İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

BİR SAĞLIK SİSTEMİNDE AKILLI KART KULLANARAK GÜVENLİK PROTOKOLÜ TASARIMI VE GERÇEKLENMESİ

YÜKSEK LİSANS TEZİ
Müh. Çiçek ÇAVDAR
(504981178)

Tezin Enstitüye Verildiği Tarih : 9 Ocak 2002
Tezin Savunulduğu Tarih : 17 Ocak 2002

Tez Danışmanı : Doç.Dr.B.Tevfik AKGÜN
Diğer Jüri Üyeleri Prof.Dr. Ümit AYGÖLÜ (İTÜ)
Yrd.Doç.Dr. Feza BUZLUCA (İTÜ)

OCAK 2002

ÖNSÖZ

Tez çalışması süresince çalışma azmimi körükleyen, dostlukları ve yardımlarıyla bana sonsuz enerji veren Hakan ve Oya'ya,

Yüksek Lisans öğrenciliğim boyunca bana her konuda destek ve yardımlarını esirgemeyen Sakine Arslan, Fatma Tutar ve tüm Fen Bilimleri Enstitüsü emekçilerine,

Tez çalışması süresince bana yol gösteren, önerileri ve görüşleri ile tezin oluşmasında katkılarını esirgemeyen, tez danışmanım Sayın Doç.Dr.B.Tevfik Akgün'e, katkıları için; tez çalışmasından bir ulusal bir de uluslararası bildiri çıkarmamı sağlayarak beni bilimsel çalışmalara teşvik ettiği için,

teşekkür ederim.

Ocak 2002 Çiçek Çavdar

İÇİNDEKİLER

KISALTMALAR

KDP	: Anahtar Türetme Parametreleri
CRC	: Cyclical Redundancy Check
ID	: Tanımlayıcı Kod
D_ID	: Doktorun Tanımlayıcı Kodu
H_ID	: Hastanın Tanımlayıcı Kodu
D_AA	: Doktorun Açık Anahtarı
D_GA	: Doktorun Gizli Anahtarı
H_AA	: Hastanın Açık Anahtarı
H_GA	: Hastanın Gizli Anahtarı
TTP	: Güvenilir Üçüncü Taraf -Thrusted Third Party
PIN	: Kişisel Tanımlama Numarası-Personal Identification Number

TABLO LİSTESİ

ŞEKİL LİSTESİ

ÖZET

Bu çalışmada bir sağlık sisteminde akıllı kart kullanarak güvenlik protokolü tasarımı yapılmış; tasarım bir pilot sistem üzerinde gerçeklenmiştir. Güvenlik protokolünde, kullanıcı akıllı kartları, istemci ve sunucu uygulama arasındaki veri akışının güvenli bir biçimde sağlanması amaçlanmıştır. Bunun için sistem bileşenlerinin birbirlerini asıllaması gerekmektedir. Hem asıllama hem de güvenli kanal oluşturulması için sistem bileşenleri kriptografik anahtarlar ve protokollerden yararlanırlar. Kullanıcıların gizli anahtarlarının, akıllı kartlar üzerinde üretilip saklanması, sistem içinde gizli anahtarların güvenliği konusunda sunulan çözümlerden birisidir.

Güvenlik protokolü altı taraftan oluşmaktadır: Akıllı kart kullanıcıları, doktor ve hasta; onların akıllı kartları, sunucu uygulama ve istemci uygulama. Doktor akıllı kartını kullanarak hasta kartındaki ve internet yoluyla sunucudaki hasta bilgilerine erişebilir ve hem hasta kartındaki hem de sunucudaki hasta verilerini güncelleyebilir. Bu çalışmada kimlik tanıma, asıllama, veri bütünlüğü gibi güvenlik problemleri tartışılarak açık anahtar ve simetrik anahtar şifreleme algoritmalarının birlikte kullanıldığı çözümler geliştirilmiştir.

Güvenlik protokolünün aktif elemanları doktor kartı, hasta kartı, istemci uygulama ve sunucu uygulamadır. Doktor, sunucuya internet üzerinden erişim sağlar. Bunun için doktor kartının ve hasta kartının varlıklarının sunucu tarafından asıllanması gerekir. Bu işlem sunucunun internet üzerinden iki kullanıcıyı birlikte asıllamasını sağlayan ikili asıllama protokolü ile gerçekleştirilir. Doktorun hasta kartı içindeki gizli bilgilere erişimi ise hasta kartının doktoru asıllamasını gerektirmektedir. Bu problemin çözümü için iki akıllı kart arasındaki asıllamayı gerçekleştirecek bir kimlik tanıma protokolü tasarlanmıştır.

Taraflar arasındaki veri akışının güvenliği hareketin her adımı için tarafların güvenilirliği ve yeteneklerine bağlı olarak ayrı ayrı ele alınmıştır. Tasarlanan güvenlik protokolünü oluşturan parçalar şu şekilde sıralanabilir: İki akıllı kart arasındaki asıllama protokolü(hasta kartı tarafından doktorun asıllanması), sunucunun doktor ve hastayı akıllı kartları aracılığıyla senkron olarak asılladığı ikili asıllama protokolü, istemci-sunucu ve sunucu-hasta kartı arasında güvenli kanal oluşturulması ve hasta kartının internet üzerinden sunucuyu asıllaması.

SECURITY PROTOCOL DESIGN AND APPLICATION USING SMART CARDS IN A HEALTH CARE SYSTEM

SUMMARY

Smart card based solutions to medicine and informatics have become prevalent in many countries in the world. The usage of smart cards in health care systems provide critical health data such as emergency data and chronical diseases of the patient to be stored in a portable, safe device. The goal of healthcard systems are to improve healthcare for people by having accurate, complete records from doctors and public health programs stored in an easy to use secure smart card. Other goals include reduced healthcare costs and to promote access to preventative health information. As more medical information is electronically stored and accessible from widely available networks, techniques to increase healthcare data privacy, confidentiality and security gained importance

In this study a security protocol is designed and developed using smart cards in a health care system. The task of the security protocol is to perform secure data transaction between users' chip cards, client and server application. System components must authenticate each other for secure transaction. Cryiptographic keys and protocols are used to perform entity authentication and to construct secure channel between the entities of the protocol. The usage of smart cards serve also for the security of secret keys except for mobile storage of health data. Secret keys of the users are generated and stored securely in the smart cards.

The security protocol involves six entities: Chipcard holders, doctor, patient, their chipcards, server application and client application. (See Fig.1) Doctor, using his smart card, reaches the private data in the patient card and through internet in the server. It can update the patient data both in the patient card and server. In this study security problems like identification, authentication and data integrity are discussed and solutions are developed using both public key and secret key cryptographic algorithms together.

Active members of the security protocol are doctor chipcard, patient chipcard, client application and server application. The server can be reached by the doctor through the internet. Doctor needs his card and patient card to be authenticated by the server to access the patient data on the server. Server authenticates through the internet both users together via the multi authentication protocol. Doctor's access to the secret information in the patient card requires authentication of doctor by the patient card. As a solution of this problem identification protocol is designed that fulfills the task of authentication among two smart cards.

The security of the data transaction between entities is examined for each step of transaction one by one considering the confidentiality and abilities of each entity

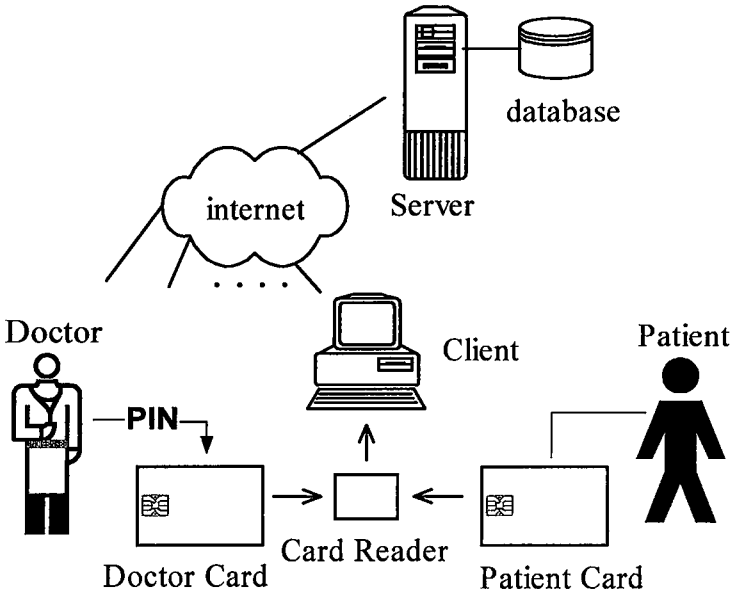


Figure 1. Elements of the system

related to that transaction. Designed security protocol is conformed of the parts below: Authentication protocol among two smart cards (doctor's authentication by patient card), multi authentication protocol by which server authenticates doctor and patient synchronously via their chipcards, construction of secure channels between client–server and server–patient chipcard, and server's authentication by the patient card through the internet. All the steps of the security protocol must be executed one after the other; each step needs the previous ones to be succeded before start to execute.

The protocol design covers authentication, data integrity, confidentiality, and nonrepudiation. Other problems of the health care system such as database management, health data organisation are out of this thesis' scope.

Health data access

Health data which will be needed by the doctor in the first hand in the process of a medical threatment and in emergency time is stored in the patient card. The history of the patient's medical treatment and all the medical test results are stored in a database in the server. The first part of the health data can be reached by the doctor depending on the conditions: the existence of a patient card, doctor's authentication by doctor card and the verification of the validity of both cards by the client programme. Doctor's access to the second part of the health data stored in the database needs authentication of both doctor and patient by the server. There's a third category of health data which is called sensitive data. Patient can prefer that a medical illness or a medical treathment to be secret; all the data about this treatment can be seen by only one special doctor who the patient prefers. Sensitive health data can also be stored in the patient card depending on the patient's request.

Design of the security protocol

The goal of the security protocol is to ensure the security of transmission between four active entities of the system. These are doctor card, patient card, client and server programs. In the sample application examined below, problems and requirements in each step are defined and a protocol is developed to meet these requirements. The healthcare application can be examined in six steps.

1. Client Program Session is Started by Doctor

Doctor accesses the system through client machine when the client program session starts. Client program session continues for the doctor until he takes his chipcard out of card reader or he closes the session manually. Client program starts in two stages. One is authentication between doctor and his chipcard; the other is authentication between doctor card and client program.

Doctor has to authenticate himself to the doctor card by entering his personal identification number (PIN) to get his chipcard active on behalf of him. Before checking the PIN, an authentication takes place between the chipcard and the client program. Both entities check each other's validity through a symmetric key which is stored in the client program and chipcard also. If the keys match, then transmission can start between the chipcard and the client. Another goal of this symmetric key is to guarantee the security of the whole transmission between the chipcard and the client program.

2. Reading the Data in the Patient Card

After doctor session is started, doctor wants to read the data in the patient card. First of all patient session must be started with the patient card, then patient card must identify the doctor via the doctor card. There are two conditions which must be achieved to start the patient session in the client program. Firstly, the doctor session must be started successfully. The other condition is the authentication between the client program and the patient card just like that in doctor's session.

3. Patient Card Authenticates Doctor

Patient card must identify the doctor to decide which data will be accessible. Sensitive health data can be accessed by only a specific doctor whose public key and identity (ID) is stored in the card. Doctor authenticates himself to patient card by his chipcard. Here the problem is how can a chip card can verify the signature of the other chipcard, on a condition that none of them makes decription with public key cryptography. The solution is an authentication protocol which makes use of both symmetric and asymmetric cryptography algorithms.

This session will be valid only for one authentication process. A new session key is produced in every session. The problem of sharing the keys is solved with the Elliptic Curve key agreement scheme, Diffie-Hellman version in P1363

It is possible for a party, who wants to authenticate himself, to produce a signature by using this session key and send it to the other side. A symmetric cypering algorithm is used for signing and having this signature be approved. In the application developed, 3DES is preferred which is supported by the smart card.

The authentication process between patient and doctor is made up of the following steps:

1. Patient card informs the client on the start of the authentication process, and sends its public key to the client.

2. Client generates a random number which will be used as the KDP for patient and doctor card.

3. Client sends KDP and public key of the patient (P_PK) to doctor card.

4. Client sends KDP to patient card. (Public key of the doctor is already kept in the patient card)

5. Patient and doctor cards generate session keys.

6. D_ID is encripted with session key in the doctor card and sent to patient card.

7. Patient card decrypts the incoming package using the same session key and compares it with the D_ID kept in itself. If the result is positive authentication is successful. **(See fig.2)**

4. Doctor and Patient's Authentication by the Server

When the doctor wants to connect the server to read the historical information of a patient or to update information there, the server has to approve both the doctor and patient synchronously

For this prupose, the signatures generated by the patient card and doctor card are sent to the server by the client. These signatures may be captured by an attack towards the communication between client and server. To prevent repetitive usage of these signatures, a time stamp is used. Expired signatures are assumed invalid by the server. Another attack may be to change the time stamp. To prevent this, the time stamp(TS) is encrypted together with P_ID or D_ID correspondingly. However if the attacker changes the system time instead of changing the TS, then this method is not preventative. In the design of the protocol, the server machine is assumed to be secure. Security issues like the usage of firewall are subjects of another study.

Server verifies the signatures by checking timestamp and comparing calculated IDs with open ones. If verification is succeded then authentication is achieved.

5. Construction of a Secure Channel Between Client-Server

Transmission of authentication data should be made over a secure channel. The reason behind that is the risk of corruption of data integrity much more than risk of capture of signature by an attack. Secure channel is established by the encryption of data by 3DES which uses a session key which is shared between client and server. Diffie-Hellman key sharing method is used again for the generation of session key. This method does not require a key exchange.

The transmission of data required for authentication from client to server and establishment of a secure channel is achieved by using only one package. This prevents the server be kept busy unnecessaryly and reduces the traffic.

The secure channel between client and server lets the patient data be sent to server securely. Server checks the identity of the doctor before sending the patient's records and lists the records which are secret to that specific doctor together with other records.

6. Patient Card Authenticates Server Over a Secure Channel

Data update in server and patient card process is formed of below steps:

1. Client sends the data update information through the secure channel between client-server.

2. Server separates the parts to be written to the patient card after it updates the patient data (doctor marks this data before sending it to the server). Server checks that if patient data update needed in the card. If not protocol ends here.

3. For the patient card data update, server sends the data to the client together with its approval including its signature through a secure channel between patient card and server.

4. Client seperates the data into blocks and transfer the blocks to card reader to operate on.

5. Patient card makes decription and verifies that the data is sent by the server.

6. Patient card updates the patient data.

Both authentication and data integrity will be ensured by the generation of shared secret and one time usable session keys. The encrypted data coming from the server side is breaked into pieces in the client side and each data block transferred to card reader one by one. Each block by using the session key is decrypted in the patient card side and after the validity check, this data is added to the necessary fields.

Verifying the integrity, hash value of data is appended to the data package in the server side. When the package is unpacked, the hash value is calculated by the patient card and it is compared with the hash value coming with the encrypted data. This way it is ensured that the data is encrypted with the correct session key.

After the design of the security protocol, Zeit Control Smart Card Application Development Kit is used in the development phase of the thesis. Enhanced Basic Card ZC3.3 is programmed which has 8K of EEPROM. Organization of EEPROM space which is used by cryptographic keys and algorithms and health data is another problem solved in the thesis.

The role of smart cards used in the security protocol is more then the other health card applications. Key generation, public key encription, symmetric key encryption and decription operations are made in the smart cards. Secret keys of the users are not stored in the server. Patient authenticates doctor via their smart cards and this authentication process achieved in smart cards. In our design a patients' sensitive data can be linked to only one doctor which is called private doctor. This restriction can be exceeded by using policy transfer mechanisms.

1. GİRİŞ

Yapılan çalışmada ele alınan sağlık sistemi, kişilere ait sağlık verilerinin kişilerin taşıyabileceği bir ortamda ve bilgi merkezlerinde saklanmasını, erişilmesini, kayıt edilmesini ve bakımının yapılmasını gerçekleyen bir bilgisayar sistemidir. Bir pilot sağlık sistemi üzerinde, sistemden hizmet alacak olan kişilerin sisteme güven duymasını sağlayabilecek bir güvenlik protokolü tasarlanmıştır. Kişisel verilerin korunması ve izinsiz ve amacı dışında kullanılmasının engellenmesi amaçlanmaktadır.

Çalışmanın kapsamında; sağlık verilerine erişimi kolaylaştırma ve hızlandırma, hizmet kalitesini arttırma, hasta bilgilerini güvenli bir ortamda saklama, sağlık verilerine değişik hastane, klinik vb. sağlık merkezlerinden erişimin ve bu verilerin hastaya yapılan işlemlerden sonra güncellemesini sağlama, hasta ile ilgili verilerin yönetiminin tek bir merkezden yapılarak tutarlılığı sağlama, hastanın geçmiş tedavi ve teşhis bilgilerini merkezi bir ortamda tutma, kişisel verilerin güvenliğini sağlama yer almaktadır.

Sağlık sisteminde akıllı kart kullanımının temel amacı, hastanın acil durumda gerekli olabilecek verilerinin üzerinde taşıyabileceği güvenli bir ortamda saklanması ve sağlık sisteminde hasta verileri üzerinde işlem yapabilmek için hastanın ve doktorun varlıklarının asıllanmasıdır.

Sistemin kullanıcıları olan hasta ve doktorun akıllı kartları aracılığıyla sistemde işlem yapma yetkilerinin belirlenmesi güvenliğin bir parçasıdır. Akıllı kart kullanımı, kullanıcıların sisteme kendilerini tanıtmalarını sağladığı gibi, hastanın doktora, onun kimliğini kontrol ederek bazı kişisel verilerine erişime izin vermesi gibi olanaklar sunmaktadır.

Sistemde saklanan ve aktarılan tüm kişisel veriler şifrelenmektedir. Veri şifreleme, akıllı kart okuyucusundan bilgi merkezine erişme olarak değişik aşamalarda kullanılmaktadır.

Pilot çalışmada üzerinde güvenlik protokolü tasarımının yapılacağı sağlık sistemi, doktor, doktorun kullandığı terminal veya istemci bilgisayar, bilgi merkezi olarak kullanılan ve terminallerin uzaktan erişebildikleri sunucu bilgisayar, hasta ve doktor ile hastanın akıllı kartlarından oluşmaktadır. Yapılan çalışma istemci ve sunucunun fiziksel konumlarına bağımlı değildir. Ele alınan sağlık sisteminde sağlık verilerinin organizasyonu, veri tabanı yönetimi gibi konular tez çalışmasının kapsamı dışındadır. Bir sağlık sistemi üzerinde akıllı kart kullanarak güvenlik protokolünün tasarlanmasına temel oluşturacak biçimde bir pilot sistem ele alınmıştır.

Tez çalışmasının başlangıcında geniş kapsamlı bir çalışma yapılmış ve öncelikle dünyadaki akıllı kart kullanan sağlık sistemlerinin yapısı incelenmiştir. Bu inceleme 2. Bölüm' de özet olarak sunulmaktadır. Akıllı kartların yapısı, çeşitleri ve standartlarla ilgili bir ön çalışma yapılmış; bu çalışmaya da kısaca 2. Bölüm' de yer verilmiştir.

Güvenlik protokolünün tasarımı yapılmadan önce kullanılacak uygun şifreleme ilkellerinin belirlenmesi ve güvenlik mekanizmalarının oluşturulması için bu konuda kapsamlı bir araştırma yapılmıştır. Akıllı kartlarda simetrik şifreleme verilerin gizliliğinin sağlanması amacıyla, uzun boyutlu verilerin şifrelenmesinde kullanılırken açık anahtar şifreleme imza oluşturma, veya anahtarlar gibi kısa verilerin şifrelenmesinde kullanılır. Akıllı kartların pek çoğunda simetrik şifreleme sistemi olarak bir standart haline gelen DES şifreleme sistemi kullanılırken benzer bir durum açık anahtar şifreleme için söz konusu değildir. Yaygın kullanımda simetrik şifreleme sistemi akıllı kartın ROM'unda yerleşik bulunurken açık anahtar şifreleme sistemleri sonradan tercihe bağlı olarak EEPROM'a yüklenmektedir.

Açık anahtar şifreleme sistemlerinin akıllı kartta kullanımı konusunda çalışmalar sürmektedir. Bu konuda son dönemde en yaygın yönelim Eliptik Eğri Şifreleme Sistemi üzerinde olmuştur. Bunun nedeni, kullanılan parametrelerin diğer sistemlerle karşılaştırıldığında daha az yer işgal etmesidir. Anahtar uzunluğunun küçük olması hesaplamaları da hızlandırmaktadır. Bu da akıllı kartlar açısından Eliptik Eğri Kriptosistemlerinin tercih edilmesine yol açmaktadır. Bu nedenle şifreleme mekanizmaları arasından eliptik eğri temelli olanların incelenmesine

ağırlık verilmiştir. Eliptik eğri şifreleme sistemi 3. Bölüm'de incelendikten sonra eliptik eğri protokolleri 5. Bölüm' de anlatılmıştır. 6. Bölüm'de ise Açık Anahtar Kriptografi için standart tanımlara ve kabullere yer verilirken bunlar arasında eliptik eğri temelli olanlar üzerinde durulmuştur.

Şifreleme sistemleri üzerine yapılan çalışmaya 3. Bölüm'de yer verilmiş, burada ağırlıklı olarak açık anahtar şifreleme sistemleri üzerinde durulmuştur. Akıllı kartlarda tampon boyunun darlığı nedeniyle veri şifrelemede kullanılan simetrik şifreleme blok şifreleme yapısında kullanılmaktadır. Bunun için simetrik şifreleme bölümünde blok şifrelemeye yer verilmiş, simetrik algoritmalar üzerinde durulmamıştır.

Algoritmaların incelenmesinin ardından 4. Bölüm'de bir güvenlik protokolünün temel elemanları incelenmiş, kullanım alanları açısından değerlendirilmiştir. Bu bölümde çırpma fonksiyonlarına yer verildikten sonra simetrik ve asimetrik şifreleme sistemleri olumlu ve olumsuz yanlarıyla, karşılaştırmalı olarak incelenmiş, hibrit kriptosistemler ele alınmıştır.

Asıllama protokolleri, sayısal imza mekanizmaları, anahtar ortaklaştırma düzenleri, tasarlanacak güvenlik protokolünün en önemli bileşenleridir. Önceki bölümlerde oluşturulan temeller üzerine 5.Bölüm'de bilgi sistemi protokolleri tanıtılmıştır.

Güvenlik protokolünün tasarımı ve gerçeklenmesi aşamasına geçmeden önce kullanılacak uygulama geliştirme ortamının ve araçlarının tanıtılmasına yer verilmiştir. Zeit Control Basic-Card üzerinde program geliştirilmiştir, bu nedenle 7.Bölüm'de Basic-Card program geliştirme kiti tanıtılmıştır. 8. Bölüm'de ise Zeit Control kitinin güvenlik mekanizmaları anlatılmıştır.

Buraya kadar tez çalışması tasarım ve gerçekleme amacına yönelik bir ön çalışma niteliğindedir. Buradan sonraki bölümlerde sırasıyla sağlık sisteminin temel bileşenleriyle birlikte tanıtılması, güvenlik protokolünün tasarımının alternatif durumlar da ele alınarak tartışılması ve gerçekleme aşaması yer almaktadır.

Tez çalışmasının önemli özellikleri aşağıdaki biçimde sıralanabilir:

Akıllı kart okuyucunun bağlı olduğu terminale güven duyulmaksızın iki akıllı kartın üzerinde yürütülen kriptografik işlemlerle birinin diğerini asıllaması sağlanmaktadır.

Güvenlik protokolünün bir diğer aşmasında sunucu, merkezi veri tabanındaki bilgilere erişimi denetlemek için istemci üzerinden ikili kimlik doğrulama yapmaktadır. İki varlığın da aynı anda asıllanması anlamına gelen bu işlem için doktorun ve hastanın akıllı kartları kullanılmaktadır.

İstemci makinaya bağlı akıllı kartın güncellenme işlemi ise uzaktan erişim yoluyla sunucu tarafından gerçekleştirilmektedir. Burada asıllama ve güvenlik problemlerinin çözümü sağlanmıştır.

Mikroişlemcili kartın üzerinde bulunan DES ve 3DES simetrik şifreleme sistemi dışında akıllı karta açık anahtar şifreleme algoritması olarak Eliptik Eğri Kriptografisine dayalı algoritmalar yüklenmiştir. Akıllı kart programlama için Basic programlama dili temelli ZC-Basic Akıllı Kart kiti kullanılmıştır.

Tasarlanan güvenlik protokolünde kullanıcıların gizli anahtarları kullanıcı akıllı kartları üzerinde üretilmekte, akıllı kartların dışında hiç bir ortamda saklanmamaktadır. Sunucuda yalnızca kullanıcıların açık anahtar bilgileri tutulmaktadır.

Sunucu, istemci üzerinden akıllı karta erişerek gerekli asıllamalar gerçekleştirildikten sonra kartı güncellemektedir. Güncelleme verilerinin güvenli bir biçimde karta iletilmesi güvenlik protokolünün çözümlediği problemler arasındadır.

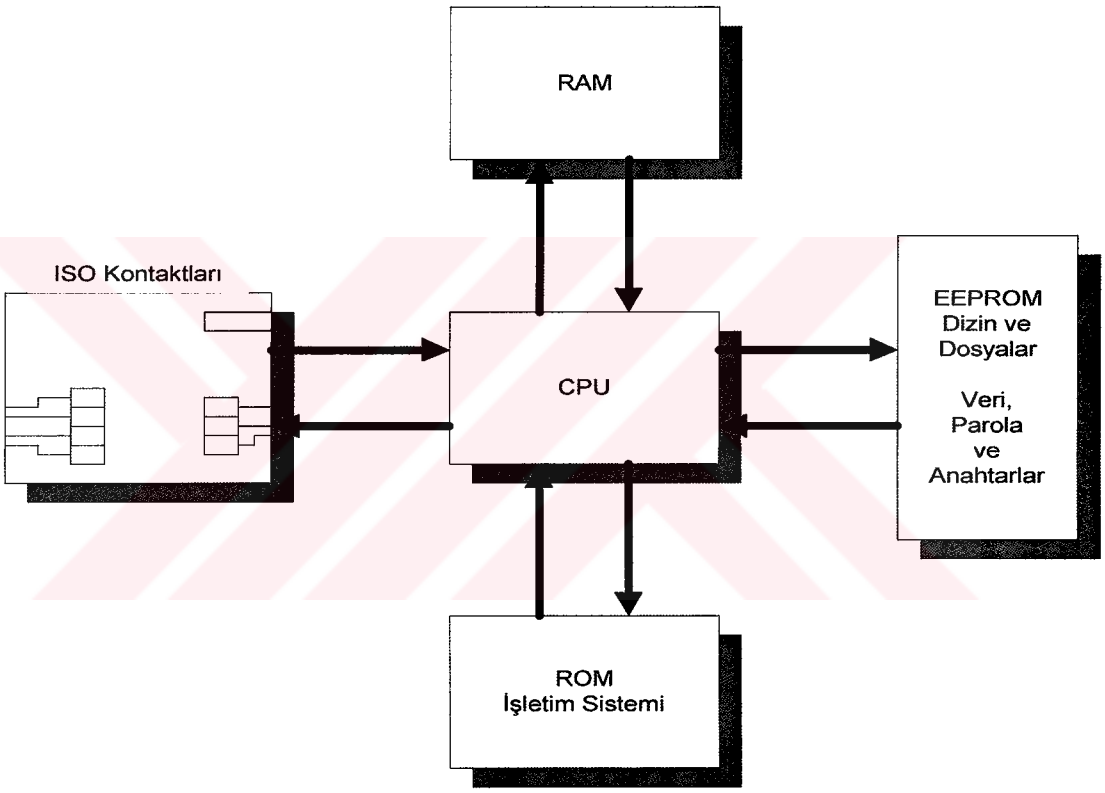
2. AKILLI KARTLAR VE SAĞLIK SİSTEMLERİNDE KULLANIMI

Akıllı Kartlar ilk kez 1974'te Moreno/Innovatron tarafından tanıtılmıştır. İlk büyük uygulaması 1982'de Fransa'da gerçeklenmiştir. Avrupa ülkelerinde GSM gibi yaygın kullanım alanları oluşurken, ABD'de yaygın kullanım çok sonraları başlamıştır. X.509 olarak da bilinen sayısal sertifikaların kullanıcı ve organizasyon tabanında asıllama amacıyla kullanımına dayalı olarak gelişen elektronik ticaretin yaygınlaşması, akıllı kartların bu alanda geniş bir pazar bulmasına neden olmuştur. Böylece akıllı kart teknolojileri hızla gelişmeye başlamıştır.

Akıllı kart tabanlı sağlık kartı uygulamaları da yaygınlaşmaktadır. Hastaların tıbbi kayıtları arasından küçük bir veri kümesi seçilerek sağlık kartına yüklenir. Böylece kart kullanımı ile, taşınabilir tıbbi kayıtlar, diğer kayıtlara erişim bağlantıları, veri tabanları için erişim anahtarları ve sağlık hizmetleri arasında bütünlüğü sağlayan bir sistem kurulmasını kolaylaştırma gibi yetenekler elde edilir. Sağlık kartı uygulamalarında internet önemli bir yere sahiptir. İntranet/internet teknolojilerinin yayılması daha yüksek seviyeden güvenliği gerektirir. Güvenlik özellikle siteler arası haberleşmenin korunmasında, kişisel bilgilerin saklanmasında, elektronik hareketlerin korunmasında önem kazanmaktadır. Bu türden uygulamalarda kullanılacak olan hizmetler; istemci işlemlerinde sunucu tanımlama, sunucu işlemlerinde istemci tanımlama, kriptografi ve elektronik imzalar, kişisel verilerin saklanması (kişisel anahtarlar), genel bilgilerin saklanması (açık anahtarlar) olarak sıralanabilir. Sağlık kartlarının uygulama alanları arasında, acil yardım, toplumsal ve genel tedavi (diabetik, özürlü vb. tedavileri), koruyucu hekimlik, nüfus toplulukları (çocuklar, yaşlılar, hamile kişiler vb.), muhasebe ve yönetimsel amaçlar, veri bankalarını sorgulama ve farklı hizmetler arasında veri alış verişi sayılabilir. Sağlık kart, sosyal güvenlik kartı, sigorta kartı olarak adlandırılan, kapsamlarında farklılıklar taşıyan uygulamalar aşağıda Bölüm 2.2' de özetlenmektedir.

2.1. Akıllı kartlar

Akıllı kart, içine bir mikroişlemci ve bir bellek tümdevresi veya programlanabilme özelliği olmadan yalnızca bellek tümdevresi yerleştirilmiş bir elektronik karttır. Mikroişlemcili kartlar, kart üzerinde bulunan veriler üzerinde değişiklik yapılmasına olanak tanırken bellek kartları yalnızca önceden tanımlanmış işlemleri yürütebilirler. Akıllı kartlar üzerinde bilgi saklayabilmesi ve verilerin güvenliğini sağlayabilmesi açısından manyetik kartlardan ayrılır.



Şekil 2.1.Akıllı kart mimarisi

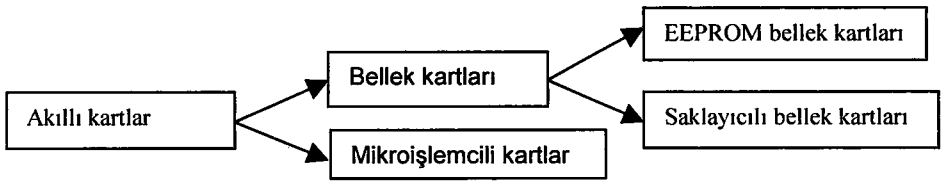
Şekil 2.1'de bir mikroişlemcili kart mimarisi görülmektedir. CPU, merkezi işlem birimi olup; ROM'da akıllı kart işletim sistemi bulunmaktadır. EEPROM'da dizin ve dosyalar, veriler, parola ve anahtarlar tutulurken sonradan karta eklenecek ek işlevler, programlar da burada saklanır. Burada tutulan veriler sonradan erişilerek değiştirilebilir. Akıllı kartın yapısı görüldüğü gibi bir bilgisayarda bulunan temel özelliklere sahiptir. Akıllı kartlar yapısal olarak Şekil 2.2'de görüldüğü gibi sınıflandırılır. Bellek kartları, genel olarak veri saklamak amacıyla kullanılır. En genel uygulama bir kullanımlık telefon kartlarıdır. Ödeme önceden yapılır,

karttaki sayaç sıfırlanana kadar kullanılır ve ardından bu sayaç tekrar kullanılmak üzere yeniden artırılamaz. Bellek kartlarında bu işlemlerin güvenliği gibi konularda henüz bir standart oluşmamıştır. Bir terminalin kontrolünde senkron çalışırlar. Mikroişlemcili kartların asenkron çalışması ve bir dizi işlem yaptıktan sonra terminale bir yanıt döndürmeleri, bellek ve mikroişlemcili kartlar arasındaki çalışma farklarından birisidir.

EEPROM bellek kartları, yeniden yazılabilir belleği olan ve bilgi saklamak amacıyla kullanılan kartlardır. Kartların içinde kullanılan dosya yapısına ilişkin olarak herhangi bir standart mevcut değildir.

Saklayıcılı bellek kartları ise sınırlı belleğe sahiptirler. Bu kartlarda abaküs tipi sayma metodu vardır. Sayaç sıfırlandığı zaman yeniden kullanılamazlar.

Mikroişlemcili kartlar gerçek anlamda "akıllı" kartlardır. Mikroişlemci birimi, RAM, ROM, veri saklama amacıyla genellikle EEPROM, giriş/çıkış arabirimi ve bir işletim sistemi vardır. Mikroişlemci birimi, matematiksel işlemleri yürütebilmek için bir dizi komut kullanır. Bazı durumlarda basit bir kesme kontrol sistemi de içerebilir. RAM, 128 sekizli ile başlayan ve giderek artan genişlikte olabilmektedir. Uygulamalar karmaşıklaştıkça bellek gereksinimi de artmaktadır. ROM'da akıllı karta sabitlenmiş programlar bulunur. Akıllı kartta, PC'deki disk sürücü yerini tutan bilgi saklamak için kullanılan EEPROM genişliği, 1K-64K arası değişebilmektedir. Giriş/çıkış arabirimi ise evrensel asenkron alıcı/gönderici, UART diye adlandırılır.



Şekil 2.2. Akıllı kartların sınıflandırılması

2.1.1. Akıllı kart ile sayısal imza

Sağlık uygulamalarında kimlik denetimi ve asıllamanın gerçekleştirilmesi için kullanılan sayısal sertifikalar, bir çeşit sayısal imza işlevini görür. Örneğin bir kişinin sayısal imzası ele geçirildiğinde o kişi yerine imza atmak ve o kişinin

kılığına bürünerek onun yetkilerini ele geçirmek mümkün hale gelir. Sayısal imza ile kişisel anahtarın bir akıllı kart üzerinde birleşimi, sayısal imzalarla ilgili bir dizi güvenlik ve uyumluluk konusunu çözüme kavuşturmaktadır. Bir sayısal sertifika bir kişi veya bir organizasyon için bir kez yayınlandıktan sonra korunur. Böylece başkaları imzayı ele geçirip imza sahibinin yerine geçemez. Akıllı kartlar, kopyası alınması zor olduğundan sayısal imzaların saklanması için en uygun mekanizmalardır. Akıllı kartlarda kullanılan PIN numarası ise kişiye ait bir bilgi olup kimliğin kontrol edilmesi için ikinci bir güvenlik mekanizması oluşturur. Kişinin kartı kullanabilmek için PIN kodunu bilmesi gerekir. Akıllı kartın sağladığı diğer bir yetenek ise sayısal imza ile birlikte ek özel verileri güvenli bir şekilde taşınabilir hale getirmesidir.

Böylece akıllı kartların kullanımı ile desteklenen sayısal imza yardımıyla gerçeklenecek internet üzerinden asıllama işlemi, günümüzde intranetler için de kullanılmaktadır. Asıllama problemlerine çözüm olarak Simetrik Anahtar kripto tekniklerine ek olarak akıllı kart üzerinde Açık Anahtar Kriptografi de kullanılır.

2.1.2. ISO-7816 Standardı

Akıllı kartların geniş alanda ve ortak uygulamalarda kullanımının yaygınlaşması için kartlarda ve kart okuyucularda bazı standartların geliştirilmesi zorunlu hale gelmiştir. Bunun için Uluslararası Standartlar Enstitüsü (ISO) kontaklı tümdevre kartları için ISO 7816 standartlarını geliştirmiştir [1]. Bu standart bellek kartları ve mikroişlemcili kartları kapsamaktadır. Her ikisi için de kontakların yerleri tanımlanır. Fakat bellek kartları için kabloların ve bağlantıların işlevi tanımlanmamıştır. ISO 7816 birden fazla bölümden oluşur. Her bölüm fiziksel karakteristikler, kontakların yönleri ve yerleri, veri erişim teknikleri, veri depolama teknikleri, sayılama sistemleri ve kaydetme işlem adımları konusunda en alt düzeyde sağlanması gereken özellikleri kapsar. Çoğunlukla kart üreticileri, kendi ürünlerini diğerlerinden ayırt etmek için yüzeyde kontakları farklılaştırmaktadırlar. Bu yüzden standart yalnızca kart ile okuyucu arasında iletişimi sağlayan elektriksel kontakların yerlerini tanımlar.

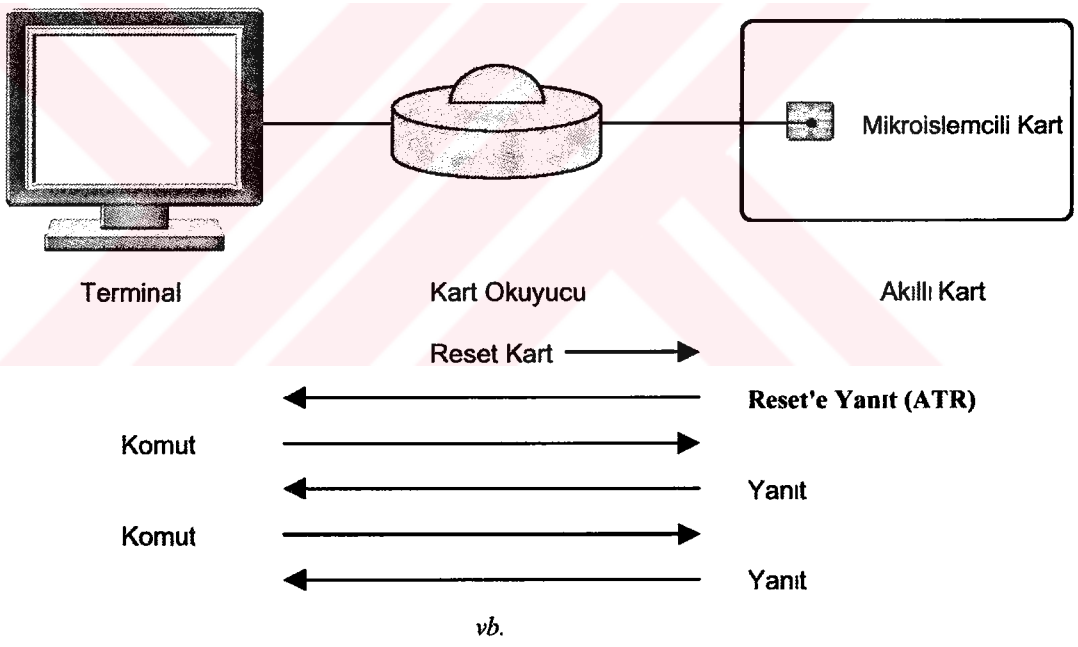
Standartlar fiziksel, elektriksel, ve veri bağlantı protokol katmanlarındaki uyumluluğu taban alarak tasarlanmıştır. Fakat aygıt bağımsız Uygulama Programı

Arayüzleri (API), geliştirme araçları, kaynak paylaşımı gibi konularda uyumluluk kapsanmamaktadır.

Bu konudaki gelişmelerin standartlarını oluşturmak için Mayıs 1996'da Microsoft, Groupe Bull, Hewlett-Packard, Schlumberger ve Siemens Nixdorf gibi akıllı kart ve kişisel bilgisayar şirketleri tarafından PC/SC(Personal Computer / Smart Card) çalışma grubu kurulmuştur.

2.1.2.1. Komut-yanıt protokolü

İşlemci kartla haberleşme bir komut-yanıt protokolü yardımıyla olur. Kart okuyucuya yerleştirildiğinde terminalin karta reset komutunu göndermesiyle komut-yanıt oturumu başlar.



Şekil 2.3 Komut-yanıt protokolü

ISO standart dokümanları ISO/IEC 7816 –3 ve 4'te [2][3] tanımlıdır ve pek çok işlemci kartta kullanılır. Komut-yanıt protokolü genel mesaj akışı şekil 2.3'de görülmektedir. Kartın terminalle iletişime geçmesi için karta Reset komutunun iletilmesi gerekir. Kart okuyucudan Reset komutunu alan akıllı kart, komut-yanıt oturumunu başlatır. Reset komutu kartı, başlangıç konumuna getirir. Reset komutu akıllı kart ve terminal arasındaki el sıkışmanın ilk adımıdır. Bu el sıkışma

sırasında kartın ve kart okuyucu terminalin teknik özelliklerine göre haberleşme ve protokoller kurulur. Reset komutuna yanıt ile gönderilen kurulum sekizlileri kartın özelliklerini tanımlayan değerleri içerir. Haberleşmenin hızının, gereksinim duyulan gerilim değerinin, mikroişlemcinin işlem hızının bildirilmesi gibi başlıklar bu özellikler arasındadır. Akıllı kart üzerinde yürütülen her işlem, terminalin gönderdiği komutlara yanıt olarak yürütülür.

2.1.3. Akıllı kart işletim sistemleri

Farklı üreticiler tarafından üretilen akıllı kart işletim sistemlerinin şimdiye kadar birbirine uyumlu olduğu görülmemektedir. Bu da uygulama geliştiricilerinin işini zorlaştırmakla birlikte maliyeti artırıcı bir etkendir. Günümüzde öne çıkan işletim sistemleri "MULTOS" ve "JavaCard" dışında yeni olarak "Smart Cards for Windows" tur.

MULTOS, yüksek güvenlikli çoklu-uygulamalara dayalı bir akıllı kart işletim sistemidir [4]. Kart fonksiyonları, bu sayede yüksek güvenlikli bir platform üzerinde geliştirilebilir. Akıllı kartlar için geliştirilmiş uluslararası standartlarla uyumludur. MULTOS'ta kart üzerine yerleştirilen her uygulama bellekte kartın geri kalanından soyutlanarak güvenlik sağlanır. Bu özellik kartın kullanımı sırasında da herhangi bir uygulamanın güvenli bir biçimde bir kişisel bilgisayar veya başka bir araç tarafından diğerlerinden bağımsız olarak silinebilmesine veya eklenebilmesine olanak verir. Multos için uygulamalar "C" gibi yüksek düzeyli bir dil kullanılarak da geliştirilebilir.

Java Card teknolojisi, platformdan bağımsız olması ve tek bir kart üzerinde farklı uygulamaların çalışmasına izin vermesi nedeni ile tercih edilmektedir [5]. Kart bir kez üretildikten sonra değişen ihtiyaçlara bağlı olarak yeni uygulamalar yüklenmesine olanak tanır. Ayrıca, akıllı kartların programlanmasında nesneye dayalı programlama yeteneklerinden dolayı bir esnekliğe sahip olması ve var olan akıllı kart standartlarıyla uyumluluğu nedeniyle işletim sistemi olarak Java Card kullanılması yararlı olmaktadır.

Ekim 1998'de Paris'te düzenlenen Cartes sempozyumunda Microsoft, akıllı kartlar için geliştirdiği Smart Cards for Windows işletim sisteminin duyurusunu

yapmıştır. Bu sistemde ağlar ve internet web sunucularına güvenli erişim, açık/ kapalı anahtara dayalı uygulamalar ve elektronik-ticaret uygulamaları yer almaktadır. Standart sürümün bir parçası olarak Windows 2000 işletim sistemi, akıllı kart tabanlı kullanıcı onayını desteklemektedir.

2.2. Akıllı kartlar ve sağlık alanındaki kullanımı

Lombardia Bölgesi'nde bir çok işlevli kart sistemi hayata geçirilmiştir [6]. Sosyal bilgi sistemi ve sağlık sistemi bir arada düşünülmüştür. Sağlık harcamalarını ve kaynak kullanımını kontrol etmek amacıyla yola çıkılarak tüm sağlık çalışanlarını birbirine bağlayan bir ağ kurulmuştur. Bu ağ, Genel Yönetim/ Sağlık Bölümü ile iletişimin güvenli akışını ve verimli haberleşmeyi denetlemektedir. Lecco'da tüm halka mikroişlemci kartlar dağıtılmıştır. Kart, sisteme girebilmek ve ağdaki servislerden yararlanabilmek için bir anahtar işlevini görmektedir. Hasta ile ilgili bilgilerin bir kısmı kart üzerinde bir kısmı da hastaneler veya diğer yapıların veri ambarlarında tutulmaktadır.

Diğer yandan ABD'de sağlık kartı uygulama projeleri hızla çeşitlilik kazanmaktadır. Eylül 1999'da Amerikan Askeri Birliği, plastik kartların yerini, çoklu uygulamalı, akıllı kartların alacağını açıklamıştır [7]. Tasarıya göre yaklaşık 800 bin personelin bu yeni kartı kullanması planlanmaktadır. Kartın mevcut ağ yapısı ile bağdaştırılması ve gelişmekte olan Açık-Anahtar Altyapı Sistemi ile uyumlu anahtarlara sahip olması planlanmaktadır. Bu projenin 2003'te tamamlanması beklenmektedir.

ABD'de akıllı kart ile sağlık uygulamaları konusunda şimdiye kadar yapılan en büyük proje Ocak 1999'da Bismark, Kuzey Dakota ve Cheyenne, Wyoming'de başlatılmıştır[7]. 25 binden fazla akıllı kart ailelere dağıtılmıştır. "Health Passport Project" adı verilen proje ile hastanın geçmiş önemli sağlık kayıtlarının akıllı kartta güvenli bir şekilde saklanması amaçlanmıştır. Ayrıca kartın yaygın olarak pek çok sağlık biriminde de kullanımı amaçlanmaktadır. "Health Passport" projesi, akıllı kartlar, uç birimler, okuyucular ve yazılımlardan oluşmaktadır.

ABD'de bir başka pilot bölgede ise çok işlevli akıllı kartların uygulaması başlatılmış. "VA/DoD" Çoklu-Uygulama kartı, hasta/ kullanıcı kimliğini

belirlemek, acil erişilmesi gereken veriyi tutmak, bilgiyi birimler arasında paylaştırmak gibi amaçların yanında genel amaçlı kullanım için bir elektronik-cüzdan uygulamasıdır.

Oklahoma'da 1997'de başlatılan sağlık alanındaki bir başka akıllı kart uygulaması ise bir acil durum projesidir. Akıllı kart birleşik sağlık hizmeti dağıtımı ağının ilk adımı olarak kullanılır. Ambulans çalışanları ve eczacılar için gerekli bilgileri içerir.

Fransa'da geçtiğimiz yıllarda uygulamaya konan akıllı kart uygulamasında ise sağlık ve sosyal güvenlik hizmetlerinin bir arada yürütülmesi amaçlanmış, 42 milyon kart halka dağıtılmıştır [8]. Sağlık personelinin güvenli yoldan haberleşmesi, ödemelerin kolayca yapılabilmesi, sağlık verilerine erişimin denetlenmesi gibi amaçlara yönelik olarak bu kartlar kriptografik işlemci içermektedir ve gizli-anahtar güvenli bir şekilde saklanmaktadır. Fransa'da bir sağlık ağı kurulmuştur. Ağdaki bilgiye erişimi denetlemek amacıyla "Healthcare Profession Card" adı verilen uzman personele ait kartlar kullanılmaktadır.

Almanya ise 1994-95 yıllarında 80 milyon kart dağıtarak bir sağlık, sosyal güvenlik uygulamasını başlatmıştır [9]. Bu kartlar sosyal sigorta bilgisini taşır. Okuyucu/yazıcı sistemi sayesinde hastanın sosyal sigorta formu yazıcıdan alınabilir. Sigorta fonu ile elektronik olarak haberleşme sağlanır. Bunun dışında Almanya'nın da uluslararası bir uygulama olarak tasarlanmakta olan NETLINK projesine dahil olabilmesi için ek işlevler geliştirilmektedir. Proje Berlin'de Sağlık Kurumu tarafından desteklenmektedir. Hem hasta hem de sağlık personeli için asıllama işlemi, sayısal imza kullanımı, internet üzerinde güvenli haberleşme konularında ilerleme sağlanmıştır.

NETLINK projesi, Avrupa Birliği tarafından desteklenen, sağlık kartındaki bilgilerin ülke dışında da geçerli olabilmesi için bir uyumlulaştırma projesidir[10]. 1998'de başlatılan projenin pilot bölgeleri, Fransa, Almanya, İtalya, ve Kanada'dır.

Bir başka çok uluslu akıllı kart projesi ise Avrupa Komisyonu tarafından desteklenen CARDLINK'tir [11]. Eylül 1994'te başlatılmış ve 5 yıllık pilot aşamasını tamamlamıştır. Bu kartlarda acil sağlık verileri ve kimlik bilgisi

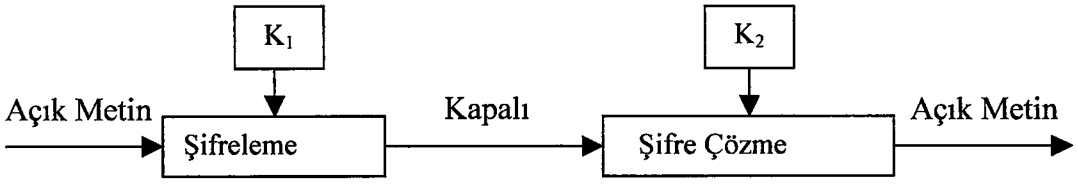
saklanmaktadır. Hastanın daha önceki tedavi bilgilerine erişim için bu bilgilerin merkezlerine internet üzerinden ulaşmayı sağlayan işaretçiler bulunmaktadır. Yaklaşık 100 bin kart dağıtılmıştır. Acil durum bilgisi katılımcı ülkelerin hangisinde kullanıldığına bağlı olarak o ülkenin diline çevrilmek üzere uyumlaştırılmıştır. 5 pilot bölge, Fransa'da Saint Nazaire, İrlanda'da Dublin, İtalya'da Milan ve Roma ve İspanya'da Valencia'dır. Aynı amaçla 1995'te başlatılan bir başka proje G-8 ülkelerini kapsamına alan G-8 Sağlık Hizmeti Verisi Kart Projesi'dir [11].

Bir başka uluslararası proje ise TrustHealth İnsiyatifi'dir [9]. Bu proje, tüm Avrupa'yı kapsayan açık sistemlerin bağlantısı içinde, modern güvenlik teknikleri kullanan güvenilir "telematik" sistemleri göstermek için tasarlanan bir çatıdır. Sağlık bilgilerinin güvenliğini sağlamak için RSA şifreleme ve akıllı kart kullanılmıştır. Burada ulusal güvenilir üçüncü şahıslar anahtarların bulunduğu kartları yayınlamak ve kullanıcı ile açık-anahtar arasındaki bağlantıyı sağlamak için gereklidir. Katılımcı 9 ülke arasında İsveç geniş rol oynamaktadır.

3. ŞİFRELEME

Şifreleme (kapatma) bir verinin bir dönüşümden geçirilerek eskisinden farklı, anlaşılamaz bir biçime dönüştürülmesidir. Şifre çözme (açma) ise şifrelenmiş bir veriyi, bir geri dönüşüm mekanizmasından geçirerek özgün haline getirmedir. Şifreleme işleminde K_1 ve K_2 anahtarlarının varlığı zorunlu değildir. (Şekil 3.1)

Bilgisayarlar şifreleme konusunda kullanılmaya başladıktan sonra şifrelenen veri sekizli kodlarından oluşmaktadır. Şifreleme için kullanılan şifreleme algoritmaları matematiksel fonksiyonlardır. Bilginin iki kişi arasındaki gizliliğini sağlamak için ilk zamanlar kullanılan yaklaşımlardan biri şifreleme algoritmasının gizliliği olmuştur. Ne var ki bilgisine yalnızca haberleşecek kişilerin sahip oldukları bu fonksiyonun gizli olması fonksiyonun güvenilirliğinin sınanması, kalite kontrol ve standardizasyon işlemlerini olanaksız kılar. Bu nedenle taraflar arasında güvenliği sağlamanın algoritmanın gizli tutulmasıyla gerçekleştirilmesi yaklaşımı bugün geçerliliğini yitirmiştir. Algoritma güvenliğin mekanizmasıdır ve mekanizmanın test edilmesi, ne derece sağlam olduğu konusunda uzmanların onayını almış olması gerekir. Bu nedenle gizlilik, mekanizmanın kendisinde değil mekanizmanın işlemesini sağlayacak bir anahtar bilgide olmalıdır. Şifreleme ve şifre çözme işlemini yerine getirebilmek için sahip olunması gereken bilgi anahtarın bilgisidir. Kriptografik anahtarlar (K) bu amaçla kullanılır. (Şekil 3.1)



Şekil 3.1. Şifreleme ve şifre çözme

Şifrelemede kullanılan algoritmalar "**açık kapılı tek yönlü fonksiyonlar**"dır. Tek yönlü fonksiyon, hesaplaması kolay olan fakat tersi alınması zor olan fonksiyonlardır. Burada "zor" un anlamı tersini hesaplayabilmek için yıllar mertebesinde zamana ihtiyaç duyulmasıdır. Açık kapılı tek yönlü fonksiyonlar ise

bu açık kapı bir gizli anahtardır. Bu anahtar bilindiğinde fonksiyonun tersini hesaplamak için bir açık kapı vardır. Kriptografik algoritmalar bu tür matematiksel fonksiyonlara dayanır.

Şifreleme işlemi, P ile simgelenen açık verinin K_1 anahtarı kullanılarak E fonksiyonundan geçirilmesi ve C sonucunun elde edilmesidir. Burada C, kapalı veridir. Kapalı veriden açık veriye ulaşılabilmesi için şifre çözme işleminin yerine getirilmesi, C'nin D fonksiyonundan K anahtarı kullanılarak geçirilmesi gerekir. Burada da K_2 anahtarının kullanımı seçilen şifreleme sistemine bağlıdır.

Anahtarları bilmeden şifrelenmiş bir metinden açık metini elde etme bilimine **kriptanaliz** denir. Başarılı bir kriptanaliz, açık metni veya anahtarı elde edebilir. Bir saldırı, kriptanaliz kullanabilir. Güvenlik testleri, algoritmaların herkes tarafından bilinebilir olduğu fakat anahtarların gizli olduğu varsayımına dayanarak yapılır. Algoritmanın bilindiği varsayımı altında saldırılar dört temel grupta toplanabilir.

1. Bilinen şifreli metin: Saldırganın elinde aynı algoritma tarafından şifrelenmiş metin örnekleri bulunmaktadır. Buradan yola çıkarak saldırganın herhangi bir örneğin açık metinini elde etme veya kapalı metinlerden açık metinleri hesaplayabilen bir algoritma bulma şansı artar.

2. Bilinen açık metin: Saldırganın elinde yalnızca kapalı metin örnekleri değil aynı zamanda ilgili açık metinler de bulunmaktadır. Buradan yola çıkarak kullanılan anahtarı elde etmeye veya aynı anahtarla şifrelenmiş metinlerin şifresini açmaya yarayan bir algoritma bulmaya çalışır.

3. Seçilen açık metin: 2. Şıktaki durumdan farklı olarak saldırgan elindeki kapalı ve açık metin örneklerini seçerek alma şansına sahiptir. Bu durumda saldırgan anahtar hakkında daha çok bilgi vereceğini düşündüğü istediği açık metinleri seçebilir.

