

T.C.

GEBZE YÜKSEK TEKNOLOJİ ENSTİTÜSÜ
MÜHENDİSLİK VE FEN BİLİMLERİ ENSTİTÜSÜ



DÜŞÜK MALİYETLİ VE GÜVENİLİR
ELEKTRONİK CÜZDAN UYGULAMASI

134332

Mehmet Ercan KURUOĞLU
YÜKSEK LİSANS TEZİ
BİLGİSAYAR MÜHENDİSLİĞİ
ANABİLİM DALI

TEZ DANIŞMANI
Yrd.Doç.Dr.İbrahim SOĞUKPINAR

GEBZE
2003

134332
GEBZE YÜKSEK TEKNOLOJİ ENSTİTÜSÜ
MÜHENDİSLİK VE FEN BİLİMLERİ ENSTİTÜSÜ

Mehmet..Erçan..KURUOĞLU.'ın tez çalışması, G.Y.T.E. Mühendislik ve Fen Bilimleri Enstitüsü Yönetim Kurulu'nun 06.02.2007 tarih ve 2007.1.5 sayılı kararıyla oluşturulan jüri tarafından BİLGİSAYAR Müh. Anabilim Dalında YÜKSEK LİSANS tezi olarak kabul edilmiştir.

JÜRI

ÜYE

(Tez Danışmanı)

:Y.Doç.Dr. İbrahim SOĞUKPINAR

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 02.06.2007 tarih ve 2007.1.16 sayılı kararı.

İmza / Mühür
Prof. Dr. Adem KILIÇ
GYTE Mühendislik ve Fen Bil.
Enstitüsü Müdürü

ÖZET

Manyetik kartlı ödeme sistemlerinin hırsızlıklara açık olması, yeni ödeme sistemleri üzerinde çalışmayı beraberinde getirdi. Son 20 yılda mikroişlemcili akıllı kartların ve akıllı kartlarda uygulanan güvenlik protokollerinin gelişmesiyle, manyetik kartla ödeme sistemlerine göre çok daha güvenli olan elektronik cüzdan ödeme sistemleri gündeme geldi. Fakat elektronik cüzdan sistemi, altyapı ve uygulama açısından daha zor ve pahalıdır. Üstelik off-line elektronik ödeme sistemlerinin güvenlik açıkları bulunur. Üçüncül kurumların devreye girmesiyle uygulanabilen PKI çözümlerinin bazı riskleri vardır. Akıllı kartlar tam güvenli değildir.

Bu çalışmada, halen dünyanın çeşitli bölgelerinde geçerli olan elektronik cüzdanla ödeme sistemlerine nispeten daha ucuz, kolay ve hızlı, aynı zamanda onlar kadar güvenli bir yöntem geliştirilmiş ve uygulamada sınanmıştır. Bu yöntemde, basit bir kullanıcı doğrulama ve yetkilendirme protokolü kullanılarak, mikroişlemcili akıllı kartlardan çok daha ucuz, fakat güvenliği daha az olan hafıza korumalı akıllı kartlar üzerinde güvenlik artırılmıştır. Kullanılan güvenlik protokolünde, şifre ve doğrulama tabloları veritabanında tutulmaz ve merkez sunucu bilgisayara kullanıcı şifresi gönderilmez. Aynı zamanda, zaman aralığı yöntemiyle güvenlik ihlalleri önlenir. Uygulamaya geçirilen bu çalışmada az güvenli kabul edilen hafıza korumalı akıllı kartlar üzerinde, mikroişlemcili akıllı kartlar kullanan elektronik cüzdan sistemlerine denk güvenlik elde edilmiştir.

Akıllı kartla bir sistem tasarlamadan önce akıllı kartları tanımak; sistem ihtiyaçlarını tespit etmek; sistemin zayıf noktalarını öngörmek gerekir. Bunun için ülke çapında makro planda bir çalışmaya gitmeden önce; mikro planda, kapalı alanda bir uygulama ortaya kondu. Akademik personel, öğrenci ve memur gibi birçok elemanın bulunduğu üniversite alanı uygulama için seçilmiştir. Elektronik cüzdan kafeterya, yemekhane ve kütüphanede kullanılabilir. Kartı yüklemek ve para değiş tokuşu yapmak için bir merkez yeterlidir.

SUMMARY

In the case of being open to intruders, payment with magnetic cards force to study on new payment systems. With the improvement of smart cards and security schemes which applied on them in the last 20 years, electronic purse payment systems which are more secure than magnetic card systems become a current issue. However electronic purse system is difficult and expensive in the case of infrastructure and application. And also off-line electronic payments have security gaps. There are some risks in PKI solutions, which are applicable with third parties included to the system. Smart cards are not fully trusted.

In this study, a new electronic purse method is improved and tested which is more cheap, simple and faster than, but also as secure as electronic purse systems still applied in some regions of the world. In this method, by using a simple user authentication scheme, security is increased on protected memory smart cards which are more cheap but less secure than smart cards with microprocessor. At the scheme, password or verification tables are not held in database and no user password is sent to the central server computer. In addition, by the time interval method, security breaks are prevented. In this application, security obtained on protected memory smart cards, which is concerned as less secure, matches the electronic purse systems which use microprocessor smart cards.

Before planning a system with smart cards, it is needed to learn smart cards, to define system requirements and to foresee the weak points of the system. So, before studying at macro plans, the whole country; an application is proposed in a micro plan, closed area. University campus which includes Academic personel, students, government workers and etc. is choosen as an application area. Electronic purse can be used in cafes, restaurants and library. To load the card and make money transactions a center place is required.

TEŞEKKÜR

Tez çalışmasında büyük emeği geçen, yılmadan yardım eden tez danışmanım Sayın
Yard. Doç. Dr. İbrahim Soğukpınar'a çok teşekkür ederim.



İÇİNDEKİLER DİZİNİ

vii

Sayfa

ÖZET	iv
SUMMARY	v
TEŞEKKÜR	vi
İÇİNDEKİLER DİZİNİ	vii
ŞEKİLLER DİZİNİ	ix
ŞEMALAR DİZİNİ	x
1. GİRİŞ	1
2. ELEKTRONİK CÜZDANDA GENEL KAVRAMLAR	4
2.1. Akıllı Kartlar	4
2.1.1. Manyetik Kartlara Göre Avantajları	4
2.1.2. Akıllı Kart Tipleri	6
2.1.3. Kart Okuyucuları	9
2.1.4. Akıllı Kartlarda Veri Güvenliği Sağlanması	10
2.1.5. Akıllı Kartlarda Güvenlik Açıkları	11
2.2. Sistem Tasarımı	12
2.3. Para Yükleme-Ödeme Sistemi ve Elektronik Cüzdanın Bazı Dezavantajları	12
2.4. Açık ve Kapalı Sistem Mimarisi	14
2.5. Sistem Yapısı, Terminal Bağlantısı	14
2.6. Sistem Seçeneklerinde Temel Mimari Yapı	15
2.7. Kullanılacak Kart Tipi	17
2.8. Veri Seti Gereksinimi	17
2.9. Güvenlik	18
2.10. Başarım Hataları	22
2.11. Bilgi Aktarımı	23
2.12. Terminaller	24
2.13. Ödeme Planı	25

2.14. İşlemlerin Genel Yapısı	26
2.15. Genel Bir Elektronik Cüzdan Uygulamasında İşlemler	27
2.15.1. Karşılıklı yetkilendirme (kart ve terminalin birbirini tanıması)	27
2.15.2. Cüzdan hesabını okuma	28
2.15.3. Borçlandırma işlemi	29
2.15.4. Borç işlemi imzası	29
3. KAMPÜS ORTAMINDA ELEKTRONİK CÜZDAN UYGULAMASI	30
3.1. Tasarım	31
3.2. Genel Yapısı	31
3.3. Genel İşleyiş	32
3.4. Kullanılan Donanım ve Kartlar	33
3.5 Güvenlik	34
3.6. Bilgi Güvenliği ve Yetkilendirme Protokolü	35
4. GÜVENLİK ANALİZİ	39
5. SONUÇ	40
KAYNAKLAR	41
ÖZGEÇMİŞ	43
EK: AKILLI KARTLA ELEKTRONİK CÜZDAN UYGULAMASI İÇİN DELPHI5 KULLANILARAK PASCAL DİLİNDE YAZILAN PROGRAM	44
EK.1. Merkez bilgisayarda bulunan sunucu programı	44
EK.2. Kart yükleyici bilgisayarda bulunan program	55

ŞEKİLLER DİZİNİ



Şekil

2.1: Hafıza kartı iç yapısı	7
2.2: MKÜ kartlarının iç yapısı	8
2.3: E-ödeme sistemi örneği. (SAM, <u>S</u> ecurity <u>A</u> ccess <u>M</u> odule: güvenlik erişim modülü) [2]	16
3.1: Üniversite alanındaki elektronik cüzdan genel yapısı	34
3.2: Yükleyici bilgisayarındaki program	37

ŞEMALAR DİZİNİ



Sema

Sayfa

2.1: Akıllı kart tipleri	6
2.2: DES Algoritması	10
2.3: Üçlü DES Algoritması	11



1. GİRİŞ



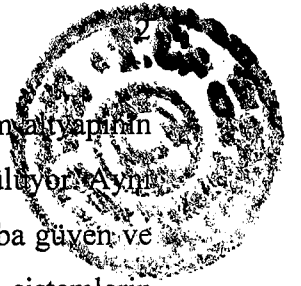
Visa ve Master Card gibi manyetik kartlı ödeme sistemlerinde birçok problemin üstesinden gelinmeye çalışılmıştır. En önemlisi güvenlik problemidir. Veri iletiminin güvenliği, merkez bilgisayarlarda şifre tutma, kimlik kontrolü gibi önlemlerin tam bir güvenlik sağlayamaması üzerine; kart güvenliği, güvenli kartlarla ödeme sistemleri üzerinde yoğunluklu çalışılmaya başlandı.

İşte, dünyanın birçok bölgesinde kredi kartlarıyla yer değiştiren akıllı kartlar yakın gelecekte, önlenemez bir şekilde para yerine kullanılmaya başlanacaktır. Literatürde elektronik cüzdan diye adlandırılan bu sistem temelde banka gibi bir merkeze yatırılan paranın akıllı kartlarda hesap bilgisi olarak tutulmasına dayanır. Müşteri, alışveriş sırasında kartını hizmet veya eşya alacağı birimin(mesela mağaza) terminaline takarak, kimlik doğrulama ve kart yetkilendirme işlemini başlatır. Kimlik doğrulanması ve kart yetkilendirmesi gerçekleşirse; karttan, üzerindeki elektronik paranın işlem için gerekli olan miktarı düşülür. Daha sonra bu para merkezden mağaza sahibine aktarılır. Hem off-line(kablo bağlantısız), hem de on-line(kablo bağlantılı, merkezle haberleşerek) ödeme yapılabilen bu sistemde güvenlik:

1. Kartların ve kartlarda kullanılan şifreleme protokollerinin güvenliğine,
2. Kart-terminal, terminal-merkez banka arasındaki veri iletiminin güvenliğine,
3. Kart-terminal-merkez üçlüsünün, işlemlerden önce birbirini doğrulama ve yetkilendirmesine dayanır[1].

Elektronik cüzdan sistemi müşterilere, nakit para taşımama, paraları güvenli bir yerde tutma, harcadıklarının hesabını bilme ve daha hızlı ödeme gibi; merkez ve mağazalara güvenilir işlem yapma, ön ödemedan dolayı daha fazla nakite kavuşma gibi birçok avantaj kazandırır. Fakat bütün bu avantajları geçersiz kılmamak için yeterli güvenlik koşullarını sağlamak gerekir.

Günümüzdeki elektronik cüzdan uygulamaları güvenlik için büyük masraflar yaptı. Açık Anahtar Altyapılı(PKI: Public Key Infrastructure) bir karşılıklı yetkilendirme sistemi ve gelişmiş mikroişlemcili kart kullanımı, dünya çapında



standart olarak kabul görüldü [2]. Fakat hem kartların pahalı olması hem altyapının pahalıya gelmesi elektronik cüzdan sisteminin bir dezavantajı olarak görülmüştür. Aynı zamanda açık anahtar sistemi için dijital sertifika dağıtan üçüncü bir gruba güven ve para yatırımı işleri daha da zorlaştırdı. Halbuki açık anahtar altyapılı sistemlerin birçok riski vardır [9]. Üstelik akıllı kartların bazı güvenlik açıkları bulunmaktadır [10], [11], [12], [13].

Bu çalışmada, halen dünyanın çeşitli bölgelerinde geçerli olan elektronik cüzdanla ödeme sistemlerine nispeten daha ucuz, kolay ve hızlı, aynı zamanda onlar kadar güvenli bir yöntem geliştirilmiş ve uygulamada sınanmıştır. Bu yöntemde, basit bir kullanıcı doğrulama ve yetkilendirme protokolü kullanılarak, mikroişlemcili akıllı kartlardan çok daha ucuz, fakat güvenliği daha az olan hafıza korumalı akıllı kartlar üzerinde güvenlik artırılmıştır.

Akıllı kartla bir uygulama veya sistem tasarlamadan önce akıllı kartları tanımak; sistem ihtiyaçlarını tespit etmek; sistemin kırılgan noktalarını öngörmek gerekir. Bunun için global, bankalar arası ve ülke çapında makro planda bir çalışmaya gitmeden önce; mikro planda, kapalı alanda bir uygulamaya gitmek gerekir. Böylelikle, akademik personel, öğrenci ve memur gibi birçok elemanın bulunduğu üniversite alanı uygulama için seçilmiştir. Elektronik cüzdan kafeterya, yemekhane ve kütüphanede kullanılabilir. Kartı yüklemek ve para değiş tokuşu yapmak için bir merkez yeterlidir.

Bu çalışmada ilk önce akıllı kartlar ve okuyucuları tanıtılıp, bilişim teknolojilerinde bilgi gizleme üzerine geliştirilen ve projede kullanılan algoritma ve uygulamalardan bahsedildi. Ardından akıllı kartlardaki güvenlik açıklarına değinildi. Daha sonra bu kartların, elektronik cüzdana yönelik dünyada ve literatürde uygulanan yöntemleri anlatıldı. Elektronik cüzdanın sahip olması gereken özellikleri (kartları, kart işletim sistemleri, okuyucuları, kart-terminal terminal-merkez haberleşmesi, verilerin şifrelenmesi ve saklanması, güvenilir sistem uygulayıcıları) standartlarda belirtilmiştir. Ayrıca elektronik cüzdan sisteminde karşılaşılan problemler konu edildi. Sonraki bölümde bu standartların dışına çok taşmadan, tam da bağlı kalmadan bir elektronik cüzdan sistemi önerildi. Merkeze bağlı, kapalı, üniversitenin sınırlı alanında, on-line(kablo bağlantılı) olarak yapılan bu uygulama,

standartların önerdiği, bankaların uyguladığı sistemlerden daha basit ve ucuz olma özelliği taşıyor. Fakat bu basitlik ve ucuzluk hiçbir şekilde sistemin, bilgilerin veya kartın güvenilir olmadığını göstermez. Nitekim [3] ve [4]'de anlatılan basit bir protokolle yeterli güvenlik elde edilmiştir.

Kullanılan güvenlik protokolünde, şifre ve doğrulama tabloları veritabanında tutulmaz. ve merkez sunucu bilgisayara kullanıcı şifresi gönderilmez. Aynı zamanda, zaman aralığı yöntemiyle güvenlik ihlalleri önlenir. Uygulamaya geçirilen bu çalışmada az güvenli kabul edilen hafıza korumalı akıllı kartlar üzerinde, mikro işlemcili akıllı kartlar kullanan elektronik cüzdan sistemlerine denk güvenlik sağlanmıştır.



2. ELEKTRONİK CÜZDANDA GENEL KAVRAMLAR

2.1. Akıllı Kartlar

Akıllı kartlar, içine mikro bilgisayar yerleştirilmiş olan plastik kartlardır. Veri iletimi, saklanması ve işlenmesi için mikro işlemci, ROM + RAM hafıza gibi elemanlara sahiptir. Kartla haberleşme yüzeyindeki kontak noktalarıyla elektriksel olarak veya kontaklı elektromanyetik dalgalarla olur. Manyetik kredi kartları, telefon kartları ve bankamatik kartları gibi kartlardan binlerce kere daha fazla bilgi yükleme ve saklama kapasitesine sahiptir. EEPROM yapılı veri hafızası, hesap numarası, hesap tutarı, ev adresi, üyelik kodu, kullanıcı giriş çıkış zamanları, sağlık bilgileri ve hatta şifrelenmiş verileri saklayacak kadar esneklik. Şu anda bazılarında 20Kbyte'ı [2] aşan hafızası gelişen teknoloji ile birlikte daha da artacaktır. İçindeki aktif mikro işlemci hafızaya ulaşmayı kontrol eder. Böylelikle gizli bilgiler, dışarıdan okunamayacak fakat kullanıcı yetkilendirmesini (kimlik doğrulama, okuma, yazma) sağlayacak şekilde hafızada saklanabilir. Bu özellikleriyle akıllı kartlar gizli veri içeren çok çeşitli uygulamalarda kullanılabilir. Yine bir elektronik cüzdan örneği olarak verilirse: Mağazalar peşin ödeme yapıp limitini belirleyen müşterilerine bu kartları dağıtır. Daha sonra müşteriler mağazanın herhangi bir bölümünde nakit para götürmeden, gereksiz bozukluklarla uğraşmadan alışveriş yapabilirler. Akıllı kartlardaki veriler güvende olduğundan mağaza dolandırıcılık yapılmayacağından emin olur.

2.1.1. Manyetik Kartlara Göre Avantajları

Akıllı kartlar, manyetik kartlardan farklı olarak off-line (kablo bağlantısız) çalışabilirler. Kart terminalleri verimliliği ve güvenliği azaltmayacak şekilde akıllı kartları işleme sokarlar. ATM, POS makineleri gibi on-line (kablo bağlantılı) servislerde, her bir terminal ana sunucuya bağlıdır ve aynı anda sadece bir işlem

gerçekleştirebilir. Akıllı kart terminalleri özellikle meşgul saatlerde sunucuya bağlı olmak zorunda değildir. Kart sahibinin bilgileri sunucu makinenin büyük veritabanında saklanmak yerine akıllı kartın hafızasında yer alır. Böylelikle kart sahibinin kişisel gizliliğini sağlar. Çünkü bilgiler her zaman kart sahibiyle beraberdir ve hiçbir yetkisiz kişi o bilgilere ulaşamaz.

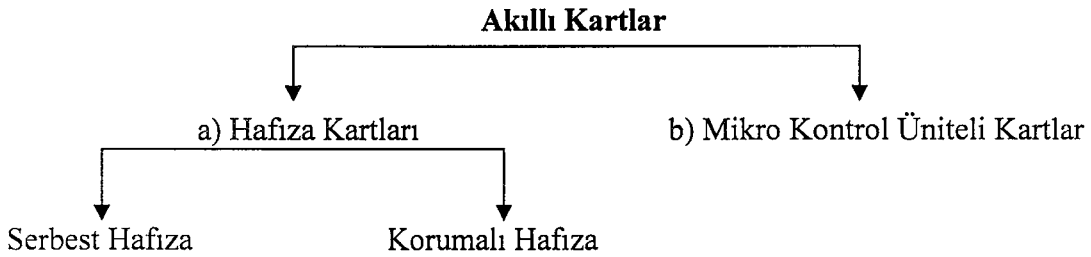
Akıllı kartların manyetik kartlara göre en önemli avantajı, içindeki bilgilere yetkisiz ulaşımına ve sahteciliğe izin vermemesidir. Veri erişimi, ancak kart içindeki işletim sistemi ve güvenlik devre sistemi aracılığıyla seri ara yüzü vasıtasıyla olur. Hafızanın belli bölümlerine ulaşmak bazı şifre kodlarıyla korunmuştur, o kodlar olmadan veriler yazılıp okunamaz. Bazı kartlarda, sadece o kartın aritmetik ünitesinin işleyebileceği gizli bilgiler saklanabilir. Yine akıllı kartların esnekliğine bağlı olarak, ulaşım kodlarını sadece kart sahipleri, kart dağıtıcıları bilebilir veya kart terminalleri uygulamalara göre kendi hesaplayabilir. Bu özellik, ulaşım kodlarının kişi belirleme numarası (Personal Identification Number: PIN) olarak kullanılıp off-line yetkilendirmesine imkan verir. Prensipte olarak, hafızadan veri okunması, yazılması, silinmesi kart donanımı ve içindeki yazılım tarafından kontrol edilir. Kripto algoritmalarını hesaplamalarıyla birlikte akıllı kartlar tam bir güvenlik modülü olarak kullanılabilir.

Manyetik kartların ömrü bir-iki yıl olduğu halde akıllı kart üreticilerinin birçoğu on yıl gibi uzun ömrü garanti ederler.

Akıllı kartlar, ilk üretildiğinde veri içermezler. Kartlar program geliştirici tarafından bütün sistemde nasıl davranacağını belirlemek üzere programlanır. Akıllı kart içeren sistemlerin tasarımında kartın bütün potansiyelini nasıl en iyi şekilde kullanılacağı çok iyi düşünülmelidir. Akıllı kart uygulamalarına örnek olarak şunlar verilebilir: Elektronik Cüzdan, Sağlık Sigorta Kartları, Üyelik Kartları, Otoyol Ücret Ödeme Kartları, Sürücü Ehliyeti, Kimlik Kartı, Pasaport, Banka Kartları.

2.1.2. Akıllı Kart Tipleri

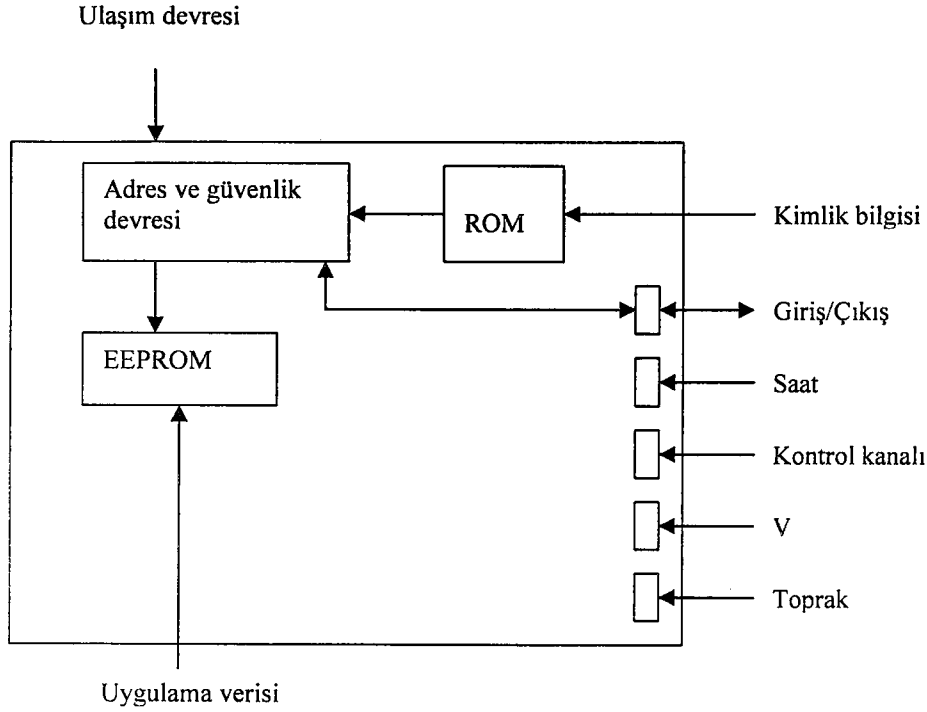
Akıllı kartların temel özellikleri ve işlevleri ISO 7816 standartlarında belirtilmiştir. Bu standartlara ve hatta fiyatlarına göre iki ana gruba ayırmak mümkündür.



Şema 2.1: Akıllı kart tipleri

a) Hafıza Kartları:

Uygulamada kullanılacak bilgiler daha çok EEPROM hafızasında saklanır. Hafızaya ulaşma, yani basit manasıyla yazma ve silme bir güvenlik devresi tarafından kontrol edilir. Okuma, yazma ve korumalı hafızaya erişimi değiştirme, bankamatik kartlarının şifresine benzeyen bir erişim kodu ile belirlenmiştir (veya engellenmiştir). Karttan okunmadan veya karta yazılmadan önce erişim kodu girilmesi gerekir, belli sayıdaki yanlış şifre girme kartı bloke eder ve kart geçici olarak sadece okunabilir. Sadece okunur kartlar kullanışsızdır, çünkü bütün işlemler karttaki veriyi değiştirmeyi gerektirir. Bazı kartlarda basit kodlama içeren daha gelişmiş devreler vardır. Veri iletimi giriş/çıkış portu veya I²C (Inter-Integrated Circuit: içe gömülmüş devre) veri yolu vasıtasıyla olur. Korumalı hafıza kartları kullanıcının eline önceden tanımlanmış 2-3 byte uzunluğundaki giriş şifresiyle gelir, kullanıcı istediği zaman bu şifreyi değiştirebilir. Serbest hafıza kartlarında ise böyle bir giriş şifresi yoktur ve veri güvenliği gerektirmeyen uygulamalarda kullanılır. Mikro işlemcili kartlara göre daha az güvenlik içerdiği, daha ucuz olduğu için ön-ödemeli telefon kartı, sağlık kartı gibi ileri seviyede güvenlik istemeyen uygulamalarda kullanılır. Fakat bu projede, koruma hafızalı akıllı kart ile ileri seviyede güvenli elektronik cüzdan uygulaması başarı ile gerçekleştirildi.



Şekil 2.1: Hafıza kartı iç yapısı

Bazı korumalı hafıza kartlarının tiplerini incelersek:

Tekrar Yüklenebilen Sayıcı Kartlar : Karta her yazıldığında sayıcısını azaltacak 104 bitten 221 bite kadar kapasiteye sahiptir. Mikro kontrol ünitesine sahip kartlar gibi yetkilendirme veya kullanıcı aşamalarında bulunabilir. Kullanıcı aşamasında iken sayıcı tekrar programlanamaz ve tekrar yetkilendirme aşamasına geçemez. Sayıcı sıfıra ulaştığında ise kart kullanılamaz duruma gelir. Örnek olarak telefon kartlarını verebiliriz. Yetkilendirme aşaması hariç kartın sayacı hiçbir şekilde arttıramaz.

