayılar

Aşağıdaki terminoloji kullanılan ikili dizgeler ve tamsayılar ile ilgilidir.

1. Bir onaltılık sayı {0,1,...,9,a,...,f} kümesinin bir elemanıdır. Bir onaltılık ikili 4 ikillik dizge ile gösterilir.. Örneğin onaltılık sayı tabanında '7' nin 4-ikillik karşılığı dize "0111", onaltılık sayı tabanında 'a' nın 4-ikillik karşılığı dize "1010" dır.
2. Bir kelime bloğu onaltılık ikiller halinde de ifade edilebilen w-ikillik dizgelerdir. Bir kelime bloğunu onaltılık ikillere çevirmek için her 4-ikillik dizge (1) de belirtildiği gibi kendi onaltılık karşılığına çevrilir. Örneğin 32-ikillik dizge

   1010 0001 0000 0011 1111 1110 0010 0011

   "a103fe23" şeklinde ifade edilebilir ve 64-ikillik dizge

   1010 0001 0000 0011 1111 1110 0010 0011
   0011 0010 1110 1111 0011 0000 0001 1010

   "a103fe2332ef301a" şeklinde ifade edilebilir.
3. Bir tamsayı bir kelime bloğu ya da kelime bloğu sıralısı şeklinde ifade edilebilir. İleti uzunluğu 'l'nin kelime bloğu gösterimi bölüm 5.1 de değinilen dolgulama tekniği için gereklidir. 0 ila 2^{32}-1 arasındaki (sınırlar dahil) tamsayılar 32 ikillik kelime bloklarıyla ifade edilirler. Tamsayının düşük anlamlı dört ikili onaltılık kelime bloğunun en sağında ifade edilir. Örneğin, 291 tamsayısı, $256+32+2+1=2^8+2^5+2^1+2^0$ "00000123" onaltılık kelime bloğu şeklinde ifade edilir. Aynı kural 0 ila 2^{64}-1 arasındaki (sınırlar dahil) tamsayılar 64 ikillik kelime blokları için de geçerlidir.

Eğer Z bir tamsayı ve $0 \le Z < 2^{64}$ ise $Z = 2^{32}X + Y$, X öyle ki $0 \le X < 2^{32}$ ve Y öyle ki $0 \le Y < 2^{32}$ X ve Y, x ve y şeklinde 32 ikillik kelime bloklarıyla ifade edilebildiğinden, Z tamsayısı da

(x,y) kelime bloğu sıralısı şeklinde gösterilebilir. Bu özellik SHA-1 ve SHA-256 için kullanılmaktadır.

Eğer Z bir tamsayı ve $0 \leq Z < 2^{128}$ ise $Z = 2^{64}X + Y$, X öyle ki $0 \leq X < 2^{64}$ ve Y öyle ki $0 \leq Y < 2^{64}$ X ve Y, x ve y şeklinde 64 ikillik kelime bloklarıyla ifade edilebildiğinden, Z tamsayısı da (x,y) kelime bloğu sıralısı şeklinde gösterilebilir. Bu özellik SHA-384 ve SHA-512 için kullanılmaktadır.

4. Güvenli hash algoritmalarında *m* ileti blok uzunluğu algoritmaya bağlıdır.
    a. SHA-1 ve SHA-256 için, her ileti bloğu on altı sıralı 32 ikilllik 512 ikille temsil edilmektedir.
    b. SHA-384 ve SHA-512 için, her ileti bloğu on altı sıralı 64-ikillik 1024 ikille temsil edilmektedir.

3.1.3.2 Kelime Blokları Üzerinde Tanımlı İşlemler

Aşağıda tanımlı işlemler dört güvenli hash algoritması için w-ikillik kelime bloklarında uygulanır. Bunlar SHA-1 ve SHA-256 için 32-ikillik kelime blokları (w=32), SHA-384 ve SHA-512 için 64-ikillik kelime blokları (w=64) üzerinde tanımlı işlemlerdir.

1. İkili sistemde mantıksal kelime bloğu işlemleri: $\wedge$, $\vee$, $\oplus$, $\neg$
2. mod 2^w de toplama işlemi
   x+y işlemi öyle ki: x ve y, X $0<=X<2^w$ ve Y $0<=Y<2^w$ tamsayılarının kelime bloğu olarak gösterimi olsun. U ve V pozitif tamsayılar için U mod V U nun V ye bölümünden kalan olarak tanımlansın.
   $Z=(X+Y) \bmod 2^w$ $0<=Z<2^w$ ise Z nin kelime bloğu karşılığı olarak z=x+y dir.
3. Sağa öteleme işlemi $SHR^n(x)$ x , w ikillik kelime bloğu ve n, 0<=n<w ise
   $SHR^n(x)=x>>n$

şeklinde tanımlanır. Bu işlem SHA-256, SHA-384, ve SHA-512 algoritmalarında kullanılır.

4. Sağa döndürme işlemi $ROTR^n(x)$ x , w ikillik kelime bloğu ve n, 0<=n<w ise
   $ROTR^n(x)=(x>>n) \vee (x<<w-n)$ şeklinde tanımlanır.

$ROTR^n(x)$ işlemi x in n kadar sağa döngüsel öteleme yapmasına denktir.
Bu işlem SHA-256, SHA-384, ve SHA-512 algoritmalarında kullanılır.

5. Sola döndürme işlemi $ROTL^n(x)$ x , w ikillik kelime bloğu ve n, 0<=n<w ise
   $ROTL^n(x)=(x<<n) \vee (x>>w-n)$

şeklinde tanımlanır.

$ROTL^n(x)$ işlemi x in n kadar sola döngüsel öteleme yapmasına denktir.

Bu işlem sadece SHA-1 algoritmasında kullanılır.

6. Aşağıda w nin her bir ilişkide sabit olduğu ilişkiler verilmiştir.
   $ROTL^n(x) \approx ROTR^{w-n}(x)$
   $ROTR^n(x) \approx ROTL^{w-n}(x)$

3.1.4 Fonksiyonlar ve Sabitler

Bu kısımda algoritmada kullanılan tanımlı fonksiyonlar ve sabitlere yer verilmiştir.

3.1.4.1 Fonksiyonlar

Bu bölüm her bir algoritmada kullanılan fonksiyonları tanımlar. SHA-256, SHA-384 ve SHA-512 algoritmaları benzer fonksiyonları kullandıkları halde bu fonksiyonların girdi ve çıktı kelime blok sayıları farklı olduğundan tanımlar SHA-256, SHA-384, SHA-512 gibi bölümlere ayrılmıştır. Her algoritma Ch(x,y,z) ve Maj(x,y,z) fonksiyonlarını içermektedir ve XOR ($\oplus$) işlemi ikili sistemde VEYA ($\vee$) işlemi ile değiştirilerek özdeş sonuçlar üretebilir.

3.1.4.1.1 SHA-1 Fonksiyonları

SHA-1 $f_0,f_1,...,f_{79}$ olarak sıralı mantıksal fonksiyonları kullanır. Her f_t fonksiyonu $0 \le t < 79$ 32 ikillik üç kelime bloğu (x,y,z) üzerinde işlem yapar ve 32-ikillik kelime bloğu sonucunu verir. $f_t(x, y, z)$ fonksiyonu aşağıdaki gösterimde tanımlanmıştır.

$$f_t(x, y, z) = \begin{cases} Ch(x, y, z)=(x \wedge y) \oplus (\neg x \wedge z) & 0 \le t \le 19 \\ Parity(x, y, z)=x \oplus y \oplus z & 20 \le t \le 39 \\ Maj(x, y, z)=(x \wedge y) \oplus (x \wedge z) \oplus (y \wedge z) & 40 \le t \le 59 \\ Parity(x, y, z)=x \oplus y \oplus z & 60 \le t \le 79. \end{cases}$$

3.1.4.1.2 SHA-256 Fonksiyonları

SHA-256 her biri 32 ikillik x,y,z ile gösterilen kelime blokları üzerinde tanımlı altı mantıksal fonksiyon kullanır. Her fonksiyon yeni bir 32 ikillik çıktı üretir.

$Ch(x, y, z)=(x \wedge y) \oplus (\neg x \wedge z)$

$Maj(x, y, z)=(x \wedge y) \oplus (x \wedge z) \oplus (y \wedge z)$

$\sum_0^{\{256\}}(x) = ROTR^{2}(x) \oplus ROTR^{13}(x) \oplus ROTR^{22}(x)$

$\sum_1^{\{256\}}(x) = ROTR^{6}(x) \oplus ROTR^{11}(x) \oplus ROTR^{25}(x)$

$\sigma_0^{\{256\}}(x) = ROTR^{7}(x) \oplus ROTR^{18}(x) \oplus SHR^{3}(x)$

$\sigma_0^{\{256\}}(x) = ROTR^{17}(x) \oplus ROTR^{19}(x) \oplus SHR^{10}(x)$

3.1.4.1.3 SHA-384 ve SHA-512 Fonksiyonları

SHA-384 ve SHA-512 her biri 64 ikillik x,y,z ile gösterilen kelime blokları üzerinde tanımlı altı mantıksal fonksiyon kullanır. Her fonksiyon yeni bir 64 ikillik çıktı üretir.

$Ch(x, y, z)=(x \wedge y) \oplus (\neg x \wedge z)$

$Maj(x, y, z)=(x \wedge y) \oplus (x \wedge z) \oplus (y \wedge z)$

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

$\sum_1^{\{512\}}(x) = ROTR^{14}(x) \oplus ROTR^{18}(x) \oplus ROTR^{41}(x)$

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

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

3.1.4.2 Sabitler

Bu kısımda standartta yer alan algoritmalarda kullanılan sabitlere yer verilmiştir.

3.1.4.2.1 SHA-1 Sabitleri

SHA-1 32 ikillik kelime blokları şeklinde seksen adet sabit kullanır; $K_0,K_1,...,K_{79}$.

$$K_t = \begin{cases} \texttt{5a827999} & 0 \le t \le 19 \\ \texttt{6ed9eba1} & 20 \le t \le 39 \\ \texttt{8f1bbcdc} & 40 \le t \le 59 \\ \texttt{ca62c1d6} & 60 \le t \le 79. \end{cases}$$

3.1.4.2.2 SHA-256 Sabitleri

SHA-256 1 32 ikillik kelime blokları şeklinde altmış dört adet sabit kullanır; $K_0^{\{256\}}$, $K_1^{\{256\}}$,..., $K_{63}^{\{256\}}$ Bu kelime blokları ilk 64 asal sayının küp köklerinin kesirli kısımlarının ilk 32 ikilidir. Onaltılık sistemde bu sabit kelime blokları (soldan sağa):

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

3.1.4.2.3 SHA-384 ve SHA-512 Sabitleri

SHA-384 ve SHA-512 32 ikillik kelime blokları şeklinde aynı 80 adet sabitleri kullanır; $K_0^{\{512\}},K_1^{\{512\}},...,K_{63}^{\{512\}}$ Bu kelime blokları ilk 80 asal sayının küp köklerinin kesirli kısımlarının ilk 32 ikilidir. Onaltılık sistemde bu sabit kelime blokları (soldan sağa):

```
428a2f98d728ae22 7137449123ef65cd b5c0fbcfec4d3b2f e9b5dba58189dbbc
3956c25bf348b538 59f111f1b605d019 923f82a4af194f9b ab1c5ed5da6d8118
d807aa98a3030242 12835b0145706fbe 243185be4ee4b28c 550c7dc3d5ffb4e2
72be5d74f27b896f 80deb1fe3b1696b1 9bdc06a725c71235 c19bf174cf692694
e49b69c19ef14ad2 efbe4786384f25e3 0fc19dc68b8cd5b5 240ca1cc77ac9c65
2de92c6f592b0275 4a7484aa6ea6e483 5cb0a9dcbd41fbd4 76f988da831153b5
983e5152ee66dfab a831c66d2db43210 b00327c898fb213f bf597fc7beef0ee4
```

```
c6e00bf33da88fc2 d5a79147930aa725 06ca6351e003826f 142929670a0e6e70
27b70a8546d22ffc 2e1b21385c26c926 4d2c6dfc5ac42aed 53380d139d95b3df
650a73548baf63de 766a0abb3c77b2a8 81c2c92e47edaee6 92722c851482353b
a2bfe8a14cf10364 a81a664bbc423001 c24b8b70d0f89791 c76c51a30654be30
d192e819d6ef5218 d69906245565a910 f40e35855771202a 106aa07032bbd1b8
19a4c116b8d2d0c8 1e376c085141ab53 2748774cdf8eeb99 34b0bcb5e19b48a8
391c0cb3c5c95a63 4ed8aa4ae3418acb 5b9cca4f7763e373 682e6ff3d6b2b8a3
748f82ee5defb2fc 78a5636f43172f60 84c87814a1f0ab72 8cc702081a6439ec
90befffa23631e28 a4506cebde82bde9 bef9a3f7b2c67915 c67178f2e372532b
ca273eceea26619c d186b8c721c0c207 eada7dd6cde0eb1e f57d4f7fee6ed178
06f067aa72176fba 0a637dc5a2c898a6 113f9804bef90dae 1b710b35131c471b
28db77f523047d84 32caab7b40c72493 3c9ebe0a15c9bebc 431d67c49c100d4c
4cc5d4becb3e42b6 597f299cfc657e2a 5fcb6fab3ad6faec 6c44198c4a475817
```

3.1.5 Önişlem

Önişlem hash hesaplaması başlamadan önce yer alır. Bu işlem: M iletisinin dolgulanması, dolgulanmış iletinin ileti bloklarına ayrıştırılması ve $H^{(0)}$ ilk hash değerinin belirlenmesi şeklinde üç aşama içerir.

3.1.5.1 İletinin dolgulanması

M iletisine hash hesaplaması başlamadan önce dolgulama yapılmalıdır. Bu dolgulamanın amacı dolgulanmış iletinin kullanılan algoritmaya göre 512 veya 1024 ikilin katları olmasının sağlanmasıdır.

3.1.5.1.1 SHA-1 ve SHA-256

M iletisinin uzunluğu l ikil olarak farz edilsin. İletinin sonuna "1" ikili ve onu takip eden k adet sıfır ikili eklenmelidir; k öyleki $l+1+k \equiv 448 \bmod 512$ denkleminin en küçük pozitif çözümüdür. Sonra l uzunluğunu ikili sayı sisteminde ifade eden 64 ikillik blok eklenmelidir. Örneğin "abc" iletisinin (8-ikillik) uzunluğu 8x3 =24 tür, iletiye bir ikili eklenecek, sonra 448-(24+1)=423 sıfır ikili ve en sonunda iletinin uzunluğu eklenerek, 512 ikillik dolgulanmış ileti oluşturulacaktır.

423 64

01100001 01100010 01100011 1 00…00 00…011000 .

"**a**" "**b**" "**c**" l=24

Böylece dolgulanmış ileti uzunluğu 512 nin katı şeklinde olacaktır.

3.1.5.1.2 SHA-384 ve SHA-512

M iletisinin uzunluğu l ikil olarak farz edilsin. İletinin sonuna "1" ikili ve onu takip eden k adet sıfır ikili eklenmelidir; k öyleki l+1+k ≡896 mod 1024 denkleminin en küçük pozitif çözümüdür. Sonra l uzunluğunu ikili sayı sisteminde ifade eden 128 ikillik blok eklenmelidir. Örneğin "abc" iletisinin (8-ikillik) uzunluğu 8x3 =24 tür, iletiye bir ikili eklenecek, sonra 896-(24+1)=871 sıfır ikili ve en sonunda iletinin uzunluğu eklenerek, 1024 ikillik dolgulanmış ileti oluşturulacaktır.

871 128

01100001 01100010 01100011 1 00…00 00…011000 .

"**a**" "**b**" "**c**" l=24

Böylece dolgulanmış ileti uzunluğu 1024 ün katı şeklinde olacaktır.

3.1.5.2 Dolgulanmış İletinin Ayrıştırılması

Bir ileti dolgulandığında, hash hesaplamasına başlanmadan önce N adet m-ikillik bloklara ayrılmalıdır.

3.1.5.2.1 SHA-1 ve SHA-256

SHA-1 ve SHA-256 algoritmaları için dolgulanmış ileti $M^{(1)}$, $M^{(2)}$,…, $M^{(N)}$ şeklinde N adet 512-ikillik bloklara ayrıştırılır. 512-ikillik girdi blokları 32-ikillik on altı kelime bloğu olarak ifade edildiğinde ilk 32-ikillik ileti bloğu i $M_0^{(i)}$ şeklinde, bir sonraki $M_1^{(i)}$ şeklinde ve en son blok $M_{15}^{(i)}$ şeklinde gösterilir.

3.1.5.2.2 SHA-384 ve SHA-512

SHA-384 ve SHA-512 algoritmaları için dolgulanmış ileti $M^{(1)}$, $M^{(2)}$,…, $M^{(N)}$ şeklinde N adet 1024-ikillik bloklara ayrıştırılır. 1024-ikillik girdi blokları 64-ikillik on altı kelime bloğu olarak ifade edildiğinde ilk 64-ikillik ileti bloğu i $M_0^{(i)}$ şeklinde, bir sonraki $M_1^{(i)}$ şeklinde ve en son blok $M_{15}^{(i)}$ şeklinde gösterilir.

3.1.5.3 İlk Hash Değeri $H^{(0)}$ ın Belirlenmesi

Hash hesaplamasına başlanmadan önce her bir güvenli hash algoritması için ilk hash değeri $H^{(0)}$ belirlenmelidir. $H^{(0)}$ ın uzunluğu ve kelime bloğu sayısı ileti özeti uzunluğuna bağlıdır.

3.1.5.3.1 SHA-1

SHA-1 için ilk $H^{(0)}$ değeri aşağıdaki beş 32-ikillik kelime bloğundan oluşur, onaltılık sistemde karşılıkları:

$H_0^{(0)}$ = `67452301`
$H_1^{(0)}$ = `efcdab89`
$H_2^{(0)}$ = `98badcfe`
$H_3^{(0)}$ = `10325476`
$H_4^{(0)}$ = `c3d2e1f0`

3.1.5.3.2 SHA-256

SHA-256 için ilk $H^{(0)}$ değeri aşağıdaki sekiz 32-ikillik kelime bloğundan oluşur, onaltılık sistemde karşılıkları:

$H_0^{(0)}$ = `6a09e667`
$H_1^{(0)}$ = `bb67ae85`
$H_2^{(0)}$ = `3c6ef372`
$H_3^{(0)}$ = `a54ff53a`
$H_4^{(0)}$ = `510e527f`

$H_5^{(0)}$ = 9b05688c

$H_6^{(0)}$ = 1f83d9ab

$H_7^{(0)}$ = 5be0cd19

Bu kelime blokları ilk sekiz asal sayının kareköklerinin kesirli kısımlarının ilk otuziki ikilinden elde edilmiştir.

3.1.5.3.3 SHA-384

SHA-384 için ilk $H^{(0)}$ değeri aşağıdaki sekiz 64-ikillik kelime bloğundan oluşur, onaltılık sistemde karşılıkları:

$H_0^{(0)}$= cbbb9d5dc1059ed8

$H_1^{(0)}$= 629a292a367cd507

$H_2^{(0)}$= 9159015a3070dd17

$H_3^{(0)}$= 152fecd8f70e5939

$H_4^{(0)}$= 67332667ffc00b31

$H_5^{(0)}$= 8eb44a8768581511

$H_6^{(0)}$= db0c2e0d64f98fa7

$H_7^{(0)}$= 47b5481dbefa4fa4

Bu kelime blokları dokuzuncu ila on altıncı arasındaki asal sayıların kareköklerinin kesirli kısımlarının ilk altmış dört ikilinden elde edilmiştir.

