

The Bucharest University of Economic Studies

The Faculty of Cybernetics, Statistics and Economic Informatics

Economic Informatics Specialization

Secure Blockchain-Based Application for Managing Electronic Health Records

Bachelor Thesis

Advisor:

Prof. univ. dr. Răzvan Daniel Zota

Graduate:

Lupșan Sabrina

Bucharest 2021

Table of Contents

[List of abbreviations - 2 -](#_Toc75195685)

[1. Introduction - 4 -](#_Toc75195686)

[2. The blockchain technology - 6 -](#_Toc75195687)

[2.1 What is Blockchain - 6 -](#_Toc75195688)

[2.2 Types of Blockchain: a comparison - 7 -](#_Toc75195689)

[2.3 Why private blockchain - 8 -](#_Toc75195690)

[3 Cybersecurity - 10 -](#_Toc75195691)

[3.1 The rise of threats - 10 -](#_Toc75195692)

[3.2 The CIA and AAA models - 11 -](#_Toc75195693)

[3.3 Encryption - 13 -](#_Toc75195694)

[3.4 Integrity with hashing - 14 -](#_Toc75195695)

[3.5 SQL Injection - 15 -](#_Toc75195696)

[3.6 Proof of work and DoS - 17 -](#_Toc75195697)

[4 Windows Application - 18 -](#_Toc75195698)

[4.1 .NET Framework, Windows Forms platform and C# Programming Language - 18 -](#_Toc75195699)

[4.2 Microsoft Azure with SQL - 18 -](#_Toc75195700)

[5 Using Blockchain to create, view and manage EHRs – an application - 20 -](#_Toc75195701)

[5.1 The classes - 21 -](#_Toc75195702)

[5.3 Navigating through the application - 23 -](#_Toc75195703)

[5.3 Logging - 24 -](#_Toc75195704)

[5.4 File encryption - 28 -](#_Toc75195705)

[5.6 The database - 29 -](#_Toc75195706)

[5.6 Design - 32 -](#_Toc75195707)

[5.7 Code obfuscation - 32 -](#_Toc75195708)

[6 Conclusions and future work - 33 -](#_Toc75195709)

[6.1 Conclusions - 33 -](#_Toc75195710)

[6.2 Future work - 34 -](#_Toc75195711)

[Bibliography - 36 -](#_Toc75195712)

[List of figures - 40 -](#_Toc75195713)

# List of abbreviations

|  |  |
| --- | --- |
| 2FA | 2 Factor Authentication |
| 3DES | Triple Data Encryption Standard |
| AAA | Authentication, Authorization and Accounting |
| AES | Advanced Encryption Standard |
| API | Application Programming Interface |
| CEO | Chief Executive Officer |
| CERT-RO | Romanian National Computer Security Incident Response Team |
| CIA | Confidentiality, Integrity and Availability |
| CLR | Common Language Runtime |
| COVID-19 | Coronavirus disease 2019 |
| CPU | Central Processing Unit |
| DBMS | Database Management System |
| DES | Data Encryption Standard |
| DoS | Denial of Service |
| DDoS | Distributed Denial of Service |
| DLL | Dynamic-Link Library |
| ECC | Elliptic Curve Cryptography |
| GUI | Graphical User Interface |
| IP | Internet Protocol |
| IPSec | Internet Protocol Security |
| IT | Information Technology |
| IV | Initialization Vector |
| LANMAN | LAN Manager |
| MD5 | Message Digest Algorithm 5 |
| PoA | Proof of Authority |
| PoET | Proof of Elapsed Time |
| PoR | Proof of Reputation |
| PoS | Proof of State |
| PoW | Proof of Work |
| PIN | Personal Identification number |
| RFC | Request for Comments |
| RSA | Rivest–Shamir–Adleman |
| SHA | Secure Shell Algorithms |
| SMTP | Simple Mail Transfer Protocol |
| SQL | Structured Query Language |
| SQLI | Structured Query Language Injection |
| SSH | Secure Shell |
| SSL | Secure Sockets Layer |
| TCP | Transmission Control Protocol |
| TLS | Transport Layer Security |
| T-SQL | Transact-SQL |

## Introduction

On the 10th of September 2020, the first victim of a cyber attack was recorded. A ransomware attack hit The University Hospital of Düsseldorf and temporarily affected thirty servers with patients’ data. [1] Consequently, a patient had to be transferred to another hospital but died in transit, making them the first casualty of a cyber security attack. Moreover, this happened during the COVID-19 pandemic, when hospitals were used at full capacity, and raised the alarm that cyber-attacks should be taken more seriously and properly addressed before giving criminals the chance to exploit vulnerabilities. Attacks in the medical system, compared to attacks in other industries, can go beyond material damages.