4. Sürekli seçilebilen açık metin: 3. şıktaki durumdan farkı saldırgana seçtiği ve üzerinde deneme yaptığı açık ve kapalı metin örneklerini bulgularına göre yeniden seçebilme olanağı sunar. Tüm örnekleri tek bir defada seçmek yerine gelişmelere göre örnekler sürekli seçilebilir.

5. Seçilebilen şifreli metin: Saldırganın bazı şifreli metin örneklerini seçerek ayrıca bu şifreli metinlerden açık metinleri elde etme olanağı vardır. Yani saldırganın elinde otomatik şifre açma yapan bir kutu vardır. Burada saldırgan anahtarı belirlemekle uğraşmaktadır.

6. Seçilen anahtar: Saldırganın anahtarı seçmesi anlamına gelmez. Saldırganın değişik anahtarlar arasındaki bağlantıya dair bilgisi olduğu durumdur.

3.1. Simetrik şifreleme sistemleri

Simetrik algoritmalar, gizli anahtar şifreleme, tek-anahtarlı algoritmalar veya geleneksel algoritmalar olarak da adlandırılmaktadır. Şifre açma ve kapatma için kullanılan anahtarlar birbirlerinde elde edilebilirler veya birbirlerinin aynıdırlar. (K_1=K_2)

Mesajı gönderen ve alan tarafın haberleşebilmeleri, yani mesajları şifreleyip açabilmeleri için aralarında bir gizli anahtar konusunda anlaşmış olmaları gerekir. Simetrik algoritmanın güvenliği anahtarların haberleşen taraflar arasında gizli olmasında yatar. Mesajı kapatma ve açma işlemi aşağıdaki şekilde gösterilir:

Kapatma: $E_K(M) = C$

Açma: $D_K(C) = M$

Simetrik algoritmalar ikiye ayrılır:

1. Seri veya sürekli algoritmalar
2. Blok algoritmalar

Seri şifrelemede birim zamanda tek bit üzerinde işlem yapılır. Blok şifrelemede ise birim zamanda bir grup bit üzerinden işlem yapılır. Blok uzunluğu değişebilir. Analizi engellemeyecek kadar büyük, işlemleri aksatmayacak kadar küçük bir blok genişliği olarak genellikle 64 sekizli tercih edilir.

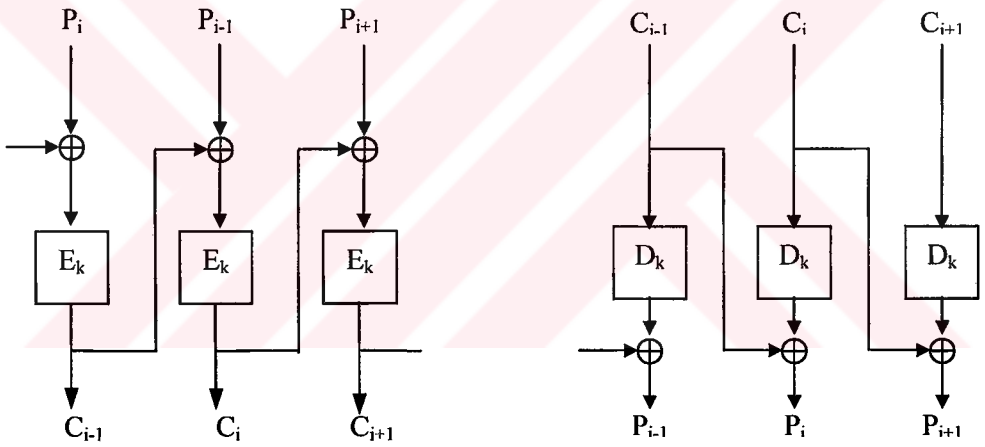
Seri şifrelemeyi de aslında tek bitlik bir blok şifreleme gibi düşünebiliriz. Şifreleme dönüşümünün, şifrelenecek her sembol için farklılaşabilir olması bu yöntemin avantajlarından biridir. Gönderim hatalarının yoğunlukta olduğu durumlarda da seri algoritmalar daha elverişlidir. Bit bit işletildiğinden, bir bitteki

hata diğerlerini etkilemez. Yani "hata yayılımı" yoktur. Seri şifreleme, bellek kısıtının olduğu veya veri tamponlama sınırlamalarının olduğu donanımlarda da tercih edilebilir.

3.1.1. Şifreli blok zincirleme modu (CBC-Cipher Block Chaining Mode)

Mesaj sabit uzunlukta bloklara bölünüp şifrelenirken, bloklar birbirinden bağımsız olarak şifrelenmek yerine bir önceki şifreleme işlemi bir sonrakini etkileyecek biçimde bir geri besleme mekanizması kurulur. Bir önceki bloğun şifrelenmesiyle oluşan sonuç, ilgili bloğun şifrelenmesi işlemine katılır. İşlem bu şekilde zincirleme devam eder.

CBC modunda açık mesaj bloğu şifrelenmeden önce bir önceki bloğun şifreli metni ile DARVEYA'lanır. Şekil 3.2.a'da CBC şifreleme yöntemi görülmektedir.



a. Şifreleme b. Şifre çözme

Şekil 3.2. Şifreli blok zincirleme modu

Şifre çözme için ise işlemler tersinden takip edilir. Şifrelenmiş blok şifre çözme işleminden (D_k) geçirildikten sonra bir önceki şifreli blokla DARVEYA'lanır (Bakınız Şekil 3.2.b).

$C_i = E_k (P_i \oplus C_{i-1})$

$P_i = C_{i-1} \oplus D_k (C_i)$

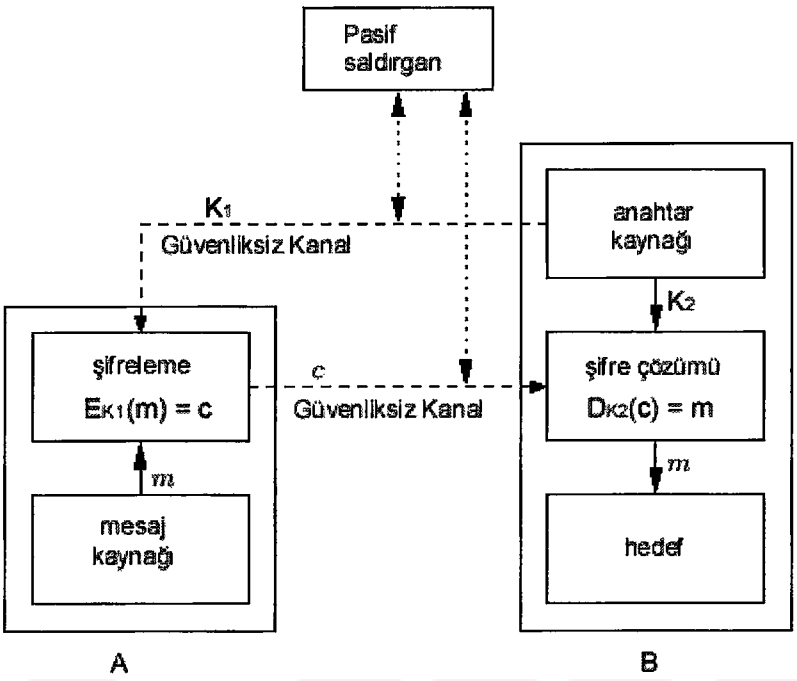
Bu yöntemde ilk başlangıç bloğunu DARVEYA'lamak için bir başlangıç vektörü kullanılır. Bu vektör sabit bir değer olabileceği gibi, her defasında farklı olarak üretilen ve şifreli mesajla birlikte karşı tarafa yollanan bir rasgele değer de olabilir. Her mesaj için yeni bir başlangıç vektörü üretimi, içeriği aynı olan mesajların şifrelenmiş halinin de birbirinin aynı olması durumunun yaratacağı riske karşı bir önlem olarak kullanılabilir. Kullanılan başlangıç vektörünün gizliliği gerekli değildir. Açık bir biçimde karşı tarafa gönderilebilir.

3.2. Asimetrik şifreleme sistemleri

Asimetrik şifreleme, açık anahtar şifreleme olarak da adlandırılır. Şifreleme ve şifre çözme işlemini yapan anahtarlar birbirinden farklıdır ($K_1 \neq K_2$).

Bu anahtarlardan birisi gizli diğeri ise açık anahtar olarak kullanılır. Açık anahtar, sistemde gizliliği gerekmeyen anahtardır. Gizli anahtar ise kişiseldir. Kriptosistemin güvenliği açısından gizli anahtar, açık anahtardan elde edilemez. En azından güvenliği sağlamaya yetecek kadar uzun bir süre bu işlemin yapılmasının mümkün olmaması gerekir. Fakat tersi mümkün olabilir. Gizli ve açık anahtar çifti üretilirken bu özellik gözetilerek anahtar üretimi yapılır.

Açık anahtarın gizliliği gerekmemekle birlikte kullanılabilmesi için geçerliliğinin devamından emin olmak gerekir. Yani A, B'ye yalnız B'nin açabileceği gizli bir mesaj göndermek istediğinde Şekil 3.3'de görüldüğü gibi B'nin açık anahtarı ile mesajı şifreler. ($E_{K1}(m) = c$) Bu durumda mesajı açanın yalnız B olabilmesi için B'nin gizli anahtarının herhangi başka bir kişinin eline geçmiş olmaması gerekir. B mesajı kendi gizli anahtarını kullanarak açar. ($E_{K2}(c) = m$). Açık anahtarın geçerliliği, ona bağlı gizli anahtarın güvenliğinin devam edip etmediğine bağlıdır.



Şekil 3.3. Açık anahtar şifreleme

Açık anahtar şifrelemenin 1976'da Whitfield Diffie ve Martin Hellman tarafından ortaya atılmasından [12] bu güne kadar pek çok açık anahtar şifreleme sistemi geliştirilmiştir. Tüm bu sistemlerin güvenliği matematiksel bir problemin çözümünün zorluğuna dayanır.

Bugüne değin geliştirilen pek çok açık anahtar şifreleme sistemi kırılmıştır veya uygulamasında yaşanan problemlerden dolayı geçerliliğini yitirmiştir. Bunlar arasında bugün yalnızca üç sistem güvenlik ve verimlilik açısından varlığını sürdürmektedir. Bu sistemler üç temel matematiksel probleme dayanmaktadır.

1. **Çarpanlara Ayırma Problemi (Integer Factorization Problem –IFP) :** RSA ve Rabin Williams.

2. **Ayrık Logaritma Problemi (Discrete Logarithm Problem – DLP) :** Sayısal İmza Algoritması (DSA), Diffie-Hellman anahtar ortaklaştırma düzeni, ElGamal şifreleme ve imza düzenleri, Schnorr imza düzeni ve Nyberg-Rueppel imzalama düzeni.

3. **Eliptik Eğri Ayrık Logaritma Problemi: (Elliptc Curve Discrete Logarithm Problem-ECDLP) :** DSA'nın Eliptik Eğri benzeşimi (ECDSA) ve Diffie-Hellman anahtar ortaklaştırma düzeni, ELGamal şifreleme ve imza düzenleri, Schnorr imza düzeni ve Nyberg-Ruppel imza düzenlerinin Eliptik Eğri benzeşimleri.

Bu problemlerin hiç birinin de çözülemez olduğu ispatlanmamakla birlikte bugüne kadar matematik ve bilgisayar bilimleri alanında yapılan tüm çalışmalara karşın problemlerin çözümü için verimli algoritmaların geliştirilmesi başarılamamıştır.

3.2.1. Çarpanlara ayırma problemi (IFP)

N tam sayısı iki büyük sayının çarpımından oluşsun:

$$p.q = n$$

N'den yola çıkarak iki büyük asal sayının bulunması problemine Çarpanlara Ayırma Problemi denir. Rivest Shamir ve Adleman RSA asimetrik şifreleme sistemini 1978'de bu problemin zorluğuna dayanarak geliştirmişlerdir. Aynı probleme dayalı bir diğer şifreleme sistemini ise Rabin ve Williams geliştirmiştir. Çarpanlara ayırma problemi Fermat ve Gauss gibi matematikçilerin de ilgisini çekmesine karşın çözümü konusunda mesafe kat edilmesi ancak geçtiğimiz 20 yıl içindedir. Bunun iki temel nedeni olmuştur. RSA şifreleme sisteminin 1978'de geliştirilmesi pek çok matematikçinin ilgisini bu problem üzerine çekmiştir. İkincisi karmaşık algoritmaların uygulanması ve test edilmesi için yüksek hızlı bilgisayarlar geliştirilmiştir.

Özel amaçlı ve genel amaçlı olmak üzere iki tür çarpanlara ayırma algoritması vardır. Özel amaçlı olanlar çarpanlara ayrılacak n sayısının özelliklerinden yararlanırlar. Genel amaçlı algoritmaların çalışma zamanı ise yalnızca n sayısının uzunluğuna bağlıdır.

En güçlü özel amaçlı çarpanlara ayırma algoritması Eliptic Eğri çarpanlara ayırma yönetimidir. (ECM) 1985'te Hendrik Lendstra Jr. tarafından geliştirilmiştir. [13] Bu metodun çalışma zamanı n sayısını oluşturan asal sayıların uzunluğuna

bağlıdır. Bu nedenle algoritma öncelikle daha küçük olan çarpanları bulmaya yönelimlidir.

Bu güne kadar ECM kullanılarak elde edilen en büyük asal çarpan 54 basamaklı (180 bit) olup, 127 basamaklı (422 bit) bir sayının asal çarpanıdır. RSA şifreleme sisteminin geliştirilmesinden önce o güne kadar geliştirilmiş en iyi çarpanlara ayırma algoritması süreğen kesir algoritmasıdır. (continued fraction algorithm) 40 ondalıklı sayıları çarpanlara ayırabilmektedir [14]. Bu algoritma, asalların faktör tabanını (factor base) kullanarak ilintili doğrusal denklem kümesini üretmeye dayanır. **Quadratic Sieve (QS)** [15] ve **Number Field Sieve (NFS)** gibi bugün en çok kullanılan algoritmalar da aynı fikirden esinlenmişlerdir. Bu algoritmaların her ikisi de paralel işlenebilir. Bunun anlamı çarpanlara ayırma işleminin, dağıtılmış ağlardaki iş makinalarında paralel olarak yürütülebilmesidir. Bu da büyük sayıların asal çarpanlara ayrılması için süper bilgisayar gerekliliğini ortadan kaldırır.

Bu problem üzerindeki çalışmaların sonuçları değerlendirildiğinde bugün modülo n'de 512 bitlik bir anahtar RSA şifreleme sisteminde kullanıldığında düşük düzeyde bir güvenlik sağlamanın ötesine geçemez. Daha uzun süreli bir güvenlik için en az 1024 bitlik anahtar kullanılmalıdır.

3.2.1.1. Asal sayıların seçimi

Asal sayılar n = p.q'nun çarpanlarına ayrılması için özel bir saldırı tehlikesi yaratmayacak şekilde seçilmelidir.

Eliptik Eğri çarpanlara ayırma algoritmasına olanak vermemek amacıyla p ve q'nun öncelikle uzunluğunun aynı veya birbirine çok yakın olması ve bu iki sayının yeterince büyük olmaları tercih edilmelidir.

Bir diğer kısıt, p-q farkının çok küçük olmasının yaratacağı güvenlik problemidir. (p-q)'nun çok küçük olduğu durumda

$$p \cong q$$

ve bu nedenle de $p \cong \sqrt{n}$ olur.

Buradan $\sqrt{n}$'e yakın olan tüm tek sayıları deneyerek bölme yöntemiyle n'in çarpanlarını bulmak mümkün olur.

Bu kısıtlara ek olarak pek çok yerde p ve q'nun güçlü asallar olması gerektiğinden söz edilmektedir. Bir asal sayının güçlü asal olması için aşağıdaki üç koşulu sağlaması gerekmektedir.

1. (p-1)'in bir büyük asal çarpanı, r vardır.
2. (p+1)'in bir büyük asal çarpanı vardır.
3. (r-1)'in bir büyük asal çarpanı vardır.

Güçlü asal sayı üretmek için asal sayı üretme algoritmaları mevcuttur.

3.2.1.2. RSA şifreleme algoritması

RSA adını fikrin yaratıcıları olan Ron Rivest, Di Shamir ve Leonard Adleman'dan almıştır. Şifreleme algoritması aynı zamanda sayısal imza üretmek amacıyla da kullanılır. Sayısal imzalara Bölüm 5'te ayrıntılı olarak değinilecektir.

RSA güvenliği, IFP'ye dayanır. Gizli anahtar ve açık anahtar bir çift büyük (100, 200 basamaklı) asal sayının fonksiyonlarından oluşmaktadır.

Anahtar çiftinin üretimi için iki rastgele asal sayı seçilir (p ve q). Güvenlik açısından seçilen sayıların birbirleriyle dengeli olmasında yarar vardır. n çarpımı hesaplanır. (n = p.q)

Rastgele bir e sayısı seçilir öyle ki (p-1) (q-1) çarpımı ile e sayısı birbirlerine göre asal olmalıdırlar.

$$e.d \equiv 1 \bmod (p-1)(q-1) \qquad (3.1)$$

olacak şekilde d sayısı hesaplanır. En büyük ortak bölen hesabı yapmak için genişletilmiş Euclid algoritması kullanılabilir.

gcd(d,n) = 1'dir. Burada e ve n sayıları açık anahtarı, d gizli anahtarı oluşturur. Bu işlemlerden sonra p ve q sayılarına şifreleme sisteminin işlemesi açısından bir daha gereksinim duyulmayacaktır. Bu nedenle gizli kalmaları gereken p ve q sayılarının saklanmalarına gerek duyulmaz.

m mesajının şifrelenmesi için önce n'de küçük bloklara bölünür. Şifrelenmiş mesaj c ile gösterilecek olursa şifreleme fonksiyonu aşağıdaki gibidir:

$$c_i = (m_i) \text{ modn} \qquad (3.2)$$

c_i mesajını çözmek için ise kullanılacak fonksiyon:

$$m_i = (c_i)^d \text{ modn} \qquad ....(3.3)$$

biçimindedir.3.3 no'lu denklemde c_i yerine 3.2 no'lu denklem konulacak olursa,

modn'de : $m_i = (m_i)^{k(p-1)(q-1)+1} = m_i\, m_i^{k(p-1)(q-1)} = m_i\, m_i^0 = m_i .1 = m_i$

bulunur.

Donanım olarak gerçeklendiğinde RSA, DES'ten 1000 kat, yazılım olarak ise 100 kat daha yavaştır [16]. Modülün büyüklüğünün iki katına çıkarılması açık anahtar işlemlerinin (şifreleme ya da imzanın doğrulanması) süresini ortalama dört katına, gizli anahtar işlemlerinin (deşifre etme ve imza üretimi) süresini ise ortalama 8 katına çıkaracaktır. Açık anahtar işlemlerinin modül uzunluğunun artırılmasından gizli anahtar işlemlerinden daha az etkilenmesinin sebebi modül büyütürken açık anahtarın sabit kalması ve gizli anahtarın aynı oranda büyümesidir. Ayrıca modülün iki kat artırılması anahtar üretim süresinin 16 kat arttırılması anlamına gelmektedir.

3.2.1.3. Rabin şifreleme algoritması

Rabin güvenliği kanıtlanmış ilk açık anahtar şifreleme sistemidir. Bunun anlamı, Rabin açık anahtar şifreleme algoritmasının kırılması çözümü zor bir matematiksel problemin çözümü ile eşdeğerdir. RSA şifreleme düzeninde ise bu eşdeğerlik kanıtlanmış değildi. Başka bir problemin çözümü üzerinden RSA şifreleme düzeninin kırılıp kırılmayacağı belirsiz olmakla birlikte bugüne kadar böyle bir yöntem bulunmuş olmadığından bu eşdeğerlik üzerinden güvenlik derecesi belirleniyordu. Rabin şifreleme düzeninde ise bu eşdeğerlik kanıtlanmıştır.

Rabin şifreleme düzeninde tarafların her biri kendi açık – gizli anahtar çiftini üretir. Anahtar üretimi için aşağıdaki işlemler takip edilir.

1. Uzunlukları birbirine yaklaşık olan iki büyük rastgele asal sayı (p,q) üretilir.
2. n = p.q hesaplanır.
3. Açık anahtar n; gizli anahtar ise (p,q) dur.

Bir m mesajının açık anahtar ile kapatılması aşağıdaki şekilde gerçekleşir.

1. Mesaj {0,1,…, n-1} kümesi içinde bir tam sayı ile gösterilir.
2. $C = M^2 \bmod n$

hesaplanır.

Şifrelenmiş mesajın (C) gizli anahtar ile açılması işlemi için çinli kalanlar teoreminden yararlanılmaktadır. Alıcı p,q çiftinden oluşan gizli anahtarı kullanarak aşağıdaki işlemleri yerine getirir.

$m_1 = C^{(p+1)/4} \bmod p$

$m_2 = (p - C^{(p+1)/4}) \bmod p$

$m_3 = C^{(q+1)/4} \bmod q$

$m_4 = (q - C^{(q+1)/4}) \bmod q$

m_1, m_2, m_3, m_4 hesaplandıktan sonra a ve b tam sayıları $ap + bq = 1$ olacak biçimde aşağıdaki şekilde hesaplanır:

$a = q\,(q^{-1} \bmod p)$, $b = p\,(p^{-1} \bmod q)$

M için dört olası çözüm vardır:

$M_1 = (am_1 + bm_3) \bmod n$

$M_2 = (am_1 + bm_4) \bmod n$

$M_3 = (am_2 + bm_3) \bmod n$

$M_4 = (am_2 + bm_4) \bmod n$

Eğer mesaj anlamlı sekizlilerden oluşuyorsa hangisinin M olduğuna karar verilebilir. Fakat mesaj rastgele bir bit dizisi ise karar vermek için ek yöntemler geliştirilmelidir. Bu problemi çözmenin bir yolu mesajın başına her iki tarafın da bildiği bir başlık bilgisi eklemektir. Hugh Williams, 1980'de Rabin şifreleme düzeninin bu eksikliğini tamamlayıcı bir yaklaşım önermiştir. [17] Ardından 1985 ve 1986'da geliştirdiği yaklaşımlarla Rabin-Willams şifreleme düzeni son halini almıştır.

3.2.2. Ayrık logaritma problemi (DLP)

Modüler aritmetiğin bir matematiksel problemi de ayrık logaritma problemidir. p belirli bir asal sayı; Z_p toplama ve çarpma işlemlerinin p modülüne göre tanımlandığı, 0 ile p-1 arasında bir tamsayı kümesi olmak üzere: Sıfıra eşit olmayan öyle bir $\alpha \in Z_p$ vardır ki α'nın p modülüne göre üsleri alınarak kümenin tüm elemanları üretilebilir. Burada α'ya Z_p'nin üretici denir.

Ayrık logaritma problemi, bir p asal sayısı ve Z_p'nin üreteci α ve sıfıra eşit olmayan bir $\beta \in Z_p$ verildiğinde,

$\beta \equiv \alpha^l \pmod{p}$ ve $o \leq l \leq p-2$

olmak üzere, l'nin bulunması problemidir. burada l tamsayısına, β'nın α tabanına göre ayrık logaritması denir.

Daha basit bir tanımlama yapılacak olursa, $y = g^x \pmod{p}$ olacak şekilde p,g ve y'nin bilindiği durumda x'in elde edilmesi problemi "p modülünde ayrık logaritma problemi" dir.

Bu problemin zorluğuna dayanarak, Diffie ve Hellman, bilinen "anahtar ortaklaştırma düzeni"ni 1976'da önerdiler [12]. Hemen ardından DLP'ye dayanarak diğer kriptografik protokol önerileri ortaya çıktı: Sayısal imza algoritması DSA [18], ElGamal şifreleme ve imza düzenleri [19], Schnorr imza düzeni [20] ve Nyberg-Rueppel imza düzeni [21].

DLP'de de çözüm için geliştirilmiş algoritmalar iki grupta toplanabilir. Özel amaçlı algoritmalar, p asal sayısının kendinden kaynaklanan özelliklerine yönelik geliştirilirken genel amaçlı algoritmalar, yalnızca p asal sayısının büyüklüğüne bağlıdır.

DLP'nin çözümü için bilinen en hızlı genel amaçlı algoritma, **index calculus** adı verilen bir yönteme dayanır. Bu yöntemde cisim elemanlarının logaritmalarının ardından kolayca elde edilebileceği biçiminde küçük asal sayıların ve onlara ilişkin logaritmaların veri tabanı üretilir. Diğerinde olduğu gibi paralel işlenebilir bir algoritmadır.

Çarpanlara ayırma probleminde olduğu gibi burada da DLP için en çok kullanılan algoritma **number field sieve**' dir[22]. Çalışma süresi çarpanlara ayırma problemi üzerine olan algoritma ile benzerdir. DLP'de de güvenliği uzun süreli korumak için 1024 bit ve daha büyük p modülü üzerinde çalışılmalıdır. Güvenlik, p modülünde ayrık logaritma problemine dayanır. Verimlilik ise p modülünde üs alma işleminin hızına dayanır.

3.2.2.1. ElGamal şifreleme düzeni

Elgamal şifreleme düzeninin güvenliği, bir sonlu cisim üzerinde ayrık logaritma probleminin çözümüne dayanmaktadır.

Anahtar çiftinin üretimi için öncelikle bir p asal sayısı ve p modülündeki tam sayıların çarpımsal grubu Z_p^* 'nin bir üreteci olacak şekilde α sayısı seçilir.

$1 \leq a \leq p-2$ olmak üzere bir rastgele sayı a seçilir ve

$y = \alpha^a \bmod p$

hesaplanır. Açık anahtar y, α ve p'den oluşmaktadır. Gizli anahtar ise a'dır.

Mesajın açık anahtar ile şifrelenerek C mesajının elde edilmesi aşağıdaki gibidir:

1. M mesajı $\{0, \ldots, p-1\}$ aralığında olmalıdır.
2. $1 \leq k \leq p-2$) olacak şekilde rastgele bir sayı k belirlenir.

3. $\gamma = \alpha^k \bmod p$ ve $\delta = m \,.\, (\alpha^a)^k$ olmak üzere $C = (\gamma,\delta)$ hesaplanır.

C'nin gizli anahtar a ile açılarak M'nin elde edilmesi için aşağıdaki işlemler yapılır:

1. $\gamma^{p-1-a} \bmod p$ hesaplanır. ($\gamma^{p-1-a} = \gamma^{-a} = \alpha^{-ak}$ eşitliğinden bulunur.)

2. $m = (\gamma^{-a}).\ \delta \bmod p$

Bu eşitliğin nedeni şuradan görülebilir:

$(\gamma^{-a}).\ \delta \bmod p = \alpha^{-ak} \,.\, m \,.\, (\alpha^a)^k \bmod p = m$

Burada açık anahtar üretecek tarafların tümü de aynı asal sayıyı ve aynı üreteç α'yı seçebilirler. p ve α'nın önceden haberleşen taraflar arasında ortaklaştırıldığı bu durumda dağıtılması gereken açık anahtarın boyu kısalmış olur. Bu parametrelerden α'nın sabitlenmesinin bir başka olumlu yanı ise üs alma işleminin önceden hesaplanabilmesi ve buradan kazanılan zamandır. ElGamal şifreleme sisteminin bir olumsuzluğu ise şifreli mesaj C'nin uzunluğunun m'nin iki katı olmasıdır.

EIGamal şifreleme sisteminin ayrık logaritma problemine dayandığı doğru olmakla birlikte sistemin çözümünün tek yolunun DLP'nin çözümü ile eşdeğer olup olmadığı henüz ispatlanmış değildir. Bugüne kadar bir başka genel amaçlı saldırı yöntemi bulunamamış olması nedeniyle güvenli olduğu varsayılmaktadır.

Parametrelerin ortak kullanıldığı durumda ELGamal şifreleme düzeninde dikkat çekilmesi gereken önemli bir nokta mesaj şifrelenirken üretilen rastgele sayının başka bir şifreleme işleminde kullanılmamasıdır. Örneğin aynı rastgele sayının (k), m_1 ve m_2 mesajlarının şifrelenmesinde kullanıldığı varsayılırsa, üretilen şifreli mesajlar (γ_1, δ_1) ve (γ_2, δ_2) olacaktır.

$\delta_1 / \delta_2 = m_1 / m_2$

Eşitliğinden mesajlardan herhangi biri bilindiğinde diğeri hesaplanabilir.

3.2.2.2. Genelleştirilmiş ElGamal şifreleme düzeni

Yukarıda, Z_p^* çarpımsal grubunda tanımlanmış şifreleme düzeninin genelleştirilmiş hali, herhangi bir sonlu çevrimsel grup G üzerinde de çalışabilmektedir.

Güvenlik burada da G grubu üzerinde tanımlı ayrık logaritma problemine dayanmaktadır. G grubunun aşağıdaki koşulları sağlayacak biçimde seçilmesi gerekmektedir:

1. Verimliliği sağlamak için, G'deki grup işleminin uygulanması kolay olmalıdır.
2. Güvenlik açısından, G grubundaki DLP'nin hesaplama yoluyla çözümü mümkün olmamalıdır.

Aşağıda, bu iki özelliği sağlayan grupların listesi bulunmaktadır. Bunlardan en çok karşılaşılanları ilk üçüdür.

1. Bir p asal sayısının modülünde tam sayılardan oluşan Z_p^* çarpımsal grubu.
2. İki karakteristiğinde bir sonlu cisim F_{2m}'nin çarpımsal grubu F_{2m}^*
3. Bir sonlu cisimde tanımlanan eliptik eğri üzerindeki noktaların oluşturdukları grup.
4. P asal ve $q = p^m$ olmak üzere F_q sonlu cisminin çarpımsal grubu F_q^*
5. n bileşik bir tamsayı olmak üzere Z_n^* birimlerinin grubu.
6. Bir sonlu cisim üzerinde tanımlı hipereliptik eğrinin jakobyeni.
7. Sanal ikili sayı cisminin sınıf grubu

Genelleştirilmiş ELGamal şifreleme düzeninde anahtar üretimi için aşağıdaki işlemler yapılır:

1. Sırası n olan, α üretecine sahip bir çevrimsel grup seçilir.
2. $1 \leq a \leq n-1$ olacak şekilde rastgele bir a tamsayısı seçilerek grup elemanı

$y = \alpha^a$ hesaplanır.

G grubunda elemanların çarpım işleminin tanım bilgisi ile birlikte açık anahtar (α,y); gizli anahtar ise a'dır.

Açık anahtar ile mesajın şifrelenmesi aşağıdaki şekilde olur:

1. Rastgele bir sayı, k belirlenir. $(1 \leq k \leq n-1)$. gcd (k, n) = 1 olmalıdır.

2. $\gamma = \alpha^k$ ve $\delta = m.(\alpha^a)^k$ olmak üzere $C = (\gamma,\delta)$ hesaplanır.

C'nin gizli anahtar a ile açılarak m'nin elde edilmesi için aşağıdaki işlemler yapılır:

1. γ^a ve γ^{-a} hesaplanır.

2. $m = (\gamma^{-a}).\ \delta$

3.2.2.3. DSA

DSA, ABD Ulusal Standartlar ve Teknoloji Enstitüsü'nün 1991'de geliştirmiş olduğu bir sayısal imza algoritması olup ABD Federal Bilgi İşleme Standardı (FIPS 186) olarak kabul edilmiştir. Sayısal İmza Standardı olarak bir devlet tarafından kabul edilen ilk algoritmadır. ElGamal sayısal imza düzeninin bir varyasyonu olup ekli bir sayısal imza düzenidir. Mesaj çözmeli mekanizmalara uygun değildir. (Sayısal imzalarla ilgili Bölüm 5'e bakınız.)

Algoritmanın güvenliği bir sonlu küme üzerinde ayrık logaritma probleminin çözümüne dayanır. Bir çırpma fonksiyonu kullanılır.

$h: \{0,1\}^* \rightarrow Z_q$ (q bir tamsayıdır)

Anahtar üretimi için aşağıdaki işlemler yapılır:

1. $2^{159} < q < 2^{160}$ olacak biçimde bir q asal sayısı seçilir.

2. $0 \leq t \leq 8$ olacak biçimde bir t sayısı seçildikten sonra $2^{511+64t} < p < 2^{512+64t}$ ve q böler (p-1) koşuluyla bir p asal sayısı seçilir.

3. Z_p^* üzerinde q sıralı bir tekil çevrimsel grubun üretici α belirlenir.

4. $1 \leq a \leq q-1$ olacak şekilde bir a tamsayısı belirlenir.

5. $y = \alpha^a \bmod p$ hesaplanır.

6. Açık anahtar (p, q, α, y); gizli anahtar a'dır.

DSA imzalama işlemi aşağıdaki adımların uygulanmasıyla yerine getirilir:

1. Gizli tutulmak koşuluyla rastgele bir k tamsayısı seçilir. (0<k<q).

2. $r = (\alpha^a \bmod p) \bmod q$ hesaplanır.

3. $k^{-1} \bmod q$ hesaplanır.

4. $s = k^{-1} \{h(m) + ar\} \bmod q$ hesaplanır. (h(m), mesajın çırpma fonksiyonudur.)

5. (r, s) ikilisi, m mesajı için imzayı oluşturur.

DSA ile A'nın imzaladığı m mesajı üzerindeki (r, s) imzasını doğrulamak için B aşağıdaki adımları yerine getirir:

1. A'nın açık anahtarı (p,q, α, y) asıl olduğundan emin olunacak biçimde elde edilir.

2. r ve q için, 0<r<q olduğu; s ve q sayıları için 0<s<q olduğu kontrol edilir. Eğer eşitsizlikler sağlanmıyorsa imza reddedilir.

3. $w = s^{-1} \bmod q$ ve h (m) hesaplanır.

4. $u_1 = w.\ h\ (m) \bmod q$ ve $u_2 = r.w \bmod q$ hesaplanır.

5. $v = (\alpha^{u1}\ y^{u2} \bmod p) \bmod q$ hesaplanır.

6. v = r eşitliği saplanıyorsa imza doğrulanır, aksi durumda reddedilir.

İmza doğrulama işleminin kanıtı aşağıdaki şekilde yapılabilir:

1. Eğer (r, s) ikilisi m mesajı üzerinde gerçekten A'nın imzası ise o zaman

h (m) ≡ - ar + ks (mod q) denkligini sağlaması gerekir.

2. Denkliğin her iki tarafı w ile çarpılırsa ve yeniden düzenlenirse

$$w.\ h\ (m) + arw \equiv k\ (mod\ q)$$

denkliği elde edilir. Fakat bu denklik basit olarak aslında

$$u_1 + a\ u_2 \equiv k\ (mod\ q)$$

denkliğine eşittir. α denklemin her iki tarafına aşağıdaki biçimde katılırsa

$$(\alpha^{u1 + a\,u2}\ modp)modq \equiv (\alpha^{k} modp)modq \Rightarrow$$

$$(\alpha^{u1}.y^{u2}\ modp)modq \equiv (\alpha^{k} modp)modq$$

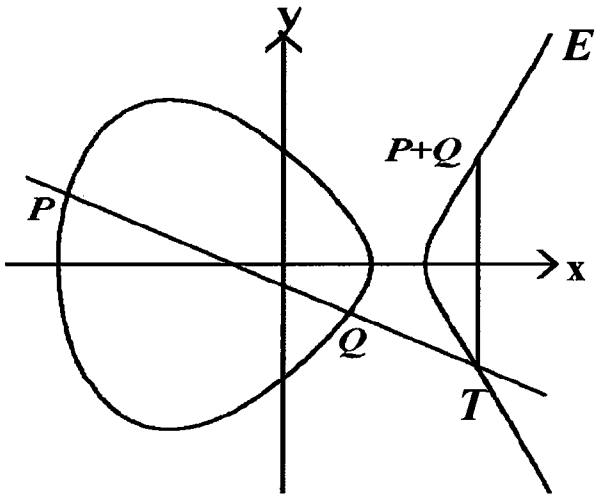
bulunur. Elde edilen denklikten v = r olduğu görülebilir.

3.2.3. Eliptik Eğri Ayrık Logaritma Problemi (ECDLP)

ECDLP, ayrık logaritma probleminin bir eliptik eğri üzerindeki noktalar aracılığıyla tanımlanmış halidir. p asal modülünde tanımlı bir eliptik eğri aşağıdaki formda bir denklemin a ve b sayıları için (x,y) çözüm kümesidir.

$y^2 = x^3+ax+b\ (modp)$

Denklemi sağlayan p(x,y) noktaları eliptik eğriyi oluşturan noktalardır. Eliptik eğri üzerindeki noktalar arasında kapalılık özelliğini sağlayan bir "toplama" işlemi tanımlamak mümkündür. Eliptik eğri üzerindeki noktaların toplanması işlemi birkaç modüler hesaplama ile yapılabilir. Şekil 3.4'te bir E eliptik eğrisi üzerinde P ve Q noktasının toplanmasının geometrik ifadesi gösterilmektedir. Geometrik olarak toplama sonucu P ve Q noktalarından geçen doğrunun eğriyi kestiği bir üçüncü noktanın, şekilde T noktası, x eksenine göre simetriği alınarak bulunan noktaya eşittir.



Şekil 3.4. E eliptik eğrisi üzerindeki Pve Q noktalarının toplanması

ECDLP'de de IFP ve DLP'de olduğu gibi oluşturulacak kriptografik sistemin verimliliği modüler aritmetiğe dayalıdır. q bir asal sayının kuvveti ve F_q, q elemanlı bir sonlu cisim olsun. Uygulamalarda tipik olarak q, 2'nin kuvveti ($q = 2^m$) veya tek bir asal sayı p'ye eşittir. F_q cismi üzerinde tanımlı bir eliptik eğri (E) verildiğinde, eğri üzerinde n dereceli bir p noktası $P \in E(F_q)$ ve q noktası $Q \in E(F_q)$, $0 \leq x \leq n-1$ olmak üzere bir x tamsayısı belirlerler, öyle ki: Q = x..P'dir.

Burada x.P işlemi, P noktasının kendi kendisi ile x defa toplanması anlamına gelen skaler çarpma işlemini ifade eder. **Eliptik Eğri Ayrık Logaritma Problemi**, P ve Q noktasına dayanarak x'in bulunması problemidir. Bu problemin zorluğuna dayanarak Neal Koblitz [23] ve Victor Miller [24], birbirlerinden bağımsız olarak yürüttükleri çalışmaları sonucunda 1985'te bir sonlu cisim üzerinde tanımlanmış bir eliptik eğri üzerindeki bir grup noktayı kullanarak değişik ayrık logaritma kriptosistemleri üretmişlerdir. Yukarıda tanımlı DLP üzerine kurulu kriptografik algoritmaların Eliptik Eğri benzeşimleri mevcuttur. DSA'nn Eliptik Eğri benzeşimi olan ve bugün bir standart olarak kabul edilen ECDSA bunlardan birisidir. 1985'ten bu yana ECDLP'yi çözmeye dönük matematiksel ilerlemeler kaydedilmiştir. En genel yöntem Pollard rho metodudur.

4. ŞİFRELEMENİN TEMEL ELEMANLARI

Bu bölümde bir güvenlik protokolü tasarımına temel oluşturacak protokollerde kullanılan ilkeller ve birbirleriyle ilişkileri karşılaştırmalı olarak ele alınacaktır. Bu ilkellerin temel işleyiş mantıkları ve nerede hangi amaçla kullanıldıkları ele alınacaktır.

4.1. Çırpma fonksiyonları

Sıkıştırma fonksiyonları, mesaj özeti, parmak izi, mesaj bütünlük kontrolü (MIC), ve müdahaleyi farketme kodu (MDC) gibi başka pek çok isimle adlandırılmaktadırlar. Bir çırpma fonksiyonu değişken uzunlukta bir giriş değeri alır ve sabit uzunlukta bir çıkış oluşturur. Genellikle çıkışın uzunluğu daha küçüktür.

En temel özelliği giren mesajın bir çeşit parmak izini oluşturmasıdır. Bir diğer özelliği ise iki farklı verinin çırpması alındığında üretilen sonucun aynı olmamasıdır. Çırpma fonksiyonları açıktır. Herhangi bir gizliliğe yer verilmez. Çırpma fonksiyonunun güvenliği tek yönlü fonksiyon seçiminden gelir. İyi bir çırpma fonksiyonunda giriş değerindeki bir bitlik değişiklik çıkış değerine yaklaşık verinin yarı uzunluğunda bit sayısının değişikliğe uğraması olarak yansıyabilmelidir, ayrıca herhangi farklı iki giriş değeri için aynı çıkış değeri üretilememelidir. Bir protokolde çırpma fonksiyonlarının kullanımı bu sayılan özellikleri üzerinden gerçekleşmektedir.

Çırpma fonksiyonunun kullanıldığı uygulamalar aşağıdaki şekilde sıralanabilir:

1. Bilginin doğrulanması-pekiştirilmesi: Burada kullanılan çırpma fonksiyonu, verinin kendisini açıklamadan bir veri değeri üzerinde taahhütte bulunmak veya bir veriye sahip olduğunu karşı tarafa göstermek için kullanılır. Örneğin gizli anahtar bilgisine sahip olduğunu karşı tarafa göstermek için anahtarın kendisini yollayarak güvenliği tehlikeye atmak yerine anahtarın çırpma değerini karşı tarafa

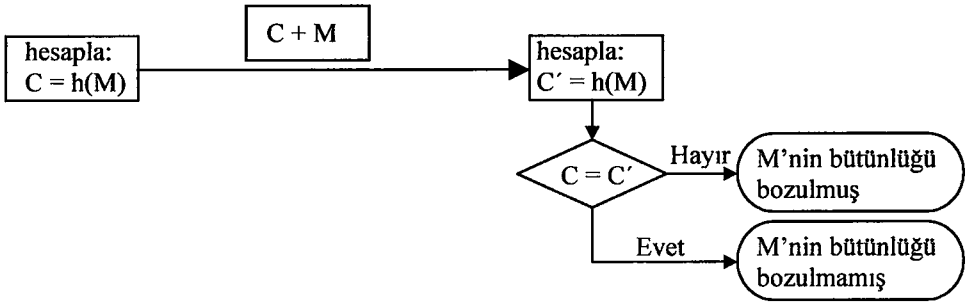
yollayarak aynı amaca ulaşmak mümkündür. Böylece çırpma fonksiyonu ile bir asıllama işlemi gerçekleştirilmiş olur. Bu anlamda diğer kullanım alanları, simetrik anahtar sayısal imza mekanizmaları, asıllamalı anahtar kurulum protokollerinde anahtar bilgisinin doğrulanması ve doküman tarihleme veya zaman pulu olarak çırpma değerinin saklanmasıdır.

2. Anahtar türetme: Giriş parametresinin uzunluğu ne olursa olsun çıkış parametresi olacak kriptografik anahtarların sabit uzunlukta olması, her giriş parametresinin ürettiği çıkış değerinin farklı olmasına duyulan ihtiyaç, üretici fonksiyon olarak çırpma fonksiyonlarının kullanımını getirmiştir.

3. Rastgele sayı üretimi: İyi bir çırpma fonksiyonunun özelliği, her giriş için farklı bir çıkış değeri üretilmesinin yanında giriş parametrelerinde meydana gelecek tek bir bit değişiminin çıkışı değerini yüksek oranda etkilemesidir. Bu nedenle bu özellikleri güçlü olan çırpma fonksiyonlarının rastgele sayı üretiminde kullanılması uygun olmaktadır.

4.1.1. Mesaj bütünlüğünün sağlanmasında çırpma fonksiyonu

Üretilen çırpma fonksiyonu çıkışına MDC (Müdahaleyi farketme kodu) denir. Mesaj bütünlük kontrol kodu (MIC) olarak da adlandırılır. Alınan mesajın bütünlüğünün bozulup bozulmadığını, değişikliğe uğrayıp uğramadığını kontrol etmek için kullanılır. Bunun için taraflar arasında önceden kararlaştırılan çırpma fonksiyonu kullanılarak mesajın çırpma değeri hesaplanır ve mesaja eklenir. Alıcı taraf mesajı elde edince çırpma değerini hesaplar, bu değeri gönderilenle karşılaştırır. Aynı sonucu elde ediyorsa, mesajın bütünlüğünün bozulmadığı sonucuna varabilir. MDC, mesajı gönderenin kimliği konusunda alıcıya bir fikir vermez.



Şekil 4.1. Mesaj bütünlüğü testi

Şekil 4.1'deki örnekte araya giren bir kişi eğer h çırpma fonksiyonunu biliyorsa, (C + M) veri paketi de güvenli bir kanaldan değil açık olarak karşı tarafa gönderiliyorsa M'yi başka bir mesajla (M′) değiştirip kendi hesapladığı çırpma değerini ekleyerek yeni bir mesaj paketi yaratabilir ve bu paket de bütünlük testinden başarıyla geçer. MDC'nin bu nedenle güvenlik protokollerinde tek başına kullanılması uygun değildir. Kimlik doğrulama mekanizmalarıyla veya iki taraf arasında yaratılan bir güvenli kanalla desteklenmelidir.

4.2. Simetrik ve açık anahtar şifrelemenin kullanımı

Bir verinin karşı tarafa gizli bir şekilde gönderilmesi istendiğinde açık anahtar şifreleme tekniklerinin simetrik şifrelemeye göre çok yavaş çalıştıkları ve metnin büyüklüğü gözönüne alınarak genellikle zorunlu olmadıkça açık anahtar şifreleme kullanılmamaktadır. Simetrik şifreleme ile çözülemeyen problemlerin açık anahtar şifreleme ile çözülebildiği görülmüş, tarihsel olarak Difffie-Hellman'ın 1976'da [12] fikri ortaya attığı günden bu yana açık anahtar şifrelemenin gelişimi de bundan dolayı olmuştur.

Açık anahtar şifrelemenin keşfedilmesi, onun gizli anahtar şifrelemenin yerini almasıyla sonuçlanmamıştır. Bunun nedeni her iki şifrelemenin olumlu ve olumsuz yanlarının birbirlerinden farklı olmasıdır.

4.2.1. Açık anahtar şifrelemenin olumlu yanları

1. Sadece kişisel anahtarın gizli tutulması yeterlidir (açık anahtarların asıllaması garanti edilmelidir).

2. Bir ağdaki anahtarların yönetimi, koşulsuz olarak güvenilen bir Güvenilir Üçüncü Taraf (Thrusted Third Party-TTP) yerine, sadece işlemlere bağlı olarak güvenilen bir TTP'nin varlığını şart koşar.

3. Kullanım kipine bağlı olarak bir kişisel anahtar/açık anahtar çifti oldukça uzun süreler boyunca, yani pek çok oturumu kapsayacak şekilde, değişmeden kalabilir.

4. Pek çok açık anahtar düzeni, göreceli olarak verimli sayısal imza mekanizmaları sağlar. Açık doğrulama fonksiyonunu tarif etmek için kullanılan anahtar, simetrik anahtar karşılığına göre oldukça kısadır.

5. Geniş bir ağda gerekli olan anahtar sayısı, simetrik anahtar senaryolarına göre oldukça düşük kalabilir.

4.2.2. Açık anahtar şifrelemede karşılaşılan problemler

1. Açık anahtar algoritmalar simetrik algoritmaların ortalama 1000'de biri kadar yavaş çalışırlar [25]. Bilgisayar hızlarındaki gelişmeler düşünülürse bu durum önemsenmeyebilir fakat bant genişliği gereksinimi de artmaktadır ve her zaman verinin daha hızlı şifrelenmesine ihtiyaç olacaktır.

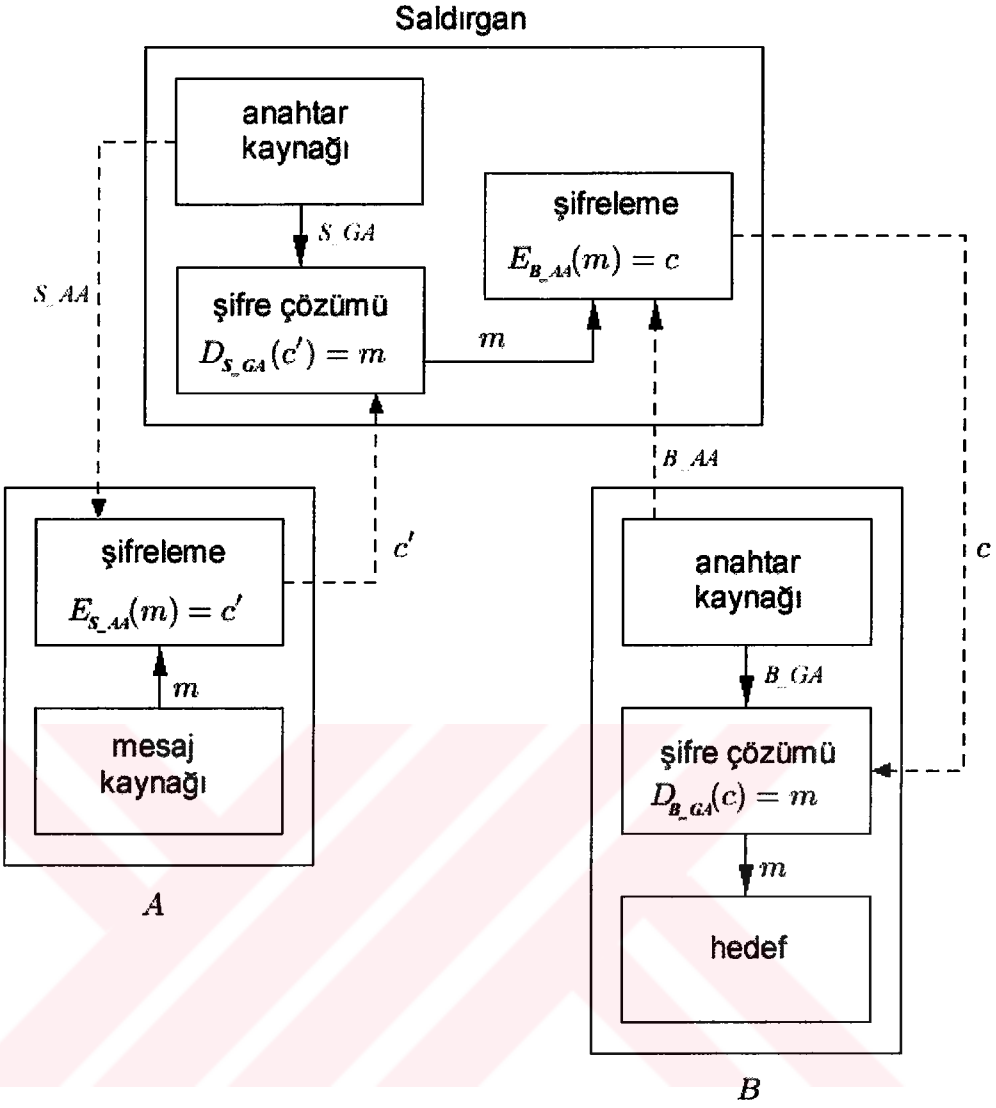
2. Açık anahtar şifreleme, seçilen açık metin saldırılarına açıktır. Eğer C=E(P) iken P, n adet olası açık metinden birisi ise kriptanalizcinin P'yi bulmak için yapması gerekli tek şey en fazla n deneme ile olası P'leri şifreleyerek C ile karşılaştırmak olacaktır. Şifreleme algoritmasının ve anahtarının açık olduğu varsayılır.

Simetrik şifrelemeye göre yavaş olan asimetrik şifreleme büyük verilerin şifrelenmesinde kullanılan simetrik algoritmaların gizli anahtarlarının taraflar arasında gizli olarak paylaşılmasında; veri bütünlüğü ve asıllamayı sağlama amaçlı diğer uygulamalarda ve kredi kart numarası veya PIN gibi kişisel küçük verileri şifrelemede kullanılır.

4.2.2.1. Açık anahtar şifrelemede asıllamanın gerekliliği

Açık anahtar kriptografi, ilk anda şifreleme anahtarını aktarmak için güvenlikli bir kanal gerektirmeyen mükemmel bir sistem gibi görünmektedir. Bunun doğru

olması, iki varlığın, hiç anahtar alışverişinde bulunmadan güvenlikli olmayan bir kanal üzerinden haberleşmelerinin mümkün olduğu anlamına gelir. Fakat Şekil 4.2'de görüldüğü gibi, etkin bir saldırgan, şifrelemeyi kırmaksızın sistemi yenebilir (taraflar arasında gönderilen mesajların şifresini çözebilir). Bu bir çeşit kılık değiştirme olarak nitelendirilir ve bir protokol hatasıdır. Bu kurguda saldırgan, B'nin kimliğine bürünerek A'ya kendi açık anahtarı S_AA'yı gönderir. A, bunu (yanlış bir şekilde) B'nin açık anahtarı zanneder. Saldırgan, A'dan B'ye yollanan mesajları yakalar, bunları şifresini kendi açık anahtarı olan S_GA ile çözer ve mesajları B'nin açık anahtarı B_AA ile tekrardan şifreleyip B'ye iletir. Bu saldırı bizzat açık anahtarların kaynaklarının asıllanmasının gerekliliğini vurgulamaktadır. A, B'nin gerçek açık anahtarıyla şifreleme yaptığından emin olmalıdır. Burada sorunun çözümü için anahtar yönetimi, kurulumu ve sertifikasyon mekanizmalarına başvurulur. (Bakınız Bölüm 5).



Şekil 4.2. Açık anahtar şifrelemeye kılık değiştirme saldırısı

4.2.3. Simetrik kriptosistemlerin olumlu yanları

1. Simetrik anahtar şifreleme, yüksek miktarlarda veri akışına sahip olacak şekilde tasarlanabilir. Bazı donanım uygulamalarında, saniyede yüzlerce megasekizli şifreleme hızlarına ulaşılabilirken, yazılımla yapılan uygulamaların performansı saniyede birkaç megasekizli ölçeğinde kalmaktadır.

2. Simetrik şifreleme anahtarları göreceli olarak daha kısadırlar.

3. Simetrik anahtar şifreler; rastgele sayı üreteçleri, çırpma fonksiyonları ve hesaplama açıdan verimli sayısal imza düzenleri de dahil olmak üzere, değişik kriptografik mekanizmalar inşa etmek için temel yapı taşı olarak kullanılabilir.

4. Simetrik anahtar şifreler, daha güçlü şifreler üretmek üzere tasarlanabilir. Kendi başlarına zayıf ve incelemesi kolay olan dönüşümler, güçlü şifreler yaratmak için kullanılabilir.

5. Simetrik anahtar şifrelemenin uzun bir geçmişi olduğu kabul edilir. Rotorlu makinaların icadını saymazsak, bu konudaki bilginin, sayısal bilgisayarın icadını ve özellikle de 1970'lerin başında Veri Şifreleme Standardı'nın tasarlanmasını takiben edinildiğini kabul etmek gerekiyor.

4.2.4. Simetrik kriptosistemlerde karşılaşılan problemler

1.Anahtar dağıtımı gizli olarak yapılmalıdır.

2.Anahtarın ele geçirilmesi durumunda saldırgan o protokolde anahtarı kullanılarak haberleşen tarafların tüm trafiğini kontrol edebilir. Taraflardan birinin yerine geçebilir.

3.Bir ağ üzerinde bulunan tüm tarafların birbirleriyle güvenli kanaldan konuşabilmeleri için her ikili için ayrı anahtar çiftlerinin tutulduğu düşünülürse bu hesap n kullanıcılı bir ağ üzerinde n(n-1)/2 anahtar anlamına gelir. 10 kullanıcı için 45, 100 kullanıcı için 4950 anahtar gereklidir.