3.1.5.3.4 SHA-512

SHA-512 için ilk $H^{(0)}$ değeri aşağıdaki sekiz 64-ikillik kelime bloğundan oluşur, onaltılık sistemde karşılıkları:

$H_0^{(0)}$ = 6a09e667f3bcc908

$H_1^{(0)}$ = bb67ae8584caa73b

$H_2^{(0)}$ = 3c6ef372fe94f82b

$H_3^{(0)}$ = a54ff53a5f1d36f1

$H_4^{(0)}$ = 510e527fade682d1

$H_5^{(0)}$ = 9b05688c2b3e6c1f

$H_6^{(0)}$ = 1f83d9abfb41bd6b

$H_7^{(0)}$ = 5be0cd19137e2179.

Bu kelime blokları ilk sekiz asal sayının kareköklerinin kesirli kısımlarının ilk altmış dört ikilinden elde edilmiştir.

3.1.6 Güvenli Hash Algoritmaları

Her bir güvenli hash algoritması için benzer sonuç veren farklı hesaplama yöntemleri bulunabilir. Bu tip farklı yaklaşımlar bu standarda uygun olarak uygulanmalıdır. (Bu dokümanda, algoritmalardan yalnızca SHA-1 ve SHA-256 ya değinilecektir. Diğer algoritmalar için referans dokümandan faydalanılabilir.)

3.1.6.1 SHA-1

SHA-1 l ($0 \leq l < 2^{64}$) uzunluğundaki iletiler için kullanılabilir. Algoritma 1) 32 ikillik seksen kelime bloğundan oluşan bir ileti çizelgesi, 2) her biri 32 ikil uzunluğundaki beş değişken ve 3) beş adet 32 ikillik kelime bloğundan oluşan bir hash değerini kullanır. SHA-1 algoritmasının sonucu 160-ikillik ileti özetidir.

İleti çizelgesine ait kelime blokları $W_0, W_1, \ldots, W_{79}$ olarak etiketlendirilir. Değişken değerler a,b,c,d,e şeklinde ve ilk hash değeri $H^{(0)}$ ı oluşturan hash kelime blokları $H_0^{(i)}$, $H_1^{i)}$,…, $H_4^{(i)}$ şeklinde etiketlendirilir bu değer her kelime bloğu işlendiğinde yeni ara hash değeri $H^{(i)}$ ile değiştirilir ve en son $H^{(N)}$ sonuç hash değerini alır. SHA-1 bunlarla beraber bir adet T geçici kelime bloğu değişkeni kullanır.

3.1.6.1.1 SHA-1 Önişlem

1. M iletisinin dolgulanması
2. Dolgulanmış iletinin $M^{(1)}, M^{(2)}, \ldots, M^{(N)}$ şeklinde N adet ileti bloğuna ayrıştırılması.
3. $H^{(0)}$ ilk hash değerinin belirlenmesi

3.1.6.1.2 SHA-1 Hash Hesaplaması

SHA-1 hash hesaplaması önceki bölümlerde tanımlanan fonksiyon ve sabitleri kullanır. Toplama işlemi (+) mod 2^{32} de gerçekleştirilir.

Önişlem tamamlandığında, her bir ileti bloğu, $M^{(1)}$, $M^{(2)}$,…, $M^{(N)}$, aşağıdaki aşamalardan sırasıyla geçer.

i=1 den N'ye

{

1. $\{W_t\}$ ileti çizelgesini hazırla

$$W_t = \begin{cases} M_t^{(i)} & 0 \le t \le 15 \\ \text{ROTL}^1(W_{t-3} \oplus W_{t-8} \oplus W_{t-14} \oplus W_{t-16}) & 16 \le t \le 79 \end{cases}$$

2. a,b,c,d,e değişkenlerine (i-1)nci hash değerini ata

   a= $H_0^{(i-1)}$

   b= $H_1^{(i-1)}$

   c= $H_2^{(i-1)}$

   d= $H_3^{(i-1)}$

   e= $H_4^{(i-1)}$

3. t=0 dan 79 a

   { $T = \text{ROTL}^5(a) + f_1(b,c,d)+e+K_t+W_t$

   e=d

   d=c

   c= $\text{ROTL}^{30}(b)$

   b=a

   a=T

   }

4. Ara i nci hash değerinin $H^{(i)}$ hesaplanması:

   $H_0^{(i-1)} = a+H_0^{(i-1)}$

   $H_1^{(i-1)} = b+H_1^{(i-1)}$

   $H_2^{(i-1)} = c+H_2^{(i-1)}$

   $H_3^{(i-1)} = d+H_3^{(i-1)}$

   $H_4^{(i-1)} = e+ H_4^{(i-1)}$

}

Tüm aşamaların birinciden dördüncüye N kez tekrarlanması sonucunda ($M^{(N)}$ nin işlenmesinden sonra) elde edilen 160 ikillik M ileti özeti: $H_0^{(N)} \| H_1^{(N)} \| H_2^{(N)} \| H_3^{(N)} \| H_4^{(N)}$ olacaktır.

(Standartta yer alan alternatif uygulama yöntemine bu kısımda yer verilmemiştir.)

3.1.6.2 SHA-256

SHA-256 l ($0 \leq l < 2^{64}$) uzunluğundaki iletiler için kullanılabilir. Algoritma 1) 32 ikillik altmış dört kelime bloğundan oluşan bir ileti çizelgesi, 2) her biri 32 ikil uzunluğundaki sekiz değişken ve 3) sekiz adet 32 ikillik kelime bloğundan oluşan bir hash değerini kullanır. SHA-256 algoritmasının sonucu 256-ikillik ileti özetidir.

İleti çizelgesine ait kelime blokları $W_0, W_1, \ldots, W_{63}$ olarak etiketlendirilir. Değişken değerler a,b,c,d,e,f,g,h şeklinde ve ilk hash değeri $H^{(0)}$ ı oluşturan hash kelime blokları $H_0^{(i)}$, $H_1^{i)}$,…, $H_7^{(i)}$ şeklinde etiketlendirilir. Bu değer her kelime bloğu işlendiğinde yeni ara hash değeri $H^{(i)}$ ile değiştirilir ve en son $H^{(N)}$ sonuç hash değerini alır. SHA-256 bunlarla beraber iki adet T_1 ve T_2 geçici kelime bloğu değişkenlerini kullanır.

3.1.6.2.1 SHA-256 Önişlem

1. M iletisinin dolgulanması
2. Dolgulanmış iletinin $M^{(1)}, M^{(2)}, \ldots, M^{(N)}$ şeklinde N adet ileti bloğuna ayrıştırılması.
3. $H^{(0)}$ ilk hash değerinin belirlenmesi

3.1.6.2.2 SHA-256 Hash Hesaplaması

SHA-256 hash hesaplaması önceki bölümlerde tanımlanan fonksiyon ve sabitleri kullanır. Toplama işlemi (+) mod 2^{32} de gerçekleştirilir.

Önişlem tamamlandığında, her bir ileti bloğu, $M^{(1)}$, $M^{(2)}$,…, $M^{(N)}$, aşağıdaki aşamalardan sırasıyla geçer.

i=1 den N'ye

{

4. $\{W_t\}$ ileti çizelgesini hazırla

$$W_t = \begin{cases} M_t^{(i)} & 0 \le t \le 15 \\ \sigma_1^{\{256\}}(W_{t-2}) + W_{t-7} + \sigma_0^{\{256\}}(W_{t-15}) + W_{t-16}) & 16 \le t \le 63 \end{cases}$$

5. a,b,c,d,e,f,g,h değişkenlerine (i-1) nci hash değerini ata

   $a= H_0^{(i-1)}$

   $b= H_1^{(i-1)}$

   $c= H_2^{(i-1)}$

   $d= H_3^{(i-1)}$

   $e= H_4^{(i-1)}$

   $f= H_5^{(i-1)}$

   $g= H_6^{(i-1)}$

   $h= H_7^{(i-1)}$

6. t=0 dan 63 e

{ $T_1 = h + \sum_1^{\{256\}}(e) + Ch(e,f,g) + K_t^{\{256\}} + W_t$

$T_2 = \sum_0^{\{256\}}(a) + Maj(a,b,c)$

$h=g$

$g=f$

$f=e$

$e=d+ T_1$

$d=c$

$c= b$

$b=a$

$a=T_1+T_2$

}

4. Ara i nci hash değerinin $H^{(i)}$ hesaplanması:

$H_0^{(i-1)} = a+H_0^{(i-1)}$

$H_1^{(i-1)} = b+H_1^{(i-1)}$

$H_2^{(i-1)} = c+H_2^{(i-1)}$

$H_3^{(i-1)} = d+H_3^{(i-1)}$

$H_4^{(i-1)} = e+ H_4^{(i-1)}$

$H_5^{(i-1)} = f+H_5^{(i-1)}$

$H_6^{(i-1)} = g+H_6^{(i-1)}$

$H_7^{(i-1)} = h+ H_7^{(i-1)}$

}

Tüm aşamaların birinciden dördüncüye N kez tekrarlanması sonucunda ($M^{(N)}$ nin işlenmesinden sonra) elde edilen 256 ikillik M ileti özeti:

$H_0^{(N)} || H_1^{(N)} || H_2^{(N)} || H_3^{(N)} || H_4^{(N)} H_5^{(N)} || H_6^{(N)} || H_7^{(N)}$ olacaktır.

3.2 HMAC Anahtarlı - Hash İleti Kimlik Denetim Kodu Standardı

Bu bölümde anahtarlı-hash ileti kimlik denetim koduna ait Amerikan Federal Bilgi İşlem Standartları 198 numaralı yayınına yer verilmiştir.

Bu standart kriptografik hash fonksiyonları kullanan ileti kimlik denetimi için bir düzenek olan anahtarlı-hash ileti kimlik denetimi kodunu HMAC(hash message authentication code) açıklar. HMAC paylaşımlı gizli anahtar birleşimi ile herhangi bir yinelemeli FIPS tarafından onaylı kriptografik hash fonksiyonu ile kullanılabilir. HMAC in kriptografik gücü kullanılan hash fonksiyonunun özelliklerine bağlıdır. Bu standarttaki HMAC belirtimi Internet RFC 2104, HMAC, İleti Kimlik Denetimi için Anahtarlı-Hash, ANSI X9.71 ve Anahtarlı-Hash İleti Kimlik Denetimi Kodlarının bir genellemesidir.

Anahtarlı-Hash İleti Kimlik Denetimi Kodu (HMAC)
Federal Bilgi İşlem Standartları Yayımları (FIPS PUBS)

Standarda ait tanımlar:
1-Standardın Adı: Anahtarlı-Hash İleti Kimlik Denetimi Kodu (HMAC) (FIPS 198)
2-Standart Kategorisi: Bilgisayar Güvenliği Standardı Alt-kategori: Kriptografi
3-Açıklama: Bu standart iletiler için kimlik denetimi gerektiren uygulamalar için bir algoritma belirler. İleti kimlik denetimi, ileti kimlik denetimi kodunun (MAC) inşasıyla elde edilir. Kriptografik hash fonksiyonlu tabanlı MACler HMAC olarak bilinir.

MAC in amacı hiçbir ilave düzeneğe gerek duyulmadan iletinin kaynağının ve bütünlüğünün denetimini sağlamaktır. HMACler fonksiyonel olarak iki farklı parametreye sahiplerdir; ileti ve sadece çağrıyı başlatan kişi ve ilgili alıcı(lar) tarafından bilinen gizli bir anahtar. Anahtarlı hash fonksiyonlarının diğer uygulamaları meydan okuma-yanıt işlevli kimlik belirleme protokolleridir.

Bir HMAC fonksiyonu, iletiyi gönderen tarafından gizli anahtar ve ileti içeriğinin yoğunlaştırılmış biçimi olan değerin (MAC) üretilmesi için kullanılır. MAC genelde alıcıya ileti ile birlikte gönderilir. Alıcı aynı anahtarı ve HMAC fonksiyonunu kullanarak alınan iletiye ait MAC i hesaplar ve sonuçla alınan MAC i karşılaştırır. Eğer iki değer eşleşiyorsa ileti doğru olarak iletilmiş ve alıcı gönderenin aynı gizli anahtarı paylaşan grubun üyesi olduğundan emin olmuş olur.

4-7 sayılı başlıkların burada belirtilmesine gerek duyulmamıştır.

8.Uygulamalar: Bu standardın uygulandığı kriptografik modüller FIPS 140-1 e uymalıdır. Bu standartta açıklanan kimlik denetimi düzeneği yazılım, bellenim, donanım veya bunların birleşimi şeklinde uygulanabilir. NIST kriptografik modül onay programını uygulamaların HMAC standardına uygunluğunu test için geliştirmiştir.

Büro HMAC uygulamalarında kullanılan anahtarların diğer sebeplerle kullanılmamasını tavsiye etmektedir.

9.Diğer Onaylanmış Güvenlik Fonksiyonları: Bu standarda uyumlu HMAC uygulamaları, Federal devletin duyarlı bilgilerini korumak için onaylanmış kriptografik algoritmaları, kriptografik anahtar üretim algoritmalarını ve anahtar yönetimi tekniklerini de sağlamalıdır. Onaylanmış kriptografik algoritmalar ve teknikler aşağıdakilerin her birini içermektedir:

a. FIPS'te belirtilmiş, veya
b. Bir FIPS'te benimsenmiş ve FIPS'te bir dokümana atıfta bulunulmuş veya bir ekte belirtilmiş,
c. FIPS 140-2 için onaylanmış güvenlik fonksiyonları listesinde belirtilmiş olmalıdır.

3.2.1. Standart Hakkında Ön Bilgi

Açık işlem ve iletişim dünyasında güvensiz bir ortamda iletilen ya da depolanan bilginin bütünlüğün kontrolünü sağlamak için bir çözüm bulmak birincil gerekliliktir. Bu tip bir bütünlüğü gizli anahtar esaslı kontrol eden düzenekler genel olarak ileti denetim kodları (MACler) olarak adlandırılırlar. Tipik olarak ileti denetim kodları iki taraf arasında paylaşılan gizli anahtar sayesinde iletilmiş bilginin (kimlik)denetimini düzenlemekte kullanılırlar. Bu standart gizli anahtarla birleşmiş bir kriptografik hash fonksiyonunu kullanan MAC yapısını tanımlamaktadır. Bu düzenek HMAC olarak adlandırılır ve HMAC genellemeleri [1.1] ve [1.3] te belirtilmiştir.

HMAC onaylanmış bir kriptografik hash fonksiyonu birleşiminde kullanılır. HMAC, MAClerin hesaplanması ve doğrulanması için gizli bir anahtar kullanır. HMAC in inşasındaki asıl hedefler:

Var olan hash fonksiyonlarının değiştirilmeden kullanılması ki; hash fonksiyonları yazılımda iyi performans sağlamakta ve kodlara yaygın olarak ve ücretsiz erişilebilmektedir.

- Hash fonksiyonunun orijinal performansını dikkate değer bir düşüşe mahal vermeden korumak.
- Anahtarların basitçe kullanılıp idaresi.
- Geçerli varsayımlara dayanan hash fonksiyonu tabanlı kimlik denetimi düzeneğinin gücünün açık kriptografik analizinin yapılabilmesi
- İleride daha hızlı ve güvenli hash fonksiyonları elde edilmesi durumunda kullanılan hash fonksiyonlarının kolayca değiştirilebilmesine imkan tanınması.

3.2.2. Terminoloji ve Kısaltmalar

Bu kısımda standartta kullanılan terim ve kısaltmalara yer verilmiştir.

3.2.2.1 Terminoloji

Aşağıda bu standartta kullanılan tanımlar yer almaktadır:

Onaylanmış: FIPS tarafından onaylanmış ya da NIST tarafından tavsiye edilmiş.
1)FIPS ya da NIST tavsiye mektubunda belirtilen veya 2) FIPS ya da NIST tarafından benimsenen ya da referans gösterilen bir algoritma veya teknik.

Kriptografik anahtar (anahtar): Bir kriptografik algoritmanın birleşiminde kullanılan ve o algoritmanın özel işlemini belirleyen parametre.

Hash fonksiyonu: Herhangi bir uzunluktaki (önceden belirlenmiş azami uzunluğa kadar sınırlanan) metine karşılık gelen sabit uzunluktaki metni oluşturan onaylanmış matematiksel fonksiyon. Uzun metin ya da iletiler için hash değeri ya da ileti özeti denilen sağlama değerini oluşturmada kullanılabilir.

Anahtarlı-Hash İleti Denetim Kodu (HMAC): Bir hash fonksiyonunun kriptografik bir anahtarla birleşimini kullanan ileti kimlik denetim kodu.

İleti Denetim Kodu (MAC): Verinin ileti denetim algoritmasından geçirilmesiyle elde edilen kriptografik sağlama. Bu standartta, HMAC'ın uygulanmasında elde edilen sonuç MAC olarak adlandırılırken, ileti denetim kodu HMAC olarak adlandırılmaktadır.

Gizli anahtar: Bir ya da daha fazla varlık ile benzersiz ilişkilendirilmiş bir kriptografik anahtardır. Bu kavramda "gizli" teriminin kullanılması bir gizlilik tasnifini belirtmez; sadece anahtarın açığa çıkmasına veya değiştirilmesine karşı koruma gerektiğini belirtir.

3.2.2.2 Kısaltmalar

Aşağıda bu standartta yer alan kısaltmalar yer almaktadır:

FIPS	Federal Bilgi İşlem Standartları (ABD)
FIPS PUB	FIPS yayınları
HMAC	Anahtarlı-Hash İleti Denetim Kodu
MAC	İleti Denetim Kodu
NIST	Ulusal Standartlar ve Teknoloji Enstitüsü (ABD)

3.2.2.3 Hmac Parametreleri ve Semboller

HMAC aşağıdaki parametreleri kullanır:

B	Onaylanmış hash fonksiyonunun girdi blok uzunluğu (byte)
H	Onaylanmış hash fonksiyonu
ipad	İç ek; B defa tekrarlanmış x'36' byte karşılığı
K	Çağrıyı başlatan ve ilgili alıcı arasında paylaşılan gizli anahtar.
K_0	B byte-anahtar oluşturmak için gerekli işlemi gördükten sonraki anahtar K.
L	Onaylanmış hash fonksiyonu çıktısı blok uzunluğu (byte)
opad	Dış ek; B defa tekrarlanmış x'5c' byte karşılığı
t	MAC e ait byte sayısı
text	HMAC'in hesaplandığı veri; text ekli anahtarı içermez. Text in uzunluğu n bittir. $0<=n<2^B-8B$

x 'N' onaltılık gösterim 'N' içinde gösterilen her sembol 4 ikili temsil eder.

||	birbirine bağlama
(+)	XOR işlemi

3.2.3 Kriptografik Anahtarlar

Anahtar (K) uzunluğu hash fonksiyonunun çıktı uzunluğu olan L'nin yarısı L/2 ye eşit ya da küçük olmalıdır. L bayttan büyük olan anahtarların fonksiyonun gücünü önemli bir şekilde artırmadığı bilinmelidir. B-bayttan uzun anahtar kullanan uygulamalarda öncelikle anahtar H

kullanılarak hashlenir ve sonra elde edilen L-bayt dizgi K, HMAC anahtarı olarak kullanılır. Anahtarlar onaylanmış anahtar üretim metotları ile rasgele üretilmeli ve periyodik olarak değiştirilmelidir. Anahtarın, güvenliği sağlanacak verinin değerine uygun olacak şekilde korunması gerektiği göz önünde tutulmalıdır.

3.2.4 Kısaltılmış Çıktı

Macler ile ilgili yaygın bir uygulama çıktının kısaltılmasıdır (örneğin, kullanılan MAC uzunluğunun, L MAC çıktı uzunluğundan kısa olması). Bu standarttaki uygulamalarda HMAC çıktısının kısaltılmasına elverişlidir. Kısaltılmış HMAC kullanıldığında, HMAC hesaplamasının en sol t baytı MAC olarak kullanılacaktır. Çıktı uzunluğu, t, dört bayttan kısa olmamalıdır (4<=t<=L). Bununla birlikte uygulama veya protokol sayısız denemenin uygulanamazlığını sağlamadığı sürece t en az L/2 bayt olmalıdır (L/2<=t<=L) . Örneğin. Bir kısa-bant genişliğindeki kanal 4 bayt MAC için sayısız denemeden korur ya da bir protokol sadece az sayıda hatalı MAC denemelerine izin verebilir.

