T C
ÇANAKKALE ONSEKİZ MART ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

AÇIK ANAHTARLI KRİPTOGRAFİ VE AĞ GÜVENLİK UYGULAMALARI

YÜKSEK LİSANS TEZİ

Ercan ÇAĞLAR

ÇANAKKALE-2004

T C
ÇANAKKALE ONSEKİZ MART ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

AÇIK ANAHTARLI KRİPTOGRAFİ VE AĞ GÜVENLİK UYGULAMALAR

YÜKSEK LİSANS TEZİ

Hazırlayan : Ercan ÇAĞLAR

Danışman : Doç. Dr. Mammadagha MAMMADOV

ÇANAKKALE-2004

Çanakkale Onsekiz Mart Üniversitesi Fen Bilimleri Enstitüsü Müdürlüğüne,

Bu araştırma jürimiz tarafından Bilgisayar Mühendisliği Anabilim Dalında Yüksek Lisans Tezi olarak kabul edilmiştir.

Başkan: Yrd. Doç. Dr. Bülent GÜLER .....

Üye : Doç. Dr. Mammadagha MAMMADOV .....

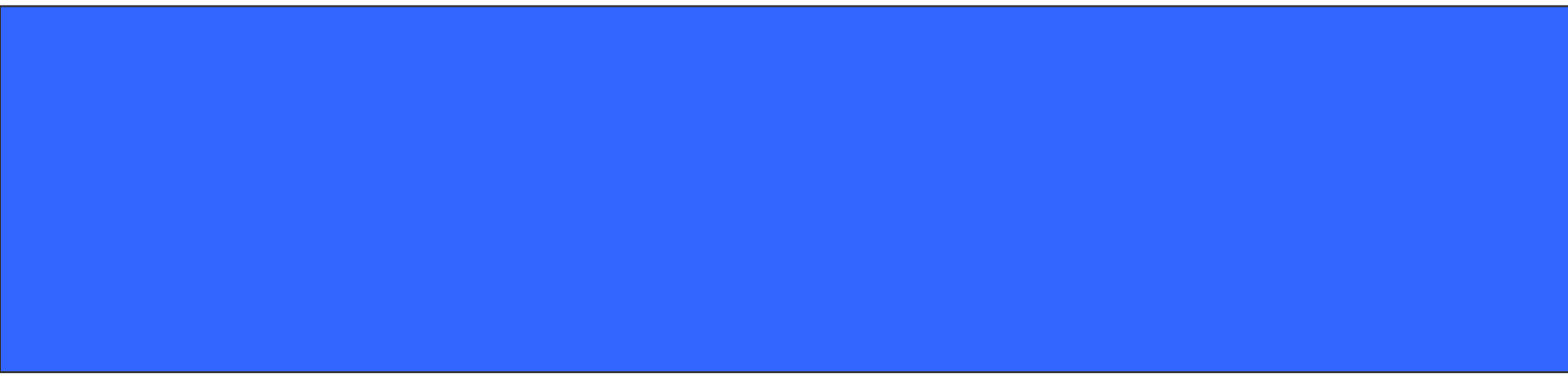
Üye : Doç. Dr. M. Ali SALAHLI .....

Kod No:

Yukarıdaki imzaların adı geçen öğretim üyelerine ait olduğunu onaylarım.

Enstitü Müdürü

Prof. Dr. Mehmet Emin ÖZEL



İÇİNDEKİLER

ÖZ

Günümüzde teknolojinin ilerlemesiyle birlikte bilgisayar kullanımı yaygınlaştı. Kullanıcıların yeni ihtiyaçları ortaya çıktı. Kullanıcılar, internet vasıtasıyla alışveriş, haberleşme ve finanssal işlemler gibi bir çok yeni olanaklara sahip oldular. Bilgisayar ağlarının kullanımının artması çıkar sağlamak isteyenlerin dikkatini çekti. İnsanların gizli bilgilerine ulaşmak, banka hesapları üzerinde işlem yapabilmek gibi uğraşlar içerisine girdiler. Bu sistemleri korumakla yükümlü olanlar, kriptografi tekniklerini kullanarak güven çözümleri üretmektedirler. Bu güven yöntemlerinden açık anahtarlı kriptografiyi temel alarak tezimi hazırladım.

Kriptografinin genel olarak amacına değinilmiştir. Açık anahtarlı kriptografinin tanımı yapılmıştır. İhtiyaçları ve algoritması üzerine bilgiler verilmiştir. Diffie-Hellman ve RSA kripto sistemleri incelenmiştir. RSA kripto sisteminde anahtar üretimi ve güvenliği incelenmiştir. Kripto sistemlerinin tarihsel gelişimi gözden geçirilmiştir. Anahtar dağıtım senaryoları incelenmiştir. Daha sonra ağ üzerindeki güvenlik uygulamalarına geçilmiştir. X509 Kimlik doğrulama servisi ve sertifikalar üzerinde tartışılmıştır. PGP üzerinde durulmuştur. IP ve WEB güvenliği konularına değinilmiştir. RSA algoritmasını temel alan yazılımlar geliştirilmiştir.

Anahtar Kelimeler : Kriptografi, Açık Anahtar Kripto Sistemler, RSA, Anahtar Yönetimi, Ağ Güvenliği

ABSTRACT

Today with the developing technology computer usage is increased. The users have possibilities for shopping, communication and financial procedures over Internet. The usage of computer networks called attention the person who wants to benefit. The hackers try to reach secret information and manage bank accounts. The cryptologists create security solutions with using cryptography techniques. The public key cryptography is the one of security methods and I prepared my thesis with using this method.

It was mentioned to Cryptography's general aim. Public key cryptography's definition was made. Information was given about its necessities and algorithms. Diffie-Hellman and RSA cryptosystems were examined. At RSA cryptosystem, key production and security was examined. The historical developments of cryptosystems were glanced over. Key distribution scenarios were examined. Then was passed to the security application over network. X509 authentication service and certificates were argued. PGP was researched. The items of IP and WEB security were mentioned. The software, which uses RSA algorithm, was developed.

Key Words: Cryptography, Public-Key Cryptosystems, RSA, Key Management, and Network Security.

KISALTMALAR :

ASCII	**:**	Bilgi Değişimi için Amerikan Standart Kodları
SPI	**:**	Güvenli Parametreler İndeksi
IKE	**:**	İnternet Anahtar Değişimi
AH	**:**	Doğrulama Başlığı
URL	**:**	Evrensel Kaynak Konumlandırıcısı
SO	**:**	Sertifika Otoritesi
IEEE	**:**	Elektrik ve Elektronik Mühendisleri Enstitüsü
IPSEC	**:**	İnternet Protokol Güvenliği
ITU	**:**	Uluslar arası Telekomünikasyon Birliği
UDP	**:**	Kullanıcı Veri Taşıma Paketi Protokolü
CCITT	**:**	Uluslar arası Telgraf ve Telefon Danışma Kurulu
CRL	**:**	Sertifika Geri Gönderim Listeleri
DES	**:**	Veri Şifreleme Standardı
DSS	**:**	Dijital İmza Standardı
HTTP	**:**	Hipertext Transfer Protokolü
IAB	**:**	İnternet Mimari Yönetim Kurulu
IDEA	**:**	Uluslar arası Veri Şifreleme Algoritması
IETF	**:**	İnternet Mühendislik Özel Görev Kuvveti
ISAKMP	**:**	İnternet Güvenlik Birliği ve Anahtar Yönetim Protokolü
ISO	**:**	İnternet Standart Organizasyonu
KDC	**:**	Anahtar Dağıtım Merkezi
LAN	**:**	Yerel Alan Ağı
MD5	**:**	Mesaj Derleme versiyon 5
NIST	**:**	Ulusal Standartlar ve Teknoloji Enstitüsü
PEM	**:**	Posta Gizliliğini Yükselmek
PGP	**:**	Oldukça İyi Gizlilik
RSA	**:**	Rivest-Shamir-Adleman (RSA algoritmasını tasarlayan kişiler)
SET	**:**	Güvenli Elektronik İşlem
SHA	**:**	Güvenli Hash Algoritması
SMTP	**:**	Basit Posta Transferi Protokolü
SSL	**:**	Güvenli Soket Düzeyi
TCP/IP	**:**	İletim Kontrol Protokol/İnternet Protokol
TLS	**:**	Ulaşım Düzeyi Güvenliği
WAN	**:**	Geniş Alan Ağı

TABLOLAR

ŞEKİLLER

1. GİRİŞ:

Bilgilerinizi paylaşıyor musunuz? O zaman kriptografiye ihtiyacınız var demektir. Gizlilik ihtiyacının ortaya çıkması ile birlikte kriptografi doğmuştur. Şimdiki teknoloji harikaları kullanılmasa da, mesajların rakiplerden korunması sağlanıyordu. Düşününki iletişim doğrudan yüz yüze yapılıyor. İletişim için herhangi bir araç kullanılmıyor. Bu durumda kendi düşüncelerimizi beynimizde saklayıp, muhatabımıza başkasının olmadığı gizli bir ortamda düşüncelerimizi aktarabiliriz. Yazının icat edilmesi ile birlikte insanlar iletişim için kağıt, kalem gibi araçlar kullanmaya başladı. İletişimin doğrudan yüz yüze değil, yazılı mesajlarla sağlanması güvenlik açıklarının ortaya çıkmasına neden oldu. Bunun en korkulanı mesajın rakiplerin eline geçme riskidir. Bunun üzerine insanlar mesajlarını şifrelemeye başladı. Bu şekilde mesaj ele geçse bile, rakip tarafından anlaşılmaması sağlandı. Aslında gizlilik insanların doğasında olan bir olgudur. İlkokul sıralarına dönelim. Alfabedeki harfleri karıştırarak yazmış olduğumuz gizli mesajları ders aralarında bir birimize iletirdik. Aslında bu yapmış olduğumuz Sezar şifresinin ta kendisiydi. Basit bir örnekle sezar şifresine bakalım. Burada amaç alfabedeki harflerin yerini değiştirmektir. Her harfin kendisinden 3 sonra gelen harfle yer değiştirilmesi sonucu şifreli alfabe oluşturulur.

Alfabe	A	B	C	Ç	D	E	F	G	Ğ	H	I	İ	J	K	L	M	N	O	Ö	P	R	S	Ş	T	U	Ü	V	Y	Z
Şifreli Alfabe	Ç	D	E	F	G	Ğ	H	I	İ	J	K	L	M	N	O	Ö	P	R	S	Ş	T	U	Ü	V	Y	Z	A	B	C

Alfabemizi oluşturduk. Şimdide kendimize mesaj belirleyip onu şifreleyelim.

Orijinal Mesaj : ÇANAKKALE ONSEKİZ MART ÜNİVERSİTESİ

Şifreli Mesaj : FÇPÇNNÇOĞ RPUĞNLC ÖÇTV ZPLAĞTULVĞUL

Şifreli mesajı alan kişi şifreli alfabedeki harflerin alfabedeki karşılığına göre mesajı deşifreler. Böylece orijinal mesajı elde eder.

Yunanca Kryptos Logos (Gizli kelime) tamlamasından gelen Kriptoloji, toplumda bir haberleşmenin gizli tutulabilmesi üzerinde çalışan bilim olarak düşünülse de aslında anlamı daha geniştir, çözülebilmesi çok zor matematik problemleri ve mekanizmaları inceleyen Kriptografiyi ve bu problemleri ve mekanizmaları çözmeyi hedefleyen ve saldırıları belirleyen Kriptoanalizi içerir. Şifreleme (Encryption), Veriyi bir anahtarla şifrelemeye verilen addır. Hedef, veriyi gerekli anahtar olmadan

çözülebilmesi imkansıza mümkün olduğunca yakın şekilde kodlamaktır. Şifre Çözme ise (Decryption) şifrelenmiş veriyi çözüp eski haline getirme işlemidir. Kriptolojide en çok kullanılan kelimelerden olan anahtar ise bir metni şifrelemekte veya açmakta kullanılan veri parçasına (sayı, kelime veya herhangi bir sayısal veri parçası) verilen isimdir.

Başlangıcı itibariyle daha çok askeri uygulamalarda kullanılmış. Günümüzde ise İnternet uygulamalarının hız kazanması kriptografinin Web güvenliği alanında uygulanabilirliğini kanıtladı. Özellikle elektronik ticaret, online banka işlemleri gibi birçok dinamik uygulamada bilginin saklanması şart. Kriptografi bu anlamda en uygun teknikleri öneriyor.

1977 yılında ABD hükümeti IBM tarafından geliştirilen bir şifreleme tekniğini standart olarak seçti. DES (Data Encryption Standart) olarak bilinen bu sistem gizli anahtar, simetrik kriptosistemiydi ve bugüne kadar en çok kullanılan kriptosistemlerden birisi oldu. 64 bitlik veri bloklarını 54 bitlik bir anahtar aracılığı ile şifreleyen DES geliştirildi.

ABD'de NIST (National Institute of Standarts and Technology) tarafından resmi makamlarca kullanılacak kriptosistemler beş yıllık süreler için standart olarak belirlenmektedir. DES'in standart olarak kullanım süresi ise en son 1993 yılında beş yıl daha uzatılmıştı. Gelecekte daha iyi bir sisteme geçilmesini isteyen NIST, bu konuda dijital imzaları (DSS) ve bilgi kodlanmasını içeren Capstone projesini başlattı.

Konu ile ilgili NIST, NSA (National Security Agency) isimli ABD devlet kuruluşuyla da yardımlaştı. NSA 1950'li yılların başlarında kurulmuş olan ve çok uzun süre gizli tutulan bir devlet kuruluşuydu. Görevi gereği ABD'nin tüm dış iletişimlerini kontrolü altında tutarak (telefon konuşmaları, email yazışmaları gibi) ülkenin güvenliği ile alakalı olanları ayırmaktadır.

1977 yılında yapılan önemli çalışmalardan biri de. Ron Rivest, Adi Shamir ve Leonard Adleman tarafından RSA isimli kriptosisteminin icadı idi. Çok karmaşık olmayan mod ve üs hesaplamalarına dayanan matematiksel işlemlerin sonucunda iki büyük asal sayıdan birer tane açık ve kişisel anahtar üreten bu kriptosistemin en büyük özelliklerinden biri kişisel anahtarın açık anahtarı oluşturan parçalardan üretilmesinin olanak dışı olmasıydı.

RSA, DES alternatifi olarak üretilmişti. Hatta DES'in daha verimli kullanımını sağlamaktadır. RSA ile bir verinin tamamını şifrelemek, gereken işlemlerin çok olması nedeniyle oldukça uzun bir zaman alıyordu. Ancak, ondan çok daha hızlı DES ile verinin şifrelenmesi ve daha sonra DES anahtarının RSA ile şifrelenerek şifreli veriye eklenmesi, işlemi oldukça hızlandırmaktadır. Fakat güvenliğin çok önemli olduğu durumlarda sadece RSA şifrelemeleri kullanılmaktadır.

RSA'ya alternatif olması amacıyla bir çok şifreleme sistemi üretildi; fakat bunlardan bir kısmı kırıldı. ElGamal tarafından üretilen ve Schnorr tarafından geliştirilen bir kriptosistem NIST'in projesinde yer alan DSS imza sistemini oluşturdu. Fakat bu sistem oldukça fazla eleştiri aldı; çünkü DSS, pek fazla test edilmemiş ve kırılamaz olduğu yeterince kanıtlanmamış bir kriptosistemdi.

Fakat, RSA'nin en büyük avantajlarından birisi hem anahtarları, hem de dijital imzayı ayni anda üretebilme kapasitesine sahip olan diğer sistemler RSA'nin yakaladığı güvenilirliği yakalayamadılar. RSA bu gücü sayesinde Unix, Linux sistemlerin neredeyse tamamında ve Microsoft, Novell, Apple tarafından kullanılmaktadır. ISO (Internet Standarts Organization) ve CCITT (Consultative Committee in International Telegraphy and Telephony) tarafından standart bir kriptosistem olarak kabul edilmiştir. Internet'te PEM (Privacy Enhanced Mail) ve PGP tarafından da kullanılmaktadır. Fakat ABD hükümeti, NSA'ya ABD dışına çıkacak kriptosistemleri kontrol yetkisini tanıdı ve NSA da neredeyse tüm önemli kriptosistemlerin ABD dışına çıkartılmasını yasakladı.

Biz öncelikle açık anahtarlı kriptografi üzerine değineceğiz. Daha sonra açık anahtarlı kriptosistemlerden Diffie-Helman ve RSA ‘nın üzerinde duracağı. Çalışmalarımız daha çok RSA kriptosistemi üzerine olacak. Anahtar dağıtımı problemine ilişkin senaryolar oluşturacağız. Açık anahtarlı kriptografini kullanıldığı ağ güvenlik uygulamalarına değindikten sonra RSA algoritmasını kullanan yazılım geliştireceğiz. Öncelikle Delphi de hazırlanmış yerel bir uygulama incelenecek. Daha sonra internet üzerinden mesaj alış verişini sağlayan bir web sayfası tasarımı üzerinde duracağız.

2. AÇIK ANAHTARLI KRİPTOGRAFİ:

Kriptografinin tarihindeki en büyük olay olarak Açık anahtarlı kriptografiyi gösterebiliriz. Kriptografinin keşfinden beri bütün kriptosistemler, yerine koyma metodunu ve permütasyonun basit araçlarını kullanmaktaydı. Şifreleme ve deşifreleme işlemlerini gerçekleştiren makinenin geliştirilmesi ile geleneksel kriptografide büyük bir gelişme meydana geldi. Elektromekanik dönüşüm karmaşık şifre sistemlerinin gelişmesini sağladı. Bilgisayarın geçerliliği ile birlikte bir çok karmaşık yapılı sistem tasarlandı. Bunlardan en önemlisi IBM tarafından yaptırılan ve DES olarak sonuçlandırılan LUCIFER projesidir. Dönüşüm makineleri ve DES'in her ikisi de kayda değer bir ilerleme göstermesine rağmen hala yerine koyma metodunu ve permütasyonu kullanıyordu. Buda onların güvenilirliğini azaltıyordu.

70'lerin başında sadece klasik blok kriptografi sistemleri bilinmekteydi. Ama ticari firmalar tarafından kabul görmediler. Sonraki 10 yıl içerisinde büyük gelişmeler oldu. Açık anahtarlı kriptografi Diffie, Merklle ve Hellman tarafından keşfedildi. Açık anahtarlı kriptografiyi, diğer kriptografi sistemlerinin hepsinden ayıran en önemli özellik; Açık anahtarlı kriptografinin algoritması, yerine koyma metodu ve permütasyondan daha çok matematiksel işlemlere dayanmasıdır. En önemlisi açık anahtar kriptografisi iki ayrı anahtar kullanımı gerektirir. Bu da asimetrik olmasını sağlar. Böylece tek anahtar kullanılan simetrik geleneksel kriptografiden daha güvenli olur . Güvenli olmasının nedeni şifre çözmek için kullanılan anahtarın paylaşılmamasıdır. İki anahtar kullanımı, anahtar dağıtımı ve kimlik doğrulama gibi güven ve gizlilik gerektiren büyük sonuçları meydana getirmiştir. Açık anahtar kriptografi sistemleri, şifreleme için açık anahtarı ve deşifreleme için özel anahtarı kullanır.

2.1. AÇIK ANAHATARLI KRİPTOSİSTEM:

Açık anahtarlı algoritmalar, şifreleme için bir anahtar ve deşifreleme için farklı fakat ilişkili anahtarı kullanırlar. Yalnızca kriptografik algoritmanın ve şifreleme anahtarının bilgisi verildiğinde, deşifreleme anahtarını tespit etmek mümkün olmamalıdır. RSA gibi bazı algoritmalarda, her iki anahtarda şifreleme ve deşifreleme için kullanılabilir. Bir anahtar şifreleme için kullanılmış ise eş anahtarı deşifreleme için kullanılır.

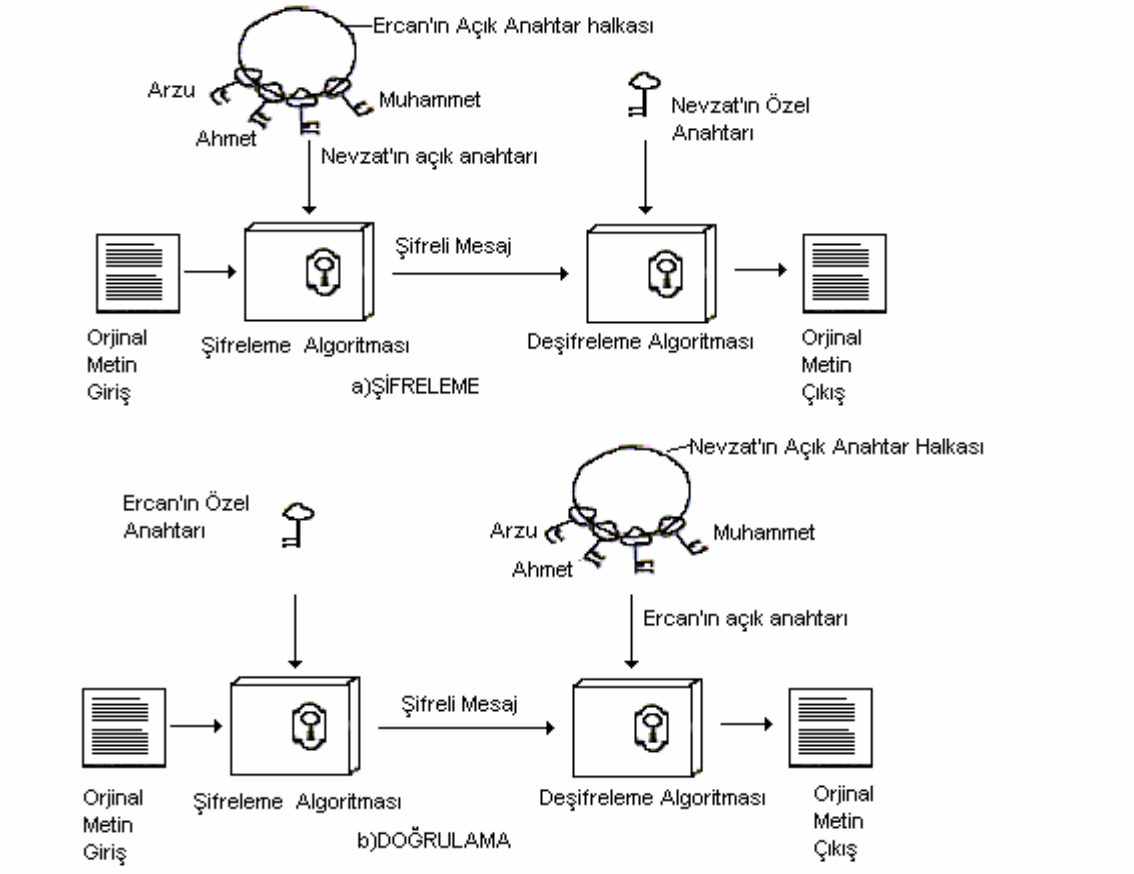
Açık anahtarlı kriptografide şifreleme işlemleri şekil 2.1.a ‘da örnek olarak gösterilmiştir. Adım adım incelersek;

1) Her kullanıcı şifreleme ve deşifreleme işlemleri için bir çift anahtar üretir.

2) Her kullanıcı şifreleme için kullanılan anahtarını, herkesçe erişilebilecek bir dosyaya kaydederek açık anahtarını yayınlar . Eş anahtarı özel olarak saklanır. Buda deşifreleme işleminde kullanılan özel anahtardır.

3) Eğer A, B’ye mesaj yollamak istiyorsa, B’nin açık anahtarını kullanarak mesajı şifreler.

4) B mesajı kabul ettiği zaman kendi özel anahtarını kullanarak onu deşifreleyecektir. B dışında hiçbir alıcı mesajı deşifreleyemez. Çünkü B’nin özel anahtarına yalnızca B sahiptir.



Şekil 2.1 : Açık Anahtar Şifreleme (a- Şifreleme işlemi b-Doğrulama işlemi)

Şekil 2.1.a da Ercan isimli kullanıcının Nevzat isimli kullanıcıya şifreli mesaj göndermesi işlemi görülmektedir. Ercan; Ahmet, Nevzat, Muhammet ve Arzu‘nun açık

anahtarlarına sahiptir. Mesaj göndereceği kişi Nevzat olduğu için, göndermek istediği mesajı Nevzat'ın açık anahtarını ve şifreleme algoritmasını kullanarak şifrelenmiş mesajı elde ederek Nevzat'a gönderir. Mesajı alan Nevzat ise kendi özel anahtarını kullanarak şifreli mesajı deşifreler ve orijinal mesaja ulaşır. Yalnız burada dikkat etmemiz gereken husus doğrulamadır. Nevzat'ın açık anahtarına sahip olan herkes Nevzat'a şifreli mesaj gönderebilir. Mesajın gerçekten Ercan'a ait olup olmadığının doğrulandığını Şekil 2.1.b de görebiliriz. Ercan göndermek istediği mesajı kendi özel anahtarı ile şifreler. Nevzat mesajı Ercan 'ın açık anahtarı ile deşifreler. Ercan'ın özel anahtarı ile şifrelendiği için Ercan tarafından gönderildiği doğrulanır. Çünkü Ercan'ın özel anahtarına yalnızca kendisi sahiptir. Mesajı Ercan'dan başkası şifreleyemez. Diyelimki mesaj herhangi bir rakip tarafından ele geçirildi ve mesajı deşifrelemeyi başardı. Mesajı okuyabilir ama Ercan'ın özel anahtarına sahip olmadığı için mesajı tekrar şifreleyip Nevzat'a gönderemez. Eğer Ercan'ın özel anahtarınıda ele geçirmeyi başardı ise Ercan için iletişim güvensizleşmiştir. Tekrar anahtar çifti üretmesi gerekmektedir.

Örnektende anlaşıldığı üzere tüm kullanıcılar açık anahtarlara sahiptir ve onları kullanabilir. Fakat özel anahtar yalnızca sahibi tarafından kullanılır. Bundan dolayı dağıtılmaya ihtiyacı yoktur. Kullanıcılar kendi özel anahtarlarını kontrol ettiği sürece iletişim güvenlidir. Bir kullanıcı istediği zaman özel anahtarını değiştirebilir ve açık anahtarını yayınlayabilir.

DES gibi geleneksel anahtarlı kriptografiyi kullanan algoritmalar tek anahtar kullandığı için güvensiz görülmektedir. Şifreleme ve deşifreleme için aynı anahtarın kullanılması, kullanılan bu anahtarın paylaşılması güvenlik açığı oluşturmaktadır. Anahtarın ele geçirilmesi kolaylaşmaktadır. Anahtarı paylaşan kişiler anahtarı yetkisiz kişilere verebilir. Ya da anahtar kullanıcılara dağıtılırken ele geçirilebilir. Bundan dolayı geleneksel kriptografi kullanılsa bile anahtar dağıtımı için açık anahtarlı kriptografi kullanılmaktadır. Peki neden geleneksel kriptografi yerine açık anahtarlı kriptografi kullanılmıyor ? Kimi uygulamalarda işlem süresi önemli hale gelmektedir. Açık anahtarlı kriptografi kompleks bir yapıya sahip olduğu için işlem süresi artmaktadır. Eş zamanlı uygulamalar için pek tercih edilmemektedir. Fakat mesajı gönderenin kendini inkar etmemesi gereken uygulamalar ve güvenliğin önemli olduğu ticari uygulamalar için açık anahtarlı kriptografi tercih edilmektedir. Şimdi tablo üzerinde iki kriptografiyi karşılaştıralım.

GELENEKSEL KRİPTOGRAFİ	AÇIK ANAHTARLI KRİPTOGRAFİ
Çalışması için ihtiyaçları : 1- Şifreleme için kullanılan anahtar ve algoritma aynı zamanda deşifreleme içinde kullanılır. 2- Gönderen ve alıcı , algoritma ve anahtarı paylaşmalılar.	**Çalışması için ihtiyaçları :** 1-Bir algoritma , deşifreleme için 1 , şifreleme için 1 olmak üzere; deşifreleme ve şifreleme için 1 çift anahtar kullanır. 2- Gönderici ve alıcıdan her birisi , eşlenen çift anahtarların birine sahip olmalıdır. (Aynı biri değil)
Güvenlik için ihtiyaçları : 1-Anahtar gizli tutulmalı. 2-Eğer diğer bilgi mevcut olmazsa mesajı çözmek olanaksız ve ya en azından elverişsiz olmalı. 3-Algoritmanın bilgisi ve ciphertext ‘in örnekleri anahtarı belirlemek için yetersiz olmalı.	**Güvenlik için ihtiyaçları :** 1-İki anahtardan biri elde gizli tutulmalıdır. 2- Eğer diğer bilgi mevcut olmazsa mesajı çözmek olanaksız ve ya en azından elverişsiz olmalı. 3-Anahtarlar ve ciphertextin örneklerinin biri ve algoritmanın bilgisi diğer anahtarı belirlemek için yetersiz olmalı.

Tablo 2.1 : Geleneksel ve Açık anahtarlı Kriptografi

Tablo 2.1 geleneksel ve açık anahtar kriptografinin bazı önemli yönlerini özetlemektedir. İkisini bir biriden ayırt etmek için geleneksel kriptografide kullanılan anahtardan gizli anahtar diye bahsedeceğiz. Açık anahtar kriptografisinde kullanılan iki anahtardan açık anahtar ve özel anahtar olarak bahsedeceğiz. İki kriptografinin değişmeyen ortak özelliği deşifreleme için kullanılan anahtarın gizli tutulmasıdır. Geleneksel kriptografide deşifreleme anahtarı aynı zamanda şifreleme içinde kullanıldığı için iletişim ağı içerisinde bulunan kullanıcılar tarafından bilimektedir ve gizli tutulması zorunludur. Açık anahtarlı kriptografide deşifreleme anahtarı yalnız deşifreleme işlemi için kullanılır. Geleneksel kriptografideki gizli anahtarla karıştırılmaması için Açık anahtarlı kriptografide deşifreleme anahtarından özel anahtar olarak bahsedeceğiz.

Geleneksel kriptografi ile açık anahtar kriptografinin algoritması brute-force saldırısına karşı aynı derecede savunmasızdır. Bu saldırıya önlem olarak ikisinde de anahtar boyutu büyük tutulmaktadır. Anahtar boyutu, pratik şifreleme ve deşifreleme yapabilecek kadar küçük, aynı zamanda Brute-Force saldırılarını elverişsiz hale

getirecek kadar büyük olmalıdır. Pratikte önerilen anahtar boyutları, brute-force saldırısı elverişsiz hale getirirler. Fakat şifreleme/deşifreleme hızlarının yavaşlamasına neden olmaktadır. Bir başka saldırı şeklide verilen açık anahtardan, özel anahtarı hesaplamaya çalışmaktır. Bu güne kadar belirli bir açık anahtar algoritması için bu saldırı şeklinin mümkün olmadığı matematiksel olarak ispatlanamadı. Stalling(1999)'e göre, Kriptoanalizin tarihine baktığımızda, bir açıdan çözümsüz görünen problemin, farklı bir yoldan incelediğimizde bir çözüm bulunabileceğini gösterir. William Stalling'inde dediği gibi günümüz koşulların da çözülemeyen problemler ileride çözülebilir. Bunu nedeniyse teknolojinin sürekli yenilenmesidir. Daha hızlı işlemcilerin piyasaya çıkması ile işlem yapma yeteneği artmıştır. Bu da kriptoanaliz işlemlerinde çözüm işlemlerinin süresini kısaltmıştır. Tabi ki şifreler kırıldıkça yeni teknikler geliştirilmiş anahtar boyutları değiştirilmiştir. Fakat çözülemiyeceğini garantisi verilememektedir.

2.2. AÇIK ANAHTARLI KRİPTOGRAFİDE GİZLİLİK VE DOĞRULAMA:

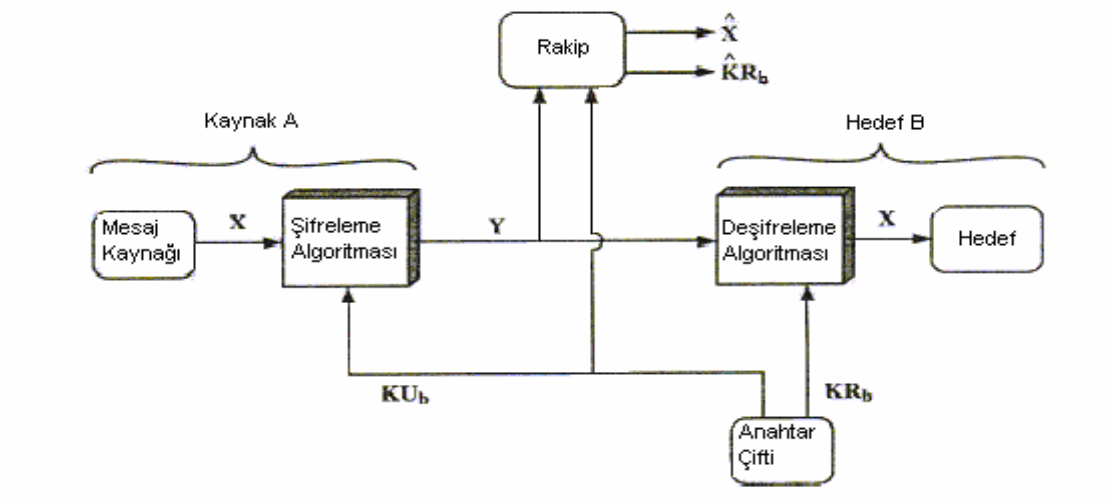
Açık anahtarlı kriptografinin yapısını oluşturan temel elemanları Şekil 2.2 vasıtasıyla inceleyelim. Şekil 2.2 'de A kullanıcısının, B kullanıcısına şifreli mesaj göndermesi esnasındaki gizlilik görülmektedir. A kullanıcısı, B kullanıcısına gönderilecek olan mesajı üretmektedir. B kullanıcısı, ilgili çift anahtarları, açık anahtar KU_b ve özel anahtar KR_b 'yi üretir. KR_b yalnız B kullanıcısı tarafından bilinmektedir. KU_b açık anahtar tüm kullanıcılar tarafından bilinmektedir. Bu nedenle KU_b açık anahtar, rakipler tarafından rahatlıkla elde edilebilir. A kullanıcısı, X mesajın B kullanıcısına göndermek için hazırlar.