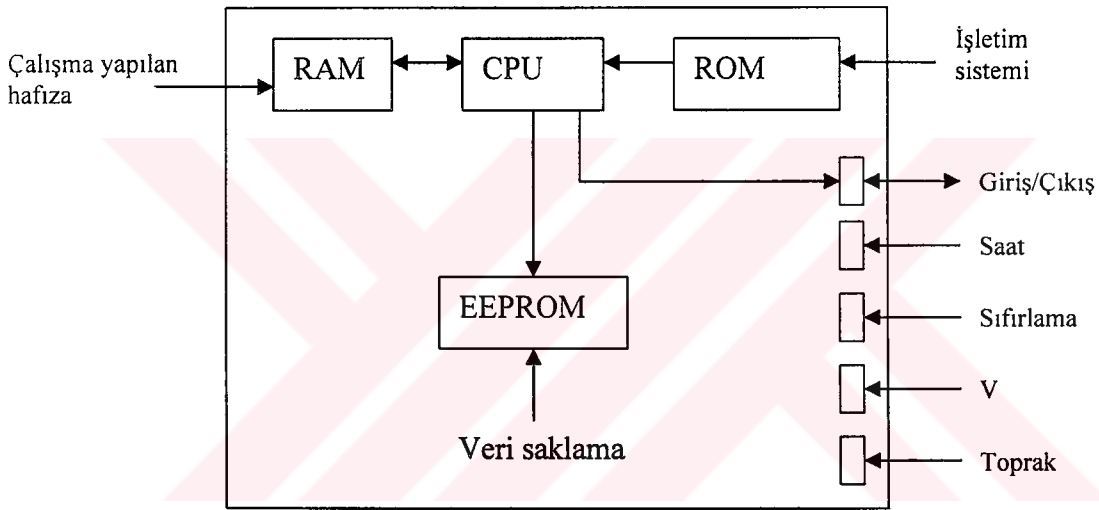
Tekrar Yüklenebilen Sayıcı Kartlar : Kapasitesi 418 bitten 896 bite kadar değişir. Karta yazmayı veya okumayı engelleyen bir güvenlik kodu vardır. Bu kod 0'dan 255'e kadar değişen iki numaradan oluşur. Ön ödemeli elektrik, su, doğalgaz kartları gibi orta güvenli uygulamalarda kullanılır.

Siemens SLE 4418/4428 Hafıza Kartı : Kapasitesi 1 Kbyte'tır. Bazı hafıza adresleri önemli veriler için yazmaya karşı korumalı yapılabilir. Bu kartlarda sağlık bilgilerini saklamak uygundur. Bu tez çalışmasında görülebileceği üzere değişik

şifreleme teknikleri kullanarak Siemens SLE 4428 hafıza kartı elektronik cüzdan uygulamasında da kullanılabilecektir.

Siemens SLE 4432/4442 Hafıza Kartı : Kapasitesi 2 K bitten 8 K bite kadar değişen şifre korumalı hafızaya sahiptir. Kartın okuma hafızası serbesttir. Yazma hafızası ise 3 byte'tan oluşan şifre ile korunur ve 4 defa yanlış şifre girme hafızayı bloke eder. Kullanıcı gerektiği zaman 2^{24} farklı kombinasyondan oluşan şifreyi değiştirebilir.

b) Mikro Kontrol Ünitesine Sahip Kartlar (MKÜ Kartlar):



Şekil 2.2: MKÜ kartlarının iç yapısı

CPU (Central Processing Unit) – Merkezi işlemci ünitesi : Veriyi işler, ROM, EPROM veya EEPROM'daki programları çalıştırır.

RAM (Random Access Memory) – Gelişigüzel erişilebilen hafıza : Hesap sonuçlarını saklar, geçici hafızadır.

ROM (Read Only Memory) – Sadece okunabilen hafıza : Kartın işletim sistemini, sabit verileri, standart fonksiyonları saklar.

Bu kartlar hafızasını dosyalara organize eder. Karta ait bütün erişimler kartın içindeki merkezi işlemci tarafından kontrol edilir. Bu işlemci erişimin

yetkilendirildiğine veya yetkilendirilmediğine karar verir, şifreleme ve onaylama işlemlerini yapar. Bu yüzden kartın hafıza adreslerine direkt ulaşım yoktur. Hafıza kapasitesi 512byte'tan 16Kbyte'a ve gelişen teknolojiyle daha ilerisine kadar çıkar. MKÜ kartlar 8 byte uzunluğunda 7 erişim kodu taşır, böylelikle kullanıcı kart dosyalarını koruma altına alır. MKÜ kartlar iki aşamada bulunabilir : **Kişileendirme** (yetkilendirme), **kullanıcı** durumları.

Yetkilendirme aşamasında, kullanıcı veya sistem geliştiricisi kartın ne kadar dosya taşıyacağını belirler. Bütün dosyalar için dosya numaraları, boyutları ve güvenlik davranışları tespit edilmelidir. Dosya güvenlik davranışları okuma ve yazma haklarını sınırlandırmak için erişim kodlarının kombinasyonundan oluşur. Sistem geliştiricisi gerekli olan erişim kodlarını koşullandırıp işleme sokmalıdır. Bundan sonra kart kullanıcı aşamasına geçer.

Kullanıcı aşamasında yetkilendirme sırasında yapılan bütün özellikler harekete geçer. Kartı dağıtan firmanın kodu girilerek tekrar yetkilendirme aşamasına dönülebilir. Bu kod 7 erişim kodundan biridir.

2.1.3. Kart Okuyucuları

Kart okuyucuları bir ana (host) makineye mesela bir bilgisayara bağlı olup gelecek komutları bekler. Akıllı kart okuyucuları genellikle seri asenkron arayüzle haberleşirler (RS-232).

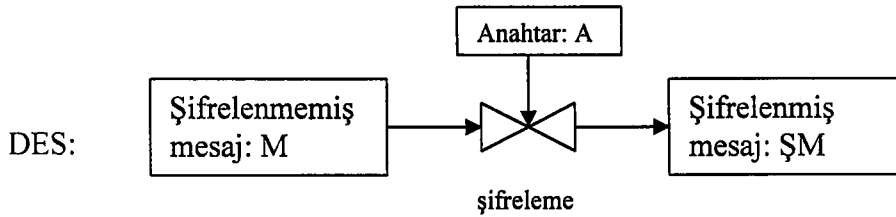
Farklı marka ve tipteki kartlar farklı komutlar ve haberleşme protokolü kullanırlar. Bu da akıllı kartlarla bilgisayarlar arasındaki direkt haberleşmeyi engeller. İşte akıllı kart okuyucuları bu haberleşmenin sabit olmasını sağlar. Karta ait özellikler dikkate alındığında, bilgisayar programcısı akıllı kartlarla ilgili pek fazla detayı bilmesine gerek yoktur.

Akıllı kartlarla okuyucuları arasındaki arayüzde ISO 7816 standartları kullanılır. Böylelikle farklı markadaki akıllı kart ve okuyucu üreticileri birbirleriyle

uyumlu olurlar. Kart ölçüleri, mikroçip pozisyonu-boyutları, markasına ve tiplerine bakmadan bütün akıllı kartlarda aynıdır. Sadece kart üreticilerinin yazılım arayüzleri farklılık gösterir. Bazı akıllı kart okuyucuları gizli kodların karşılıklı değişimi üzerine kurulan karşılıklı yetkilendirme özellikleri sunarlar.

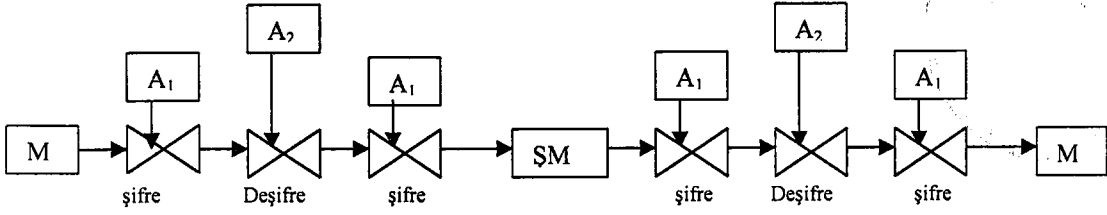
2.1.4. Akıllı Kartlarda Veri Güvenliği Sağlanması

DES(Data Encryption Standard) – Üçlü DES Algoritmaları: Veriyi şifrelemek ve şifreyi çözmek için aynı(simetrik) anahtar kullanma prensibine dayanır. Kerckhof'un prensibine göre düzenlenmiştir [2], güvenlikten bir şey kaybetmeden yayınlanabilir. Auguste Kerckhof (1835 - 1903) prensibi ise algoritmanın bütün güvenliğinin algoritmaya değil, kullanılan anahtarın gizliliğine dayandığını söyler. Algoritma karışıklık ve dağılma üzerine kuruludur. Karışıklık, şifrelenmemiş mesajla şifrelenmiş mesajın istatistiklerinin birbiriyle alakalı olmasıdır. Böylelikle şifreyi kırmak isteyen fayda sağlanmamış olur. Dağınıklık ise, şifrelenmemiş mesajın hemen bütün bitlerinin, şifrelenmiş mesaja etkisine dayanır. Algoritmada şifrelenmemiş ve şifrelenmiş mesaj bloklarının ve anahtarın uzunluğu 8byte'dır.



Şema 2.2: DES Algoritması

Üçlü DES'in DES'ten farkı, 16byte'lık iki anahtar kullanması ve işlemleri bir döngü içerisinde gerçekleştirmesidir. Bütün üç anahtarda aynı olduğunda yine DES algoritması sonucunu verir:



Şema 2.3: Üçlü DES Algoritması

Tek Yönlü (Hash) fonksiyonları: Değişken uzunluktaki bir dokümandan veya veriden sabit uzunlukta bir çıktı alır. Çıkış sıkıştırılmıştır ve giriş elde etmek için geri dönüşüm yapılamaz. Giriş blokları istenilen boyutta değilse ekleme yapılır.

2.1.5. Akıllı Kartlarda Güvenlik Açıkları

Bütün önlemlere rağmen akıllı kartlarda bazı güvenlik açıkları bulunmaktadır. Dolandırıcılar gün geçtikçe daha yetenekli bir şekilde güvenlik saldırıları yapmaktadırlar. Birkaçını ele alırsak: korumalı hafıza hücrelerinin içeriği okunabilir hatta değiştirilebilir. Bunun için veri yollarına problemler takılır, bazı veri yolları kesilebilir, yeni devreler eklenebilir. Bu tür fiziksel saldırılar karta zarar verir ve kart kullanımdışı kalır, ayrıca uygulanması pahalıdır [10]. Ayrıca, kriptanaliz sırasında zaman düzensizliklerini [11] ve enerji harcama sırasındaki güç kullanımını düzensizliklerini [12] tespit etmek mümkündür. Osiloskopla bile yapılabilen bu işlem, hesaplamalardaki verileri, hatta anahtarları ortaya çıkarabilir. Diğer bir işlem ise akıllı kartı alışılmışın dışında kullanarak işlemcinin hata yapmasını sağlamaktır [13]. Bu tür hileler sistemde ciddi kayıplara yol açmaktadır. Akıllı kartlarda para bilgisi saklandığı düşünülürse kart güvenliğinin yanında başka işlemler de gerekmektedir.

2.2. Sistem Tasarımı

Akıllı kart sistem uygulamasını başlatmadan önce bazı problemleri göz önüne almak gerekir. Sistemin altyapısı, akıllı kartın tipini ve yapılacak olan masrafları etkiler. Bütün sistemin işleyişi ve yapısı üzerinde yeterince düşünmek ve şu sorulara cevap vermek gerekir:

- a) Sistem tamamen off-line veya on-line mı, yoksa ikisinin kombinezonu şeklinde mi olacak?
- b) Veriler kartın üzerinde mi kalacak yoksa Server(sunucu) makinesine mi gönderilecek?
- c) Eğer bu çok uygulamalı bir sistemse, veriler değişik bilgisayarlarda mı yoksa kartın değişik bölgelerinde mi duracak?
- d) İlk kullanım ve sonrasında ne gibi güvenlik fonksiyonları uygulanacak?
- e) Gelecek güvenlik yapıları için ne gibi alt yapılar hazırlanacak?
- f) Hangi tip kart okuyucuları kullanılacak ve erişim kodları nasıl olacak?
- g) Kart okuyucu terminalleri hazır bulunmalı mı?
- h) Sistem harici terminaller devreye girebilmeli mi? Yetkilendirilmesi, sertifika verilmesi nasıl olacak?

Her bir alternatif kendine göre ekonomik veya uygulama olarak avantaj veya dezavantaja sahiptir. Elektronik cüzdan olarak bütün alternatifler karşılaştırılacaktır.

2.3. Para Yükleme-Ödeme Sistemi ve Elektronik Cüzdanın Bazı Dezavantajları

Kart tabanlı ödemelerde en basit yöntem, on-line yetkilendirme ile gerçekleşen manyetik kartlı ödeme sistemidir. Kart sahibinin kimliği, son kullanma tarihi, hesabın geçerliliği gibi şartlar karşılaştırılıp; kart sahibinin banka hesabından,

mağaza (POS terminal) sahibine yeterli miktarda para aktarılır. Akıllı kartlarda sistem prensip olarak aynıdır. Karşılıklı veya bağımsız yetkilendirme gerçekleştirildikten sonra, daha önce karta aktarılan elektronik para işlem görür; kullanıcının karttaki hesabından alacağı servis veya eşyanın parası düşülür, bu para terminal sahibinin hesabına geçer. Gerekirse kart veya okuyucu tarafından PIN (Personel Identification Number: kişi kimlik numarası) kontrolü yapılır. Daha sonra terminal sahibi bu değerın nakit olarak karşılığını merkezden alır. Fakat sistem kart kullanıcısına bazı dezavantajlar sunar.

İlk önce, karta elektronik yükleme için peşin ödeme yapılır. Yani gerçek para elektronik para ile değişecektir. Kart kullanıcısının bu kartı, en azından içindeki değerin bir kısmını birkaç hafta hatta daha uzun süre kullanmadığını düşünürsek finansal olarak zarara uğrayacaktır. Mesela paranın değerinin düşmesi (devalüasyon), ödeme yaptığı yerin faiz geliri gibi.

İkinci etapta, karta elektronik para yükleyen firma iflas edebilir. Böylelikle kart kullanıcısı, karttaki parayı kullanamaz hale gelecek, belki bütün parasını kaybedecektir. Akıllı kartın kullanıldığı ülkelerde bunun yasal çerçevesi bulunmaya çalışılıyor. Bu firmaların devlet güvencesi altındaki banka ve benzeri kuruluşlar olmasına dikkat ediliyor.

Üçüncü olarak, kart bozulursa içindeki değerler-veriler kaybolur. Eğer kart sahibi anonim ise (telefon kartı gibi) bu değer hiçbir zaman geri döndürülemez. Kart kullanıcısının belli olduğu durumlarda, kartı yükleyen operatördeki bilgiler ve son işlemler sayesinde, kartın içindeki değer yaklaşık olarak bulunabilir.

Uygulama ilk defa bakıldığında kolay görünse de, gerçeğe dönüştürmek için arka planda önceki soruları cevaplayacak ciddi bir çalışma yapmak gerekir: sistem mimarisi, kart tipi, hafıza kapasitesi, veri içeriği, güvenlik, hatalar, bilgi aktarımı, terminaller, uygulamaya geçiş, standartlar, alternatif teknolojiler...

2.4. Açık ve Kapalı Sistem Mimarisi

Açık sistem, temelde uygulamayı yürüten birçok firmanın idaresiyle, genel ödeme işlemlerinin organizasyonlar arasında yapılmasıyla gerçekleşir. Kapalı sistemde ise ödemeler tek bir sistem operatörüne yapılır.

Kapalı sisteme örnek olarak hafıza çipli telefon kartlarını verebiliriz. Kart terminale (yani kartlı telefona) takıldığında, görüşme yapılırsa içindeki sayaç belli miktarda azaltılır. Terminal haricinde bu sayaç azaltılamaz ve operatör haricinde kart tekrar doldurulamaz. Operatör ve terminaller tek bir telefon şirketine aittir. Böylelikle dolandırıcılık önlenir.

Açık sistemlerde terminaller ve elektronik cüzdan dağıtıcıları, operatörleri çok çeşitli olabilir. Bu yüzden elektronik cüzdan dağıtıcıları terminalden gelen bilgilerin, işlemlerin doğruluğunu tespit etmelidir. Şu andaki Visa ve Master kart sistemleri açık sisteme örnek olabilir.

2.5. Sistem Yapısı, Terminal Bağlantısı

Merkezi bir sistemde, terminaldeki bütün işlemler sistem operatörü tarafından anında denetlenmeli; operatörün izin vermediği veya bağlantının yapılamadığı durumlarda işlem gerçekleşmemelidir. Yazılım değişiklikleri, şifreleme işlemleri gecikmeye uğramadan yapılır bu da sistem güvenilirliğini artırır.

Fakat merkezi sistemde, on-line altyapının kurulması ve bütün haberleşmenin üstesinden gelinmesi, kurulacak ağın güvenlik problemlerinin giderilmesi için firmalar ciddi bir ödenek ayırmalıdır. Bu da merkezi olmayan, on-line veya off-line çalışabilen güvenli akıllı kartları gündeme getirir. Örnek olarak, haftalarca veya aylarca bağlanmayı (veya kontrol edilmeyi) gerektirmeyen ve zaten haberleşme linklerine ihtiyacı olmayan kola ve çukolata makinelerini verebiliriz.

Merkezi sistemde bazı terminallerin göçmesiyle bütün e-ödeme sistemi bloke olabilir. Merkezi olmayan sistemde bu sadece arıza veren terminali etkiler. Fakat merkezi olmayan sistemde yönetim boşluklarını gidermek için belli periyotlarla haberleşecek on-line linkler kurulmalıdır. Bazı yerlerde en az günde bir işlem verisi terminalden merkeze, yönetim verileri (yeni yazılımlar, anahtar setleri, yenilenen hesap listeleri...) de merkezden terminale aktarılır. Böylelikle sistem güvenliği artırılır.

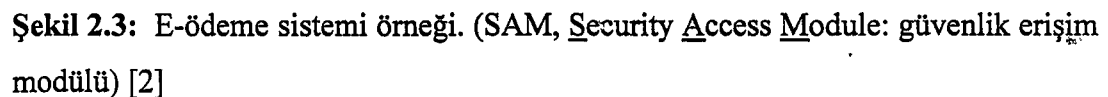
Yapılan alışverişin içeriğine göre işlemin merkezi olup olamayacağına karar verilmeli, merkezi olacak işlemde bağlantı sağlanamazsa işlem gerçekleşmemelidir. Bağlantı gerektiren işlemler genelde belli miktarı aşan büyük ölçekli (makro) ödeme dediğimiz kısımdır. Küçük ölçekli (mikro) ödemede buna ihtiyaç yoktur. Bu değerler her bir kart için farklı ölçeklenebilir. Ayrıca yapılan bağlantısız işlemlerin sayısı, en son yapılan bağlantılı harcamadaki geçen süre veya kartta tutulan rasgele bir sayıyla veya kasiyerin kart kullanıcılarından şüphelenmesiyle on-line sisteme geçilebilir. Hatta kola ve çukolata makinelerinde on-line altyapıyı kurmaya hiç gerek yoktur, satış elemanları sistemi arasına kontrol eder.

2.6. Sistem Seçeneklerinde Temel Mimari Yapı

Çok çeşitli yapılabilmelerine karşın ekonomik olması için mevcut altyapı kullanılmalıdır. Arka planda temizleme yani işlemlerin doğrulanıp gerçekleşmesi ve yönetim gelir. Mesela e-cüzdana yüklenenden fazla para harcanmamalıdır. Yönetim kısmında ise yeni uyarı listelerinin dağıtımı, yeni anahtar setlerinin belirlenmesi, kartın kişiselleştirilmesi esnasında alınacak verilerin belirlenmesi, güvenlik ve işlevliğinin tespiti gibi işler yapılır.

Ağ yapısı ister kablo yapısında (ISDN) ister paket yapısında (X25) olsun, temel olarak mesajlar geçirgen olmalıdır. Şifreleme hariç mesajlar bozulmaya uğramamalıdır.

Şekil 2.3’de sistem elemanları ve lojik bağlantıları gösteriliyor. Arka plandaki sistem geçirgen ağı çeşitli elemanlarla bağlanıyor. Hesaba para ekleme işi genellikle on-line olarak yapılır. Güvenlik modülüne sahip olduğu için ağıın bozulması durumunda off-line olarak da çalışabilir. Parkmetreler ve taksi terminalleri örnek olarak verilebilir. Transfer kartları güvenlik modülü yardımıyla işlem verisini transfer edebilir.



Şekil 2.3: E-ödeme sistemi örneği. (SAM, Security Access Module: güvenlik erişim modülü) [2]

Bu sistem çip içeren Eurocheque kartı ile Avusturya'da gerçekleştirilmiş. Daha büyük uygulamalar için arka planda ayırık ve paralel çalışan dağıtık sistem yapıları kullanılabilir.

2.7. Kullanılacak Kart Tipi

Elektronik cüzdan uygulamalarında genel olarak T=0 veya T=1 protokolüne sahip mikroişlemcili, yüksek güvenli kartlar seçilir. Bu kartlar diğerlerine göre çok daha pahalıdır. Fakat yeni teknikler geliştirilerek kartın donanımına inilmesi, şifreleme algoritmalarının ve şifrelerin kırılması, gün geçtikçe hızlanmaktadır. Böylelikle yüzde yüz güvenliğin sağlanamayacağı düşünülmektedir. Burada güvenliğin ne kadar zamanda kırılacağı önem kazanmaktadır. Fakat bu projede bütün genel kanılara ters olarak, elektronik cüzdan sistemi için 1Kbyte hafızaya sahip "Siemens SLE 4428 Hafıza Kartları" kullanılacaktır.

Manyetik kartlar dolandırıcılığa çok açıktır, hiçbir güvenlik ünitesi içermezler ve sadece on-line çalışabilirler.

MKÜ kartları ise hafıza kartlarına göre çok pahalıdır. Üstelik yetkilendirme, bilgi koruma için için güçlü bir şifreleme ve protokol altyapısı ister.

2.8. Veri Seti Gereksinimi

Veri setinden kastedilen kimlik bilgileri, hesap veya sayaç durumu, şifre gibi verilerdir. Minimum veri seti gereksinimi uygulamayla değişir. Bu çalışma, ticari olarak mevcut olan pek çok elektronik cüzdan sistemleri gibi, hafızanın yaklaşık 1Kbyte'ına gerek duyar. Kapalı sistem çevrelerde tasarımcı, kartın anonim olup olmayacağına karar vermelidir.. Bir anonim-kart yaklaşımında sistem kart kullanıcısının kim olduğunu bilemez ve umursamaz. Sonuç olarak, hafıza alanı isim, adres ya da hesap bilgilerine ayrılmak zorunda kalmaz. Diğer yandan, kayıp bir

kartın geri döndürülebilmesi, hırsızlığın önlenmesi veya elektronik olarak kilitlenebilmesi için, eğer kart kullanıcısı tespit edilmesi gerekirse, bazı kimlik bilgileri kartta saklanmalıdır. Bu senaryo, tasarımcının göz önünde bulundurması gereken ücret / verimlilik seçimlerinden sadece biridir.

Bu çalışmada hesap bilgisinin yanında, yetkilendirme için kimlik bilgisi ve şifre değerine ihtiyaç vardır.

2.9. Güvenlik

Elektronik cüzdan, her kart sahibinin ayrı bir hesaba sahip olma prensibine dayanarak çalışır. “Para” hem yasalar hem de kart yenileme açısından, merkez bilgisayarda tutulur. Kartı yüklemek, para eklemek ve boşaltmak, ödeme yapmak için gerekli olan güvenlik anahtarlarının yanında bir de bellek-hesap değeri, yani merkez bilgisayardaki hesabın bir kopyasını da saklar. Bu cüzdan dolayısıyla denetlenmiştir, yani değer kart sahibi ve satıcılar arasında hareket ederken para ana bilgisayarda izlenir. Fakat, kart hesapları şifrelendiğinden veya karıştırıldığından, sistem operatörünün belli bir kart sahibini bilmesi veya onu işleme sokmasının yolu yoktur.

Kart güncelleme, bir karttaki elektronik para hesabına ekleme veya hesaptan düşme yeteneğine sahip finansal servis işlemleri için ekstra güvenlik gereklidir. Günümüzde güvenlik sağlamak için en sık kullanılan metot bir güvenli erişim modülü kullanmaktır (endüstride SAM(Secure Access Modül) olarak bilinir).

Güvenlik kartları akıllı kartlarla aynı yapı ve dizayna sahipken, paketlenmesi ve yazılımı akıllı kartlardan farklıdır. Genellikle darbelere karşı korumalıdırlar. Güvenlik modülleri Siemens, Landis&Gyr, Schlumberger ve Gemplus gibi birçok üreticide mevcuttur.

Bu üreticiler yazılımda kart kaplamasına benzeyen bir güvenlik modülü kaplaması sunarlar. Güvenlik modülleri, kript algoritmaları için sunulan işlemciler,

cevap veren doğrulayıcılar ve anahtar üretimi için gerekli yapıya sahiptir. Bu bölümde sistemin diğer bölümlerinden daha fazla ek ve daha pahalı donanım ve yazılım koruması vardır.

Fakat hiçbir unsur tamamıyla güvenilir değildir. Tasarımcı sistem bütünlüğünün zarar görmemesini sağlamak için özellikle güvenlik modüllerinde kontrol ve dengeleri uygulamak zorundadır.

