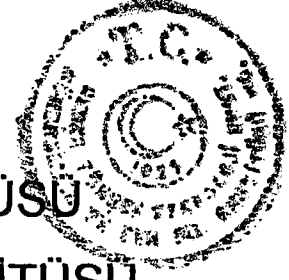


T.C.

GEBZE YÜKSEK TEKNOLOJİ ENSTİTÜSÜ
MÜHENDİSLİK VE FEN BİLİMLERİ ENSTİTÜSÜ



MALZEME SİPARİŞ SİSTEMİ YAZILIMI
VE SAYISAL İMZA UYGULAMASI

Levent YALÇINKAYA
YÜKSEK LİSANS TEZİ
BİLGİSAYAR MÜHENDİSLİĞİ
ANABİLİM DALI

109910

T.C. YÜKSEKÖĞRETİM KURULU
DOKÜMANTASYON MERKEZİ

TEZ DANIŞMANI
Dr. İbrahim SOĞUKPINAR

GEBZE

2001

Levent YALÇINKAYA.....'ın tez çalışması, G.Y.T.E. Mühendislik ve Fen Bilimleri Enstitüsü Yönetim Kurulu'nun 09.09.2001..... tarih ve 2001/121..... sayılı kararıyla oluşturulan jüri tarafından Bilgiyağır Mahallesi..... Anabilim Dalında YÜKSEK LİSANS tezi olarak kabul edilmiştir.

JÜRİ

ÜYE
(Tez Danışmanı)

: Yrd. Doç. Dr. İbrahim SOĞUKPINAR



ÜYE

: Yrd. Doç. Dr. Mehmet GÖKTÜRK



ÜYE

: Yrd. Doç. Dr. Hakan HOCAOĞLU



ONAY

G.Y.T.E. Mühendislik ve Fen Bilimleri Enstitüsü Yönetim Kurulu'nun 13.09.2001..... tarih ve 2001/114..... sayılı kararı.



ÖZET

Bu çalışmanın amacı merkezi bir organizasyon için, kendine bağlı birimlerinden gelen malzeme siparişlerinin WEB ortamında yapılmasını sağlayacak bir yazılım geliştirmek ve sayısal imza metodunu kullanarak güvenilir bir şekilde siparişleri toplayan Malzeme Sipariş Sistemini oluşturmaktır.

Malzeme Sipariş Sistemi iki ana yazılım modülünden oluşmaktadır. Bunlardan ilki uç kullanıcıların siparişlerini girebileceği ve sorgulayabileceği kullanıcı arayüzleri, diğeri de yapılan siparişleri sayısal imzası ile beraber merkezi otoriteye gönderen ve merkezi otorite tarafından gelen bilgilerin gerçek sahibinin kontrol edildiği güvenlik modülüdür. Güvenliği arttırmak için sadece kullanıcı kodu ve şifre ile sınırlı kalmadan sayısal imza metodu da sisteme dahil edilmiştir.

Temelde el ile atılan imzalar ile aynı işlevi gören sayısal imzaların bazı önemli avantajları vardır. Sayısal imzalar, el ile atılan imzalar gibi sabit olmayıp, bir başka deyişle sadece imzayı atan kişiye bağlı olmayıp, mesajdan mesaja değişiklik göstermektedir. Ayrıca sayısal imza mesajın imzalandıktan sonra bir bit'inin bile değişmediği garantisini taşır. Çünkü sayısal imza sadece imzayı atan kişiye bağımlı değil, aynı zamanda mesajın kendisinin bir özetinden de oluşmaktadır.

Önerilen yapının ilk modülü olan Malzeme Sipariş Sistemi 27 Mart 2000 tarihinde uygulamaya konmuş ve halen merkezi organizasyona bağlı Türkiye genelinde 31 birim tarafından kullanılmaktadır. Uygulama Extranet/İntranet ortamında Merkezi Otorite ve buna bağlı birimler tasarlanarak geliştirilmiş olmakla beraber, İnternet ortamından da müşteri kaydı yapılması, kaydedilen müşterinin siparişlerinin alınması için herhangi bir değişiklik ve ilaveye ihtiyaç göstermemektedir.

İkinci modül olan Güvenlik Modülünün test aşaması tamamlanmıştır. 2002 Ocak ayından itibaren Malzeme Sipariş Sistemi ile bütünleştirilerek işleme alınacaktır.

SUMMARY



The aim of this thesis is to produce a software for central authority that collects orders securely from business parts connected to it over Extranet/Internet by using digital signature technology and build Order Managment System.

Order Managment System is consist of two main modules. One of them is for Extranet/Internet end users to make their orders and query the orders, the other one is security module that sends the orders from end user to central authority with digital signature appended and verifies the signature on the server side (central authority). To increase the security digital signature technology is added to the System instead of using user id and password only.

Hand made signature and digital signature almost have the same funtionality but digital signatures have some advantages over hand made signatures. Digital signatures are not static as hand made ones. Digital signatures change from message to message, they are not depend oo the owner only. Digital signatures guerenty that message was not changed after it is signed, because as I mentioned before digital signature is not depend on the owner only, it is the summary of the message also.

Order Managment System is a live system since 27 March 2000 and is used by 31 business part of the central authority all around the Turkey. The System is designed for central authority and connected business parts of that authority over Extranet/Intranet. But you can use it over Internet for order managment of customers without any modification.

TEŞEKKÜR

Tez çalışmalarım süresince görüş ve önerileri ile beni yönlendiren danışmanım Yrd.Doç.Dr. İbrahim SOĞUKPINAR hocama, özellikle Malzeme Sipariş Sistemi Yazılımı'nın Güvenlik Modülünün kodlaması aşamasında JAVA programlama dili bilgisi ile bana yardımcı olan Müh. Yzb. Akif İZGİ' ye ve her konuda bana destek olan eşim Müh. Esin YALÇINKAYA' ya teşekkürlerimi sunarım.



İÇİNDEKİLER DİZİNİ



ÖZET	iv
SUMMARY	v
TEŞEKKÜR	vi
İÇİNDEKİLER DİZİNİ	vii
SİMGELER VE KISALTMALAR DİZİNİ	viii
ŞEKİLLER DİZİNİ	ix
1. GİRİŞ	1
2. BİLGİ GÜVENLİĞİ	3
2.1. Kripto Sistemleri Ve Uygulamaları	3
2.1.1. Gizli Anahtarlı (Simetrik) Kripto Sistemleri	4
2.1.2. Açık Anahtarlı (Asimetrik) Kripto Sistemleri	5
2.1.3. Mesaj Özetleme Tekniği	6
2.1.4. Sayısal İmza Teknolojisi	9
2.1.5. Anahtar Dağıtım Tekniği	10
2.1.5.1. Açık Dağıtım	11
2.1.5.2. Açık Klasör Yapılanması	11
2.1.5.3 Açık-Anahtar Otorite Kontrolü	12
2.1.5.4 Açık Anahtar Sertifikaları	12
3. MALZEME SİPARİŞ SİSTEMİNDE SAYISAL İMZA UYGULAMASI	14
3.1. Sistemin Yapısı	14
3.2 Sistemin İşleyişi	15
4. MALZEME SİPARİŞ SİSTEMİNİN TEST VE GERÇEKLENMESİ	21
5. SONUÇ VE ÖNERİLER	25
KAYNAKLAR	29
ÖZGEÇMİŞ	30
EKLER	
EK 1. GÜVENLİK MODÜLÜ KAYNAK KODLARI	

SİMGELER VE KISALTMALAR DİZİNİ



Simgeler	Açıklama
GYTE	Gebze Yüksek Teknoloji Enstitüsü
PC	Personal Computer
RSA	Rivest, Shamir, Adleman
DES	Data Encryption Standard
MD4/5	Message Digest 4/5
IP	Internet Protocol
TCG	Türkiye Cumhuriyeti Gemisi
TCSG	Türkiye Cumhuriyeti Sahil Güvenlik
OBI	Otomatik Bilgi İşlem
MSS	Malzeme Sipariş Sistemi

ŞEKİLLER DİZİNİ



Şekil	Sayfa
2.1 Gizli Anahtarlı Kripto Sistemi	4
2.2 Özet Fonksiyonlarının Temel İlkeleri	7
2.3 Sayısal İmzanın İşleyişi	10
2.4 Açık Dağıtım	11
2.5 Açık Klasör Yapılanması	11
2.6 Açık-Anahtar Otorite Kontrolü	12
2.7 Açık Anahtar Sertifikaları	13
3.1 Malzeme Sipariş Sisteminin Yapısı	15
3.2 MSS WEB Sayfası	16
3.3 İlk Kullanıcı Bilgi Mesajı	16
3.4 Gizli Anahtar Kayıt Ekranı	17
3.5 Açık-Gizli Anahtarların Konumları	17
3.6 MSS Ana Menüsü	18
3.7 Sayısal İmza Uygulaması	19
4.1 Giriş Ekranı	21
4.2 Test Kurulumu	22
4.3 Test Sipariş Giriş Ekranı	22
4.4 Uyarı Mesajı	23
5.1 MSS- Yıllara Göre Sipariş Miktarları	25
5.2 MSS- Bölgelere Göre Sipariş Dağılımı	26
5.3 On-line/Kağıt Ortamında Sipariş Oranları	27



1. GİRİŞ

Türkiye'nin özellikle yazılım geliştirme açısından sayılı kuruluşlarından olan Envanter Kontrol Merkezi Komutanlığı'nca malzeme siparişlerinin toplanmasına yönelik günün teknolojik imkanları kullanılarak hazırlanmış Yedek Malzeme Sipariş Sistemi 10 yılı aşkın süredir hizmet vermiştir. Bahse konu yazılım Türkiye'nin çeşitli bölgelerinden Envanter Kontrol Merkezi Komutanlığı'ndaki ana çatı (main frame) bilgisayarına terminal olarak bağlanıp ihtiyaç sahibinin malzeme talebini iletmesi üzerine tasarlanmıştır.

1990'lı yılların sonundan itibaren Deniz Kuvvetleri Komutanlığı'nın artan gemi adedi ve seyir saatlerine paralel olarak özellikle yedek parça siparişlerinde hissedilir artışlar meydana gelmiştir.

Gerek siparişlerdeki bu artışlar gerekse mevcut yazılım/donanım teknolojisinin güncellenmesi ihtiyacından dolayı yeni bir yazılım hazırlanması gerekliliği ortaya çıkmıştır. Çağımızda bilginin ne kadar önemli hale geldiği aşikardır. Bu nedenle bilgi güvenliğinin de ön plana alınması gerekliliği kabul görmüştür.

Bilhassa sayısal imza teknolojisinin oldukça yeni olması nedeniyle ülkeler ve hatta firmalar bazında henüz oturmuş bir standardizasyonun oluşmadığı görülmüştür. Özellikle standardizasyonun sağlanmasında öncü devletlerden biri olan ABD'de Ulusal Standartlar ve Teknoloji Geliştirme Enstitüsü(NIST), elektronik kimlik belgesi hazırlayan, içinde AT&T, Motorola, VeriSign gibi kuruluşların da bulunduğu 10 onay kurumunun yazılımlarını inceleyerek, bütün bu yazılımların uyum içinde çalışması için gerekli ölçütleri belirleme çalışmaları yapmaktadır [Elektronik Ticaret Koordinasyon Kurulu Teknik Raporu, 1998].

Bu araştırmalar sonucunda, geliştirilen Malzeme Sipariş Sistemi için hizmet verecek sayısal imza modüllerinin de hazırlanmasına karar verilmiştir. Böylece herhangi bir firmanın sunduğu ve özellikle sayısal imzanın saklama ve taşıma

2

kriterlerine baęlı kalmadan, hazırlanan gvenlik yazılımı sayesinde uygulama bazında esnek bir yapı oluřturulmuřtur. Kullanıcı kendisi iin verilen ve sayısal imzayı oluřturmada ana girdi olan gizli anahtarını istedięi bir ortamda (disket, sabit disk, cd-rom vb) saklayabilecek ve gerektięinde istedięi bilgisayardan kullanabilecek imkana kavuřmuřtur.



2. BİLGİ GÜVENLİĞİ



Bilgisayar dünyasının en önemli ilerlemelerinden biri olan ağ kavramının ortaya çıkışı 1960 yıllarına uzanmaktadır [Soğukpınar ve Çenesiz, 2000]. Bilgisayar ağlarının çok yaygın olarak kullanılmaya başlanması ise 1980'lerde PC'lerin yaygınlaşmasıyla olmuştur. Böylece başta güvenlik olmak üzere önceden yeterince ele birçok sorun da ortaya çıkmaya başlamıştır. Büyük ağlardaki güvenlik ve güvenilirlik sorunları ve bunlara yönelik çözümler ilk kez 1976 yılında ortaya kondu [Diffie and Helman, 1976]. 1977 yılında Ronald Rivest, Adi Shamir ve Len Adleman RSA Açık Anahtarlı Kripto Sistemini geliştirerek uygulamaya soktular.

Çok geniş bir kullanıcı grubuna hizmet eden ve yoğun olarak kullanılan Malzeme Sipariş Sisteminde, güvenliği ve güvenilirliği arttırmak için sayısal imza teknolojisi kullanılmıştır. Sadece kullanıcı kodu ve şifresi ile sağlanan güvenlikteki eksiklikler çeşitli uygulamalarda görülmüştür. Kullanıcılar şifrelerinde çocuklarının adları, doğum günü gibi kolay tahmin edilebilen yapılar yeğlediklerinden sistem dışardan gelebilecek sızmalara karşı duyarlıdır. Özellikle kritik malzemelerin hatalı veya kötü niyetli siparişlerini önlemek, sipariş bilgilerinin güvenilirliğini sağlamak amacıyla Malzeme Sipariş Sisteminde sayısal imza teknolojisi kullanılmıştır.

Sayısal imza teknolojisini anlayabilmek için, buna temel oluşturan Kripto Sistemini ve Mesaj Özetleme Tekniklerini iyi anlamak gerekir.