A kullanıcısı, şifreleme algoritmasını ve B kullanıcısının açık anahtarını kullanarak X mesajından Y şifreli mesajını elde eder ve B kullanıcısına gönderir.

$$Y = E_{KU_b} (X)$$

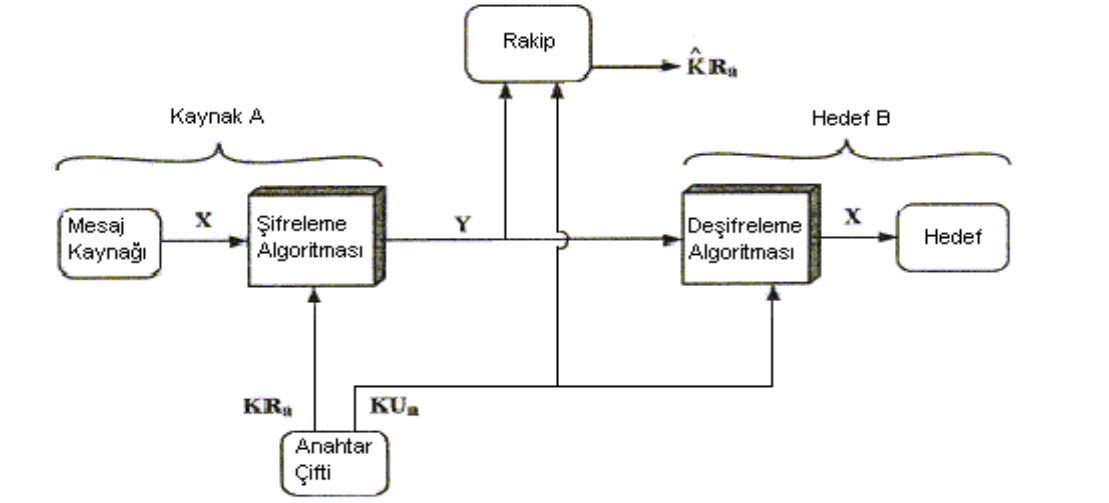
B kullanıcısı da aldığı Y mesajından deşifreleme algoritması ve özel anahtarı ile X'i elde eder.

$$X = D_{KR_b} (Y)$$



Şekil 2.2 : Açık Anahtarlı Kriptosistemde Gizlilik

Burada bizim dikkat etmemiz gereken nokta, şifreli mesaj Y ve açık anahtar KU_b 'yi ele geçirmiş olan rakiplerdir. Y ve KU_b 'ye sahip olan rakip, orijinal mesaj X'i ve özel anahtar KR_b 'yi elde etmeye çalışacaktır. Rakibin şifreleme ve deşifreleme algoritmalarının bilgisine sahip olduğu varsayılır. Rakip yalnızca belirli bir mesaj ile ilgileniyorsa, şifreli mesaj üzerine yapacağı hesaplar ile X'i oluşturmaya çalışacaktır. Ama asıl amaç sonraki mesajları da okuyabilmektir. Bunu yapabilmesi içinde KR_b'yi elde etmesi gerekmektedir.



Şekil 2.3 : Açık Anahtarlı Kriptosistemde Doğrulama

Şekil 2.2'de gizliliğin sağlanmasını görmüştük. B kullanıcısının açık anahtarına sahip olan A kullanıcısı mesajı şifreleyerek gizliliği sağlamıştı. B kullanıcısının açık anahtarına sahip olan rakip kendisini A kullanıcısı olarak tanıtarak B kullanıcısına şifreli mesaj gönderebilir. Bu nedenle gizliliğin yanında dikkat etmemiz gereken önemli

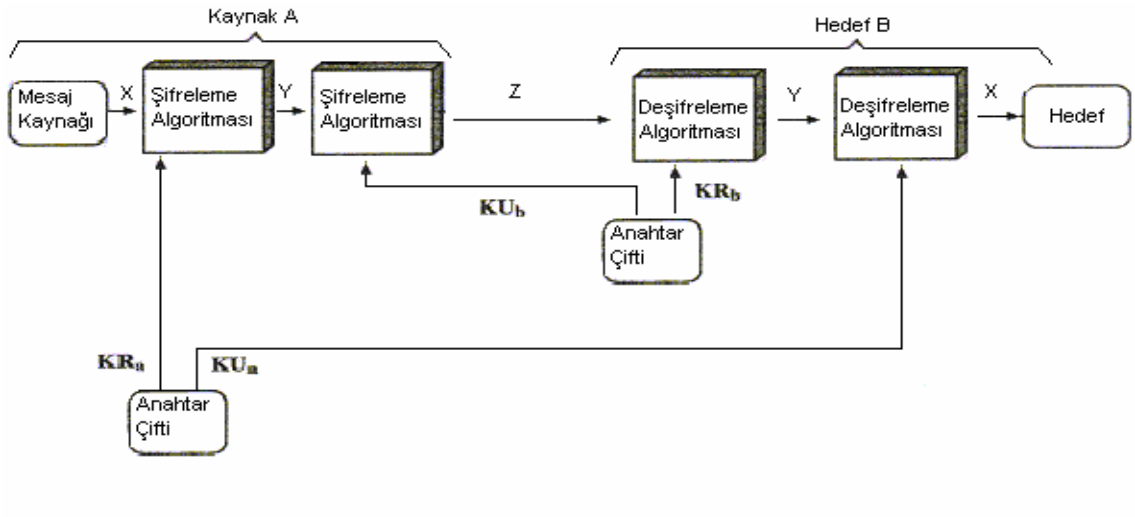
bir nokta da doğrulamadır. Şekil 2.1.b ve 2.3 'de doğrulama işlemi görülmektedir. Doğrulama için A kullanıcısı X mesajını kendi özel anahtarı KR_a ile şifreleyerek Y'yi elde eder.

$$Y = E_{KR_a} (X)$$

Mesajı alan B kullanıcısı A kullanıcısının açık anahtarı ile Y şifreli mesajını deşifreleyerek X'i elde eder.

$$X = D_{KU_a} (Y)$$

A kullanıcısının özel anahtarı sadece kendisinde olduğu ve B kullanıcısı da mesajı A kullanıcısının açık anahtarı ile deşifrelediği için mesajın kaynağı doğrulanır. A kullanıcısının özel anahtarına sahip olmadan mesajını içeriği değiştirilemeyeceği için içerikte doğrulanmış olur. A kullanıcısının açık anahtarına sahip bir rakip mesajın içeriğini görebilir. Ama mesajın içeriğini değiştiremez değiştirebilmesi için A kullanıcısının özel anahtarına sahip olması gerekmektedir. Rakip, A'nın özel anahtarını elde etmek için çeşitli hesaplar yapar. Yalnızca doğrulama veya gizlilik tek başlarına tam olarak güvenliği sağlamaz. Şekil 2.4'te ikisinin birden kullanımı görülmektedir.



Şekil 2.4 : Açık Anahtarlı Kriptosistemde Gizlilik ve Doğrulama

Açık anahtar algoritmasının iki kere üst üste kullanılması ile hem doğruluğunu ispat etme hem de güvenliğini sağlamak mümkündür. (Şekil 2.4)

$$Z = E_{KU_b} [E_{KR_a} (X)]$$

$$X = D_{KU_a} [D_{KR_b} (Z)]$$

Öncelikle göndericinin özel anahtarını kullanarak şifreleme işlemi ile başlarız. Bu digital imzayı sağlar. Daha sonra alıcının açık anahtarını kullanarak tekrar şifreleriz. Şifrelenmiş son metni yalnızca ilişkili özel anahtara sahip olan hedef alıcı tarafından deşifrelenebilir. Böylece güvenlik sağlanır. Bu yaklaşımın dezavantajı , kompleks yapıda olan açık anahtar algoritmasının her iletişimde iki yerine dört kez uygulanmak zorunda olmasıdır.

Şekil 2.4 'te, A kullanıcısı göndermek istediği X mesajını öncelikle kendi özel anahtarı ile şifreleyerek Y'yi elde eder. Daha sonra Y mesajını B kullanıcısının açık anahtarı ile şifreleyerek Z'yi elde eder. Z mesajını alan B kullanıcısı öncelikle kendi özel anahtarı ile Z mesajını deşifreleyerek Y mesajını elde eder. Y mesajını A kullanıcısının açık anahtarı ile deşifreleyerek X orijinal mesajını elde eder.

2.3. AÇIK ANAHTAR KRİPTOGRAFİSİNİN İHTİYAÇLARI:

Açık anahtarlı kriptosistemleri, şekiller üzerinde ilişkili iki anahtara bağlı olarak inceledik. İki anahtar kullanımı ile gizlilik ve doğrulamanın nasıl yapıldığını gördük. Şimdide bu bilgilerimizin ışığında, açık anahtar kriptografisini temel alan bir kriptosistemin ihtiyaçlarını belirleyelim.

1) Alıcı B için, bir çift (açık anahtar KU_b ve özel anahtar KR_b) anahtar üretmek kolay olmalıdır.

2) Gönderen A için, açık anahtarın ve şifrelenecek mesajın (M) bilinmesi durumunda şifreli mesajı üretmek kolay olmalıdır. $C=E_{KU_b}(M)$

3) Alıcı B'nin orijinal mesajı elde edebilmesi için, özel anahtarını kullanarak şifreli mesajı deşifrelemesi kolay olmalıdır.

$$M=D_{KR_b}(C) = D_{KR_b}[E_{KU_b}(M)]$$

4) Bir rakibin açık anahtarı (KU_b) kullanarak özel anahtarı (KR_b) hesaplaması mümkün olmamalıdır.

5) Bir rakibin açık anahtarı (KU_b) ve şifreli metni (C) kullanarak orijinal mesajı (M) elde etmesi mümkün olmamalıdır.

6) Şifreleme ve deşifreleme görevleri arka arkaya uygulanabilir.

$$M=E_{KU_b}[D_{KR_b}(M)]$$

3. AÇIK ANAHTARLI KRİPTOSİSTEMLER:

3.1. DIFFIE-HELLMAN KRİPTO SİSTEMLERİ:

İlk yayınlanan açık anahtar algoritması, Diffie ve Hellman tarafından kriptografide yeni yöntemler olarak tanımlanan makalelerinde(Diffie ve Hellman,1976) görüldü ve genellikle Diffie-Hellman anahtar değişimi olarak bilindi. Bir çok ticari uygulamada bu anahtar değiştirme tekniği kullanıldı. Algoritmanın amacı, iki kullanıcının anahtarı emniyetli bir şekilde değiştirmesini sağlamak ve daha sonra mesajın gizliliği için kullanmaktı.

Diffie-Hellman algoritmasının etkinlik derecesi ayrık logaritmaların hesaplanmasının güçlüğüne bağlıdır. Kısaca, ayrık logaritmaları tanımlayalım. İlk olarak, q asal sayısını ve 1 den q-1 e kadar tüm tam sayıları üretebilen, q asal sayısının bir primitif kökünü α tanımlarız. α , q asal sayısının bir primitif kökü ise, α mod q , α^2 mod q,...., α^{q-1} mod q sayıları birbirinden farklıdır ve 1 'den q-1 'e doğru tamsayılar kapsar. q asal sayısının bir α primitif kökü ve herhangi bir tamsayısı b için eşitliği sağlayacak yalnızca bir tek i üssü vardır. i üssünü bulmak için $b=\alpha^i \bmod q \quad 0 \leq i \leq (q-1)$ denklemini kullanırız.

Global Açık Elemanlar	
q	Asal Sayı
α	α<q ve α primitif kök
Kullanıcı A Anahtar Üretimi	
Özel X_A 'yı seç	X_A <q
Açık Y_A 'yı hesapla	$Y_A = \alpha^{X_A} \bmod q$
Kullanıcı B Anahtar Üretimi	
Özel X_B 'yi seç	X_B <q
Açık Y_B 'yi hesapla	$Y_B = \alpha^{X_B} \bmod q$
Kullanıcı A Tarafından Gizli Anahtarın Üretilmesi	
$K=(Y_B)^{X_A} \bmod q$	
Kullanıcı B Tarafından Gizli Anahtarın Üretilmesi	
$K=(Y_A)^{X_B} \bmod q$	

Şekil 3.1: Diffie-Hellman Anahtar Değişim Algoritması

Temel bilgileri aldıktan sonra şekil 3.1 de özetlenen Diffie-Hellman anahtar değişimini tanımlayalım. Bu yöntemde herkes tarafından bilinen iki sayı vardır: biri q asal sayısı ve diğeri q 'nun primitif bir kökü olan bir α tamsayısı. A ve B kullanıcılarının bir anahtarı değiştirmek istediklerini varsayalım. A kullanıcısı rastgele bir $X_A<q$ tamsayısı seçsin ve $Y_A=\alpha^{X_A} \bmod q$ 'yu hesaplasın. Benzer bir şekilde, B kullanıcısı, A kullanıcısından bağımsız bir şekilde $X_B<q$ olan rastgele bir tamsayı seçsin ve $Y_B= a^{X_B} \bmod q$ 'yu hesaplasın. Her iki kullanıcıda X değerini özel olarak saklar ve Y değerini diğer kullanıcı ile paylaşır. A kullanıcısı $K=(Y_B)^{X_A} \bmod q$ yoluyla ve B kullanıcısı $K=(Y)^{X_A} \bmod q$ yoluyla temel gizli anahtarı hesaplar. Bu iki işlem özdeş sonuçlar üretir:

$$K =(Y_B)^{X_A} \bmod q$$

$$= (\alpha^{X_B} \bmod q)^{X_A} \bmod q$$

$$= (\alpha^{X_B})^{X_A} \bmod q$$

$$= \alpha^{X_B X_A} \bmod q$$

$$= (\alpha^{X_A})^{X_B} \bmod q$$

$$= (\alpha^{X_A} \bmod q)^{X_B} \bmod q$$

$$= (Y_A)^{X_B} \bmod q$$

Böylece, iki taraf gizli bir anahtarı değiştirmiş olur. Ayrıca, X_A ve X_B gizlidir. Bir rakip q,a,Y_A ve Y_B değerlerini bilebilir ve çalışmalarını bu değerler üzerinden yapar. Rakip bu değerleri kullanarak gizli anahtarı belirlemeye çalışacaktır. Gizli anahtarı belirlemek için, X_A ve ya X_B değerlerinden birini hesaplamalıdır. X_A 'yı hesaplamak için Y_A 'yı, X_B 'yi hesaplamak için Y_B 'yi kullanacaktır.

Diffie-Hellman anahtar değişim güvenliği şu gerçeğe dayanır: primitif kökün üssünü alıp modülünü hesaplamak kolayken, modülün sonucundan primitif kökün üssünü (X_A ve ya X_B) hesaplamak çok zordur. Örneğin, Anahtar değişimi için q=97 asal sayısını ve 97 'nin primitif kökü α=5 olsun. A ve B kullanıcıları, gizli anahtarlarını X_A=36 ve X_B =58 sırasıyla seçerler. Her biri kendi açık anahtarını hesaplar:

$$Y_A = 5^{36} = 50 \bmod 97$$

$$Y_B = 5^{58} = 44 \bmod 97$$

Açık anahtarları değiştirdikten sonra her biri temel gizli anahtarı hesaplayabilir.

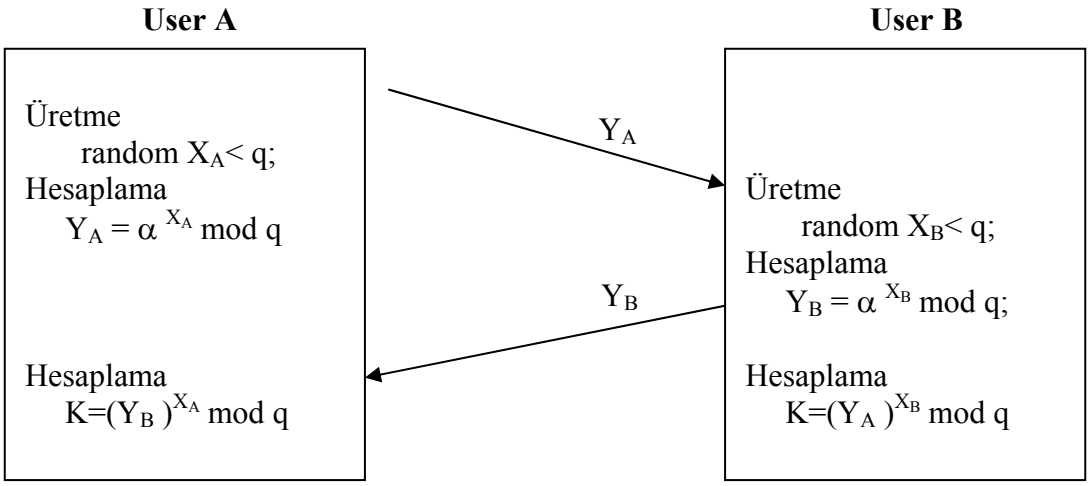
$K = (Y_B)^{X_A} \bmod 97$

$= 44^{36} = 75 \bmod 97$

$K = (Y_A)^{X_B} \bmod 97$

$= 50^{58} = 75 \bmod 97$

Şekil 3.2 'de Diffie-Hellman anahtar değişimini basit bir şekilde ifade etmektedir. A kullanıcısının, B kullanıcısı ile aralarında bir bağlantı kurmak istediğini varsayalım. A kullanıcısı, özel bir X_A anahtarı kullanır ve Y_A 'yı hesaplayarak, bunu B kullanıcısına gönderir. B kullanıcısı, özel bir X_B anahtarı kullanır ve Y_B 'yı hesaplayarak, bunu A kullanıcısına gönderir. Her iki kullanıcıda şimdi anahtarı elde edebilir. Diffie-Hellman algoritmasının diğer bir kullanım örneği de şudur: bir grup kullanıcı varsayalım(LAN Ağındaki tüm kullanıcılar gibi) her kullanıcı dayanıklı ve uzun birer X_A özel anahtarını üretir ve Y_A açık anahtarını hesaplar. Bu açık anahtarlar, q ve α değerleri ile birlikte herkese açık olan merkezi bir rehberde depolanır. Herhangi bir zamanda, B kullanıcısı A nın genel değerine erişebilir, gizli bir anahtar hesaplayabilir, ve A kullanıcısına şifreli bir mesaj göndermek için kullanabilir. Merkezi rehber güvenli ise, bu iletişim iki tarafa da gizlilik sağlar ve kullanıcının gerçekliğini kanıtlar. Çünkü sadece A ve B anahtarı belirler, diğer kullanıcılar mesajı okuyamazlar. Alıcı A, sadece B kullanıcısının bu anahtarla bir mesaj oluşturabileceğini bilmektedir(Kimlik Doğrulama).



Şekil 3.2 Diffie-Hellman Anahtar Değişimi

Örnek.3.1: α=2 ve q=11 alalım. Tablo 3.1 α = 2 (mod 11) 'in üslerini üretilir.

α^{λ}, $1 \le \lambda \le$ q-1	2^1	2^2	2^3	2^4	2^5	2^6	2^7	2^8	2^9	2^{10}
α^{λ} mod 11	2	4	8	5	10	9	7	3	6	1

Tablo 3.1: Primitif Kökün İspatlanması

α , q asal sayısının bir primitif kökü ise, α mod q , α^2 mod q,...., α^{q-1} mod q sayıları birbirinden farklı ve 1 'den q-1 'e doğru tamsayılar kapsaması gerektiğini söylemiştik. Tablo 3.1 'e baktığımızda p=11 asal sayısı için α=2 'nin primitif kök olduğunun ispatlandığını görüyoruz.

İletişimi başlatmak için kullanıcı A, tablo 3.1'deki 2^{λ} mod 11 ={1,2,3,...,10} tamsayı kümesinden rastgele bir X_A=5 sayısını seçer ve bunu gizli tutar. Kullanıcı B'ye $Y_A = \alpha^{X_A}$ mod q 'yu gönderir. Benzer biçimde kullanıcı B, rastgele X_B=7 sayısını seçer. Ve kullanıcı A'ya $Y_B = \alpha^{X_A}$ mod q 'yu gönderir. Şimdi iki kullanıcıda $K = \alpha^{X_A X_B}$ mod q 'yu hesaplayabilir. $Y_A = 2^5$ mod 11 = 10 ve $Y_B = 2^7$ mod 11 = 7 'dir. Sonuçta aşağıda gösterildiği gibi K 'yı hesaplarız.

$K = Y_B^{X_A}$ mod 11
$= 7^5$ mod 11
$= 2^{35}$ mod11
$= 10$

$K = Y_A^{X_B}$ mod 11
$= 10^7$ mod 11
$= (2^5)^7$ mod 11
$= 2^{35}$ mod11
$= 10$

Böylece her bir kullanıcı ortak anahtar sahip olur.

3.2. RSA KRİPTO SİSTEMİ:

Diffie ve Hellman'ın öncülük ettiği çalışmalar, kriptografiye yeni bir yaklaşım getirdi. Şifreleyiciler(kriptolojistler), Açık anahtar sistemlerinin ihtiyaçlarını karşılayan bir şifreleme algoritmasında görüş birliğine vardılar ve meydan okudular. Meydan okuma yanıtlarından ilki 1977'de Ron Rivest , Adi Shamir ve Len Adleman tarafından, MIT de gerçekleştirildi. 1978'de ilk yayın yapıldı(Rivest ve ark.,1978). RSA (Rivest-

Shamir-Adleman) algoritması, geniş bir kitle tarafından kabul edildi ve açık anahtar şifrelemede her amaca uygun yaklaşıma sahiptir.

3.2.1. ALGORİTMANIN TANIMLANMASI:

Algoritmanın yapısı Rivest, Shamir ve Adleman tarafından geliştirildi. Destekleyici görüşlerle birlikte ifadenin kullanımı meydana getirildi. Orjinal mesaj bloklar halinde şifrelenir. Şifreleme ve deşifreleme, herhangi bir orjinal mesaj bloğu M ve şifreli mesaj bloğu C için birbirlerini takip eden yapılardır.

$C=M^e \bmod n$

$M=C^d \bmod n$

$= (M^e)^d \bmod n$

$=M^{e.d} \bmod n$

Gönderici ve alıcının her ikisi de n'nin değerini bilmek zorundadır. Gönderici e'nin değerini bilmektedir ve yalnız alıcı d'nin değerini biliyor. Böylece bu bir KU={e,n} açık anahtarı ve bir KR={d,n} özel anahtarı ile birlikte açık anahtar şifreleme algoritmasıdır. Bu algoritmanın açık anahtar kriptografisinin şartlarını yerine getirmesi için aşağıdaki ihtiyaçlar sağlanmalıdır.

1- n>M'nin sağlandığı tüm durumlarda $M^{ed} =M \bmod n$ iken e,d ve n'nin değerleri tespit edilebiilmelidir.

2- n>M 'nin tüm değerleri için M^e ve C^d 'yi hesaplamak kolay olmalıdır.

3- Verilen e ve n ile d'nin ne olduğunun tespit edilmesi mümkün olmamalıdır.

Şuanda ilk ihtiyacı düşünelim.

$M^{ed} =M \bmod n$

p ve q iki asal sayı olsun. n ve m iki tamsayı olmak üzere n=p.q ve 0<m<n olduğu durumda, keyfi seçilmiş bir k tamsayısı aşağıdaki gibi bir denklem oluşturur.

$m^{k\Phi(n)+1} = m^{k(p-1)(q-1)+1} \equiv m \bmod n$

$\Phi(n)$ sayısı n'den daha küçük pozitif bir tam sayıdır. Ve n ile kendi aralarında asaldır. p ve q asal sayı olmak üzere $\Phi(pq)= (p-1)(q-1)$ 'dir.

Böylece, eğer e.d=k.$\Phi(n)$ +1 olursa ilişkinin isteklerini başarabiliriz. Bu durumda aşağıdaki denkliklerden bahsedebiliriz.

$ed \equiv 1 \bmod \Phi(n)$

$d \equiv e^{-1} \bmod \Phi(n)$

e ve d , mod $\Phi(n)$'de çarpmaya göre ters elemandır.

Dikkat etmemiz gereken bir hususta, modüler aritmetiğin kurallarına göre eğer d (ve sonucunda e'de) ile $\Phi(n)$ aralarında asal olması durumunda bu gerçekleşir. Bu durumda, OBEB($\Phi(n)$,d)=1'dir.

p,q iki asal sayı (özel , seçilmiş)

n=p.q (Açık , hesaplanmış)

e , OBEB($\Phi(n)$,e) =1 ; 1<e<$\Phi(n)$ (Açık , seçilmiş)

d, e^{-1} mod $\Phi(n)$ (özel , hesaplanmış)

{d,n} çiftinden özel anahtar, {e,n} çiftinden açık anahtar oluşur. A kullanıcısının, açık anahtarını yayınladığını varsayalım. B kullanıcısı, A kullanıcısına mesaj M'yi göndermek istiyor. B kullanıcısı, $C=M^e \pmod n$'yi hesaplar ve C'yi gönderir. Şifreli mesaj alındıktan sonra kullanıcı A tarafından $M=C^d \pmod n$ hesaplanarak deşifreleme yapılır.

e ve d 'yi aşağıdaki denkliği sağlayacak şekilde seçtik.

$d \equiv e^{-1} \bmod \Phi(n)$

e ve d 'nin mod $\Phi(n)$ 'de çarpmaya göre ters eleman olmasından dolayı;

$e.d \equiv 1 \bmod \Phi(n)$

Bu yüzden e.d , k$\Phi(n)$+1 'in bir şeklidir. İki asal sayı olan p ve q, birer tamsayı olan n (n = p.q) ve M (0 < m < n) alınarak ispatlanmıştır.

$M^{k.\Phi(n)+1} = M^{k.(p-1).(q-1)} \equiv M \bmod n$

Bundan dolayı $M^{ed} \equiv M \bmod n$;

$C = M^e \bmod n$

$M = C^d \bmod n \equiv (M^e)^d \bmod n \equiv M^{ed} \bmod n \equiv M \bmod n$

Anahtar Üretimi		
p,q seç (p ve q ikisi de asal)		
n = p x q ve $\Phi(n)=(p-1)(q-1)$ hesapla		
e tamsayı seç	OBEB($\Phi(n)$,e) =1; 1< e< $\Phi(n)$	
d'yi hesapla	$d= e^{-1} \bmod \Phi(n)$	
Açık Anahtar KU={e,n} ;	Özel Anahtar KR={d,n}	
	Şifreleme	**Deşifreleme**
Orjinal mesaj	M<n	C
Şifreli mesaj	$C=M^e \pmod n$	$M=C^d \pmod n$

Şekil 3.3: RSA Algoritması

Şekil 3.3'de RSA algoritması özetlenir.

Şekil 3.4'de ki örneği inceleyelim. Bu örnek için anahtarların üretim adımlar aşağıda verilmiştir.

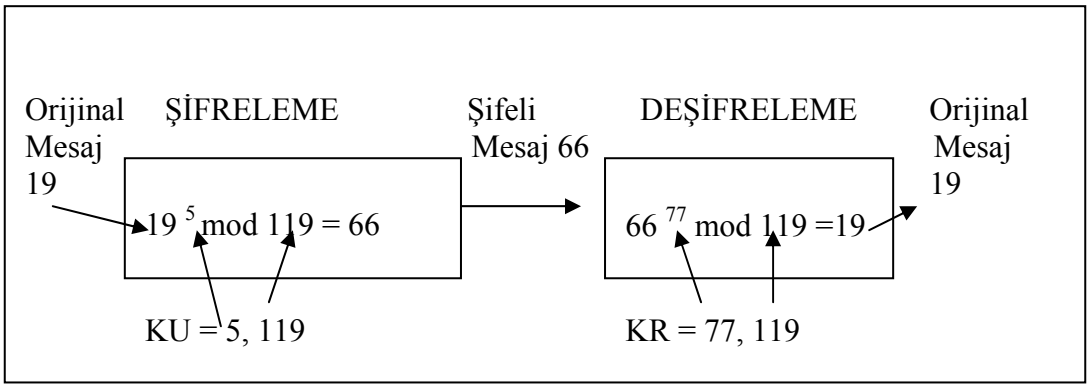
1- İki asal sayı seçilir p=7 ve q=17

2- n = p.q = 7x17 = 119 hesaplanır.

3- $\Phi(n) = (p-1)(q-1) = 96$ hesaplanır.

4- Öyle bir e seçki ; e $\Phi(n)=96$ ya asal olsun aynı zamanda $\Phi(n)$ den küçük olsun . Bu durumda e=5'tir.

5- Öyle bir d belirleyelim ki d.e = 1 mod 96 ve d<96 olsun. Doğru değer d=77 'dir. Çünkü 77x5 = 385 = 4 x 96 + 1



Şekil 3.4: RSA Algoritmasına örnek

Sonuçta anahtarlar; açık anahtar KU ={5,119} ve özel anahtar KR={77,119} olur. Şekil 3.4 deki örnek, M=19 orjinal mesajı için anahtarların kullanımını gösterir. Şifreleme için 19'un beşinci kuvveti alınır ve 2476099 sayısı elde edilir. Bu sayının 119 tarafından bölünmesi sonucunda elde edilen kalan bulunur. Kalan olarak bulunan 66 'tı sayısı bizim şifreli mesajımızdır. $C=M^e \pmod n$ formülünü uygulayarak $C= 19^5 \bmod 119 \equiv 66 \bmod 119$ şifreli mesajımızı elde ettik. Elde edilen şifreli mesajımızı, deşifrelemek için $M=C^d \pmod n$ formülünü uygularız. $M= 66^{77} \bmod 119 \equiv 19 \bmod 119$ işleminin sonucunda orijinal mesajımız olan 19 sayısını elde ederiz.

3.2.2. ÜS ALMA İŞLEMİ:

RSA'da şifreleme ve deşifrelemenin her ikisi de tamsayıların tamsayı kuvvetlerini alma ve mod alma işlemlerinden ibarettir. Eğer üs alma işlemi yapılır ve sonra mod n ile indirgenirse ara değerler büyük olurlar. Neyse ki modüler aritmetiğin özelliklerini kullanarak bu sorunu çözeriz.

$[(a \bmod n) \times (b \bmod n)] \bmod n = (axb) \bmod n$

Böylece, ara değerler mod n'ye indirgenebilir. Bu hesaplamayı pratik yapar.

Dikkat edeceğimiz başka bir konuda üs alma işleminin verimliliğidir. Çünkü RSA ile imkan dahilinde geniş üsler ile uğraşıyoruz. Verimliliği nasıl arttırabildiğimizi görmek için , X^{16}'yı hesaplamayı düşünelim. Doğru yaklaşım 15 çarpma gerektirir :

$X^{16}= X.X.X.....X$

Ancak, eğer biz her bir kısmi sonucun karesini X^2,X^4,X^8,X^{16} 'da olduğu gibi defalarca alırsak yalnız dört çarpma ile aynı sonuca ulaşabiliriz.

Genel olarak a^m 'nin değerini bulmak istediğimizi varsayalım. a ve m 'nin pozitif tamsayıları olduğunu biliyoruz. Eğer m 'yi ikili sayı sistemi gibi $b_k b_{k-1}....b_0$ ifade edersek ;

$m = \sum_{bi \neq 0} 2^i$ yi elde ederiz.

Böylece; $a^m = a^{\left(\sum_{bi \neq 0} 2^i\right)} = \prod_{bi \neq 0} a^{2^i}$

$a^m = \prod_{bi \neq 0} a^{(2^i)} \bmod n = \prod_{bi \neq 0} [a^{(2^i)} \bmod n]$

Şekil 3.5 gösterildiği gibi a^b mod n 'yi hesaplayabilmek için algoritma geliştirebiliriz. Şekil 3.6 'da bu algoritma uygulamasının bir örneği gösterilir. Değişken c'nin gerekmediğini, açıklayıcı amaçlar ihtiva ettiğini unutmayalım. c'nin son değeri üssünün değeridir.

```
c← 0; d←1
for i←k downto 0
    do c←2 x c
    d←(d x d) mod n
    if  bi=1
        then c← c+1
             d←(d x a) mod n
return d
```

Şekil 3.5 a^b mod n yi hesaplamak için algoritma

İ	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
$\mathbf{b_i}$	1	0	0	0	1	1	0	0	0	0
C	1	2	4	8	17	35	70	140	280	560
D	7	49	157	526	160	241	298	166	67	1

Şekil 3.6: a^b mod n için hızlı modüler üstel algoritmasının sonucu, a=7 , b=560=1000110000, n=561

3.2.3. ANAHTAR ÜRETİMİ:

Açık anahtar kripto sistem uygulamasından önce, her kullanıcı bir çift anahtar üretmek zorundadır. Bu üretim işlemi aşağıdaki görevleri içerir.

- iki asal sayı tespit edilir. p ve q
- e veya d seçilir diğeri hesaplanır.

İlk olarak p ve q seçimini düşünürüz. Çünkü n=p.q 'nın değeri herhangi bir potansiyel rakip tarafından bilinebilir. Kapsamlı metotlar kullanarak p ve q nun keşiflerini önlememiz gerekir. Bu asal sayılar yeterince büyük bir kümeden (p ve q büyük sayılar olmak zorundadır) seçilmek zorundadır. Diğer bir yandan büyük asal sayıları bulmak için kullanılan yöntem yeterince verimli olmadır.

Günümüzde kullanılan teknikler, keyfi olarak büyük asal sayılar ürettiği için hiç kullanışlı değillerdir. Genellikle kullanılan metot, istenen büyüklükte tek sayısı rastgele seçilir. Ve o sayının asal olmasına dikkat edilir. Eğer sayı bulunamazsa , asal olan bir sayı bulununcaya kadar ardarda rastgele sayılar seçeriz.

Asal sayılar p ve q 'yu belirledikten sonra anahtar üretimi işlemi, e 'nin seçilip d 'nin değerini hesaplanmasıyla ve ya d 'nin seçilip e 'nin değerini hesaplanmasıyla tamamlanır. e 'yi seçtiğimizi varsayalım. Öyle e seçmeliyiz ki OBEB($\Phi(n)$,e)=1 olmalı ve $d=e^{-1} \bmod \Phi(n)$ 'yi işlemini yaparak d 'yi hesaplarız. Eğer OBEB($\Phi(n)$,e)=1 olursa, diğer tamsayıyı modüle göre elimizdeki tamsayının ters çevrilmişidir. $\Phi(n)$ ile aralarında asal sayı bulunana dek , $\Phi(n)$ ile karşılaştırılarak her birini test ederiz.

3.2.4. RSA DOĞRULUĞUNU İSPATLAMA PLANI:

RSA kriptosistemi, hem gizlilik hem de doğruluğunu ispatlama için kullanılabilen tek açık anahtar kriptosistemidir. Şimdi RSA doğruluğunu ispatlama metodunu inceleyeceğiz.

Y şifreli mesaji, orjinal mesaj X 'in, özel anahtar KR{d,n}'nin kullanılmasıyla elde edilir. Orjinal mesaj X, şifreli mesaj Y 'nin açık anahtar KU{e,n}kullanılması ile elde edilir. Aşağıdaki gibi;

$(X^d \bmod n)^e \bmod n = X$

X<n olduğu yerde $d.e=1 \bmod \phi(n)$ 'dir. Denklemi şöyle yazabiliriz.