3.2.5 HMAC Özellikleri

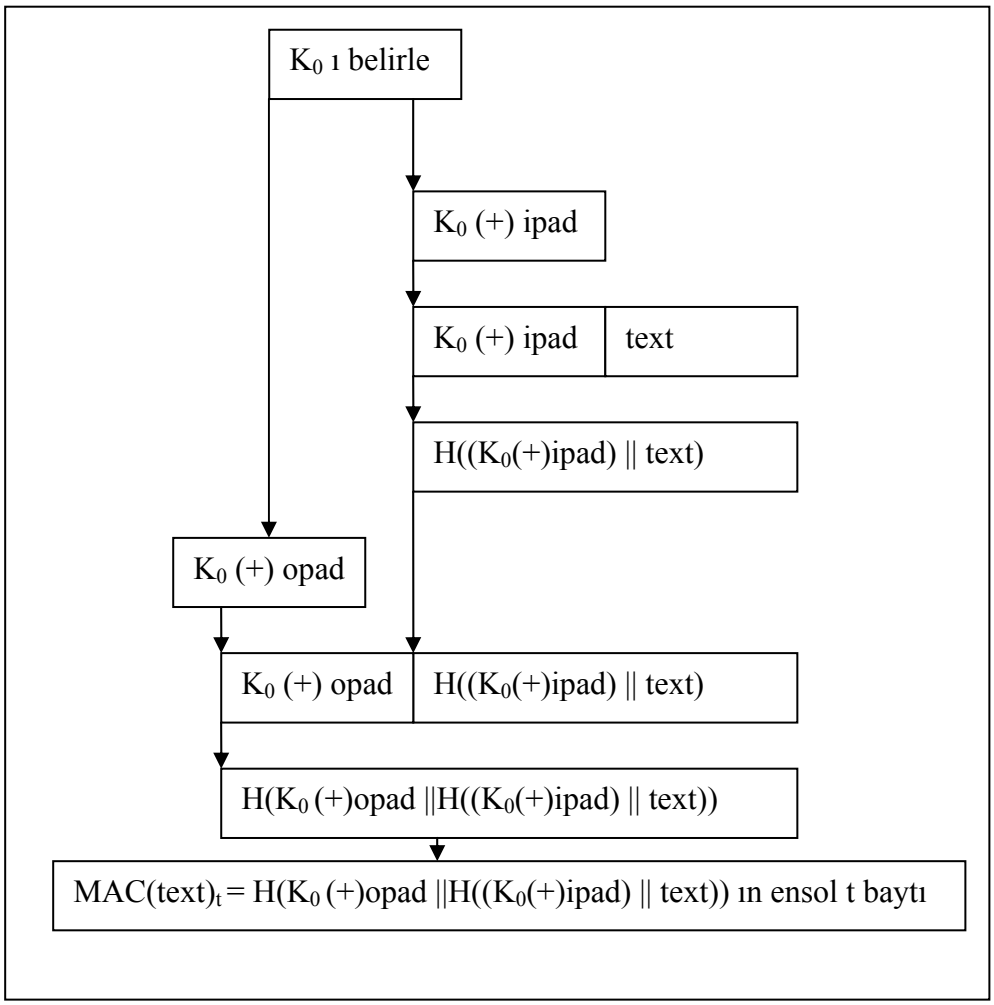
'text' verisinden HMAC fonksiyonunu kullanarak MAC hesaplamak için aşağıdaki işlem gerçekleştirilir: $\mathbf{MAC(text)_t = HMAC(K,text)_t = H((K0\ (+)\ opad)||\ H((K_0\ (+)\ ipad)||\ text))_t}$

Çizelge 3.2, Şekil 1 de tarif edilen HMAC algoritmasındaki işlemleri adım adım örneklemektedir.

Çizelge 3.2 HMAC Algoritması

ADIMLAR	ADIM ADIM TARİF
1. Adım	Eğer K nın uzunluğu B'ye eşitse (K=B) K_0 ı K ya eşitle (K_0=K) 4.adıma git
2. Adım	Eğer K'nın uzunluğu B'den büyükse (K>B) L bayt dizge eldesi için K yı hash işlemine tabi tut, sonra B bayt uzunluğunda K_0 oluşturmak için (B-L) adet sıfırı ekle (örn. K_0=H(K)||00…00). 4. adıma git.
3. Adım	Eğer K'nın uzunluğu B'den küçükse (K<B) B bayt uzunluğunda K_0 oluşturmak için K nın sonuna sıfır ekle (örn. Eğer K 20 bayt uzunluğunda ve B=64 ise K ya 44 bayt sıfır 0x00 eklenir.

4. Adım	K_0 ı ipadle B bayt uzunluğunda dizge oluşturan XOR işlemine tabi tut: K_0(+)ipad
5. Adım	'text' verisini 4.adımda elde edilen dizgeye ekle (K_0(+)ipad) || text
6. Adım	5. adımda elde edilen dizgeyi H ye tabi tut : H((K_0(+)ipad) || text)
7. Adım	K_0 ı opad ile XOR işlemine tabi tut K_0 (+)opad
8. Adım	6. ve 7. adımların sonuçlarını birleştir: K_0 (+)opad ||H((K_0(+)ipad) || text)
9. Adım	8. Adımın sonucunu H işlemine tabi tut : H(K_0 (+)opad ||H((K_0(+)ipad) || text))
10. Adım	9. Adımın sonucunun en soldan t baytını MAC olarak seç



Şekil 3.1 : HMAC inşası

3.2.6 Uygulama Notu

HMAC algoritması herhangi bir onaylanmış kriptografik hash fonksiyonu H için belirlenmiştir. Küçük değişikliklerle, bir HMAC uygulaması H hash fonksiyonuyla H' hash fonksiyonunun kolayca değiştirebilir.

Kavramsal olarak, B-baytlık bloklarda ((K0 (+) ipad)ve(K0 (+) opad)) sıkıştırma fonksiyonunun ara sonuçları K anahtarı üretilirken ya da kullanılmadan önceden bir kez hesaplanabilir. Bu ara sonuçlar depolanıp aynı anahtarla ileti denetimi her gerektiği zaman H için kullanılabilir. K anahtarını kullanan her bir denetimden geçmiş ileti için, bu yöntem H hash fonksiyonunu iki B-bayt uzunluğundaki bloklarda saklar. Bu saklama kısa veri dizgelerinde önemli olabilir. Bu saklanan ara değerler gizli anahtarlar gibi muamele görmeli ve korunmalıdır.

3.2.7 Mac Algoritmalarının Sınırları

Bir MAC in başarılı gerçeklenmesi ilgili iletinin güvenilir olduğunu tamamen garanti etmez: anahtarı bilmeyen bir kaynak anlamlı görünebilen bir MAC üretebilir ve bu da doğrulama işlemini geçebilir. Örneğin, herhangi bir metnin, herhangi bir anlamlı MAC inin t bitinin başarılı şekilde doğrulamayı geçmesi $(½)^{t}$ dir. Bu sınırlama her MAC algoritmasının doğasında vardır.

Eğer uygulama güvensiz iletinin tekrar tekrar farklı anlamlı MAClerine izin veriyorsa bu sınırlama büyür. Her ayrık deneme sadece $(½)^{t}$ kadar başarı şansına sahiptir; ancak, tekrarlı denemelerde olasılık artar ve kaçınılmaz olarak bir MAC doğrulanır. Buna benzer, eğer bir uygulama anlamlı bir MACe karşılık güvensiz farklı iletilere de izin veriyorsa, olasılık artar ve sonuçta MAC doğrulanabilir.

Bu sebepten, genelde, eğer MAC kısaltılmışsa, t uzunluğu olabildiğince büyük tutulmalı ve en az L çıkış blok uzunluğunun yarı uzunluğunda olmalıdır. Yukarıda belirtilen her iki tekrar denemeli başarılı şekilde kısıtlandığı takdirde asgari t uzunluğu 32 bit olabilir. Örneğin uygulama veya uygulamayı kontrol eden protokol tüm düz iletileri ve MACleri izleyip sıkça

çok fazla başarısız deneme içerenleri reddedebilir. Bir başka örnek de her iki tip denemeyi imkansız kılan düşük bant genişliğindeki kanallardır. Her iki durumda, öngörülen azami başarısız deneme sayısı verinin hassaslığı ve MAC algoritmasında kullanılan t uzunluğu ile ilgili riskler dikkate alınarak önceden belirlenmelidir.

3.3 RC6 Blok Şifreleme Algoritması

Bu algoritma Rivest, Robshaw ve diğ. (1998) tarafından yeni gelişmiş şifreleme standardı (AES) adayı olarak Amerikan Ulusal Standartlar ve Teknolojiler Enstitüsü'ne sunulmuştur. Daha önceden yine aday olarak sunulan RC5 algoritmasının performans ve güvenlik bakımından geliştirilmesi ile oluşturulan bu algoritma ile ilgili detaylara bu kısımda yer verilecektir.

RC6 algoritması genel olarak RC6 –w/r/b şeklinde parametreli olarak gösterilmektedir. Algoritmada kullanılan kelime bloğu w ikili uzunluğunda, şifrelemede kullanılacak pozitif, döngü sayısı r ve yine şifrelemede kullanılan anahtar uzunluğu b dir. AES te hedeflenen değerler doğrultusunda genel olarak RC6 algoritmasında bu parametreler w=32 , r=20 değerlerini almaktadır.

RC6 algoritmasının parametrelere bağlı tüm türevleri dört w-ikil kelime bloğu üzerinde tanımlı 6 işleme tabidirler. lgw w üzerindeki iki tabanlı logaritmik fonksiyonu ifade etmektedir.

a + b mod 2w de tamsayı toplama işlemi
a - b mod 2w de tamsayı çıkarma işlemi
a ⊕ b w-ikil kelime bloklarında tanımlı XOR işlemi
a x b mod 2w de tamsayı çarpma işlemi
a<<<b a nın döngüsel olarak b ikilinin en az anlamlı lgw ikili kadar sola ötelenmesi işlemi
a>>>b a nın döngüsel olarak b ikilinin en az anlamlı lgw ikili kadar sağa ötelenmesi işlemi

Algoritma anahtar çizelgesi, şifreleme ve çözme bölümlerinden oluşmaktadır. Aşağıda bu bölümler ile ilgili açıklamalar ve bölüm sonunda şifreleme algoritmasına ait bir kesite şekil-1 de yer verilmiştir.

3.3.1 Anahtar Çizelgesi

RC6 algoritmasının anahtar çizelgesi RC5 algoritmasında kullanılan anahtar çizelgesi ile aynıdır. Tek fark şifreleme ve çözme safhalarında anahtar çizelgesinden daha fazla kelime bloğu elde edilmektedir.

Kullanıcıdan elde edilen b sekiz-ikillik anahtar gerekli sıfır ikilleri dolgulanarak kelime bloğu yağısı elde edilir. Daha sonra bu anahtar blokları w-ikillik c dizisine L[0],...,L[c-1] şeklinde yüklenir. Döngülere katılacak w-ikillik kelime blokları şeklindeki 2r+4 anahtar, S[0,..., 2r+3] şeklinde S dizisinde saklanır.

Algoritmada kullanılan sihirli sabitler P_{32}=B7E15163 ve Q_{32}=9E3779B9 dur. P_{32} değeri e, doğal logaritmik fonksiyonunu ifade etmek üzere e-2 değerinin ikili açılımından, Q_{32} değeri ise Ø altın oranı ifade etmek üzere Ø –1 değerinden elde edilmiştir.

RC6-w/r/b için anahtar çizelgesi:

Girdi: c kelime bloklarına yüklenmiş kullanıcıdan elde edilen b sekiz-ikilllik anahtar
L[0,..., c -1]dizisi
r döngü sayısı
Çıktı: w-ikillik döngü anahtarları S[0,..., 2r + 3]
İşlem: S[0] = Pw
i = 1 den 2r + 3 e
S[i] = S[i-1] + Qw
A = B = i = j = 0
v = 3x max { c, 2r + 4}
s = 1 den v ye
{A = S[i] = (S[i] + A + B)<<<3
B = L[j] = (L[j] + A + B)<<<(A + B)
i = (i + 1)mod (2r + 4)
j = (j + 1)mod c
}

3.3.2. Şifreleme ve Çözme

RC6 algoritması şifrelemede girdi olarak düz metni taşıyan ve sonuçta şifreli metni depolayan dört adet w-ikillik A,B,C,D yazmaçlarını kullanır.

RC6-w/r/b ile şifreleme:

Girdi: Düz metin w-ikillik dört adet (A,B,C,D) girdi yazmaçlarına yüklenir.
Döngü sayısı r
w-ikillik döngü anahtarları S[0,..., 2r + 3]
Çıktı: Şifreli metin A,B,C,D de depolanır.
İşlem: B = B + S[0]
D = D + S[1]
i = 1 den r ye
{
t = (B x(2B + 1))<<<lg w
u = (D x(2D + 1))<<<lg w
A = ((A ⊕ t)<<<u) + S[2i]
C = ((C ⊕u)<<<t) + S[2i+ 1]
(A,B,C,D) = (B,C,D,A)
}
A = A + S[2r + 2]
C = C + S[2r + 3]

RC6-w/r/b ile çözme:

Girdi: Şifreli metin w-ikillik dört adet (A,B,C,D) girdi yazmaçlarına yüklenir.
Döngü sayısı r
w-ikillik döngü anahtarları S[0,..., 2r + 3]
Çıktı: Düz metin A,B,C,D de depolanır.

İşlem: C = C - S[2r + 3]
A = A - S[2r + 2]

i = r den 1 e

{

(A,B,C,D) = (D,A,B,C)

u = (D x (2D + 1))<<<lg w

t = (B x (2B + 1))<<<lg w

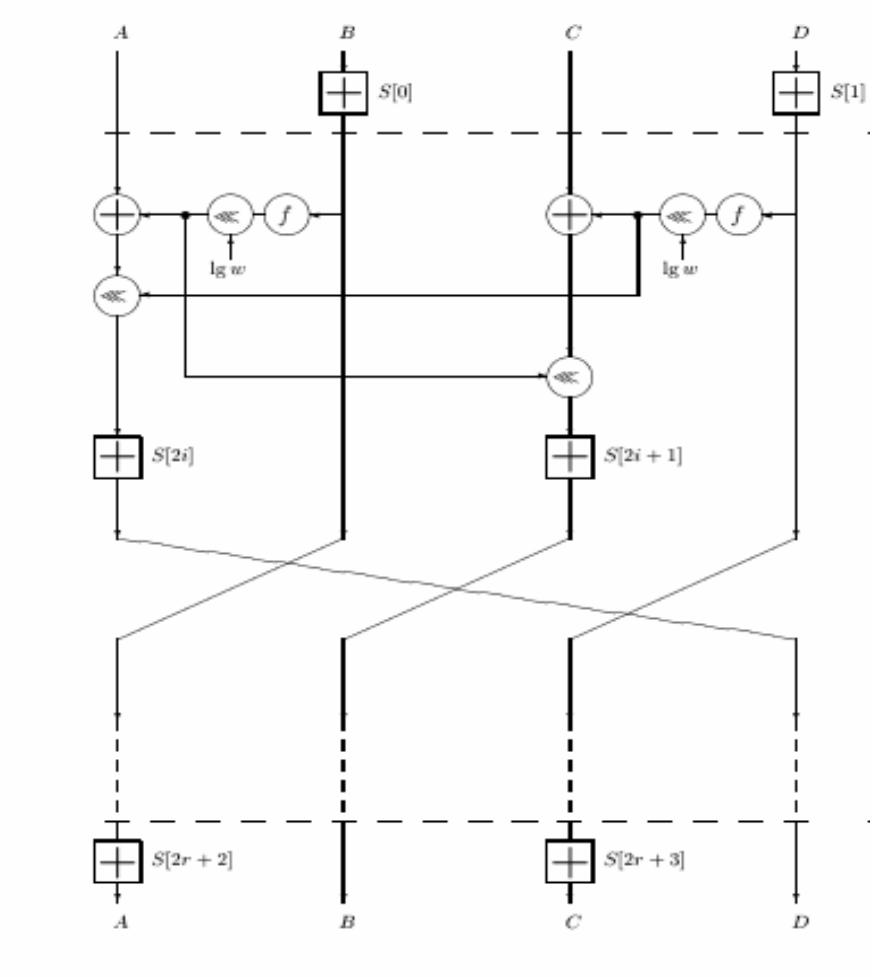
C = ((C - S[2i + 1])>>>t) ⊕u

A = ((A - S[2i])>>>u) ⊕ t

}

D = D - S[1]

B = B - S[0]



Şekil 3.2 RC6 algoritmasının döngü kesiti (Rivest ve diğ, 1998)

DÖRDÜNCÜ BÖLÜM
UYGULAMA

4. UYGULAMA

Bir önceki bölümde kriptografik haberleşme protokolü detaylandırılmış olup, bu bölümde ise sunulan protokolün .NET ortamında ve TCP/IP protokolü üzerine inşa edilen uygulamaya, protokolün temelini oluşturan kriptografik kütüphaneye ve protokolden bağımsız olarak uygulama safhasında gereksinim duyulan güvenlik çözümlerine yer verilecektir.

4.1 Uygulama İçin Ortam ve Programlama Dili Seçimi

Uygulamayı temel alan kriptografik kütüphane kodların tekrar kullanılabilirlik, platform bağımsızlığı gereksinimlerinden dolayı programlama dili olarak nesne tabanlı bir dil seçimine gidilmiştir. Böylece programlama dili olarak C# ve Java dilleri ön plana çıkmış olup kriptografik kütüphaneye ileride eklenecek yeni güvenlik şemaları ve algoritmik çözümlerin yazılmış olan dildeki bağımlılığını da kaldırmak için kütüphanenin birçok dili bünyesinde bulunduran ve farklı dil kodları arasında geçiş ve uyumunu sağlayan .NET ortamında yazılması uygun bulunmuştur.

Protokol benzetiminin konsol uygulaması yerine bir görsel uygulama olarak gerçeklenmesi, benzetimin günlük uygulamalara yakınlığını ve gerçeklenebilir işlem paletinin genişlemesini sağlamıştır.

Benzetim .NET ortamının özelliklerinden azami derecede faydalanılması ve TCP/IP protokolünü temel haberleşme protokolü kabul ettiğinden dolayı Microsoft Windows XP işletim sistemi üzerinde gerçeklenmiştir.

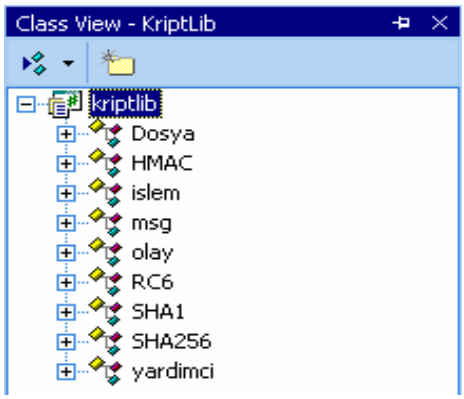
4.2. Kriptografik Kütüphane

Protokol ileti gizliliği, veri bütünlüğü ve kimlik doğrulama temelleri üzerine kurulduğundan hazırlanacak kütüphanede bu işlevleri yerine getirebilecek yöntemlere yer verilmiştir.

Kütüphanede dönüşüm, matematiksel işlemler gibi temel fonksiyonlar ileriye dönük kullanılmaya ve genişletilmeye uygun olarak modüller ve nesneler halinde kodlanmıştır. Nesne tabanlı dillerin çok şekillilik ve aşırı-yükleme özellikleri kullanılarak kodlanan metotların farklı parametre kullanımlarına cevap vermeleri sağlanmıştır. Kriptografik algoritmaların yapılarındaki her aşama modüler olarak kodlanmış bu algoritmalarda ileride yapılabilecek değişiklik ya da geliştirme çalışmaların kütüphaneye yansıtılmasına kolaylık sağlanmıştır.