2.1. Kripto Sistemleri ve Uygulamaları

Günümüzde elektronik güvenlik iki ana alanda ön plana çıkmaktadır. Bunlar, bilginin bir sistemden diğerine aktarıldığı andaki iletişim güvenliği ve bilginin bilgisayarda depolandığı süreç içindeki güvenliğidir. Her iki aşamada da güvenliği arttırmak ve riskleri en aza indirmek için çeşitli kripto yöntemleri uygulanmaktadır.

Kripto Sistemleri üç ayrı bölümde katagorize edilebilirler:

- a. Açık metni şifreli metne çevirmekte kullanılan algoritma tipine göre: Bütün şifreleme algoritmaları iki genel prensibe göre çalışır. Bunlardan

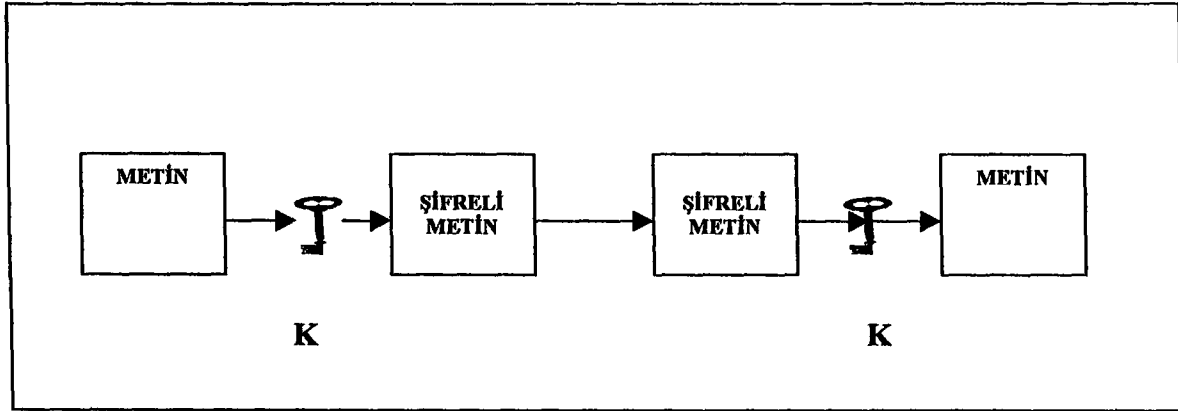
biri, açık metindeki her elemanın başka bir elemana dönüştürülmesi, diğeri ise açık metindeki elemanların yerlerinin değiştirilmesi.

- b. Kullanılan anahtar sayısına göre: Hem gönderen hem de alıcı aynı anahtarı kullanıyor ise sisteme simetrik, tek-anahtarlı, gizli anahtarlı sistem denir. Eğer gönderici ve alıcı ayrı anahtarları kullanıyor ise sisteme asimetrik, iki-anahtarlı veya açık-anahtarlı sistem denir.
- c. Açık metnin işleniş yöntemine göre: Blok şifreleme girdi olarak bir eleman grubu alarak her girdi bloğu için bir çıktı bloğu üretir [Feistel, 1973]. Dizi şifreleme her elemanı tek tek girdi alır ve her biri için çıktı elemanı üretir.

Sayısal imzanın dayanağı Açık Anahtarlı Sistemler olduğundan Kripto Sistemlerini kullanılan anahtar sayısına göre katagorize edilmiş olarak incelenecektir.

2.1.1. Gizli Anahtarlı (Simetrik) Kripto Sistemleri

Tek anahtarlı olarak da adlandırılan Gizli Anahtarlı Kripto Sisteminde, sistemin tek anahtarı kullanıcılara gizli olarak dağıtılır. Metin bu anahtarla şifrelenir ve yine bu anahtar kullanılarak açılır (Şekil 2.1).



Şekil 2.1: Gizli Anahtarlı Kripto Sistemi.

Bu sistemde anahtar hem şifreleme hem de deşifreleme işlemi yaptığı için çok iyi korunmalı ve yetkisiz kişilerin eline geçmemelidir. 1977 yılında National Bureau of Standards tarafından geliştirilen Data Encryption Standard (DES) Gizli Anahtarlı Kripto Sistemlerinin içinde en yaygın olarak kullanılanıdır. Bu tür sistemlerin

performansı genelde iyi olmakla birlikte tüm kullanıcılar yeni anahtarı aynı anda değiştirmek durumunda olduklarından, sistemin genel performansı düşmektedir.



2.1.2. Açık Anahtarlı (Asimetrik) Kripto Sistemleri

İki anahtarlı sistem olarak da adlandırılabilen Açık Anahtarlı Kripto Sistemi bir çift anahtar kullanmaktadır. Bunlardan ilki açık olarak dağıtılan ve her kullanıcının elde edebileceği açık anahtar, diğeri ise gizli ve kullanıcıya özgü olan anahtardır. Açık ve Gizli anahtar ikilisi sistem içinde birlikte kullanılmakta ve birinin şifrelediği metni diğeri açabilmektedir.

E ve D sırasıyla şifreleme ve deşifreleme işlemini göstermek için kullanılırsa ve M de gönderilecek metin olarak belirlenirse Açık Anahtarlı Kripto Sistemi şu dört özellikte tanımlanabilir [Rivest et al, 1977]:

- * Şifrelenmiş M mesajının deşifre edilmesi tekrar M mesajını oluşturur;
 $D(E(M)) = M$.
- * Hem E , hem de D işlemleri kolay hesaplanabilir olmalıdır.
- * Açık dağıtılan anahtardan gizli anahtarın hesaplanamaması gerekir.
- * M mesajı önce deşifrelenir sonra şifrelenir ise yine M elde edilmelidir;
 $E(D(M)) = M$.

1977 yılında Rivest, Shamir ve Adleman tarafından geliştirilen RSA ve 1978 yılında El-Gamal tarafından geliştirilen El-Gamal algoritmaları günümüzde en yaygın olarak kullanılan Açık Anahtarlı Kripto Sistemi algoritmalarıdır.

Açık anahtarlı kripto sisteminde ' M ' metnini şifrelemek için önce mesajı 0 ile $n-1$ arasında sayı cinsinden ifade etmek gerekir. Eğer metin çok uzun ise bloklara ayrılabilir.

Sonra metnin e üzeri alınarak bulunan sayı modulo n işlemine tabi tutulur ve şifreli metin ' C ' elde edilmiş olur. Burada bahsi geçen n ve e pozitif sayılardır.

Deşifreleme için şifreli metnin ('C') d üzeri alınır ve bulunan sayı modulo n işlemine tabi tutulur. Böylece orijinal metin 'M' elde edilir.

Şifrelemek metnin boyutunda bir artışa neden olmaz gerek metin ve gerekse şifreli metnin boyutu 0 ile n-1 sayısı arasındadır. Şifreleme ve deşifreleme işlemlerinden de anlaşılacağı gibi e şifreleme anahtarı, d ise deşifreleme anahtarıdır. Peki bu iki anahtarı nasıl elde edebiliriz?

İlk önce n bulunur. Bunun için iki çok büyük asal sayı p ve q rastgele olarak seçilir. Bu iki sayının çarpımından n elde edilir.

$$n = p * q$$

Daha sonra d sayısı, rastgele olarak büyük bir asal sayı, seçilir ve bu sayının $(p - 1) * (q - 1)$ eşitliği ile En Büyük Ortak Bölünü'nin (EBOB) 1 olduğu görülür.

$$\text{EBOB}(d, (p - 1) * (q - 1)) = 1$$

Son olarak d sayısı ile çarpımın sonucunda bulunan ve $(p - 1) * (q - 1)$ ile modulo işlemine tabi tutulduktan sonra 1'i veren e sayısı hesaplanır.

$$(e * d) \bmod ((p - 1) * (q - 1)) = 1$$

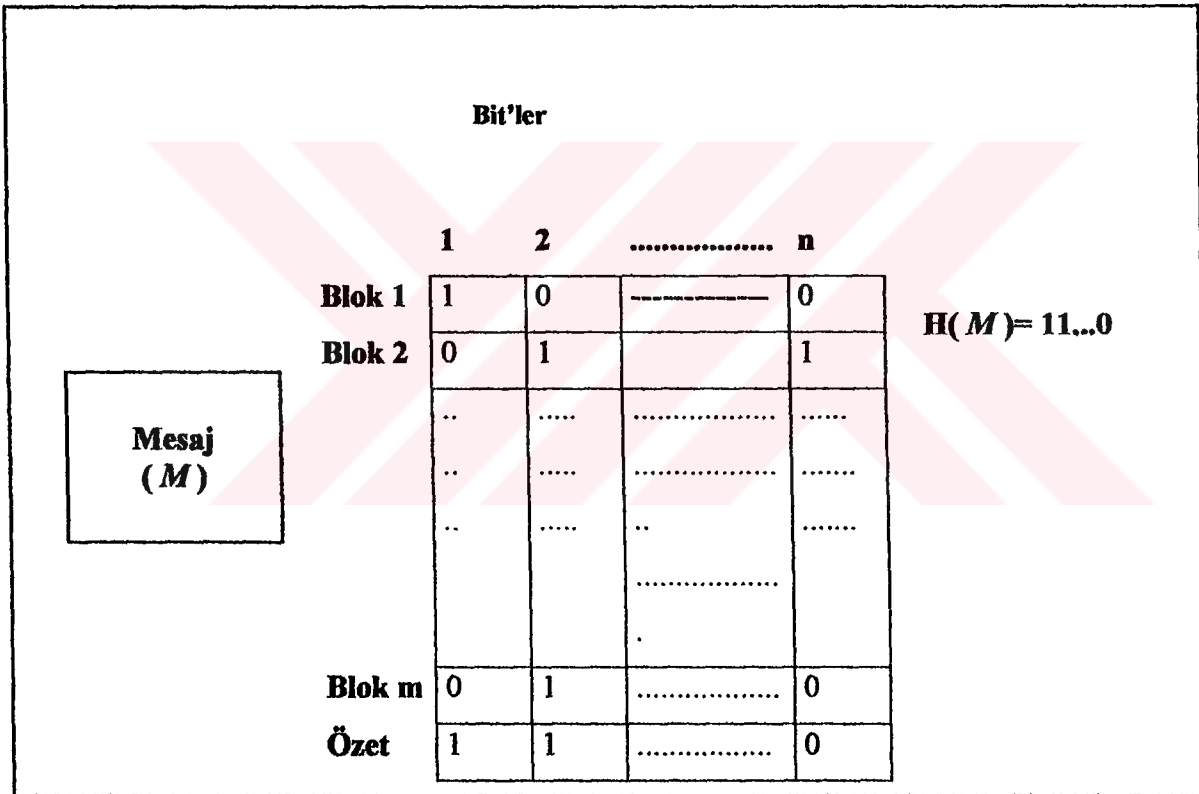
Artık metni şifreleyecek ve deşifreleyecek anahtar ikilisi oluşmuştur. Kullanıcı e'yi açık olarak yayımlar ve d'yi gizli anahtarı olarak tutar.

2.1.3. Mesaj Özetleme Tekniği

Bu teknikle, mesaj özetleme fonksiyonu ile değişken uzunlukta bir mesaj M girdi olarak alıp sabit uzunlukta bir özet $H(M)$ elde edilir. Mesaj içindeki her değişiklik farklı bir özetin oluşmasına neden olur. Mesajın parmak izini üretmek amacıyla olan özetleme fonksiyonu şu özelliklere sahip olmalıdır [Simmons, 1992].

- * H fonksiyonu herhangi uzunluktaki bir mesaja uygulanabilmelidir.
- * H sabit uzunlukta bir sonuç üretmelidir.
- * Herhangi bir mesaj M için, $H(M)$ kolay hesaplanabilir olmalıdır.
- * Mesajın özetinden mesajın kendisi elde edilememelidir.
- * Farklı iki mesajdan aynı özet elde edilememelidir.
- * Bir mesajdan farklı iki özet çıkarılamamalıdır.

Tüm özetleme fonksiyonları ortak ilkeler üzerinde çalışmaktadır (Şekil 2.2). Girdi olan mesaj n -bit' ten oluşan bloklara ayrılmakta ve her bir blok tek tek işleminden geçirilerek sonuçta n -bit' lik özet elde edilmektedir [Stallings, 1998].



Şekil 2.2. Özet Fonksiyonlarının Temel İlkeleri.

Ron Rivest tarafından geliştirilen MD4 ve MD5 [8], National Institute of Standards and Technology tarafından geliştirilen SHA-1 ve European RACE Integrity Primitives Evaluation proje grubu tarafından geliştirilen RIPEMD-160 [9] günümüzde en yaygın olarak kullanılan özetleme fonksiyonlarıdır.

Malzeme Sipariş Sistemi'nin sayısal imza modülünün temelini oluşturan MD5 algoritmasıdır. Bu algoritma değişken uzunlukta bir metni girdi olarak alır ve 128

bit'lik bir özet çıktı hazırlar. Hem algoritmanın incelenmesi hem de örnek olması amacıyla, elimizde b-bit'lik bir mesaj olduğunu varsayalım ve bu mesajın özetinin çıkarılması işlemini aşağıdaki 5 adımda inceleyelim:

- a. 1.Adım- Bit ilavesi: İlk önce mesajın boyu (bit olarak) 512 ile modulo işlemine tutulduğunda 448'i verecek şekilde bit ilavesi ile uzatılır. İlk olarak 1 bit'i daha sonra gerektiği kadar 0 bit'i ilave edilir.
- b. 2.Adım- Mesajın Boyunun Büyütülmesi: Mesajın orijinal boyutunu belirten 'b' sayısının 64-bit'lik gösterim şekli mesajın sonuna ilave edilir. Böylece mesaj 512 bit veya katları boyutuna ulaşır.
- c. 3.Adım- Mesaj Özet Buffer'larının Düzenlenmesi: Her biri 32'ser bit'lik 4 adet buffer (A, B, C, D) kullanılır. Bu buffer'lar şu şekilde düzenlenir (hexadecimal).

