Prescriptions Management

Pharmacy module

Olga Dzięgielewska
Marcin Klepaczka
Andrzej Rybczak
Jan Szajda

Wrocław, 2014

Contents

Part I

Introduction

Chapter 1

Current Situation

Current system strongly depends on paper prescriptions. Each prescriptions carries a lot of data, which some can be treated as private data of patients:

- prescription's creation date,
- patient's personal data:
  - name and surname
  - address
  - PESEL
- number of the prescription, specific for each doctor [1],
- list of medicines with refoundation level,
- signature and stamp of the doctor.

The patient, who was given the prescription by the doctor, goes to the pharmacy to buy the medicines. He gives his prescription to a pharmacist and says which of the medicines from the list he wants to buy. The pharmacist checks if the medicines are

[1] NFZ generates a list of prescription for each doctor. Every prescription has the unique identifier number. During the refoundation process, NFZ checks, if the number on the prescription, the doctor name, signature and stamp are correct. Only if thy are valid, the refoundation is granted.

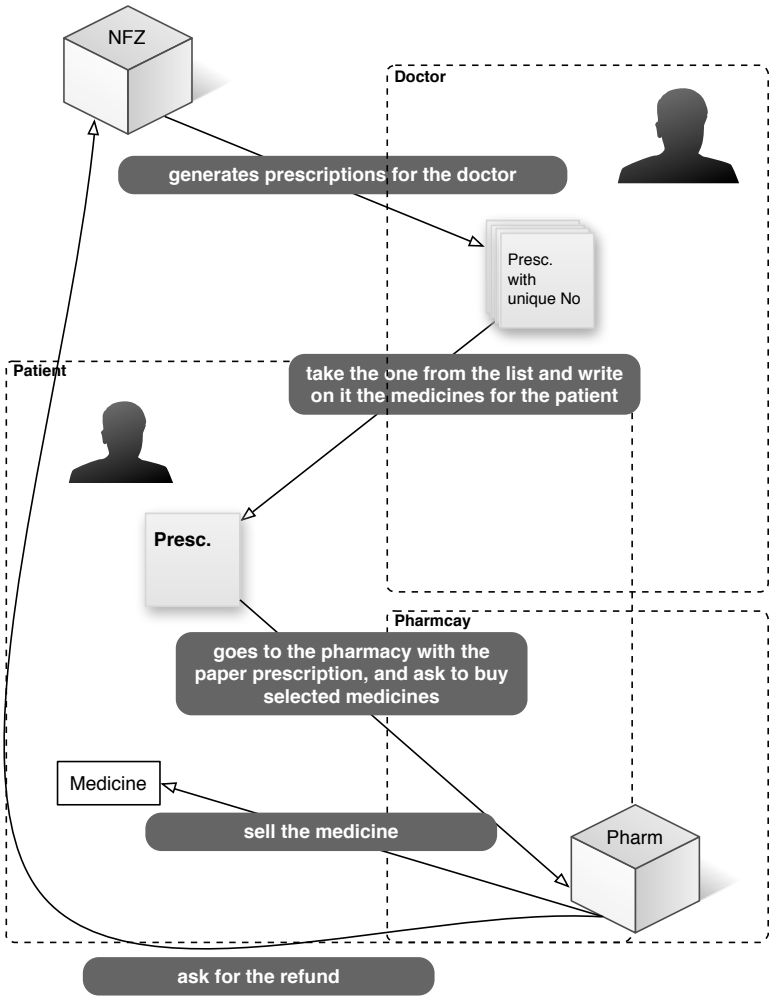


Figure 1.1: The main points of currently used system

available and if yes, he sells them. Next, he takes the prescription and makes a signature next to the each of the medicine he sold. He also inputs to the software installed on computers in the pharmacy, which of the medicine was sold, for who, who gave the prescription and what are the refoundation costs.

Each month in every pharmacy a report, consisting of the set of the information about each prescription sold in the pharmacy is generated. This report is sent to the NFZ central database. Based on this, the NFZ refunds costs of the medicines. Each prescription has to be kept for at least five years in the pharmacy, and be ready for checking during controls made by NFZ representatives.

Chapter 2

Threats and Inconveniences

The way prescriptions are currently processed is vulnerable to many threats, and brings many inconveniences. The most important ones are listed below.

2.1 NFZ

2.1.1 Defraudation

Significant amount of money is being defrauded from NFZ because the current system does not verify if the patient himself has bought the medicine or the pharmacists has made a false call for the medicine having some patient's prescription, prepared by the doctor (who is also a part of the defraudation scheme).

2.2 Patient

2.2.1 Losing a prescription

The patient can lose the prescription and he cannot buy the medicines, even if they are life-saving, he has to go to the doctor again and ask for the new prescription.

If someone finds lost prescription, he can buy this medicines; what is more, this person can get to know, who takes which medicines and in this way, he can get to know, what is wrong with the person described on the prescription.

2.3 Pharmacy

2.3.1 Refundation delay

The pharmacy has to wait long time to refund costs for the medicines from NFZ.

2.3.2 Prescriptions with mistakes

Prescription tend to contain mistakes which makes it useless. In this situation the patient has to go to the doctor again so it's fixed.

2.4 System

2.4.1 Prescription forgery

Patient can try to copy the prescription and try to buy the medicines few times in different pharmacies.

2.4.2 Pretending that prescription was lost

Patient can claim that he has lost his prescription and ask the doctor to give him another one. Then he can buy the medicines twice.

Chapter 3

System Goals

The system has to eliminate each of the flaws described in previous chapter. It will meet each of following requirements:

1. Prescriptions will be digitalized.
2. Prescriptions will be hard to forge.
3. Doctors won't be able to create prescriptions without knowledge of patient.
4. Prescriptions will be realized only by users with right credentials.
5. Patients and doctors will be able to browse history of prescriptions.
6. System will be secured with most up-to-date measures.
7. System will provide anonymous big data statistics.

3.1 Central Server Objectives

The central server will be the core component of the whole digital prescriptions system. Key features of central server are:

- Holding data of patients, doctors and pharmacists
- Allowing doctors to create prescriptions
- Allowing doctors and patients to review history of created prescriptions
- Allowing patients to transfer the ownership of prescription in secure, controlable manner
- Allowing pharmacists to review prescriptions yet to be realized
- Allowing prescription realization only if patient will be present at this event
- Validating the signatures of each party
- Providing annonymous statistics

3.2 Pharmacy Module Objectives

The main objectives of our new design of the pharmacy module is to limit the impact of the threats listed in Chapter 2 and improve the usability of the current system.
The patient has to be sure that his sensitive data is stored in a secure way, and unauthorized person cannot get to know anything about his medicines and illnesses.
The pharmacist has to be sure that he sells the right medicines only for the right patient.
The refund process should be quicker and easier.
The possibility of making **mistakes** on the prescription should be eliminated.
The number of **defraudations** should be significantly limited.

Part II

Project

Chapter 4

Project

4.1 Environment Requirements

4.1.1 Smart Cards

The main reason we decided to use smart cards is that smart card solutions, which employs two factor authentication, i.e. "something you have and something you know", provide a high security level which is crucial for the health's systems sensitive data.

All the system's users will be given personalized smart cards which will store their identification data: names, surnames, PESEL and digital certificates. Each card will be assigned PIN and PUK numbers. The first one will be used to initialize authentication process, the second one will be used for unblocking a card[1].

To improve the security level of the system, the data stored on smart cards should be enciphered. Users' private keys need to be stored in a secure memory which cannot be directly read out.

Keys used for connecting and authorising should have sufficient length to provide security. If the RSA key is used it should have length of at least 2048 bits.