$(X^e \bmod n)^d \bmod n \equiv X$

A kullanıcısı p ve q 'nun asal olduğu yerde uygun bir n=p.q seçer. A göndericisi p-1 ve q-1 'in çarpımından $\phi(n)$ 'yi hesaplar. Şimdi A göndericisi özel ve açık anahtarları olan d ve e 'yi seçer. Şöyle ki $d.e=1 \bmod \phi(n)$ 'dir. $KU_a\{e,n\}$ açık anahtarı yayınlanır. A, M mesajını kendi özel anahtarı $KR_a\{d,n\}$ ile şifreleyip Y 'yi elde. Şöyle ki;

$Y=X^d \bmod n$

Y mesajını B alıcısına gönderir. B alıcısı Y ve A'nın açık anahtarı $KU_a\{e,n\}$'yi kullanarak X mesajını yeniden oluşturur. Şöyle ki;

$X = Y^e \bmod n$

$= (X^d)^e \bmod n$

$= X^{d.e} \bmod n$

$= X$

Gönderenin çift anahtarlarından d.e = 1 mod $\phi(n)$ olduğunda dolayı d.e =1 oldu ve X 'i elde ettik. A 'nın özel anahtarı yalnızca A da olduğu ve biz mesajı A 'nın açık anahtarı ile deşifrelediğimiz için mesajı gönderenin A kullanıcısı olduğu doğrulanmış olur.

Örnek 3.2: p=11 ve q=17 alalım. n = p.q =187 bağıntısından n 'yi hesaplarız.

$\phi(n) = (p-1).(q-1) = 10.16 = 160$ e=23 seçin.

Sonra d.e = 1 mod $\phi(n)$

d.23 = 1 mod 160

Ondan sonra (23)(7) = 161 mod 160 =1 , d = 7 olarak belirlenir. X=55 farz edelim. Sonra

$Y = X^d \mod 187 = 55^7 \mod 187 = 132$

Kabul edilen mesaj $X = Y^e \mod n = (132)^{27} \mod 187 = 55$ olarak yeniden oluşturulur. Böylece X mesajı güvenilir olarak kabul edilir.

Örnek 3.3: p=41 ve q=59 asal sayıları seçelim, n = p.q = 2419 ve $\phi(n) = (p-1)(q-1) = 2320$ olur. d =151 özel anahtar olsun. Açık anahtar e, d =151 ile ilgili olduğundan e 'yi 151 mod 2320 'nin çarpımsal ters çevrilmişini bularak elde ederiz. OBEB$(d, \phi(n))$ OBEB(151,2320)=1 olmalıdır. e 'nin değerinin hesaplanması aşağıdadır.

2320=(151).15+55

151=(55).2+41

55=(41).1+14

41=(14).2+13

14=(13).1+1

13=(1).13 Sıfır olmayan son kalan

Sıfır olmayan son kalan 1 olduğunda,OBEB(151,2320)

Bu yüzden , iki tam sayı olan 151 ve 2320 aralarında asaldır. Şimdi 1 'i , 2320 ve 151'in doğrusal birleşimi gibi OBEB $(d, \phi(n)) = a.d + b.\phi(n)$ bağıntısıyla ifade edebiliriz.

1 =14 – (13) 1

1 = 14 – (41 – 14 x 2) = (14) 3 – 41

1 = (55 – 41)3 – 41 = (55) 3 – (41) 4

1 = (55) 3 – (151 – 55 x 2) 4 = (55) 11 – (151) 4

$1 = (2320 - 151 \times 15)11 - (151)\,4$

$1 = (2320)\,11 - (151)\,169$ **(3.1)**

a=169 =e ve b=11 olduğunda 3.1 eşitliği $1 \equiv -(151)(169) \pmod{2320}$ şeklinde tekrar yazılabilir. –169 mod 2320 'ye göre 2151 'e denk olduğu için, açık anahtar e =2151 'i özel anahtar d=151 modulo $\phi(n)$=2320 'nin çarpma işleminin tersi olarak görebiliriz.

HARF	KOD	HARF	KOD	HARF	KOD	HARF	KOD	HARF	KOD	HARF	KOD
Bank	00	E	05	J	10	O	15	T	20	Y	25
A	01	F	06	K	11	P	16	U	21	Z	26
B	02	G	07	L	12	Q	17	V	22		
C	03	H	08	M	13	R	18	W	23		
D	04	I	09	N	14	S	19	X	24		

Örnek 3.4: Seçilen p=41 ve q=59 'dan, n=(41)(59)=2419 ve $\phi(n)$=(40)(58)=2320 'yi hesaplayalım. e=169 seçildiğinde d=2169 olur. X = "PUBLIC KEY CRYPTOGRAPHY" orjinal mesajını, her bloğu 4 karakterden oluşan bloklar halinde aşağıdaki gibi ifade edebiliriz.

1621 0212 0903 0011 0525 0003

1825 1620 1507 1801 1608 2500

İlk blok 1621 'in, e=169 'a göre üssünü alırız. Çıkan sonucu n=2419'a böldüğümüzde kalan 1757 bize şifreli mesaj Y 'yi verir. $1621^{169} \pmod{2419} = 1757$ şifreleme işlemini yapmış oluruz. Aynı işlemi ilk bloktan başlayarak tüm bloklara uygularız. Böylece tüm X orjinal mesajı aşağıdaki gibi şifrelenir.

1757 0874 1272 1447 0241 1315

1843 2376 1931 1842 0788 1393

Bunun gibi, şifreli mesaj Y 'nin ilk bloğu 1757 'nin d=2169 'a göre üssünü alırız. Çıkan sonucu n=2419 'a böldüğümüzde kalan 1621 bize orijinal mesaj X'i verir. $1757^{2169} \pmod{2419} = 1621$ deşifreleme işlemini yapmış oluruz. Aynı işlemi ilk bloktan başlayarak tüm bloklara uygularız. Böylece tüm Y şifreli mesajı aşağıdaki gibi deşifrelenir.

1621 0212 0903 0011 0525 0003

1825 1620 1507 1801 1608 2500

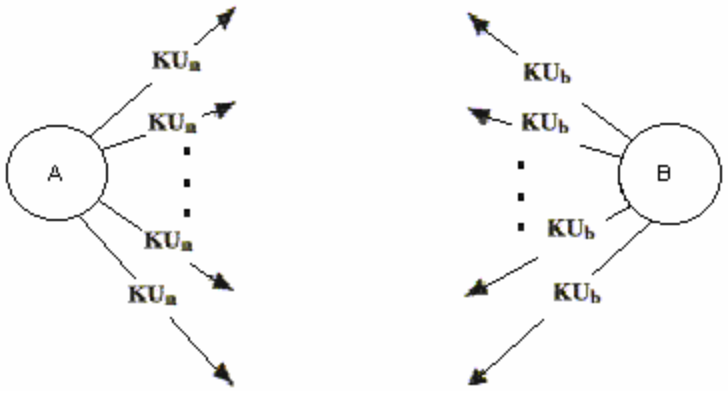
4. AÇIK ANAHTAR DAĞITIMI:

Açık anahtar kriptosistemlerinde ana rollerden biride, anahtar dağıtım probleminin çözülmesidir. Bu bağlamda açık anahtar kriptosistemlerinde kullanılan, açık anahtar dağıtımı için birkaç teknik ileri sürülmektedir. Bu görüşler aşağıda listelenmiştir.

- Genel Duyuru
- Adres Rehberi
- Açık anahtar yetkilisi
- Açık anahtar sertifikası

4.1. AÇIK ANAHTARLARIN DUYURULMASI:

Açık anahtarlı kriptografinin temelini, açık anahtarın herkesin kullanımına açık olması oluşturmaktadır. Bundan dolayı, eğer açık anahtar kullanımının tüm kuralları kabul etmek gerekirse, RSA gibi olmalıdır. Herhangi bir kullanıcı kendi açık anahtarını diğer bir kullanıcıya gönderebilir. Ve ya yayın yolu ile geniş bir kitleye yayabilir.(Şekil 4.1) Örnek olarak RSA'nın kullanımını sağlayan PGP'nin, popülaritesinin yükselmesinden dolayı, bir çok PGP kullanıcıları kendi açık anahtar ekleme uygulamalarını benimsemişler ve USENET haber grubu ve internet mail listelerine gönderildiği gibi, mesaj olarak genel forma da gönderilmişlerdir.



Şekil 4.1: Kontrolsüz Açık Anahtar Dağıtımı

Bu yaklaşımın uygun olmasına rağmen büyük bir açığı vardır. Herkes genel bir duyuruyu taklit edebilir. Bu olay şöyle açıklanabilir, örneğin bazı kullanıcılar kendilerini kullanıcı A olarak tanıtabilirler. Açık anahtarlarını bir diğer kullanıcıya

gönderebilirler ve ya yayınlayabilirler. Kullanıcı A sahtekarlığı fark edene ve ya ikaz edilene kadar, taklitçi kişi, kullanıcı A için gelmiş olan tüm şifreli mesajları okuyabilir ve belgelemek için taklit edilmiş anahtar takımlarını kullanabilirler.

4.2. ADRES REHBER:

Açık anahtar bilgilerini içeren ve yüksek derecede güvenliği sağlanmış bir adres rehberi oluşturulur. (Şekil 4.2) Anahtarların dağıtımı ve korunması kişi ve ya kuruluşların sorumluluğunda olmalıdır. Böyle bir hizmet için aşağıdaki koşullar sağlanmalıdır.

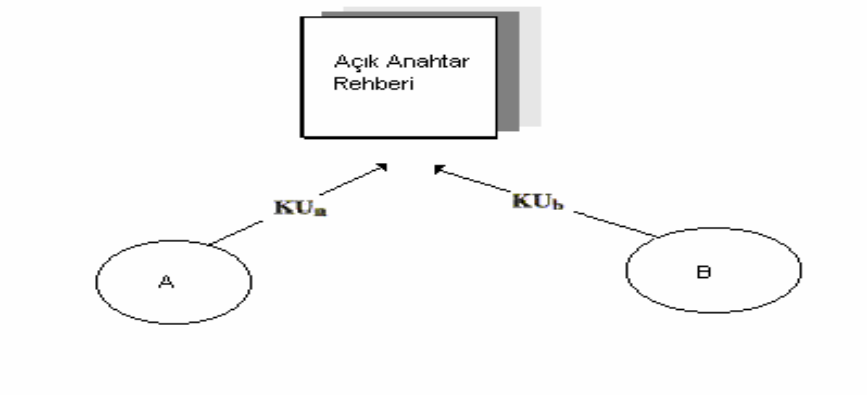
1- Her bir kullanıcı için ayrı ayrı girilerek isim ve anahtarlardan rehber oluşturulur.

2- Her bir kullanıcı yetkisiyle açık anahtarı kaydeder. Kaydetme bir kişi için gizli olarak belgelenmiş iletişim formlarıyla yapılır.

3- Kullanıcı her zaman için açık anahtarını yenisi ile değiştirebilir. Kullanıcı anahtar çiftinin güvenliğinin kaybolduğuna inanabilir. Rakip gizli anahtarını ele geçirmiş olabilir. Bu nedenle yeni bir anahtar çifti oluşturulur. Yeni açık anahtarı rehbere ekler.

4- Periyodik olarak yetkililer tüm rehberi ilan edebilir. Yeni kullanıcılar ekleneceği ve zaman zaman kullanıcılar açık anahtarlarını değiştirebilecekleri için adres rehberi güncellenmeli ve ilan edilmelidir.

5- Kullanıcı adres rehberine elektronik olarak erişebilmelidir. Bunun için yetki, gizlilik ve kimlik denetimini gerektiren güvenli bir iletişim metodu kullanılmalıdır.



Şekil 4.2: Açık Anahtarın Yayınlanması

Bu yöntem birbirinden bağımsız yapılan genel duyurulardan çok daha fazla güvenlidir. Ama hala sorunlu olabilir. Şayet rakip, rehber otoritesinin özel anahtarını elde etmeyi ve hesaplamayı başarırsa, rakip kendi yetkisiyle sahte açık anahtarlar dağıtabilir. Ve buna ek olarak herhangi bir kullanıcının taklidini yapabilir. Diğer kullanıcılara giden mesajları gizlice dinleyebilir. Rakibe karşı başarılı olmanın bir diğer yolu ise yetki ile kayıtları değiştirmektir.

4.3. AÇIK ANAHTAR YETKİLİSİ:

Bu yöntemde de merkezi bir rehber vardır. Açık anahtar dağıtımı için güvenilirliğin oluşması sıkı bir kontrole bağlıdır. Herkes tarafından erişilebilen adres rehberinin güvenliği sağlanmalıdır. Bunu sağlamak için her kullanıcıya yöneticinin açık anahtarı verilir. Anahtar dağıtımı aşağıda gösterildiği gibi yapılır.

1- A kullanıcısı, B kullanıcısının açık anahtarını öğrenmek istediğine dair yöneticiye zaman imzalı mesaj gönderir.

2- Yönetici, kendi özel anahtarı (KR_{auth}) ile şifrelenmiş bir mesajı A kullanıcısına cevap olarak gönderir. A kullanıcısı, bu mesajı yöneticinin açık anahtarı ile deşifreler. Yöneticinin özel anahtarına kendisinden başka kimse sahip olamayacağı için mesajın yöneticiden geldiği doğrulanır. Bu mesajda;

- B kullanıcısının açık anahtarı. KU_b, B 'ye tahsis edilmiş olup, mesajı şifrelemek için A kullanıcısı tarafından kullanılabilecektir.
- A kullanıcısının yöneticiye göndermiş olduğu mesajın orjinali. Böylelikle mesaj kıyaslanır isteği yapanın A kullanıcısı olduğu garanti edilir. Orijinal isteğin, yetkilinin eline geçmeden önce değiştirilmemiş olduğu doğrulanmış olur.
- Zaman imzası; bu sayede A kullanıcısına gönderilen mesajın eski olmadığı ve gönderilen açık anahtarın B kullanıcısına ait olduğu tespit edilir.

3- A kullanıcısı B kullanıcısının açık anahtarını kullanarak, B kullanıcısına mesaj gönderir. Bu mesajın içerisinde, A kullanıcısının kimliği (ID_A) ve bu iletişimi tanımlayıcısı olan bir kelime (N_1) bulunur.

4- B kullanıcısı, A kullanıcısının açık anahtarını öğrenmek istediğine dair yöneticiye zaman imzalı mesaj gönderir.

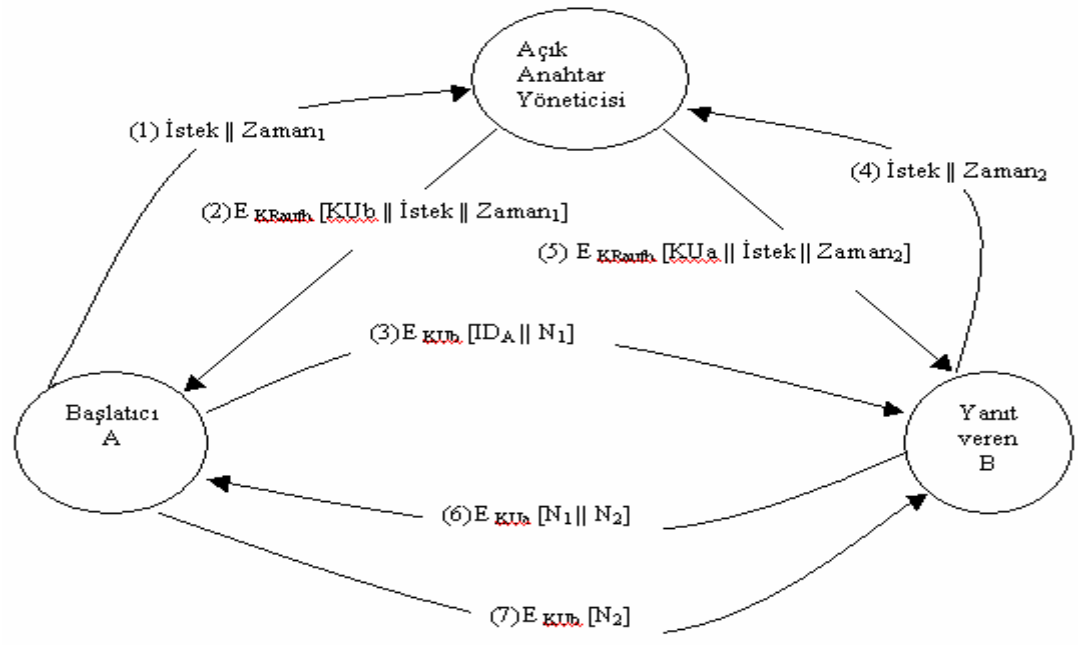
5- Yetkili, B kullanıcının mesajını cevaplar. 2. Adımda A'nın mesajını nasıl cevapladı ise aynı yöntemi kullanır.

Bu noktada, açık anahtarlar A ve B kullanıcılarına güvenli bir şekilde teslim edilmiştir. Kullanıcılar güvenli iletişime geçebilir, birbirlerine şifreli mesaj gönderebilirler. Ancak, iki ek adımın kullanılması tavsiye edilir.

6- B kullanıcısı, A kullanıcısının açık anahtarını (KU_a) kullanarak A kullanıcısına şifrelenmiş mesaj gönderir. Bu mesajda A 'nın göndermiş olduğu N_1 kelimesi ve kendi oluşturduğu N_2 kelimesi bulunmaktadır. A kullanıcısı, mesajı alıp kendi göndermiş olduğu N_1 tanımlayıcı kelimesini görünce, B 'nin umduğu kişi olduğundan emin olur. Çünkü N_1 kelimesini B 'den başka kimse deşifreleyemez.

7- A kullanıcısı, B kullanıcısına güven vermek için B 'nin tanımlayıcı kelimesi N_2 'yi B 'nin açık anahtarı ile şifreler ve B 'ye mesaj olarak gönderir.

Bu yedi adımda güvenli bir şekilde açık anahtarlar elde edildi. Bundan sonra A ve B kullanıcıları birbirlerinin açık anahtarlarını kullanarak güvenli iletişim kurabilirler. Açık anahtarlar kullanıcılar tarafından saklanacağı için bu işlemler iletişimin başlangıcında olur. Kullanıcılar arasında iletişim mevcut anahtarlar ile sağlanır. Fakat periyodik olarak açık anahtarların güncellenmesi gerekir. Çünkü kullanıcılar anahtar çiftlerini değiştirebilir.



Şekil 4.3: Açık Anahtar Dağıtım Senaryosu

4.4. AÇIK-ANAHTAR SERTİFİKALARI:

Açık anahtar yetkilisinin olduğu bir önceki görüşte yetkili bir dar boğaz oluşturmaktadır. Bir kullanıcının, haberleşeceği herkes için yetkili ile temasa geçmesi lazım. Buna alternatif olarak açık anahtar sertifikaları geliştirildi. Bu yöntemde, katılımcılar anahtarları değiştirmek için bir açık anahtar yetkilisiyle bağlantıya geçmelerine gerek yoktur. Anahtar değişimini gerçekleştirebilecekleri sertifikalar vardır. Bu da açık anahtar yetkilisinin elinden alınmış anahtarlar kadar güvenlidir.

Sertifika yöneticisi her kullanıcıya sertifika verir. Bu sertifikada kullanıcının kimlik tanımları, açık anahtarı ve zaman pulu bulunur. Sertifika, yöneticinin özel anahtarı ile şifrelenmiştir. Sertifika ile birlikte yöneticinin açık anahtarı verilir. Bu sayede diğer kullanıcılar sertifikanın yetkili tarafından oluşturulduğunu doğrulayabilirler. Bu yöntem için şartlar aşağıda belirtilmiştir

1. Her kullanıcı sertifika okuma yetkisine sahiptir. Yöneticinin açık anahtarı ile sertifikayı deşifreler. Haberleşeceği kullanıcının açık anahtarını ve kimlik bilgilerini öğrenir.

2. Herhangi bir kullanıcı sertifikanın orjinalliğini kanıtlayabilir. Sertifika, yöneticinin özel anahtarı ile şifrelenir. ($C_A=E_{KRauth}[T,ID_A,KU_a]$) Şifreli mesaj ancak yöneticinin açık anahtarı ile deşifrelenir. Kullanıcılar yöneticinin açık anahtarına sahip oldukları için mesajın orjinalliğini kanıtlar. $\{ D_{KUauth}[C_A] = D_{KUauth} [E_{KRauth} [T,ID_A,KU_a]] = (T,ID_A,KU_a)\}$

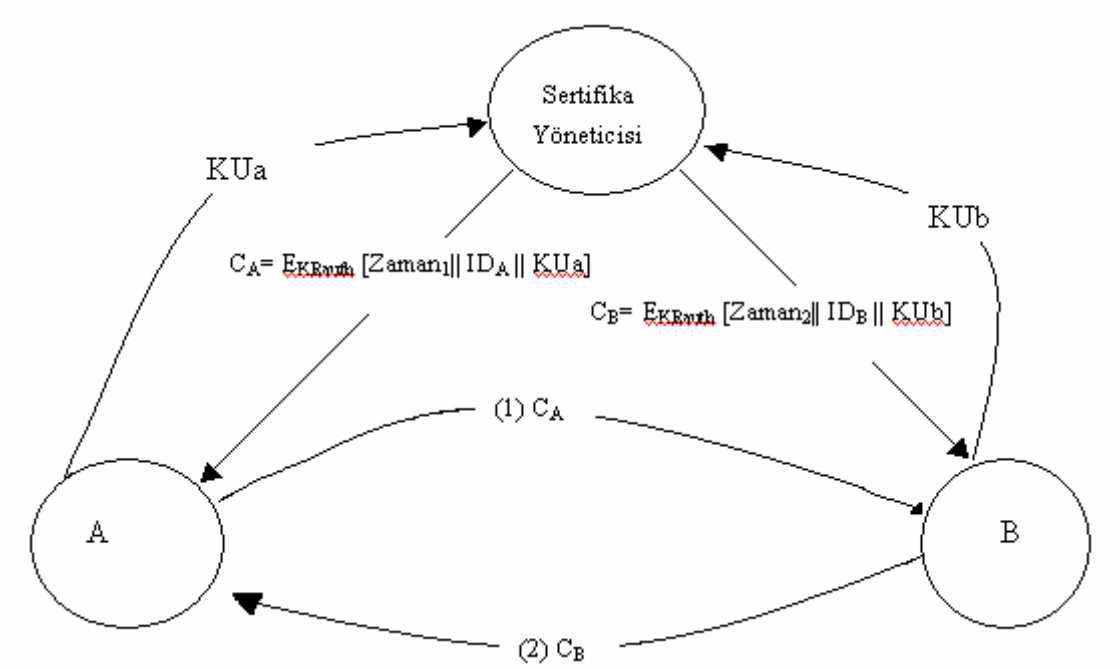
3. Sertifika sadece yönetici tarafından oluşturulur. Gerekli güncellemeler sadece yönetici tarafından yapılır.

4. Herhangi bir kullanıcı zaman pulu sayesinde sertifikanın geçerliliğini kanıtlayabilir. Zaman pulunda sertifikanın geçerli olduğu tarih vardır.

Sertifika planı Şekil 4.4 ‘de gösterilmiştir. Her kullanıcı,yüz yüze yada kimlik denetimi ve gizliliğin sağlandığı güvenli bir haberleşme yöntemi kullanarak sertifika yöneticisi ile temasa geçer. Yönetici sertifikasını oluşturup kullanıcıya teslim eder. Kullanıcı A için yönetici bir sertifika oluşturur. Yönetici özel anahtarını (KR_{auth}) kullanarak sertifikayı şifreler. A kullanıcısı, daha sonra bu sertifikayı aşağıdaki anlatıldığı gibi diğer kullanıcılara gönderebilir.

Örneğin A kullanıcısı, B kullanıcısı ile haberleşmek istiyor. Her iki kullanıcıda sertifika yöneticisinden sertifikalarını almıştır. A kullanıcısı B kullanıcısına sertifikasını gönderir. B kullanıcısı, A'nın sertifikasını yöneticinin açık anahtarını kullanarak deşifreler. B, A'nın kimlik bilgilerine ve açık anahtarına sahip olur. Zaman pulunda sertifikanın geçerlilik tarihi bulunmaktadır. Bu sayede sertifikanın güncel olduğu anlaşılır. B kullanıcısı, A kullanıcısına kendi sertifikasını gönderir. Bu şekilde A kullanıcısı B'nin açık anahtarına sahip olur. Bu noktadan sonra iki kullanıcıda birbirlerinin açık anahtarlarına sahiptir. Açık anahtarları kullanarak güvenli bir şekilde haberleşebilirler.

Alıcı yöneticinin açık anahtarını sertifikayı deşifrelemek için kullanır. Çünkü sertifika yalnızca yöneticinin açık anahtarı (KU_{auth}) ile okunabilir. Bu şunu doğrular: sertifika yöneticiden gelmiştir. ID_A ve KU_a, elemanları sertifika sahibinin açık anahtarı ve kimliğini doğrular. Son olarak, zaman pulu T sertifikanın sürümünü onaylar. Zaman pulu(timestamp) aşağıdaki planı karşılar: A 'nın özel anahtarı rakip tarafından öğrenilirse, A yeni bir özel/açık anahtar çifti üretir ve sertifika yetkilisine yeni bir sertifika için başvurur. Bu arada rakip eski sertifikayı B 'ye yerini değiştirerek gönderir. Eğer B uzlaştıkları eski açık anahtarı kullanarak mesajları açıklarsa, rakip bu mesajların hepsini okuyabilir.



Şekil 4.4: Açık Anahtar Sertifikasının Değişimi

Temelde, bir özel anahtarın rakip tarafından öğrenilmesi ile bir kredi kartının kaybedilmesi birbirine benzer. Kredi kartı kaybedildiğinde, sahibi tarafında kredi kartı iptal edebilir fakat kredi kartının geçersiz olduğu haberdar edilene kadar kullanıma açık olacaktır. Bu süre içerisinde kartın kullanılması kart sahibi için sorunlar çıkaracaktır. Böylece, zaman pulu son tarihi veren bir sayaç gibi hizmet eder. Eğer bir sertifika yeterli derecede eskiyse, onun sona erdirilmiş olduğu farz edilir.

5.AÇIK ANAHTARLI KRİPTOGRAFİNİN KULLANILDIĞI AĞ GÜVENLİK UYGULAMALARI

5.1 X509 KİMLİK DOĞRULAMA SERVİSİ

X509, bir servis dizinini tanımlar. Dizin, bir sunucu ve ya kullanıcılar hakkındaki bilgilerin tutulduğu veritabanını oluşturan sunucuların dağıtılmış kümesidir. Kullanıcı isminden ağ adreslerine kadar ek olarak da diğer nitelikleri ve kullanıcılar hakkındaki bilgileri içeren veritabanı oluşturulur. Dizin, bir açık anahtar sertifikaları deposu gibi düşünülebilir ve açık anahtarların dağıtıldığı gibi dağıtılabilir. Her sertifika bir kullanıcının açık anahtarını içerir ve güvenilen bir yöneticinin özel anahtarıyla şifrelenmiştir. X.509 açık anahtar kriptografisinin kullanımına ve dijital imzalara bağlıdır. Standart belirli bir algoritma kullanımını gerektirmez ama RSA'yı tavsiye eder.

International Telecommunication Union (ITU) ve International Standards Organization (ISO) tarafından geliştirilen X.509 standart önerisi (ITU ve ISO,1997), sertifika yapılarına ve güven kavramına ortak bir yaklaşım getirmektedir. Sertifika yapılarındaki bu standart oluşum, değişik amaçlı uygulamaların tek bir sertifika yapısı ile çalışabilmesini hedeflemektedir. Bu amaca ne kadar ulaştığı tartışılsada, konusundaki uluslar arası otoritelerce kabul edilmiş tek standart olması, X.509 'un açık anahtar tabanlı değişik sistemlerdeki uygulanırlığını arttımıştır.

5.1.1 Sertifikalar

X.509 planının en önemli noktası, her kullanıcıyla ilişkilendirilmiş açık anahtar sertifikasıdır. Bu kullanıcı sertifikaları, bazı güvenilir sertifika otoriteleri (SO) tarafından oluşturulur. SO veya kullanıcı tarafından dizine yerleştirilir. Dizin sunucusunun kendisi açık anahtar yapımından veya belge görevinden sorumlu değildir. O sadece, kullanıcıların belgeleri elde edebilmesi için kolaylıkla ulaşılabilir yerler sağlar. Bir sertifikanın doğrulanabilmesinin temel şartı üzerindeki imzanın geçerliliğidir. Sertifikanın geçerliliğini kontrol eden kullanıcı, SO'nu açık anahtarını kullanarak imzayı doğrular. Doğrulayıcının bir sertifikayı doğrulayabilmesinin başka bir şartıda SO 'nun güvenli olmasıdır.

Bir X.509 sertifikasındaki temel elemanları şekil 5.1 de görebiliriz.

Versiyon
Seri Numarası
Algoritmanın Tanımlayıcısı • Algoritma Adı • Parametreler
Sertifika Otoritesi
Geçerlilik Periyodu • Başlangıç Tarihi • Bitiş Tarihi
Sertifika Sahibi
Açık Anahtar Bilgileri • Algoritma • Parametreler • Açık Anahtar
Otoritenin Sayısal İmzası

Şekil 5.1:X.509 Sertifika Standardı

5.1.2. Bir Kullanıcı Sertifikasının Elde Edilmesi

Bir SO tarafından oluşturulan kullanıcı sertifikaları şu özelliklere sahiptir:

• SO'nin açık anahtarına sahip olan herhangi bir kullanıcı, iletişim kurmak istediği kullanıcıya ait sertifikadan, kullanıcının açık anahtarını elde edebilir.

• Belirtilmedikçe, SO dışındaki hiçbir kimse sertifikayı değiştiremez.

Sertifikalar değiştirilemeyeceği için, bir dizin içinde o dizini korumak için özel bir çaba harcamadan saklanabilirler.

Bütün kullanıcılar aynı SO' den sertifikalarını alırlarsa, o SO için ortak bir güven vardır. Bütün kullanıcı sertifikaları, bütün kullanıcıların erişebileceği bir dizinde saklanabilirler. Ek olarak, bir kullanıcı sertifikasını diğer kullanıcılara direk olarak iletebilir. Her iki durumda da, B, A'nin sertifikasına sahipse, A'nın açık anahtarıyla şifrelediği mesajların, bir saldırıya karşı güvenliği olduğundan ve A'nin özel anahtarıyla imzalanmış mesajların değiştirilmemiş olduğundan emindir.

Çok geniş bir kullanıcı kitlesi varsa, bunların, sertifikalarını aynı SO'den almaları çok pratik bir uygulama değildir. Bu SO, sertifikaları imzalayan SO olduğundan, her bir kullanıcının, imzaları doğrulamak için, SO'nın açık anahtarının bir kopyasına sahip olması gerekir. Bu açık anahtar, her bir kullanıcıya o kadar güvenli bir şekilde sağlanmalıdır ki, kullanıcı, ilgili sertifikaların doğruluğundan emin olsun. Kullanıcı sayısının çok fazla olduğu düşünülürse, birden fazla SO'nin kullanılması ve

her birinin açık anahtarlarını belirli kullanıcı kesimlerine güvenli bir şekilde iletmeleri daha pratik olacaktır.

Şimdi, A'nın X_1, SO'sinden ve B'nin X_2, SO'sinden birer sertifika elde ettiklerini varsayalım. A, X_2'nin açık anahtarını güvenli bir şekilde bilmiyorsa, B'nin X_2 tarafından yayınlanan sertifikası A için bir anlam taşımaz. A, B'nin sertifikasını okuyabilir fakat imzayı doğrulayamaz. İki SO güvenli bir şekilde kendi açık anahtarlarını birbirlerine iletebilirlerse, aşağıdaki prosedür A'nın B'nin açık anahtarını elde etmesini sağlar:

1.A, dizinden X_2'nin X_1 tarafından imzalanmış sertifikasını alır. A, X_1'in sertifikasını güvenli bir şekilde bildiği için, X_2'nin açık anahtarını sertifikasından elde edebilir ve sertifikadaki X_1 'in imzasıyla doğrulayabilir.

2.Ardından A, dizine geri döner ve B'nin X_2 tarafından imzalanmış sertifikasını alır. A, şu anda X_2 'nin açık anahtarının kopyasına güvenli bir şekilde sahip olduğu için, imzayı doğrular ve B'nin açık anahtarını elde eder.

A, B'nin açık anahtarını elde etmek için bir sertifika zincirini kullandı. Bu zincir X.509 notasyonuyla şöyle ifade edilebilir:

$X_1<<X_2>>X_2<<B>>$

Aynı prosedürle B, A'nın açık anahtarını ters zincirle şu şekilde elde edebilir:

$X_2<<X_1>>X_1<<A>>$

Bu plan iki sertifikalık bir zincirle sınırlandırılamaz. Bir zincir üretmek için, keyfi uzunluktaki bir SO'lar yolu takip edilebilir. N elemanlı bir zincir şu şekilde ifade edilebilir:

$X_1<<X_2>>X_2<<X_3>>..... X_N<<B>>$

Bu durumda zincirdeki her bir SO 'nun sertifikalarına sahip olmak gereklidir. Tüm bu SO sertifikalarının dizinde yer almaları gerekir ve kullanıcının diğer kullanıcının açık anahtarına giden yolu takip edebilmesi için, SO sertifikalarının birbirlerine nasıl bağlandığını bilmesi gerekir. X.509, SO sertifikalarının, sertifikalar arasındaki gezintinin ileriye doğru olmasını sağlayacak biçimde bir hiyerarşiyle düzenlenmesini tavsiye eder.

X.509'dan alınana Şekil 5.2 böyle bir hiyerarşi örneğidir. Birbirlerine bağlanmış daireler SO'leri arasındaki ilişkiyi gösterir; dairelere bağlanmış kutular her bir SO'nin dizininde barındırılan sertifikaları gösterir. Dizinlerde iki tip sertifika bulunabilir:

•**İleriye doğru sertifikalar:** X'in diğer SO'leri tarafından oluşturulmuş sertifikaları.

•**Geriye doğru sertifikalar:** X tarafından yaratılmış diğer SO'lerine ait sertifikalar.

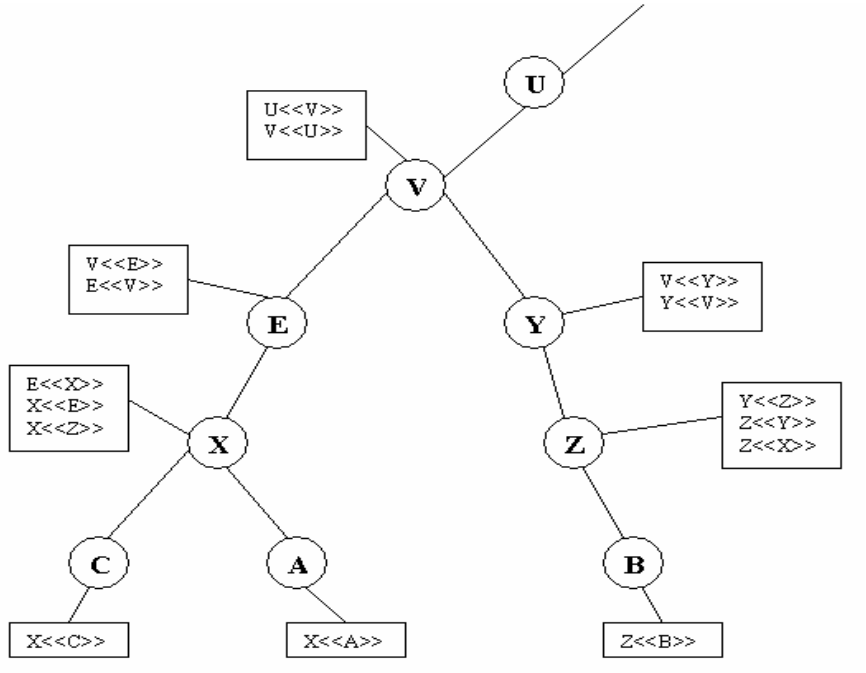
Bu örnekte, A kullanıcısı için B kullanıcısının sertifikasına giden yol :

X<<E>>E<<V>>V<<Y>>Y<<Z>>Z<<B>>

B de, A'nin sertifikasını elde etmek için şu yolu takip etmelidir:

Z<<Y>>Y<<V>>V<<E>>E<<X>>X<<A>>

İkisi de birbirlerinin sertifikalarını bu şekilde aldıkları takdirde birbirlerinin açık anahtarına sahip olurlar. Birbirleri için veri şifreleyip imzalayabilirler.