A = 01 23 45 67

B = 89 ab cd ef

C = fe dc ba 98

D = 76 54 32 10

- d. 4.Adım- Mesajın 16 word'lük Bloklar Halinde İşlenmesi: Her biri 3 adet 32 bit'lik word alıp bir adet word üreten dört adet fonksiyon işletilir.

$$F(X, Y, Z) = X \text{ and } Y \text{ or not}(X) \text{ and } Z$$

$$G(X, Y, Z) = X \text{ and } Z \text{ or } Y \text{ and not}(Z)$$

$$H(X, Y, Z) = X \text{ xor } Y \text{ xor } Z$$

$$I(X, Y, Z) = Y \text{ xor } (X \text{ or not}(Z))$$

Elde edilen sonuçlar sırası ile A, B, C ve D buffer'larında saklanır.

- e. 5.Adım- Özet: Mesajın özeti A, B, C ve D buffer'larında oluşmuştur. A'dan başlayarak D buffer'ının sonuna kadar bit'ler birleştirilir ve mesaj özeti elde edilir.

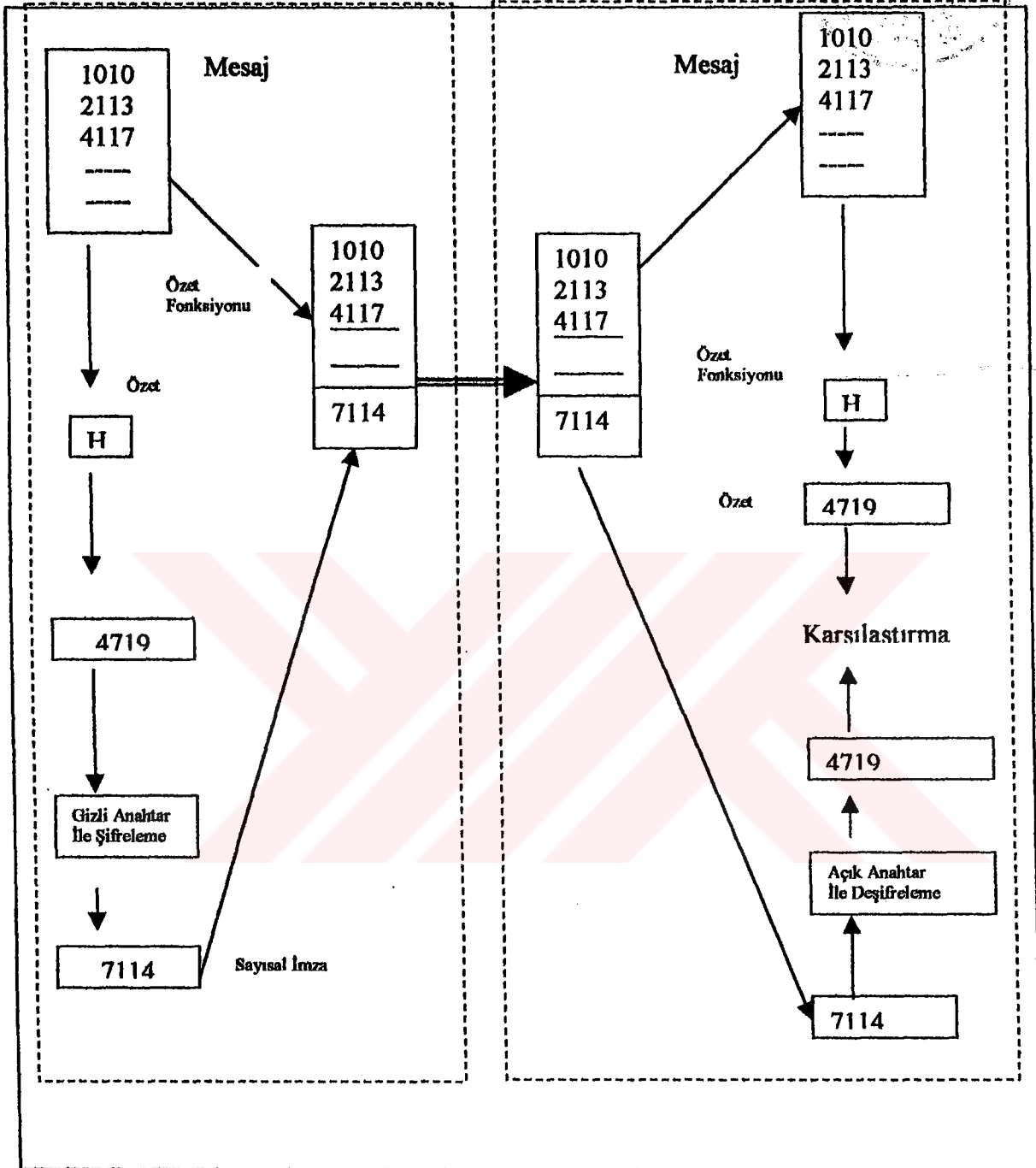
2.1.4. Sayı sal İmza Teknolojisi

Açık Anahtarlı Kripto Sisteminin ve Mesaj Özetleme Tekniğinin beraber kullanılması sayısal imza tekniği olarak tanımlanabilir. Sayısal imzalar şu üç özelliği sağlamalıdır [Garfinkel and Spafford, 1997].

- * Alıcı, gönderenin imzasını onaylama olanağına sahip olmalıdır.
- * İmza taklit edilememeli, bir başka deyişle gönderici dışında hiç kimse tarafından oluşturulamamalıdır.
- * İmzalanmış mesajı gönderici daha sonra reddedememelidir.

Temelde el ile atılan imza ile aynı işlevi görmelerine karşın sayısal imzaların önemli bazı özellikleri vardır. Sayısal imzalar el ile atılan imzalar gibi sabit olmayıp mesajdan mesaja değişmektedir. Ayrıca sayısal imza mesajın imzalandıktan sonra bir bit' inin bile değişmediği garantisini taşır. Çünkü sayısal imza mesajın kendisinden oluşturulmaktadır. Mesajdaki en küçük değişiklik farklı bir sayısal imza üretilmesine neden olur.

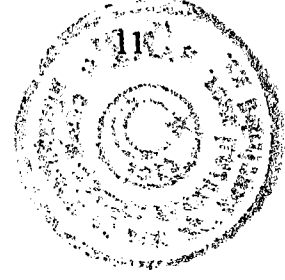
Sayısal imzanın genel işleyişi Şekil 2.3'te betimlenmiştir. Mesaja (M) özet fonksiyonu (H) uygulanarak sabit uzunlukta bir mesaj özeti elde edilmektedir. Daha sonra gönderici gizli anahtarı kullanarak bu özeti şifrelemektedir. Şifrelenmiş mesaj özeti sayısal imza olarak tanımlanmaktadır. Şifrelenmiş mesaj özeti, mesajın aslına ilave edilerek alıcıya gönderilmekte ve alıcı da gönderenin açık anahtarını kullanarak imzayı deşifre etmektedir. Yalnızca göndericinin açık anahtarı deşifre işlemini yapabileceği için, böylece gönderici doğrulanmış olur. Bu işlemten sonra mesajın kendisinden özet fonksiyonu sayesinde yeni bir özet çıkarılmaktadır. Bu özet ve deşifre edilen imza karşılaştırılmakta ve her ikisi de aynı ise gönderilen mesajın imzalandıktan sonra hiç değişmediği anlaşılmaktadır.



Şekil 2.3. Sayısal İmza'nın İşleyişi.

2.1.5. Anahtar Dağıtım Tekniği

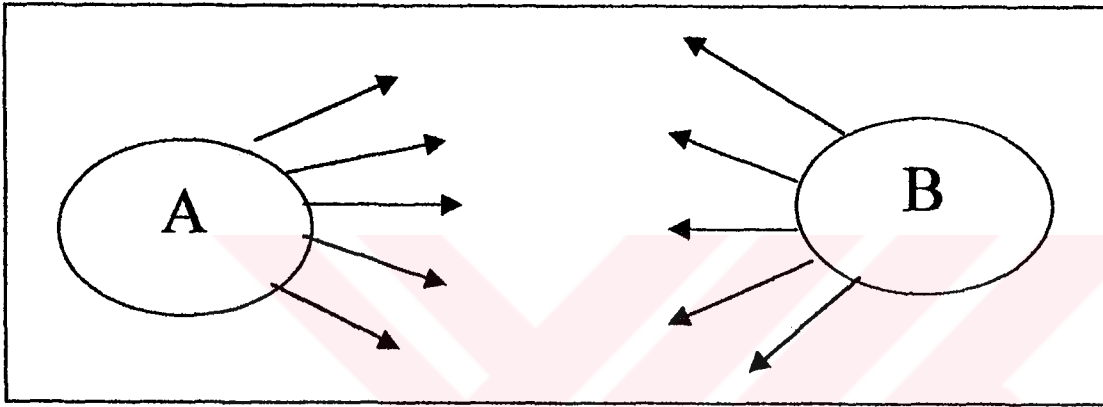
Kullanılan anahtarların dağıtımı için birçok teknik mevcuttur. Tüm bu teknikleri şu 4 kategoride toplamak mümkündür:



- a. Açık dağıtım
- b. Açık klasör yapılanması
- c. Açık-anahtar otorite kontrolü
- d. Açık-anahtar sertifikaları

2.1.5.1. Açık Dağıtım

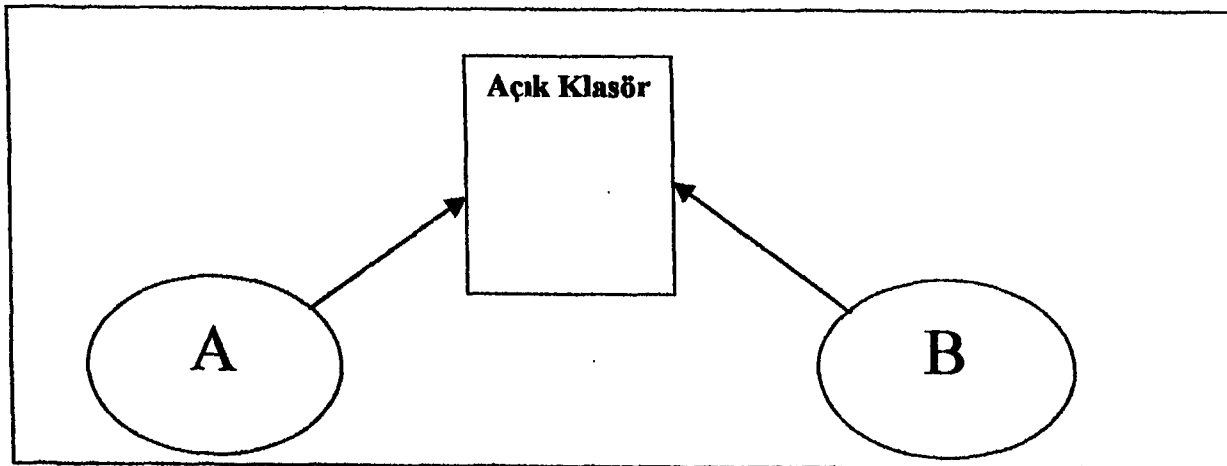
Ana çıkış noktası; iki anahtarlı krypto sistemlerinde açık anahtarın adı üstünde herkese açık olmasındadır. Açık anahtar ortama yayımlanır (broadcast edilir).



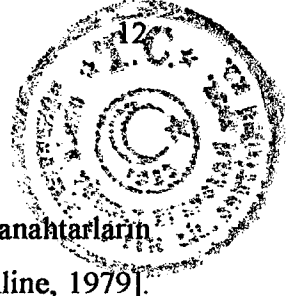
Şekil 2.4. Açık Dağıtım.

2.1.5.2. Açık Klasör Yapılanması

Açık dağıtımdan daha güvenli bir tekniktir. Belirli bir organizasyon tarafından idame edilen açık anahtarları içeren klasörlerin kullanıcılara açık olarak sunulmasıdır.

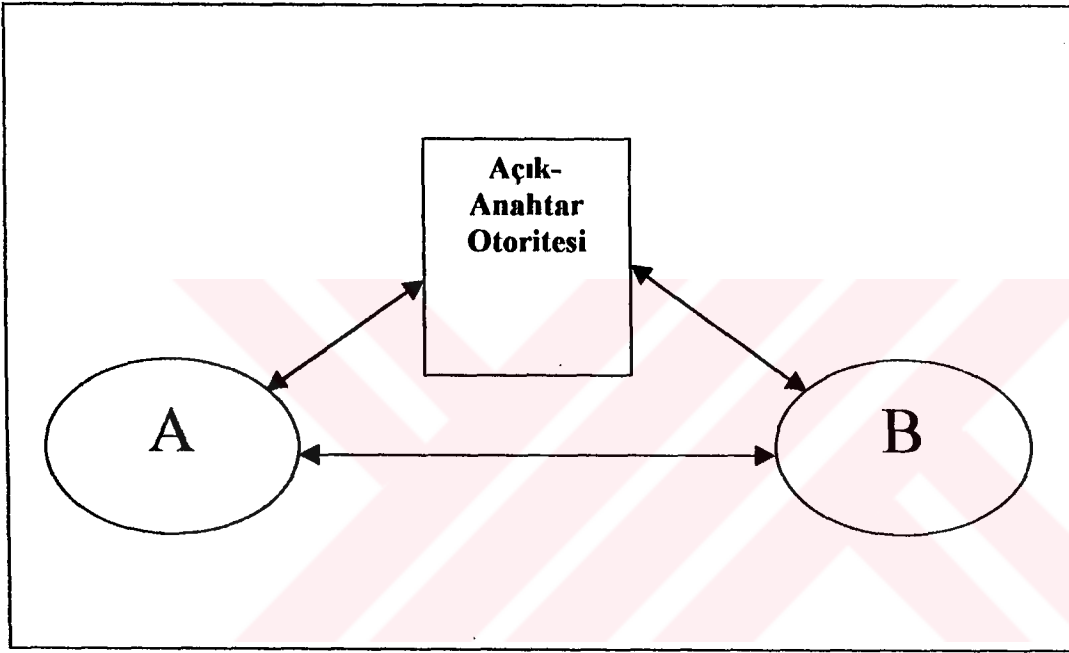


Şekil 2.5. Açık Klasör Yapılanması.