[1] Unblocking procedure can be performed in the two following situation: when a user inputs wrong PIN number three times in a row or when he blocks his card after loosing it.

In case of loosing a smart card, a user should perform a standardized revocation procedure. First, he should block a card in the assigned institution and while doing this he should be able to select whether he want to block the card temporarily or permanently. In the first case, after finding the card it is possible to unblock it with card's PUK number. In the second case it is necessary to generate new user's card and even after finding the card it will not be possible to unblock it.

4.1.2 Two-Way SSL

Two-Way SSL provides the same functionalities as SSL, with the addition of authentication and non-repudiation of the client authentication, using digital signatures. When mutual authentication is used the server would request the client to provide a certificate in addition to the server certificate issued to the client. The main advantages of client-certificate authentication are:

1. The private information (the private key) is never sent to the server. The client doesn't let its secret out at all during the authentication.
2. A server that doesn't know a user with that certificate can still authenticate that user, provided it trusts the CA (Certificate Authority) that issued the certificate (and that the certificate is valid). This is very similar to the way passports are used: you may have never met a person showing you a passport, but because you trust the issuing authority, you're able to link the identity to the person.

4.1.3 Certificates

In Two-Way SSL both parties (client and server) need the certificates. Each user has his digital certificate on his smart card. All the user's certificates must be given by a defined certification authority and regularly[2] updated. The CA also generates a keypair for the database server.

In case of selecting permanent blocking option during the revocation procedure, a new certificate is generated for such user.

[2]The CA should define a standard validity period for the patient's, pharmacist's and doctor's certificates.

The certificate's validity should be checked at each use of the user's smartcard. The validity check is performed in the database module.

4.1.4 Pharmacy

All the pharmacies which will be using the system must have broadband internet access, two smart card readers and two terminals: one for a pharmacist and one for a customer. The terminals apart from displaying the data need to handle all the confirmation actions on both sides.

4.2 Architecture

4.2.1 Central Server Architecture

Central server will be constructed of several components. In order to provide all necessary data and functionalities to the users this is system will be a cooperation of system's logic, specific APIs and database. In the project the following components were chosen:

1. **server layer** - Apache HTTP Server ("Apache") version 2.4.9
2. **database layer** - PostgreSQL version 9.3

4.2.2 The Pharmacy Module

The pharmacy module architecture consists of the following elements:

1. **smart cards** with personal certificate, used for the authentication and signing, and an application which allows to read certain data from the card;
2. **pharmacist's PC** with a pharmacy module application which provides all of the functionalities which satisfy all the operation performed in a pharmacy; provides

two user-friendly interfaces: one for a patient and one for a pharmacist; is connected with patient's and pharmacist's terminals and the central database; is able to execute SIGMA protocol, handle secure keys storage and establish SSL connection;

3. **central DB** is a central element of the whole system; stores the data and handles all the necessary database I/O functions.

4.3 Protection and Security

Below we describe entities used in the system, how we choose to protect them and why.

4.3.1 Protection methods

4.3.1.1 Patient's card

Patient's card stores private key along with the certificate. Elements of the certificate are as follows (*text in parentheses describes what is used*):

- **Serial Number**: Used to uniquely identify the certificate.
- **Subject**: The person, or entity identified (*personal data of the patient*).
- **Signature Algorithm**: The algorithm used to create the signature (*RSA*).
- **Signature**: The actual signature to verify that it came from the issuer.
- **Issuer**: The entity that verified the information and issued the certificate (*CA for the patient*).
- **Valid-From**: The date the certificate is first valid from.
- **Valid-To**: The expiration date.
- **Public Key**: The public key.
- **Thumbprint Algorithm**: The algorithm used to hash the public key certificate (*SHA256*).
- **Thumbprint (also known as fingerprint)**: The hash itself, used as an abbreviated form of the public key certificate.

4.3.1.2 Pharmacist's card

Pharmacist's card stores the same information as patient's card, with exception to several certificate fields being different:

- **Subject:** *Personal data of the pharmacist and pharmacy*
- **Issuer:** *CA for the pharmacies*

4.3.1.3 Card's data access

Card is read only is the sense that patients/pharmacists are not able to modify the data that is stored on it. They do, however (after successful authentication), have access to certificate stored on the card as well as the function to sign arbitrary input data with its private key.

4.3.1.4 PIN

The certificate access/signing input data can be performed after inputting a PIN. The user is given 4-digit PIN number and the verification system will allow three attempts of typing the correct number before the card is blocked.

4.3.1.5 Patient authentication

Two factor authentication is used:

- Something you have - smart card (containing user's certificate)
- Something you know - PIN number used to access the certificate on the card

4.3.1.6 Pharmacist authentication

Two factor authentication is used:

- Something you have - smart card (containing pharmacist's certificate)
- Something you know - PIN number used to access the certificate on the card

4.3.1.7 Connection between card and application

After successful authentication we establish a session key and the communication is encrypted with it. For that AKE protocol „SIGMA" is utilized. (Note that encryption is optional if we assume that no eavesdropping can take place or the data exchanged isn't considered confidential).

4.3.1.8 Connection between application and central database

Two-way SSL connection is used along with additional nonce-based authentication.

4.3.2 Justification

4.3.2.1 PIN protection

The PIN number is used to authenticate the card holder. In case the card was lost and found by someone else, he won't be able to use the card without knowing the PIN. We propose 4-digit PIN number with three subsequent incorrect attempts before the card is blocked as it's already used e.g. in ATM cards and proven to work there.

4.3.2.2 SIGMA protocol

This AKE protocol (we choose to use SIGMA, but that is by no means the ultimate choice. It's been chosen due to convenience of having the implementation already in place. If one wishes, it can be replaced by other AKE protocol, e.g. NAXOS) will be used to secure the communication channels between parties existing in the pharmacy, i.e. cards and application. AKE protocols provide not only secure communication but also authentication mechanism, preventing not only eavesdropping or man-in-the-middle attacks but also party substitution.

4.3.2.3 Secure key disposal

All short term keys, i.e. ephemeral keys used using AKE protocol or session keys which are the result of the protocol are erased from memory immediately after they are no longer needed.

4.3.2.4 Secure communication with the database

The communication channel between database and pharmacy is secured with an SSL connection. We assume the SSL provides all the necessary mechanisms to protect the channel from attacks. To strengthen the security of the channel all the requests from any valid party must contain the signature (RSA signature) over the nonce provided by database system. This solution ensure that no unauthorized party is able to get access to database.

4.3.2.5 Protection against defraudation

Each transaction has to be signed by all the participating parties. In this setting it is impossible for the doctor/pharmacist to fake the medicaments sale and deceive NFZ into giving them money for refunding the nonexistent costs, as signature of the patient is also required. Additionally a token that has to be signed changes with each transaction, so protection from repetition attacks is also gained.