Kütüphanede yer alan kriptografik algoritmalar kullanılan standartlara ve standart adayı olarak kabul edilmiş yayınlara uygun olarak kodlanmıştır. (FIPS 180-2, FIPS 198) Tez sonundaki ekler kısmında bu algoritmalara ayrıntılı olarak yer verilmiştir.

Kütüphanede yer alan işlevsel ve güvenlik algoritmalarına ait sınıflar Şekil 4.1 de verilmiştir.

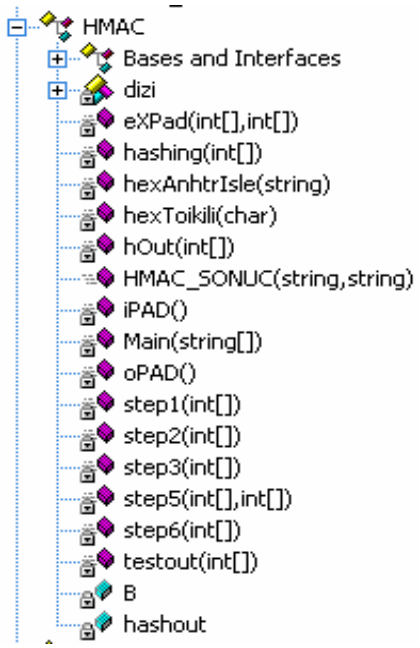


Şekil 4. 1: Kriptografik Kütüphane'ye ait sınıflar

Bu sınıflar;

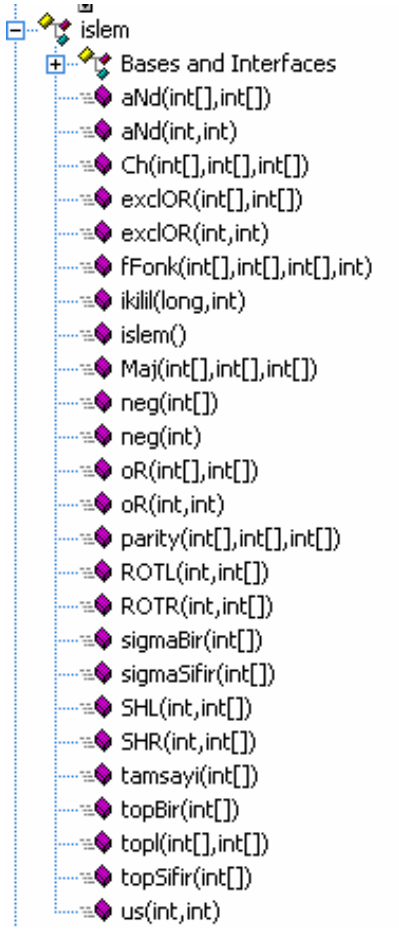
- Dosya Sınıfı: Algoritmaların kullanımında dosya yapısındaki girdiler üzerinde okuma, belleğe aktarma ve hata denetleme gibi metotları barındıran bir sınıftır.

- HMAC Sınıfı: Anahtarlı-hash ileti kimlik denetim kodu algoritmasını gerçekleyen sınıftır. Metin dolgulama, hashleme, hash anahtarı işleme, algoritmaya ait 6 basamak, sonuç ve test metotlarını barındırır.



Şekil 4.2: HMAC sınıfı

- İşlem Sınıfı: Kütüphanenin matematiksel ve işlevsel temelini oluşturan, algoritmalarda kullanılan işlemleri destekleyen sınıftır. İkili, ondalık ve tamsayı dönüşümlerini, matematiksel üs ve toplama işlemlerini, algoritmalarda ortak kullanılan bool mantıksal VE, VEYA, XOR, tersleyen işlemlerini, kaydırma, döndürme işlemlerini, eşlik ve algoritmalara özel metotları içermektedir.

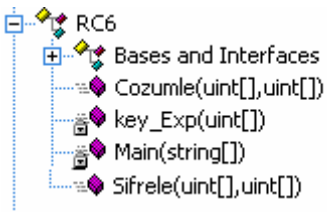


Şekil 4.3: İşlem Sınıfı

- Msg Sınıfı: Bu sınıf metin veya veri blok yapısını oluşturur. Blok yapısındaki ikili sayısı ve ileti yapısındaki blok sayısına göre değişken bir nesne tanımlar.

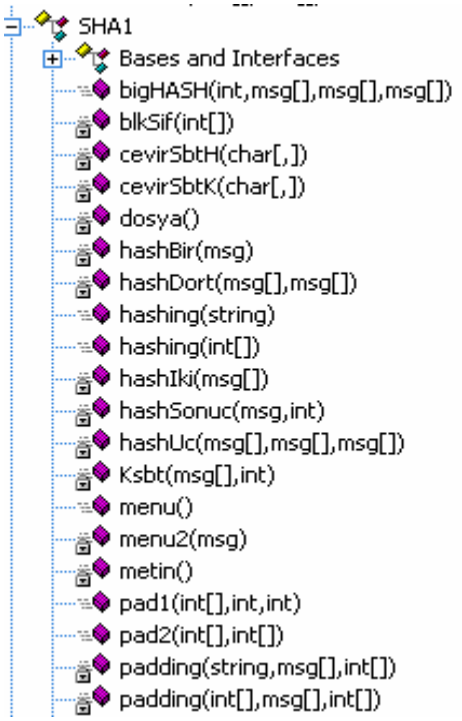
- Olay sınıfı: Uygulamanın işleyişinde ihtiyaç duyulabilecek durum, bekleme, yenileme, çıkış vb. metotları içermektedir.

- RC6 sınıfı: Şifreleme algoritmasına ait şifreleme, çözme ve anahtar üretimi metotlarını içermektedir.



Şekil 4.4: RC6 Sınıfı

- SHA1 Sınıfı: Kütüphanede yer alan güvenli hash algoritmalarından SHA-1'e ait olan bu sınıf, hashleme sabitleri ve basamaklarını, çevirme ve dolgulama gibi metotları içermektedir.



Şekil 4.5: SHA1 Sınıfı

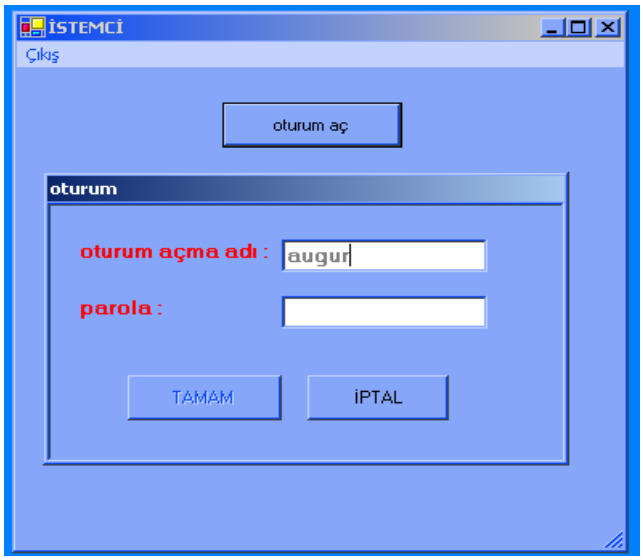
- SHA256 Sınıfı: Kütüphanede yer alan güvenli hash algoritmalarından SHA-256' ya ait olan bu sınıf, SHA1 sınıfına ek olarak bu algoritmaya özel metotları içermektedir.

4.3. İstemci Modülü

İki farklı modülden oluşan uygulamanın ilki uzaktan erişim isteğinde bulunan ve protokolün başlamasını tetikleyen istemci arayüzüdür. Uygulamanın bu kısmı erişime yetki sağlayacağından sistem güvenliğinden başka uygulama için artı güvenlik modülleri hazırlanmış ve uygulamaya eklenmiştir.

Bunlardan güvenlik için en önde gelmesi gereken, uygulamayı kullanma yetkisi sağlayan oturum açma adı ve parola sorgu ekranıdır (Şekil 4.6). Bilindiği gibi makine üzerinde yetki kontrolü kullanıcıyı tanımlayan bir kullanıcı kimliği ve sadece kullanıcının bildiği varsayılan,

sistemin yetki isteyen kişinin doğru kişi olup olmadığını belirlemesine yarayan gizli kelime ya da cümle ile sağlanmaktadır. Sistem edindiği (kullanıcı adı, parola) veri çiftini bünyesinde tuttuğu kayıt ile karşılaştırır ve eşleşme halinde yapılacak işlemi onaylar.

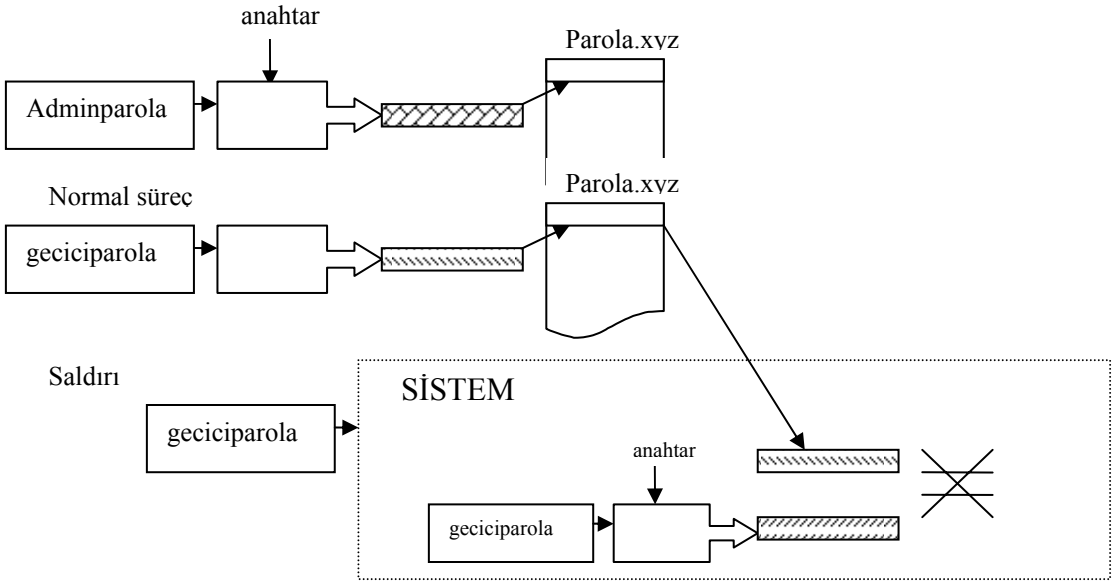


Şekil 4.6: Oturum açma ekranı

Parolalar sistem üzerinde genelde açık, şifreli ya da tek yönlü fonksiyonlardan geçirilmiş olarak dosyalarda tutulurlar. Parolaların açık metin halinde sadece okuma ve yazma korumalı olarak sistem erişim haklarıyla korunması ve alınacak arşiv yedeklerinin de açık metin üzerinden olması güvenliği tehlikeye atmaktadır. Bunun önüne geçebilmek için her bir parola tek yönlü fonksiyonlardan geçirilerek dosyada asıl parolanın yerine yazılır. Sistem kontrolü kullanıcı tarafından girilen parolanın aynı tek yönlü fonksiyondan geçirilmesi ve dosyadaki karşılığıyla eşleştirilmesi esasına dayanır. Parolalar gizlenirken geri dönüşümlü şifreleme algoritmaları yerine tek yönlü fonksiyonların kullanımı şifreleme algoritmaları üzerindeki ihraç kısıtlamaları ve anahtarlama malzemesinin gerekmesinin önüne geçilmesi sebebindendir (Menezes ve diğ, 1997).

Gerçeklenen uygulamada, sistem izin kontrolleri ile erişim hakları kısıtlanan parola dosyasında parolalar, kütüphanede yer alan HMAC modülünden geçirilerek saklanmaktadır. Sadece bir hash algoritması kullanmak yerine uygulamaya gömülen bir anahtar ile anahtarlı HMAC kullanılmasında amaçlanan, sistem erişim haklarının ihlali halinde bile uygulama güvenliğinin dayanıklılığını artırmaktır.

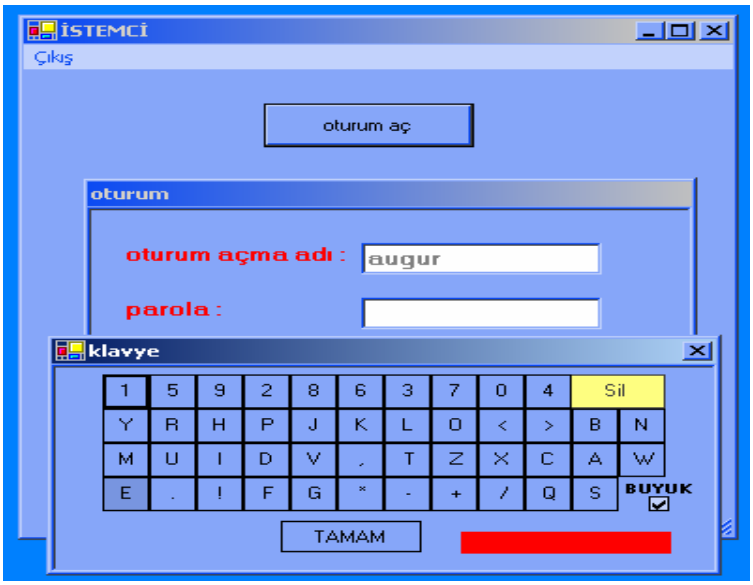
Bir saldırganın parola dosyası üzerindeki sistem izin kontrollerini atlatması ile elde ettiği haklarla yeni bir kullanıcı parolasını oluşturması ve bu dosyaya hash karşılığını kayıt etmesi olasıdır. Ama anahtarlı HMAC kullanıldığı için saldırgan tarafından kaydedilen parola anahtar bilgisi gizli kaldığından geçersiz olacaktır. Şekil 4.7 de saldırı ve uygulama cevabı gösterilmiştir.



Şekil 4.7: Parola saldırısı ve uygulama cevap şeması

Parola saldırılarında yeni bir yetkili parolaya ulaşmak yerine var olan yetkili parolayı ele geçirmeye çalışmak da kullanılan yöntemlerin başında yer alır. Sisteme yerleştirilen ve arka planda çalışan keylogger programları klavye kullanılarak girilen parolalarda basılan tuşları günlüğe atarak parola elde etmeyi amaçlarlar. Bu programlara karşı geliştirilen ekran klavyeleri kullanıcının klavye kullanmadan parolasını güvenli bir şekilde girmesini sağlar. Şekil 4.8 uygulama için oluşturulmuş ekran klavyesini göstermektedir.

Klavye, fare tıklamaları ile koordinat tabanlı parola eldesi olasılığına karşı klavye üzerindeki harfleri her seferinde dinamik olarak dağıtacak şekilde tasarlanmıştır. Bu ekran klavyelerinin en büyük dezavantajları kullanıcı tarafında kullanım zorluğu oluşturmalarıdır. Bu sebepten güvenlik seviyesini yakalamak için yeterli uzunlukta belirlenmesi gereken parolalar, kullanıcılar tarafından kısaltılma yoluna gidilmektedir. Yönetici tarafından parola poliçeleri ve kullanıcılara verilecek eğitim ile bu gibi sorunların üstesinden gelinecektir.



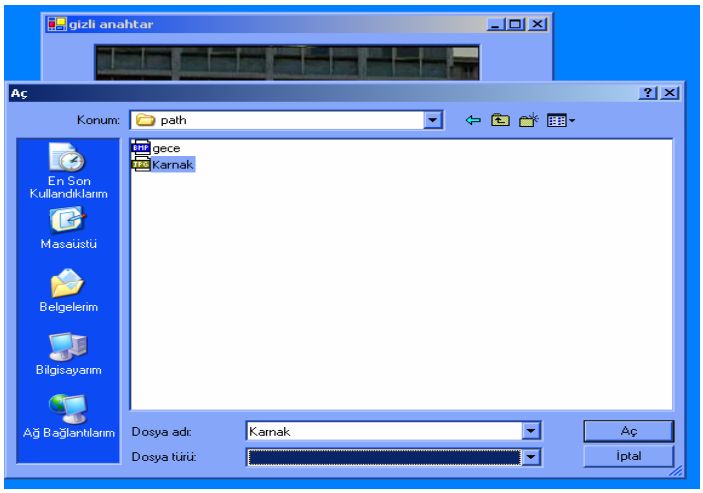
Şekil 4.8: Ekran klavyesi

4.4. Yönetim Arayüzü

Bu arayüz istemci modülü ile tümleşik çalışmakta ve işlevsel olarak kullanıcı adı, kullanıcı parolası ve gizli anahtar değiştirme modüllerini içermektedir. Bu veri değişikliği modülleri eski bilgilerin kullanıcı tarafından girilen yeni değerlerle değiştirilmesini temel alırlar. Gizli anahtar değiştirme modülü ise güvenli anahtar üretimi için anahtar değerlerinin yenilenmesinde farklı bir mekanizma içermektedir.

Bu mekanizmada sistemde yer alan ve kullanıcı tarafından seçilebilen görüntülerden uygulamada gömülü fonksiyonlar çerçevesinde elde edilen veriler hash fonksiyonundan geçirilerek anahtar üretilmesi söz konusudur. Şekil 4.9, 4.10 ve 4.11 gizli anahtar değiştirme arayüzünün işleyiş aşamalarını göstermektedir.

Şekil 4.9: Gizli anahtar değiştirme açılış ekranı



Şekil 4.10: Anahtar verisi için görüntü seçimi

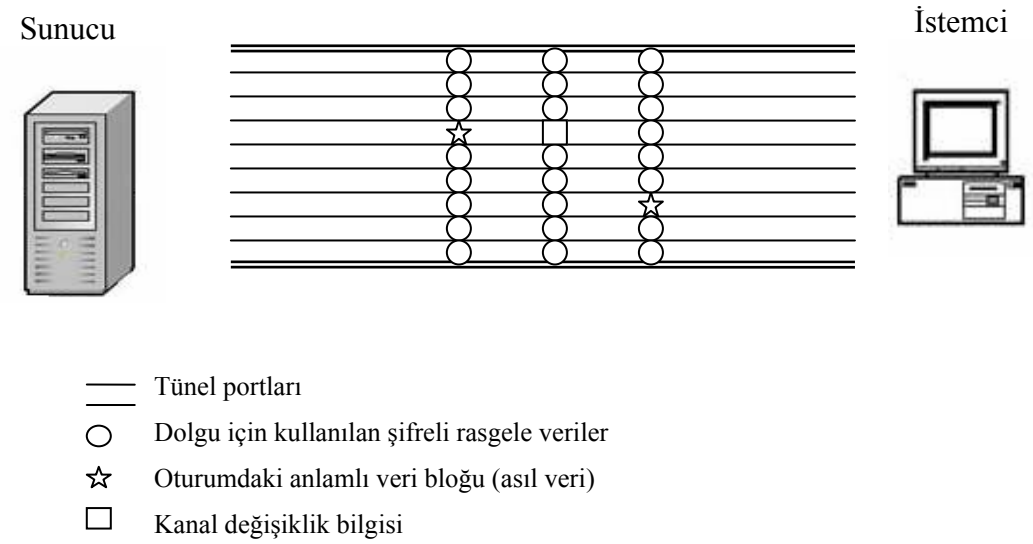
Şekil 4.11: Gizli anahtar üretimi sonuç ekranı

Gizli anahtarı oluşturacak verilerin elde edileceği görüntü havuzundan seçilen görüntü, onaylama emri ile işlenmeye başlanır. Görüntünün üzerinde belirlenen başlangıç ve bitiş koordinatları arasındaki alan arasındaki renk kodları tek yönlü hash fonksiyonundan geçirilerek sıkıştırılamaz gizli anahtarın üretilmesini sağlar. Üretilen bu anahtar protokol esnasında sunucu ile şifreli olacak şekilde paylaşılarak istemci-sunucu arasındaki senkronizasyon da sağlanacaktır.

