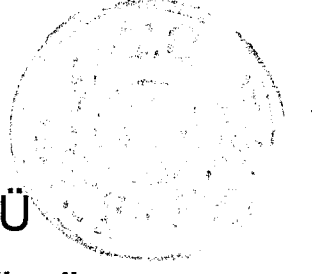


T.C.

GEBZE YÜKSEK TEKNOLOJİ ENSTİTÜSÜ
MÜHENDİSLİK VE FEN BİLİMLERİ ENSTİTÜSÜ



ELEKTRONİK VERİ DEĞİŞİMİNDE (EVD)
GÜVENLİK ÖNLEMLERİ

121 333

Kemal Akgün TAŞKIRAN
YÜKSEK LİSANS TEZİ
BİLGİSAYAR MÜHENDİSLİĞİ
ANABİLİM DALI

TEZ DANIŞMANI

Yrd.Doç. Dr. İbrahim SOĞUKPINAR

**TC. YÜKSEKÖĞRETİM KURULU
DOKÜMANTASYON MERKEZİ**

GEBZE

2002

121333

Kemal Akgün TAŞKIRAN'ın tez çalışması, G.Y.T.E. Mühendislik ve Fen Bilimleri Enstitüsü Yönetim Kurulu'nun ..26..07..2002..... tarih ve ..2002/125..... sayılı kararıyla oluşturulan jüri tarafından Bilgisayar Mühendisliği Anabilim Dalında YÜKSEK LİSANS tezi olarak kabul edilmiştir.

JÜRİ

ÜYE : Y. Doç. Dr. İbrahim SOĞUKPINAR
(Tez Danışmanı)



ÜYE : Y. Doç. Dr. Mehmet GÖKTÜRK



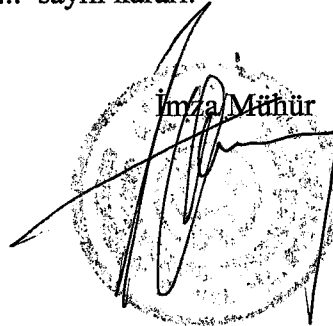
ÜYE : Y. Doç. Dr. İlyas KANDEMİR



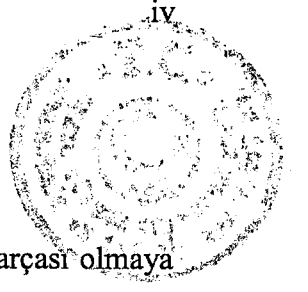
ONAY

G.Y.T.E. Mühendislik ve Fen Bilimleri Enstitüsü Yönetim Kurulu'nun ..03..09..2002..
Tarih ve ..2002/128..... sayılı kararı.

İmza/Mühür



ÖZET



“Elektronik ticaret” son yıllarda iş hayatının vazgeçilmez bir parçası olmaya başlamıştır. Elektronik Veri Değişimi EVD (EDI-Electronic Data Interchange) elektronik ticaretin en önemli parçalarından biri olup, elektronik ortamda ticari işlemlerin yapılmasını sağlayan bir mesajlaşma sistemidir. EVD elektronik olarak belli bir formatı olan iş dokümanlarının, bir firmayla ortağı olan diğer bir firma arasında değiştirilmesidir. Örneğin satın alma siparişi gibi. EVD ’nin temel aldığı fikir, asıl doküman bir kere girilmesi ve bu dokümanın sonraki gelen ilişkili yerlere sırayla elektronik ortamda gitmesidir. EVD ilk kez Kuzey Amerika’da 1970’lerde ortaya çıktı. İlk olarak taşımacılık sektöründe uygulandı. Bunu gıda, ilaç dağıtımı, otomobil, banka ve finans sektörleri izledi. İşlerin elektronik ortamda yapılması elle yapılmasına göre daha hızlı, daha etkili ve daha doğru olarak gelişti. Böylece verimlilik arttı, siparişler daha hızlı yapıldı ve de bunlar rekabette avantaj sağladılar.[5]

EVD bu amaçlar için kullanılırken karşımıza bir çok problem çıkmaktadır. Farklı gerçeklemeler büyük bir problemdir, mesajların tanımlanması ayrı ve de güvenlik te başka bir önemli problemdir EVD ‘yi kullanmak için bunu kullanacak olan endüstrinin EVD ‘nin mesaj formatlarının standartları hakkında uzlaşması gerekmektedir. Kuzey Amerika’da kabul edilen standart ‘ANSI X12’ dir. Bununla birlikte uluslararası iş çevrelerinde kabul edilen standart ‘UN/EDIFACT ‘ tır. Bu standartlaşmadan sonra önümüze çok büyük bir problem daha çıkar, o da güvenlidir.

EVD gerçekleştirilirken karşımıza çıkan güvenlik problemleri şunlardır:

- Bütünlük

Mesajın içeriğinin değişip değişmediğini anlama

- Onaylama

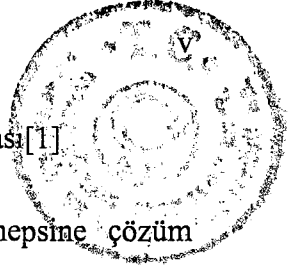
Mesajın sahibini anlama, bunun için bazı mekanizmalar kullanılır örneğin mesaj onaylama kodu (MAC)...

- Reddedemezlik

Mesajın sahibinin delili, böylece mesaj sahibi mesajını reddedemez

- Gizlilik

Mesajın ortaklar arasında deęiştirilirken, içerięinin dıřa karřı korunması[1]



Bu alıřmada, yukarıda geen gvenlik problemlerinin hepsine czm bulacak bir yntem geliřtirilmiřtir. Bu yntemde, EVD bir nesne olarak dřnlmřtr ve nesnenin gvenlięi saęlanmıřtır. Teoride, nesne oluřturulduktan sonra gvenli bir iletiřim kanalından dięer tarafa ulařtırılması gerekiyordu, bunun iin gvenli kanal řifreli haberleřmeli, verinin btnlęn korumalı, onay mekanizmasını saęlamalı ve de kullanacaęı yntemlerle mesaj sahibini belirlemeli ki daha sonrası iin delil teřkil edebilsin. Mesajları tek tek řifrelemektense, o soketten geen her veriyi řifreleyip iletiřimi gvenli kılmak ve mesajları bu soketten gndermek, o hattın gvenlięini daha ok arttırmakta ve de atak ihtimallerini azaltmaktadır. Ayrıca nesneye ynelik olması koda esneklik getirir ve de gereklemeyi kolaylařtırır.

Pratikte, bu yntemi gereklemenin en iyi yolu olarak, nesneye ynelik yapısı ve Gvenli Soket Katmanı GSK (SSL-Secure Socket Layer) zellięi nedeniyle, Java ve Java'da yazılmıř Gvenli Soket Katmanı GSK (SSL-Secure Socket Layer) seilmiřtir. GSK opsiyonlu olarak her iki tarafa da dięer tarafın kimlięini onaylama mekanizması saęlar. Tarafların birbirini onaylamasından sonra, GSK gvenli iletiřim iin, řifreli haberleřmeyi saęlar. Taraflar arasında řifreli haberleřme gizlilięi ilk planda saęlamakla beraber, ikinci olarak adreslemeyi yapar. Ayrıca řifreleme algoritması zetleme fonksiyonu kullanır, bu da verilerin iletiřim esnasında deęiřip deęiřmedięini kontrol iin kullanılır. Bu gvenlik zet fonksiyonu veri btnlęn saęlar. GSK kullanmakla onaylama, gizlilik ve verilerin btnlę saęlanmıřtır, fakat bununla reddedemezlik servisi saęlanmaz. GSK tek bařına reddedemezlięi saęlayamaz, bundan dolayı Java'nın sayısal imza mekanizması kullanılarak bu zellik saęlanmıřtır.

SUMMARY



E-business has gained very important role in the business. EDI (electronic data interchange) is the one of the important part of e-business, and it is a protocol that is used in e-business message exchange. EDI is defined as the communication of standard business transaction in a standard format between companies, for example purchase order. The basic idea in the EDI is that in the beginning document is entered than all issues done in electronically without human intervention. First of all EDI was used in North America at 1970's. The first area that was used is transportation, and then it was used other areas, like food, medicine, car industry, bank and finance.

When we compare the works done with EDI with done with manual, efficiency, correctness and time was improved, therefore productivity developed, works were done faster and by the help of this in competition the firms that use EDI have gained advantages. [5] When EDI is used for this aim there are many difficulties are happened. Firstly, different implementations are a big problem, and message definitions are other and security is the one of the major problem. While using EDI, sides must agree on the message standards. In North America, the standard is 'ANSI X12', and in Europe is 'UN/EDIFACT'.

While implementing EDI, relating with security some problems we must struggle. These are:

- Integrity

Understanding of message context was changed or not.

- Authentication

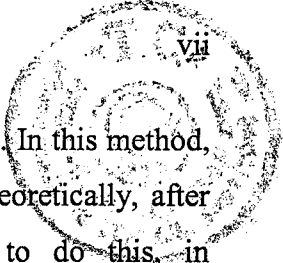
Understanding of message owner, some mechanism is used for this like message authentication code (MAC)...

- non-repudiation

Evidence of the message owner, by the way message owner cannot deny the sending messages.

- confidentiality

While changing the message between partners, securely protect message context. [1]



In this paper, a method developed for solving these problems. In this method, EDI was thought as an object, and object security is provided. Theoretically, after object constituted, it must securely send to other sides, so to do this, in communication channel messages must be encrypted, integrity and authentication must be provided, and owner of message must be detected. Instead of encrypt messages one by one, encrypt the all data in the socket, and sending the messages in this socket is more secure, and it is decreased the attack possibilities. In addition to this, object oriented code provide flexibility and easy implementations.

Practically, in the implementation it is seen that the best way is using Java and SSL written in Java, because their security mechanism provides them, and language is object oriented. SSL provides authentication optionally both sides. After authentication, SSL provides secure connection within cipher mode. Cipher connection firstly provides security, but also it is used for addressing. Adding to this, the algorithm which is used in crypto, use hash function, therefore it is understood that data is changed or not. This function provides the integrity, but non-repudiation is not provided, so digital signature mechanism of java is used for this.



TEŞEKKÜR

Tez için yaptığım araştırmalarımda desteğini hiçbir zaman esirgemeyen danışmanım Yrd. Doç. Dr. İbrahim SOĞUKPINAR' a çok teşekkür ederim.



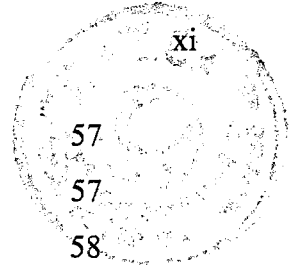
İÇİNDEKİLER DİZİNİ



Sayfa

ÖZET	IV
SUMMARY	VI
TEŞEKKÜR	VIII
İÇİNDEKİLER DİZİNİ	IX
SİMGELER VE KISALTMALAR DİZİNİ	XII
ŞEKİLLER DİZİNİ	XIII
TABLolar DİZİNİ	XIV
1. GİRİŞ	1
1.1. EVD Nedir?	1
1.2. EVD Ne Değildir?	2
1.3. EVD Nasıl Çalışır?	2
1.4. EVD' de Haberleşme	2
2. GÜVENLİK PROBLEMLERİ VE EVD'DE GÜVENLİK	7
2.1 EVD' deki İnsan Merkezli Mimari ve Güven	8
2.2 Güvenin Değişimleri ve Bunların Gelişim Evreleri	8
2.2.1 Güven Evreleri (1,2,3)	8
2.2.2 Pilot Çalışma	9
2.3 EVD' de Güvenin Önemi ve Faydaları	11
2.4 EVD' de Güvenlik İşleri	11
2.5 EVD Mesajlarının Taşınmasındaki Tehlikeler	15
2.6 EVD Sisteminin Diğer İş Sistemleriyle İlişkilerindeki Güvenliği	15
2.6.1 Gidecek Mesajın Güvenli Oluşturulması	15
2.6.2 Gelen Mesajın Güvenli İşlenmesi	16
2.6.3 Güvenliğin Yönetimi	20
2.7 EVD Kontratlarının Çeşitleri ve Güvenlik Riskleri	21
2.8 EVD Güvenliği ile İlgili Yapılan Çalışmalar	25
2.8.1 EDIFACT Güvenliği	25
2.8.2 EDI5	27
2.8.3 CRYPTIT-EDI5	28

2.8.4 ETEBAC5	28
2.8.5 X400 Mesajlaşma Sistemi	29
2.8.6 X435 EVD Mesajlaşma Sistemi	29
2.8.7 X500 Dosyalama Sistemi	29
3. UYGULAMANIN KURAMSAL YAPISI	30
3.1 EVD ve Problemler	30
3.2 Kuramsal Yapı	30
3.2.1 Uygulamanın Genel Özellikleri	30
3.2.2 Uygulamanın Güvenlik Özellikleri	31
3.2.3 Güvenliğin Uygulanışı	36
3.3 Kullanılan Araçlar ve Sebepleri	37
3.3.1 Java	37
3.3.2 Java Platform Güvenliği	37
3.3.3 Java Güvenlik Ekleri	38
3.3.4 GSK	38
4. UYGULAMANIN TEKNİK ÖZELLİKLERİ VE DİĞER UYGULAMALARLA KARŞILAŞTIRILMASI	42
4.1 Yazılım Araçları	42
4.1.1 Oracle 8i	42
4.1.2 Java ve JBuilder6	42
4.2 Uygulamanın Güvenlik Özellikleri	42
4.2.1 GSK	43
4.2.2 Sayısal İmzalama ve Onaylama	45
4.3 Uygulamanın Genel Özellikleri	47
4.3.1 Sunucu ve İstemci	47
4.3.2 EVD Nesnesi	48
4.3.3 Veritabanı	48
4.4 Örnek Uygulama	49
4.4.1 Sunucu	49
4.4.2 İstemci	53
4.5 Diğer Sistemler ile Karşılaştırılması	55
4.6 Güvenlik Analizi	56
4.6.1 El Sıkışma Protokolü	56
4.6.2 Onaylama ve Anahtar Değişimi	57



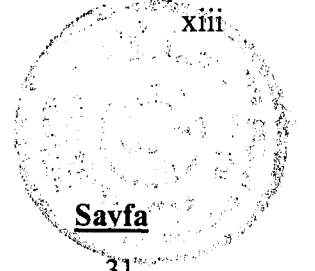
4.6.3 Anonim Anahtar Değişimi	57
4.6.4 RSA Anahtar Değişimi ve Onaylaması	57
4.6.5 El Sıkışma Protokolüne Yapılan Atakların Fark Edilmesi	58
4.6.6 Yeniden Oturum Kurmak	58
4.6.7 Uygulamanın Verilerinin Korunması	58
4.6.8 Sonuç	58
5. SONUÇ VE ÖNERİLER	60
KAYNAKLAR	61
ÖZGEÇMİŞ	63
EKLER	64
EK 1-EDI Standartlarının Formatları	
1.1-ANSI X12	
1.2-UN/EDIFACT	
EK 2- Program Kodları	
2.1-Client	
2.2 EDI	
2.3 GUI	
2.4 SECURITY	
2.5 SERVER	

SİMGELER VE KISALTMALAR DİZİNİ



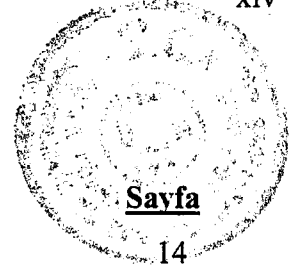
ANSI	: American National Standard Institute
AUTACK	: Authentication Acknowledge
DES	: Data Encryption Standart
EDI	: Electronic Data Interchange
EFT	: Electronic Found Transfer
EVD	: Elektronik Veri Değişimi
FTP	: File Transfer Protocol
GSK	: Güvenli Soket Katmanı
HTTP	: Hyper Text Transfer Protocol
IETF	: Engineering Task Force
JAAS	: Java Authentication and Authorization Service
JCE	: Java Cryptography Extension
JSSE	: Java Secure Socket Extension
JVM	: Java Virtual Machine
KEYMAN	: Key Management
MAC	: Message Authentication Code
RSA	: Rivest Shamir Adleman
SSL	: Secure Sockets Layer
TCP/IP	: Transport Control Protocol / Internet Protocol
TLS	: Transport Layer Security
UN/EDIFACT	: United Nation/Electronic Data Interchange for Administration, Commerce and Transport
VAN	: Value Added Network

ŞEKİLLER DİZİNİ



<u>Sekil</u>	<u>Savfa</u>
3.1 Güvenli EVD Mesajı	31
3.2 DSA Algoritması ile İmza ve Onay Mekanizması	35
3.3 GSK Mesajları Yukarıda ki Sırada Gönderilir	40
4.1 Konfigurasyon Ayarları	43
4.2 GSK Konteksti	44
4.3 Sunucu Oluşturulması	45
4.4 İstemci Oluşturma	45
4.5 İmzalama Fonksiyonları	46
4.6 İmzalı Obje	46
4.7 Objeyi Onaylama	46
4.8 Veri Gönderme	47
4.9 Veri Alma	47
4.10 İstemci Servisi Oluşturma	48
4.11 Sunucu Mönüsü	50
4.12 Sorgulama	51
4.13 Bağlı İstemciler	52
4.14 Alım İşlemleri	53
4.15 İstemci Başlatma Mönüsü	54
4.16 İstemci İşlemleri	55

TABLÖLAR DİZİNİ



Tablo

2.1 EVD ile İlgili Riskler	14
2.2 EVD Ticari Anlaşmalarının Çeşitleri	23
2.3 Olan EVD Anlaşmalarının Risk Seviyeleri	24
2.4 Etaback5	28
4.1 EVD Sistemlerinin Karşılaştırılması	56



1. GİRİŞ

Ticari firmalar ortaklarına ulaşmak istedikçe, İnternet üzerindeki ticari aktiviteler hızla büyümektedir. EVD elektronik ortamda ticari işlemlerin yapılmasını sağlayan bir mesajlaşma sistemidir.

1.1 EVD Nedir?

EVD elektronik olarak belli bir formatı olan iş dokümanlarının, bir firmayla ortağı olan diğer bir firma arasında değiştirilmesidir. Örneğin satın alma siparişi gibi. Bu iş dokümanı hiçbir insan katkısı olmadan, bir yerden diğer bir yere gönderilmektedir. EVD kağıtla yapılan iş dokümanlarının yerini almakta ve normal elle yapılan işlerdeki yavaşlıkları ortadan kaldırmaktadır.

EVD ortaya çıkmış yeni bir kavram değildir. EVD 'nin temel aldığı fikir, asıl dokümanın bir kere girilmesi ve bu dokümanın sonraki gelen ilişkili yerlere sırayla elektronik ortamda gitmesidir. Genel olan EVD senaryosu, satın alacak kişinin satıcıya satın alma siparişini göndermesiyle başlar. Satıcı satın alacak kişiye satın alma onayı gönderir. Bundan sonra nakliye zamanında, satıcı satın alıcıya EVD 'nin daha gelişmiş nakliye ilanını gönderir. Bunu EVD 'nin faturası takip eder, ve en sonunda da satın alıcı bankasına ödeme talimatını gönderir, ve para satıcının hesabına satın alanın hesabından elektronik olarak (EFT) veya mali EVD ile aktarılır. EVD 'nin diğer kullanma şekilleri:

- Üniversiteler transcriptleri hızla değiştirirken
- Fabrikatörler karmaşık mühendislik dizaynlarını iletirken
- Tüketiciler kredi kartları ve banka işlemlerini yaparken
- Hükümetler gelir vergisi kazançlarını elektronik ortamda toplarken

EVD ilk kez Kuzey Amerika'da 1970 lerde ortaya çıktı. İlk olarak taşımacılık sektöründe uygulandı. Bunu gıda, ilaç dağıtımı, otomobil, banka ve finans sektörleri izledi. İşlerin elektronik ortamda yapılması elle yapılmasına göre daha hızlı, daha etkili ve daha doğru olarak gelişti. Böylece verimlilik arttı, siparişler daha hızlı yapıldı ve de bunlar rekabette avantaj sağladılar. Endüstride normal olarak kağıt üzerindeki işlemlerle yapılan işlerin maliyeti \$70 iken, EVD 'yle \$1 a geliyor. 1980 lerde EVD böylece Kanada iş çevresinin bir parçası haline geldi.[5]



1.2 EVD Ne Değildir?

EVD deki amaç ticari işlerin otomasyonudur. Bundan dolayı elektronik posta, faks veya benzeri şeyler EVD kapsamında değildir, çünkü bunlarda insan katkısı vardır.

1.3 EVD Nasıl Çalışır?

EVD 'yi kullanmak için bunu kullanacak olan endüstrinin EVD 'nin mesaj formatlarının standartları hakkında uzlaşması gerekmektedir. Ticareti yapanların böylece aynı sistemleri kullanmasına gerek kalmayacaktır, çünkü standartlaşma bilgisayar donanım ve yazılımına bağımlılığını ortadan kaldırır. Kuzey Amerika'da kabul edilen standart 'ANSI X12' dir. Bununla birlikte uluslararası iş çevrelerinde kabul edilen standart 'UN/EDIFACT ' tır. Yöresel sistemlerden gönderilen veriler için de ayrıca tercüme eden yazılımlara gerek vardır.

1.4 EVD'de Haberleşme

EVD'de datanın dağıtımı değişik şekillerde olmaktadır. Bunların en yaygın olanları:

1. Noktadan Noktaya EVD

Adındanda anlaşılabilceği gibi iki bilgisayar arasında direkt olarak bağlantı vardır.

2. Özel Ağlar

Bir tane özel ağ 'elektronik postakutusu' veya 'depolayıp iletme' faaliyeti sağlar, bunu kullananlar aylık bir ücret öderler.

3. İnternet

İnternet EVD için bir potansiyeldir, çünkü mesajlar hızlı ve düşük maliyetle dağıtılır[Ratnasingham, 1998].

Bunlarla beraber EVD uygulamalarında bazı problemlerle karşı karşıya kalırız, örneğin data değişiminde yetkili kişilerin onaylanması, elektronik imza yetersizliği, tam bir standartlaşmanın olmayışı , yıllık hesapların denetiminin

izlenmesindeki eksiklikler. Büyük organizasyonların bu konulardaki hassasiyetlerini düşünürsek, EVD uygulamalarının başarısı ve büyümesi bu sorunları etkili bir şekilde çözmesiyle ortaya çıkacaktır. Ayrıca EVD' nin haberleşmesindeki güven tabii ki yazılım ve donanımla da yakından ilgilidir.

EVD iletişimdeki risklerde önemlidir. İşlemleri izleme, firmanın kontrolü bu riskleri çok önlememektedir. Örneğin firma tedarikçisinden 2000 tane cd sürücüsü istesin, fakat iletişim problemlerinden dolayı bunun yerine 200 tane bilgisayar gönderilsin. Bu durumda önümüze resmi problemler çıkacaktır, 1) cd sürücülerinin gelmemesinden dolayı oluşan kazanç kaybı, 2) istenmeden gönderilen bilgisayarların parası, 3) sistemi izlemeden kim sorumlu? Tabii ki bu soruların cevapları EVD ile ilgili anlaşma kontratında belirlenmiş olmalıdır. Ayrıca yapılan araştırmalara göre tam bir EVD kontratı güvenlik önlemlerini de kapsamalı. Çünkü güvenlik önlemlerindeki açıklıklar kullanıcı taraflar arasında anlaşmazlıklara yol açmaktadır. Sonuçta kullanışlı bir EVD sözleşmesinde farklı anlaşmalarla birlikte bunlarla ilgili güvenlik önlemleri de yer almalıdır.

EVD gerçekleştirilirken güvenlik problemleri olarak karşımıza Bütünlük , Onaylama, Reddedemezlik, ve Gizlilik çıkar, ve bunlar EVD standartları oluşturulurken ortaya çıkan 10 bölümün 5 tanesini içerir.

- Bütünlük

Mesajın içeriğinin değişip değişmediğini anlama

- Onaylama

Mesajın sahibini anlama, bunun için bazı mekanizmalar kullanılır örneğin mesaj onaylama kodu (MAC)...

- Reddedemezlik

Mesajın sahibinin delili, böylece mesaj sahibi mesajını reddedemez

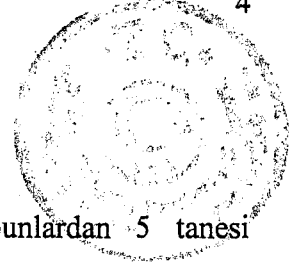
- Gizlilik

Mesajın ortaklar arasında değiştirilirken, içeriğinin dışı karşı korunması

- Anahtar yönetimi

Güvenlik problemi olmamakla beraber onunla ilgilidir, EVD standartlarında KEYMAN bunun için kullanılır[Ratnasingham, 1998]

Bu problemleri çözmek için sunulan bazı standartlar ve uygulamalar aşağıda anlatılacaktır:



‘EDIFACT’ Güvenlik Standartları

Yeni EDIFACT standartları 10 bölümden oluşmaktadır, bunlardan 5 tanesi güvenlikle alakalıdır:

- takım EVD için güvenlik kuralları (onaylama,bütünlük, reddedemezlik)
- güvenlik onaylama ve alındı mesajı
- takım EVD için güvenlik kuralları
- güvenlik anahtarı ve sertifika yönetimi
- etkileşimli EVD için güvenlik kuralları

EDI5

EDI5 protokolü Fransa’da ulusal ve de finanssal ortaklık projelerini içine alan EVD protokolünün özel bir halidir.

CRYPTIT-EDI5

CRYPTIT-EDI5 ürünü EDI5® gibi projelere gerekli güvenlik mekanizmalarını sağlar.

ETEBAC5

ETEBAC5 Fransa’da bankalar ve müşterileri arasında kullanılan güvenli dosya alışveriş sistemidir. Güvenlik servisleri aşağıdadır[EaseIt homepage]:

X400 Mesajlaşma Sistemi

Bu mesajlaşma sisteminin EVD’ da kullanım şekli, EVD mesajının X400 mesajlaşma formatına göre sayısal zarflanıp, karşı tarafa iletilmesidir [Humpreys].

X435 EVD mesajlaşma sistemi

X400 mesajlaşma sisteminin daha çok fonksiyon içeren halidir[Humpreys].