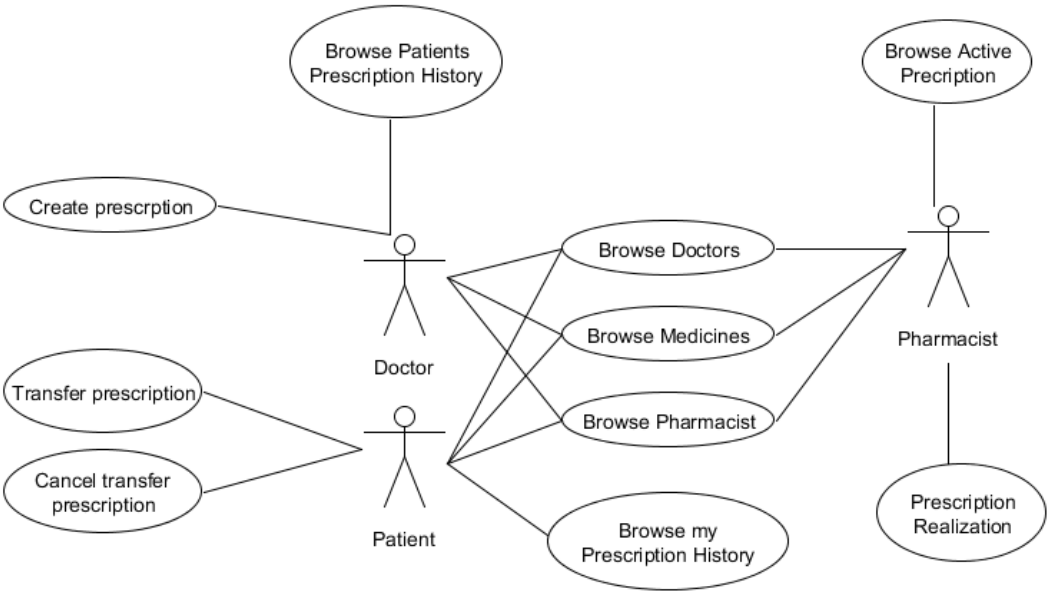
Part III

Database

Chapter 5

Use cases

We define two groups of actors - clients like patients, doctors and pharmacists which benefit from the system on daily basis and third parties - administrators, analytic tools and goverment authorities which cope with the system on special ocassions.



Every use case requires the users to establish secure channel of communication with central server and have to be logged in which will be described more thorougly in section 6.1.

5.1 Shared use cases

Three use cases are applicable for patient, pharmacist and doctors and they consider browsing informations which can be publicly accessible, that is:

- Browse Doctors
- Browse Medicines
- Browse Pharmacists

Rest of use cases which are applicable to only one actor is described in their respective subsections.

Actors: Patient, Doctor, Pharmacist	Title: Browse Doctors
Goal:	Allows to find doctor with specific name, address or license number.
Scenario:	User enters any or all of name, address and license number of searched doctor.
Result:	List of doctors corresponding to the query.
Database method:	browse_doctors

Actors: Patient, Doctor, Pharmacist	Title: Browse Pharmacies
Goal:	Allows to find pharmacist and pharmacy with specific name, address or license number.
Scenario:	User enters any or all of name, address, license number of searched pharmacist or pharmacy name.
Result:	List of pharmacists corresponding to the query.
Database method:	browse_pharmacies

Actors: Patient, Doctor, Pharmacist	Title: Browse Medicines
Goal:	Allows to find medicine with specific name or type.
Scenario:	User enters name or/and type of medicine he is searching.
Result:	List of medicines corresponding to the query.
Database method:	browse_medicines

5.2 Patient

Actors: Patient	Title: Transfer prescription
Goal:	Allows to transfer a prescription to another patient and give him credentials to realize this prescription. Patient who transferred the prescription losses his right to realize it by himself. If he wants the prescription back he has to cancel the transfer (next use case).
Scenario:	Patient enters his id, id of new owner, the prescription id he wants to transfer and his signature.
Result:	OK response from database and iId of new owner of prescription.
Database method:	transfer_prescription

Actors: Patient	Title: Cancel Transfer Prescription
Goal:	Allows to revert transfering of prescription to another patient.
Scenario:	Patient enters his id, prescription id he wants transfers to revert and his signature.
Result:	OK response from database.
Database method:	cancel_prescription_transfer

Actors: Patient	Title: Browse My Prescriptions History
Goal:	Patient can see his history of realized and created prescriptions.
Scenario:	Patient sends his id which is signed by his key from smartcard. Patient can define the time span of returned prescriptions as also a filter to only return prescriptions which aren't realized yet.
Result:	List of prescriptions for the patient.
Database method:	browse_prescription_history

5.3 Doctor

Actors: Doctor	Title: Create prescription
Goal:	Allows to create a new prescription in database for selected patient..
Scenario:	Doctor enters his and patients ids, as well as the data specific to the medicine - id, dosage, unit and quanitity. Everything is signed by his key.
Result:	OK response from database.
Database method:	create_prescription

Actors: Doctor	Title: Browse Patients Prescriptions History
Goal:	Doctor can see patient history of realized and created prescriptions..
Scenario:	Doctor sends his id - he will see all prescriptions created by him. If he will add the id of patient with patients signature, he will see the full history of prescriptions of current patient. Doctor can define the time span of returned prescriptions as also a filter to only return prescriptions which aren't realized yet.
Result:	List of prescriptions for the patient.
Database method:	browse_patient_prescription_history

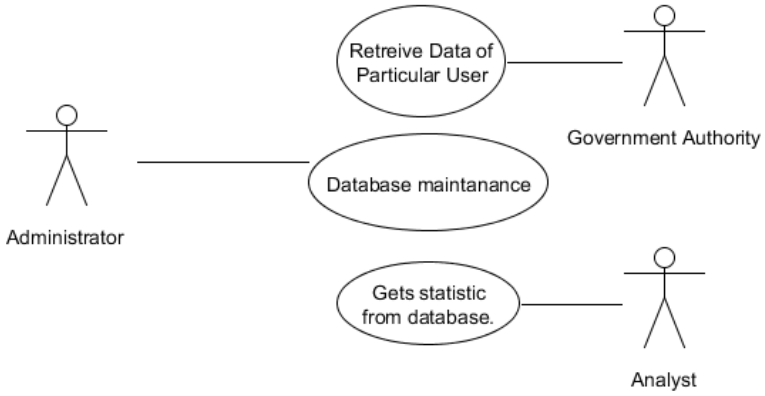
5.4 Pharmacist

Actors: Pharmacist	Title: Prescription realization
Goal:	Pharmacist realizes the prescription. DB checks if the request can be verified and if the prescription is valid.
Scenario:	Pharmacist enters his id, prescription id as well as drugs id, dosage and qunatity of medicine. Everything is signed by pharmacist key.
Result:	OK response from database if operation was successful.
Database method:	prescription_realization

Actors: Pharmacist	Title: Browse Active Prescriptions
Goal:	Pharmacist can see prescriptions which are not yet realized.
Scenario:	Pharmacist sends his id and id of current patient which are signed by both of their keys. Pharmacist can see only prescriptions which are not yet realized.
Result:	List of prescriptions for the patient.
Database method:	browse_active_prescriptions

5.5 Special Users

There are also defined three other users which cope with the system on special ocassions. These are - administrator, which maintains the system, analytic tools which can be used to obtain statistical data and the goverment authority which has super access to all the data after acquiring proper permissions from court or police.



Actor: Administrator	Title: Central Server maintenance
Goal:	Administrator modifies the database, upgrades software etc.

Actor: Goverment Authority	Title: Retreive Data Of Particular User
Goal:	Goverment Authority (GA) can retreive all sensitive data of every user after showing permission to do so e.g. court order. GA account password can be separated into several pieces to ensure that one attacker won't be in possesion of the key.

Actor: Analytic tools	Title: Obtaining statistics from DB
Goal:	Analyst can query the database for statistical data e.g. number of medicines sold in last month. Analyst can't query patients or link prescriptions data to particular person.

Chapter 6

Communication

Every communication with database can be described by one abstract scenario. First central server and client establish session via SSL. After correct establishment of session, client chooses one of database functions that he can execute with appropriate arguments. Before using methods requiring signatures, client has to ask server for nonce, generated specially for the user. After obtaining the nonce, client can execute selected function. Database verifies the correctness of signature and data passed in arguments and returns the result, or if one of the verification steps failed, error message.