4.5. Haberleşme Tüneli

İstemci ve sunucu arasında, güvenli bir oturum sağlanması amacıyla bir haberleşme tüneli şeması kurulmuştur. Bu tünel, sunucu tarafından seçilen portlarda protokolde yer alan trafik dolgulama şemasını ve şifreli verilerin gönderimini sağlamaktadır.

Oturum kanalı değişikliği sunucunun istemciye veri paketlerinde bir sonraki pakette hangi portun oturum için kullanılacağını ve şu an kullanılan portun rasgele şifrelenmiş verilerle dolgulanacağını bildirmesi ve istemcinin bunu onaylamasıyla sağlanır. Sunucu, oturumda dinleme yaptığı port numarasını, yeni port numarasıyla değiştirerek oturumu devam ettirir. Şekil 4.12'de istemci tarafından tünelin kullanılması ve oturum kanalının değişikliği simgesel olarak gösterilmiştir.

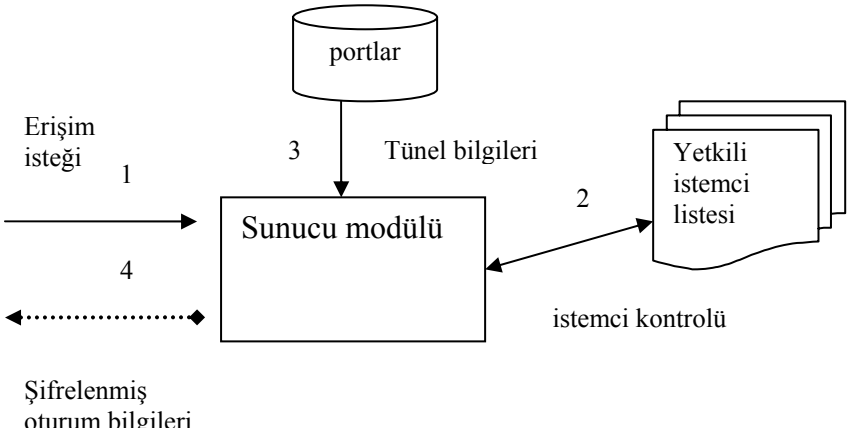


Şekil 4.12: Tünel şeması

4.6. Sunucu Modülü

Uygulamanın bu modülünde, istemci ve sunucu arasında kurulacak olan tünel şemasının belirlenmesi ve istemcinin yetki kontrolü alt modülleri yer almaktadır. Sunucu modülünde bağlantı kuran istemciye ait bilgiler, yetkili istemci listesi kontrol edilip istemcinin hangi kaynaklara erişim hakkı olduğu belirlenmektedir. Eğer erişim izni isteyen istemci bu listede yer almıyorsa istemcinin isteği reddedilecektir. Aynı listede istemci bilgilerine bağlantılı olarak bu istemciye ait gizli anahtar bilgileri de tutulmaktadır.

Sunucu, servis trafiğini kontrol ederek, oturum için oluşturulacak tünel için uygun olan port listesini çıkarır. Tünel şemasındaki dolgu portlarını ve haberleşmenin yapılacağı asıl portu bu listeden elde eder. Bu haberleşme verilerini ve oturum zaman-pulunu, yetkili istemci listesinden ilgili istemciye ait olan anahtarla şifreleyerek gönderir. Şekil 4.13'te sunucu modülünde tünel tespit şeması gösterilmiştir.



Şekil 4.13: Sunucu modülü

BEŞİNCİ BÖLÜM
SONUÇ VE ÖNERİLER

5. SONUÇ VE ÖNERİLER

Bu çalışmada tanınmayan uzak bir bilgisayarın sunuculardaki kaynaklara erişimini amaçlayan bir protokol yapısında; kimlik belirleme, doğrulama ve yetkilendirme problemleri ile trafik analizi ve yineleme gibi saldırılara karşı çözümler ele alınmıştır. Güvenli haberleşmenin sağlanabilmesi için gerekli ileti gizliliği ve veri bütünlüğü kavramları ön plana çıkarılmıştır.

Bir protokol tasarımının hangi kriterler üzerine kurulması ve tasarım yapılırken göz önünde bulundurulması gereken temel esaslar bu çalışmada bir araya getirilmiştir.Tasarlanan protokolde, kimlik denetimi ve ileti bütünlüğü rasgele sayı üreteci ve hash algoritmaları kullanılarak, ileti ve haberleşme gizliliği ise simetrik şifreleme yöntemleriyle sağlanmıştır. Sunucu ve istemci arasında oluşturulan haberleşme tünelinde gerçekleştirilen trafik dolgulama yöntemiyle ise trafik analizi saldırılarına karşı önlemler alınmıştır.

Kriptografik protokollerde kullanılan algoritmalar standartlara uygun olarak kodlanıp kriptografik hash kütüphanesi oluşturulmuştur. Hazırlanan kütüphanenin ileride planlanan araştırmalarda da kullanılabilir olması bu çalışmanın önemli faydalarındandır.

Tüm kriptografik haberleşme protokollerinde yer alan rasgele sayı serileri için görüntü uzayının kullanılması yeni bir yaklaşımdır. Böylece birbirinden bağımsız ve tahmin edilemez sayı serileri üretilerek, haberleşme güvenliği tehlikeye atılmayacaktır.

5.2. Öneriler ve Gelecek Çalışmalar

Protokol tasarımında ön planda tutulmayan uygulama performansı, anahtarlama ve rasgele sayı üretimi önceden yapılarak artırılabilir. Protokol grup iletişimine uyarlanarak anahtar dağıtım şeması uygulanabilir.

Kriptografik kütüphaneye açık anahtarlı şifreleme modülleri eklenmesi sertifikasyon ve sayısal imza uygulamalarını gerçekleyen haberleşme protokollerinin güvenliğini artıracaktır.

Yapılan sayısal içerik güvenliğinin lisans dağıtımı ve paylaşımı çalışmalarının kimlik denetimi ve yetkilendirme kısımlarında ve taşınabilir etmen teknolojisi tabanlı uzman sistemlerde etmenlerin kendi aralarındaki ve platformlarla kurdukları haberleşme ve etmen göçlerinin kriptografik olarak güvenilirliğinin sağlanmasında çalışmada yer alan ve test vektörleriyle geçerlilikleri sınanmış kriptografik kütüphaneden faydalanılacaktır.

EKLER
TEST VEKTÖRLERİ

Bu bölümde protokolde ve uygulamanın güvenliğini sağlamak için kullanılan SHA, HMAC ve RC6 algoritmalarının standartlarda geçen test vektörlerine yer verilmiştir.

EK.A SHA Test Vektörleri

A.1. SHA-1 Test Vektörleri

Örnekler bilgi vermek içindir ve standartlarla uyuşması gerekmemektedir.

A.1.1 Tek-blok İleti

M, iletisi 24-ikil (l=24) uzunluğundaki "abc" olsun. İletinin ikili sistemdeki karşılığı:
`01100001 01100010 01100011`

İleti "1" ve devamında 423 "0" ikilleri ile dolgulanmış iletinin sonuna ise uzunluğun (24) iki 32 ikillik kelime bloğunun onaltılık sistemdeki karşılığı `00000000 00000018` eklenmiştir.

Sonuçta elde edilen dolgulanmış ileti tek bir bloktan oluşmaktadır. (N=1)

SHA-1 için ilk hash değeri $H^{(0)}$:

$H_0^{(0)} =$ `67452301`

$H_1^{(0)} =$ `efcdab89`

$H_2^{(0)} =$ `98badcfe`

$H_3^{(0)} =$ `10325476`

$H_4^{(0)} =$ `c3d2e1f0`

Bundan sonra dolgulanmış iletinin kelime blokları ileti çizelgesine ait $W_0,\ldots,W_{15}$ kelime bloklarına atanır.

W_0 = 61626380

W_1 = 00000000

W_2 = 00000000

W_3 = 00000000

W_4 = 00000000

W_5 = 00000000

W_6 = 00000000

W_7 = 00000000

W_8 = 00000000

W_9 = 00000000

W_{10} = 00000000

W_{11} = 00000000

W_{12} = 00000000

W_{13} = 00000000

W_{14} = 00000000

W_{15} = 00000018.

Aşağıdaki çizelgede a,b,c,d,e nin daha önce belirtilen t=0 dan 79'a döngüsünden çıktıktan sonraki onaltılık değerleri verilmiştir.

	a	***b***	***c***	***d***	***e***
t = 0 :	0116fc33	67452301	7bf36ae2	98badcfe	10325476
t = 1 :	8990536d	0116fc33	59d148c0	7bf36ae2	98badcfe
t = 2 :	a1390f08	8990536d	c045bf0c	59d148c0	7bf36ae2
t = 3 :	cdd8e11b	a1390f08	626414db	c045bf0c	59d148c0
t = 4 :	cfd499de	cdd8e11b	284e43c2	626414db	c045bf0c
t = 5 :	3fc7ca40	cfd499de	f3763846	284e43c2	626414db
t = 6 :	993e30c1	3fc7ca40	b3f52677	f3763846	284e43c2
t = 7 :	9e8c07d4	993e30c1	0ff1f290	b3f52677	f3763846
t = 8 :	4b6ae328	9e8c07d4	664f8c30	0ff1f290	b3f52677
t = 9 :	8351f929	4b6ae328	27a301f5	664f8c30	0ff1f290
t = 10 :	fbda9e89	8351f929	12dab8ca	27a301f5	664f8c30
t = 11 :	63188fe4	fbda9e89	60d47e4a	12dab8ca	27a301f5
t = 12 :	4607b664	63188fe4	7ef6a7a2	60d47e4a	12dab8ca
t = 13 :	9128f695	4607b664	18c623f9	7ef6a7a2	60d47e4a
t = 14 :	196bee77	9128f695	1181ed99	18c623f9	7ef6a7a2

$t = 15$: 20bdd62f 196bee77 644a3da5 1181ed99 18c623f9

$t = 16$: 4e925823 20bdd62f c65afb9d 644a3da5 1181ed99

$t = 17$: 82aa6728 4e925823 c82f758b c65afb9d 644a3da5

$t = 18$: dc64901d 82aa6728 d3a49608 c82f758b c65afb9d

$t = 19$: fd9e1d7d dc64901d 20aa99ca d3a49608 c82f758b

$t = 20$: 1a37b0ca fd9e1d7d 77192407 20aa99ca d3a49608

$t = 21$: 33a23bfc 1a37b0ca 7f67875f 77192407 20aa99ca

$t = 22$: 21283486 33a23bfc 868dec32 7f67875f 77192407

$t = 23$: d541f12d 21283486 0ce88eff 868dec32 7f67875f

$t = 24$: c7567dc6 d541f12d 884a0d21 0ce88eff 868dec32

$t = 25$: 48413ba4 c7567dc6 75507c4b 884a0d21 0ce88eff

$t = 26$: be35fbd5 48413ba4 b1d59f71 75507c4b 884a0d21

$t = 27$: 4aa84d97 be35fbd5 12104ee9 b1d59f71 75507c4b

$t = 28$: 8370b52e 4aa84d97 6f8d7ef5 12104ee9 b1d59f71

$t = 29$: c5fbaf5d 8370b52e d2aa1365 6f8d7ef5 12104ee9

$t = 30$: 1267b407 c5fbaf5d a0dc2d4b d2aa1365 6f8d7ef5

$t = 31$: 3b845d33 1267b407 717eebd7 a0dc2d4b d2aa1365

$t = 32$: 046faa0a 3b845d33 c499ed01 717eebd7 a0dc2d4b

$t = 33$: 2c0ebc11 046faa0a cee1174c c499ed01 717eebd7

$t = 34$: 21796ad4 2c0ebc11 811bea82 cee1174c c499ed01

$t = 35$: dcbbb0cb 21796ad4 4b03af04 811bea82 cee1174c

$t = 36$: 0f511fd8 dcbbb0cb 085e5ab5 4b03af04 811bea82

$t = 37$: dc63973f 0f511fd8 f72eec32 085e5ab5 4b03af04

$t = 38$: 4c986405 dc63973f 03d447f6 f72eec32 085e5ab5

$t = 39$: 32de1cba 4c986405 f718e5cf 03d447f6 f72eec32

$t = 40$: fc87dedf 32de1cba 53261901 f718e5cf 03d447f6

$t = 41$: 970a0d5c fc87dedf 8cb7872e 53261901 f718e5cf

$t = 42$: 7f193dc5 970a0d5c ff21f7b7 8cb7872e 53261901

$t = 43$: ee1b1aaf 7f193dc5 25c28357 ff21f7b7 8cb7872e

$t = 44$: 40f28e09 ee1b1aaf 5fc64f71 25c28357 ff21f7b7

$t = 45$: 1c51e1f2 40f28e09 fb86c6ab 5fc64f71 25c28357

$t = 46$: a01b846c 1c51e1f2 503ca382 fb86c6ab 5fc64f71

$t = 47$: bead02ca a01b846c 8714787c 503ca382 fb86c6ab

$t = 48$: baf39337 bead02ca 2806e11b 8714787c 503ca382

$t = 49$: 120731c5 baf39337 afab40b2 2806e11b 8714787c

$t = 50$: 641db2ce 120731c5 eebce4cd afab40b2 2806e11b

$t = 51$: 3847ad66 641db2ce 4481cc71 eebce4cd afab40b2

$t = 52$: e490436d 3847ad66 99076cb3 4481cc71 eebce4cd

$t = 53$: 27e9f1d8 e490436d 8e11eb59 99076cb3 4481cc71

$t = 54$: 7b71f76d 27e9f1d8 792410db 8e11eb59 99076cb3
$t = 55$: 5e6456af 7b71f76d 09fa7c76 792410db 8e11eb59
$t = 56$: c846093f 5e6456af 5edc7ddb 09fa7c76 792410db
$t = 57$: d262ff50 c846093f d79915ab 5edc7ddb 09fa7c76
$t = 58$: 09d785fd d262ff50 f211824f d79915ab 5edc7ddb
$t = 59$: 3f52de5a 09d785fd 3498bfd4 f211824f d79915ab
$t = 60$: d756c147 3f52de5a 4275e17f 3498bfd4 f211824f
$t = 61$: 548c9cb2 d756c147 8fd4b796 4275e17f 3498bfd4
$t = 62$: b66c020b 548c9cb2 f5d5b051 8fd4b796 4275e17f
$t = 63$: 6b61c9e1 b66c020b 9523272c f5d5b051 8fd4b796
$t = 64$: 19dfa7ac 6b61c9e1 ed9b0082 9523272c f5d5b051
$t = 65$: 101655f9 19dfa7ac 5ad87278 ed9b0082 9523272c
$t = 66$: 0c3df2b4 101655f9 0677e9eb 5ad87278 ed9b0082
$t = 67$: 78dd4d2b 0c3df2b4 4405957e 0677e9eb 5ad87278
$t = 68$: 497093c0 78dd4d2b 030f7cad 4405957e 0677e9eb
$t = 69$: 3f2588c2 497093c0 de37534a 030f7cad 4405957e
$t = 70$: c199f8c7 3f2588c2 125c24f0 de37534a 030f7cad
$t = 71$: 39859de7 c199f8c7 8fc96230 125c24f0 de37534a
$t = 72$: edb42de4 39859de7 f0667e31 8fc96230 125c24f0
$t = 73$: 11793f6f edb42de4 ce616779 f0667e31 8fc96230
$t = 74$: 5ee76897 11793f6f 3b6d0b79 ce616779 f0667e31
$t = 75$: 63f7dab7 5ee76897 c45e4fdb 3b6d0b79 ce616779
$t = 76$: a079b7d9 63f7dab7 d7b9da25 c45e4fdb 3b6d0b79
$t = 77$: 860d21cc a079b7d9 d8fdf6ad d7b9da25 c45e4fdb
$t = 78$: 5738d5e1 860d21cc 681e6df6 d8fdf6ad d7b9da25
$t = 79$: 42541b35 5738d5e1 21834873 681e6df6 d8fdf6ad

Böylece ilk ve tek ileti bloğu $M^{(1)}$ in işlenmesi tamamlanmış olacaktır. Elde edilecek son hash değeri $H^{(1)}$:

$H_0^{(1)} =$ 67452301 + 42541b35 = a9993e36
$H_1^{(1)} =$ efcdab89 + 5738d5e1 = 4706816a
$H_2^{(1)} =$ 98badcfe + 21834873 = ba3e2571
$H_3^{(1)} =$ 10325476 + 681e6df6 = 7850c26c
$H_4^{(1)} =$ c3d2e1f0 + d8fdf6ad = 9cd0d89d

ve 160 ikillik ileti özeti:

a9993e36 4706816a ba3e2571 7850c26c 9cd0d89d

A.1.2. Çok-bloklu İleti

M, iletisi 448-ikil (l=448) uzunluğundaki "**abcdbcdecdefdefgefghfghighijhijkijkljkl mklmnlmnomnopnopq**" olsun.

İleti "1" ve devamında 511 "0" ikilleri ile dolgulanmış, iletinin sonuna ise uzunluğun (448) iki 32 ikillik kelime bloğunun onaltılık sistemdeki karşılığı `00000000 000001c0` eklenmiştir.

Sonuçta elde edilen dolgulanmış ileti iki bloktan oluşmaktadır. (N=2)

SHA-1 için ilk hash değeri $H^{(0)}$:

$H_0^{(0)}$ = `67452301`
$H_1^{(0)}$ = `efcdab89`
$H_2^{(0)}$ = `98badcfe`
$H_3^{(0)}$ = `10325476`
$H_4^{(0)}$ = `c3d2e1f0`

Bundan sonra dolgulanmış iletinin ilk kelime bloğu $M^{(1)}$ ileti çizelgesine ait $W_0,\ldots,W_{15}$ kelime bloklarına atanır:

W_0 = `61626364`
W_1 = `62636465`
W_2 = `63646566`
W_3 = `64656667`
W_4 = `65666768`
W_5 = `66676869`
W_6 = `6768696a`
W_7 = `68696a6b`
W_8 = `696a6b6c`
W_9 = `6a6b6c6d`

W_{10} = 6b6c6d6e

W_{11} = 6c6d6e6f

W_{12} = 6d6e6f70

W_{13} = 6e6f7071

W_{14} = 80000000

W_{15} = 00000000.