2.1.5.3. Açık Anahtar Otorite Kontrolü

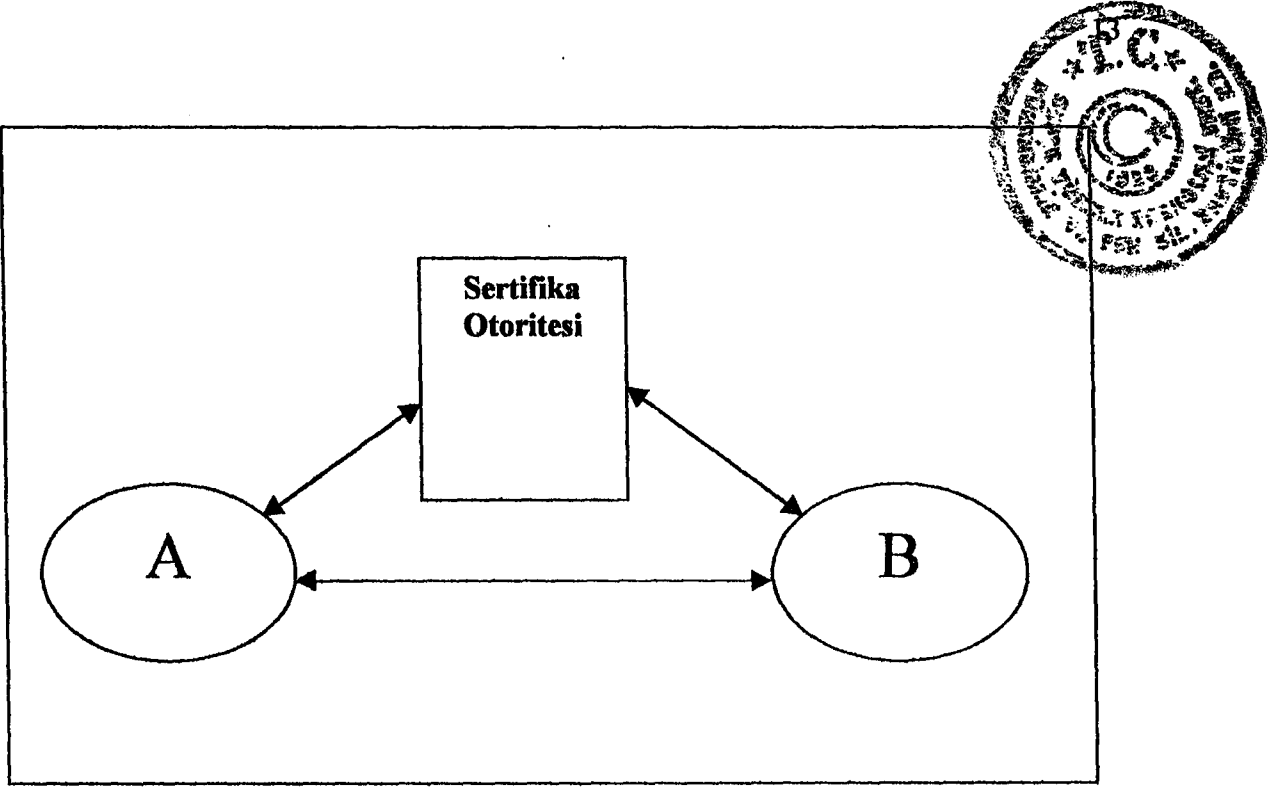
Açık klasör yapılanmasından daha güvenli bir tekniktir. Açık anahtarların bulunduğu klasör/dosyalara kontrollü erişim olanağı sağlar [Popek and Kline, 1979]. Herhangi bir kullanıcı başka bir kullanıcının açık-anahtarı için otoriteye başvurur. Bu başvuru sonucunda ancak iletişime geçeceği diğer kullanıcının açık anahtarını elde edebilir.



Şekil 2.6. Açık Anahtar Otorite Kontrolü.

2.1.5.4. Açık Anahtar Sertifikaları

Herhangi bir açık anahtar otoritesine ihtiyaç duymadan iletişime geçecek iki kullanıcının, sertifika otoritesinden aldıkları sertifika yardımı ile birbirlerine açık anahtarlarını gönderme metodudur [Kohnfelder, 1978].



Şekil 2.7. Açık Anahtar Sertifikaları.

3. MALZEME SİPARİŞ SİSTEMİNDE SAYISAL İMZA UYGULAMASI

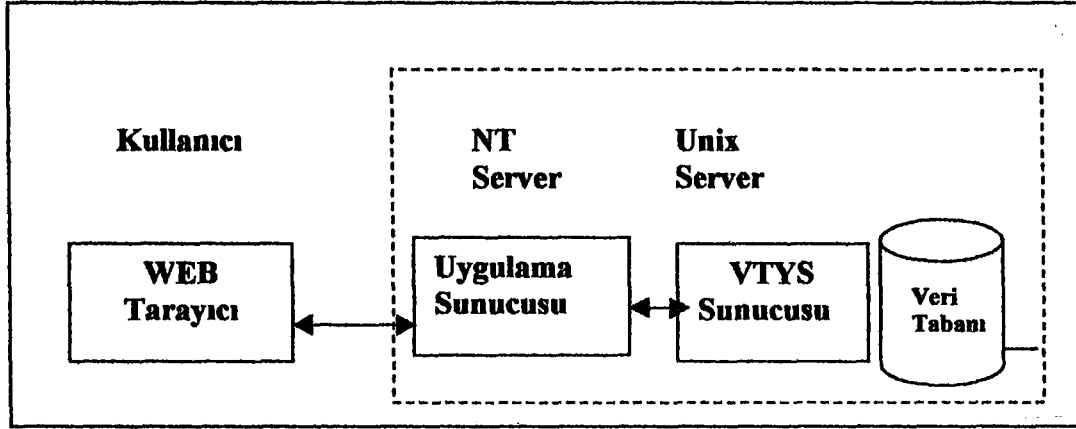
Extranet/İntranet altyapısını kullanan Malzeme Sipariş Sistemi uygulamasının amacı Merkezi Otoriteye bağlı birimlerce yapılan malzeme siparişlerinin varolan güvenlik teknolojilerini kullanarak, en etkin ve en güvenilir bir ortamda yapılmasını sağlamaktır. Uygulama çok geniş bir kitleye hitap ettiğinden extranet/intranet ortamında IP tabanlı olarak geliştirilmiştir.

3.1. Sistem Altyapısı

Malzeme Sipariş Sistemi SUN Solaris V.2.6 Unix işletim sistemli SUN Enterprise 3500 bilgisayarı üzerinde kurulu Oracle 8i veritabanı sunumcusu ve Microsoft Windows NT işletim sistemli bilgisayar üzerinde bulunan uygulama yazılımları ile Oracle iAS uygulama sunucusundan oluşmaktadır (Şekil 3.1).

İstemci tarafında herhangi bir yazılıma gereksinim duyulmamakla birlikte Windows Explorer veya Netscape Communicator WEB tarayıcıları ile sistemi kullanmak olasıdır. Gerek uygulama yazılımları gerekse güvenlik modülleri JAVA programlama diliyle yazılmıştır. Açık Anahtarlı Kripto Sistemi olarak RSA, Mesaj Özet Fonksiyonu olarak SHA-1 kullanılmıştır.

Güvenlik modülünde 2.Bölümde değinilen anahtar dağıtım tekniklerinden biri olan açık-anahtar otoritesi tekniği kullanılmıştır. Tüm kullanıcıların açık anahtarları bir otoritenin gözetiminde merkezi olarak saklanmaktadır. Açık anahtara ihtiyaç duyan kullanıcı, ki güvenlik modülünde siparişleri toplayan sunumcu yazılımıdır, açık anahtar otoritesinin gözetiminde diğer kullanıcının, ki siparişte bulunan kullanıcıdır, açık-anahtarını elde etmektedir.



Şekil 3.1. Malzeme Sipariş Sisteminin Altyapısı.

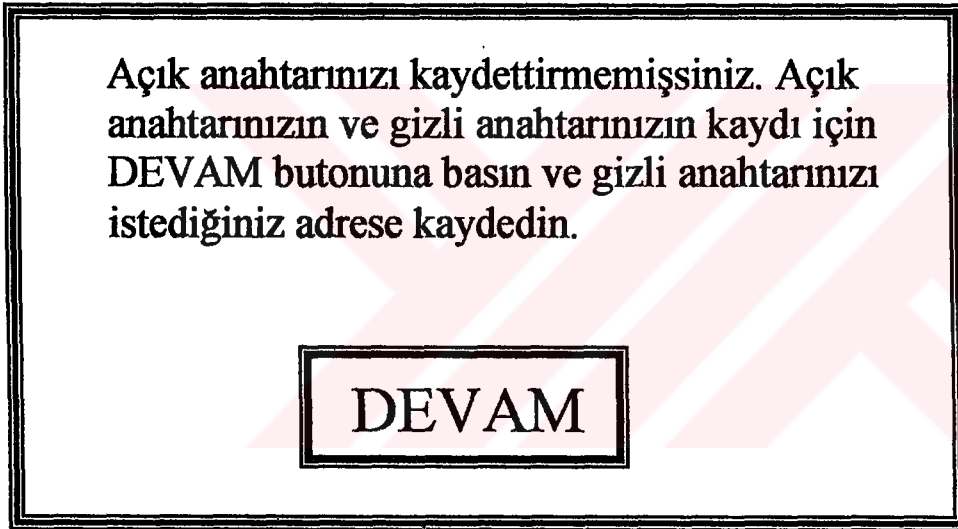
3.2. Sistemin İşleyişi

Malzeme Sipariş Sistemin güvenliğinde yalnızca kullanıcı kodu ve şifresine bağlı kalmanın bilgi teknolojilerinin hızla geliştiği günümüzde yeterli olmadığı gerçeğinden yola çıkılarak sistemde Açık Anahtarlı Şifreleme ve Sayısal İmza Teknolojisi kullanılmıştır. Açık anahtarlı şifrelemenin ve sayısal imzanın getirileri olan, sisteme giren kişinin gerçekten yetkili kişi olup olmadığı denetimi, kullanıcı tarafından gönderilen bilgilerin kaynağından çıktıktan sonra değiştirilmesinin önlenmesi ve bilgiyi gönderenin gönderimini daha sonra inkar edememesi işlevlerinden bu sistemde tam olarak yararlanılmıştır.

Kullanıcı kodu ve şifre ile sisteme girildiğinde (Şekil 3.2) sunumcu tarafında çalışan program kullanıcı kodu ve şifre kontrolü yapar. Herhangi birisinin hatalı olması durumunda sisteme girişi engeller. Kullanıcı kodu ve şifre doğru ise sunumcu programı kullanıcının açık anahtarının olup olmadığını kontrol eder ve eğer yok ise kullanıcının sistemi ilk olarak kullandığını değerlendirip Şekil 3.3’ de gösterilen bilgi mesajını verir.

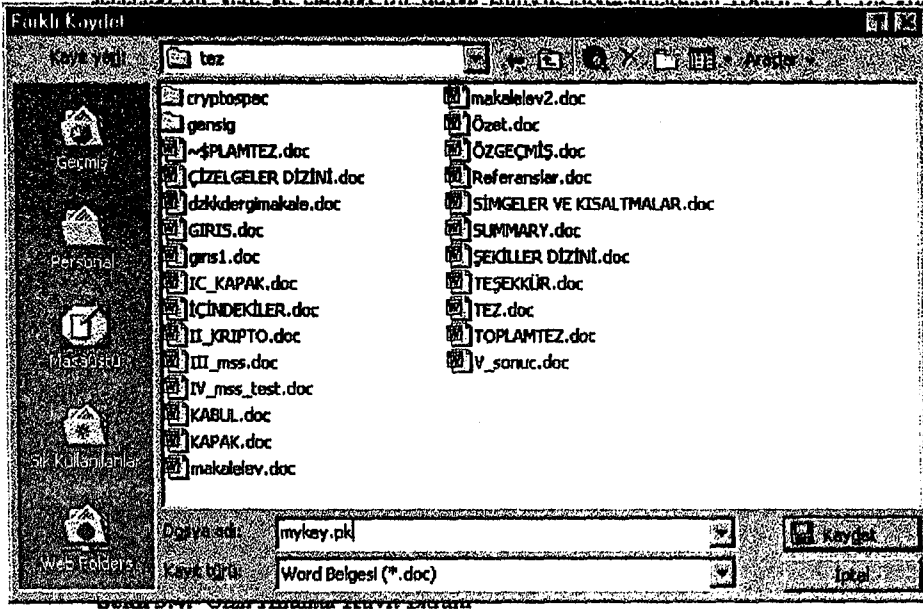


Şekil 3.2. MSS WEB Sayfası

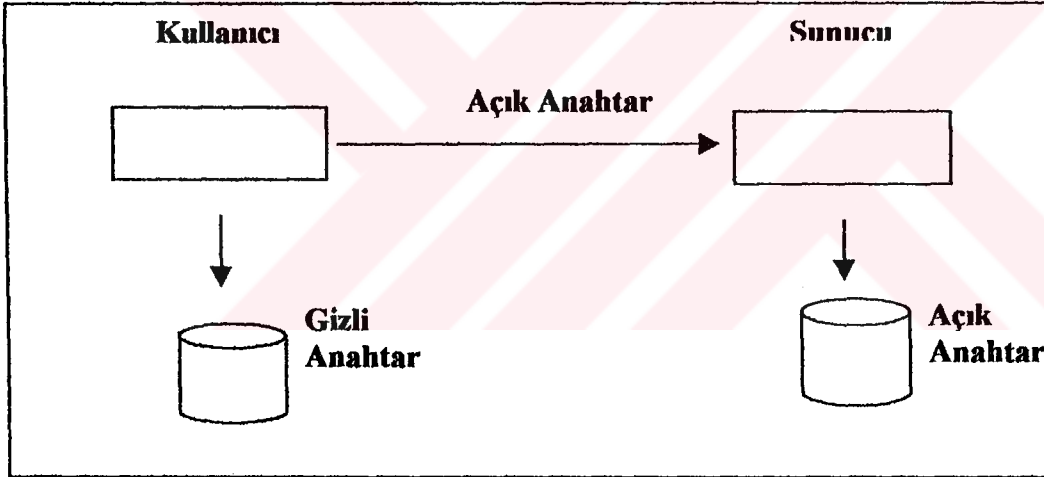


Şekil 3.3. İlk Kullanıcı Bilgi Mesajı

Devam düğmesine basıldığında istemci tarafında çalışan program açık ve gizli anahtar ikilisini oluşturur. Yazılım açık anahtar sunumcudaki veri tabanına; gizli anahtar da kullanıcının Şekil 3.4'deki ekranı kullanarak kendi bilgisayarında seçeceği bir yere ve istediği bir dosya ismiyle kaydetmektedir (Şekil 3.5). Bu gizli anahtarın başka bilgisayarlardan da sipariş açılırken kullanılması için, ilgili bilgisayara taşınması gereklidir. Bunun için gizli anahtar bir diskete kopyalanmalıdır. Diğer bilgisayarda ister disketten ister o bilgisayardaki herhangi bir yere kopyalamak suretiyle kullanılabilir.