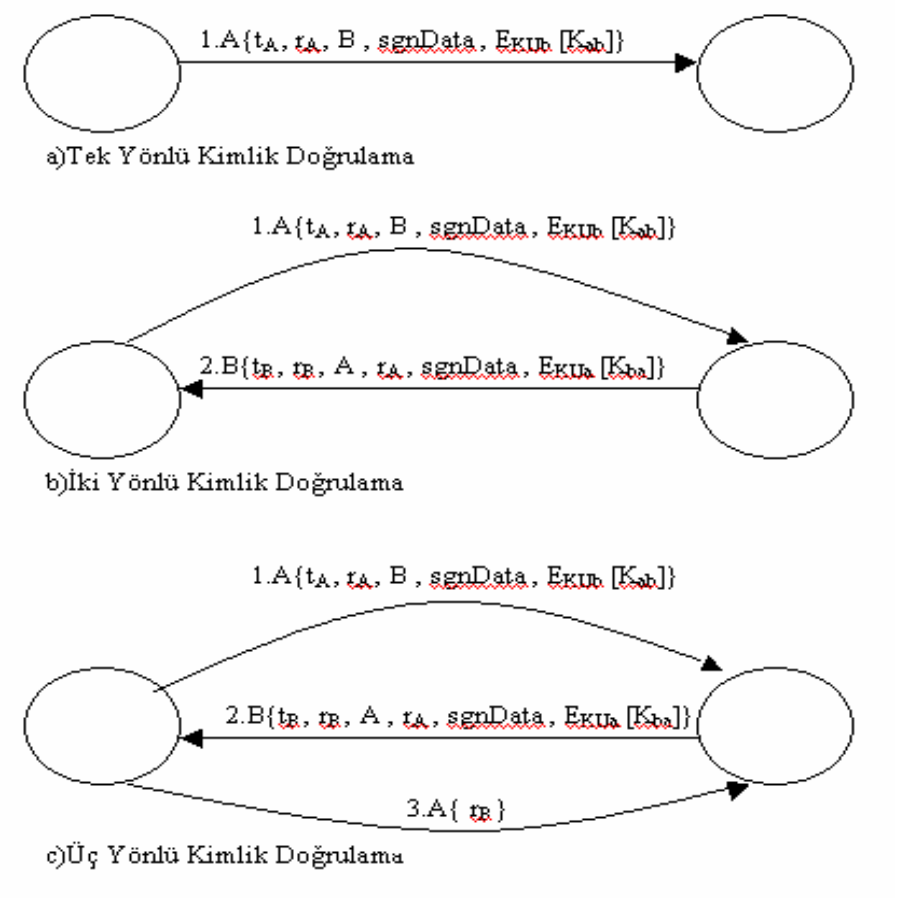
Şekil 5.2: X.509 Hiyerarşisi: Hipotetik örnek

5.1.3. Sertifikaların İptal Edilmesi

Her bir sertifika bir kredi kartı gibi geçerlilik süresi bilgisi içerir. Tipik olarak yeni bir sertifika eskisinin süresi dolmadan hemen önce yayınlanır. Kullanıcının özel anahtarının güvenliği yitirildiğinde, kullanıcı bu SO tarafından imzalanmak istenmezse

ve ya SO'nin sertifikasının güvenliği yitirildiğinde sertifikanın süresinin dolması beklenmeden iptal edilir. Her bir SO, kullanıcıları ve diğer SO'leri için, iptal edilmiş sertifikalar için bir liste tutmalıdır. Bu listeler ayrıca dizine de yerleştirilmelidir.

Her bir sertifika iptal listesi (=certificate revocation list: CRL) yayıncı tarafından imzalanmalıdır ve yayıncı ismi, listenin oluşturulma tarihi, sonraki CRL'nin oluşturulup yayınlanacağı tarih ve iptal edilmiş sertifikalar, bilgilerini içerir. İptal edilmiş sertifikalar bilgisinde her bir sertifika için, sertifikanın seri numarası ve iptal edilme tarihi bilgileri saklanır. Seri numaraları bir SO için tekil olduklarından, seri numarası sertifikayı tanımlamak için yeterlidir.Bir kullanıcı bir sertifika aldığında, bu sertifikanın iptal edilip edilmediğini kontrol etmelidir.



Şekil 5.3 : X.509 Kimlik Doğrulama Prosedürleri

5.1.4. Kimlik Doğrulama Prosedürleri

X.509 üç alternatif kimlik doğrulama prosedürü içerir. Bunların hepsi açık anahtarlı imzaları kullanır. İki kullanıcının bir birlerinin sertifikalarına sahip olduğu varsayılır. Şekil5.3 bu üç prosedürü gösterir.

Tek Yönlü Kimlik Doğrulama

Tek yönlü kimlik doğrulama, bir kullanıcıdan (A) diğerine (B) tek bir bilgi transferinden oluşur ve aşağıdakilerin doğruluğunu sağlar:

1. A'nın kimliği ve mesajın, A tarafından oluşturulduğu
2. Mesajın B için olduğu
3. Mesajın bütünlüğü ve orijinalliği

Yalnızca başlatan kullanıcının kimliği doğrulanıyor. Mesaj, bir zaman damgası t_A, bir nonce r_A ve B'nin kimliğinden meydana gelir ve A'nın açık anahtarıyla imzalanmıştır. Zaman damgası, üretim zamanı ve bitiş zamanından meydana gelir. Bu gecikmiş mesajların sebep olabilecekleri sorunları engeller. Nonce replay ataklarını belirlemek için kullanılır. Nonce değeri, mesajın bitiş süresi içinde tekil olmalıdır. Böylece, B nonce'i süresi doluncaya kadar saklayıp, aynı nonce' a sahip olan mesajları reddedebilir.

İki Yönlü Kimlik Doğrulama

Yukarıda listelenen üç elemana ek olarak, iki yönlü kimlik doğrulama aşağıdakilerin doğruluğunu sağlar:

4. B'nin kimliği ve replay mesajının B tarafından oluşturulduğu
5. Mesajın A için olduğu
6. Replayin bütünlüğü ve orijinalliği

Böylece iki yönlü kimlik doğrulamada, iki kullanıcının da kimliğinin doğrulaması sağlanır. Replay mesajı A'dan gelen nonce'i içerir. Ayrıca bir zaman damgası ve B tarafından üretilmiş bir nonce içerir.

Üç Yönlü Kimlik Doğrulama

Üç yönlü kimlik doğrulamada A'dan B'ye r_B nonce'ının imzalanmış bir kopyası olan bir son mesaj içerir. Bu tasarımın amacı zaman damgalarının kontrolünü ortadan kaldırmaktır. İki nonce da diğer kullanıcı tarafından geriye döndürüldükleri için, her bir taraf, replay ataklarını belirlemek için, döndürülen nonce'ları kontrol edebilir. Bu yaklaşım, eş zamanlı saatler olmadığı zaman gereklidir.

5.2. PGP (PRETTY GOOD PRIVACY, Oldukça İyi Gizlilik):

Elektronik posta, neredeyse tüm dağıtım alanlarında en sık kullanılan ağ bazlı uygulamadır. Aynı zamanda tüm yapılarda ve satış platformunda yaygın bir şekilde kullanılmakta olan tek uygulamadır. Kullanıcılar, yerel işletim sistemini veya iletişim maliyetini sorun etmeden, internete doğrudan yada dolaylı olarak bağlanan diğer insanlara mail gönderebilmeyi beklemektedirler.

PGP, dikkate değer bir olgudur. Büyük ölçüde sadece bir tek kişinin, Phil Zimmermann'ın gayretleriyle, PGP, elektronik posta ve dosya saklama uygulamalarında kullanılabilecek olan güvenlik ve gizlilik servisi sunmaktadır. Zimmermann özetle şunları yapmıştır:

1- Yapı taşları olarak en uygun kriptografik algoritmaları seçti.

2- Bu algoritmaları, işletim sisteminden ve işlemciden bağımsız olarak ve ufak bir dizinin kolay kullanım komutlarına dayanarak genel kullanım uygulamalarıyla bütünleştirdi.

3- İnternet sayesinde kaynak kodunu ücretsiz olarak bildirdi ve ticari ağları içeren paketi ve dokümanları hazırladı.

4- PGP'nin, tamamen uyumlu düşük maliyetli ticari versiyonunu oluşturmak için bir şirketle anlaşma imzaladı.

PGP hızla büyüdü ve şimdi yaygın ölçüde kullanılmakta. Bu büyümenin sebebi olarak şunlar sıralanabilir:

1- Dos\Windows'u , UNIX'i, Macintosh'u ve diğerlerini içeren çeşitli platformlarda dünya çapında bedelsiz versiyonları mümkün kılmaktadır.

2- Kapsamlı eleştiriler sonucunda ayakta kalan algoritmalara dayanmaktadır ve oldukça güvenlidir.

3- İnternet ve diğer ağlar yoluyla başkalarıyla güvenli bir şekilde iletişim kurmak isteyen bireylerden, belgeleri ve mesajları iletmek için standart bir düzenek seçmek isteyen herkese yönelik geniş bir uygulanabilirliğe sahiptir.

İdari yada standartçı bir kurum tarafından geliştirilmemiştir ve kontrol altında değildir. 'Kuruluşlara' yönelik içgüdüsel bir güvensizliğe sahip olanlar için PGP çekicidir. Aslında temel olarak bakıldığında, dosya bazında işlem yapan bir şifreleme ve sayısal imzalama programıdır. E-posta gönderme özelliği yoktur. Sadece E-posta olarak gönderilebilir halde dosyalar oluşturur.

PGP'nin ticari olamayan kişiler ve kuruluşlar tarafından edinilmesi ve kullanımı tüm dünyada ücretsizdir. PGP'nin tüm ticari hakları ViaCrypt adlı bir ABD şirketinindir. O yüzden, ticari kullanım için bu şirketten lisans alınması gereklidir. PGP'nin ABD dışına çıkması yasak olduğu için ViaCrypt'in ABD dışına lisans vermesi de şu anda mümkün gözükmemektedir. PGP, RSA ve IDEA adlı iki patentli şifreleme algoritması kullanmaktadır. RSA'nın patenti ABD dışında geçerli değildir. Ancak IDEA'nın patenti Ascom-Tech AG isimli bir Isviçreli şirkete aittir. ABD ve Kanada dışındaki ülkelerde, bu şirketten IDEA lisansı alarak ticari olarak PGP'yi kullanmak mümkündür.

PGP emir bazında işlem yapan bir arayüze sahiptir. Orjinal PGP'nin grafik arayüzü yoktur. O yüzden kullanımı gayet zordur. PGP'nin program kodunun da dağıtılması, Phil Zimmermann'ın dışında, bu konu ile ilgilenen başka insanları PGP uyumlu programlar yazmaya teşvik etmiştir. Böylelikle, PGP Eudora ve Pine gibi E-posta işlemleri yapan diğer programlarla entegre edilmiş ve birtakım Windows ön uçları üretilmiştir.

5.2.1. Şifreleme ve İmzalama

PGP, temel olarak RSA algoritmasına dayanır. RSA algoritması ile mesajı şifrelemek performansı etkileyecektir. Bundan dolayı gönderici, mesajı şifrelemek için oturum anahtarı ile birlikte CAST-128, IDEA yada 3DES geleneksel kriptografik algoritmalarından birisi kullanılır. Kullanılan bu oturum anahtarı rastgele oluşturulur ve bir kere kullanılır. Şifreleme işlemi yaptığımız oturum anahtarı RSA kullanılarak şifrelenip mesajın başına ilave edilir. MD5 ve ya SHA1 algoritmalarından birisi kullanılarak mesajın 128 bitlik bir hash kodu (özü) oluşturulur. Oluşturulan bu hash kodu RSA kullanılarak şifrelenir ve imza oluşturulur. Oluşturulan bu imza pakete yerleştirilir. Hash kodunu kullanarak ana mesajı elde etmek imkansızdır. İmza oluşturma ve şifreleme işlemlerinin bu şekilde kullanılması PGP'nin performansını artırmıştır.

PGP'nin başka bir özelliği ise mesajı şifrelemeden önce ZIP yöntemiyle sıkıştırmasıdır. Bu şekildedosya boyutu küçültülmüş olup yerden tasarruf sağlanmış olur. Kimi rakipler şifreyi çözmek için harflerin kullanım sıklığına bakmakta ve o şekilde ataklar yapmaktalar. ZIP yöntemi ile sıkıştırmamız sayesinde bu tür tehditlerden korunmuş oluruz.

Tabiki tehditler bundan ibaret değildir. Rakiplerimiz ağı dinleyebilir. Şifrelenmiş ve imzalanmış mesajımız ağ üzerinde alıcıya iletilirken rakipler tarafından kopyalanabilir. Kopyası alınan mesajımız ileriki bir tarihte alıcıya gönderilebilir. Mesajın üzerindeki şifre ve imza doğru olduğu için, alıcı mesajın gönderenden geldiğini zannedebilir. Bu sorunun çözümü için mesajımızın başına zaman pulu(time stamp) konur. Böylelikle tarihin güncelliği sayesinde mesajın orjnalliği kabul edilir. Ve gönderme zamanından emin olunur.

PGP'de eğer bir mesaj hem imzalanıp hem de şifrelenecekse, önce gönderenin özel anahtarı ile mesajın hash kodu imzalanır. Daha sonra mesaj ile imza birlikte sıkıştırılır ve şifrelenir. Bu şifrelemede kullanılan oturum anahtarı ise alıcının açık anahtarı ile şifrelendikten sonra paketin başına konur. Bu işlemden sonra mesaj ASCII formatına çevrilir. Alıcı mesajı aldıktan sonra gönderenin yapmış olduğu işlemleri sondan başlayarak tersini yapacaktır. İlk olarak ASCII formatından ikili formata çevirir. Daha sonra, kendi özel anahtarını kullanarak oturum anahtarını deşifreler. Deşifrelemiş olduğu oturum anahtarını kullanarak, imza ve mesajı deşifreler. Artık, orjinal mesaj alıcının elindedir. PGP, imzayı doğrulamak için gönderenin açık anahtarını kullanarak imzayı deşifreler. Deşifreleme sonucunda orijinal mesajımızın hash kodunu elde ederiz. Alıcı orijinal mesajın hash kodunu üretir. Deşifrelemiş olduğu hash koduyla karşılaştırır. Eğer ikisi birbirine eşitse imzanın doğruluğunu ispatlamıştır. Tam tersi olurda eşit çıkmaz ise bu mesaj bizim için güvensiz demektir ve herhangi bir değeri yoktur.

5.2.2. Anahtar İşlemleri

PGP, RSA algoritmasının kullandığı özel anahtar ve eş anahtarı olan açık anahtarı oluşturabilir. Anahtar boyutlarının büyük olması güvenliği arttıracaktır. Bundan dolayı anahtarları ezberlemek ve ya akıl da tutmak mümkün değildir. PGP anahtarları dosyalar halinde tutar. Kullanıcıya ait özel anahtar şifrelenip secring.pgp isimli dosyada tutulur. Gerektiğinde bu dosyadan okunarak işlem yapılır. Özel anahtarı şifrelemek için CAST-128, IDEA yada 3DES algoritmalarından birisi kullanılabilir.

PGP, açık anahtarları pubring.pgp isimli bir dosyada saklar. Açık anahtarların gizlilik gibi bir gereksinimleri olmadıkları için şifrelenmelerine de gerek yoktur. Açık anahtarlarla ilgili tek sorun yanlış açık anahtar sorunudur. Gönderenin veya alıcının, şifreleme ve imza kontrollerinde yanlış açık anahtar kullanmaları yanlış sonuçlara sebep olabilir. PGP bu sorunu aşmak için açık anahtar imzalama (public key signatures),

başka bir deyişle sertifika mekanizmasını kullanır. Aslında en iyi açık anahtar edinme yolu sahibinden doğrudan almaktır. Bu şekilde elde edilen açık anahtarlar kullanıcı tarafından imzalanarak pubring.pgp dosyasına kaydedilir. Bunun imkansız olduğu durumlarda ise sertifika yöntemi kullanılır. Sertifika, bir açık anahtarın doğruluğuna bir başkasının verdiği garantidir. Garantiyi veren sertifika otoritesidir. SO kendi özel anahtarını kullanarak, açık anahtarı ve sahibinin kimliğini şifreler. Böylece sertifika SO tarafından imzalanır. Açık anahtar ve anahtara atılan imzalar pubring.pgp dosyası içinde tutulur.

Şimdi de sertifika mekanizmasının PGP'de nasıl çalıştığına bakalım. Bir kullanıcı, başka bir kullanıcıya ait olan açık anahtara doğrudan güvenmeyebilir. Bundan dolayı o kullanıcıya ait sertifikayı inceler. Sertifikada imzası bulunan kişileri tanıyıp tanımadığına bakar. Bunu pubring.pgp dosyası içinde imzalayıcı ile ilgili bir anahtarın olup olmadığına bakarak yapar. Eğer imzalayıcının açık anahtarı varsa ve o anahtar üzerinde kullanıcının tanıdığı ve güvendiği başka birinin (veya kendisinin) imzası varsa, kullanıcının imzalayan insanı tanıdığına hükmedilir. Bundan sonraki kontrol ise tanınan imzalayıcının imzaladığı açık anahtarlara güvenilip güvenilmeyeceğinin kontrolüdür. Eğer o anahtarı güvenilen biri imzalamışsa, kullanıcı sözkonusu açık anahtarı güvenerek kullanabilir. Bu mekanizmanın işlemesi için pubring.pgp içindeki her anahtar için bir güven bilgisi tutulur ve bu bilgiye göre kullanıcının açık anahtar sahibinin verdiği imzalara güvenip güvenmeyeceği hesaplanır. Bu hesaplama PGP tarafından otomatik olarak yapılır. Eğer sonuçta açık anahtarın doğruluğuna karar verilmişse kullanıcının sözkonusu anahtarı imzalayarak pubring.pgp dosyasına kaydetmesinde bir sakınca yoktur. Burada önemli olan diğer bir konu ise zincirleme güven konusudur. Bir kullanıcı bir başka kullanıcıya güveniyorsa onun güvendiği başka kullanıcılara da güvenmeli midir? Bu sorunun cevabı evet ise kullanıcı zincirleme sertifika izin veriyor demektir. Burada hiyerarşik bir geçiş söz konusudur. Bu hiyerarşide kaç seviye gidileceğini ise kullanıcı belirler.

Özetlemek gerekirse, bir kullanıcının doğruluğuna güvenmediği bir açık anahtarı çekinmeden kullanabilmesi için ortakça tanınan ve güvenilen bir veya birkaç kişinin, ya doğrudan ya da hiyerarşik bir düzende birkaç seviyede geçişli olarak, sözkonusu açık anahtarı imzalayarak sertifika vermesi gerekir.

5.3. INTERNET PROTOKOL GÜVENLİĞİ (IPSec):

IP seviyesinde paketlerin güvenilir bir şekilde iletilmesi için IETF (Internet Engineering Task Force) tarafından geliştirilen protokoldür. LAN'larda, WAN'larda veya her ikisinde de kullanılır. 10 yıl öncesine kadar Internet Protokol (IP) şebekeleri, asıl olarak akademik ve araştırma alanlarında kullanılıyordu. Birçok kuralları tanımlanmış olmasına rağmen, dünyada çok az insan "internet" hakkındaki teknolojiden haberdardı. İnternet kullanımındaki artış ve bunun iş dünyasında kullanımı ile; günümüzde güvenlik son derece önem kazandı. Geçen birkaç yıl içinde; Internet hizmeti veren kuruluşlar, internette güvenlik ile ilgili çözümler geliştirmişlerdir. IP şebekede taşınan trafiğin güvenliği için geliştirilen teknolojilerden en önemlileri; SSL (Secure Socket Layer) ve IPSec (Internet Protocol Security)'dir. IPSec, network seviyesinde uygulandığı için IP üzerinden taşınan her türlü trafik için güvenlik sağlamaktadır. IPSec; IP network üzerinde data iletiminde bilginin, gizliliğini, bütünlüğünü ve güvenirliğini sağlar. IPSec; gizliliğini, bütünlüğünü ve güvenirliği tüm sistemde sağlamak için birkaç değişik güvenlik teknolojilerini birlikte kullanır. Bulk algoritması, Açık anahtarlı kriptografi, Diffie-Hellman anahtar değişimi ve dijital sertifikaları kullanır.

IPSec'in faydaları

- IPSec bir firewall veya yönlendirici içinde uygulandığında tüm trafiğe uygulanabilen güçlü bir güvenlik sağlar.
- Eğer İnternet üzerinden organizasyona girmek için firewall tek giriş ise ve dışardan gelen tüm trafik IP kullanmak zorunda ise firewall içindeki IPSec baypasa karşı dayanıklıdır.
- IP iletişim katmanının altındadır (TCP,UDP) ve tüm uygulamalara açıktır. Kullanıcı veya servis sağlayıcı sistemde herhangi bir yazılım değişikliği gerektirmez.
- IPSec son kullanıcılar tarafından kullanılabilir. Eğitim gerektirmez.
- Eğer gerekli ise IPSec bireysel kullanıcılar için de güvenlik sağlayabilir.Bu özellik hassas uygulamalar için bir organizasyon içinde güvenli bir sanal alt ağ kurmada ve merkez dışında çalışanlar için faydalıdır.

Yönlendirme uygulamaları

IPSec, ortam sistemlerini ve ağlarını koruma ve son kullanıcıları destekleme dışında internet çalışması için gerekli yönlendirme yapısında önemli bir rol oynar.

IPSec aşağıdakileri garantiler:

- Bir yönlendirici ilanının yetkili bir yönlendiriciden gelmesini.
- Komşu bir ilanın yetkili bir yönlendiriciden gelmesini
- İlk paketin gönderildiği yönlendiriciden yeniden yönlendirilen bir mesajın gelmesini.
- Bir yönlendirici güncelleştirmesinin sahte olmadığını.

Bu tip güvenlik önlemleri yoksa karşı taraf iletişimi bozabilir veya trafiğin bir kısmının yönünü değiştirebilir.

5.3.1. Güven Birlikleri (Security Association):

Bir IPSec bağlantısında, iki nokta arasında hangi algoritmanın kullanılacağı IP datagramı gönderilmeden önce belirlenmesi gerekir. İki nokta arasında kullanılan bu algoritmada; hangi açık anahtar, özel anahtar, anahtar değişim algoritması, şifreleme algoritması ve oturum anahtarı kullanılacağı ve SA yenileme zamanının belirlenmesi gerekir.

Güvenli bir iletişim için hangi güvenlik algoritmalarının kullanılacağını belirleyen iki nokta arasındaki ilişki Güven Birliği(SA)'dir. IPSec uç noktaları arasında kabul edilen bir anlaşmadır da denilebilir.

SA'lar; Security Parameter Index (SPI) ile veritabanında her bağlantı için tek bir parametre olarak belirlenir.

SA'lar tekyönlüdür. A noktasından B noktasına ESP uygulanırken, B noktasından A noktasına AH uygulanabilir. SA'lar tek yönlü oldukları için veritabanında gelen trafik ve giden trafik için ayrı iki tablo tutulur.

5.3.2. IKE (Internet Anahtar Değişimi):

IETF tarafından, IPSec'de SA'nın kurulması için seçilmiş bir protokoldür.

İki nokta arasında doğrulanmış ve güvenli tünel kurulmasını, IPSec'de SA için görüşmeler yapılmasını sağlar. Bu işlem iki noktanın birbirlerine kim olduklarını ve oturum anahtarını belirlemelerini sağlar.

Doğrulama: IKE farklı doğrulama metotlarını destekler, iki nokta arasında aynı doğrulama metodu kullanılmalıdır. Bu metotlar; Açık Anahtarlı Kriptografi ve dijital imzalar.

Anahtar Değişimi : Her iki taraf da; IKE tünelini şifrelemek için ortak anahtar olan oturum anahtarını bilmesi gerekir. Bunun için Diffie-Hellman anahtar değişim protokolü kullanılır.

Bu iki adım, Doğrulama ve anahtar değişimi iki nokta arasında güvenli tünel olan IKE SA'nın belirlenmesini sağlar.

A, B'ye veri paketi göndereceği zaman A'nın ve B'nin PC'leri arasında bir tünel kurulur. Bu tünel üzerinden SA belirlenir. A ve B arasında güvenli bir mesajlaşma işlemi sağlanır. IPSec kullanıldığında IP paketine yeni başlıklar eklenir. Bu yeni başlıklar IP paketinin güvenliği için gerekli bilgileri içerir. IP paketine eklenen yeni başlıklar;

AH (Doğrulama Başlığı): Bu başlık kısmı IP paketine eklendiğinde, verinin değiştirilmemesini ve verinin doğru kişiden doğru kişiye iletilmesini sağlar. Verinin şifrelenerek gizlenmesiyle ilgilenmez.

ESP (Encapsulating Security Payload): Bu başlık kısmı IP paketine eklendiğinde, verinin gizliliğini, bütünlüğünü ve güvenirliğini sağlar.

AH ve ESP ayrı ayrı veya her ikisi birlikte kullanılabilir.

AH ve ESP'nin her ikisi de mesaj bütünlüğünü korumak için dijital imza kullanır.

AH'de; IP paketlerinin bir kısmı veya tamamı alınır, hash kodu üretilir ve bir oturum anahtarı konularak karşı tarafa gönderilir. Bu oturum anahtarı sadece alıcı ve gönderici tarafından bilinmektedir.

ESP'de; IP paketlerinin bir kısmı veya tamamı alınır, DES (Data Encrytion Standart) veya 3DES ile şifrelenir. Daha sonra paketin hash kodu üretilir ve oturum anahtarı konularak gönderilir. Bu oturum anahtarı sadece alıcı ve gönderici tarafından bilinmektedir. Ayrıca alıcı tarafında paketin şifresi çözülür.

AH ve ESP'de mesaj bütünlüğünü korumak için kullanılan, Dijital İmzayı üretmek için kullanılan hash kodları; MD5 ve SHA1 algoritmaları tarafından oluşturulur. Gönderici ve alıcı tarafında hangi oturum anahtarı algoritmasının kullanılacağını seçmek şirketin politikasına göre değişir.

Bilginin şifrelenmesi için kullanılan DES ve 3DES standartlarından 3DES daha pahalıdır. 3DES güçlü bir şifreleme işlemi yaptığı için güçlü bir işlemci kapasitesi gerektirir. Bu nedenle DES'e göre daha pahalıdır. Bu nedenle şirketler para ile ilgili bilgileri 3DES ile şifrelerler. Daha önemsiz olan firma, kişisel bilgiler gibi verileri DES ile şifreleyebilirler.

IPSec'de IP paketlerinin güvenliği 2 tipte sağlanır, bunlar;

İletim Modu: IP paketi içerisinde bulunan; orjinal IP başlık kısmı şifrelenmez, sadece veri kısmı şifrelenir. IP paketine daha az veri ilave edilmesi avantajıdır. IP başlık kısmı şifrelenmediği için; kaynak ve hedef IP adresleri açıktır. Rakiplerin trafik analizi yapmasına olanak sağlar. Rakipler veri paketinin kimden kime gittiğini öğrenebilir ama içindeki bilgiye erişemez. Ör: Windows 2000 Professional

Tünel Modu: IP paketi içerisinde bulunan; hem veri kısmı hem de başlık kısmı şifrelenir. Şifrelenen bu kısım yeni IP paketinin veri kısmını oluşturur. IP paketine iletim moduna göre daha fazla veri ilave ederiz. Router gibi cihazlar üzerinde şifreleme işlemi yapılır. Kaynak router paketi şifreler ve IPSec Tünel üzerinden hedef routera iletilir. Hedef router paketi deşifreler ve hedef sisteme gönderir. Tünel modun en büyük avantajı uç sistemlerde IPSec'in kullanımı için herhangi bir değişiklik gerektirmemesidir. Rakiplerin trafik analizi yapmasına izin vermez. Ör: Windows 2000 Server

5.4. GÜVENLİ SOKET DÜZEYİ (SSL-Secure Socket Layer)

SSL teknolojisi TCP/IP protokolü üzerinden çalışan, web sunucusu ve web tarayıcısı arasındaki tüm bilgi akışını koruyan bir güvenlik protokolüdür. Bütün popüler web tarayıcılarda ve web sunucularda uygulanmaktadır. Bugünün web üzerinden elektronik ticaret ve elektronik iş uygulamalarında önemli bir rolü vardır. SSL protokolü iki taraf arasında güvenli ve gizli iletişimin sağlanmasında elektronik kimlik belgelerini kullanır. SSL bağlantısı üzerinden gönderilen veriler üçüncü şahıslar tarafından bozguna uğratıldığında, bundan tarafların anında haberi olur.

Bir web sunucusunun kullanıcılarıyla SSL bağlantısı sağlayabilmesi için öncelikle sunucu tarafında bir sunucu e-kimliğinin bulunması gereklidir. SSL ile güvenliği sağlanmış bir siteye bağlanan kullanıcı sitenin URL adresinin "https:" ile başladığını görür. Bundan sonraki aşamalarda sunucu kendi e-kimliğini kullanıcıya göndererek kendini tanıtır. Eğer sunucunun e-kimliği geçerliyse ve kullanıcının tarayıcısında sunucuya e-kimlik veren Onay Kurumunun e-kimliği tanımlıysa tarayıcı kullanıcıya güvenilir bir siteye bağlandığına dair bilgi verir. Daha sonra SSL bağlantısı kurulur ve sunucu-tarayıcı arasındaki tüm veriler üçüncü şahısların mesajı okumasını önlemek amacıyla şifrelenir. Mesaj içeriğinin yolda herhangi bir şekilde değiştirilme olasılığına karşı olarak da sayısal imza ile imzalanır. Şifreli mesaj imzasıyla birlikte gönderilir ve ya alınır. Sunucu Kimlik Doğrulaması olarak adlandırılan bu işlemlerin amacı kullanıcıya bağlandığı sitenin gerçekten bağlandığını düşündüğü site olduğunun

ve sunucuya gönderilen bilgilerin gerçekten de sadece o sunucu tarafından okunabileceğinin ispatının sağlanmasıdır. Kullanıcı bundan emin olmak için SSL bağlantısı süresince sunucudan gelen her bilgiyi web sayfasının güvenlikle ilgili özelliklerine bakarak kontrol etmelidir. Web sayfalarının güvenlik bilgileri sunucunun e-kimliğini kontrol imkanı sağlar. Eğer yabancı veya farklı bir sunucunun kimliğiyle karşılaşılırsa bağlanılan sunucu bağlanıldığı sanılan sunucu değildir ve iletişimin güvenliği tehdit altındadır.

SSL'in sağladığı imkanlardan bir diğeri de kullanıcı e-kimliğinin sunucuya gönderilerek kullanıcının kendisini sunucuya tanıtmasıdır. Kullanıcı Kimlik Doğrulaması olarak adlandırılan bu işlemde ise sunucu kendini kullanıcıya tanıttığı gibi kullanıcıdan da kendisini tanıtmasını ister. Bunun için kullanıcının bir e-kimliğinin olması gerekmektedir. Sunucu tarafında yapılacak bazı ayarlar ile e-kimliği olan hangi kullanıcıların siteye bağlanmalarına izin verileceği tanımlanabilir.

5.5.GÜVENLİ ELEKTRONİK İŞLEM (SET- Secure Electronic Transaction):

Özel şirketlerin, çoğu kamu kuruluşlarının ve bir çok bireyin Web sitesi bulunmaktadır. İnternet erişimine sahip olan bireylerin ve şirketlerin sayısı gittikçe artan bir hızla büyümektedir. Bunun bir sonucu olarak şirketler elektronik ticaret için Web'te sayfalar oluşturmaya çok hevesliler. Fakat İnternet ve Web çeşitli tiplerde saldırılara çok açıktır. Bunun fark edilmesi ile güvenli Web servisleri için olan talep artmıştır.

Dünya Çapında Web, İnternet ve TCP/IP İntranetler üzerinden çalışan bir kullanıcı/servis sağlayıcı uygulamasıdır. Web, gittikçe artan bir şekilde şirket ve üretim bilgileri için ve iş dünyası işlemleri için bir platform olarak yüksek oranda şeffaf bir çıkış olarak hizmet vermektedir. Web servis sağlayıcıları yanlış yönlendirilirse ünleri zarar görebilir ve para kaybedilebilirler.

Web tarayıcılarının kullanımı kolay olmasına, Web servis sağlayıcılarının konfigürasyonu ve yönetimi kolay olmasına ve Web içeriğinin geliştirilmesi gittikçe kolaylaşmasına karşın bütün bunların altında yatan yazılım olağanüstü komplekstir. Bu kompleks yazılım bir çok potansiyel güvenlik açıklarını gizleyebilir. Web'in kısa tarihi, doğru kurulumu yapılmış ve çeşitli güvenlik saldırılarına açık yeni ve güncelleştirilmiş sistemler ile doludur.

Web servis sağlayıcısı, bir kere yanlış yönlendirildi mi, saldırgan kendisi Web'in bir parçası olmayan fakat lokal sitede servis sağlayıcıya bağlı olan verilere ve sistemlere

erişim sağlayabilir. Dikkatsiz ve eğitimsiz (güvenlik konularında) kullanıcılar Web temelli servisler için sıkça rastlanan kullanıcılardır. Bu tip kullanıcılar var olan güvenlik risklerinden haberdar değildirler ve etkili karşı önlemler almak için gerekli araçlara ve bilgiye sahip değildirler.

SET (Secure ElectronicTransaction), İnternet üzerinde kredi kartı işlemlerinin korunması için tasarlanmış açık bir güvenlik ve şifreleme şartnamesidir. MasterCard ve Visa tarafından şubat 1996'da güvenlik standartları için bir çağrı sonucu oluşmuştur. Şimdiki versiyon SETv1 ortaya çıktı. IBM, Microsoft, Netscape, RSA, Terisa ve VeriSign dahil olmak üzere ilk şartnameyi geliştirmede bir çok şirket çalışmalara katıldı. 1996'dan başlayarak kavram bir çok teste tabi tutuldu ve 1998 ile ilk SET uyumlu ürünlerin nesli piyasaya çıktı.