Aşağıdaki çizelgede a,b,c,d,e nin daha önce belirtilen t=0 dan 79'a döngüsünden çıktıktan sonraki onaltılık değerleri verilmiştir.

	a	***b***	***c***	***d***	***e***
$t = 0$:	0116fc17	67452301	7bf36ae2	98badcfe	10325476
$t = 1$:	ebf3b452	0116fc17	59d148c0	7bf36ae2	98badcfe
$t = 2$:	5109913a	ebf3b452	c045bf05	59d148c0	7bf36ae2
$t = 3$:	2c4f6eac	5109913a	bafced14	c045bf05	59d148c0
$t = 4$:	33f4ae5b	2c4f6eac	9442644e	bafced14	c045bf05
$t = 5$:	96b85189	33f4ae5b	0b13dbab	9442644e	bafced14
$t = 6$:	db04cb58	96b85189	ccfd2b96	0b13dbab	9442644e
$t = 7$:	45833f0f	db04cb58	65ae1462	ccfd2b96	0b13dbab
$t = 8$:	c565c35e	45833f0f	36c132d6	65ae1462	ccfd2b96
$t = 9$:	6350afda	c565c35e	d160cfc3	36c132d6	65ae1462
$t = 10$:	8993ea77	6350afda	b15970d7	d160cfc3	36c132d6
$t = 11$:	e19ecaa2	8993ea77	98d42bf6	b15970d7	d160cfc3
$t = 12$:	8603481e	e19ecaa2	e264fa9d	98d42bf6	b15970d7
$t = 13$:	32f94a85	8603481e	b867b2a8	e264fa9d	98d42bf6
$t = 14$:	b2e7a8be	32f94a85	a180d207	b867b2a8	e264fa9d
$t = 15$:	42637e39	b2e7a8be	4cbe52a1	a180d207	b867b2a8
$t = 16$:	6b068048	42637e39	acb9ea2f	4cbe52a1	a180d207
$t = 17$:	426b9c35	6b068048	5098df8e	acb9ea2f	4cbe52a1
$t = 18$:	944b1bd1	426b9c35	1ac1a012	5098df8e	acb9ea2f
$t = 19$:	6c445652	944b1bd1	509ae70d	1ac1a012	5098df8e
$t = 20$:	95836da5	6c445652	6512c6f4	509ae70d	1ac1a012
$t = 21$:	09511177	95836da5	9b111594	6512c6f4	509ae70d
$t = 22$:	e2b92dc4	09511177	6560db69	9b111594	6512c6f4
$t = 23$:	fd224575	e2b92dc4	c254445d	6560db69	9b111594
$t = 24$:	eeb82d9a	fd224575	38ae4b71	c254445d	6560db69
$t = 25$:	5a142c1a	eeb82d9a	7f48915d	38ae4b71	c254445d

$t = 26$: 2972f7c7 5a142c1a bbae0b66 7f48915d 38ae4b71

$t = 27$: d526a644 2972f7c7 96850b06 bbae0b66 7f48915d

$t = 28$: e1122421 d526a644 ca5cbdf1 96850b06 bbae0b66

$t = 29$: 05b457b2 e1122421 3549a991 ca5cbdf1 96850b06

$t = 30$: a9c84bec 05b457b2 78448908 3549a991 ca5cbdf1

$t = 31$: 52e31f60 a9c84bec 816d15ec 78448908 3549a991

$t = 32$: 5af3242c 52e31f60 2a7212fb 816d15ec 78448908

$t = 33$: 31c756a9 5af3242c 14b8c7d8 2a7212fb 816d15ec

$t = 34$: e9ac987c 31c756a9 16bcc90b 14b8c7d8 2a7212fb

$t = 35$: ab7c32ee e9ac987c 4c71d5aa 16bcc90b 14b8c7d8

$t = 36$: 5933fc99 ab7c32ee 3a6b261f 4c71d5aa 16bcc90b

$t = 37$: 43f87ae9 5933fc99 aadf0cbb 3a6b261f 4c71d5aa

$t = 38$: 24957f22 43f87ae9 564cff26 aadf0cbb 3a6b261f

$t = 39$: adeb7478 24957f22 50fe1eba 564cff26 aadf0cbb

$t = 40$: d70e5010 adeb7478 89255fc8 50fe1eba 564cff26

$t = 41$: 79bcfb08 d70e5010 2b7add1e 89255fc8 50fe1eba

$t = 42$: f9bcb8de 79bcfb08 35c39404 2b7add1e 89255fc8

$t = 43$: 633e9561 f9bcb8de 1e6f3ec2 35c39404 2b7add1e

$t = 44$: 98c1ea64 633e9561 be6f2e37 1e6f3ec2 35c39404

$t = 45$: c6ea241e 98c1ea64 58cfa558 be6f2e37 1e6f3ec2

$t = 46$: a2ad4f02 c6ea241e 26307a99 58cfa558 be6f2e37

$t = 47$: c8a69090 a2ad4f02 b1ba8907 26307a99 58cfa558

$t = 48$: 88341600 c8a69090 a8ab53c0 b1ba8907 26307a99

$t = 49$: 7e846f58 88341600 3229a424 a8ab53c0 b1ba8907

$t = 50$: 86e358ba 7e846f58 220d0580 3229a424 a8ab53c0

$t = 51$: 8d2e76c8 86e358ba 1fa11bd6 220d0580 3229a424

$t = 52$: ce892e10 8d2e76c8 a1b8d62e 1fa11bd6 220d0580

$t = 53$: edea95b1 ce892e10 234b9db2 a1b8d62e 1fa11bd6

$t = 54$: 36d1230a edea95b1 33a24b84 234b9db2 a1b8d62e

$t = 55$: 776c3910 36d1230a 7b7aa56c 33a24b84 234b9db2

$t = 56$: a681b723 776c3910 8db448c2 7b7aa56c 33a24b84

$t = 57$: ac0a794f a681b723 1ddb0e44 8db448c2 7b7aa56c

$t = 58$: f03d3782 ac0a794f e9a06dc8 1ddb0e44 8db448c2

$t = 59$: 9ef775c3 f03d3782 eb029e53 e9a06dc8 1ddb0e44

$t = 60$: 36254b13 9ef775c3 bc0f4de0 eb029e53 e9a06dc8

$t = 61$: 4080d4dc 36254b13 e7bddd70 bc0f4de0 eb029e53

$t = 62$: 2bfaf7a8 4080d4dc cd8952c4 e7bddd70 bc0f4de0

$t = 63$: 513f9ca0 2bfaf7a8 10203537 cd8952c4 e7bddd70

$t = 64$: e5895c81 513f9ca0 0afebdea 10203537 cd8952c4

$t = 65$: 1037d2d5 e5895c81 144fe728 0afebdea 10203537
$t = 66$: 14a82da9 1037d2d5 79625720 144fe728 0afebdea
$t = 67$: 6d17c9fd 14a82da9 440df4b5 79625720 144fe728
$t = 68$: 2c7b07bd 6d17c9fd 452a0b6a 440df4b5 79625720
$t = 69$: fdf6efff 2c7b07bd 5b45f27f 452a0b6a 440df4b5
$t = 70$: 112b96e3 fdf6efff 4b1ec1ef 5b45f27f 452a0b6a
$t = 71$: 84065712 112b96e3 ff7dbbff 4b1ec1ef 5b45f27f
$t = 72$: ab89fb71 84065712 c44ae5b8 ff7dbbff 4b1ec1ef
$t = 73$: c5210e35 ab89fb71 a10195c4 c44ae5b8 ff7dbbff
$t = 74$: 352d9f4b c5210e35 6ae27edc a10195c4 c44ae5b8
$t = 75$: 1a0e0e0a 352d9f4b 7148438d 6ae27edc a10195c4
$t = 76$: d0d47349 1a0e0e0a cd4b67d2 7148438d 6ae27edc
$t = 77$: ad38620d d0d47349 86838382 cd4b67d2 7148438d
$t = 78$: d3ad7c25 ad38620d 74351cd2 86838382 cd4b67d2
$t = 79$: 8ce34517 d3ad7c25 6b4e1883 74351cd2 86838382

Böylece ilk ileti bloğu $M^{(1)}$ in işlenmesi tamamlanmış olacaktır. Elde edilecek ara hash değeri $H^{(1)}$:

$H_0^{(1)} =$ 67452301 + 8ce34517 = f4286818
$H_1^{(1)} =$ efcdab89 + d3ad7c25 = c37b27ae
$H_2^{(1)} =$ 98badcfe + 6b4e1883 = 0408f581
$H_3^{(1)} =$ 10325476 + 74351cd2 = 84677148
$H_4^{(1)} =$ c3d2e1f0 + 86838382 = 4a566572

Dolgulanmış iletinin ikinci kelime bloğu $M^{(2)}$ ileti çizelgesine ait $W_0,\ldots,W_{15}$ kelime bloklarına atanır:

$W_0 =$ 00000000
$W_1 =$ 00000000
$W_2 =$ 00000000
$W_3 =$ 00000000
$W_4 =$ 00000000
$W_5 =$ 00000000
$W_6 =$ 00000000

W_7 = 00000000

W_8 = 00000000

W_9 = 00000000

W_{10} = 00000000

W_{11} = 00000000

W_{12} = 00000000

W_{13} = 00000000

W_{14} = 00000000

W_{15} = 000001c0.

Aşağıdaki çizelgede a,b,c,d,e nin daha önce belirtilen t=0 dan 79'a döngüsünden çıktıktan sonraki onaltılık değerleri verilmiştir.

	a	***b***	***c***	***d***	***e***
t = 0 :	2df257e9	f4286818	b0dec9eb	0408f581	84677148
t = 1 :	4d3dc58f	2df257e9	3d0a1a06	b0dec9eb	0408f581
t = 2 :	c352bb05	4d3dc58f	4b7c95fa	3d0a1a06	b0dec9eb
t = 3 :	eef743c6	c352bb05	d34f7163	4b7c95fa	3d0a1a06
t = 4 :	41e34277	eef743c6	70d4aec1	d34f7163	4b7c95fa
t = 5 :	5443915c	41e34277	bbbdd0f1	70d4aec1	d34f7163
t = 6 :	e7fa0377	5443915c	d078d09d	bbbdd0f1	70d4aec1
t = 7 :	c6946813	e7fa0377	1510e457	d078d09d	bbbdd0f1
t = 8 :	fdde1de1	c6946813	f9fe80dd	1510e457	d078d09d
t = 9 :	b8538aca	fdde1de1	f1a51a04	f9fe80dd	1510e457
t = 10 :	6ba94f63	b8538aca	7f778778	f1a51a04	f9fe80dd
t = 11 :	43a2792f	6ba94f63	ae14e2b2	7f778778	f1a51a04
t = 12 :	fecd7bbf	43a2792f	daea53d8	ae14e2b2	7f778778
t = 13 :	a2604ca8	fecd7bbf	d0e89e4b	daea53d8	ae14e2b2
t = 14 :	258b0baa	a2604ca8	ffb35eef	d0e89e4b	daea53d8
t = 15 :	d9772360	258b0baa	2898132a	ffb35eef	d0e89e4b
t = 16 :	5507db6e	d9772360	8962c2ea	2898132a	ffb35eef
t = 17 :	a51b58bc	5507db6e	365dc8d8	8962c2ea	2898132a
t = 18 :	c2eb709f	a51b58bc	9541f6db	365dc8d8	8962c2ea
t = 19 :	d8992153	c2eb709f	2946d62f	9541f6db	365dc8d8
t = 20 :	37482f5f	d8992153	f0badc27	2946d62f	9541f6db

$t = 21$: ee8700bd 37482f5f f6264854 f0badc27 2946d62f

$t = 22$: 9ad594b9 ee8700bd cdd20bd7 f6264854 f0badc27

$t = 23$: 8fbaa5b9 9ad594b9 7ba1c02f cdd20bd7 f6264854

$t = 24$: 88fb5867 8fbaa5b9 66b5652e 7ba1c02f cdd20bd7

$t = 25$: eec50521 88fb5867 63eea96e 66b5652e 7ba1c02f

$t = 26$: 50bce434 eec50521 e23ed619 63eea96e 66b5652e

$t = 27$: 5c416daf 50bce434 7bb14148 e23ed619 63eea96e

$t = 28$: 2429be5f 5c416daf 142f390d 7bb14148 e23ed619

$t = 29$: 0a2fb108 2429be5f d7105b6b 142f390d 7bb14148

$t = 30$: 17986223 0a2fb108 c90a6f97 d7105b6b 142f390d

$t = 31$: 8a4af384 17986223 028bec42 c90a6f97 d7105b6b

$t = 32$: 6b629993 8a4af384 c5e61888 028bec42 c90a6f97

$t = 33$: f15f04f3 6b629993 2292bce1 c5e61888 028bec42

$t = 34$: 295cc25b f15f04f3 dad8a664 2292bce1 c5e61888

$t = 35$: 696da404 295cc25b fc57c13c dad8a664 2292bce1

$t = 36$: cef5ae12 696da404 ca573096 fc57c13c dad8a664

$t = 37$: 87d5b80c cef5ae12 1a5b6901 ca573096 fc57c13c

$t = 38$: 84e2a5f2 87d5b80c b3bd6b84 1a5b6901 ca573096

$t = 39$: 03bb6310 84e2a5f2 21f56e03 b3bd6b84 1a5b6901

$t = 40$: c2d8f75f 03bb6310 a138a97c 21f56e03 b3bd6b84

$t = 41$: bfb25768 c2d8f75f 00eed8c4 a138a97c 21f56e03

$t = 42$: 28589152 bfb25768 f0b63dd7 00eed8c4 a138a97c

$t = 43$: ec1d3d61 28589152 2fec95da f0b63dd7 00eed8c4

$t = 44$: 3caed7af ec1d3d61 8a162454 2fec95da f0b63dd7

$t = 45$: c3d033ea 3caed7af 7b074f58 8a162454 2fec95da

$t = 46$: 7316056a c3d033ea cf2bb5eb 7b074f58 8a162454

$t = 47$: 46f93b68 7316056a b0f40cfa cf2bb5eb 7b074f58

$t = 48$: dc8e7f26 46f93b68 9cc5815a b0f40cfa cf2bb5eb

$t = 49$: 850d411c dc8e7f26 11be4eda 9cc5815a b0f40cfa

$t = 50$: 7e4672c0 850d411c b7239fc9 11be4eda 9cc5815a

$t = 51$: 89fbd41d 7e4672c0 21435047 b7239fc9 11be4eda

$t = 52$: 1797e228 89fbd41d 1f919cb0 21435047 b7239fc9

$t = 53$: 431d65bc 1797e228 627ef507 1f919cb0 21435047

$t = 54$: 2bdbb8cb 431d65bc 05e5f88a 627ef507 1f919cb0

$t = 55$: 6da72e7f 2bdbb8cb 10c7596f 05e5f88a 627ef507

$t = 56$: a8495a9b 6da72e7f caf6ee32 10c7596f 05e5f88a

$t = 57$: e785655a a8495a9b db69cb9f caf6ee32 10c7596f

$t = 58$: 5b086c42 e785655a ea1256a6 db69cb9f caf6ee32

$t = 59$: a65818f7 5b086c42 b9e15956 ea1256a6 db69cb9f

$t = 60$: 7aab101b a65818f7 96c21b10 b9e15956 ea1256a6
$t = 61$: 93614c9c 7aab101b e996063d 96c21b10 b9e15956
$t = 62$: f66d9bf4 93614c9c deaac406 e996063d 96c21b10
$t = 63$: d504902b f66d9bf4 24d85327 deaac406 e996063d
$t = 64$: 60a9da62 d504902b 3d9b66fd 24d85327 deaac406
$t = 65$: 8b687819 60a9da62 f541240a 3d9b66fd 24d85327
$t = 66$: 083e90c3 8b687819 982a7698 f541240a 3d9b66fd
$t = 67$: f6226bbf 083e90c3 62da1e06 982a7698 f541240a
$t = 68$: 76c0563b f6226bbf c20fa430 62da1e06 982a7698
$t = 69$: 989dd165 76c0563b fd889aef c20fa430 62da1e06
$t = 70$: 8b2c7573 989dd165 ddb0158e fd889aef c20fa430
$t = 71$: ae1b8e7b 8b2c7573 66277459 ddb0158e fd889aef
$t = 72$: ca1840de ae1b8e7b e2cb1d5c 66277459 ddb0158e
$t = 73$: 16f3babb ca1840de eb86e39e e2cb1d5c 66277459
$t = 74$: d28d83ad 16f3babb b2861037 eb86e39e e2cb1d5c
$t = 75$: 6bc02dfe d28d83ad c5bceeae b2861037 eb86e39e
$t = 76$: d3a6e275 6bc02dfe 74a360eb c5bceeae b2861037
$t = 77$: da955482 d3a6e275 9af00b7f 74a360eb c5bceeae
$t = 78$: 58c0aac0 da955482 74e9b89d 9af00b7f 74a360eb
$t = 79$: 906fd62c 58c0aac0 b6a55520 74e9b89d 9af00b7f

Böylece ikinci ve son ileti bloğu $M^{(2)}$ in işlenmesi tamamlanmış olacaktır. Elde edilecek son hash değeri $H^{(2)}$:

$H_0^{(1)}$= f4286818 + 906fd62c = 84983e44
$H_1^{(1)}$= c37b27ae + 58c0aac0 = 1c3bd26e
$H_2^{(1)}$= 0408f581 + b6a55520 = baae4aa1
$H_3^{(1)}$= 84677148 + 74e9b89d = f95129e5
$H_4^{(1)}$= 4a566572 + 9af00b7f = e54670f1

ve 160 ikillik ileti özeti:
84983e44 1c3bd26e baae4aa1 f95129e5 e54670f1

A.1.3 Uzun İleti

M, iletisi 1000000 kez "a" karakterinin tekrarı ile oluşturulmuş olsun. SHA-1 ileti özeti:
34aa973c d4c4daa4 f61eeb2b dbad2731 6534016f

A.2 SHA-256 Test Vektörleri

Örnekler bilgi vermek içindir ve standartlarla uyuşması gerekmemektedir.

A.2.1 Tek-blok İleti

M, iletisi 24-ikil (l=24) uzunluğundaki "abc" olsun. İletinin ikili sistemdeki karşılığı:
`01100001 01100010 01100011`

İleti "1" ve devamında 423 "0" ikilleri ile dolgulanmış iletinin sonuna ise uzunluğun (24)iki 32 ikillik kelime bloğunun onaltılık sistemdeki karşılığı `00000000 00000018` eklenmiştir.

Sonuçta elde edilen dolgulanmış ileti tek bir bloktan oluşmaktadır. (N=1)

SHA-256 için ilk hash değeri $H^{(0)}$:

$H_0^{(0)} =$ `6a09e667`
$H_1^{(0)} =$ `bb67ae85`
$H_2^{(0)} =$ `3c6ef372`
$H_3^{(0)} =$ `a54ff53a`
$H_4^{(0)} =$ `510e527f`
$H5^{(0)} =$ `9b05688c`
$H_6^{(0)} =$ `1f83d9ab`
$H_7^{(0)} =$ `5be0cd19`

Bundan sonra dolgulanmış iletinin kelime bloğu $M^{(1)}$ ileti çizelgesine ait $W_0,\ldots,W_{15}$ kelime bloklarına atanır:

$W_0 =$ `61626380`
$W_1 =$ `00000000`
$W_2 =$ `00000000`
$W_3 =$ `00000000`
$W_4 =$ `00000000`

W_5 = 00000000

W_6 = 00000000

W_7 = 00000000

W_8 = 00000000

W_9 = 00000000

W_{10} = 00000000

W_{11} = 00000000

W_{12} = 00000000

W_{13} = 00000000

W_{14} = 00000000

W_{15} = 00000018

Aşağıdaki çizelgede a,b,c,d,e,f,g,h nin daha önce belirtilen t=0 dan 63'e döngüsünden çıktıktan sonraki onaltılık değerleri verilmiştir.

	a	***b***	***c***	***d***	***e***	***f***	***g***	***h***
t = 0 :	5d6aebcd	6a09e667	bb67ae85	3c6ef372	fa2a4622	510e527f	9b05688c	1f83d9ab
t = 1 :	5a6ad9ad	5d6aebcd	6a09e667	bb67ae85	78ce7989	fa2a4622	510e527f	9b05688c
t = 2 :	c8c347a7	5a6ad9ad	5d6aebcd	6a09e667	f92939eb	78ce7989	fa2a4622	510e527f
t = 3 :	d550f666	c8c347a7	5a6ad9ad	5d6aebcd	24e00850	f92939eb	78ce7989	fa2a4622
t = 4 :	04409a6a	d550f666	c8c347a7	5a6ad9ad	43ada245	24e00850	f92939eb	78ce7989
t = 5 :	2b4209f5	04409a6a	d550f666	c8c347a7	714260ad	43ada245	24e00850	f92939eb
t = 6 :	e5030380	2b4209f5	04409a6a	d550f666	9b27a401	714260ad	43ada245	24e00850
t = 7 :	85a07b5f	e5030380	2b4209f5	04409a6a	0c657a79	9b27a401	714260ad	43ada245
t = 8 :	8e04ecb9	85a07b5f	e5030380	2b4209f5	32ca2d8c	0c657a79	9b27a401	714260ad
t = 9 :	8c87346b	8e04ecb9	85a07b5f	e5030380	1cc92596	32ca2d8c	0c657a79	9b27a401
t = 10 :	4798a3f4	8c87346b	8e04ecb9	85a07b5f	436b23e8	1cc92596	32ca2d8c	0c657a79
t = 11 :	f71fc5a9	4798a3f4	8c87346b	8e04ecb9	816fd6e9	436b23e8	1cc92596	32ca2d8c
t = 12 :	87912990	f71fc5a9	4798a3f4	8c87346b	1e578218	816fd6e9	436b23e8	1cc92596
t = 13 :	d932eb16	87912990	f71fc5a9	4798a3f4	745a48de	1e578218	816fd6e9	436b23e8
t = 14 :	c0645fde	d932eb16	87912990	f71fc5a9	0b92f20c	745a48de	1e578218	816fd6e9
t = 15 :	b0fa238e	c0645fde	d932eb16	87912990	07590dcd	0b92f20c	745a48de	1e578218
t = 16 :	21da9a9b	b0fa238e	c0645fde	d932eb16	8034229c	07590dcd	0b92f20c	745a48de
t = 17 :	c2fbd9d1	21da9a9b	b0fa238e	c0645fde	846ee454	8034229c	07590dcd	0b92f20c
t = 18 :	fe777bbf	c2fbd9d1	21da9a9b	b0fa238e	cc899961	846ee454	8034229c	07590dcd
t = 19 :	e1f20c33	fe777bbf	c2fbd9d1	21da9a9b	b0638179	cc899961	846ee454	8034229c
t = 20 :	9dc68b63	e1f20c33	fe777bbf	c2fbd9d1	8ada8930	b0638179	cc899961	846ee454

$t = 21$: c2606d6d 9dc68b63 e1f20c33 fe777bbf e1257970 8ada8930 b0638179 cc899961

$t = 22$: a7a3623f c2606d6d 9dc68b63 e1f20c33 49f5114a e1257970 8ada8930 b0638179

$t = 23$: c5d53d8d a7a3623f c2606d6d 9dc68b63 aa47c347 49f5114a e1257970 8ada8930

$t = 24$: 1c2c2838 c5d53d8d a7a3623f c2606d6d 2823ef91 aa47c347 49f5114a e1257970

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Böylece ilk ve tek ileti bloğu $M^{(1)}$ in işlenmesi tamamlanmış olacaktır. Elde edilecek son hash değeri $H^{(1)}$:

$H_0^{(1)} =$ 6a09e667 + 506e3058 = ba7816bf

$H_1^{(1)} =$ bb67ae85 + d39a2165 = 8f01cfea

$H_2^{(1)} =$ 3c6ef372 + 04d24d6c = 414140de

$H_3^{(1)} =$ a54ff53a + b85e2ce9 = 5dae2223

$H_4^{(1)} =$ 510e527f + 5ef50f24 = b00361a3

$H_5^{(1)} =$ 9b05688c + fb121210 = 96177a9c

$H_6^{(1)} =$ 1f83d9ab + 948d25b6 = b410ff61

$H_7^{(1)} =$ 5be0cd19 + 961f4894 = f20015ad.

ve elde edilen 256-ikillik ileti özeti:

ba7816bf 8f01cfea 414140de 5dae2223 b00361a3 96177a9c b410ff61 f20015ad

A.2.2 Çok-bloklu İleti

M, iletisi 448-ikil (l=448) uzunluğundaki "**abcdbcdecdefdefgefghfghighijhijkijkljkl mklmnlmnomnopnopq**" olsun.