Güvenlik modüllerinin kendi kaplamaları, işletim sistemleri ve iletişim protokolleri vardır. Modüller üreticiden üreticiye değişir ve kart terminalinin donanım ve yazılıma eklenme metodu dikkatli bir analiz gerektirir. Güvenlik modül protokolleri genelde çok basittir. Akıllı kart bir anahtar mesajı doğrulama ya da sorgulama ve cevap isteği talebini başlatır. Güvenlik modülü talebi işleme koyar ve bir sonuç çıkarır. Bu kara kutular yazılıma uygulanabilir ve bazı üreticiler kendi güvenlik modüllerini terminalin yazılımına yerleştirmeyi tercih etmişlerdir.

Açıkçası, güvenlik modüllerini kart, mini-kart veya plug-in araçları şeklinde kullanmak bir fiziksel tedbir avantajını sunmaktadır. Eğer güvenlik yazılımı çözülmüş olsa, sadece yazılım değiştirme çözümünün aksine, modüller bütün sistemin entegresini bozmadan değiştirilebilir.

Güvenlik modüllerinin bilgi saklama ve işleme koyma yeteneği normal akıllı kartların yeteneğine göre çok azdır. Güvenlik modülü kullanan bir sistem tasarlanırken bu eksiklik dikkate alınmalıdır. Örneğin, eğer sistem tasarımı birçok değişik kartın terminal tarafından kabul edilmesine izin veriyorsa, güvenlik modülünün sadece iki ya da üç değişik tipte kartı destekleyecek bir üst sınırı olabilir. Ek kartları kabul etmek için, terminal devresine, ek güvenlik modüllerine izin verecek daha fazla boşluklar eklenebilir. Uygulamaya daha sonra eklendiğinde çok masraf verdiği için, daha tasarım aşamasında sistem içine konmalıdır.

Akıllı kartlar yaklaşık 20 yıldır piyasada olmasına rağmen, son beş-on yılda devlet kurumları, büyük telefon operatörleri, bankalar ve sigorta şirketleri PK (Public Key: açık anahtar) altyapılı tiplerine yöneldi. Son zamanlarda ise elektronik ticaret adı altında İnternet üzerinden evlere kadar ulaşan yeni bir pazar açıldı. Son olarak

gezici şirket veya şahıslar arasında ileri uygulama, haberleşme veya ticarete imkan veren mobil kablosuz altyapı hızla gelişti.

Son yıllarda İnternet gibi global ağlar araştırmacıları ve bilim adamlarını fikirleri ve bilgileri paylaşmaya yöneltti. Çünkü ticari aktivitelerde devlet organizasyonları, finans enstitüleri ve çeşitli işyerleri birbirleriyle haberleşmeye ve dahası ortak hareket etmeye başladılar. Hatta elektronik ticaret için altyapısı hazır (İnternet) ağı bağlanarak, gerekli bilgileri paylaşmak ve müşterilerle satıcılar arasında iş ilişkileri kurmak yeterli oldu. Fakat bu ilişkileri kurmak ve iletişimi sağlamak için bütün taraflar sanki yüz yüze işlem yapıyormuş gibi, elektronik işlemlerde de aynı güvene sahip olmalılar.

PKI [16], [17], (Public Key InfraStructure: açık anahtar altyapısı) ve dijital sertifikalar bu güveni sağlamak için oluşturuldu. Elektronik cüzdan sağlayıcısı sertifika almak için Sertifika Otoritesi'ne başvurur. Sertifika, sertifika dağıtıcısının gizli anahtarıyla sayısal olarak imzalanan açık anahtardır. Bu imzadan sonra akıllı kartın sayısal kimliği hazırdır. Sayısal kimlik ve gizli anahtar akıllı karta yazıldıktan sonra, kart kullanıcılara dağıtılır [15]. Sertifika Otoritesi sisteme sonradan dahil edilen üçüncül bir kurumdur. Fakat bu kuruma güvenmek ve yüklüce bir miktar ödeme yapmak elektronik cüzdan sistemlerinde üstesinden gelinmesi gereken problemlerdendir. Bunun yanısıra, yazarlar C. Ellison ve B. Schneier'e göre PKI sisteminin birçok riski vardır [9].

Bu sertifikalar kimliği, pasaport ve ehliyet gibi güvenilir ve ispat edilir hale getirir. PKI işler hale getirildikten sonra, güvenilir dijital imzalar emniyetli iletişimi sağlar. Bu elektronik imzanın önemi, sipariş, çek, kontrat ve para gibi kağıt doküman ve nesneler elektronik ortama geçtiği içindir. Elektronik ticaret, ev bankacılığı, İntranet ve İnternet, kimlik, sağlık bilgileri ve fiziksel erişimli kontrol servisleri gibi uygulamalarda dijital imzaya gereksinim çok fazladır.

Yazılım sistemleri uzaktan aktive edilebilen truva atları ve virüs saldırılarına karşı çok duyarlıdır. Dijital sertifikalar bilgisayarın hard diskinde bulunur ve hileli girişlere, kullanıma karşı hassastır. Güvenilir bir şekilde kimliği ispat eden bu sertifikaların kendisi, geniş bir alanda kullanılmadan önce korunmaya alınmalıdır.



Akıllı kartlar taşınabilir ve güvenli bir ortamda dijital sertifikayı kilitleyerek ve hakkı olmayanlara erişim izni vermemeyerek bu korumayı sağlarlar.

Açık anahtarları korumak için yapılan hesaplar, anahtar tarafından şifrelenen veya imzalanan mesajların güvenliği veya yetkilendirilmesi kadar güçlü olmalıdır. Akıllı kartlar açık anahtarlar için üst bir koruma düzeyi sunar. Sadece şifreyi veya PIN numarasını bilmek yetmez aynı zamanda karta da sahip olmak gerekir. Hatta ileri uygulamalarda parmak izi, yüz şekli gibi ekstra biyometrik sorgulama eklenebilir [14].

Akıllı kartlar taşınabilir ve rahat kullanımı vardır. Aynı zamanda tektir, yani kopyalanması neredeyse imkansızdır. Belli sayıdaki yanlış şifre girme ile erişim kullanıcıya kapanır. Akıllı kart sayesinde kullanıcı günlük hayatta daha az şifre ezberler. Bilgi teknolojileri yardım bölümüne gelen telefonların %40'ı çalınmış veya unutulmuş şifreler sebebiyledir. En azından buradaki para ve nakit kaybı azaltılır.

Bazı organizasyonlar güvenlik anahtarlarının akıllı kartlarda olmasını talep eder. Bir bankada kayıtlı olan firmanın elemanları, başka bir bankanın müşterisi olan bir firmanın temsilcisiyle iş yapar. Bankaların altyapısı işlemlere aracılık yapar ve işlemlerin içindeki riski yönetir.

Akıllı kartların güvenliğine gelince; bütün elemanları basit ve tek bir çipte toplanmıştır. Birden fazla çip olsaydı bağlantı kabloları güvenlik kaybına yol açacaktı. Aynı zamanda standartlarda belirtilen belli bir fiziki güce dayanıklıdır.

Akıllı kartların çipleri de yapı bakımından güvenlidir, birçok üretici en az 10 yıllık elektriksel bozulmamayı garanti eder. Böylelikle zaten arıza yapmadan önce modası geçer. Örnek olarak Fransız telefon kartının hata oranı 10000'de 3'ten daha azdır [2].

Çeşitli saldırılara karşı özel korumaları göz önüne alırsak:

• **Rasgele istatistikler:** Yazılım modüllerinin test fonksiyonları, hatalı verileri yakalamak için rasgele sayı üreticinin çıkışında çalıştırılırlar.

- **Filtreler:** Çipi istenildiği zaman, UV (Ultra Viole: mor ötesi) ışığının altında öldürebilmek için, bazı yazılımlar EEPROM'da konuşlandırılır. (Güvenlik verileriyle beraber çalışan kodlar yok olur)

- **Cevap zamanı toplama:** Bazı durumlarda kötü niyetli terminalleri tespit etmek için cevap zamanı hesaplanır.

- **Hafıza matris güvenliği:** RAM, ROM veya EEPROM hafızalarındaki yazılımların hangisinin, hangilerini, kontrol edeceği, sınırlandıracağı belirtilir.

- **Isı detektörleri:** Düşük ısı (RAM şarjının azalma oranının tespiti) ve yüksek ısı detektörleri.

- **Voltaj detektörleri:** Genel olarak rasgele sayı üretici korunur. Düşük voltaj detektörü EEPROM hafızayı yazılmaktan, yüksek voltaj detektörü de silinmekten korunur.

- **Saat detektörleri:** Hızlı saat detektörleri sinyali sıkıştırmaktan (veri kaybından) korur, yavaş saat detektörleri de adım adım hata ayıklamayı engeller.

Bütün bu güvenlik önlemlerine rağmen bölüm 2.1.5.'de bahsedildiği gibi akıllı kartların güvenlik açıkları vardır. Bu açıklar özellikle off-line ödemelerde günyüzüne çıkar.

2.10. Başarım Hataları

Kesik bir işlem, terminal ya da kartta olabilecek donanım veya yazılım başarısızlıklarını da içeren, değişik sebeplerden dolayı başarısız olan işlem demektir. Kesik işlem, terminalin kart üzerine son bilgilerini yazıp dosyaları kapamasından önce kartı terminalden çeken kart sahibi tarafından da oluşabilir.

Örneğin bir kamu telefon uygulamasında, telefon servisi için iki hesap çıkartma metodu vardır. Sistem telefon konuşması yapıldıktan sonra konuşma ücretininin karttan çekilmesine göre dizayn edilebilir. Eğer parçalanma durumu tesbit edilirse, bir sonraki artış ya da değer müteakip bir işlem için kart yerleştirildiğinde

çekilir ve işlem, örneğin, kart hesabını göstermeden ve ek telefon görüşmelerine izin vermeden önce tamamlanmaya zorlanabilir.

Telefon hesabını ödeme örneğinde, ikinci yöntem değeri geri çekmektir. Bazı kartlı telefon sistemleri telefon görüşmesini yapmadan önce ücreti karttan düşerler. Bu senaryoda, kart telefon görüşmesi veya zaman birimi için, önceden ödeme yapar. Örneğin telefon hiç cevaplanmamışsa veya kart sahibi bitiş zamanından önce kartı telefondan çekerse, değer karta tekrar geri yüklenir . Recovery (tamir etme) fonksiyonu, kart tekrar kullandığında değer karta geri döndürülmesini garantiler.

Donanım istisnaları, saat frekansının bozulmasından düşük voltaj durumlarının tespitine kadar çeşitlidir. Düşük voltaj tespiti, özellikle pille çalışan terminallerin çalışmasında önemlidir. Test tam bir işlemi garantilemelidir. Eğer tam işlem mümkün değilse başlatılmamalıdır.

Düşük voltaj tespiti, yarı iletken endüstrisi tarafından, tüm yeni çip kartı tasarımları için bir standart olarak sunulmaktadır.

Tasarım süreci, haberleşmenin tamamlanamaması gibi durumlarda, terminal ve kartın statüsünü etkileyecek hatadan geri dönme işlemi içermelidir. Bu da daha önce bahsedilen fonksiyonlar veya verileri birleştirip tamamlama şeklinde olabilir.

2.11. Bilgi Aktarımı

Kart okuyucudan karta veya karttan kart okuyucuya bilgi aktarımı, tasarım aşamasında düşünülmesi gereken ayrı bir olaydır. Genel olarak, hataları ana terminale gönderme sistemin düzgün çalışması için önemlidir. Veri alışverişini taklit etmeye çalışan bilgisayar korsanlarını tespit etmenin yanında, istisna durumlarını analiz etme, ekstra kart ve terminallerin sisteme eklendiğinde ortaya çıkan problemleri belirler. Sistem operatörü, veri kaybını önlemek ve ana veritabanının eş zamanlı işlemlerini bozmamak için istisna durumlarını tanımlamalıdır.

Veri tasarımı ve düzeni, sistem bütünlüğünü kontrol eden, hatadan koruma ve hata düzeltme fonksiyonları içermelidir. Bu, tasarımın karşılıklı iki yetkilendirme olayını gerçekleştirme kapasitesine sahip olması demektir. Böylelikle sistem ve sistemin işletim elemanlarının, tasarlandığı şekilde çalıştığı doğrulanır. Kart ve terminalleri içeren fiziksel hatalarda ve tamamlanamamış veri alışverişi olaylarında alışveriş tekrar sağlanmalıdır.

Bu çalışmada on-line sistem üzerinde durulacağı için, merkez sunucuyla haberleşme bitmeden işlem tamamen gerçekleşmiş olmayacaktır. Mesela ilk önce kartın doğrulanması gerçekleşecek, daha sonra hesap değiştirme isteği sunulacak, istek gerçekleşirse karta yeni hesap yazılacaktır. Karta yazma bittikten sonra merkeze 'doğru yazıldı' haberi gönderilirse ancak işlem tamamen bitmiş olacaktır. Bütün bu işlemler özel komut setiyle gerçekleştirilir, hata anında ise geriye dönüş yapılır.

2.12. Terminaller

Elektronik cüzdan uygulamalarında, ek olarak güvenlik modülü içeren, ilgili marketlerde rahatça bulunabilecek standart terminaller kullanılır. Alım satım işlemleri genel olarak off-line olarak yapılır. Fakat yine de bilgisayara ulaşma ve ödemelerde on-line sisteme geçilebilir. Tekrar yükleme işlemleri nakit paradan elektronik cüzdana çevirme dışında on-line olarak yapılır.

Kartla terminaller arasındaki muameleler periyodik bir şekilde ana bilgisayarlara aktarılacak için terminallerde saklanır. Her bir terminal, daha önce belirlenen gece vakitlerinde ana bilgisayara işlemleri aktarır. Mesela hafızanın dolduğu ve başka işlem yapamayacağı gibi durumlarda bu aktarım işi başka zamanlara kaydırılabilir.

Terminaldeki veriler toplandığında, ana bilgisayar alacak ve verecekleri kartın hesabına geçer ve satıcıya (mağaza sahibine) ücretleri düşerek elektronik ödemeyi yapar. Terminal sahiplerinin eline, aylık veya üç aylık periyotlar halinde, numaralar

ve hesaplardan oluşan raporlar geçer. Böylelikle el ile veya elektronik olarak hesapları kontrol ederler.

Sisteme kaçak girme, illegal kartlar, çalınmış kart veya terminaller gibi herhangi bir istisnai durumlar ertesi sabah rapor edilir. On-line ortamlarda ise bu kaçaklar, ani bir şekilde güvenlikten sorumlu şahıslara ulaştırılır.

Herhangi bir akıllı kart uygulaması çok çeşitli kart okuyucularına ihtiyaç duyar. Bakiye kontrol ediciler, elektronik cüzdan veya kartla çalışan yazıcılar bunlardan bazılarıdır. Bu terminaller kartın içindeki alışveriş verilerini kart taşıyıcısının görmesine izin verirler.

Selfservis makineleri ikinci bir terminal ailesi tipi oluştururlar. Selfservis terminalleri, ödemeli telefonlar, otoparklar, çukolata-bisküvi, çay-kahve makineleri, fotokopiler, bilgisayar servisleri gibi çok çeşitli uygulamaları desteklerler.

Perakende satış yapan yerlerdeki terminallerin de farklı kullanımları vardır. Tezgahtar kartı okutur kart sahibi de onayını verir.

Bütün bu terminal tiplerinin dizaynı, yapacağı fonksiyona dayalı olarak değişir. Mesela bilanço okuyan terminallerin tuşlara ihtiyacı yoktur, kart basitçe okuyucuya takılır ve bakiye ortaya çıkar.

2.13. Ödeme Planı

Elektronik cüzdan uygulaması kendine ait vazifeleri olan birçok iştirakçiyi içerir. Bazen ticari banka gibi tek bir kuruluş bazı iştirakçilerin görevini yerine getirir. Beş ana grupta toplanırsa:

- a) **Cüzdan tedarikçisi:** Cüzdan uygulaması boyunca, cüzdanı tedarik eden ve içindeki değeri garantileyen organizasyondur. İşletim kurallarını belirler ve genel

güvenlikten sorumludur. Bu, bir grup bankayı temsil eden bir organizasyon da olabilir.

- b) **Yükleme merkezi:** Cüzdan tedarikçisinin, elektronik cüzdan kullanıcısındaki(taşıyıcısı) cüzdana parayı yükleyen güvenilir bir şubesidir. Gerekğinde tekrar yükleme yapar. Bu birim cüzdan tedarikçisinin kendisi veya cüzdan dağıtan bir banka da olabilir. Aynı zamanda kartı kişiselleştirme ve ilk değerleri yükleme işi de yaparlar.
- c) **Hizmet/Eşya sunucular:** Eşyayı satan veya hizmet sunan firma veya mağazalardır. İşlemler için terminalleri kullanırlar.
- d) **Kazanç tedarikçileri:** Servis sunucuların terminallerinden işlem ödemelerini toplayan cüzdan tedarikçisinin güvenilir şubeleridir. Genel olarak ödemeleri kredi gibi diğer kazançlarla servis sunucunun hesabına değiş tokuş yaparlar.
- e) **Cüzdan taşıyıcılar:** E-cüzdan olarak kullanılan akıllı kartları taşıyıp onunla alışveriş yapan kişilerdir.

2.14. İşlemlerin Genel Yapısı

Birçok fonksiyonla birlikte bir defada gerçekleşen işlemlerdir:

- **İşlem hesabının görülmesi:** Servis sunucuların terminallerinden alınan işlemlerin kredi ve borçlanma dönüşüm işlemlerinin yapılmasıdır.
- **Hesapların gözden geçirilmesi ve sonuçların rapor edilmesi:** Cüzdan tedarikçisinin yapılan bütün işlemleri, aktarılan paraları gözden geçirmesidir. Böylelikle güvenlik ihlali gibi anormal durumlar tespit edilir ve diğer birimlerin problemleri giderilir.
- **Kara liste yönetimi:** Cüzdan tedarikçisine çalınan, kaybolan veya kopyalanan cüzdanların iptaline imkan verir.
- **Yükleme yönetimi fonksiyonları:** Cüzdan tedarikçisinin yükleme merkezlerinin kapasiteleri gözönüne alınarak güvenli bir şekilde cüzdanı yükleme imkanı verir.

2.15. Genel Bir Elektronik Cüzdan Uygulamasında İşlemler

MPCOS-EMV (Multi-application Payment Chip Operating System: birden fazla uygulamalı ödeme çipi işletim sistemi) çok amaçlı kullanım ve ödeme için düzenlenmiştir. Bazı özellik ve standartlarını sıralarsak:

- **ISO 7816 (1-2-3) uyumluluğu:** Genel olarak $T = 0$ protokolünü kullanır fakat $T = 1$ ve 14 protokollerini de destekler.
- **ISO 7816 (4) uyumluluğu:** Komutlar, veri yapıları ve sonuç kodları yazılımcı ve terminal üreticileri tarafından kabul görür.
- **EMV (Europay, Master, Visa) uyumluluğu**
- 115200 baud'a kadar çıkabilen giriş, çıkış fonksiyonları, hızlı kriptografik rutinler.
- İspatlanmış güvenlik özellikleri:
 - Gizli kod / anahtar yönetimi ve doğrulanması
 - Duyarlı sistem veri koruması
 - Yetkilendirme için 3DES algoritması, güvenli mesaj ve ödeme sertifikaları
 - Difransiyel güç analizine karşı uygulama düzeyinde karşı tedbirler
- Kişiselleştirme için gelişmiş yönetim komutları

Off-line satın alma işlemi dört kademedé incelenebilir.

2.15.1. Karşılıklı yetkilendirme (kart ve terminalin birbirini tanınması)

Sahte kartları, taklit güvenlik erişim modüllerini yakalayan kriptografik protokoller, gizli anahtar kriptosuna dayanan sorgu-cevap protokolleridir. 16 Byte

anahtarla şifre-deşifre-şifre modunda 3DES algoritmasıyla gerçekleştirilir. Ana anahtarlar güvenlik erişim modülünde saklanır.

Terminal kartın gerçek olup olmadığını şöyle kontrol eder:

- Kartın seri numarası okunup, SAM tarafından ana anahtardan (MK_{auth}) kart anahtarı (K_{auth}) elde etmek için kullanılır. Kart seri numarası daha sonraki işlemler için terminalde saklanır.

- SAM kart tarafından şifrelenmesi için rasgele sayı üretir.
- Kart (K_{auth})'ı kullanarak rasgele sayıyı şifreler
- Kartın ürettiği kriptogram SAM'ın ürettiğiyle karşılaştırılmak üzere terminale geri gönderilir.

Yetkilendirme başarısız olarak sonuçlanırsa işlem durdurulur ve kart geri iade edilir. Kart aynı işlemleri, sahte olup olmadığını tespit için terminale karşı da uygular.

2.15.2. Cüzdan hesabını okuma

Satın alma işleminde cüzdandaki değer okunur ve kart CBC (Card Balance Certificate: Kart hesabı sertifikası) ile değişik bir anahtar (K_{bal} : hesap anahtarı) kullanarak hesaplamayı yapar. Daha sonra CBC, SAM tarafından doğrulanır.

- SAM rasgele bir sayı üretir, bu sayı cüzdan ve anahtar seç komutuyla karta gönderilir.
- Kart işlem sayacını artırır, yeni geçici işlem anahtarı oluşturur ve yetkilendirme kriptogramını hesap eder.
- Daha sonra bu kriptogram SAM tarafından kontrol edilmek üzere terminale gönderilir.
- Sonuç doğrulanırsa, hesabı oku komutu ve terminal işlem sayaç değeri karta gönderilir.

- Kart sertifikayı (CBC) hesap eder ve sonucu hesap değeri ile beraber terminale gönderir.
- SAM kartın CBC'sini kendi hesapladığıyla kontrol eder.

2.15.3. Borçlandırma işlemi

Kart K_{pay} (ödeme anahtarı) kullanarak CDC (Card Debit Certificate: kart borçlandırma sertifikası)'yi hesap eder. Bu sertifika SAM tarafından kontrol edilmelidir. K_{pay} ana ödeme anahtarından (MK_{pay}) ve CSN (Card Serial Number: Kart seri numarası)'den hesaplanır.

- SAM rasgele sayı üretir ve SelPk komutuyla beraber karta gönderir.
- Kart CTC (Card Transaction Count: Kart işlem sayısı)'sini artırır ve geçici ödeme işlemi anahtarını üretir. Yetkilendirme kriptogramını hesaplar.
- Kriptogram SAM'ın ürettiğiyle karşılaştırılmak üzere terminale gönderilir.
- Sonuç doğrulanırsa borçlandırma komutu terminal işlem değeri ile beraber karta gönderilir.
- Kart borç kriptogramını hesaplayarak kontrol edilmek üzere terminale gönderilir. Terminal bunun üzerine yeni cüzdan hesap değerini gösterir.

2.15.4. Borç işlemi imzası

Borçlandırma imzası, sahte kartları tespit etmek ve terminal tarafından yanlış işlem yapılmadığını kontrol etmek için gereklidir. Bu imza sadece tedarikçiler tarafından kontrol edilir.

İmza e-cüzdan tarafından işlem elemanları ve imza anahtarı ile hesaplanır. İmza sertifikası, ilgili CTC, işlem, hesap değerleri ve CSN numarasıyla ana (host) makineye gönderilmelidir.

3. KAMPÜS ORTAMINDA ELEKTRONİK CÜZDAN UYGULAMASI

Bu projenin amacı, üniversite içerisindeki harcamalarda nakit paranın yerine elektronik cüzdanı kullanmaktır. Uygulamada üniversite elemanları (akademik personel, öğrenciler, memur ve diğer çalışanlar) kafeteryada veya yemekhanede yiyecek ödemelerinde, kütüphanedeki fotokopi çekimlerinde gerekli donanım çalışması yapılırsa kola veya bisküvi makinelerinde nakit para yerine, ön ödemesi yapılmış hesap bilgisi içeren akıllı kartı kullanacaklardır.

Akıllı kartlar tam bir güvenlik sağlamadığı, yüksek güvenli mikroişlemcili akıllı kartların fiyatının da diğerlerine göre çok yüksek olduğu için, sistemde düşük fiyatlı, hafıza korumalı akıllı kartlar kullanılmıştır. Böylelikle karta güvenmek yerine, kart kapasitesi kullanılarak uygulanan şifreleme protokolünün güvenliği üzerinde durulmuştur.

Kartlar tekrar doldurulabilir olacaktır. Kartın içindeki tutar, kullanım sonucu tükenirse, üniversite içinde belirli noktalardan ödemesi yapılarak tekrar yüklenebilir. Yükleme işi ya nakit parayla ya da kredi kartı ile yapılabilir. Kullanıcı mezun olduğunda veya enstitüden ayrıldığında karttaki elektronik para nakit paraya çevrilip kullanıcıya geri ödenir.

Bütün bir sistemi kontrol altına almak, ayrıca güvenlik açıklarına sebep vermemek için off-line ödeme hesaba katılmadı. Önerilen sistem manyetik kredi kartı on-line yetkilendirme ve ödeme altyapısına dayanıyor. Bu altyapı düşük maliyetli ve güvenilirdir çünkü bütün dünyada geçerlidir. Sadece uygun kart okuyucuları veya POS makineli terminaller eklenecektir.

Kullanılan yetkilendirme protokolüne göre, sayısal sertifika almak için sertifika otoritelerine veya üçüncül kurumlara ihtiyaç yoktur. Bu sistemi PKI sorunlarıyla karşılaşmayacak şekilde basit hale getirir.

3.1. Sistem Tasarımı

Proje, ana başlılarıyla şu özelliklerle tasarlandı:

- a) Sistem tamamen manyetik kart altyapısına uygun on-line altyapıda olacaktır.
- b) Bazı veriler yetkilendirme ve doğrulama için hem kartın üzerinde kalacak, hem de sunucu makinesine gönderilecektir.
- c) Sadece elektronik cüzdan için düşünülmesine rağmen, taşıdığı kimlik bilgileriyle üniversite içinde kimlik tanımlama kartı olarak da kullanılabilir.
- d) Genel elektronik cüzdan sistemlerinde kullanılan açık anahtarlı altyapılar yerine daha basit ve değiştirilmesine gerek duyulmayacak bir güvenlik kullanılacaktır.
- e) Güvenlik altyapısında, haberleşme ve veritabanındaki bilgilerin güvenliğine dikkat edilecektir.