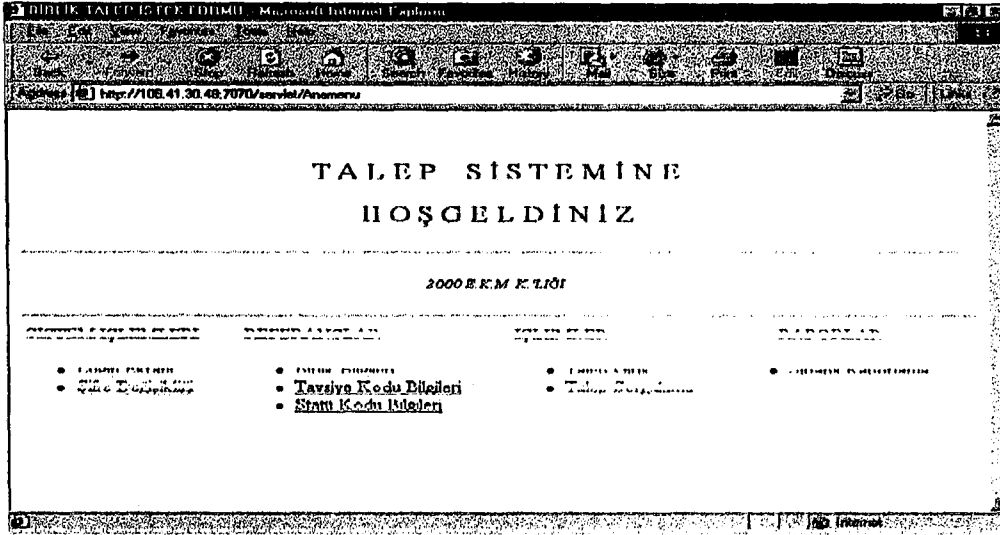


Şekil 3.4. Gizli Anahtar Kayıt Ekranı



Şekil 3.5. Açık – Gizli Anahtarın Konumları.

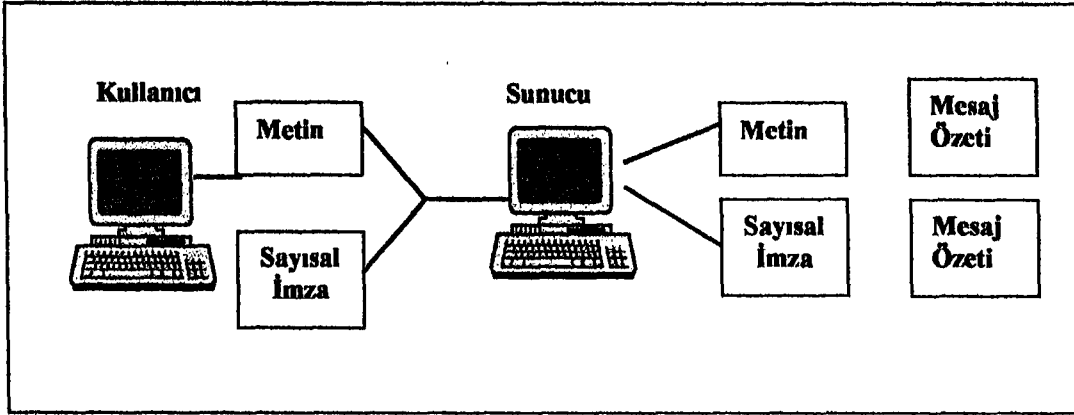
Açık ve gizli anahtarın oluşturulmasından sonra kullanıcı kodu ve şifre ile sisteme girildiğinde ve ekrana Malzeme Sipariş Sistemi Ana Menü'sü (Şekil 3.6) gelir.



Şekil 3.6. MSS Ana Menü'sü

Bu menü ile sipariş girişi yapılabilir, yapılacak ilk siparişle birlikte istemci tarafında çalışan program kullanıcının gizli anahtarının bulunduğu yeri kullanıcıya sorar. Aynı bağlantı sırasında yapılacak diğer siparişlerde gizli anahtarın yeri tekrar sorulmaz. Bağlantı herhangi bir şekilde kesilir ise her yeni bağlantıdan sonraki ilk sipariş için gizli anahtarın yeri sorulur.

Kullanıcı tarafından gönderilen sipariş bilgileri, yazılım vasıtası ile özet işlemine tabi tutularak bilgilerin bir özeti çıkartılmakta ve bilgiler kullanıcının gizli anahtarı ile şifrelenerek sayısal imza oluşturulmaktadır. Gönderilen bilgi ile birlikte, oluşturulan bu sayısal imza da alıcıya gönderilmektedir. Sunumcu tarafına gelen metin yeniden özet işlemine tabi tutulmakta ve yeni bir özet çıkartılmaktadır. Metnin arkasından gelen sayısal imza açık anahtar ile deşifre edilerek metin özeti ile karşılaştırılmakta; iki özet birbirinin aynısı ise metin kabul edilmekte ve bir tabloya gelen metin, sayısal imzası, o anki açık anahtar, kullanıcı kodu ve tarih kaydedilmektedir. İki özet birbirini tutmaz ise kullanıcının isteği geri çevrilmektedir (Şekil 3.7).



Şekil 3.7. Sayısal İmza Uygulaması.

Güvenlik modülünü biraz daha detaylı inceleyelim. Bu modül tamamen nesne tabanlı yazılım geliştirme teknolojisi ile hazırlanmıştır. Genel olarak, güvenlik hizmetlerini sunan bir paket, uç kullanıcı tarafında gizli-açık anahtarları oluşturup açık anahtarı açık-anahtar otoritesine kaydettiren ve sipariş sırasında imzayı oluşturan bir paket ve sunumcu tarafında yapılacak siparişleri ve ilgili imzaları toplayıp açık-anahtar otoritesinden kullanıcının açık anahtarını alıp imzayı kontrol eden paketten oluşmuştur.

Kullanıcı Sipariş Sistemine giriş için kullanıcı kodu ve şifresini girmek zorundadır. Kullanıcı tarafında çalışan güvenlik paketi bu kullanıcı kodu ve şifresini sunumcu tarafındaki pakete aktarmaktadır. Sunumcu tarafında çalışan paket gelen kullanıcı kodu ve şifresini gerçeklemede, eğer kullanıcı kodu ve şifresi hatalı ise kullanıcıyı bir mesaj ile uymaktadır. Kullanıcı kodu ve şifresi doğru ise ve fakat henüz açık-anahtar otoritesine kendi açık anahtarını kaydetmedi ise kullanıcı tarafındaki güvenlik paketine mesaj göndererek kendisini kaydettirmesi için uymaktadır. Kullanıcı tarafındaki paket bu uyarı ile otomatik olarak gizli-açık anahtar ikilisini oluşturmakta ve açık anahtarı açık-anahtar otoritesine kaydettirmektedir. Gizli anahtar ise kullanıcının belirleyeceği adreste saklanmaktadır.

Sipariş sistemine tam olarak girişe ancak, kullanıcı kodu ve şifresi doğru ise ve kullanıcı açık anahtarını açık-anahtar otoritesine kaydettirdiyse mümkün olmaktadır.

Sisteme giren kullanıcıdan, yapacağı ilk sipariş öncesinde, kullanıcı tarafındaki paket vasıtasıyla gizli anahtarının yeri sorulur ve yapmakta olduğu siparişin imzası oluşturulur. Sistemden çıkmadan yapacağı müteakip siparişler için tekrar gizli anahtarın adresi sorulmaz. Sipariş bilgileri ve imza sunumcu tarafındaki güvenlik paketi tarafından onaylanır, bu onaylama için gerekli olan kullanıcının açık anahtarı açık-anahtar otoritesinden alınır. Güvenlik modülünün kaynak kodları Ek-1’de verilmiştir.



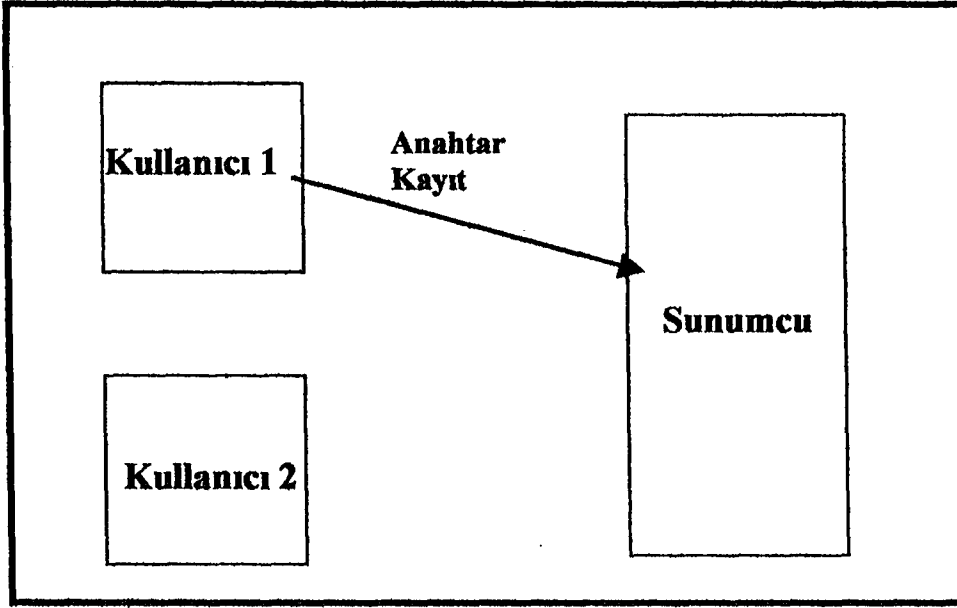
4. MALZEME SİPARİŞ SİSTEMİNİN GERÇEKLENMESİ VE TESTİ

Geliştirilen yazılımın test edilmesi amacıyla bir veritabanı sunucusu, uygulama sunucusu ve sistemin uç kullanıcılarını canlandırmak amacıyla iki istemci bilgisayar kurulmuştur. Veritabanı olarak Oracle 8i VTYS kurulmuştur. Bir kullanıcı Sipariş Sistemine Giriş Ekranını (Şekil 4.1) kullanarak girip açık ve gizli anahtar ikilisini yazılım aracılığı ile oluşturarak kendini sisteme tanıtmıştır. Diğer kullanıcı ise kullanıcı kodu ve şifresi olmasına rağmen kendini sisteme tanıtmamıştır.



Şekil 4.1. Giriş Ekranı

Birinci kullanıcı sisteme kendini tanıtırken, yani açık ve gizli anahtarını oluşturup açık anahtarını açık-anahtar otoritesine kaydettirirken, gizli anahtarını da kendi bilgisayarının diskine kaydetmiştir (Şekil 4.2).



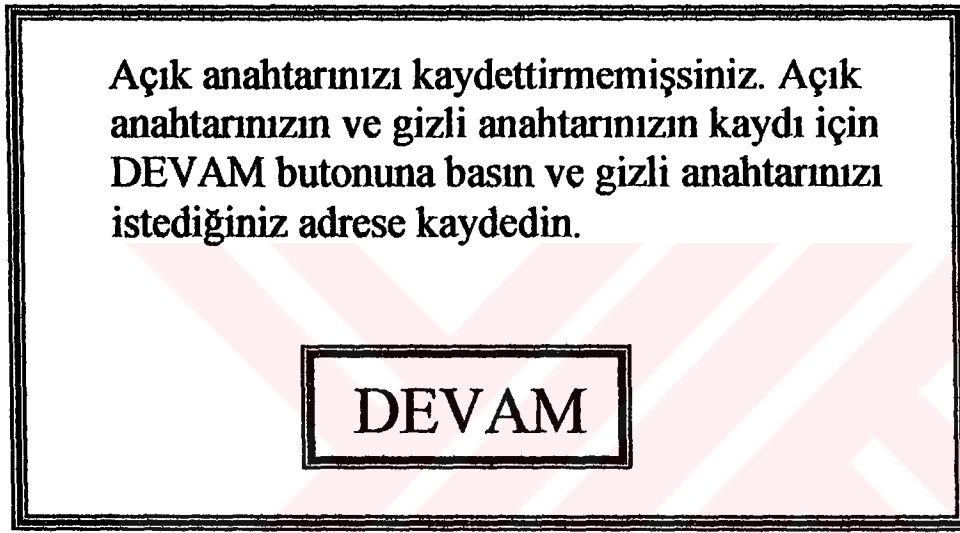
Şekil 4.2. Test kurulumu.