İleti "1" ve devamında 511 "0" ikilleri ile dolgulanmış, iletinin sonuna ise uzunluğun (448) iki 32 ikillik kelime bloğunun onaltılık sistemdeki karşılığı 00000000 000001c0 eklenmiştir.

Sonuçta elde edilen dolgulanmış ileti iki bloktan oluşmaktadır. (N=2)

SHA-256 için ilk hash değeri $H^{(0)}$:

$H_0^{(0)}$ = 6a09e667

$H_1^{(0)}$ = bb67ae85

$H_2^{(0)}$ = 3c6ef372

$H_3^{(0)}$ = a54ff53a

$H_4^{(0)}$ = 510e527f

$H_5^{(0)}$ = 9b05688c

$H_6^{(0)}$ = 1f83d9ab

$H_7^{(0)}$ = 5be0cd19

Bundan sonra dolgulanmış iletinin ilk kelime bloğu $M^{(1)}$ ileti çizelgesine ait $W_0,\ldots,W_{15}$ kelime bloklarına atanır:

W_0 = 61626364

W_1 = 62636465

W_2 = 63646566

W_3 = 64656667

W_4 = 65666768

W_5 = 66676869

W_6 = 6768696a

W_7 = 68696a6b

W_8 = 696a6b6c

W_9 = 6a6b6c6d

W_{10} = 6b6c6d6e

W_{11} = 6c6d6e6f

W_{12} = 6d6e6f70

W_{13} = 6e6f7071

W_{14} = 80000000

W_{15} = 00000000.

Aşağıdaki çizelgede a,b,c,d,e,f,g,h nin daha önce belirtilen t=0 dan 63'e döngüsünden çıktıktan sonraki onaltılık değerleri verilmiştir.

a ***b*** ***c*** ***d*** ***e*** ***f*** ***g*** ***h***

$t = 0$: 5d6aebb1 6a09e667 bb67ae85 3c6ef372 fa2a4606 510e527f 9b05688c 1f83d9ab

$t = 1$: 2f2d5fcf 5d6aebb1 6a09e667 bb67ae85 4eb1cfce fa2a4606 510e527f 9b05688c

$t = 2$: 97651825 2f2d5fcf 5d6aebb1 6a09e667 62d5c49e 4eb1cfce fa2a4606 510e527f

$t = 3$: 4a8d64d5 97651825 2f2d5fcf 5d6aebb1 6494841b 62d5c49e 4eb1cfce fa2a4606

$t = 4$: f921c212 4a8d64d5 97651825 2f2d5fcf 05c4f88a 6494841b 62d5c49e 4eb1cfce

$t = 5$: 55c8ef48 f921c212 4a8d64d5 97651825 7ff91c94 05c4f88a 6494841b 62d5c49e

$t = 6$: 485835b7 55c8ef48 f921c212 4a8d64d5 39a5b2ca 7ff91c94 05c4f88a 6494841b

$t = 7$: d237e6db 485835b7 55c8ef48 f921c212 a401d211 39a5b2ca 7ff91c94 05c4f88a

$t = 8$: 359f2bce d237e6db 485835b7 55c8ef48 c09ffec4 a401d211 39a5b2ca 7ff91c94

$t = 9$: 3a474b2b 359f2bce d237e6db 485835b7 9037b3b8 c09ffec4 a401d211 39a5b2ca

$t = 10$: b8e2b4cb 3a474b2b 359f2bce d237e6db 443ed29e 9037b3b8 c09ffec4 a401d211

$t = 11$: 1762215c b8e2b4cb 3a474b2b 359f2bce ee1c97a8 443ed29e 9037b3b8 c09ffec4

$t = 12$: 101a4861 1762215c b8e2b4cb 3a474b2b 839a0fc9 ee1c97a8 443ed29e 9037b3b8

$t = 13$: d68e6457 101a4861 1762215c b8e2b4cb 9243f8af 839a0fc9 ee1c97a8 443ed29e

$t = 14$: dd16cbb3 d68e6457 101a4861 1762215c 9162aded 9243f8af 839a0fc9 ee1c97a8

$t = 15$: c3486194 dd16cbb3 d68e6457 101a4861 1496a54f 9162aded 9243f8af 839a0fc9

$t = 16$: b9dcacb1 c3486194 dd16cbb3 d68e6457 d4f64250 1496a54f 9162aded 9243f8af

$t = 17$: 046a193e b9dcacb1 c3486194 dd16cbb3 885370b6 d4f64250 1496a54f 9162aded

$t = 18$: f402f058 046a193e b9dcacb1 c3486194 6f433549 885370b6 d4f64250 1496a54f

$t = 19$: 2139187b f402f058 046a193e b9dcacb1 7c304206 6f433549 885370b6 d4f64250

$t = 20$: d70ac17d 2139187b f402f058 046a193e 7cc6b262 7c304206 6f433549 885370b6

$t = 21$: 1b2b66b8 d70ac17d 2139187b f402f058 d560b028 7cc6b262 7c304206 6f433549

$t = 22$: ae2e2d4f 1b2b66b8 d70ac17d 2139187b f074fc95 d560b028 7cc6b262 7c304206

$t = 23$: 59fce6b9 ae2e2d4f 1b2b66b8 d70ac17d a2c7d51d f074fc95 d560b028 7cc6b262

$t = 24$: 4a885065 59fce6b9 ae2e2d4f 1b2b66b8 763597fb a2c7d51d f074fc95 d560b028

$t = 25$: 573221da 4a885065 59fce6b9 ae2e2d4f 36e74eb4 763597fb a2c7d51d f074fc95

$t = 26$: 128661da 573221da 4a885065 59fce6b9 1162d575 36e74eb4 763597fb a2c7d51d

$t = 27$: 73f858af 128661da 573221da 4a885065 e77c797f 1162d575 36e74eb4 763597fb

$t = 28$: 74bcf468 73f858af 128661da 573221da 72abaecd e77c797f 1162d575 36e74eb4

$t = 29$: df7151a0 74bcf468 73f858af 128661da 7629c961 72abaecd e77c797f 1162d575

$t = 30$: eb43f3ed df7151a0 74bcf468 73f858af 0635d880 7629c961 72abaecd e77c797f

$t = 31$: 5581ab07 eb43f3ed df7151a0 74bcf468 df980085 0635d880 7629c961 72abaecd

$t = 32$: 9fc905c8 5581ab07 eb43f3ed df7151a0 a94d2af1 df980085 0635d880 7629c961

$t = 33$: 9ce5a62f 9fc905c8 5581ab07 eb43f3ed 6ef3b6bd a94d2af1 df980085 0635d880

$t = 34$: 1df8e885 9ce5a62f 9fc905c8 5581ab07 2a9e048e 6ef3b6bd a94d2af1 df980085

$t = 35$: 0786dce8 1df8e885 9ce5a62f 9fc905c8 de2a21d1 2a9e048e 6ef3b6bd a94d2af1

$t = 36$: 2c55d3a6 0786dce8 1df8e885 9ce5a62f b067c1af de2a21d1 2a9e048e 6ef3b6bd

$t = 37$: a985b4be 2c55d3a6 0786dce8 1df8e885 f72bf353 b067c1af de2a21d1 2a9e048e

$t = 38$: 91ac9d5d a985b4be 2c55d3a6 0786dce8 68d8d590 f72bf353 b067c1af de2a21d1

$t = 39$: 7e4d30b8 91ac9d5d a985b4be 2c55d3a6 9f5b9b6d 68d8d590 f72bf353 b067c1af

$t = 40$: 7e056794 7e4d30b8 91ac9d5d a985b4be 423b26c0 9f5b9b6d 68d8d590 f72bf353

$t = 41$: 508a16ab 7e056794 7e4d30b8 91ac9d5d 45459d97 423b26c0 9f5b9b6d 68d8d590

$t = 42$: b62c7013 508a16ab 7e056794 7e4d30b8 80a92a00 45459d97 423b26c0 9f5b9b6d

$t = 43$: 167361de b62c7013 508a16ab 7e056794 41dd3844 80a92a00 45459d97 423b26c0

$t = 44$: de71e2f2 167361de b62c7013 508a16ab ff61c636 41dd3844 80a92a00 45459d97

$t = 45$: 18f0d19d de71e2f2 167361de b62c7013 6b88472c ff61c636 41dd3844 80a92a00

$t = 46$: 165be9cd 18f0d19d de71e2f2 167361de a483f080 6b88472c ff61c636 41dd3844

$t = 47$: 13d82741 165be9cd 18f0d19d de71e2f2 a7802a4d a483f080 6b88472c ff61c636

$t = 48$: 017b9d99 13d82741 165be9cd 18f0d19d aeb10b60 a7802a4d a483f080 6b88472c

$t = 49$: 543c99a1 017b9d99 13d82741 165be9cd 16f134b6 aeb10b60 a7802a4d a483f080

$t = 50$: 758ca97a 543c99a1 017b9d99 13d82741 100cf2ea 16f134b6 aeb10b60 a7802a4d

$t = 51$: 81c1cde0 758ca97a 543c99a1 017b9d99 5c47eb7b 100cf2ea 16f134b6 aeb10b60

$t = 52$: b8d55619 81c1cde0 758ca97a 543c99a1 1c806a61 5c47eb7b 100cf2ea 16f134b6

$t = 53$: 1d6de87a b8d55619 81c1cde0 758ca97a 3443bed4 1c806a61 5c47eb7b 100cf2ea

$t = 54$: f907b313 1d6de87a b8d55619 81c1cde0 61a41711 3443bed4 1c806a61 5c47eb7b

$t = 55$: 9e57c4a0 f907b313 1d6de87a b8d55619 eec13548 61a41711 3443bed4 1c806a61

$t = 56$: 71629856 9e57c4a0 f907b313 1d6de87a 2f6c8c4e eec13548 61a41711 3443bed4

$t = 57$: 7c015a2c 71629856 9e57c4a0 f907b313 cb9d3dd0 2f6c8c4e eec13548 61a41711

$t = 58$: 921fccb6 7c015a2c 71629856 9e57c4a0 43d8a034 cb9d3dd0 2f6c8c4e eec13548

$t = 59$: e18f259a 921fccb6 7c015a2c 71629856 51e15869 43d8a034 cb9d3dd0 2f6c8c4e

$t = 60$: bcfce922 e18f259a 921fccb6 7c015a2c 962d8621 51e15869 43d8a034 cb9d3dd0

$t = 61$: f6f443f8 bcfce922 e18f259a 921fccb6 acc75916 962d8621 51e15869 43d8a034

$t = 62$: 86126910 f6f443f8 bcfce922 e18f259a 2fc08f85 acc75916 962d8621 51e15869

$t = 63$: 1bdc6f6f 86126910 f6f443f8 bcfce922 25d2430a 2fc08f85 acc75916 962d8621

Böylece ilk ve tek ileti bloğu $M^{(1)}$ in işlenmesi tamamlanmış olacaktır. Elde edilecek son hash değeri $H^{(1)}$:

$H_0^{(1)} =$ 6a09e667 + 1bdc6f6f = 85e655d6

$H_1^{(1)} =$ bb67ae85 + 86126910 = 417a1795

$H_2^{(1)} =$ 3c6ef372 + f6f443f8 = 3363376a

$H_3^{(1)} =$ a54ff53a + bcfce922 = 624cde5c

$H_4^{(1)} =$ 510e527f + 25d2430a = 76e09589

$H_5^{(1)} =$ 9b05688c + 2fc08f85 = cac5f811

$H_6^{(1)} =$ 1f83d9ab + acc75916 = cc4b32c1

$H_7^{(1)} =$ 5be0cd19 + 962d8621 = f20e533a

Dolgulanmış iletinin ikinci kelime bloğu $M^{(2)}$ ileti çizelgesine ait $W_0,\ldots,W_{15}$ kelime bloklarına atanır:

$W_0 =$ 00000000
$W_1 =$ 00000000
$W_2 =$ 00000000
$W_3 =$ 00000000
$W_4 =$ 00000000
$W_5 =$ 00000000
$W_6 =$ 00000000
$W_7 =$ 00000000
$W_8 =$ 00000000
$W_9 =$ 00000000
$W_{10} =$ 00000000
$W_{11} =$ 00000000
$W_{12} =$ 00000000
$W_{13} =$ 00000000
$W_{14} =$ 00000000
$W_{15} =$ 000001c0

Aşağıdaki çizelgede a,b,c,d,e,f,g,h nin daha önce belirtilen t=0 dan 63'e döngüsünden çıktıktan sonraki onaltılık değerleri verilmiştir.

	a	***b***	***c***	***d***	***e***	***f***	***g***	***h***
$t = 0$:	7c20c838	85e655d6	417a1795	3363376a	4670ae6e	76e09589	cac5f811	cc4b32c1
$t = 1$:	7c3c0f86	7c20c838	85e655d6	417a1795	8c51be64	4670ae6e	76e09589	cac5f811
$t = 2$:	fd1eebdc	7c3c0f86	7c20c838	85e655d6	af71b9ea	8c51be64	4670ae6e	76e09589
$t = 3$:	f268faa9	fd1eebdc	7c3c0f86	7c20c838	e20362ef	af71b9ea	8c51be64	4670ae6e
$t = 4$:	185a5d79	f268faa9	fd1eebdc	7c3c0f86	8dff3001	e20362ef	af71b9ea	8c51be64
$t = 5$:	3eeb6c06	185a5d79	f268faa9	fd1eebdc	fe20cda6	8dff3001	e20362ef	af71b9ea
$t = 6$:	89bba3f1	3eeb6c06	185a5d79	f268faa9	0a34df03	fe20cda6	8dff3001	e20362ef
$t = 7$:	bf9a93a0	89bba3f1	3eeb6c06	185a5d79	059abdd1	0a34df03	fe20cda6	8dff3001
$t = 8$:	2c096744	bf9a93a0	89bba3f1	3eeb6c06	abfa465b	059abdd1	0a34df03	fe20cda6
$t = 9$:	2d964e86	2c096744	bf9a93a0	89bba3f1	aa27ed82	abfa465b	059abdd1	0a34df03
$t = 10$:	5b35025b	2d964e86	2c096744	bf9a93a0	10e77723	aa27ed82	abfa465b	059abdd1

$t = 11$: 5eb4ec40 5b35025b 2d964e86 2c096744 e11b4548 10e77723 aa27ed82 abfa465b

$t = 12$: 35ee996d 5eb4ec40 5b35025b 2d964e86 5c24e2a2 e11b4548 10e77723 aa27ed82

$t = 13$: d74080fa 35ee996d 5eb4ec40 5b35025b 68aa893f 5c24e2a2 e11b4548 10e77723

$t = 14$: 0cea5cbc d74080fa 35ee996d 5eb4ec40 60356548 68aa893f 5c24e2a2 e11b4548

$t = 15$: 16a8cc79 0cea5cbc d74080fa 35ee996d 0fcb1f6f 60356548 68aa893f 5c24e2a2

$t = 16$: f16f634e 16a8cc79 0cea5cbc d74080fa 8b21cdc1 0fcb1f6f 60356548 68aa893f

$t = 17$: 23dcb6c2 f16f634e 16a8cc79 0cea5cbc ca9182d3 8b21cdc1 0fcb1f6f 60356548

$t = 18$: dcff40fd 23dcb6c2 f16f634e 16a8cc79 69bf7b95 ca9182d3 8b21cdc1 0fcb1f6f

$t = 19$: 76f1a2bc dcff40fd 23dcb6c2 f16f634e 0dc84bb1 69bf7b95 ca9182d3 8b21cdc1

$t = 20$: 20aad899 76f1a2bc dcff40fd 23dcb6c2 cc4769f2 0dc84bb1 69bf7b95 ca9182d3

$t = 21$: d44dc81a 20aad899 76f1a2bc dcff40fd 5bace62d cc4769f2 0dc84bb1 69bf7b95

$t = 22$: f13ae55b d44dc81a 20aad899 76f1a2bc 966aa287 5bace62d cc4769f2 0dc84bb1

$t = 23$: a4195b91 f13ae55b d44dc81a 20aad899 eddbd6ed 966aa287 5bace62d cc4769f2

$t = 24$: 4984fa79 a4195b91 f13ae55b d44dc81a a530d939 eddbd6ed 966aa287 5bace62d

$t = 25$: aa6cb982 4984fa79 a4195b91 f13ae55b 0b5eeea4 a530d939 eddbd6ed 966aa287

$t = 26$: 9450fbbc aa6cb982 4984fa79 a4195b91 09166dda 0b5eeea4 a530d939 eddbd6ed

$t = 27$: 0d936bab 9450fbbc aa6cb982 4984fa79 6e495d4b 09166dda 0b5eeea4 a530d939

$t = 28$: d958b529 0d936bab 9450fbbc aa6cb982 c2fa99b1 6e495d4b 09166dda 0b5eeea4

$t = 29$: 1cfa5eb0 d958b529 0d936bab 9450fbbc 6c49db9f c2fa99b1 6e495d4b 09166dda

$t = 30$: 02ef3a5f 1cfa5eb0 d958b529 0d936bab 5da10665 6c49db9f c2fa99b1 6e495d4b

$t = 31$: b0eab1c5 02ef3a5f 1cfa5eb0 d958b529 f6d93952 5da10665 6c49db9f c2fa99b1