4.2.5. Açık ve simetrik anahtar kriptosistemlerinin kullanımı

Açık anahtar (asimetrik) algoritmalar gerçek hayatta simetrik algoritmaların yerine kullanılmazlar. İkisinin olumlu ve olumsuz yanları birbirini tamamlayan biçimde değişik amaçlara yönelik olarak tercih edilirler. Kredi kart numarası veya PIN gibi kişisel küçük verileri şifrelemede açık anahtar şifreleme algoritmaları tercih edilirken büyük verilerin şifrelenmesinde simetrik algoritmalar tercih edilir. Simetrik algoritmalarda karşılaşılan anahtar dağıtımı problemini çözmek için açık anahtar şifreleme kullanılması oldukça yaygındır. Veri bütünlüğü, asıllamayı

sağlama ve sayısal imza sistemleri gibi diğer uygulamalarda da asimetrik şifreleme algoritmaları yaygın olarak tercih edilmektedir.

4.2.5.1. Oturum anahtarı kullanımı

Oturum anahtarı, tek bir oturum için üretilen gizli anahtardır. Bir terminalin birden fazla terminalle konuştuğu durumda, sunucu ve istemci arası protokolde olduğu gibi bir terminalde çok sayıda gizli anahtarın saklanması zorunluluğunu ortadan kaldırmak için kullanılır.

4.2.5.2. Melez kriptosistemler

Simetrik şifrelemede kullanılacak anahtarın taraflar arasında paylaşımı veya dağıtımı probleminin çözümünde açık anahtar şifreleme kullanılmasına melez kriptosistemler denir. Simetrik şifrelemede tarafların daha önceden bildikleri gizli anahtarlar kullanıldığında bu anahtarların güvenli bir biçimde saklanması, bir zaman sonra anahtarların güvenliğinin tehlikeye girmesi gibi problemlerle karşılaşılacağından her oturum için ayrı bir gizli anahtar üretilmesi ve uzun süreli güvenlik sağlama probleminin de bu şekilde çözülmesi tercih edilir. Bu defa da oturum anahtarının üretildikten sonra paylaşılması problemi ortaya çıkmaktadır. Bu problemin çözümü için aşağıdaki basamaklardan oluşan melez kriptosistem kullanılır.

1. B, A'ya kendi açık anahtarını(B_AA) gönderir.
2. A, rastgele bir oturum anahtarı(K) üretir. K'yı B'nin açık anahtarı ile şifreleyerek B'ye gönderir.

$$E_{B\_AA}(K)$$

3. B, A'dan gelen mesajı kendi gizli anahtarını kullanarak açar ve oturum anahtarını elde eder.

$$D_{B\_GA}\,(E_{B\_AA}(K)) = K$$

A ve B bundan sonraki haberleşmelerini K anahtarı ile simetrik şifreleme kullanarak güvenli bir biçimde yapabilirler. Bu sayede önemli bir anahtar

yönetimi problemi, oturum anahtarının paylaştırılması problemi çözülmüş olur. Açık anahtar şifreleme ile paylaşım problemi çözüldüğünden, simetrik anahtar sadece kullanılacağı zaman üretilerek, işi bitince bir kenara atılabilir.

Yukarıdaki melez kriptosistem, eğer anahtar kurulumu, yönetimi ve sertifikasyon mekanizmaları ile desteklenmez ise araya giren bir saldırganın tüm haberleşmeyi kontrol etmesine ve mesajları başka mesajlarla değiştirmesine olanak verir. Bu durum Bölüm 4.2.2.1'de açıklanmıştır.

5. BİLGİ SİSTEMİ PROTOKOLLERİ

En temel anlamıyla tanımı yapılacak olursa protokol, bir işin tamamlanması amacıyla hareket eden iki veya daha fazla tarafı kapsayan bir basamaklar dizisidir. Bir başlangıcı ve bitişi olan sıralı bir düzeni vardır. Her basamağın sırasıyla işletilmesi gerekir ve hiç bir basamak bir önceki basamak sonlanmadan başlayamaz. Bir protokolün tamamlanması için en az iki tarafın varlığı gereklidir. Bir taraf tek başına bir protokol oluşturamaz. Son olarak bir protokol bir işi gerçeklemek amacıyla yapılır. Bir dizi basamaktan oluşan işlem sonunda hiçbir iş ortaya çıkmıyorsa bu da bir protokol olmaz.

Protokolün temel özellikleri şöyle sıralanabilir:

- Protokolü kullanan taraflar, protokolün hangi işlem basamaklarına dayandığını bilmelidirler.
- Protokolü kullanan taraflar, protokolü oluşturan işlem basamaklarını takip etmeyi kabul etmelidirler.
- Protokolü oluşturan tüm basamakların iyi tanımlanması gereklidir. Basamakların farklı taraflarca farklı anlaşılma ve farklı uygulanma olasılığı ortadan kaldırılmalıdır.
- Protokol tamamlanmış olmalıdır. Tüm olası durumlar için belirlenmiş bir işlem olmalıdır. Herhangi bir olası durum için belirlenmemiş işlem varsa taraflardan birisi o durumla karşılaştığında protokol tamamlanamaz [25].

Protokolün uygulanması, farklı bir basamağa dallanma gösteren komutlarla karşılaştırılmadıkça, basamaklar üzerinden doğrusal olarak devam etmektedir. Her basamak en az iki işlemden birini içerir: Taraflardan biri veya daha fazlası tarafından yürütülen işlemler veya taraflar arasında mesaj gönderilmesi.

Kriptografik protokoller, kriptografik algoritmaların kullanıldığı protokollerdir. Protokolde birbirine güven duyan veya güven duymayan taraflar olabilir. Protokolün tarafları, bir hesap yapmak için birbirlerinin gizli verilerinin bir bölümünü paylaşmak, birlikte rastgele bir dizi üretmek, birbirlerini kimliklerinin doğruluğuna ikna etmek veya yanında bir kontratı imzalamak isteyebilirler. Bir protokolde kriptografi kullanımının amacı dışarıdan veya içeriden kötü niyetli kullanımları engellemektedir.

Bir sistemin güvenliğini sağlamak üzere, sistemin analizi yapıldıktan sonra taraflar arasında güvenli iletişimi sağlamak üzere bir kriptografik protokol tasarlanır. Protokol tasarımında temel iki ilke izlenir:

1. Protokolün veya mekanizmanın tasarımındaki tüm koşulların tanımlanması,
2. Bu koşullardan herhangi birinin çiğnenmesi durumunda güvenlik hedefini nasıl etkileyeceğinin belirlenmesi.

Bilgi güvenliği protokollerinin temellerini aşağıdaki dört güvenlik hizmeti oluşturur. Bu hizmetler asıllama, inkar edememe, veri bütünlüğü ve güvenilirlik/gizliliktir. Güvenlik mekanizmaları bu hizmetler üzerinden tasarlanır. Bugün akıllı kart ile güvenlik çözümleri de genel olarak bu dört farklı kriptografik hizmeti kullanır:

Asıllama (authentication): Varlıkların kendisine veya bilgiye yönelik tanımlama servisidir. Karşılıklı haberleşmeye başlayan iki tarafın birbirlerini tanımaları gerekir. Bir kanal üzerinden edinilen bilginin kaynağı bakımından, hangi tarihte üretildiği, içeriği, gönderiliş zamanı gibi başlıklarda asıllanması gerekir. Bu nedenlerle asıllama genellikle iki temel alt başlığa ayrılır: *varlık asıllama ve bilgi kaynağını asıllama*. Veri kaynağı asıllama dolaylı olarak veri bütünlüğünü de sağlar. Gerçekleştirmek için tipik olarak sayısal imza kullanılır.

İnkar edememe (nonrepudiation): Herhangi bir tarafın geçmişe dönük verdiği taahhütleri ve yaptığı işlemleri inkar edememesidir. Bu servis kart sahibinin bir elektronik veri aktarımı sırasında yaptığı işlemleri inkar etmesini engeller. Bu sayede yaptığı işlemler kart sahibini yasal olarak bağlar. Gerçekleştirmek için

güvenilir bir üçüncü kişinin hareketleri onaylamasına başvurulabilir. Tipik olarak sayısal imza kullanılır.

Veri bütünlüğü (data integrity): Elektronik veri aktarımı sırasında yer değiştiren verinin kontrol dışında herhangi bir değişikliğe uğramadan hedefe ulaşmasını sağlar. Gerçekleştirmek için tipik olarak sayısal imza kullanılır.

Güvenilirlik-Gizlilik (confidentiality): Elektronik işlem içerisindeki verilerin görmeye yetkili tarafların dışında gizli tutulmasını sağlar. Gizlilik mesajın şifrelenmesi ile sağlanır.

5.1. Protokollere yapılan saldırılar

Saldırılar kriptografik algoritmalara veya bu algoritmaların kullanıldığı protokollere yönelik olabilir. Bu bölümde ve daha sonra protokol tasarımı sırasında da olasılıklar değerlendirilirken gözetilen saldırılar arasıda algoritmadan kaynaklanan açıklar yer almamaktadır. Algoritmanın güvenli olduğu varsayılmaktadır.

Protokollere yapılan saldırılar pasif ve aktif saldırılar olmak üzere ikiye ayrılabilir:

1. Pasif saldırı. Saldırganın protokolün işletilmesi konusunda olumsuz bir etki meydana getirmeksizin sisteme bir yerine dahil olması ve dinleyici veya gözlemci konumunda kalmasıdır. Bu saldırı biçimi Bölüm 3'te sözü geçen saldırı durumlarından birincisine denk düşer. Pasif saldırılar farkedilmesi güç saldırılar olduğundan protokoller bu tür saldırıların farkedilmesi değil doğrudan engellenmesi üzerine kurulmayı çalışılır.

2. Aktif saldırı: Saldırgan protokole bir başkasının taklidini yaparak dahil olabilir ve protokole yeni mesajlar gönderir, var olan mesajları siler, bir mesajın yerini başka mesajlarla değiştirir, eski mesajları yeniden gönderir, bir haberleşme kanalında gidip gelen mesajları engelleyerek bu kanalı tıkar, veya bir bilgisayarda saklı tutulan bilgileri değiştirebilir. Bu yapıdaki saldırılar ağa bağlı saldırılardır.

Özellikle protokolün taraflarının birbirlerine tam güven duymadıkları sistemlerde aktif saldırılar daha da önem kazanır. Saldırganın her zaman dışarıdan birisi olması beklenmemelidir. Saldırgan sistem kullanıcısı hatta sistem yöneticisi de olabilir. Protokolün taraflarından biri olup diğer taraflara yalan da söyleyebilir.

5.2. Asıllama ve kimlik tanıma

Asıllama, çok geniş bir anlamda kullanılan ve sık sık da yanılgıya düşülen bir terim. Asıllama, tarafların iddia ettikleri kimliklerin doğruluğunu garanti etme imkanı sağladığı gibi, tarafların gönderilen bilginin saldırıya uğramadan kimliği doğrulanmış kişiden geldiği konusunda güvence duymalarını sağlar. Asıllama, güvenlik hedefine bağlı özellikler taşır. Bu hedefler arasında erişim kontrolü, varlık asıllama, mesaj asıllama, veri bütünlüğü, tekrar edilemezlik ve anahtar asıllama sayılabilir. Bu hedeflere bağlı olarak asıllama yöntemleri de değişebilmektedir.

Asıllama, bilgi güvenliği hedefleri arasında en önemli başlıklardan birisidir. 1970'lerin ortalarına kadar, asıllama ve gizliliğin birbirlerine sıkı bir biçimde bağlı oldukları düşünülmüştür. Çırpma fonksiyonlarının ve sayısal imzaların keşfinden sonra, asıllama ve gizliliğin birbirlerinden tamamen ayrı ve bağımsız iki bilgi güvenliği hedefi olduğu anlaşılmıştır. İkisi arasındaki ayrım ilk anda önemsiz gibi görünebilir fakat bazı durumlarda bu ayrımın zorunluluk yarattığı görülmektedir. Örneğin eğer A ve B arasındaki iki taraflı haberleşme ikisinin de farklı ülkelerde olduğu bir ortamda gerçekleşiyorsa, ülkeler, kanaldaki gizli haberleşmeye izin vermeyebilirler, içeriği görmek isteyebilirler. A ve B bu durumda birbirlerinin kimliğinden emin olabilecekleri asıllama yöntemini haberleşmenin gizliliğinden bağımsız bir biçimde tasarlamak zorunda kalacaklardır. Asıllama işlemi iki koşulda gerçekleştirilir:

1. A ve B'nin ikisi de haberleşme sırasında aktiftirler ve haberleşme bir zaman aşımına uğramadan "gerçek zamanda" meydana gelir.

2. A veya B, bir gecikme ile mesaj alışverişinde bulunur. Bu durumda mesaj çeşitli ağlardan geçerek yönlendirilir, saklanır ve sonraki bir zamanda iletilir.

Birinci durumda A ve B kimliklerini gerçek zamanda onaylamak isteyeceklerdir. Bu durumda örneğin A, B'ye yalnızca B'nin yanıtını bildiği bir sorgulama mesajı gönderir. Bu biçimde gerçekleştirilen asıllamaya **varlık asıllama** veya **kimlik doğrulama** denir.

İkinci olasılıkta ise bir sorgu gönderip yanıt beklemeye geçmek doğru değildir. Haberleşme yolu tek yönlü açık olabilir. Yani karşı taraf mesajı alamıyor olabilir. Bu durumda mesajın kaynağını asıllayabilmek için değişik teknikler geliştirmek gerekir. Bu yapıdaki asıllamaya **veri kaynağını asıllama** denir.

5.2.1. Kimlik tanıma

Kimlik tanıma veya varlık asıllama tekniği, (onaylı kanıtlar aracılığıyla) haberleşen iki tarafın her birine diğerinin varlığı ve kanıtın yaratıldığı veya elde edildiği anda diğer tarafın etkin olduğu güvencesini verir. Olağan bir durumda aktarılan yegane veri, haberleşen tarafların kimliğini saptamak için gerekli olan veridir. Varlıkların ikisi de haberleşmede etkindir, bu da zaman sınırlaması olmayan bir garanti sağlar.

Örneğin A, B'yi telefonla aradığında A ve B birbirini tanıyorsa, kimlik tespiti ses tanıma ile sağlanır. Tamamen hatasız çalışmasa da uygulamada etkin bir şekilde kullanılabilen bir yöntemdir.

Bir başka örnek ise bankamatiklerdir. A kişisi, bir bankamatiğe kişisel kimlik numarasını (PIN) ve A hakkında bilgi taşıyan manyetik bir şerit verir. Bankamatik, kart üzerindeki bilgiyi ve PIN' i kullanarak kart sahibinin kimliğini doğrulamaya çalışır. Doğrulama başarıyla gerçekleşirse A'ya, makinanın çeşitli hizmetlerine erişim hakkı tanınır.

Burada tarif edilen durumlardan ilki, karşılıklı asıllamaya, ikincisi ise tek yönlü asıllamaya bir örnektir.

5.2.2. Bilgi kaynağını asıllama

Veri kaynağı asıllaması veya mesaj asıllaması teknikleri, mesaj alan bir tarafa, mesajı gönderen tarafın kimliği hakkında onaylı kanıtlar aracılığıyla güvence

verir. Çoğu kez B'ye iletilen mesajın yanında, B'nin mesajı gönderenin kimliğini tespit etmesini sağlayacak ek bilgiler sağlanır. Bu çeşit bir asıllama, dakiklik konusunda güvence vermez, ancak haberleşen taraflardan birinin etkin olmadığı durumlarda faydalıdır.

5.3. Sayısal imzalar

Sayısal imzalar bilgisayarların hayatın her alanına girdiği günümüzde sayısal bilgi aktarımında dokümanları imzalamak amacıyla kullanılır. Bu anlamda bir imzanın sağladığı özellikleri sayısal imzanın sağlaması gerekir. Bu özellikler aşağıdaki gibi sıralanabilir:

1. İmzanın asıllığından emin olunmalıdır. İmza dokümanı alan kişiyi, imzacının dokümanı bilinçli bir biçimde imzaladığına inandırmalıdır.
2. İmza taklit edilemez olmalıdır.
3. İmza yeniden kullanılamaz olmalıdır. İmza bir dokümanın parçasıdır. İmzanın kötü niyetli bir kişi tarafından bir başka dokümana taşınması mümkün olmamalıdır.
4. İmzalanmış doküman değiştirilemez olmalıdır.
5. İmzalayan kişi dokümanı imzaladıktan sonra imzasını reddedememelidir.

Sayısal imzaların, bilgi güvenliği alanında, asıllama, veri bütünlüğü, tekrar edilemezlik gibi pek çok uygulaması vardır. Sayısal imzaların en çok karşılaşılan uygulaması büyük ağlarda açık anahtarların sertifikasyonudur. Sertifikasyon, üçüncü bir güvenli kişi tarafından bir kullanıcının kimlik bilgisini bir açık anahtara bağlamak için kullanılır. Böylece bu kullanıcının asıl anahtarı daha sonraki işlemler sırasında bir güvenli üçüncü kişiye gereksinim duyulmaksızın asıllanabilir.

5.3.1. Gizli anahtar şifreleme ve doküman imzalama

Yalnızca simetrik şifreleme düzeninin kullanıldığı sayısal imzalarda her imzanın yalnızca tek bir kişiye özgü olduğu düşünülürse her imzacının ayrı gizli anahtarı

olması gereklidir. Gizli anahtarın iki kişi arasında paylaşılamaması imzasını doğrulanması işleminin nasıl yapılacağı konusunda bir problem ortaya çıkarır. Bu problemin ek bir yöntem kullanmadan çözümü için bir güvenilir üçüncü kişi kullanılmaktadır. Bu üçüncü kişi birbirlerine imzalı doküman göndermek isteyen tarafların açık anahtarının bilgisine sahip olacaktır. Güvenilir üçüncü kişinin bir başkası yerine dokümanları imzalayamayacağına güvenilmektedir.

A, B'ye imzalı doküman göndermek istediğinde aşağıdaki adımlar izlenir:

1. A, dokümanı (P) kendi gizli anahtarı A_GA'yı kullanarak şifreler ve güvenilir üçüncü kişi G'ye gönderir.

$$E_{A\text{-}GA}(P) = D$$

2. G, A_GA'yı bilmektedir. Mesajı açarak mesajın A'dan gelip gelmediğine karar verebilir. Mesajın anlamlı bir metin olduğu varsayılır.

$$D_{A\text{-}GA}(D)=P$$

3. G, imzayı onayladıktan sonra P'nin sonuna imzanın A'ya ait olduğunu söyleyen bir ifade ekler ve dokümanı B'nin gizli anahtarı ile şifreleyerek B'ye gönderir.

$$E_{B\text{-}GA}(P')=D'$$

4. B, mesajı gizli anahtarı ile açar ve dokümanı okur. Dokümanın G'den geldiğine emindir çünkü kendi gizli anahtarını bilen tek kişi G'dir.

$$D_{B\text{-}GA}(D')=P'$$

Bu yöntemle bir önceki bölümde sıralanan imza özellikleri sağlanmış olur. Fakat B, A'dan gelen imzalı metni C'ye göstermek istediğinde C'ye metnin A tarafından imzalanmış olduğunu elindeki metinle kanıtlayamaz. Bunun için yeniden güvenilir üçüncü kişiye başvurması gerekir.

1. P' mesajını G'ye kendi gizli anahtarı B_GA ile şifreleyerek gönderir.

2. G mesajı B_GA ile açar ve veri tabanının kontrol ederek mesajın gerçekten A tarafından imzalanmış olduğunu onaylar. Elde etmiş olduğu P´ mesajını C'nin gizli anahtarı ile kapatarak C'ye yollar.

3. C mesajı gizli anahtarı ile açar ve G'nin onayıyla birlikte dokümana ulaşmış olur.

Tüm gizli anahtarların bir üçüncü kişinin elinde bulunması ve protokolde üçüncü kişinin sürekli aktif bir rolünün olması problemli durumlardır. G çok kullanıcının olduğu bir sistemde mesaj şifreleyip açma işine zaman ve kaynak yetiremez. Bu nedenle bugün sayısal imzalarda yalnızca simetrik şifrelemeye dayanan protokollerin kullanılması tercih edilen bir durum değildir.

5.3.2. Açık anahtar şifreleme ile doküman imzalama

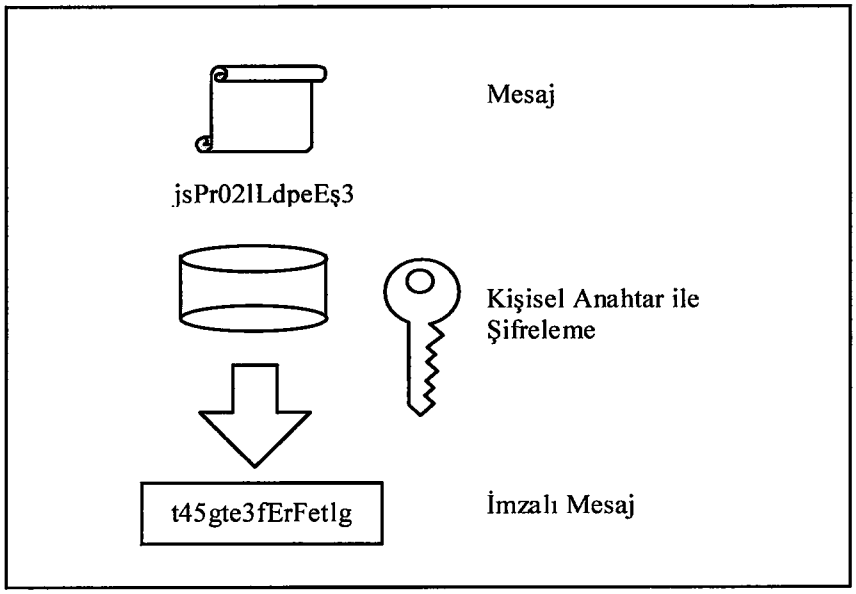
RSA gibi algoritmalarda mesaj gizli anahtar kullanılarak şifrelendiğinde imzalanmış olur. DSA gibi şifreleme düzenlerinde ise mesajın imzalanması için, mesaj şifreleme işleminden ayrı bir imzalama algoritması kullanılır. Açık anahtar şifrelemenin doğrudan imzalamada kullanılması aşağıdaki biçimde olur:

1. A, mesajı kendi gizli anahtarı ile şifreler. Böylece mesajı imzalamış olur.

2. A, imzalanmış mesajı B'ye gönderir.

3. B imzalanmış mesajı A'nın açık anahtarı ile açar ve mesajın anlamlı olduğu kabul edildiğinden A'nın imzasını kontrol etmiş olur.

Burada G'ye ihtiyaç kalmamıştır. İmzalama işlemi ve imzanın onaylanması farklı anahtarlarla yapıldığından B imzalı doküman göndermek istediği kişilere kendi açık anahtarını dağıtabilir. Burada da dağıtım işlemi için bir güvenli kişiye gereksinim vardır. B elindeki açık anahtarın gerçekten A'ya ait olduğundan nasıl emin olacaktır? Anahtar dağıtım problemini çözmek üzere başvurulan güvenli kişiye **sertifika otoritesi** adı verilir (Şekil 5.1).



Şekil 5.1. Açık anahtar şifreleme ile doküman imzalama

5.3.3. Sayısal zarf

Mesajı gönderen kişi (A), kendi sayısal imzası ile alıcıyı mesajın kendisinden geldiği konusunda ikna etmiş olur. Fakat mesajın gizliliğini sayısal imza ile sağlamış olmaz. A'nın açık anahtarını bilen herhangi bir kişi imzayı onaylayabileceği gibi mesajı da açmış olur. Mesajın gizliliğini sağlamak için kullanılan mekanizmalardan birisi açık anahtar şifrelemedir. Mesajın imzalandıktan sonra yeniden bu defa gizliliği sağlamak üzere şifrelemesine sayısal zarf oluşturma da denilmektedir. Bunun için aşağıdaki basamaklar yürütülür:

1. A, mesajı kendi gizli anahtarı ile imzalar (Açık anahtar şifreleme yöntemini kullanarak mesajı şifreler).

2. A, imzalı mesajı, B'nin açık anahtarı ile kapatır ve B'ye gönderir.

3. B aldığı mesajı önce kendi gizli anahtarı ile açar ve sonucu A'nın açık anahtarı ile deşifre eder (Burada açık anahtar şifre çözme fonksiyonu, farklı anahtar ile iki kez kullanılmış olur).

4. B, anlamlı bir metne ulaştığı durumda imzayı onaylar.

5.3.4. Tek yönlü çırpma fonksiyonlarının imzalamada kullanımı

Açık anahtar şifreleme algoritmaları uzun dokümanların imzalanmasında çok yavaş ve verimsizlidirler. Bu durumda imzalanacak dokümanın küçültülmesi gerekmektedir. Zaman kazanmak açısından imzalama protokollerinde genellikle bir yönlü çırpma fonksiyonları kullanılır. Dokümanı imzalamak yerine A dokümanın çırpmasını imzalar. Bunun için her iki taraf da önceden ortak bir çırpma fonksiyonu ve sayısal imza algoritması üzerinde anlaşırlar. A, B'ye imzalı bir doküman göndermek istediğinde:

1. A, dokümanın tek yönlü çırpmasını hesaplar.
2. A, çırpma değerini kendi gizli anahtarı ile şifreler. Böylece dokümanı imzalamış olur.
3. A, imzalanmış çırpma değeri ile birlikte dokümanın kendisini B'ye gönderir.
4. B,A'nın gönderdiği dokümanın tek yönlü çırpma değerini hesaplar. Sayısal imza algoritmasını kullanarak B, imzalanmış çırpma değerini A'nın açık anahtarı ile açar. Deşifre edilen değerle çırpma sonucunda ulaştığı değerleri karşılaştırır. Birbirini tutuyorsa imza doğrulanır.

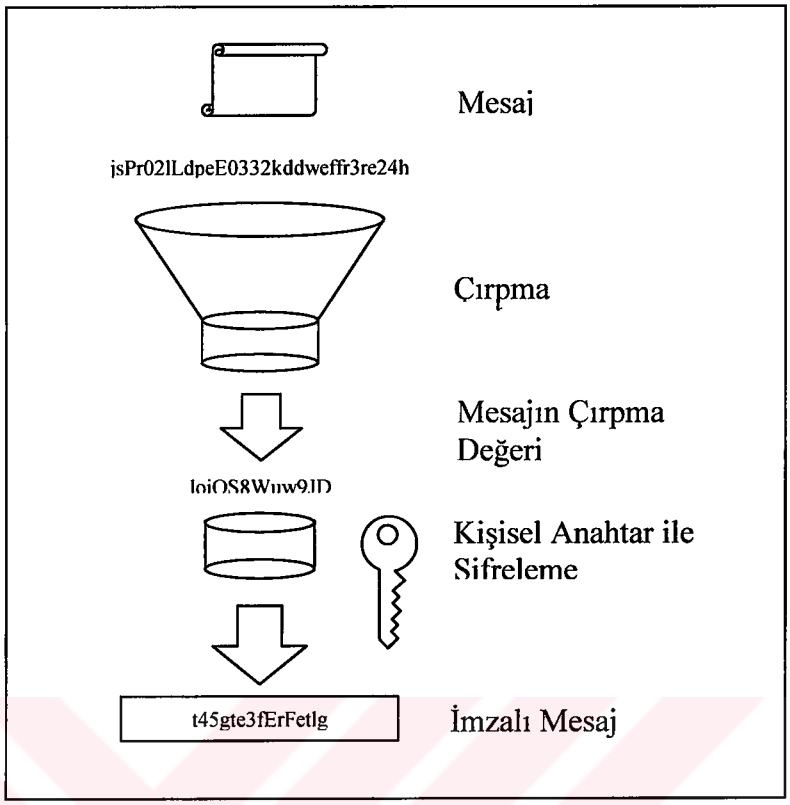
Bu yöntemin pek çok ulumlu yanı vardır. Birincisi, imza dokümandan ayrı tutulmuş olur. İkincisi alıcının doküman ve imza için ayırması gereken yer küçülmüş olur.

5.3.5. Sayısal imza düzenlerinin sınıflandırılması

Sayısal imza düzeni, imzalama ve imzanın doğrulanması için kullanılan mekanizmaların bütünüdür.

Sayısal imza düzenlerini iki ayrı sınıfta toplayabiliriz:

1. Ekli sayısal imza düzenleri, orijinal mesajı imza doğrulama algoritmasının girişinde kullanır. Pratikte en çok kullanılan imza düzenleridir. Çırpma fonksiyonlarının kullanımına dayanırlar. DSA, ElGamal, Schnorr imza düzenleri örnek olarak verilebilir (Şekil 5.2).



Şekil 5.2. Çırpma fonksiyonunun mesaj imzalamada kullanılması

2. Mesaj çözmeli sayısal imza düzenleri, imza doğrulama algoritmasının girişinde orijinal mesaja gerek duymaz. Bu durumda imza doğrulama algoritmasının kendisi mesajın deşifre edilmesi için kullanılır. Mesaj, imzanın kendisinden elde edilebilir. RSA, Rabin, Nyberg-Rueppel açık anahtar imza düzenleri örnek verilebilir (Şekil 5.1).

5.3.6. Zaman pulu ve sayısal imzalar

A, B'ye imzalı bir mesaj gönderirken yolda bu mesajın ele geçirilmesi riski vardır. İmzalı mesaj ele geçiren kişi tarafından kötüye kullanılabilir. Bu kişi imzalı mesajı açmayı, imza ve mesajı birbirinden ayırmayı başaramasa da elindeki imzalı dokümanı kullanabilir Örneğin imzalı mesaj sistemde bir işlemi yapma yetkisi kazandırıyor olabilir. Bu mesajı eline geçiren bir başkası elindekini göstererek yetkiyi devralmayı başarmış olur. Bu durumda imzalı mesajın yeniden bir başkası tarafından kullanımını engellemek amacıyla zaman pulu kullanılır. **Zaman pulu,** imzanın içerisine yerleştirilen imzalama zamanı bilgisidir. Bu sayede süresi geçen

imzalı metinler bir başkası tarafından kullanılsalar bile artık geçersizdirler. İmzayı onaylayan taraf, zamanı da kontrol eder.

5.3.7. Çoklu sayısal imzalar

Birden fazla kişinin aynı dokümanı imzalaması durumudur. Örneğin A ve B dokümanın ayrı ayrı kopyalarını imzalayarak karşı tarafa gönderebilirler. Fakat bu durumda sonuç verisi orijinal dokümanın iki katı büyüklüğünde olur. Çırpma fonksiyonu kullanmadan izlenebilecek bir başka yöntem A'nın önce imzaladığı dokümanı B'nin imzalamasıdır. Bu yöntemin özelliği ise mesajı alan tarafın A'nın imzasını onaylayabilmek için B'nin imzasını onaylaması gerekliliğidir. B'nin imzası onaylanmadan A'nın dokümanı imzalayıp imzalamadığını öğrenmek mümkün değildir.

Birinci yöntem mesajın kendisi yerine sıkıştırılmış hali olan çırpma değeri imzalanarak uygulanırsa uzunluk problemi çözümlenmiş olur.

5.3.8. İnkar edememe ve sayısal imzalar

Bir sayısal imzanın temel özelliklerinden biri de imzayı atan tarafın daha sonra imzanın kendisinin olduğunu reddetmesinin mümkün olmamasıdır. Zaman pulu kullanımı bu tür kötüye kullanımları sınırlar fakat engellemez. İmzalayan kişi mesajın gösterdiği zamandan daha önce imzasının çalınmış olduğunu iddia edebilir. Burada uygulanabilecek çözümlerden biri haberleşme sırasında bir üçüncü kişinin onayını almaktır. Yani bir hakem kullanılır.

İmzalayan tarafa A, İmzayı onaylayan tarafa B ve hakeme de T diyecek olursak, genel olarak protokol aşağıdaki basamaklardan oluşur:

1. A, mesajı imzalar
2. A, kimliğine dair tanımlayıcı bilgi içeren bir başlık bilgisi oluşturarak imzalı mesaja ekler ve tümünü birden imzalayarak yeni bir paket oluşturur. Paketi T'ye gönderir.

3. T, dıştaki paketi açarak imzayı ve gelen tanımlayıcı bilgiyi onaylar. A'nın imzalı mesajına bir zaman pulu ve tanım bilgisi ekler. Son olarak tüm paketi imzalayarak hem A, hem de B'ye gönderir.

4. B, T'nin imzasını onayladıktan sonra zaman pulunu ve son olarak da A'nın imzasını onaylar.

5. A, T'nin B'ye gönderdiği mesajı açarak T'nin imzasını onaylar. Eğer gönderilen pakette itiraz edeceği bir bilgi varsa mesajı reddeder.

A, son basamak da işletildikten sonra B'ye gönderdiği mesajdaki imzayı reddetme hakkına sahip değildir.

Bu şekilde oluşturulan protokol her işlemde bir güvenilir üçüncü kişinin hazır bulunmasını gerektirmesi bakımından her zaman tercih edilen bir yöntem değildir.

5.3.9. Sayısal imza düzenlerine yönelik saldırılar

Sayısal imza düzenlerine yönelik saldırılar imzanın ele geçirilmesi amacına dönüktür. Sayısal imzaların kırılmasını üç ayrı kategoride toplayabiliriz:

1. Tümden kırılma: saldırgan, imzacının açık anahtar bilgisini elde etmeyi başarmış veya imzalama algoritmasına eşdeğer bir hesaplama yöntemi bulmuştur.

2. Seçimli sahtekarlık: Saldırgan seçilen bazı mesajları veya belirli bir tür mesajı imzalamayı başarmıştır. İmzanın asıl sahibi bu aldatmacanın içinde yer almaz.

3. Varoluşsal sahtekarlık: Saldırgan en az bir mesaj için imza taklit etmeyi başarmıştır. İmzası ele geçirilen mesaj üzerinde saldırgan tam kontrol sahibi değildir. İmzanın sahibi bu aldatmacanın bir parçası olabilir.

Açık anahtar sayısal imza istemlerine uygulanan iki temel saldırı yöntemi vardır:

1. Anahtar saldırıları: saldırgan yalnızca imza sahibinin açık anahtarını bilmektedir.

2. Mesaj saldırıları: Saldırgan bilinen veya seçilen mesajların imzalarını analiz etme imkanına sahiptir.

5.4. Kimlik doğrulama ve imzalama düzenleri arasındaki ilişki

Doğrulama düzenleri, sayısal imza düzenleri ile çok yakından ilişkili olup daha basittirler. Değişken bir mesajın varlığı ve imzanın reddedilemezliği özellikleri sayısal imzaları daha karmaşık kılar. Kimlik doğrulama düzenlerinde mesajın semantiği temelde sabittir. Mesaj, o an için kullanılacak iddia edilen kimlik bilgisidir. İddia o anda ilgili ayrıcalık veya haklarının verilip verilmemesi şeklinde gerçek zamanda onaylanır veya reddedilir.

5.5. Anahtar kurulum protokolleri

Anahtar kurulumu, şifrelemede kullanılmak üzere iki veya daha fazla taraf arasında ortaklaşa bir gizli değerin belirlendiği süreç veya protokole verilen addır. Anahtar kurulumu, genel olarak anahtar gönderimi ve anahtar anlaşması olmak üzere iki ayrı biçimde yürütülebilir.

Anahtar gönderme protokolü veya mekanizması, bir tarafın gizli bir değer oluşturarak veya edinerek diğerine gizli bir kanaldan göndermesini sağlayan bir anahtar kurulum tekniğidir.

Anahtar ortaklaştırma protokolü veya mekanizması ise herhangi birinin tek başına önceden belirlemesi mümkün olmayan ortaklaşa bir gizli değerin iki veya daha fazla taraf arasında türetildiği bir anahtar kurulum tekniğidir.

5.5.1. Diffie-Hellman anahtar ortaklaştırma protokolü

Diffie-Hellman anahtar ortaklaştırma (üstel anahtar değişimi adı da verilir) asıllamasız anahtar ortaklaştırma işleminde kullanılan en temel tekniktir. Bu anlamda pek çok anahtar kurulum protokolünün temelini oluşturur.

A ve B aralarında bir gizli anahtar üzerinde anlaşma sağlayacak taraflar olsun.

p asal sayı ve α, Z_p^* ($2 \leq \alpha \leq p-2$)çarpımsal grubunun üreteci olmak üzere temel protokol adımları aşağıdaki gibidir:

1. A, $1 \leq x \leq p-2$ olacak şekilde bir x gizli değeri belirler ve α^x modp değerini hesaplayarak B'ye gönderir.

2. B, $1 \leq y \leq p-2$ olacak şekilde bir y gizli değeri belirler ve α^y modp değerini hesaplayarak A'ya gönderir.

3. B, α^x değerini alır ve $K = (\alpha^x)^y$ modp değerini hesaplar.

4. A, α^y değerini alır ve $K = (\alpha^y)^x$ modp değerini hesaplar.

Bu şekilde hesaplanan K, A ve B arasındaki ortak gizli anahtardır.

Bu protokolü açık anahtar şifreleme yöntemine göre düşünerek geliştirecek olursak, A ve B arasında bir veri akışı olmaksızın da K ortak gizli değeri hesaplanabilir.

A'nın (açık,gizli) anahtar çifti: (α^x modp, x);
B'nin (açık,gizli) anahtar çifti: (α^y modp, y) olsun.

A ve B, açık anahtarlarını imzalı sertifikalar aracılığıyla birbirlerine bildirebilirler ve bu sayede her iki taraf da kendi gizli anahtarını ve karşı tarafın açık anahtarını kullanarak K'yı hesaplayabilir. Bu şekilde hesaplanan K değerine "ortak giz" adı verilir.

5.6. ISO asıllama çerçevesi

ITU-X..509 protokolleri olarak da bilinir. X509, ilk olarak 1989'da yayınlanmıştır[26]. 1993'te güvenlikle ilgili eleştiriler de değerlendirilerek yapılan bazı düzeltmelerle yenilenmiştir.

5.6.1. Sertifikalar

X.509'un en önemli yanı, açık anahtar sertifikalar için kurulan yapıdır. Her kullanıcının ayrı bir adı vardır. Güvenilir bir sertifika otoritesi, her kullanıcıya bir

ad atar ve kullanıcının açık anahtarını ve adını içeren imzalı bir sertifika yayınlar. Bir X.509 sertifikası Şekil 5.3'te görülen bölümlerden oluşur.

Versiyon
Seri numarası
Algoritma Tanımlayıcı -Algoritma -Parametreler
Yayımcı
Geçerlilik Süresi -Başlangıç tarihi -Bitiş Tarihi
Hedef
Hedefin açık anahtarı: -Algoritma -Parametreler -Açık Anahtar
İmza

Şekil 5.3 Bir X.509 Sertifikası

Versiyon, sertifika formatını tanımlar.

Seri numarası, sertifika otoritesi içinde tektir

Algoritma, parametrelerle birlikte sertifikayı imzalamak için kullanılan algoritmayı tanımlar

Yayımcı, sertifika otoritesinin adıdır.

Geçerlilik süresi, sertifikanın geçerli olduğu başlangıç ve bitiş tarihini belirtir.

Hedef, kullanıcı adıdır.

Kullanıcının açık anahtar bilgisi, algoritma adını, gerekli parametreleri ve açık anahtarı içerir.

İmza, bu en son alan ise sertifika otoritesinin imzası için ayrılmıştır.

Eğer A, B ile haberleşmek istiyorsa öncelikle onun sertifikasını veri tabanından alır. Ardından geçerliliğini onaylar. Eğer her iki taraf da aynı sertifika otoritesini kullanıyorlarsa işlemler daha kolay yürür. A, sertifika otoritesinin imzasını, B'nin sertifikası üzerinde onaylar. Eğer farklı sertifika otoritelerini kullanıyorlarsa daha

karmaşık bir işlem yürütülür. Bu durumda A, B'nin sertifika otoritesine ulaşana kadar birbirinin sertifikalarını üreten sertifika otoriteleri ile zincirleme bir işlem yürütür.

Bu sertifika yönteminin bir problemi vardır. Sertifika süresi dolmadan önce bir nedenle geçersiz duruma geçebilir. Örneğin sertifika otoritesi ilgili kullanıcıya ait sertifikayı feshedebilir. Bu durumdan o sertifikayı kullanan tarafların haberdar edilmeleri gereklidir.

5.6.2. Asıllama protokolleri

A, B ile haberleşmek istediğinde veri tabanına gider ve A'dan B'ye giden "sertifikasyon yolu" ile birlikte B'nin açık anahtarını edinir. Bu noktada A, bir yollu, iki yollu, üç yollu asıllama protokolü başlatabilir.

Bir yollu protokol, A'dan B'ye doğru basit bir haberleşmedir. A ve B'nin kimliğini tesbit eder ve A'dan B'ye gidecek verinin bütünlüğünü sağlar. Bağlantı üzerine yapılacak "tekrar" saldırılarını engeller.

İki yollu protokol, B'nin A'ya yanıtını da organize eder. Yanıtın B'nin kılığına girmiş bir saldırgandan değil B'nin kendisinden geldiğini denetler. İki taraflı haberleşmenin de gizliliğini ve verinin bütünlüğünü sağlar.

Hem bir yollu, hem de iki yollu protokoller zaman pulu kullanırlar. Üç yollu protokolde A'dan B'ye gidecek veriye bir başka mesaj eklenir, zaman pulu ihtiyacı ortadan kalkar.

5.6.2.1. Tek yollu protokol

1. A, rastgele bir sayı üretir (R_A)
2. A, M = (T_A, R_A, I_A, d) mesajını üretir, T_A A'nın zaman puludur. I_A, A'nın kimliği; d, herhangi bir veri parçasıdır Verinin güvenliği isteniyorsa d, B'nin açık anahtarı ile kapatılabilir ($E_{B\text{-}AA}$).
3. A, (C_A, $E_{A\text{-}GA}$(M))'yi B'ye gönderir. C_A, A'nın sertifikasıdır. A_GA, A'nın gizli anahtarıdır.

4. M, C_A'yı doğrulayarak, A_AA'yı elde etmiş olur. Bu anahtarın geçerliliğin yitirmediğini kontrol eder. A_AA, A'nın açık anahtarıdır.

5. B, A_AA'yı E_{A_GA}(M)'yi açmak için kullanır. Bu işlemle hem A'nın imzasını onaylamış hem de verinin bütünlüğünü kontrol etmiş olur.

6. B, M'nin içinde bulunan I_B'yi kontrol ederek M'nin doğruluğunu test eder.

7. B, M'nin içindeki T_A'yi kontrol eder ve mesajın geçerliliğini yitirip yitirmediğini anlar.

8. Bir seçenek olarak B, R_A'yı daha önce gelen mesajların R_A'larını tuttuğu veri tabanıyla karşılaştırır ve bu mesajın daha önce gelen mesajlardan biri olup olmadığının da kontrol etmiş olur.

5.6.2.2. İki yollu protokol

Yukarıda anlatılan bir yollu protokolün çift yönlü işleyen biçimidir. İki yollu protokolde yukarıdaki 8 basamak işletildikten sonra ek olarak aşağıdaki basamaklar işletilir:

9. B, rastgele bir sayı üretir (R_B).

10. B, M^1=(T_B, R_B, I_A, R_A, d) mesajını üretir. T_B, B'nın zaman puludur. I_A, A'nin kimliği; d, herhangi bir veri parçasıdır. Verinin güvenliği isteniyorsa d, A'nın açık anahtarı ile kapatılabilir (E_{A_AA}), R_A, A'nın birinci adımda ürettiği rastgele sayıdır.

11. B, E_{B_GA} (M')'yi B'ye gönderir.

12. A, B_AA'yı E_{B_GA}(M')'yi açmak için kullanır. Bu işlemle hem B'nın imzasını onaylamış hem de verinin bütünlüğünü kontrol etmiş olur.

13. B, M' 'nin içinde bulunan I_A'yı doğruluk açısından kontrol eder.

14. B, M' 'nin içinde ki T_B'yi kontrol eder ve mesajın geçerliliğin yitirip yitirmediğin anlar.

15. Bir seçenek olarak A, R_B'yi daha önce gelen mesajların R_B'lerini tuttuğu veri tabanıyla karşılaştırır ve bu mesajın daha önce gelen mesajlardan biri olup olmadığını da kontrol etmiş olur.

5.6.2.3. Üç yollu protokol

İki yollu protokolle aynı işi yapar fakat zaman pulu kullanılmaz. 1'den 15'e kadar olan adımlar $T_A=T_B=0$ olmak koşuluyla aynıdır.

16. A, B'den geri dönen R_A'yı B'ye gönderdiği rastgele sayıyla karşılaştırır.

17. A, B'ye $E_{A\_GA}(R_B)$ gönderir.

18. B, A'nın açık anahtarı ile $E_{A\_GA}((_B)$'yi açar. Bu işlem hem A'nın imzasının hem de imzalanan verinin bütünlüğünün doğrulanmasını sağlar.

19. B, 10.adımda A'ya yolladığı R_B ile geri dönen R_B'nin aynı olup olmadığını kontrol eder.

5.7. Eliptik eğri protokolleri: ECDH, ECDSA, ECAES

Eliptik eğri üzerine kurulu üç temel protokol bulunmaktadır:

1. Elliptik Curve Diffie Hellman (ECDH)
2. Elliptic Curve Digital Signature Algorithnm (ECDSA)
3. Elliptic Curve Authenticated Encryption Scheme (ECAES)

ECDH, Diffie-Hellman anahtar anlaşma yönteminin Eliptik Eğri versiyonudur. ECDSA, DSA'nın 1992'de Scott Vanstone [27] tarafından önerilen Eliptik Eğri uygulamasıdır. ECAES ise Abdalla, Bellare ve Rogaway tarafından 1999'da [28] önerilen ElGamal açık anahtar şifreleme düzeninin bir başka şeklidir.

Anahtar üretimi:

A'nın açık ve gizli anahtar çifti, belirli Eliptik Eğri alan parametreleri ile ilişkilidir: $(q,FR,a,b,G,n,h)^5$.

Bir anahtar çifti üretmek için A'nın yapması gereken işlemler aşağıdaki gibidir:

1. [1, n-1] aralığında bir rastgele sayı belirlenir,
2. Q=d.G hesaplanır,
3. A'nın açık anahtarı Q, gizli anahtarı ise d'dir.

Açık anahtarın geçerliliğinin doğrulanması:

Bu işlem, üretilen açık anahtarın Eliptik Eğri açık anahtarının aritmetik gereklerini yerine getirip getirmediğini kontrol eder. Alan parametrelerine (q,FR,a,b,G,n,h) ilişkin bir açık anahtar $Q=(x_Q y_Q)$'nun uygun olup olmadığını saptamak için aşağıdaki işlemler uygulanır:

1. $Q \neq O$ olduğu kontrol edilir,
2. x_Q ve y_Q'nun F_q'nun uygun biçimde temsil edilen elemanları olup olmadığı kontrol edilir,
3. Q'nun a ve b tarafından tanımlanan eliptik eğri üzerinde olup olmadığı kontrol edilir,
4. $n.Q=O$ olduğu kontrol edilir.

Dördüncü şıktaki kontrol yapılmadığında buna "yarı doğrulama" denir.

5.7.1. ECDH

Bu ilkenin temel mantığı, tarafların kendi gizli anahtarları ve karşı tarafın açık anahtarını kullanarak aynı anda ve birbirine eşit değere sahip "paylaşılan gizli veri" üretmelerini sağlamaktır. A tarafının alan parametrelerinin D=(q,FR,a,b.G,n,h), gizli anahtarının d_A olduğunu; B tarafının D'ye ilişkin açık anahtarının Q_B olduğunu varsayalım. (Q_B'nin en azından yarı doğrulanmış olması gereklidir.)

A tarafı, B tarafı ile ortak gizli veriyi hesaplamak için aşağıdaki işlemleri yürütür:

1. $P=d_A Q_B= (x_p, y_p)$ noktası hesaplanır,

2. P≠O olduğu kontrol edilir,

3. Ortak giz, $z=x_p$'dir.

Birinci basamaktaki hesaplama $P=h\ d_A\ Q_B= (x_p, y_p)$ şeklinde yapılacak olursa, bu ilkele "Eliptik Eğri Diffie Hellman kofaktörü" adı verilir. Kofaktör h'nin hesaplamaya dahil edilmesinin nedeni küçük alt grup saldırıları [29] gibi saldırılara karşı yeterli gelebilecek bir direnç oluşturabilmektir.

5.7.2. ECAES

Şifreleme (E) ve şifre çizme (D) için başlangıç kabulleri yapılacak olursa. Alıcı taraf B'nin alan parametreleri D= (q,FR,a,b,G,n,h) ve açık anahtarı Q_B olsun, Gönderici taraf A'nın, D ve Q_B'nin asıllanmış kopyalarına sahip olduğunu varsayalım. Burada MAC, HMAC gibi mesaj asıllama kodu anlamında kullanılmıştır. ENC ise TDES veya ortak-giz'den kriptografik anahtarları üreten Anahtar Türetme Fonksiyonu KDF gibi bir simetrik şifreleme düzeni anlamında kullanılmıştır.

B için m mesajını şifrelemek için A aşağıdaki işlemleri yürütür:

1. [1, n-1] aralığında bir rastgele r sayısı üretilir,

2. R=r.G'yi hesaplanır,

3. $K=h.r.Q_B=(K_x, K_y)$ hesaplanır. K'nın sıfıra eşit olup olmadığı kontrol edilir,

4. $k_1 \mid\mid k_2 = KDF\ (K_x)$ hesaplanır,

5. $c=ENC_{kl}\ (m)$ hesaplanır,

6. $t=MAC_{k2}\ (c)$ hesaplanır,

7. (R,c,t), B'ye gönderilir.

B, kapatılmış mesajı (R, c, t) açmak için aşağıdaki işlemleri yürütür:

8. R üzerinde anahtarı yarı doğrulama işlemi yürütülür,

9. $K=h.d_B^{1}=(K_x, K_y)$ hesaplanır, K'nın O'ya eşit olmadığı kontrol edilir,

10. $k_1 \mid\mid k_2$ =KDF (K_x) hesaplanır,

11. t=MAC_{k2} (c) olduğu doğrulanır,

12. m=$ENC^{-1}{}_{k1}$ (c) hesaplanır.

5.7.3. ECDSA

İmzalayan taraf A'nın alan parametreleri D= (q,FR,a,b,G,n,h) ve açık anahtarı Q_A olsun. İmzayı doğrulayan taraf B'de bu iki bilgi D ve Q_A 'nın asıllanmış olarak var olduğunu kabul edelim. SHA-1 160 bitlik çırpma fonksiyonunu göstermek üzere, A'nın m mesajını imzalamak için yapacağı işlemler aşağıdaki gibidir:

1. [1, n-1] aralığında rastgele bir tam sayı k seçilir,

2. kG = (x_1, y_1) ve r = x_1 modn hesaplanır,

Eğer r = 0 sonucu elde edilirse birinci basamağa geri dönülür, değilse aşağıdaki basamaklar yürütülür

3. Hesapla: k^{-1} modn

4. Hesapla: e = SHA-1(m)

5. Hesapla: s = k^{-1} {e+d_A.r}modn

Eğer s = 0 bulunursa birinci basamağa geri dönülür.

6. m mesajı için A'nın imzası: (r,s)

İmzayı doğrulamak için B aşağıdaki işlem basamaklarını yürütür:

1. r ve s sayılarının [1, n-1] aralığında tam sayılar olup olmadığını kontrol et

2. Hesapla: e=SHA-1(m)

3. Hesapla: w= s^{-1} modn

4. Hesapla: u_1=ew modn ve u_2=rw modn

5. Hesapla: $u_1G + u_2Q_A$= (x_1, y_1)

6. Hesapla: $v = x_1 \bmod n$

7. İmzayı yalnız ve yalnız v=r eşitliği sağlanıyor ise kabul et

İmza üretme ve doğrulama işlemleri sırasında geçen zamanı 2. ve 11. adımda yürütülen skaler çarpma işlemleri belirler [30].

6. ELİPTİK EĞRİ ŞİFRELEME İLKELLERİ VE DÜZENLERİ

P1363, açık anahtar kriptografi için geliştirilmiş bir IEEE standardıdır. Bu standartta açık anahtar şifreleme üzerine kurulu kriptografik teknikler belirtilir. Gizli değer türetme, açık anahtar şifreleme ve sayısal imzalar ve anahtar ortaklaştırma için matematiksel ilkeller ve bu ilkeller üzerine kurulu kriptografik düzenler tanımlanır. Standart aynı zamanda ilgili kriptografik parametreleri, açık ve gizli anahtarları tanımlar. P1363'ün amacı, uygulamaların seçim yapabilecekleri çeşitli tekniklerin özellikleri hakkında bir referans oluşturmaktır.

Tez çalışmasının gerçeklenmesi aşamasında bu standarttan yararlanılmıştır. Bu bölümde gerçekleme esnasında başvurulan kriptografik teknikler tanıtılacaktır.

Anahtar ortaklaştırma amacıyla Eliptik Eğri Anahtar Ortaklaştırma Düzeni, Diffie Hellman versiyonu olan ECKAS-DH1 [31] kullanılmıştır. Sayısal imza mekanizması olarak bir ekli sayısal imza düzeni olan Eliptik Eğri Ekli Sayısal İmza Düzeni, ECSSA [32] kullanılmıştır. Protokolde tarafların karşılıklı ortak giz türetmeleri amacıyla Eliptik Eğri Gizli Değer Türetme İlkeli-Diffie Hellman, ECSVDP-DH [33] kullanılmıştır. Sayısal imza düzeninde imza üretme ve imza doğrulama amacıyla kriptografik ilkeller kullanılmaktadır. Uygulamada Eliptik Eğri İmzalama İlkeli Nyberg-Rueppel versiyonu, ECSP-NR [34] ve Eliptik Eğri İmza Onaylama İlkeli Nyberg-Rueppel versiyonu, ECVP-NR kullanılmıştır[34].

6.1. Anahtar ortaklaştırma düzenleri (Key Agreement Schemes)

Bir anahtar ortaklaştırma düzeninde tüm taraflar kendilerine ait gizli anahtar(lar)ı karşı tarafın açık anahtar(lar)ı ile bileştirip bir gizli anahtar oluştururlar. Bu işlem sırasında kullanılan bir diğer bilgi ise taraflarca bilinen, dışarıya açık veya gizli olabilen anahtar türetme parametreleridir. Taraflar uygun anahtarları, birbirinin aynı anahtar türetme parametrelerini kullanarak düzeni doğru bir biçimde işletirlerse sonuçta aynı gizli anahtar verisine ulaşırlar. Anahtar ortaklaştırma

düzeni, tarafların öncül bir ortak gizli veriden yola çıkmalarına ihtiyaç duymadan bir ortak gizli anahtar türetmelerini sağlar.

Anahtar ortaklaştırma düzeninin gereksindiği ilkeller alan parametrelerinin ve anahtar çiftlerinin üretimidir.

Bir anahtar ortaklaştırma işlemi tüm düzenler için aşağıdaki yapıya sahiptir:

1. Tarafların anahtar çiftlerinin üretilmesinde kullanılacak bir veya daha fazla sayıda geçerli alan parametresi kümesi oluşturulur.

2. Birinci basamakta üretilen alan parametrelerinden yararlanılarak ortaklaştırma işleminde kullanılmak üzere bir veya daha fazla sayıda geçerli kişisel anahtar (ve bazı durumlarda karşılık gelen açık anahtar) seçimi yapılır.

3. Karşı tarafın veya tarafların açık anahtar(lar)ı elde edilir.

4. Beşinci adımdaki kriptografik işlemlere bağlı olarak açık anahtarların ve alan parametrelerinin geçerliliğini test etmek için bir yöntem belirlenir.

5. Belirli kriptografik işlemlerin ilgili gizli anahtar ve açık anahtarlara uygulanmasıyla ortak gizli değer üretilir.

6. Ortak gizli anahtarın elde edilmesi için taraflar aralarında ortaklaşılan anahtar türetme parametreleri ve bir önceki basamakta üretilen ortak gizli değer, anahtar türetme fonksiyonundan geçirilir.

Anahtar türetme parametrelerinin nasıl yorumlanacağı uygulamaya göre değişir. Her iki taraf da belirli koşullar altında hata üretebilirler. Koşullar aşağıdaki gibi belirlenebilir:

- İkinci basamakta gizli anahtar bulunamadı
- Üçüncü basamakta açık anahtar bulunamadı
- Dördüncü basamakta açık anahtar geçersiz
- Beşinci basamakta açık anahtar veya gizli anahtar desteklenmiyor
- Altıncı basamakta anahtar türetme parametresi desteklenmiyor

Bu hataları yakalamaya dönük üretilecek özel metotlar uygulamaya göre değişebilir ve P1363 Açık Anahtar Şifreleme standardına dahil değildir.