X500 dosyalama sistemi

Açık anahtar şifrelemesine ve sayısal imzalamaya dayanır, mesajlaşma da diğer güvenlikler için X400 mesajlaşma sisemini kullanır[Humpreys]



Uygulama

Bu çalışmada, yukarıda anlatılan güvenlik önlemleri göz önüne alınmış olup, bunların hepsine çözüm olacak bir yöntem geliştirilmiştir. Normalde EVD standartlarına göre güvenlik, mesajın başına ve sonuna eklenen başlık ve kuyruklarda ki parametrelere göre sağlanmakta, burada ki parametrelerle kullanılan algoritmalar belirlenmektedir ve güvenlikle EVD' in diğer fonksiyonlarının içice girdiği gözlenmektedir. Burada sunulan çözüm EVD' yi bir nesne olarak görmekte ve nesnenin güvenliğini sağlamaya çalışmaktadır. Teorik olarak düşünüldüğünde, EVD sisteminde haberleşecek taraflar ilkönce kendi aralarında güvenli bir iletişim kanalı oluşturmaları, bunun için güvenli kanalda şifreli haberleşme olmalı, verinin bütünlüğü korunmalı, onay mekanizması sağlanmalı, ve kullanılan yöntemlerle mesaj sahibi belli olmalı ki daha sonra çıkabilecek herhangi bir uzlaşmazlık için mesajlar delil teşkil edebilsin. Daha sonra oluşturulan EVD mesajı taşıyan nesne, bu sokete verilir ve güvenli şekilde karşı tarafa ulaştırılması sağlanmış olur. Normalde EVD standartlarında ki gibi mesajları tek tek şifrelemektense, o soketten geçen her veriyi şifreleyip iletişimi güvenli kılmak ve mesajları bu soketten göndermek, o hattın güvenliğini daha çok arttırmakta ve de atak ihtimallerini azaltmaktadır. Ayrıca nesneye yönelik olması koda esneklik getirir ve de gerçeklemeyi kolaylaştırır.

Pratikte, bu yöntemi gerçekleyebilmek için ilkönce bir EVD sistemine ihtiyaç duyulmuş olup, bunun için bir EVD simülasyon programı yazılmıştır. Bunun için bilgisayar parçaları sağlayan genel bir şirket ve onun ürünlerini müşterilere satan bayileri düşünülmüştür, istemci sunucu mantığıyla çalışan yazılıma karar verilmiştir. EVD standartlarına göre haberleşmeleri sağlanmıştır. Nesneye yönelik olması açısından gerçekleştirme dili olarak Java kullanılmıştır.

Bunlardan sonra, asıl iş olan güvenlik sağlamaya geçilmiştir. Java programlama dilinin nesneye yönelik olması ve de güvenlik mekanizmalarının gelişmiş olmasından dolayı, Java ve Java'da yazılmış GSK kullanılmasına karar verilmiştir. GSK opsiyonlu olarak her iki tarafada diğer tarafın kimliğini onaylama

mekanizması sağlar. Tarafların birbirini onaylamasından sonra, GSK güvenli iletişim için, şifreli haberleşmeyi sağlar. Taraflar arasında şifreli haberleşme gizliliği ilk planda sağlamakla beraber, ikinci olarakta adreslemeyi yapar. Ayrıca şifreleme algoritması özetleme fonksiyonu kullanır, bu da verilerin iletişim esnasında değişip değişmediğini kontrol için kullanılır. Bu güvenlik özet fonksiyonu veri bütünlüğünü sağlar. GSK kullanmakla onaylama, gizlilik ve verilerin bütünlüğü sağlanmıştır. GSK reddedemezliği sağlayamaz, bundan dolayı Java'nın sayısal imza mekanizması kullanılarak bu özellik sağlanmıştır.





2. GÜVENLİK PROBLEMLERİ VE EVD'DE GÜVENLİK

Bu bölümde EVD'deki teknolojinin değerlendirilmesi, birbiriyle alakalı taraflar arasında ki güven ilişkileri ve EVD'deki güvenlik işleri ele alınacaktır. Güven ticari çevrelerde başarılı olmak için çok önemli olup, ortaklar bunun sayesinde ciddi başarılar elde etmektedir. Güven ilk başta yapılmış bir anlaşmadır. Bununla beraber EVD' yi bütün bir sistem olarak düşündüğümüzde başkalarına karşı güvenliğinin sağlanması şarttır. Tam güvenlik hiçbir zaman sağlanamayacağı bilinmektedir, amaç olabildiğince güvenli bir sistem oluşturmaktır. EVD belli standartlara göre üretilmiş olan verilerin farklı yerlerden farklı yerlere taşınmasıdır. Bu bölümde ilk olarak anlatılmak istenen EVD'deki güvenin tam olduğudur. EVD yazılımı üç parçadan oluşmaktadır. Dokümanların haritalanması, verilerin çevirisi ve haberleşme. Dokümanların haritalanması ticari ortaklar tarafından gönderilen mesajlar ile üçüncü şahıs olan ağlar (VANs) birleştirir. Bu ağlar elektronik olarak haberleşmek için gerekli cihazlarla donatılmışlardır ve de gerekli servisleri sağlamaktadırlar, örneğin: EVD çevirme standartları, uluslararası bağlantılar, ve diğer üçüncü parti taşıyıcı ağlara olan bağlantılar.

Kağıda dayalı sistemlerde hataların ortaya çıkarıldığını ve düzeltildiğini anlamak için bir çok kontrol yapılmaktadır. Bu düzeltmeler deneyimli çalışanlar tarafından yapılmaktadır. EVD sisteminde de bu kontrol ve düzeltmeleri yapacak prosedürler bu kişilerin yerlerini almadırlar.

EVD sisteminde önemli bir konuda güvenlik prosedürlerinin sistemin gereken her yerinde yer almasıdır. Bununla eş değerde önemli bir konuda bu prosedürlerin birleştiğinde bütün sistemin güvenliğini sağlayabilmesidir. Sistemdeki her bölümde farklı olayların önceliği farklıdır, buna göre güvenlik bir firmada diğerine göre daha önemli olabilir. Sonuçta tam bir güvenlik sağlamak imkansızdır – hatalar yapılır, makineler bozulabilir, yazılımlarda hatalar olur. Bu yüzden, yönetim etkili bir şekilde kabul edilebilir güvenlik seviyesini tespit etmelidir.

Bu bölümde ilk olarak güven ilişkilerinden söz edilecek, ikinci kısımda EVD' da ki güvenlik riskleri anlatılacak, üçüncü kısımda yürürlükte olan EVD anlaşmalarından bahsedilecek ve de dördüncü ve son bölümde 'EDIFACT' in kabul ettiği riskler ve de bunlara karşı önlem almak için yapılmış standartlar belirtilecektir.

2.1 EVD' deki İnsan Merkezli Mimari ve Güven

EVD' de diğer bilgi teknolojilerine aksine yenilikler hemen uygulanamaz ve tek taraflı kullanılamaz. Bu nedenle EVD oluşturulurken firmalar ortaklarıyla stratejik işbirlikleriyle, birlik anlaşmalarına yönelmeye konsantre olmalıdırlar. EVD kullanıcılarından güvenlik seviyesini yükseltmek için birlik ve söz talep etmektedir. Ticari ortaklar arasındaki birlik güvenlik açısından izlenen yol olup, fırsatların yerine herkes için uygun değişiklikleri takip etmektedirler.

2.2 Güvenin Değişimleri ve Bunların Gelişim Evreleri

Güven zaman içinde belli evreler geçtikçe gelişir. Bununla birlikte başta caydırma tabanlı ve sonda tanıma tabanlı evreye sahip olma en yüksek güvendir ve de en gelişmiş güven türü bütün elektronik haberleşmedeki ilişkilerde bu evrelerin aynı olmasıdır. Güven EVD' de önemli bir önceliktir ki prosedürel değişiklikler kesin saldırılara yol açabilir.

2.2.1 Güven Evreleri (1,2,3)

(1) Caydırmaya dayalı güven

(2) Bilgiye dayalı güven

(3) Kimliğe dayalı güven

Güven Çeşitleri

Üç çeşit güven şekli vardır, ve bir çeşit diğer bir çeşide liderlik eder.

Caydırmaya dayalı güven: ticari ortaklar cezalandırılmaktan korktuklarından (kontratın iptali gibi) ne yapacaklarsa onları söylerler. Cezalandırma tehdidi negatif faktör olarak görülebilir, bununla birlikte hesaba dayalı güvende başarmaya dayalı ödül pozitif faktör olarak görülebilir.

Bilgiye dayalı güven: bilgilerini diğer ticari ortak ile birleştirilmiştir (emanetçi), bu güvenen tarafın emanetçi tarafı denemesi ve hareketlerini tahmin etmesi için bir fırsattır. Bu şekildeki güvende anahtar faktör zaman içinde ilişkiden dolayı oluşan bilgidir ticari ortakların birbirlerinin hakkındaki davranışlarını tahmin etmeleridir.

Kimliğe dayalı güven: başkalarının özelliklerini kendinde görmeye ve ticari ortaklarıyla ortak değerleri temel alır. Kimliğe dayalı güvende ortaklarla birebir iş temeli yerine genel işler etrafında dönmeğe dayalı yönelim vardır.

Çalışmaların sonucunda birinci evrede güven inançla ilgili, ve birlikteki başarısızlığa belli bir oranda cezalandırma tehdidi var. İkinci evrede ticari ortağın eğilimleri bilindiğinden, ne yapacağı kolayca tahmin edilebilmekte. Üçüncü evrede bir ticari ortağın bir diğerinin kişisel başarısına ihtiyaç duyduğunda olur.

2.2.2 Pilot çalışma

EVD' nin otomotiv endüstrisinde kullanılmasıyla bulunan ilk verilere göre ticari ortaklar arasında güven risk içeren proseslerde çok önemli. Bunun ilerisinde, kontrol mekanizmalarının güven seviyelerinde önemli etkisi vardır.

Bu çalışma bir çok evreden oluşmaktadır. İlk olarak, EVD ile alakalı olarak varolan ve risk içeren proseslerin analiz edilmesi gerekmektedir. Sonra güvenle ilgili olarak yapılan görüşmelerde anketler yapılmalı. Açık ve kapalı çeşitte sorulardan oluşmalı ve görüşmeler gündelik tartışmalar şeklinde olmalı. Böylece nitelikli bir çok veri toplanmış olur ve bunlar güvenlik ve güven arasında bağlantı kurmamızda önemli rol oynarlar. Bulgular güven arttıkça risklerin ve kontrollerin azaldığını gösterir. Sonuçta, çoğu risk iş riskine girer, teknoloji ile ilgili güvenlik risklerinden ziyade eşyaların dağıtılması, mesajın gizliliği ile ilgilidir.

EVD minimum değişiklikler (aradaki anlaşmazlıkları çözecek prosedürlerde, ticari prosedürlerde) içerek şekilde oluşturulur. Ticari ortaklar arasındaki

anlaşmalarda, VAN anlaşması, EVD standartlarında uzlaşma, iç siyasetlerinde, prosedürlerinde, ve yasal anlaşmalarla ön ticari anlaşmalar yapılmış olur. Bu kontratlar ticari ortaklar arasında yapılacak olan birlik anlaşmaları için iskelet görevi yaparlar.

Yüksek güven ticari ortaklar arasında umulan davranış olarak görülür ve bunun devamında ticari ortaklar diğerlerinin beklentilerine uyarlar, örneğin, eşyaların zamanında ve olması gereken yere ulaştırılması gibi. Bu nedenle, güven ve kararlılık arasında karşılıklı ilişki vardır ki bu kontratları yenilemeyle oluşacak ilişkilerdeki devam ihtimaline delildir.

Gerçekte aşağıdaki nedenlerden dolayı EVD organizasyonları düzenli şekilde yıllık hesapları kontrol edilememektedir:

- Ticari ortağına bağlı olma isteği, kendini adanmış ve kooperatifmiş gibi.
- Tedarik ediciler performans kalitesi kontrol listesiyle izlenir.
- Ticari ortağıyla uzun süredir ticaret yaptığından dolayı (10 yıldan fazla).
- Ticari ortağıyla aynı stratejik gayeyi paylaşması ve aynı endüstriyel sektörde bulunmaları.
- Haberleşmenin yoğunluğu ve frekansı yüksektir. Telefon, faks, elektronik posta EVD' ye ek olarak, işlemlerin onaylanması sırasında ve olumsuzlukları gidermede kullanılmaktadır. Kısaca, olumsuzluklar oluştuğunda acil faaliyetler olur. Daha ilersinde, ticari ortaklar düzenli olarak toplanıp en çok ilgilendikleri alanlardaki ticari prosedürlerini tartışır.
- Malzeme tedarik çizelgesi sipariş miktarından, dağıtımın çeşidinden, ve de gelişmiş taşıma ilanının gösterdiği zaman ve dağıtımın tarihinden oluşur ve bunlar gelişmiş şekilde on iki ay içinde hazırlanırlar. Arabanın oluşması için motorun zamanında ulaştırılması çok önemli ve değerlidir.

Pozitif güven ticari ortaklar arasında tutarlılık ve sigorta olarak görülebilir. Bu ticari ortakların birbirine dayandıkları ve sözlerini tuttukları anlamına gelir, sonuçta birliktelikteki seviyenin artması aradaki güveni artırır.

2.3 EVD’de Güvenin Önemi ve Faydaları

Birliktelikteki ilişkilerdeki güven sonucu birçok etkili fayda elde edilebilir:

- Detaylı kontratın sadece taslağı çıkarılır, resmi olarak yapılırsa hem çok zaman gerektirir hem de pahalıya mal olur,
- Birlikteliğin alanı genişletilebilir ve ticari firmalara ortaklığa girmeye izin verir, yoksa eşitlik kontratı olsa da bu sağlanamaz;
- işlem masrafları düşürülebilir.

Daha ilerisinde, güven iş ilişkilerinde gizliliği ve güvenliğini artırır, etkili bilgilerin değiş tokuşu iletilir, kuşkulu durumları giderme yönetimine yardımcı olur, be ticari ortaklar arasında koordinasyon ve birliktelik için fırsatlar sağlar.

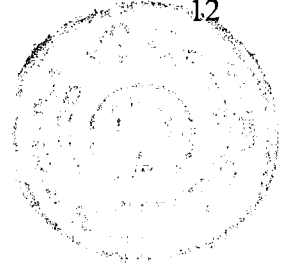
2.4 EVD’ de Güvenlik İşleri

EVD sisteminde güvenlik prosedürlerinin her tarafta olması çok önemlidir. Her kullanıcıda bu prosedürler eşit önemlidirler çünkü birleşince tüm sistemin güvenliğini sağlarlar. Kullanıcıları düşündüğümüzde her biri için farklı prosedürlerin önceliği olabilir ve bazılarında güvenlik diğerlerine göre daha önemlidir. Kesin güvenlik imkansızdır, bir takım problemler çıkması normaldir, bundan dolayı kabul edilebilir güvenlik önlemleri yapılmaya çalışılmalıdır.

EVD işlemlerin sisteme giriş şeklini değiştirir. Sistemdeki yetkili kısım tarafından girdinin doğruluğunun ve bütünlüğünün kesinleştirilmesi için denetim ve kontrol prosedürleri gereklidir. Bununla beraber bazı EVD riskleri bu denetim ve kontrol prosedürleriyle ortaya çıkarılamamaktadır. Güvenlik riskleri iş yapan taraflar arasındaki haberleşmedeki etkililikte zarar vermekte, bunlar arasındaki ilişkileri tehlikeye atmaktadır.

EVD deki güvenlik riskleri 6 alt grupta toplanabilir:

- 1) mesajı açığa vurmak



- 2) mesajın içeriğinin değişmesi
- 3) mesajın sırasının değişmesi
- 4) gönderici gibi davranmak
- 5) mesajın kaynağını, alındı mesajını reddetmek
- 6) işleri inkar etme, yapmama

Mesajı açığa vurmak herhangi bir elektronik haberleşme şeklinde en sık görülen güvenlik problemidir. Bu olay genelde yetkisiz birinin mesajlara ulaşma hakkını elde etmesiyle oluşur. Disclosure iş yapanlar arasındaki işlemleri rakiplerine karşı açığa vurur. Hassas bilgilere ulaşmak ilerideki iş görüşmelerinde rakiplere haksız avantajlar verecektir. Mesajı açığa vurmak riski verileri şifrelemekle, dijital imzalamayla ve de şifre kullanmakla azaltılsa da bütün problemleri gideren sağlam bir mekanizma yoktur.

Mesajın içeriğinin değişmesi bütün sistemi etkileyen diğer bir güvenlik ihlalidir. Mesajların içeriğini kurcalamak iki grup tarafından yapılabilir, birisi sistemde mesajın içeriğini değiştirme hakları bulunan kişilerce, veya yasal olmadan sisteme erişme ve mesaj içeriğini değiştirme hakkını kazanmış (atakçılar gibi). Bazı bilgiler böylece değiştirilebilmektedir, örneğin alınacak olan adet sayısı değiştirilebilmektedir. Sonuç olarak üretici firma doğru olmaya sayıda parça dağıtımını yapmakta, ve bu da satıcıların ürün dağıtım trafiğini bozmaktadır. Bu riski minimuma indirmek için, firma şifre kullanabilir buna ek olarak ta mesajların kopyalarını tutabilir.

Mesajın sırasının değişmesi hangi mesajın alındığının düzenini bozar veya aynı mesajın tekrarlanmasına sebep olur. Mesajı geciktirmenin alıcıların ürün programını bozması, veya finanssal bir sistemde sürekli para meseleleriyle ilgili bir mesajı göndermek bu riske bir örnektir. Bunlarla beraber üretici firmanın üretim programının bozulması içinde potansiyel bir tehlikedir. Ağ güvenliğinin artırılması için firma bu riski azaltmalıdır.

Gönderici gibi davranmak olayı sahte kullanıcıların mesajın yasal alıcısı veya göndericisi gibi davrandığı zaman ortaya çıkar. Sistem güvenliğindeki bir gedikten kaynaklanır. Sistem şifresinin ele geçirilmesiyle bu olay kolayca ortaya çıkabilir. Gönderici gibi davranma ile aslında gerçek kullanıcı tarafından yapılmayan siparişler

istenebilir. Bu olay alıcı ve satıcı arasındaki ilişkileri ciddi şekilde bozabilir. EVD sisteminde mesajların onaylanması bu riski azaltabilir.

Mesajın kaynağını veya alındı mesajını reddetmek alıcı tarafın mesajı aldığını reddetmesiyle oluşur. Örneğin firma EVD yoluyla gelen stoklarını azalt mesajını inkar edebilir. Bu riski düşürmenin bir yolu ağdaki ticari işlemlerin kayıtlarının yedeklenmesidir. Bunlarla beraber bu yedekleme işlemleri ve bu mesajların ne kadar tutulacağını hesaplanması yüksek maliyet gerektiren işlerdir.

İşleri inkar etme, yapmama kullanıcının yetkisiz erişim hakkı kazanmasıyla veya bazı özel hakları elde edip sistem kaynaklarını kullanmayı reddetmesiyle oluşur. Bu olay kolayca anlaşılabilir ve sistemce rapor edilebilir, bundan dolayı sistemde düşük bir risk taşımaktadır. Yasal kullanıcılar en kötü şartlar altında başka yollarla iletişim kurup doküman değişimini minimuma düşürüp tehlikeyi en az seviyeye indirebilirler.

X12, EVD güvenlik standartları ticari ortaklar arasında güvenlik konuları olan onaylama, bütünlük, ve gizliliği sağlar. Bununla beraber sayısal imzayı sağlamaz. Bu gibi işlerin yasal yorumları kullanıcıların ve de yasal çevrelerin ilgi alanına girmektedir. Bir sonraki bölümde tartışılacağı gibi güvenlik risklerinin seviyeleri EVD kullanıcılarının aralarında yaptıkları anlaşmalara bağlıdır.

Güvenlik işleri aşağıdaki alanlarda değerlendirilebilir:

- EVD mesajlarının taşınmasındaki tehlikeler
- EVD sisteminin diğer iş sistemleriyle ilişkilerindeki güvenliği
- Güvenlik mekanizmaları ve prosedürleri
- Güvenliğin yönetimi.

Tablo.2.1 EVD ile ilgili riskler

Güvenlik Riskleri	Tanımı	Organizasyonlara Etkileri	Potansiyel Çözüm
mesajı açığa vurmak (Disclosure)	Yetkisiz kişilerin mesaja ulaşması	Rakiplere haksız avantaj sağlaması	Veri şifreleme, parola
mesajın içeriğinin değişmesi (modification of message contents)	Mesajın içeriğinin değişmesi	İstenmeyen dökümanların veya yanlış miktarda gönderilmesi, alıcıya da satıcıya da etkiler	Parola, mesajların birden fazla kopyası ve ağ gönderilen lerin kopyesini saklar
mesajın sırasının değişmesi (modification of message sequence)	Mesajın gecikmesi veya mesajın sırasının değişmesi	Çizelgenin bozulması, gönderilen mesajın tekrar edilmesi için iki defa yapılması	Ağ güvenliği ile sağlanmalıdır
gönderici gibi davranmak (sender masquerade)	Bir taraf diğer bir tarafmış gibi davranır	Siparişi yanlış vererek daha fazla satış yaptırma, sipariş üreticiye ulaşmaz üretim akışı durur	Mesajın parola ile onaylanması
mesajın kaynağını ve alındı mesajını reddetmek (repudation of message origin or receipt)	Mesajın kaynağını reddetmek	Para akışının bozulması, malzeme yokluğu üretim çizelgesini bozar	Ağ yedeklemeyle kopyaları saklamalı
işleri inkar etme, yapmama (denial of services)	Yasal kullanıcının erişimini engellemek	Kolayca fark edilebilir ve ağ tarafından ani olarak yapılacak aksiyonlarla etkiler en aza indirilebilir	Ağ ın güvenliğince sağlanmalıdır

2.5 EVD Mesajlarının Taşınmasındaki Tehlikeler

EVD mesajlarının taşınması sırasında karşılaşılabilecek tehlikeler şunlardır:

- Mesaj gizli bir veri taşırken, bunu açıkça ifşa etmemesi önemlidir
- Orijinal mesaj elde edildiğinde, kritik veriler varacağı yere ulaşmadan önce değiştirilebilir
- Mesaj kazayla veya bilerek tekrarlanabilir,ve tekrarlanan mesaj alıcı tarafından gerçek sanılabilir, çünkü iki mesajda gerçek mesaj formatında.
- Organizasyonun durumundan mesajın alındığı garanti edilmeli, aksi takdirde hassas bilgiler rakip firmalar tarafından EVD kullanılarak elde edilebilir.
- Mesajın sahibinin şimdi veya daha sonra mesajı veya mesajın içeriğini inkar edememesi önemlidir.
- Mesajın dağıtımındaki gecikme önemli zıt etkilere sebep olabilir.
- İzleyeceği yol yanlış olan mesaj değiştirebilir veya varması gereken yere hiç ulaşmayabilir.

Bir organizasyonun EVD' nin tüm avantajlarını elde etmesinden sonra, bu problemlerle uğraşmak için prosedürleri tanımlamaya hazır olması gerekir. Bazı prosedürler ve teknikler güvenlik mekanizması ve prosedürleri adı altında tartışılmıştır.

2.6 EVD Sisteminin Diğer İş Sistemleriyle İlişkilerindeki Güvenliği

Mesajın güvenliği yalıtım olarak anlaşılmamalıdır, mesajın güvenliğiyle uğraşacak sistemler ve prosesler iletişim prosesleri kadar önemlidir. Bu proseslerden bazıları:

2.6.1 Gidecek Mesajın Güvenli Oluşturulması

EVD sistemine elle data girilirken gerçek tehlike yanlış verilerin girilmesidir, tür hatalarına sebep olur. Otomatik girdiler kullanılırsa, mesajın kaynağı olan sistemde hata vardır ve bu doğru olmayan girdilerin sisteme girmesine sebep olur. Doğru olmayan girdilerin girilmesiyle oluşan veya EVD' deki çevirme aletlerinden

dolayı oluşabilecek hatalardan dolayı mesajın iletilmesinde yanlışlıklara ve anlaşılan standartlarda uzlaşmazlıklara sebep olur.

Mesajı alanlar mesajın sistemdeki bir kullanıcıdan geldiğini bilmesi gerekmektedir beraber, gönderen tarafta yetkili olmayan kişiler tarafından kendi sistemlerinden mesaj gönderilmesini engellemelidirler. Buradaki tehlike hacker'ların sisteme girip orijinal mesaj üretmeleridir.

Mesaj oluşumunda yapılması gereken kontroller kolayca başarılabilir. Bunlar EVD' nin gelişimi sırasında varolan sistem gelişim ve sistem devam kontrolleri uygulanarak yapılır. Kontroller gönderilmiş mesajların gönderildiğinden emin olur. Bununla birlikte, sıkı koruma prosedürleri kimlik belirleme numarası girilirken, kullanıcılar girerken ve yetkilendirme sırasında kullanılır.

2.6.2 Gelen Mesajın Güvenli İşlenmesi

Bir mesaj alıcısına vardığı zaman, göndericinin, mesajın bütünlüğünün, ve doğruluğunun da onaylanması gerekir. Gerekli olursa da, mesajın alındığı bildirilir. Mesaj alıcının sistemine uygun şekilde çevrilmelidir. Mesajın bütünlüğü ve onaylanması ancak güvenlik mekanizmaları ve prosedürleriyle sağlanabilir.

Gelen mesaj alıcı bilgisayarın anlayacağı formata uygun hale çevrilmelidir. Bu çevirme prosedürü çevrilemeyen mesajları reddetme kapasitesine sahip olmalıdır. Bu reddetmeden sonra göndericiye doğru mesaj gönderilmesi sorulmalıdır. Çoğu çevirme programları hata bulma ve bunları görüntülemeyi birlikte yapar. Bu değişiklikler ve açıklamalar sistemde kaydedilir.

Bu gelen mesajın işlenmesi üzerindeki kontroller elle veya hem el hem de otomatik kontrollerin kullanılmasıyla yapılabilir. Bu kontrollerin çalıştığını anlamak için, mesaj alındıktan sonra gideceği uygulamaya kadar kaybolup kaybolmadığını kontrol etmek bir yöntemdir.