6.1 Connecting to Central Server

1. Enter smartcard with users private key and certificate (or establish paths to them)
2. set path of PostgreSQL to environment variable PATH.
3. in command line write $psql\ 'host = hosts_ip\ port = port\_address\ dbname = database\_name\ user = username\ sslmode = require\ sslcert = user.crt\ sslkey = user.key\ sslrootcert = ca.crt'$ where:
   - *host* - IP of server where database is
   - *dbname* - is the name of database to which we want to connect
   - *user* - name of user which want to connect. Each part will have its own user name.

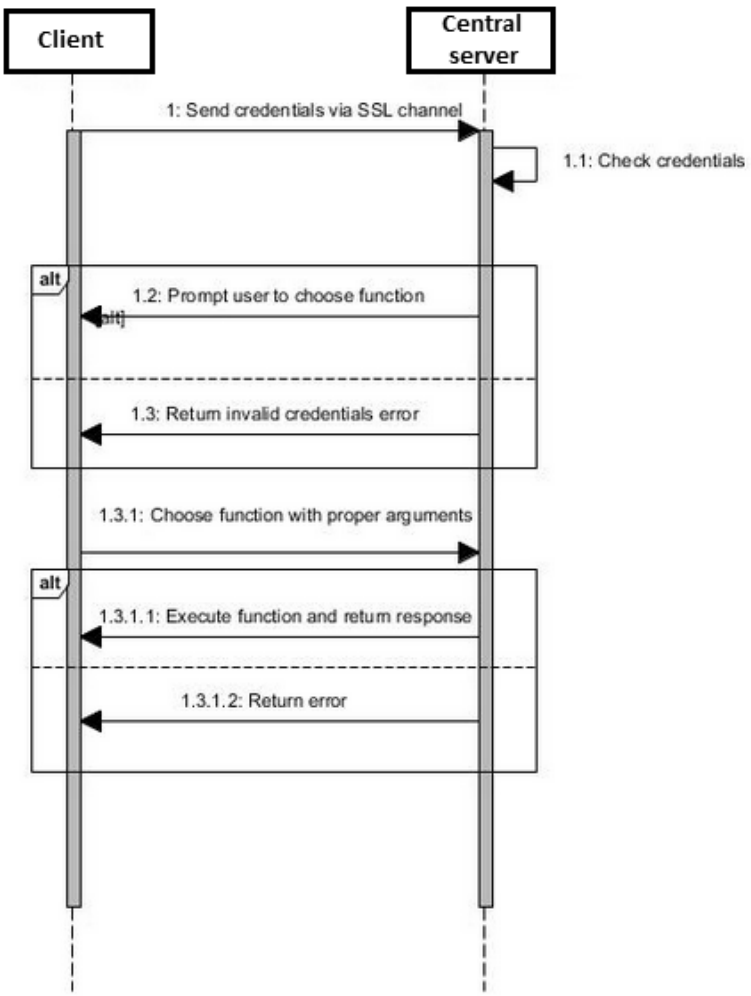


Figure 6.1: General communication diagram for patient, pharmacist and doctor

- *sslcert* - certificate of user.
- *sslkey* - private key of user.
- *sslrootcert* - Certificate of CA.

Example login: $psql\ 'host = 95.85.28.156\ port = 5432\ dbname = PrescriptionSystemMk2\ user = patient\ sslmode = require\ sslcert = patient.crt\ sslkey = patient.key\ sslrootcert = ca.crt'$

4. enter password

6.2 Nonces & Verification Proccess

Randomly generated nonces are part of challenge-response protocol used in communication with database layer. Nonce are security measure against the replay attack. If a request require signature of any party, client has to ask database for generated nonce for given ID. After nonce is return, client has to:

1. Conacatenate function name,
2. function arguments,
3. nonce.
4. Calculate SHA-1 sum over the concatenated elements.
5. Sign the with appropriate key[1].
6. Add the signature as the corresponding argument in function.
7. Send the request.

When the server obtains the request:

1. Takes users key from the database
2. Validates the signature
3. If the signature is validated, constructs SHA-1 sum in the same way as user
4. compares the verified, signed sum with one calculated in previous point
5. Executes the query if the sums are equal
6. Returns the result to the user

[1]Signing method should be equal to invoking openssl command "openssl rsautl sign" with necessary parameters only

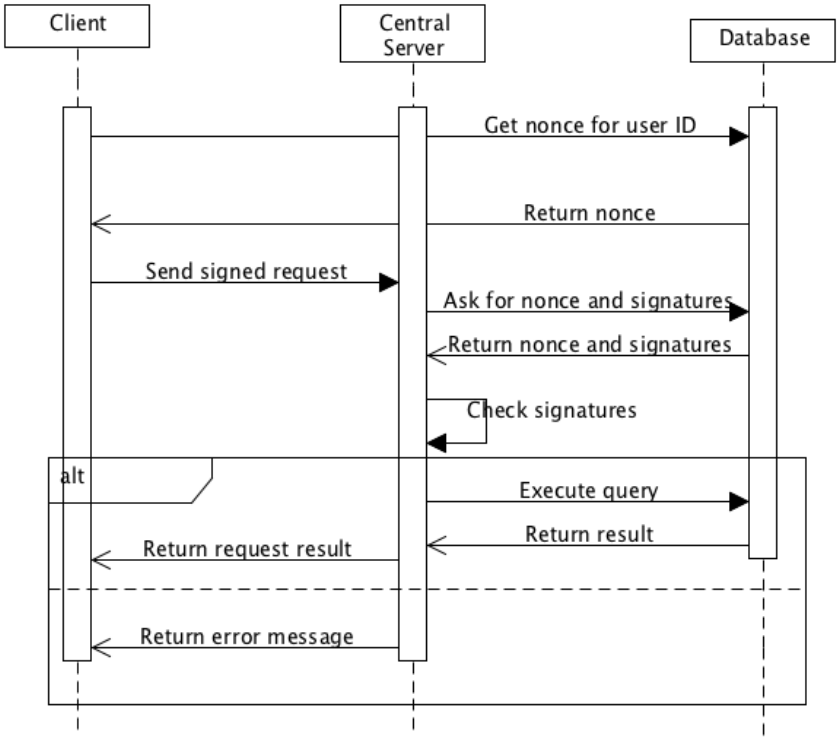


Figure 6.2: Sequence diagram of executing request with nonce signature

Chapter 7

Database Functions and Schema

7.1 Database Functions

After veryfing credentials sent by user to Central Server via SSL secure channel, user, depending on its role, will be able to execute set of functions, which will serve single purpose each (e.g. creation of new prescription).

7.1.1 Shared Functions

	browse_medicines
Arguments:	• name (string, optional, default = None) • type (string, optional, default = None)
Usage:	Pharmacist sends his id and id of current patient which are signed by both of their keys. Pharmacist can see only prescriptions which are not yet realized.

Result:	• medicine_id • name • prescription requirement • medicine type • maximum dosage • unit

Note: Multiple records may be returned at single request.

	browse_doctors
Arguments:	• name (string, optional, default = None) • address (string, optional, default = None) • license_number (string, optional, default = None)
Usage:	Entity using this function performs simple query which return all public data regarding registered doctors stored in DB. Arguments name, adress and license_number narrows down result applying filters to the executed query.
Result:	• doctor_id • name • address • license_number • certificate • public_key

Note: Multiple records may be returned at single request.

	browse_pharmacists
Arguments:	• pharmacist_name (string, optional, default = None) • address (string, optional, default = None) • license_number (string, optional, default = None) • pharmacy_name (string, optional, default = None)
Usage:	Entity using this function performs simple query which return all public data regarding registered pharmacists stored in DB. Arguments name, adress and license_number narrows down result applying filters to the executed query.
Result:	• pharmacist_id • name • address • license_number • certificate • public_key • pharmacy_name

Note: Multiple records may be returned at single request.