Daha sonra Sipariş Sistemine girerek test amacıyla hazırlanan Sipariş Giriş Ekranını (Şekil 4.3) kullanarak stok numarası, miktar ve dağıtım birimi bilgilerini doldurmuş ve sipariş işlemini başarıyla gerçekleştirmiştir.

Stok No	
DB	
Miktar	
Kaydet Vazgeç	

Şekil 4.3. Test Sipariş Giriş Ekranı.

İkinci kullanıcı sistemin tanıdığı kullanıcı kodu ve şifresi ile sisteme girmiş ancak açık anahtarını kaydettirmediği için Sipariş Giriş Ekranına ulaşması mümkün olmamıştır. Çünkü sunumcu tarafında çalışan güvenlik paketi bu kullanıcının girdiği kullanıcı kodu ve şifreyi kontrol edip sisteme girişine müsaade etmeden önce kullanıcının açık anahtarının kayıtlı olup olmadığını da kontrol etmiş ve herhangi bir kayıt bulamayınca kullanıcı tarafında çalışan güvenlik paketine mesaj yollayarak kullanıcının açık gizli anahtar ikilisini oluşturarak açık anahtarını kaydettirmesini istemiştir (Şekil 4.4).



Şekil 4.4. Uyarı Mesajı.

Test gereği kullanıcı bu uyarıyı dikkate almadan sipariş giriş ekranına geçmeye çalışmış ancak Sipariş Giriş Ekranına geçememiştir. Kullanıcılar bilgisayarları değiştirmişler ve tekrar sipariş işlemi yapmaya çalışmışlardır. Her iki kullanıcı da başarılı olamamıştır. İkinci kullanıcı zaten kendinin açık anahtarını kaydettirmediği için bir önceki denemede olduğu gibi Sipariş Giriş Ekranına dahi geçememiştir. Birinci kullanıcı ise açık anahtarını sisteme daha önce tanıttığı için Sipariş Giriş Ekranına girmiştir. Ancak sistem ilk siparişi yaparken imzayı oluşturmak için kullanıcının gizli anahtarının adresini sormuş ama kullanıcının gizli anahtarı diğer bilgisayarda kaldığından sipariş yapılamamıştır. Gizli anahtar yerine verilmek istenen bir başka dosya da ise imza gerçekleştirilemedi hatası almıştır. Kullanıcının gizli anahtarı yeni bilgisayarına kopyalandığında sipariş işleminin başarıyla yapıldığı gözlemlenmiştir.

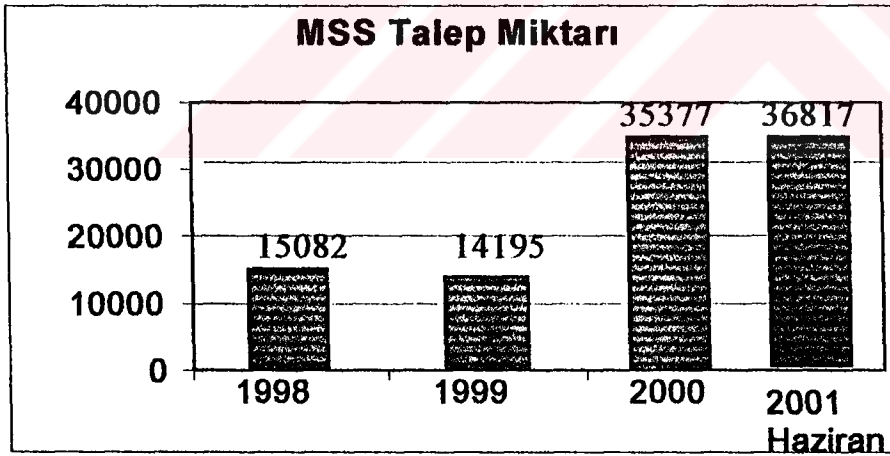
Testin başka bir aşamasında kullanıcının yaptığı sipariş miktarları bilerek değiştirilmiş ve kullanıcıya bu hatalı miktarlar bildirilmiştir. Kullanıcı hatalı miktarlara itiraz edince ana sistemdeki kayıtlı değiştirilmiş siparişten oluşturulan yeni mesaj özeti ile kullanıcının ilk gönderdiği sayısal imzadan elde edilen özetin tutmadığı görülmüş ve ancak kullanıcının belirttiği miktarlar girildiğinde bu özetlerin aynı oldu görülmüştür.



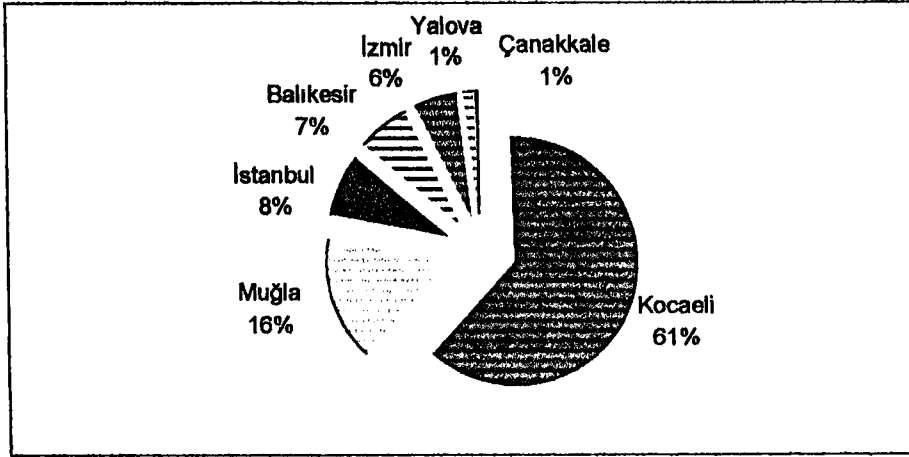
5. SONUÇ VE ÖNERİLER

Bu çalışma sonucunda merkezi bir otorite ve buna bağlı birimlerce kullanılacak Malzeme Sipariş Sistemi yazılımı ve Güvenlik Modülü hazırlanmıştır. Yazılım Java Programlama Dili ile applet tekniği kullanılarak hazırlanmış ve Java'nın WEB tabanlı uygulamalardaki esnekliği ve platform bağımsızlığı özelliklerinden faydalanılmıştır. Java'nın benzer uygulamalar için uygun bir yazılım geliştirme ortamı olduğu görülmüştür.

Malzeme Sipariş Sistemi 27 Mart 2000 tarihinde uygulamaya konmuştur. Malzeme Sipariş Sistemi (MSS)'nin uygulamaya geçişinden sonrası ve öncesi dönem içindeki sipariş miktarları ve siparişlerin dağılımı Şekil 5.1 ve Şekil 5.2'de betimlenmiştir.



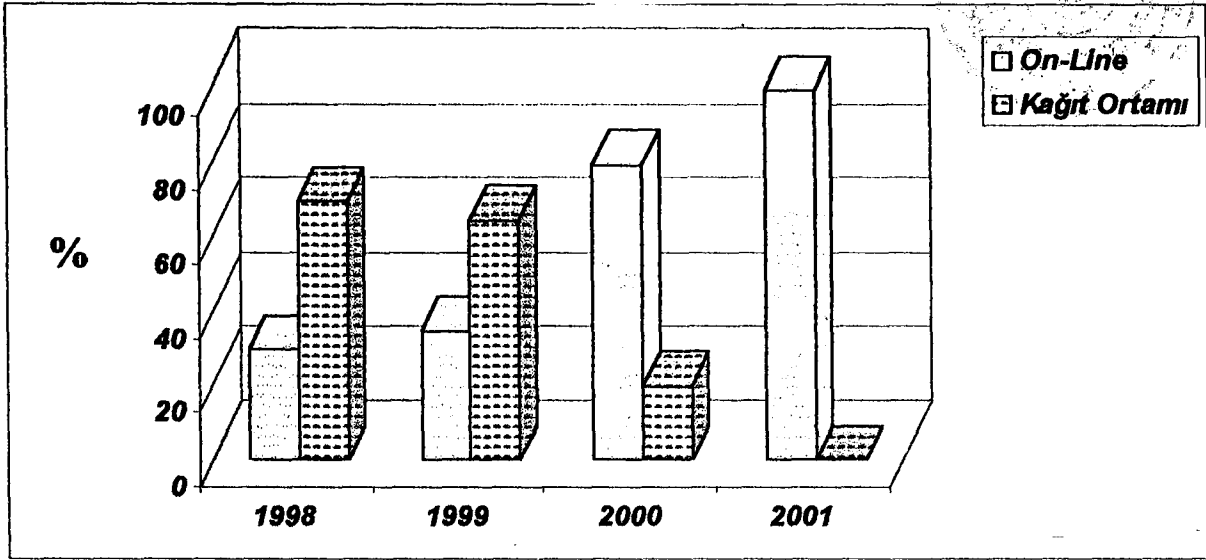
Şekil 5.1. Malzeme Sipariş Sistemi (MSS) – Yıllara Göre Sipariş Miktarı.



Şekil 5.2. MSS- Bölgelere Göre Sipariş Dağılımı.

Sistemin uygulamaya konulması ile çeşitli ara kontrollere son verilmiş ve sipariş işlemleri basite indirilmiştir. Bu sayede geçmiş yıllara göre tam iki katına varan sipariş artışları tespit edilmiştir. Böylece geniş bir sipariş istatistiği oluşturulmaya çalışılmıştır.

Sistem uygulamaya konmadan önce sadece 24 uç birim merkezi birime terminal client olarak bağlanarak, siparişleri on-line girmekte ve diğer tüm uç birimler siparişleri kağıt ortamında göndermekteydi. On-line giriş yapanlar ile kağıt ortamında siparişleri gönderen birimlerin yıllara göre oranlar Şekil 5.3.' de gösterilmiştir.



Şekil 5.3. On-line /Kağıt Ortamında Sipariş Oranları.

27.Mart.2000 tarihinden itibaren uygulamanın WEB üzerine taşınması ile birlikte 135 uç birimin siparişleri on-line işlemeye başlamıştır. Siparişlerin kağıt ortamında gelmesine son verilmiştir. Daha önceden kağıt ortamında gelen siparişlerin merkezi birimdeki görevliler tarafından bilgisayara girilmesinde meydana gelen operatör hatalarına son verilmiştir.

Ayrıca önceki sistemde gerek sipariş sahibinin fikir değiştirerek miktar ve malzemede yaptığı düzeltmeleri merkezi sisteme yansıtılmaması gerekse operatör hatalarından dolayı merkezi birimler ile uç birimler arasında çeşitli uyumsuzluklar oluşturmakta ve problem çıkmaktaydı. Artık tüm kullanıcıların siparişlerini kendileri girmesinden sonra operatör hataları ortadan kaldırılmıştır. Sayısal imza uygulaması ile de sipariş sahibinin yaptığı siparişin içeriğini reddetmesi imkanı ortadan kaldırılmıştır.

Önceki sistemde her yıl ortalama 750 – 1000 problemlili sipariş (% 5 – 8) üzerinde ilave çalışma yapmak gerekmekte idi. SIUMSIS'e geçiş ile birlikte 2000 yılında toplam siparişler 35377' e çıkmasına rağmen, sadece 200 kadar problemlili sipariş olmuştur. (~% 0,6) Bu problemlili siparişlerin hepsinin de uç kullanıcıların hatalı girişleri olduğu görülmüş ve 2001 yılında uç kullanıcıların eğitim düzeylerinin artırılması ile bu hataların da düşürüleceği sonucuna varılmıştır. Problemlili

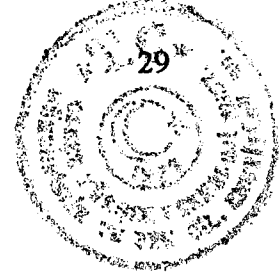
siparişlerin bazılarında uç kullanıcıların miktar ve malzemelere itirazları olmuş fakat kullanılan sayısal imza teknolojisi ile yapılan siparişlerin orijinal kayıtlı haliyle ilgili uç kullanıcıların ait olduğu gerçekleşmiştir.

Günümüzde ve gelecekte elektronik haberleşmenin insan yaşamının her alanında vazgeçilmez bir yer tutacağı açıktır. Bu nedenle ağ altyapılarının sağlamlığı ve uygulama yazılımlarının hatasız işleyişi kadar ya da daha da önemli olmak üzere haberleşmenin gizliliği ve güvenilirliği sorunu karşımıza çıkmaktadır. Gelecekte bilgiyi bir yerden başka bir yere göndermek yeter şart olmadığı bu bilgiyi tutarlı ve güvenli bir şekilde göndermek gereği daha fazla ön plana çıkacaktır.

Kişiye ve metne özel şifreli özet olarak tanımlayabileceğimiz sayısal imza gizlilik ve güvenirliliğin sağlanması konusunda çok etkili bir araçtır. Bu çalışma sonucu oluşturulan Malzeme Sipariş Sisteminde de sayısal imza teknolojisi kullanılarak, yapılan siparişlerin gerçek sahiplerinin onaylanması, sipariş bilgisinin güvenilirliğinin denetlenmesi ve yapılan siparişte olası uzlaşmazlıklarda ispat olanağına ulaşılmıştır.

Özellikle güvenlik modülünün yazılımı sırasında internet browserların (Internet Explorer, Netscape Navigator vb.) henüz aynı standartları desteklememesinden dolayı sorunlar ile karşılaşmıştır. Kullanıcı bilgisayarındaki kaynakların (disk, disket vb.) güvenlik modülünce kullanılabilmesi için applet'lerin imzalanması her browser'a göre farklılık arz etmiştir. Aynı şekilde kullanılan Java paketleri de gene her browser'a göre değişmesi gerekmiştir. Gelecekte tüm browser'ların aynı standartları desteklemesiyle harcanacak emek kaybı en aza indirilecektir.