3.2. Sistemin Genel Yapısı

Bu çalışmadaki görev dağılımı göz önünde bulundurularak, sistem yapısı hakkında bilgi sahibi olunabilir:

- **Cüzdan tedarikçisi:** Okul yönetimi. Sisteme onay verip; kartları temin eden, gerekli altyapı için finansman ve eleman sağlayan birim.
- **Yükleme merkezi:** Okulun belirleyeceği kartı dağıtan, kart yükleyen, silen, parayı toplayıp geri veren, hizmet / eşya sunuculara müşteri hesabından parayı gönderen bir merkez. Aynı zamanda kazanç tedarikçisinin de görevini yapar.
- **Hizmet/Eşya sunucular:** Kafeterya, yemekhane, kütüphane, bilgisayar laboratuvarları, öğrenci işleri. Özel donanım eklenirse çukolata, kola ve bisküvi makineleri, kontürlü telefonlar aynı vazifeyi yapabilirler.
- **Cüzdan taşıyıcılar:** Akademik personel, öğrenciler, memurlar, işçiler kısaca üniversite alanı içinde uzun süreli bulunan herkes.

Merkez sunucu, para ve diğer bilgileri karta yükleyen, alışveriş yapılan birimler üniversite alanı içinde bulunuyor. Kartlar ve elektronik cüzdan üniversite dışında işlemeyecek. Böylelikle mikro plandaki, merkezi ve kapalı sistemlere örnek teşkil ediyor. Sistem zaten altyapıda mevcut olan telefon hatlarına ek yapılarak on-line bir platforma oturacak. On-line platforma oturmasının sebebi de daha sonra açıklanacağı gibi, yetkilendirme için gerekli olmasıdır. Şekil 3.1’de sistemin genel yapısı gösterilmiştir.

Bölüm 2.3.’deki beklenen dezavantajlar bu sistemde kesinlikle ön plana çıkmaz. Üniversitede harcanan para bir insanın toplam harcayacağı paranın çok az bir kısmını oluşturur. Yani karta yüklenecek tutar enflasyondan ve paranın değer kaybetmesinden fazla etkilenmez. Ayrıca okul yönetiminin iflas etmesi beklenemez. En azından devlet güvencesi altındadır. Sistem on-line çalıştığı, hesap bilgisi kartla birlikte merkez sunucu bilgisayarda tutulduğu için kartın kaybolması ve çalınması durumunda, kart kullanıcısının parası geri iade edilebilir veya kullanıcıya yeni bir kart verilebilir.

3.3. Sistemin Genel İşleyişi

Kart kullanıcısı (üniversite elemanı), kart yükleyicisine giderek kimlik, hesap ve şifre bilgilerinin yükleneceği bir kart ister. Kart yükleyicisi merkez sunucunun onayı ile hesap miktarını kart kullanıcısından alarak bilgileri karta yükler. Kart böylelikle kullanıcıya verilir. Kart kullanıcısı okuldaki uzun süreli ayrıldığında, yatırdığı parayı geri almak istediğinde, kartını kaybettiğinde veya çaldırdığında, karta tekrar yükleme yapmak istediğinde yine kart yükleyicisine başvurur.

Kart kullanıcısı üniversite içersinde alışveriş yapmak istediğinde hizmet veya eşya veren birime kartını verir. O birim kimlik bilgilerini kontrol ettikten sonra karttaki bilgileri on-line olarak merkez sunucuya gönderir. Merkez sunucudan yetkilendirme aldıktan sonra, hesap düşme veya işlemi iptal bilgisini tekrar merkez

sunucuya gönderir. Merkez sunucudan onay gelirse yeni hesabı karta yazar ve alışverişi gerçekleştirir.

3.4. Sistemde Kullanılan Donanım ve Kartlar

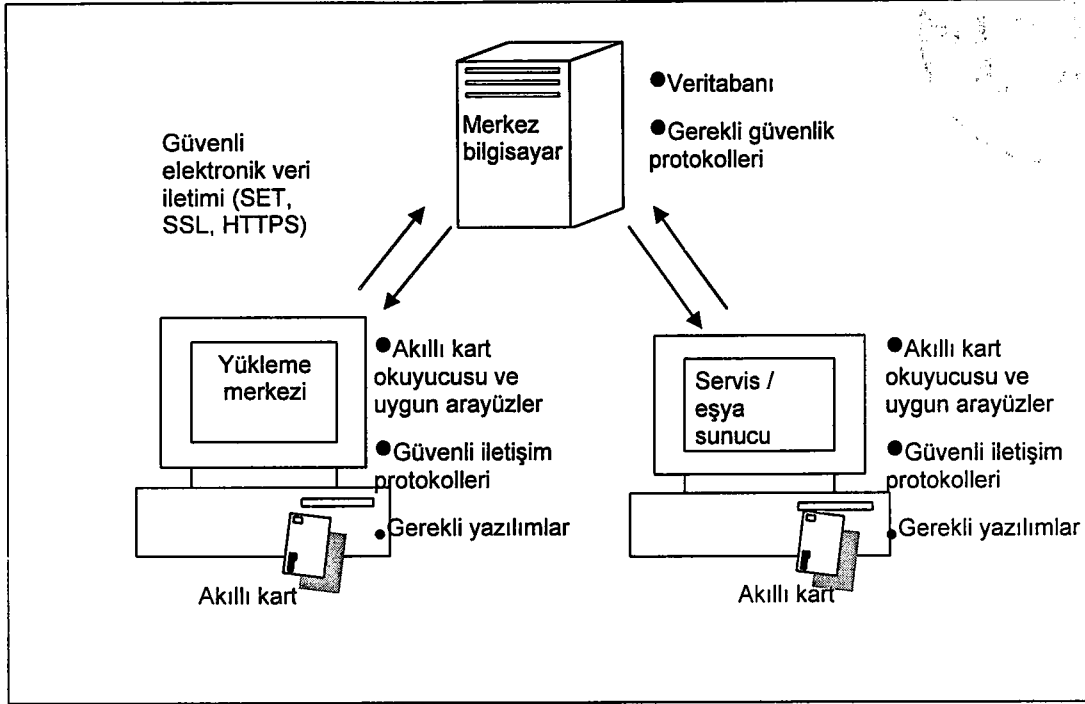
Sistemde merkez sunucu yüksek işlem gücüne, kırılgan olmayan bir işletim sistemine sahip bilgisayardan oluşur. Sonuçta verileri saklayacak bir veri tabanına, yetkisiz girilmesine izin verilmeyecek bir yönetime ihtiyaç vardır. Aynı zamanda on-line olarak diğer bilgisayarlarla haberleşecek bir donanıma sahip olmalıdır. Bu çalışmada merkez sunucunun işletim sistemi Microsoft Windows 2000 Server, veritabanı ise Microsoft SQL Server 2000'dir (Şekil 3.1).

Kart yükleyicide yine güvenli bir işletim sistemine, merkez sunucuyla haberleşecek yeterli donanıma sahip bir bilgisayar, ve bir kart okuyucusu bulunmaktadır. Kartla ulaşacak gerekli kütüphane ve yazılımla bu sistem bir POS terminalini andırır. Kartı yükleme ve yetkilendirme işlemi sadece bu ve bununla aynı özellik ve görevlere sahip terminallerde olur. Kart yükleyici bilgisayarları merkez sunucuda kayıtlıdır. Dışarıdan kaçak olarak sisteme giren terminaller merkez sunucu tarafından tanınmayacağı ve yetki verilmeyeceği için sahtekarlıklar önlenir.

Alışverişin yapıldığı yerlerdeki düzenek de aynı kart yükleyicideki gibidir. Buralarda sadece kart yükleme işlemi yapılamaz.

Kart okuyucusu olarak "TOWITOKO" firmasının "ChipDrive micro 120" markası seçildi. Gerekli kütüphaneleri bulunursa pazarda satılan herhangi bir kart okuyucusu da aynı işlevi görebilir.

Sistemde "Siemens SLE 4418/4428 veya 4432/4442 Hafıza Kartları" kullanılacaktır. Bu kartların mikroişlemcili veya güvenli kartlara göre bazı üstünlükleri vardır. Mesela 1Kbyte'lık 4428 kartını ele aldığımızda:



Sekil 3.1: Üniversite alanındaki elektronik cüzdan genel yapısı

- Ürün mikroişlemcili kartlara göre çok daha ucuzdur (1/10 seviyesinde).
- PIN içeren ilk 3 Byte'ı yazma ve okumaya karşı korumalıdır.
- Sonraki 1021 Byte'lık hafıza kısmının ilk 32 Byte'ı yazmaya karşı korumalıdır.
- Geriye kalan 989 Byte'a, ancak PIN doğrulaması sonucunda yazılabilir.
- Gerekli şifreleme ve yetkilendirme protokolleri eklendiğinde MKÜ kartlarını aratmayacaktır.

3.5. Güvenlik

Güvenliği beş ana başlık altında toplayabiliriz:

- Bilgisayar ve yazılımların güvenliği:** Bir önceki bölümde bahsedildiği gibi yetkisiz hiç kimse bilgisayarları izinsiz kullanamamalıdır. Bu güvenli bir işletim sisteminde kullanıcı ve şifresini tanımlamakla olur. Ayrıca kullanıcıların hatta yöneticilerin gerçekleştirdiği işlemler veritabanında tutulmalıdır. Daha da ötesi, akıllı kartlarla, parmak izi gibi biyometrik kontrollerle işletim sistemi ortamına giriş sağlanabilir. Aynı bir çalışmayı

veya yazılım-donanım satın almayı gerektiren bu durumda operatör kartını bilgisayardan çektiğinde, program ve işletim sisteminden çıkarılır. Kullanıcı yetkisi olmayan hiç kimse böylelikle ortama giriş sağlayamaz. Yazılımlar .exe uzantılı kodlar olduğu için kaynağına ulaşamaz.

2. **Veri iletim güvenliği:** Halihazırda mevcut olan telefon altyapısı kullanılacağı için, bankaların İnternet üzerinden alışverişlerde sunduğu 128bit SSL kullanılmalıdır. Bu sistem de satın alınacaktır.
3. **Kart Güvenliği:** Daha önce de bahsedildiği gibi PIN kontrolü olmadan karta yazma engellenmiştir.
4. **Bilgi Güvenliği ve Yetkilendirme:** İlk üç güvenlik sağlandıktan sonra en çok üzerinde durulması gereken konu budur. Zira kart gerekli donanım (kart okuyucusu) ve yazılım (uygulama arayüzü) satın alınarak sistem dışında okunup, PIN doğrulaması yapılıncaya da karta yazılabilir. En nihayetinde PIN dört veya altı karakterli bir şifredir. Bir sonraki bölümde bilgi güvenliği ve yetkilendirme için gerekli olan protokol ve plan sunulacaktır.

3.6. Bilgi Güvenliği ve Yetkilendirme Protokolü

Manyetik kart tabanlı ödeme sistemlerinde bölüm 3.2’de anlatıldığı gibi, kart kullanıcısı sisteme girmek için kimlik ve şifresini merkeze göndermek zorundadır. Merkezdeki bilgilerle gönderilenler birbirini tutarsa kullanıcı sisteme girebilir. Şifre ve kimlik bilgilerinin güvenli ve hatta kriptolu bir ortamda iletilmesine rağmen tamamen güvenilir değildir. Veri iletimi arada kesilip taklit edilebilir, böylelikle sisteme tekrar girilebilir (tekrar para çekilebilir, harcama yapılabilir). Günümüzdeki taklit ve sahtekarlık olaylarının çokluğu bunu kanıtlar.

Bu problemi çözmek için çeşitli Lombart 1981’de yayınlanan makalesinde bazı yöntemler öne sürülmüştür[4]. Bu çözümün tek eksikliği merkezde doğrulama tablolarının tutulmasıdır. Bu tablolara ulaşırsa sistem güvenliği sona erer. [5], [6], [7]’de sözü geçen araştırmacılar bu problemin de üstesinden gelmiştir.

Elektronik cüzdan uygulamalarında yaygın olan güvenlik bölüm 3.8’de bahsedildiği gibi ‘Açık Anahtar Altyapılı’ sistemdir. Fakat bunun için pahalı bir altyapı, dijital sertifika dağıtan kurumlara ihtiyaç vardır.

Bu projede basit, kolay ve ucuz bir yöntem denenecektir[3]. Makalede sözü geçen yetkilendirme protokolüne göre:

- a) Kullanıcı şifresi ve doğrulama tabloları merkez sunucuda yer almaz,
- b) Tekrar eden işlem isteği (veya kandırması) tamamen çözülmüştür,
- c) Herhangi bir kullanıcı şifresini istediği gibi seçip,değiştirebilir,
- d) Şifre merkez sunucuya gönderilmez. Böylelikle şifreyi yalnız kullanıcı bilir, merkez bilemez (Zero-knowledge proof [8]).
- e) Karşılıklı yetkilendirme ile merkez de yetkilendirilir [4].

Bu karakterlerin ışığında akıllı kartla yetkilendirme işlemi üç bölümden oluşur:

Yükleme: Yüklemiş işlemi elektronik cüzdan görevlileri açısından baktığımızda, yüklemiş merkezinde gerçekleşir. Akıllı kart kullanıcısı K_i , kayıt olmak istediğinde kimlik bilgileri yüklemiş bilgisayarına girilir. Sonra kendisinden bir şifre seçmesi istenilir $\mathcal{S}_i$, seçilen şifre tek yönlü(geri dönüşü olmayan) fonksiyondan $f(\mathcal{S}_i)$ geçirilir. Bu fonksiyon, standartlarda yer alan 128bit çıkışlı MD5 veya 160bit çıkışlı SHA1 fonksiyonu olabilir. Literatürde Hash adıyla geçen fonksiyonların çıkışından girişi elde etmek mümkün değildir. Daha sonra kullanıcı kimlik bilgileri, K_i ve $f(\mathcal{S}_i)$ merkeze gönderilir. Merkez bu bilgilerle aşağıdaki fonksiyonu kullanarak değiştirilmiş şifreyi ($D\mathcal{S}_i$) hesaplar:

$$D\mathcal{S}_i = f(K_i \oplus H_i \oplus A_g) \oplus f(\mathcal{S}_i),$$

Burada A_g merkezin gizli şifresini, H_i ise ilk hesap bilgisini temsil ediyor. Yüklemiş merkezi $D\mathcal{S}_i$ ’yi merkezden alarak, kimlik K_i , hesap bilgisi ile birlikte tekrar 3DES algoritmasıyla şifreleyerek karta yazar. Kartın PIN’i de $\mathcal{S}_i$ ’den elde edilir.

Banka			
100% İşlem tamamlandı!			100%
         			
Müşteri Bilgileri:			
Kart No:	1	Son Kullanma Tarihi:	01/01/2003
Hesap:	999.999.999	TL	
Ad:	Mehmet Ercan	Doğum Tarihi:	04/10/1973
Soyad:	Kuruoğlu	Doğum Yeri:	Burhaniye
Nüfus No:	A02-3414513		
Baba Adı:	Niyazi		
Ev Adresi:	Osman Yılmaz mah.		Ev Tel:
İş Adresi:		Tübitak UEKAE, P.K. 21	İş Tel:
E-mekrup:	mekuruoglu@hotmail.com		Cep Tel:
-> Boş kart, yeni müşteri girerek yükleme yapın! -> ClientSocketToServer: Connecting -> ClientSocketToServer: Connect -> ClientSocketToServer: Write			
			Kapat
Yeni kart yükleme ve müşteri girme işlemi			

Sekil 3.2: Yükleyici bilgisayarındaki program

Sisteme Giriş ve Tanınma: Alışveriş sırasında karttan hesap düşülürken veya iptalinde hesap geri yüklenirken; yükleme merkezinde nakit para elektronik cüzdana yüklenirken veya kart iptali gibi durumlarda elektronik para nakit paraya çevrilirken karşılaşılabacak durumlardır.

Kart terminale takıldığında terminaldeki yazılım, okuyucu vasıtasıyla kartın içindeki bilgileri alır ve deşifre eder. Böylelikle K_i , kimlik ve hesap bilgilerini, aynı zamanda $D\mathcal{S}_i$ bilgisini elde eder. Kullanıcın şifresini ($\mathcal{S}_i$) girmesi üzerine terminal aşağıdaki hesapları yapar:

$$TD\mathcal{S}_i = D\mathcal{S}_i \oplus f(\mathcal{S}_i) = f(K_i \oplus H_i \oplus A_g),$$

$$Y = f(TD\mathcal{S}_i \oplus TZ)$$

$TD\mathcal{S}_i$: Tekrar Değişen Şifre

TZ : Terminal işlem Zamanı

Y : Yükleme ifade eder

Daha sonra (K_i , Y , TZ) ve hesap bilgisi merkeze gönderilir. Doğrulanma safhasında merkezde şu işlemler yapılır:

1. K_i ve hesap bilgisi kontrol edilir. Yanlışlık veya uyumsuzluk varsa kullanıcı ve kart reddedilir.
2. Merkeze ulaşan bilginin zamanı (MZ) ile TZ arasındaki fark belli bir değeri aşmamalıdır. Eğer: $MZ - TZ \geq \Delta T$ ise tekrar veya sahte işlem yapma durumu olabileceği için kullanıcı ve kart reddedilir.
3. Alttaki eşitlik sağlanırsa kullanıcı ve kart yetkilendirilir:

$$Y = f(f(K_i \oplus H_i \oplus A_g) \oplus TZ)$$

Karşılıklı yetkilendirme için merkez kendi işlem zamanını (MZ^m) kullanarak, kendi fonksiyonunu (MY^m) elde eder:

$$MY^m = f(f(K_i \oplus H_i \oplus A_g) \oplus MZ^m)$$

Sonraki aşamada MZ^m ve MY^m 'yi terminale gönderir. Terminal kendi zamanı ile MZ^m arasındaki farkı kontrol eder. Eğer bu fark ΔT 'yi aşıyorsa merkez reddedilir. Son olarak merkezi yetkilendirmek için MY^m alttaki eşitliği sağlamalıdır:

$$MY^m = f(TD\$_i \oplus MZ^m)$$

Bundan sonra harcama veya para yükleme işlemleri gerçekleştirilebilir. Doğrulama sağlanmadığı durumlarda kara liste ekleme ve haber verme işlemleri yapılmalıdır.

Şifre Değiştirme: Kullanıcı doğrulanması gerçekleştikten sonra eğer şifre değiştirmek isterse aşağıdaki hesaplar yapılmalıdır:

1. $D\$_i \oplus f(\$_i) \{ = f(K_i \oplus H_i \oplus A_g) \}$ hesaplanır,
2. Yeni şifre $Y\$_i$ seçilir ve $f(Y\$_i)$ hesaplanır,
3. $DY\$_i = f(K_i \oplus H_i \oplus A_g) \oplus f(Y\$_i) \{ = D\$_i \oplus f(\$_i) \oplus f(Y\$_i) \}$ hesaplanarak $D\$_i$ yerine karta yazılır.

4. Güvenlik Analizi

Merkez-Kart yükleyici veya Merkez-Alışveriş yeri arasındaki haberleşmeleri, İnternet bankacılığında olduğu gibi en başta güvenli olarak kabul ediyoruz. Bu yüzden veri iletiminde hata ve kaçak olmadığı düşünülmelidir.

Merkezin gizli anahtarı A_g , tek yönlü Hash fonksiyonu tarafından korunduğu için, herhangi birinin ona ulaşması neredeyse imkansızdır. Basit hafıza kartının çalınma durumunda, hırsız üçlü DES şifrelemesini kırsa bile Hash fonksiyonundan dolayı merkezin gizli anahtarına erişemez. Üçlü DES şifresinin güvenlik analizleri yapıldığında, çok güçlü ve paralel çalışan birçok sayıda bilgisayarla bile kısa sürede kırılmaz[2]. Güvenliği arttırmak için mikro işlemcili kartlar düşünülebilir.

TZ, tekrar veya (mesajda oynama yaparak) uzun süre sonra işleme girme riskini ortadan kaldırır. Doğrulama safhasını geçmek için kaçak, TZ'yi değiştirmek zorundadır. TZ değişince de Y bulunamaz.

Kart çalınsa bile hırsız şifreyi bilmediği için $TD\hat{S}_i$ 'yi hesaplayamaz. Böylelikle doğrulama safhasını geçemez. Hırsız $D\hat{S}_i$ 'yi karttan okusa bile yek yönlü fonksiyonla korunan A_g ve $\hat{S}_i$ 'yi ele geçiremez.

Peki hırsız kullanıcının kendisi olursa, mesela hesap tutarını illegal olarak artırmak isterse ne olur? $\hat{S}_i$ 'yi zaten biliyordur. Standart uygulanan tek yönlü fonksiyonları tespit ederek $f(\hat{S}_i)$ 'yi yaklaşık olarak bulabilir veya tahmin edebilir. Belki çok uzun uğraşlar sonucunda, üçlü DES algoritmasını kırarak $D\hat{S}_i$ 'ye ulaşabilir. Fakat en sonunda gelip takılacağı yer, başta bahsettiğimiz merkez-kart yükleyici ve merkez-alışveriş yeri arasındaki güvenli haberleşme protokolüdür. Çünkü bilgisayarlar arasında da doğrulama protokolleri vardır.



5. SONUÇ

Bu çalışmada hafıza korumalı bir akıllı kartla, basit bir kullanıcı doğrulama algoritması kullanarak elektronik cüzdan uygulamasını gerçekleştirdik. Algoritmanın basitliği uzun işlemler içermemesinden, kısa zamanda gerçekleşmesinden kaynaklanıyor.

Daha fazla güvenlik için hafıza korumalı kartın yerine mikroişlemcili akıllı kartlar düşünülebilir. Böylelikle çok uygulamalı sistemlere bir geçiş sağlanır. Fakat böyle bir kartın maliyeti hafıza korumalılarına göre 10 katına çıkar.

Global elektronik cüzdan sistemlerinde olduğu gibi, Açık Anahtar Altyapılı sistemler hem çok pahalıya patlayacak, hem de uzun bir çalışma gerektirecektir. Üniversite ortamında hayata geçirilecek olan bu sistem çok ucuza mal edilmesinin yanında elemanlarına büyük kolaylıklar sağlar.

KAYNAKLAR



- [1]. Henry Dreifus, J. Thomas Monk, "Smart Cards – A Guide To Building and Managing Smart Card Applications", John Wiley & Sons, 1997
- [2]. W. Rankl, W. Effing, "Smart Card Handbook", John Wiley & Sons, 1997
- [3]. Min-Shiang Hwang, Cheng-Chi Lee, Yuan-Liang Tang, "A Simple Remote User Authentication Scheme", Mathematical and Computer Modelling 36, s:103-107, 2002
- [4]. L. Lamport, "Password Authentication with insecure communication", Communications of the ACM 24, s:770-772, Ekim 1981
- [5]. H.-Y. Chien, J.-K. Jan, Y.-M. Tseng, "A modified remote log in authentication scheme based on geometric approach", Journal of System and Software 55, s:287-290, 2001
- [6]. G.Horng, "Password authentication without using password table", Information Processing Letters 55, s:247-250, 1995
- [7]. J.K. Jan, Y.Y. Chen, "Paramita wisdom- password authentication scheme without verification tables, The Journal of Systems and Software 42, s:45-57, 1998
- [8]. O. Geldreich, "Modern Cryptography, Probabilistic Proofs and Pseudo-Randomness", Springer Verlag, 1999
- [9]. Carl Ellison, Bruce Schneier, "Ten Risks of PKI: What You're not Being Told about Public Key Infrastructure", Computer Security Journal, Volume XVI, No. 1, 2001
- [10]. R. Anderson: "Security Engineering". John Wiley 2001
- [11]. P. Kocher, "Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems". In: N. Koblitz, editor, Proceedings of Crypto'96. Lecture Notes in Computer Science, 104-113, 1109, Springer Verlag, 1996
- [12]. P. Kocher, J. Jaffe, B. Jun, "Differential Power Analysis". In: M. Wiener, editor, Proceedings of Crypto'99. Lecture Notes in Computer Science, 388-397, 1666, Springer Verlag, 1999
- [13]. D. Boneh, R. DeMillo and R. Lipton, "The Importance of Checking Cryptographic Protocols for Faults". In: W. Fumy, editor, Proceedings of Eurocrypt'97. Lecture Notes in Computer Science, 37-51, 1233, Springer Verlag, 1997



[14]. David Corcoran, David Sims and Bob Hillhouse: Smart Cards and Biometrics: Your Key to PKI. <http://www.linuxjournal.com/article.php?sid=3013>.

Issue 59: March 01, 1999

[15]. Smart Cards in PKI. Available at: <http://deepnight.org/publications/smartcard-howto/smartpki.html>.

[16]. W. Diffie, M. E. Hellman: New Directions in Cryptography. IEEE Trans. Inform. Theory, IT, Vol. 22, No. 6, pp. 644-654, November (1976)

[17]. R. L. Rivest, A. Shamir, L. Adleman: A Method for Obtaining Digital Signatures and Public Key Cryptosystem. Communication of the ACM, Vol. 21, No. 2, pp. 120-126, 1978



ÖZGEÇMİŞ

4 Ekim 1973'de Aydın ilinin Burhaniye (daha sonra Buharkent adını alarak ilçe oldu) kasabasında doğdum. İlkokulu doğduğum kasabada okuduktan sonra Aydın Anadolu Lisesini kazandım. Üniversite lisans eğitimimi Ankara'da Bilkent Elektrik ve Elektronik Mühendisliğinde alarak 1997 yılında mezun oldum. Kocaeli Gebze'de bulunan Tübitak Ulusal Elektronik ve Kriptoloji Araştırma Enstitüsü'nde 1997 yılından beri araştırmacı olarak çalışmaktayım. Yayınladığım makaleler:

1. Mehmet Ercan KURUOĞLU ve İbrahim SOĞUKPINAR: Düşük Maliyetli ve Güvenilir Elektronik Cüzdan. Pamukkale Üniversitesi Bilgi Teknolojileri Kongresi tarafından 2003 yılında kabul edildi. Makaleyi 2 Mayıs 2003 tarihinde Pamukkale Üniversitesindeki kongrede sundum.
2. Mehmet Ercan KURUOĞLU and İbrahim SOĞUKPINAR: Low Cost and Trusted Electronic Purse System Design. Send to ISCIS XVIII – Eighteenth International Symposium on Computer and Information Sciences: <http://www.iscis03.metu.edu.tr> at Middle East Technical University.