SET kendisi bir ödeme sistemi değildir. İnternet gibi açık bir ağ üzerinde kullanıcıların var olan kredi kartı ödeme alt yapısını güvenli bir şekilde kullanmaları için gerekli olan güvenlik protokollerinin ve formatlarının bir bütünüdür. SET üç hizmet sağlar:

- Bir işlemde katılımda bulunan tüm taraflar arasında güvenli bir iletişim kanalı sağlar.
- X.509v3 dijital sertifikalarının kullanımı ile güven sağlar.
- Bilgi, taraflara işlem için gerekli olduğu zaman ve yerde sağlandığı için gizliliği korur.

SET 'in Yerine Getirmesi Gereken Şartlar

SET şartnameleri, İnternet üzerinden ve diğer ağlar üzerinden kredi kartlarının ödeme işlemlerinin güvenliği için gerekli temel iş şartlarını listeler:

- Ödeme ve sipariş bilgilerinin gizliliğini sağlama.
- Tüm iletilen verilerin bütünlüğünü garantileme.
- Kart sahibinin kredi kartı hesabının legal kullanıcısı olduğunun kanıtlanmasını garantileme.
- Bir ticaret işletmesinin, finanssal bir kurumla ilişkileri vasıtasıyla kredi kartı işlemlerini kabul edebilmesinin doğrulanması.
- Bir elektronik ticaret işleminde tüm legal tarafların korunması için gerekli en iyi güvenlik çalışmalarının ve sistem tasarım tekniklerinin kullanılmasının garanti edilmesi.

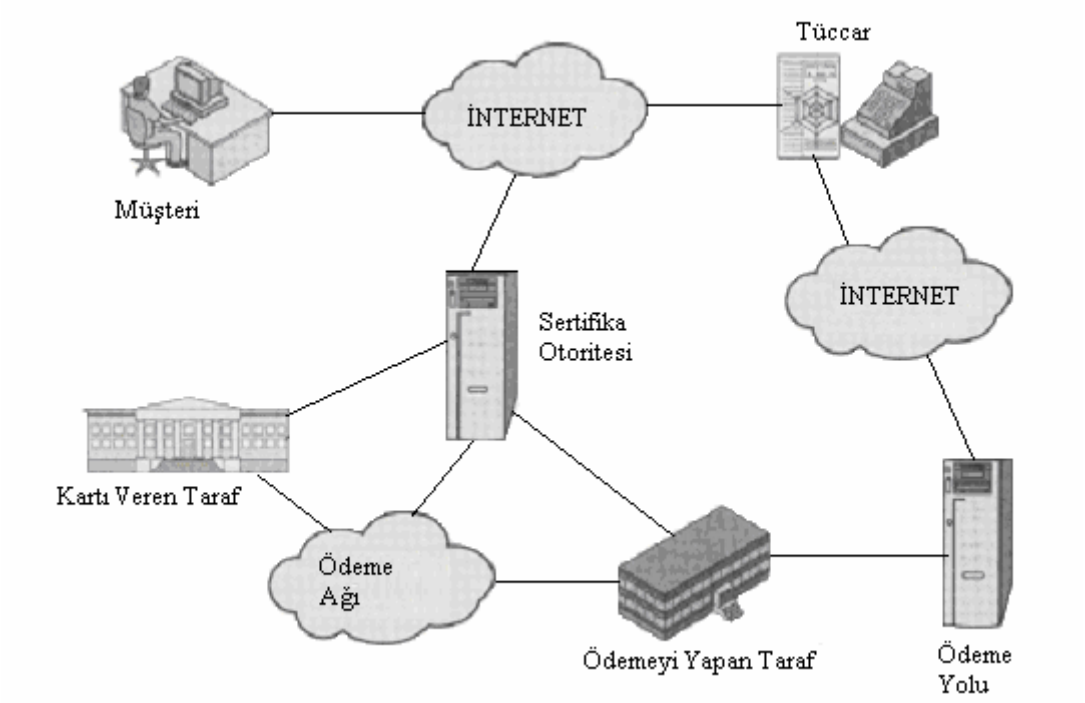
- Ne ulaşım güvenlik mekanizmalarına ne de onların kullanılmasına bağımlı olmayan bir protokol oluşturma.
- Yazılım ve ağ sağlayıcılar arasında bir arada çalıştırılabilmeyi kolaylaştırma ve destekleme.

SET'in anahtar özellikleri

- Bilgilerin gizliliği
- Verilerin bütünlüğü
- Kart sahibinin doğrulanması
- Ticaret işletmesi doğrulama

SET katılımcıları

- Kart sahipleri,Müşteri (CardHolder)
- Ticaret işletmesi sahibi, Tüccar (Merchant)
- Kartı veren taraf (Issuer)
- Ödeme yapan taraf (Acquirer)
- Ödeme yolu (Payment Gateway)
- Sertifika Otoritesi (Certificate Authority)

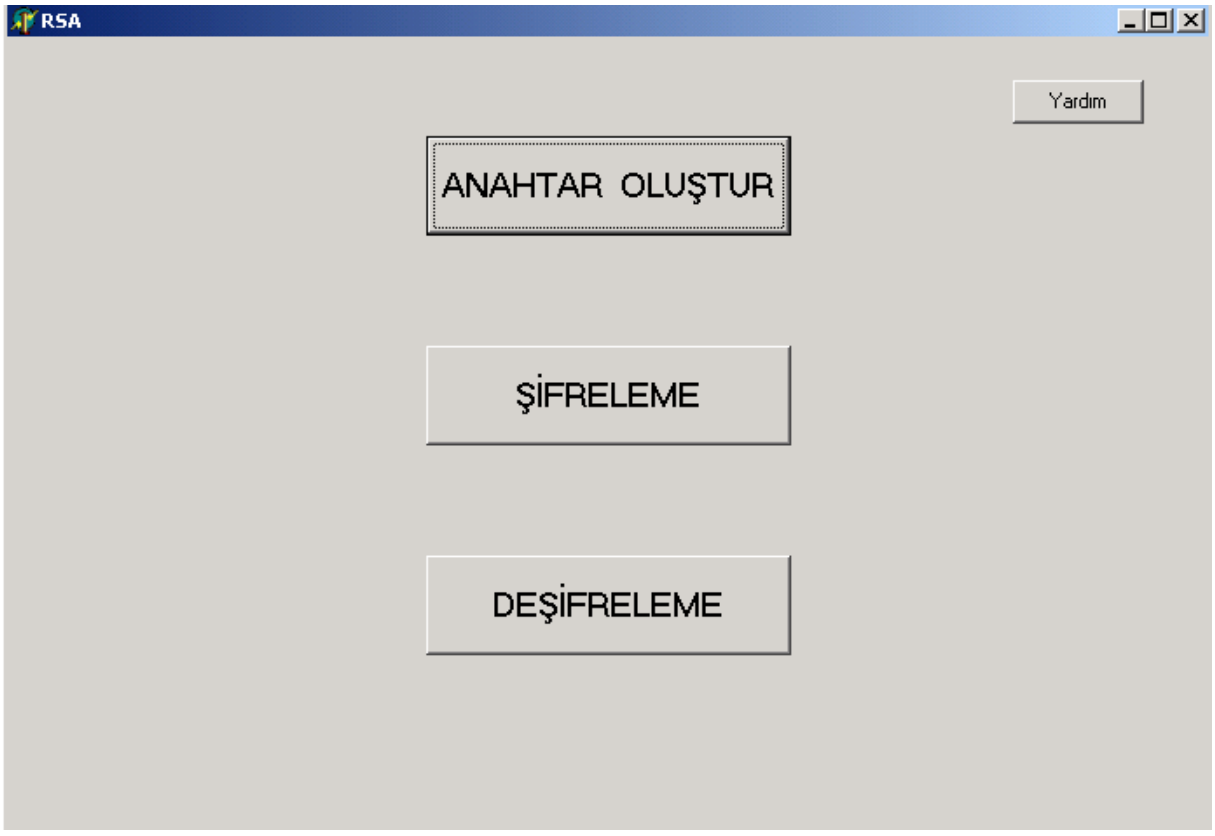


Şekil 5.4: SET Bileşenleri

Şimdi bir işlem için gerekli olan olayların inceleyelim. Öncelikle müşteri bir hesap açar. Daha sonra müşteri bir sertifika alır. Tüccar kendi sertifikalarına zaten sahiptir. Kaydını daha önceden yaptırmış sertifikalarını almıştır. Müşteri bir siparişte bulunur. Tüccarın sertifikasına bakılarak kimliği doğrulanır. Sipariş ve ödeme gönderilir. Tüccar ödeme yetkisi talep eder. Tüccar siparişi onaylar. Tüccar malları veya hizmetleri sağlar. Tüccar ödemeyi talep eder.

6. RSA'NIN YEREL OLARAK UYGULANMASI:

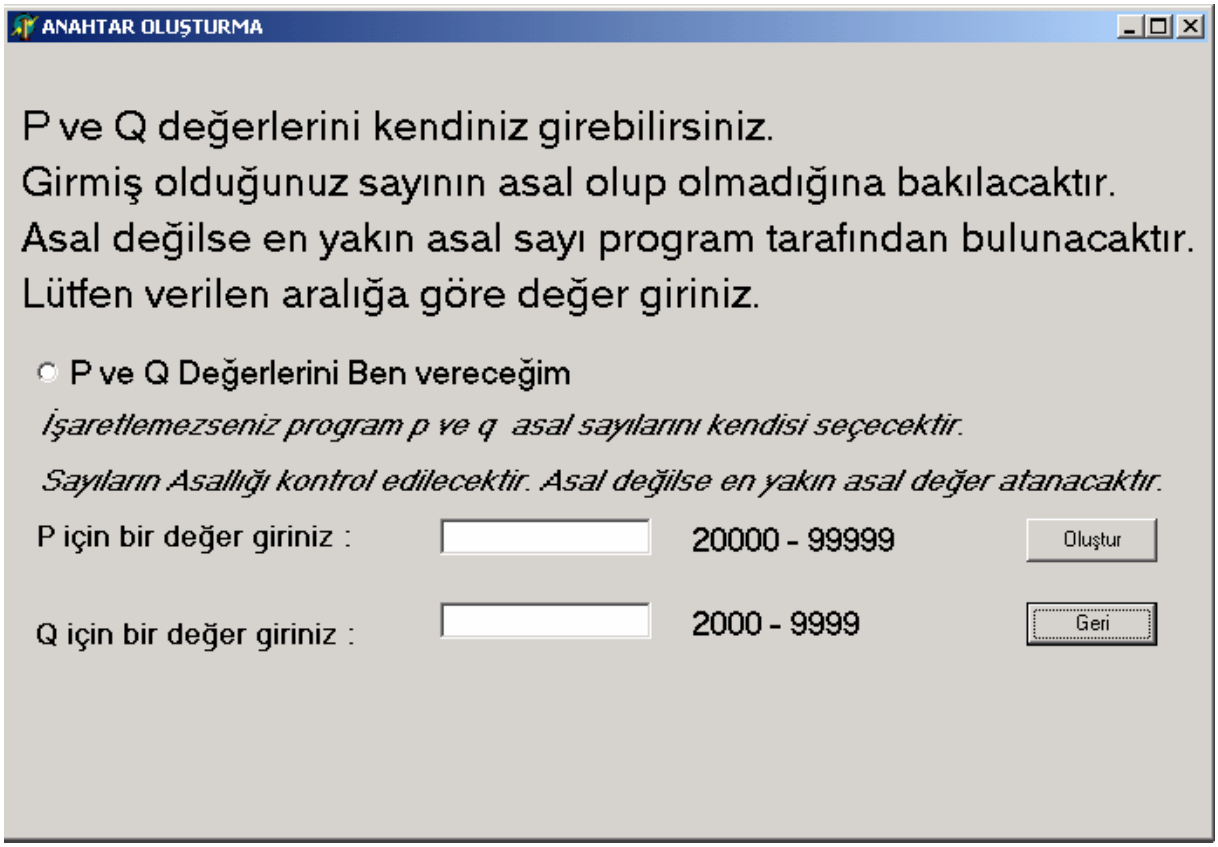
RSA algoritmasını kullanan, bir yazılım geliştirdim. Yazılımı Borland Delphi 6 programlama dilinde hazırladım. Programın amacı anahtar üretimini gerçekleştirmek. Anahtar çiftine bağlı olarak şifreleme ve deşifreleme işlemi yapmak. Tabi insan ilk etapta şu soruyo soruyor kimseyle iletişim kurmayıp yerel olarak çalışıyorsam şifrelemeye ne gerek var. Aslında yerel olarak çalışsa bile ağ üzerinde kullanamaz diye bir şart yok. Aslında yerel olması hem programın kodunu hem de özel anahtarımızı korumuş olacaktır. Anahtar çiftimi oluşturduktan sonra benim gibi programa sahip olan arkadaşlarıma açık anahtarımın olduğu txt dosyayı gönderirim. Onların açık anahtarlarınıda txt olarak alırım. Tabi kime mesaj yazacaksam onun açık anahtarını PublicKey.txt isimli dosyaya kopyalarım. Mesajımı şifreledikten sonra şifreli mesajımı normal mail yoluyla arkadaşıma gönderirim. Oda program ve özel anahtarını kullanarak mesajı deşifreler. Şimdi prgramımızı inceleyelim. RSA.exe isimli dosyayı çalıştırırız. Aşağıdaki form karşımıza çıkacaktır.



Şekil 6.1 : Ana Form

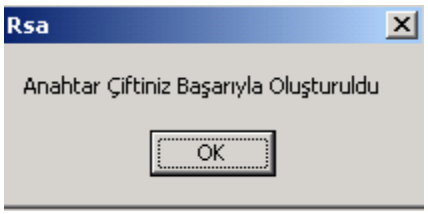
Burada "ANAHTAR OLUŞTUR" düğmesine bastığımızda anahtar üretimi formu, "ŞİFRELEME" düğmesine bastığımızda şifreleme formu, "DEŞİFRELEME"

düğmesine bastığımızda deşifreleme formu, “Yardım” düğmesine bastığımızda yardım formu açılacaktır. Aşağıdaki formu “ANAHTAR OLUŞTUR” düğmesine basınca elde ettim.

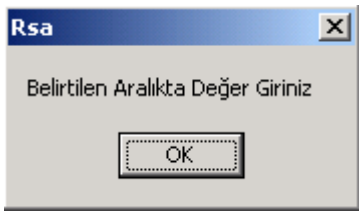


Şekil 6.2 : Anahtar Oluşturma Formu

Bu formda “OLUŞTUR” düğmesine bastığımızda anahtarı oluşturacak ve aşağıdaki mesajı verecektir.



“Geri” düğmesi ise ana forma dönmemizi sağlar. P ve Q değerlerini kendimiz girmek istiyorsak “P ve Q Değerlerini Ben vereceğim” işaretlenmeli ve belirtilen aralıkta değerler girilmelidir. Girilen değerlerin asallığı kontrol edilip en yakın asal sayı bulunacaktır. Belirtilen aralıkta değer girilmezse aşağıdaki mesajı verir.



Şimdi oluştur düğmesinin altında çalışan kodu inceleyelim:

```
var
  Form2: TForm2;
  RandSeed,p,q,s,e,d,e1,sayac4,n,Fin,se: Longword;
  k,sayac,sayac2,sayac3,s1,s2,s3:integer;
  mesaj:string;
  Dosya,Dosya2:Text;
```

Yukarıda var ile gerekli tanımlamalar yapılmıştır. Anahtarlarımı kaydedeceğim txt dosyaları oluşturmakiçin Dosya, Dosya2 değişkenlerini Text olarak atadım. Diğer değişkenleri ise sayının büyüklüğüne göre integer ve ya Longword tanımladım.

Aşağıda ise düğme tıklandığında çalışan program kodu verilmiştir.

```
procedure TForm2.Button2Click(Sender: TObject);
begin
  randomize();   {Rastgele değeri farklı almak için sistemi kullanır }
  s1:=random(9);
  s2:=random(9999);
  s3:=random(999); {s ler rastgele seçilen sayılar p ve q için}
  if (s1=0) or (s1=1) then s1:=2;
  p:=s1*10000+s2;
  q:=s1*1000+s3;
  k:=0;
```

p ve q asal sayılarını oluşturmak için s'leri kullandım. s'ler random olarak belirleniyor. Dikkat etmemiz gereken nokta p ve q nun asal olduğundan şu anda emin değiliz aşağıdaki testleri geçerlerse asal olacaklar.

```
  if radiobutton1.checked=True then
  Begin
    p:= StrToInt(Edit1.Text);
    q:= StrToInt(Edit2.Text);
  end;
```

Yukarıda değerlerin kullanıcı tarafından girilip girilmediğine bakılıyor. Kullanıcının girdiği değerleri alıyor. Aşağıda ise şart koşulan aralıkta girilip girilmediği kontrol ediliyor. İstenilen aralıkta değilse tekrar girmesini sağlıyor.

```
  if (p>=20000) and (p<=99999) and (q>=2000) and (q<=9999) then
  Begin
    repeat
      for sayac3:=2 to 330 do
      begin
        if (p mod sayac3=0) then
        begin
          k:=0;
          p:=p-1;
          break;
        end;
        k:=1;
      end;
    until k=1;
    k:=0;
    repeat  {aşağıdaki döngüde asal olan q yu bul}
      for sayac3:=2 to 100 do
      begin
```

```
      if (q mod sayac3=0) then
      begin
        k:=0;
        q:=q+1;
        break;
      end;
      k:=1;
      if (q=p) then k:=0;
    end;
  until k=1;
```

Yukarıdaki repeat-until bloklarında, p ve q'nun asallıklarına bakılıyor. Asal değillerse en yakın asalı buluyor. Daha sonra p ve q değerlerini kullanarak n ile $\Phi(n)$ değişkenleri hesaplanır. Açık anahataın parçası olan e rastgele seçilir. e ve $\Phi(n)$ kullanılarak d hesaplanır.

```
n:=p*q;
Fin:=(p-1)*(q-1);
randomize();
e1:=random(9);
if (e1=0) or (e1=1) then e1:=2;
e:=random(999999);
e:=e+e1*1000000;
k:=0;
repeat
  for sayac3:=2 to 10000 do
  begin
    if (e mod sayac3=0)and(Fin mod sayac3=0)then
    begin
      k:=0;
      e:=e-1;
      break;
    end;
    k:=1;
  end;
until k=1;
s:=e;
for sayac4:=2 to Fin-1 do
Begin
  s:=s+e;
  if s<>0 then s:=s mod Fin;
  if (s=1) then
  begin
    d:=Sayac4;
    break;
  end;
end;
```

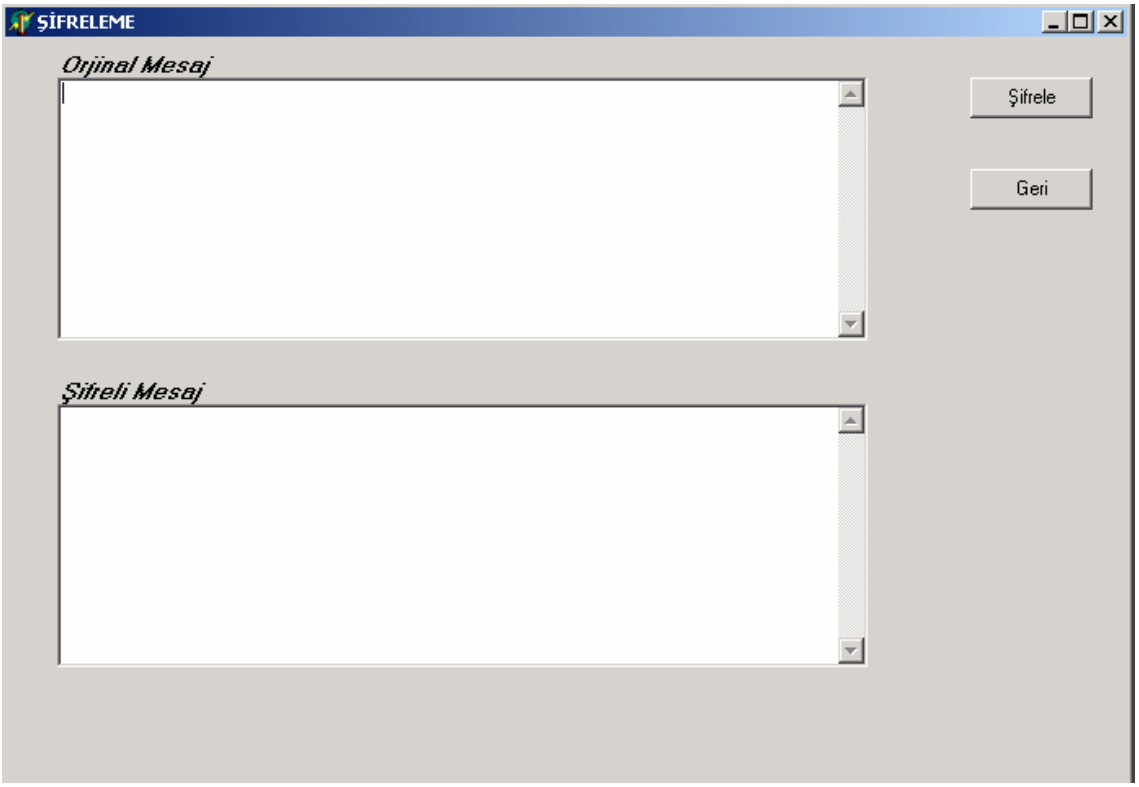
Tüm hesaplamalar bittikten sonra PublicKey.txt ve PrivateKey.txt dosyaları oluşturulur. PublicKey.txt 'nin içerisine e ve n, PrivateKey.txt'nin içerisine d ve n kaydedilir.

```
AssignFile(Dosya,'PublicKey.txt');
Rewrite(Dosya);
writeln(Dosya,e,';',n);
CloseFile(Dosya);
AssignFile(Dosya2,'PrivateKey.txt');
Rewrite(Dosya2);
writeln(Dosya2,d,';',n);
CloseFile(Dosya2);
```

```
    mesaj:='Anahtar Çiftiniz Başarıyla Oluşturuldu';
    ShowMessage(mesaj);
  end else showmessage('Belirtilen Aralıkta Değer Giriniz');
end;
```

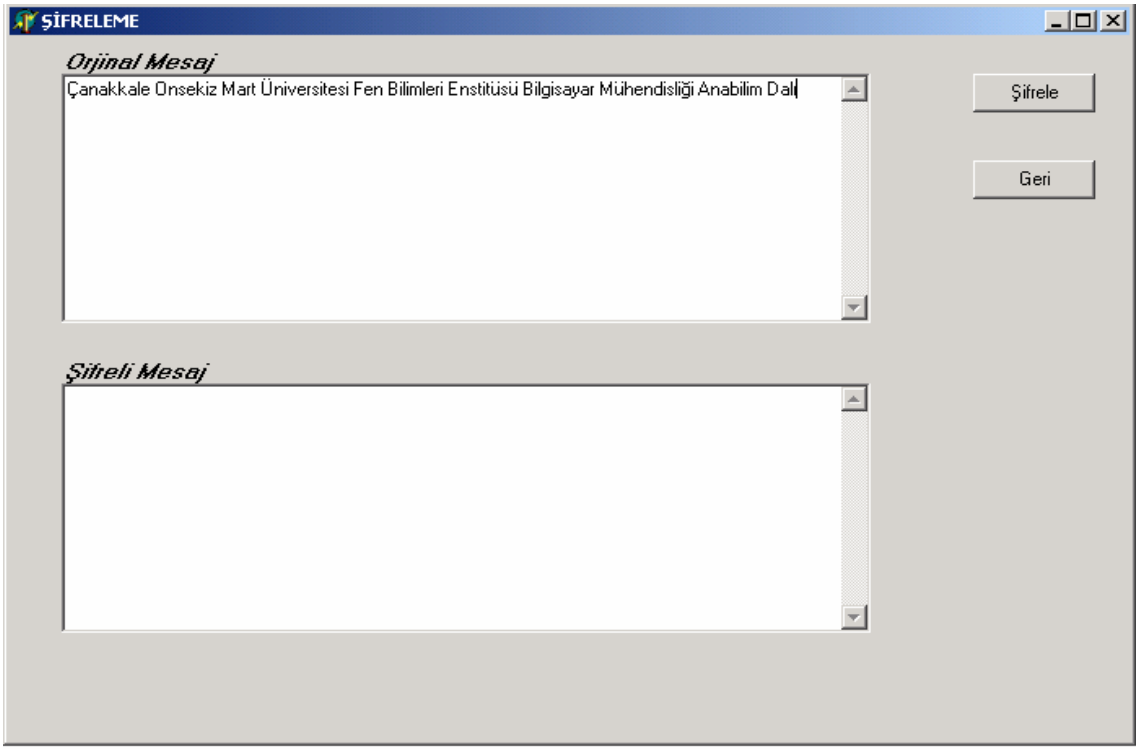
Şimdi tekrar gözden geçirelim. Rasgele seçilen s'lerden, p ve q asal sayıları oluşturulur. Eğer kullanıcı kendi girmek istiyorsa, if ile radiobutton kontrol edilip klavyeden girilen değerler p ve q ya atanır. Değer aralığı if ile kontrol edilir. Belirtilen aralıkta değilse if bloğunun içerisine girmez. Kullanıcıya hata mesajı verir. Repeat-until bloğu içerisinde p ve q ların asallığı tespit edilir. p ve q seçildikten sonra n değişkeni ve Fin değişkeni hesaplanır. Daha sonra açık anahtarı oluşturan e değişkeni seçilir. e değişkeninin asallığı kontrol edilir. $d=e^{-1} \bmod \Phi(n)$ formülüne göre e ve fin değişkenlerinden d hesaplanır. Bu hesaplamalar bittikten sonra PublicKey.txt ye açık anahtar, PrivateKey.txt ye özel anahtar kaydedilir. PublicKey.txt, e ve n değişkenlerinden; PrivateKey.txt, d ve n değişkenlerinden oluşur.

Aşağıdaki formu "ŞİFRELEME" düğmesine basınca elde ettim.



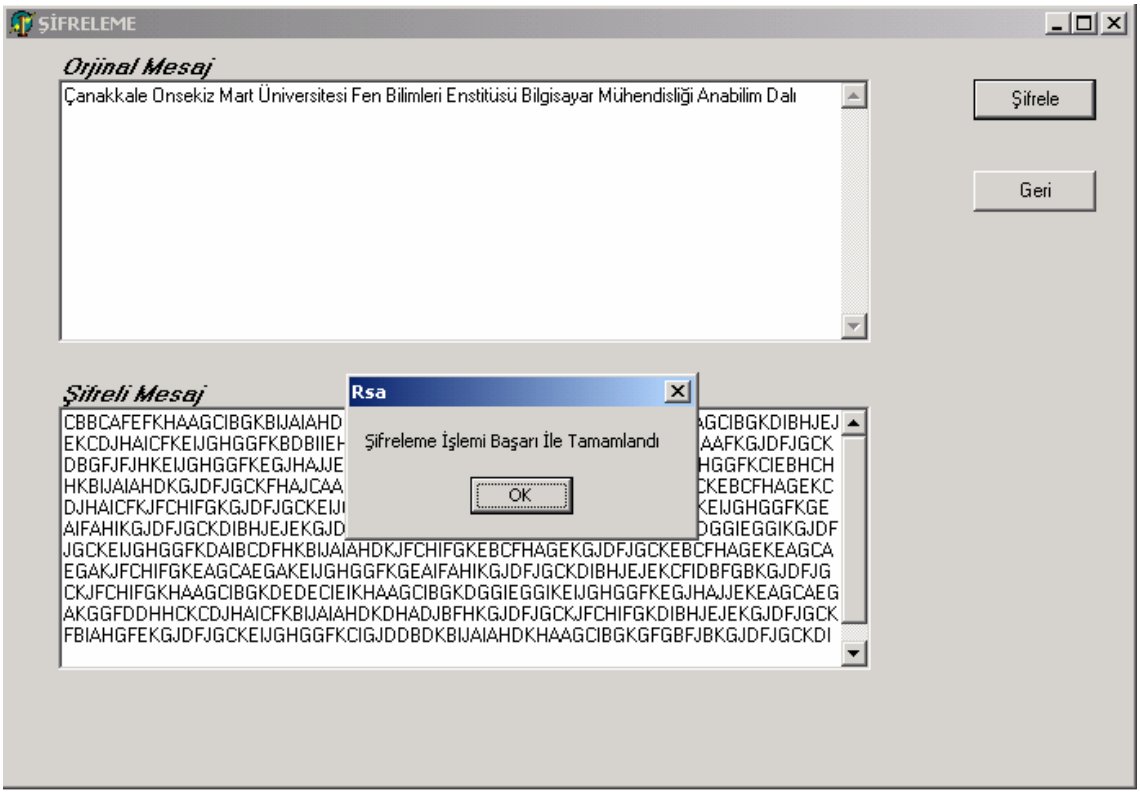
Şekil 6.3 :Şifreleme Formu

Orijinal mesaj alanına şifrelemek istediğimiz mesajı yazarız. "Şifrele" düğmesine basınca mesajımızı şifreleyip şifreli mesaj alanına yazar. Geri düğmesi ana forma döndürür.



Şekil 6.4 : Şifreleme Formuna Mesaj Yazılması

Yukarıdaki formda şifrele düğmesine basınca aşağıdaki formu elde ettik.



Şekil 6.5 : Mesajın Şifrelenmesi

Şimdi şifrele düğmesinin altında çalışan program koduna bakalım.

```
var
  Form3: TForm3;
  mesaj,satir,satire,satirn,orjinal,sifreli,sifreli2,ikili:string;
  s,i,j,bayrak,sorjinal,sifuz,kalan,us,sayac:integer;
  dosya:Text;
  n,e,e2,M,esayac,tam,aradeger:longword;
  C:Int64;
```

Yukarıda var la gerekli tanımlamalar yapıldı.

```
procedure TForm3.Button1Click(Sender: TObject);
begin
  AssignFile(Dosya,'PublicKey.txt');
  Reset(Dosya);
  Readln(Dosya,Satir);
  s:=Length(satir);
  bayrak:=0;
  for i:=1 to s do
  Begin
    if (satir[i]=';') then
    Begin
     bayrak:=1;
     continue;
    end;
    if (bayrak=1) then satirn:=satirn+satir[i];
    if (bayrak=0) then satire:=satire+satir[i];
  End;
  e:=StrToInt(satire);
  n:=StrToInt(satirn);
  satire:='';
  satirn:='';
```

Yukarıda öncelikle PublicKey.Txt isimli dosyayı okuduk. Daha sonra txt ifademizdeki karakterleri tektek ayırarak e ve n ‘yi elde ettik.

```
  orjinal:=memo1.Text;
  sorjinal:=Length(orjinal);
  e2:=e;
  ikili:='';
  repeat
  begin
    tam:=e2 Div 2;
    kalan:= e2 Mod 2;
    ikili:= IntToStr(kalan)+ikili;
    e2:=tam;
  end;
  until tam=0;
  us:=length(ikili);
```

Yukarıda öncelikle orijinal mesajımızı okuduk ve uzunluğunu tespit ettik. Daha sonra e sayımızı ikili hale çevirdik. Uzunluğunu us isimli değişkene atadık.

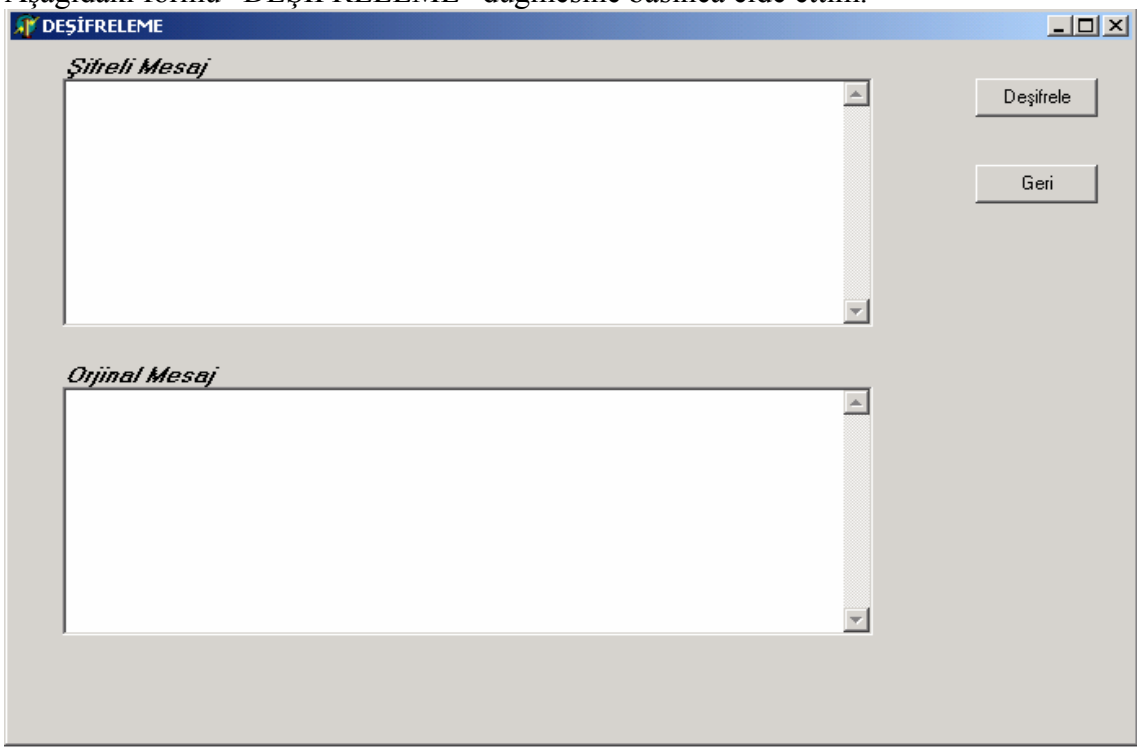
```
  for j:=1 to sorjinal do
  Begin
    M:=Ord(Orjinal[j]);
    C:=1;
    aradeger:=0;
    for sayac:=1 to us do
    Begin
      aradeger:=aradeger*2;
      C:=(C*C) mod n;
      if ikili[sayac]='1' then
```

```
      Begin
         aradeger:=aradeger+1;
         C:=(C*M)mod n
      end;
    end;
    sifreli2:=IntToStr(C);
    sifuz:=Length(sifreli2);
    for i:=1 to sifuz do
    Begin
      sifreli:=sifreli+Chr(65+StrToInt(sifreli2[i]));
    end;
    Sifreli:=Sifreli+'K';
  end;
```

Yukarıda orjinal mesajımızı karakter karakter ayırıp önce ASCII kodunu aldık. Daha sonra gelen sayısal değerin e. Kuvvetini mod n ye göre bulduk. Üs alma işlemini e 'nin ikili halini kullanarak yaptık. Böylece işlem süresini kısalttık. RSA konusu dönülüp incelendiğinde ikili alma yöntemiyle üs hesabı görülecektir. Sonuçta C değişkenini elde edeceğiz. C değişkeni mod n ye uyan birtamsayı olacaktır. C değişkenini de karakter karakter ayırıp 65 ilave edip ASCII karşılığı olan harf bulunur. K ise karakterlerin ayrım noktası olarak kullanılır. Döngüler bittiğinde şifreli mesaj elde edilecektir. Şifreli mesajı yazdırıp şifreleme işini bitiririz.