Verilerin tutulmasının amacı:

- Kuralları yerine getirmek ve yasal gerekler.
- İşlem yapıldığına ait delil
- Organizasyonel aktiviteleri kaydetmek için bilgi kaynağıdır
- Piyasa ve planlama aktiviteleri için bilgi kaynağıdır

Organizasyonlar verilerini yitirmemek ve veri geliştirmek için gereklerini ve veri tutma siyasetini belirlemelidir. Aynı zamanda çok fazla verini gereksiz yere tutulmasından kaçınılmalıdır. Eğer az veri üretilirse, her şeyi tutmak gerekli olan verileri özel olarak tutmaktan daha kolay olur. Tutulan verinin güvenliği kullanıcının niyetine göre değişir. Tutulan verinin değiştirilmesine, kaybolmasına, ve ifşa edilmesine karşı korunması gerekir.

Güvenlik mekanizmaları ve prosedürleri

Kullanılan güvenlik prosedürleri aşağıdakileri içerir:

- Şifreleme
- Sayısal imza
- Anahtar yönetimi
- Damgalamak

Bu prosedürleri destekleyen teknolojilerde şunlardır:

- Smartcards
- Wormdisks
- Biometrics

Şifreleme mesajın yetkili olmayanların anlamaması için belli algoritmalarla anlaşılmayacak halde karışık hale getirilmesidir ve aşağıdakiler için kullanılır:

- Kullanıcıyı onaylama: alıcı tarafından onaylama, gönderenin gerçek olması.
- Mesajın onaylanması: mesajın haberleşme sırasında kaybolmadığını anlamak için veya oynanmadığını anlamak için onaylanması.
- Gizlilik: şifrelenmiş veri alıcı ve gönderici dışında başka kişiler tarafından normal yollar ile algılanamazlar.
- Hata bulma: mesajın içeriğinin kaza ile değişmediğinin kontrol edilmesi,

- Kaynağın ispatlanması: üçüncü kişilere mesajın sistemde var olan bir kişi tarafından gönderildiğinin ispatlanması.

İki çeşit şifreleme vardır: (1) gizli anahtarlara dayanan şifreleme sistemi ve (2) açık anahtarlara dayanan şifreleme sistemi. Gizli anahtar sisteminde şifreleme ve şifreyi açmak için aynı anahtar kullanılır. Gizli anahtar sistemi aynı zamanda simetrik sistem olarak bilinir çünkü iki tarafta da aynı anahtar kullanılmaktadır. Aynı zamanda göndericiden alıcıya kullanılacak anahtar gönderilirken güvenlik sağlanmalıdır. En çok kullanılan gizli anahtar sistemi DES' tir (data encryption standard).

Açık anahtarlara dayanan şifreleme sisteminde açık anahtar herkes tarafından bilinebilir, fakat gizli anahtar tektir ve sadece gönderen taraf bilir. Açık anahtar sistemi aynı zamanda asimetrik sistem olarak bilinir, iki tarafta da farklı anahtarlar kullanılır. Açık anahtarların dağıtımı kolaydır çünkü bunları gizli olarak saklamaya gerek yoktur. Bununla beraber gizli anahtarların güvenliği sağlanmalıdır.

Sayısal imzalar asimetrik şifreleme kullanırlar ve mesajın kaynağının onaylanmasının garanti edilmesi ve de mesajın bütünlüğünün sağlanması için kullanılır. Ayrıca tanımama olayını engellemek için kullanılır (gizli anahtar başkaları tarafından elde edilmediyse), mesaj gönderenin imzasıyla kapanmış olur.

Sayısal imzada, gizli anahtar belli algoritmalar ile üretilir, mesajın özeti bu anahtar ile şifrelenir. Alıcı tarafındaki açık anahtar ile imza açılır ve onaylanır.

Anahtar yönetimi anahtarların üretilmesi, depolanması, değiştirilmesi. Arşivlenmesi ve silinmesini yapan prosedürlerdir. Şifrelemede güvenlik mekanizmasının başarılı olması çok yüksek oranda bu prosedürlerin doğru çalışmasına bağlıdır. Gereksinimler kullanılan algoritmaya göre ve anahtarların kullanılış amaçlarına göre çeşitlilikler göstermektedir.

Simetrik şifreleme sisteminde, şifrelenmiş veri ile haberleşen taraflar arasında anahtarları gizli tutmak için güvenlik gereksinimleri umursanmaz. Asimetrik sistemde ise elde ettiği anahtarın şifrelemek için gerekli açık anahtar olmalıdır, yoksa sahte kullanıcı gibi davranır.

Anahtarlar güvenli bir şekilde üretilmelidir ve yetkili olmayan kişiler bu anahtarların nasıl üretildiğini, ve bunları nasıl elde edebileceklerini bilmemeleri gerekir.

Sonra gelen güvenlik önlemide üretilen açık veya gizli anahtarların elle veya elektronik olarak dağıtılacağıdır:

- Elle dağıtım ve alma: Elle dağıtılan anahtarlar iki tarafa damgalanmış gizli zarflarla gönderilirler. Alıcı bunların gerçek olduğundan emin olmalıdır, belki kişisel imzalarla bu sağlanabilir.
- Elektronik dağıtım: Dağıtılacak gizli anahtarlar şifrelenip öyle gönderilebilirler. Verileri şifreleyecek anahtarlar şifrelenmiş mesajların içinde gönderilebilir.

Sertifika otoritesi, alıcı ve gönderici tarafından güvenilen üçüncü taraf olup, açık anahtarların sertifikalanmasından sorumludur. X500 gereksinimlerinden (X.509) bir kısmı dosyaların göndericinin açık anahtarlarını sertifikalarken nasıl kullanılacağını belirler. Mesaj alındığında, alıcı taraf gönderenin açık anahtarını talep eder. Şifreleme mekanizması onaylama için kullanılacağı zaman çeşitli anahtarların güvenli bir şekilde arşivlenmesi önemlidir.

Mesajın damgalanması, mesajın içeriğinin kazayla veya bilerek değişmediğinin görülmesi için kullanılır. Bu algoritma da kontrol toplamları kullanır ve bu kontrol toplamları mesajla birlikte veya ayrı olarak gönderilir. Kontrol toplamları mesajdaki veriler kullanılarak yapılan bir hesaplamanın sonucudur. Mesajın değişip değişmediğini anlamak için kontrol toplamları alıcıda tekrar hesaplanır. Eğer veriler değişmişse, hesaplanan kontrol toplamları farklı olacaktır ve buda damganın kırıldığını gösterir

Bu prosedürleri destekleyen teknolojik mekanizmalar şunlardır:

Smart kartlar kredi kartına benzeyen plastik kartlar olup, manyetik şerit yerine hafızalı mikroçip taşırlar. Smart kartın mikroişlemcisi kartı okuyan aletin bunu yapmaya yetkisinin olup olmadığını kontrol eder. Smart kartlar şunlar için kullanılırlar:

- EVD kullanıcılarını onaylamak ve tanımak için;
- Verilere ve kaynaklara erişimi denetlemek için;
- İşlemlerin imzalanması ve damgalanması sırasında

WORM diskler (bir kez yazılabilen ve çok kere okunabilen) depolama cihazları olup bir kere yazılır ve bir daha üzerinde değişiklik yapılamaz. Bununla birlikte, normal depolama araçlarından sınırlı olarak daha güvenlidir.

Biyometrik teknikler kullanıcıların kimliğini belirlemek için doğal özelliklerini kullanılarak yapılır. Bu doğal özellikler parmak izi, ses algılama ve benzeri teknikleri kapsar. Bir çok teknik pilot bölgelerde denenmekte, fakat maliyetinin yüksek olması çekiciliğini azaltmaktadır.

2.6.3 Güvenliğin Yönetimi

Organizasyonun güvenlik politikası güvenliğe bakış açısına göre geliştirilir ve hazırlanır. Daha detaylı güvenlik gereksinimleri ve prosedürleri özel EVD uygulamalarında bu politikadan türer. Bu gereksinimler hangi güvenlik aletlerinin prosedürlerde kullanılacağını belirlemek için önemlidir. Bu politikanın tanımlanmasıyla resmi analiz metotları kullanılarak güvenlik risklerinin analizi yapılır.

Son birkaç yıl içinde risk analiz metodolojileri geliştirilmiştir. Bu prosedürler risklerin belirlenme ve ne kadar zamanda çözüldüğünü ve de bunun maliyetini ölçmeyi hedefler. Bu analizler EVD mesajlarında ve prosedürlerinde oluşturulmasından alınmasına kadar uygulanabilir. Riskler mesajın cinsine ve organizasyonun iş sahasına göre değişiklikler gösterebilir. Risk analizi kullanma organizasyonun hangi risklerle karşılaşma olasılığı olmasına ve hangi risklerin öncelikle azaltılması gerektiğiyle ilgili olarak kullanılabilir.

Organizasyonda azaltılması gereken ilk riskler içerden kontrol edilemeyenlerdir. Bu riskler de sınırlandırılmalıdır. Örneğin çevirme programının yazılımının düzgün olarak çalışması, fiziksel ve mantıksal ulaşım kontrollerinin uygun olması, yedekleme uygun olmalı ve test edilmeli.

Güvenlik bakış açısından atılmamalıdır. Güvenlik iş sistemlerinde ana gereksinimlerden biridir ve EVD sistemindeki fonksiyonların tamamlayıcısıdır. Güvenlik EVD sistemlerinde kesinlik ciddiye alınmalıdır.

2.7 EVD Kontratlarının Çeşitleri ve Güvenlik Riskleri

Elektronik kontratlardaki temel düşünce iş belgelerini EVD' nin kopyalamasıdır. Bu nedenle, diğer iş belgelerine uygulanan yasal işlemler EVD deki iş belgelerine de uygulanabilir olur. Bununla beraber kontratlarla ilgili yasal kurallar genelde yazılı ve imzalı kağıtlar üzerine dayanmaktadır. Bunların elektronik ortama uygun çeşitleri daha yürürlüğe girmemiştir. Bunlar yürürlüğe girene kadar EVD kontratları bu açığı gidermektedir.

EVD var olan bir ağ ile veya ticari ortaklar arasındaki sadece bu iş için kurulmuş özel bir ağ ile gerçekleştirilebilir. Eğer özel bir ağ ile gerçekleştirilecekse, ticari ortaklar aynı haberleşme kanalını kullanmaları şart değildir ama aynı haberleşme protokolü, iletişim hızını kullanmaları ve uyumlu donanıma sahip olmaları şarttır. Özel ağlar bu şartları sağlayan servisler sağlayarak bu problemleri çözerler. Özel ağlar ayrıca posta kutusu servisi sağlayarak mesajların belli bir süre için geçici olarak saklanmalarını sağlarlar. Güvenlik riskleri özel ağlarda daha azdır. Bununla birlikte özel ağlarda esneklik daha azdır yani bir istasyonun yerini değiştirmek zordur ve de gerçekleştirmek pahalıya mal olmaktadır. Diğer yandan özel ağlar kullanmak EVD yi gerçekleştirmekteki esnekliği artırmakla beraber güvenlik risklerini de artırmaktadır. Bu nedenlerle, EVD kontratına karar vermeden önce, yönetim var olan özel ağların ve özel ağların risklerini araştırmalıdır. Bütün tehditleri elemek imkansızdır. Sonuç olarak, EVD yi kullanacak olan ticari ortaklar için riskleri ve sorumlulukları değerlendirmek önemlidir ve böylece uygun olan EVD kontratına ulaşılabilir.

EVD kontratları 5 ana grup altında toplanabilir:

- 1) özel iki yönlü
- 2) özel tek taraflı
- 3) özel çok yanlı
- 4) özel üçüncü taraf taşıyıcı
- 5) genel taşıyıcı çok yönlü

Özel iki yönlü anlaşma genelde yakın ticari ortaklar arasında yaygındır. Bu anlaşma ticari iki taraf arsında EVD mesajlarının başka taraflar giremeyecek şekilde direkt olarak haberleşmesini sağlar. Bu şekilde anlaşmalar dışardan birinin girmesine karşı risksizdir.

Özel tek taraflı anlaşmada bir taraf bir protokol belirler ve bununla iş yapacak grup bu protokolle haberleşerek bu tarafla iş yapar. Bu anlaşmada ağ birden çok grupla haberleşecek şekilde olmalıdır, ve ağ sadece haberleşmek için görevlidir, yedekleme veya başka bir görevden sorumlu değildir. Ancak piyasadaki hakim müşteriler böyle bir anlaşma sunabilirler. Ağ herhangi bir ek servis sağlamadığı için en yüksek risk bu anlaşmadadır.

Özel çok yanlı anlaşma tipik hub ile olan konuşma düzenidir. Büyük üreticilerin özel ağlarıyla EVD sistemini kurmasıyla olur. Buradaki ağ mesajları alıcılarına göndermekle ve bunları depolayacak alan sağlamakla görevlidir. Bununla birlikte gönderilen mesajların kopyalarını karışıklıkları önlemek için belli bir süre tutar. VAN bu anlaşmada baskın bir rol oynar. Özel iki yönlü anlaşmayla karşılaştırıldığında güvenlik riski fazladır ve taşıyıcı tarafından sağlanan servise bağlıdır. Bu da taşıyıcının yasal yükümlülüklerinin anlaşmada açık bir şekilde açıklanmasını gerektirir.

Genel taşıyıcı çok yönlü anlaşmalarda taşıyıcıların farklı EVD standartlarında haberleşmeyi sağlayacak proselere ihtiyaçları vardır. EVD kullanıcıları için taşıyıcı kontrol etmek zor olduğundan, taşıyıcılar bilgileri depolarken hileye yatkındırlar. Bununla beraber taşıyıcılar farklı kullanıcıların farklı mesajlarını ayırt etmek zorunludurlar. Aktarılan mesajlar burada da depo edilmek zorundadır. Bu sistem genel taşımadan daha güvenli olmakla beraber özel iki yönlü kadar güvenli değildir.

Tablo 2.2 EVD ticari anlaşmalarının çeşitleri

EVD anlaşmalarının çeşitleri	Şu anki durumları	Network çeşitleri	Haberleşme şekilleri	Ağın sorumlulukları
özel iki yönlü (private bilateral)	Bir çok şirket tarafından kullanılmakta	İki taraf arasında özel ve bu iş için kullanılan ağ	İki taraf arasında kabul edilmiş ve üstünde anlaşılmış bir protokolle	Yok, çünkü ticari ortaklar bunu direkt olarak halleder
özel tek taraflı (private unilateral)	Genelde kullanılmaz	Çok taraflı ağ	Teklif eden taraf bir protokol sunar bu bununla anlaşılan herkes için açıktır	Sadece taşıyıcı olarak sorumluluğu vardır
özel çok yönlü (private multilateral)	Tipik hub ile konuşmayla	Bir alıcı ve birden fazla tedarikçi arasında olan özel ve bu iş için kullanılan ağ ile	Tedarikçi bir tane alıcıyla bağlantı kurar ve kendi aralarında haberleşmezler	Yok, protokol ve güvenlik alıcı tarafından sağlanır
özel üçüncü taraf taşıyıcı (private third-party carrier)	Bir çok şirket tarafından kullanılmakta	Çok taraflı VAN	Aynı taşıma protokolünü kullanarak çoklu taraf diğer biriyle haberleşir	Taşıyıcıya bağlıdır, fakat erişimler kontrol altında olmalıdır, mesajlar yedeklenmeli ...
genel taşıyıcı çok yönlü (common carrier-multilateral agreements)	Gelecekte kullanılmaya aday	Çok taraflı – sadece EVD için adanmış	Farklı taraflar farklı protokoller kullanırlar. Ağ sorumlulukları ve diğer işleri üstlenir	Taşıma için herşey ve cevrim işleri

Tablo 2.3 Olan EVD anlaşmalarının risk seviyeleri

			Güvenlik	Riskleri		
Anlaşma çeşitleri	mesajı açığa vurmak (disclosure of messages)	mesajın içeriğinin değişmesi (modification of message contents)	mesajın sırasının değişmesi (modification of message sequence)	gönderici gibi davranmak (sender masquerade)	mesajın kaynağını ve alındı mesajını reddetmek (repudation of message origin or receipt)	işleri inkar etme, yapmama (denial of services)
özel iki yönlü (private bilateral)	Düşük	Düşük	Sıfır	Düşük	Düşük	Düşük
özel tek taraflı (private unilateral)	Çok yüksek	Çok yüksek	Çok yüksek	Çok yüksek	Çok yüksek	Yüksek
özel çok yönlü (private multilateral)	Düşük	Düşük	Olası	Düşük	Olası	Olası
özel üçüncü taraf taşıyıcı (private third-party carrier)	Orta	Orta	Taşıyıcıya bağlı	Orta yüksek arası	Düşük orta arası	Olası

genel taşıyıcı çok yönlü (common carrier-multilateral agreements)	Zor	Zor	Taşıyıcıya bağlı	Olası	Düşük	Olası
--	-----	-----	------------------	-------	-------	-------

2.8 EVD Güvenliği ile İlgili Yapılan Çalışmalar

2.8.1 EDIFACT Güvenliği [UN/EDIFACT DRAFT DIRECTORY, 1990]

EVD gerçekleştirilirken karşımıza çıkan güvenlik problemleri şunlardır

- Bütünlük

Mesajın içeriğinin değişip değişmediğini anlama

- Onaylama

Mesajın sahibini anlama, bunun için bazı mekanizmalar kullanılır örneğin mesaj onaylama kodu (MAC)...

- Reddedemezlik

Mesajın sahibinin delili, böylece mesaj sahibi mesajını reddedemez

- Gizlilik

Mesajın ortaklar arasında değiştirilirken, içeriğinin dışa karşı korunması

- Anahtar yönetimi

Güvenlik problemi olmamakla beraber onunla ilgilidir, EVD standartlarında KEYMAN bunun için kullanılır[Ratnasingham, 1998]

EDIFACT Güvenlik Standartları

Yeni EDIFACT standartları 10 bölümden oluşmaktadır, bunlardan 5 tanesi güvenlikle alakalıdır:

- takım EVD için güvenlik kuralları (onaylama,bütünlük, reddedemezlik)

- güvenlik onaylama ve alındı mesajı (AUTACK)
- takım EVD için güvenlik kuralları (gizlilik)
- güvenlik anahtarı ve sertifika yönetimi (KEYMAN)
- etkileşimli EVD için güvenlik kuralları

Güvenlik Teknolojileri

Mesajları şifreleyip şifrelerini açmak için iki yöntem vardır, simetrik ve asimetrik kriptoloji. DES simetrik kriptolojiye bir örnektir, bu teknikte haberleşecek tarafların ikisinde de aynı anahtar bulunmalıdır, bu iki taraf haberleşmeye başlamadan önce haberleşecekleri anahtarları kendi aralarındaki bir protokolle belirlemelidirler, örneğin diffie-hellman. RSA diğer asimetrik kriptolojinin bir örneği olup, burada alıcı tarafın kendine özel bir anahtarı olup, mesajı gönderecek tarafta da karşının genel anahtarı olacaktır. Gönderici genel anahtarla şifreler, alıcı tarafta kendi özel anahtarı ile mesajın şifresini açar, mesaj ancak bu özel anahtarla açılabilir. Asimetrik kriptoloji genellikle sayısal imzalamalarda kullanılır. İki tarafta özetleme fonksiyonu kullanır. Gönderici taraf mesajın özetini alır ve mesajın sonuna ekler, alıcı taraf mesajı alınca mesajın özetini alır ve mesajdaki sayısal imzayla karşılaştırır, eğer aynıysa mesaj onaylanmış olur. Simetrik kriptoloji ise genelde mesajları şifreleme ve şifresini açmakta kullanılır[Ratnasingham, 1998].

EDIFACT Güvenlik Standartlarının Uygulanışı

Onaylama, Bütünlük ve Mesajın Sahibinin Reddedememesi

Bu bölümde güvenlik başlığı ve kuyruğundan bahsedilecektir. Güvenlik başlığı göndereni ve alıcıyı belirlemek için bölümler içerir, bunlar;

USH bölümü başlık bölümü, bu bölümle bazı öze bilgiler belirlenir, örneğin sayısal imza gibi

USA bölümü (ilki) bütünlük mekanizması için kullanılır (özet)

USA bölümü (ikincisi) onaylama mekanizması için kullanılır (MAC)

USA bölümü reddedemezlik için kullanılır (sayısal imza)

USA bölümü(leri) veri sıkıştırması ve gizlilik için kullanılır

USC bölümü sertifika referansları için kullanılır

Güvenlik kuyruğu başlığı kapamak için kullanılır ve güvenlik başlığındaki güvenlik işlerinin sonuçları bu kuyrukta görülür, bunun için sonuç bölümü kullanılır (USR)

Güvenlik Onaylama ve Alındı Mesajı (AUTACK)

Bu mesaj iki amaç için kullanılır, birincisi güvenlik göndermekte, diğeri ise güvenlikle ilgili hataları ve onay alındıları raporlamaktadır.

USB bölümü değişimin olduğunu anlamakta kullanılır

USX bölümü EDIFACT güvenli mesajına referans olmak için kullanılır

USY bölümü güvenlik sonuçları için kullanılır

Gizlilik

Güvenlik başlığı ve kuyruğu gizlilik servisinde kullanılacak algoritmayı belirlemekte kullanılır. EDIFACT yapısı veri şifreleme başlığı ve şifreli mesaj için USD bölümü, veri şifreleme kuyruğu içinde USU bölümü içerir.

Güvenlik Anahtarı ve Sertifika Yönetimi (KEYMAN)

Anahtar yönetimi mesajıyla yapılır (KEYMAN). Üç amaç için kullanılır, istek yapmakta, isteğe karşılık vermekte, ve de sertifikaları listelemekte. KEYMAN mesajının bir özelliği, EVD formatı kullanmaz, X509 sertifikası için ASN.1 kullanır.

Etkileşimli EVD İçin Güvenlik Kuralları

Interactive EVD tüm EVD mesajlarını değişim sürecinde içerir. batch EVD den en büyük farkı, işlerin etkileşimli olması bu yüzden zaman önemlidir. Interactive EVD iki kısımdan oluşur, başlatıcı ve karşılayıcı. Oturum değişim başlıklarının değişimi ile başlar. Oturum kurulur, uygulama mesajları değiştirilir, ve bu diyalog değişim kuyruklarının değiştirilmesi ile bitirilir[5].

2.8.2 EDI5

EDI5 protokolü Fransa'da ulusal ve de finanssal ortaklık projelerini içine alan EVD protokolünün özel bir halidir.

Kullandığı güvenlik mekanizmaları normalden farklıdır.

CRYPTIT-EDI5 ürününün destekledikleri:

- Çeşitli aygıtları desteklemesi, disket sürücüsü gibi...
- Ek özet fonksiyonları vardır, MDx ve RIPEMDx.
- Ek akıllı kart okuyucuları.

CRYPTIT-EDI5 içerdikleri:

- Müşteri ve sunucu için ETB5V2.
- Onaylayıcı için AUTHORITY.[EaseIt homepage]

2.8.3 CRYPTIT-EDI5

CRYPTIT-EDI5 ürünü EDI5® gibi projelere gerekli güvenlik mekanizmalarını sağlar.

Güvenlik Servisleri

- Erişim izni,
- Onaylama,
- bütünlük,
- gizlilik,
- reddedemezlik. [EaseIt homepage]

2.8.4 ETEBAC5

ETEBAC5 Fransa'da bankalar ve müşterileri arasında kullanılan güvenli dosya alışveriş sistemidir. Güvenlik servisleri aşağıdadır[EaseIt homepage]:

Tablo 2.4 Etaback5

Güvenlik servisleri	Güvenlik teknikleri
Erişim izni	Akıllı kart, kullanıcı listesi
Onaylama	RSA512 ve 1024 bits
Bütünlük	DES-CBC, DES-MDC ve SHA1

Gizlilik	DES-CBC 56bits ve Triple DES-CBC 112 bits
Reddedemezlik	RSA512 ve 1024 bits
Onay mekanizması	DXP akıllı kart(müşteri) and D9 akıllı kart(bankalar).

2.8.5 X400 Mesajlaşma Sistemi

Bu mesajlaşma sisteminin EVD' da kullanım şekli, EVD mesajının X400 mesajlaşma formatına göre sayısal zarflanıp, karşı tarafa iletilmesidir.

Bu sistemde sağlana güvenlik özellikleri:

- Onaylama,
- bütünlük,
- gizlilik,
- reddedemezliktir[Humpreys].

2.8.6 X435 EVD Mesajlaşma Sistemi

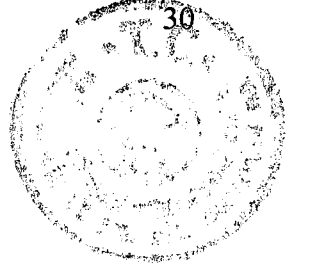
X400 mesajlaşma sisteminin daha çok fonksiyon içeren halidir. X400' e ek olarak aşağıdaki özellikleri içerir:

- EVD mesajlaşmasının sorumluluk onayı
- Bu onayın reddedemezliği[Humpreys]

2.8.7 X500 Dosyalama Sistemi

Açık anahtar şifrelemesine ve sayısal imzalamaya dayanır, mesajlaşma da diğer güvenlikler için X400 mesajlaşma sisemini kullanır[Humpreys].

3. UYGULAMANIN KURAMSAL YAPISI



3.1 EVD ve Problemler

EVD gereklenirken karřımıza birok problem ıkmaktadır. Farklı platformlar bařta en nemli problemdir. Farklı dillerde geliřtirilen EVD yazılımları ve aralarında tam bir standartlařmanın olmayıřı, mesajların tam tanımlı olmayıřı ve verilerin gvenlięinin saęlanması bařa ıkmamız gereken problemlerdir. Bunun iin EVD kullanacak taraflar ilk nce standartlar hakkında uzlařması gerekmektedir. Genel olarak kullanılan standartlar ANSI X12 ve UN/EDIFACT ' tır. Bu standartlařmalardan sonra verilerin gvenlięi saęlanması gerekmektedir.

EVD gerekleřtirilirken karřımıza ıkan gvenlik problemleri řunlardır:

- Btnlk

Mesajın ierięinin deęiřip deęiřmedięini anlama

- Onaylama

Mesajın sahibini anlama, bunun iin bazı mekanizmalar kullanılır rneęin mesaj onaylama kodu (MAC)...