EK: AKILLI KARTLA ELEKTRONİK CÜZDAN UYGULAMASI İÇİN DELPHİ5 KULLANILARAK PASCAL DİLİNDE YAZILAN PROGRAM

EK.1. Merkez bilgisayarda bulunan sunucu programı:

```

unit BankServerMain;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  ExtCtrls, StdCtrls, ScktComp, Db, ADODB, ComCtrls, Buttons, HCMngr;

type
  Arr8 = Array [1..8] of BYTE;
  ByteArr10 = Array [1..10] of BYTE;
  ByteArr16 = Array [1..16] of BYTE;
  ByteArr32 = Array [1..32] of BYTE;
  ByteArr64 = Array [1..64] of BYTE;
  PByte = ^byte;

type
  TFormBankServer = class(TForm)
    ServerSocketOfProvider: TServerSocket;
    MemoServer: TMemo;
    Panel1: TPanel;
    Panel2: TPanel;
    ADOConnectionToCustomer: TADOConnection;
    ADOQueryCustomer: TADOQuery;
    DataSourceCustomer: TDataSource;
    ServerSocketOfMerchant: TServerSocket;
    StatusBar1: TStatusBar;
    SpeedButtonClearMemo: TSpeedButton;
    HashManager1: THashManager;
    procedure FormCreate(Sender: TObject);
    procedure FormDestroy(Sender: TObject);
    procedure ServerSocketOfProviderAccept(Sender: TObject;
      Socket: TCustomWinSocket);
    procedure ServerSocketOfProviderClientConnect(Sender: TObject;
      Socket: TCustomWinSocket);
    procedure ServerSocketOfProviderClientDisconnect(Sender: TObject;
      Socket: TCustomWinSocket);
    procedure ServerSocketOfProviderClientError(Sender: TObject;
      Socket: TCustomWinSocket; ErrorEvent: TErrorEvent;
      var ErrorCode: Integer);
    procedure ServerSocketOfProviderClientRead(Sender: TObject;
      Socket: TCustomWinSocket);
    procedure ServerSocketOfProviderListen(Sender: TObject;
      Socket: TCustomWinSocket);
    procedure SpeedButtonClearMemoClick(Sender: TObject);

  private
    { Private declarations }
    procedure Authenticate;
    procedure InsertNewCustomer;
    procedure UpdateBalance;
    procedure UpdateCustomerInfo;
    procedure DeleteCustomer;
    function IsSenderValid(SenderIDNo: Byte): Boolean;
    procedure ClearMessageSend;
    procedure CalculateSecretInfo(ID: String; var ByteData32: ByteArr32);
  end;

```

```

public
{ Public declarations }
end;

type
  CustomerInfo = Record
    CustID      : ByteArr16;
    CustName    : ByteArr16;
    CustSurname : ByteArr16;
    CustBirthday : ByteArr10;
    CustBirthPlace: ByteArr16;
    CustFatherName: ByteArr16;
    CustTaxNo    : ByteArr16;
    CustNationalID: ByteArr16;
    CustHomeAdr  : ByteArr64;
    CustHomeTel  : ByteArr16;
    CustWorkAdr  : ByteArr64;
    CustWorkTel  : ByteArr16;
    CustMobileTel : ByteArr16;
    CustEmail    : ByteArr32;
    CustDueDate  : ByteArr10;
    CustBalance  : ByteArr16;
  end;

  BalanceInfo = Record
    OldBalance: ByteArr16;
    NewBalance: ByteArr16;
  end;

  CryptoInfo = Record
    HashValue  : ByteArr32;
    CryptoValue : ByteArr32;
    ActionTime  : TTimeStamp;
  end;

  DataSend = Record
    CustomerDataExists: Byte;
    CustomerData      : CustomerInfo;
    CryptoDataExists  : Byte;
    CryptoData        : CryptoInfo;
    BalanceDataExists : Byte;
    BalanceData       : BalanceInfo;
  end;

  MessageInfo = Record
    Command  : Byte;
    SenderID : Byte;
    DataExists: Byte;
    DataTag  : DataSend;
  end;

const
  BufferSize = 32768;
  HashSize  = 20;
  SecretKey = 'ThisIsSecretKeyWhichIs32InLength';

// Client Desires:
ClientHello      = $01;
ClientGoodbye    = $02;
AuthenticateAsk  = $03;
NewCustomerInsert = $04;
CustomerInfoUpdate = $05;
BalanceUpdate    = $06;
CustomerDelete   = $07;
BlacklistAlert   = $08;

// Server Responses:
ResponseHello      = $21;
ResponseGoodbye    = $22;
ResponseAuthenticate = $23;
ResponseInsert     = $24;
ResponseUpdateInfo = $25;
ResponseUpdateBalance = $26;
ResponseDelete     = $27;
BlacklistSend      = $28;
BlacklistDelete    = $29;

```

[illegible]

```

procedure StrToByteArr10(Str: String; var ByteData10: ByteArr10);
var
  Cnt: Integer;
begin
  for Cnt:=1 to Length(Str) do begin
    if (Cnt > 10) then break;
    ByteData10[Cnt] := Ord(Str[Cnt]);
  end;
end;

procedure StrToByteArr16(Str: String; var ByteData16: ByteArr16);
var
  Cnt: Integer;
begin
  for Cnt:=1 to Length(Str) do begin
    if (Cnt > 16) then break;
    ByteData16[Cnt] := Ord(Str[Cnt]);
  end;
end;

procedure StrToByteArr32(Str: String; var ByteData32: ByteArr32);
var
  Cnt: Integer;
begin
  for Cnt:=1 to Length(Str) do begin
    if (Cnt > 32) then break;
    ByteData32[Cnt] := Ord(Str[Cnt]);
  end;
end;

procedure StrToByteArr64(Str: String; var ByteData64: ByteArr64);
var
  Cnt: Integer;
begin
  for Cnt:=1 to Length(Str) do begin
    if (Cnt > 64) then break;
    ByteData64[Cnt] := Ord(Str[Cnt]);
  end;
end;

procedure TFormBankServer.FormCreate(Sender: TObject);
begin
  ServerSocketOfProvider.Open;
  //ServerSocketOfMerchant.Open;
  //ReceiveBytes := AllocMem(BufferSize);
  //SendBytes := AllocMem(BufferSize);
  //HashValue := AllocMem(HashSize);
  isTakeHello := False;
end;

procedure TFormBankServer.FormDestroy(Sender: TObject);
begin
  ServerSocketOfProvider.Close;
  //ServerSocketOfMerchant.Close;
  //FreeMem(ReceiveBytes, BufferSize);
  //FreeMem(SendBytes, BufferSize);
  //FreeMem(HashValue, HashSize);
end;

procedure TFormBankServer.ServerSocketOfProviderAccept(Sender: TObject;
  Socket: TCustomWinSocket);
begin
  MemoServer.Lines.Add('ServerSocketBank:-> Accept');
end;

procedure TFormBankServer.ServerSocketOfProviderClientConnect(Sender: TObject;
  Socket: TCustomWinSocket);
begin
  MemoServer.Lines.Add('ServerSocketBank:-> ClientConnect');
end;

procedure TFormBankServer.ServerSocketOfProviderClientDisconnect(Sender: TObject;
  Socket: TCustomWinSocket);
begin
  MemoServer.Lines.Add('ServerSocketBank:-> ClientDisconnect');
end;

```

```

procedure TFormBankServer.ServerSocketOfProviderClientError(Sender: TObject;
  Socket: TCustomWinSocket; ErrorEvent: TErrorEvent;
  var ErrorCode: Integer);
begin
  MemoServer.Lines.Add('ServerSocketBank:-> ClientError, Code: ' +
    IntToStr(ErrorCode));
end;

procedure TFormBankServer.ServerSocketOfProviderClientRead(Sender: TObject;
  Socket: TCustomWinSocket);
var
  ReceiveLen: LongInt;
  strTemp: String;
begin
  MemoServer.Lines.Add('Client:-> Read');
  ReceiveLen := Socket.ReceiveBuf(MessageReceive, SizeOf(MessageInfo));
  MemoServer.Lines.Add(IntToStr(MessageReceive.Command));
  MemoServer.Lines.Add(IntToStr(MessageReceive.SenderID));
  if (MessageReceive.DataExists = Present) then begin
    if (MessageReceive.DataTag.CustomerDataExists = Present) then begin
      strTemp := ByteArrToStr(@MessageReceive.DataTag.CustomerData.CustName, 16);
      MemoServer.Lines.Append(strTemp);
      strTemp := ByteArrToStr(@MessageReceive.DataTag.CustomerData.CustSurname, 16);
      MemoServer.Lines.Append(strTemp);
      strTemp := ByteArrToStr(@MessageReceive.DataTag.CustomerData.CustBirthDay, 10);
      MemoServer.Lines.Append(strTemp);
      strTemp :=
        ByteArrToStr(@MessageReceive.DataTag.CustomerData.CustBirthPlace, 16);
      MemoServer.Lines.Append(strTemp);
      strTemp :=
        ByteArrToStr(@MessageReceive.DataTag.CustomerData.CustFatherName, 16);
      MemoServer.Lines.Append(strTemp);
      strTemp := ByteArrToStr(@MessageReceive.DataTag.CustomerData.CustTaxNo, 16);
      MemoServer.Lines.Append(strTemp);
      strTemp :=
        ByteArrToStr(@MessageReceive.DataTag.CustomerData.CustNationalID, 16);
      MemoServer.Lines.Append(strTemp);
      strTemp := ByteArrToStr(@MessageReceive.DataTag.CustomerData.CustHomeAdr, 64);
      MemoServer.Lines.Append(strTemp);
      strTemp := ByteArrToStr(@MessageReceive.DataTag.CustomerData.CustHomeTel, 16);
      MemoServer.Lines.Append(strTemp);
      strTemp := ByteArrToStr(@MessageReceive.DataTag.CustomerData.CustWorkAdr, 64);
      MemoServer.Lines.Append(strTemp);
      strTemp := ByteArrToStr(@MessageReceive.DataTag.CustomerData.CustWorkTel, 16);
      MemoServer.Lines.Append(strTemp);
      strTemp := ByteArrToStr(@MessageReceive.DataTag.CustomerData.CustMobileTel, 16);
      MemoServer.Lines.Append(strTemp);
      strTemp := ByteArrToStr(@MessageReceive.DataTag.CustomerData.CustEmail, 32);
      MemoServer.Lines.Append(strTemp);
      strTemp := ByteArrToStr(@MessageReceive.DataTag.CustomerData.CustDueDate, 10);
      MemoServer.Lines.Append(strTemp);
      strTemp := ByteArrToStr(@MessageReceive.DataTag.CustomerData.CustBalance, 16);
      MemoServer.Lines.Append(strTemp);
    end;
    if (MessageReceive.DataTag.CryptoDataExists = Present) then begin
      strTemp := ByteArrToStr(@MessageReceive.DataTag.CryptoData.HashValue, 32);
      MemoServer.Lines.Append(strTemp);
      strTemp := ByteArrToStr(@MessageReceive.DataTag.CryptoData.CryptoValue, 32);
      MemoServer.Lines.Append(strTemp);
      strTemp := IntToStr(MessageReceive.DataTag.CryptoData.ActionTime.Time);
      MemoServer.Lines.Append(strTemp);
      strTemp := IntToStr(MessageReceive.DataTag.CryptoData.ActionTime.Date);
      MemoServer.Lines.Append(strTemp);
    end;
    if (MessageReceive.DataTag.BalanceDataExists = Present) then begin
      strTemp := ByteArrToStr(@MessageReceive.DataTag.BalanceData.OldBalance, 16);
      MemoServer.Lines.Append(strTemp);
      strTemp := ByteArrToStr(@MessageReceive.DataTag.BalanceData.NewBalance, 16);
      MemoServer.Lines.Append(strTemp);
    end;
  end;

  isTakeHello := True;
  isAuthenticate := True;
  if (ReceiveLen > 0) then begin
    case MessageReceive.Command of
      ClientHello: begin

```

```

isTakeHello := IsSenderValid(MessageReceive.SenderID);
if (isTakeHello) then begin
    ClearMessageSend;
    MessageSend.Command := ResponseHello;
    MessageSend.SenderID := MessageReceive.SenderID;
end;
end;
ClientGoodbye: begin
    if (isTakeHello and IsSenderValid(MessageReceive.SenderID)) then begin
        ClearMessageSend;
        MessageSend.Command := ResponseGoodbye;
        MessageSend.SenderID := MessageReceive.SenderID;
        isTakeHello := False;
        isAuthenticate := False;
    end;
end;
AuthenticateAsk:
    if (isTakeHello and IsSenderValid(MessageReceive.SenderID)) then
        Authenticate;
NewCustomerInsert:
    if (isTakeHello and IsSenderValid(MessageReceive.SenderID) and
        isAuthenticate) then
        InsertNewCustomer;
CustomerInfoUpdate:
    if (isTakeHello and IsSenderValid(MessageReceive.SenderID) and
        isAuthenticate) then
        UpdateCustomerInfo;
BalanceUpdate:
    if (isTakeHello and IsSenderValid(MessageReceive.SenderID) and
        isAuthenticate) then
        UpdateBalance;
CustomerDelete:
    if (isTakeHello and IsSenderValid(MessageReceive.SenderID) and
        isAuthenticate) then
        DeleteCustomer;
//BlacklistAlert: // do something
end;
Socket.SendBuf(MessageSend, SizeOf(MessageInfo));
end;
end;

procedure TFormBankServer.ServerSocketOfProviderListen(Sender: TObject;
    Socket: TCustomWinSocket);
begin
    MemoServer.Lines.Add('ServerSocketBank:-> Listen');
end;

procedure TFormBankServer.Authenticate;
var
    SecretData: ByteArr32;
    ServerTime: TTimeStamp;
    strSQL: String;
    TimeStr: String;
begin
    strSQL := 'SELECT CustomerID FROM CustomerInfo WHERE (CustomerName = ''';
    strSQL := strSQL + ByteArrToStr(@MessageReceive.DataTag.CustomerData.CustName, 16);
    strSQL := strSQL + ''') and (Surname = ''';
    strSQL := strSQL +
    ByteArrToStr(@MessageReceive.DataTag.CustomerData.CustSurname, 16);
    strSQL := strSQL + ''') and (NationalID = ''';
    strSQL := strSQL +
    ByteArrToStr(@MessageReceive.DataTag.CustomerData.CustNationalID, 16);
    strSQL := strSQL + ''')';

    ADOConnectionToCustomer.Open;
    ADOQueryCustomer.SQL.Clear;
    ADOQueryCustomer.SQL.Add(strSQL);
    ADOQueryCustomer.Open;

    if (ADOQueryCustomer.RowsAffected = 1) then begin
        // calculate h(ID + K)

        CalculateSecretInfo(ADOQueryCustomer.FieldByName('CustomerID').AsString, SecretData);
        MemoServer.Lines.Add('-> Customer ID: ' +
        ADOQueryCustomer.FieldByName('CustomerID').AsString);
        MemoServer.Lines.Add('-> Customer Balance: ' +
        ByteArrToStr(@MessageReceive.DataTag.CustomerData.CustBalance, 16));
    end;
end;

```

```

MemoServer.Lines.Add('-> Secret Key : ' + SecretKey);
MemoServer.Lines.Add('-> Secret Info : ' + ByteArrToStr(@SecretData,32));

ServerTime := DateTimeToTimeStamp(Now);
if ((MessageReceive.DataTag.CryptoData.ActionTime.Date = ServerTime.Date)
    and ((MessageReceive.DataTag.CryptoData.ActionTime.Time - ServerTime.Time) <
50)) then begin

    TimeStr := IntToStr(MessageReceive.DataTag.CryptoData.ActionTime.Date);
    TimeStr := TimeStr +
IntToStr(MessageReceive.DataTag.CryptoData.ActionTime.Time);
    CalculateSecretInfo(TimeStr,SecretData);
    if (ByteArrToStr(@MessageReceive.DataTag.CryptoData.CryptoValue,32)
        = ByteArrToStr(@SecretData,32)) then begin
        ClearMessageSend;
        MessageSend.Command := ResponseAuthenticate;
        MessageSend.SenderID := 255;
        MessageSend.DataExists := Absent;
        isAuthenticate := True;
    end else begin
        ClearMessageSend;
        MessageSend.Command := UnidentifiedCustomer;
        MessageSend.SenderID := 255;
        MessageSend.DataExists := Absent;
    end;
end else begin
    ClearMessageSend;
    MessageSend.Command := UnidentifiedCustomer;
    MessageSend.SenderID := 255;
    MessageSend.DataExists := Absent;
end;
end else begin
    ClearMessageSend;
    MessageSend.Command := UnidentifiedCustomer;
    MessageSend.SenderID := 255;
    MessageSend.DataExists := Absent;
end;

ADOConnectionToCustomer.Close;
end;

procedure TFormBankServer.InsertNewCustomer;
var
    strSQL: String;
    SecretData: ByteArr32;
begin
    strSQL := 'INSERT INTO CustomerInfo (CustomerName,Surname,Birthday,';
    strSQL := strSQL + 'BirthPlace,FatherName,TaxNo,NationalID,HomeAdr,';
    strSQL := strSQL + 'HomeTel,WorkAdr,WorkTel,MobileTel,Email,';
    strSQL := strSQL + 'DueDate,Balance) VALUES ('';
    strSQL := strSQL + ByteArrToStr(@MessageReceive.DataTag.CustomerData.CustName,16);
    strSQL := strSQL + ', '';
    strSQL := strSQL +
ByteArrToStr(@MessageReceive.DataTag.CustomerData.CustSurname,16);
    strSQL := strSQL + ', '';
    strSQL := strSQL +
ByteArrToStr(@MessageReceive.DataTag.CustomerData.CustBirthday,10);
    strSQL := strSQL + ', '';
    strSQL := strSQL +
ByteArrToStr(@MessageReceive.DataTag.CustomerData.CustBirthPlace,16);
    strSQL := strSQL + ', '';
    strSQL := strSQL +
ByteArrToStr(@MessageReceive.DataTag.CustomerData.CustFatherName,16);
    strSQL := strSQL + ', '';
    strSQL := strSQL + ByteArrToStr(@MessageReceive.DataTag.CustomerData.CustTaxNo,16);
    strSQL := strSQL + ', '';
    strSQL := strSQL +
ByteArrToStr(@MessageReceive.DataTag.CustomerData.CustNationalID,16);
    strSQL := strSQL + ', '';
    strSQL := strSQL +
ByteArrToStr(@MessageReceive.DataTag.CustomerData.CustHomeAdr,64);
    strSQL := strSQL + ', '';
    strSQL := strSQL +
ByteArrToStr(@MessageReceive.DataTag.CustomerData.CustHomeTel,16);
    strSQL := strSQL + ', '';
    strSQL := strSQL +
ByteArrToStr(@MessageReceive.DataTag.CustomerData.CustWorkAdr,64);

```

```

    strSQL := strSQL + ', ';;
    strSQL := strSQL +
ByteArrToStr(@MessageReceive.DataTag.CustomerData.CustWorkTel,16);
    strSQL := strSQL + ', ';;
    strSQL := strSQL +
ByteArrToStr(@MessageReceive.DataTag.CustomerData.CustMobileTel,16);
    strSQL := strSQL + ', ';;
    strSQL := strSQL + ByteArrToStr(@MessageReceive.DataTag.CustomerData.CustEmail,32);
    strSQL := strSQL + ', ';;
    strSQL := strSQL +
ByteArrToStr(@MessageReceive.DataTag.CustomerData.CustDueDate,10);
    strSQL := strSQL + ', ';;
    strSQL := strSQL +
ByteArrToStr(@MessageReceive.DataTag.CustomerData.CustBalance,16);
    strSQL := strSQL + '));';

    ADOConnectionToCustomer.Open;
    ADOQueryCustomer.SQL.Clear;
    ADOQueryCustomer.SQL.Add(strSQL);
    ADOQueryCustomer.ExecSQL;

    if (ADOQueryCustomer.RowsAffected = 1) then begin
        MemoServer.Lines.Add('-> Yeni müşteri veritabanına girildi.');
```

strSQL := 'SELECT CustomerID FROM CustomerInfo WHERE (CustomerName = ''';
 strSQL := strSQL +
 ByteArrToStr(@MessageReceive.DataTag.CustomerData.CustName,16);
 strSQL := strSQL + ''') and (Surname = ''';
 strSQL := strSQL +
 ByteArrToStr(@MessageReceive.DataTag.CustomerData.CustSurname,16);
 strSQL := strSQL + ''') and (NationalID = ''';
 strSQL := strSQL +
 ByteArrToStr(@MessageReceive.DataTag.CustomerData.CustNationalID,16);
 strSQL := strSQL + '));';

ADOQueryCustomer.SQL.Clear;
 ADOQueryCustomer.SQL.Add(strSQL);
 ADOQueryCustomer.Open;

// calculate h(ID + K)

CalculateSecretInfo(ADOQueryCustomer.FieldByName('CustomerID').AsString,SecretData);
 MemoServer.Lines.Add('-> Customer ID: ' +
 ADOQueryCustomer.FieldByName('CustomerID').AsString);
 MemoServer.Lines.Add('-> Customer Balance: ' +
 ByteArrToStr(@MessageReceive.DataTag.CustomerData.CustBalance,16));
 MemoServer.Lines.Add('-> Secret Key : ' + SecretKey);
 MemoServer.Lines.Add('-> Secret Info : ' + ByteArrToStr(@SecretData,32));

ClearMessageSend;
 MessageSend.Command := ResponseInsert;
 MessageSend.SenderID := 255;
 MessageSend.DataExists := Present;
 MessageSend.DataTag.CryptoDataExists := Present;
 MessageSend.DataTag.CryptoData.CryptoValue := SecretData;
 end else begin
 ClearMessageSend;
 MessageSend.Command := UnknownError;
 MessageSend.SenderID := 255;
 MessageSend.DataExists := Absent;
 end;

 ADOConnectionToCustomer.Close;
end;

procedure TFormBankServer.UpdateBalance;
var
 strSQL: String;
begin
 strSQL := 'SELECT Balance FROM CustomerInfo WHERE (CustomerName = ''';
 strSQL := strSQL + ByteArrToStr(@MessageReceive.DataTag.CustomerData.CustName,16);
 strSQL := strSQL + ''') and (Surname = ''';
 strSQL := strSQL +
 ByteArrToStr(@MessageReceive.DataTag.CustomerData.CustSurname,16);
 strSQL := strSQL + ''') and (NationalID = ''';
 strSQL := strSQL +
 ByteArrToStr(@MessageReceive.DataTag.CustomerData.CustNationalID,16);
 strSQL := strSQL + '));';

```

ADOConnectionToCustomer.Open;
ADOQueryCustomer.SQL.Clear;
ADOQueryCustomer.SQL.Add(strSQL);
ADOQueryCustomer.Open;

if (ADOQueryCustomer.RecordCount = 1) then begin
  if (ADOQueryCustomer.Fields.Fields[0].AsString =
    ByteArrToStr(@MessageReceive.DataTag.BalanceData.OldBalance,16)) then begin
    strSQL := 'UPDATE CustomerInfo SET Balance = ''';
    strSQL := strSQL +
ByteArrToStr(@MessageReceive.DataTag.BalanceData.NewBalance,16);
    strSQL := strSQL + ''', WHERE (CustomerID = ''';
    strSQL := strSQL + ByteArrToStr(@MessageSend.DataTag.CustomerData.CustID,16);
    strSQL := strSQL + ''')';

    ADOQueryCustomer.SQL.Clear;
    ADOQueryCustomer.SQL.Add(strSQL);
    ADOQueryCustomer.ExecSQL;

    if (ADOQueryCustomer.RowsAffected = 1) then begin
      MemoServer.Lines.Add('-> Müşteri hesabı değiştirildi.');
```

X

```

      ClearMessageSend;
      // MessageSend hazırla
      MessageSend.Command := ResponseUpdateBlance;
      MessageSend.SenderID := 255;
      MessageSend.DataExists := Present;
      MessageSend.DataTag.BalanceDataExists := Present;
      MessageSend.DataTag.BalanceData.OldBalance :=
MessageReceive.DataTag.BalanceData.NewBalance;
      MessageSend.DataTag.BalanceData.NewBalance :=
MessageReceive.DataTag.BalanceData.NewBalance;

      end else begin
        ClearMessageSend;
        // MessageSend hazırla
        MessageSend.Command := UnKnownError;
        MessageSend.SenderID := 255;
        MessageSend.DataExists := Absent;
      end;
    end else begin
      MemoServer.Lines.Add('-> Hata! Kart hesabı, müşteri hesabından farklı.');
```

X

```

      ClearMessageSend;
      // MessageSend hazırla
      MessageSend.Command := BalanceError;
      MessageSend.SenderID := 255;
      MessageSend.DataExists := Absent;
    end;
  end else begin
    MemoServer.Lines.Add('-> Hata! Müşteri bulunamadı.');
```