A study conducted by the Romanian National Computer Security Incident Response Team in 2020 uncovered 7670 vulnerabilities in hospitals and clinics across Romania. Out of those, 1337 were considered to pose a high risk, 4241 were deemed medium-risk and 2092 were low-risk. [2]. These vulnerabilities may range from computers with old or unpatched operating systems, missing hardware, the use of common or simple passwords, and others. The systemic lack of investment in the cybersecurity infrastructure is now brought to light as more and more attackers find the medical system an area of interest.

Health records contain sensitive information about patients and can become very dangerous vectors of attacks when fallen into the wrong hands. For example, a spear-fishing attack (meaning, an attack directed to a specific person with the means of tricking them into paying an amount of money, downloading a document, or doing a certain action favourable to a hacker) can be very convincing if the attacker would possess key information such as the name of a medicine a patient urgently needs to buy. Such phishing attacks can be very dangerous, especially for elders, who are not aware of the sophistication of the cybernetic attacks that are being conducted nowadays and do not know how easily hackers can perform successful data theft. Deception schemes are more and more creative and effective and should be stopped from the root, before the attacker even obtains access to confidential information. Therefore, several protection layers should be used to properly secure an application used for such high sensitivity data such as medical charts, appointments, future surgeries, blood tests.

The scope of this paper is to present a secure solution for the management of electronic health records, which does not allow the modification or deletion of records or permit unauthorized persons to access personal data. Several cybersecurity principles such as AAA (Authentication, Authorization and Accounting) and CIA (Confidentiality, Integrity and Availability) were implemented as well as other security practices like file encryption, password hashing, code obfuscating and others. The featured solution presents a secure implementation of a blockchain-based application that stores the database data on the cloud to assure high availability and protection. The application has a user-friendly interface but modern technologies in the background and is designed to be used by medical staff (doctors, nurses) as well as patients of that hospital.

A secure application however is not enough and should go together with personnel training, awareness-raising campaigns and the constant reminder of good practices for every user of an application. However good multi-factor authentication is or how many protection layers are used in a network if a person discloses their credentials to other individuals their personal data is no longer safe. Social engineering is based on human interaction and should not be disregarded as not significant; it is just as important, if not more, than technical measures. Attackers use panic or fear-inducing scenarios to convince their victim to provide information or money as fast as possible, without letting them properly think or ask professionals. [3] This is especially dangerous and threatening in the health domain, where fear, sadness and stress are very present emotions and can be used as attack vectors, since ill-intended people do not refrain from using medical conditions, accidents, and surgeries as a pretext.

## The blockchain technology

## What is Blockchain

Blockchain is a technology that defines a secure, immutable solution to store data in a database. Every row is a block chained together with the other blocks in chronological order; the link between the rows, or the chain, represents the hash of the previous block, which is computed using all the fields from that record. This feature provides the owner or the participants of the blockchain with a way to check the integrity of the entire chain, block by block, and to identify the exact point if a change were to happen. The immutability of this technology can be very useful in the healthcare sector, where medical charts should not be changed or erased after they are issued.

It is a secure storage technology for sensitive information because it disallows the modification or deletion of already-added records, as well as forbids rogue nodes to contribute to the database. Because every block is linked to the previous one, if one change is performed at any point of the blockchain, all the nodes after it change; in this way, if an attacker were to make a modification in the blockchain, they would have to recompute the entire chain of blocks before any node rejects the records. This is not computationally feasible if the difficulty of the blockchain is set at an appropriate value, high enough to enable security, but low enough to not cause great performance issues for the users.

The difficulty of a blockchain represents the complexity of the hashing rule used to compute the block hashes. For example, let’s say that the rule is the hash of every block has to start with at least one 0. A node that tries to add a block with a hash that does not obey the rule is declined; this means that the hash needs to be recomputed several times until the desired value is reached. Now let’s say that the rule becomes more complex; after that 0, two 1’s and one letter need to be present in the hash. The difficulty also increases, because it would take the participating node exponentially more time to compute a fitting hash value. This difficulty can increase with time; the higher it is, the more secure the blockchain is. However, this means that more resources have to be allocated, such as time and CPU; this is a trade-off between security and resources.