- Reddedemezlik

Mesajın sahibinin delili, bylece mesaj sahibi mesajını reddedemez

- Gizlilik

Mesajın ortaklar arasında deęiřtirilirken, ierięinin dıřa karřı korunması[Dosdale, 1998]

3.2 Kuramsal Yapı

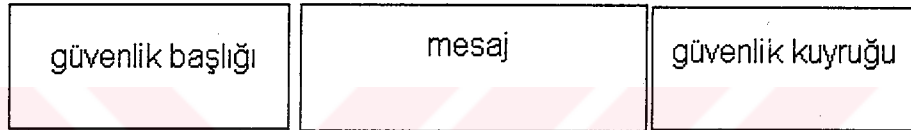
3.2.1 Uygulamanın Genel zellikleri

Bu alıřmada, ilk nce bir EVD simlasyon programı gereklenmiřtir. Bunun iin ilk olarak EVD uygulaması yapılabilcek bir uygulama dřnlmesi ve buna uygun olarak gereklenmesi řarttır. Bunun iin kullanıcılara bilgisayar paraları saęlayan bayiler ile bayilere bu paraları saęlayan tedariki bir firma arasında ki iřlemleri gerekleyen bir simlasyon programı bu gerekli řartları yerine getirir. Bu

programın istemci-sunucu mantığına uygun bir programın olması gerekmektedir. EVD mesajlaşma standartlarına göre gerçekleşmesi ile, elimizde EVD’da güvenlik hakkında çalışabileceğimiz bir ortam oluşturur.

3.2.2 Uygulamanın Güvenlik Özellikleri

Normalde EVD standartlarında, oluşturulan EVD mesajının önüne güvenlik başlığı, sonuna da güvenlik kuyruğu eklenmektedir. Başlıkta kullanılan şifreleme algoritmasının parametreleri, onaylama ile ilgili parametreler, kuyrukta ise bu işlemlerin başarılı olup olmadığı bilgileri bulunmaktadır. Bu bilgiler sayesinde her EVD mesajı tek tek şifrlenmekte ve bu şifrelenen mesaj karşı tarafa gönderilmektedir.



Şekil 3.1 Güvenli EVD Mesajı

Veri Bütünlüğü

Bu çalışmada bulunacak yöntemin yukarı da geçen tüm güvenlik problemlerine çözüm getirmesi gerekmektedir. Öncelikle verinin bütünlüğü sağlanması gerekmektedir, bunun için özetleme fonksiyonu kullanmak en güvenli yoldur, mesajın küçük bir özeti alınır ve mesaja eklenir. Burada özetleme fonksiyonu kullanmak güvenlidir, çünkü özetleme fonksiyonlarından elde edilen özetlerden asıl mesajı elde etmek imkansızdır, bu fonksiyonlar tek yönlüdürler, ve de bir mesajın tek bir özeti elde edilebilir, bu yüzden mesajın içeriğinin değişip değişmediğinin, yani verinin bütünlüğünün korunması için en güvenli yollardan birisi özetleme fonksiyonları kullanmaktır. SHA en sık kullanılan özetlem fonksiyonlarından biridir.

SHA ile Mesaj Özeti Alma

Mesaj özeti doldurularak 512 bit’ in katı hale getirilen mesajdan hesaplanır. Hesaplama metodu iki tane tampon kullanır, herbiri 32-bit lik 5 tane kelime ve 80 tane 32 bit lik kelime dizisi içerir. İlk beş kelime A,B,C,D,E,F olarak adlandırılır,

ikinci beş kelime de H_0, H_1, H_2, H_3, H_4 olarak adlandırılır. Seksen kelimelik diziler ise $K_0, K_1, \dots, K_{79}$ olarak adlandırılır.

Mesaj özeti oluşturmak için, 16 kelimelik bloklar $M_1, M_2, \dots, M_n$ işlenirler. Herbir işleme 80 adım gerektirir.

Herhangi bir bloğu işlemeyen önce $\{H_i\}$ serisi iklendirilir(hex olarak), örneğin:

$$H_0 = 67452301$$

$$H_1 = EFCDAB89$$

$$H_2 = 98BADCFE$$

$$H_3 = 10325476$$

$$H_4 = C3D2E1F0.$$

sonra:

a. M_i 16 kelimeye bölünür $K_0, K_1, \dots, K_{15}$, K_0 en soldaki kelimedir.

b. $t = 16$ dan 79 e kadar $K_t = S^1(K_{t-3} \text{ XOR } K_{t-8} \text{ XOR } K_{t-14} \text{ XOR } K_{t-16})$.

c. $A = H_0, B = H_1, C = H_2, D = H_3, E = H_4$.

d. $t = 0$ dan 79 a kadar

$$\text{GEÇİCİ} = S^5(A) + f_t(B, C, D) + E + K_t + K_t;$$

$$E = D; D = C; C = S^{30}(B); B = A; A = \text{GEÇİCİ};$$

e. $H_0 = H_0 + A, H_1 = H_1 + B, H_2 = H_2 + C, H_3 = H_3 + D, H_4 = H_4 + E$.

M_n işledikten sonra, mesaj özeti 160-bit 5 kelime ile belirtilir:

$$H_0 H_1 H_2 H_3 H_4 [\text{SECURE HASH STANDARD, 1995}].$$

Onaylama

Veri bütünlüğünden sonra, tarafların birbirini onaylaması ikinci güvenlik problemidir. Bunun için en güvenli yollardan biri asimetrik şifrelemedir, çünkü bu yöntemin kırılması simetrik şifrelemeye göre çok daha zordur, kullandığı anahtar uzunluğu yeterince büyüktür. Anahtarının uzunluğunun büyük olması sebebiyle

yavaş olmasına karşın, sadece ilk başta istemci ve sunucuyu onaylayacağı için ve bunu bir kere yaptıktan sonra heberleşmeye devam edeceği için bu yavaşlık önemsizdir. Bu işlem için RSA algoritması kullanılmaktadır.

RSA

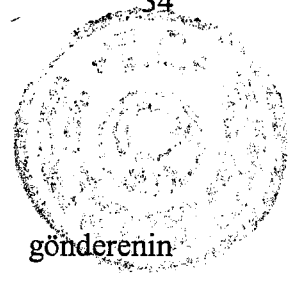
RSA algoritması 1978 yılında Ron Rivest, Adi Shamir ve Leonard Adleman tarafından bulunmuştur.

Bu algoritmanın altında yatan matematiksel kuram aşağıda ki maddelerdir:

1. İki tane büyük asal sayı bulunur, bunlar P ve R olsun (örneğin 1024-bit).
2. Bir E bu sayı 1'den büyük, PR çarpımından küçük olsun, öyle ki E ve $(P-1)(R-1)$ nispeten asal sayı olsunlar, yani aynı asal sayı bölenleri olmasın. E 'nin asal sayı olması gerekmiyor, ama mutlaka tek sayı olmalıdır. $(P-1)(R-1)$ asal sayı olamaz çünkü çift sayıdır.
3. Bir D sayısı hesaplanır, öyle ki $(DE-1) (P-1)(R-1)$ 'e bölünebilsin. Bu eşitlik matematikçiler tarafından $DE = 1 \pmod{(P-1)(R-1)}$ şeklinde yazılır ve D E 'nin çarpmaya göre tersi olur. Bu işlemi yapmak kolaydır, bir K sayısı bulunur, bu sayı şu eşitliği sağlamalıdır, $D = (K(P-1)(R-1) + 1)/E$.
4. Şifrelem fonksiyonu, $C = (T^E) \pmod{PR}$ dir, burada C şifreli metin, T normal metin, $^$ üs alma fonksiyonunu belirtir. Şifrelenen mesaj T , PR 'dan küçük olmak zorundadır.
5. Şifreyi açma fonksiyonu $T = (C^D) \pmod{PR}$, C şifreli metindir, T normal metin, $^$ üs alma fonksiyonunu belirtir.

Burada açık anahtar çifti (PR, E) 'dir. Burada ki gizli anahtar D 'dir. $PR \pmod{}$ dur, E 'de açık çarpan, D 'de gizli çarpandır.

Açık anahtarların başkaları tarafından bilinmesi önemli değildir, çünkü (PR, E) açık anahtar çiftiyle D, P veya R kolay bir şekilde hesaplanamaz [DIGITAL SIGNATURE STANDARD (DSS), 1994].



Reddedemezlik

Üçüncü problem reddedemezliktir, burada mesajı gönderenin belirlenemesidir. Burada sayısal imzalama kullanmak mesajların sahibini belirleyici rol oynar. Gönderen taraf kendi gizli anahtarı ile mesajı imzalar ve imzasını mesajın sonuna ekler, karşı taraf mesajı aldığı anda imzayı ve mesajı ayırır, mesajı karşı tarafın açık anahtarı ile işleyip bir özet değer elde eder ve imza ile karşılaştırır, aynıysa mesaj onaylanır. Burada da asimetrik şifreleme kullanılır. Sayısal imzalama da DSA kullanılmaktadır.

DSA

DSA'da aşağıda ki parametreler kullanılır:

1. $p = \text{asal mod}, 2^{L-1} < p < 2^L$ ve $512 \leq L \leq 1024$ ve de L 64'ün katıdır
2. $s = p - 1$ 'in asal bölenidir, $2^{159} < s < 2^{160}$
3. $g = h^{(p-1)/s} \text{ mod } p$, h herhangi bir sayıdır $1 < h < p - 1$ ve $h^{(p-1)/s} \text{ mod } p > 1$
4. $x = \text{rasgele üretilmiş bir sayıdır ve } 0 < x < s$
5. $y = g^x \text{ mod } p$
6. $k = \text{rasgele üretilmiş bir sayıdır ve } 0 < k < s$

p, s ve g sayıları açık olabilir, kullanıcının gizli ve açık anahtarları x ve y 'dir. x ve k parametreleri sadece imza üretmede kullanılır, ve gizli olmalıdırlar. k parametresi her imza için ayrıca üretilmelidir.

M metninin imzası ve r, s sayıları aşağıda ki eşitliklere göre üretilmektedir:

$$r = (g^k \text{ mod } p) \text{ mod } s \text{ and}$$

$$s = (k^{-1}(\text{SHA}(M) + xr)) \text{ mod } s.$$

Yukarıda, k^{-1} k sayısının mod s'ya göre çarpmada tersidir, SHA bu alitrmada kullanılan özetlem fonksiyonudur.

Sayısal imza mesajla birlikte onaylanmak için karşı tarafa iletilir. Onaylama için, ilkönce p,s,g değerleri, gönderenin açık anahtarı ve kimliği alıcı tarafta bulunmalıdır.

M' , r' ve s' - M , r , ve s 'nin alınmış versiyonları olsun,ve y 'de açık anahtar olsun. İlkönce $0 < r' < s$ ve $0 < s' < s$ koşulları kontrol edilir; bir tanesi bile sağlanmırsa imza onaylanmaz. Eğer iki koşulda sağlanırsa, aşağıdakiler hesaplanır:

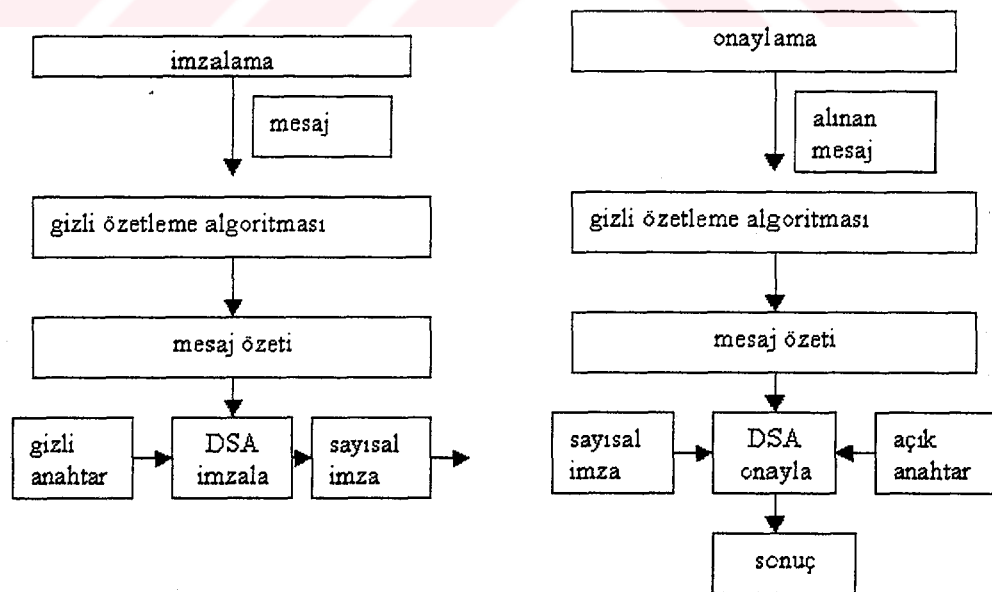
$$w = (s')^{-1} \text{ mod } s$$

$$u1 = ((\text{SHA}(M')w) \text{ mod } s$$

$$u2 = ((r')w) \text{ mod } s$$

$$v = (((g)^{u1} (y)^{u2}) \text{ mod } p) \text{ mod } s.$$

Eğer $v = r'$, imza onaylanmış demektir, değilse mesaj değişmiş, veya mesaj yanlış imzalanmış veya da başka birisi imzalamış olabilir, böyle olursa mesaj geçersiz sayılır [DIGITAL SIGNATURE STANDARD (DSS), 1994].



Şekil 3.2 DSA Algoritması ile İmza ve Onay Mekanizması

Gizlilik

Şon olarakta gizliliğin sağlanması gerekmektedir, bunun için hızlı şifreleme algoritmaları kullanılmalıdır, bu yüzden anahtar uzunluğu küçük olan simetrik şifreleme metodları kullanmak mantıklıdır. Bir taraf göndereceği mesajı anahtarla şifreler ve karşıya gönderir, karşı tarafta aynı anahtarla bu mesajı açarlar. Burada üçlü DES (TDES) algoritması kullanılır.

TDES

TDES algoritması aşağıda ki gibidir:

$E_K(I)$ ve $D_K(I)$ DES algoritmasının şifreleme ve şifre açma fonksiyonları, I şifrelenen ve şifresi açılacak olan metin, ve de anahtar K olsun. Aşağıda ki işlemler yapılır:

1. TDESşifreleme operasyonu: 64-bit lik mesajı I 64-bit'lik mesaj O ' ya çevrilmesi:

$$O = E_{K3}(D_{K2}(E_{K1}(I)))$$

2. TDES deşifreleme operasyonu: 64-bit'lik I mesajının 64-bit'lik O mesaja döndürülmesi:

$$O = D_{K1}(E_{K2}(D_{K3}(I))) \text{ [DATA ENCRYPTION STANDARD (DES), 1999].}$$

3.2.3 Güvenliğin Uygulanışı

Bunların bu programda uygulanışı olarak, iki taraf birbiriyle haberleşeceği zaman ilk önce birbirlerini onaylaması gerekmektedir, birbirlerini onayladıktan sonra el sıkışma yöntemiyle şifreli haberleşmeye geçmeli, veri bütünlüğü sağlanmalı ve de bundan sonra reddedemezlik için imza ve onay metodları kullanılmalıdır.

Burada normal EVD standartlarından farklı olarak mesajların tek tek şifrelenmesi yerine, ilk başta EVD nesnesi oluşturulmuş ve nesnenin güvenliği

sağlanmış, böylece mesajlaşma ve güvenlik, işleri ayrı olarak düşünülmüş, böylece gerçekleştirme de kolaylık ve de esneklik sağlanmıştır. Teorikte, nesne oluşturulmuş, daha önce den güvenli hale gelmiş bir kanala imzalanıp verilmiş, bu güvenli iletişim kanalından şifreli halde karşıya gönderilmiş, bütünlüğü korunmuş ve karşı tarafa ulaştığında mesajın imzası kontrol edilerek mesajın sahibi belirlenmiştir. Bütün bunlar gerçekleşirken nesneye yönelik yapılması kodun esnekliğini arttırmıştır. İletişim kanalının güvenliğinin sağlanması GSK ile sağlanmıştır, GSK sayesinde veri bütünlüğü, onaylama da gerçekleşmiştir, oluşturulan EVD nesneleri iletişim kanalına verilmeden önce imzalanarak reddedemezlik sağlanmıştır.

Bütün bunlar yapılırken, burada Java programlama dili ve GSK teknolojisinin kullanılması, uygulamanın teorisine uygun olduğundan seçilmiştir. Java nesneye yönelik programlama için en uygun dildir, GSK' de soket haberleşmesinde o soketin güvenliğini sağlamakta kullanılan en güvenli yöntemlerden biridir.

3.3 Kullanılan Araçlar ve Sebepleri

3.3.1 Java

Java genel amaçlı, nesneye yönelik, bir programlama dili olup, Sun Microsystems firmasınca geliştirilmiştir, ve temel özellikleri her ortamda çalışabilmesi, multithread olması ve ağ yapısını desteklemesidir. Her ortamda çalışabilmesi sayesinde Java da kod yazıp her yerde çalıştırabilirsiniz.

Java güvenlikle ilgili iki bakış açısı vardır:

- Java'nın güvenli çalışabileceği ortamlar yapmak.
- Güvenlik ekleri yapmak, bu kütüphanelerle, gerekli araçlar oluşturarak, servisler gerçekleştirerek olur.

3.3.2 Java Platform Güvenliği

İlkönce, java type-safe yani güvenli tiplerden oluşur, böylece kullanımı kolaydır ve tiplerle ilgili hata yapma oranı diğer dillere göre daha düşüktür.

Yazılımcılara güvenli kod yazmak konusunda özellikleriyle yardımcı olmaktadır, bu özelliklere otomatik hafıza kontrolü, çöp toplaması, güvenli tip örnektir.

Ayrıca sistemin kaynaklarına ulaşma JVM tarafından engellenmiş olup, SecurityManager sınıfı tarafından gelişmiş şekilde kontrol edilmekte, böylece sınırlama aksiyonlarıyla güvenilmez kodlar en aza indirilmiştir.

Java 2 platform güvenlik mimarisi aşağıdaki amaçlar için geliştirilmiştir:

- Son derece iyi erişim kontrolü
- Güvenlik politikaları şekillendirilebilir
- Ek erişim kontrol yapıları kolaydır
- Güvenlik ekleri bütün Java programlarını denetler

3.3.3 Java Güvenlik Ekleri

Sun tarafından Java için geliştirilmiş güvenlik ekleri vardır, örneğin *Java Cryptography Extension* (JCE), *Java Authentication and Authorization Service* (JAAS) ve *Java Secure Socket Extension* (JSSE).

Java Secure Socket Extension (JSSE) Java paketi olup, güvenli internet iletişimi kurmakta kullanılır. GSK'in Java versiyonu ve Transport Layer Security (TLS) protokolünü gerçekler ve veri şifreleme, sunucu onaylama, mesajın veri bütünlüğünü sağlama ve de istemci onaylamayı opsiyonel olarak sağlayan fonksiyonları vardır. JSSE, geliştirmeciler tarafından istemci ve sunucu arasında güvenli veri alışverişleri sağlamakta herhangi bir protokolle kullanılır, örneğin HTTP, Telnet, ve TCP/IP üstünde FTP [Java™ 2 SDK, 2001].

3.3.4 GSK

Güvenli Soket Katmanı (GSK) Protokolü

Secure Sockets Layer (GSK) yaygın olarak web tabanlı uygulamalarda kriptoloji protokollerini gerçeklemede kullanılır. GSK ağlarda güvenli iletişim sağlamak için çeşitli kriptografik yöntemler kullanır.

GSK 'Netscape' tarafından 1994 yılında geliştirildi, ve de sonra bir standart haline geldi. Şu an IETF (Internet Engineering Task Force) tarafından kontrol edilmektedir ve ismide TLS (Transport Layer Security) olarak değişmiştir. Ocak 1999'ta TLS 1.0 piyasaya en son sürüm olarak çıkmıştır.

Niçin GSK?

Aşağıdaki üç sebepten dolayı önemli bilgileri ağ üzerinde aktarmak güvensiz olabilir:

- Her zaman için kiminle haberleştiğinizden emin olamazsınız.
- Veriler yetkisi olmayan üçüncü kişiler tarafından okunabilir, bunlar atakçılar olarak bilinirler..
- Atakçılar verileri ele geçirirse, veri alıcıya ulaşmadan önce içeriğini değiştirebilirler.

GSK bu problemlerin hepsine çözüm getirmektedir. Opsiyonlu olarak her iki tarafada diğer tarafın kimliğini onaylama mekanizması sağlar. Tarafların birbirini onaylamasından sonra, GSK güvenli iletişim için, şifreli haberleşmeyi sağlar. Taraflar arasında şifreli haberleşme gizliliği ilk planda sağlamakla beraber, ikinci olarakta adreslemeyi yapar. Ayrıca şifreleme algoritması özetleme fonksiyonu kullanır, bu da verilerin iletişim esnasında değişip değişmediğini kontrol için kullanılır. Bu güvenlik özet fonksiyonu veri bütünlüğünü sağlar.

Öncelikle onaylama sağlandıktan sonra, GSK gizlilik ve veri bütünlüğünü kullandığı şifreleme algoritmalarıyla sağlar.

GSK' in Çalışma Mekanizması

GSK farklı kriptografik yöntemler kullanmaktadır. GSK açık anahtar kriptolojisini onaylama mekanizmasında, gizli anahtar kriptolojisini ve sayısal imzayı gizlilik ve data bütünlüğünü sağlamakta kullanmaktadır.

GSK kullanarak oluşturulan haberleşme istemci ve sunucu arasında bilgi değişimi ile başlar, buna GSK el sıkışması denir.

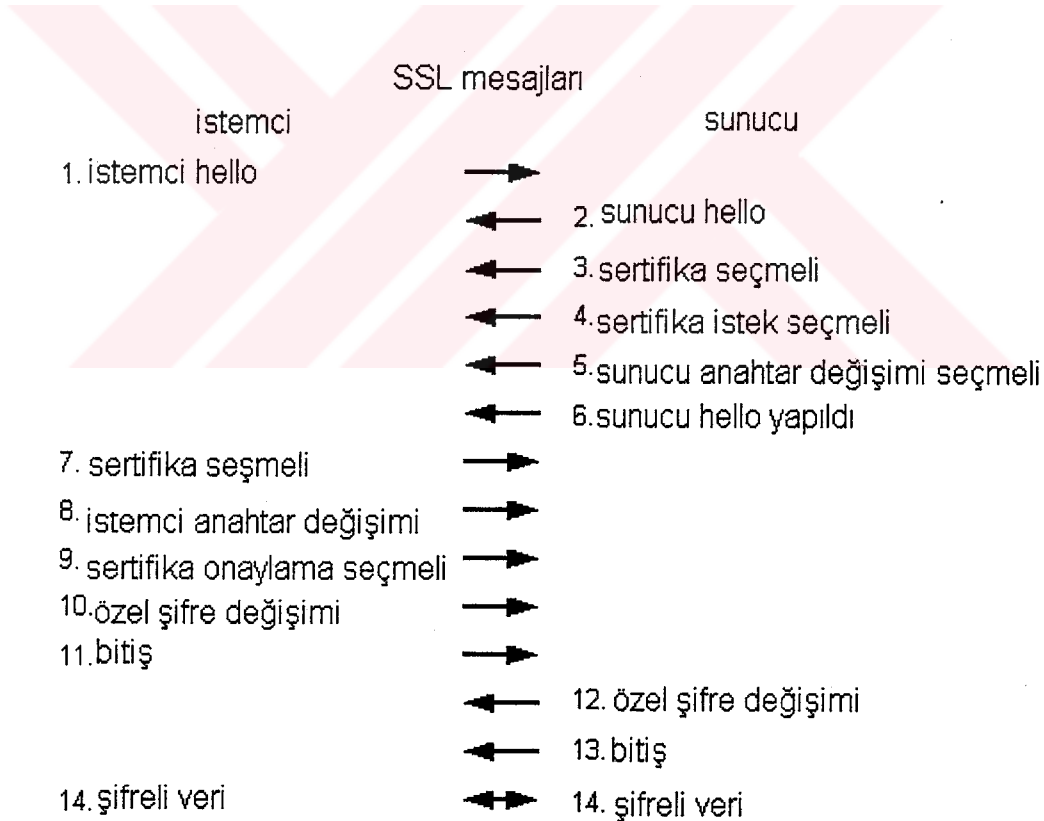
GSK el sıkışmasının üç tane ana amacı vardır:

- Şifreli ortamı yapmak
- Otorite onaylaması yapmak (opsiyonel)
- Bilgi güvenliğini kurmak [Ratnasingham, 1998]

GSK Protokolü

Daha önceki bölümlerde GSK el sıkışması üst seviyede anlatılmıştır, mesajlaşmanın istemci ve sunucu arasında şifreli ve güvenli olarak değişimlerinden söz edilmişti. Bu bölümde biraz daha detaylı anlatılacaktır.

“GSK Mesajları” figüründe GSK el sıkışmasında sırasında değiştirilen mesajların akış şeması gösterilmektedir. Sadece kesin durumlarda gönderilen mesajlar aşağıda seçmeli olarak belirtilmiştir. Aşağıda ki figürde bütün GSK mesajları gösterilmiştir:



Şekil 3.3 GSK Mesajları Yukarıda ki Sırada Gönderilir

1. **istemci hello** – İstemci sunucuya içerisinde desteklediği en son versiyon olan GSK bilgilerini ve kriptografik özelliklerinin listesini gönderir. Bu liste kriptografik algoritmaları ve anahtar uzunluklarını içerir.