X

```

    ClearMessageSend;
    // MessageSend hazırla
    MessageSend.Command := UnIdentifiedCustomer;
    MessageSend.SenderID := 255;
    MessageSend.DataExists := Absent;
  end;

  ADOConnectionToCustomer.Close;
end;

procedure TFormBankServer.UpdateCustomerInfo;
var
  strSQL: String;
begin
  strSQL := 'UPDATE CustomerInfo SET HomeAdr = ''';
  strSQL := strSQL + ByteArrToStr(@MessageSend.DataTag.CustomerData.CustHomeAdr,64);
  strSQL := strSQL + ''', HomeTel = ''';
  strSQL := strSQL + ByteArrToStr(@MessageSend.DataTag.CustomerData.CustHomeTel,16);
  strSQL := strSQL + ''', WorkAdr = ''';
  strSQL := strSQL + ByteArrToStr(@MessageSend.DataTag.CustomerData.CustWorkAdr,64);
  strSQL := strSQL + ''', WorkTel = ''';
  strSQL := strSQL + ByteArrToStr(@MessageSend.DataTag.CustomerData.CustWorkTel,64);
  strSQL := strSQL + ''', MobileTel = ''';
  strSQL := strSQL +
ByteArrToStr(@MessageSend.DataTag.CustomerData.CustMobileTel,16);
  strSQL := strSQL + ''', Email = ''';

```

```

    strSQL := strSQL + ByteArrToStr(@MessageSend.DataTag.CustomerData.CustEmail,32);
    strSQL := strSQL + ''', DueDate = ''';
    strSQL := strSQL + ByteArrToStr(@MessageSend.DataTag.CustomerData.CustDueDate,10);
    strSQL := strSQL + ''', WHERE (CustomerName = ''';
    strSQL := strSQL + ByteArrToStr(@MessageReceive.DataTag.CustomerData.CustName,16);
    strSQL := strSQL + ''') and (Surname = ''';
    strSQL := strSQL +
ByteArrToStr(@MessageReceive.DataTag.CustomerData.CustSurname,16);
    strSQL := strSQL + ''') and (NationalID = ''';
    strSQL := strSQL +
ByteArrToStr(@MessageReceive.DataTag.CustomerData.CustNationalID,16);
    strSQL := strSQL + ''')';

    ADOConnectionToCustomer.Open;
    ADOQueryCustomer.SQL.Clear;
    ADOQueryCustomer.SQL.Add(strSQL);
    ADOQueryCustomer.ExecSQL;

    if (ADOQueryCustomer.RowsAffected = 1) then begin
        MemoServer.Lines.Add('-> Müşteri bilgileri değiştirildi.');
```

ClearMessageSend;
 MessageSend.Command := ResponseUpdateInfo;
 MessageSend.SenderID := 255;
 MessageSend.DataExists := Absent;
 end else begin
 MemoServer.Lines.Add('-> Müşteri bulunamadı veya bilgileri değiştirilemedi');

ClearMessageSend;
 MessageSend.Command := UnIdentifiedCustomer;
 MessageSend.SenderID := 255;
 MessageSend.DataExists := Absent;
 end;

ADOConnectionToCustomer.Close;

end;

```

procedure TFormBankServer.DeleteCustomer;
var
    strSQL: String;
begin
    strSQL := 'DELETE FROM CustomerInfo WHERE (CustomerName = ''';
    strSQL := strSQL + ByteArrToStr(@MessageReceive.DataTag.CustomerData.CustName,16);
    strSQL := strSQL + ''') and (Surname = ''';
    strSQL := strSQL +
ByteArrToStr(@MessageReceive.DataTag.CustomerData.CustSurname,16);
    strSQL := strSQL + ''') and (NationalID = ''';
    strSQL := strSQL +
ByteArrToStr(@MessageReceive.DataTag.CustomerData.CustNationalID,16);
    strSQL := strSQL + ''')';

    ADOConnectionToCustomer.Open;
    ADOQueryCustomer.SQL.Clear;
    ADOQueryCustomer.SQL.Add(strSQL);
    ADOQueryCustomer.ExecSQL;

    if (ADOQueryCustomer.RowsAffected = 1) then begin
        MemoServer.Lines.Add('-> Müşteri silindi.');
```

ClearMessageSend;
 MessageSend.Command := ResponseDelete;
 MessageSend.SenderID := 255;
 MessageSend.DataExists := Absent;
 end else begin
 MemoServer.Lines.Add('-> Müşteri bulunamadı veya silinemedi.');

ClearMessageSend;
 MessageSend.Command := UnIdentifiedCustomer;
 MessageSend.SenderID := 255;
 MessageSend.DataExists := Absent;
 end;

ADOConnectionToCustomer.Close;

end; *

```

function TFormBankServer.IsSenderValid(SenderIDNo: Byte): Boolean;
```

```

begin
  case SenderIDNo of
    100..110: Result := True;
  else
    Result := False;
  end;
end;

procedure TFormBankServer.ClearMessageSend;
begin
  MessageSend.Command := Absent;
  MessageSend.SenderID := Absent;
  MessageSend.DataExists := Absent;
  MessageSend.DataTag.CustomerDataExists := Absent;
  MessageSend.DataTag.CustomerData.CustID := ZeroArr16;
  MessageSend.DataTag.CustomerData.CustName := ZeroArr16;
  MessageSend.DataTag.CustomerData.CustSurname := ZeroArr16;
  MessageSend.DataTag.CustomerData.CustBirthDay := ZeroArr10;
  MessageSend.DataTag.CustomerData.CustBirthPlace := ZeroArr16;
  MessageSend.DataTag.CustomerData.CustFatherName := ZeroArr16;
  MessageSend.DataTag.CustomerData.CustTaxNo := ZeroArr16;
  MessageSend.DataTag.CustomerData.CustNationalID := ZeroArr16;
  MessageSend.DataTag.CustomerData.CustHomeAdr := ZeroArr64;
  MessageSend.DataTag.CustomerData.CustHomeTel := ZeroArr16;
  MessageSend.DataTag.CustomerData.CustWorkAdr := ZeroArr64;
  MessageSend.DataTag.CustomerData.CustWorkTel := ZeroArr16;
  MessageSend.DataTag.CustomerData.CustMobileTel := ZeroArr16;
  MessageSend.DataTag.CustomerData.CustEmail := ZeroArr32;
  MessageSend.DataTag.CustomerData.CustDueDate := ZeroArr10;
  MessageSend.DataTag.CustomerData.CustBalance := ZeroArr16;
  MessageSend.DataTag.CryptoDataExists := Absent;
  MessageSend.DataTag.CryptoData.HashValue := ZeroArr32;
  MessageSend.DataTag.CryptoData.CryptoValue := ZeroArr32;
  MessageSend.DataTag.CryptoData.ActionTime.Time := 0;
  MessageSend.DataTag.CryptoData.ActionTime.Date := 0;
  MessageSend.DataTag.BalanceDataExists := Absent;
  MessageSend.DataTag.BalanceData.OldBalance := ZeroArr16;
  MessageSend.DataTag.BalanceData.NewBalance := ZeroArr16;
end;

procedure TFormBankServer.SpeedButtonClearMemoClick(Sender: TObject);
begin
  MemoServer.Lines.Clear;
end;

procedure TFormBankServer.CalculateSecretInfo(ID: String;
                                              var ByteData32: ByteArr32);
var
  Cnt: Integer;
  HashStr: String;
begin
  for Cnt:=1 to 16 do begin
    ByteData32[Cnt] := Ord(ID[Cnt]) xor Ord(SecretKey[Cnt]);
  end;
  for Cnt:=1 to 16 do begin
    ByteData32[Cnt+16] := Ord(ID[Cnt]) xor Ord(SecretKey[Cnt+16]);
  end;
  HashStr := ByteArrToStr(@ByteData32, 32);
  HashManager1.CalcString(HashStr);
  HashStr := HashManager1.DigestString[16];
  StrToByteArr32(HashStr, ByteData32);
end;

end.

```

EK.2. Kart yükleyici bilgisayarda bulunan program:

```

unit ProviderMain;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  Buttons, Mask, StdCtrls, ComCtrls, ToolWin, ExtCtrls, Menus, Db, ADODB,
  UserLogin, PasswordCheck, UpdateBalance, SCardC32, Gauges, ChangePIN,
  ScktComp, GetPIN, HCMngr;

type
  TFormProvider = class(TForm)
    PanelActionInformation: TPanel;
    PanelCustomerInformation: TPanel;
    PanelFormActions: TPanel;
    StatusBarSQL: TStatusBar;
    MainMenu1: TMainMenu;
    PanelCardActions: TPanel;
    BitBtn1: TBitBtn;
    SpeedButtonLoad: TSpeedButton;
    SpeedButtonUpdateInfo: TSpeedButton;
    SpeedButtonDeleteCustomer: TSpeedButton;
    GroupBox1: TGroupBox;
    Panel4: TPanel;
    Label1: TLabel;
    Panel5: TPanel;
    Label2: TLabel;
    EditName: TEdit;
    Panel6: TPanel;
    Label3: TLabel;
    EditSurname: TEdit;
    Panel7: TPanel;
    Label4: TLabel;
    MaskEditBalance: TMaskEdit;
    SmartCard1: TSmartCard;
    LabelSCardInfo: TLabel;
    GaugeSCard: TGauge;
    MaskEditCardNo: TMaskEdit;
    Label5: TLabel;
    SpeedButtonChangePIN: TSpeedButton;
    MemoOperationInf: TMemo;
    Bevel1: TBevel;
    Bevel2: TBevel;
    Bevel3: TBevel;
    SpeedButtonCredit: TSpeedButton;
    SpeedButtonDebit: TSpeedButton;
    SpeedButtonConfirmUpdateInfo: TSpeedButton;
    Panel11: TPanel;
    Label8: TLabel;
    MaskEditBirthDay: TMaskEdit;
    Panel3: TPanel;
    Label10: TLabel;
    EditPlaceofBirth: TEdit;
    Panel19: TPanel;
    Label20: TLabel;
    EditFatherName: TEdit;
    Panel10: TPanel;
    Label11: TLabel;
    EditTaxNo: TEdit;
    Panel11: TPanel;
    Label12: TLabel;
    EditIdNo: TEdit;
    Panel13: TPanel;
    Label14: TLabel;
    MaskEditHomeTel: TMaskEdit;
    Panel17: TPanel;
    Label18: TLabel;
    MaskEditWorkTel: TMaskEdit;
    Panel18: TPanel;
    Label19: TLabel;
    MaskEditMobileTel: TMaskEdit;
    Panel15: TPanel;
  end;

```

```

Label16: TLabel;
EditEmail: TEdit;
Panel2: TPanel;
Label9: TLabel;
EditHomeAddress: TEdit;
Panel16: TPanel;
Label17: TLabel;
EditWorkAddress: TEdit;
Panel9: TPanel;
Label7: TLabel;
MaskEditDueDate: TMaskEdit;
Bevel5: TBevel;
Bevel6: TBevel;
ClientSocketToServer: TClientSocket;
SpeedButtonClearMemo: TSpeedButton;
SpeedButtonConfirmLoad: TSpeedButton;
SpeedButtonCancelLoad: TSpeedButton;
SpeedButtonCancelUpdateInfo: TSpeedButton;
HashManager1: THashManager;
procedure SpeedButtonLoadClick(Sender: TObject);
procedure SpeedButtonUpdateInfoClick(Sender: TObject);
procedure FormShow(Sender: TObject);
procedure SpeedButtonDeleteCustomerClick(Sender: TObject);
procedure SmartCard1CardActive(Sender: TObject; DeviceIndex: Integer);
procedure SmartCard1Progress(Sender: TObject; DeviceIndex,
    Progress: Integer);
procedure SmartCard1CardDetect(Sender: TObject; DeviceIndex: Integer);
procedure SmartCard1CardInvalid(Sender: TObject; DeviceIndex: Integer);
procedure SmartCard1CardLock(Sender: TObject; DeviceIndex,
    TaskIndex: Integer);
procedure SmartCard1CardValid(Sender: TObject; DeviceIndex: Integer);
procedure SmartCard1CardWait(Sender: TObject; DeviceIndex: Integer);
procedure SmartCard1DeviceError(Sender: TObject; DeviceIndex: Integer);
procedure SmartCard1DeviceSearch(Sender: TObject; Port,
    Progress: Integer);
function ReadDataFromSCard(StartAdr, Len: String): Boolean;
function WriteDataToSCard(DataIn, StartAdr, Len: String): Boolean;
function VerifyPinOfSCard: Boolean;
function ChangePinOfSCard(OldPIN, NewPIN: String): Boolean;
function ExecSCardCmd(Cmd, DataIn: String): Boolean;
procedure SpeedButtonChangePINClick(Sender: TObject);
procedure ClientSocketToServerConnect(Sender: TObject;
    Socket: TCustomWinSocket);
procedure ClientSocketToServerConnecting(Sender: TObject;
    Socket: TCustomWinSocket);
procedure ClientSocketToServerDisconnect(Sender: TObject;
    Socket: TCustomWinSocket);
procedure ClientSocketToServerError(Sender: TObject;
    Socket: TCustomWinSocket; ErrorEvent: TErrorEvent;
    var ErrorCode: Integer);
procedure ClientSocketToServerRead(Sender: TObject;
    Socket: TCustomWinSocket);
procedure ClientSocketToServerWrite(Sender: TObject;
    Socket: TCustomWinSocket);
procedure SpeedButtonClearMemoClick(Sender: TObject);
procedure SpeedButtonConfirmLoadClick(Sender: TObject);
procedure SpeedButtonCancelLoadClick(Sender: TObject);
procedure SpeedButtonConfirmUpdateInfoClick(Sender: TObject);
function isFilled: Boolean;
function isChanged: Boolean;
procedure SpeedButtonCancelUpdateInfoClick(Sender: TObject);
procedure SpeedButtonCreditClick(Sender: TObject);
procedure SpeedButtonDebitClick(Sender: TObject);
procedure ClearMessageSend;
function IsSenderValid(SenderIDNo: Byte): Boolean;
function AuthenticateCustomer: Boolean;
procedure InsertNewCustomer;
procedure UpdateCustomerInfo;
procedure UpdateCustomerBalance;
procedure DeleteCustomer;
procedure SendHello;
private
{ Private declarations }
public
{ Public declarations }
end;

```

```

type
  Arr8 = Array [1..8] of BYTE;
  ByteArr10 = Array [1..10] of BYTE;
  ByteArr16 = Array [1..16] of BYTE;
  ByteArr32 = Array [1..32] of BYTE;
  ByteArr64 = Array [1..64] of BYTE;
  PByte = ^byte;

  CustomerInfo = Record
    CustID      : ByteArr16;
    CustName    : ByteArr16;
    CustSurname : ByteArr16;
    CustBirthday : ByteArr10;
    CustBirthPlace: ByteArr16;
    CustFatherName: ByteArr16;
    CustTaxNo   : ByteArr16;
    CustNationalID: ByteArr16;
    CustHomeAdr : ByteArr64;
    CustHomeTel : ByteArr16;
    CustWorkAdr : ByteArr64;
    CustWorkTel : ByteArr16;
    CustMobileTel : ByteArr16;
    CustEmail   : ByteArr32;
    CustDueDate : ByteArr10;
    CustBalance : ByteArr16;
  end;

  BalanceInfo = Record
    OldBalance: ByteArr16;
    NewBalance: ByteArr16;
  end;

  CryptoInfo = Record
    HashValue : ByteArr32;
    CryptoValue : ByteArr32;
    ActionTime : TTimeStamp;
  end;

  DataSend = Record
    CustomerDataExists: Byte;
    CustomerData      : CustomerInfo;
    CryptoDataExists  : Byte;
    CryptoData        : CryptoInfo;
    BalanceDataExists : Byte;
    BalanceData       : BalanceInfo;
  end;

  MessageInfo = Record
    Command : Byte;
    SenderID : Byte;
    DataExists: Byte;
    DataTag : DataSend;
  end;

const
  BufferSize = 32768;
  HashSize = 20;

  ClientID = $64;

// Client Desires:
  ClientHello      = $01;
  ClientGoodbye    = $02;
  AuthenticateAsk  = $03;
  NewCustomerInsert = $04;
  CustomerInfoUpdate = $05;
  BalanceUpdate    = $06;
  CustomerDelete   = $07;
  BlacklistAlert   = $08;

// Server Responses:
  ResponseHello      = $21;
  ResponseGoodbye    = $22;
  ResponseAuthenticate = $23;
  ResponseInsert     = $24;
  ResponseUpdateInfo = $25;
  ResponseUpdateBlance = $26;

```


59

```

// This are keys to be written to the card during personalization.
Keys : array [1..9,1..8] of BYTE = (($59,$49,$4E,$25,$43,$4F,$44,$45), // "PIN-
CODE" = PIN
                                ($41,$55,$54,$48,$43,$41,$52,$44), //
"AUTHCARD" = Auth Key Kc
                                ($41,$55,$54,$48,$54,$45,$52,$4D), //
"AUTHTERM" = Auth Key Kt
                                ($52,$41,$4E,$44,$53,$45,$45,$44), //
"RANDSEED" = RND seed (this can be any value)
                                ($41,$50,$50,$43,$4F,$44,$45,$31), //
"APPCODE1" = Application Code 1
                                ($41,$50,$50,$43,$4F,$44,$45,$32), //
"APPCODE2" = Application Code 2
                                ($41,$50,$50,$43,$4F,$44,$45,$33), //
"APPCODE3" = Application Code 3
                                ($41,$50,$50,$43,$4F,$44,$45,$34), //
"APPCODE4" = Application Code 4
                                ($41,$50,$50,$43,$4F,$44,$45,$35)); //
"APPCODE5" = Application Code 5

procedure Decrypt(data : PByte; Key : PByte); cdecl; external 'des.dll';
procedure Encrypt(data : PByte; Key : PByte); cdecl; external 'des.dll';

implementation

{$R *.DFM}

function ByteArrToStr(ByteData: PByteArray; DataLen: Integer): String;
var
  Cnt: Integer;
begin
  Result := '';
  for Cnt:=1 to DataLen do begin
    if (ByteData[Cnt] = $00) then Continue;
    Result := Result + Chr(ByteData[Cnt]);
  end;
end;

procedure StrToByteArr10(Str: String; var ByteData10: ByteArr10);
var
  Cnt: Integer;
begin
  for Cnt:=1 to Length(Str) do begin
    if (Cnt > 10) then break;
    ByteData10[Cnt] := Ord(Str[Cnt]);
  end;
end;

procedure StrToByteArr16(Str: String; var ByteData16: ByteArr16);
var
  Cnt: Integer;
begin
  for Cnt:=1 to Length(Str) do begin
    if (Cnt > 16) then break;
    ByteData16[Cnt] := Ord(Str[Cnt]);
  end;
end;

procedure StrToByteArr32(Str: String; var ByteData32: ByteArr32);
var
  Cnt: Integer;
begin
  for Cnt:=1 to Length(Str) do begin
    if (Cnt > 32) then break;
    ByteData32[Cnt] := Ord(Str[Cnt]);
  end;
end;

procedure StrToByteArr64(Str: String; var ByteData64: ByteArr64);
var
  Cnt: Integer;
begin
  for Cnt:=1 to Length(Str) do begin
    if (Cnt > 64) then break;
    ByteData64[Cnt] := Ord(Str[Cnt]);
  end;
end;

```

60.

```

function IntToStrEx(Val,Digits: LongInt): String;
var A: LongInt;
begin
  A:=Abs(Val);
  Str(A,Result);
  while Length(Result) < Digits do Result:='0'+Result;
  if Val < 0 then Result:='-'+Result;
end;

function HexChrVal (HexChr: Char): Byte;
begin
  case UpCase(HexChr) OF
    '0'..'9': HexChrVal:=Byte(HexChr) - Byte('0');
    'A'..'F': HexChrVal:=Byte(UpCase(HexChr)) - Byte('A') + 10;
    else HexChrVal:=$FF;
  end;
end;

function TxtToHexStr(const Txt:String): String;
var C: LongInt;
begin
  Result:='';
  for C:=1 to Length(Txt) do Result := Result + IntToHex(Ord(Txt[C]),2) + ' ';
end;

function HexStrToTxt(Hex:String): String;
var I:LongInt;
    TmpChr, TmpVal, Counter: Byte;
begin
  Result:=''; TmpChr:=0; Counter:=0; //TmpVal:=0;
  for I:=1 to Length(Hex) do begin
    if (Hex[I]=' ') then begin
      if (Counter>0) then begin Result:=Result+Chr(TmpChr); Counter:=0; end;
      TmpChr:=0;
    end else begin
      TmpVal:=HexChrVal(Hex[I]);
      if (TmpVal=$FF) then TmpVal:=0;
      TmpChr:=(TmpChr shl 4)+ TmpVal;
      Inc(Counter);
      if (Counter=2) then begin
        Result:=Result+Chr(TmpChr); Counter:=0; TmpChr:=0;
      end;
    end;
  end;
  if (Counter>0) then Result:=Result+Chr(TmpChr)
end;

procedure TFormProvider.SpeedButtonLoadClick(Sender: TObject);
var
  DueDate: TDateTime;
  Year, Month, Day: Word;
begin
  StatusBarSQL.SimpleText := 'Yeni kart yükleme ve müşteri girme işlemi';
  FormPassword.Caption := strUserID + ', yeni müşteri eklemek için şifre giriniz!';
  FormPassword.ShowModal;
  if (FormPassword.ModalResult = mrOk) then begin
    // daha başka eklenecekler olabilir mi?
    EditName.Enabled := True;
    EditSurname.Enabled := True;
    EditFatherName.Enabled := True;
    MaskEditBirthDay.Enabled := True;
    EditPlaceofBirth.Enabled := True;
    EditTaxNo.Enabled := True;
    EditIdNo.Enabled := True;
    EditHomeAddress.Enabled := True;
    MaskEditHomeTel.Enabled := True;
    EditWorkAddress.Enabled := True;
    MaskEditWorkTel.Enabled := True;
    EditEMail.Enabled := True;
    MaskEditMobileTel.Enabled := True;
    MaskEditBalance.Enabled := True;

    DueDate := Now;
    DecodeDate(DueDate, Year, Month, Day);
    Year := Year + 1;
    DueDate := EncodeDate(Year, Month, Day);
  end;
end;

```

MÜHÜR KULLANILMAMIŞTIR
T.C. MİLLÎ EĞİTİM BAKANLIĞI

```

MaskEditDueDate.Text := DateToStr(DueDate);

EditName.SetFocus;
SpeedButtonLoad.Enabled := False;
SpeedButtonConfirmLoad.Enabled := True;
SpeedButtonCancelLoad.Enabled := True;
if not (isConnectedToServer) then begin
    if (ClientSocketToServer.Socket <> nil) then
        ClientSocketToServer.Open;
    end;
end
else if (FormPassword.ModalResult = mrAbort) then
    Close;
end;

procedure TFormProvider.SpeedButtonUpdateInfoClick(Sender: TObject);
begin
    StatusBarSQL.SimpleText := 'Müşteri bilgilerini değiştirme işlemi';
    FormPassword.Caption := strUserID + ', müşteri bilgilerini değiştirmek için şifre giriniz!';
    FormPassword.ShowModal;
    if (FormPassword.ModalResult = mrOk) then begin
        // daha başka eklenecekler olabilir mi?
        EditHomeAddress.Enabled := True;
        MaskEditHomeTel.Enabled := True;
        EditWorkAddress.Enabled := True;
        MaskEditWorkTel.Enabled := True;
        EditEmail.Enabled := True;
        MaskEditMobileTel.Enabled := True;

        SpeedButtonUpdateInfo.Enabled := False;
        SpeedButtonConfirmUpdateInfo.Enabled := True;
        SpeedButtonCancelUpdateInfo.Enabled := True;
        SpeedButtonCredit.Enabled := False;
        SpeedButtonDebit.Enabled := False;
        SpeedButtonChangePIN.Enabled := False;
        SpeedButtonDeleteCustomer.Enabled := False;
        if not (isConnectedToServer) then begin
            if (ClientSocketToServer.Socket <> nil) then
                ClientSocketToServer.Open;
            end;
        end
        else if (FormPassword.ModalResult = mrAbort) then
            Close;
        end;
end;

procedure TFormProvider.FormShow(Sender: TObject);
begin
    if (FormUserLogin.ShowModal() = mrOk) then begin
        strUserID := FormUserLogin.EditUserID.Text;
        FormPassword.strPasswordCheck := FormUserLogin.EditPassword.Text;
    end else
        Close();

    isAuthenticated := False;
    StatusBarSQL.SimpleText := strUserID + ', sisteme hoşgeldiniz';
    Application.ProcessMessages;
    SmartCard1.Active := True;
end;

procedure TFormProvider.SpeedButtonDeleteCustomerClick(Sender: TObject);
begin
    StatusBarSQL.SimpleText := 'Müşteri ve kart silme işlemi';
    FormPassword.Caption := strUserID + ', müşteri bilgilerini silmek için şifre giriniz!';
    FormPassword.ShowModal;
    if (FormPassword.ModalResult = mrOk) then begin
        // daha başka eklenecekler olabilir mi?
        ClearMessageSend;
        MessageSend.Command := CustomerDelete;
        MessageSend.SenderID := ClientID;
        MessageSend.DataExists := Present;
        MessageSend.DataTag.CustomerDataExists := Present;
        StrToByteArr16(EditName.Text, MessageSend.DataTag.CustomerData.CustName);
        StrToByteArr16(EditSurname.Text, MessageSend.DataTag.CustomerData.CustSurname);
        StrToByteArr16(EditName.Text, MessageSend.DataTag.CustomerData.CustNationalID);
        if (isConnectedToServer) then begin