6.1.1. DL/ECKAS-DH1

DL/ECKAS-DH1, Ayrık Logaritma ve Eliptik Eğri Anahtar Ortaklaştırma Düzeni (Discrete Logarithm and Elliptic Curve Key Agreement Scheme), Diffie-Hellman versiyonudur.

6.1.1.1. Düzen seçenekleri

Aşağıdaki seçenekler üzerinde taraflar arasında bir kurgu yapılmalı veya anlaşmaya varılmalıdır.

- Bir gizli değer türetme ilkeli : DLSVDP-DH, DLSVDP-DHC, ECSVDP-DH, veya ECSVDP-DHC
- Hepsi için DH ilkeli istenir
- DL/ECKAS-DH1 ile birlikte kullanılmak üzere bir anahtar türetme fonksiyonu olarak KDF1 veya standarda uygun bir başka fonksiyon kullanılabilir.

Yukarıdaki seçenekler, anahtar ortaklaştırma düzeninin her işletilişinde aynı kalabilir veya belirli periyotlarla değiştirilebilir. Bu bilginin gizli tutulmasına gerek yoktur.

6.1.1.2. Anahtar ortaklaştırma işlemi

Taraflar, aşağıdaki basamak sekansını veya benzer bir işlem dizisini işleterek bir ortak gizli anahtar listesi üretirler: $K_1, K_2, K_3, \ldots, K_t$

1. Tarafların anahtar çiftlerinin oluşturulmasında kullanılacak geçerli DL veya EC alan parametreleri kurulur.
2. Birinci basamakta kurulan alan parametrelerine uygun, işlem için geçerli bir gizli anahtar s seçimi yapılır.

3. Diğer tarafın birinci basamakta kurulan alan parametrelerine uygun açık anahtarı, W, elde edilir.

4. *(Seçime bağlı)* Eğer seçilen gizli değer türetme ilkeli DLSVDP-DHC veya ECSVDP-DHC ise W'nin ilgili grubun bir elemanı olup olmadığı kontrol edilir (örn., DL için GF(q)'nun elemanı veya EC için eliptik eğri üzerinde olup olmadığına bakılır.) ;diğer durumda W'nin geçerli bir açık anahtar olup olmadığı kontrol edilir. Eğer geçerli olmadığı saptanırsa, "geçersiz açık anahtar" hata mesajı üretilir ve işlem sonlandırılır.

5. s gizli anahtarı, w diğer tarafın açık anahtarı ve seçilen gizli değer türetme ilkelleri kullanılarak z ortak gizli değeri hesaplanır.

6. z ortak gizli değeri, FE2OSP'yi kullanarak bir karakter katarına dönüştürülür.

7. Üzerinde anlaşmaya varılan her ortak gizli anahtar için

    a. Anahtar türetme parametreleri P_i belirlenir.

    b. Karakter katarı Z'den ve anahtar türetme parametreleri P_i'den , seçilen anahtar türetme fonksiyonu kullanılarak K_i elde edilir.

6.2. İmzalama düzenleri

P1363'te tanımlanan genel model, bir imzacı ve imza onaylayıcıdan oluşmaktadır. İmzacı gizli anahtarı ile mesajı imzalar, onaylayıcı ise imzacının açık anahtarı ile imzayı onaylar. Bir imza düzeni, bir mesajın kaynağı ve bütünlüğü konusunda güvence verir ve mesajın müdahaleye uğramadan doğru kişi tarafından gönderildiğinden karşı tarafın emin olmasını sağlar.

Daha önce anlatıldığı gibi P1363 standardında da imza düzenleri önceki bölümde anlatıldığı gibi ekli imza düzenleri ve mesaj çözmeli imza düzenleri olmak üzere ikiye ayrılmıştır. Fakat bu standartta yalnızca ekli imza düzenleri tanımlanmıştır. Bir imza düzeni, anahtar yönetimini de destekleyecek biçimde imza türetme ve imza onaylama işlemlerinden oluşmaktadır. İmza düzenlerinde anahtar çiftlerinin ve alan parametrelerinin üretimi, Ayrık Logaritma, Eliptik Eğri veya Çarpanlara Ayırma "aileleri"ne bağımlı olarak tanımlanmaktadır. P1363 standardında bir

imza doğrulama düzeninin genel işlemleri tanımlanmıştır. Bir imza üretme işlemi standartta tanımlı tüm düzenler için aşağıdaki yapıdadır:

1. Geçerli bir gizli anahtar seçilir (Varsa ona bağlı alan parametreleriyle birlikte).
2. Mesaja ve gizli anahtara bilinen kriptografik işlemler uygulanır.
3. İmza çıkışa aktarılır.

Ekli imza düzeninde bir imza onaylama işlemi aşağıdaki yapıdadır:

1. İmzacının açık anahtarı elde edilir (Varsa ona bağlı alan parametreleri ile birlikte).
2. (Bu adım seçime bağlıdır) Açık anahtar ve ona bağlı alan parametrelerinin geçerliliği onaylanır. Olumsuz sonuç elde edilirse imza geçersiz kabul edilir.
3. Mesaj, açık anahtar ve imza üzerinde bilinen kriptografik işlemler uygulanır.
4. Üçüncü adımdaki sonuca göre "geçerli" veya "geçersiz" sonucu çıkışa aktarılır.

İmzacı ve onaylayıcı taraflar, aşağıdaki seçenekler üzerinde anlaşmalıdırlar:

- İmzalama ve onaylama ilkelleri olarak ikililerden birini seçmek durumundadırlar: DLSP-NR ve DLVP-NR, DLSP-DSA ve DLVP-DSA, ECSP-NR ve ECVP-NR, veya ECSP-DSA ve ECVP-DSA.
- Ekli imza düzenleri için mesaj kodlama yöntemi seçilir: EMSAI [35] veya DL/ECSSA ile birlikte kullanılmak üzere tasarlanmış bir yöntem.

P1363 standardında iki imza düzeni tanımlanmıştır. Birincisi DL/ECSSA, ikincisi ise IFSSA'dır. Birincisi matematiksel olarak eliptik eğri ayrık logaritma, diğeri ise çarpanlara ayırma problemine dayanmaktadır.

6.2.1. DL/ECSSA

DL/ECSSA, Ayrık Logaritma ve Eliptik Eğri Ekli İmza Düzenidir [32]. Taraflar arasında yukarıda belirtilen başlıklar üzerinde anlaşma sağlanmalıdır:

6.2.1.1. İmza üretme işlemi

M mesajı ile yola çıkarak (c, d) imzası aşağıdaki işlemler yürütülerek oluşturulur:

1. Geçerli bir gizli anahtar, s ve ona bağlı alan parametrelerinin seçimi yapılır.

2. Eğer seçilen imzalama ilkeli DLSP-NR veya ECSP-NR ise

(mesaj örneğinin maksimum uzunluğu) = (r nin bit cinsinden uzunluğu-1)

olacak şekilde ayarlanır. Burada r Eliptik Eğri veya Ayrık Logaritma taban noktasının derecesidir.

3. Eğer seçilen imza ilkeli DLSP-DSA veya ECSP-DSA ise, mesaj örneğinin maksimum uzunluğu l, r' nin bit cinsinden uzunluğuna eşit olacak şekilde ayarlanır (l = r).

4. M mesajından sabit uzunlukta bir f mesaj örneği elde edilir.

5. Gizli anahtar s ve mesaj örneği f üzerinde seçilen imza ilkeli uygulanarak (c,d) imzası oluşturulur.

6. İmza çıkışa aktarılır.

6.2.1.2. İmza onaylama işlemi

M mesajı üzerinde bir (c, d) imzası aşağıdaki şekilde onaylanır:

1. İmzacının açık anahtarı w ve ona bağlı alan parametreleri elde edilir

2. *(Seçime bağlıdır)*) Açık anahtar *w* ve ona bağlı alan parametrelerinin geçerliliği onaylanır. Geçersiz sonucuna ulaşılırsa burada imza onaylama işlemine son verilir.

3. Seçilen onaylama ilkeli DLVP-NR veya ECVP-NR ise

   a. Seçilen onaylama ilkeli, f mesaj örneğini geri elde etmek amacıyla imzaya ve imzacının açık anahtarına uygulanır. İlkelin çıkışı geçersiz değer döndürürse, imza geçersizdir.

b. Mesaj örneğinin maksimum uzunluğu l = (r'nin bit uzunluğu)-1 olarak ayarlanır. r, Ayrık Logaritma veya Eliptik Eğri alan parametreleri kümesinin taban değerinin derecesidir.

c. Seçilen mesaj kodlama metodunun onaylama işlemi kullanılarak f tamsayısının doğru mesaj örneği olup olmadığı kontrol edilir. "Geçerli" veya "Geçersiz" çıkışı üretilir.

4. Seçilen onaylama ilkeli DLVP-DSA veya ECVP-DSA ise

a. Mesaj örneğinin maksimum uzunluğunu l = (r'nin bit uzunluğu) olarak ayarlanır. r, Ayrık Logaritma veya Eliptik Eğri alan parameteleri kümesinin taban değerinin derecesidir.

b. Seçilen mesaj kodlama metodu kullanılarak M mesajından maksimum uzunluğu l olan bir f mesaj örneği üretilir.

c. Seçilen onaylama ilkeli f tamsayısına, imzaya (c,d) ve imzacının açık anahtarına uygulanarak bu değerlerin uygun olup olmadıkları kontrol edilir. İlkelin çıkışı geçersiz değer döndürürse, imza geçersizdir.

Ekli imza düzenlerinde mesaj kodlama metodu olarak genellikle çırpma fonksiyonları kullanılır.

6.3. Eliptik Eğri Ayrık Logaritma Problemine dayanan ilkeller

Eliptik Eğri Grupları üzerinde tanımlı ayrık logaritma problemine dayalı ilkellerdir. Eliptik eğri protokolleri veya düzenleri, bu ilkelleri kullanır. Bu nedenle öncelikle bu ilkellerden tez çalışmasında kullanılanların tanıtımına yer verilecektir.

6.3.1. Eliptik Eğri alan parametreleri

Eliptik Eğri alan parametreleri tüm Eliptik Eğri ilkelleri ve düzenlerinde kullanılır. Alan parametreleri kümesi, bir GF(q) alanı belirler.

Burada p poziitf bir asal sayı, m de pozitif tam sayı olmak üzere q = p veya q = 2^m 'dir. Eliptik Eğri katsayıları a ve b GF(q)' nun elemanları, E eliptik eğrisini tanımlarlar. Bir pozitif asal tamsayı olan r, E eliptik eğrisi üzerindeki nokta sayısının asal böleni, G noktasının derecesidir. G, r derecesinde bir alt grup üreten bir eğri noktasıdır.

6.3.2. ECSVDP-DH

ECSVDP-DH, Eliptik Eğri Gizli Değer Türetme İlkeli, Diffie-Hellman versiyonudur. [33] Diffie-Hellman[12], Koblitz[23] ve Miller[24]'in çalışmalarına dayanır. Bu ilkel, taraflardan birinin gizli anahtarını karşı tarafın açık anahtarını kullanarak bir ortak giz türetir. Her iki tarafın Eliptik Eğri alan parametrelerinin ortak olduğu varsayılmıştır. Her iki taraf da bu ilkeli hatasız işlettiklerinde aynı sonucu elde ederler. Bu ilkel bir ortak giz türetme düzeni tarafından çağrılabilir. Özel olarak ECKAS-DH1 ve DL/ECKAS-DH2 anahtar ortaklaştırma düzenleri tarafından çağrılabilir. Giriş anahtarlarının geçerli olduğu varsayılır. Eliptik Eğri Gizli Değer Türetme İlkelinin giriş değerleri, varsayımlar, çıkış değerleri, ve işlemler aşağıda tanıtılmıştır.

Giriş:

-Eliptik Eğri alan parametreleri q, a, b, r ve G

-ilkeli işleten tarafın kendi gizli anahtarı s

-karşı tarafın açık anahtarı W

(Açık anahtarın derecesi r ile gösterilirse gizli anahtar s, [1, r-1] aralığındadır.

Varsayımlar: Gizli anahtar s, Eliptik Eğri alan parametreleri q, a, b, r, ve G; açık anahtar W'nin geçerli olduğu kabul edilir. Her iki anahtar da alan parametreleri ile uyumludur.

Çıkış: Ortak giz değeri (z) türetilir veya hata çıkışı üretilir.

İşlem: Ortak giz değeri aşağıdaki işlem dizisinin takip edilmesiyle türetilir.

1. P = sW olacak biçimde bir Eliptik Eğri noktası hesaplanır.

2. Eğer P = O ise hata çıkışı üretilerek işlem sonlandırılır.

3. z, P noktasının x koordinatına eşittir. (x_p)

4. z, ortak giz değeri olarak çıkışa aktarılır.

6.3.3. ECSP - NR

ECSP-NR, Eliptik Eğri İmza İlkeli , Nyberg-Rueppel versiyonudur. [34] Koblitz [24], Miller [25], ve Nyberg ve Rueppel'in [36] çalışmalarından temel alır. Bir düzenin içinde imzacının gizli anahtarı ile mesaj örneğini imzalamak için çağrılabilir. Mesaj örneğini imzadan yola çıkarak geri elde etmek için imzacının açık anahtarı kullanılır. Bu işlem ECVP-NR ilkeli ile yapılır. P1363 standardında mesaj çözmeli Eliptik Eğri imza düzenleri tanımlanmamıştır. ECSP-NR ekli imza düzenlerinde kullanılabilir ve DLSSA düzeninin içinde imzalama amacıyla çağrılabilir.

Giriş:

- Eliptik Eğri alan parametreleri q, a, b, r, ve G

- İmzacının gizli anahtarı s

- mesaj örneği f, [0,r-1] aralığında bir tamsayıdır.

Varsayımlar: Gizli anahtar *s* ve Eliptik Eğri alan parametrelerinin (q, a, b, r, ve G) geçerli ve birbirleri ile uyumlu oldukları varsayılmaktadır.

Çıkış: İmza, (c,d) tamsayı çiftinden oluşur. ($1 \leq c < r$ ve $0 \leq d \leq r$)

İşlem: İmza, (c, d) aşağıdaki işlem adımları işletilerek oluşturulur.

1. Bir anahtar çifti (u, V) türetilir. (Bu anahtar çifti bir kullanımlıktır.) Bu anahtar çifti, s gizli anahtarı ile aynı alan parametrelerine sahiptir. V noktasının koordinatları (x_v,y_v) olarak tanımlanabilir.

2. x_v, FE2IP ilkelini kullanarak bir i tamsayısına dönüştürülür (x_v ,GF(q)'nun bir elemanıdır).

3. c = i + f mod r olacak biçimde c tamsayısı hesaplanır. Eğer $c = 0$ sonucu elde edilirse 1. adıma geri dönülür.

4. d = u – s.c mod r olacak biçimde d tamsayısı hesaplanır.

5. (c, d) tamsayı çifti imza olarak çıkışa aktarılır.

6.3.4. ECVP-NR

ECVP-NR, Eliptik Eğri Onaylama İlkeli, Nyberg-Rueppel versiyonudur [34]. Koblitz [24], Miller [25], ve Nyberg ile Rueppel'in [36] çalışmalarından temel alır. İmzacının imzası ve açık anahtarını kullanarak ECSP-NR ilkeli ile imzalanmış mesaj örneğini geri elde eder. Bir düzen içinde imza onaylama veya mesaj çözme amacıyla çağrılabilir. P1363 standardının bu versiyonunda Eliptik Eğri imzalama düzenleri mesaj çözmeli olarak tanımlanmamıştır. ECVP-NR, ekli imza düzenleri içinde kullanılabilir ve DLSSA içinde imza doğrulama amacıyla çağrılabilir.

Giriş:

– Eliptik Eğri alan parametreleri q, a, b, r, ve G

- İmzacının açık anahtarı, W

- Onaylanacak imza, (c, d) tamsayı ikilisi

Varsayımlar: Açık anahtar W ve Eliptik Eğri alan parametrelerinin (q, a, b, r, ve G) birbirleriyle uyumlu ve geçerli oldukları kabul edilir.

Çıkış: Mesaj örneği, $0 \leq f < r$ olacak biçimde bir f tamsayısıdır, veya imzanın geçersiz olduğunu belirten mesaj.

İşlem: Mesaj örneği aşağıdaki işlem adımları izlenerek hesaplanır:

1. Eğer c, $[1, r - 1]$ aralığında değilse veya d, $[0, r - 1]$ aralığında değilse, "imza geçersizdir"çıkışı üretilir.

2. Eliptik Eğri üzerinde bir nokta P = d.G + c.W hesaplanır. Eğer P = O sonucu elde edilirse , "imza geçersizdir"çıkışı üretilir; diğer durumda hesaplanan noktanın koordinatları P = (x_p,y_p) olarak gösterilebilir.

3. x_p, FE2IP ilkeli yardımıyla i tamsayısına dönüştürülür.

4. f = c – i mod r tamsayısı hesaplanır.

5. f mesaj örneği olarak çıkışa aktarılır.

7. ZEIT CONTROL KİTİ VE AKILLI KART PROGRAMLAMA

BasicCard, Zeit Control firmasının ürettiği programlanabilir bir işlemcili karttır. Zeit Control Kiti, bireysel program geliştirme aracı olarak tasarlanmıştır. Akıllı kart üzerindeki programlar Basic dilinde yazılır. Zeit Control bu yapıda iki tür akıllı kart üretmiştir. Compact BasicCard ve Enhanced BasicCard. Compact BasicCard daha basit işlevlere sahip, 1K EEPROM'a sahipken Enhanced BasicCard 8K ve 16K EEPROM genişliğine sahip ve daha karmaşık işlevleri olan bir üründür.

7.1. BasicCard'ın yapısı

Basic Card'ın 256 Byte RAM ve 1K veya 16K olarak tercih edilebilen EEPROM genişliği vardır. EEPROM'un 1K'lık olduğu durumda veriler bir değişken olarak tutulmaktadır, 16K'lık olduğu durumda ise dosya yapısı içinde saklanabilmektedir.

RAM üzerinde çalışma sırasındaki (run-time) veriler ve P-code yığın tutulur. P-code, Basic kodunun derlenmesi sonucu oluşan bir sanal makina dilidir. (Java da kullanılan teknolojinin benzeri burada da kullanılmaktadır.) P-code'u oluşturmak ve akıllı karta yüklemek için Zeit Control'ün MSDOS ve Windows altında çalışabilen destek yazılımı kullanılabilir. Bu yazılım paketleri, kart okuyucu olmadan da yazılımın test edilebilmesini sağlar.

İşlemci kart tasarımında en önemli başlık güvenliktir. İşlemci kartların kullanımında asıl amaç güvenliktir. Karttaki işlemci, belleğe erişimi engeller.

7.1.1. BasicCard işletim sistemi

BasicCard işletim sistemi, tüm haberleşmeyi yürütür. Komutları çözer, şifreler ve yanıtlar.

Haberleşme çift yönlü I/O kontakları üzerinden 9600 baud' ta T=1 protokolüne göre sağlanır. Protokol, ISO standartlarına göre tanımlıdır[2,3]. Bu protokol kullanıcı için arka planda kalır. Yapılması gereken, kartta işleyecek komutun tanımlanması ve bu komutun bir Basic prosedürü olarak programlanmasıdır.

İşletim sisteminde akıllı karta programları ve verileri yüklemeyi sağlayan, kart üzerindeki şifrelemeyi aktif hale getiren, vb önceden tanımlı komutlar bulunmaktadır. Zeit Control P-Code'u çalıştırmak üzere bir sanal makina tanımlanmıştır. Derleyici, **WZCBASIC.EXE** ZC-Basic kaynak kodunu derleyerek ve P-Code'u oluşturur. P-Code, bir sanal makina üzerinde çalışan makina kodu olarak düşünülebilir. P-Code **WBCLOAD.EXE** kart yükleme programı kullanılarak karta yüklenir. Bu sayede BasicCard'ta bulunan sanal makina, P-Code komutlarını çalışma süresince işletebilir.

Basic Card'da ayrıca tanımlanacak her komut için komut tanım kodu (ID) atanır. Her komut için farklı iki sekizli'den oluşur. Bir işlemci kartla haberleşme, Komut-Yanıt protokolü ile yapılır. Kart okuyucuya takılır takılmaz protokol oturumu başlatılır. Protokol, reset komutu ve kartın yanıtı ile başlar. Akıllı kart üzerinde herhangi bir işlemin gerçekleşebilmesi için komutların çağrılması gerekir. PC üzerinde koşan ZC-Basic programından bir komut çağrılır. Tanımlanan her komut için farklı bir iki-byte ID atanır. Buradaki amaç, akıllı kart üzerinde komutların fazla yer tutmasını engellemektir.

Enhanced BasicCard'ta 8K ve 16K olarak değişen EEPROM, 17K ROM ve 256K RAM vardır. Bunlar içinde Basic Card işletim sistemi 107 Byte RAM ve 338 Byte EEPROM kullanır.

7.1.2. ROM'daki algoritmalar

Şifreleme ve şifre çözme komutları, ROM'da tutulur. Bazı komutlara akıllı kartın vereceği yanıtların DES kullanması gerekir. Bu yanıtların kullandıkları kod ROM'da tutulur. Kullanılan akıllı kart **tekli DES** ve üçlü **DES** desteklidir. ROM'da kataloglama tabanlı, DOS benzeri bir dosya sistemi vardır. IEEE uyumlu, kayan noktalı aritmetik de desteklenmektedir.

7.1.3. Ek şifreleme özellikleri

Kartın yapısında olmamakla birlikte EEPROM'a yüklenecek kütüphanelerle şifreleme ve şifre çözme algoritmaları geliştirilebilir. Zeit Control BasicCard program geliştirme kiti için bazı şifreleme algoritmalarının desteklendiği kütüphaneler bulunmaktadır. Bu kütüphaneler aşağıdaki şekilde sıralanabilir:

EC–161: 161-bit Eliptik Eğri Kriptografi üzerinde kurulu algoritmaları içerir. IEEE standardı P1363'e göre yazılmıştır (Bölüm 6'ya bakınız).

SHA–1: MD4 algoritması üzerine kurulu bir güvenli çırpma algoritmasıdır [37].

IDEA: Uluslararası veri şifreleme algoritması olarak bilinir. İlk kez 1990'da ortaya atılmış bir blok şifreleme algoritmasıdır [38].

7.2. Yazılım destek paketi

ZCBASIC: ZC Basic programlama dili derleyicisi.

ZCDD split ekran "Double Debugger": Windows işletim sistemi altında çalışır. Terminal kodunu ve Basic Card kodunu simültane olarak izlemeyi sağlar.

ZCDOS: (P-Code interpreter) derlenmiş ZC-Basic programlarını MSDOS altında çalıştırır. Terminal programını ve BasicCard programını simültane olarak çalıştırır. Bunun için BasicCard'ı simüle eder ve seri port üzerinden BasicCard ile haberleşmeyi de sağlar.

BCLOAD: P-Code'u Akıllı karta yüklemek için kullanılır.

KEYGEN: Şifrelemede kullanılmak üzere rastgele anahtar ve ilkel polinom üretir.

BCKEYS: Cryptographic anahtarları akıllı karta yükler.

7.3. BasicCard Programlama arayüzü

Basic Card programları ZC Basic dilinde yazılır.

7.3.1. Başlangıç kodu

Terminalden kullanıcı tanımlı ilk kod çağrıldığında işletilen kod, başlangıç kodudur. Her zaman gerekli olmamakla birlikte kartın yayımcı tarafından iptal edilip edilmediğini kontrol etmek için veya beklenen dosya veya klasörlerin varlığını test etmek için kullanılabilir.

7.3.2. Prosedür tanımları

ZC Basic'te üç tip prosedür vardır. İçiçe prosedür tanımlarına izin verilmez.

Subroutines: Diğer prosedürler tarafından çağrılabilen program parçasıdır.

Functions: Çağrıldığı yere bir değer döndüren **Subroutine**'dir.

Commands: ZC-Basic diline özgü bir yapıdır. Terminal programının işlemci kart programıyla haberleştiği mekanizmadır.

ISO/IEC 7816-4 [3]'e göre, her komuta bu komutun hangi sınıfa ait olduğunu ve bu sınıfın içinde hangi komut olduğunu belirtmek üzere iki sekizliden oluşan ID atanmıştır (CLA-class, INS-instruction).

"Command" anahtar sözcüğü ile komutun adının arasına yazılarak komutun ID'si belirtilir. &H heksadesimal yazılımı göstermek üzere aşağıdaki örnekte :

```
Command &H80 &H10 GetCustomerName (Name$)
    Name$ = CustomerName$
End Command
```

BasicCard, Terminalden CLA=&80 ve INS=&H10 olan bir komut aldığında, akıllı kartın işletim sistemi otomatik olarak **GetCustomerName** komutunu işletecektir.

Her komut, fonksiyon ve alt-yordam arasında geçiş noktasıdır. Yukarıdaki örnekte olduğu gibi alt-yordama benzer biçimde tanımlanır fakat çağrılırken fonksiyon

çağrısına benzer biçimde çağrılır. BasicCard işletim sistemi Terminal programına geri döndürülecek dönüş değerini atar. Bu dönüş değeri iki durum bitinden oluşur: SW1 ve SW2. Durum bitleri ISO/IEC 7816-4 standardında tanımlanmıştır. Bir komutun dönüş değerinin her durumda kontrol edilmesi gerekir. Örneğin kart, okuyucudan çıkartılmış olabilir veya okuyucunun gücü bir nedenle kesilmiş olabilir. Eğer SW1 = &H90 ve SW2= &H00 değerlerine sahipse veya eğer SW1 = &H61 değerine sahipse, komutun başarımı doğrulanmış olur. Diğer durumda komut başarıma ulaşmamış demektir.

Bu iki durum biti akıllı kart içinde önceden tanımlanmış değişkenlerdir. Bu sayede hata kodu tanımlanabilmektedir. Erişim kolaylığı açısından iki sekizliden oluşan tamsayı değişken SW1 ve SW2' de tanımlıdır. Komut çağrılmadan önce bu iki değer H90 ve H00 olarak belirlenir. Eğer aşağıda olduğu gibi SW1 ve SW2 'ye farklı değerler atanırsa işletim sistemi yanıt verisini bir kenara atar ve Terminal programına bu iki durum bilgisini gönderir. Burada durum bilgileri hata mesajı anlamına gelir. Bu ISO/IEC 7816-4 standartları ile belirlenmiş bir özelliktir. Örnek:

```
Eeprom Balance As Long : Rem Declare permanent (Eeprom)
variable
Const InsufficientCredit = &H6F00
Command &H80 &H20 DebitAccount (Amount As Long)
    If Balance < Amount Then
        SW1SW2 = InsufficientCredit
    Else
        Balance = Balance - Amount
    End If
End Command
```

Hata kodu olarak istenen değer atanabilmesine rağmen ISO standartları ile uyum gözetilmiştir. ISO'ya göre SWI'nın yüksek anlamlı biti en fazla 6 olabilir. Ayrıca dikkat etmek gereken bir diğer nokta da ZC-Basic'in önceden tanımlı hata kodlarına yeni hata mesajları atanmamasıdır. ZC-Basic hata kodları listesi mevcuttur [39]. Hata kodu olarak **SW1 = &H6B, &H6C, veya &H6F** kullanılırsa çakışma önlenmiş olur.

7.3.3. Dosya tanımlama

Enhanced Basic Card'ta ağaç yapısında düzenlenmiş DOS benzeri bir dosya sistemi mevcuttur. Dosyalara erişmek için çeşitli yöntemler izlenebilir:

BasicCard'ın içinde, DOS veya Windows altında çalışan bir Basic programındakine benzer biçimde dosya yaratılabilir, okunabilir ve yazılabilir. Dosya ve dizinlere terminal programlarından erişimi kısıtlamak için bazı özel ifadeler mevcuttur. Bu erişim koşulları kriptografik anahtarlar, kullanıcı şifresi vb. kullanılarak düzenlenir.

Terminal programı tarafından bakıldığında BasicCard'a erişim hakları düzenlenmiş olmakla birlikte bir sürücüye erişmek gibi düşünülebilir.

Dosyaları ve dizin yapılarının başlangıç durumlarını belirlemek için BasicCard programında Dosya Tanımlama Bölümleri kullanılır.

7.3.4. Kalıcı veriler

Akıllı kartta saklanması düşünülen veriler EEPROM'da tutulur. Enhanced BasicCard'ta bu veriler dosyalanarak tutulabilir. Daha düşük kapasiteli (1K) Compact Basic Card'ta dosya yapısı yoktur. Örneğin aşağıda tanımlanan "balance" isimli bir değişken tanımı yer almaktadır:

Eeprom Balance As Long : Rem Declare permanent (Eeprom) variable

EEPROM'da string veya array tipinde değişken tanımlamak da mümkündür. EEPROM'a bir yazma işleminin süresi 6 milisaniye'dir. Bu nedenle yazma işlemi sırasında kartın elektriğinin kesilme riski her zaman mevcuttur. Enhanced BasicCard böyle bir durumda hatayı onarabilmek için otomatik olarak tüm EEPROM yazma işlemlerinin kayıtlarını tutar. Compact Basic Card'ta ise böyle bir düzeltme mekanizması yoktur. Bu nedenle bu tür akıllı kartlarda veri kaybını engellemek amacıyla kaybedilmesi istenmeyen verilerin yedeğinin uygun bir yerde tutulması gerekir.

7.4. Terminal tarafı program geliştirme

Terminal programı, istemci tarafında, PC üzerinde çalışan ZC-Basic programıdır. Akıllı kart üzerinde çalışan diğer bir programla haberleşir. Derleyici, Terminal program kaynak kodundan çalışabilir dosya veya bir görüntü dosya üretir.

Çalışabilir dosya:

ZCBasic derleyicisi, standart .exe uzantılı dosya üretebilir. Bu tür programlar simülasyon üzerinde çalışamazlar. Akıllı kartın karşı tarafta var olması gerekir. Ayrıca bu programlar, Write Eeprom ifadelerini de çalıştıramazlar.

Görüntü Dosyalar:

Program geliştirmede esneklik açısından derleyici Terminal programı kaynak dosyasını kullanarak (.IMG uzantılı) ZeitControl Görüntü Dosyası oluşturur. ZCDOS P-Code yorumlayıcısı, bu Terminal programını Akıllı Kart programıyla birlikte Gerçek veya Simüle edilmiş Akıllı Kart üzerinde çalıştırır.

7.4.1. Terminal programı arayüzü

Bir Terminal Programı ana program ve alt program tanımlarından oluşur. Akıllı kart komutları, "command declarations" içinde tanımlanır. Komutlar tanımlandıktan sonra fonksiyonlar gibi çağrılabilirler.

Komut Bildirimleri (Command Declarations):

Bir akıllı kart komutunu çağırmadan önce, ZC-Basic derleyicisinin komutun ID'sini ve komut parametrelerini bilmesi gerekir. Bunun için komutun önceden tanımlanması gerekir. ID bölümü hariç Command Declaration, bir alt program tanımına çok benzer.

```
Declare Command &H80 &H10 GetCustomerName (Name$)
Declare Command &H80 &H20 DebitAccount (Amount As Long)
```

Komut çağrıları ise fonksiyon çağrısı gibidir:

```
Status = GetCustomerName (Name$)
```

```
If Status <> &H9000 And (Status And &HFF00) <> &H6100
Then
    Print "GetCustomerName: Status = &H"; Hex$ (Status)
    GoTo Retry
End If
```

Komutun kendisinin hata koşulları olmasa da bağlantı problemi olasılığına karşı her durumda dönüş değeri kontrol edilmelidir. COMMANDS.DEF dosyası, durum kodlarını tanımlar. Bu dosya program içerisine eklendiğinde &H9000 ve &H61 kodları **swCommandOK** ve **sw1LeWarning** komutları için sırasıyla tanımlanmış olur. Bir başka yöntem de COMMERR.DEF dosyasında tanımlanmış olan subroutine **CheckSW1SW2()**'yi çağırmaktır. Haberleşme hatası oluşması durumunda bu alt program, uygun bir hata mesajı üretilerek programdan çıkışı sağlar.

Karta veri yazılması:

Terminal programında, EEPROM'a yazılacak veri, iki koşulda görüntü dosyaya yazılır:

1.Programdan çıkarken, eğer uygun seçenekler belirtilmişse,

2.Terminal Programı, Write Eeprom ifadesini çalıştırdığında. (Bu ifade Terminal Programının yalnızca ZCDOS P-code yorumlayıcısı veya Windows Debugger içinde çalıştığı koşullarda geçerlidir. Bu ifadeyi içeren programlar .exe dosya içine derlenemez.)

7.5. ZC-BASIC Dili

ZC-Basic programlama dili fonksiyon ve alt-program çağrıları, kullanıcı tanımlı veri tipleri, dosya G/Ç, ve işlemci öncesi komutlar gibi yapılardan oluşmaktadır. Ek olarak akıllı kart çevre birimi için bazı özellikler içermektedir. Bunlar akıllı kart için komut tanımlama, G/Ç şifreleme, dosya erişim kontrolü ve EEPROM yazma günlüğü olarak sıralanabilir.

7.5.1. Kaynak dosya

ZC-Basic programlama dilinde tek bir derleme birimi vardır. Bağlama (linking) aşaması yoktur. Bu durum derleyicinin 256 byte RAM'in tamamını tek başına kullanması olanağını yaratır. Buna karşın kaynak kodu çeşitli alt birimlere bölerek tek bir ana kaynak dosya altına #include ile toplamak mümkündür.

7.5.2. Jeton yapısı

En düşük düzeyde bir kaynak kod, bir dizi jetondan oluşur. Dört çeşit jeton vardır: Sabitler, tanımlayıcılar, rezerve kelimeler ve özel semboller. Katar sabitler hariç jetonlar boşluk içermeyebilir.

Sabitlerin veri tipi tam sayı, kayan noktalı veya katar olabilir. Tam sayı sabitler varsayım olarak ondalıklı kabul edilir. &O öneki 8'lik, &H öneki ise 16'lık düzende olduğunu gösterir. Kayan noktalı sayılarda ZC-Basic tek basamakta hassasiyeti destekler. Tanımlayıcılar, harf, rakam ve ardından veri tipini tanımlayan bir özel sembolden oluşur.

Karakter:	@	%	&	!	$
Veri tipi:	**Byte**	**Integer**	**Long**	**Single**	**String**

Eğer tipi belirten karakter mevcut değilse, veri tipi integer varsayılır. Bu varsayım DefByte, DefLng vb. kullanılarak değiştirilebilir. ZC-Basic'in bir başka özelliği, küçük ve büyük harf ayrımı yoktur. Tanımlayıcıların rezerve kelimelerle çakışmamasına dikkat etmek gerekir.

Tanımlayıcılar, sabitler ve rezerve kelimeler dışında aritmetik, lojik işlem sembolleri ve başka özel semboller tanımlıdır.

7.5.3. İşlemci öncesi komutlar

ZC-Basic derleyicisine derleme öncesi verilen talimatlardır. '#' karakteri ile başlayan satırlar derleyici tarafından "işlemci öncesi komutlar" olarak değerlendirilir.

1. Kaynak dosya ekleme: **#include** *filename*

Bu şekilde eklenen dosyadaki kaynak kod, kodun bir parçasıymış gibi derlenir. Derleyici yukarıdaki biçimde adı belirtilen dosyayı ararken önce içeren dosyanın bulunduğu dizinin altına bakar. Ardından komut satırında yazıldıkları sıraya göre -I parametreleri ile özelleştirilen dizinde eklenecek dosyayı arar. Daha sonra içinde bulunulan dizinin altına, orada bulamadığında ise ZCINC çevre değişkeninde tanımlanmış dizinlere bakar.

2. Kütüphane ekleme: **#Library** *filename*,

talimatı, ZeitControl Plug-In kütüphanelerini ekler. Zeit Control'de her kütüphane dosyası library.lib için bir tanım dosyası library.def vardır. Tanım dosyası, tüm gerekli tanımlarla birlikte tahsis edilen #Library talimatını içerir.

3. Koşullu Derleme:

Komut satırında veya daha önce tanımlanmış sabitlerin değerine göre derleyici kodun bazı parçalarını derleyip derlememeye karar verir. Derleme aşamasında yapılan bu içleme dışlama işlemi için if, elseif, else ve endif komutları kullanılır. Koşullu derleme de derleme öncesi talimat olarak yorumlanacağından başlarına # sembolü getirilir.

4. Listing talimatları:

Kodun bazı parçaları listing dosyasından çıkarılmak istendiğinde **#NoList**, talimatı kullanılır.

5. Kart durumu:

ZC-Basic programı akıllı karta yüklendikten sonra kart TEST durumuna geçer. Eğer kartın başka bir duruma geçmesi tercih ediliyorsa **#state** talimatı kullanılarak derleyiciye bildirilir.

6. Açık dosya sayısı belirleme:

ZC-Basic programında Terminal tarafı için aynı anda 32 dosya açık tutulabilir yani açık dosya slotu sayısı 32'dir. Bu sayı Enhanced Basic Card için 2'dir. Açık Dosya Slotu sayısının varsayılan değerleri derleyici talimatları kullanılarak

0<=nfiles<=16 olmak koşuluyla değiştirilebilir. Bunun için aşağıdaki talimat kullanılır:

#Files *dosyasayısı*

Bu durumda dosya sistemi tarafından kullanılacak RAM, *6*dosyasayısı+7* ile sınırlanmış olur.

7. Yığın Büyüklüğü:

P-Code yığını için ayrılacak alan **#stack** *yığın-genişliği*, talimatı ile belirlenebilir.

8. EEPROM Genişliği:

Kartın EEPROM'unun başlangıç ve bitiş adresini belirler.

#Eeprom [başlangıç] [**To** bitiş]

9. Mesaj talimatı:

Derleme sırasında bir mesaj çıktısı üretmek mümkündür. Derleme sürerken mesaj ekrana bastırılabilir. **#Message** *mesaj*

10. Hata talimatı:

#Error, komutu kullanılarak hata mesajlarını kendi istenen biçimde tanımlamak veya kendi hata mesajlarımızı üretmek mümkündür.

```
#If MaxLineLength > 80
#Error MaxLineLength too big (max 80)
#EndIf
```

Örnek program MaxLineLength=100 olacak biçimde derlendiğinde, derleyici "**#Error MaxLineLength too big (max 80)**" hata mesajını üreterek derlemeyi sonlandıracaktır.

11. Blok bekleme zamanı:

Bir BasicCard programında "reset'e yanıt" içindeki BWT(Blok Bekleme Zamanı) alanı aşağıdaki şekilde özelleştirilebilir. $1 < 2^n \leq 256$ olmak üzere,

#BWT *n*

Blok bekleme zamanı, kart okuyucu **swCardTimedOut** durumuna dönmeden önce karta bir komutu işlemek için tanınan süredir. Saniyenin ondabiri ile ifade edilir. Varsayılan değeri Compact Basic Card'ta 16, Enhanced Basic Card'ta ise 128'dir. (1.6 saniye ve 12.8 saniye anlamına gelir.) Blok bekleme zamanının alabileceği en büyük değer 51.2 saniyedir.

12. Önceden tanımlı sabitler:

Programın çalışacağı ortama göre (Terminal, Compact Basic Card veya Enhanced Basic Card) üç sabitten birinin değeri derleyici tarafından 1 olarak önceden tanımlanır. Bu üç sabit, *TerminalProgram, CompactBasicCard* ve *EnhancedBasicCard* 'tır.

7.5.4. Veri saklama

ZC-Basic programlama dilinde tüm değişkenler dört veri saklama sınıfından birine aittir: *Eeprom, Public, Static*, veya *Private.*

1.Eeprom verileri:

EEPROM'da saklananlar: derlenerek P-Code'u oluşturulmuş ZC-Basic programı, dizinler, dosyalar ve kalıcı değişkenlerdir. Program derlenirken değişkenlere bir başlangıç değeri atanmadığı durumda P-Code karta yüklenirken başlangıç değeri olarak sıfır atanır.

Eeprom verileri global değişkenlerdir. Programdaki tüm alt-programlar tarafından erişilebilirler.

2.Public ve Static veriler:

RAM'de işlenen veriler **public** veya **static** olarak ikiye ayrılırlar. **Public** değişkenler global, **static** değişkenler yerel değişkenlerdir.

3.Private veriler:

Prosedür içinde "private" olarak tanımlanan veriler yalnızca tanımlandıkları alt program içinde geçerlidirler. Alt-programdan çağrılan yere dönüldüğünde geçerliliklerini yitirirler. Bu nedenle P-Code yığınında alt programa dallanıldığında yer alınır ve alt programda dönülürken de yığından alınan alan yığına geri verilir.

Tüm verilerin private, public veya static olarak tanımlanması zorunlu değildir. Derleyici tanımlanmamış değişkenleri varsayım olarak "private" kabul eder.

7.5.5. Veri tipleri

ZC-Basic'te 6 tip veri desteklenir: Byte, integer, Long, Single, String, String*n. Ayrıca bunların dışında kullanıcı kendisi tip tanımlayabilmektedir.

7.5.6. BasicCard'a özgü yapılar

1.Özelleştirilmiş ATR:
BasicCard resetlendiğinde, kendisi hakkında bilgiyi ATR(Answer to Reset-Reset'e yanıt) aracılığıyla verir. **ATR,** akıllı kartın kullanacağı haberleşme parametreleri hakkında bilgi içerir. 15 sekizliye kadar kullanılabilecek "tarih karakterleri" adı verilen bu alanda kart kendini tanıtır. BasicCard içindeki tarih-karakterleri'nin "**BasicCard ZC***vvv*" biçimindedir. Burada *vvv* kartın yazılım versiyon numarasıdır. (Compact BasicCard'ın yazılım versiyon numarası 1.1.'dir. EnhancedBasicCard'ların yazılım versiyon numaraları ise 2.3 veya 2.4.) "Tarih karakterleri" **Declare ATR** ifadesi kullanılarak değiştirilebilir.

Declare ATR = *veri*

Veri, **Byte** ve **String** sabitlerden oluşan, toplam uzunluğu 15'ten küçük olan herhangi bir diziliş(sekans).

2. Uygulama ID'si:
BasicCard'ta önceden tanımlı **GET APPLICATION ID** komutu vardır. ZC-Basic kaynak kodunda belirlenmiş uygulama ID'sini döndürür. Bu komut kart okuyucudaki akıllı kartın istenen uygulamayı içerip içermediğini kontrol etmek için kullanılır. Bir uygulamanın ID'sini düzenlemek için :

Declare ApplicationID = *veri*

veri, **Byte** ve **String** sabitlerden oluşan, toplam uzunluğu ≤ 127 olan herhangi bir diziliştir(sekans).

3. Şifreleme algoritmalarını etkinleştirme ve etkisizleştirme:

{**Enable** | **Disable**} **Encryption** [*AlgoritmaID* [, *AlgoritmaID*, . . .]]

AlgoritmaID, Bir şifreleme algoritmasının ID'sidir. Eğer herhangi bir algoritma özelleştirilmemişse tüm var olan algoritmalar etkin veya etkisiz hale getirilir. Aşağıda var olan algoritma ID'leri verilmiştir:

Enhanced BasicCard: &H21 **Single DES**
&H22 **Triple DES**

Maksimum güvenlik açısından kullanılmayacak şifreleme algoritmalarının etkisizleştirilmesi gerekir. (En güvenli algoritma Enhanced BasicCard'ta &H22'dir.)

Bu komut, program derlenirken çalıştırılır ve kartın yaşamsüresi boyunca geçerlidir. Algoritmalar işlem sırasında (run-time) etkin veya etkisiz hale getirilemezler.

4. Terminalden süre istemek:

Eğer bir komutun işlem zamanı verilen süreden fazla olacaksa çalışabilmek için ek süre istemesi gerekir; aksi taktirde caller time-out olur. Bunun için WTX komutu kullanılır. BasicCard'ta bir Blok Bekleme Süresi, **BWT** (Block Waiting Time), mevcuttur. BWT, Compact BasicCard'ta 1.6 saniye Enhanced BasicCard'ta 12.8 saniyedir.

5. Önceden tanımlı değişkenler:

BasicCard işletim sisteminde ZC-Basic dilinden erişilebilen iç değişkenler vardır. Çoğu haberleşme ile ilgilidir.

7.5.7. Terminal tarafına özgü yapılar

1.Ekran Çıkışı:

Ekran çıkışı için **Cls** ve **Print** ifadeleri önceden tanımlanmış dört değişkenle birlikte kullanılır: **FgCol**, **BgCol**, **CursorX**, ve **CursorY.**

Cls komutu ekranı temizler ve **CursorX** ve **CursorY** değişkenlerini 1'e eşitler. **Print** komutu:

Print [*alan* | *ayraç*] [*alan* | *ayraç*] . . .

alan, **Byte**, **Integer**, **Long**, **Single**, veya **String** tipinde herhangi bir ifade

ayraç, ';' (noktalı-virgül), çıkışta kursörün değişmeden kalmasını sağlar.

',' (virgül), çıkışta kursörü bir çıkış katarı boyu kadar ilerletir. (Çıkış katarı 14 karakter genişliğindedir.

Spc(*n*), *n* tane boşluk basar.

Tab(*n*), çıkış kursörünü n. pozisyona konumlandırır.

Print ifadesinden sonra ayraç kullanılmadığı durumda kursör bir sonraki satırın başına ilerler.

2. Klavye girişi:

InKey$, 0, 1, veya 2 sekizliden oluşan dizi döndürür: Klavye tamponunda bekleyen karakter yoksa 0 byte; sıradan bir tuşa basıldıysa 1; extended-ASCII tuşuna basıldıysa 2 (ilk byte ın 0 olduğu durum) döndürür.

LineInput *X$,* klavyeden bir karakter katarı değişkeni *X$*'a return tuşuna basılan yere kadar satır okur.

Input *değişken-liste,* klavyeden girilen bir dizi değişkeni okur. Kullanıcı birden fazla değer girişi yaparken arada virgül veya boşluk kullanır.

3. Haberleşme:

Kart okuyucu ve kartın durumunu belirten üç fonksiyon vardır. Bu fonksiyonlar komut çağrıları gibi **SW1–SW2** içinde durum kodunu döndürürler:

CardReader [(*isim$*)]

Seri port üzerinden kart okuyucunun varlığını denetler. Eğer bir karakter katarı parametresi geçerse, kart okuyucuyu tanımlayıcı karakter katarı döner. BasicCard PC'de simüle ediliyorsa "Simulated Card Reader" ifadesi *name$* parametresi içinde döndürülür. SW1-SW2'deki durum kodları:

swCommandOK Kart okuyucu bulundu
swNoCardReader Kart okuyucu bulunamadı
swCardReaderError Kart okuyucu geçersiz yanıt veriyor

CardInReader:

Kart okuyucuya takılıysa **swCommandOK** (&H9000) döndürür. SW1-SW2'deki durum kodları:

swCommandOK Kart okuyucuya takılı
swNoCardReader Kart okuyucu bulunamıyor
swCardReaderError Geçeriz yanıt
swNoCardInReader Okuyucuda kart yok

ResetCard [(*ATR$*)]

Kartı reset'ler. Kart reset isteğine geçerli yanıt üretirse **swCommandOK** (&H9000) döndürür. Eğer string parametre aktarıldıysa ATR "tarih bitleri" geri döndürülür. SW1-SW2'deki durum kodları:

swCommandOK ATR geçerli
swNoCardReader Kart okuyucu bulunamadı
swCardReaderError Kart okuyucu geçersiz yanıt üretiyor
swNoCardInReader Okuyucuda kart yok
swT1Error **T=1** protokol hatası
swCardError Karttan geçersiz yanıt
swCardTimedOut Kart beklenen zamanda ATR gönderemedi

4. PC/SC Fonksiyonları:

Sisteme konfigürasyonu yapılmış PC/SC uyumlu kart okuyucular hakkında bilgi derlemek için iki fonksiyon sağlanmıştır.

nReaders = **PcscCount**

Konfigüre edilmiş kart okuyucuların sayısını tamsayı değer olarak döndürür. SW1-SW2'deki durum kodları :

swNoPcscDriver Sistemde PC/SC sürücüsü yüklü değil.

swPcscError PC/SC sürücüsü beklenmeyen hata mesajı gönderdi.

ReaderName = **PcscReader(***Okuyucuno***)**

Okuyucuno'su verilen PC/SC kart okuyucunun adını **String** olarak döndürür. Okuyucuno sıfıra eşitse varsayılan PC/SC okuyucusunun adı döndürülür. PC/SC okuyucu numarasına erişmek için, önceden tanımlanmış değişken **ComPort** *Okuyucuno*+100'e eşitlenir. SW1-SW2'deki durum kodları:

swNoCardReader *Okuyucuno < 0 veya okuyucuno > nReaders.*

swNoPcscDriver PC/SC sürücüsü sistemde yüklü değil.

swPcscError PC/SC sürücüsü beklenmeyen bir hata kodu döndürdü.

5. I/O Logging:

Open Log File ifadesi Terminal Programı ve BasicCard programı arasındaki tüm I/O logging işlemini başlatır.

Open Log File *dosyaadı,* Log dosyasının geçmişteki içeriği silinir. Dosya açma işlemi başarısız olursa önceden tanımlı değişken **FileError** sıfır olmayan bir değere, hata koduna eşitlenir.

Close Log File ifadesi I/O kaydetme işlemini bitirir ve kayıt dosyasını kapatır.

6. Tarih ve Zaman:

Time$ fonksiyonu 24-karakter sabit formatta o andaki tarih ve zamanı döndürür:

7. Eeprom Veriyi kaydetmek:

Write Eeprom [(*dosyaadı*)]

Kalıcı **Eeprom** verisini Terminal programda bir disk dosyasına yazar. *Dosyaadı* verilmediyse veri asıl görüntü dosyaya (veya izleme dosyasına) geri yazılır. Eğer dosya herhangi bir nedenle açılamadıysa, önceden tanımlı değişken **FileError** bir hata kodu değerine eşitlenir.

Write Eeprom ifadesi yalnızca Terminal programı **WZCDOS** P-Code yorumlayıcısı veya Double Debugger'da çalışıyorsa geçerlidir. **Write Eeprom** ifadelerini içeren programlar exe dosyalar içine derlenemezler.

8. Otomatik Şifreleme:

{ **Enable** | **Disable** } **Encryption**

Terminal programını çalıştıran P-Code yorumlayıcısı tüm komutları **START ENCRYPTION** ve **END ENCRYPTION** komutlarını izlemek için Akıllı Karta monitörler. Akıllı karttan geçerli yanıtlar(response) alan bir **START ENCRYPTION** komutu gördüğünde **END ENCRYPTION** komutu görene kadar otomatik olarak komutların şifrelenmesini ve yanıtların(response) şifrelerinin çözülmesini etkinleştirir. Bu monitör herhangi bir nedenle etkisiz hale getirilmek istendiğinde bu işlem **Disable Encryption** komutu ile yapılabilir. **Enable Encryption** ile istenildiği zaman monitörü etkinleştirmek mümkündür.

9. Karta daha fazla zaman vermek:

Bazen akıllı kart bir komutu çalıştırmak için BWT'den daha fazla zamana ihtiyaç duyabilir. Kural olarak bu sürenin artırılmasını istemek kartın görevidir. **WTX** ifadesi bu amaçla kullanılır. Varsayılan BWT değerini Terminal programdan da **WTX** ifadesini kullanarak değiştirmek mümkündür:

WTX *saniye*

Burada *saniye* değeri 255'e eşit yapılırsa karta tanınan yanıt süresi sınırsız yapılmış olur.

10. Önceden tanımlı Değişkenler:
Terminal P-Code yorumlayıcısı Byte tipinde, önceden tanımlı **Public** değişkenleri içerir.

8. AKILLI KARTTA KULLANILAN GÜVENLİK FONKSİYONLARI

8.1. Komut-yanıt protokolünde şifreleme

Hem Compact BasicCard hem de Enhanced BasicCard için komutların ve yanıtların şifrelendiği bir mekanizma bulunmaktadır. Bu mekanizmayı işler hale getirmek için öncelikle kullanılan iki program çağrısı bulunmaktadır. Bunlar StartEncryption ve EndEncryption'dur [39].

1- Haberleşmeyi yürüten tarafların her ikisinde de kriptografik anahtarları içeren bir anahtar dosyası yaratmak için KEYGEN programı kullanılır.

2- Yaratılan anahtar dosyasını hem terminal hem de akıllı kart tarafına dahil etmek gerekir.

3- StartEncryption ve EndEncryption komutlarını tanımlayabilmek için COMMANDS.DEF dosyası, Terminal programı içerisine dahil edilir.

4- Terminal programı içinde otomatik şifrelemeyi kapatıp açmak için aşağıdaki ifadeler kullanılır:

Call StartEncryption (P1=*algoritma*, P2=*anahtar_no*)

Call EndEncryption ()

Bu bölümde kullanılan $, karakter katarı tipinde değişkenleri belirtmek için kullanılmıştır.

8.1.1. Anahtar bildirimi

Terminal ve akıllı kart programlarının karşılıklı komut ve yanıtları şifreleyebilmeleri için iki tarafın da ortak kullandığı anahtarlar vardır. Compact

BasicCard'ta tüm anahtarlar 8 sekizli uzunluğundadır. **Triple DES** şifreleme için 16 sekizli uzunluğunda anahtarlar kullanılır.

Declare Key ***anahtar_no*** [(*uzunluk,* [*sayaç*])] [= *b1*, *b2*, *b3*, . . .]

anahtar_no: Anahtarın özelleştirilebilmesi için bir anahtar numarası gereklidir. (**StartEncryption** komutunda olduğu gibi) 0-255 arasında herhangi bir değer alabilir.

Uzunluk: Anahtarın uzunluğudur. Eğer belirtilmemişse anahtar uzunluğu sekiz byte varsayılmaktadır. Eğer bir başlangıç değişkeni alanı mevcutsa (*b1*, *b2*, *b3*, . . .) ve hiç bir uzunluk özelleştirilmemişse anahtar uzunluğu başlangıç değeri alanı içindeki byte sayısına eşitlenir. (Eğer uzunluk belirlenmişse başlangıç değeri alanı, istenen uzunlukta sıfırla doldurulur.)

Sayaç: Anahtar için hata sayacıdır. (0 <= sayaç <= 15). Eğer sayaç sıfırsa, anahtar başlangıçta kullanılamaz durumdadır. Eğer *sayaç* yoksa, anahtar hata sayacı aktif değildir.

b1, *b2*, *b3*,...Anahtarın başlangıç değeridir. Başlangıç değerinin verili olmadığı durumda varsayılan değer sıfırdır. Daha sonra anahtara yeni değer verilmesi için izlenebilecek üç yol vardır:

- **Key**(*anahtarno*) = ***string*** ifadesi ile, Terminal programında veya Enhanced BasicCard programında
- **Read Key File** ifadesi ile Terminal programında
- **BCKEYS** programı ile, BasicCard'ta.

8.1.2. Çalışma sırasında (run-time) anahtar konfigürasyonu

Terminal programı, bir anahtar dosyasından çalışma sırasında (run time) anahtarları yükleyebilir:

Read Key File *dosyaadı*

Eğer komut çalışmazsa Dosya Sistemi değişkeni **FileError** sıfırdan farklı bir kod (hatayı oluşturan nedeni işaret eden bir hata kodu) üretir. FILEIO.DEF hata kodlarının tanımlarını içerir. Terminal veya Enhanced BasicCard programlarında anahtarlara string olarak: **Key(***anahtarno***)** fonksiyonu yardımıyla erişilebilir.

8.1.3. Anahtar hata sayacı

Enhanced BasicCard'ta her kriptografik anahtarın bir hata sayacı vardır. Herhangi bir anahtar için hata sayacı aktif duruma geçtiğinde, Terminal programın anahtarı deneyerek bulma riskine karşı deneme sayısını sınırlar.