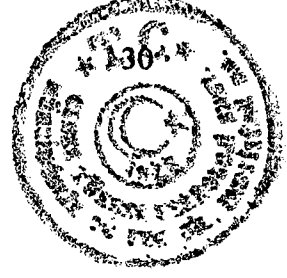
Sistemin gerçekleşmesi sırasında sayısal imzanın kullanılan bilgisayara bağlı kalmadan her bilgisayarda sorunsuz kullanılabilmesi için güvenli bir şekilde taşınması zorluğu ortaya çıkmıştır. Bu problemi aşmak için akıllı kart, biometrik veriler (klavye yazım ritmi vb.) ya da parmak izi, göz retinasından tanıma olanağı sağlayan donanımlardan yararlanılması çalışmaları varolan teknoloji, maliyet ve uygulanabilirlik ölçütleri ışığında incelenmelidir.



KAYNAKLAR

1. C. Simmons, Contemporary Cryptology: The Science of Information Integrity, IEEE Press, 1992.
2. Elektronik Ticaret Koordinasyon Kurulu Teknik Raporu-1998
3. F. Çenesiz, İ. Soğukpınar, Kurumsal Ağ Güvenliğinde Sayısal İmza Kullanımı: Tasarım ve Uygulama, 5.Bilgisayar Ağları Sempozyumu BAS2000, Bilkent Üniversitesi/ANKARA, 15-16 Haziran 2000,.
4. G. Popek and C. Kline, Encryption and Secure Computer Networks, ACM Computing Surveys, December 1979.
5. H. Dobbertin, A. Bosselaers, B. Preneel, RIPEMD-160 A Strengthened Version of RIPEMD, III. International Workshop on Fast Software Encryption, 1996, Springer- Verlag, 1996.
6. H. Feistel, Cryptography and Computer Privacy, Scientific American, May 1973.
7. L. Kohnfelder, Towards a Practical Public-Key Cryptosystem, Bachelor's Thesis, M.I.T. May 1978.
8. R. L. Rivest, A. Shamir, L. Adleman, A Method for Obtaining Digital Signatures and Public- Key Cryptosystem, Commun. ACM, vol. 21, no. 2, pp. 120-127, Feb. 1977.
9. R. Rivest, The MD4 Message Digest Algorithm, Crypto'90, Santa Barbara, CA, Aug. 11-15, 1990.
10. S. Garfinkel, G. Spafford, WEB Security & Commerce , O'Reilly & Associates Inc., 1997.
11. W. Stallings, 2 ed., Cryptography and Network Security: Principles and Practice, Prentice-Hall.Inc, 1998.
12. W.Diffie, M. Helman, New Directions in Cryptography, IEEE Trans. Inform. Theory, vol. IT-22, no. 6, pp. 644-654, Nov. 1976.

ÖZGEÇMİŞ



Levent YALÇINKAYA 1986 yılında Deniz Harp Okulu Komutanlığı Yöneylem Araştırma/Yönetim bölümünden mezun olmuştur.

Mezuniyetini müteakip sırasıyla TCG Tekirdağ K.lığı II. Çarkçılık, TCSG 71 K.lığı II. Komutanlık görevlerini yapmıştır.

1989 yılında Orta Doğu Teknik Üniversitesi Bilgisayar Mühendisliği Bölümünde açılan "OBİ Subay Temel Kursu"nu kazanmış ve 1990 yılında mezun olmuştur. Daha sonra sırasıyla Deniz Harp Akademisi K.lığı, Deniz Harp Okulu K.lığı ve Gölcük Bilgi İşlem Merkezi Komutanlığı'nda bilgisayar subaylığı görevlerinde bulunmuştur.

Halen Envanter Kontrol Merkezi K.lığı'nda Yazılım Proje Planlama ve Yönetim Şube Müdürlüğü görevini yapmaktadır.

EK-1



GÜVENLİK MODÜLÜ KAYNAK KODLARI

```
/*
 *
 *
 * Açık-Gizli anahtar ikilisini kullanıcı tarafında oluşturan
 * yazılım
 */
// Copyright (c) 2000
package DigitalSignature;

import javax.swing.*;
import java.awt.*;
import oracle.jdeveloper.layout.*;
import java.awt.event.*;

/**
 * A Swing-based dialog class.
 * <P>
 * @author
 */
public class AnahtarOlustur extends JDialog {
    JPanel jPanel1 = new JPanel();
    JTextArea jTextArea1 = new JTextArea();
    JButton devam = new JButton();
    GridBagLayout gridBagLayout1 = new GridBagLayout();
    String fileName = "User.PK";

    public void setFileName(String name) {
        fileName = name;
    }

    public String getFileName() {
        return fileName;
    }

    public AnahtarOlustur(Frame parent, String title, boolean modal) {
        super(parent, title, modal);
        try {
            jbInit();
            pack();
        }
        catch (Exception e) {
            e.printStackTrace();
        }
    }

    public AnahtarOlustur() {
        this(null, "", false);
    }

    private void jbInit() throws Exception {
        jPanel1.setLayout(gridBagLayout1);
        jTextArea1.setLineWrap(true);
        jTextArea1.setWrapStyleWord(true);
    }
}
```

```

jTextArea1.setRows(13);
jTextArea1.setBackground(Color.lightGray);
jTextArea1.setText("Malzeme Talep Sistemini kullanmak için
gerekli bir gizli ve bir açık " +
    "anahtar sizin için oluşturulacaktır. Bu anahtarlardan açık olan
Envanter " +
    "Kontrol Merkezi Komutanlığı Ana Sistemine kaydedilecektir.
Sizde " +
    "kalacak olan gizli anahtarı, istediğiniz ortamda ve lokasyonda
gizlilikle " +
    "saklayınız. Malzeme talebini herhangi bir bilgisayar üzerinden
yapabilirsiniz. " +
    "Ancak bu gizli anahtarınızı tasimanız gerektiğini unutmayınız.
Gizli " +
    "anahtarın sorumluluğu kullanıcıya aittir. Çalınması ya da
desifre " +
    "olması durumunda Envanter Kontrol Merkezi Komutanlığına evrakla
ivedi " +
    "olarak başvurmanız gerekir. Gizli ve açık anahtarların
oluşturulması " +
    "için DEVAM tusuna basarak gizli anahtarın saklanacağı lokasyonu
belirtiniz.");
jTextArea1.setEditable(false);
devam.setText("devam");
devam.setLabel("Devam");
devam.addActionListener(new java.awt.event.ActionListener() {

    public void actionPerformed(ActionEvent e) {
        devam_actionPerformed(e);
    }
});
getContentPane().add(jPanel1);
jPanel1.add(jTextArea1, new GridBagConstraints2(0, 0, 1, 1, 1.0,
1.0,
    GridBagConstraints.CENTER, GridBagConstraints.BOTH, new
Insets(31, 16, 0, 26), 0, 1));
jPanel1.add(devam, new GridBagConstraints2(0, 1, 1, 1, 0.0, 0.0,
    GridBagConstraints.CENTER, GridBagConstraints.NONE, new
Insets(29, 413, 24, 30), 0, 0));
}

void devam_actionPerformed(ActionEvent e) {
    FileDialog fd = new FileDialog((new Frame()), "Gizli Anahtar
Kaydet", FileDialog.SAVE);
    fd.setFile(fileName);
    fd.show();
    if (fd.getFile() == null)
        setFileName(null);
    else setFileName(fd.getDirectory()+"\\"+fd.getFile());
    dispose();
}
}

```



```

        String message
getAuthentication(kullanici,sifre);
        Arac.writeBuffer(out,message);
    }
    else if (islem.equals("PK---")) {
        byte[] publicKey = Arac.readBuffer(in);
        setPK(kullanici,publicKey);
    }
    else if (islem.equals("TALEP")) {
        String Cumle = s.substring(5,38);
        System.out.println(Cumle);
        imza = Arac.readBuffer(in);
        boolean OK
guvenlik.verifySignature(Cumle.getBytes(),imza);
        System.out.println(OK);
        if (OK)
            Arac.writeBuffer(out,"OK----");
        else Arac.writeBuffer(out,"ERROR-");
    }
}
catch (Exception e) {
    done = true;
    System.out.println(e);
}
}
in.close();
out.close();
incoming.close();
System.out.println(socketNumber+
    " : LOGGED OUT at "+(new
java.util.Date()).toString());
}
catch (Exception e) {
    System.out.println(socketNumber+
    " : STREAM/DB ACCESS ERROR at "+(new
java.util.Date()).toString());
}
}

public String getAuthentication(String kullanici, String sifre) {
    try {
        Statement st = connection.createStatement();
        ResultSet rs = st.executeQuery("Select public_key
from kullanici "+
        "where kullanici kodu = '"+kullanici+
        "' and sifre = '"+sifre+"'");
        if (rs.next()) {
            oracle.sql.BLOB pk
(oracle.sql.BLOB)rs.getBlob(1);
            rs.close();
            st.close();
            if (pk == null) {
                return "NEEDPK";
            }
            else {
                convertToPK(pk);
                return "AUTHED";
            }
        }
        else {
            rs.close();

```

```

        st.close();
        return "NOAUTH";
    }
}
catch (Exception e) {
    System.out.println("Select public_key from kullanici
"+
        "where kullanici_kodu = '"+kullanici+
        "' and sifre = '"+sifre+"'");
    return "ERROR-"+e.toString();
}

}

private void convertToPK(oracle.sql.BLOB pk) throws Exception {
    int length = (int) pk.length();
    System.out.println("Uzunluk : "+length);
    byte[] buffer = new byte[length];
    buffer = pk.getBytes(1,length);
    guvenlik.setPublicKey(buffer);
}

private void setPK(String kullanici, byte[] publicKey) throws
Exception {
    guvenlik.setPublicKey(publicKey);
    Statement st = connection.createStatement();
    st.executeUpdate("update kullanici set public_key = '0' "+
        "where kullanici_kodu = '"+kullanici+"'");
    ResultSet rs = st.executeQuery("Select public_key from
kullanici "+
        "where kullanici_kodu = '"+kullanici+"'");
    rs.next();
    oracle.sql.BLOB pk = (oracle.sql.BLOB)rs.getBlob(1);
    OutputStream os = pk.getBinaryOutputStream();
    os.write(publicKey);
    os.close();
    connection.commit();
}

}

package DigitalSignature;

import java.net.*;
import java.sql.*;

public class ServerProg extends Object {

    public static void main(String arg[]) throws Exception {
        int i=1;
        ServerSocket s = new ServerSocket(1978);
        for (;;) {
            Socket incoming = s.accept();
            System.out.println(i+
                " : LOGGED ON at "+(new
java.util.Date()).toString());
            new ThreadedServerProg(incoming,i).start();
            ++i;
        }
    }
}

```

```

/*
 *
 *
 * Güvenlik Modülü kütüphanesi
 *
 */
package DigitalSignature;

import java.security.*;
import java.security.spec.*;

public class Guvenlik {
    private PrivateKey privateKey = null;
    private PublicKey publicKey = null;

    public void setPublicKey( byte[] buffer) throws Exception {
        X509EncodedKeySpec      publicKeySpec      =      new
X509EncodedKeySpec(buffer);
        KeyFactory keyFactory = KeyFactory.getInstance("DSA","SUN");
        publicKey = keyFactory.generatePublic(publicKeySpec);
    }

    public void setPrivateKey( byte[] buffer) throws Exception {
        PKCS8EncodedKeySpec      privateKeySpec      =      new
PKCS8EncodedKeySpec(buffer);
        KeyFactory keyFactory = KeyFactory.getInstance("DSA","SUN");
        privateKey = keyFactory.generatePrivate(privateKeySpec);
    }

    public byte[] getPrivateKeyByteStream() throws Exception {
        return privateKey.getEncoded();
    }

    public byte[] getPublicKeyByteStream() throws Exception {
        return publicKey.getEncoded();
    }

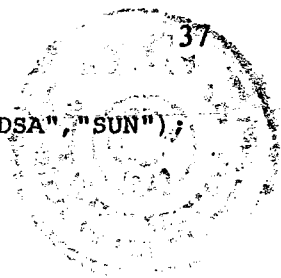
    public void generateKeyPair() throws Exception {
        KeyPairGenerator      keyGenerator      =
KeyPairGenerator.getInstance("DSA","SUN");
        SecureRandom      random      =
SecureRandom.getInstance("SHA1PRNG","SUN");
        keyGenerator.initialize(1024,random);
        KeyPair pair = keyGenerator.generateKeyPair();
        privateKey = pair.getPrivate();
        publicKey = pair.getPublic();
    }

    public byte[] getSignature( byte[] buffer) throws Exception {
        Signature dsa = Signature.getInstance("SHA1withDSA","SUN");
        dsa.initSign(privateKey);
        dsa.update(buffer);
        return dsa.sign();
    }

    public boolean verifySignature( byte[] buffer, byte[] signature)
throws Exception {