```



```
ClientSocketToServer.Socket.SendBuf(MessageSend,SizeOf(MessageInfo));
MemoOperationInf.Lines.Add('-> Server''a müşteri ve kart silme bilgisi
gönderildi.'');
end else if (ClientSocketToServer.Socket <> nil) then begin
ClientSocketToServer.Open;
if (IsConnectedToServer) then begin
ClientSocketToServer.Socket.SendBuf(MessageSend,SizeOf(MessageInfo));
MemoOperationInf.Lines.Add('-> Server''a müşteri ve kart silme bilgisi
gönderildi.'');
end else
MemoOperationInf.Lines.Add('-> Server''la bağlantı kurulamadı.'');
end else
MemoOperationInf.Lines.Add('-> Server''la bağlantı kurulamadı.'');
end
else if (FormPassword.ModalResult = mrAbort) then
Close;

end;

procedure TFormProvider.SmartCard1CardActive(Sender: TObject;
DeviceIndex: Integer);
var
Protocol, CardType, PinSize: String;
begin
StatusBarSQL.SimpleText := 'Kart aktif ediliyor';
MemoOperationInf.Lines.Clear;
Protocol:=SmartCard1.ComandStr(cmCard+cmInfo+cmProtocol,'');
CardType:=SmartCard1.ComandStr(cmCard+cmInfo+cmType,'');
PinSize:=SmartCard1.ComandStr(cmCard+cmInfo+cmPinSize,'');
if ((PinSize = '2') and ((Protocol='2W') or (Protocol='3W'))) then begin
if (ReadDataFromSCard('32','1')) then begin
if (DataHex = '30 ') then begin
// read secret data and user info from the card
if (ReadDataFromSCard('33','32')) then begin
if (AuthenticateCustomer) then begin
ReadDataFromSCard('65','16');
EditName.Text := DataASCII;

ReadDataFromSCard('81','16');
EditSurname.Text := DataASCII;

ReadDataFromSCard('97','10');
MaskEditBirthday.Text := DataASCII;

ReadDataFromSCard('107','16');
EditPlaceofBirth.Text := DataASCII;

ReadDataFromSCard('123','16');
EditFatherName.Text := DataASCII;

ReadDataFromSCard('139','16');
EditTaxNo.Text := DataASCII;

ReadDataFromSCard('155','16');
EditIdNo.Text := DataASCII;

ReadDataFromSCard('171','64');
EditHomeAddress.Text := DataASCII;

ReadDataFromSCard('235','16');
MaskEditHomeTel.Text := DataASCII;

ReadDataFromSCard('251','64');
EditWorkAddress.Text := DataASCII;

ReadDataFromSCard('315','16');
MaskEditWorkTel.Text := DataASCII;

ReadDataFromSCard('331','16');
MaskEditMobileTel.Text := DataASCII;

ReadDataFromSCard('347','32');
EditEmail.Text := DataASCII;

ReadDataFromSCard('379','10');
MaskEditDueDate.Text := DataASCII;
```



```
ReadDataFromSCard('389','16');
MaskEditBalance.Text := DataASCII;

SpeedButtonUpdateInfo.Enabled := True;
SpeedButtonCredit.Enabled := True;
SpeedButtonDebit.Enabled := True;
SpeedButtonChangePIN.Enabled := True;
SpeedButtonDeleteCustomer.Enabled := True;
MemoOperationInf.Lines.Add('-> Geçerli kart ve veritabanında müşteri
tanındı.');
```

```
    end else begin
        SpeedButtonLoad.Enabled := True;
        MemoOperationInf.Lines.Add('-> Geçersiz kart veya kullanıcı, yeniden
yükleme yapın!');
```

```
    end;
    end else
        MemoOperationInf.Lines.Add('-> Geçersiz veya arızalı kart! Kart
okunamadı.');
```

```
    end else begin
        SpeedButtonLoad.Enabled := True;
        MemoOperationInf.Lines.Add('-> Boş veya geçersiz kart, (yeniden) yükleme
yapın!');
```

```
    end;
    end else
        MemoOperationInf.Lines.Add('-> Geçersiz veya arızalı kart! Kart okunamadı.');
```

```
    end else begin
        StatusBarSQL.SimpleText := 'Hata! Yanlış kart tipi!';
        MemoOperationInf.Lines.Add('-> Yanlış kart girildi! PIN 4 harfli, Protokol 2W
veya 3W olacak.');
```

```
    end;
end;
```

```
procedure TFormProvider.SmartCard1Progress(Sender: TObject; DeviceIndex,
    Progress: Integer);
begin
    GaugeSCard.Progress:=Progress;
end;
```

```
procedure TFormProvider.SmartCard1CardDetect(Sender: TObject;
    DeviceIndex: Integer);
begin
    //
end;
```

```
procedure TFormProvider.SmartCard1CardInvalid(Sender: TObject;
    DeviceIndex: Integer);
begin
    //
end;
```

```
procedure TFormProvider.SmartCard1CardLock(Sender: TObject; DeviceIndex,
    TaskIndex: Integer);
begin
    //
end;
```

```
procedure TFormProvider.SmartCard1CardValid(Sender: TObject;
    DeviceIndex: Integer);
begin
    //
end;
```

```
procedure TFormProvider.SmartCard1CardWait(Sender: TObject;
    DeviceIndex: Integer);
begin
    MaskEditCardNo.Text := '';
    MaskEditBalance.Text := '';
    EditName.Text := '';
    EditSurname.Text := '';
    EditFatherName.Text := '';
    MaskEditBirthDay.Text := '';
    EditPlaceofBirth.Text := '';
    EditTaxNo.Text := '';
    EditIdNo.Text := '';
    EditHomeAddress.Text := '';
    MaskEditHomeTel.Text := '';
    EditWorkAddress.Text := '';
```

```

MaskEditWorkTel.Text := '';
EditEmail.Text := '';
MaskEditMobileTel.Text := '';
MaskEditDueDate.Text := '';

SpeedButtonLoad.Enabled := False;
SpeedButtonUpdateInfo.Enabled := False;
SpeedButtonDeleteCustomer.Enabled := False;
SpeedButtonChangePIN.Enabled := False;
end;

procedure TFormProvider.SmartCardDeviceError(Sender: TObject;
  DeviceIndex: Integer);
begin
  //
end;

procedure TFormProvider.SmartCardDeviceSearch(Sender: TObject; Port,
  Progress: Integer);
begin
  //
end;

function TFormProvider.ReadDataFromSCard(StartAdr, Len: String): Boolean;
var
  Cmd: String;
begin
  Cmd := 'Card,MemRead,';
  Cmd := Cmd + StartAdr;
  Cmd := Cmd + ',';
  Cmd := Cmd + Len;
  if (ExecSCardCmd(Cmd, '')) then
    Result := True
  else begin
    Result := False;
    StatusBarSQL.SimpleText := 'Hata! Kart okunamadı!';
    end;
    Sleep(1000);
  end;

function TFormProvider.WriteDataToSCard(DataIn, StartAdr, Len: String): Boolean;
var
  Cmd: String;
begin
  Cmd := 'Card,MemWrite,';
  Cmd := Cmd + StartAdr;
  Cmd := Cmd + ',';
  Cmd := Cmd + Len;
  if (ExecSCardCmd(Cmd, DataIn)) then
    Result := True
  else begin
    Result := False;
    StatusBarSQL.SimpleText := 'Hata! Karta yazılamadı!';
    end;
    Sleep(1000);
  end;

function TFormProvider.VerifyPinOfSCard: Boolean;
var
  Cmd: String;
begin
  Cmd := 'Card,MemVerifyPin,';
  Cmd := Cmd + PIN;
  if (ExecSCardCmd(Cmd, '')) then
    Result := True
  else begin
    Result := False;
    StatusBarSQL.SimpleText := 'Hata! Girilen PIN yanlış!';
    end;
    Sleep(1000);
  end;

function TFormProvider.ChangePinOfSCard(OldPIN, NewPIN: String): Boolean;
var
  Cmd: String;
begin
  Cmd := 'Card,MemChangePin,';

```

```

Cmd := Cmd + OldPIN;
Cmd := Cmd + ',';
Cmd := Cmd + NewPIN;
if (ExecSCardCmd(Cmd, '')) then
    Result := True
else begin
    Result := False;
    StatusBarSQL.SimpleText := 'Hata! Kart PIN'i deđiřtirilemedi!';
end;
Sleep(1000);
end;

function TFormProvider.ExecSCardCmd(Cmd, DataIn: String): Boolean;
const BufferSize = 32768;
var P, pDataIn: PChar;
    //DataInHex: String;
    C: Integer;
    Len, LenIn: LongInt;
begin
    P:=AllocMem(BufferSize);
    if DataIn='' then begin
        pDataIn:=nil;
        LenIn:=0;
    end else begin
        LenIn := Length(DataIn);
        pDataIn:=@DataIn[1];
    end;

    (* If Cmd is an invalid command, the SCardServer will create a message box
    which must be confirmed. After confirmation the program will continue
    and the component raises an ESmartCard exception.
    So, if this is the only line in the Memo field, please check for any
    message box from the SCardServer. This can happen during your development
    phase, but your final release should never use any invalid commands! *)
    try

        (* now we execute the command, Len returns the length of the data in P. *)
        Len:=SmartCard1.Comand(Cmd, pDataIn, LenIn, P, BufferSize);
        (* The component throws an exception if an error occurs. Here
        we don't do any detailed error handling, but our try-finally
        block take care that at least the buffer p is freed again *)

        Result := True;
        if Len <> 0 then begin
            (*DataOut in HEX *)
            DataHEX:='';
            for C:=0 to Len-1 do
                DataHEX:=DataHEX+IntToHex(Ord(P[C]), 2)+' ';

            (*DataOut in ASCII, non printable characters become "" *)
            DataASCII:='';
            for C:=0 to Len-1 do
                case P[C] of
                    #0..#31: DataASCII:=DataASCII+'^';
                    else DataASCII:=DataASCII+P[C];
                end;
            end;
        end;
    except
        on E:ESmartCard do begin (* the component creates this exception *)
            Result := False;
            Application.MessageBox(PChar(E.Message), 'Akıllı Kart hatası!', MB_OK);
        end;
        on E:Exception do begin (* we'd better catch all exceptions here *)
            Result := False;
            Application.MessageBox(PChar(E.Message), 'Sistem hatası!', MB_OK);
        end;
    end;
    FreeMem(P, BufferSize);
end;

procedure TFormProvider.SpeedButtonChangePINClick(Sender: TObject);
var
    Cnt: Integer;
    HashStr: String;
    SecretInfo: ByteArr32;
begin
    StatusBarSQL.SimpleText := 'PIN deđiřtirme iřlemi';

```

```

FormPassword.Caption := strUserID + ', PIN değiştirmek için şifre giriniz!';
FormPassword.ShowModal;
if (FormPassword.ModalResult = mrOk) then begin
    FormChangePIN.EditOldPIN.Text := '';
    if (FormChangePIN.ShowModal = mrOk) then begin
        if (ChangePinOfSCard(FormChangePIN.EditOldPIN.Text,
            FormChangePIN.EditNewPIN.Text)) then begin
            if (ReadDataFromSCard('33','32')) then begin
                HashManager1.CalcString(FormChangePIN.EditOldPIN.Text);
                HashStr := HashManager1.DigestString[16];
                for Cnt:=1 to 32 do begin
                    SecretInfo[Cnt] := Ord(DataASCII[Cnt]) xor Ord(HashStr[Cnt]);
                end;
                HashManager1.CalcString(FormChangePIN.EditNewPIN.Text);
                HashStr := HashManager1.DigestString[16];
                for Cnt:=1 to 32 do begin
                    SecretInfo[Cnt] := SecretInfo[Cnt] xor Ord(HashStr[Cnt]);
                end;
                HashStr := ByteArrToStr(@SecretInfo,32);
                if (WriteDataToSCard(HashStr,'33','32')) then
                    MemoOperationInf.Lines.Add('-> Müşteri PIN'i başarıyla değiştirildi.')
                else
                    MemoOperationInf.Lines.Add('-> HATA! Kart PIN'i değiştirildi. Fakat
karta yazılamadı.');
```

X

```

                    end;
                end;
            end;
        end;
    else if (FormPassword.ModalResult = mrAbort) then
        Close;
end;

end;

procedure TFormProvider.ClientSocketToServerConnect(Sender: TObject;
    Socket: TCustomWinSocket);
begin
    MemoOperationInf.Lines.Add('-> ClientSocketToServer: Connect');
    isConnectedToServer := True;
end;

procedure TFormProvider.ClientSocketToServerConnecting(Sender: TObject;
    Socket: TCustomWinSocket);
begin
    MemoOperationInf.Lines.Add('-> ClientSocketToServer: Connecting');
    // no code need I suppose
end;

procedure TFormProvider.ClientSocketToServerDisconnect(Sender: TObject;
    Socket: TCustomWinSocket);
begin
    MemoOperationInf.Lines.Add('-> ClientSocketToServer: Disconnect');
    isConnectedToServer := False;
end;

procedure TFormProvider.ClientSocketToServerError(Sender: TObject;
    Socket: TCustomWinSocket; ErrorEvent: TErrorEvent;
    var ErrorCode: Integer);
begin
    MemoOperationInf.Lines.Add('-> ClientSocketToServer: Error');
    isConnectedToServer := False;
end;

procedure TFormProvider.ClientSocketToServerRead(Sender: TObject;
    Socket: TCustomWinSocket);
var
    ReceiveLen: LongInt;
begin
    // çok önemli tekrar bak:

    MemoOperationInf.Lines.Add('-> ClientSocketToServer: Read');
    ReceiveLen := Socket.ReceiveBuf(MessageReceive,SizeOf(MessageInfo));

    isTakeHello := True;
    isAuthenticate := True;
    if (ReceiveLen > 0) then begin
        MemoOperationInf.Lines.Add('-> Komut:      ' + IntToStr(MessageReceive.Command));
    end;

```

```

MemoOperationInf.Lines.Add('-> Gönderen: ' + IntToStr(MessageReceive.SenderID));
if (MessageReceive.DataExists = Present) then begin
  if (MessageReceive.DataTag.CustomerDataExists = Present) then begin
    MemoOperationInf.Lines.Add('->
'+ByteArrToStr(@MessageReceive.DataTag.CustomerData.CustName,16));
    MemoOperationInf.Lines.Add('->
'+ByteArrToStr(@MessageReceive.DataTag.CustomerData.CustSurname,16));
    MemoOperationInf.Lines.Add('->
'+ByteArrToStr(@MessageReceive.DataTag.CustomerData.CustBirthday,8));
    MemoOperationInf.Lines.Add('->
'+ByteArrToStr(@MessageReceive.DataTag.CustomerData.CustBirthPlace,16));
    MemoOperationInf.Lines.Add('->
'+ByteArrToStr(@MessageReceive.DataTag.CustomerData.CustFatherName,16));
    MemoOperationInf.Lines.Add('->
'+ByteArrToStr(@MessageReceive.DataTag.CustomerData.CustTaxNo,16));
    MemoOperationInf.Lines.Add('->
'+ByteArrToStr(@MessageReceive.DataTag.CustomerData.CustNationalID,16));
    MemoOperationInf.Lines.Add('->
'+ByteArrToStr(@MessageReceive.DataTag.CustomerData.CustHomeAdr,64));
    MemoOperationInf.Lines.Add('->
'+ByteArrToStr(@MessageReceive.DataTag.CustomerData.CustHomeTel,16));
    MemoOperationInf.Lines.Add('->
'+ByteArrToStr(@MessageReceive.DataTag.CustomerData.CustWorkAdr,64));
    MemoOperationInf.Lines.Add('->
'+ByteArrToStr(@MessageReceive.DataTag.CustomerData.CustWorkTel,16));
    MemoOperationInf.Lines.Add('->
'+ByteArrToStr(@MessageReceive.DataTag.CustomerData.CustMobileTel,16));
    MemoOperationInf.Lines.Add('->
'+ByteArrToStr(@MessageReceive.DataTag.CustomerData.CustEmail,64));
    MemoOperationInf.Lines.Add('->
Tar: '-ByteArrToStr(@MessageReceive.DataTag.CustomerData.CustDueDate,8));
    MemoOperationInf.Lines.Add('->
'+ByteArrToStr(@MessageReceive.DataTag.CustomerData.CustBalance,16));
  end;
  if (MessageReceive.DataTag.BalanceDataExists = Present) then begin
    MemoOperationInf.Lines.Add('->
'+ByteArrToStr(@MessageReceive.DataTag.BalanceData.OldBalance,16));
    MemoOperationInf.Lines.Add('->
'+ByteArrToStr(@MessageReceive.DataTag.BalanceData.NewBalance,16));
  end;
  if (MessageReceive.DataTag.CryptoDataExists = Present) then begin
    MemoOperationInf.Lines.Add('->
'+ByteArrToStr(@MessageReceive.DataTag.CryptoData.HashValue,32));
    MemoOperationInf.Lines.Add('->
Crypto: '+ByteArrToStr(@MessageReceive.DataTag.CryptoData.CryptoValue,32));
    MemoOperationInf.Lines.Add('->
'+IntToStr(MessageReceive.DataTag.CryptoData.ActionTime.Time));
    MemoOperationInf.Lines.Add('->
'+IntToStr(MessageReceive.DataTag.CryptoData.ActionTime.Date));
  end;
end;

isTakeHello := True;
case MessageReceive.Command of
  ResponseHello:
    isTakeHello := IsSenderValid(MessageReceive.SenderID);
  ResponseGoodbye:
    if isTakeHello and IsSenderValid(MessageReceive.SenderID) then begin
      ClientSocketToServer.Close;
      isTakeHello := False;
      isAuthenticate := False;
    end;
  ResponseAuthenticate:
    if isTakeHello and IsSenderValid(MessageReceive.SenderID) then
      AuthenticateCustomer;
  ResponseInsert:
    if isTakeHello and IsSenderValid(MessageReceive.SenderID) and
      isAuthenticate then
      InsertNewCustomer;
  ResponseUpdateInfo:
    if isTakeHello and IsSenderValid(MessageReceive.SenderID) and
      isAuthenticate then
      UpdateCustomerInfo;
  ResponseUpdateBlance:
    if isTakeHello and IsSenderValid(MessageReceive.SenderID) and
      isAuthenticate then
      UpdateCustomerBalance;

```

```

ResponseDelete:
    if isTakeHello and IsSenderValid(MessageReceive.SenderID) and
        isAuthenticate then
        DeleteCustomer;
    //SendBlacklist:    // do something
    //DeleteBlacklist:  // do something
end;
end;
end;

procedure TFormProvider.ClientSocketToServerWrite(Sender: TObject;
    Socket: TCustomWinSocket);
begin
    // no code need I suppose
    if (isConnectedToServer) then begin
        MemoOperationInf.Lines.Add('-> ClientSocketToServer: Write');
    end;
end;

procedure TFormProvider.SpeedButtonClearMemoClick(Sender: TObject);
begin
    MemoOperationInf.Lines.Clear;
end;

procedure TFormProvider.SpeedButtonConfirmLoadClick(Sender: TObject);
var
    HashStr: String;
begin
    if (isFilled) then begin
        FormChangePIN.Caption := 'PIN Girişi';
        FormChangePIN.EditOldPIN.Text := '0000';
        if (FormChangePIN.ShowModal = mrOk) then begin
            if (ChangePinOfSCard(FormChangePIN.EditOldPIN.Text,
                FormChangePIN.EditNewPIN.Text)) then begin
                MemoOperationInf.Lines.Add('-> Karta yeni PIN girişi başarı ile
                    gerçekleşti. ');
                MemoOperationInf.Lines.Add('-> Yeni PIN: ' + FormChangePIN.EditNewPIN.Text);
                HashManager1.CalcString(FormGetPIN.EditPIN.Text);
                HashStr := HashManager1.DigestString[16];
                PasswdHashStr := HashStr;
                MemoOperationInf.Lines.Add('-> h(PIN) : ' + HashStr);

                EditName.Enabled := False;
                EditSurname.Enabled := False;
                EditFatherName.Enabled := False;
                MaskEditBirthDay.Enabled := False;
                EditPlaceofBirth.Enabled := False;
                EditTaxNo.Enabled := False;
                EditIdNo.Enabled := False;
                EditHomeAddress.Enabled := False;
                MaskEditHomeTel.Enabled := False;
                EditWorkAddress.Enabled := False;
                MaskEditWorkTel.Enabled := False;
                EditEMail.Enabled := False;
                MaskEditMobileTel.Enabled := False;
                MaskEditBalance.Enabled := False;
                MaskEditDueDate.Enabled := False;

                ClearMessageSend;
                // MessageSend hazırla
                MessageSend.Command := NewCustomerInsert;
                MessageSend.SenderID := ClientID;
                MessageSend.DataExists := Present;

                MessageSend.DataTag.CustomerDataExists := Present;
                StrToByteArr16(EditName.Text, MessageSend.DataTag.CustomerData.CustName);

                StrToByteArr16(EditSurname.Text, MessageSend.DataTag.CustomerData.CustSurname);

                StrToByteArr10(MaskEditBirthDay.Text, MessageSend.DataTag.CustomerData.CustBirthDay);

                StrToByteArr16(EditPlaceofBirth.Text, MessageSend.DataTag.CustomerData.CustBirthPlace);
                ;

                StrToByteArr16(EditFatherName.Text, MessageSend.DataTag.CustomerData.CustFatherName);
            end;
        end;
    end;
end;

```

```

StrToByteArr16(EditTaxNo.Text,MessageSend.DataTag.CustomerData.CustTaxNo);
StrToByteArr16(EditIdNo.Text,MessageSend.DataTag.CustomerData.CustNationalID);
StrToByteArr64(EditHomeAddress.Text,MessageSend.DataTag.CustomerData.CustHomeAdr);
StrToByteArr16(MaskEditHomeTel.Text,MessageSend.DataTag.CustomerData.CustHomeTel);
StrToByteArr64(EditWorkAddress.Text,MessageSend.DataTag.CustomerData.CustWorkAdr);
StrToByteArr16(MaskEditWorkTel.Text,MessageSend.DataTag.CustomerData.CustWorkTel);
StrToByteArr32(EditEmail.Text,MessageSend.DataTag.CustomerData.CustEmail);
StrToByteArr16(MaskEditMobileTel.Text,MessageSend.DataTag.CustomerData.CustMobileTel);
;
StrToByteArr10(MaskEditDueDate.Text,MessageSend.DataTag.CustomerData.CustDueDate);
StrToByteArr16(MaskEditBalance.Text,MessageSend.DataTag.CustomerData.CustBalance);

//MessageSend.DataTag.CryptoDataExists := Present;
//StrToByteArr16(HashStr,MessageSend.DataTag.CryptoData.HashValue);

if (isConnectedToServer) then begin
  ClientSocketToServer.Socket.SendBuf(MessageSend,SizeOf(MessageInfo));
  MemoOperationInf.Lines.Add('-> Server''a yeni müşteri ve kart girme bilgisi
gönderildi.'');
end else if (ClientSocketToServer.Socket <> nil) then begin
  ClientSocketToServer.Socket.Open;
  if (isConnectedToServer) then begin
    ClientSocketToServer.Socket.SendBuf(MessageSend,SizeOf(MessageInfo));
    MemoOperationInf.Lines.Add('-> Server''a yeni müşteri ve kart girme
bilgisi gönderildi.'');
  end else
    MemoOperationInf.Lines.Add('-> Server''la bağlantı kurulamadı.'');
  end else
    MemoOperationInf.Lines.Add('-> Server''la bağlantı kurulamadı.'');

//Onay mesajı clientread'de alırsan aşağıdaki işlemler yapılacaktır
// yani yerleri burası değil
{} SpeedButtonConfirmLoad.Enabled := False;
{} SpeedButtonCancelLoad.Enabled := False;
{} SpeedButtonUpdateInfo.Enabled := True;
{} SpeedButtonCredit.Enabled := True;
{} SpeedButtonDebit.Enabled := True;
{} SpeedButtonChangePIN.Enabled := True;
{} SpeedButtonDeleteCustomer.Enabled := True;

{} MemoOperationInf.Lines.Add('-> Yeni kart yükleme ve müşteri girme işlemi
başarıyla gerçekleşti');
end;
end;
end;
end;

procedure TFormProvider.SpeedButtonCancelLoadClick(Sender: TObject);
begin
  MaskEditCardNo.Text := '';
  MaskEditBalance.Text := '';
  EditName.Text := '';
  EditSurname.Text := '';
  EditFatherName.Text := '';
  MaskEditBirthDay.Text := '';
  EditPlaceofBirth.Text := '';
  EditTaxNo.Text := '';
  EditIdNo.Text := '';
  EditHomeAddress.Text := '';
  MaskEditHomeTel.Text := '';
  EditWorkAddress.Text := '';
  MaskEditWorkTel.Text := '';
  EditEmail.Text := '';
  MaskEditMobileTel.Text := '';
  MaskEditDueDate.Text := '';

  EditName.Enabled := False;
  EditSurname.Enabled := False;
  EditFatherName.Enabled := False;