Şifrelemenin geçersiz olduğu durumda hata sayacı bir azaltılır ve Akıllı kart durum kodu **SW1-SW2 = swRetriesRemaining+*X*** (&H63C0+*X*) döndürür. Burada *X* hata sayacının yeni değeridir. Hata sayacı sıfır olduğunda anahtar geçersiz hale getirilir. Akıllı kart programında **Enable Key** komutu çalıştırılana kadar anahtar geçerli duruma geçmez. Şifreleme geçerliyse, hata sayacı başlangıç değerine getirilir.

Anahtar geçersiz durumda ise (hata sayacı daha başlangıçta sıfıra eşitse) akıllı kart durum kodunu **SW1-SW2 = swKeyDisabled** (&H6614) yaparak yanıt döndürür. Enhanced BasicCard programında , anahtarın hata sayacını etkin hale getirmek içi iki komut vardır:

Enable Key *anahtarno* **[(***sayaç***)]** : Anahtarı geçerli hale getirir. *Sayaç* varsa, anahtar hata sayacı aktif duruma getirilir ve başlangıç değeri **Max** (*sayaç*, 15) olarak etkinleştirilir. *Sayaç* yoksa veya 255'e eşitse anahtar hata sayacı yeniden aktif hale getirilir.

Disable Key *anahtarno* : **Enable Key** komutu işletilene kadar anahtarı geçersiz hale getirir.

Bu hata sayacı mekanizması yalnızca komutların şifrelenmesi için kullanılır. Bunun dışında bir anahtar geçersiz konuma getirilmiş olsa da Enhanced BasicCard programı içinden kullanılabilir. Kriptografik anahtarları kullanan ZC-Basic fonksiyonları **Şifreleme Fonksiyonları** bölümünde listelenmiştir.

8.1.4. DES şifreleme ilkelleri

DES mesaj şifreleme ve şifre çözme dört blok şifreleme ilkeline dayanır: $\mathbf{E_K}$, $\mathbf{D_K}$, $\mathbf{E^3_K}$, ve $\mathbf{D^3_K}$ Akıllı kart ve terminal programlarında bu ilkeller ZC-Basic programcısına **DES** fonksiyonu ile sağlanır.

sonuç$ = **DES**(*tip*, *anahtar*, *blok$*)

tip, ilkelin tipi +1, –1, +3, veya –3 ile numaralandırılır:

+1: $\mathbf{E_K}$(*blok*) **Tekli DES** şifreleme

–1: $\mathbf{D_K}$(*blok*) **Tekli DES** şifre çözme

+3: $\mathbf{E^3_K}$(*blok*) **Üçlü DES** şifreleme

–3: $\mathbf{D^3_K}$(*blok*) **Üçlü DES** şifre çözme

anahtar, 0-255 arasında bir sayı veya şifreleme anahtarını içeren bir karakter katarı olabilir. Tekli DES için anahtarın en az 8 sekizli uzunluğunda, Üçlü DES için ise en az 16 sekizli uzunluğunda olması gerekmektedir.

blok$, Şifrelenecek veya çözülecek bloğu içeren 8-sekizli uzunluğundaki karakter katarından oluşur. 8 sekizliden uzun olursa sadece ilk 8 sekizli kullanılır, az olursa, P-Code hatası **pcBadStringCall** (&H0D) üretilir.

sonuç$, DES şifreleme veya şifre çözme fonksiyonunun 8 sekizliden oluşan sonucudur.

8.1.5. Sertifika üretimi

Terminal programı ve Enhanced BasicCard'ta kriptografik anahtarlar kullanılarak "sayısal imza" üretilebilir. Burada sertifika üretimi olarak tanımlanan fonksiyon daha önceki bölümlerde sayısal imza olarak anlatılmıştı. Sertifika üretimi için:

S$ = **Certificate**(*anahtar*, *veri*)

anahtar 0-255 arasında bir sayıyı veya anahtarı içeren karakter katarını, *veri* ise doğrulanacak veriyi göstermek üzere yukarıdaki komut kullanılır. Burada veri,

karakter katarı tipinde bir ifade, sabit uzunluklu bir veri veya bir dizi elemanı olabilir. Anahtarın boyuna göre DES veya Üçlü DES sertifikası üretilir. Sonuç, *S$*, her durumda 8 sekizli uzunluğundadır.

8.1.6. Rastgele sayı üretimi

Rnd built-in fonksiyonu 4-sekizli boyunda bir rastgele sayı üretir. Terminal ve BasicCard'ın rastgele sayı üretimi için değişik mekanizmaları vardır.

8.1.6.1. Terminal

Terminal programı sistem saatini temel alarak bir rastgele sayı üretir. Program her çalıştığında **Rnd** fonksiyonunun üreteceği değer bu nedenle farklı olur. **Randomize** komutu ile bu üretim biçimi değiştirilebilir.

Randomize *kaynak*: Burada *kaynak* **Long** veya **String** tipinde herhangi bir ifadedir.

Rastgele sayı üretiminde kullanılan kaynağın değiştirilmek istenmesinin çeşitli nedenleri vardır:

1. Program geliştirirken izleme-hata ayıklama işleminin daha kolay yapılabilmesi için üretilecek rastgele sayı dizisinin önceden tahmin edilebilir olması istenebilir.

2. Güvenlik açısından sistem saatinden üretilecek sayının bilinebilirlik riski vardır.

Rastgele sayı üretecinin varsayılan davranışı akıllı kart haberleşmesinde kullanılan şifreleme algoritmaları açısından yeterlidir. Bu algoritmalar açısından **RA** ve **RB** rastgele sayılarının başlangıç değerleri kritik öneme sahip değildir. Güvenlik açısından kullanılacak anahtarların gizliliği önemlidir. Bu nedenle anahtar üretimi için yüksek kalitede bir rastgele sayı üreteci tercih edilmektedir.

8.1.6.2. Akıllı kart

Her BasicCard'ın içine gömülü bir seri numarası vardır. BasicCard üretiminden sonra ilk kez rastgele sayı üretiyorsa bu seri numarası kaynak olarak kullanılır. Daha sonra kaynak güncellenerek bir sonraki rastgele sayı üretimi için EEPROM'a kaydedilir. Bu sayede her BasicCard değişik sekansta rastgele sayı üretir. Verilen bir BasicCard her reset işleminden sonra aynı sekansta rastgele sayı üretmemiş olur.

Randomize komutu BasicCard'ın içinde bulunmaz, yüklenmesi gerekir. ZCDOS ve ZCDD programlarının içindeki BasicCard simülatörleri her çalıştırıldıklarında aynı dizilişte rastgele sayı üretirler. Bunun nedeni üretim mekanizmasını başlatacak kaynak olarak kullanmak için tekil bir seri numarasına erişimlerinin olmayışıdır. Fakat program bir akıllı karta yüklendiğinde rastgele sayı dizilişi önceden bilinemez hale gelir.

8.2. Eliptik Eğri kriptografi kütüphanesi: EC-161

EC-161 Kütüphanesi, terminal ve akıllı kart tarafı için kullanılacak 161-bit Eliptik Eğri Şifreleme kodlarını içerir. Kütüphane, aşağıdaki işlemleri destekler:

- kapalı/açık anahtar çifti üretimi
- paylaşılan gizli veri türetme
- oturum anahtarı üretimi
- sayısal imza üretimi
- sayısal imza doğrulama (Yalnızca terminal tarafında mümkündür)

Eliptik Eğri Kütüphanesi'nde kullanılacak alt programlar üç bölümde incelenebilir: Başlatma altprogramları, şifreleme düzenleri ve şifreleme ilkelleri[40].

8.2.1. Başlatma altprogramları

Sub EC161SetCurve (*Parametreler* **As EC161DomainParams**)

Elliptic Eğri için alan parametrelerini özelleştirmek amacıyla kullanılır. Bu alt programa yalnızca terminal programında gerek duyulur. Akıllı kart programlarında gerek duyulmaz.

Sub EC161GenerateKeyPair(*Kaynak*$)

Bu alt program, gizli/açık anahtar çiftini üreterek, sonuçları public String değişkenleri olan EC161PrivateKey ve EC161PublicKey değişkenlerine atar.

Sub EC161SetPrivateKey(*Key*$)

Özel bir gizli anahtar saptar ve ona bağlı açık anahtarı üreterek sonuçları public String değişkenleri olan **EC161PrivateKey** ve **EC161PublicKey** değişkenlerine atar.

8.2.2. Şifreleme düzenleri (Cryptographic Schemes)

ZC-Basic Card üzerinde Eliptik Eğri kütüphane fonksiyonları yardımıyla oturum anahtarı üretimi amacıyla ortak giz oluşturulması, imzalama ve imza onaylama düzenleri bulunmaktadır. Bu mekanizmaların işletilmesi amacıyla geliştirilmiş fonksiyonlar kısaca aşağıdaki gibidir:

Function EC161SessionKey(*KDP$, SharedSecret$*) **As String**

Anahtar türetme parametreleri KDP$(Key Derivation Parameters) ve verilen *SharedSecret$* değerlerini kullanarak 20-sekizli uzunluğunda bir oturum anahtarı üretir. Bu ECKAS-DH1 P1363 düzenidir: Her iki tarafın da bir anahtar çifti kullandığı, Elliptic Curve Key Agreement Scheme, Diffie-Helman versiyonudur.

Sub EC161HashAndSign(*İmza$, Mesaj$)* **As String**

Verilen mesajın SHA-1 çırpma değerini hesaplar ve EC161Sign ile imzalar. Bu yöntem, ECSSA P1363 düzeninin İmza Üretme İşlemidir: "Elliptic Curve Signature Scheme with Appendix."

Function EC161HashAndVerify (*İmza$, Mesaj$, AçıkAnahtar$*)

Verilen mesajın SHA-1 çırpma değerini hesaplar ve EC161Verify ile imzanın doğruluğunu test eder. Bu yöntem, ECSSA P1363 düzeninin İmza Doğrulama İşlemidir: "Elliptic Curve Signature Scheme with Appendix."

8.2.3. Şifreleme ilkelleri

Eliptik Eğri kriptografiye dayanan şifreleme düzenleri, yine Eliptik Eğri şifreleme ilkellerinden oluşmaktadır. Yukarıdaki bölümde tanıtılan düzenler aşağıdaki ilkelleri kullanarak oluşturulur.

Function EC161SharedSecret (AçıkAnahtar$) **As String**

Verilen AçıkAnahtar$'a bağlı olarak ona bağlı shared secret 'i üretir. Bu yöntem P1363 ilkeli ECSVDPDH'dır (Elliptic Curve Signature Primitive, Nyberg-Rueppel versiyonu).

Sub EC161Sign(*İmza$, Çırpma$*) **As String**

Verilen *Çırpma$* değeri için imzayı hesaplar. Bu yöntem P1363 ilkeli ECSP-NR'dir (Elliptic Curve Signature Primitiv, Nyberg-Rueppel versiyonu).

Function EC161Verify(*İmza$, Çırpma$, AçıkAnahtar$)*

Verilen *Çırpma$* değeri için imzanın doğruluğunu test eder. Bu yöntem P1363 ilkeli ECVP-NR'dir (Elliptic Curve Verification Primitive, Nyberg-Rueppel versiyonu).

Burada tanıtılan bireysel kütüphanelerin tanımlarında hata kodları da tanımlanabilir. Bu hata kodları ZC-Basic değişkeni olan **LibError**'a atılabilir. Bu değişken kütüphane alt programı tarafından işaretlenmiş en yakın zamandaki hata kodunu içerir.

8.3. EC-161: Eliptik Eğri kütüphanesi ile uygulama geliştirme

Bu bölümde altprogramlar daha ayrıntılı olarak incelenecek, uygulama geliştirilme açısından kütüphanenin kullanımı ele alınacaktır. Eliptic Eğri

kütüphanesini yüklemek için program koduna #Include EC-161.DEF, program satırı eklenir.

8.3.1. Eliptik Eğri parametrelerinin saptanması

Bir eliptik eğri, alan parametreleri ile tanımlanır. Kütüphanelerde üç uygun eliptik eğri bulunmaktadır: EC161-1.CRV..EC161-3.CRV, ZC-Basic dilinde eğri tanımlarını içerir. Bu tanımlardan biri kaynak programa dahil edilebilir. EC161.BIN dosyası üç eğri için de Terminal programına çalışma-zamanında yüklenecek ikili veriyi içerir.

Enhanced BasicCard programı içinde Eliptik Eğri belirlemek için program koduna EC161-X.DEF dosyasını eklemek gerekir. Bir akıllı kart programında eğrinin derleme sırasında seçilmesi gerekir çünkü çalışma sırasında (Run-time) yeniden karta yüklenemez. Terminal programı içinde Eliptik Eğri, EC161SetCurve kullanılarak yüklenir. Eliptik Eğri'yi yüklemenin üç yolu vardır:

- Hangi eğrinin kullanılacağı biliniyorsa tanım dosyası eklenir ve EC161SetCurve(*parametreler*) çağrılır. Bir programda yalnız bir tanım dosyasına yer verilebilir.

- Eğer kartın uygun komutu varsa, eğri karttan yüklenebilir. Örneğin:

```
Private Curve As EC161DomainParams
Call GetCurve (Curve) : Call CheckSW1SW2()
Call EC161SetCurve (Curve)
```

- Eğri, EC-161.BIN binary dosyasından okunabilir. Örneğin:

```
Private Curve As EC161DomainParams
Open "EC-161.BIN" For Random As  #1 Len=64
Get #1, 3, Curve ' Read Elliptic Curve #3
Close #1
Call CheckFileError ( )
Call EC161SetCurve (Curve)
```

Eğer alan parametreleri (DomainParams) geçersizse altprogram **EC161SetCurve**; **LibError** değişkeni içerisinde hata kodu **EC161BadCurveParams** döndürür.

Terminal programı içinde EC-161 kütüphanesi içinden diğer alt programlar çağrılmadan önce **EC161SetCurve** çağrılarak eğri tanımlanmalıdır. Aksi halde **LibError** değişkeni içerisinde **EC161CurveNotInitialised** hata kodu döndürülür.

8.3.2. Anahtar üretimi

Eliptik Eğri açık anahtar şifreleme için anahtarların üretilmesi işlemi iki biçiminde yapılabilir. Bu anahtarlar anahtar üretme fonksiyonları yardımıyla kart içinde üretilebileceği dışarıdan belirlenerek karta da verilebilir. Tez çalışmasında güvenlik açısından birinci yöntem tercih edilmiştir.

Açık/Gizli anahtar çiftini üretmek için:

Call EC161GenerateKeyPair(*kaynak$*),

altprogram çağrısı kullanılır. Bu altprogram güvenli çırpma algoritması kütüphanesi SHA-1'i kullanır. SHA-1, *kaynak$* değerini kullanarak geçici rastgele sayı üretir. Üretilen sayı gizli anahtar olarak kullanılacak sayıdır. 21-sekizli uzunluğundaki gizli anahtar ve ona bağlı 22 sekizli uzunluğundaki açık anahtar Eeprom string değişkenleri **EC161PrivateKey** ve **EC161PublicKey** içinde saklanır.

Gizli Anahtar değerini dışarıdan belirlemek için:

Call EC161SetPrivateKey(*anahtar$*),

alt program çağrısı kullanılır. Bu alt program (mod r'de hesaplanan) *anahtar$* 'ı 21-sekizli uzunluğundaki bir EEPROM katarı olan **EC161PrivateKey**'e kopyalar ve ilgili açık anahtar değerini hesaplayarak 22-sekizli uzunluğundaki EEPROM katarı **EC161PublicKey** değişkenine kaydeder. (Burada r, seçilen Eliptik Eğri parametrelerinden G noktasının derecesidir.)

Eğer *anahtar$* mod r değeri sıfır bulunursa, **EC161BadProcParams** hata kodu **LibError** değişkeni ile döndürülür. Burada eğer ilgili açık anahtar değeri hesaplanmak istenmiyorsa modül kontrolüne de gerek yoktur. Gizli anahtar değeri doğrudan değişkene kopyalanabilir. Modül işleminin aldığı süre kısalmış olur.

8.3.3. Sayısal imza üretimi

Sayısal imza üretiminde gizli-anahtar kullanılır. Karakter katarı tipinde bir mesajın imzalanması için:

Call EC161HashAndSign (*imza$, mesaj$*),

alt-program çağrısı kullanılır. Bu alt-program *imza$* parametresi içinde 42-sekizli uzunluğunda bir katar döndürür.

Fakat bu algoritmada aslında isminde olduğu gibi çırpma değeri hesaplanmaz. 42-sekizliden uzun mesajların imzalanmasında öncelikle mesajın çırpma fonksiyonu hesaplanarak mesajın kısaltılması gerekir. (Güvenli Çırpma Algoritması kütüphanesi SHA-1 kullanılır.) Daha sonra hesaplanan çırpma değeri imzalanır. Yani çırpma değeri hesaplama ve imzalama şeklinde işlem iki basamakta yürütülür. Çırpma değerinin imzalanması için:

Call EC161Sign(*imza$, çırpma$*),

altprogram çağrısı kullanılır. Burada oluşturulan imza da 42-sekizli uzunluğundadır. Eğer burada asimetrik şifreleme yapılırken kullanılacak gizli anahtar daha önceden belirlenmediyse, bu imzalama fonksiyonları, **LibError** değişkeni içinde **EC161KeyNotInitialised** hata kodunu döndürür.

Enhanced Basic Card versiyonları ZC2.3 ve ZC2.4 için sayısal imzanın oluşturulması 2.5 saniye veya 3.57MHz saat zamanı tutar. EC-FSA kartlarda bu süre 1.2 saniyedir.

8.3.4. Bir sayısal imzanın onaylanması

Bunun için onaylayacak tarafta gönderici tarafın açık anahtarının tutulması gerekir. Bir sayısal imzanın onaylanması için aşağıdaki program çağrısına başvurulur:

Status = **EC161HashAndVerify** (*imza$, mesaj$, AçıkAnahtar$*)

imza$, 42 sekizli uzunluğundaki onaylanacak imzadır.

message$, imzalanmış verinin kendisidir.

AçıkAnahtar$, imzalayan tarafın 22-sekizli uzunluğundaki açık-anahtarıdır.

Bu fonksiyonun dönüş değeri imzanın geçerli olup olmadığını bildirmek üzere **True** veya **False** olur. Onaylanacak mesaj 42-sekizliden uzunsa, önce mesajın çırpma değeri hesaplanır, (Güvenli Çırpma Algoritması Kütüphanesinde bulunan SHA-1 kullanılır.) ardından imzayı doğrulamak üzere aşağıdaki program çağrısı kullanılır:

Status = **EC161Verify** (*imza$, çırpma$, AçıkAnahtar$*)

Burada eğer imza 42-sekizli uzunluğunda değilse veya AçıkAnahtar$ 22-sekizli uzunluğunda değilse hata mesajı **LibError** değişkeninde **EC161BadProcParams** hata kodu döndürülür.

8.3.5. Oturum-anahtarı üretimi

İki taraf da birbirlerinin açık anahtarlarının bilgisine sahipse karşılıklı güvenli bir kanaldan konuşmalarını sağlamak için bir oturum anahtarı üzerinde anlaşmaları mümkündür. Bu anlaşma için birbirlerine herhangi bir veri aktarımı yapmaları gerekmez (Bölüm 5.6.1'e bakınız). Burada kullanılan fonksiyonda üzerinde anlaşma sağlanan gizli değer 21-sekizli uzunluğundadır. Bu değere **ortak giz** (shared secret) denir. **Ortak giz**, karşı tarafın açık anahtarı ve hesaplayan tarafın gizli anahtarı kullanılarak hesaplanır. Bunun için kullanılan kütüphane fonksiyonu çağrısı aşağıdaki gibidir.

OrtakGiz$* = **EC161SharedSecret**(*AçıkAnahtar$)

AçıkAnahtar$, karşı tarafın 22-sekizli uzunluğundaki açık anahtarıdır.

OrtakGiz$, 21-sekizli uzunluğundaki **ortak giz**'dir.

Eğer *AçıkAnahtar$* 22-sekizli uzunluğunda değilse veya Eliptik Eğri Algoritmasında kullanılan eğri üzerinde değilse **LibError** değişkeninde hata kodu **EC161BadProcParams** döndürülür.

Ortak giz daha sonra 20-sekizli uzunluğunda **oturum-anahtarlarının** üretiminde kullanılır. Ortak giz, karşılıklı açık ve gizli anahtar çiftleri değişmediği sürece değişmez ama oturum anahtarları, ortak giz değişmese de her üretimde değişir ve tek bir oturum için güvenli haberleşmede kullanılır.

Oturum anahtarı üretebilmek için taraflar Anahtar Türetme Parametreleri (KDP-Key Derivation Parameters) üzerinde anlaşırlar. Anahtar Türetme Parametreleri herhangi bir sekizli dizisi olabilir ve gizli tutulması gerekmez. Yüksek güvenlik açısından her oturum anahtarı üretiminde bu parametrenin de değiştirilmesinde yarar vardır. Örneğin standart bir başlığa eklenmiş tarih ve zaman verisi her defasında yeniden üretilebilir veya üretimden önce kullanıcıdan rastgele tuşlara basması istenebilir. Burada ikinci yöntem kullanılmıştır. Bu sayede sürecin sonunda üretilecek oturum anahtarının daha önceki bir anahtarla aynı olma tehlikesi ortadan kaldırılmıştır.

Oturum anahtarı üretmek üzere kullanılan alt program çağrısı aşağıdaki gibidir:

OturumAnahtarı$ = **EC161SessionKey** (*KDP$, OrtakGiz$*)

KDP$, Anahtar Türetme Parametresi, herhangi uzunlukta veri topluluğudur.

OrtakGiz$, **EC161SharedSecret** alt programı tarafından üretilen ortak giz değeridir.

Akıllı kartta ortak gizin üretilmesi, 3.57 MHz saat-zamanında yaklaşık yedi saniye sürer. Fakat verilen bir açık anahtardan bir kez karşılıklı haberleşme için bu değer üretildikten sonra, oturum anahtarının üretimi aynı saat-zamanında yarım saniyeden az sürmektedir. Buradaki uygulamada olduğu gibi genel olarak akıllı kart uygulamalarında da ortak gizin üretimi kart ömrü boyunca bir kez olmaktadır.

9. BİR GÜVENLİKLİ SAĞLIK SİSTEMİ

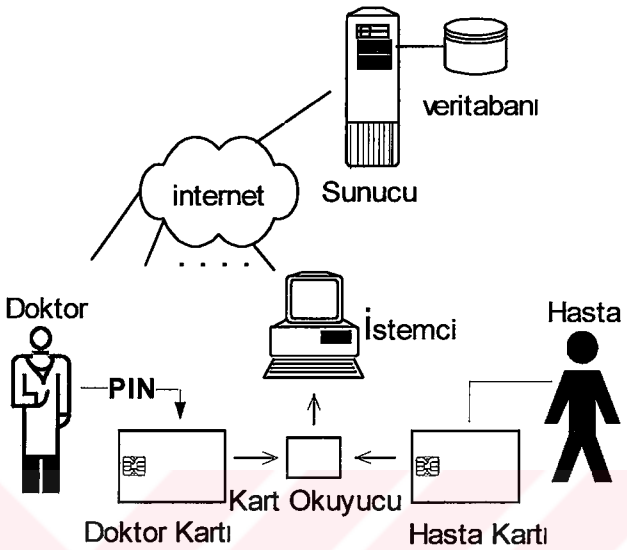
Yapılan çalışmada ele alınan sağlık sistemi, kişilere ait sağlık verilerinin kişilerin taşıyabileceği bir ortamda ve bilgi merkezlerinde saklanmasını, erişilmesini ve kayıt edilmesini gerçekleyen bir bilgisayar sistemidir. Sistemi kullanacak olan kişilerin sisteme güven duyabileceği bir ortam sağlanacaktır. Kişisel verilerin korunması ve izinsiz ve amacı dışında kullanılmamasının sağlanması amaçlanmaktadır. Ancak bu amacın, kullanıcı ve/veya yetkili hataları gibi bazı olağan dışı konularda sistemin işletilmesi ve denetlenebilen bir sistem yaratılması gerekir. Bunun için uygulamaya yönelik en elverişli çalışma biçimi oluşturulmalıdır.

Çalışmanın kapsamında; sağlık verilerine erişimi kolaylaştırma ve hızlandırma, hizmet kalitesini arttırma, hasta bilgilerini güvenli bir ortamda saklama, sağlık verilerine değişik hastane, klinik vb. sağlık merkezlerinden erişimin ve bu verilerin hastaya yapılan işlemlerden sonra güncellemesini sağlama, hasta ile ilgili verilerin yönetiminin tek bir merkezden yapılarak tutarlılığı sağlama, hastanın geçmiş tedavi ve teşhis bilgilerini merkezi bir ortamda tutma, kişisel verilerin güvenliğini sağlama yer almaktadır.

Sağlık sisteminde akıllı kart kullanımının temel amacı, hastanın acil durumda gerekli olabilecek verilerinin üzerinde taşıyabileceği güvenli bir ortamda saklanması ve sağlık sisteminde hasta verileri üzerinde işlem yapabilmek için hastanın ve doktorun varlıklarının asıllanmasıdır.

Sistemin kullanıcıları olan hasta ve doktorun akıllı kartları aracılığıyla sistemde işlem yapma yetkilerinin belirlenmesi güvenliğin bir parçasıdır. Akıllı kart kullanımı, kullanıcıların sisteme kendilerini tanıtmalarını sağladığı gibi, hastanın doktora, onun kimliğini kontrol ederek bazı kişisel verilerine erişime izin vermesi gibi olanaklar sunmaktadır.

Sistemde saklanan ve aktarılan tüm kişisel veriler şifrelenmektedir. Veri şifreleme, akıllı kart okuyucusundan bilgi merkezine erişme olarak değişik aşamalarda kullanılmaktadır.



Şekil 9.1. Sistemin temel bileşenleri

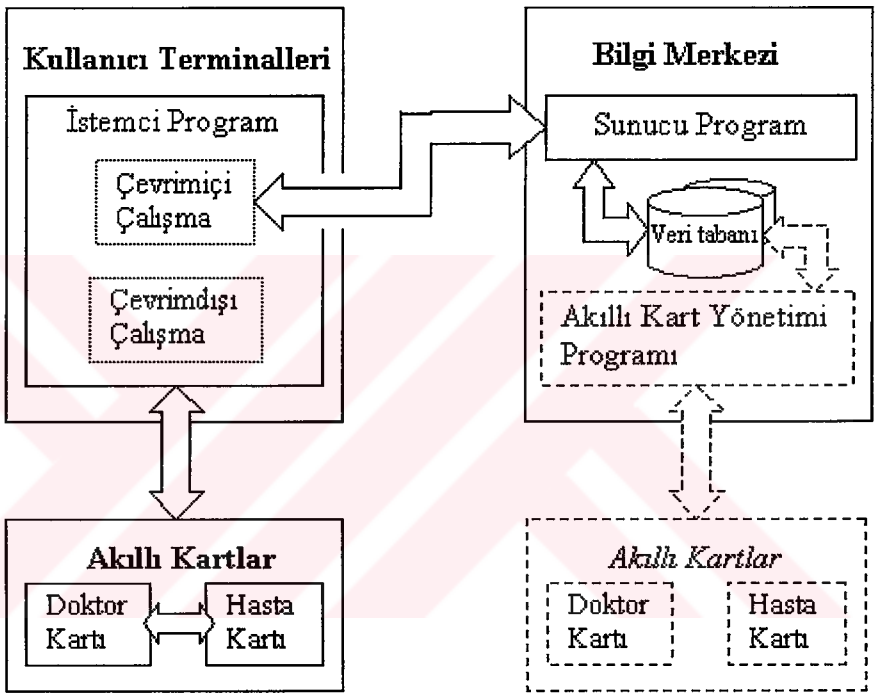
Güvenlik protokolünün tasarlanacağı pilot sistemin ana şeması Şekil 9.1'de görülmektedir. Pilot çalışmada üzerinde güvenlik protokolü tasarımının yapılacağı sağlık sistemi, şekilde de görüldüğü üzere doktor, doktorun kullandığı terminal veya istemci bilgisayar, bilgi merkezi olarak kullanılan ve terminallerin uzaktan erişebildikleri sunucu bilgisayar, hasta ve doktor ile hastanın akıllı kartlarından oluşmaktadır. Yapılan çalışma istemci ve sunucunun fiziksel konumlarına bağımlı değildir.

Uygulamanın temel özellikleri ve tasarım karakteristiklerinin anlaşılabilmesi amacıyla; sistem işlevsel olarak bütünleşik bir sistem olsa da mimari olarak Şekil 9.2'de görüldüğü gibi üç alt sisteme ayrılarak incelenebilir:

1. Bilgi Merkezi (sunucu bilgisayar): Hasta ve sağlık personeline ilişkin kişisel verilerin organizasyonu için veri tabanı ve güvenlik protokolünde kullanılmak üzere kullanıcı sertifika bilgilerinin bulunduğu merkezi sunucudur. Kullanıcı akıllı kartlarının yönetimi; kart basımı, kart silme, kilitlenen kartları çözme gibi işlemler için de özel bir yazılım bulunmaktadır.

2. İnteraktif kullanıcı terminalleri (istemci bilgisayarlar): Hasta kartı üzerindeki bilgilere çevrimiçi ve çevrimdışı olarak iki türlü erişimin gerekli güvenlik kontrolleri yapılarak sağlandığı terminallerdir.

3. Akıllı kartlar (Hasta kartı ve doktor kartı): Akıllı kartlarda kullanıcıların kişisel bilgileri dışında kriptografik anahtarlar da tutulur. Şifreleme işlemleri için doktor ve hasta kartında kullanılan algoritmalar da güvenlik mekanizmasında üstlendikleri roller gözetilerek kartlara yüklenir.



Şekil 9.2. Sistem Mimarisi

9.1. Bilgi merkezi

Bilgi merkezinde hasta ve sağlık personeline ait verilerin organizasyonunun yapıldığı bir merkezi veri tabanı bulunmaktadır. Veri tabanında tutulan veriler temel olarak aşağıdaki biçimde sıralanabilir:

1. Hastaların kimlik bilgileri

2. Hastaların geçmiş tedavi bilgileri

3. Hastaların akıllı kartlarında da aynı biçimde tutulan acil durum sağlık bilgileri (alerji bilgileri, önemli hastalıklar, kullanılan ilaçlar, arızalar)

4. Doktorların kişisel bilgileri

Bilgi merkezinde, sunucu programı ve akıllı kart yönetimi programı olmak üzere iki farklı işlem yürütülür. Sunucu programı, terminallerden gelen isteklere yanıt üreterek veri tabanına erişimi denetler. Akıllı kart yönetimi programı ise offline çalışır. Bilgi merkezi üzerinden yürütülen akıllı kartların kişiselleştirilmesi, güncellenmesi gibi işlemleri yürütür. Bu iki program da birbirinden bağımsız süreçler olarak işletilir. Sunucu programı, istemci programı ve terminale bağlı olan doktor ve hasta akıllı kartları ile iletişim içindedir. Güvenlik protokolü bu iletişimin güvenliği üzerine kuruludur. Akıllı kart yönetimi programı ise tamamen ayrı işleyen bir süreçtir.

9.1.1. Sunucu programı

Sunucu programının görevleri kendisine gelen istek paketlerine bağlı olarak aşağıdaki şekilde sıralanabilir:

1. Sunucu ile istemci arasında gerçekleştirilen tüm işlemler güvenli kanal üzerinden yapılır. Gelen istek üzerine oturumu başlatmadan önce öncelikle istemci ile sunucu programı arasında bir güvenli kanal oluşturulur, güvenli kanal sayesinde asıllama verisinin bütünlüğü de korunmuş olur, sunucu asıllama paketini açar ve kullanıcı kimlik bilgilerini alarak asıllama işlemini gerçekleştirir. Burada doktor ve hastanın aynı anda istemci programı tarafında var olmaları koşulunun sağlanıp sağlanmadığının kontrolü yapılır.

2. Asıllama işleminin başarılı olmaması durumunda karşı tarafa asıllamanın başarısız olduğunu belirten bir mesaj yollanır ve oturum açılmaz.

3. Asıllama işlemi başarılı olmuşsa sunucu program istek paketini değerlendirir.

4. İlgili hastanın bilgilerini veri tabanından çekerek oluşturulan güvenli kanal üzerinden istemciye gönderir.

5. Doktorun hasta bilgileri üzerinde yaptığı güncelleme işlemini değerlendirerek bu veriler arasında akıllı kartta da tutulması gereken veri olup olmadığını belirler. Güncelleme işleminin bilgi merkezinde başarıyla gerçekleştirilmesinin ardından akıllı kartta da güncellemenin yapılmasına yönelik işlemleri başlatır. Bu sıralama, akıllı kart ve bilgi merkezindeki değişikliklerin birbirini tutması, veri tutarlılığının bozulmaması açısından tercih edilmiştir. Bu işlemlerin doğru kişiler tarafından güvenli bir biçimde yerine getirilebilmesi için sunucu programında da güvenlik mekanizmaları işletilmekte, gerekli kontroller yapılmaktadır. Sunucu programı güvenlik protokolünde aktif taraflardan biridir.

9.1.2. Akıllı kart programı

Bilgi merkezi, aşağıdaki işlemleri gerçekleştirmek üzere özel akıllı kart programı ile desteklenmiştir:

1. Akıllı kartların yayınlanması: Bu işlem sonucunda, boş bir ZC-Basic akıllı kartı bir doktor veya hastaya ait bir akıllı kart olarak kullanıma hazır hale getirilir. Bunun için hastaya veya doktora ait kişisel bilgiler merkezi veri tabanından çekilerek akıllı kartta EEPROM'a yazılır. Diğer önemli işlemse kriptografik anahtarların akıllı kart üzerinde üretiminin ve saklanmasının sağlanmasıdır. Anahtarların kart dışında değil kart üzerinde üretimi güvenlik protokolünde akıllı kartın sağladığı en önemli güvenlik özelliğidir.

2. Akıllı kartın güncellenmesi işlemi: Kullanıcıların merkezi veri tabanında tutulan kişisel bilgileri değiştiğinde bu bilgilerin silinmesi, değiştirilmesi gibi işlemler için akıllı kartı ile birlikte bilgi merkezine başvurur ve bu gibi değişiklikler merkezi olarak yapılır. Özellikle kimlik bilgisi üzerinde güncelleme yalnız merkezi olarak yapılabilir.

3. Akıllı karttaki bilgilerin silinmesinin sağlanması

4. Akılı kart kilidinin açılması: Kart kullanıcısı üç kez yanlış PIN kodu girdiğinde kartın gerçek sahibi dışında bir başkası tarafından kullanımını engellemek üzere kart kendini kilitler. Kilitlenmiş bir akıllı kartı açtırabilmek için kart kullanıcısının kimliği ile birlikte bilgi merkezine başvurması

gerekmektedir. PIN girişi doktor kartında gerek duyulurken hasta kartında, hastanın acil durum verilerine erişimin sağlanabilmesi açısından kullanılmamaktadır.

5. Merkezi olarak akıllı kart yönetimi programındaki işlemleri gerçekleyecek kullanıcının yetkilerinin sınanması, kimlik denetiminin sistem tarafından gerçekleştirilmesi gerekir. Burada gereksinim duyulan varlık asıllama işlemidir. Kullanıcıya ise "sistem yöneticisi" adı verilir. Sistem yöneticisinin asıllanması için PIN numarası ve kullanıcı adı girmesi beklenir.

9.2.Kullanıcı terminalleri

Sağlık sistemlerinde sağlık personelinin kullanabileceği çeşitli terminaller ve üzerinde işlem yapılan değişik veri çeşitleri bulunmakla birlikte konuya yalnızca güvenlik başlığı üzerinden yaklaşılacağından terminaller üzerinden akan verilerin çeşitliliği ve organizasyonu gözetilmeksizin problem olabildiğince basitleştirilerek ele alınmıştır.

Terminal veya istemci bilgisayarlar, çevrimdışı veya çevrimiçi çalışabilir. İstemci bilgisayarı üzerinde her iki durum için de bir istemci programı koşmaktadır. Çevrimdışı çalışma acil durumda internet bağlantısının olmadığı varsayılarak hasta kartı üzerindeki verilerin doktor kartı yardımıyla okunmasını sağlar. Hastanın verilerine erişim her durumda bir doktorun varlığını gerektirir. Burada asıllama işlemi doktorun akıllı kartı yardımıyla yerine getirilir.

Çevrimiçi çalışma durumunda ise internet üzerinden uzaktaki hasta verilerine erişim sağlanarak doktorun yetkileri dahilinde bilgi edinmesine, merkezi veri tabanındaki bilgileri güncellemesine, veri tutarlılığı açısından gerektiğinde doktorun hasta kartı üzerindeki bilgileri de güncellemesine izin verilir.

İstemci programı aşağıdaki işlemleri yapar:

1. Doktor kartı ve hasta kartının takılı olduğu kart okuyucu ile haberleşerek asıllama işlemlerini gerçekleştirir. Önce PIN sorgulama ile doktor kartı okunarak doktor istemci program oturumu başlatılır. Ardından hasta kartı okunur. Buraya kadar olan kısım çevrimdışı çalışma kısmıdır.

2. Doktor hastaya ilişkin geçmiş tedavi bilgilerini de görmek istediğinde veya veri tabanında güncelleme yapmak istediğinde istemci programı internet üzerinden sunucu ile bağlantı kurar. Buradan sonra istemci programının işlemlere devam edebilmesi için sunucu tarafından doktor ve hastanın birlikte asıllanması gerekir. İstemci programı kart okuyucu ile bağlantı kurarak doktor ve hastanın asıllama bilgilerini edinir ve sunucuya güvenli bir kanal üzerinden gönderir.

3. Sunucu programından hasta ve doktorun akıllı kartları asıllanarak onay alındığında sunucudan gelen hastanın geçmiş tedavi bilgileri ekranda görüntülenir.

4. Doktorun muayene sonunda oluşan hasta bilgilerini sunucuya göndermesini sağlar. Bu sırada akıllı karta da eklenmesi gereken bilgiler varsa doktor bu bilgileri sunucuya gönderilecek formun üzerinde işaretler. İstemci program, terminale bağlı olan hasta kartının güncellemesi işlemini başlatmış olur. Bu işlem sırasında istemci program sadece bir aracı işlevi görmektedir. Akıllı kartın güvenli bir biçimde doktor tarafından güncellenmesi işlemi ileride güvenlik protokolünün tasarımı ile ilgili bölümde anlatılacaktır.

9.3. Akıllı kartlar

Doktor ve hastaya ilişkin "varlık asıllama" ve "veri kaynağı asıllama" işlemlerini gerçekleştirmek, güvenlik protokolünde doktora ve hastaya ilişkin kriptografik anahtar ve sertifikaları güvenli bir biçimde saklamak amacıyla akıllı kart kullanılır. Sağlık sistemlerinde akıllı kart kullanımının yararları üzerine önceki bölümlerde de gerekli açıklamalar yapılmıştır.

9.3.1.Doktor kartı

Sağlık sisteminde doktorların kendilerini sisteme tanıtmaları ve hasta bilgilerine erişim için kullandıkları akıllı karta doktor kartı denilir. İlk olarak akıllı kart ile doktor arasında bir asıllama işlemi gerçekleştirilir. Bunun için doktorun kişisel kimlik kodunu (PIN) doğru bir biçimde akıllı karta bildirmesi gerekir.

Doktor kartında tutulan veriler aşağıdaki biçimde sıralanabilir:

1- Doktorun kimlik bilgisi

2- Doktora ilişkin kriptografik anahtarlar

Doktor kartından bilgi okumak için bir terminal programının varlığı ve PIN kodunun doğru girilmesi yeterlidir. Doktor kartındaki verilerin güncellenmesi ancak sistem yöneticisi tarafından bilgi merkezinde yapılabilir.

9.3.2. Hasta kartı

Hastanın kullandığı akıllı karta hasta kartı denir. Hasta kartında tutulan veriler aşağıdaki biçimde sıralanabilir:

1. Hastanın önemli sağlık bilgileri: Hastalık, alerji, kullandığı önemli ilaçlar, eksik veya sakatlanmış organ olup olmadığına ilişkin bilgiler
2. Hastanın kimlik bilgileri
3. Acil durumda aranabilecek yakınlarının telefonları
4. Hastanın kendisine ilişkin kriptografik anahtarlar
5. Sunucuya ve hastanın "kişisel doktor"larına ilişkin sertifika bilgileri

Hasta kartı üzerine bilgi yazma iki şekilde yapılabilir:

1. Sistem yöneticisi tarafından bilgi merkezinde,
2. Hastanın doktoru tarafından sağlık kontrolü sonrasında verilerin güncellenmesi gerektiğinde.

Birincisi hastanın tüm bilgilerini güncelleme yetkisine sahipken hastanın doktoru yalnızca hastanın sağlık verilerini güncelleme yetkisine sahiptir.

Doktorun kartı güncellemesi işlemi bilgi merkezi ve hasta kartı arasında veri tutarsızlığı problemi yaratacaktır. Bu problemin çözümlenmesi için öncelikle hasta bilgilerinin bilgi merkezinde internet üzerinden güncellenmesi, bu işlemin başarı ile gerçekleştirildiğinden emin olunduktan sonra hasta kartında güncelleme

yapılması uygun bulunmuştur. Bu işlem sırasında yine tutarlılığın sağlanması açısından her iki tarafa da yapılacak güncellemenin doktor açısından tek bir vuruşta yapılması gerekmektedir. İki ayrı güncelleme işlemi olarak doktorun kendisine bırakmak yerine doktorun bir kez "güncelle" butonuna basmasının ardından sistem tarafından işlemlerin yürütülmesi gerekir.

Bu noktada sistemin hangi elemanına ne kadar güvenileceği ve ne kadar yetki verileceği problemi ortaya çıkmaktadır. İstemci programa aktarma görevi dışında herhangi bir yetki verilemez. Doktor, yazılacak verilere karar verme ve işlemi başlatma yetkisine sahiptir. Burada yetkili taraf sunucudur çünkü buraya kadar doktor ve hastanın ikisini birden asıllamış, onaylarını almış durumdadır. Fakat sunucunun uzak mesafeden bu yetkiyi nasıl kullanacağı ve kime devredeceği problemi güvenlik probleminin tasarımı sırasında çözülecek bir problem olarak ortaya çıkmaktadır.

Karşılaşılabilecek bir başka olası durum bilgi merkezinde güncelleme yapılmasının ardından terminalde meydana gelecek bir fiziksel aksaklık nedeniyle akıllı kartta güncellemenin başarıma ulaşamamasıdır. Bu durumda oluşacak tutarsızlık hastanın bilgi merkezine başvurmasıyla veya kart üzerinde herhangi bir yerde yapılacak bir sonraki güncelleme işlemiyle çözülmektedir.

Hasta kartında tutulan veriler erişim izni bakımından iki kategoriye ayrılır:

1. Her doktorun görebileceği veriler: Bu verilere erişim için sistemde bir doktorun ve hasta kartının varlığı yeterlidir. Terminal programı yardımıyla verilere erişim mümkündür. Bilgi merkezinde de yukarıda anlatıldığı biçimde erişim mümkündür. Bu verilerden kimlik bilgilerinin güncellenmesi yalnızca bilgi merkezinde yapılabilirken sağlık bilgileri doktor tarafından da güncellenebilir.

2. "Kişisel doktorlar"ın görebileceği veriler: Hasta kendi seçimine bağlı olarak bazı hastalık kayıtlarının yalnızca o hastalığa ilişkin tedaviyi uygulayan veya uygulamış olan doktorlar tarafından görülmesini isteyebilir. Bu hastalıklar, hastanın akıllı kartına işlenmesi gerekli "önemli hastalıklar" kategorisinde olabilir. Bu durumda hasta kartı, kendisine veri okuma isteği geldiği zaman bu isteğin kimden, hangi doktordan geldiğini kontrol ederek bilgileri ekranda

görüntülemelidir. Hasta kartının bu denetimi yapabilmesi için hasta kartında "kişisel doktorlar"a ilişkin sertifika bilgilerinin, açık anahtar bilgilerinin, bulunması gereklidir. "Hassas hastalık" bilgilerine erişim denetimi yapılırken terminal programına güven duyulmamaktadır. Terminal programı burada yalnızca aracı görevini görebilir fakat bilgileri gösterme yetkisi onaylanıp, gösterme aşamasına gelinene kadar terminal programı herhangi bir yetkiye sahip değildir.

9.4. Uygulamanın önemli özellikleri

Sağlık sisteminin diğer akıllı kart kullanılan güvenlikli sağlık sistemleri ile karşılaştırıldığında taşıdığı özellikler aşağıdaki şekilde sıralanabilir:

1. Akıllı kart okuyucunun bağlı olduğu terminale güven duyulmaksızın iki akıllı kartın üzerinde yürütülen kriptografik işlemlerle birinin diğerini asıllaması sağlanmaktadır.

2. Sunucu, merkezi veri tabanındaki bilgilere erişimi denetlemek için istemci üzerinden ikili kimlik doğrulama yapmaktadır. İki varlığın da aynı anda asıllanması anlamına gelen bu işlem için doktorun ve hastanın akıllı kartları kullanılmaktadır.

3. İstemci makinaya bağlı akıllı kartın güncellenme işlemi uzaktan erişim yoluyla sunucu tarafından gerçekleştirilmektedir. Burada asıllama ve güvenlik problemlerinin çözümü sağlanmıştır.

9.5. Güvenlik servisleri

Sağlık sisteminde yukarıda tanıtılan sistem bileşenleri arasında bir güvenlik protokolü tasarlanarak gerçeklenmiştir. Güvenlik protokolü aşağıda tanıtılan güvenlik servislerini sağlamaktadır.

1.Kimlik tanıma: Doktor ve hasta akıllı kartları aynı zamanda kimlik olarak kullanılırlar. Kartın üzerinde kişisel kimlik bilgileri de tutulur. Bu sayede doktorun ve hastanın kendilerini sisteme tanıtmaları mümkün olur.

2. Kart sahibinin doğrulanması: Sistem, akıllı kartın kart sahibi adına sistemde tüm asıllama işlemlerinde kullanılması üzerine kuruludur. Akıllı kart kart sahibinin sistemdeki varlığının kanıtıdır. Fakat bu durum şöyle bir tehlike yaratmaktadır: Kartın bir başkası tarafından kullanılması. Buna karşı kart sahibinin kartı kullanabilmesi için bir koşulu yerine getirmesi, karta kendisini onaylatması gereklidir. Kart ve kart sahibi arasında gerçekleştirilen bu asıllama işlemi, kart sahibinin kişisel şifresini girmesi ile başlar. Girilen şifre, kullanıcı kartında saklanan kullanıcı şifresi ile karşılaştırılarak kartın kart kullanıcısını asıllaması işlemi gerçekleştirilir. Burada kullanılan şifreye genel olarak verilen adlandırma kişisel tanıtım numarası anlamında PIN'dir. Bu işlem hasta verilerinin acil durumda doktor tarafından kullanılacağı gözetilerek hasta ve hasta kartı arasında yapılmaz. Buna karşılık doktorun PIN numarası girerek kendini doktor kartına asıllaması gerekir. Yerel sistemde gerçekleşen bu asıllama işleminden sonra akıllı kart, sahibini sistemde tanıtma ve onun adına asıllama işlemlerini yürütme yetkisine sahip olur.

3.Asıllama: Sistemde doktor kartı, hasta kartı, istemci ve sunucu arasında sağlıklı haberleşmenin yürüyebilmesi için tarafların karşılarındakinin varlığından, gelen mesajın doğru taraftan geldiğinden emin olmaları gerekir. Bunun için taraflar arasında çok yönlü, ikili ve üçlü, çeşitli ilkellerin kullanıldığı asıllama işlemleri gerçekleşir. Tasarlanan güvenlik protokolü bu asıllama işlemleri üzerine kurulmuştur.

4. Akıllı kart ile okuyucu yazılımı arasındaki asıllama: Kullanıcıların asıllamasından önce akıllı kartın EEPROM'unda tutulan verilere erişimin denetlenmesi amacıyla öncelikle kart üzerinde işlem (okuma, güncelleme) yapacak olan programla akıllı kart arasında bir yetki sınama işlemi yapılır. Akıllı kart, okuyucu programın tipine göre o programın kartın üzerinde hangi işlemleri yapmaya izinli olduğunu öğrenir.

Uygulamada iki tip kart okuma programı vardır. Bunlardan birincisi akıllı kart yönetimi programı, ikincisi ise istemci programdır. Kart yönetimi programının akıllı kart üzerinde her tür işlem yapma yeteneği mevcutken, istemci programın yetenekleri sınırlandırılmıştır. Bir okuyucu programın akıllı kart tarafından

asıllanması anlamına gelen bu işlem kullanıcı asıllaması yapılmadığı durumda elbette ki bu programların yalnız başına işlem yapabilmesi anlamına gelmez.

Bunun için akıllı kart üzerinde tutulan dosyaların her biri için erişim hakları belirlenerek anahtarlama yapılır. Her okuyucu programda bir anahtar tutulmaktadır. Okuyucu programın kartla ilişki kurabilmesi için anahtar bilgisinin karşılıklı birbiriyle uyumlu olması gerekir. Aynı anahtar akıllı kart ve terminal arasında gidip gelen verilerin şifrelenmesi için de kullanılmaktadır. Burada aynı mekanizma hem gizlilik hem de asıllama amacıyla kullanılmış olur.

5. Hareketin asıllanması: Hareketin asıllanması için oturum anahtarı kullanılır. Bunun anlamı yeni hareketler için bir oturum anahtarının üretilmesini gerektirmesidir. Tarafların her birisi, oturum anahtarını, ortak gizi kullanarak üretir.

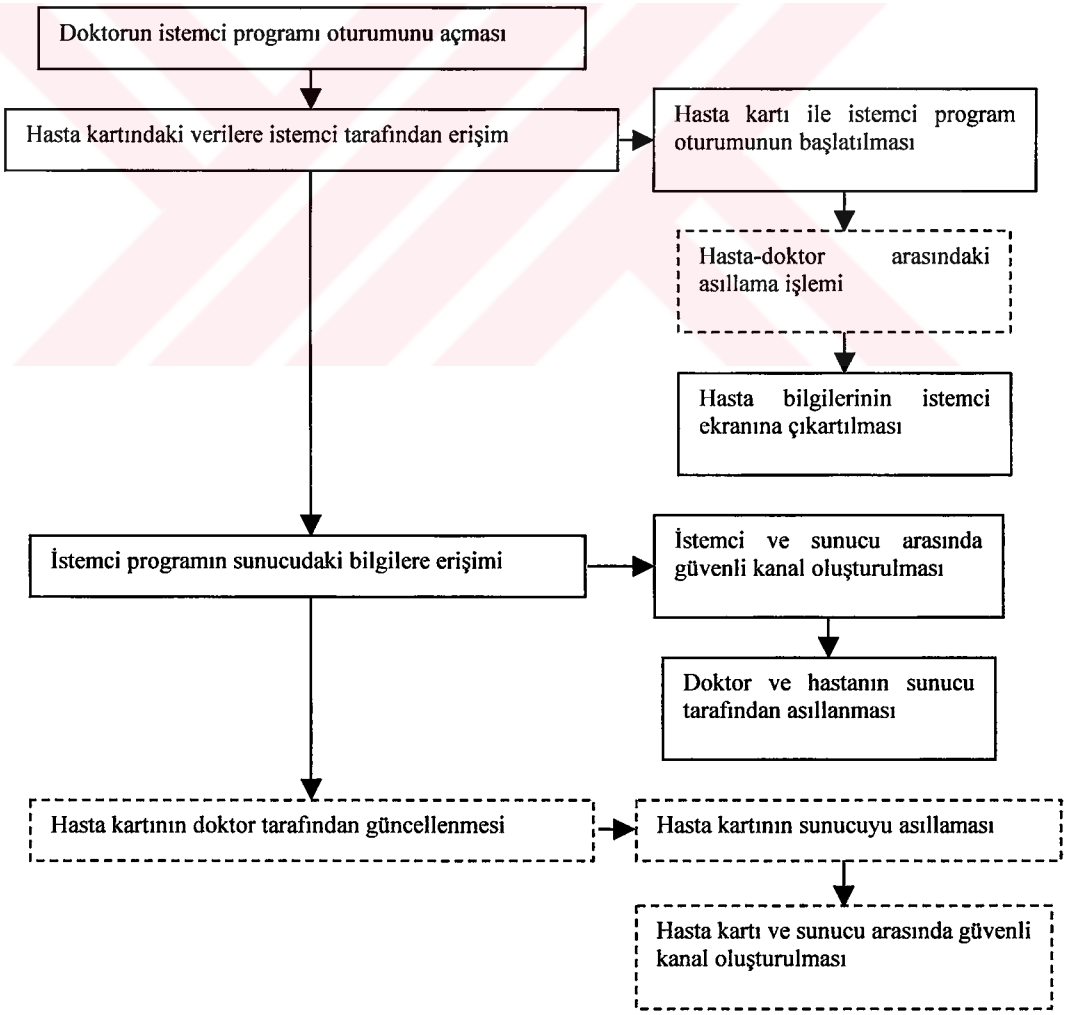
6. Oturumun sürekliliği ve hareketin güvenliği: Her oturum adımı, sunucu, istemci veya akıllı kartlar tarafından şifrelenir veya imzalanır.

7. Üç taraflı asıllama: Güvenilir üçüncü taraf (TTP-Thrusted Third Party) şifreleme ve şifre çözme yeteneklerine sahiptir ve anahtarların saklanması işlemini kontrol eder. Bu üçüncü tarafın da kendisini sunucuya asıllaması gerekir. Tez çalışmasında sertifika dağıtımını yapan güvenilir üçüncü taraf merkezi bilgi sisteminin içinde bir modül olarak tasarlanmıştır. Bu durumda sunucu programının ve kart basma programının bulunduğu merkez güvenilir kabul edilmiştir.

10. GÜVENLİK PROTOKOLÜNÜN TASARIMI

Bir önceki bölümde ele alınan sağlık sisteminde sistem bileşenleri tanıtılarak bileşenlerin koşullara bağlı olarak verilere erişim ve işlem yapma konularındaki yetkileri ortaya konmuştu. Ortaya çıkan gereksinimler ve problemlerinin çözümleri doğrultusunda bu bölümde tasarlanan güvenlik protokolü tanıtılacaktır.

Protokol, Bölüm 3'te de anlatıldığı üzere "taraflar" arasında kurulur ve farklı işlevlere sahip adımlardan oluşur. Adımlar, işlevlerin sırasına göre birbirini takip



Şekil.10.1. İşlem akışı

eder. Sağlık sisteminde taraflar arasında yürütülen işlemlerin akışı ortaya konduktan sonra her işlemin yürütülme koşulları, hangi taraflar arasında yürütüldüğü, tarafların işlem üzerindeki yetkilerine bağlı olarak güvenlik protokolünün tasarımı yapılmıştır. Sözü edilen her işlemin başarımı için kendi içinde bir güvenlik protokolü tasarımı yapılmıştır. Sistemde gerçekleştirilen işlemlerin akışı Şekil 10.1'de görülmektedir. Kesikli çizgiler içinde yer alan işlemler, her durumda gerçekleştirilen işlemlerden ayrılmıştır. Hasta kartının doktor tarafından güncellenmesi veya doktor kartının hasta kartı tarafından asıllanmasına her örnekte gerek duyulmayabilir.

10.1. Doktor'un İstemci Programı oturumunu açması

Doktorun istemci makinadan sisteme girişi, istemci programı oturumunun açılmasıyla sağlanır. Oturum açıldıktan sonra, doktor kartındaki veriler istemciye aktarılır. İstemci programda doktor oturumunun açık kalması doktor kartının okuyucudan çıkartılması veya doktorun kendi isteği ile oturumu sonlandırması ile son bulur. İstemci programı oturumunun açılması iki farklı asıllama işleminin başarımını gerektirir:

1- Doktor - doktor kartı arasındaki asıllama işlemi

2- Doktor kartı - istemci program arasındaki asıllama işlemi

Doktor - doktor kartı arasındaki asıllama işleminde bir önceki başlıkta anlatıldığı gibi doktorun PIN numarasını girmesi ile kartın kart sahibini doğrulaması istenir. Burada gerçekleştirilen işlem varlık asıllamadır.