7.1.2 Patient Functions

	get_patient_nonce
Arguments:	• patient_id (integer, mandatory)
Usage:	Function returns 1024 bit nonce for given patient_id.
Result:	• nonce
Comment:	New nonce is generated only if the last request was successfully verified.

	browse_my_prescriptions_history
Arguments:	• patient_id (integer, mandatory) • executed (boolean, optional, default = None) • start (date, optional, default = None) • end (date, optional, default = None) • patient_signature (byte, mandatory)
Usage:	Patient requires history of his prescriptions. In order to get access to this kind of data, patient needs to sign his request using his secret key. Next, the signature will be veryfied by database. If signature will be acknowledged as genuine, database will return data about patient prescription history. Database provides patient the ability to filter his history by mean of time span and by the information about execution of prescriptions.
Result:	• prescription_id • doctor_id • doctor name • doctor address • doctor license number • prescription_owner_id • drug id • dosage • max dosage • unit • quantity • execution • time of execution • pharmacy_id • pharmacy_name • pharmacy_adress

Note: Multiple records may be returned at single request.

	transfer_prescription
Arguments:	• patient_id (integer, mandatory) • owner_PESEL (integer, mandatory) • prescription_id (integer, mandatory) • patient_signature (byte, mandatory)
Usage:	Patient changes prescription owner to another patient, therefore allowing him to buy out specific prescription. It is important note, that after changing owner of prescription, original owner is NOT able to buy out his prescription until transfer is cancelled.
Result:	• new_owner_id • "OK"
Comment:	Prescription in database structure has two fields indicating prescription ownership - patientID (non-changeable, indicates the patient to which the medicine was prescribed) and owner_PESEL (patient which will may realize the prescription). Transfering the right will only apply if both of these fields point to same id - thus we exclude the scenario when patients can pass the prescription to yet another person. After this operation the transferring patient losses right to realize the prescription - prevention from cloning the prescription.

	cancel_prescription_transfer
Arguments:	• patient_id (integer, mandatory) • prescription_id (integer, mandatory) • patient_signature (byte, mandatory)
Usage:	Patient changes actual owner of his prescription back to the original one (the patient himself) allowing him to buy out prescription and disallowing former owner of prescription to do so.
Result:	• "OK"

Comment:	Prescription in database structure has two fields indicating prescription ownership - patientID (non-changeable, indicates the patient to which the medicine was prescribed) and ownerID (patient which will may realize the prescription). Cancelling will only work if patientID and ownerID are different and the signature over request is verified.

7.1.3 Doctor Functions

	get_doctor_nonce
Arguments:	• doctor_id (integer, mandatory)
Usage:	Function returns 1024 bit nonce for given doctor_id.
Result:	• nonce
Comment:	New nonce is generated only if the last request was successfully verified.

	create_prescription
Arguments:	• doctor_id (integer, mandatory) • patient_id (integer, mandatory) • drug_id (integer, mandatory) • dosage (integer, mandatory) • unit (integer, mandatory) • quantity (integer, mandatory) • doctor_signature (byte, mandatory)
Usage:	Doctor prescribe single medicine to the patient, describing medicine, quantity and dosage.
Result:	• "OK"

Comment:	Database does not requires patient signature to create a prescription for him - Prescription realization will require his key (thus his smartcard) so the medicine can't be bought without his knowledge. Also the doctor can create prescription without the need of meeting the patient face to face - which is and advantage for chronically ill patients.

	browse_patient_prescription_history
Arguments:	• doctor_id (integer, mandatory) • patient_id (integer, mandatory) • start (date, optional, default = None) • end (date, optional, default = None) • bought(boolean, optional, default = None) • doctor_signature (byte, mandatory) • patient_signature (byte, optional)
Usage:	Doctor downloads patient prescription history. Doctor (unlike pharmacist) do not needs patient signature to browse history od prescription that he has created. If he wants the full history, patients signature is needed.

Result:	• prescription_id • doctor_id • doctor name • doctor address • doctor license number • prescription_owner_id • drug id • dosage • max dosage • unit • quantity • execution • time of execution • pharmacy_id • pharmacy_name • pharmacy_adress
Comment:	If the patient signature is missing, database will only return prescriptions which were created by the doctor. If the patient signature is present and can be verified, doctor will receive the full history of patient. In case of any errors on verification, the request will be canceled.

Note: Multiple records may be returned at single request.

7.1.4 Pharmacist functions

	get_pharmacist_nonce
Arguments:	• pharmacist_id (integer, mandatory)
Usage:	Function returns 1024 bit nonce for given pharmacist_id.
Result:	• nonce

Comment:	New nonce is generated only if the last request was successfully verified.

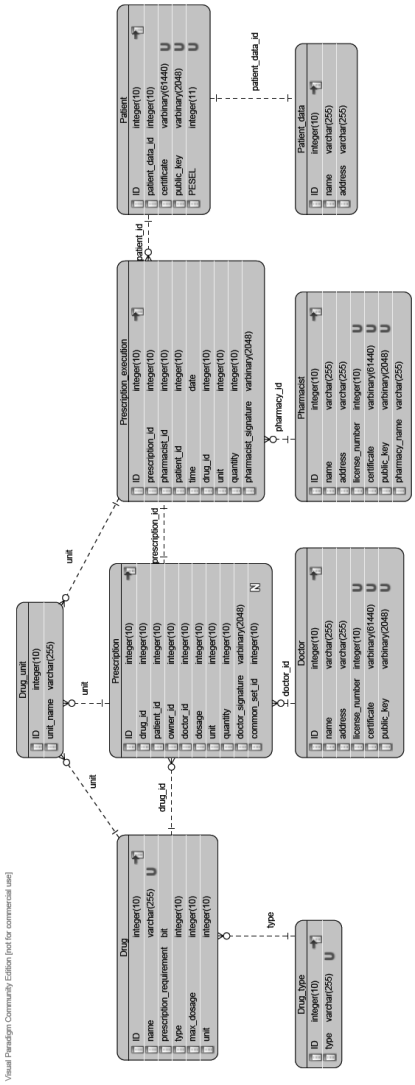
	prescription_realization
Arguments:	• prescription_id (integer, mandatory) • pharmacist_id (integer, mandatory) • drug_id (integer, mandatory) • unit (integer, mandatory) • quantity (integer, mandatory) • pharmacist_signature (byte, mandatory) • patient_signature (byte, mandatory)
Usage:	Pharmacist will be able to realize patient prescription by pointing right prescription by giving its id, choose proper medicine (not necessairly the same as medicine prescribed by doctor, this check will be done by database), describe how many medicine is sold.
Result:	• "OK"
Comment:	Request has to be signed by both patient's and pharmacist's keys. If the signature is incorrect, the database will return error message and the medicine shouldn't be given away.

	browse_active_prescriptions
Arguments:	• pharmacist_id (integer, mandatory) • patient_id (integer, mandatory) • pharmacist_signature (byte, mandatory) • patient_signature (byte, mandatory)
Usage:	Pharmacy is able to see all active (not bought) prescriptions of current patient, which agrees to show this data to the pharmacy by signing request.

Result:	• prescription_id • doctor_id • doctor name • doctor address • doctor license number • prescription_owner_id • drug id • dosage • max dosage • unit • quantity • execution • time of execution
Comment:	If the signatures of patient or pharmacist are incorrect, database will return an error. If there are no non-realized prescriptions, database will return empty list.

Note: Multiple records may be returned at single request.