```

```
MaskEditBirthday.Enabled := False;
EditPlaceofBirth.Enabled := False;
EditTaxNo.Enabled := False;
EditIdNo.Enabled := False;
EditHomeAddress.Enabled := False;
MaskEditHomeTel.Enabled := False;
EditWorkAddress.Enabled := False;
MaskEditWorkTel.Enabled := False;
EditEMail.Enabled := False;
MaskEditMobileTel.Enabled := False;
MaskEditBalance.Enabled := False;
MaskEditDueDate.Enabled := False;
```

```
SpeedButtonLoad.Enabled := True;
SpeedButtonConfirmLoad.Enabled := False;
SpeedButtonCancelLoad.Enabled := False;
```

```
MemoOperationInf.Lines.Add('-> Yeni kart yükleme ve müşteri girme işlemi iptal edildi');
end;
```

```
procedure TFormProvider.SpeedButtonConfirmUpdateInfoClick(Sender: TObject);
begin
```

```
  if (isFilled) then begin
    EditHomeAddress.Enabled := False;
    MaskEditHomeTel.Enabled := False;
    EditWorkAddress.Enabled := False;
    MaskEditWorkTel.Enabled := False;
    EditEMail.Enabled := False;
    MaskEditMobileTel.Enabled := False;
```

```
    //Onay mesajı clientread'de alırsan yukarıdaki işlemler yapılacak
    // yani yerleri burası değil
    SpeedButtonUpdateInfo.Enabled := True;
    SpeedButtonConfirmUpdateInfo.Enabled := False;
    SpeedButtonCancelUpdateInfo.Enabled := False;
    SpeedButtonCredit.Enabled := True;
    SpeedButtonDebit.Enabled := True;
    SpeedButtonChangePIN.Enabled := True;
    SpeedButtonDeleteCustomer.Enabled := True;
```

```
  if (isChanged) then begin
    ClearMessageSend;
    // MessageSend hazırla
    MessageSend.Command := CustomerInfoUpdate;
    MessageSend.SenderID := ClientID;
    MessageSend.DataExists := Present;
```

```
    MessageSend.DataTag.CustomerDataExists := Present;
    StrToByteArr16(EditName.Text, MessageSend.DataTag.CustomerData.CustName);
    StrToByteArr16(EditSurname.Text, MessageSend.DataTag.CustomerData.CustSurname);
```

```
    StrToByteArr10(MaskEditBirthday.Text, MessageSend.DataTag.CustomerData.CustBirthday);
```

```
    StrToByteArr16(EditPlaceofBirth.Text, MessageSend.DataTag.CustomerData.CustBirthPlace);
  ;
```

```
    StrToByteArr16(EditFatherName.Text, MessageSend.DataTag.CustomerData.CustFatherName);
    StrToByteArr16(EditTaxNo.Text, MessageSend.DataTag.CustomerData.CustTaxNo);
    StrToByteArr16(EditIdNo.Text, MessageSend.DataTag.CustomerData.CustNationalID);
```

```
    StrToByteArr64(EditHomeAddress.Text, MessageSend.DataTag.CustomerData.CustHomeAdr);
```

```
    StrToByteArr16(MaskEditHomeTel.Text, MessageSend.DataTag.CustomerData.CustHomeTel);
```

```
    StrToByteArr64(EditWorkAddress.Text, MessageSend.DataTag.CustomerData.CustWorkAdr);
```

```
    StrToByteArr16(MaskEditWorkTel.Text, MessageSend.DataTag.CustomerData.CustWorkTel);
    StrToByteArr32(EditEMail.Text, MessageSend.DataTag.CustomerData.CustEmail);
```

```
    StrToByteArr16(MaskEditMobileTel.Text, MessageSend.DataTag.CustomerData.CustMobileTel);
  ;
```

```
    StrToByteArr10(MaskEditDueDate.Text, MessageSend.DataTag.CustomerData.CustDueDate);
```

```
    StrToByteArr16(MaskEditBalance.Text, MessageSend.DataTag.CustomerData.CustBalance);
```

```

    if (isConnectedToServer) then begin
        ClientSocketToServer.Socket.SendBuf(MessageSend, SizeOf(MessageInfo));
        MemoOperationInf.Lines.Add('-> Server''a müşteri bilgileri güncelleme isteği
gönderildi.');
```

end else if (ClientSocketToServer.Socket <> nil) then begin
 ClientSocketToServer.Open;
 if (isConnectedToServer) then begin
 ClientSocketToServer.Socket.SendBuf(MessageSend, SizeOf(MessageInfo));
 MemoOperationInf.Lines.Add('-> Server''a müşteri bilgileri güncelleme
isteği gönderildi.');

end else
 MemoOperationInf.Lines.Add('-> Server''la bağlantı kurulamadı.');

end else
 MemoOperationInf.Lines.Add('-> Server''la bağlantı kurulamadı.');

end;
end;
end;

```

function TFormProvider.isFilled: Boolean;
begin
    isFilled := False;
    if (EditName.Text = '') then begin
        Application.MessageBox("Ad" kısmı boş bırakılmış, tekrar deneyiniz.',
                                'Dikkat !', MB_OK);
        EditName.SetFocus;
    end else if (EditSurname.Text = '') then begin
        Application.MessageBox("Soyad" kısmı boş bırakılmış, tekrar deneyiniz.',
                                'Dikkat !', MB_OK);
        EditSurname.SetFocus;
    end else if ((MaskEditBirthDay.Text = ' . . ') or
        (MaskEditBirthDay.Text = ' / / ') or
        (MaskEditBirthDay.Text = '')) or
        (MaskEditBirthDay.Text = ' - - ')) then begin
        Application.MessageBox("Doğum Tarihi" kısmı boş bırakılmış, tekrar
deneyiniz.',
                                'Dikkat !', MB_OK);
        MaskEditBirthDay.SetFocus;
    end else if (EditPlaceofBirth.Text = '') then begin
        Application.MessageBox("Doğum Yeri" kısmı boş bırakılmış, tekrar deneyiniz.',
                                'Dikkat !', MB_OK);
        EditPlaceofBirth.SetFocus;
    end else if (EditFatherName.Text = '') then begin
        Application.MessageBox("Baba Adı" kısmı boş bırakılmış, tekrar deneyiniz.',
                                'Dikkat !', MB_OK);
        EditFatherName.SetFocus;
    end else if (EditTaxNo.Text = '') then begin
        Application.MessageBox("Vergi No" kısmı boş bırakılmış, tekrar deneyiniz.',
                                'Dikkat !', MB_OK);
        EditTaxNo.SetFocus;
    end else if (EditIdNo.Text = '') then begin
        Application.MessageBox("Nüfus No" kısmı boş bırakılmış, tekrar deneyiniz.',
                                'Dikkat !', MB_OK);
        EditIdNo.SetFocus;
    end else if ((MaskEditHomeTel.Text = '( ) - ') and
        (MaskEditWorkTel.Text = '( ) - ') and
        (MaskEditMobileTel.Text = '( ) - ')) then begin
        Application.MessageBox("Ev, İş veya Cep" telefonlarından en az birisi
girilmelidir, tekrar deneyiniz.',
                                'Dikkat !', MB_OK);
        MaskEditHomeTel.SetFocus;
    end else if ((EditHomeAddress.Text = '') and (EditWorkAddress.Text = '')) then
begin
        Application.MessageBox("Ev veya İş" adreslerinden en az birisi girilmelidir,
tekrar deneyiniz.',
                                'Dikkat !', MB_OK);
        EditHomeAddress.SetFocus;
    end else
        isFilled := True;
end;
```

```

function TFormProvider.isChanged: Boolean;
begin
    isChanged := False;
    if (EditHomeAddress.Modified) then isChanged:= True;
    if (MaskEditHomeTel.Modified) then isChanged:= True;
    if (EditWorkAddress.Modified) then isChanged:= True;
```

```

if (MaskEditWorkTel.Modified) then isChanged:= True;
if (EditEMail.Modified) then isChanged:= True;
if (MaskEditMobileTel.Modified) then isChanged:= True;
end;

```

```

procedure TFormProvider.SpeedButtonCancelUpdateInfoClick(Sender: TObject);
begin

```

```

    EditHomeAddress.Undo;
    MaskEditHomeTel.Undo;
    EditWorkAddress.Undo;
    MaskEditWorkTel.Undo;
    EditEMail.Undo;
    MaskEditMobileTel.Undo;

```

```

    EditHomeAddress.Enabled := False;
    MaskEditHomeTel.Enabled := False;
    EditWorkAddress.Enabled := False;
    MaskEditWorkTel.Enabled := False;
    EditEMail.Enabled := False;
    MaskEditMobileTel.Enabled := False;

```

```

    SpeedButtonUpdateInfo.Enabled := True;
    SpeedButtonConfirmUpdateInfo.Enabled := False;
    SpeedButtonCancelUpdateInfo.Enabled := False;
    SpeedButtonCredit.Enabled := True;
    SpeedButtonDebit.Enabled := True;
    SpeedButtonChangePIN.Enabled := True;
    SpeedButtonDeleteCustomer.Enabled := True;
end;

```

```

procedure TFormProvider.SpeedButtonCreditClick(Sender: TObject);

```

```

var

```

```

    NewBal: String;

```

```

begin

```

```

    StatusBarSQL.SimpleText := 'Hesap ekleme işlemi';
    FormPassword.Caption := strUserID + ', hesap ekleme için şifre giriniz!';
    FormPassword.ShowModal;
    if (FormPassword.ModalResult = mrOk) then begin

```

```

        FormUpdateBalance.Caption := 'Hesap Ekleme';
        FormUpdateBalance.LabelCreditDebit.Caption := 'Eklenecek tutar:';
        FormUpdateBalance.MaskEditCardNo.Text := MaskEditCardNo.Text;
        FormUpdateBalance.EditName.Text := EditName.Text;
        FormUpdateBalance.EditSurname.Text := EditSurname.Text;

```

```

        FormUpdateBalance.MaskEditBalance.Text := Trim(MaskEditBalance.Text);
        FormUpdateBalance.MaskEditCreditDebitValue.Text := '';
        FormUpdateBalance.ShowModal;

```

```

        if (FormUpdateBalance.ModalResult = mrOk) then begin
            if ((StrToInt(Trim(MaskEditBalance.Text))) +
StrToInt(Trim(FormUpdateBalance.MaskEditCreditDebitValue.Text))) < 999999999) then
begin

```

```

                ClearMessageSend;
                // MessageSend hazırla
                MessageSend.Command := BalanceUpdate;
                MessageSend.SenderID := ClientID;
                MessageSend.DataExists := Present;

```

```

                MessageSend.DataTag.CustomerDataExists := Present;
                StrToByteArr16(EditName.Text, MessageSend.DataTag.CustomerData.CustName);

```

```

                StrToByteArr16(EditSurname.Text, MessageSend.DataTag.CustomerData.CustSurname);

```

```

                StrToByteArr10(MaskEditBirthday.Text, MessageSend.DataTag.CustomerData.CustBirthday);

```

```

                StrToByteArr16(EditPlaceofBirth.Text, MessageSend.DataTag.CustomerData.CustBirthPlace);

```

```

                StrToByteArr16(EditFatherName.Text, MessageSend.DataTag.CustomerData.CustFatherName);
                StrToByteArr16(EditTaxNo.Text, MessageSend.DataTag.CustomerData.CustTaxNo);

```

```

                StrToByteArr16(EditIdNo.Text, MessageSend.DataTag.CustomerData.CustNationalID);

```

```

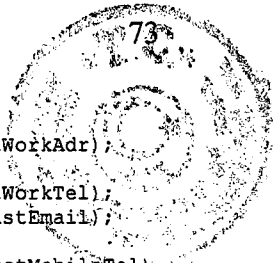
                StrToByteArr64(EditHomeAddress.Text, MessageSend.DataTag.CustomerData.CustHomeAdr);

```

```

                StrToByteArr16(MaskEditHomeTel.Text, MessageSend.DataTag.CustomerData.CustHomeTel);

```



```

StrToByteArr64(EditWorkAddress.Text,MessageSend.DataTag.CustomerData.CustWorkAdr);
StrToByteArr16(MaskEditWorkTel.Text,MessageSend.DataTag.CustomerData.CustWorkTel);
StrToByteArr32(EditEmail.Text,MessageSend.DataTag.CustomerData.CustEmail);

StrToByteArr16(MaskEditMobileTel.Text,MessageSend.DataTag.CustomerData.CustMobileTel);
;

StrToByteArr10(MaskEditDueDate.Text,MessageSend.DataTag.CustomerData.CustDueDate);

NewBal      :=      IntToStr(StrToInt(Trim(MaskEditBalance.Text))      +
StrToInt(Trim(FormUpdateBalance.MaskEditCreditDebitValue.Text)));
MessageSend.DataTag.BalanceDataExists := Present;

StrToByteArr16(MaskEditBalance.Text,MessageSend.DataTag.BalanceData.OldBalance);
MaskEditBalance.Text := NewBal;

StrToByteArr16(MaskEditBalance.Text,MessageSend.DataTag.BalanceData.NewBalance);

if (isConnectedToServer) then begin
  ClientSocketToServer.Socket.SendBuf(MessageSend,SizeOf(MessageInfo));
  MemoOperationInf.Lines.Add('-> Server'a müşteri hesabına ekleme isteği
gönderildi.');
```

```

end else if (ClientSocketToServer.Socket <> nil) then begin
  ClientSocketToServer.Open;
  if (isConnectedToServer) then begin
    ClientSocketToServer.Socket.SendBuf(MessageSend,SizeOf(MessageInfo));
    MemoOperationInf.Lines.Add('-> Server'a müşteri hesabına ekleme isteği
gönderildi.');
```

```

  end else
    MemoOperationInf.Lines.Add('-> Server''la bağlantı kurulamadı.');
```

```

end else
  MemoOperationInf.Lines.Add('-> Server''la bağlantı kurulamadı.');
```

```

end else
  MemoOperationInf.Lines.Add('-> HATA! En büyük miktar 999,999,999 olmalı.');
```

```

end;
end;
end;

procedure TFormProvider.SpeedButtonDebitClick(Sender: TObject);
var
  NewBal: String;
begin
  StatusBarSQL.SimpleText := 'Hesap çıkarma işlemi';
  FormPassword.Caption := strUserID + ', hesap çıkarma için şifre giriniz!';
  FormPassword.ShowModal;
  if (FormPassword.ModalResult = mrOk) then begin

    FormUpdateBalance.Caption := 'Hesap Çıkarma';
    FormUpdateBalance.LabelCreditDebit.Caption := 'Çıkarılacak tutar:';
    FormUpdateBalance.MaskEditCardNo.Text := MaskEditCardNo.Text;
    FormUpdateBalance.EditName.Text := EditName.Text;
    FormUpdateBalance.EditSurname.Text := EditSurname.Text;
    FormUpdateBalance.MaskEditBalance.Text := MaskEditBalance.Text;
    FormUpdateBalance.MaskEditCreditDebitValue.Text := '';
    FormUpdateBalance.ShowModal();

    if (FormUpdateBalance.ModalResult = mrOk) then begin
      if
        ((StrToInt(MaskEditBalance.Text)
StrToInt(FormUpdateBalance.MaskEditCreditDebitValue.Text)) >= 0) then begin
        ClearMessageSend;
        // MessageSend hazırla
        MessageSend.Command := BalanceUpdate;
        MessageSend.SenderID := ClientID;
        MessageSend.DataExists := Present;

        MessageSend.DataTag.CustomerDataExists := Present;
        StrToByteArr16(EditName.Text,MessageSend.DataTag.CustomerData.CustName);

        StrToByteArr16(EditSurname.Text,MessageSend.DataTag.CustomerData.CustSurname);

        StrToByteArr10(MaskEditBirthday.Text,MessageSend.DataTag.CustomerData.CustBirthday);

```



```
StrToByteArr16(EditPlaceOfBirth.Text,MessageSend.DataTag.CustomerData.CustBirthPlace);
;

StrToByteArr16(EditFatherName.Text,MessageSend.DataTag.CustomerData.CustFatherName);
StrToByteArr16(EditTaxNo.Text,MessageSend.DataTag.CustomerData.CustTaxNo);

StrToByteArr16(EditIdNo.Text,MessageSend.DataTag.CustomerData.CustNationalID);

StrToByteArr64(EditHomeAddress.Text,MessageSend.DataTag.CustomerData.CustHomeAdr);

StrToByteArr16(MaskEditHomeTel.Text,MessageSend.DataTag.CustomerData.CustHomeTel);

StrToByteArr64(EditWorkAddress.Text,MessageSend.DataTag.CustomerData.CustWorkAdr);

StrToByteArr16(MaskEditWorkTel.Text,MessageSend.DataTag.CustomerData.CustWorkTel);
StrToByteArr32(EditEmail.Text,MessageSend.DataTag.CustomerData.CustEmail);

StrToByteArr16(MaskEditMobileTel.Text,MessageSend.DataTag.CustomerData.CustMobileTel);
;

StrToByteArr10(MaskEditDueDate.Text,MessageSend.DataTag.CustomerData.CustDueDate);

NewBal := IntToStr(StrToInt(MaskEditBalance.Text) -
StrToInt(FormUpdateBalance.MaskEditCreditDebitValue.Text));
MessageSend.DataTag.BalanceDataExists := Present;

StrToByteArr16(MaskEditBalance.Text,MessageSend.DataTag.BalanceData.OldBalance);
MaskEditBalance.Text := NewBal;

StrToByteArr16(MaskEditBalance.Text,MessageSend.DataTag.BalanceData.NewBalance);

if (isConnectedToServer) then begin
ClientSocketToServer.Socket.SendBuf(MessageSend,SizeOf(MessageInfo));
MemoOperationInf.Lines.Add('-> Server''a müşteri hesabı çıkarma isteği
gonderildi.'');
end else if (ClientSocketToServer.Socket <> nil) then begin
ClientSocketToServer.Open;
if (isConnectedToServer) then begin
ClientSocketToServer.Socket.SendBuf(MessageSend,SizeOf(MessageInfo));
MemoOperationInf.Lines.Add('-> Server''a müşteri hesabı çıkarma isteği
gönderildi.'');
end else
MemoOperationInf.Lines.Add('-> Server''la bağlantı kurulamadı.'');
end else
MemoOperationInf.Lines.Add('-> Server''la bağlantı kurulamadı.'');

end else
MemoOperationInf.Lines.Add('-> HATA! Miktar 0''dan küçük olmamalı.'');

end;
end;
end;

procedure TFormProvider.ClearMessageSend;
begin
MessageSend.Command := Absent;
MessageSend.SenderID := Absent;
MessageSend.DataExists := Absent;
MessageSend.DataTag.CustomerDataExists := Absent;
MessageSend.DataTag.CustomerData.CustID := ZeroArr16;
MessageSend.DataTag.CustomerData.CustName := ZeroArr16;
MessageSend.DataTag.CustomerData.CustSurname := ZeroArr16;
MessageSend.DataTag.CustomerData.CustBirthDay := ZeroArr10;
MessageSend.DataTag.CustomerData.CustBirthPlace := ZeroArr16;
MessageSend.DataTag.CustomerData.CustFatherName := ZeroArr16;
MessageSend.DataTag.CustomerData.CustTaxNo := ZeroArr16;
MessageSend.DataTag.CustomerData.CustNationalID := ZeroArr16;
MessageSend.DataTag.CustomerData.CustHomeAdr := ZeroArr64;
MessageSend.DataTag.CustomerData.CustHomeTel := ZeroArr16;
MessageSend.DataTag.CustomerData.CustWorkAdr := ZeroArr64;
MessageSend.DataTag.CustomerData.CustWorkTel := ZeroArr16;
MessageSend.DataTag.CustomerData.CustMobileTel := ZeroArr16;
MessageSend.DataTag.CustomerData.CustEmail := ZeroArr32;
MessageSend.DataTag.CustomerData.CustDueDate := ZeroArr10;
MessageSend.DataTag.CustomerData.CustBalance := ZeroArr16;
MessageSend.DataTag.CryptoDataExists := Absent;
```



```
MessageSend.DataTag.CryptoData.HashValue := ZeroArr32;
MessageSend.DataTag.CryptoData.CryptoValue := ZeroArr32;
MessageSend.DataTag.CryptoData.ActionTime.Time := 0;
MessageSend.DataTag.CryptoData.ActionTime.Date := 0;
MessageSend.DataTag.BalanceDataExists := Absent;
MessageSend.DataTag.BalanceData.OldBalance := ZeroArr16;
MessageSend.DataTag.BalanceData.NewBalance := ZeroArr16;
end;

function TFormProvider.IsSenderValid(SenderIDNo: Byte): Boolean;
begin
  case SenderIDNo of
    255: Result := True;
    else
      Result := False;
  end;
end;

function TFormProvider.AuthenticateCustomer: Boolean;
var
  HashStr: String;
begin
  if (FormGetPIN.ShowModal = mrOk) then begin
    PIN := FormGetPIN.EditPIN.Text;
    if (VerifyPinOfSCard) then begin
      HashManager1.CalcString(PIN);
      HashStr := HashManager1.DigestString[16];
      //if (HashStr = DataASCII) then
        Result := True
      //else
        //Result := False;
      end else
        Result := False;
    end else
      Result := False;
  end;
end;

procedure TFormProvider.InsertNewCustomer;
var
  HashByte: ByteArr32;
  HashStr: String;
  Cnt: Integer;
  StrTemp: String;
begin
  if (WriteDataToSCard('00','32','1')) then begin
    MemoOperationInf.Lines.Add('-> Karta "00" yazıldı');
    for Cnt:=1 to 32 do begin
      HashByte[Cnt] := Ord(PasswdHashStr[Cnt]) xor
MessageReceive.DataTag.CryptoData.CryptoValue[Cnt];
    end;
    HashStr := ByteArrToStr(@HashByte,32);
    if (WriteDataToSCard(PasswdHashStr,'33','32')) then begin
      MemoOperationInf.Lines.Add('-> Karta yazılan hash fonk.: ' + PasswdHashStr);

      StrTemp := EditName.Text;
      for Cnt:=Length(EditName.Text) to 16 do
        StrTemp := StrTemp + ' ';
      if (WriteDataToSCard(StrTemp,'65','16')) then
        MemoOperationInf.Lines.Add('-> Ad: ' + StrTemp);

      StrTemp := EditSurname.Text;
      for Cnt:=Length(EditSurname.Text) to 16 do
        StrTemp := StrTemp + ' ';
      if (WriteDataToSCard(StrTemp,'81','16')) then
        MemoOperationInf.Lines.Add('-> Soyad: ' + StrTemp);

      StrTemp := MaskEditBirthday.Text;
      for Cnt:=Length(MaskEditBirthday.Text) to 10 do
        StrTemp := StrTemp + ' ';
      if (WriteDataToSCard(StrTemp,'97','10')) then
        MemoOperationInf.Lines.Add('-> Doğumgünü: ' + StrTemp);

      StrTemp := EditPlaceofBirth.Text;
      for Cnt:=Length(EditPlaceofBirth.Text) to 16 do
        StrTemp := StrTemp + ' ';
      if (WriteDataToSCard(StrTemp,'107','16')) then
        MemoOperationInf.Lines.Add('-> Doğumyeri: ' + StrTemp);
```

```

StrTemp := EditFatherName.Text;
for Cnt:=Length(EditFatherName.Text) to 16 do
  StrTemp := StrTemp + ' ';
if (WriteDataToSCard(StrTemp,'123','16')) then
  MemoOperationInf.Lines.Add('-> Baba adı:      '+StrTemp);

StrTemp := EditTaxNo.Text;
for Cnt:=Length(EditTaxNo.Text) to 16 do
  StrTemp := StrTemp + ' ';
if (WriteDataToSCard(StrTemp,'139','16')) then
  MemoOperationInf.Lines.Add('-> Vergi no:      '+StrTemp);

StrTemp := EditIdNo.Text;
for Cnt:=Length(EditIdNo.Text) to 16 do
  StrTemp := StrTemp + ' ';
if (WriteDataToSCard(StrTemp,'155','16')) then
  MemoOperationInf.Lines.Add('-> Kimlik no:      '+StrTemp);

StrTemp := EditHomeAddress.Text;
for Cnt:=Length(EditHomeAddress.Text) to 64 do
  StrTemp := StrTemp + ' ';
if (WriteDataToSCard(StrTemp,'171','64')) then
  MemoOperationInf.Lines.Add('-> Ev adresi:      '+StrTemp);

StrTemp := MaskEditHomeTel.Text;
for Cnt:=Length(MaskEditHomeTel.Text) to 16 do
  StrTemp := StrTemp + ' ';
if (WriteDataToSCard(StrTemp,'235','16')) then
  MemoOperationInf.Lines.Add('-> Ev tel:          '+StrTemp);

StrTemp := EditWorkAddress.Text;
for Cnt:=Length(EditWorkAddress.Text) to 64 do
  StrTemp := StrTemp + ' ';
if (WriteDataToSCard(StrTemp,'251','64')) then
  MemoOperationInf.Lines.Add('-> İş adresi:      '+StrTemp);

StrTemp := MaskEditWorkTel.Text;
for Cnt:=Length(MaskEditWorkTel.Text) to 16 do
  StrTemp := StrTemp + ' ';
if (WriteDataToSCard(StrTemp,'315','16')) then
  MemoOperationInf.Lines.Add('-> İş tel:          '+StrTemp);

StrTemp := MaskEditMobileTel.Text;
for Cnt:=Length(MaskEditMobileTel.Text) to 16 do
  StrTemp := StrTemp + ' ';
if (WriteDataToSCard(StrTemp,'331','16')) then
  MemoOperationInf.Lines.Add('-> Cep tel:        '+StrTemp);

StrTemp := EditEmail.Text;
for Cnt:=Length(EditEmail.Text) to 32 do
  StrTemp := StrTemp + ' ';
if (WriteDataToSCard(StrTemp,'347','32')) then
  MemoOperationInf.Lines.Add('-> E-mail:          '+StrTemp);

StrTemp := MaskEditDueDate.Text;
for Cnt:=Length(MaskEditDueDate.Text) to 10 do
  StrTemp := StrTemp + ' ';
if (WriteDataToSCard(StrTemp,'379','10')) then
  MemoOperationInf.Lines.Add('-> Son K. Tar:      '+StrTemp);

StrTemp := MaskEditBalance.Text;
for Cnt:=Length(MaskEditBalance.Text) to 16 do
  StrTemp := StrTemp + ' ';
if (WriteDataToSCard(StrTemp,'389','16')) then
  MemoOperationInf.Lines.Add('-> Hesap:          '+StrTemp);
end;
end;

end;

procedure TFormProvider.UpdateCustomerInfo;
var
  StrTemp: String;
begin
  StrTemp := ByteArrToStr(@MessageReceive.DataTag.CustomerData.CustHomeAdr, 64);

```

[illegible]



```
        MemoOperationInf.Lines.Add('-> Kart başarıyla silindi.');
```

```
    end;
```

```
end else
```

```
    MemoOperationInf.Lines.Add('-> Kart PIN'i hatalı');
```

```
end;
```

```
end;
```

```
procedure TFormProvider.SendHello;
```

```
begin
```

```
    MessageSend.Command      := ClientHello;
```

```
    MessageSend.SenderID     := ClientID;
```

```
    MessageSend.DataExists   := Absent;
```

```
    if (isConnectedToServer) then begin
```

```
        ClientSocketToServer.Socket.SendBuf(MessageSend, SizeOf(MessageInfo));
```

```
        MemoOperationInf.Lines.Add('-> Server''a "Merhaba!" denildi.');
```

```
    end else if (ClientSocketToServer.Socket <> nil) then begin
```

```
        ClientSocketToServer.Open;
```

```
        if (isConnectedToServer) then begin
```

```
            ClientSocketToServer.Socket.SendBuf(MessageSend, SizeOf(MessageInfo));
```

```
            MemoOperationInf.Lines.Add('-> Server''a "Merhaba!" denildi.');
```

```
        end else
```

```
            MemoOperationInf.Lines.Add('-> Server''la bağlantı kurulamadı.');
```

```
    end else
```

```
        MemoOperationInf.Lines.Add('-> Server''la bağlantı kurulamadı.');
```

```
end;
```

```
end.
```