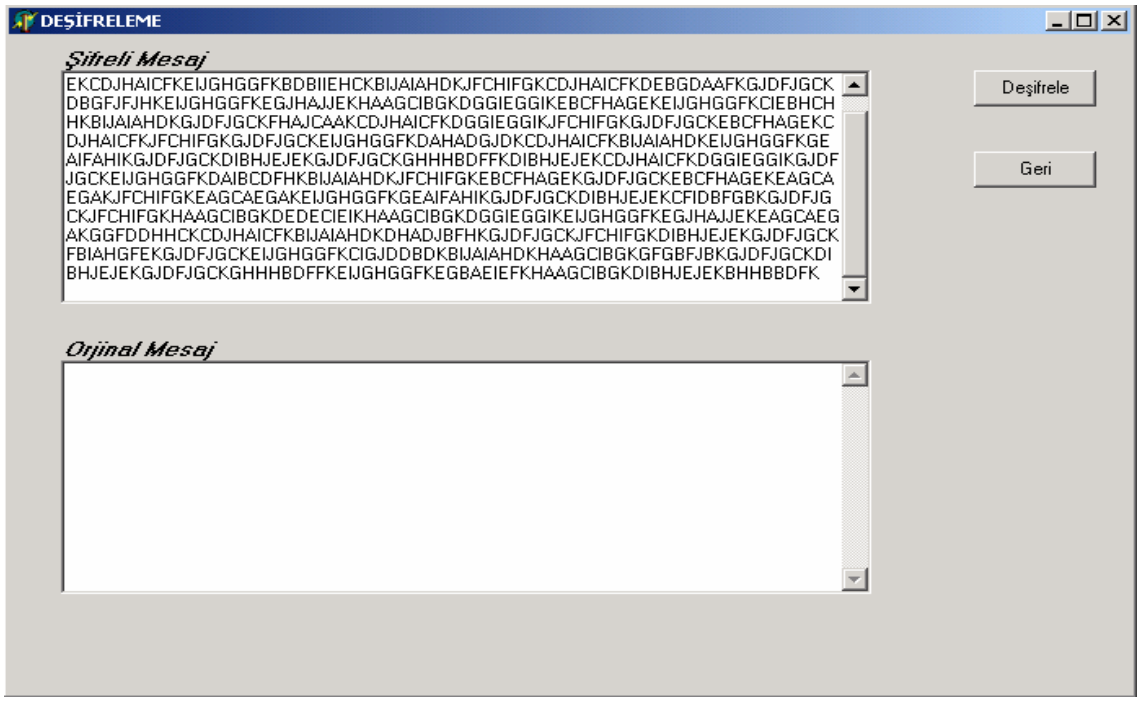
```
  memo2.Text:=Sifreli;
  sifreli:='';
  mesaj:='Şifreleme İşlemi Başarı İle Tamamlandı';
  ShowMessage(mesaj);
end;
```

Aşağıdaki formu "DEŞİFRELEME" düğmesine basınca elde ettim.



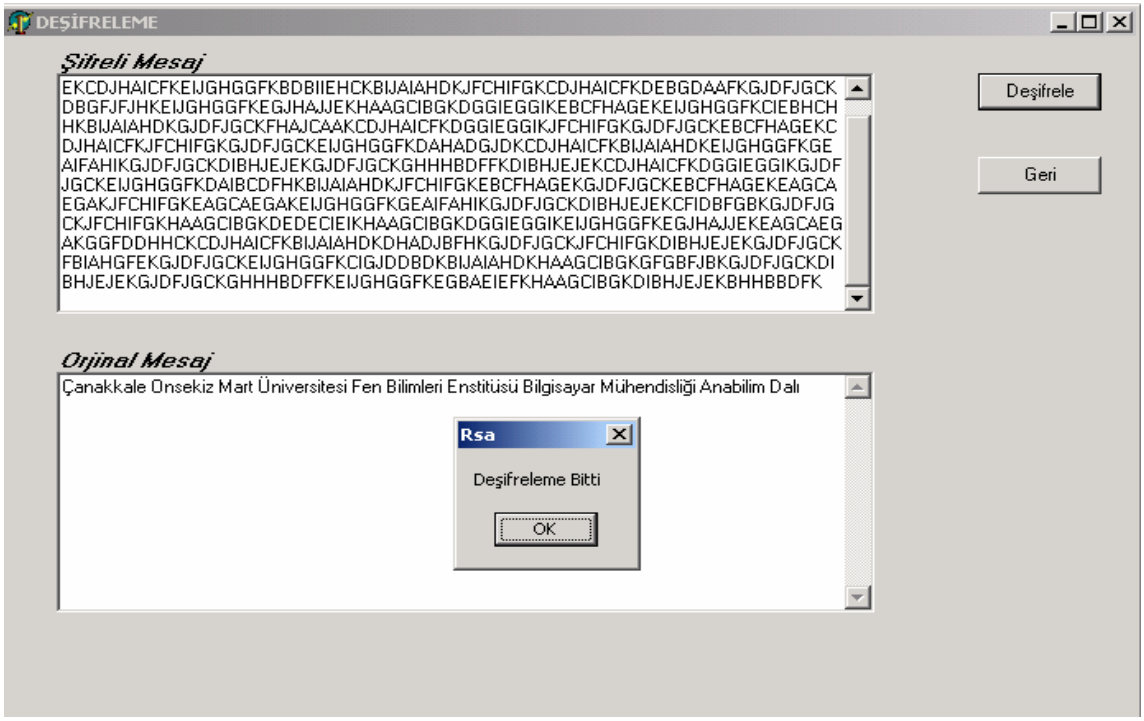
Şekil 6.6 : Deşifreleme Form

Şifreli mesaj alanına deşifrelemek istediğimiz mesajı kopyalarız. “Deşifrele” düğmesine basınca mesajımızı deşifreleyip Orjinal mesaj alanına yazar. Geri düğmesi ana forma döndürür.



Şekil 6.7 : Deşifreleme Formuna Mesaj Yazılması

Yukarıdaki formda deşifrele düğmesine basınca aşağıdaki formu elde ettik.



Şekil 6.8 : Deşifreleme İşlemi

Şimdi Deşifrele düğmesinin altında çalışan program koduna bakalım.

```
var
  Form4: TForm4;
  Dosya:Text;
  mesaj,satir,satird,satirn,sifreli,sifreli2,orjinal,ikili:string;
  s,bayrak,i,sifuz,j,sifdeger,dsayac,kalan,us,sayac:integer;
  d,n,C,tam,aradeger,d2:Longword;
  M:int64;
```

Yukarıda var la gerekli tanımlamalar yapıldı.

```
procedure TForm4.Button1Click(Sender: TObject);
begin
  AssignFile(Dosya,'PrivateKey.txt');
  Reset(Dosya);
  Readln(Dosya,Satir);
  s:=Length(satir);
  bayrak:=0;
  for i:=1 to s do
  Begin
    if (satir[i]=';') then
    Begin
     bayrak:=1;
     continue;
    end;
    if (bayrak=1) then satirn:=satirn+satir[i];
    if (bayrak=0) then satird:=satird+satir[i];
  End;
  d:=StrToInt(satird);
  n:=StrToInt(satirn);
  satird:='';
  satirn:='';
```

Yukarıda öncelikle PrivateKey.Txt isimli dosyayı okuduk. Daha sonra txt ifademizdeki karakterleri tektek ayırarak d ve n ‘yi elde ettik.

```
  d2:=d;
  ikili:='';
  repeat
  begin
    tam:=d2 Div 2;
    kalan:= d2 Mod 2;
    ikili:= IntToStr(kalan)+ikili;
    d2:=tam;
  end;
  until tam=0;
  us:=length(ikili);
  sifreli:=memo1.Text;
  sifuz:=length(sifreli);
  bayrak:=0;
  sifreli2:='';
```

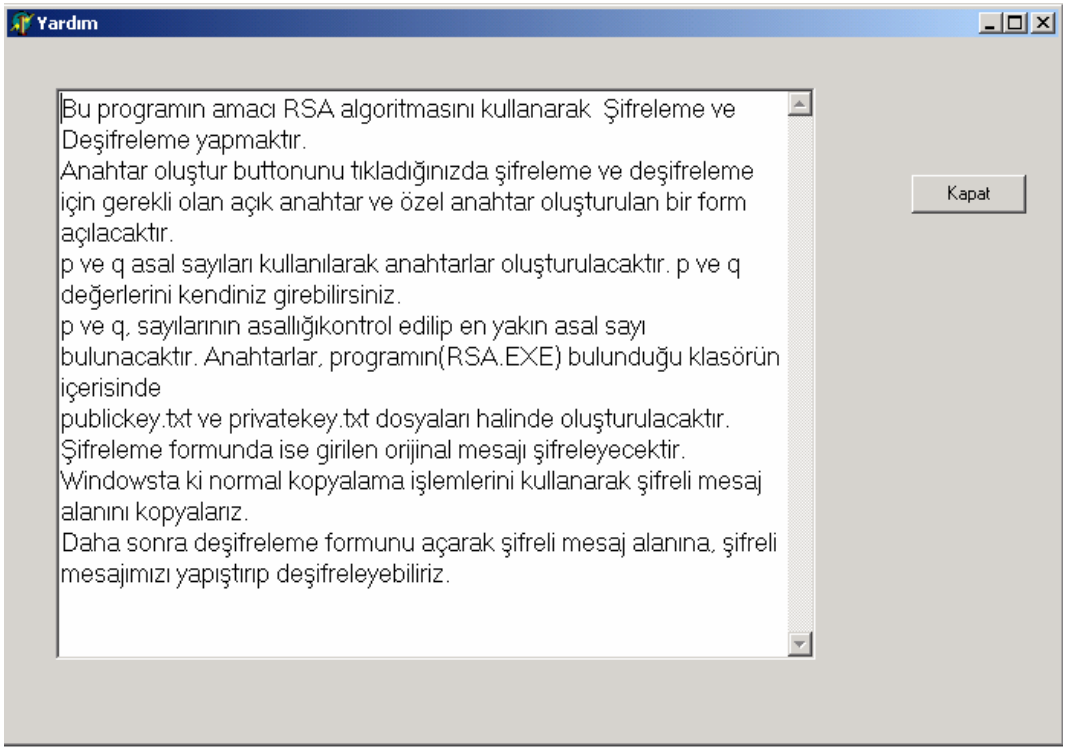
Yukarıda öncelikle d sayımızı ikili hale çevirdik. Uzunluğunu us isimli değişkene atadık. Daha sonra şifreli mesajımızı okuduk ve uzunluğunu tespit ettik.

```
  for j:=1 to sifuz do
  Begin
    if (sifreli[j]='K') then
    Begin
     bayrak:=1;
    end;
    if (bayrak=0) then
    Begin
      sifdeger:=Ord(sifreli[j])-65;
      sifreli2:=sifreli2+IntToStr(sifdeger);
    end;
    if (bayrak=1) then
    Begin
      C:=StrToInt(sifreli2);
      M:=1;
      aradeger:=0;
      for sayac:=1 to us do
      Begin
        aradeger:=aradeger*2;
        M:=(M*M) mod n;
        if ikili[sayac]='1' then
        Begin
          aradeger:=aradeger+1;
          M:=(M*C)mod n;
        end;
      end;
      orjinal:=orjinal+Chr(M);
      bayrak:=0;
      sifreli2:='';
    end;
  End;
```

K ayrımlarına dikkat ederek şifrelerken karakterlerini ayırdığımız C tamsayısını bulduk. Daha sonra C tamsayılarının d. kuvvetlerini aldık. Üs alma işlemini şifreleme yaparken kullandığımız yolla yaptık. Sonuçta M tamsayısını elde ettik. M tamsayısı şifrelenmemiş karakterimizin ASCII kodu karşılığıdır. Bizde bu tamsayımızı karaktere çevirdik. Döngüler sayesinde tüm karakterler için aynı işlemleri yaptık. Sonuçta orijinal mesajımızı elde ettik. Ve memo kullanarak ekrana yazdırdık.

```
  memo2.Text:=Orjinal;
  Orjinal:='';
  mesaj:='Deşifreleme Bitti';
  ShowMessage(Mesaj);
end;
```

Aşağıda yardım düğmesine basılınca gelen form gözükmektedir.

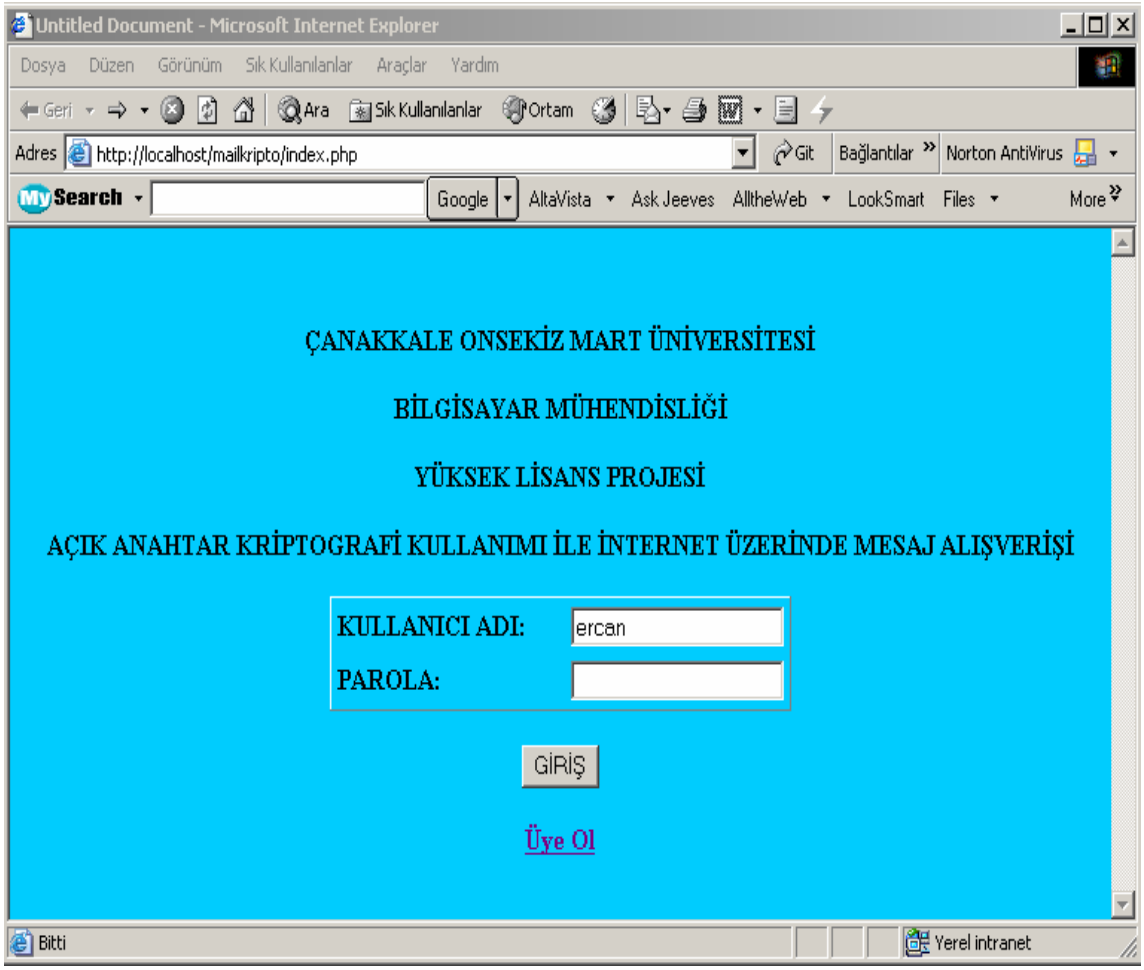


Şekil 6.9: Yardım Formu

7- RSA ALGORİTMASININ KULLANIMI İLE İNTERNET ÜZERİNDEN ŞİFRELİ MESAJ ALIŞ VERİŞİ YAPILMASI:

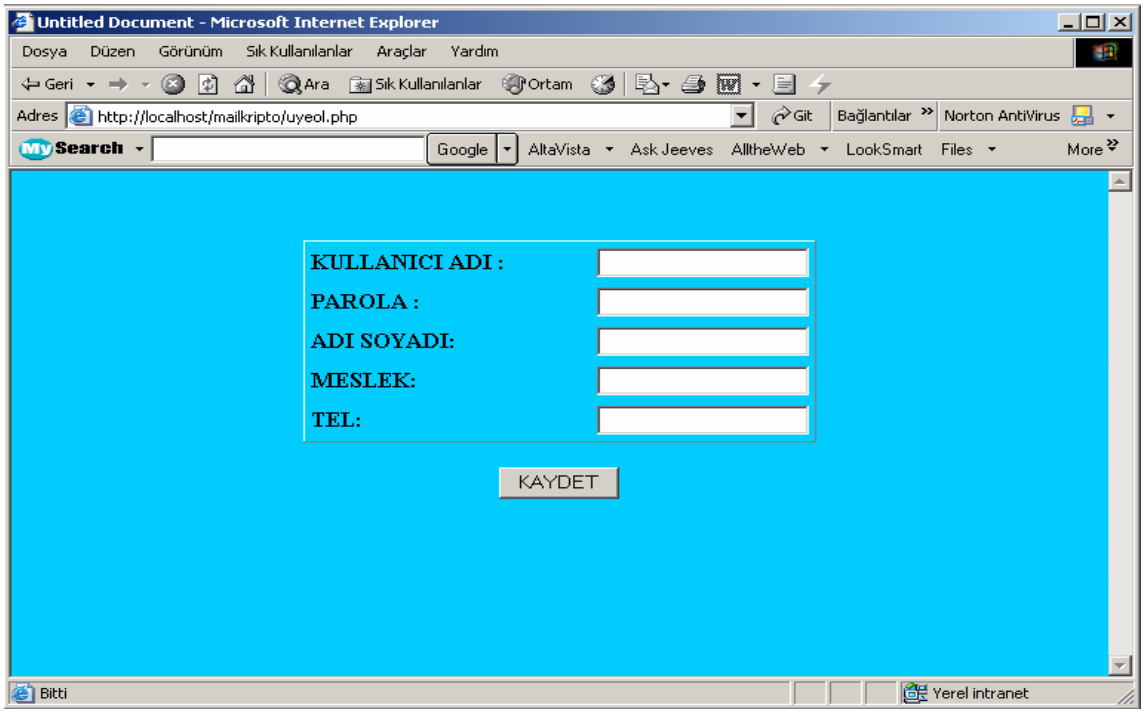
Bu uygulamadaki amacımız kullanıcıların internet üzerinden meajlarını şifreleyerek iletmeleridir. Uygulamayı hazırlarken Dreamweaver, JavaScript, PHP ve MySQL kullandım. MySQL ile veri tabanını oluşturdum. Dreamweaver vasıtasıyla görsellik sağlayıp JavaScript ve PHP kodu kullandım. Yapılacak işleme göre Java Script ve ya PHP kullandım. PHP server üzerinde, JavaScript ise kullanıcının bilgisayarında çalışmaktadır. Anahtar oluşturma , mesajın şifrelenmesi gibi gizliliğin önemli olduğu uygulamaları JavaScript kullanarak yaptım. http://193.255.97.173/mailkripto/index.php adresinden uygulamamıza erişebiliriz. Şimdi uygulamamızı inceleyelim.

İlk olarak Şekil 7.1'deki sayfa karşımıza çıkar. Kullanıcı adı ve parolamızı girerek giriş yapabiliriz. Üye değilsek Üye Ol linkine tıklayarak üye olabiliriz.



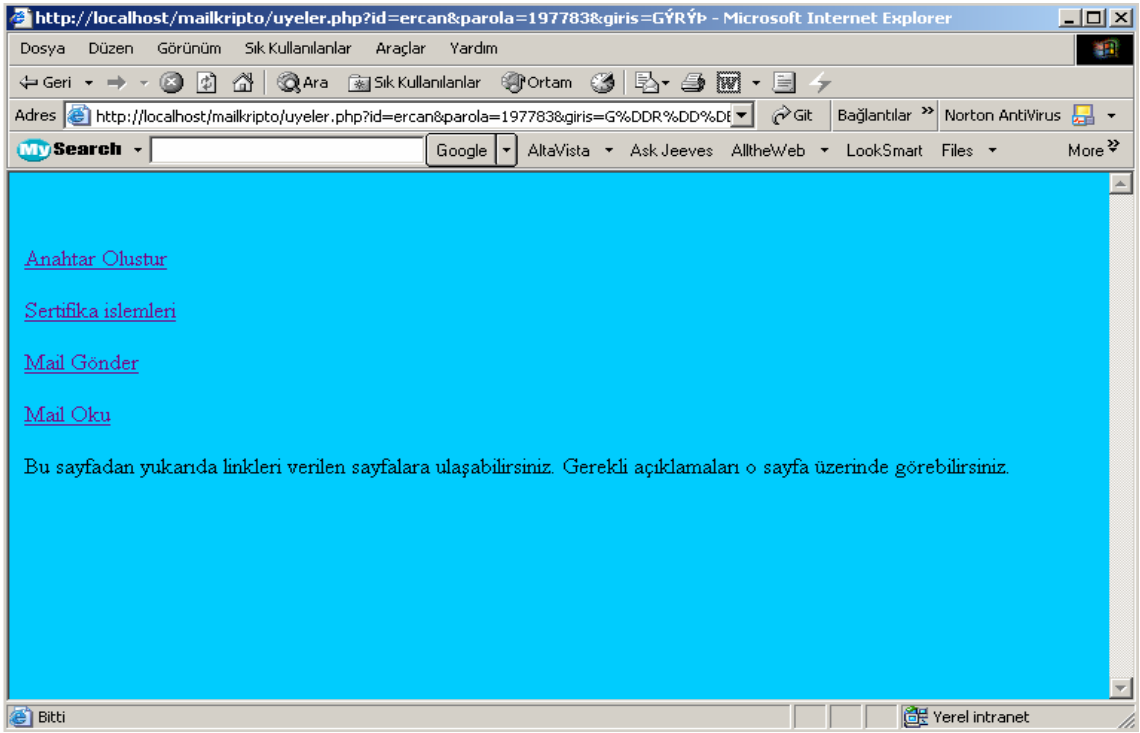
Şekil 7.1: Kullanıcı girişi

Şekil 7.2'deki sayfayı üye ol linkine tıklayınca elde ettim. Gerekli alanları doldurup kaydet dediğimde üye olabiliriz. MySQL 'de user isimli tabloya kayıt yapacaktır.



Şekil 7.2: Üye Kayıt Formu

Ana formdan kullanıcı adı ve parolayı girip, GİRİŞ butonuna tıkladığımda Şekil 7.3 'deki sayfayı elde ettim. Yapmak istediğim işlemin linkine tıklarım.

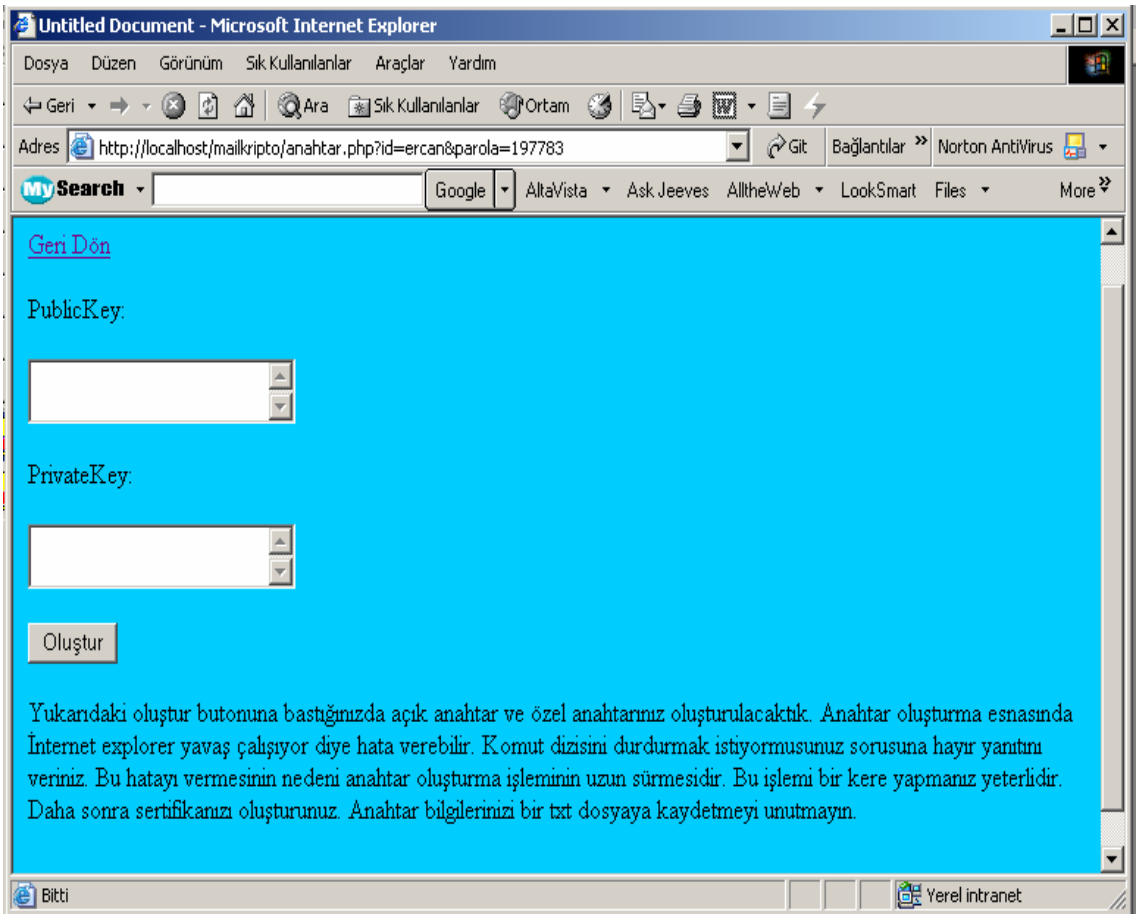


Şekil 7.3: İşlem seçenekleri

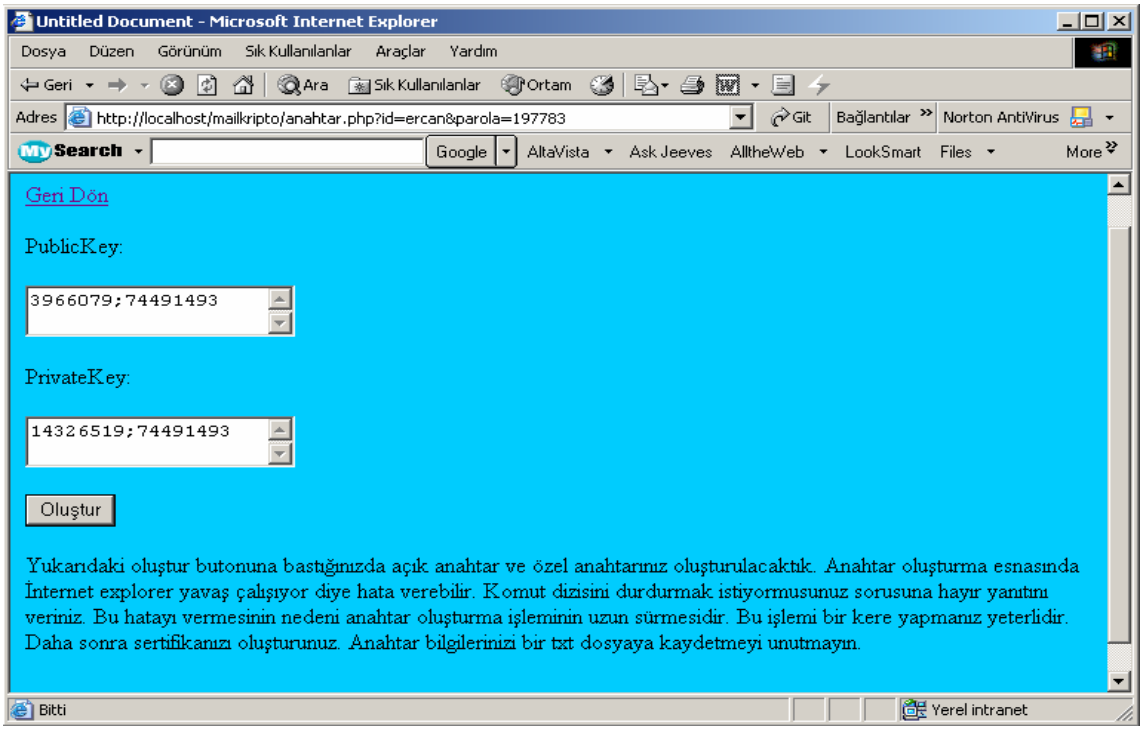
Anahtar Oluştur linkini tıklayınca Şekil7.4 de görülen sayfayı elde ettim. Burada oluştur butonuna basarak anahtarlarımı oluşturacağım. Her kullanıcının bir çift anahtarı vardır.

Açık anahtar ve özel anahtar. Açık anahtar kullanılarak sertifika oluşturulur. Mesajın gönderen kişi sertifikamızın içerisindeki açık anahtar vasıtasıyla bize gönderceği mesajı şifreler. Açık anahtar, sertifikanın dağıtılması nedeniyle herkes tarafından bilinebilir. Özel anahtarsa deşifreleme için kullanılacağından yalnızca sahibi tarafından bilinmelidir. Bu işlemler kullanıcının bilgisayarında yapılması gerektiği için JavaSciptle yaptım. Şekil 7.5'de oluştur butonuna bastıktan sonraki sayfa görülmektedir.

Anahtar oluştururken önelikle p ve q isimli iki asal sayı belirledim. Daha sonra bu asal sayıların çarpımıyla n=p*q elde ettim . Fin=(p-1)*(q-1) 'i buldum. Fin ile aralarında asal olmak koşuluyla e değişkenini hesapladım. d değişkenini e*d mod Fin=1 olamk koşuluyla hesapladım. e,n açık anahtarımı; d,n ise özel anahtarımı oluşturdu. Tezimin ek kısmında bulunan anahtar.php 'nin kaynak kodunu inceleyerek yapılan işlemleri görebilirsiniz.

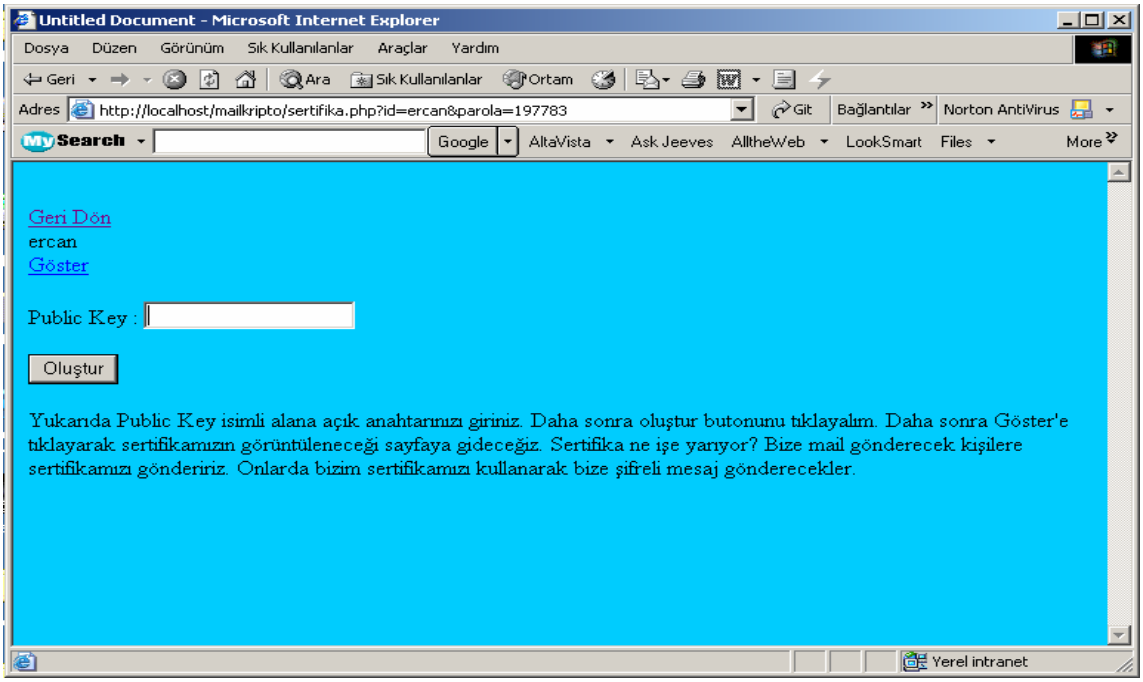


Şekil 7.4: Anahtar Oluşturma Formu



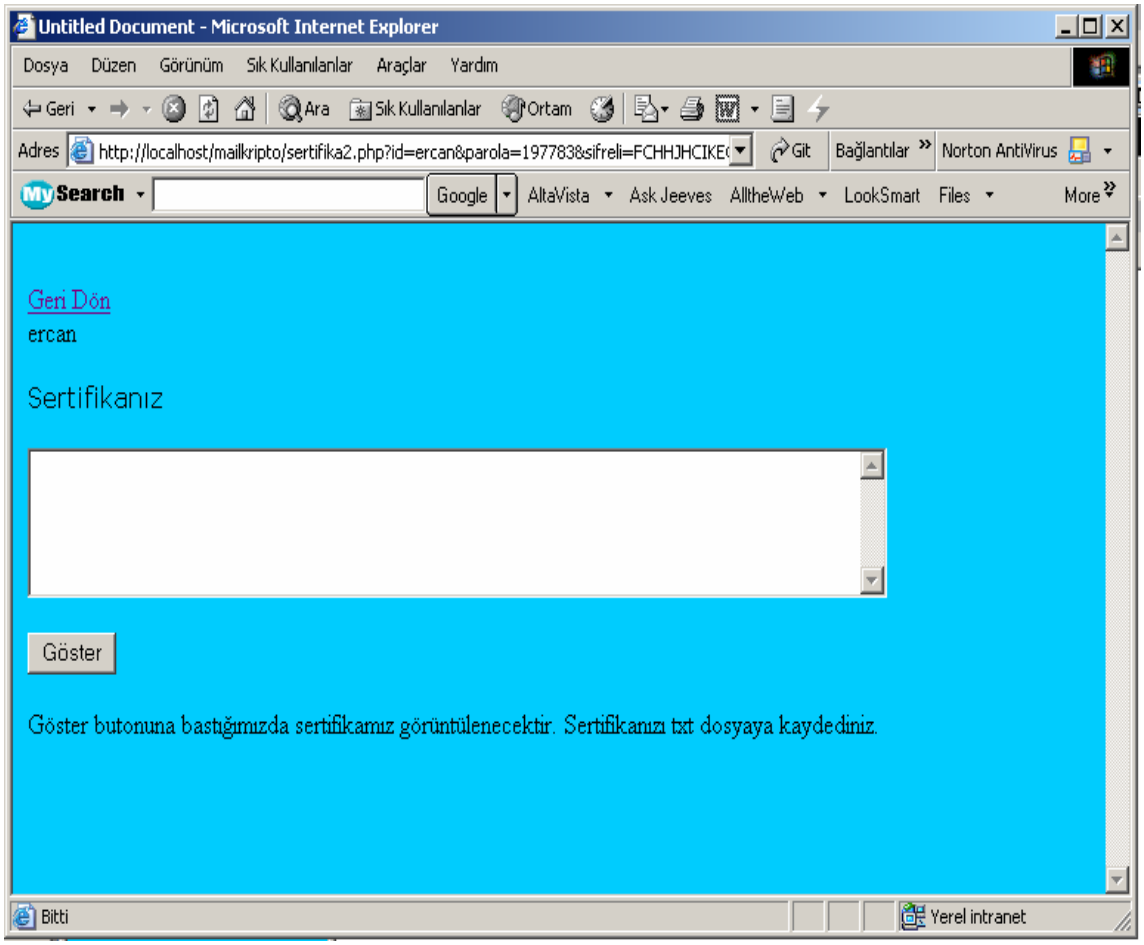
Şekil 7.5: Anahtar Oluşturulduktan sonraki durum

Sertifika işlemleri linkine tıklayınca Şekil 7.6 'daki sayfayı elde ettim. Public Key isimli alana açık anahtarımızı kopyalayıp oluştur dediğimizde sertifikamız oluşturulacaktır. Sertifikamız oluşturuluken kullanıcı bilgisi, açık anahtar ve tarih yöneticinin özel anahtarı ile şifrelenecektir. Bu işlemler serverda yapılması gerektiği için PHP kullandım. Peki neden serverda yapılmalı?