7.2 Database schema



Chapter 8

Central Server Security Standards

8.1 Physical Security

- Servers is protected by backup and offsite data storage. The offsite storage of backup media is in a secure backup-vendor secure facility.
- A facility with Uninterruptible Power Supply (UPS) supporting all servers and essential peripheral equipment (console servers, etc).
- A facility with a climate controlled environment separate from the building HVAC, (dedicated air conditioning with in-room temperature controls).
- A facility with cooling and electrical capacity that is planned and monitored for outages.
- Secured access to the facility with documentation listing all individuals who currently have access and monitoring/auditing of ingress/egress via staff/video/etc.
- Servers in the facility must require authentication for local access (i.e. consoles are not left logged in while unattended).
- For facilities that use access codes, the capability to quickly change the access codes if personnel changes warrant is required. Access codes must be changed at least annually.
- A facility with automated fire detection and suppression systems.

8.2 Data Encryption

- Hard disks, on which are stored databases, will be encrypted by external program TrueCrypt. TrueCrypt encrypts whole data on hard disk in real time.
- Databases will be encrypted by TDE (Transparent Data Encryption). TDE encrypts:
  - Database files
  - Database Snapshots
  - Transaction Log File
  - Backups

  using DEK (Database Encryption Key) which is protected by certificate.

8.3 Backup Procedure

- To ensure no data loss, database is replicated in real-time to a server in another location - this location meets conditions mentioned in section 5.1.
- Additionally, regular backups are made every day.
- Backups are kept for reasonable amount of time:
  - Daily backups - 1 week
  - Weekly backups - 1 month
  - Monthly backups - 1 year
  - Annual backups - forever
- All backups are encrypted with measures described in section 5.2.

Part IV

Pharmacy Module

Chapter 9

Data Flows

9.1 Use Cases

The way prescriptions are currently processed is vulnerable to many threats, and brings many inconveniences. The most important ones are listed below.

1. **System:**
    - **pharmacist's verification** - system is ableto check that pharmacist has permissions to sell the drugs;
    - **buyer's verification** - system is able to check that the buye's card is valid and entered PIN number was correct;
    - **prescriptions update** - system can change the state of prescriptions(to either 'bought' or 'invalid') or attach additional info to them, like the fact that drug's substitute was sold instead of prescribed one;
2. **Pharmacist:**
    - **reading available prescriptions** - a pharmacist is able to seebuyer's prescriptions
    - **modifying the prescriptions** - a pharmacist is able to update the prescriptions (changing their state/attaching info that substitute was sold instead)

- **signing the prescriptions** - a pharmacist is able to sign prescription to confirm that he's the one who sold them

3. **Customer:**

- **reading available prescriptions** - a customer is able to see/select prescriptions that haven't yet been bought
- **confirming pharmacist's changes** - a customer is obliged to confirm possible changes made to the prescriptions by the pharmacist
- **signing the prescriptions** - a customer is able to sign prescription to confirm that he got the certain medicines

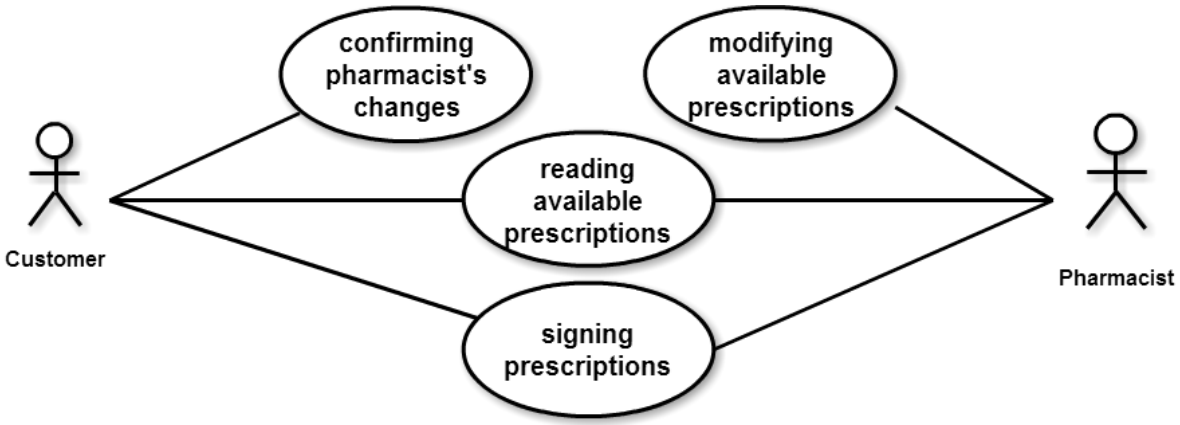


Figure 9.1: Patient's and pharmacist's use cases

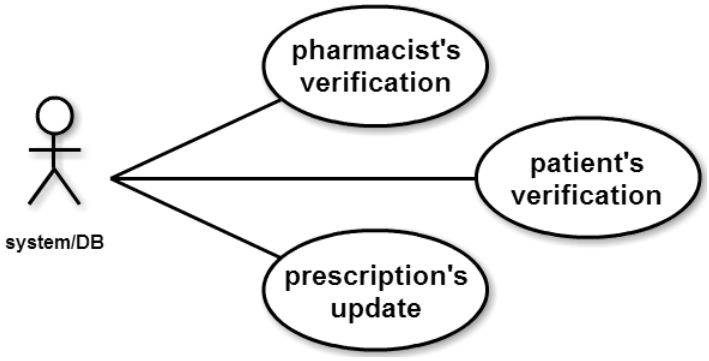


Figure 9.2: System's use cases

9.2 Scenario

1. Customer inserts his card into the reader and enters PIN number.
   (a) System checks whether PIN is correct (if it is not, an appropriate message is displayed and the process cannot be continued).
2. Terminal displays list of active prescriptions to both buyer and pharmacist.
3. Buyer selects prescriptions to buy.
4. Pharmacist inserts his card into his reader and authenticates himself to the system (assuming that the card is not already inserted).
   (a) If authentication is not possible (eg. card of the pharmacist is invalid), an appropriate error message appears on the screen and the process can't be continued.
5. The pharmacist marks prescriptions selected by the customers as 'to be bought'.
6. System checks whether prescriptions have already been bought.
7. System verifies validity of prescriptions (expiration date, credentials of the doctor etc.)
   (a) If some prescriptions are invalid, an appropriate message appears on the screen and system marks the prescriptions as 'invalid'.
8. If the drug from the prescription is not available (or the buyer does not want it for some reason), pharmacist can instead sell a substitute. For that, he is able to write information about selling a substitute to the system.
9. Buyer confirms the prescriptions to be bought (including possible substitute replacements).
10. Pharmacist gives the drugs to the buyer, confirms the selling and the system marks the prescriptions as 'bought'.
11. Buyer takes the drugs and removes his card from the reader.

If the customer's or pharmacist's card is removed from the reader before the step 10, the process is aborted and the initial state of the prescriptions is not changed.