Doktor kartı ile istemci programı arasındaki asıllama işlemi, akıllı kartın istemci programındaki şifreyi kontrol ederek üzerindeki verilerin program tarafından okunmasına izin vermesidir. İstemci programı ile doktor kartı üzerinde tutulan gizli anahtarlar aracılığıyla, simetrik şifreleme (3DES) kullanılarak doktor kartı ile istemci programının veri aktarımına izin verilir. Bu anahtarların uyuşmadığı durumda istemci programı, doktor kartındaki verileri okuyamaz. İstemci program ile akıllı kart arasındaki verilerin şifrelenerek gönderilmesinin yanında kartın

okuyucu programın tipini öğrenmesi ile birlikte hangi veri üzerinde ne kadar erişim izni vereceğini de belirlemesi sağlanır.

Doktorun istemci programı oturumunu açması için aşağıdaki işlemler yürütülür:

İstemci program doktorun PIN kodunu öğrendikten sonra akıllı karta göndererek kartla iletişime geçer. PIN kodunu gizli anahtarla şifreleyerek akıllı karta gönderir.

Akıllı kart şifreyi çözerek hem istemci programı ile akıllı kart arasındaki asıllama işlemini gerçekleştirir hem de PIN kodunu öğrenmiş olur.

Alınan PIN ile kartta saklı olan kullanıcı PIN karşılaştırılarak doktorun akıllı kartın gerçek sahibi olup olmadığı test edilmiş olur.

Bu aşamadan sonra artık güvenlik protokolünde doktoru, onun akıllı kartı temsil edecektir. Buradan sonraki adımlarda doktordan sözedildiğinde aslında işlemlerin onun akıllı kartı üzerinden yapıldığı unutulmamalıdır. Hasta için ise PIN kontrolü olmadığından başlangıçtan bu yana sistemde hasta kendi akıllı kartı tarafından temsil edilmektedir. Protokolün tarafları olarak hasta ile hasta kartı, doktor ile de doktor kartı aynı anlamda kullanılacaktır.

10.2. İstemci Programının hasta kartındaki verilere erişimi

Doktor kartının okunmasının ardından hasta bilgilerine erişim mümkün hale gelir. Hasta kartındaki bilgilerin doktor tarafından okunabilmesi için kart okuyucu üzerinde bir ekran olmadığından bilgiler terminal ekranından gösterilir. Ekranda verilerin gösterilmesi aşamasında istemci programına güven duyulur. İstemci program ile kart okuyucu arasında veriler şifrelenir. İstemci programının hasta kartındaki verilere erişimi üç adımda yürütülür:

1- Hasta kartı ile istemci program oturumunun başlatılması,

2- Hasta-doktor arasındaki asıllama işlemi,

3- Hasta bilgilerinin istemci ekranında görüntülenmesi.

10.2.1 Hasta kartı ile İstemci Program oturumunun başlatılması

Hasta kartı ile istemci program oturumunun açılması iki ayrı koşulun sağlanmasını gerektirir.

Bu oturumun açılması için doktor oturumunun açık olması koşulu aranır. İstemci programı ile oturumunu başlatmış olan herhangi bir doktor yoksa hasta kartı da istemci programı aracılığıyla okunamaz.

Bir diğer kontrol, hasta kartı ile istemci programı arasındaki asıllamadır. Bunun için hasta kartında ve istemci programında tutulan gizli anahtar kullanılır. İstemci programının gizli veriler hariç diğer kimlik ve hasta bilgilerine erişimi için hasta kartında da bulunan gizli anahtara sahip olması gerekir. Simetrik şifreleme yöntemi ile anahtarlanan hasta bilgileri bu koşul sağlandığında istemci program tarafından ekrana çıkartılır.

Hasta bilgileri arasında gizli veriler de mevcuttur (hassas hastalık bilgileri). Bu verilerin okunabilmesi için hasta kartı tarafından doktorun kimliğinin doğrulanması gerekir.

10.2.2. Hasta ile Doktor arasındaki asıllama işlemi

Hasta kartında tutulan acil durum verileri dışında bazı veriler, yalnızca o hastalığın takibini yapan doktor tarafından görülebilir. Hassas veriler ile diğer veriler arasında ve hassas verilerin kendi aralarındaki ayrım, doktor kimliğine bağlı olarak yapılmaktadır. Hassas verilere erişim hakkı olan doktor kimlikleri hasta kartında tutulmaktadır. Bunun için hassas veriler hasta kartı tarafından istemci programına gönderilmeden önce doktorun asıllaması yapılır. Eğer doktor hastanın kişisel doktorlarından birisi değilse asıllama yapılmaz.

Hastanın doktoru asıllaması işlemi, istemci programının kullandığı doktor kimliğinin hasta kartı tarafından doğrulanmasıyla gerçeklenir.

Hasta kartı ile doktor kartı arasında yürütülecek asıllama işlemine başlamadan önce hasta kartını okumak isteyen doktorun tanımlayıcı kimlik kodunun (D_ID) hasta kartındaki doktor listesinde olup olmadığı kontrol edilir. Eğer bu listede yoksa asıllama işlemine gerek duyulmaz. Listede doktorun tanımlayıcı kodu

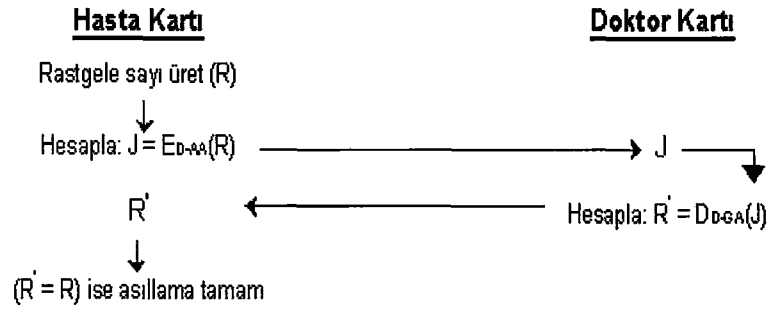
varsa, doktora ilişkin gizli bir verinin kartta bulunduğu sonucuna varılır ve bu verinin de istemci programına gönderilebilmesi için doktorun kimliğinin doğrulanması gerekir. Doktorun kimliğinin hasta kartı tarafından doğrulanması için izlenebilecek farklı tutumlar vardır. Bunlardan ikisi açık anahtar şifrelemenin farklı iki kullanımıdır. Diğer öneriler ise bu iki yöntemde karşılaşılan problemleri çözebilmek için geliştirilmeye çalışılmıştır.

Birinci yöntem, hasta kartı tarafından doktorun açık anahtarı kullanılarak bir jeton üretilmesi ve bu jetonun doktor kartı tarafından imzalanmasının istenmesidir. İmzalanan jeton hasta kartında bir karşılaştırma işlemi yapılarak kontrol edilir ve doktorun kimliği doğrulanır.

İkinci yöntemde ise, doktorun ürettiği imzanın hasta kartı tarafından doktorun açık anahtarı kullanılarak açılması ve elde edilen verinin karşılaştırılarak doktorun kimliğinin onaylanmasıdır. Her iki yöntem aşağıda açıklanmaktadır.

10.2.2.1 Jeton imzalama yöntemi

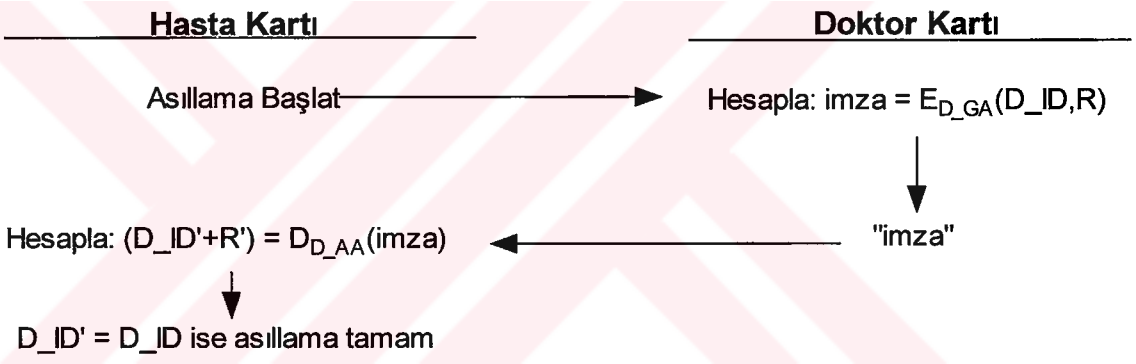
Hasta kartında rastgele sayı (R) üretilir. R, doktorun açık anahtarı ile Eliptik Eğri şifreleme algoritması kullanılarak şifrelenir ($E_{D\text{-}AA}$) ve Jeton (J) oluşturulur. Doktor kartı jetonu aldıktan sonra doktorun gizli anahtarı ile Eliptik Eğri şifre çözme algoritmasını kullanarak şifreyi çözer ($D_{D\text{-}GA}$) ve R' hesaplanmış olur. Hasta kartı kendisine gönderilen bu değeri R ile karşılaştırır. Karşılaştırma sonucu olumlu ise asıllama başarıyla tamamlanmış olur. (Bakınız Şekil.10.2)



Şekil.10.2. Jeton imzalama

10.2.2.2. Kimlik bilgisi imzalama

Doktorun asıllanması için hasta kartı, doktorun asıllama verisini ister. Asıllama verisi, doktorun tanımlayıcı kodunun (D_ID) doktorun gizli anahtarı ile şifrelenmiş ($E_{D\text{-}GA}$) halidir ve doktor kartı tarafından oluşturulur. Doktorun sayısal imzasıdır. Asıllama işleminin yapılabilmesi için burada hasta kartının doktorun sertifikasını doktorun açık anahtarı ile açması ($D_{D\text{-}AA}$) ve bulduğu sonucu(D_ID'), D_ID ile karşılaştırması gerekir. Karşılaştırma işleminin sonucuna göre asıllama işlemi başarıma ulaşır (Şekil.10.3). Doktor kartının ürettiği mesajın her seferinde aynı olması bu mesajın ele geçirilmesi durumunda daha sonra aynı amaçla bir saldırgan tarafından da kullanılması riskini ortaya çıkarır. Bu amaçla bir zaman pulu veya rastgele sayı üretilerek D_ID'ye eklenir.



Şekil.10.3. Kimlik bilgisi imzalama

10.2.2.3.Açık anahtar kriptografi ile imzalamanın uygulamada kısıtları

Yukarıda tanıtılan, kullanılabilecek iki yöntemin de akıllı kart açısından kısıtları vardır. Akıllı kartta asimetrik ve simetrik şifreleme desteklenmesine karşın asimetrik şifre çözme desteklenmemektedir. Bunun nedeni bellek yetersizliğidir. Bu problemin çözümü için öncelikle yukarıda genel olarak kullanılabilecek her iki yöntemdeki kısıtlar ve olası çözüm yolları incelenmiştir.

Birinci yönteme bakılırsa burada hasta kartı tarafında problem olmamakla birlikte, doktor kartı tarafında şifre çözme işlemi gerekmektedir. Bunun yerine şifre çözme işleminin doktor kartı yerine terminalde yapılması bir çözüm yolu olarak tercih edilebilir mi? Hayır, çünkü bu durum doktorun açık anahtarının terminale

alınmasını yani akıllı kartın dışına çıkarılmasını gerektireceğinden sistemin güvenliğinin sarsılması anlamına gelecektir.

İkinci asıllama yönteminde ise hasta kartının asıllamayı yapabilmek için doktor sertifikasını doktorun açık anahtarı ile çözmesi gerekmektedir. Şifre çözme işleminin akıllı kart üzerinde yapılması durumu burada da karşımıza çıkmaktadır. Burada da şifre çözme işleminin akıllı kart dışına alınması düşünülemez çünkü asıllamayı yapacak olan hasta kartıdır.

Karşılıklı asıllama işleminin iki akıllı kart arasında yapılıyor olmasından kaynaklanan bu problem iki yöntemi de geçersiz kılmaktadır. Kullanılacak akıllı kartın belleğini arttırıp şifre çözme algoritmasını da karta yüklemek dışında iki akıllı kart arasında gerçekleşecek asıllama işlemi için yeni bir asıllama protokolü tasarlanması mümkündür.

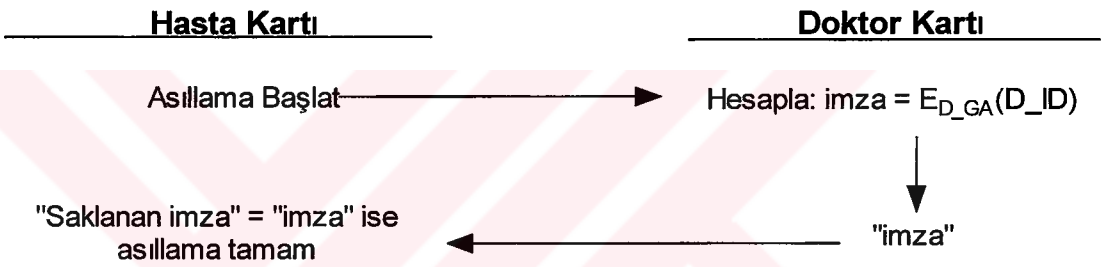
10.2.2.4. Şifre çözme kullanmadan asıllama

Asıllama işleminin gerçekleştirileceği taraflardan ikisinin de akıllı kart olması güvenlik sağlaması açısından aslında bir olumluluktur. Buradan yola çıkılarak bir asıllama protokolü tasarlanabilir.

Doktorun asıllama verisi hasta kartında tutulacak, doktor kartı tarafından gönderilen asıllama verisi ile karşılaştırılarak asıllama yapılacaktır (Şekil.10.4). Burada doktorun asıllama verisinin hastanın elinde olması hastanın tek başına sistemde sahip olamayacağı erişim haklarını elde etmesi riskini ortaya çıkaracaktır. Bu riski ortadan kaldırmak için bir çözüm mevcuttur. Hasta kartı - doktor kartı arasındaki bu asıllama işlemi dışında doktorun asıllama verisine (imza) ihtiyaç duyan tüm kimlik denetleme ve asıllama işlemleri için rastgele sayı ve/veya zaman pulu kullanılabilir. Belirlenen süreyi aşan asıllama verisi, onaylayıcı tarafından zaman pulu kontrol edilerek belirlenir ve geçersiz kabul edilir. Yalnızca hasta kartına yüklenen ilgili doktorun asıllama verisi zaman pulu içermez. Çünkü uzun süreli olarak kartta tutulacak bu bilgi ile karşılaştırma yapılarak asıllama yapılacaktır.

Bunun dışında ortaya çıkacak güvenliği tehdit eden durum yoktur. Asıllama verisi, akıllı kartın dışına çıkarılarak herhangi bir program tarafından kullanılamaz, akıllı kartta güvenli bir biçimde saklanmaktadır.

Doktorun asıllama verisinin oluşturulması işlemi ise doktorun gizli anahtarına sahip olan yalnızca doktor kartında yapılabilir. Doktorun gizli anahtarı kullanılarak, zaman pulu eklenmeden doktorun tanımlayıcı kodu D_ID şifrelenir. Bu durumda doktor kartının hasta kartı tarafından asıllanması işlemi sadece daha önce hasta kartına kayıtlı olan bilgi ile doktor kartında oluşturulan asıllama verisinin karşılaştırılması işleminden ibaret olacaktır. Burada asıllama işleminin iki akıllı kart arasında gerçekleşmesi, şifre çözme işleminin gerekliliğini ortadan kaldırmıştır.



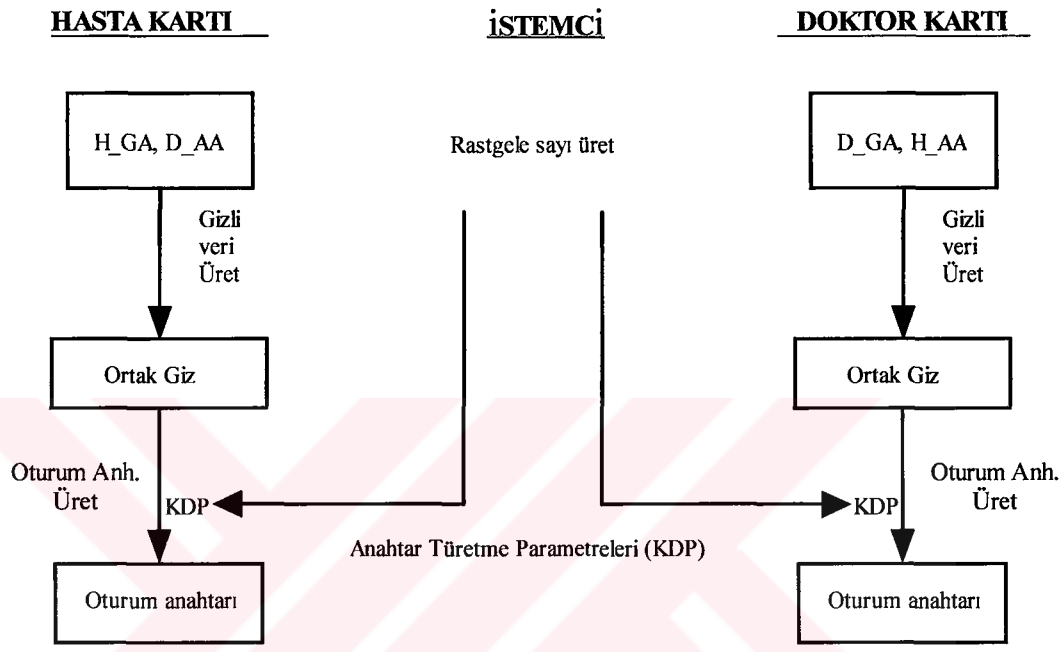
Şekil.10.4. Şifre çözme kullanmadan asıllama

Kart okuyucunun bağlı olduğu terminalden asıllama verisine bir saldırının gerçekleşme olasılığı vardır. Terminal üzerinden bu bilgiyi ele geçiren kişi zaman pulu olmadığından istediği zaman hassas verilere ilişkin doktorun bulunmadığı durumda da bu doktorun yerine geçerek hassas bilgileri okuyabilir. Doktorun imza bilgisi protokolün diğer adımlarında bu biçimde kullanılmayacak olsa da buradaki kullanım açısından oluşan risk gizliliği ortadan kaldırmaktadır.

Burada varlık asıllamada kullanılabilecek herhangi bir gizli veri (şifre veya PIN gibi) de tutulabilirdi. Fakat bu her doktor için sadece bu işlem için ayrı bir gizli anahtar tutulması anlamına geleceğinden gizli bilginin kartta zaten kayıtlı bulunan anahtar ve algoritmalarla türetilmesi düşünülmüştür.

10.2.2.5. Önerilen asıllama protokolü

Yukarıda açıklandığı üzere tek başına açık anahtar şifreleme yöntemine dayanan asıllama işleminde engeller veya güvenlik problemleri ile karşılaşılmaktadır. Bunun için açık ve gizli anahtar şifreleme yönteminin birlikte kullanıldığı bir asıllama protokolü tasarlanmış ve yapılan uygulamada sözedilen nedenlerle bu tasarımın kullanılması tercih edilmiştir.



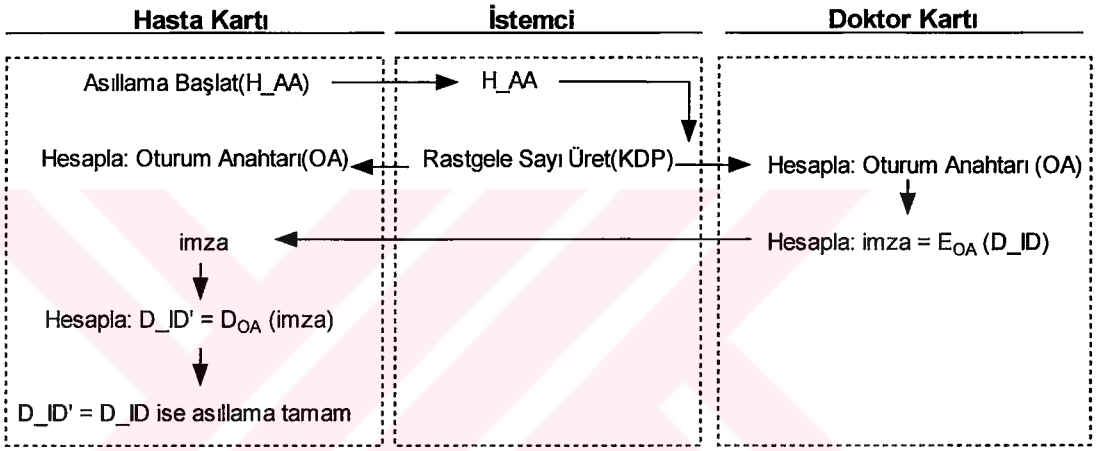
Şekil.10.5. Hasta-doktor arasındaki asıllama için oturum anahtarı üretimi

Hasta-kartı, ilgili doktoru asıllamak için onun açık anahtarını(D_AA) ve kişisel tanımlayıcı kodunu(D_ID) doktor tablosunda tutmaktadır. Her iki taraf da karşı tarafın açık anahtarı, kendi gizli anahtarı ve rastgele üretilen anahtar türetme parametrelerini kullanarak bir oturum anahtarı üretir (Şekil.10.5). Bu oturum anahtarı yalnızca tek bir asıllama için geçerli olacaktır. Her defasında yeni bir oturum anahtarı üretilir. Burada oturum anahtarı paylaşımı sorunu Diffie Hellman anahtar paylaşım protokolü kullanılarak çözülmüştür. Bu protokolde her iki tarafın kendi gizli anahtarı ve diğer verileri kullanarak ürettiği oturum anahtarı birbirinin aynı olacağından bu anahtarın paylaşımı sorunu ortadan kalkar. Üretimde kullanılan Eliptik Eğri anahtar çiftinin gizli anahtarları ise her iki taraf açısından tek ve tarafların kimliğini doğrulamakta kullanılan verilerdir. Bu sayede

üretilen oturum anahtarı, kaynağında iki tarafın gizli anahtarları olduğundan iki tarafın birbirini asıllaması için de kullanılabilir.

Kendini asıllamak isteyen tarafın bu anahtarı kullanarak kendi imzasını oluşturması ve karşı tarafa göndermesi mümkündür. Burada imzalama ve imzayı onaylama işlemi için simetrik şifreleme yöntemi kullanılmalıdır. Yapılan uygulamada, akıllı kartın da desteklediği 3DES tercih edilmiştir.

Hasta - doktor arasındaki asıllama işlemi Şekil 10.6'da görüldüğü gibi aşağıdaki basamaklardan oluşur :



Şekil.10.6. Hasta-Doktor arasında asıllama protokolü

1- Hasta kartı istemciye asıllamanın başlayacağını bildirir ve bu bildirim sırasında hastanın açık anahtarını istemciye gönderir.

2- İstemci, hasta kartı ve doktor kartının anahtar türetme parametresi (KDP) olarak kullanmaları için rastgele bir sayı üretir.

3- İstemci, doktor kartına KDP'yi ve hastanın açık anahtarını (H_AA) gönderir.

4- İstemci, hasta kartına KDP'yi gönderir. (asıllanacak doktorun açık anahtarı hasta kartında zaten tutulmakta)

5- Hasta ve doktor kartları oturum anahtarı üretirler.

6- Doktor kartı oturum anahtarı ile D_ID 'yi birlikte kapatır ve hasta-kartına gönderir.

7- Hasta kartı, aynı oturum anahtarını kullanarak gelen paketi açar, kendisinde kayıtlı D_ID ile sonucu karşılaştırır. Sonuç olumlu ise asıllama başarıma ulaşır.

10.2.3. Doktor bilgilerinin hasta kartında saklanması

Hasta bilgilerinin arasında gizli kalması gereken bilgilerin herhangi bir doktorun okuyabileceği bilgilerden ayrılabilmesi için hasta kartında kişisel doktorların tablosu tutulur.

Doktor tablosunun alanları Tablo 10.1'de gösterilmiştir. Tabloda, bu gizli bilgilere ilişkin doktorların açık anahtarları, doktorların tanımlayıcı kodları(D_ID) ve hasta kartından başka bir yerde kullanılmayan doktor_no bulunmaktadır. Gizli veriler, diğer verilerle aynı tabloda tutulmakla birlikte bu verilerin doktor_no alanı 0'dan farklıdır. Doktor_no alanı 0'a eşit olan veriler ise istemci tarafından herhangi bir doktorun varlığında okunabilir. Doktor_no kontrol edilerek doktora ilişkin gizli veriler bulunur ve doktorun diğer verilerle birlikte bu verilere de erişimine izin verilir.

Tablo.10.1.Kişisel Doktor Tablosu

Doktor_no	Doktor_ID	Doktor_Açık Anahtarı
:	:	:
:	:	:

10.3. İstemci programın sunucudaki bilgilere erişimi

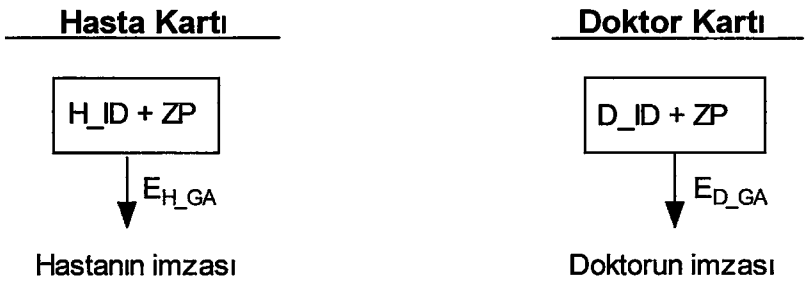
Doktorun hastanın geçmiş tedavi bilgilerine erişmek istediği veya muayene sonucunda bilgileri güncellemek istediği durumda istemci programı sunucuya erişim isteği sözkonusu olur. Bunun için gerekli olan işlemler istemci ve sunucu arasında güvenli kanal oluşturulması ve doktor ile hastanın sunucu tarafından asıllanmasıdır.

10.3.1. Doktor+hasta'nın sunucu tarafından asıllanması

Doktor, sunucuya bağlanarak hastanın geçmiş bilgilerini okumak veya hasta bilgilerini güncellemek istediğinde kimliğini belirten doktorun iddia edildiği gibi

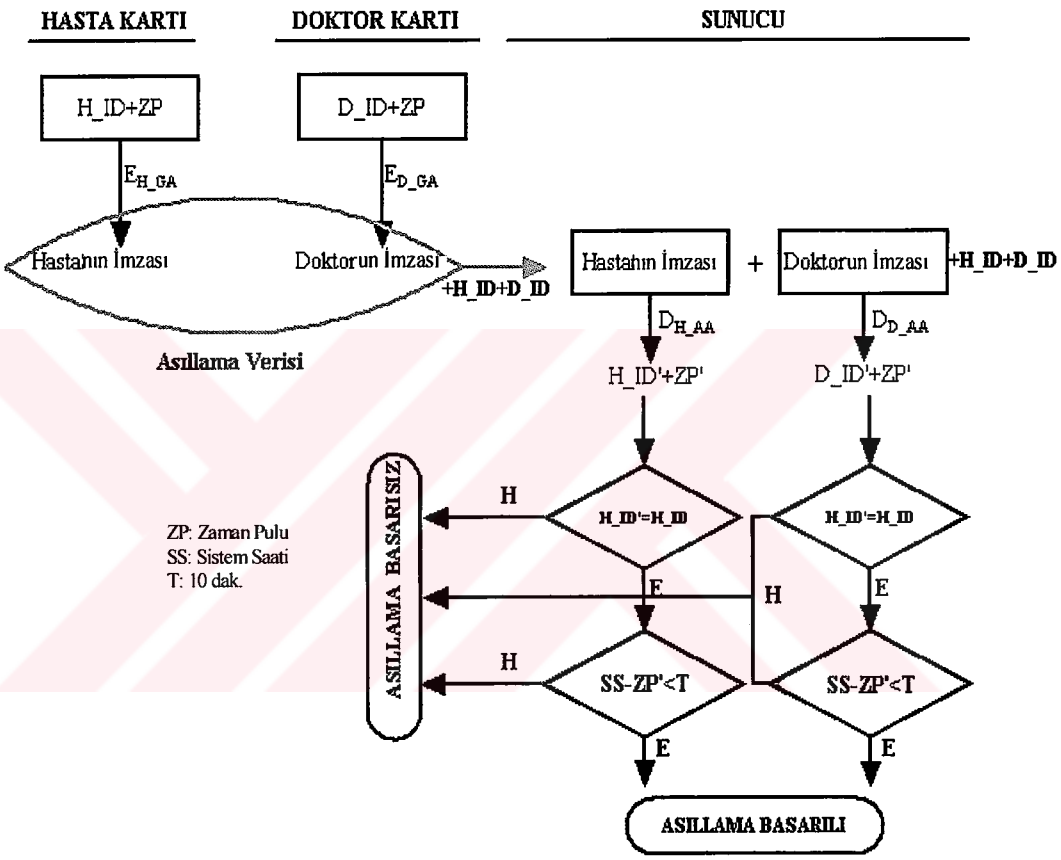
gerçekten sistemin diğer ucundaki kişi olup olmadığından sunucu emin olmalıdır. Ayrıca tek başına doktorun kimliğinden ve varlığından emin olmak da burada sistemin güvenliğini açısından yeterli değildir. Güncellenecek veya öğrenilecek olan hastanın bilgileri olduğuna göre hastanın bu işi yapması için söz konusu doktora izin veriyor olması gerekir. Bu nedenle doktorun yanında gerçekten söz konusu hastanın var olup olmadığından sunucunun emin olmasıyla doktora hasta hakkında istediği işlemi yapması için izin verilir. Sunucunun istenen işlemi yerine getirmesi için hem doktoru hem de hastayı aynı anda asıllaması gerekmektedir.

Bunun için hasta ve doktor kartları tarafından üretilen imzalar, istemci tarafından sunucuya yollanır. Burada istemci makinaya yapılacak bir saldırı ile veya istemci-sunucu arasındaki haberleşme sırasında imzalar ele geçirilebilir. Ele geçiren kişi, daha sonra sunumcuyu doktor ve hastanın varlığına dair kandırma olanağına sahip olur. Bu riske karşı çözüm olarak zaman pulu kullanımı uygundur. Sunucuyu kandırmaya çalışan kişi ele geçirdiği imzayı daha sonra kullanmaya kalktığında sunucu imzanın üretim süresine bakarak geçersiz kabul edecektir. Saldırganın yapabileceği bir diğer işlem de zaman pulunu değiştirmek olabilir. Fakat zaman pulu (ZP) kişinin tanımlayıcı verisi ile birlikte imzanın içine gömülüdür (Şekil.10.7). Bu nedenle ele geçirdiği veri paketini açamadığı sürece imzayı ele geçirmiş olması sisteme zarar vermez. Fakat söz konusu saldırgan zaman pulunu değiştirmek yerine sunucunun sistem saatini değiştirmeyi başarırsa bu durumda sistemi korumak için zaman pulu kullanımı işe yaramayacaktır. Ayrıca doğrudan sunucuya yapılan bir saldırının başarımı söz konusu olduğunda sunucudaki verilerin güvenliğinin sağlanması ayrı bir güvenlik başlığıdır. Protokolün tasarımında sunucu taraf güvenli kabul edilmiştir.



Şekil.10.7. Hasta ve doktor imza üretimi

Şekil 10.7'de doktor ve hasta kartında imza üretimi gösterilmiştir. Her iki taraf da kendi gizli anahtarını kullanarak Eliptik Eğri şifreleme yöntemi ile kendi tanımlayıcı verisini ve zaman pulunu kapatır. Sunucuya ulaşan imzalar, ayrı ayrı doktor ve hastanın açık anahtarları kullanılarak sunucu tarafından açılır. Zaman pulları kontrol edilir, sonuçlar doktor ve hastanın ID'leri ile karşılaştırılır. Karşılaştırmaların sonucunda asıllama işlemi başarıma ulaşmış olur (Bakınız Şekil.10.8).



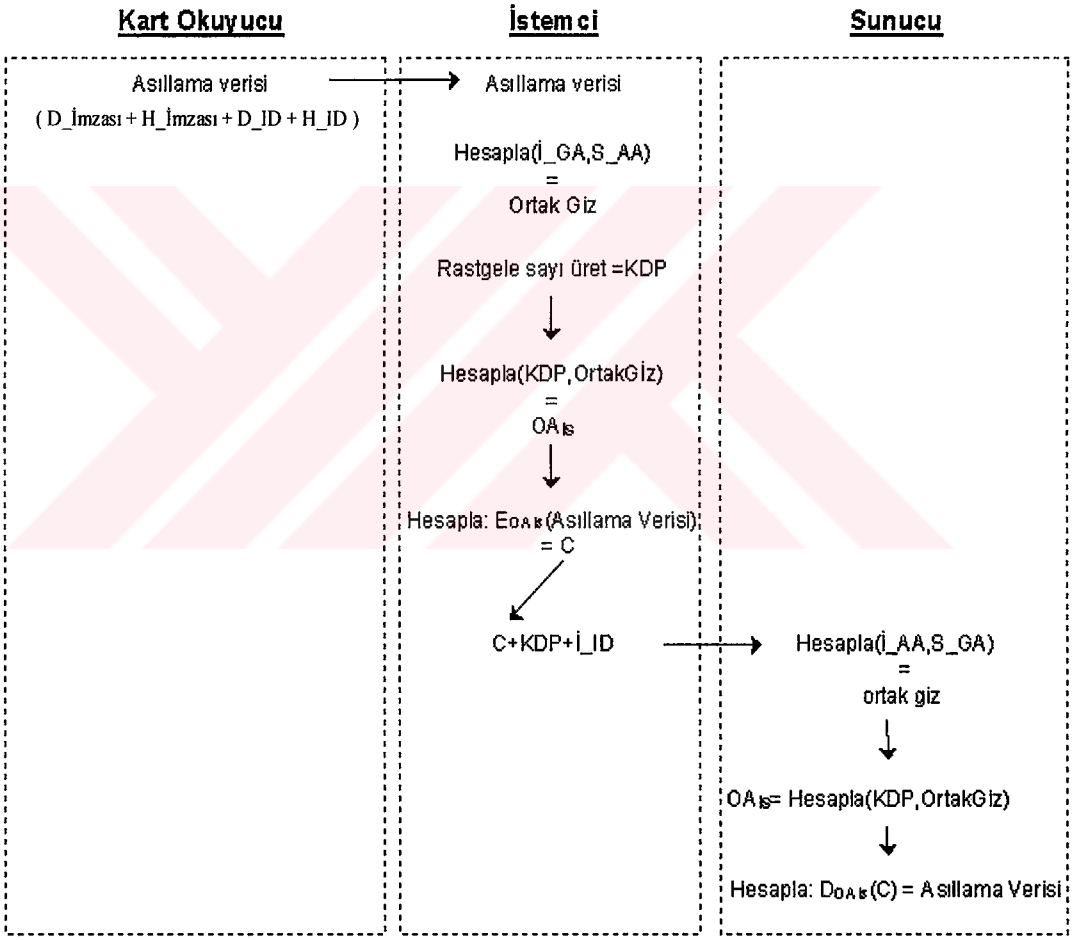
Şekil.10.8. Sunucunun hasta+doktor'u asıllaması

10.3.2. İstemci-Sunucu arasında güvenli kanal oluşturulması

Doktor ve hasta imzalarının (asıllama verisi) gönderimi bir güvenli kanal üzerinden olmalıdır. Bunun nedeni sertifikaların ele geçirilmesinde oluşacak riskten çok veri bütünlüğünün bozulması riskidir. Asıllama verisinin bütünlüğü bozulmadan sunucuya ulaşması için istemci ve sunucu arasında bir güvenli kanal oluşturulmalıdır. Güvenli kanal, istemci ve sunucu arasında paylaşılacak bir oturum anahtarı yardımıyla verilerin 3DES ile şifrelenmesi sonucu oluşturulur.

Oturum anahtarının oluşturulması için yine anahtar alışverişinin gerekliliğini ortadan kaldıran Diffie Hellman anahtar paylaşımı yöntemi kullanılmıştır.

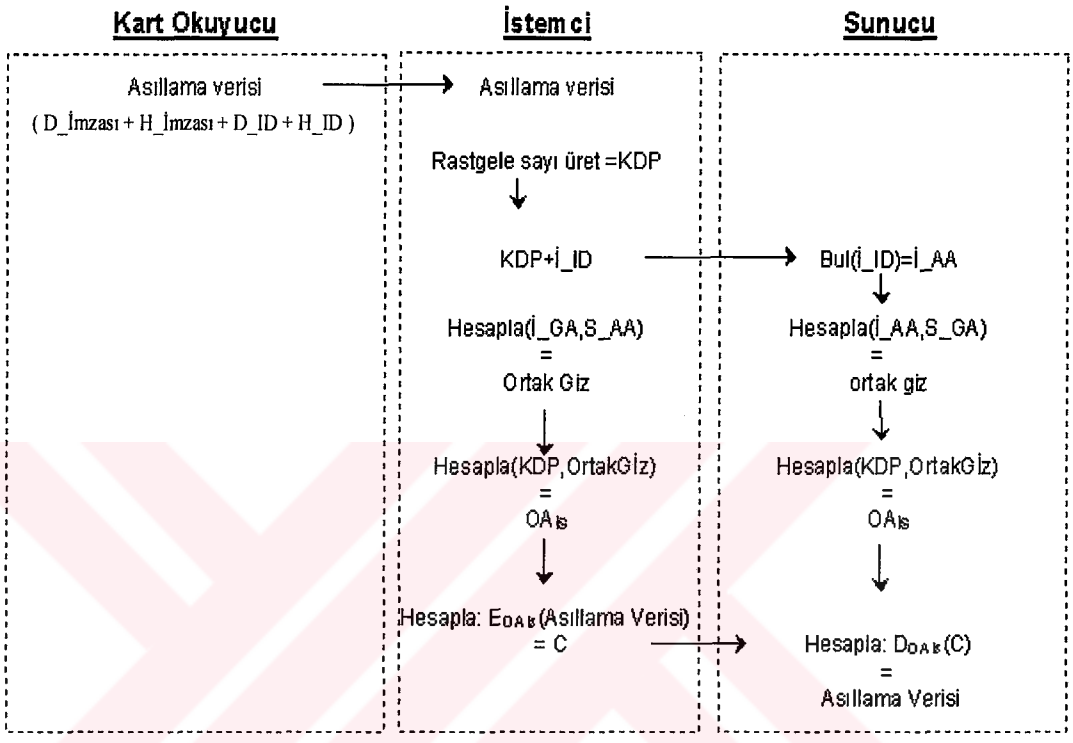
Şekil.10.9'da görüldüğü gibi istemci asıllama verisini paketleyebilmek için istemci-sunucu oturum anahtarını (OA_{IS}) üretir. Burada kullanılacak "ortak giz"in hesabı istemci tarafından istemci gizli anahtarı (İ_GA) ve sunucu açık anahtarı (S_AA) kullanılarak hesaplanır. Veri akışını başlatan taraf istemci olduğundan oturum anahtarının üretilmesi için gerekli olan anahtar türetme parametresini de istemci üreterek sunucuya diğer bilgilerle birlikte gönderir.



Şekil.10.9. İstemciden sunucuya güvenli kanal oluşturulması-1

Şekil.10.10'da görüldüğü üzere, sunucu oturum anahtarının hesaplanması işlemine ancak istemci tüm paketi hazırlayıp gönderdikten sonra başlayabilir. Bu ise gereksiz yere zaman kaybına neden olacaktır. Oysa istemci sunucuya KDP ve istemci tanıma kodunu (İ_ID) ne kadar erken gönderir ve istemci ile arasındaki

protokolü başlatırsa, istemci de oturum anahtarını üretme işlemine daha önce başlayabilir. Bu nedenle "ortak giz"in hesaplanmasından önce KDP hesaplanabilir ve İ_ID ile birlikte Şekil 10.10'da görüldüğü gibi sunucuya gönderilerek sunucunun da protokolü başlatması sağlanabilir.



Şekil.10.10. İstemciden sunucuya güvenli kanal oluşturulması-2

Sonuç olarak bu iki yöntem arasında uygulama açısından bir karşılaştırma yapıldığında birincisinin tercih edilmesine karar verilmiştir. Sunucunun aynı anda birçok kullanıcıya hizmet verdiği gözönüne alındığında sunucunun tek bir istemciden aynı işlem için bir yerine iki paket alması sunucuyu gereksiz yere meşgul edecektir.

Sunucu-istemci arasında oluşturulan güvenli kanal, sunucudan istemciye gönderilecek hasta verilerinin de güvenli bir biçimde istemciye ulaşmasını sağlar. Sunucu geçmiş hasta kayıtlarını istemciye gönderirken doktorun kimliğini kontrol eder ve eğer doktora ilişkin gizli tutulan bir hasta kaydı varsa bu kayıtları da diğer kayıtlarla birlikte listeler.

10.4. Hasta kartının sunucu tarafından güncellenmesi

Kullanılan önemli bir ilaç olduğunda, acil durumda önemi olan bir hastalık tespit edildiğinde, hastanın acil durumda önem kazanabilecek bir alerjisi tespit edildiğinde, veya bir ameliyat sonrası bir organ eksikliği ortaya çıktığında hasta kartının güncellenmesi gerekir. Burada bu işlemin bilgi merkezinde yapılması değil Şekil 10.1' de belirtilen işlem akışı içinde doktor tarafından istemci program aracılığıyla güncelleme yapılması tartışılacaktır.

İstemcinin sunucuyu güncelleme isteği ile birlikte gönderdiği veri paketi istemci-sunucu arasında bir önceki işlem adımında oluşturulan güvenli kanaldan gönderilir. Bu aşamada akıllı karta yazılması gereken veriler olabilir. Bu verilerin kimin tarafından nasıl yazılacağı, veri bütünlüğünün nasıl sağlanacağı, güvenlik protokolünün çözmesi gereken başlıklar olarak ortaya çıkmaktadır. Akıllı karta hangi verilerin yazılması gerektiğine karar verecek kişi doktor olacaktır. Fakat hasta kartının yalnızca gizli bilgilere ilişkin doktorları asıllayabildiği düşünülürse hasta kartı üzerinde güncelleme yapacak doktoru asıllama yeteneği hasta kartında bulunmamaktadır. Bu durumda asıllama işlemini istemcinin de yapamayacağı ortadadır. Protokolü oluşturan taraflar arasında bu işlemi yapabilecek tek taraf sunucu olarak karşımıza çıkmaktadır. Sunucunun doktoru asıllaması yukarıda anlatıldığı gibi asıllama verisi üzerinde sunucunun yaptığı işlemler sonucu gerçekleşmiş olur.

Fakat burada sunucunun doktoru asıllaması işlemi doktorun hasta kartına yazması için yeterli değildir. Sunucu doktoru asılladığını hasta kartına bildirmelidir. Ayrıca sunucu hasta kartına doktorun kimliğini doğruladığını öyle bir biçimde bildirmelidir ki bu bildirimin sunucudan geldiğine hasta kartı emin olmalıdır. İstemci üzerinden herhangi bir kişi doktorun sunucu tarafından asıllandığı konusunda hasta kartını kandırabilir.

İstemci program ve doktorun yetkilerinin sınırlandırılmasının bir diğer nedeni, hasta kartı ile bilgi merkezi arasında oluşabilecek veri tutarsızlığıdır. Doktor, sunucuya güncellemek üzere verileri gönderip sunucudan da hasta kartına yazmak üzere yazma iznini aldıktan sonra hasta kartına yazılacak veriler üzerinde değişiklik yapabilir. Yani sunucu verileri kendi üzerinde güncelledikten sonra

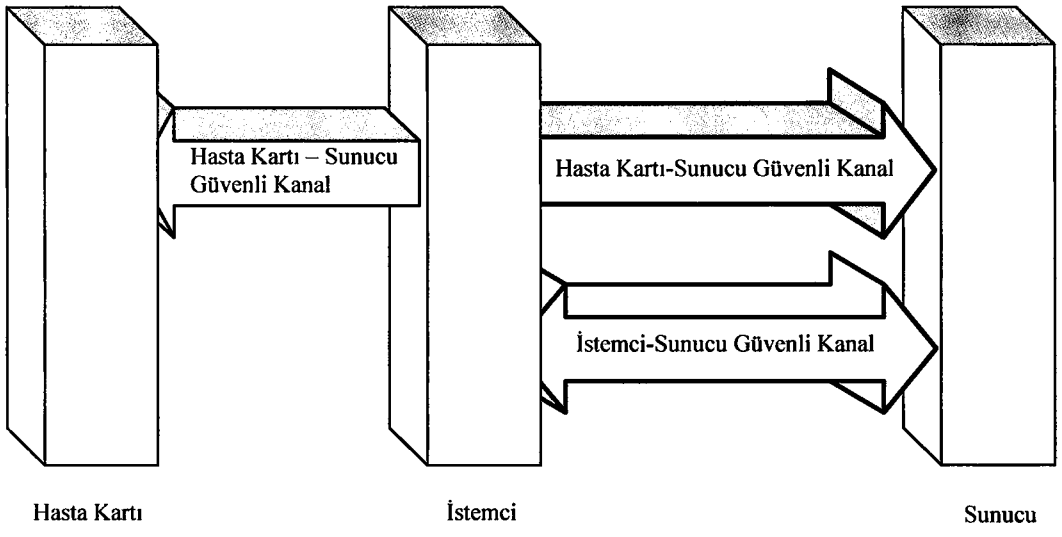
doktor karta yazacağı verileri değiştirebilir. Bu da sunucu ve hasta kartı arasında güvenlik protokolünden kaynaklı veri tutarsızlığına neden olur.

Bunu engellemek için doktor güncelleme işlemini tek vuruşta başlatmalı ve bundan sonrasını yeniden doktordan gelecek herhangi bir işleme gerek duymadan sistem kendisi çözmelidir.

Veri tutarlılığı açısından veri güncelleme işleminin öncelikle sunucuda başarıma ulaşması ardından karta yazılması; yani sunucuya yazılamamış verinin hasta kartına da yazılamaması gerekmektedir. Burada hasta kartı değil sunucu belirlenimli bir organizasyon tercih edilmiştir.

Bu türden olasılıklara karşı hasta kartının korunması ve veri tutarsızlığının önüne geçebilmek için hasta kartının sunumcuyu asıllaması gereği ortaya çıkmıştır.

Bu nedenle sunucu hasta bilgilerini güncelledikten sonra hasta kartına yazılması gerekenleri ayırır. İstemci paketi yollamadan önce doktor bu verileri işaretlemiştir. Bunları yazılması için hasta kartına kendi imzasını taşıyan onayıyla birlikte gönderir. Gönderim sırasında karta yazılacak verilerin bütünlüğünün bozulmaması yani yolda herhangi bir değişikliğe ve bozulmaya uğramadan hasta kartına ulaşması ise bir başka gözetilmesi gereken konudur. Burada sunucu ve hasta kartı arasında yürütülecek bir veri aktarımı söz konusudur ve protokolün diğer taraflarının müdahalesi engellenmelidir. Daha önce oluşturulan istemci-



Şekil 10.11. Protokol tarafları arasında oluşturulan güvenli kanalların simgesel gösterimi

sunucu arasındaki güvenli kanal bu işlem açısından güvenliğini yitirmiştir. Burada istemci programa yalnızca verileri uygun paketler halinde hasta kartına iletmek görevi düşmektedir.

Böylece hasta kartındaki veriler uzaktan erişimle sunucu tarafından güncellenir. Bu işlem basamağında gözetilmesi gereken iki önemli gereksinim şöyle ortaya çıkmaktadır:

1- Verilerin protokolün tarafları da dahil herhangi bir müdahaleye olanak bırakmayan güvenli bir kanaldan hasta kartına ulaşması, sunucu ile hasta kartı arasında Şekil 10.11' de görüldüğü gibi bir güvenli kanalın oluşturulması,

2- Hasta kartının bu verilerin sunucudan geldiğinden emin olması gerekmektedir.

Bu iki başlık üzerinden hasta kartının sunucuyu asıllaması için değişik yöntemler geliştirilebilir.

10.4.1.Açık anahtar şifreleme ile imza ve güvenli kanal oluşturulması

Akıllı karta açık anahtar şifre açma algoritmasının yüklendiği kabul edilirse; sunucunun kişisel anahtarı ile şifreleyeceği bir veriyi hasta kartının asimetrik şifreleme ile sunucunun açık anahtarını kullanarak açması sunucunun imzasını onaylamak için yeterli olabilir, fakat imza iletilen verinin güvenliğini sağlamayacaktır. Sunucunun açık anahtarını elinde bulunduran herhangi bir kimse veriyi açabilir. Çözüm olarak sunucu veriyi göndermeden kendi gizli anahtarıyla kapattıktan sonra bir de hasta kartının açık anahtarı ile kapatır. Böylece veriyi yalnızca hasta kartı kendi gizli anahtarı ile açabilir. Veri gönderilmeden önce sunucunun gizli anahtarı ile şifrelenecek, ardından oluşan imzalı metin, hastanın açık anahtarı ile şifrelenecektir:

Güncelleme veri paketi = E_{H_AA} [E_{S_GA}(Güncelleme verisi)]

Güncelleme veri paketini açmak için ise D_{S_AA} [D_{H_GA}(Güncelleme veri Paketi)] şifre çözme işlemleri yürütülerek güncelleme verisine ulaşılır. Bu işlem için her iki tarafın da gizli ve açık anahtar çiftleri mevcuttur. Bölüm 3.3'te tartışıldığı üzere akıllı kartların açık anahtar şifreleme algoritmasıyla şifrelenmiş bir veriyi çözme

yetenekleri yoktur. Bellek yetersizliğinden dolayı bu algoritmanın akıllı karta yüklenmesi tercih edilmemektedir.

10.4.2. Ortak giz kulanılarak oturum anahtarı üretilmesi

Hem veri bütünlüğünün ve güvenliğin sağlanması hem de asıllama yapılması gereksinimlerinin ikisini birden karşılayabilecek bir yöntem olarak sunucu ve hasta kartı arasında "ortak giz" üretilerek buradan her ikisinin de bir kullanımlık olmak üzere üretecekleri oturum anahtarı (OA_{SH}) ile verinin şifrelenmesi uygun bulunmuştur. Oturum anahtarı ile sunucu tarafından şifrelenen veri hasta kartında açılmakta, anlamlı bir veri elde edilip edilmediği kontrol edilerek veriler hasta kartı tarafından ilgili alanlara eklenmektedir. Verinin gerçekten sunucu tarafından paketlenip paketlenmediğini anlamak üzere iki yöntem kullanılabilir.

Birincisi verinin başında iki tarafın da bildikleri bir başlık bilgisi tutulmasıdır. Bu bilgi örneğin hastanın kimlik tanımlayıcı kodu (H_ID) olabilir. Veri paketi açıldıktan sonra hasta kartı başlığı kontrol ederek verinin sunucudan geldiği konusunda bir sonuca varabilir.

Bir diğer yöntem ise gönderilecek verinin çırpma değerinin hesaplanarak paketin arkasına eklenmesidir. Paket açıldığında hasta kartı tarafından çırpma değeri hesaplanır ve paketin arkasına eklenen değerle karşılaştırılarak verinin doğru oturum anahtarı ile şifrelenip şifrelenmediği kontrol edilmiş olur.

Uygulamada ikinci yöntem tercih edilmiştir. Çünkü birinci yöntem "bilinen kapalı veri" saldırısına açıktır. Bu saldırı biçimi arada duran bir kişinin verinin açık ve kapalı hallerini ele geçirmesi durumunda kullanılan algoritmaya ve anahtara ulaşmak için denemeler yapabilmesi anlamına gelir. Aynı başlık bilgisinin kullanıldığını ve bu başlığın bilinen bir değer olduğunu düşünürsek, verinin ilk bloğu, şifrelemeyi bu saldırıya açık hale getirir. Diğer yandan her oturum için ayrı anahtar kullanılıyor olması bu riski azaltıcı bir faktördür.

Sunucuda ve hasta kartında verilerin güncellenmesi işlemi aşağıda listelenen basamaklardan oluşur. Burada hasta ve doktorun sunucu tarafından başarıyla asıllandığını kabul ediyoruz.):

1- İstemci, hasta verilerini güncelleme bilgilerini istemci-sunucu arasında oluşturulan güvenli kanal üzerinden sunucuya gönderir. Bunun için istemcide de bir oturum anahtarı üretilir ve sunucuya yazılacak veriler bununla şifrelenir.

2- Sunucu bu paketi açtıktan sonra güncelleme işlemini yapar ve güncelleme başarıyla tamamlandıysa karta yazılması için gerekli işlemlere başlanır.

3- Sunucu doktor tarafından işaretlenen veri olup olmadığına bakar. Hasta kartına güncelleme isteği yoksa istemciye bildirimde bulunulur ve protokol burada sonlandırılır. Güncelleme isteği bulunuyorsa, sunucu hasta kartında tutulan verilerin güncellenmiş halini hasta kartı ile arasında oluşturulan oturum anahtarı ile verileri şifreleyerek hasta kartına gönderir. Böylece aynı zamanda verileri imzalamış olur. Burada istemci gönderilen veriyi açamaz çünkü gelen veri hasta kartında üretilen oturum anahtarı ile şifrelenmiştir.

4- İstemci sunucudan gelen veri paketini hasta kartına iletir. Hasta kartı oturum anahtarı ile paketi açar. Böylece sunucunun imzasını onaylayarak asıllamayı gerçekleştirir.

5- Sunucunun asıllaması sağlandıktan sonra hasta kartı gönderilen verilerin yazımına izin verir. Sunucunun asıllanması sağlanmadığı durumda yazma işlemi yapılmaz.

11. GÜVENLİK PROTOKOLÜNÜN GERÇEKLENMESİ

Önceki bölümde yapılan güvenlik protokolü tasarımı tanıtılan pilot sistem üzerinde gerçeklenmiştir. Bunun için bir gerçekleme ortamı olarak Zeit Control Basic Card program geliştirme kiti kullanılmıştır. Zeit Control akıllı kart program geliştirme kitinin sunduğu araçlardan yararlanılarak bir istemci/sunucu uygulama yazılımı geliştirilmiş, güvenlik protokolünün tasarımı gerçeklenmiştir.

Akıllı kartlarla ilgili uygulama geliştirmek için Zeit Control BasicCard gibi hazır bir uygulama geliştirme ortamı tercih edilmeseydi tez konusunun dışında pek çok başlıkta çalışmak gerekecekti. Bu başlıklardan birkaçı aşağıdaki şekile sıralanabilir:

- Makina dilinde programlama: Bazı işlemci kartları için C derleyicileri bulunsa da tüm kart işletim sistemini C'de yazmak mümkün değildi.
- T=1 protokolü gibi sekizli düzeyinde haberleşme protokolleri üzerine ayrıntılı bilgi sahibi olmak,
- Komut-Yanıt düzeyinde Blok-düzeyli haberleşme protokolleri üzerine ayrıntılı bilgi sahibi olmak,
- EEPROM'a yazmak için donanım düzeyinde programlama,
- Güvenlik algoritmalarının yazılması.

Tüm bunların yanında karmaşık ve pahalı geliştirme araçlarına ihtiyaç vardı. Zeit Control Programlanabilir işlemci kart ile bu konularda pek çok zorluk aşılmış bulunmaktadır.

11.1. Kullanılan akıllı kartın yapısı

Tez çalışmasında kullanıcı tarafından programlanabilir 8K EEPROM genişliği olan **Enhanced BasicCard ZC3.3**. kullanılmıştır.