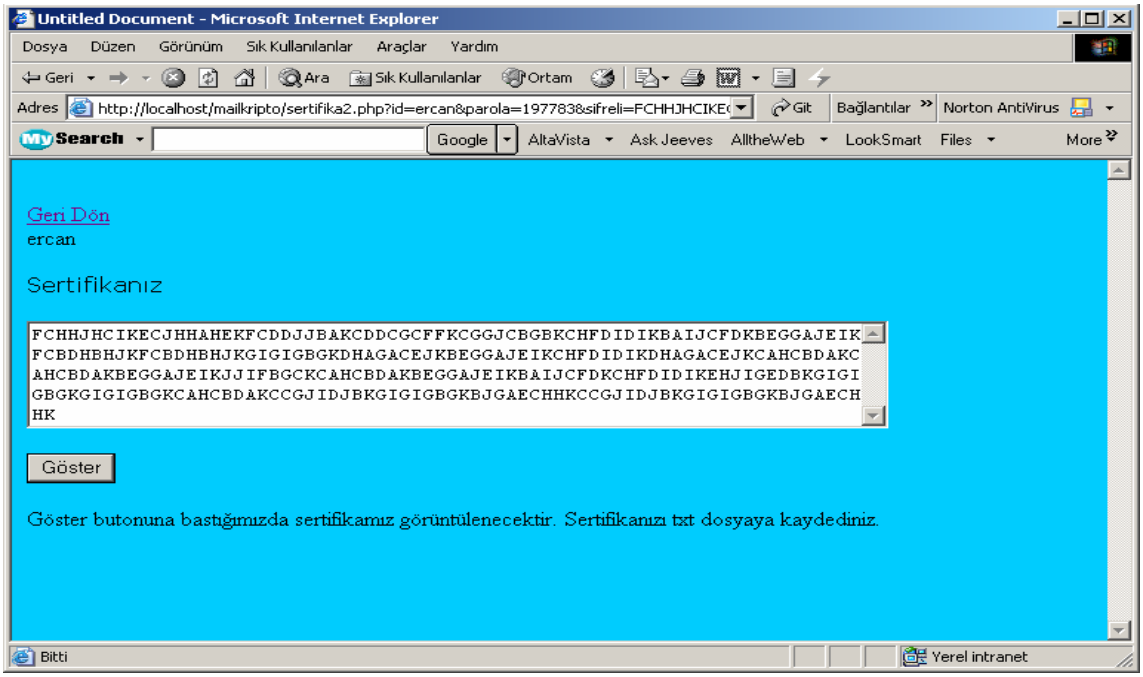


Şekil 7.6: Sertifika oluşturulması

MySQL veritabanında yönetici isimli tablodan yöneticinin özel anahtarı alınacak. Eğer ağ üzerinde yöneticinin özel anahtarını kullanıcının bilgisayarına götürüp orda işlem yaparsam, ağı dinleyen bir rakip yöneticinin özel anahtarını ele geçirebilir. Bundan dolayı sertifikanın imzalanması serverda yapıldı. Sertifika sahibi mesajlaşmak istediği kişilere sertifikasını dağıtır. Sertifikayı alanlar ise yöneticinin imzası olduğu için sertifikaya güvenirler. Setifikaları mesaj göndermek için kullanacaklardır. Şekil 7.6 daki göster linkine tıklayınca, Şekil 7.7 deki sayfa açılır. Bu sayfada Göster butonuna tıkladığımızda Şekil 7.8'de gördüğümüz sayfa açılacak ve sertifikamız görülecektir. Sertifikamızı txt bir dosyaya kaydedip dostlarımıza dağıtabiliriz.

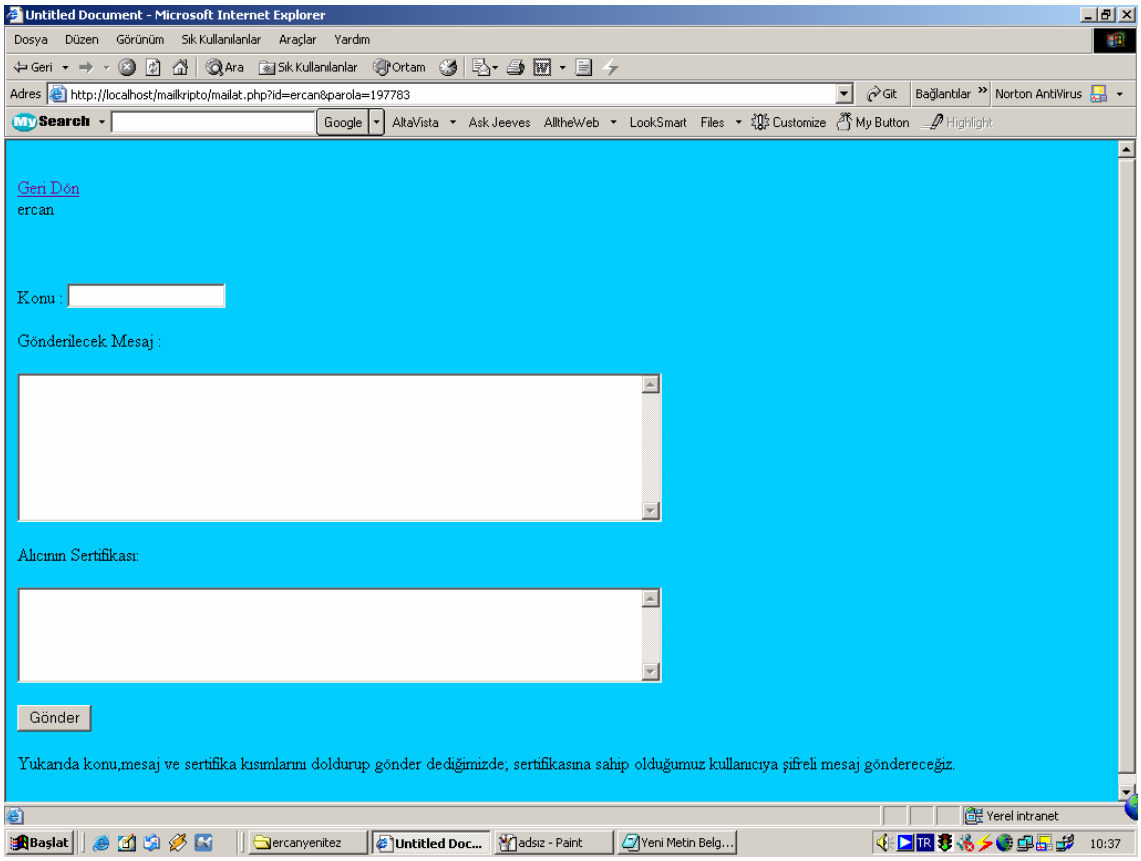


Şekil 7.7: Sertifikanı Gösterileceği sayfa



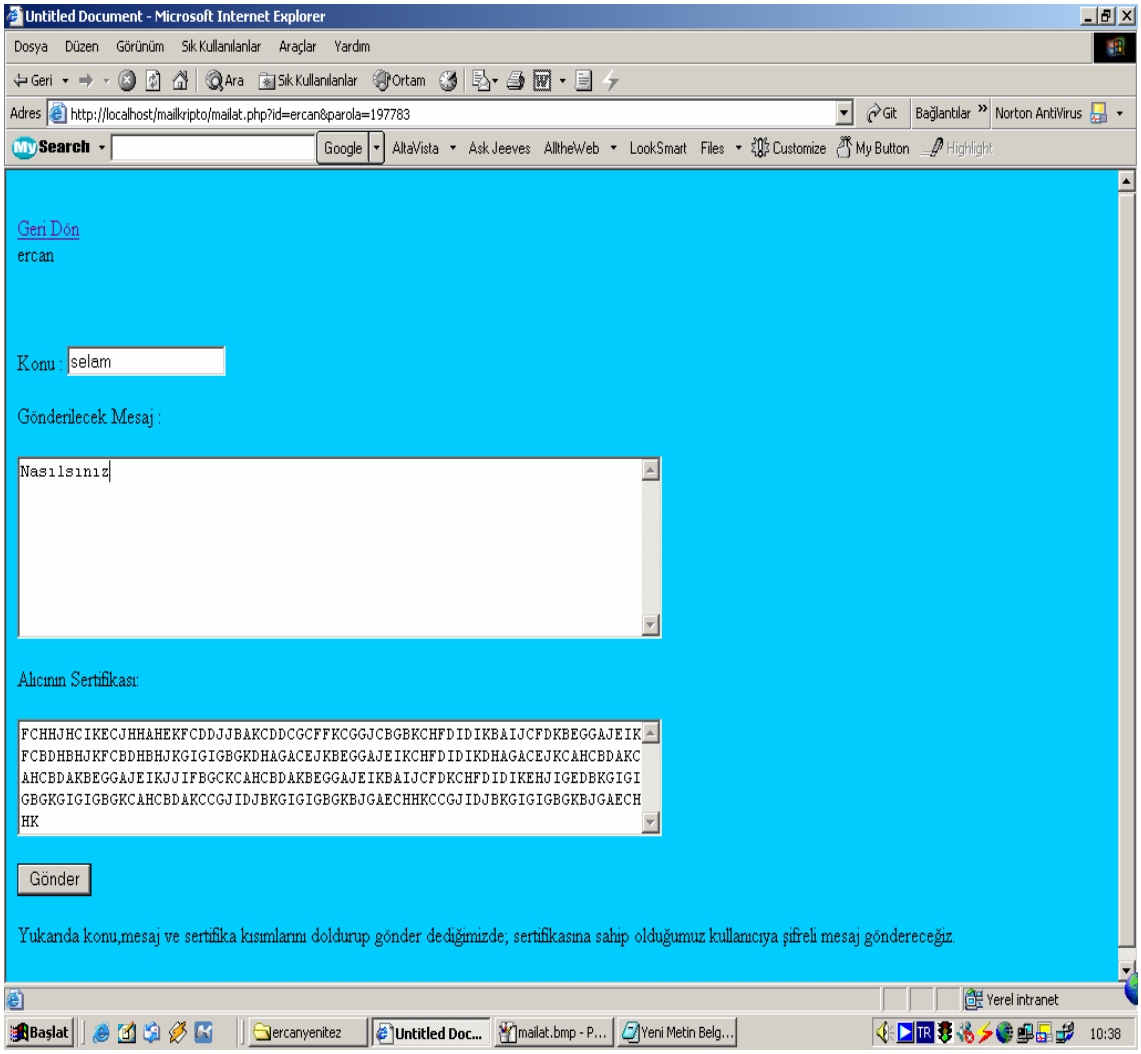
Şekil 7.8: Sertifikanın Gösterilmesi

Şekil 7.3'deki Mail Gönder linkine tıkladığımızda Şekil 7.9 'daki sayfa karşımıza çıkar. Gerekli alanları doldurunca şekil 7.10'daki sayfa karşımıza çıkacaktır.



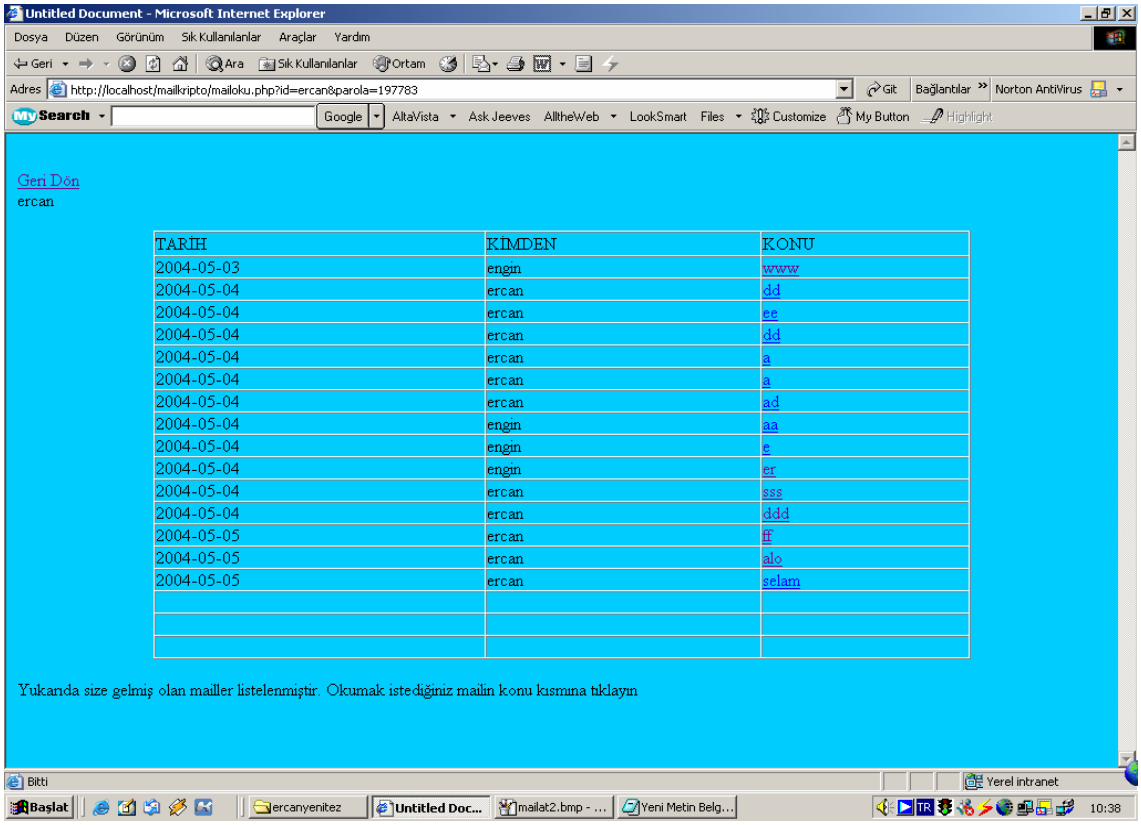
Şekil 7.9: Mail Gönderme Sayfası

Şekil 7.10'daki gönder butonuna bastığımızda mesajı gönderecektir. Öncelikle yöneticinin açık anahtarını kulanarak sertifikayı deşifreleriz. Deşifrelenmiş sertifikadan, kullanıcı kimliği, açık anahtarı ve sertifika oluşum tarihi gelir. Gönderilecek mesaj alanındaki metni açık anahtarı kullanarak şifreleriz. Daha sonra şifrelenmiş mesajı MySQL'de mesaj tablosuna kaydeder. Kimden kısmına gönderenin kimliği, kime kımına alıcının kimliği, bunu sertifikadan elde ettik, mesaj kısmına şifrelenmiş mesajı, konu kısmınada konuyu kaydeder. Şifreleme ve deşifreleme işlemleri JavaScritle kullanıcının bilgisayarında, MySQL veri tabanı işlemlerini PHP 'de serverda yaptım.



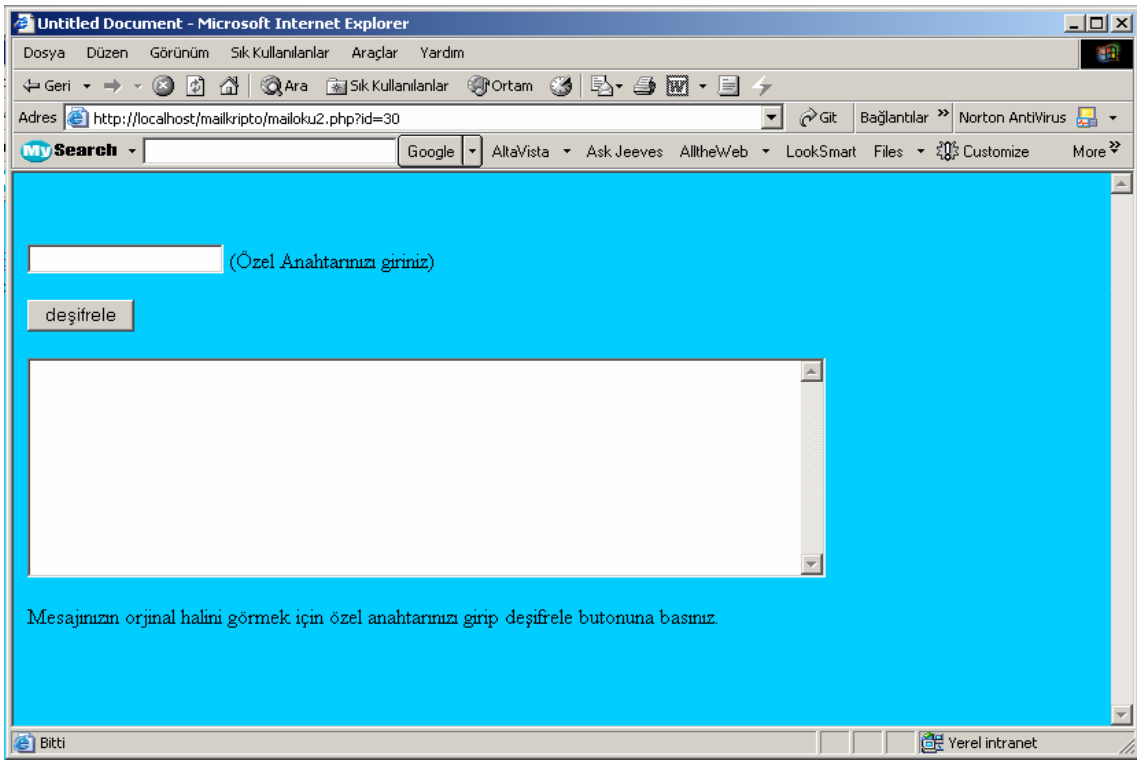
Şekil 7.10: Mailin Gönderilmesi

Sonuçta mesajımızı şifreledik ve veritabanına kaydettik. Artık gelen mailleri okuma zamanı. Gelen mailleri okumak için Şekil 7.3 deki Mail oku linkine tıklarız. Karşımıza şekil 7.11'deki sayfa çıkacaktır. Bu sayfada MySQL veritabanına bağlanıp mesaj tablosunda kime alanında kulanıcının adı gözüken satırları sorgu sonucu seçecek ve tabloda listeleyecek. Bu işlemleri PHP de yaptım.



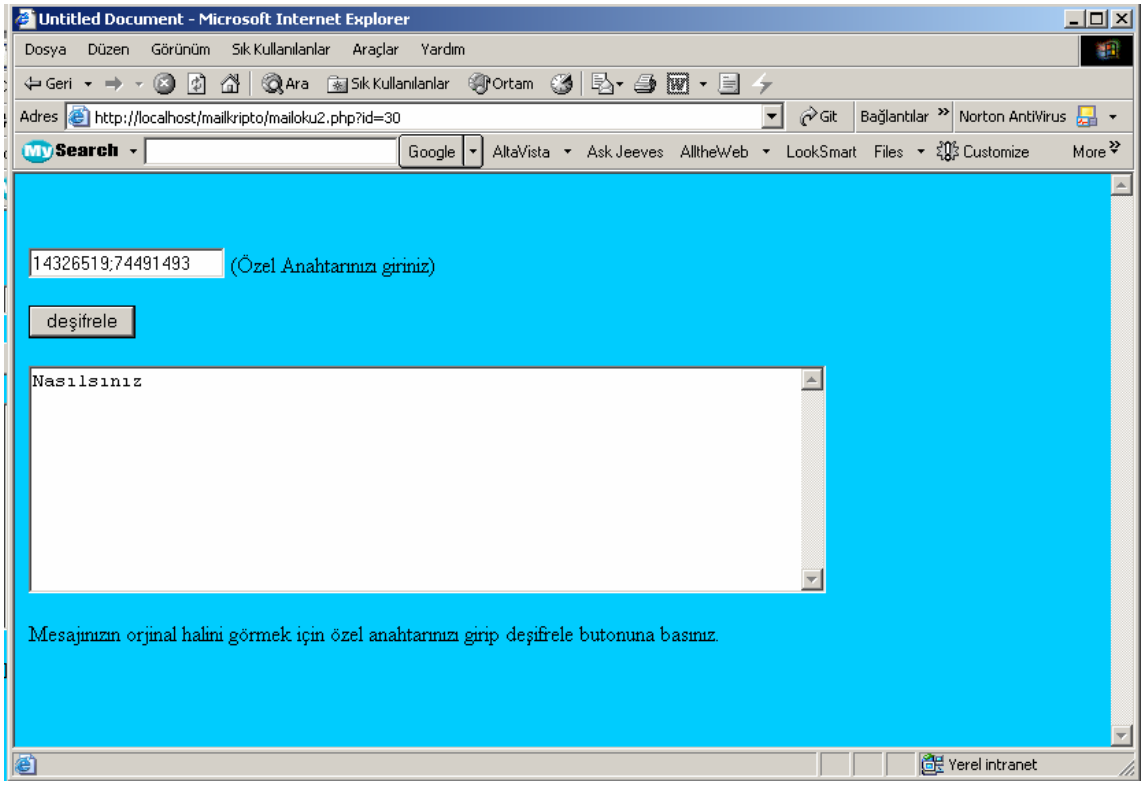
Şekil 7.11: Gelen Mailler

Şekil 7.11 de listelenen maillerden hangisini okumak istiyorsam onu konusuna tıklayacağım. Şekil 7.12 deki sayfa açılacak.



Şekil 7.12 : Mail Okuma sayfası

Bu sayfada özel anahtar kısmına kendimize ait olan özel anatarı gireriz. Deşifreleme butonuna bastığımızda mesajı deşifreleyip yazdıracak. Bunu da Şekil 7.13'de görebiliriz. Öncelikle PHP kullanarak MySQL'den şifreli mesajı alırız. Daha sonra kullanıcının özel anahtarı ile mesajı deşifreleriz. Orijinal mesajı ekrana yazdırırız. Deşifreleme işlemini kullanıcının bilgisayarında JavaScript kullanarak yaptım.



Şekil 7.13 : Mesajın deşifrelenmesi

TARTIŞMA VE SONUÇ

Günümüz iletişim dünyasının ağ tabanlı yapıları kullanması güvenlik ihtiyacını doğurdu. Güvenli bir haberleşme alt yapısı kurmadığımız taktirde her türlü saldırıya açığız demektir. Gizli bilgilerimizi, rakiplerden korumak için saklamalıyız. Aksi taktirde büyük zararlara uğrayabiliriz. Tabi bilginin gizlilik derecesi kuruma ve ya kişilere bağlı olarak değişecektir. Kişisel kullanıcılar bazında düşündüğümüzde saklayabilecekleri kredi kartı bilgileri ve ya özel mailleri olabilir.Fakat milyarlarca dolarlık ciro yapan kuruluşlar içinse bilginin gizli tutulması son derece önemli olacaktır. En basitinden istenmeyen bir bilginin sızması borsa üzerinde puan kaybetmelerine ve zarar etmelerine neden olacaktır. Biz de güvenlik için çözüm önerilerinde açık anahtarlı kriptografiyi inceledik. Peki çok mu güvenli? diye sorarsanız. Bugün güvenli olanın yarın güvenli olacağını bilemeyiz. Tabi açık anahtarlı kriptografiyi kullanan bir çok algoritma hackerler tarafından kırılmış bugün kırılamayanlar ise ileri de mutlaka kırılacaktır. Tabi buna paralel olarak yeni güvenlik teknikleri üretilecektir. Aslında temel mantık, hiç çözülemez değil çözümü o kadar uzun sürmeli ki çözmenin bir yararı olmasın. Yeni ürünlerin ortaya çıkması ile işlemci hızları artmış ve şifre çözüm süreleri azalmıştır. Peki kriptografi ile uğraşanlar boş mu duracaklar? Onlarda yeni donanımlarla daha güvenli yöntemler üretecekler. Sonuç olarak birileri güven için uğraşırken, bazıları da onların güvensizliğini ispatlamaya çalışacaktır.

ÖZET

İnsanlık tarihinde gizliliğin ortaya çıkması ve bununla beraber kriptografinin gelişmesine değinildi. Açık anahtarlı kriptografi algoritması incelendi ve gereklilikleri tespit edildi. Geleneksel kriptografi ile karşılaştırıldığında daha güvenli olduğu görüldü. Açık anahtarlı kriptografide iki anahtardan oluşan anahtar çiftinin bulunduğu, açık anahtarın herkesçe bilindiği ve şifreleme işleminde kullanıldığı, özel anahtarınsa deşifreleme işleminde kullanıldığı ve yalnızca anahtar çiftinin sahibi olan kullanıcıda bulunduğu tespit edildi. Diffi-Hellman kripto sistemi incelendi. Gerekli olan hesaplamalar belirtildi ve örneklendirildi. Ron Rivest, Adi Shamir ve Len Adleman Tarafından tasarlanan RSA algoritması incelendi. Anahtar üretimi için gerekli olan iki asal sayının bulunması ve anahtar üretimi incelendi. Uygulama yazılımı üzerinde, iki asal sayı tespitinin yapılması incelendiğinde, büyük sayılarla çalışmanın güvenliği artırdığı fakat anahtar üretimi için gerekli süreyi de artırdığı görüldü. Anahtar üretimi, her mesajın şifrelenmesi için gerekli olmadığından büyük sayılarla çalışmanın avantajlı olduğu tespit edildi. Şifreleme ve ya deşifreleme işlemlerinin üstsel ifadeler kullanılarak yapıldığı belirlendi. Üs alma işleminin uzun bir süre alması nedeniyle çeşitli mantıklar üzerine duruldu. X gibi bir sayının 16. kuvvetini tek tek çarparak almak yerine, X^2, X^4, X^8, X^{16} kullanılarak yalnızca dört çarpmada sonuca ulaşıldığı görüldü. Bu işlem hesaplamalar için gerekli olan süreyi kısalttı. Uygulama yazılımı buna göre geliştirildi ve şifreleme/deşifreleme işlemlerinin daha kısa sürede yapıldığı tespit edildi. Güvenliği artıran en önemli etkenlerden birisi anahtar yönetimidir. Açık anahtarların dağıtımı üzerine dört tane senaryo incelendi. Açık anahtar sertifikası oluşturmanın avantajlı olduğu görüldü. X509 Kimlik doğrulama servisi ve sertifikalar incelendi. Tek yönlü, İki yönlü ve Üç yönlü Kimlik doğrulama konularına değinildi. Elektronik posta güvenliği konusu incelendi. Phil Zimmermann tarafından geliştirilen PGP uygulamasının, kimlik doğrulama, güvenlik, anahtar halkaları gibi konuları incelendi. IP güvenliği ve WEB güvenliği konularına değinildi. İnternet üzerinden alış veriş senaryosu oluşturuldu.

SUMMARY

At the humanity history occurring privacy developed cryptography. The public key cryptography algorithm was examined and its necessity was fixed. When it was compared with conventional cryptography, it was seen that the public key cryptography is safer than conventional cryptography. In the public key cryptography there is a pair of key that is made up of two keys, the public key is known by everybody and it is used for encryption. Privacy key is used for decryption and only the person who is the own of the pair of key can see the privacy key. Diffie-Hellman cryptosystem was examined. The necessary computation was shown and illustrated. The RSA algorithm that was envisaged by Ron Rivest, Adi Shamir and Len Adleman was examined. Finding two prime numbers, which are necessary for key production, and key production was examined. When finding two prime numbers with application software was examined, it was seen that working with big numbers increased the security but the time that is necessary for key production increased too. Key production isn't necessary for all message encryption. Therefore it was seen that working with the big numbers are advantageous. For encryption and decryption operations exponential expression were used. Because of computing exponential expression were wasting time, the different methods were examined. To compute X^{16}, instead of multiplying 16 X one by one using X^2, X^4, X^8 and X^{16} with four multiplications it was concluded. This operation shortened the necessary time which for computations. Application software was developed according to this and it was found that encryption/decryption operations were made in a shorter time. One of the most important factors is key management. The four methods were examined about the public key distribution. It was seen that making public key certificate is advantageous. At X509 authentication service and certificates were examined. One-way, two-way and three-way authentication items were mentioned. Electronic mail security was examined. PGP application that was developed by Phil Zimmerman and as the items of PGP application's authentication, security, and key rings was examined. The items of IP and WEB security were mentioned. Shopping scenario with Internet was constituted.

EKLER : RSA algoritmasının kullanımı ile internet üzerinden şifreli mesaj alış verişi yapılmasını sağlayan uygulamanın kaynak kodları

INDEX.PHP

```
<html>
<head>
<title>Untitled Document</title>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1254">
<script language="JavaScript" type="text/JavaScript">
function kontrol()
{

		if (document.form1.id.value=="")
		{
				alert("Kullanıcı adını Giriniz");
				form1.id.focus();
				form1.id.select();
				return false;
		}

		if (document.form1.parola.value=="")
		{
				alert("Parola Alanı Boş Bırakılamaz");
				form1.parola.focus();
				form1.parola.select();
				return false;
		}

return true;
}

    </script>
</head>
```

```
<body bgcolor="#00CCFF" text="#333333">
<form name="form1" method="get" onSubmit="return kontrol()" action="uyeler.php">
<p align="center"> </p>
<p align="center"><font face="Times New Roman"><strong><font color="#000000">&Ccedil;ANAKKALE
    ONSEKİZ MART ÜNİVERSİTESİ</font></strong></font></p>
<p align="center"><font color="#000000"><strong><font face="Times New Roman">BİLGİSAYAR MÜHENDİSLİĞİ</font></strong></font></p>
<p align="center"><font color="#000000"><strong><font face="Times New Roman">YÜKSEK LİSANS PROJESİ</font></strong></font></p>
<p align="center"><font color="#000000"><strong><font face="Times New Roman">AÇIK ANAHTAR KRİPTOGRAFİ KULLANIMI
  İLE İNTERNET ÜZERİNDE MESAJ ALIŞVERİŞİ</font></strong></font></p>
<table width="43%" border="1" align="center">
 <tr align="left" bordercolor="#00CCFF">
  <td width="53%"><font color="#000000"><strong>K<font face="Times New Roman">ULLANICI ADI:</font></strong></font></td>
  <td width="47%"><font color="#000000">
    <input name="id" type="text" id="id">
  </font></td>
 </tr>
 <tr align="left" bordercolor="#00CCFF">
  <td><font color="#000000"><strong>PAROLA:</strong></font></td>
  <td><font color="#000000">
    <input name="parola" type="password" id="parola">
  </font></td>
 </tr>
</table>
<p align="center">
 <input name="giris" type="submit" id="giris" value="GİRİŞ">
</p>
<p align="center"><font color="#000000" face="Times New Roman"><strong><a href="uyeol.php">Üye
```

```
      Ol</a></strong></font></p>
</form>
</body>
</html>
```