$t = 32$: 0bfba73c b0eab1c5 02ef3a5f 1cfa5eb0 8b99e3a9 f6d93952 5da10665 6c49db9f

$t = 33$: 4bd1df96 0bfba73c b0eab1c5 02ef3a5f 905e44ac 8b99e3a9 f6d93952 5da10665

$t = 34$: 9907f1b6 4bd1df96 0bfba73c b0eab1c5 66c3043d 905e44ac 8b99e3a9 f6d93952

$t = 35$: ecde4e0d 9907f1b6 4bd1df96 0bfba73c 5dc119e6 66c3043d 905e44ac 8b99e3a9

$t = 36$: 2f11c939 ecde4e0d 9907f1b6 4bd1df96 fed4ce1d 5dc119e6 66c3043d 905e44ac

$t = 37$: d949682b 2f11c939 ecde4e0d 9907f1b6 32d99008 fed4ce1d 5dc119e6 66c3043d

$t = 38$: adca7a96 d949682b 2f11c939 ecde4e0d c6cce4ff 32d99008 fed4ce1d 5dc119e6

$t = 39$: 221b8a5a adca7a96 d949682b 2f11c939 0b82c5eb c6cce4ff 32d99008 fed4ce1d

$t = 40$: 12d97845 221b8a5a adca7a96 d949682b e4213ca2 0b82c5eb c6cce4ff 32d99008

$t = 41$: 2c794876 12d97845 221b8a5a adca7a96 ff6759ba e4213ca2 0b82c5eb c6cce4ff

$t = 42$: 8300fca2 2c794876 12d97845 221b8a5a e0e3457c ff6759ba e4213ca2 0b82c5eb

$t = 43$: f2ad6322 8300fca2 2c794876 12d97845 cc48c7f3 e0e3457c ff6759ba e4213ca2

$t = 44$: 0f154e11 f2ad6322 8300fca2 2c794876 6f9517cb cc48c7f3 e0e3457c ff6759ba

$t = 45$: 104a7db4 0f154e11 f2ad6322 8300fca2 5348e8f6 6f9517cb cc48c7f3 e0e3457c

$t = 46$: 0b3303a7 104a7db4 0f154e11 f2ad6322 bbe1c39a 5348e8f6 6f9517cb cc48c7f3

$t = 47$: d7354d5b 0b3303a7 104a7db4 0f154e11 aad55b6b bbe1c39a 5348e8f6 6f9517cb

$t = 48$: b736d7a6 d7354d5b 0b3303a7 104a7db4 68f25260 aad55b6b bbe1c39a 5348e8f6

$t = 49$: 2748e5ec b736d7a6 d7354d5b 0b3303a7 d4b58576 68f25260 aad55b6b bbe1c39a

$t = 50$: d8aabcf9 2748e5ec b736d7a6 d7354d5b 27844711 d4b58576 68f25260 aad55b6b

$t = 51$: 1a6bcf6a d8aabcf9 2748e5ec b736d7a6 ff5e99d0 27844711 d4b58576 68f25260

$t = 52$: 4eca6fa0 1a6bcf6a d8aabcf9 2748e5ec 989ed071 ff5e99d0 27844711 d4b58576

$t = 53$: ec02560a 4eca6fa0 1a6bcf6a d8aabcf9 7151df8e 989ed071 ff5e99d0 27844711

$t = 54$: d9f0c115 ec02560a 4eca6fa0 1a6bcf6a 624150c4 7151df8e 989ed071 ff5e99d0

$t = 55$: 92952710 d9f0c115 ec02560a 4eca6fa0 226806d6 624150c4 7151df8e 989ed071

$t = 56$: 20d4d0e4 92952710 d9f0c115 ec02560a 4e515a4d 226806d6 624150c4 7151df8e

$t = 57$: 4348eb1f 20d4d0e4 92952710 d9f0c115 c21eddf9 4e515a4d 226806d6 624150c4

$t = 58$: 286fe5f0 4348eb1f 20d4d0e4 92952710 54076664 c21eddf9 4e515a4d 226806d6

$t = 59$: 1c4cddd9 286fe5f0 4348eb1f 20d4d0e4 f487a853 54076664 c21eddf9 4e515a4d

$t = 60$: a9f181dd 1c4cddd9 286fe5f0 4348eb1f 27ccb387 f487a853 54076664 c21eddf9

$t = 61$: b25cef29 a9f181dd 1c4cddd9 286fe5f0 2aa1bb13 27ccb387 f487a853 54076664

$t = 62$: 908c2123 b25cef29 a9f181dd 1c4cddd9 9a392956 2aa1bb13 27ccb387 f487a853

$t = 63$: 9ea7148b 908c2123 b25cef29 a9f181dd 2c5c4ed0 9a392956 2aa1bb13 27ccb387

Böylece ikinci ve son ileti bloğu $M^{(2)}$ in işlenmesi tamamlanmış olacaktır. Elde edilecek son hash değeri $H^{(2)}$:

$H_0^{(2)}$ = 85e655d6 + 9ea7148b = 248d6a61

$H_1^{(2)}$ = 417a1795 + 908c2123 = d20638b8

$H_2^{(2)}$ = 3363376a + b25cef29 = e5c02693

$H_3^{(2)}$ = 624cde5c + a9f181dd = 0c3e6039

$H_4^{(2)}$ = 76e09589 + 2c5c4ed0 = a33ce459

$H_5^{(2)}$ = cac5f811 + 9a392956 = 64ff2167

$H_6^{(2)}$ = cc4b32c1 + 2aa1bb13 = f6ecedd4

$H_7^{(2)}$ = f20e533a + 27ccb387 = 19db06c1

ve elde edilen 256-ikillik ileti özeti:

248d6a61 d20638b8 e5c02693 0c3e6039 a33ce459 64ff2167 f6ecedd4 19db06c1

A.2.3 Uzun İleti

M, iletisi 1000000 kez “a” karakterinin tekrarı ile oluşturulmuş olsun. SHA-256 ileti özeti:
cdc76e5c 9914fb92 81a1c7e2 84d73e67 f1809a48 a497200e 046d39cc c7112cd0

EK.B HMAC Test Vektörleri

Bu örnekler HMAC uygulamalarının doğruluklarını desteklemek için sunulmuştur. Bu örneklerde SHA-1 hash fonksiyonu kullanılmıştır.

B.1 64-Bayt Anahtarlı SHA-1

Text: "Sample #1"

Key:

```
00010203 04050607 08090a0b 0c0d0e0f
10111213 14151617 18191a1b 1c1d1e1f
20212223 24252627 28292a2b 2c2d2e2f
30313233 34353637 38393a3b 3c3d3e3f
```

K_0:

```
00010203 04050607 08090a0b 0c0d0e0f
10111213 14151617 18191a1b 1c1d1e1f
20212223 24252627 28292a2b 2c2d2e2f
30313233 34353637 38393a3b 3c3d3e3f
```

$K_0 \oplus$ ipad:

```
36373435 32333031 3e3f3c3d 3a3b3839
26272425 22232021 2e2f2c2d 2a2b2829
16171415 12131011 1e1f1c1d 1a1b1819
06070405 02030001 0e0f0c0d 0a0b0809
```

(Key $\oplus$ ipad)||text:

```
36373435 32333031 3e3f3c3d 3a3b3839
26272425 22232021 2e2f2c2d 2a2b2829
16171415 12131011 1e1f1c1d 1a1b1819
06070405 02030001 0e0f0c0d 0a0b0809
53616d70 6c652023 31
```

Hash((Key $\oplus$ ipad)||text):

```
bcc2c68c abbbf1c3 f5b05d8e 7e73a4d2
7b7e1b20
```

$K_0 \oplus$ opad:

```
5c5d5e5f 58595a5b 54555657 50515253
4c4d4e4f 48494a4b 44454647 40414243
7c7d7e7f 78797a7b 74757677 70717273
```

```
6c6d6e6f 68696a6b 64656667 60616263
```

($K_0 \oplus$ opad) || Hash(($K_0 \oplus$ ipad)||text):

```
5c5d5e5f 58595a5b 54555657 50515253
4c4d4e4f 48494a4b 44454647 40414243
7c7d7e7f 78797a7b 74757677 70717273
6c6d6e6f 68696a6b 64656667 60616263
bcc2c68c abbbf1c3 f5b05d8e 7e73a4d2
7b7e1b20
```

HMAC(Key, Text) = Hash(($K_0 \oplus$ opad) || Hash(($K_0 \oplus$ ipad)||text)):

```
4f4ca3d5 d68ba7cc 0a1208c9 c61e9c5d
a0403c0a
```

20-byte HMAC(Key, Text):

```
4f4ca3d5 d68ba7cc 0a1208c9 c61e9c5d
a0403c0a
```

B.2 20-Bayt Anahtarlı SHA-1

Text: "Sample #2"

Key:

```
30313233 34353637 38393a3b 3c3d3e3f
40414243
```

K_0:

```
30313233 34353637 38393a3b 3c3d3e3f
40414243 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
```

$K_0 \oplus$ ipad:

```
06070405 02030001 0e0f0c0d 0a0b0809
76777475 36363636 36363636 36363636
36363636 36363636 36363636 36363636
36363636 36363636 36363636 36363636
```

($K_0 \oplus$ ipad)||text:

```
06070405 02030001 0e0f0c0d 0a0b0809
76777475 36363636 36363636 36363636
36363636 36363636 36363636 36363636
36363636 36363636 36363636 36363636
53616d70 6c652023 32800000 00000000
00000000 00000000 00000000 00000000
```

```
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000248
```

Hash((Key ⊕ ipad)||text):

```
74766e5f 6913e8cb 6f7f108a 11298b15
010c353a
```

K_0 ⊕ opad:

```
6c6d6e6f 68696a6b 64656667 60616263
1c1d1e1f 5c5c5c5c 5c5c5c5c 5c5c5c5c
5c5c5c5c 5c5c5c5c 5c5c5c5c 5c5c5c5c
5c5c5c5c 5c5c5c5c 5c5c5c5c 5c5c5c5c
```

(K_0 ⊕ opad) || Hash((Key ⊕ ipad)||text):

```
6c6d6e6f 68696a6b 64656667 60616263
1c1d1e1f 5c5c5c5c 5c5c5c5c 5c5c5c5c
5c5c5c5c 5c5c5c5c 5c5c5c5c 5c5c5c5c
5c5c5c5c 5c5c5c5c 5c5c5c5c 5c5c5c5c
74766e5f 6913e8cb 6f7f108a 11298b15
010c353a
```

HMAC(Key, Text) = Hash((K_0 ⊕ opad) || Hash((Key ⊕ ipad)||text)):

```
0922d340 5faa3d19 4f82a458 30737d5c
c6c75d24
```

20-byte HMAC(Key, Text):

```
0922d340 5faa3d19 4f82a458 30737d5c
c6c75d24
```

B.3 100-Bayt Anahtarlı SHA-1

Text: "Sample #3"

Key:

```
50515253 54555657 58595a5b 5c5d5e5f
60616263 64656667 68696a6b 6c6d6e6f
70717273 74757677 78797a7b 7c7d7e7f
80818283 84858687 88898a8b 8c8d8e8f
90919293 94959697 98999a9b 9c9d9e9f
a0a1a2a3 a4a5a6a7 a8a9aaab acadaeaf
b0b1b2b3
```

Hash(Key):

```
a4aabe16 54e78da4 40d2a403 015636bf
4bb2f329
```

K_0:

```
a4aabe16 54e78da4 40d2a403 015636bf
4bb2f329 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
```

$K_0 \oplus$ ipad:

```
929c8820 62d1bb92 76e49235 37600089
7d84c51f 36363636 36363636 36363636
36363636 36363636 36363636 36363636
36363636 36363636 36363636 36363636
```

(Key $\oplus$ ipad)||text:

```
929c8820 62d1bb92 76e49235 37600089
7d84c51f 36363636 36363636 36363636
36363636 36363636 36363636 36363636
36363636 36363636 36363636 36363636
53616d70 6c652023 33
```

Hash((Key $\oplus$ ipad)||text):

```
d98315c4 2152bea0 d057de97 84427676
2a1a5576
```

$K_0 \oplus$ opad:

```
f8f6e24a 08bbd1f8 1c8ef85f 5d0a6ae3
17eeaf75 5c5c5c5c 5c5c5c5c 5c5c5c5c
5c5c5c5c 5c5c5c5c 5c5c5c5c 5c5c5c5c
5c5c5c5c 5c5c5c5c 5c5c5c5c 5c5c5c5c
```

($K_0 \oplus$ opad) || Hash((Key $\oplus$ ipad)||text):

```
f8f6e24a 08bbd1f8 1c8ef85f 5d0a6ae3
17eeaf75 5c5c5c5c 5c5c5c5c 5c5c5c5c
5c5c5c5c 5c5c5c5c 5c5c5c5c 5c5c5c5c
5c5c5c5c 5c5c5c5c 5c5c5c5c 5c5c5c5c
d98315c4 2152bea0 d057de97 84427676
2a1a5576
```

HMAC(Key, Text) = Hash(($K_0 \oplus$ opad) || Hash((Key $\oplus$ ipad)||text)):

```
bcf41eab 8bb2d802 f3d05caf 7cb092ec
f8d1a3aa
```

20-byte HMAC(Key, Text):

```
bcf41eab 8bb2d802 f3d05caf 7cb092ec
f8d1a3aa
```

B.4 49-Bayt Anahtarlı SHA-1, 12-Bayt Kısaltılmış HMAC

Text: "Sample #4"

Key:

```
70717273 74757677 78797a7b 7c7d7e7f
80818283 84858687 88898a8b 8c8d8e8f
90919293 94959697 98999a9b 9c9d9e9f
a0
```

K_0:

```
70717273 74757677 78797a7b 7c7d7e7f
80818283 84858687 88898a8b 8c8d8e8f
90919293 94959697 98999a9b 9c9d9e9f
a0000000 00000000 00000000 00000000
```

$K_0 \oplus$ ipad:

```
46474445 42434041 4e4f4c4d 4a4b4849
b6b7b4b5 b2b3b0b1 bebfbcbd babbb8b9
a6a7a4a5 a2a3a0a1 aeafacad aaaba8a9
96363636 36363636 36363636 36363636
```

(Key $\oplus$ ipad)||text:

```
46474445 42434041 4e4f4c4d 4a4b4849
b6b7b4b5 b2b3b0b1 bebfbcbd babbb8b9
a6a7a4a5 a2a3a0a1 aeafacad aaaba8a9
96363636 36363636 36363636 36363636
53616d70 6c652023 34
```

Hash((Key $\oplus$ ipad)||text):

```
bf1e889d 876c34b7 bef3496e d998c8d1
16673a2e
```

$K_0 \oplus$ opad:

```
2c2d2e2f 28292a2b 24252627 20212223
dcdddedf d8d9dadb d4d5d6d7 d0d1d2d3
cccdcecf c8c9cacb c4c5c6c7 c0c1c2c3
fc5c5c5c 5c5c5c5c 5c5c5c5c 5c5c5c5c
```

($K_0 \oplus$ opad) || Hash((Key $\oplus$ ipad)||text):

```
2c2d2e2f 28292a2b 24252627 20212223
dcdddedf d8d9dadb d4d5d6d7 d0d1d2d3
cccdcecf c8c9cacb c4c5c6c7 c0c1c2c3
fc5c5c5c 5c5c5c5c 5c5c5c5c 5c5c5c5c
bf1e889d 876c34b7 bef3496e d998c8d1
```

```
16673a2e
```

HMAC(Key, Text) = Hash((K0 ⊕ opad) || Hash((Key ⊕ ipad)||text)):

```
9ea886ef e268dbec ce420c75 24df32e0
751a2a26
```

12-byte HMAC(Key, Text):

```
9ea886ef e268dbec ce420c75
```

EK.C RC6 Blok Şifreleme Algoritması Test Vektörleri

C.1 RC6 Algoritması Şifreleme/Deşifreleme Test Vektörleri

Bu bölümde RC6 algoritmasına ait test vektörlerine yer verilmiştir.

düz metin	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
anahtar	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
şifreli metin	8f c3 a5 36 56 b1 f7 78 c1 29 df 4e 98 48 a4 1e

düz metin	02 13 24 35 46 57 68 79 8a 9b ac bd ce df e0 f1
anahtar	01 23 45 67 89 ab cd ef 01 12 23 34 45 56 67 78
şifreli metin	52 4e 19 2f 47 15 c6 23 1f 51 f6 36 7e a4 3f 18

düz metin	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
anahtar	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
şifreli metin	6c d6 1b cb 19 0b 30 38 4e 8a 3f 16 86 90 ae 82

düz metin	02 13 24 35 46 57 68 79 8a 9b ac bd ce df e0 f1
anahtar	01 23 45 67 89 ab cd ef 01 12 23 34 45 56 67 78 89 9a ab bc cd de ef f0
şifreli metin	68 83 29 d0 19 e5 05 04 1e 52 e9 2a f9 52 91 d4

düz metin	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
anahtar	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
şifreli metin	8f 5f bd 05 10 d1 5f a8 93 fa 3f da 6e 85 7e c2

düz metin	02 13 24 35 46 57 68 79 8a 9b ac bd ce df e0 f1
anahtar	01 23 45 67 89 ab cd ef 01 12 23 34 45 56 67 78 89 9a ab bc cd de ef f0 10 32 54 76 98 ba dc fe
şifreli metin	c8 24 18 16 f0 d7 e4 89 20 ad 16 a1 67 4e 5d 48

KAYNAKLAR

FIPS PUB180-2, Federal Information Processing Standards Publication, Secure Hash Standard, Ağustos 2002

FIPS PUB198, Federal Information Processing Standards Publication, The Keyed- Hash Message Authentication Code (HMAC), Mart 2002

Lee, A., Guideline for Implementing Cryptography in the Federal Government, NIST Special Publication 800-21, 1999

Menezes J.A, Van Oorschot P.C, Vanstone S.A, Handbook of Applied Cryptography, CRC Press, 1997

Pratt W.K., Digital Image Processing: PIKS Inside, Third Edition, John Wiley & Sons, 2001

Rivest R.L., Robshaw, M.J.B., Sydney R., Yin R.L., The RC6 Block Cipher, (http://www.rsa.security.com/rsalabs/rc6),1998

Schneier, B., Applied Cryptography, Second Edition: Protocols, Algorithms, and Source Code in C , John Wiley & Sons, 1996

Stallings W., Cryptography and Network Security: Principles and Practice, Second Edition, Prentice Hall, 1998

Uğur A., Aydos M., Güvenli Haberleşme Protokollerinde Görüntü Uzayının Kullanımı, Ulusal Ağ ve Bilgi Güvenliği Sempozyumu, Haziran 2005

ÖZGEÇMİŞ

Adı, soyadı: Alper UĞUR

Ana adı: Abugül

Baba adı: H. Hüseyin

Doğum yeri ve tarihi: Sivas, 1978.

Lisans eğitimi ve mezuniyet tarihi: Yeditepe Üniversitesi, Bilgisayar Mühendisliği, 2001.

Çalıştığı yer: Pamukkale Üniversitesi, Bilgisayar Mühendisliği Bölümü

Bildiği yabancı dil, aldığı belgeler: İngilizce

Mesleki Etkinlikleri:

- A. Uğur and M. Aydos. Sayısal İçerik Güvenliği: Uygulama Yöntemleri. BİLGİTEK Kongresi, 2004
- A. Uğur and M. Aydos. Digital Content Protection Via Intelligent Agents In Campus Networks, IV. Uluslar Arası İleri Teknolojiler Sempozyumu, 2005
- A. Uğur and M. Aydos. Güvenli Haberleşme Protokollerinde Görüntü Uzayının Kullanımı, Ağ Ve Bilgi Güvenliği Ulusal Sempozyumu ve Sergisi, 2005
- A. Uğur and M. Aydos. Taşınabilir Etmenlerle Çok Katılımcılı Sözleşmelerin Sayısal İmzalanması, Elektrik-Elektronik-Bilgisayar Mühendisliği 11. Ulusal Kongresi, 2005