```

```
Signature dsa = Signature.getInstance("SHA1withDSA", "SUN");  
dsa.initVerify(publicKey);  
dsa.update(buffer);  
return dsa.verify(signature);  
}  
}
```





```
/*
 *
 *
 *   Güvenlik modülünün testi için geliştirilen kullanıcı giriş
yazılımı
 *
 */
// Copyright (c) 2000
package DigitalSignature;

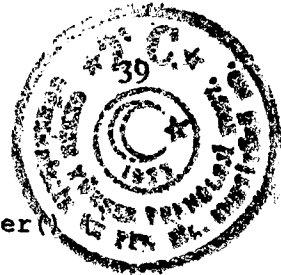
import javax.swing.*.*;
import java.awt.*.*;
import oracle.jdeveloper.layout.*.*;
import java.awt.event.*.*;

/**
 * A Swing-based dialog class.
 * <P>
 * @author
 */
public class KullaniciGiris extends JDialog {
    JPanel jPanel1 = new JPanel();
    JLabel jLabel1 = new JLabel();
    JLabel jLabel2 = new JLabel();
    JLabel jLabel3 = new JLabel();
    JTextField kullaniciKodu = new JTextField();
    JPasswordField kullaniciSifresi = new JPasswordField();
    JButton tamam = new JButton();
    JButton iptal = new JButton();
    GridBagLayout gridBagLayout1 = new GridBagLayout();
    String kullanici = null;
    String sifre = null;
    public static int OK = 1;
    public static int CANCELED = 0;
    int state = CANCELED;

    public KullaniciGiris(Frame parent, String title, boolean modal) {
        super(parent, title, modal);
        try {
            jbInit();
            pack();
        }
        catch (Exception e) {
            e.printStackTrace();
        }
    }

    public KullaniciGiris() {
        this(null, "", false);
    }

    private void jbInit() throws Exception {
        jPanel1.setLayout(gridBagLayout1);
        jLabel1.setText("Malzeme Talep Sistemi");
        jLabel1.setFont(new Font("Dialog", 0, 16));
        jLabel2.setText("Kullanici Kodu");
        jLabel3.setText("Kullanici Sifresi");
        kullaniciKodu.setColumns(15);
        kullaniciSifresi.setColumns(15);
    }
}
```



```
tamam.setText("tamam");
tamam.setLabel("Tamam");
tamam.addActionListener(new java.awt.event.ActionListener() {

    public void actionPerformed(ActionEvent e) {
        tamam_actionPerformed(e);
    }
});
iptal.setText("iptal");
iptal.setLabel(" Iptal ");
iptal.addActionListener(new java.awt.event.ActionListener() {

    public void actionPerformed(ActionEvent e) {
        iptal_actionPerformed(e);
    }
});
getContentPane().add(jPanell1);
jPanell1.add(jLabel1, new GridBagConstraints2(0, 0, 3, 1, 0.0,
0.0,
    GridBagConstraints.WEST, GridBagConstraints.NONE, new
Insets(27, 111, 0, 0), 0, 0));
jPanell1.add(jLabel2, new GridBagConstraints2(0, 1, 1, 1, 0.0,
0.0,
    GridBagConstraints.WEST, GridBagConstraints.NONE, new
Insets(59, 80, 0, 0), 0, 0));
jPanell1.add(jLabel3, new GridBagConstraints2(0, 2, 1, 1, 0.0,
0.0,
    GridBagConstraints.WEST, GridBagConstraints.NONE, new
Insets(38, 80, 0, 0), 0, 0));
jPanell1.add(kullaniciKodu, new GridBagConstraints2(1, 1, 4, 1,
1.0, 0.0,
    GridBagConstraints.WEST, GridBagConstraints.HORIZONTAL,
new Insets(57, 19, 0, 54), 0, 0));
jPanell1.add(kullaniciSifresi, new GridBagConstraints2(1, 2, 4,
1, 1.0, 0.0,
    GridBagConstraints.WEST, GridBagConstraints.HORIZONTAL,
new Insets(36, 19, 0, 54), 0, 0));
jPanell1.add(tamam, new GridBagConstraints2(1, 3, 1, 1, 0.0, 0.0,
    GridBagConstraints.CENTER, GridBagConstraints.NONE, new
Insets(66, 19, 27, 0), 0, 0));
jPanell1.add(iptal, new GridBagConstraints2(3, 3, 1, 1, 0.0, 0.0,
    GridBagConstraints.CENTER, GridBagConstraints.NONE, new
Insets(66, 0, 27, 0), 0, 0));
}

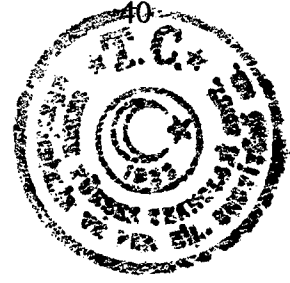
void iptal_actionPerformed(ActionEvent e) {
    setState(CANCELED);
    dispose();
}

void tamam_actionPerformed(ActionEvent e) {
    setState(OK);
    dispose();
}

public String getKullanici() {
    return kullanici;
}

public String getSifre() {
    return sifre;
}
```

```
}  
  
private void setState(int state) {  
    this.state = state;  
    if (state == CANCELED) {  
        kullanıcı = null;  
        sifre = null;  
    }  
    else {  
        kullanıcı = kullanıcıKodu.getText();  
        sifre = kullanıcıSifresi.getText();  
    }  
}  
  
public int getState() {  
    return state;  
}  
}
```



```
/*
 *
 *
 * Güvenlik modülünün testi için geliştirilen sipariş giriş yazılımı
 *
 */
```

```
// Copyright (c) 2000
package DigitalSignature;
```

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.net.*;
import java.io.*;
import oracle.jdeveloper.layout.*;
```

```
public class Talep extends JApplet {
    Guvenlik guvenlik = new Guvenlik();
    boolean ilktalep = true;
    boolean isStandalone = false;
    JLabel jLabel1 = new JLabel();
    JTextField stokNo = new JTextField();
    JTextField miktar = new JTextField();
    JLabel jLabel3 = new JLabel();
    JLabel jLabel4 = new JLabel();
    JLabel jLabel2 = new JLabel();
    JButton Kaydet = new JButton();
    JButton Vazgec = new JButton();
    JComboBox dagitimBirimi = new JComboBox();
    XYLayout xYLayout1 = new XYLayout();
    JLabel jLabel5 = new JLabel();

    Socket s = null;
    private DataInputStream input;
    private DataOutputStream output;
    private byte inputBuffer[];
    private byte outputBuffer[];
    private final static int BUFFER_LENGTH = 1096;
```

```
public String getParameter(String key, String def) {
    if (isStandalone) {
        return System.getProperty(key, def);
    }
    if (getParameter(key) != null) {
        return getParameter(key);
    }
    return def;
}
```

```
public Talep() {
}
```

```
public void init() {
    try {
        s = new Socket(InetAddress.getByName("vtysbmd_laptop"), 1978);
        input = new DataInputStream(s.getInputStream());
        output = new DataOutputStream(s.getOutputStream());
        String cevap = null;
```

42

```

String kullanıcı=null;

KullaniciGiris prompt = new KullaniciGiris(null,"Kullanıcı
Kodunu ve Sifresini Giriniz",true);
int devam = 0;
while (devam == 0) {
    prompt.show();
    if (prompt.getState()== KullaniciGiris.OK) {
        kullanıcı = prompt.getKullanici();
        if (kullanıcı.length() > 15) kullanıcı =
kullanıcı.substring(0,15);
        else for (int k=kullanıcı.length(); k < 15; ++k) kullanıcı
+= " ";
        String sifre = prompt.getSifre();
        if (sifre.length() > 15) sifre = sifre.substring(0,15);
        else for (int k=sifre.length(); k < 15; ++k) sifre += " ";

        String message = "USER-"+kullanıcı+sifre;
        Arac.writeBuffer(output,message);
        cevap = new String(Arac.readBuffer(input));
        System.out.println(cevap);
        if (cevap.equals("NOAUTH"))
            devam = 0;
        else devam = 2;
        System.out.println(devam);
    }
    else devam = 1;
}
if (devam == 2) {
    if (cevap.equals("NEEDPK")) {
        AnahtarOlustur anahtarDialog = new
AnahtarOlustur(null,"Gizli ve Açık Anahtar Yaratma",true);
        anahtarDialog.setFileName(kullanıcı.trim()+".PK");
        anahtarDialog.show();
        String fileName =anahtarDialog.getFileName();
        System.out.println(fileName);
        if (fileName == null) devam = 1;
        else {
            guvenlik.generateKeyPair();
            FileOutputStream outputFile = new
FileOutputStream(fileName);
            outputFile.write(guvenlik.getPrivateKeyByteStream());
            outputFile.close();
            Arac.writeBuffer(output,"PK---");

            Arac.writeBuffer(output,guvenlik.getPublicKeyByteStream());
            ilktalep = false;
        }
    }
    if (devam == 2)
        jbInit();
}
catch (Exception e) {
    e.printStackTrace();
}

public void destroy() {
    try {
        Arac.writeBuffer(output,"KAPAT");
    }
}

```



```
        input.close();
        output.close();
        s.close();
    }
    catch (Exception e) {
        System.out.println(e);
    }
}

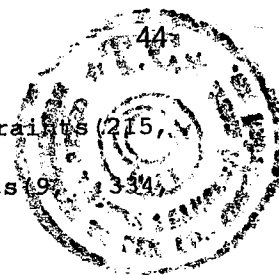
private void jbInit() throws Exception {
    this.getContentPane().setLayout(xYLayout1);
    this.setSize(new Dimension(336, 400));
    jLabel1.setText("Stok Numarası");
    stokNo.setColumns(13);
    miktar.setColumns(5);
    jLabel3.setText("Dagitim Birimi");
    jLabel4.setText("Talep Edeceginiz Malzemeyi Giriniz :");
    jLabel2.setText("Talep Miktari");
    Kaydet.setText("Kaydet");
    Kaydet.setLabel("Kaydet");
    Kaydet.addActionListener(new java.awt.event.ActionListener() {

        public void actionPerformed(ActionEvent e) {
            Kaydet_actionPerformed(e);
        }
    });
    Vazgec.setText("Vazgec");
    Vazgec.setLabel("Vazgec");
    Vazgec.addActionListener(new java.awt.event.ActionListener() {

        public void actionPerformed(ActionEvent e) {
            Vazgec_actionPerformed(e);
        }
    });
    xYLayout1.setHeight(370);
    jLabel5.setText("...");
    dagitimBirimi.addItem("Adet");
    dagitimBirimi.addItem("Kilo");
    dagitimBirimi.addItem("Paket");
    dagitimBirimi.addItem("Düzine");
    dagitimBirimi.addItem("Litre");
    dagitimBirimi.addItem("Metre");
    dagitimBirimi.addItem("Metre Küp");

    xYLayout1.setWidth(393);
    this.getContentPane().add(jLabel1, new XYConstraints(19, 125, 263, -1));
    this.getContentPane().add(stokNo, new XYConstraints(19, 148, 138, -1));
    this.getContentPane().add(miktar, new XYConstraints(19, 219, 142, -1));
    this.getContentPane().add(jLabel3, new XYConstraints(215, 195, -1, -1));
    this.getContentPane().add(jLabel4, new XYConstraints(19, 87, 294, -1));
    this.getContentPane().add(jLabel2, new XYConstraints(19, 195, -1, -1));
    this.getContentPane().add(Kaydet, new XYConstraints(94, 274, -1, -1));
    this.getContentPane().add(Vazgec, new XYConstraints(233, 274, -1, -1));
```

44



```

    this.getContentPane().add(dagitimBirimi, new XYConstraints(215,
219, 121, -1));
    this.getContentPane().add(jLabel15, new XYConstraints(97, 334,
381, -1));
}

public String getAppletInfo() {
    return "Applet Information";
}

public String[][] getParameterInfo() {
    return null;
}

public static void main(String[] args) {
    Talep applet = new Talep();
    applet.isStandalone = true;
    JFrame frame = new JFrame();
    frame.setTitle("Applet Frame");
    frame.getContentPane().add(applet, BorderLayout.CENTER);
    applet.init();
    applet.start();
    frame.setSize(400, 420);
    Dimension d = Toolkit.getDefaultToolkit().getScreenSize();
    frame.setLocation((d.width - frame.getSize().width) / 2,
(d.height - frame.getSize().height) / 2);
    frame.setVisible(true);
    frame.addWindowListener(new WindowAdapter() { public void
windowClosing(WindowEvent e) { System.exit(0); } });
}

static {
    try {
        UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());
    }
    catch(Exception e) {
        e.printStackTrace();
    }
}

void Vazgec_actionPerformed(ActionEvent e) {
    stokNo.setText("");
    miktar.setText("");
    dagitimBirimi.setSelectedIndex(0);
}

void Kaydet_actionPerformed(ActionEvent e) {
    String tempStokNo = stokNo.getText();
    if (tempStokNo == null) {
    }
    else if (tempStokNo.length() > 13)
        tempStokNo = tempStokNo.substring(0,13);
    else for (int k=tempStokNo.length(); k < 13; ++k)
        tempStokNo += " ";
    String tempMiktar = miktar.getText();
    if (tempMiktar == null) {
    }
    else if (tempMiktar.length() > 10)
        tempMiktar = tempMiktar.substring(0,10);
    else for (int k=tempMiktar.length(); k < 10; ++k)

```



```
tempMiktar += " ";
String tempDagitimBirimi
dagitimBirimi.getSelectedItem().toString();
for (int k=tempDagitimBirimi.length(); k < 10; ++k)
    tempDagitimBirimi += " ";
String mesaj = tempStokNo+tempMiktar+tempDagitimBirimi;
if (ilktalep) {
    FileDialog fd = new FileDialog((new Frame()), "Gizli Anahtar
Oku", FileDialog.LOAD);
    fd.setFile("");
    fd.show();
    if (fd.getFile() == null) {
        //////////////////////////////////hata üret //////////////////////////////////
    }
    else { try {
        FileInputStream fileInput = new
FileInputStream(fd.getDirectory()+"\\"+fd.getFile());
        byte[] privateKey = new byte[fileInput.available()];
        System.out.println(privateKey.length);
        fileInput.read(privateKey);
        fileInput.close();
        guvenlik.setPrivateKey(privateKey);
        ilktalep = false;
    } catch (Exception ex) { System.out.println("1:  "+ex);}
    }
    try {
        byte[] imza = guvenlik.getSignature(mesaj.getBytes());
        Arac.writeBuffer(output, "TALEP"+mesaj);
        Arac.writeBuffer(output, imza);
        String cevap = new String(Arac.readBuffer(input));
        System.out.println(cevap);
    } catch (Exception ex) { System.out.println("2 :  "+ex); }
}
}
```