UYEOL.PHP

```
<?php require_once('Connections/link.php'); ?>
<?php
function GetSQLValueString($theValue, $theType, $theDefinedValue = "",
$theNotDefinedValue = "")
{
 $theValue = (!get_magic_quotes_gpc()) ? addslashes($theValue) : $theValue;

 switch ($theType) {
  case "text":
   $theValue = ($theValue != "") ? "'" . $theValue . "'" : "NULL";
   break;
  case "long":
  case "int":
   $theValue = ($theValue != "") ? intval($theValue) : "NULL";
   break;
  case "double":
   $theValue = ($theValue != "") ? "'" . doubleval($theValue) . "'" : "NULL";
   break;
  case "date":
   $theValue = ($theValue != "") ? "'" . $theValue . "'" : "NULL";
   break;
  case "defined":
   $theValue = ($theValue != "") ? $theDefinedValue : $theNotDefinedValue;
   break;
 }
 return $theValue;
```

```
}

$editFormAction = $HTTP_SERVER_VARS['PHP_SELF'];
if (isset($HTTP_SERVER_VARS['QUERY_STRING'])) {
  $editFormAction .= "?" . $HTTP_SERVER_VARS['QUERY_STRING'];
}

if ((isset($HTTP_POST_VARS["MM_insert"])) &&
($HTTP_POST_VARS["MM_insert"] == "form1")) {
  $insertSQL = sprintf("INSERT INTO user (ID, Adsoyad, Meslek, Tel, parola)
VALUES (%s, %s, %s, %s, %s)",
                GetSQLValueString($HTTP_POST_VARS['kuladi'], "text"),
                GetSQLValueString($HTTP_POST_VARS['adsoyad'], "text"),
                GetSQLValueString($HTTP_POST_VARS['meslek'], "text"),
                GetSQLValueString($HTTP_POST_VARS['tel'], "text"),
                GetSQLValueString($HTTP_POST_VARS['parola'], "text"));

  mysql_select_db($database_link, $link);
  $Result1 = mysql_query($insertSQL, $link) or die("Bu Kullanıcı Adı Kullanılmakta
Başka Bir Kullanıcı Adı Giriniz");//mysql_error());

  $insertGoTo = "ok.php";
  if (isset($HTTP_SERVER_VARS['QUERY_STRING'])) {
    $insertGoTo .= (strpos($insertGoTo, '?')) ? "&" : "?";
    $insertGoTo .= $HTTP_SERVER_VARS['QUERY_STRING'];
  }
  header(sprintf("Location: %s", $insertGoTo));
}
?>
<html>
<head>
<title>Untitled Document</title>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1254">
```

```
<script language="JavaScript" type="text/JavaScript">
function kontrol()
{

		if (document.form1.kuladi.value=="")
		{
			alert("Kullanıcı adını Giriniz");
			form1.kuladi.focus();
			form1.kuladi.select();
			return false;
		}

		if (document.form1.parola.value=="")
		{
			alert("Parola Alanı Boş Bırakılamaz");
			form1.parola.focus();
			form1.parola.select();
			return false;
		}
		if (document.form1.adsoyad.value=="")
		{
			alert("Adınızı ve Soaydınızı Giriniz");
			form1.adsoyad.focus();
			form1.adsoyad.select();
			return false;
		}
		if (document.form1.meslek.value=="")
		{
			alert("Mesleğinizi Giriniz");
			form1.meslek.focus();
			form1.meslek.select();
			return false;
		}
```

```
                if (document.form1.tel.value=="")
                {
                        alert("Telefon Numaranızı Giriniz");
                        form1.tel.focus();
                        form1.tel.select();
                        return false;
                }


return true;
}

    </script>
</head>

<body bgcolor="#00CCFF">
<form name="form1" method="POST" onSubmit="return kontrol()" action="<?php
echo $editFormAction; ?>" >
<p> </p>
<table width="48%" border="1" align="center">
  <tr align="left" bordercolor="#00CCFF">
    <td width="58%"><font face="Times New Roman"><strong><font
color="#000000">KULLANICI
      ADI : </font></strong></font></td>
    <td width="42%"><font face="Times New Roman"><strong><font
color="#000000">
    </font><font face="Times New Roman"><strong><font color="#000000">
    <input name="kuladi" type="text" id="kuladi">
    </font></strong></font></strong></font></td>
  </tr>
  <tr align="left" bordercolor="#00CCFF">
    <td><font color="#000000"><strong>PAROLA : </strong></font></td>
    <td><font color="#000000"><strong><font face="Times New Roman">
```

```
    <input name="parola" type="password" id="parola">
   </font></strong></font></td>
  </tr>
  <tr align="left" bordercolor="#00CCFF">
   <td><font color="#000000"><strong><font face="Times New Roman">ADI
SOYADI:</font></strong></font></td>
   <td><font color="#000000"><strong><font face="Times New Roman">
    <input name="adsoyad" type="text" id="adsoyad">
   </font></strong></font></td>
  </tr>
  <tr align="left" bordercolor="#00CCFF">
   <td><font color="#000000"><strong><font face="Times New
Roman">MESLEK:</font></strong></font></td>
   <td><font color="#000000"><strong><font face="Times New Roman">
    <input name="meslek" type="text" id="meslek">
   </font></strong></font></td>
  </tr>
  <tr align="left" bordercolor="#00CCFF">
   <td><font color="#000000"><strong><font face="Times New
Roman">TEL:</font></strong></font></td>
   <td><input name="tel" type="text" id="tel">
    </td>
  </tr>
</table>
<div align="center"></div>
<p align="center">
  <input name="Kaydet" type="submit" value="KAYDET">
</p>
<p align="left"><font face="Times New Roman"><strong><font color="#000000">
</font></strong></font></p>
<p align="left"><font color="#000000"></font></p>
<p align="left"><font color="#000000"><strong><font face="Times New Roman">
</font></strong></font></p>
```

```
<p align="left"><font color="#000000"><strong><font face="Times New Roman">
</font></strong></font></p>
<p align="left"><font color="#000000"><strong><font face="Times New Roman">
</font></strong></font></p>
<p align="left">
  <input type="hidden" name="MM_insert" value="form1">
</p>
</form>
<p> </p>
<p> </p>
<div align="center"></div>
</body>
</html>
```

UYELER.PHP

```
<body bgcolor="#00CCFF"><?php require_once('Connections/link.php'); ?>

<?php
$id=$HTTP_GET_VARS['id'];
$parola=$HTTP_GET_VARS['parola'];
mysql_select_db($database_link, $link);
$query_Recordset1 = "SELECT * FROM `user` WHERE ID = '$id' AND parola =
'$parola' ";
$Recordset1 = mysql_query($query_Recordset1, $link) or die(mysql_error());
$row_Recordset1 = mysql_fetch_assoc($Recordset1);
$totalRows_Recordset1 = mysql_num_rows($Recordset1);
if ($row_Recordset1['ID']!="")
{
echo '<br><br><a href=anahtar.php?id=';echo $id;echo'&parola=';echo
$parola;echo'>Anahtar Olustur</a>';
echo '<br><br><a href=sertifika.php?id=';echo $id;echo '&parola=';echo
$parola;echo'>Sertifika islemleri</a>';
```

```
echo '<br><br><a href=mailat.php?id=';echo $id;echo '&parola=';echo $parola;echo'>Mail Gönder</a>';
echo '<br><br><a href=mailoku.php?id=';echo $id;echo '&parola=';echo $parola;echo'>Mail Oku</a><br><br>';
echo 'Bu sayfadan yukarýda linkleri verilen sayfalara ulaþabilirsiniz. Gerekli açýklamalarý o sayfa üzerinde görebilirsiniz. ';
}
else
{
echo "Girmiþ Olduðunuz Kullanýcý Adý Geçerli Deðil";
echo '<br><br><a href=index.php>Geri DÖN</a>';
}
?>
<?php
mysql_free_result($Recordset1);
?>
<form name="form1" method="post" onSubmit="" action="">
</form>
```

ANAHTAR.PHP

```
<?php
$id=$HTTP_GET_VARS['id'];
$parola=$HTTP_GET_VARS['parola'];
echo '<br><a href=uyeler.php?id=';echo $id;echo'&parola=';echo $parola;echo'>Geri Dön</a><br>';

?>
<html>
<head>
<title>Untitled Document</title>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1254">
</head>
<script language="javascript" type="text/JavaScript">
```

```
var yeni=null;
function anahtar()
{  var s1=Math.floor(Math.random()*10);
  var s2=Math.floor(Math.random()*1000);
  var s3=Math.floor(Math.random()*1000);
  if (s1<4) s1=4;
  var p=s1*1000+s2;
  var q=s1*1000+s3;
  var k=0;
  while(k<1)
  { for(var sayac1=2;sayac1<100;sayac1++)
    { if (p%sayac1==0)
            {k=0;
             p=p-1;
             break;
                    }
   k=1;
          }
  }
  var k2=0;
  while(k2<1)
  { for(var sayac3=2;sayac3<100;sayac3++)
    {if (q%sayac3==0)
           {k2=0;
           q=q-1;
           break;
           }
          k2=1;
          }
          if (q==p) k2=0;
  }
  var n=p*q;
  var Fin=(p-1)*(q-1);
```

```
  var e1=Math.floor(Math.random()*10);
  if (e1==0 || e1==1) e1=2;
  var e=Math.floor(Math.random()*1000000);
  e=e+e1*1000000;
  k=0;
  do
   { for(var sayac3=2;sayac3<10000;sayac3++)
          {
      if ((e%sayac3==0)&&(Fin%sayac3==0))
      { k=0;
        e=e-1;
        break;
      }
                k=1;
          }
   } while (k==0);
    k=0;
         var f=1;
         do
   {Fin2=f*Fin+1;
          var d=Math.floor(Fin2/e);
          if (e*d==Fin2)
          {k=1;}
          f=f+1;
   } while (k==0);
         var metin=';';
         document.form1.textarea.value=e+metin+n;
         document.form1.textarea2.value=d+metin+n;
         return false;
}

 </script>
<body bgcolor="#00CCFF">
```

```
<form name="form1" method="POST" onSubmit="return anahtar()" action="">
<p>
  <input type="hidden" name="hiddenField">
  <input type="hidden" name="hiddenField2">
  <input type="hidden" name="hiddenField3">
</p>
<p>PublicKey:</p>
<p>
  <textarea name="textarea"></textarea>
</p>
<p>PrivateKey:</p>
<p>
  <textarea name="textarea2"></textarea>
</p>
<p>
  <input type="submit" name="Submit" value="Oluştur">
</p>
<p>Yukarıdaki oluştur butonuna bastığınızda açık anahtar ve özel anahtarınız
  oluşturulacaktık. Anahtar oluşturma esnasında İnternet explorer yavaş çalışıyor
  diye hata verebilir. Komut dizisini durdurmak istiyormusunuz sorusuna hayır
  yanıtını veriniz. Bu hatayı vermesinin nedeni anahtar oluşturma işleminin
  uzun sürmesidir. Bu işlemi bir kere yapmanız yeterlidir. Daha sonra sertifikanızı
  oluşturunuz. Anahtar bilgilerinizi bir txt dosyaya kaydetmeyi unutmayın.</p>
<p> </p>
</form>
</body>
</html>
```

SERTIFIKA.PHP

```
<?php require_once('Connections/link.php'); ?> <?php
mysql_select_db($database_link, $link);
$query_Recordset1 = "SELECT privatekey FROM yonetici";
$Recordset1 = mysql_query($query_Recordset1, $link) or die(mysql_error());
```

```
$row_Recordset1 = mysql_fetch_assoc($Recordset1);
$totalRows_Recordset1 = mysql_num_rows($Recordset1);

$id=$HTTP_GET_VARS['id'];
$parola=$HTTP_GET_VARS['parola'];
echo '<br><a href=uyeler.php?id=';echo $id;echo'&parola=';echo $parola;echo'>Geri Dön</a><br>';
echo $id;
function GetSQLValueString($theValue, $theType, $theDefinedValue = "", $theNotDefinedValue = "")
{
 $theValue = (!get_magic_quotes_gpc()) ? addslashes($theValue) : $theValue;

 switch ($theType) {
  case "text":
   $theValue = ($theValue != "") ? "'" . $theValue . "'" : "NULL";
   break;
  case "long":
  case "int":
   $theValue = ($theValue != "") ? intval($theValue) : "NULL";
   break;
  case "double":
   $theValue = ($theValue != "") ? "'" . doubleval($theValue) . "'" : "NULL";
   break;
  case "date":
   $theValue = ($theValue != "") ? "'" . $theValue . "'" : "NULL";
   break;
  case "defined":
   $theValue = ($theValue != "") ? $theDefinedValue : $theNotDefinedValue;
   break;
 }
 return $theValue;
}
```

```
$editFormAction = $HTTP_SERVER_VARS['PHP_SELF'];
if (isset($HTTP_SERVER_VARS['QUERY_STRING'])) {
 $editFormAction .= "?" . $HTTP_SERVER_VARS['QUERY_STRING'];
}
 $metin=$HTTP_POST_VARS['publickey'];
 $metin3=date("Y/m/d");
 $metin4=";";
 $metin2=$id.$metin4.$metin.$metin4.$metin3;
 $privatekey= $row_Recordset1['privatekey'];
 $privatesayi=strlen($privatekey);
 $bayrak=0;
 for ($i=0; $i<$privatesayi; $i++)
 {
  if ($privatekey[$i]==';')
   {
    $bayrak=1;
    continue;
   }
   if ($bayrak==1)  $satirn=$satirn.$privatekey[$i];
   if ($bayrak==0)  $satird=$satird.$privatekey[$i];
 }
 $d=$satird;
 $n=$satirn;
 $d2=$d;
 do
 {
  $tam=floor($d2/2);
  $kalan=$d2%2;
  $ikili=$kalan.$ikili;
  $d2=$tam;
 }
 while ($tam!=0);
```

```
 $us=strlen($ikili);
 $sorjinal=strlen($metin2);
 for($j=0;$j<$sorjinal;$j++)
 {       $M=Ord($metin2[$j]);
         $C=1;
  $aradeger=0;
         for($sayac=0;$sayac<$us;$sayac++)
         {
           $aradeger=$aradeger*2;
           $C=($C*$C);
           $C=fmod($C,$n);
           if($ikili[$sayac]=='1')
    { $aradeger=$aradeger+1;
      $C=($C*$M);
      $C=fmod($C,$n);
           }
  }
         $sifreli2=strval($C);
         $sifuz=strlen($sifreli2);
         for($l=0;$l<$sifuz;$l++)
  { $sifreli3=$sifreli2[$l];
           $sifreli4=65+intval($sifreli3);
           $sifreli=$sifreli.Chr($sifreli4);
         }
  $sifreli=$sifreli.'K';
 }

 echo '<br><a href=sertifika2.php?id=';echo $id;echo'&parola=';echo
$parola;echo'&sifreli=';echo $sifreli;echo'>Göster</a><br>';

if ((isset($HTTP_POST_VARS["MM_insert"])) &&
($HTTP_POST_VARS["MM_insert"] == "form1")) {
```

```
 $insertSQL = sprintf("INSERT INTO sertifika (ID,`Time`,sertifika) VALUES (%s,
%s, %s)",
                GetSQLValueString($id, "text"),
                GetSQLValueString(date("Y/m/d"), "date"),
                                        GetSQLValueString($metin2, "text"));
 mysql_select_db($database_link, $link);
 $Result1 = mysql_query($insertSQL, $link) or die(mysql_error());
  $insertGoTo = "ok2.php";
}
?>
<html>
<head>
<title>Untitled Document</title>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1254">
<script language="JavaScript" type="text/JavaScript">

function kontrol()
{               if (document.form1.publickey.value=="")
                {
                        alert("Açık anahtarınızı Giriniz");
                        form1.publickey.focus();
                        form1.publickey.select();
                        return false;
                }



return true;

}


    </script>
```

```
</head>

<body bgcolor="#00CCFF">
<form name="form1" method="POST" onSubmit="return kontrol()" action="<?php
echo $editFormAction; ?>">

<p>Public Key :
  <input name="publickey" type="text" id="publickey">
</p>
<p>
  <input name="Kaydet" type="submit" id="Kaydet" value="Oluştur">
</p>
<p>Yukarıda Public Key isimli alana açık anahtarınızı giriniz. Daha sonra oluştur
  butonunu tıklayalım. Daha sonra Göster'e tıklayarak sertifikamızın görüntüleneceği
  sayfaya gideceğiz. Sertifika ne işe yarıyor? Bize mail gönderecek kişilere
  sertifikamızı göndeririz. Onlarda bizim sertifikamızı kullanarak bize şifreli
  mesaj gönderecekler.</p>
<p>
    <input type="hidden" name="MM_insert" value="form1">
</p>
</form>
<p> </p>
</body>
</html>
<?php
mysql_free_result($Recordset1);
?>
```

SERTIFIKA2.PHP

```
<?php
$id=$HTTP_GET_VARS['id'];
$parola=$HTTP_GET_VARS['parola'];
$sifreli=$HTTP_GET_VARS['sifreli'];
```

```
echo '<br><a href=sertifika.php?id=';echo $id;echo'&parola=';echo $parola;echo'>Geri Dön</a><br>';
echo $id;
?>
<html>
<head>
<title>Untitled Document</title>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1254">
</head>
<script language="javascript" type="text/JavaScript">
var yeni=null;
function goster()
{

        document.form1.textarea.value="<?echo $sifreli;?>";

        return false;


}

 </script>

<body bgcolor="#00CCFF">
<form name="form1" onSubmit="return goster()" method="post" action="">
<p><font face="Verdana, Arial, Helvetica, sans-serif, Albertus">Sertifikanız</font></p>
<p>
  <textarea name="textarea" cols="70" rows="5"></textarea>
</p>
<p>
  <input type="submit" name="Submit" value="Göster">
</p>
```

```
<p>Göster butonuna bastığımızda sertifikamız görüntülenecektir. Sertifikanızı
 txt dosyaya kaydediniz.</p>
</form>
</body>
</html>
```

MAILAT.PHP

```
<?php require_once('Connections/link.php'); ?>
<?php
$id=$HTTP_GET_VARS['id'];
$parola=$HTTP_GET_VARS['parola'];
echo '<br><a href=uyeler.php?id=';echo $id;echo'&parola=';echo $parola;echo'>Geri
Dön</a><br>';
function GetSQLValueString($theValue, $theType, $theDefinedValue = "",
$theNotDefinedValue = "")
{
 $theValue = (!get_magic_quotes_gpc()) ? addslashes($theValue) : $theValue;

 switch ($theType) {
  case "text":
   $theValue = ($theValue != "") ? "'" . $theValue . "'" : "NULL";
   break;
  case "long":
  case "int":
   $theValue = ($theValue != "") ? intval($theValue) : "NULL";
   break;
  case "double":
   $theValue = ($theValue != "") ? "'" . doubleval($theValue) . "'" : "NULL";
   break;
  case "date":
   $theValue = ($theValue != "") ? "'" . $theValue . "'" : "NULL";
   break;
  case "defined":
```

```
    $theValue = ($theValue != "") ? $theDefinedValue : $theNotDefinedValue;
    break;
  }
  return $theValue;
}

$editFormAction = $HTTP_SERVER_VARS['PHP_SELF'];
if (isset($HTTP_SERVER_VARS['QUERY_STRING'])) {
  $editFormAction .= "?" . $HTTP_SERVER_VARS['QUERY_STRING'];
}

if ((isset($HTTP_POST_VARS["MM_insert"])) &&
($HTTP_POST_VARS["MM_insert"] == "form1")) {
  $insertSQL = sprintf("INSERT INTO mesaj (kimden, kime, mesaj,`time`,konu)
VALUES (%s, %s, %s, %s, %s)",
                GetSQLValueString($HTTP_POST_VARS['hiddenField2'], "text"),
                GetSQLValueString($HTTP_POST_VARS['hiddenField3'], "text"),
                GetSQLValueString($HTTP_POST_VARS['hiddenField'], "text"),
                GetSQLValueString(date("Y/m/d"), "date"),
                GetSQLValueString($HTTP_POST_VARS['textfield'], "text"));
  mysql_select_db($database_link, $link);
  $Result1 = mysql_query($insertSQL, $link) or die(mysql_error());
}



mysql_select_db($database_link, $link);
$query_Recordset1 = "SELECT publickey FROM yonetici";
$Recordset1 = mysql_query($query_Recordset1, $link) or die(mysql_error());
$row_Recordset1 = mysql_fetch_assoc($Recordset1);
$totalRows_Recordset1 = mysql_num_rows($Recordset1);
$yon=$row_Recordset1['publickey'];
$idgon=$HTTP_GET_VARS['id'];
```

```
echo $idgon;

?>
<html>

<head>
<title>Untitled Document</title>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1254">
</head>
<script language="javascript" type="text/JavaScript">

function kontrol()
{       if (document.form1.textarea.value=="")
                {
                        alert("Mesaj alanı Boş bırakılamaz");
                        form1.textarea.focus();
                        form1.textarea.select();
                        return false;
                }

        if (document.form1.textarea2.value=="")
                {
                        alert(" Alıcının Sertifikasını Giriniz");
                        form1.textarea2.focus();
                        form1.textarea2.select();
                        return false;
                }

var yon2 ="<?echo $yon;?>";
var mesaj=document.form1.textarea.value;
var alici=document.form1.textarea2.value;
var idgon2="<? echo $idgon;?>";
var s=yon2.length,bayrak,i;
```

```
var yone,yonn;
var yonsatire='',yonsatirn='';
 bayrak=0;
 for(i=0;i<s;i++)
 { if (yon2.charAt(i)==';')
   {
    bayrak=1;
    continue;
   }
   if (bayrak==1) {yonsatirn=yonsatirn+yon2.charAt(i);}
   if (bayrak==0) yonsatire=yonsatire+yon2.charAt(i);
 }
 yone=yonsatire;
 yonn=yonsatirn;
 yone2=yone;
 var ikili=' ';
 var tam,kalan;
 do
 {tam=Math.floor(yone2/2);
  kalan=yone2% 2;
  ikili=kalan+ikili;
  yone2=tam;
 }
 while (tam !=0);
 var us=ikili.length-1;
 var aliciuz=alici.length;
 bayrak=0;
 var alici2='';
 var sifdeger,sifreli;
 var orjinal='';
 for(var j=0;j<aliciuz;j++)
 {
        if (alici.charAt(j)=='K')
```

```
 {
  bayrak=1;
 }
 if (bayrak=='0')
 {
  sifdeger=((alici.charAt(j)).charCodeAt())-65;
        alici2=alici2+sifdeger;
       }
 if (bayrak==1)
 { var C=parseFloat(alici2);
        var yonn2=parseFloat('yonn');
  var M=parseFloat('1');
        var aradeger=0;
  for(var sayac=0;sayac<us;sayac++)
  {
   aradeger=aradeger*2;
   M=(M*M)%yonn;
              if (ikili.charAt(sayac)=="1")
   {
    aradeger=aradeger+1;
    M=(M*C)%yonn;
              }
  }
        orjinal=orjinal+String.fromCharCode(M);
        bayrak=0;
  alici2='';
 }
}
var orjuz=orjinal.length;
var aliciID='',alicie='',alicin='',alicitime='';
var bayrak2=0;
for(i2=0;i2<orjuz;i2++)
{ if (orjinal.charAt(i2)==';')
```

```
  {
   bayrak2=bayrak2+1;
   continue;
  }
  if (bayrak2==0) aliciID=aliciID+orjinal.charAt(i2);
        if (bayrak2==1) alicie=alicie+orjinal.charAt(i2);
        if (bayrak2==2) alicin=alicin+orjinal.charAt(i2);
        if (bayrak2==3) alicitime=alicitime+orjinal.charAt(i2);
}
var alicie2=alicie;
var ikili2='';
var tam2,kalan2;
do
{tam2=Math.floor(alicie2/2);
 kalan2=alicie2% 2;
 ikili2=kalan2+ikili2;
 alicie2=tam2;
} while(tam2!=0);
var us2=ikili2.length;
var mesajuz=mesaj.length;
bayrak=0;
var mesaj2='';
var sifdeger2,sifreli2='';
var sifreli='';
for(var j=0;j<mesajuz;j++)
{ var M2=parseFloat((mesaj.charAt(j)).charCodeAt());
        var C2=parseFloat('1');
        var alicin2=parseFloat('alicin');
  var aradeger=0;
  for(var sayac2=0;sayac2<us2;sayac2++)
  {
    aradeger=aradeger*2;
    C2=(C2*C2)%alicin;
```

```
            if (ikili2.charAt(sayac2)=="1")
     {
      aradeger=aradeger+1;
      C2=(M2*C2)%alicin;
     }
    }
          var sifreli2="+C2;
   var sifuz=sifreli2.length;
          for(i=0;i<sifuz;i++)
   {
    sifreli=sifreli+String.fromCharCode(65+parseInt(sifreli2.charAt(i)));
          }
   sifreli=sifreli+'K';
 }
 document.form1.hiddenField.value=sifreli;
 document.form1.hiddenField2.value=idgon2;
 document.form1.hiddenField3.value=aliciID;


return true;
}

     </script>

<body bgcolor="#00CCFF">
<p> </p>
<form name="form1" method="POST" onSubmit="return kontrol()" action="<?php
echo $editFormAction; ?>">
<p>Konu :
  <input type="text" name="textfield">
</p>
<p><font face="Times New Roman">Gönderilecek Mesaj :</font></p>
<p><font face="Times New Roman">
```

```
  <textarea name="textarea" cols="70" rows="8"></textarea>
  <input type="hidden" name="hiddenField">
  <input type="hidden" name="hiddenField2">
  <input type="hidden" name="hiddenField3">
</font></p>
<p><font face="Times New Roman">Alıcının Sertifikası:</font></p>
<p><font face="Times New Roman">
  <textarea name="textarea2" cols="70" rows="5"></textarea>
</font></p>
<p>
  <input name="Submit" type="submit" id="Submit" value="Gönder">
</p>
<p>Yukarıda konu,mesaj ve sertifika kısımlarını doldurup gönder dediğimizde;
  sertifikasına sahip olduğumuz kullanıcıya şifreli mesaj göndereceğiz. </p>
<input type="hidden" name="MM_insert" value="form1">
</form>
</body>
</html>
<?php
mysql_free_result($Recordset1);
?>
```

MAILOKU.PHP

```
<?php require_once('Connections/link.php'); ?>
<?php
$id=$HTTP_GET_VARS['id'];
$parola=$HTTP_GET_VARS['parola'];
echo '<br><a href=uyeler.php?id=';echo $id;echo'&parola=';echo $parola;echo'>Geri
Dön</a><br>';
$idkime=$HTTP_GET_VARS['id'];
echo $idkime;
mysql_select_db($database_link, $link);
$query_Recordset1 = "SELECT * FROM mesaj WHERE kime = '$idkime'";
```

```
$Recordset1 = mysql_query($query_Recordset1, $link) or die(mysql_error());
$row_Recordset1 = mysql_fetch_assoc($Recordset1);
$totalRows_Recordset1 = mysql_num_rows($Recordset1);

?>
<html>
<head>
<title>Untitled Document</title>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1254">
</head>

<body bgcolor="#00CCFF">
<form name="form1" method="post" action="">
<table width="75%" border="1" align="center" cellpadding="0" cellspacing="0">
  <tr>
    <td height="23">TARİH</td>
    <td>KİMDEN</td>
    <td>KONU</td>
  </tr>
  <?php do { ?>
  <tr>
      <td><?php echo $row_Recordset1['time']; ?></td>
      <td><?php echo $row_Recordset1['kimden']; ?></td>
      <td><a  href="mailoku2.php?id=<?  echo  $row_Recordset1['mesajID'];?>"><?php
echo $row_Recordset1['konu']; ?></a></td>
  </tr>
  <?php } while ($row_Recordset1 = mysql_fetch_assoc($Recordset1)); ?>
  <tr>
    <td> </td>
    <td> </td>
    <td> </td>
  </tr>
  <tr>
```

```
    <td> </td>
    <td> </td>
    <td> </td>
  </tr>
  <tr>
    <td> </td>
    <td> </td>
    <td> </td>
  </tr>
</table>
<p>Yukarıda size gelmiş olan mailler listelenmiştir. Okumak istediğiniz mailin
  konu kısmına tıklayın</p>
</form>
</body>
</html>
<?php
mysql_free_result($Recordset1);
?>
```

MAILOKU2.PHP

```
<?php require_once('Connections/link.php'); ?>
<?php
 $idmesaj=$HTTP_GET_VARS['id']; //echo $idmesaj;
mysql_select_db($database_link, $link);
$query_Recordset1 = sprintf("SELECT * FROM mesaj WHERE mesajID =
'$idmesaj'");
$Recordset1 = mysql_query($query_Recordset1, $link) or die(mysql_error());
$row_Recordset1 = mysql_fetch_assoc($Recordset1);
$totalRows_Recordset1 = mysql_num_rows($Recordset1);
$gelen= $row_Recordset1['mesaj'];?>
<html>
<head>
<title>Untitled Document</title>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1254">
</head>
<script language="javascript" type="text/JavaScript">
function anahtar()
{
if (document.form1.textfield.value=="")
```

```
                {
                        alert("Özel Anahtar alanı Boş bırakılamaz");
                        form1.textarea.focus();
                        form1.textarea.select();
                        return false;
                }
var sifrelimesaj="<? echo $gelen;?>";
var privatekey=document.form1.textfield.value;
var alicisatirn='';
var alicisatire='';
var s=privatekey.length;
var bayrak=0;
 for(var i=0;i<s;i++)
 { if (privatekey.charAt(i)==';')
   {
    bayrak=1;
    continue;
   }
  if (bayrak==1) {alicisatirn=alicisatirn+privatekey.charAt(i);}
  if (bayrak==0) alicisatire=alicisatire+privatekey.charAt(i);
 }
 var alicie=alicisatire;
 var alicin=alicisatirn;
 var alicie2=alicie;
 var ikili=' ';
 var tam,kalan;
 do
 {tam=Math.floor(alicie2/2);
  kalan=alicie2% 2;
  ikili=kalan+ikili;
  alicie2=tam;
 }
 while (tam !=0);
 var us=ikili.length-1;
 var sifmesuz=sifrelimesaj.length;
 bayrak=0;
 var alici2='';
 var sifdeger,sifreli;
 var orjinal='';
 for(var j=0;j<sifmesuz;j++)
 {
        if (sifrelimesaj.charAt(j)=='K')
  {
   bayrak=1;
  }
  if (bayrak=='0')
  {
   sifdeger=((sifrelimesaj.charAt(j)).charCodeAt())-65;
         alici2=alici2+sifdeger;
```

```
        }
   if (bayrak==1)
   { var C=parseFloat(alici2);
          var alicin2=parseFloat('alicin');
    var M=parseFloat('1');
          var aradeger=0;
    for(var sayac=0;sayac<us;sayac++)
    {
     aradeger=aradeger*2;
     M=(M*M)%alicin;
                if (ikili.charAt(sayac)=="1")
     {
      aradeger=aradeger+1;
      M=(M*C)%alicin;
                }
    }
          orjinal=orjinal+String.fromCharCode(M);
          bayrak=0;
    alici2='';
   }
 }

 document.form1.textarea.value=orjinal;
 return false;
}

 </script>

<body bgcolor="#00CCFF">
<form name="form1" method="post" onSubmit="return anahtar()" action="">
<p> </p>
<p>
  <input type="text" name="textfield">
(Özel Anahtarınızı giriniz)</p>
<p>
  <input name="Submit" type="submit" id="Submit2" value="deşifrele">
</p>
<p>
  <textarea name="textarea" cols="70" rows="10"></textarea>
</p>
<p>Mesajınızın orjinal halini görmek için özel anahtarınızı girip deşifrele butonuna
  basınız.</p>
</form>
</body>
</html>
<?php
mysql_free_result($Recordset1);
?>
```

KAYNAKLAR

1- Stallings,William., 1999.Cryptography and Network Security:161-203,323-473

2- Rhee, M. Y., 1994. Cryptography and Secure Communications: 169-204

3- Diffie W., Helmann M.E., Kasım 1976. "New Directions in Cryptography", IEEE Transactions on Information Theory, Vol. 22, pp. 644-654

4- Rivest R., Shamir A. Ve Adleman L., Şubat 1978." A Method for Obtaining Digital Signatures and Public Key Cryptosystems", Commucations of the ACM, vol.21, no.2, pp.120-126

5- ITU-T Recommendation X509, ISO/IEC 9594-8 Information Technology-Open System Interconnection-The Directory: Authentication Framework, 1997

6- Menezes,A., Oorschot P.V.ve Vanstone S.,1996. Handbook of Applied Cryptography: 283-319

7- PGP Corporation, 2002. An Introduction to Cryptography : 33-57

8- Diffie, W., 1998. The First Ten Years of Public Key Cryptography, Procedings of te IEEE., May 1998

9- Salomaa, A.,1996.Public Key Cryptography, New York: Springer-Verlag

11- Çenesiz, F., 2001. Kurumsal Ağlarda Sayısal İmza ve Güvenlik. Gebze İleri teknoloji Enstitüsü, (Yüksek Lisans Tezi)

12- Değermen, M. Ç., 1996. Açık Sistem Veri İletişim Ağlarında Kriptografik Anahtar Yönetimi

13- Levi, A. Ve Özcan, M., 2003. Güvenli E-posta için Anahtar Yönetim Protokolleri, Bilişim 2003, 2-6 Eylül 2003, İstanbul

14- Levi, A., Nasıl bir e-posta güvenliği, Bilişim Güvenliği, Mart/Nisan 2003: 38-40

15- Levi, A. Ve Özcan, M., 2002. Açık Anahtar Tabanlı Şifreleme Neden Zordur?, Bilişim 2002, 3-6 Eylül 2002,İstanbul:41-45

16- Levi, A. Ve Cağlayan M.U., 1998. Türkiye için bir Açık Anahtar Altyapısı Modeli, Bilişim 98, 2-6 Eylül1998,İstanbul:354-361

17- Levi, A. Ve Cağlayan M.U., 1997. Elektronik Posta Güvenliği ve Açık Anahtar Sunucuları, Bilişim 97, 3-6 Eylül 1997,İstanbul:114-117

18- Levi, A. Ve Cağlayan M.U., 1997. Elektronik Posta Güvenliği için PGP kullanımı, Açık Sistem Sempozyumu, 19-21 Mart 1997,İstanbul:39-46

19- Levi, A. Ve Cağlayan M.U., 1996. Bilgisayar İletişiminde Kimlik Kanıtlama ve Güvenli Üçüncü Kişi Sorunları, 4. Bilgisayar-Haberleşme Sempozyumu, 11-15 Aralık 1996,Bursa:85-88

20- Cantu, M., Çeviren: Çölekçi, M.,2000. Delphi 5 Uygulama Geliştirme Kılavuzu:870

21- Yanık, M.,1997. Borland Delphi ile Görsel Programlama:568

22- Şamlı, M., 2002. PHP ile Web Programcılığı:390

23- Tarhan, C., 2001., Dreamweaver:273

24- Pekgöz, N., 2001.,JavaScript:250

25- Balaban, E., 2003. Web Tasarım Kılavuzu:1094

26- Greenspan, J. Ve Bulger, B., 2001. MySQL/PHP DatabaseApplications:622

27- Welling, L. Ve Thomson, L., 2001. PHP and MySQL Web Development:893

28- Goodman, Danny., 2001. JavaScript Bible, Gold Edition:2177

29- Wootton, C., 2001. JavaScript Programmer's Reference: 2625

30- Ratschiller, T. Ve Gerken T., 200. Web Application Development with PHP 4.0:416

TEŞEKKÜR

Bu çalışmayı yaparken değerli bilgilerine ve görüşlerine başvurduğum, tezin her aşamasında yardımlarını esirgemeyen ve yol gösteren tez danışmanım Doç. Dr. Mammadagha MAMMADOV'a,

Çalışmalarımda her türlü yardımı ve desteği sağlayan, başta bölüm başkanımız Prof. Dr. Hülya YILDIRIM ve çalışma arkadaşlarıma,

Ayrıca maddi ve manevi desteğiyle her zaman yanımda olan, rahat ve huzurlu bir çalışma ortamı sağlayıp, tezin her aşamasında beni destekleyen aileme teşekkürlerimi bir borç bilirim.

ÖZGEÇMİŞ

Adı Soyadı: Ercan ÇAĞLAR

Doğum Yeri ve Yılı: Ankara – 25.09.1977

Adres: Esenler mah. Karaca Sitesi A Blok D:9 ÇANAKKALE

Eğitim Durumu :

1983 – 1988 : Bir Eylül İlkokulu UŞAK

1988 – 1991 : Atatürk Ortaokulu KİLİS

1991 – 1994 : Manavgat Lisesi Manavgat/ANTALYA

Staj ve Kurslar:

06.07.1998 – 06.08.1998 : ADD-ON Bilgisayar & Yazılım Firmasında Staj

06.02.2002 – 08.02.2002 : IV. Akademik Bilişim Konferansı, Selçuk Üniversitesi

27.01.2003 – 31.01.2003 : Coğrafi Bilgi Sistemleri Kursu, İşlem Coğrafi Bilgi Sistemleri ve Mühendislik Ltd. Şti. ANKARA

28.02.2004 – 18.04.2004 : Görüntü İşleme Kursu, Çanakkale Onsekiz Mart Üniversitesi, ÇANAKKALE

Mesleki Deneyim:

2000 – 2001 : Çanakkale Onsekiz Mart Üniversitesi, Bilgi İşlem Dairesi, Çözümleyici

2001 – : Çanakkale Onsekiz Mart Üniversitesi, Eğitim Fakültesi, Bilgisayar ve Öğretim Teknolojileri Eğitimi Bölümü, Öğretim Görevlisi

Çalışma ve İlgi Alanları:

Yazılım, Nesneye Yönelik Programlama, Veri Tabanları, Kriptografi, Coğrafi Bilgi Sistemleri