2. **sunucu hello** – Sunucu GSK'in en yüksek versiyonunu ve en iyi kriptografik özellikleri seçer ve bunu istemciye gönderir.
3. **sertifika** – Sunucu istemciye sertifika veya sertifika zinciri gönderir. Sertifika sunucunun açık anahtarı ile başlar ve sertifika makamının kök sertifikası ile biter. Bu mesaj seçmeli olmakla beraber sunucu onaylama istendiğinde gereklidir.
4. **sertifika istek** – Eğer sunucu istemciyi onaylamaya gerek duyarsa, sertifika istek mesajı gönderir.
5. **sunucu anahtar değişimi** – Sunucu istemciye 3 nolu maddedeki bilgi anahtar değişimi için yeterli gelmezse sunucu anahtar değişimi mesajı gönderir.
6. **sunucu hello yapıldı** – Sunucu istemciye ilk görüşme mesajlarının bittiğini söyler.
7. **sertifika** – Eğer sunucu isterse, istemci kendi setifikasını sunucuya gönderir.
8. **istemci anahtar değişimi** – İstemci simetrik şifrelemede kullanılacak anahtar için bilgileri oluştururlar. RSA için, istemci bu anahtar bilgilerini sunucunun açık anahtarı ile şifreler ve sunucuya gönderir.
9. **sertifika onay** – u mesajın amacı sunucunun istemciyi onayladığının bitimi içindir. Bu mesaj kullanıldığında, istemci bilgilerini kriptografik özetleme fonksiyonuyla imzalayarak gönderir. Sunucu bu bilgiyi istemcinin açık anahtarı ile açtığında istemciyi onaylamaya hazırdır.
10. **özel şifre değişimi** – İstemci sunucuya gönderdiği bu mesajla şifreli moda geçmeye hazır olduğunu bildirir.
11. **bitiş** – İstemci sunucuya güvenli haberleşmeye başlamaya hazır olduğunu bildirir.
12. **özel şifre değişimi** – Sunucu istemciye mesajla şifreli moda geçmeye hazır ol der.
13. **bitiş** – Sunucu istemciye güvenli habrleşmeye başlamaya hazır ol der. Bu mesaj da GSK el sıkışmasının sonudur.

şifreli veri – İstemci ve sunucu simetrik şifreleme algoritması, kriptografik özetleme fonksiyonu ve gizli anahtar ile haberleşmeye başlar [Java™ 2 SDK, 2001].



4. UYGULAMANIN TEKNİK ÖZELLİKLERİ VE DİĞER UYGULAMALARLA KARŞILAŞTIRILMASI

4.1 Yazılım Araçları

Uygulama temel olarak üç ana bölümde incelenebilir. EVD mesajlaşma sistemi ve bununla oluşturulmuş uygulama, uygulamada ki verilerin saklanması için veritabanı ve de mesajlaşmada gerekli güvenliği sağlayan kısım.

Uygulama genel olarak ağ üzerinde ve nesneye yönelik olduğundan bu şartları sağlayacak, gerektiğinde esnek olabilecek, her ortamda çalışabilecek bir dilde yazılması gerekmektedir, çünkü tam bir farklı standartlarla da gerektiğinde haberleşebilmesi için esnek yapıya sahip olmalıdır. Bu özellikleri genel kapsayan dil Java olduğundan bu dil kullanıldı.

Veritabanı uygulamaları için Oracle veritabanı kullanılmıştır. Oracle' ın güvenilir, büyük veritabanlarını desteklemesi bu seçimde rol oynamıştır. Bu uygulamada kullanılan yazılım araçları ve versiyonları aşağıda belirtilmiştir:

4.1.1 Oracle 8i

Veritabanı olarak “Oracle Enterprise Manager Release 2.2.0.0.0 Production “ kullanılmaktadır.

4.1.2 Java ve JBuilder6

Yazılım dili olarak “Java™ 2 SDK, Standard Edition Version 1.3.0_02” ve de yazılım geliştirme editörü olarak “Borland JBuilder6 “ kullanılmıştır.

4.2 Uygulamanın Güvenlik Özellikleri

EVD gerçekleştirilirken karşımıza çıkan güvenlik problemleri daha öncede belirtildiği gibi bütünlük, onaylama, reddedemezlik ve gizliliklerdir.

İki yöntem kullanılarak bu problemler çözülmüştür. GSK kullanılarak bütünlük, onaylama ve gizlilik sağlanmıştır, sayısal imzalama yöntemiyle de reddedemezlik sağlanmıştır.

Java'da bu işlemleri yapabilmek için dilin kendi kütüphaneleri kullanılmıştır. Bu kütüphanelerin kullanılması için ilkönce konfigürasyon dosyasında bir takım işlemler yapılmıştır, bu dosya `JDK_HOME\jre\lib\security\java.security` dosyasıdır ve yapılan işlemlerde aşağıda verilmiştir. Burada güvenlik sağlayıcılar belirtilmiştir, aksi takdirde bu özellikleri kullanamayız.

```
security.provider.1=sun.security.provider.Sun
security.provider.2=com.sun.net.ssl.internal.ssl.Provider
security.provider.3=com.sun.rsa.jca.Provider
security.provider.4=org.bouncycastle.provider.BCJSSE
security.provider.5=com.sun.crypto.provider.SunJCE
```

Şekil 4.1 Konfigürasyon Ayarları

4.2.1 GSK

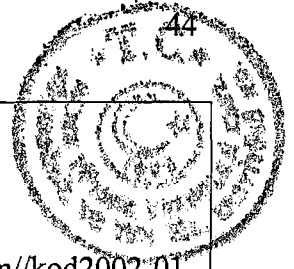
JSSE (Java Secure Socket Extension) Java paketi olup, güvenli ağ iletişimi kurmakta kullanılır. GSK'in Java versiyonu ve TLS (Transport Layer Security) protokolünü gerçekler.

Bu uygulamada da bu paketin bu özellikleri kullanılmaktadır. İlkönce hem istemci hem de kullanıcının GSK haberleşmesinde kullanacağı GSK konteksti oluşturulmalıdır.

Bu işlem aşağıdaki program parçasındaki sırada oluşturulmaktadır;

İlkönce değişken boş değerli tanımlanır, anahtarlar yüklenir, sertifika tipi belirlenir, GSK haberleşmesi yapacağı belirtilir ve bu bilgilerin içinde olduğu konteksts döndürülür.

```
protected com.sun.net.ssl.SSLContext getSSLContext(){
```



```
com.sun.net.ssl.SSLContext sslContext = null;

KeyStore keyStore = KeyStore.getInstance("JKS");

keyStore.load(new FileInputStream("J://proje//tezprogram//kod2002-01-02//kod//testkeys"), "passphrase".toCharArray());

KeyManagerFactory keyManagerFactory =
KeyManagerFactory.getInstance("SunX509");

keyManagerFactory.init(keyStore, "passphrase".toCharArray());

TrustManagerFactory trustManagerFactory =
TrustManagerFactory.getInstance("SunX509");

trustManagerFactory.init(keyStore);

sslContext = com.sun.net.ssl.SSLContext.getInstance("SSL");

sslContext.init(keyManagerFactory.getKeyManagers(),
trustManagerFactory.getTrustManagers(), null);

return sslContext;

}
```

Şekil 4.2 GSK Konteksti

Bu bağlam oluşturulduktan sonra, hem istemcide hem de sunucuda kullanılmak üzere hazırdır. Sunucuda kullanım şekli aşağıdaki gibidir, bir tane sunucu soketi oluşturulur ve bu bağlamanın özellikleri sokete aktarılır, sunucu yeni istemciler kabul etmeye hazırdır.

```
public void startServer() {

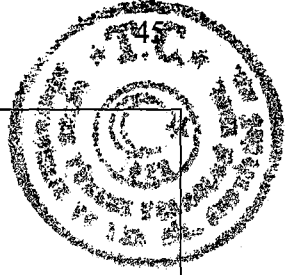
.....

ServerSocketFactory
serverSocketFactory=getSSLContext().getServerSocketFactory();

serverSocket = serverSocketFactory.createServerSocket( serverSocketPort);

.....

}
```



```
Socket socket;  
.....  
socket = serverSocket.accept();  
clientService client = new clientService( socket );  
.....  
}
```

Şekil 4.3 Sunucu Oluşturulması

İstemcide de sunucuda olduğu gibi daha önceden oluşturulan bağlamın özellikleri istemci sunucusuna aktarılır, istemci sunucuya bağlantı yapmaya hazırdır.

```
public void clientStart(){  
.....  
SocketFactory socketFactory = getSSLContext().getSocketFactory();  
clientSocket = socketFactory.createSocket( serverName,serverSocket );  
.....  
clientSocket = new Socket( serverName,serverSocket );  
.....  
}
```

Şekil 4.4 İstemci Oluşturma

4.2.2 Sayısal İmzalama ve Onaylama

Sayısal imzalama yapan sınıf ve metodları aşağıda verilmiştir. 'keyGen' fonksiyonu sayısal imzalama anahtarını ve buna bağlı olarak bu imzayı onaylayacak

onay anahtarını üretir. İmzalama yapılan metot 'signObj', onaylamasının yapıldığı metotta 'verifyObject' dir.

```
public class sign {
    public KeyPair keyGen(){
    }
    public SignedObject signObj(Object data,KeyPair pair){
    }
    public void verifyObject(SignedObject data,KeyPair pair){
    }
}
```

Şekil 4.5 İmzalama Fonksiyonları

İmzalamada imzası alınacak nesne alınır, özel anahtar ve imzalama algoritmasının özelliği girilir, sonuç olarak nesne ve imzasını içeren imzalanmış nesne döner.

```
SignedObject so = new SignedObject((Serializable)data,
    (PrivateKey)pair.getPrivate(),
    Signature.getInstance("SHA1withDSA"));
```

Şekil 4.6 İmzalı Obje

Onaylama işleminde, imzalanmış nesnenin onaylama metodu çağrılır, açık anahtar ve algoritmanın özellikleri girilir, onaylanır ise doğru sonucunu onaylanmazsa yanlış sonucunu döndürür.

```
boolean verifies = data.verify( pair.getPublic(),
    Signature.getInstance("SHA1withDSA"));
```

Şekil 4.7 Objeyi Onaylama

4.3 Uygulamanın Genel Özellikleri

4.3.1 Sunucu ve İstemci

İstemci ve sunucunun mesajlaşmasının dayandığı temel soket üzerinden nesne alış-verişidir. İki tarafta da kullanılan yöntemler aynıdır.

Mesaj gönderme de kullanılan yöntem aşağıda gösterilmiştir:

```
public void sendData(Object o){
    .....
    output.writeObject(o);
    .....
}
```

Şekil 4.8 Veri Gönderme

Mesaj almada kullanılan yöntemde aşağıda gösterilmiştir:

```
public void getData(){
    .....
    o = input.readObject();
    .....
}
```

Şekil 4.9 Veri Alma

Sunucu birden çok bağlantı kabul ederken, her istemci için yeni bir 'clientService' sınıfı oluşturmakta ve de bunlar için bir 'thread' atamaktadır, bundan sonra bu istemci ile ilgili işleri bu 'thread' halletmektedir, bu işlemlerin başlangıç aşaması aşağıda gösterilmektedir:

```

socket = serverSocket.accept();
clientService client = new clientService( socket );
clientList.addElement( client );
client.start();

```

Şekil 4.10 İstemci Servisi Oluşturma

4.3.2 EVD Nesnesi

Mesajlaşmada ve de uygulamanın genelinde kullanılan nesne EVD nesnesidir. EVD mesajı UN bölümü(UN Segments)ve Veri bölümünden(Data segments) oluşur. Bölümün (SEGMENT)içeriği: Bölüm etiketi, Basit veri elemanı, Karma(Composite) veri elemanı veya ikisidedir.

Bölüm etiketi içeriği: Bölüm kodu ve, açıkça belirtiyorsa, tekrar eden ve art arda değerlerdir.

4.3.3 Veritabanı

Veritabanı olarak Oracle kullanılmaktadır, Java'dan Oracle' a bağlanırken 'oracle.jdbc.driver.OracleDriver' sürücüsü kullanılmaktadır, bağlanma adreside 'jdbc:oracle:thin:@localhost:1521:EVD' dir. Kullanıcı adı 'EDI' ve parolası da 'EDI' dir.

Veri tabanında kullanılan tablolar, içerikleri ve amaçları aşağıda açıklanmıştır:

KULLANICI:

Sitemdeki kullanıcıların kendilerine ait kullanıcıya özel numaralarını, isim, soy isim ve sisteme giriş isimlerini tutan veri tabanı tablosudur.

SIFRE:

Sistemdeki kullanıcıların kendi özel numaralarına göre şifrelerini tutan tablodur.

ALIM:

Sunucunun stokları azaldığında, alım yaparsa bu işlemlerin tutulduğu tablodur, burada hangi tip ürün, hangi markadan olduğu, alış satış fiyatları ve de tarihi tutulur.

CIHAZTAB:

Bu tabloda sunucunun stoklarında bulunan cihazların, tipleri, markaları, satış fiyatları ve de sayısı tutulur.

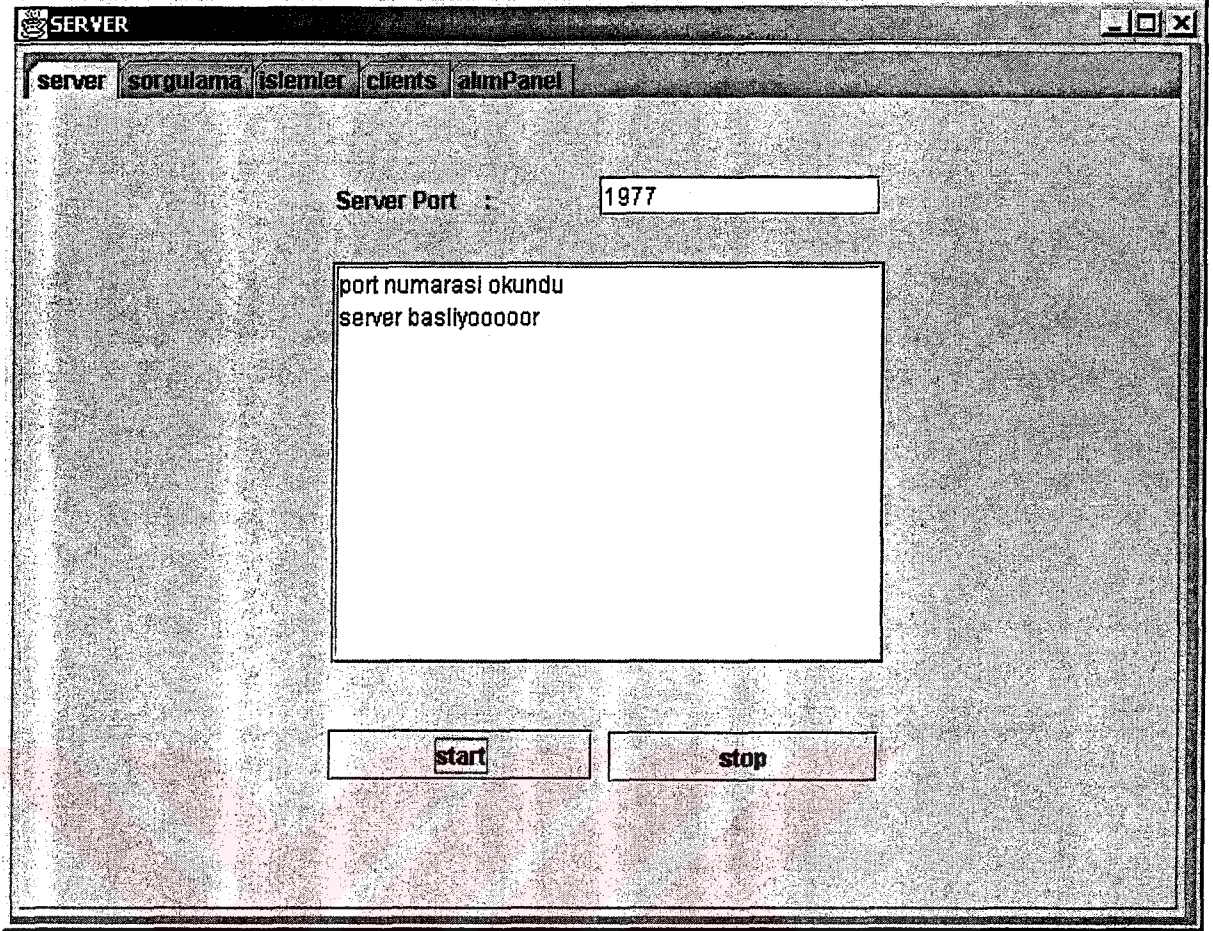
SATIM:

Bu tabloda satılmış olan malların tipleri, markaları, adetleri, satış fiyatı, hangi tarihte satıldığı ve kime satıldığını belirlemek için satın alanın numarası tutulur.

4.4 Örnek Uygulama

4.4.1 Sunucu

Sunucuda yapılabilecek uygulamalar bu bölümde anlatılacaktır. İstemcilerin bağlanabilmesi için, sunucunun başlatılması ilk iştir, sunucu istenen porttan bağlantıları kabul eder, istenilen bir vakitte sunucu kapatılabilir.



Şekil 4.11 Sunucu Mönüsü

Sunucuda yapılabilecek diğer bir uygulamada istemciler tarafından yapılmış olan işlemleri görebilmektir, kullanıcı ismini sol kolondan seçildiğinde, sağ taraftaki kolonda o kişinin şimdiye kadar yaptığı işlemler tarih sırasına göre dizilmiş olarak görünürler.

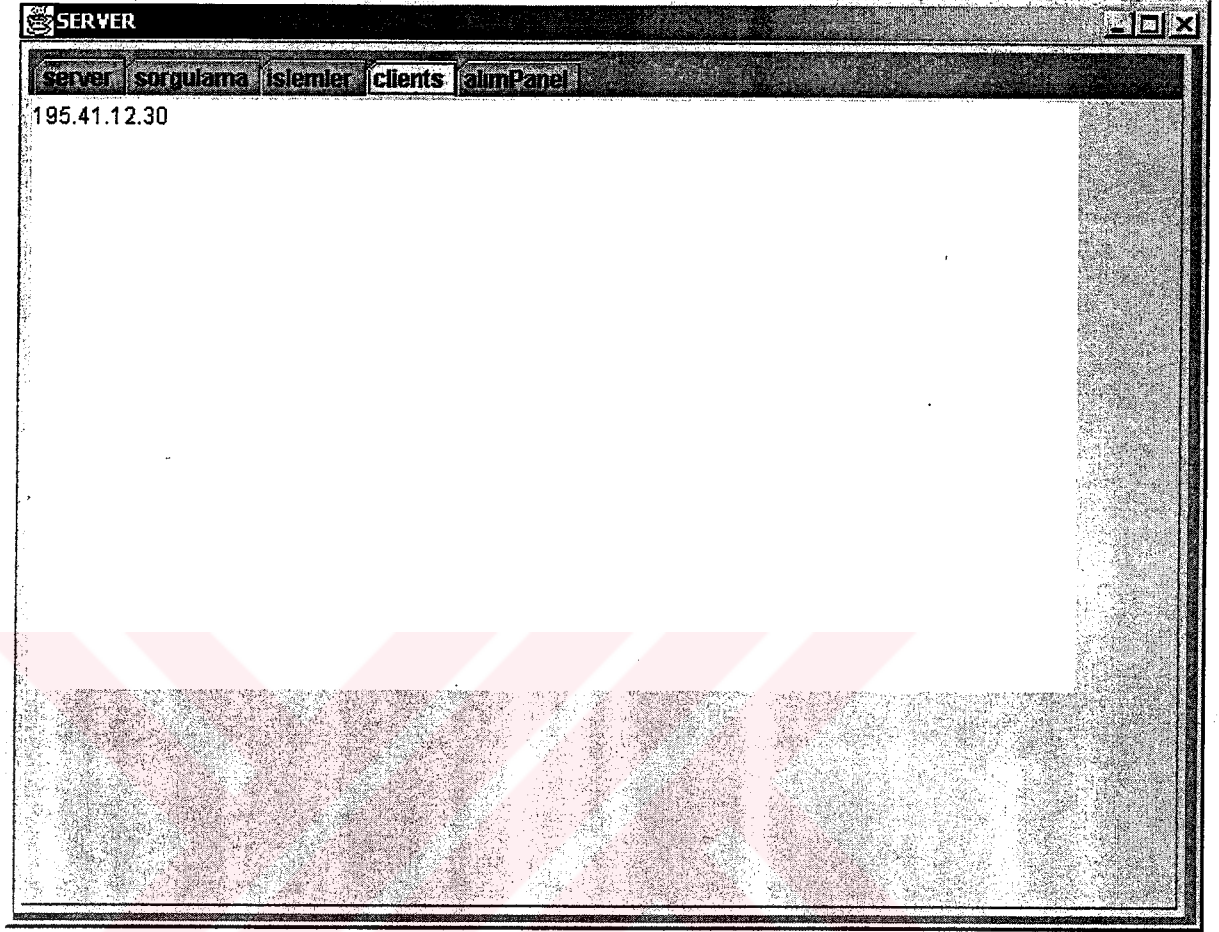
SERVER

server sorgulama islemler clients alimPanel

	ALICI	TIP	MARKA	ADET	FIAT
1	ahmet	ekran	sony	3	3575.20
2	ahmet				
3	veli				

Şekil 4.12 Sorgulama

Sunucuya baęlı istemciler 'clients' tuşuna basılarak görülebılır.



Şekil 4.13 Baęlı İstemciler

Stoklarda az kalan veya biten bir para alındığı zaman bu işlem veritabanına alım panelinden girilir, bu panelde girilecek bilgiler yazılmıştır.

The screenshot shows a software window titled 'SERVER'. The menu bar includes 'server', 'sorgulama', 'işlemler', 'clients', and 'alımPanel'. The 'alımPanel' is the active window. It contains a form with the following fields and controls:

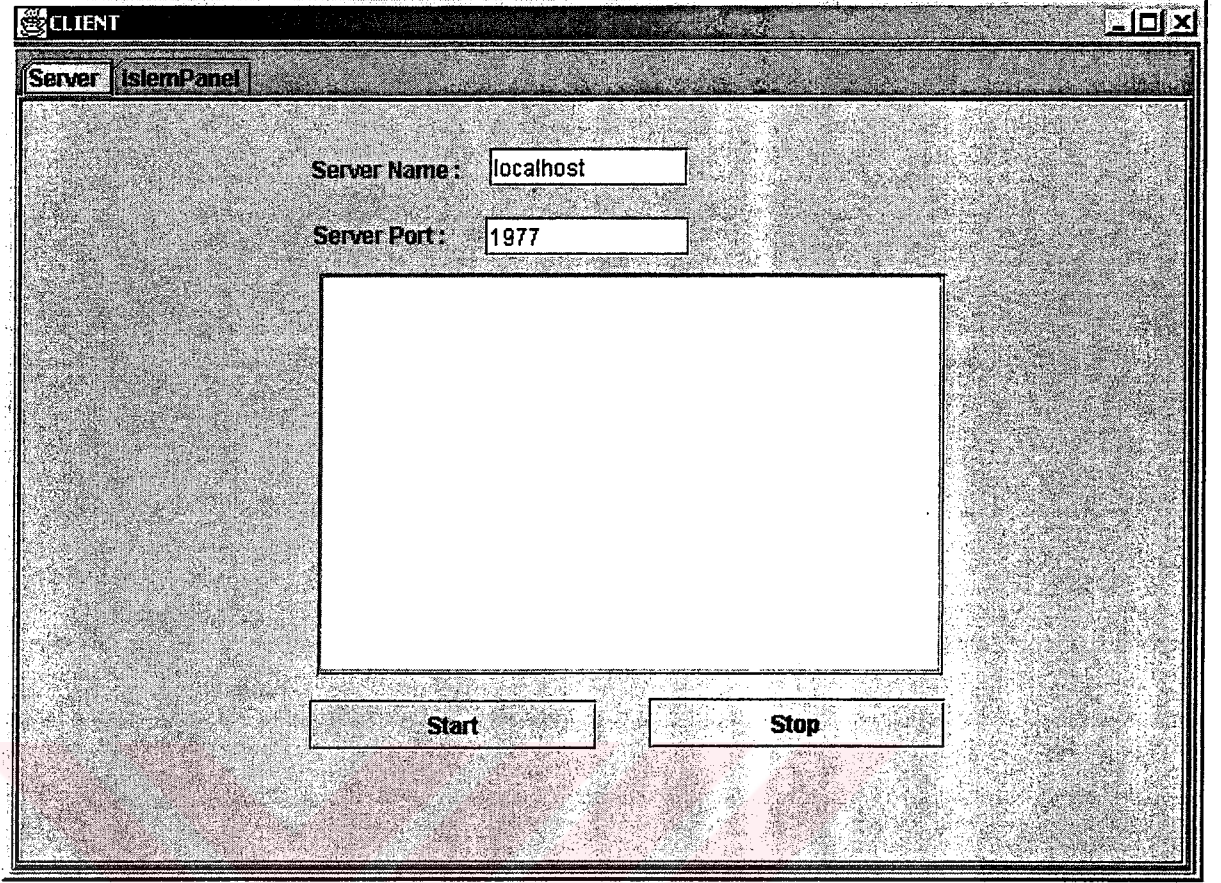
- tip**: A dropdown menu.
- marka**: A dropdown menu.
- alım fiyat**: A text input field.
- satım fiyat**: A text input field.
- adet**: A text input field.
- tarih**: A text input field.
- ekle**: A button located below the input fields.