The most popular implementation of a blockchain is Bitcoin, but not limited to that, because other cryptocurrencies use this technology as well. These are public blockchains that store the data in a decentralized manner where all the nodes involved in the transactions can contribute to the blockchain in real-time. In this way, a rogue node cannot change the blockchain because at least 51% of the blocks of the chain would have to be recomputed and replaced in the database, in order for the block to be considered as valid. In the case of Bitcoin, the data represents the transactions performed. [4]

Centralized storage can also be available under the blockchain technology. This solution has a considerably lower cost and will be assessed in the following chapter, where the advantages and disadvantages of both public and private blockchain will be weighted against the final goals of the presented project.

## Types of Blockchain: a comparison

There are two types of Blockchain: public and private. Both types serve different needs and provide different measures of security. There is a big debate on which blockchain is better and, after analyzing the characteristics of both, I decided which type is more suited for this application’s goals.

Firstly, the problem of decentralization of data: in a public blockchain, the data is not stored on a server, in a single place, but rather everywhere: anyone that follows the rules imposed can step in and verify, add and read the data on the blockchain, as well as store it. It is also not owned by a single entity. This is the case for the Bitcoin cryptocurrency. On the other hand, in a private blockchain, decentralization cannot be kept. However, only trusted entities can participate in its manipulation. This capability of the private blockchain increases access control, because not anyone can participate in the actions specific to a blockchain, but introduces the problem that if a rogue node gains the trust of the program in any way, the blockchain is compromised. This access can be gained, for example, by obtaining credentials to access the application; this is why security measures should heavily accompany the decentralized variant and act as a supplicant to the centralized version’s features.

Secondly, a public blockchain’s security increases as the number of peers that take part in it increases. This introduces the problem of scalability since the transactions can only occur at a slow pace. As a result, every time an entity wants to contribute with a block, it must be approved by everyone in the blockchain. This aspect can severely impact performance, which is not the case for a private blockchain. In the permissioned one, the number of authorized nodes is much smaller and established only once, therefore the data is processed much faster, even if the number of nodes increases by a lot. [5] High responsiveness is critical in the healthcare sector, where medical charts have to be delivered very fast, especially if there are medical emergencies.

Thirdly, the kind of data stored in hospitals is very sensitive and highly classified and no one should read the information without permission. This is a big impediment in the case of the public blockchain, as this version stores it openly and transparently. In this case, the permissioned technology would be more appropriate for the storage of medical charts and documents. [6]

Lastly, a private blockchain is prone to authenticated vulnerabilities. If a hacker would get access to a trusted entity’s credentials, they would have access to that person’s data and rights and could further take advantage of that to escalate their privileges. This solution raises security issues that do not cause problems in the unpermissioned version. The latter one can only be attacked if over 51% of the participants in the transactions would have the same ill-intention. In order to fully attack a blockchain, all of the blocks have to be tampered with and the proof of work should be completely redone for the entire blockchain. In most of the cases this cannot be accomplished neither it is feasible to try to do it. [7]

## Why private blockchain

After considering the advantages and disadvantages of both permissioned and unpermissioned blockchains, I have decided that the private version of this technology would be more fit for the application presented in this paper and its main goals. I believe this would accommodate all the needs for an application for the healthcare system made to be used by patients and doctors from a hospital.

One key capability I considered is the invitation-only aspect of the private variant, which allows the trusted entities to keep the privacy of data in an enclosed circle, which permits an organization to only include its employees, suppliers and clients. Otherwise, in a public environment, the transactions would not be kept private. Acknowledging this aspect, a private blockchain is a secure tool to manage sensitive data in an enterprise. In this specific case, the only trusted nodes would be the doctor, the patient and the administrators, who would have credentials-based access by logging into the application with multi-factor authentication. Establishing the trusted parties in a safe environment closes the circle of trusted nodes and assures privacy within it, with the premise that the credentials used to authenticate are only known by the owner and no one else.

There is also the possibility of implementing different levels of access and a different set of transactions for entrants. For example, in a company, an employee would have different rights and needs than a client. [8] This can be achieved by creating different profiles and assigning them correctly to participants. The presented application has different levels of access for the patients, the doctors and the administrator. The patients can only see their own medical records, the doctors can access the medical records of their patients and the administrator is only responsible with registering new patients or doctors into the system and assuring the database backup is done regularly, but without actually accessing the private records of the users of the application. Different levels of access are very important in order to limit the amount a user can see or alter to exactly what they need.

Another decisional factor in differentiating the two blockchains is the cost of the technology. In an unpermissioned implementation, validation and proof of work are essential. Legitimizing transactions in a public blockchain is time-costly because every entrant is considered untrusted. In the private version, this is exactly the opposite, as the contributors to transactions are already trusted nodes, thus speeding up the process of reading and writing to the database. As the blockchain grows more and more, the difference in the quality of service provided is even more visible between the two options, and using the public variant can significantly impact the performance of the presented solution. Choosing the private blockchain assures scalability and good performance, because no matter how many new users are added, the performance of the application is not affected, compared to the public blockchain.