Programlanabilir işlemci kartın temeli P-code kod yorumlayıcısına dayanmaktadır. Program derlendiğinde P-code oluşmaktadır. P-code makinadan bağımsız ve makina koduna benzeyen bir kodtur. Kod, karta yüklenerek burada kod yorumlayıcı tarafından çalıştırılır. Kodun çalışmadığı durumda yenileme olanağı bulunmaktadır.

ZCBASIC.exe derleyici, ZC-Basic kaynak kodunu P-code'a çevirir. P-Code, BCLOAD.exe kart yükleme programı kullanılarak akıllı karta yüklenir. Akıllı karttaki "Sanal Makina" P-Code komutlarını çalışma süresi boyunca çalıştırır.

Seçilen akıllı kartta uygulamaya uygun olarak 8K EEPROM, 17K ROM ve 256K RAM vardır. Bunlar içinde Basic Card işletim sistemi 107 Byte RAM ve 338 Byte EEPROM kullanır.

11.2. Akıllı kart EEPROM alanının organizasyonu

Akıllı kartta 8K'lık EEPROM alanının büyük bir kısmı güvenlik protokolünün gereksindiği algoritmaların yüklenmesi ile birlikte işgal edilmektedir. İşletim sistemi tarafından kullanılan alan yalnızca 338 sekizliden oluşmaktadır. Geriye kalan alanda hastaya veya doktora ilişkin verilerle birlikte akıllı kart uygulama programının tutulması sağlanmaktadır. Doktor kartında boş alan problemi yaşanmamıştır. Hasta kartına yüklenen uygulama yazılımı hasta kartındakinden daha geniş yer kaplamaktadır. Hastanın doktoru asıllaması, sunucuyu asıllaması gibi işlemler hasta kartı üzerinde yürütülürken, doktor kartının üzerinde bir PIN kodunun kontrolü dışında herhangi bir asıllama işlemi yürütülmemektedir. Hasta kartının veri organizasyonu bu nedenle bir problem olarak ortaya çıkmaktadır. Hastanın önemli hastalık verileri ve acil durum bilgilerinin kartta tutulduğu düşünüldüğünde hastalık verileri için geriye kalan alanın tamamının kullanılması gerekmiştir. 8K genişliğindeki EEPROM alanı bu organizasyon sonucunda sonuna kadar değerlendirilmiştir.

EEPROM'da 338 sekizli işletim sistemi tarafından kullanılmaktadır. En büyük alanı açık anahtar şifreleme, imzalama, oturum anahtarı oluşturma işlemlerini yerine getiren ilkelleri içeren Eliptik Eğri kütüphanesi kullanmaktadır. Eliptik Eğri kütüphanesi, kullandığı güvenli çırpma algoritması kütüphanesi ile birlikte 5K büyüklüğünde yer tutmaktadır. Eliptik Eğri şifreleme ilkellereinde algoritmanın hızlı

çalışması için bazı noktaların değerleri sabit eğri parametrelerine göre önceden hesaplanarak karta yüklenebilir. Bu şekilde önceden hesaplanarak yüklenebilecek noktaların sayısı 16-64 arasında değişmektedir. Ancak her nokta kart üzerinde 21 sekizli yer işgal edeceğinden önceden karta yüklenmeleri hızdan kazanırken yerden kaybetmeye yol açacaktır. EEPROM'daki yer sıkıntısından dolayı hasta kartında önceden yüklenen nokta sayısı 16 olarak belirlenmiştir. Bunun anlamı EEPROM üzerinde 16.21=346 sekizlidir.

Hasta kartı üzerinde:

338 sekizli, işletim sistemi
259 sekizli hasta verileri
346 sekizli önceden hesaplanan eğri noktaları
96 sekizli kriptografik anahtarlar

tarafından kullanılmaktadır. Toplam hesaplanan EEPROM alanı 1039 sekizlidir. Hasta kartındaki uygulama yazılımı ise 1960 sekizli yer tutmaktadır.

11.2.1 Kart üzerinde kriptografik anahtarların organizasyonu

Güvenlik protokolünde kullanılacak kriptografik anahtarlar her kullanıcı için bir açık gizli anahtar çifti, kartla haberleşen her uygulama için ise kar üzerinde simetrik şifrelemeyi sağlayacak bir simetrik gizli anahtardan oluşmaktadır. Kartla haberleşen bir istemci uygulama bir de kart yöneticisi olmak üzere iki tip uygulama bulunmaktadır. Bunun anlamı kart üzerinde iki adet üçlü DES anahtarının tutulmasıdır.

Kullanıcı açık anahtarı, 22 sekizli, gizli anahtarı, 21 sekizli, 3DES anahtarı 16 sekizli boyunda yer tutar. Sunucu ve hasta kartı arasında oluşturulacak oturum anahtarı için kullanılan ortak giz de bir kez kartın bilgi merkezinde basımı sırasında oluşturularak karta yüklenmektedir. Her defasında ortak gizi kart üzerinde oluşturmak için bu işlemin harcadığı zaman 7sn'dir. Bu zamandan kazanmak üzere sunucunun açık anahtarını (22 sekizli boyunda) akıllı karta yüklemek yerine 21 sekizli boyundaki ortak giz, bilgi merkezinde sunucu gizli anahtarı ve akıllı kartın açık anahtarı kullanılarak bir defa üretilir ve karta yüklenir. Karta işlem sırasında oluşturulan imza 42 sekizli, oturum anahtarları ise daha sonra 4 sekizlisi atılmak üzere 20 sekizli yer

tutmaktadır fakat bunlar çalışma süresi boyunca RAM'de tutulurlar. Sonuç olarak EEPROM'da tutulan anahtarlar için 21+21+22+ 2.16 = 96 sekizli yer bulunması gereklidir.

11.2.2. Hasta kartında veri organizasyonu

Hasta kartındaki uygulama yazılımı doktor kartına göre daha fazla işleve sahip olduğundan daha fazla yer işgal etmektedir. Bu nedenle gerçekleme aşamasında hasta kartında veri organizasyonu karşılaşılan önemli problemlerden birisi olmuştur. Hasta verilerinin organizasyonu yer kısıtı parametresine öncelilik verilerek yapılmıştır. Algoritmaların verimliliği ikinci aşamada gözetilmiştir. Bu nedenle her tip hasta verisi için ayrı bir organizasyon yapılarak bu verileri ucuca sıkıştırarak hasta kartında bir karakter katarı halinde tutan bir program yazılmıştır.

Kart üzerinde hasta bilgilerinin saklanışı mantıksal olarak aşağıdaki tablolarda gösterildiği gibidir. Fakat kart üzerinde bu biçimde bir tablolama yapısının bulunmadığı bir ortamda aşağıda görülen her tablo için ayrı bir karakter katarı oluşturulmaktadır. Bu katarın boyu ise değişkendir. Bu sayede EEPROM alanı minimum düzeyde işgal edilmektedir.

Tablo 11.1. Hastalık Tablosu (8 Elemanlı)

Alan	**Tip**	**Uzunluk(sekizli)**
Code	Integer	2
BegDate	Integer	2
EndDate	Integer	2
Doktor	Integer	2

Hastanın önemli hastalıkları hasta kartı üzerinde saklanmaktadır. Tablo 11.1.'de görüldüğü gibi hastalık bilgisi içinde hastalığın başlangıç tarihi, geçirilmiş bir hastalık ise bitiş tarihi ve hastalığa ilişkin doktor_no bilgisi tutulmaktadır. Bu kayıt yapısında 8 adet hastalık kaydı tutulabilmektedir. Tablo 11.1'de görülen doktor bilgisi, 0'a veya D_ID'ye eşittir. Eğer 0'a eşit ise o hastalık hassas değildir. Yani herhangi bir kişisel doktora bağlı değildir. Hastalık kayıtlarının boyu en fazla 64 sekizli olabilmektedir

Tablo 11.2 Doktor Tablosu (İki Elemanlı)

Alan	**Tip**	**Uzunluk(sekizli)**
DocId	Integer	2
PubKey	String	22

Hastanın kişisel doktorları için hasta kartında bir doktor tablosu tutulmaktadır. (Bakınız Tablo 11.2) Böylece hasta kartını okumak isteyen bir doktorun D_ID'sinin tabloda olup olmadığı kontrol edilerek doktorun kişisel doktorlar arasında olup olmadığına karar verilir. Tabloda bulunamazsa asıllamaya gerek duyulmaz. Asıllama işlemi için kullanılacak doktorun açık anahtar bilgisi de doktor tablosunda tutulur. EEPROM'daki yer kısıtından dolayı bir hastanın yalnızca iki adet kişisel doktoru olabilmektedir. Bu kaydın boyu en fazla 48 sekizli olabilmektedir.

Tablo 11.3 Kişi Tablosu (Bir Elemanlı)

Alan	**Tip**	**Uzunluk(sekizli)**
PatID	Long	4
FName	String	12
LName	String	12
POB	String	10
DOB	İnteger	2
Tel	String	10
Sex	Byte	1
Married	Byte	1
Blood	Byte	1
PName	String	12
Aname	String	24
ATel	String	10

Kişi tablosu (Tablo 11.3.) hastanın kimlik bilgilerinin yanında kan grubu, ve acil durumda aranacak kişi ve telefon numarasını da içermektedir. Toplam kayıt uzunluğu 99 sekizlidir. EEPROM genişliği 8K'dan daha büyük bir kart kullanıldığında tutulan kayıtların uzunluğu arttırılabilir.

Tablo 11.4 Alerji Tablosu (8 Elemanlı)

Alan	Tip	Uzunluk(sekizli)
Alerji	Integer	2

Hastanın alerji bilgileri ise ayrı bir tabloda tutulur (Tablo 11.4). Bu kaydın boyu en fala 16 sekizli olabilmektedir. Hastalık bilgisinde olduğu gibi başlangıç ve bitiş tarihi bulunmaz, alerji ortadan kalktığında hasta kartındaki kayıt da silinir.

Tablo 11.5 Arıza Tablosu (8 Elemanlı)

Alan	Tip	Uzunluk(sekizli)
Ariza	Integer	2

Hastanın arıza tablosu, (tablo 11.5) eksik organ veya bir sakatlık bilgisini içerir. Arıza kaydı en fazla 16 sekizli olabilmektedir.

Tablo 11.6 İlaç Tablosu (8 elemanlı)

Alan	Tip	Uzunluk(sekizli)
Ilac	Integer	2

İlaç tablosunda ise (tablo 11.6) hastanın sürekli kullandığı ilaçlar tutulmaktadır. İlaç kaydı da 0-16 sekizli arasında değişen uzunluktadır.

Hasta kayıtlarının EEPROM üzerinde toplam işgal ettikleri alan 99-259 sekizli arasında değişmektedir.

11.2.3.Seçilen Eliptik Eğri Parametreleri

EC-161.DEF içinde kullanıcı tarafından tanımlanan eliptik eğri parametreleri bulunur. Eliptik eğri parametreleri *a* (1 sekizli), *b* (21 sekizli), *r* (21 sekizli), *k* (1 sekizli), ve *G* (22 sekizli) olmak üzere toplam 66 sekizli yer tutar. Kullanılan eliptik eğriler $GF(2^m)$ (m=168) cismi üzerinde tanımlanabilir. Polinom tabanlı gösterilim şu şekildedir:

$$P(t) = t^{168} + t^{15} + t^3 + t^2 + 1$$

$GF(2^m)$ üzerinde tanımlanan eliptik eğri E'nin denklem yapısı aşağıdaki gibidir:

$$y^2 + xy = x^3 + ax^2 + b$$

11.3. İkili kart okuyucunun simülasyonu

Uygulamada çift kart girişli kart okuyucu bulunamaması nedeniyle ikili kart okuyucu simülasyon üzerinden gerçeklenmiştir. Gerçeklemede tasarımdan farklı olarak doktor oturumu bir kez açıldıktan sonra doktor kapatana kadar açık kalmaktadır. Kartın takılı olma koşulunun bu kısıttan dolayı doktor kartı için değil, hasta kartı için aranması tercih edilmiştir. Bir doktor kendi oturumunu açtıktan sonra doktor kartı bilgileri istemci makinaya alınarak doktor kartının simülasyonu sağlanmıştır.

11.4. Çok katmanlı uygulama yapısı

Uygulma yazılımının çok katmanlı veri tabanı yapısında olması tercih edilmiştir. Bir çok-katmanlı istemci/sunucu uygulaması, farklı makinalarda birbirine bağlı olarak çalışan mantıksal birimlere parçalanmıştır. Çok-katmanlı uygulamalar, aralarında bazı verileri ortaklaşa kullanırlar, internet veya bir yerel ağ üzerinden birbirleriyle haberleşirler. Çok-katmanlı veri tabanı modelinde ise veri tabanı mantıksal parçalara bölünmüştür. Sunucu uygulama (merkezi katman), birden fazla istemciden gelen istekleri ve güncelleme bildirimlerini işler ve koordine eder. Veri kümelerinin tanımlanmasının detaylarıyla ilgilenir ve uzak veritabanı sunucusu ile bağlantıya geçer. Bu çok katlı modelin olumlu yanları şu şekilde sıralanabilir:

- Değişik istemci uygulamalar, aynı merkezi katmana ulaşırlar, bu sayede veri tabanına erişim kuralları her istemci uygulama için tanımlanmak yerine tek bir merkezi katmanda tanımlanır.

- İstemci uygulamalar üzerindeki yük azaltılmış olur. İstemci uygulamalar veri tabanı bağlantı yazılımından muaftırlar. Hafifletilmiş istemci uygulamaları internet ortamında dağıtmak da kolaylaşır.

- Bir uygulamanın yükünü birden fazla makinaya dağıtmak, performansı artırabilir.

- Hassas fonksiyonlar değişik erişim kısıtlarına sahip olan katmanlara bölünerek izole edilebilirler. Bu yapı, dinamik ve değişken güvenlik katmanları yaratmaya olanak sağlar. Merkezi katmanlar, hassas veriye erişimin giriş noktalarını kısıtlayabilirler.

Tez çalışmasında bu yapının tercih edilmesinin en önemli nedeni güvenliktir. İstemci uygulamaların veri tabanı sunucusuna doğrudan erişiminin engellenmesi gerekmektedir. Güvenlik protokolünün fonksiyonları veri tabanı sunucusuna bırakılmaksızın ayrı bir sunucu uygulama üzerinden yürütülmektedir. Tez çalışmasında çok katmanlı yapının en basit formundan yararlanılmıştır. Üç katmanlı model adı verilen yapı aşağıdaki katmanlardan oluşmaktadır:

- **İstemci Uygulama**: Kullanıcı terminalleri üzerinde çalışır. Kullanıcı arabirimini sağlar. İstemci tarafındaki güvenlik fonksiyonlarını işletir ve kart okuyucu ile haberleşir.

- **Sunucu Uygulama**: Tüm istemci uygulamaların erişebileceği Bilgi Merkezi'nde bulunur. Veri tabanı sunucusunun bulunduğu makinanın üzerinde çalışmaktadır. Veri tabanı sunucusuna erişim için yürütülen kimlik doğrulama, asıllama, veri güvenliği için şifreleme ve şifre çözme gibi güvenlik fonksiyonlarını yürütür. Veri tabanına erişim servislerini yüklenir.

- **Veri Tabanı Sunucusu**: İlişkisel veri tabanı yönetim sistemini sağlar.

Çok katmanlı uygulamalar için Delphi 5'te MIDAS teknolojisi(Multi-tier Distributed Application Services Suite-Çok katmanlı Dağıtılmış Uygulama Servis Seti) kullanılmaktadır. Sunucu uygulama ve istemci uygulamaların birbirleriyle haberleşmeleri için soket haberleşmesi kullanılmıştır. Bunun için TCP/IP üzerinde soket bağlantısı ile sunucu program ile istemci programın haberleşmesini sağlayan MIDAS nesnelerinden yararlanılmıştır. Veri tabanı sunucusu olarak ise SQL Server 7.0 kullanılmıştır.

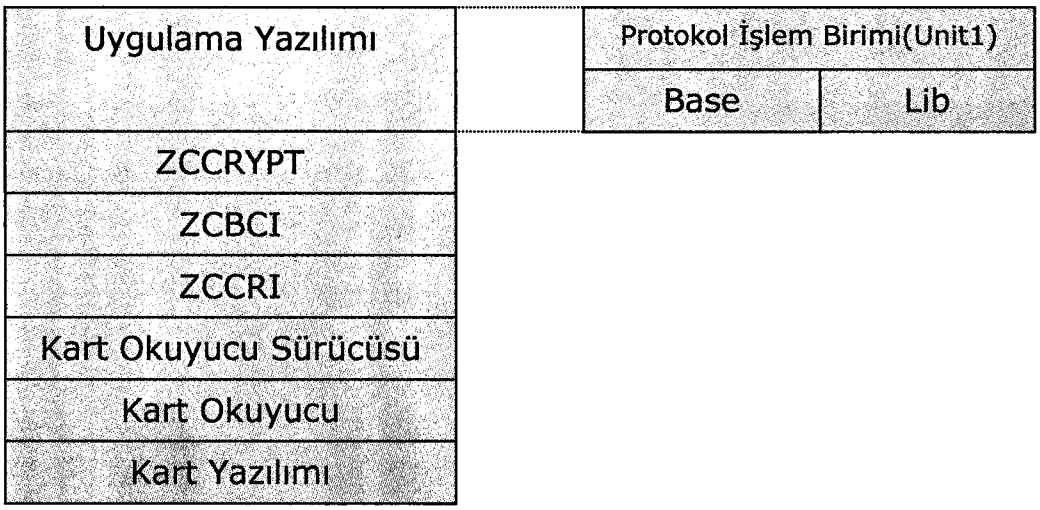
11.5. Uygulama geliştirme

Geliştirilen uygulama yazılımı, kart okuyucu üzerinden akıllı kartla ilişki kurabilmek için iki DLL kullanır: Zeit Control Basic Card Command Interface (ZCBCI.DLL) ve

Common Reader Interface (ZCCRI.DLL). ZCBCI, akıllı karta terminal tarafından gönderilen önceden tanımlı komutları içerir. ZCCRI ise bir kart okuyucuyla kurulan bağlantılarda başvurulan bir yapıdır. ZCCRI, bir kart okuyucuyu açmak, kapatmak, bir kartın okuyucuya yerleştirilip yerleştirilmediği konusunda beklemeye geçmek, karta bağlanmak, karta yeniden bağlanmak (reset) veya kartla bağlantıyı kesmek gibi fonksiyonları içerir. Bunların dışında ZCBCI tarafından çağrılmak üzere basit işlem fonksiyonları da desteklenir. Kullanıcı tarafından bu fonksiyonların çağrılmasına gerek duyulmaz.

ZCBCI, kartla ve okuyucu ile bağlantıya geçmek için ZCCRI'yı kullanır. Kart ve okuyucu ile bir bağlantı yaratmak için önce ZCCRI katmanından yararlanılır. Yaratılan kart nesnesine ilk önce eklenmesi gereken aygıtlardan birisidir. ZCBCI, dosya sistemi desteği de dahil olmak üzere tüm önceden tanımlı komutları içerir. Bunların üzerine akıllı kartta işlem yapabilmek için komut tanımlanmak istendiğinde küçük bir işlem fonksiyonu yardımıyla bu komutlar işletilebilir.

Kart okuyucu sürücü katmanı, okuyucu ile terminal arasıdaki bağlantıyı sağlar. Okuyucunun sistem tarafından tanınmasını sağlar. Desteklenen her kart okuyucu için bu katmanda bir DLL bulunur. Uygulama geliştirirken bu kütüphanelerin doğrudan kullanımı sözkonusu değildir. ZCCRYPT'de ise uygulama yazılımı tarafından kullanılabilecek çeşitli kriptografik fonksiyonlar bulunur.



Şekil 11.1 Yazılım katmanlarının yapısı

Gerçeklenen uygulamadaki yazılım katmanları Şekil 11.1'de gösterilmektedir. Uygulama yazılımı şekilde görüldüğü gibi üç bölümden oluşur. Bunlar kullanıcı önyüzü, Base ve Lib olarak adlandırılan parçalardır.

11.5.1 Uygulama yazılımı

Uygulama yazılımı, üç bölüme ayrılarak geliştirilmiştir. Öncelikle akıllı kart ve kart okuyucu ile kurulan tüm bağlantılar Base ünitesinde toplanmıştır. Bu nedenle base ünitesinin her fonksiyonunda ZCCRI.DLL fonksiyonları kullanılır. Lib ünitesinde bazı kriptografik ilkeller ve temel olarak diğer fonksiyonlar tarafından çağrılan yardımcı fonksiyonlar bulunmaktadır. Protokol İşlem Birimi ise güvenlik protokolü işlemlerini yürütmek için kullanılan fonksiyonları içerir. Aşağıda bu üç ünitede kullanılan fonksiyon ve prosedürler tanıtılmıştır. Program akış diyagramları ise EK-A'da görülebilir.

11.5.1.1 Base ünitesi

Bu ünite uygulama yazılımı içinde kartla bağlantıya geçen tüm alt programların toplandığı bölümdür. Kartla bağlantıya geçmek, bağlantıyı koparmak, karta PIN numarasını onaylatmak, kartın PIN numarasını değiştirmek, kartı formatlamak, karttan açık anahtarını okumak, karttan asıllama verisi veya kullanıcının imzasını almak, karttan oturum anahtarı oluşturmasını istemek, kart üzerindeki verileri güncellemek, karttaki verileri okumak, doktor-hasta arasındaki asıllama işleminin hasta kartı tarafında yapılmasını sağlamak amaçlarıyla base ünitesine başvurulur.

Bu ünitedeki yordamlar kart iletişimini kolaylaştırmak ve kullanıcı arayüzünü kart iletişim detaylarından yalıtmak üzere tasarlanmıştır. Tüm yordamlar hata kontrolü yapar ve işlemin başarılı olması durumunda sıfır hata durumunda ise hata kodu döndürür. ConnectCard ve DisConnectCard dışındaki tüm yordamlar kart üzerindeki bir yordamla örtüşür yani kart üzerindeki bu yordamı tetikler ve sonuçlarını geri döndürür. Komut-yanıt protokolünün bir parçası olarak yürütülür. Tüm yordamlarda geçirilen ilk parametre olan Card nesnesi kart iletişimi sırasında kullanılan kaynakları ve kart statüsünü tutar. Aşağıda kısaca işlevleri üzerinden tanıtılacak yordamların akıllı kart tarafındaki karşılıklarının tanıtıldığı kart yazılımı, Bölüm 11.5.2'de giriş ve çıkış parametreleri ile açıklanacaktır.

- **ConnectCard (Card : TCard ; Key : Integer) : Integer;** verilen no ile tanımlı anahtarı kullanarak kart oturumunu başlatır.

- **DisConnectCard (Card : TCard) : Integer;** kart oturumunu sonlandırır ve hata kontrolü yapar.

- **VerifyPIN (Card : TCard ; TestPin : String) : Integer;** verilen PIN numarasını kart üzerinde kontrol eder.

- **ChangePIN (Card : TCard ; NewPIN : String) : Integer;** kart üzerindeki PIN numarasını değiştirir. Bu yordam ancak Kart üretici anahtarı ile bağlanılması halinde çağrılabilir.

- **FormatCard (Card : TCard ; Seed : String ; PubKey : String) : dWord;** kartı ilk kullanıma hazırlar. Kart gizli/açık anahtar çifti oluşturulur, sunucu açık anahtarı karta kaydedilir ve sunucu ile kart arasındaki ortak giz hesaplanır.

- **GetCardPublicKey (Card : TCard ; var CardPubKey : String) : dWord;** FormatCard yordamı ile oluşturulan kart açık anahtarı istemciye alınır. Bu yordam Kart üretim sistemi tarafından kullanılır ve elde edilen kart açık anahtarı ilerde sunucu-kart arasında şifreli iletişim sağlanabilmesi için veritabanına kaydedilir.

- **GetSign (Card : TCard ; var Sign : String) : dWord;** verilen tarih saat bilgisinin kart numarası da kart tarafından eklenmek üzere kart tarafından imzalanır. Bu imza istemci tarafından kart sahibinin istemci noktasında bulunduğunu ispatlamak üzere kullanılır.

- **SetSessionKey (Card : TCard ; var Stamp : String) : dWord;** istemci sunucu-kart arasındaki şifreli iletişime şifreli veri paketlerini aktararak katılır. Şifreli oturumu kart tarafında başlatmak için kart üzerinde sunucu-kart oturum anahtarının hesaplanmasını bu komutla tetikler.

- **PutData (Card : TCard ; Data : String) : dWord;** sunucu-kart oturumunda sunucudan gelen şifreli veri paketlerini karta aktarmak için kullanılır.

- **GetData (Card : TCard ; z : Byte ; var d : String) : dWord;** kart üzerindeki bilgilerin alınması için kullanılır. Gelen bilgi ekranda gösterilir. Gelen bilgi herkese açık bilgi yanında kişisel doktora özel bilgileri içerir.

- **AuthenticateDoctor (Card : TCard ; z : Integer ; KDP : String ; Data : String) : dWord;** Data'yı parçalayarak D_ID'yi doktor tablosundaki ID'lerle karşılaştırır. Tablo'da D_ID ile karşılaşırsa doktor kartının hasta kartı tarafından asıllanması işlemini yürütür. Bunun için istemcinin ürettiği KDP'yi ve doktor açık anahtarını kullanarak bir oturum anahtarı üretir ve gelen asıllama verisini onaylar. Kart verilen imzayı tanır ve doktoru onaylarsa oturum boyunca (yani kart çıkana dek) doktor kendine özel bilgiye erişebilir.

11.5.1.2. Lib ünitesi

Bu ünitede genel amaçlı destek ve çevirme yordamları bulunmaktadır.

Destek alt programları:

- **StringToHex(Val : String) : String ;** uygulama geliştirme aşamalarında kart iletişiminde gelip giden verinin kolay takibi ve test amaçlı kullanılmıştır. Verilen karakter katarı içerisindeki her bir karakteri onaltılık tabanda gösteren bir dizi döndürür.

- **crc (d : String) : char ;** şifreli verinin paketlerinin açıldığında anlamlı olduğunun doğrulanması için kullanılan basit darveya tabanlı 1 sekizli uzunluğunda çıkış üreten fonksiyondur.

- **MakeStamp (dt : TDateTime) : String ;** gün-saat verisinde saat parçasını beş dakika hassasiyetine düşürür.

Çevirme alt programları:

Bu yordamlar sayısal veri tiplerini katarlar üzerinde taşımak için sayısal veri tiplerinin bellek görüntülerini bellek kopyalama yöntemi ile harf katarına dönüştürür yada bellek görüntüsü harf katarından sayısal veri tipine dönüştürür (Tablo 11.7).

Tablo 11.7. Çevirme yordamları

SmallToBin	(val : SmallInt) :	String ;
BinToSmall	(str : String) :	SmallInt ;
LongToBin	(val : Integer) :	String ;
BinToLong	(str : String) :	Integer ;
DwordToBin	(val : DWord) :	String ;
BinToDWord	(str : String) :	DWord ;
DateToBin	(val : TDateTime) :	String ;
BinToDate	(str : String) :	TDateTime ;
StringToBin	(val : String ;	String ;
	len : Integer):	

11.5.1.3. Protokol İşlem Birimi

Protokol İşlem Birimi, güvenlik protokolünde sunucuyla haberleşme, şifreleme, şifre çözme ve imza üretme, asıllama işlemlerini yürütür. Akıllı kartla bağlantı kurularak yürütülen işlemler Base ünitesi üzerinden komut yanıt protokolü ile gerçekleştirilir. Güvenlik fonksiyonlarının bazıları lib ünitesindeki yardımcı fonksiyonları kullanır. Kullanıcı önyüzüne ilişkin fonksiyonlar aşağıda tanıtılmıştır. Bu fonksiyonlar, sistemi oluşturan tüm taraflarca kullanılmaktadır. Bunlar sunucu uygulama, akıllı kart yönetimi programı ve istemci uygulamadır.

- **Timer:** Timer1.Timer olay prosedürü, bir Delphi zamanlayıcı nesnesi olan Timer1 tetiklendiğinde (500 ms'de bir) çalıştırılır ve kart okuyucuda bir kart olup olmadığını kontrol eder. Kartın okuyucuya takılı olup olmaması durumuna göre **Cardin** fonksiyonunu true veya false parametresi ile çağırır. Kartın takılı olup olmadığını anlamak için Zeit kütüphanesindeki ZCCriWaitCard fonksiyonunu çağırır. İstemci uygulama ve kart yönetimi programı tarafından kullanılır. Timer prosedürünün akış diyagramı Şekil A.1'de görülmektedir.

- **Cardin:** Timer1.Timer olay prosedürü tarafından çağrılır. Verilen "true" ya da "false" parametresine göre ekranda kart durumunu (kart var/kart yok) tazeler. İstemci uygulama ve kart yönetim programı tarafından kullanılır.

- **Create Session Key:** Oturum anahtarı üretecek taraflardan herhangi birisi çağırabilir. Kendi gizli anahtarı, karşı tarafın açık anahtarı ve KDP giriş parametresi olarak alınarak oturum anahtarı fonksiyonun çıkışında üretilir. CryptEC161SessionKey kütüphane fonksiyonunu çağırarak girişlerini besler. Bu fonksiyonun oturum anahtarını saklayabilmesi için 20 sekizliden oluşan bir string değişkenine yer ayırır. Geri dönen sonucun ise ilk 16 sekizlik bölümü ayrılır. Bunun nedeni ileride kullanılacak 3DES simetrik şifreleme fonksiyonunun 16 sekizli bir oturum anahtarı istemesidir.

- **Encrypt:** Şifrelenecek metni ve anahtar değerini alarak ZCCryptDesEncryptOnce kütüphane fonksiyonunun girişlerini besler. Böylece verilen metnin tümünün şifrelenmesi sağlanır. Bu fonksiyon akıllı kart üzerinde şifreleme için çağrılmaz, istemci ve sunucu tarafında kullanılır. Şifrelenecek metin açıldığında anlamlı bir paket olarak algılanabilsin diye önüne onun hasta bilgileri içerisinde hangi parça olduğunu belirten veri kodu ve verinin çırpma değeri ve uzunluk bilgisi eklenir. Şifrelenecek veri paketi oluşturulurken uzunluğu 64 bit ve katları olacak şeklinde boşluk doldurulur.

- **Decrypt:** Şifreli metin ve anahtarı giriş değeri olarak alır ve 3DES şifre çözme işlemi için ZCCryptDesDecryptOnce kütüphane fonksiyonunun girişlerini besler. Bu fonksiyonun çıkışında elde edilen metni, kod bilgisi, çırpma değeri, uzunluk bilgisi ve verinin kendisi olarak anlamlı parçalara ayırır. Verinin çırpma değerini hesaplayarak ZCCryptDesDecryptOnce şifre çözme fonksiyonundan geri dönen değerle karşılaştırır. Sonuç farklı ise şifreli metnin doğru taraftan gelmediği sonucuna varılır.

- **Create Server Keys:** Yalnızca sistemin kurulumu sırasında bir defalığına çağrılan bir fonksiyondur. Sunucunun anahtar çifti üretilir. Bunun için EC161 kütüphanesindeki CreateKeyPair fonksiyonu çağrılır.

- **Create Server Session Key:** Sunucu için oturum anahtarı üretmek amacıyla çağrılır. CreateSessionKey fonksiyonunu çağırarak giriş değerlerini aktarır. Çıkışta hata kontrolü için yazılmıştır. Sunucunun gizli anahtar bilgisini, kartın açık anahtarını ve KDP değerini alarak bir oturum anahtarı üretilir.

- **Create Doctor Keys:** Kart yönetici programı tarafından kullanılır. Doktorun gizli ve açık anahtarının oluşturulması işini yapar. Zccrypt kütüphane fonksiyonlarından CreateKeyPair fonksiyonunu çağırır. Hata kontrolü ve test amacıyla kullanılır.

- **CreateDoctorSign:** Doktor kartının simülasyonunda doktorun imzasını oluşturur. Doktor kartı tarafında hasta-doktor oturum anahtarını oluşturduktan sonra bu anahtarla D_ID'yi ve zaman pulu oluşturarak sonucu şifreler. Şifreleme işleminden önce 4 sekizliden oluşan D_ID, 8 sekizliye tamamlanır. Şifreleme için ZccryptOnce kütüphane fonksiyonunu çağırır.

- **Initialize Card:** Bir akıllı kartın kimlik bilgileri yüklenmeden önceki kriptografik fonksiyonlar açısından hazırlanması işini yapan bir nesnedir. Kartla bağlantıya geçmeyi ve kartta kriptografik işlemlere hazırlık olarak rasgele sayı üretecinin çalışmasını başlatır. Bu nesne birçok alt programa çağrı gönderir. Bunlar ConnectCard, ZCCryptRandomSeed, ZCCryptRandom, FormatCard ve DisconnectCard'tır. FormatCard alt programına, sunucunun açık anahtarı ve bir rastgele değer gönderilerek hangi kartla bağlantıya geçileceği bilgisi verilir. Bu altprogramlar Base ünitesinde tanıtılmıştır.

- **GetCardPublicKey:** Akıllı kart yönetim programı tarafından çağrılır. Kartın açık anahtarını karttan okuyan alt programdır. Önce karta bağlanmak için Base ünitesindeki ConnectCard alt programına çağrı gönderilir. Ardından yine Base ünitesindeki GetCardPublicKey al programına çağrı gönderilerek kartın (açık,gizli) anahtar çiftlerinden açık anahtarı okunur. Kartla bağlantıyı kapatmak için yine base ünitesindeki DisConnectCard alt programa çağrı gönderilir.

- **Create Card Session Key:** Kart-sunucu iletişimine ön hazırlık olarak kart üzerinde kart-sunucu arasındaki oturum anahtarı oluşturulmasını sağlar. Bu amaçla Base ünitesindeki karşılık gelen SetSessionKey fonksiyonu yoluyla kart üzerindeki SetSessionKey komutunu tetikler. SetSessionKey fonksiyonuna bir anahtar türetme parametresi, KDP oluşturarak gönderir.

- **Put Data Ex:** Karta bilgi yazmak amacıyla sunucu program veya kart yöneticisi tarafından kullanılır. Kod bilgisi ile birlikte bir veri parçası alarak kod+uzunluk+veri şeklinde paketler. Paketleme işlemi sırasında bilgi 64

sekizliye tamamlanır. Paketi ZCCryptDesEncryptOnce fonksiyonunu çağırarak şifreler. Şifrelenmiş paketi Base ünitesindeki PutData fonksiyonunu kullanarak karta aktarır. Paketlenen bilgi hasta kaydını oluşturan parçalardan (Hastalık, Alerji, Arıza vb) biridir. Şifreleme sunucu – kart oturum anahtarıyla (3DES) yapıldığından kart bu veriyi açarak kendi üzerinde saklayabilecektir.

- **Put Datum:** Bu fonksiyon hasta bilgisi parçalarını şifreleyerek yazabilen PutDataEx fonksiyonunu her bir parça için çağırarak tüm hasta kaydını karta geçirir.

- **Get Datum:** Bu fonksiyon tüm hasta bilgisi parçaları için Base ünitesindeki GetData 'yı çağırır ve karttan hasta bilgileri bir bütün olarak alınmış olur ve ekranda görüntüler.

- **Get Card Sign:** Base ünitesindeki GetSign prosedürüne tarih ve saat girişi yaparak akıllı kartın oluşturacağı imzayı alır. Saat bilgisi Makestamp fonksiyonu çağrılarak üretilir.

- **VerifyPatientSign:** Giriş değeri olarak hastanın tanımlayıcı kodunu (H_ID), hastanın açık anahtarını(H_AA) ve imzayı alır. Çıkış olarak imzanın geçerli olup olmadığına dair boolean bir sonuç üretir. MakeStamp fonksiyonunu çağırarak zaman pulu üretir. O anki zamanı kullanarak zaman pulu üretir ve imzanın geçerliliğini kontrol eder. Olumsuz sonuç alındığı durumda kullanılan zaman değerinden beş dakika çıkartılarak yeniden bir kontrol yapılır. Böylece 10 dakikalık bir zaman çerçevesinde imzanın doğrulaması sağlanmış olur. İmzanın geçerliliğini kontrol için ZC kütüphane fonksiyonu olan ZCCryptEC161Verify çağrılır. İki kez kontrol yapılarak 10 dakikalık bir süre elde edilmesinin nedeni MakeStamp fonksiyonunun 5 dakikalık bir hassasiyetle çalışmasıdır.

- **VerifyDoktorSign:** VerifyPatientSign prosedürü ile aynı işi yapar. Ayrı olarak yazılmasının nedeni doktorun tanımlayıcı kodunun uzunluğunun hastanınkinden iki sekizli daha kısa olmasıdır. Akıllı kart üzerindeki yer sıkıntısı dolayısıyla böyle bir farklılık yaratılmıştır.

- **Decode:** İstemciden gelen asıllama paketinin sunucu tarafında parçalanması işlemini yürütür. Asıllama verisini doktor ve hastanın imzaları ve ID bilgileri olmak üzere dört parçaya ayırır.

- **Encode:** Bu prosedür istemci programında kullanılır. Doktor ve hasta kartlarından gelen imza bilgileri ve doktor ile hastanın ID bilgilerini birleştirerek asıllama verisini oluşturur.

- **EncodeHastalik:** Veri tabanını sorgulayarak hastaya ait hastalık bilgilerini veri tabanından çeker ve gelen bilgileri bir string üzerinde toplayarak geri döndürür. Prosedürün giriş parametresi hastanın tanımlayıcı kodu (H_ID) ve veri tabanının bulunduğu dizini belirten adres bilgisidir; çıkış parametresi ise bir string değişkenidir.

- **EncodeAriza, EncodeIlac, EncodeKisi1, EncodeKisi2:** Sunucuda Hastanın tüm arıza, ilaç, ve kişisel bilgilerini ardışık olarak bir karakter katarına ekliyor. Tüm bu prosedürler Encodehastalik ile aynı mantıkta çalışır. Ayrı ayrı yazılmalarının nedeni, farklı türde farklı içerikteki veriler üzerinde işlem yapmalarıdır. Akıllı kart üzerinde alan sıkıntısı olduğundan her veri türü için ayrı organizasyon yapılır.

- **EncodeDoktor:** Doktor verilerini sunucuda veri tabanından çekerek karta yazmak üzere bir stringe ekler. Kart yönetici programı dışında kullanılmaz.

- **EncodeData:** Parçalar halinde oluşturulan hasta bilgisi parçaları tek tek şifrelenip tek blok halinde bir çıkış üretilir. Yukarıda tanıtılan EncodeHastalık, EncodeAriza,... prosedürlerini çağırarak bu prosedürlerin geriye döndürdüğü veri paketlerini oturum anahtarı ile şifreler. Giriş parametresi olarak H_ID, veri tabanı adresi ve anahtarı alır, çıkış olarak tek bir veri bloğu üretilir.

- **ExtractData:** Sunucudan gelen blok halindeki hasta bilgisi içinden istenen koda ilişkin parçayı çıkartarak geriye döndürür.

- **DecodeHastalik, DecodeAlerji, DecodeAriza, DecodeIlac, DecodeKisi1, DecodeKisi2:** Hastalık, alerji, arıza gibi hasta bilgileri bloklarını alanlarına

parçalar ve her blok içindeki alanları kullanılabilir hale getirir. Giriş olarak ilgili hasta bilgisi paketi alınır ve bir string listesi olarak geri döndürülür.

- **DecodeDoktor:** Doktor bilgilerinden oluşan veri bloğunu, kendi içinde alanlara göre parçalayarak bir karakter katarı listesi haline getirir.

- **DecodeData:** Yukarıda tanımlanan decode fonksiyonlarına veri bloğunun kodunu giriş parametresi olarak alıp, dallanmayı sağlar. Örneğin 1 numaralı kod hastalık bilgilerinin kodu olduğundan DecodeHastalik prosedürünü çağırır ve hastalıklardan oluşan veri bloğunun bir karakter katarı listesine parçalanması işlemi yürütülür.

11.5.2 Hasta Kartı Yazılımı

Kart yazılımındaki fonksiyonlar base ünitesindeki komutlara karşılık gelen yanıtların tanımlanmasıdır. Yazılan her komut için bir komut kodu tanımlanır. Kart üzerinde işletilen komutlar, parametreleri ile birlikte aşağıdaki verilmiştir.

- **Decrypt**

Amaç : Blok zincirleme yaparak 8 sekizliden daha uzun blokların açılmasını sağlar
Girdi : data : açılacak veri
key : anahtar
Çıktı : data : açık veri

İşlem : Gelen veri sekizlik bloklar halinde DES altyordamı ile açılır ve blok zincirleme yöntemi ile de daha uzun bloklar açılır.

- **CRC**

Amaç : Gelen verinin anlamlı olduğunun doğrulanması için kullanılan basit darveya tabanlı CRC hesaplanır.
Girdi : data : doğrulanacak veri bloğu
Çıktı : Bir sekizli boyunda toplam darveya değeri.
İşlem : Gelen data bloğunun tüm elemanları sıfır değerinden başlanarak ardışık olarak darveyalanır.

- **FormatCard**

Girdi : Kaynak : Rasgele kaynak

SunucuAcikAnahtar : Sunucu Açik Anahtarı

Çıktı : Yok

İşlem : 1. Açık/Gizli Anahtar cifti üret

2. Sunucu Açık Anahtarını kullanarak Ortak Giz üret

Alt Yordamlar : EC161GenerateKeyPair : BasicCard EE Kütüphane yordamı

EC161SharedSecret : BasicCard EE Kütüphane yordamı

- **GetCardPublicKey**

Girdi : Yok

Çıktı : KartAcikAnahtar : Kart Açık Anahtarı

İşlem : Kart Açık Anahtarını döndür

- **SetSessionKey**

Girdi : KDP : Eliptik Egri Anahtar Türetme Parametreleri

Çıktı : Yok

İşlem : kart-sunucu oturum anahtarını oluşturur ve Key 3 olarak kaydeder

Alt yordamlar: EC161SessionKey : BasicCard EE Kütüphane yordamı

- **GetSign**

Girdi : AsıllamaVerisi : Terminal tarafından sağlanan [Tarih + Saat]

Çıktı: AsıllamaVerisi : Kart Asıllama Verisi

İşlem: imzala [Terminal tarafından sağlanan bolum + Hasta No]

Alt yordamlar: EC161Sign : BasicCard EE Kütüphane yordamı

- **PutData**

Girdi : d : Karta yazılacak bilgi paketi

Çıktı : Yok

İşlem : 1. Bilgi paketini parçala [verikodu , Uzunluk , Bilgi]

2. Bilgi doğruluğunu kontrol et

3. Bilgiyi ilgili yazma yordamına aktar

Alt yordamlar :

PutHastalik : Hastalık Bilgisi Yaz

PutAlerji : Alerji Bilgisi Yaz

PutAriza : Arıza Bilgisi Yaz

PutIlac : İlaç Bilgisi Yaz

PutKisisel1 : Kişisel Bilgi Yaz - Bolum 1 Bellek probleminde dolayi

PutKisisel2 : Kişisel Bilgi Yaz - Bolum 2

PutDoktor : Doktor Bilgisi Yaz

- **GetData**

Girdi : z : İstenen Bilgi Kodu

Çıktı : d : İstenen Bilgi Paketi

İşlem : 1. İstenen bilgi koduna karşılık gelen rutini çağır

2. Bilgi paketini döndür

Alt yordamlar :

GetHastalik : Hastalık Bilgisi Oku

GetAlerji : Alerji Bilgisi Oku

GetAriza : Arıza Bilgisi Oku

GetIlac : İlaç Bilgisi Oku

GetKisisel1 : Kişisel Bilgi Oku - Bolum 1 - Bellek probleminden dolayı

GetKisisel2 : Kişisel Bilgi Oku - Bolum 1 - iki parcaya bolundu

GetDoktor : Doktor Bilgisi Oku

- **AuthenticateDoctor**

Girdi : z : D_ID

KDP : EE Anahtar Türetme Parametreleri

d : Şifreli Doktor No - Doktor asıllama bilgisi

Çıktı : Yok

İşlem : 1. Verilen D_ID için tanımlı doktorları tara, doktor tabloda yoksa yordamdan çık

2. D_ID doktor tablosunda varsa D_AA'yı tablodan al ve doktor-hasta oturum anahtarı üret

3. Oturum anahtarı ile doktor asıllama bilgisini aç ve doktorun kimliğini doğrula.

11.6. Kullanıcı arayüzü

İstemci makina ve bilgi merkezi için hazırlanan kullanıcı arayüzlerinin görünümleri EK B'de verilmiştir. Şekil B.1'de görülmekte olan Sunucu Paneli, sunucu üzerinde çalışır ve hata ayıklama amacıyla kullanılır. İstemci tarafı kullanıcı arayüzü şekil B.2'de görülmektedir. Hasta Kartı takıldığında alan detayları karttan alınan bilgilerle doldurulur. Kart çıktığınında bilgiler ekrandan ve bellekten silinir. Şekil B.3'te doktorun sisteme girişi ve oturumu başlatmasını sağlayan doktor PIN giriş ekranı görülmektedir. Şekil B.4'te Bilgi Merkezi tarafından yönetici programın hastaların listesi gösteren bir ekran çıktısı görülebilir. Şekil B.5-B.11 arası kart yönetici programında seçilen hastaya ilişkin detayları gösteren ekranlar görülmektedir. Şekil B.12'de ise kart yönetimi programından doktorların listesi veren ekran görülmektedir.

12. SONUÇLAR VE TARTIŞMA

Tez çalışmasında bir sağlık sistemi üzerinde güvenlik protokolü tasarlanmış ve gerçeklenmiştir. Tasarım aşamasında sunucu uygulamanın doktor ve hasta kartlarını asıllaması üzerinden yapılan ikili asıllama kurgusu gerçekleme aşamasında ikili kart okuyucunun simülasyonu ile yerine getirilmiştir. Ele alınan pilot sağlık sistemi, incelenen örneklerden farklı olarak sağlık hizmetinin ücretsiz olarak yürütüldüğü merkezi olarak planlanan bir sağlık sistemini temel alır. Para veya sigorta bilgileri işlenmemektedir. Tez çalışmasındaki bilgi merkezi yerel bir sunucu uygulamayı içeren dağıtılmış bir veri tabanının parçası olarak düşünülebilir. Bu şekilde yerel veri tabanındaki hasta verilerinin tüm ülke çapındaki merkezi organizasyonu ayrı bir çalışma olarak incelenebilir.

Doktorun karta yazımı işlemi sırasında akıllı kartın internet üzerinden güvenlikli haberleşmeyle veri bütünlüğü korunarak sunucuyu asıllaması sağlanmıştır. Protokolün bu adımında sunucu güvenilir üçüncü taraf olarak bir işlev de görmüştür.

İstemci uygulama, sistem açısından yetkileri en kısıtlı olan taraftır. Sistem üzerinde verilere erişim ve güncelleme ile ilgili işlem yapma yetkisi yoktur. Yetki akıllı kartlar tarafından alındığında istemci uygulama veri paketlerini iletme, parçalama, karta yazılabilecek hale getirme veya karttan asıllama verilerini alarak sunucuya gönderme gibi yetki gerektirmeyen yükleri üstlenmektedir. İstemci uygulama sistemde aracı taraf olarak çalışmaktadır. Ekranda verileri gösterme veya klavyeden doktorun hastaya ilişkin bilgileri girmesi işlemleri kart okuyucunun ekran ve klavye donanımı olmadığından istemci tarafında gerçekleştirilir. Bu açık güvenilir bir işletim sistemi ile çözülebilir.

Güvenlik protokolünün bir başka önemli adımı da bir akıllı kartın diğerini asıllayarak, kendi üzerindeki bilgilerin okunmasına izin vermesidir. Yine veriler istemci uygulama üzerinden görüntülenir fakat yetki kullanıcı kartlarındadır. Genellikle akıllı kart kullanarak yapılan güvenlik protokolü tasarımlarında imza

üretme işlemi akıllı kartta mümkün iken imza doğrulama işlemi akıllı kart üzerinde yürütülememektedir. Bunun nedeni açık anahtar şifrelemenin zaman ve bellek kısıtları açısından akıllı kartlar üzerindeki problemlerinin henüz yeterince çözümlenememiş olmasıdır. Açık anahtar şifreleme ile imza doğrulama işlemi sunucu tarafında yapılabilirken kart tarafında imza doğrulama için şifre çözme algoritması olarak simetrik anahtar kriptografiden yararlanılmıştır.

Karta ek olarak yüklenen açık anahtar şifreleme mekanizmaları ile kullanıcı yetkilerinin sınanması, varlık asıllama ve veri kaynağı asıllama işlemleri güvenlikli bir biçimde yürütülebilmektedir. Melez kriptosistemler kullanılarak anahtar gönderimi olmaksızın ortak oturum anahtarı hesaplanabilmekte, sistemde kullanılan asimetrik gizli anahtarlar hiçbir biçimde kart dışında herhangi bir yerde kayıtlı bulunmamaktadır. Simetrik anahtar şifreleme anahtarları ise her oturumda yeniden üretilmekte; melez kriptosistemlerin kullanımı sayesinde anahtar dağılımı ve güvenliği problemi çözülmektedir.

Karta yüklenen açık anahtar şifreleme sistemi Eliptik Eğri kriptosistemlerdir. Eliptik Eğri Kriptografide kullanılan anahtarlar diğer açık anahtar şifrelerine oranla daha az yer kapladığından akıllı kartlarda eliptik eğri kullanımı bugün oldukça yaygındır. Eliptik Eğri kriptosisteminde 161 bitlik bir anahtarın güvenliği RSA şifreleme sisteminde 1024 bitlik bir anahtarın güvenliğine denktir. Ne var ki akıllı kart üzerinde eliptik eğri kriptosistemin gerçeklenmesi performansının artırılması açısından henüz üzerinde çalışılmakta olan bir konudur. Bu konu tez kapsamının dışında olduğundan hazır geliştirilmiş algoritmalar kullanılmıştır.

İstemci tarafta hasta kartında veri güncelleme isteğinde bulunulduğunda sunucu taraftaki verilerin güncellenmesinin ardından hasta kartındaki veriler güncellenmektedir. Hasta kartına veri yazımı sırasında elektrik kesintisi gibi fiziksel bir problem ortaya çıktığında oluşacak veri tutarsızlığı problemi için ek mekanizmalar geliştirilebilir. Doktorun tüm işlemleri baştan başlayarak yeniden yapması yerine farklı bir biçimde problemin çözümü sağlanabilir. Bir diğer geliştirilmesi gereken konu ise yetki transferidir. Hastanın kişisel doktoru, hasta bilgi merkezine giderek bir değişiklik talep etmedikçe değişmemektedir. Oysa kişisel doktor hastanın kontrolü dahilinde bir başka doktora yetki transferi yapabilmelidir.

KAYNAKLAR

[1] **Dreifus, H. ve Monk, J.T.**, *1997.* Smart Cards, Wiley Computer Publishing, 29-46

[2] **ISO/IEC 7816-3**, *Electronic Signals and Transmission Protocols,* International Standards Organisation.

[3] **ISO/IEC 7816-4**, *Interindustry commands for interchange,* International Standards Organisation.

[4] http://www.multos.com/

[5] http://java.sun.com/products/javacard/

[6] **Amigoni, M., Luzzi, L.**, The new Health and Social Information System of Region Lombardia The Pilot Project of Lecco, *Proc. of CARTES 99*

[7] **Maloney, Daniel L.**, Card Technology in Healthcare, *Proc. of CARTES 99*

[8] http://www.cp8.bull.net/sct/uk/sante/page/ c_sesame.html

[9] **Project Office Telemedicine**, 2000. Health Care Professionals' Protocol, *From a lecture at Cards Australia 2000,* Melbourne, http://www.hcpprotokoll.de/ smartcardseng.htm

[10] http://www.sesam-vitale.fr/html/projets/ netlink/netlk_pres.htm

[11] **Maloney, Daniel L.**, May 16 2001. Card Technology in Healthcare, *CardTech/SecureTech 2001.*

[12] **Diffie, W. and Hellman, M.E.**, Nov 1976. New Directions on Cryptography, IEEE Transactions on Information Theory,v.IT-22, n.6, pp.644-654.

[13] **Lenstra, H.W.**, 1987. Factoring Integers With Elliptic Curves, Annals of Mathematics, Volume 126, pp. 649-673.

[14] **Morrison, M.A. and Brillhart, J.,** 1975. A method of factoring and the factorization of F_7, *Mathematics of Computation*, volume 29, 183-205.

[15] **Pomerance, C.,** 1985. The guadratic sieve factoring algorithm, *Advances in Cryptology, EUROCRYPT"84, Lecture Notes in Computer Science,* volume 209, Springer-Verlag, 169-182.

[16] **Schneier, B.,** 1996. Applied Cryptography, Second Edition, Protocols, Algorithms and Source Code in C, John Wiley & Sons, Inc.469.

[17] **Willams, H.C.,** Nov 1980. A Modification of the RSA Public-Key Encryption Procedure, *IEEE Transactions on Information Theory*, v. IT-26, n 6, 726-729.

[18] **FIPS 186**, 1993. Digital Signature Standard, National Institute for Standards and Tecnology, http://csrc.ncsl.nist.gov/fips/

[19] **ElGamal, T.**, 1985. A public key cryptosystem and a signature scheme based on discrete logarithms, *IEEE Transactions on Information Theory,* vol 31, 469-472.

[20] **Schnorr, C.P.**, 1991. Efficient signature generation by smart cards, *Journal of Cryptology*, vol 4, 161-174.

[21] **Nyberg, K. and Rueppel, R.**, 1996, Message recovery for signature schemes based on the discret logarithm problem, *Designs, Codes and Cryptography*, Vol 7, 61-81.

[22] **Gordon, D.**, 1993. Discrete logarithms in GF (p) using the number field sieve, *SIAM Journal on Discrete Mathematics*, vol 6, 124-138.

[23] **Koblitz, N.,** 1987. Elliptic curve cryptosystems, *Mathematics of Computation*, Vol 48, 203-209.

[24] **Miller, V.,** 1986. Uses of elliptic curves on cryptograpy, *Advances in Cryptology,* CRYPTO'85, Lecture notes in Computer Science, vol 218, Springer-Verlag, 417-426.

[25] **Schneier, B.,** 1996. Applied Cryptography, Second Edition, Protocols, Algorithms and Source Code in C, John Wiley & Sons, Inc, 21-33.

[26] **CCCITT**, 1989. Draft Recommendation X.509, The Directory-Authentication Framework, *Consultation Committee, International Telephone and Telegraph, Intenational Telecommunications Union*, Geneva.

[27] **Solinas, J.,** 2000. Efficient Arithmetic on Koblitz curves, *Designs, Codes and Cryptography,* 19, pp. 195-249

[28] **Abdalla, M., Bellare, M. ve Ragaway, P.,** DHIES: An encryption scheme based on the Diffie-Hellman Problem, *http://www-cse.ucsd.edu/users/mihir/papers/dhies.html*

[29] **SEC 1**, September, 1999. Elliptic Curve Cryptography, *Standards for Efficiecncy Cryptography Group,* Working Draft. http://www.secg.org

[30] **Lopez J., Dahab, R.** Maio de 2000. An overview of Elliptic Curve Cryptography, *Relatorio Tecnico IC-00-10.*

[31] **ECKAS-DH1**, Jan 2000, IEEE Standart Specifications for Public-Key Cryptography, IEEE-SA Standarts Board, 9.2, p. 47.

[32] **ECSSA**, Jan 2000, IEEE Standart Specifications for Public-Key Cryptography, IEEE-SA Standarts Board, 10.2, p. 53.

[33] **ECSVDP-DH**, Jan 2000, IEEE Standart Specifications for Public-Key Cryptography, IEEE-SA Standarts Board, 7.2.1, p.29.

[34] **ECSP-NR, ECVP-NR,** Jan 2000. IEEE Standart Specifications for Public-Key Cryptography, IEEE-SA Standarts Board, 7.2.5-6, pp. 34-35

[35] **EMSAI**, Jan 2000. IEEE Standart Specifications for Public-Key Cryptography, IEEE-SA Standarts Board, 12.1.1.