Şekil 4.14 Alım İşlemleri

4.4.2 İstemci

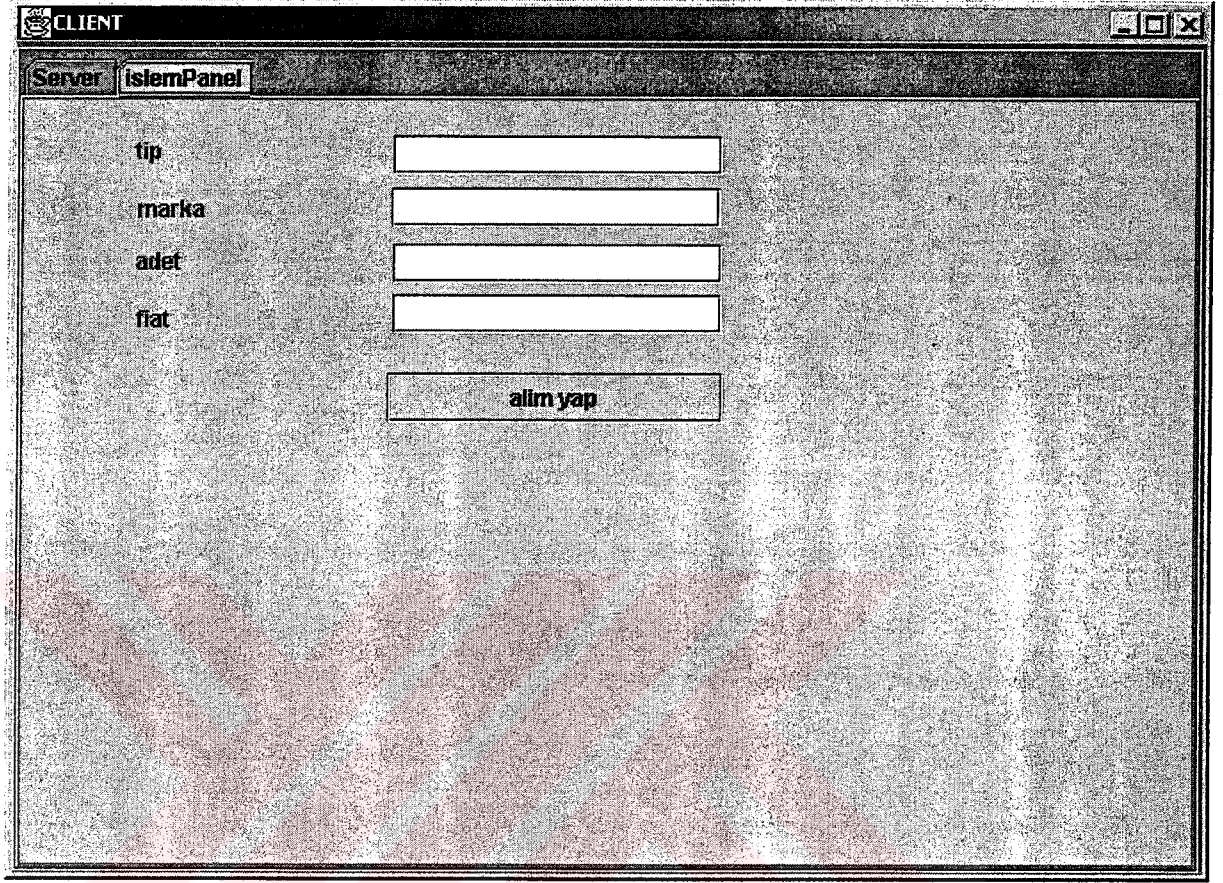
İstemcide yapılabilecek işler aşağıda belirtilmiştir.

Kullanıcı sunucuya, sunucunun adresini ve de port numarasını vermek suretiyle bağlanır.



Şekil 4.15 İstemci Başlatma Mönüsü

İstemci stoklarında herhangi bir mal biteceği veya azaldığı vakit sunucudan alım işlemini yapabilir.



The screenshot shows a window titled "CLIENT" with a tabbed interface. The active tab is "İşlemPanel". Inside the panel, there are four text input fields stacked vertically, each preceded by a label: "tip", "marka", "adet", and "fiat". Below these fields is a button labeled "alın yap". The window has standard Windows-style window controls (minimize, maximize, close) in the top right corner.

Şekil 4.16 İstemci İşlemleri

4.5 Diğer Sistemler ile Karşılaştırılması

Bu bölümde bu uygulama ile diğer iki uygulamanın güvenlik yönünden karşılaştırılması yapılmaktadır. Güvenlik işleri olarak bütünlük, onaylama, reddedemezlik, gizlilik belirlenmiştir, ayrıca kodda güvenlik işleri ile ilgili olarak kodun ileriki uygulamaları destekleyebilmesi açısından esneklik ve ortamdan bağımsızlığı da ele alınmıştır.

Tablo 4.1 EVD Sistemlerinin Karşılaştırılması

<i>EVD sistemi</i>	<i>Bütünlük</i>	<i>onaylama</i>	<i>reddedemezlik</i>	<i>gizlilik</i>	<i>esneklik</i>	<i>Ortamdan bağımsızlık</i>
<i>Uygulama</i>	<i>GSK ile halledilir</i>	<i>GSK ile halledilir</i>	<i>Sayısal imzalama DSA algoritması ile halledilir</i>	<i>GSK ile halledilir</i>	<i>sağlanır</i>	<i>sağlanır</i>
<i>UN/EDI FACT</i>	<i>Özetleme algoritması</i>	<i>MAC kullanılır</i>	<i>Sayısal imzalama</i>	<i>Simetrik şifreleme</i>	<i>Değildir</i>	<i>Değildir</i>
<i>X400</i>	<i>Halledilir</i>	<i>halledilir</i>	<i>halledilir</i>	<i>halledilir</i>	<i>Değildir</i>	<i>Değildir</i>
<i>X435</i>	<i>Halledilir</i>	<i>halledilir</i>	<i>halledilir</i>	<i>halledilir</i>	<i>Değildir</i>	<i>Değildir</i>
<i>X500</i>	<i>Halledilir</i>	<i>halledilir</i>	<i>halledilir</i>	<i>Kendi halletmez, X400 kullanır</i>	<i>Değildir</i>	<i>Değildir</i>
<i>EDI5</i>	<i>Halledilir</i>	<i>halledilir</i>	<i>halledilir</i>	<i>halledilir</i>	<i>Değildir</i>	<i>Değildir</i>
<i>CRYPT IT-EDI5</i>	<i>Halledilir</i>	<i>halledilir</i>	<i>halledilir</i>	<i>halledilir</i>	<i>Değildir</i>	<i>Değildir</i>
<i>ETEBA C5</i>	<i>Halledilir</i>	<i>halledilir</i>	<i>halledilir</i>	<i>halledilir</i>	<i>Değildir</i>	<i>Değildir</i>

4.6 Güvenlik Analizi

SSL protokolü istemci ve sunucu arasında güvensiz ortamda güvenli bir iletişim kurmak için tasarlanmıştır. Bu kısımda SSL protokolünün ataklara karşı nasıl direndiği anlatılacaktır.

4.6.1 El Sıkışma Protokolü

El sıkışma protokolü şifreleme özelliklerini seçmek ve yönetici gizliyi üretmek ile sorumludur, bu parametreler kriptolu haberleşme için gerekli değerlerdir.

4.6.2 Onaylama ve Anahtar Değişimi

SSL üç çeşit onaylamayı destekler: iki tarafında onaylanması, sunucunun onaylanması ve de iki tarafında onaylanmaması. Sunucu onaylandığı zaman iletişim kanalı ortadaki adam atağına karşı korunmuş olur.

Sunucu onaylanacağı zaman , sunucunun sertifika mesajı sertifika otoritelerince kabul edilebilir bir sertifika zinciri oluşturmalıdır. Aynı şekilde istemci de sunucuya kabul edilebilir bir sertifika sunmalıdır. İki tarafta diğer tarafı onaylamakla sorumludur.

Anahtar değişiminin esas amacı yönetici gizli öncesini oluşturmaktır. Yönetici gizli öncesini yönetici gizli oluşturmakta kullanılır. Yönetici gizli bitiş mesajı, şifreleme anahtarları ve makine ulaşım kodu üretmek için gereklidir. Bitiş mesajı gönderilince, iki tarafta doğru yönetici gizli öncesini aldığından emin olur.

4.6.3 Anonim Anahtar Değişimi

Tamamen anonim olan oturumlar RSA, Diffie-Hellman kullanılarak oluşturulurlar. RSA ile istemci yönetici gizli öncesini sunucunun anahtar değişim mesajı ile gelen sertifikalanmamış açık anahtarı ile şifreler ve sonuç istemcinin anahtar değişim mesajı ile iletilir. Hattı dinleyenler sunucunun özel anahtarını bilmediklerinden şifresini açmaları mümkün değildir.

4.6.4 RSA Anahtar Değişimi ve Onaylaması

RSA'da anahtar değişimi ve onaylama birlikte olur. Açık anahtar sunucunun sertifikasında veya geçici anahtar sunucu anahtar değişimi mesajı ile gönderilir. Geçici açık anahtar kullanıldığında sunucunun anahtarı ile imzalanır. Sunucu geçici açık anahtarını birden çok bağlantıda kullanır. Sunucunun sertifikası onaylandıktan sonra, istemci sunucunun açık anahtarı ile yönetici gizli öncesini şifreler. Sunucu bunun şifresini açıp aldığı anda bitiş mesajı üretir.

4.6.5 El Sıkışma Protokolüne Yapılan Atakların Fark Edilmesi

Atak yapan kişi el sıkışma değişimini etkilemeye çalışabilir, ve burada değişik algoritma seçtirmeye çalışabilir. Çünkü çoğu algoritma 40-bit şifrelemeyi destekler. Bu atak için, atakçı el sıkışma mesajlarından bazılarını değiştirmek zorundadır. Eğer bu olursa istemci ve sunucu el sıkışma mesajları için değişik değerler hesaplayacaktır. Bundan dolayı taraflar birbirlerinin bitiş mesajlarını kabul etmeyeceklerdir. yönetici gizli olmadan atakçı bitiş mesajını yeniden yapamaz, yapsa da hemen fark edilir.

4.6.6 Yeniden Oturum Kurmak

Yeniden oturum açıldığında, oluşturulan özel değerler bir önceki bağlantıdan farklıdır. Atakçılar bilinen şifreleme anahtarlarını yeni bağlantı oluşturma sırasında kullanamazlar, bunları kullanabilmeleri için özetleme algoritmalarını kırmaları gerekir.

Oturum iki tarafta tamamen anlaşıncaya kadar başlamaz. Eğer iki taraftan birisi herhangi bir şüphe duyarsa, tam el sıkışma yöntemine gider. Bir oturumun hayatı en fazla 24 saat sürer.

4.6.7 Uygulamanın Verilerinin Korunması

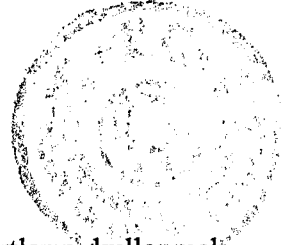
Yönetici gizli özeti ilk mesajlarla birlikte alınır ve bu değer özel şifreleme anahtarı için ve de mesaj onaylama kodu için kullanılır. Bu her bağlantıda ayrı olarak yapılır. Giden mesaj, gitmeden önce mesaj onaylama kodu ile korunur, mesaj onaylama kodu özel değerler ile hazırlanır. Bir tarafın mesajı diğer tarafın mesajı ile birleşemez, çünkü her birinin ayrı mesaj onaylama kodu vardır. Mesaj içeriğini değiştirme atakları için şifreleme algoritmasını ve mesaj onaylama kodunu kırması gerekir, bunun içinde şifreleri ve özel bilgileri bilmelidir.

4.6.8 Sonuç

SSL' in istemci ve sunucu için güvenli bağlantı sağlaması için anahtarların, sistemlerin ve uygulamaların güvenli olması lazımdır. Ayrıca yazılım güvenlik

hatalarından arınmış olmalıdır. Bu uygulamada da sistem ve uygulama güvenli bir şekilde yapılmış olup güvenlik hatalarından da arındırılmıştır.





5. SONUÇ VE ÖNERİLER

Bu makalede görüldüğü gibi, EDIFACT' in güvenlik standartlarını kullanmak güvenlik servislerini büyük ölçüde karşılamaktadır, fakat gerçekleştirmedeki zorluk, süreklilik, ve ortama bağlı olmak karşımıza büyük problemler olarak çıkmaktadır. GSK kullanmak, bunlara göre en güvenli metodudur, çünkü bu teknoloji bilhassa internette kullanılmakta ki bu ortam özel ağdaki ortamdan çok daha fazla risk içermektedir. Bununla ilgilendirirsek GSK daha güvenlidir, fakat GSK onaylama mekanizması sağlamamaktadır, bunu sağlamakta Java'nın onaylama mekanizması kullanılmaktadır. Ve programın tamamı sadece Java kullanılarak gerçekleştirilmiştir, bu süreklilik, ortamdaki bağımsız olmayı sağlamakta, ayrıca bu program istemci sunucu mantığıyla yazılmış olmasına rağmen Java'nın esnek yapısı sayesinde web tabanlı EVD' ye dönüştürülebilir, ayrıca bazı değişikliklerle konfigüre edilebilir ve diğer Elektronik Ticaret programlarıyla uyumlu hale getirilebilir, örneğin xml tabanlı iş programlarında.

Sonuç olarak bu uygulama da güvenlik ile ilgili yapılan işler:

- GSK kullanılarak veri bütünlüğü sağlanmıştır
- GSK kullanarak onaylama sağlanmıştır
- Sayısal imzalama ile reddedemezlik sağlanmıştır
- GSK kullanarak gizlilik sağlanmıştır

Bunlara ek olarak yapılan işlerde:

- Kodda ve dolaylı olarak güvenlik işlerinde esneklik sağlanmıştır, uygulama değişik algoritmalar ve değişik yöntemler kullanmaya açıktır
- Uygulama ortamdaki bağımsızdır

Yapılanlarla beraber gelecekte yapılabilecek olan işlerde:

- Diğer elektronik ticaret programlarına uygun hale getirilebilir
- PKI programa eklenebilir

Tasarlanan uygulamada dezavantajları söylemek gerekirse:

- Sayısal imzalamalarda kullanılan şifreleme algoritmaları yavaştır.



KAYNAKLAR

- [1] *Banerjee , Snehamay and Golha, Damodar Y.,1995. Security issues in the EDI environment. Information Management & Computer Security, Vol. 3 No. 2, 1995, pp. 27-33 © MCB University Press Limited, 0968-5227*
- [2] Bottomley L., 1995. 'An Introduction to Electronic Data Interchange' *Network Notes* #6 ISSN: 1201-4338, Information Technology Services, National Library of Canada.
- [3] Data Encryption Standard (DES), 1999. DRAFT Federal Information Processing Standards Publication 46-3.
- [4] Digital Signature Standard (DSS), 1994. Federal Information Processing Standards Publication 186.
- [5] Dosdale, T., 1998. EDI security. Information Security Technical Report, Vol.3, No. 2 (1998) 98-110. Axion Services Ltd. And C. N. Rasmussen, Etrust Technologies Europe.
- [6] EaseIt homepage, The EDI and e-bussiness Security. <http://www.easit.fr/english>.
- [7] Electronic Commerce Agreement, Draft,1999-01-23-Rev1.
- [8] Fedorowicz, Jane, 2000. Discussion of adoption of Just-in-Time and Electronic Data Interchange systems and perceptions of Cost Management Systems effectiveness. . Information Security Technical Report, Vol.1, No. 4 (2000) 47-55.
- [9] Humpreys, Ted, 1997. EDI messaging security. Information Security Technical Report, Vol.4, No. 2 (1997) 38-53.
- [10] Jui-Lin Lu, Eric and Hwang, Rong-Ji, 2000. A distributed EDI model. Information Management Department, Chaoyang University of Technology, Taiwan. Information Management & Computer Security, Vol. 1 No. 2, 2000, pp. 57-63
- [11] Litterio, Francis, 1999. The Mathematical Guts of RSA Encryption. <http://www.std.com>.
- [12] Liu, Duen-Ren and Wu, I-Chin and Hsieh, Sung-Ting, 2001. Integrating SET and EDI for secure healthcare commerce. Information Management & Computer Security [2001] 25-35

- [13] Ratnasingham Pauline and Swatman, Paul, 1997. EDI security: a model of EDI risks and associated controls. *Information Management & Computer Security* 5/2 [1997] 63–71
- [14] Ratnasingham, Pauline, 1998. Internet-based EDI trust and security
Department of Information Systems. *Information Management & Computer Security* 6/1 [1998] 33–39.
- [15] Ratnasingham, Pauline, 1998. EDI Security: The Influences of the Trust on EDI Risks. *Computers & Security*, 17 (1998) 313-324
- [16] Ratnasingham, Pauline, 1998. Implicit Trust Levels in EDI Security. . *Computers & Security*, 17 (1998) 350-381
- [17] Rawlings, Jane, 1998. Electronic Commerce On The Internet – Part 1, Network Security.
- [18] Rubia, Montserrat and Cruellas, Juan Carlos, Technical description of X.509 and UN/EDIFACT certificates. *Network Notes* #6 ISSN: 1205-4238.
- [19] Secure Hash Standard, 1995. Federal Information Processing Standards Publication 180-1.
- [20] Un/Edifact Draft Directory, Part 4, 1990. United Nations Rules for Electronic Data Interchange for Administration, Commerce and Transport
- [21] Zhaou, Jianying and Gollman, Dieter, 1998. Certified Electronic Mail. Department of computer science Royal Holloway, University of London. *Information Management & Computer Security*, Vol. 1 No. 3, 1998, pp. 42-53

ÖZGEÇMİŞ

1977 yılında İzmir’de doğdu. İzmir Bornova Suphi Koyuncuoğlu Lisesini 1994’de bitirdi ve aynı sene Bilkent Üniversitesi Müh-Mim Fakültesi Bilgisayar ve Enformatik Mühendisliği bölümüne girdi. 1999 yılında lisans eğitimini tamamladı; aynı yıl Gebze Yüksek Teknoloji Enstitüsünde Elektronik Anabilim Dalında yüksek lisans öğrenimine başladı ve halen devam etmektedir.

1999’dan beri TÜBİTAK Ulusal Elektronik ve Kriptoloji Araştırma Enstitüsünde araştırmacı olarak görevine devam etmektedir.



EKLER

EK 1-EDI Standartlarının Formatları

1.1-ANSI X12

X12 (ve EDIFACT) standartlarının yapısı hiyerarşik olarak düşünülebilir. En başta basit olan veri yapısı kompleks veri yapısına dönüşür. Hiyerarşinin en üstünde bilgi ünitesi ticari ortaklar arasında değiştirilir.

Bu bölümde X12 standartlarının hiyerarşisi baştan sona incelenecek. EDIFACT'te farklı veri yapısı kullanmakla beraber, hiyerarşik yapı aynıdır.

X12 standartlarının temeli basit veri elemanları sözlüğüdür (simple data element dictionary). Basit yapı elemanı X12 standartlarında en küçük isimli parçadır. Vasıf, değer veya tanım belirtir. Örnek olarak basit veri yapısı şunları içerir:

- Fatura tarihi (an invoice date),
- Ağırlık kapasitesi (a weight capacity),
- Değiştirme oranı (an exchange rate),
- Riskli materyallerin sınıflandırması (a hazardous material classification), or
- Ölçme birimlerinin kodları (a unit of measurement code (e.g., pounds, dozens, cubic feet, gallons, etc.)).

X12 standartlarının 3. versiyonunda yeni veri elemanı sunuldu: composite data element. Composite data element tek isimli unsurları ifade eden basit yapıları elemanlar kümesidir.

Veri elemanları (data elements) fonksiyonel ilişkileriyle alakalı olarak, veri bölümleri (data segments) altında gruplaşırlar. Örneğin bir veri bölümü coğrafi konumdur, bu veri elemanları olarak il ismini, posta kodunu, ülke kodunu içerir. Veri bölümleri X12 bölüm dosyasında tanımlanmıştır, ve burada düzenli olarak veri elemanları bölümleriyle listelenmiştir.

Veri elemanları işlem bölümlerinin (transaction sets) altında gruplaşmıştır. İşlem bölümü ticari ortaklar arasında değiştirilen anlamlı en küçük bilgidir. Bu

yaygın bir iş dokümanını temsil eder, örneğin satın alma siparişi gibi. Birçok işlem bölümü üç kısma bölünmüştür. Başlık bölümü (heading area), bütün işlemle ilgili bilginin yerleştirildiği. Detaylı bölüm (detail area), bir tane büyük döngü için. Özet bölümü (summary area), bilgi sağlar, örneğin işlem bölümünde kaç tane veri bölümü kullanıldığı gibi

İşlem bölümü damgalı doküman ifade etse de, EDI işlemlerinde kullanılan bilgi ünitesi değildir. Örneğin, A şirketi B şirketine iki tane maliyet talebi ve beş tane satın alma siparişi gönderse, iki maliyet talebi bir fonksiyonel grup, diğer beş tane satın alma siparişi ayrı bir fonksiyonel grup olur. Bu fonksiyonel gruplar EDI değiş tokuşu (EDI interchange) adı altındaki kısımda toplanır. Bunlar ticari ortaklar arasında değiştirilir.

Bu fonksiyonel gruplar değişim başlığı ve kuyruğuyla sarmalanır. Başlık çeşitli kontrol bilgileri içerir, örneğin kimlik saptamak:

- Değişimin alıcısı ve göndericisi (the sender and receiver of the interchange),
- Değişimin tarihi ve zamanı (the date and time of the interchange),
- Değişimin standartı ve versiyonu (the standard and version of the interchange),
- Yetki ve güvenlik bilgisi (authorization and security information),
- Tekil kontrol numarası (a unique control number by which the interchange can be tracked).

Kuyruk değişimin sonlandırır, değişimdeki fonksiyonel grupların sayısını tutar, ayrıca tekil değişim kontrol numaralarını tekrar eder.

1.2-UN/EDIFACT

Genel UN/EDIFACT mesajı bölümlerin (segments) birleşiminden oluşup, basit değerler, kodlanmış değerler, veya karma yapılı basit, kodlanmış değerler taşırlar. Bu bölümlerden bazıları özel dikkat gerektirirler. Bu bölümler United Nations tarafından ilk iki karakteri UN olacak şekilde belirlenmiştir ve Annex B nin altında D422 de tanımlamışlardır.

Her bölüm üç karakterlik etiketle başlar. Artı işareti karma veri elemanlarının ayırıcı (COMPONENT DATA ELEMENT SEPARATOR) olarak, çift kolon veri

elamanı ayırıcı (DATA ELEMENT SEPARATOR), ve doğrulama işareti (tick) bölümün sonunu (SEGMENT TERMINATOR) belirler. Bu karakterlerden biri özel manada kullanılacaksa bildirme sinyali (RELEASE INDICATOR) olarak soru işareti konur. Numaralar ondalık sisteme (DECIMAL NOTATION) uymak için kodlanır, nokta en çok kullanılır. UNA bölümü (segment) isteğe bağlı olarak bu karakterleri tekrar tanımlamak için mesaja başlık olabilir.

EK 2- Program Kodları

2.1-Client

```
package client;

import java.net.Socket;
import java.io.InputStream;
import java.io.OutputStream;
import java.io.IOException;
import gui.clientgui;
import java.net.SocketException;

import java.security.*;
import java.io.FileInputStream;
import com.sun.net.ssl.*;
import javax.net.SocketFactory;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.io.InterruptedIOException;
import java.io.EOFException;
import java.util.Hashtable;
import edi.message;
import java.security.KeyPair;
import security.sign;
import security.Certificate;
import java.security.PublicKey;

public class client extends Thread{

    public Socket clientSocket;
    public InputStream clientInputStream;
    public OutputStream clientOutputStream;
    private byte[] buffer=new byte[1024];
    private boolean alive = true;
    String serverName;
    int serverSocket;
```

```

private ObjectInputStream input;
private ObjectOutputStream output;
protected Hashtable certificates = new Hashtable();
private sign keypair = new sign();
private KeyPair kp;
private Certificate cer = new Certificate();
PublicKey serverPK;
message mes = new message();
public client(){
    kp = keypair.keyGen();
    cer.setPublicKey(kp.getPublic());
}

public KeyPair getKP(){
    return kp;
}
public sign getSignKP(){
    return keypair;
}

public void setClient(String serName, int serSocket) {
    serverName = serName;
    serverSocket = serSocket;
}

//istemci oluřturur
public void clientStart(){
    try {
        if ( true ){
            SocketFactory socketFactory = getSSLContext().getSocketFactory();
            clientSocket = socketFactory.createSocket( serverName,serverSocket );
        }else{
            clientSocket = new Socket( serverName,serverSocket );
        }
    }

    clientInputStream = clientSocket.getInputStream();
    clientOutputStream = clientSocket.getOutputStream();
    output = new ObjectOutputStream( clientOutputStream );
    clientgui.getInstance().writer("client basladi");
    Object obj = new Object();
    obj = cer;
    sendData( obj );
}
catch(IOException ioEx){
}
}

//kořturma metodu
public void run(){
    clientStart();
    getData();
}

```



```

}
//veri gönderir
public void sendData(Object o){
    try {
        output.writeObject(o);
        output.flush();
    }
    catch(IOException ioEx){
    }
}
//veri alır
public void getData(){
    Object o;

    try{
        input = new ObjectInputStream( clientInputStream );
    }catch(IOException ioEx){}

    while ( alive )
    {
        try {
            o = input.readObject();
            if (o instanceof message){
                message mes = new message();
                mes =(message) o;
                clientgui.getInstance().writer("mesaj geldi");
            }
            else if (o instanceof Certificate){
                clientgui.getInstance().writer("sertifika geldi");
                Certificate cer = new Certificate();
                cer =(Certificate) o;
                serverPK = cer.getPublicKey();
                clientgui.getInstance().writer("sertifika set edildi");
            }

            else if (o instanceof SignedObject){
                SignedObject so;
                so =(SignedObject) o;
                keypair.verifyObject(so,serverPK);
                clientgui.getInstance().writer("imzali mesaj geldi ve onaylandi");
                mes =(message) so.getObject();

                clientgui.getInstance().writerIslem(mes.getmessagesSegments().getSimpleDataElement().getValue());

            }

        }
        catch(SocketException socEx)
        {

```

```

        clientgui.getInstance().writer("server kapandi, client ta kapandi");
        alive=false;
    }

    catch(InterruptedException ioEx) {
        clientgui.getInstance().writer("interrupted io exception");
    }
    catch(EOFException ioEx) {
        //clientgui.getInstance().writer("eof exception");
        //try {sleep(1000);
        //clientgui.getInstance().writer("eof exception");
        clientgui.getInstance().writer("client kapandi");
        //alive=false;
        //} catch(Exception e){clientgui.getInstance().writer("eof exception 2");
        //e.printStackTrace();
        //}
    }
    catch(IOException ioEx) {
        clientgui.getInstance().writer("client kapandi");
        alive=false;
    }
    catch(Exception ex)
    {
        clientgui.getInstance().writer("client kapandi\n"+ex);
        ex.printStackTrace();
        alive=false;
    }
}
try {
    clientSocket.close();
    clientgui.getInstance().writer("client kapandi");
}
catch(IOException ioEx) {
}
}

//istemci durdurur
public void clientStop(){
    alive = false;
    try{
        input.close();
        output.close();
    }catch(IOException ioEx){}
}

/**
 * Returns the SSL context to be used in SSL communication for both the client
 * and the server side.
 */
protected com.sun.net.ssl.SSLContext getSSLContext(){
    long before = System.currentTimeMillis();