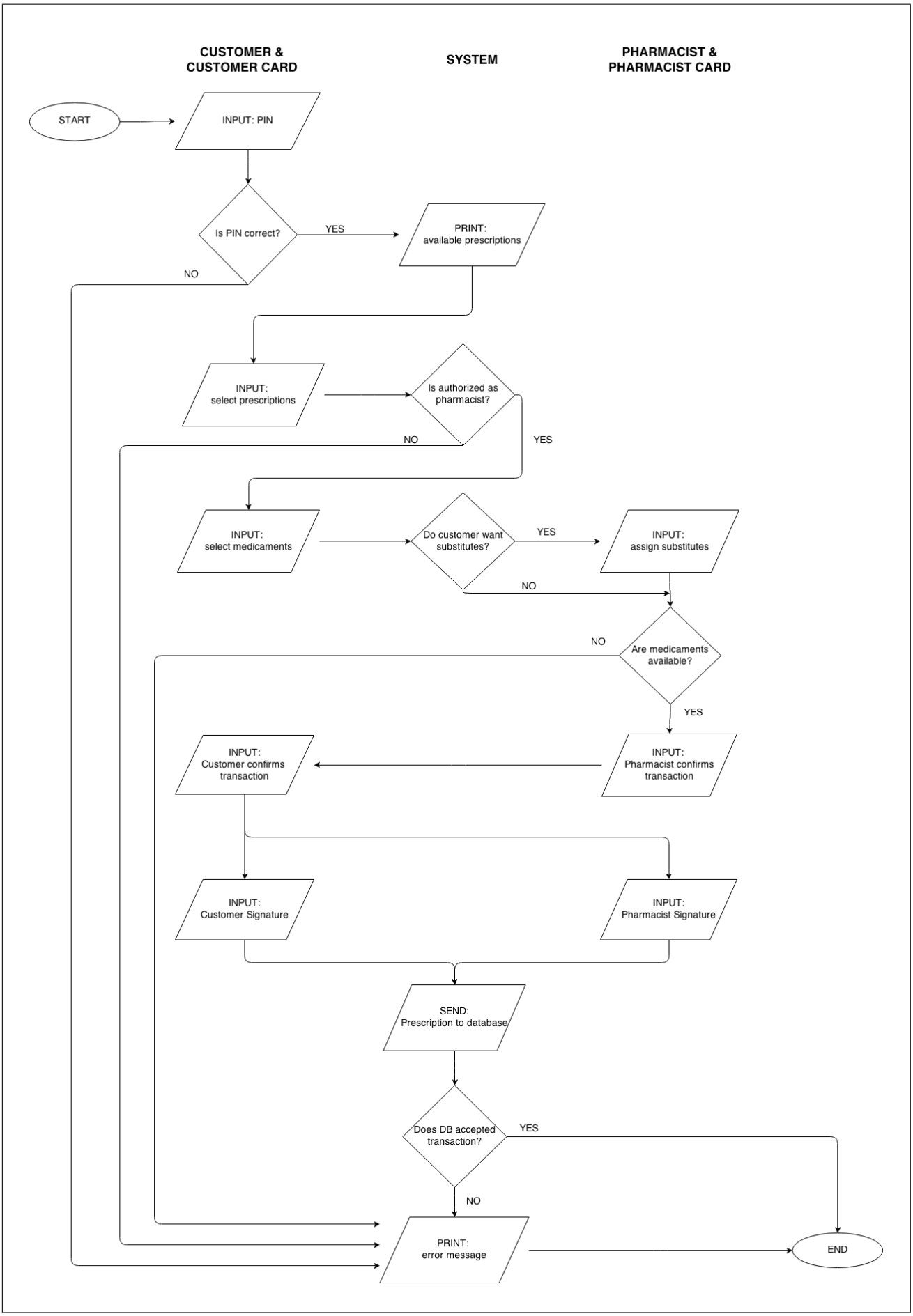


Figure 9.3: Flow chart

Chapter 10

Sequence Diagram

In his chapter we present sequence diagram of the actions performed in the range of Pharmacy Module. Each step is described in details. Not all the actions are obligatory, i.e. some procedures can be performed or omitted depending on the required security level and a budget.

10.1 Communication initialization

The first step is communication initialization. Actions performed in this step by the system elements are presented in the figure 10.1

At the beginning, a patient puts his personal card to a terminal and he enters his PIN as usual, e.g. in the ATM. If the PIN is correct, the user can see appropriate message on the terminal screen. Also a pharmacist have to use his card and enters his PIN in the second terminal. Then, the system is ready to work.

PINs are preventing from unauthorized usage of cards, e.g. when a card was stolen or lost.

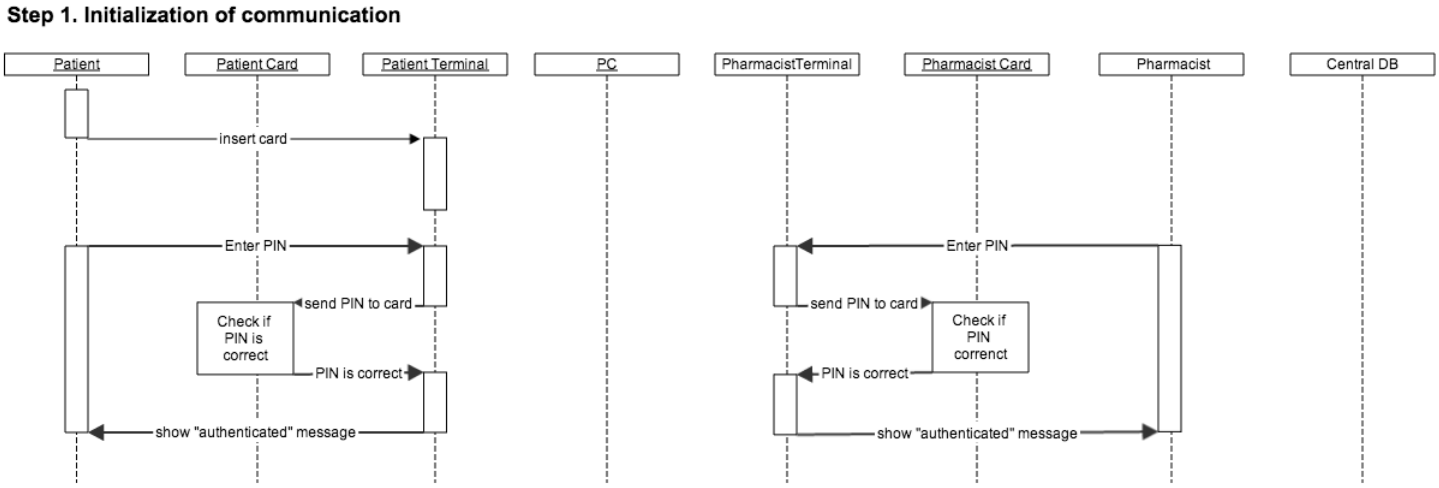


Figure 10.1: Sequence diagram - step 1

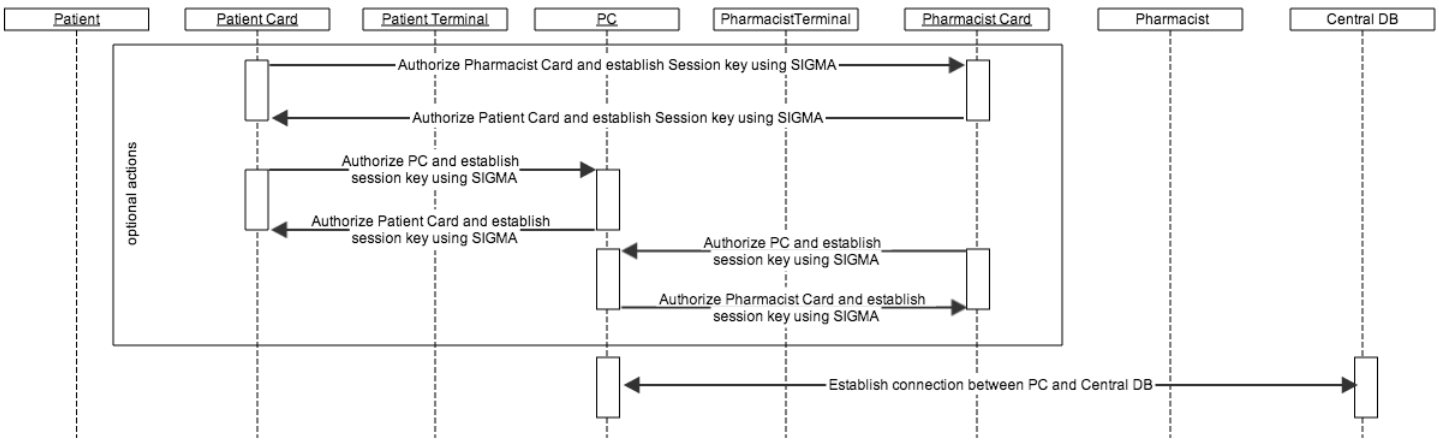


Figure 10.2: Sequence diagram - step 2

10.2 Establish a secure communication

The second step, presented on the figure 10.2, contains actions related with establishing secure communication between the system parties. Part of the actions marked there, are optional and are not required for the system to work properly. Establishing a secure communication between the cards allows the participant to be sure, that the patient's and pharmacist's cards are not forged and they are authenticated to each other. Similarly, suing the SIGMA protocol between a card (patient's or pharmacist's) and the application installed on the PC, allows to authorize the application by the card and the card by the application. These two sub-steps can be implemented, if a very-high level of the security is required.