[36] **Nyberg, K., and Rueppel, R.**, 1993. "A New Signature Scheme Based on the DSA Giving Message Recovery," *Proceedings of First ACM Conference on Computer and Communications Security,* ACM Press, 58-61.

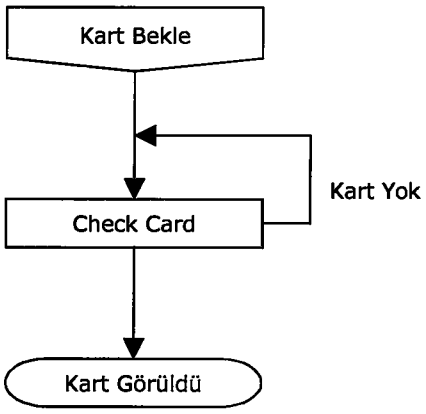
[37] **Menezes, A., Oorschot, P. and Vanstone, S.**, 1996. Hash Functions and Data Integrity, in *Handbook of Applied Cryptography*, CRC Press, 343-349.

[38] **Menezes, A., Oorschot, P. and Vanstone, S.**, 1996. Block Ciphers, in *Handbook of Applied Cryptography*, CRC Press, 163-166.

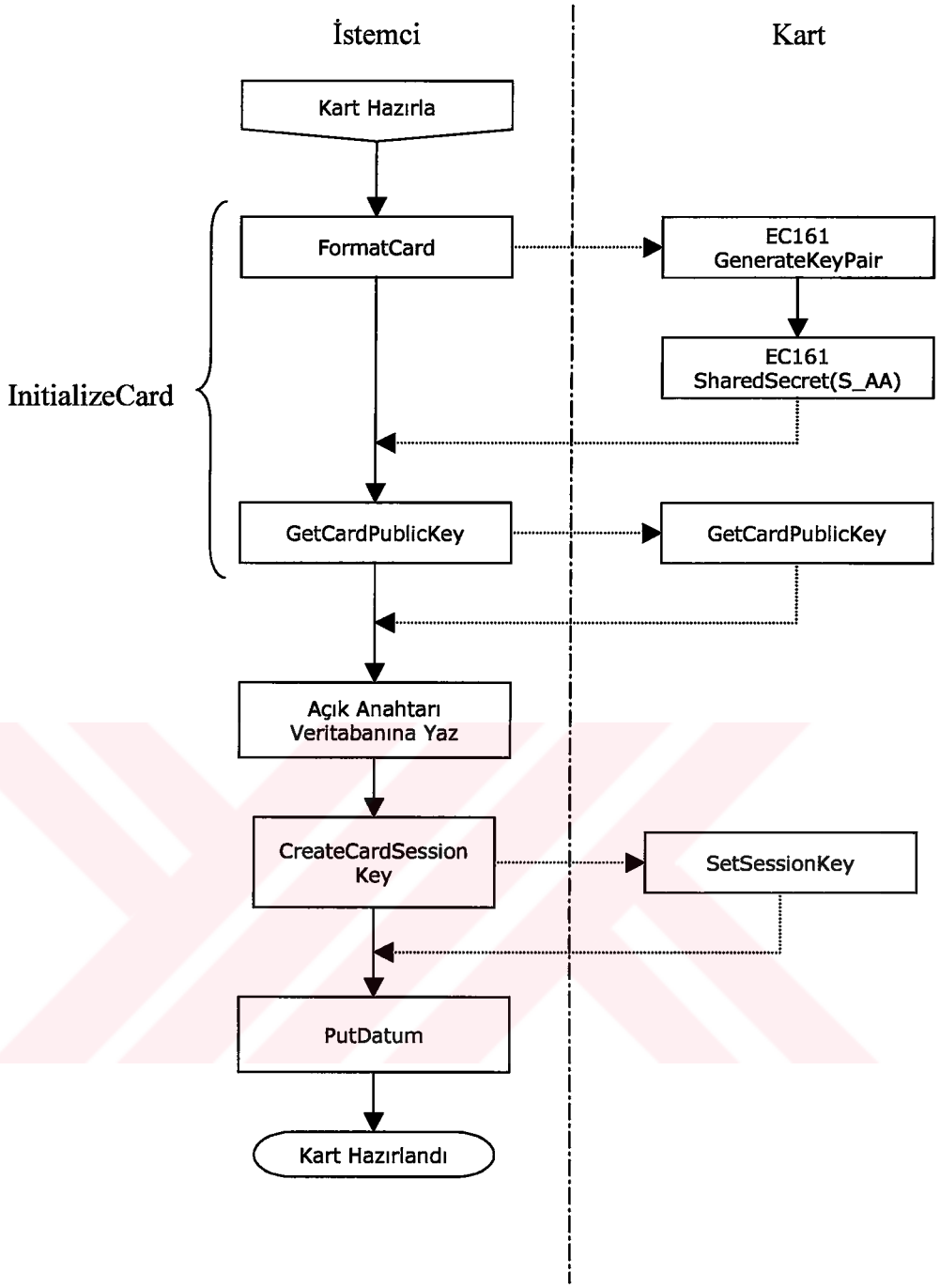
[39] **Guilfoyle, T.,** 20 th February 2001. The ZeitControl Compact and Enhanced BasicCards, *Product Catalogue,* pp. 91-93, ZeitControl cardsystems GmbH, Document version 3.20.

[40] **Guilfoyle, T.,** 14 th February 2000. Elliptic Curves in the BasicCard, *Product Catalogue,* ZeitControl cardsystems GmbH, Germany.

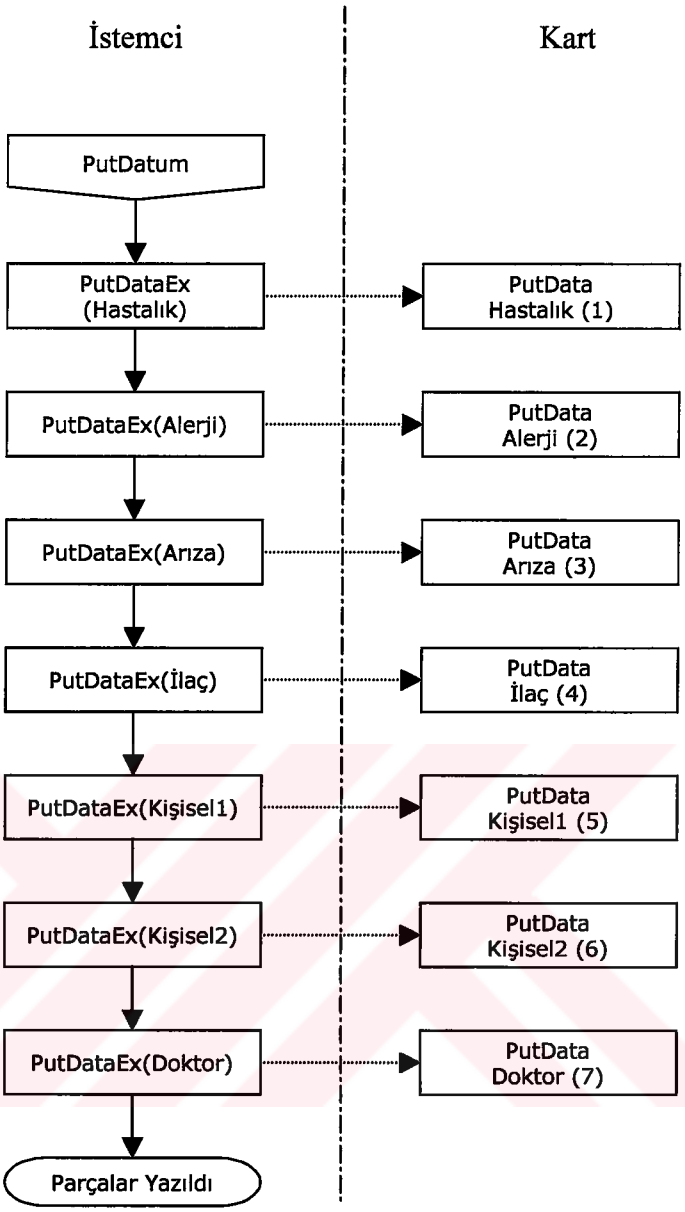
EK-A UYGULAMA YAZILIMI PROGRAM AKIŞ DİYAGRAMLARI



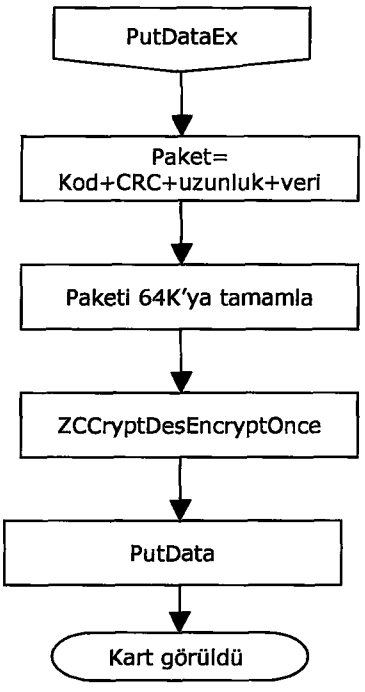
Şekil A.1 Timer prosedürü akış diyagramı



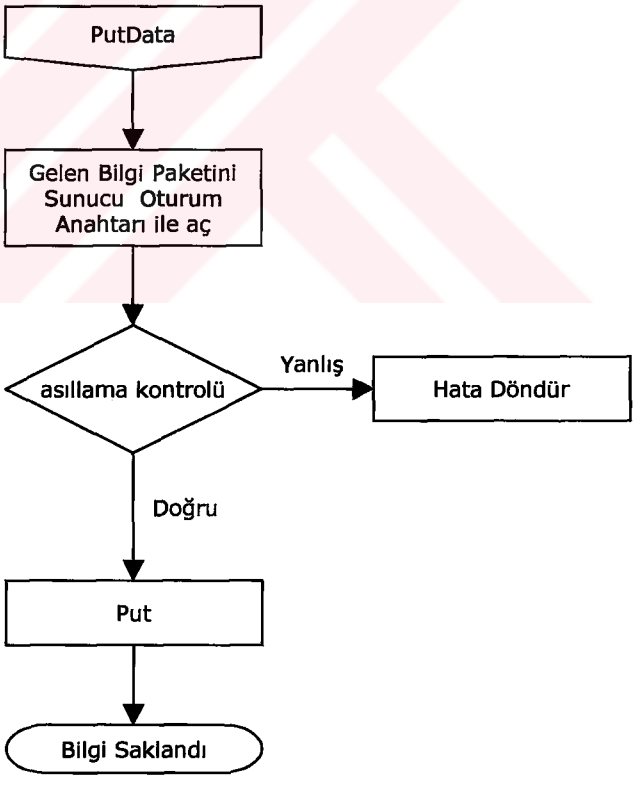
Şekil A.2. Kart yönetici programında kart basımı yordamı



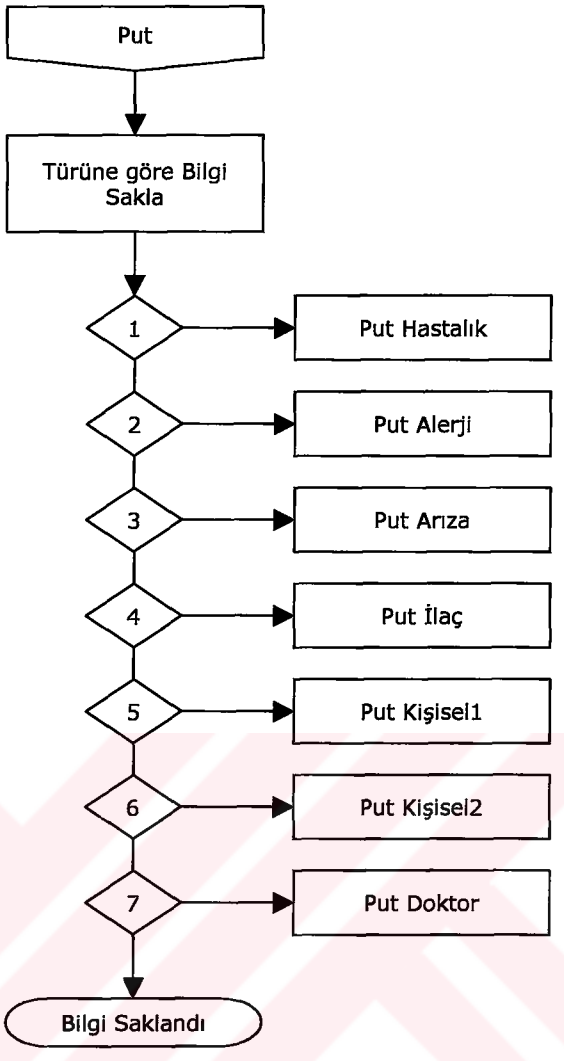
Şekil A.3. Hasta bilgilerinin karta yazılması



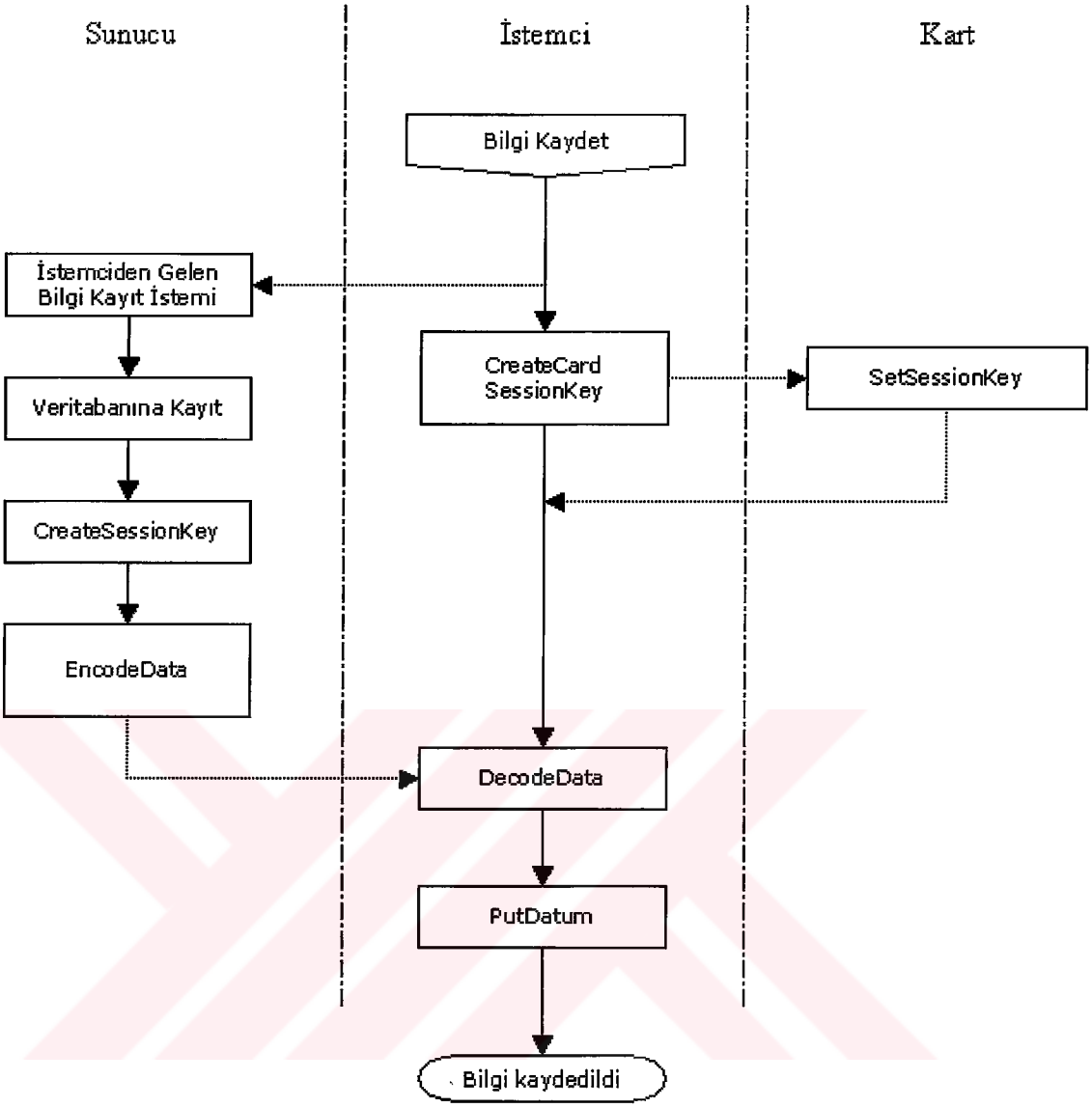
Şekil A.4. Hasta veri parçalarını şifreleyerek karta yollar



Şekil A.5. Kartta asıllama ve bilgi güncelleme

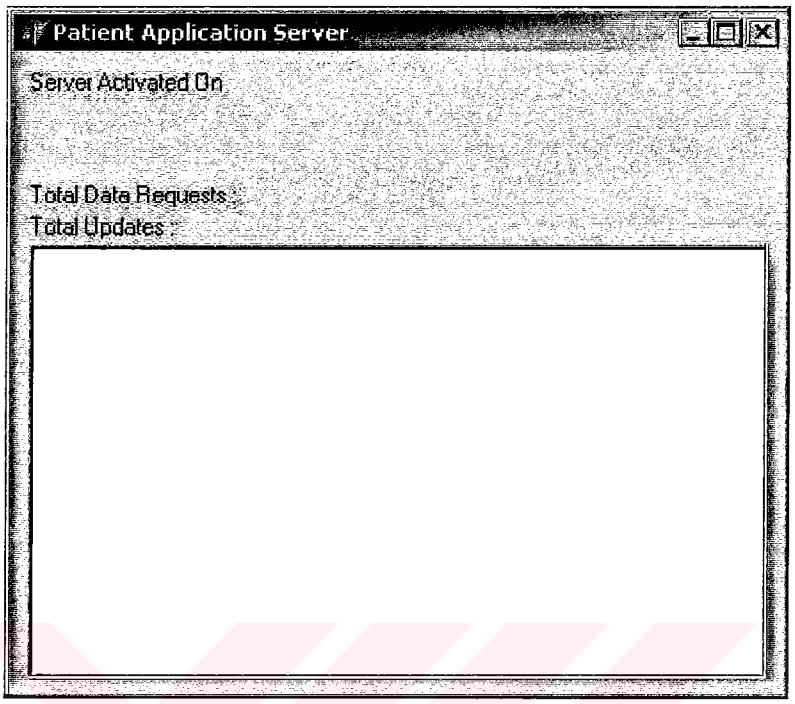


Şekil A.6. Kartta hasta bilgi paketlerinin yazımı

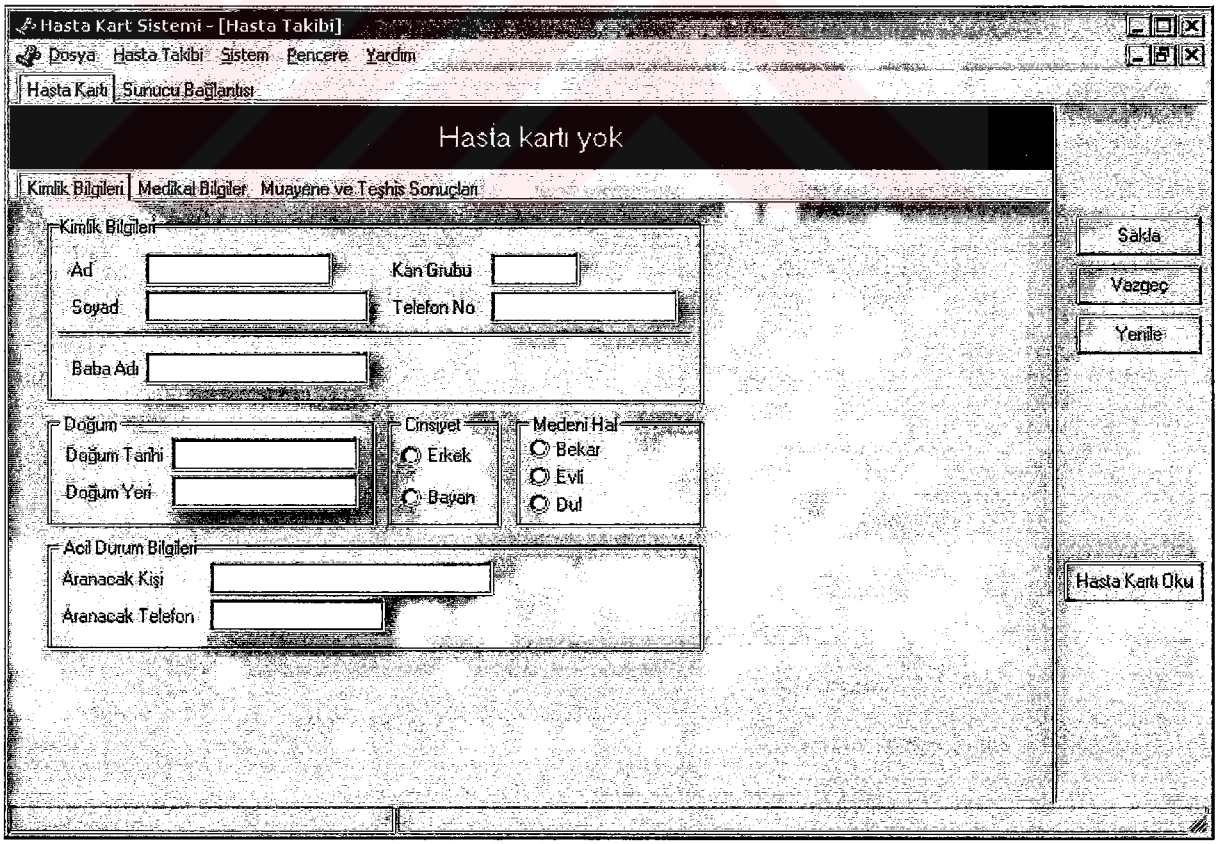


Şekil A.7 Doktorun hasta kartını güncellemesi

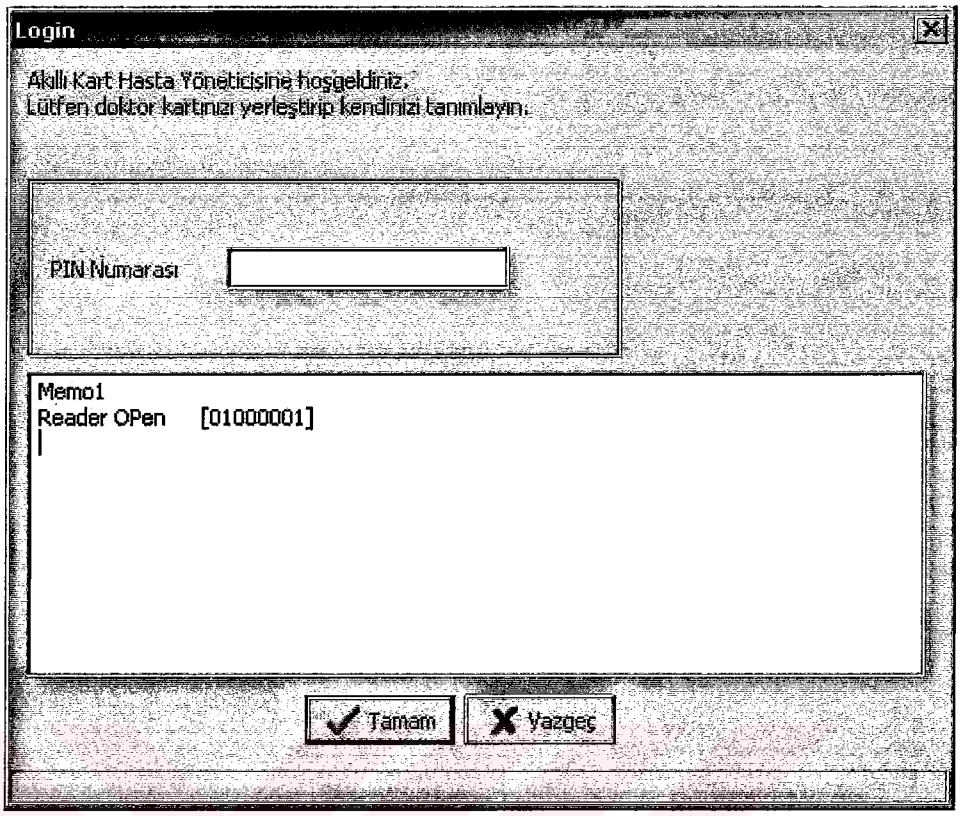
EK-B SAĞLIK SİSTEMİ EKRAN GÖRÜNTÜLERİ



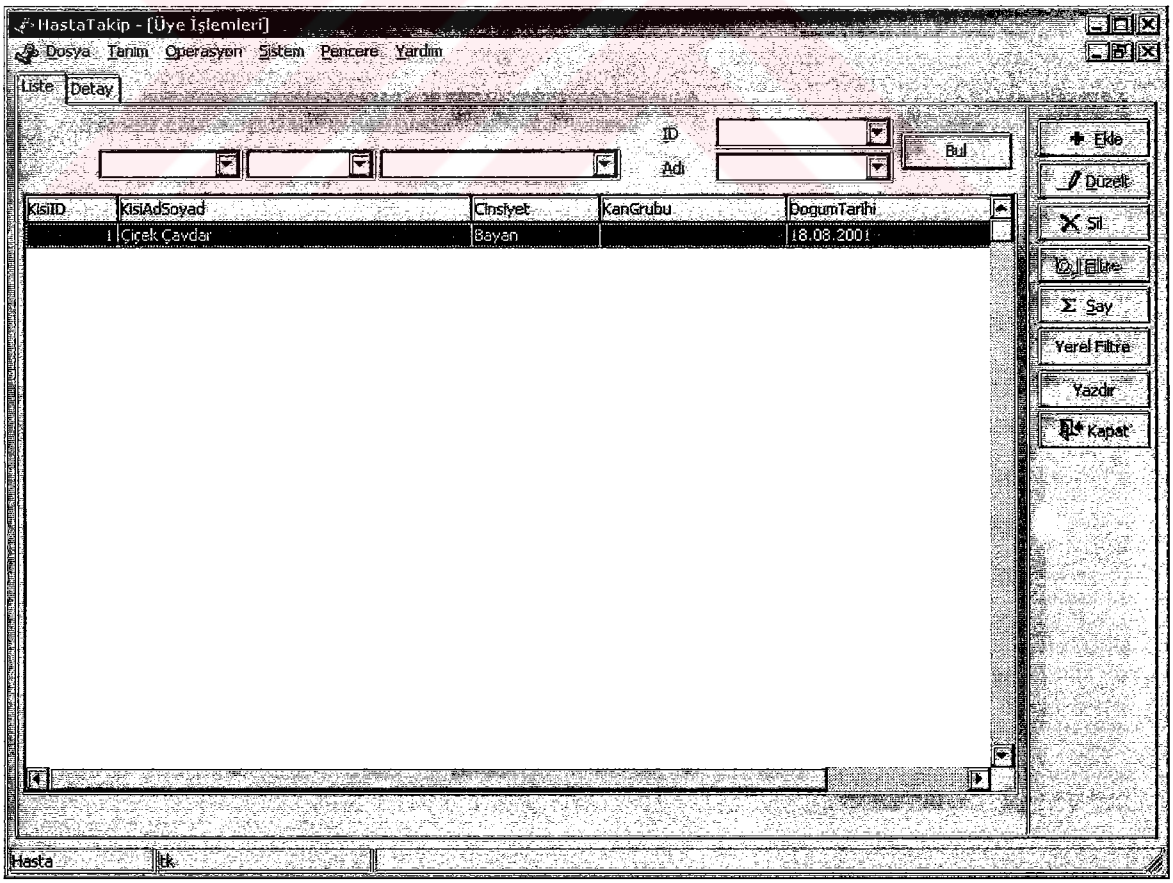
Şekil B.1 Sunucu Paneli



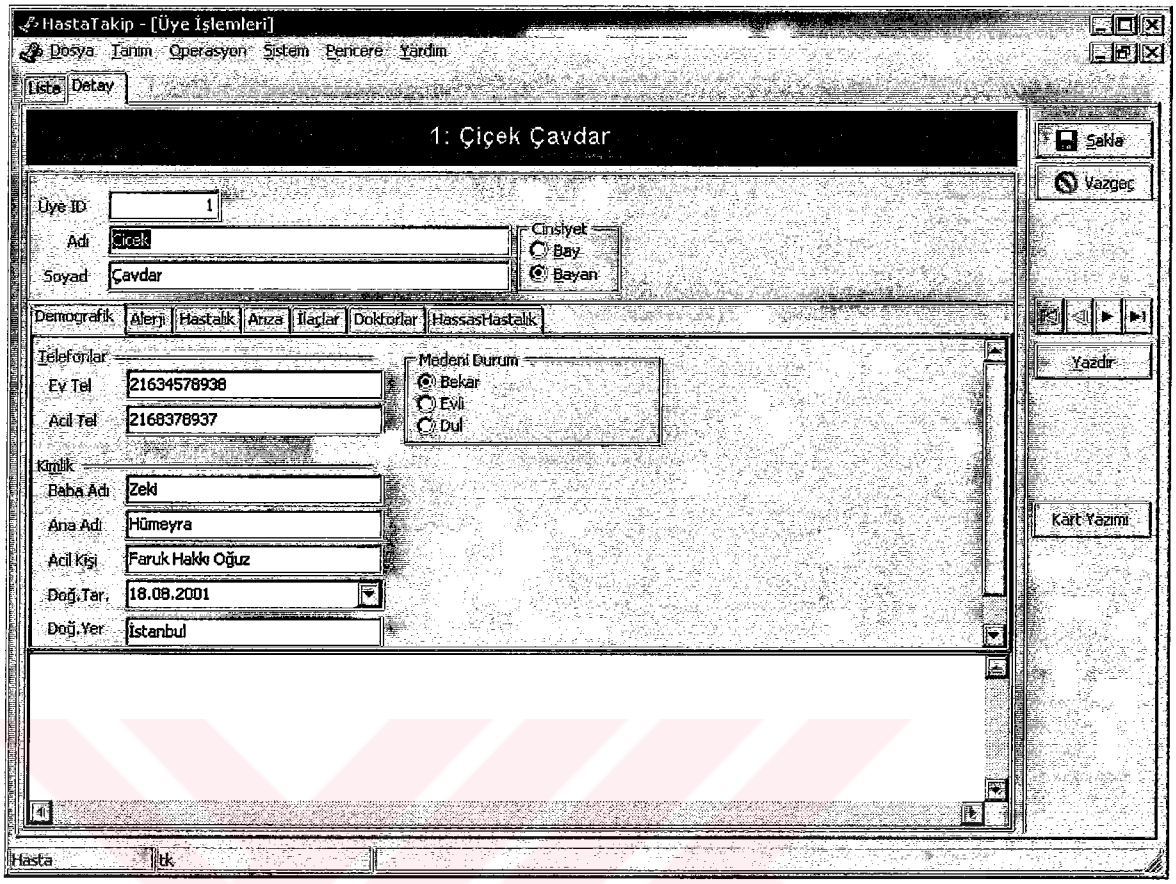
Şekil B.2. İstemci uygulama kullanıcı arayüzü hasta kimlik bilgileri ekranı



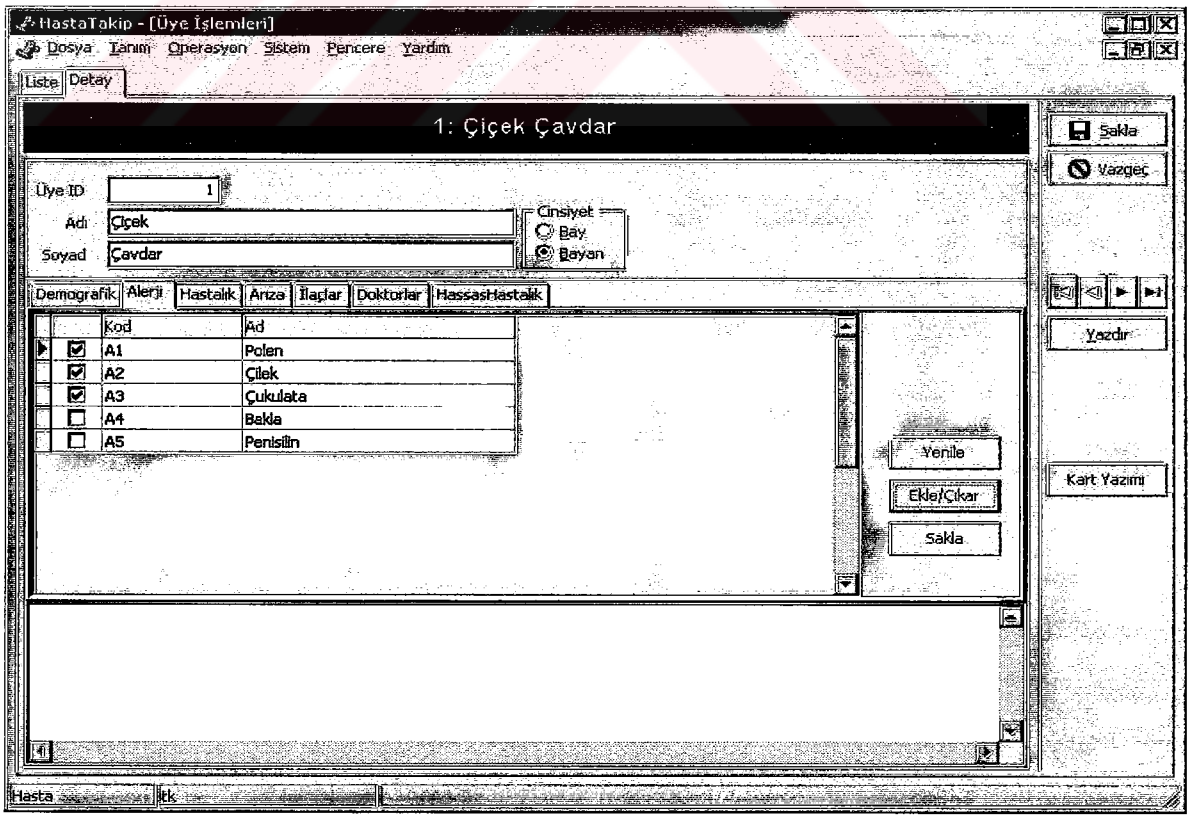
Şekil B.3. Doktor PIN Giriş ekranı



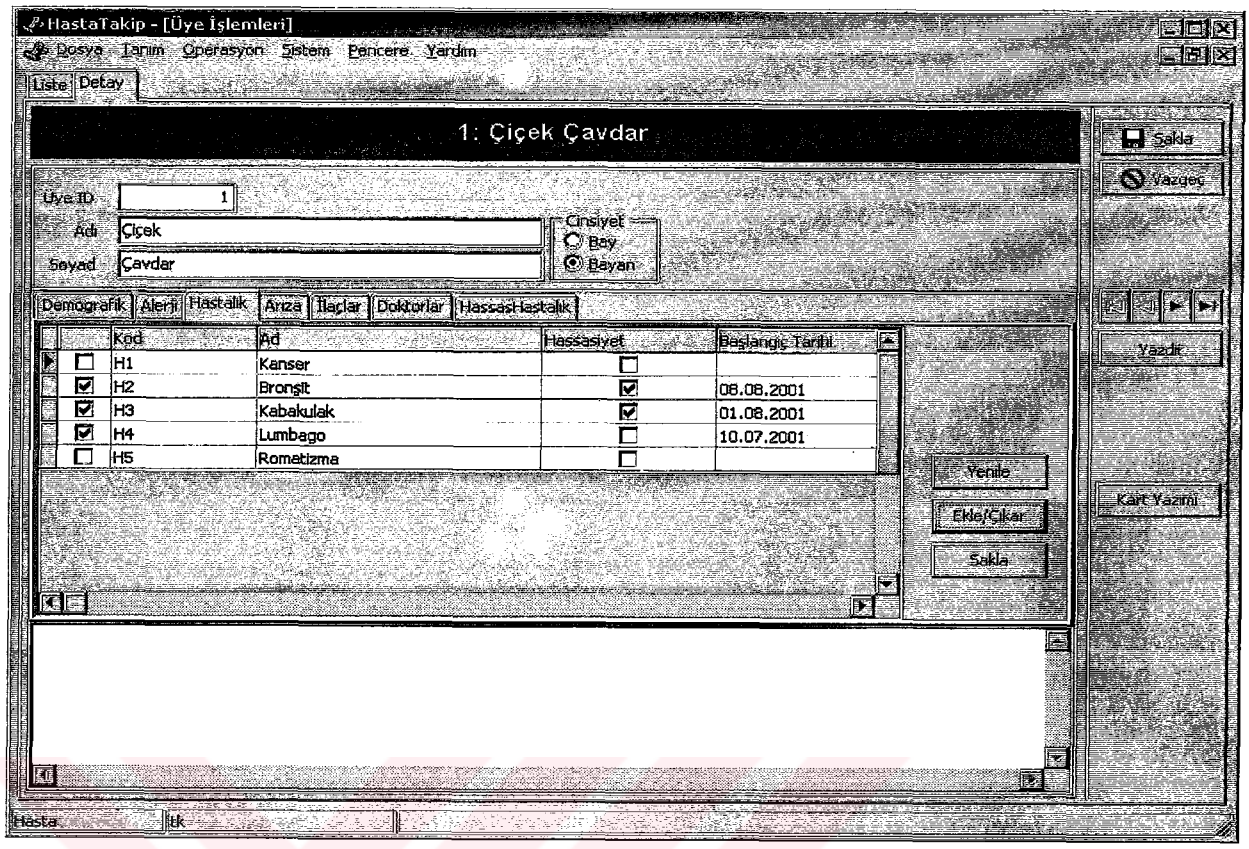
Şekil B.4. Bilgi Merkezi Hasta Listesi



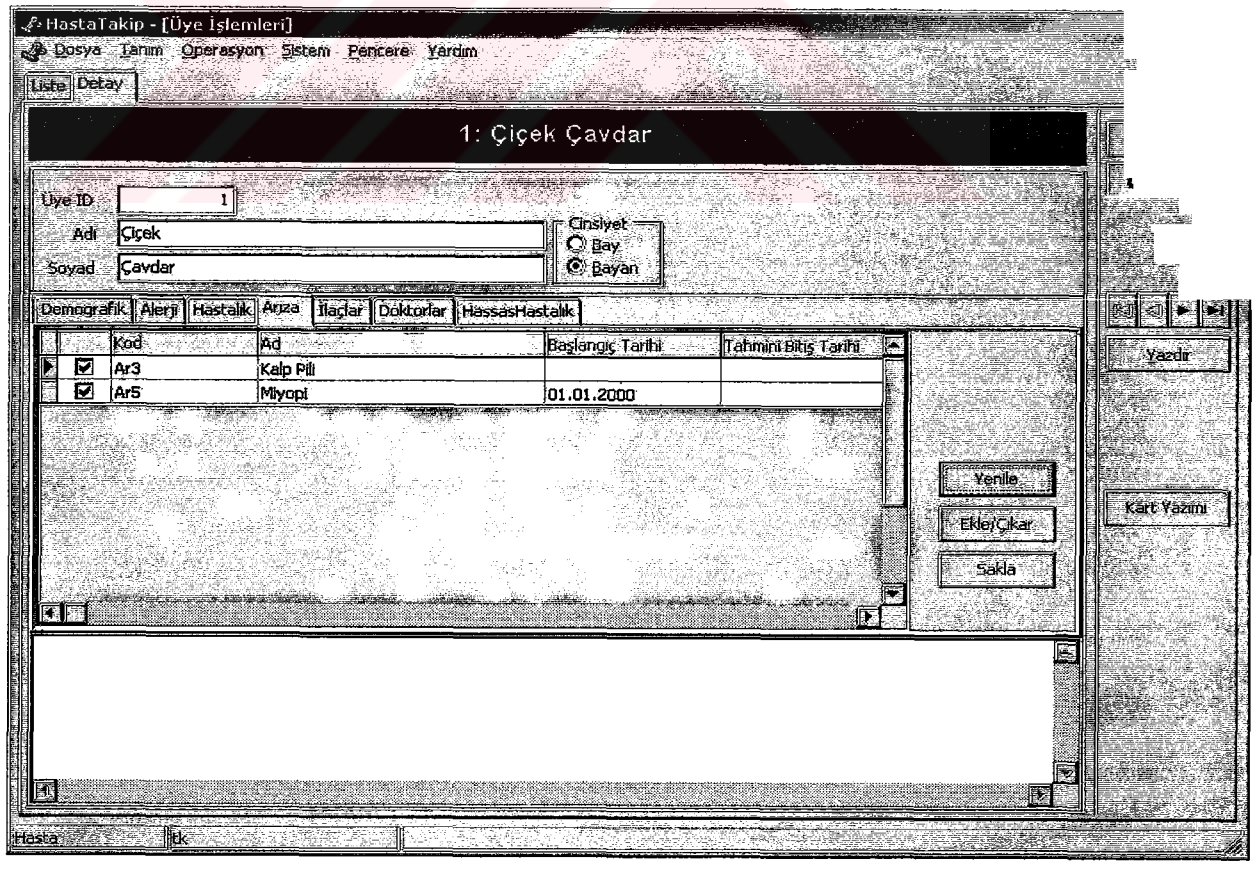
Şekil B.5. Hasta Detay – Kişisel Bilgiler



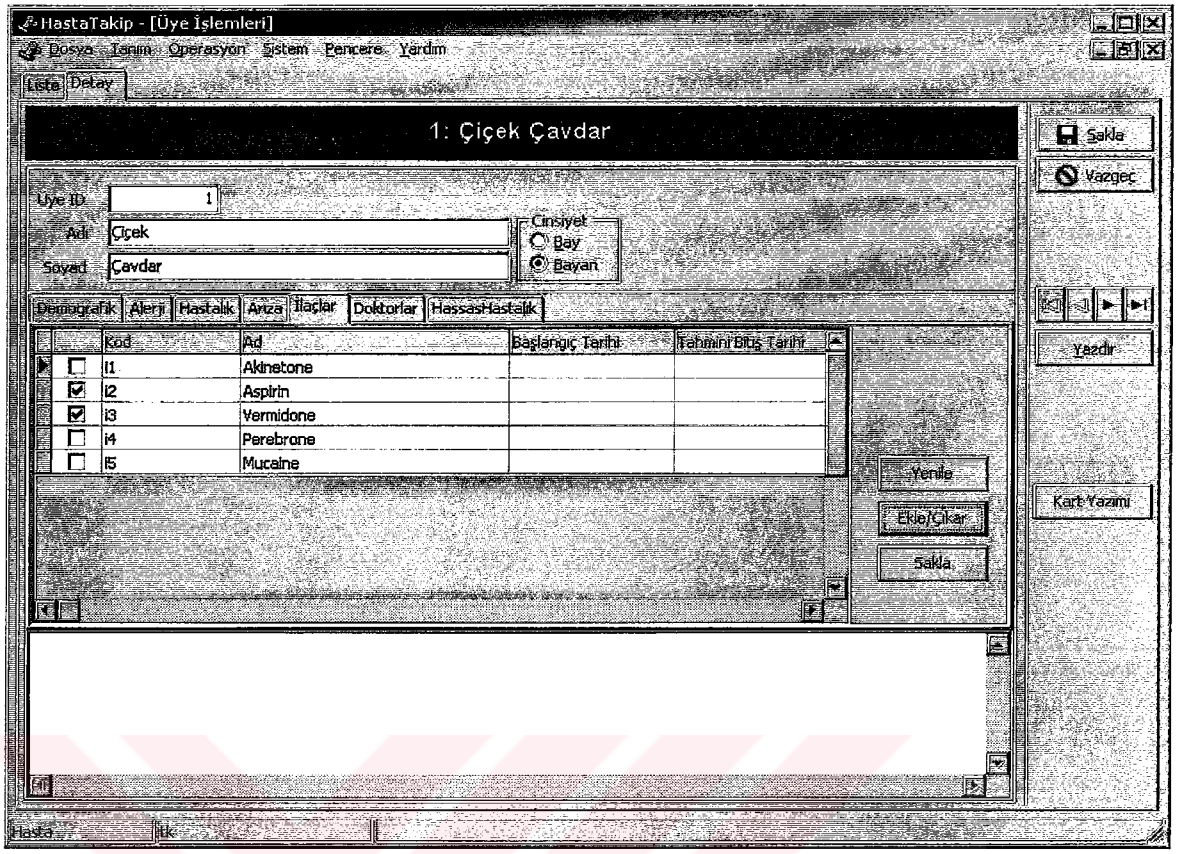
Şekil B.6. Hasta Detay – Alerji Bilgileri



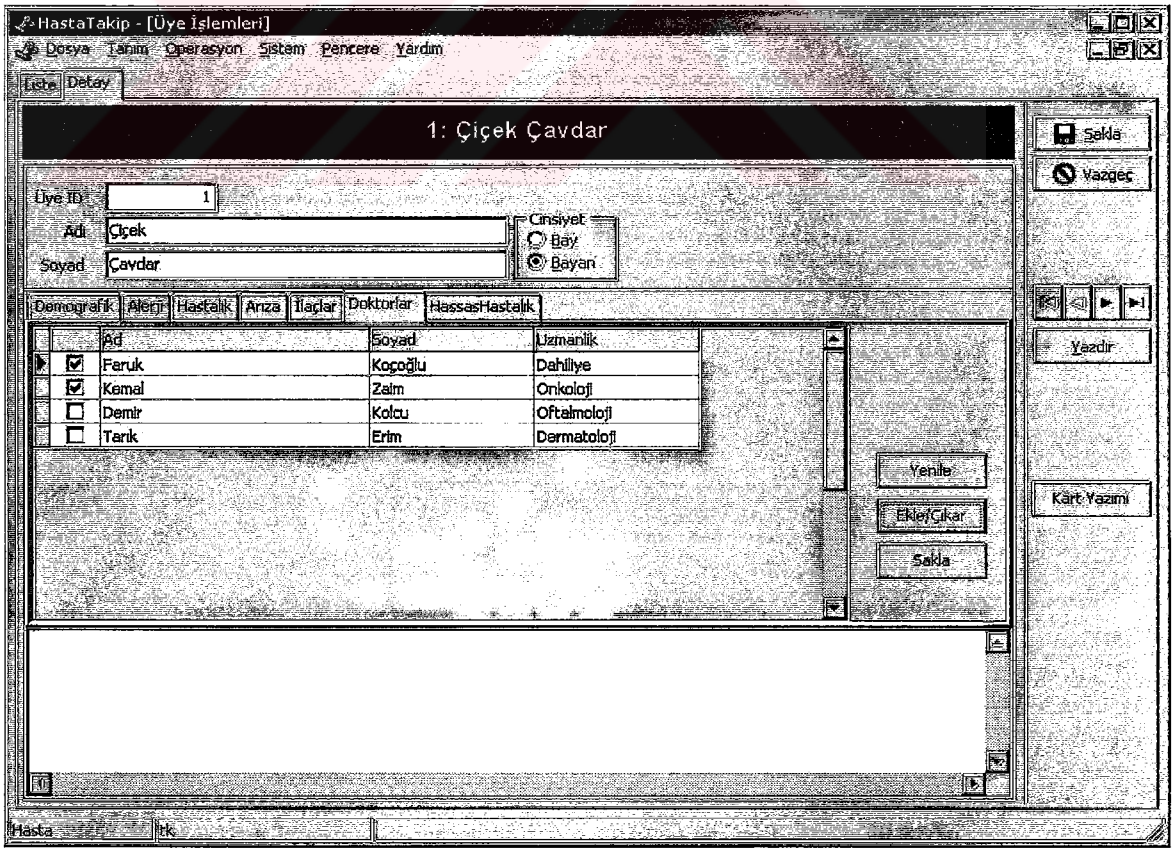
Şekil B.7 Hasta Detay – HastalıkBilgileri



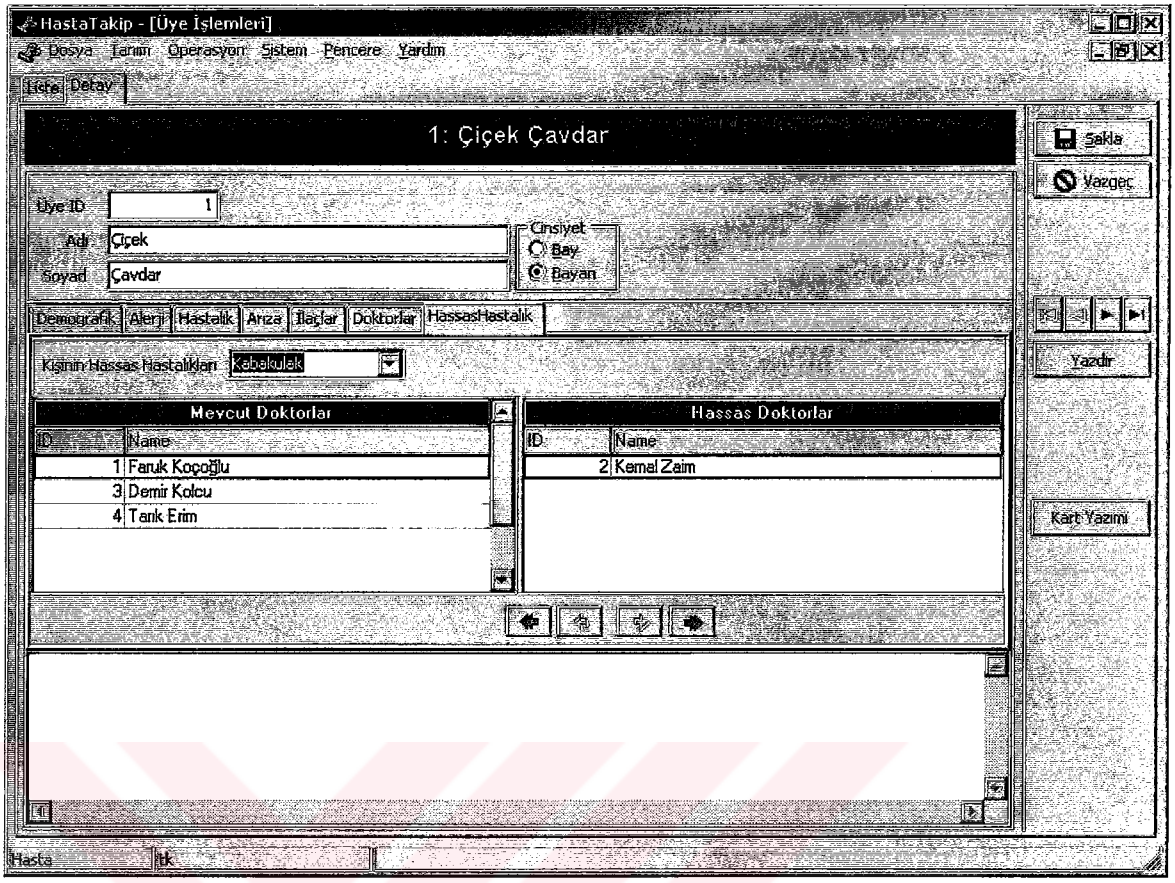
Şekil B.8. Hasta Detay – Arıza Bilgileri



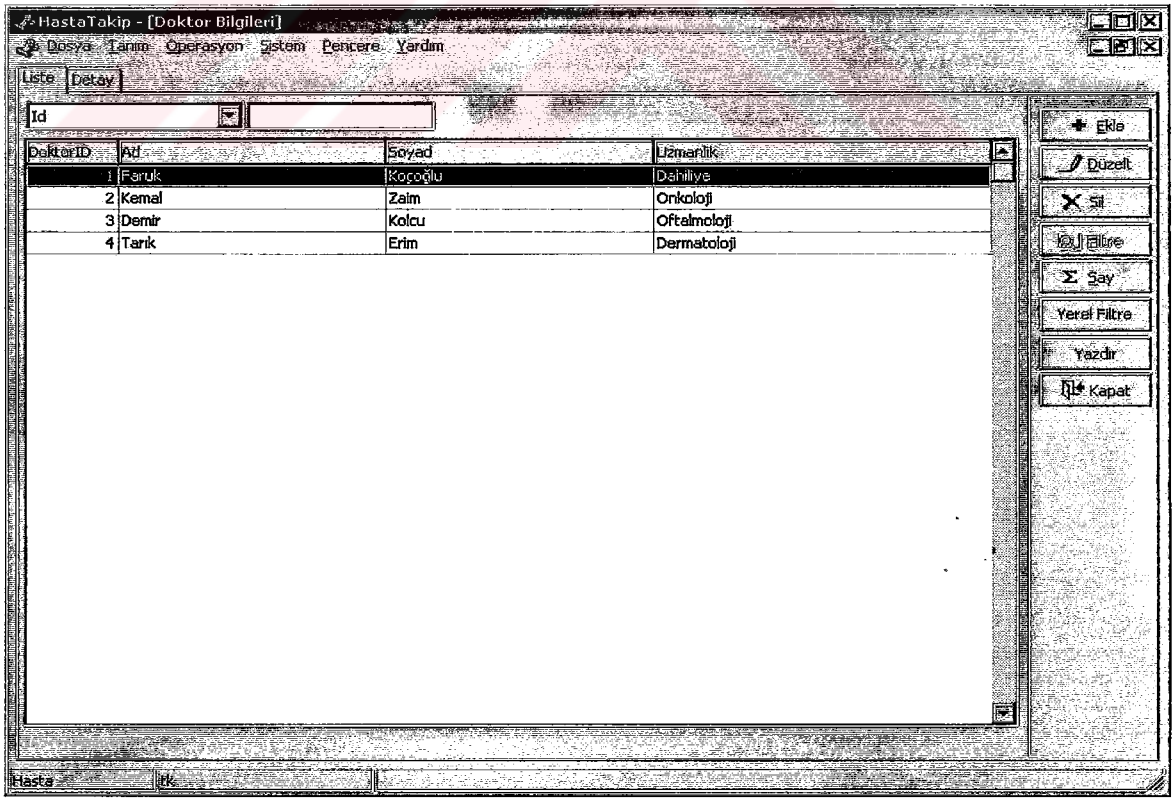
Şekil B.9. Hasta Detay – İlaç Bilgileri



Şekil B.10 Hasta Detay – Doktor Bilgileri



Şekil B.11. Hasta Detay – Alerji Bilgileri



Şekil B.12. Bilgi Merkezi-Doktor Listesi

ÖZGEÇMİŞ

1977 yılında Kütahya'da doğan Çiçek Çavdar ilköğrenimini Tavşanlı'da tamamladıktan sonra orta ve lise öğrenimine İzmir'de devam etti.

Fen Lisesi öğrencilik dönemi boyunca bilimsel çalışma ve araştırma ortamı içinde yer aldı. 1992'de 9 Eylül Üniversitesi'nde Prof.Dr. Ata Önvural'ın danışmanlığında Tüp Bebek konusunda araştırma yapan Çiçek Çavdar, 1993'te Ege Üniversitesi Kimya Bölümü'nde Parafin Oksidasyonu konusunda deneysel çalışmalar yaptı. Çalışmanın sonuçları, MEF Yayınları'nda Lise Öğrencileri Araştırma Projeleri yarışmasının sonucunda 1993 yılında yayınlandı.

1994-1998 yılları arasında İstanbul Teknik Üniversitesi Bilgisayar Mühendisliği Bölümü'nde lisans öğrenimini tamamladı. Çiçek Çavdar aynı bölümde araştırma görevlisi olarak görev yapmaktadır. Bilgisayar Müh. Yüksek Lisans Bölümü'nden 2002 Ocak dönemi mezun olmak üzere tez çalışmasını hazırlamıştır. Tez çalışması döneminde bir ulusal bir uluslararası olmak üzere iki konferansta bildiri sunmuştur. "Sağlık Alanında Akıllı Kart Uygulamaları" başlıklı bildirisi Bilişim 2000 Konferansı'nda; "Security Protocol Design in a Health Care System Using Smart Cards" başlıklı bildirisi ise IASTED International Conference on Applied Informatics'te yer almıştır.