```

```

com.sun.net.ssl.SSLContext sslContext = null;
try {
    KeyStore keyStore = KeyStore.getInstance("JKS");
    keyStore.load(new FileInputStream("c://kernal//tezprogram//kod2002-01-
02//kod//testkeys"), "passphrase".toCharArray());
    KeyManagerFactory keyManagerFactory =
    KeyManagerFactory.getInstance("SunX509");
    keyManagerFactory.init(keyStore, "passphrase".toCharArray());
    TrustManagerFactory trustManagerFactory =
    TrustManagerFactory.getInstance("SunX509");
    trustManagerFactory.init(keyStore);
    sslContext = com.sun.net.ssl.SSLContext.getInstance("SSL");
    sslContext.init(keyManagerFactory.getKeyManagers(),
    trustManagerFactory.getTrustManagers(), null);
} catch (Exception ie) {ie.printStackTrace();}

return sslContext;
}
//nesneyi halleder
protected void handleObject( Object o){
if (o instanceof Certificate) {
    Certificate cer = (Certificate) o;
    //certificates.put( cer.getHostInfo().getHostId(), cer );
}
else if (o instanceof message){
    message m = (message) o;
    //handleMessage(m);
}
}
}
}

```

2.2 EDI

```

package edi;

import java.util.Vector;
import java.io.Serializable;

public class compositedataelement implements Serializable{

    Vector cdeVector = new Vector();

    public compositedataelement() {
    }

    public void setCDEVVector(Vector v){

```

```

        cdeVector = v;
    }
    public Vector getCDEVector(){
        return cdeVector;
    }
    //vektöre eleman ekler
    public void addElementtoVector( simplicatedataelement sde){
        cdeVector.addElement( sde );
    }

}

package edi;

import java.io.Serializable;

public class message implements Serializable{

    segments messagesSegments = new segments();
    unb messageUnb = new unb();
    unh messageUnh = new unh();
    unt messageUnt = new unt();
    unz messageUnz = new unz();

    public message() {
    }
    //mesajın set metodları
    public void setMessagesSegments(segments seg){
        messagesSegments = seg;
    }
    public void setmessageUnb(unb un){
        messageUnb = un;
    }
    public void setmessageUnh(unh un){
        messageUnh = un;
    }
    public void setmessageUnt(unt un){
        messageUnt = un;
    }
    public void setmessageUnz(unz un){
        messageUnz = un;
    }
    //mesajın get metodları
    public segments getmessagesSegments(){
        return messagesSegments;
    }
    public unb getmessageUnb(){
        return messageUnb;
    }
}

```

```

    public unh getMessageUnh(){
        return messageUnh;
    }
    public unt getMessageUnt(){
        return messageUnt;
    }
    public unz getMessageUnz(){
        return messageUnz;
    }
}

package edi;

import java.io.Serializable;

public class segments implements Serializable{

    tag segmentTag = new tag();
    simpledataelement segmentSDE = new simpledataelement();
    //compositedataelement segmentCDE= new compositedataelement();

    public segments() {
    }
    //mesajın set metodları
    public void setTag(tag t){
        segmentTag = t;
    }
    public void setSimpleDataElement(simpledataelement sde){
        segmentSDE = sde;
    }
    //public void setCompositeDataElement(compositedataelement cde){
    // segmentCDE = cde;
    //}

    public tag getTag(){
        return segmentTag;
    }
    public simpledataelement getSimpleDataElement(){
        return segmentSDE;
    }

    //public compositedataelement getCompositeDataElement(){
    // return segmentCDE;
    //}

}

package edi;

```

```
import java.io.Serializable;

public class simpledataelement implements Serializable{

    String value = new String();

    public simpledataelement() {
    }

    public void setValue(String s){
        value = s;
    }

    public String getValue(){
        return value;
    }

}
```

```
package edi;

import java.lang.String;
import java.io.Serializable;

public class tag implements Serializable{

    int code = 0;
    int value = 0;

    public tag() {
    }

    public void setValue(int i){
        value = i;
    }
    public void setCode(int i){
        code = i;
    }

    public int getValue(){
        return value;
    }
    public int getCode(){
        return code;
    }

}
```

```
package edi;
```

```
import java.lang.String;
import java.io.Serializable;
//import java.util.Date;
/*
```

Segment: UNB, Interchange Header

Function: To start, identify and specify an interchange

```
*/
public class unb implements Serializable{

    public unb() {
    }
    /*
S001      M  SYNTAX IDENTIFIER
0001  a4    M  Syntax identifier    a3, upper case
                        Controlling Agency (e.g.
                        UNO=UN/ECE) and a1 stating
                        level (e.g. A) (which
                        together give UNOA)
0002  n1    M  Syntax version number Increments 1 for each new
                        version. Shall be 2 to
                        indicate this version

    */

    String syntaxIdentifier = new String();
    int syntaxVersionNumber = 0; //default

    public void setSyntaxIdentifier( String si){
        syntaxIdentifier = si;
    }
    public String getSyntaxIdentifier(){
        return syntaxIdentifier;
    }

    public void setSyntaxVersionNumber( int svn){
        syntaxVersionNumber = svn;
    }
    public int getSyntaxVersionNumber(){
        return syntaxVersionNumber;
    }

    /*
S002      M  INTERCHANGE SENDER
0004  an..35 M  Sender identification Code or name as specified
                        in IA
```

0007 an..4 C Partner identification Used with sender
code qualifier identification code
0008 an..14 C Address for reverse
routing

*/

```
String senderIdentification = new String();
String partnerIdentification = new String();
String Address = new String();
public void setAddress(String s){
    Address = s;
}
public String getAddress(){
    return Address;
}
/*
```

S004 M DATE/TIME OF PREPARATION
0017 n6 M Date YYMMDD
0019 n4 M Time HHMM

*/

/*

```
Date dateandTime = new Date();
public void setDateandTime(Date d){
    dateandTime = d;
}
public Date getDateandTime(){
    return dateandTime;
}
*/
```

/*

0020 an..14 M INTERCHANGE CONTROL Unique reference
REFERENCE assigned by sender

*/

int reference = 0;

/*

0031 n1 C ACKNOWLEDGEMENT REQUEST Set = 1 if sender
requests
acknowledgement, i.e.
UNB and UNZ
segments received
and identified

*/

int REQUEST = 0;

}

```
package edi;
```

```
import java.io.Serializable;
```

```
/*
```

```
Segment: UNH, Message Header
```

```
Function: To head, identify and specify a Message
```

```
*/
```

```
public class unh implements Serializable{
```

```
    public unh() {
    }

```

```
/*
```

```
0062 an..14 M MESSAGE REFERENCE NUMBER A sender's unique
           message reference
```

```
*/
```

```
int referenceNumber = 0;
```

```
/*
```

```
S009      M MESSAGE IDENTIFIER
```

```
0065 an..6 M Message type      Type of message being
           transmitted
```

```
0052 an..3 M Message version number Version number of the
           message type. If UNG is
           used, 0052 shall be
           identical
```

```
0054 an..3 M Message release number Release number within
           current version number
```

```
*/
```

```
int messageType = 0;
```

```
int messageVersionNumber = 0;
```

```
int messageReleaseNumber = 0;
```

```
public void setMessageType(int i){
    messageType = i;
}

```

```
public void setMessageVersionNumber(int i){
    messageVersionNumber = i;
}

```

```
public void setMessageReleaseNumber(int i){
    messageReleaseNumber = i;
}

```

```
public int getMessageType(){
    return messageType;
}

```



```
public int getMessageVersionNumber(){
    return messageVersionNumber;
}
```

```
public int setMessageReleaseNumber(){
    return messageReleaseNumber;
}
```

```
/*
S010      C  STATUS OF THE TRANSFER
0070  n..2  M  Sequence of transfers   Starts at 1 and is
                                incremented by 1 for
                                each transfer
```

```
*/
int sequenceofTransfers = 0;
public void setSequenceofTransfers(int i){
    sequenceofTransfers = i;
}
```

```
public int getSequenceofTransfers(){
    return sequenceofTransfers;
}
public void sequenceofTransfersCount(){
    sequenceofTransfers++;
}

}
```

```
package edi;
```

```
import java.io.Serializable;
```

```
public class unt implements Serializable{
/*
```

```
Segment: UNT, Message Trailer
```

```
Function: To end and check the completeness of a Message
```

```
*/
public unt() {
}
```

```
/*
0074  n..6  M  NUMBER OF SEGMENTS IN THE Control count
                MESSAGE                including UNH and UNT
```

```
*/
int numberofSegments = 0;
```



```
public void setNumberOfSegments(int i){  
    numberOfSegments = i;  
}
```

```
public int getNumberOfSegments(){  
    return numberOfSegments;  
}  
/*
```

0062 an..14 M MESSAGE REFERENCE NUMBER Shall be identical to
0062 in UNH

```
*/  
int reference = 0;  
  
}
```

```
package edi;
```

```
import java.io.Serializable;
```

```
public class unz implements Serializable{  
/*
```

Segment: UNZ, Interchange Trailer

Function: To end and check the completeness of an interchange

```
*/  
public unz() {  
}
```

```
/*  
0036 n..6 M INTERCHANGE CONTROL COUNT The count of the number  
of messages or, if  
used, the number of  
functional groups in  
the interchange.  
One of these counts  
shall appear.
```

```
*/  
int count = 0;
```

```
public void setCount(int i){  
    count = i;  
}
```

```
public int getCount(){  
    return count;
```

```

}

/*
0020 an..14 M INTERCHANGE CONTROL    Shall be identical to
REFERENCE                             0020 in UNB

*/
int reference = 0;

}

```

2.3 GUI

```

package gui;

import java.awt.*;
import javax.swing.*;
import com.borland.jbcl.layout.*;
import javax.swing.border.*;
import java.awt.event.*;
import client.client;
import java.lang.String;
import edi.message;
import java.security.SignedObject;
import java.util.StringTokenizer;

/**
 * Title:
 * Description:
 * Copyright: Copyright (c) 2000
 * Company:
 * @author
 * @version 1.0
 */

public class clientgui extends JFrame {
    XYLayout xYLayout1 = new XYLayout();
    Border border1;
    JTabbedPane jTabbedPane1 = new JTabbedPane();
    JPanel serverPanel = new JPanel();
    JPanel islemPanel = new JPanel();
    Border border2;
    XYLayout xYLayout2 = new XYLayout();
    JTextField serverNameTextField = new JTextField();
    JLabel serverNameLabel = new JLabel();
    JLabel serverPortLabel = new JLabel();
    JTextField serverPortTextField = new JTextField();
    JTextArea serverTextArea = new JTextArea();
    JButton serverStartButton = new JButton();

```

```

JButton serverStopButton = new JButton();
message mes = new message();
client Client;

```

```

private static clientgui singleInstance = null;
Box box1;
JScrollPane jScrollPane1 = new JScrollPane();
JTextField jTextField1 = new JTextField();
XYLayout xYLayout3 = new XYLayout();
JTextField jTextField3 = new JTextField();
JTextField jTextField4 = new JTextField();
JTextField jTextField5 = new JTextField();
JLabel jLabel1 = new JLabel();
JLabel jLabel2 = new JLabel();
JLabel jLabel4 = new JLabel();
JLabel jLabel5 = new JLabel();
JLabel jLabel3 = new JLabel();
JTextField jTextField2 = new JTextField();
JLabel jLabel6 = new JLabel();
JPasswordField jPasswordField1 = new JPasswordField();
JScrollPane jScrollPane2 = new JScrollPane();
JTextArea jTextArea1 = new JTextArea();
JButton jButton1 = new JButton();
JButton jButton2 = new JButton();

```

```

public static clientgui getInstance(){
    if (singleInstance == null)
        singleInstance = new clientgui();
    return singleInstance;
}

```

```

public clientgui() {
    try {
        jbInit();
    }
    catch(Exception e) {
        e.printStackTrace();
    }
}

private void jbInit() throws Exception {
    border1 = new EtchedBorder(EtchedBorder.RAISED,Color.white,new Color(148,
145, 140));
    border2 = new EtchedBorder(EtchedBorder.RAISED,Color.white,new Color(148,
145, 140));
    box1 = Box.createVerticalBox();
    this.getContentPane().setLayout(xYLayout1);
    xYLayout1.setWidth(645);
    xYLayout1.setHeight(446);
    jTabbedPane1.setBorder(border2);
    serverPanel.setBorder(BorderFactory.createEtchedBorder());
}

```

```

serverPanel.setLayout(xYLayout2);
serverNameLabel.setText("Server Name :");
serverPortLabel.setText("Server Port :");
serverStartButton.setText("Start");
serverStartButton.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(ActionEvent e) {
        serverStartButton_actionPerformed(e);
    }
});
serverStopButton.setText("Stop");
serverStopButton.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(ActionEvent e) {
        serverStopButton_actionPerformed(e);
    }
});
this.getContentPane().setBackground(new Color(144, 151, 217));
this.setTitle("CLIENT");
serverTextArea.setBorder(BorderFactory.createEtchedBorder());
serverTextArea.setRows(8);
serverNameTextField.setText("localhost");
serverPortTextField.setText("1977");
islemPanel.setLayout(xYLayout3);
jLabel1.setText("marka");
jLabel2.setText("fiat");
jLabel4.setText("adet");
jLabel5.setText("tip");
jLabel3.setText("Kullanici Adi");
jLabel6.setText("Kullanici Sifresi");
jButton1.setText("Satin Al");
jButton1.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(ActionEvent e) {
        jButton1_actionPerformed(e);
    }
});
jButton2.setText("Sorgula");
jButton2.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(ActionEvent e) {
        jButton2_actionPerformed(e);
    }
});
this.getContentPane().add(jTabbedPane1, new XYConstraints(0, 3, 644, 443));
jTabbedPane1.add(serverPanel, "Server");
serverPanel.add(serverNameTextField, new XYConstraints(251, 24, 109, -1));
serverPanel.add(serverNameLabel, new XYConstraints(155, 27, 91, -1));
serverPanel.add(serverPortTextField, new XYConstraints(249, 61, 112, -1));
serverPanel.add(serverPortLabel, new XYConstraints(156, 62, -1, -1));
serverPanel.add(serverStartButton, new XYConstraints(155, 320, 157, -1));
serverPanel.add(serverStopButton, new XYConstraints(340, 319, 161, -1));
serverPanel.add(box1, new XYConstraints(573, 124, -1, -1));
serverPanel.add(jScrollPane1, new XYConstraints(159, 91, 341, 216));

```

```

serverPanel.add(jLabel3, new XYConstraints(17, 104, 86, -1));
serverPanel.add(jTextField2, new XYConstraints(16, 134, 105, -1));
serverPanel.add(jLabel6, new XYConstraints(17, 170, 106, -1));
serverPanel.add(jPasswordField1, new XYConstraints(17, 193, 107, -1));
jTabbedPane1.add(islemPanel, "islemPanel");
jScrollPane1.getViewport().add(serverTextArea, null);
islemPanel.add(jTextField1, new XYConstraints(201, 19, 177, -1));
islemPanel.add(jTextField4, new XYConstraints(200, 47, 177, -1));
islemPanel.add(jLabel5, new XYConstraints(60, 19, 107, -1));
islemPanel.add(jLabel1, new XYConstraints(62, 50, 111, -1));
islemPanel.add(jScrollPane2, new XYConstraints(63, 216, 322, 171));
islemPanel.add(jButton1, new XYConstraints(278, 175, 100, -1));
islemPanel.add(jLabel4, new XYConstraints(61, 120, 107, -1));
islemPanel.add(jTextField3, new XYConstraints(201, 146, 177, -1));
islemPanel.add(jTextField5, new XYConstraints(201, 119, 177, -1));
islemPanel.add(jLabel2, new XYConstraints(61, 151, 116, -1));
islemPanel.add(jButton2, new XYConstraints(282, 81, 96, -1));
jScrollPane2.getViewport().add(jTextArea1, null);
}

```

```

void serverStartButton_actionPerformed(ActionEvent e) {

    //if ( ) {
        Client = new client();
        Client.setClient( serverNameTextField.getText(), Integer.parseInt(
serverPortTextField.getText() ) );
        Client.start();
    //}

}

```

```

void serverStopButton_actionPerformed(ActionEvent e) {
    Client.clientStop();
}

```

```

public void writer(String anything){
    serverTextArea.append( anything+"\n" );
}
public void writerIslem(String anything){
    StringTokenizer st = new StringTokenizer(anything);

    jTextField5.setText(st.nextToken());
    jTextField3.setText(st.nextToken());
}
void jButton1_actionPerformed(ActionEvent e) {
    String alim = new String("Select ");
}

```

```

void jButton2_actionPerformed(ActionEvent e) {
    message mes = new message();
}

```

```

Object obj = new Object();
String sorgula = new String("Select adet,fiat from cihaztab where tip =
"+jTextField1.getText()+" and marka = "+jTextField4.getText()+"");

mes.getmessagesSegments().getSimpleDataElement().setValue( sorgula );
mes.getmessagesSegments().getTag().setValue(1);

mes.getMessageUnb().setAddress(Client.clientSocket.getLocalAddress().toString());

mes.getMessageUnh().setMessageType(1);
mes.getMessageUnt().setNumberOfSegments(1);
mes.getMessageUnz().setCount(1);
obj = mes;
SignedObject so;
so = Client.getSignKP().signObj(obj,Client.getKP());
Client.sendData( so );

}

public void constructMessage(String s, int i,String adr, int typ, int nos, int co){

mes.getmessagesSegments().getSimpleDataElement().setValue( s );
mes.getmessagesSegments().getTag().setValue(i);
mes.getMessageUnb().setAddress(adr);

mes.getMessageUnh().setMessageType(typ);
mes.getMessageUnt().setNumberOfSegments(nos);
mes.getMessageUnz().setCount(co);
}

}

package gui;

import javax.swing.UIManager;
import java.awt.*;

public class clientguiclass {
    boolean packFrame = false;

    //Construct the application
    public clientguiclass() {
        clientgui frame = clientgui.getInstance();
        //Validate frames that have preset sizes
        //Pack frames that have useful preferred size info, e.g. from their layout
        if (!packFrame) {
            frame.pack();
        }
    }
}

```

```

    else {
        frame.validate();
    }
    //Center the window
    Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();
    Dimension frameSize = frame.getSize();
    if (frameSize.height > screenSize.height) {
        frameSize.height = screenSize.height;
    }
    if (frameSize.width > screenSize.width) {
        frameSize.width = screenSize.width;
    }
    frame.setLocation((screenSize.width - frameSize.width) / 2, (screenSize.height -
frameSize.height) / 2);
    frame.setVisible(true);
}

//Main method
public static void main(String[] args) {
    try {

        UIManager.setLookAndFeel(UIManager.getCrossPlatformLookAndFeelClassName(
));
    }
    catch(Exception e) {
        e.printStackTrace();
    }
    new clientguiclass();
}
}

```

```

package gui;

import java.sql.*;
import java.awt.*;
import com.borland.jbcl.layout.*;
import javax.swing.*;
import javax.swing.border.*;
import javax.swing.tree.*;
import java.awt.event.*;
import server.server;
import java.lang.NumberFormatException;
import com.borland.dbswing.*;
import com.borland.dx.sql.dataset.*;
import com.borland.internetbeans.*;
import com.borland.dx.dataset.*;
import com.borland.datastore.*;

```

```

/**
 * Title:

```

```

* Description:
* Copyright: Copyright (c) 2000
* Company:
* @author
* @version 1.0
*/

```

```

public class servergui extends JFrame {

    XYLayout xYLayout1 = new XYLayout();
    JTabbedPane jTabbedPane1 = new JTabbedPane();
    TitledBorder titledBorder1;
    JPanel server = new JPanel();
    XYLayout xYLayout2 = new XYLayout();
    JButton start = new JButton();
    JPanel clientPanel = new JPanel();
    JPanel islemlerPanel = new JPanel();
    JPanel sorgulamaPanel = new JPanel();
    JPanel alimPanel = new JPanel();
    JButton stop = new JButton();
    JLabel serverPortLabel = new JLabel();
    JTextField serverPortField = new JTextField();
    public JTextArea serverTextArea = new JTextArea();
    XYLayout xYLayout3 = new XYLayout();

    DefaultMutableTreeNode top = new DefaultMutableTreeNode("Merkez");

    server Server=null;

    private static servergui singleInstance = null;
    Border border1;
    JScrollPane jScrollPane1 = new JScrollPane();
    XYLayout xYLayout4 = new XYLayout();
    Database database1 = new Database();
    QueryDataSet queryDataSet1 = new QueryDataSet();
    JdbTable jdbTable1 = new JdbTable();
    TableScrollPane tableScrollPane1 = new TableScrollPane();
    QueryDataSet queryDataSet2 = new QueryDataSet();
    JdbTable jdbTable2 = new JdbTable();
    TableScrollPane tableScrollPane2 = new TableScrollPane();
    XYLayout xYLayout5 = new XYLayout();
    JdbTextField jdbTextField1 = new JdbTextField();
    JdbTextField jdbTextField2 = new JdbTextField();
    JdbTextField jdbTextField3 = new JdbTextField();
    JdbTextField jdbTextField4 = new JdbTextField();
    JdbToggleButton jdbToggleButton1 = new JdbToggleButton();
    JdbLabel jdbLabel1 = new JdbLabel();
    JdbLabel jdbLabel2 = new JdbLabel();
    JdbLabel jdbLabel3 = new JdbLabel();

```

```

JdbLabel jdbLabel4 = new JdbLabel();
JdbLabel jdbLabel5 = new JdbLabel();
JdbLabel jdbLabel6 = new JdbLabel();
QueryDataSet queryDataSet3 = new QueryDataSet();
QueryDataSet queryDataSet4 = new QueryDataSet();
JdbNavComboBox jdbNavComboBox1 = new JdbNavComboBox();
JdbNavComboBox jdbNavComboBox2 = new JdbNavComboBox();
JdbTextField userName = new JdbTextField();
JPasswordField sifre = new JPasswordField();
JdbLabel UserName = new JdbLabel();
JdbLabel Sifre = new JdbLabel();
JdbTable jdbTable3 = new JdbTable();
JdbNavComboBox jdbNavComboBox4 = new JdbNavComboBox();
ParameterRow parameterRow1 = new ParameterRow();
Column column1 = new Column();
Column column2 = new Column();
QueryDataSet queryDataSet5 = new QueryDataSet();
DBDisposeMonitor dbDisposeMonitor1 = new DBDisposeMonitor();
QueryDataSet queryDataSet6 = new QueryDataSet();
QueryDataSet queryDataSet7 = new QueryDataSet();
XYLayout xYLayout6 = new XYLayout();
TableScrollPane tableScrollPane3 = new TableScrollPane();
JdbTable jdbTable4 = new JdbTable();
JdbNavComboBox jdbNavComboBox3 = new JdbNavComboBox();
JdbNavComboBox jdbNavComboBox5 = new JdbNavComboBox();
JButton jButton1 = new JButton();
JTextArea jTextArea1 = new JTextArea();

```

```

public static servergui getInstance(){
    if(singleInstance == null)
        singleInstance = new servergui();
    return singleInstance;
}

```

```

public servergui() {
    try {
        jbInit();
    }
    catch(Exception e) {
        e.printStackTrace();
    }
}

```

```

private void jbInit() throws Exception {
    titledBorder1 = new TitledBorder("");
    border1 = new EtchedBorder(EtchedBorder.RAISED,Color.white,new Color(178,
178, 178));
    this.getContentPane().setBackground(new Color(144, 151, 217));
    this.setTitle("SERVER");
}

```

```

this.getContentPane().setLayout(xYLayout1);
jTabbedPane1.setBorder(titledBorder1);
jTabbedPane1.setDebugGraphicsOptions(DebugGraphics.NONE_OPTION);
jTabbedPane1.setOpaque(true);
jTabbedPane1.setToolTipText("");
server.setLayout(xYLayout2);
start.setActionCommand("start");
start.setText("start");
start.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(ActionEvent e) {
        start_actionPerformed(e);
    }
});
stop.setActionCommand("stop");
stop.setText("stop");
stop.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(ActionEvent e) {
        stop_actionPerformed(e);
    }
});
serverPortLabel.setText("Server Port :");
clientPanel.setLayout(xYLayout3);

```

```

serverTextArea.setBorder(border1);
xYLayout1.setWidth(630);
xYLayout1.setHeight(462);
serverPortField.setText("1977");
islemlerPanel.setLayout(xYLayout4);
database1.setConnection(new
com.borland.dx.sql.dataset.ConnectionDescriptor("jdbc:oracle:thin:@localhost:1521:
EDI", "EDI", "edi", false, "oracle.jdbc.driver.OracleDriver"));
database1.setDatabaseName("");
queryDataSet1.setSort(new com.borland.dx.dataset.SortDescriptor("", new
String[] { "USERNAME" }, new boolean[] { false, }, null, 0));
queryDataSet1.setQuery(new
com.borland.dx.sql.dataset.QueryDescriptor(database1, "SELECT
KULLANICI.ID,KULLANICI.USERNAME FROM EDI.KULLANICI ORDER BY
" +
    "KULLANICI.USERNAME", null, true, Load.ALL));
jdbTable1.setDataSet(queryDataSet1);
queryDataSet2.setSchemaName("");
queryDataSet2.setTableName("SATIM");
queryDataSet2.setStoreName("");
queryDataSet2.setQuery(new
com.borland.dx.sql.dataset.QueryDescriptor(database1, "SELECT
TIP,MARKA,ADET,FIAT,TARIH,ID FROM SATIM WHERE ID=:ID ORDER " +
    "BY SATIM.TARIH", null, true, Load.ALL));
jdbTable2.setDataSet(queryDataSet2);