The communication between the application on the PC and the Central Database is performed in the way described in the Central Database Module Documentation.

10.3 Select prescription to buy

The figure 10.3 presents a point in the protocol, in which user's prescriptions are downloaded from the Central Database and are shown on the screen. After that the patient selects one or more of them to realize them. User's identification data are stored on his card. They are used to authenticate the patient and to download appropriate prescriptions.

10.4 Realize prescription

The last step is presented on the figure 10.4. This scheme is repeated for the each prescription. At the beginning, the system shows available substitutions for the medicine. Then, the pharmacist can select original medicine or one of the substitutions and the patient can confirm this choose.

Then, the application ask the patient and pharmacist cards to sign selected data. After it receives a response, it sends this signed data to the Central Database. The data

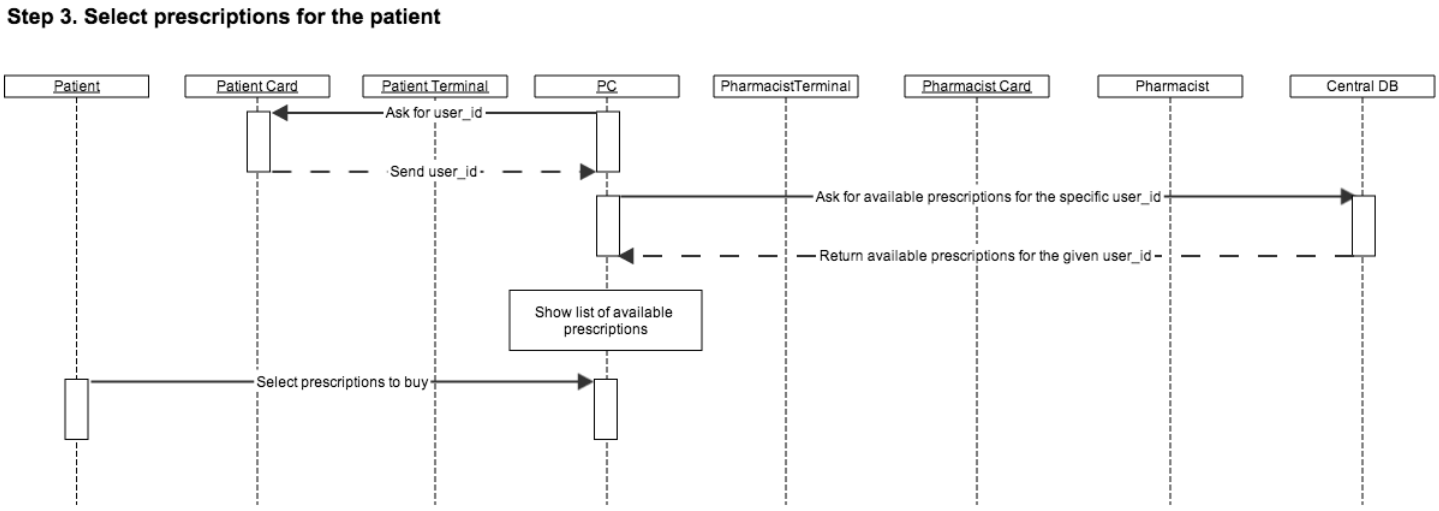


Figure 10.3: Sequence diagram - step 3

are saved there. Because of that, it is impossible to simulate buying process, without patient's personal card. The prescription's data have to be signed by the patient to be inserted into a database as a bought prescription. Without a valid insert, the refund will not be granted.

10.5 End of the protocol

At the end of the protocol, the communications channels are closed and all ephemeral keys are destroyed.

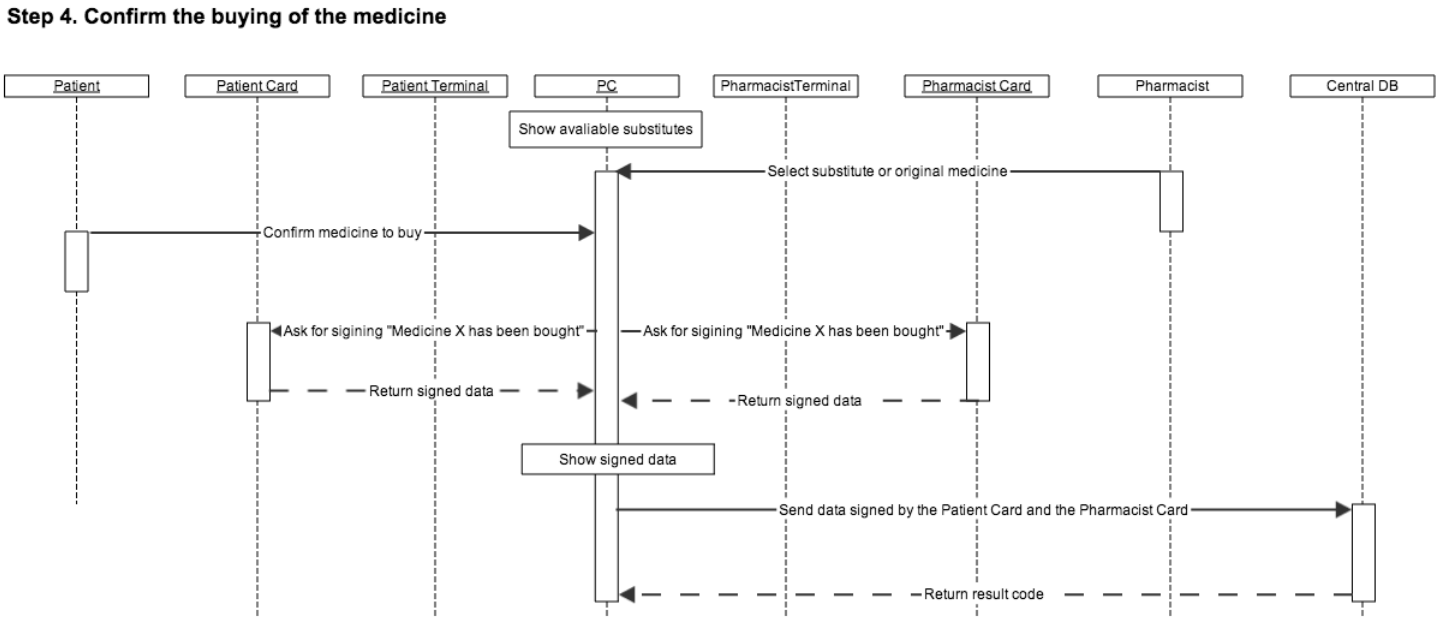


Figure 10.4: Sequence diagram - step 4

Part V

Patient

Part VI

Doctor