```

```

alimPanel.setLayout(xYLayout5);
sorgulamaPanel.setLayout(xYLayout6);
jdbTextField3.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(ActionEvent e) {
        jdbTextField3_actionPerformed(e);
    }
});
jdbToggleButton1.setText("ekle");
jdbToggleButton1.setColumnName("");
jdbToggleButton1.addMouseListener(new java.awt.event.MouseAdapter() {
    public void mouseClicked(MouseEvent e) {
        jdbToggleButton1_mouseClicked(e);
    }
});
/*jdbToggleButton1.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(ActionEvent e) {
        jdbToggleButton1_actionPerformed(e);
    }
});*/
jdbLabel1.setText("tip");
jdbLabel2.setText("marka");
jdbLabel3.setText("alim fiat");
jdbLabel4.setText("satim fiat");
jdbLabel5.setText("adet");
jdbLabel6.setText("tarih");
queryDataSet3.setQuery(new
com.borland.dx.sql.dataset.QueryDescriptor(database1, "SELECT distinct TIP
FROM ALIM", null, true, Load.ALL));
queryDataSet3.setSort(new com.borland.dx.dataset.SortDescriptor("", new
String[] { "TIP" }, new boolean[] { false, }, null, 0));
queryDataSet4.setQuery(new
com.borland.dx.sql.dataset.QueryDescriptor(database1, "SELECT distinct MARKA
FROM ALIM", null, true, Load.ALL));
queryDataSet4.setSort(new com.borland.dx.dataset.SortDescriptor("", new
String[] { "MARKA" }, new boolean[] { false, }, null, 0));
//jdbComboBox1.setItems(null);
jdbNavComboBox1.setEditable(true);
jdbNavComboBox1.setColumnName("TIP");
jdbNavComboBox1.setDataSet(queryDataSet3);
jdbNavComboBox2.setEditable(true);
jdbNavComboBox2.setColumnName("MARKA");
jdbNavComboBox2.setDataSet(queryDataSet4);

UserName.setText("Kullanici Adi");
Sifre.setText("Sifre");
userName.setColumnName("");
alimPanel.setEnabled(false);
clientPanel.setEnabled(false);
islemlerPanel.setEnabled(false);
sorgulamaPanel.setEnabled(false);

```

```

jdbNavComboBox4.setEditable(true);
jdbNavComboBox4.setColumnName("MARKA");
jdbNavComboBox4.setDataSet(queryDataSet7);
column1.setColumnName("tip");
column1.setDataType(com.borland.dx.dataset.Variant.STRING);
column1.setTableName("");
column1.setServerColumnName("NewColumn1");
column1.setSqlType(0);
column2.setColumnName("marka");
column2.setDataType(com.borland.dx.dataset.Variant.STRING);
column2.setServerColumnName("NewColumn2");
column2.setSqlType(0);
queryDataSet5.setQuery(new
com.borland.dx.sql.dataset.QueryDescriptor(database1, "SELECT * FROM
CIHAZTAB WHERE TIP=:TIP AND MARKA=:MARKA", parameterRow1, true,
Load.ALL));
dBDisposeMonitor1.setDataAwareComponentContainer(this);
queryDataSet6.setQuery(new
com.borland.dx.sql.dataset.QueryDescriptor(database1, "select distinct tip from
cihaztab", null, true, Load.ALL));
queryDataSet7.setQuery(new
com.borland.dx.sql.dataset.QueryDescriptor(database1, "select distinct marka from
cihaztab", null, true, Load.ALL));
jdbNavComboBox3.setColumnName("TIP");
jdbNavComboBox3.setDataSet(queryDataSet6);
jdbNavComboBox5.setColumnName("MARKA");
jdbNavComboBox5.setDataSet(queryDataSet7);
jdbTable4.setDataSet(queryDataSet5);
jButton1.setToolTipText("");
jButton1.setText("Sorgula");
jButton1.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(ActionEvent e) {
        jButton1_actionPerformed(e);
    }
});
alimPanel.add(jdbTextField2, new XYConstraints(193, 106, 212, -1));
alimPanel.add(jdbTextField4, new XYConstraints(193, 165, 214, -1));
alimPanel.add(jdbTextField3, new XYConstraints(192, 136, 213, -1));
alimPanel.add(jdbTextField1, new XYConstraints(193, 76, 209, -1));
alimPanel.add(jdbToggleButton1, new XYConstraints(190, 207, 220, -1));
alimPanel.add(jdbLabel1, new XYConstraints(17, 16, 129, -1));
alimPanel.add(jdbLabel2, new XYConstraints(15, 52, 129, -1));
alimPanel.add(jdbLabel3, new XYConstraints(14, 81, 129, -1));
alimPanel.add(jdbLabel4, new XYConstraints(13, 109, 129, -1));
alimPanel.add(jdbLabel6, new XYConstraints(13, 168, 129, -1));
alimPanel.add(jdbLabel5, new XYConstraints(13, 140, 129, -1));
alimPanel.add(jdbNavComboBox1, new XYConstraints(190, 15, 211, -1));
alimPanel.add(jdbNavComboBox2, new XYConstraints(191, 44, 210, -1));
jTabbedPane1.add(sorgulamaPanel, "sorgulamaPanel");

```

```

sorgulamaPanel.add(tableScrollPane3, new XYConstraints(179, 12, 411,
108));
sorgulamaPanel.add(jdbNavComboBox3, new XYConstraints(22, 27, 124, -1));
sorgulamaPanel.add(jdbNavComboBox5, new XYConstraints(21, 58, 123, -1));
sorgulamaPanel.add(jButton1, new XYConstraints(20, 92, 131, -1));
tableScrollPane3.getViewPort().add(jdbTable4, null);
jTabbedPane1.add(server, "server");

islemlerPanel.add(tableScrollPane2, new XYConstraints(140, 1, -1, 352));
islemlerPanel.add(tableScrollPane1, new XYConstraints(3, 0, 141, 354));
jTabbedPane1.add(alimPanel, "alimPanel");
tableScrollPane1.getViewPort().add(jdbTable1, null);
tableScrollPane2.getViewPort().add(jdbTable2, null);
server.add(stop, new XYConstraints(313, 330, 142, -1));
server.add(start, new XYConstraints(163, 329, 142, -1));
server.add(serverPortLabel, new XYConstraints(166, 43, 98, -1));
server.add(serverPortField, new XYConstraints(307, 39, 148, -1));
server.add(jScrollPane1, new XYConstraints(164, 84, 294, 211));
server.add(userName, new XYConstraints(12, 129, 126, 23));
server.add(sifre, new XYConstraints(11, 191, 128, 23));
server.add(UserName, new XYConstraints(11, 102, 129, 24));
server.add(Sifre, new XYConstraints(10, 164, 129, 21));
jTabbedPane1.add(islemlerPanel, "islemlerPanel");
jScrollPane1.getViewPort().add(serverTextArea, null);
this.getContentPane().add(jTabbedPane1, new XYConstraints(0, 0, 735, 457));
jTabbedPane1.add(clientPanel, "clientPanel");
clientPanel.add(jTextArea1, new XYConstraints(-1, 0, 529, 269));
queryDataSet2.setMasterLink(new
com.borland.dx.dataset.MasterLinkDescriptor(queryDataSet1, new String[] {"ID"},
new String[] {"ID"}, true, false, false));
parameterRow1.setColumns(new Column[] {column1, column2});
}

void start_actionPerformed(ActionEvent e) {
    boolean kullaniciOk = false;
    try {
        Class.forName("oracle.jdbc.driver.OracleDriver");
        Connection con =
DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:EDI", "edi",
"edi");
        Statement stmt = con.createStatement();
        ResultSet res1 = stmt.executeQuery("SELECT ID FROM KULLANICI
WHERE KULLANICI.USERNAME = '"+userName.getText()+"'");
        if ( res1.next() ){
            ResultSet res = stmt.executeQuery("SELECT SIFRE FROM SIFRE WHERE
SIFRE.ID = '"+res1.getInt(1)+"'");
            if ( res.next() && (sifre.getText().equals(res.getString(1))) ){
                serverTextArea.append( "sifre dogrulandi\n" );
                kullaniciOk = true;
            } else {

```

```

        serverTextArea.append( "sifre dogrulanamadi\nserver baslatilamadi\n");
    }}
    else
        serverTextArea.append( "kullanici bulunamadi\nserver baslatilamadi\n");
    stmt.close();
    con.close();
}
catch (Exception ex) {
    ex.printStackTrace();
}
if ( kullanıcıOk ){
    Server = new server();
    try
    {
        Server.setServerPort( Integer.parseInt(serverPortField.getText()) );
        serverTextArea.append( "port numarasi okundu\n" );
        Server.start();
    }
    catch(NumberFormatException nuEx)
    {
        serverTextArea.append( nuEx.toString() );
    }
}
}

void stop_actionPerformed(ActionEvent e) {

    if (Server!=null)
    {
        serverTextArea.append( "server kapatildi\n" );
        Server.stopServer();
        Server = null;
    }
    else
        serverTextArea.append( "server yokki\n" );
}

public void writer(String anything){
    serverTextArea.append( anything+"\n" );
}

public void writer_client(String anything){
    jTextArea1.append( anything+"\n" );
}

public void jdbTextField3_actionPerformed(ActionEvent e) {

}

void jdbToggleButton1_mouseClicked(MouseEvent e) {

```

```

try {
    Class.forName("oracle.jdbc.driver.OracleDriver");
    Connection con =
DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:EDI", "edi",
"edi");
    Statement stmt = con.createStatement();
    int res = stmt.executeUpdate("INSERT INTO ALIM
(TIP,MARKA,ADET,ALIMFIAT, "+
    "SATIMFIAT,TARIH)
VALUES('"+jdbNavComboBox1.getSelectedItemAt(0).toString()+"','"+
    ""+jdbNavComboBox2.getSelectedItemAt(0).toString()+"','"+jdbTextField1.getText()+"",
    ""+jdbTextField2.getText()+"','"+jdbTextField3.getText()+"','"+jdbTextField4.getText(
    )+"')");
    stmt.close();
    con.close();
}
catch (Exception ex) {
    ex.printStackTrace();
}
}

```

```

void jdbToggleButton2_actionPerformed(ActionEvent e) {
try {
    // change the values in the parameter row
    // and refresh the display
    parameterRow1.setString("tip",
        jdbNavComboBox3.getSelectedItemAt(0).toString());
    parameterRow1.setString("marka",
        jdbNavComboBox4.getSelectedItemAt(0).toString());
    queryDataSet5.refresh();
}
catch (Exception ex) {
    ex.printStackTrace();
}
}

```

```

void jButton1_actionPerformed(ActionEvent e) {
try {
    // change the values in the parameter row
    // and refresh the display
    parameterRow1.setString("tip",
        jdbNavComboBox3.getSelectedItemAt(0).toString());
    parameterRow1.setString("marka",
        jdbNavComboBox4.getSelectedItemAt(0).toString());
    queryDataSet5.refresh();
}
catch (Exception ex) {
    ex.printStackTrace();
}
}

```

```

    }

}

package gui;

import javax.swing.UIManager;
import java.awt.*;

public class serverguiclass {
    boolean packFrame = false;

    //Construct the application
    public serverguiclass() {
        servergui frame = servergui.getInstance();
        //Validate frames that have preset sizes
        //Pack frames that have useful preferred size info, e.g. from their layout
        if (!packFrame) {
            frame.pack();
        }
        else {
            frame.validate();
        }
        //Center the window
        Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();
        Dimension frameSize = frame.getSize();
        if (frameSize.height > screenSize.height) {
            frameSize.height = screenSize.height;
        }
        if (frameSize.width > screenSize.width) {
            frameSize.width = screenSize.width;
        }
        frame.setLocation((screenSize.width - frameSize.width) / 2, (screenSize.height -
frameSize.height) / 2);
        frame.setVisible(true);
    }

    //Main method
    public static void main(String[] args) {
        try {

            UIManager.setLookAndFeel(UIManager.getCrossPlatformLookAndFeelClassName(
));
        }
        catch(Exception e) {
            e.printStackTrace();
        }
        new serverguiclass();
    }
}

```

```
}
}
```

2.4 SECURITY

```
package security;
```

```
import java.io.*;
import java.security.PublicKey;
```

```
public class Certificate implements Serializable {
```

```
    private PublicKey publicKey;
```

```
    //private HostInfo hostInfo;
```

```
    public Certificate() {
    }
```

```
    //sertifikayı set eder
```

```
    public Certificate(PublicKey newPublicKey) {
        publicKey = newPublicKey;
    }
```

```
    // public Certificate(HostInfo hostInf, PublicKey newPublicKey) {
    //     hostInfo = hostInf;
    //     publicKey = newPublicKey;
    // }
```

```
    //sertifikayı döndürür
```

```
    public PublicKey getPublicKey() {
        return publicKey;
    }
```

```
    public void setPublicKey(PublicKey pk) {
        publicKey = pk;
    }
```

```
    //public HostInfo getHostInfo() {
    //     return hostInfo;
    // }
```

```
    //public String toString(){
    //     return "( Certificate of host "+ getHostInfo().getHostId()+", "
    //         +getHostInfo().getHostname()+ ":"+getHostInfo().getPort() +" )";
    // }
```

```
}
```

```
package security;
```

```
import java.security.KeyPairGenerator;
import java.security.SecureRandom;
```

```
import java.security.spec.DSAParameterSpec;
import java.security.KeyPair;
import java.security.Signature;
import java.security.PrivateKey;
import java.security.PublicKey;
import java.math.BigInteger;
import java.security.SignedObject;
import java.io.Serializable;
```

```
/**
 * <p>Title: </p>
 * <p>Description: </p>
 * <p>Copyright: Copyright (c) 2000</p>
 * <p>Company: </p>
 * @author unascribed
 * @version 1.0
 */
```

```
public class sign {
//anahtarlar
    BigInteger userSeed = new
    BigInteger("8d5155894229d5e689ee01e6018a237e2cae64cd",16);
```

```
//counter = 92
```

```
    BigInteger p = new
    BigInteger("fd7f53811d75122952df4a9c2eece4e7f611b7523cef4400c31e3f80"+
    "b6512669455d402251fb593d8d58fabfc5f5ba30f6cb9b556cd7813b"+
    "801d346ff26660b76b9950a5a49f9fe8047b1022c24fbba9d7feb7c6"+
    "1bf83b57e7c6a8a6150f04fb83f6d3c51ec3023554135a169132f675"+
    "f3ae2b61d72aeff22203199dd14801c7",16);
```

```
    BigInteger q = new
    BigInteger("9760508f15230bccb292b982a2eb840bf0581cf5",16);
```

```
    BigInteger g = new
    BigInteger("f7e1a085d69b3ddecbbcab5c36b857b97994afbbfa3aea82f9574c0b"+
    "3d0782675159578ebad4594fe67107108180b449167123e84c281613"+
    "b7cf09328cc8a6e13c167a8b547c8d28e0a3ae1e2bb3a675916ea37f"+
    "0bfa213562f1fb627a01243bcc4f1bea8519089a883dfe15ae59f06"+
    "928b665e807b552564014c3bfecf492a",16);
```

```
public sign() {
    String b = new String ("1,1,1,1,1,1,1,1,1,1,1,1");
    SignedObject so;
    KeyPair pair;
    pair = keyGen();
    so = signObj(b,pair);
```

```

        verifyObject(so,pair.getPublic());
    }
//anahtar üretir
public KeyPair keyGen(){
    try{
        KeyPairGenerator keyGen = KeyPairGenerator.getInstance("DSA");
        SecureRandom random1 = SecureRandom.getInstance("SHA1PRNG", "SUN");
        random1.setSeed(userSeed.toByteArray());
        keyGen.initialize(1024, random1);
        DSAParameterSpec dsaSpec = new DSAParameterSpec(p, q, g);
        SecureRandom random = SecureRandom.getInstance("SHA1PRNG", "SUN");
        random.setSeed(userSeed.toByteArray());
        keyGen.initialize(dsaSpec, random);
        KeyPair pair = keyGen.generateKeyPair();
        return pair;}
    catch(Exception ex){return null;}

}
//nesneyi imzalar
public SignedObject signObj(Object data,KeyPair pair){
    try
    {
        /*Signature dsa = Signature.getInstance("SHA1withDSA");
        PrivateKey priv = pair.getPrivate();
        dsa.initSign(priv);
        dsa.update(data);
        byte[] sig = dsa.sign();*/
        SignedObject so = new SignedObject((Serializable)data,
            (PrivateKey)pair.getPrivate(),
            Signature.getInstance("SHA1withDSA"));

        return so;}
    catch(Exception ex){ return null;}
}
//nesneyi onaylar
public void verifyObject(SignedObject data,PublicKey pub){
    try {
        /* Signature dsa = Signature.getInstance("SHA1withDSA");
        PublicKey pub = pair.getPublic();
        dsa.initVerify(pub);
        dsa.update(data);
        boolean verifies = dsa.verify(sig);*/

        boolean verifies = data.verify( pub, Signature.getInstance("SHA1withDSA"));
        System.out.println("signature verifies: " + verifies);

    }
    catch(Exception ex){return;}
}

```



```
}

public static void main(String[] args) {
    sign sign1 = new sign();
}
}
```

2.5 SERVER

```
package server;

import java.io.InputStream;
import java.io.OutputStream;
import java.io.IOException;
import java.net.Socket;
import java.net.SocketException;
import gui.servergui;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.io.InterruptedIOException;
import java.io.EOFException;
import edi.message;
import security.Certificate;
import security.sign;
import java.security.KeyPair;
import java.security.PublicKey;
import java.security.SignedObject;
import java.lang.Class;
import java.sql.Statement;
import java.sql.DriverManager;
import java.sql.Connection;
import java.sql.ResultSet;

/**
 * Title:
 * Description:
 * Copyright: Copyright (c) 2000
 * Company:
 * @author
 * @version 1.0
 */

public class clientService extends Thread{

    private InputStream inputStream;
    private OutputStream outputStream;
    private Socket clientServiceSocket;
    public boolean alive = false;
    public boolean online = true;
```

```

public int id;
static int i=0;
private ObjectInputStream input;
private ObjectOutputStream output;
private sign keypair = new sign();
private KeyPair kp;
private Certificate cer = new Certificate();
PublicKey clientPK;
message mes1 = new message();
public clientService(Socket socket) {
    clientServiceSocket = socket;
    i++;
    id=i;
    kp = keypair.keyGen();
    cer.setPublicKey(kp.getPublic());
}
public KeyPair getKP(){
    return kp;
}
public sign getSignKP(){
    return keypair;
}
public void run(){
    try{
        alive = true;
        startClientService();
        getData();
    }
    catch(RuntimeException runtimeEx){

    }
    finally {
        alive = false;
    }
}
//sunucu istemci için bir sınıf açar
public void startClientService(){

    try{
        //clientServiceSocket.setSoTimeout( 1000 );
        inputStream = clientServiceSocket.getInputStream();
        outputStream = clientServiceSocket.getOutputStream();
        output = new ObjectOutputStream( outputStream );
        Object obj = new Object();
        obj = cer;
        sendData( obj );
    }
    catch (IOException ioEx){

```

```

        ioEx.printStackTrace();
    }
}

```

//veri gönderir

```

public void sendData(Object o){
    try{
        output.writeObject(o);
        output.flush();
    }
    catch(IOException ioEx){
    }
}

```

//veri alır

```

public void getData(){

```

```

    int timeout = 10000;
    Object o;

```

```

    try{
        input = new ObjectInputStream( inputStream );
    }catch(IOException ioEx){}

```

```

while(alive){

```

```

    try{
        o = input.readObject();
        if (o instanceof message){
            //message mes = new message();
            message mes =(message) o;
            servergui.getInstance().writer("mesaj geldi");
            if (mes.getMessageUnh().getMessageType() == 1)
            {

```

```

                try {
                    Class.forName("oracle.jdbc.driver.OracleDriver");
                    Connection con =

```

```

DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:EDI", "edi",
"edi");

```

```

        Statement stmt = con.createStatement();
        ResultSet res =

```

```

stmt.executeQuery(mes.getmessagesSegments().getSimpleDataElement().getValue()
);

```

```

        servergui.getInstance().writer("mesaj isleniyor");
        stmt.close();
        con.close();
    }

```

```

    catch (Exception ex) {
        ex.printStackTrace();
    }
}

```

```

    }
    }
    else if (o instanceof SignedObject){
        SignedObject so;
        so =(SignedObject) o;
        keypair.verifyObject(so,clientPK);
        servergui.getInstance().writer("imzali mesaj geldi ve onaylandi");
        message mes =(message) so.getObject();
        if (mes.getMessageUnh().getMessageType() == 1)
        {
            try {
                Class.forName("oracle.jdbc.driver.OracleDriver");
                Connection con =
                DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:EDI", "edi",
                "edi");
                Statement stmt = con.createStatement();
                ResultSet res =
                stmt.executeQuery(""+mes.getMessagesSegments().getSimpleDataElement().getValue()
                ue()+"");
                servergui.getInstance().writer("mesaj
                isleniyor\n"+mes.getMessagesSegments().getSimpleDataElement().getValue());
                if (res.next()){
                    int s1 = res.getInt(1);
                    int s2 = res.getInt(2);
                    String s = s1+" "+s2;

                constructMessage(s,2,this.getClientServiceSocket().getInetAddress().toString(),2,1,1);
                    servergui.getInstance().writer("mesaj
                    isleniyor\n"+mes1.getMessagesSegments().getSimpleDataElement().getValue());
                    SignedObject so1;
                    so1 = this.getSignKP().signObj(mes1,this.getKP());
                    this.sendData(so1);
                }
                else
                    servergui.getInstance().writer("birsey bulunamadi");
                stmt.close();
                con.close();
            }
            catch (Exception ex) {
                ex.printStackTrace();
            }
        }
    }

    else if (o instanceof Certificate){
        servergui.getInstance().writer("sertifika geldi");
        Certificate cer = new Certificate();
        cer =(Certificate) o;
        clientPK = cer.getPublicKey();
        servergui.getInstance().writer("sertifika set edildi");
    }

```

```

    }

    //servergui.getInstance().writer( o.toString() );
    }
    catch(SocketException socEx)
    {
        servergui.getInstance().writer("client kapandi-socketexception");
        alive=false;
    }

    catch(InterruptedException ioEx){
        servergui.getInstance().writer("interrupted io exception");

    }
    catch(EOFException ioEx) {
        //clientgui.getInstance().writer("eof exception");
        //try {sleep(1000);
        servergui.getInstance().writer("client kapandi-
eof\n"+this.clientServiceSocket.getInetAddress().toString());
        //alive=false;
        //} catch(Exception e){servergui.getInstance().writer("eof exception 2");
        //e.printStackTrace();
        //}
    }

    catch(IOException ioEx){
        servergui.getInstance().writer("client kapandi-ioexception\n");
        ioEx.printStackTrace();

        alive=false;
    }
    catch(Exception ex)
    {
        servergui.getInstance().writer("client kapandi\n"+ ex);
        alive=false;
        ex.printStackTrace();
    }
    }
    //servergui.getInstance().writer("client kapandi");
}

public void clientServiceStop(){
    alive=false;
    try{
        clientServiceSocket.close();
        input.close();
        output.close();
    }
}

```

```
        catch(IOException ioEx){
        }
    }

    public void constructMessage(String s, int i,String adr, int typ, int nos, int co){
        mes1.getmessagesSegments().getSimpleDataElement().setValue( s );
        mes1.getmessagesSegments().getTag().setValue(i);
        mes1.getmessageUnb().setAddress(adr);

        mes1.getmessageUnh().setMessageType(typ);
        mes1.getmessageUnt().setNumberOfSegments(nos);
        mes1.getmessageUnz().setCount(co);
    }
}
```

```
package server;
```

```
import java.net.ServerSocket;
import java.net.Socket;
import java.io.IOException;
import gui.servergui;
import javax.swing.SwingConstants;
import javax.swing.SwingUtilities;
import java.util.Vector;

import java.security.*;
import java.io.FileInputStream;
import com.sun.net.ssl.*;
import javax.net.ServerSocketFactory;
```

```
/**
 * Title:
 * Description:
 * Copyright: Copyright (c) 2000
 * Company:
 * @author
 * @version 1.0
 */
```

```
public class server extends Thread {
```

```
    ServerSocket serverSocket;
    private static int serverSocketPort;
    private boolean serverPortIsUsed = false;
    private boolean isOpened = true;
    public Vector clientList = new Vector();
```

```
    public server() {
```

```
}
```

```
public void setServerPort(int port){
    serverSocketPort = port;
}
public int getServerPort(){
    return serverSocketPort;
}
```

```
public void run() {
    startServer();
}
```

```
//sunucuyu başlatır
```

```
public void startServer() {
```

```
    if (serverPortIsUsed)
```

```
        return;
```

```
    servergui.getInstance().writer("server basliyooooor");
```

```
    serverPortIsUsed = true;
```

```
    try{
```

```
        if ( true ){
```

```
            ServerSocketFactory
```

```
serverSocketFactory=getSSLContext().getServerSocketFactory();
```

```
        serverSocket = serverSocketFactory.createServerSocket( serverSocketPort);
```

```
    }else {
```

```
        serverSocket = new ServerSocket( serverSocketPort );
```

```
    }
```

```
//serverSocket.setSoTimeout(1500);
```

```
while (isOpened){
```

```
    Socket socket;
```

```
    try{
```

```
        socket = serverSocket.accept();
```

```
        clientService client = new clientService( socket );
```

```
        servergui.getInstance().writer_client(""+socket.getInetAddress()+"");
```

```
        clientList.addElement( client );
```

```
        client.start();
```

```
        servergui.getInstance().writer("client baglandi");
```

```
    }
```

```
    catch(IOException ioEx){
```

```
    }
```

```
}
```

```
serverSocket.close();
```

```
}
```

```
catch(IOException ioEx){
```

```
}
```

```

    }

//sunucuyu durdurur.
public void stopServer() {
    isOpened = false;
    clientService temp = null;
    for ( int i=0; i<clientList.size(); i++) {
        temp = (clientService)clientList.elementAt( i );
        if ( temp==null )
            continue;
        if ( temp.alive )
            temp.clientServiceStop();
    }
}
/**
 * Returns the SSL context to be used in SSL communication for both the client
 * and the server side.
 */
protected com.sun.net.ssl.SSLContext getSSLContext(){
    long before = System.currentTimeMillis();

    com.sun.net.ssl.SSLContext sslContext = null;
    try {
        KeyStore keyStore = KeyStore.getInstance("JKS");
        keyStore.load(new FileInputStream("c://kemal//tezprogram//kod2002-01-
02//kod//testkeys"), "passphrase".toCharArray());
        KeyManagerFactory keyManagerFactory =
        KeyManagerFactory.getInstance("SunX509");
        keyManagerFactory.init(keyStore, "passphrase".toCharArray());
        TrustManagerFactory trustManagerFactory =
        TrustManagerFactory.getInstance("SunX509");
        trustManagerFactory.init(keyStore);
        sslContext = com.sun.net.ssl.SSLContext.getInstance("SSL");
        sslContext.init(keyManagerFactory.getKeyManagers(),
        trustManagerFactory.getTrustManagers(), null);
    } catch (Exception ie) {ie.printStackTrace();}

    return sslContext;
}
}

```