However, while having trusted nodes access their rights by authenticating, even if this can be heavily layered with security features, it is still an increased risk; I consider this the biggest drawback of the private version: the fact that it is credentials-based, which means that anyone that has access to credentials of a trusted person (that can be obtained through phishing, for example) or gains the trust of the already-existing participants can access the database records. As mentioned before, in order to mitigate this kind of threat, awareness programs and training should be conducted not only for the employees of the hospitals but also for the patients, who might not be knowledgeable about this kind of threat and how easy it is to be tricked by hackers.

Taking into consideration the risks and the benefits of this type of blockchain, many companies have adopted the discussed technology. Walmart, Spotify, DHL are just a few examples of the giants that acquired a permissioned blockchain. [9] I conclusioned that the private blockchain would be an appropriate solution for this project, and, knowing the possible vulnerabilities of the current plan is a good place to begin designing and implementing an application.

## Cybersecurity

## The rise of threats

Malware is software designed to harm applications and devices and comes in many different types: trojans, ransomware, adware, viruses, spyware etc.

Ransomware is one of the most common types of malware. It is malicious software that encrypts files on a host or multiple hosts with the intent of receiving a ransom from the victim in exchange for the key that would decrypt those files. However, after receiving the ransom, the attacker may or not provide that key, making this type of cyber attack extremely impacting. Paying the ransom is strongly discouraged because it does not guarantee that the encrypted files will ever be recovered and encourages the perpetrator to seek additional victims. This type of malicious software can be found in 7 out of 10 malware payloads. [10].

Another type of widely used cybernetic attack is a less technical one that relies fully on human error and a lack of proper security training: phishing. It is typically used as a transmitter tool to deliver malware payloads to users, and it accomplishes this by deceiving users to click, download something or to provide authentication credentials or sensitive information such as bank account information. This type of attack is usually extremely efficient if the perpetrator has any inside information about the victim they are attacking. For example, if the attacker would have information about any type of medical condition the victim has, they could easily falsify an email or a call offering promotions for the medicine and pretending they are hospital employees.

In the following chart, based on data collected for the period 2008-2019, we can see that the number of malware infections started increasing at a rapid rate from 2012 ($82.62 million) and have continued to increase with a slight decrease in the growth rate.

Figure 1: My own data processing based on data collected from Purplesec [59]

The information security landscape changes constantly. The impact of these intrusions also evolves – theft is replaced by damaged servers and integrity disruption.

## The CIA and AAA models

In order to deal with the constant rise of threats and the numerous types of complex attacks, cybersecurity provides two models to guide professionals. The CIA model represents the principle that an IT component should provide the following characteristics: Confidentiality, Integrity, and Availability. The AAA model provides the means to achieve the goals of cybersecurity described in the CIA model. The triple-A concept refers to Authentication, Authorization, and Accounting. [11] These models are the baseline of a secure application.

**CIA Model**

The first goal in the CIA model expresses the need for privacy of the data. Protecting data over the internet is crucial and the first step in a hacker’s malicious plans includes gathering sensitive information about their victim. Therefore, an attack can be stopped from the reconnaissance stage if an attacker cannot gain confidential information. This element of the model can be assured using data encryption, both in transit and while stores, and access control, among others.

Integrity assures that the original data is not tampered with by a third party. This function of the CIA model establishes the rule that someone unauthorized should not be allowed to alter or delete information. Integrity can be achieved through hashing, which is the process of introducing data in a hash function that produces a unique output for every input. If you have the initial hash, you can compare it against the hash of the data you have; if they do not match, then there is no data integrity and it should not be trusted.

The last principle in the CIA model is Availability. This component assumes that the owner or users of some application or data can access them and the needed resources whenever they want. DoS (Denial of Service) or DDoS (Distributed DoS) attacks disrupt the availability function and are built to bring down a system and block the users from accessing the needed resources. Availability can be provided when a system maintains redundancy, fault tolerance (which means not having a single point of failure that, if compromised, can bring the whole system down with it), access lists, etc. [12]

**AAA Model**

The first letter in the triple-A model stands for Authentication and describes the way a user can be identified in an information system. Authentication serves to uniquely identify a user on the internet through the use of secret credentials. Experts strongly advise using multi-factor authentication along with very strong passwords in order to prevent unwanted individuals from accessing someone’s account just by guessing the password. Multifactor authentication, or its more popular version, Two-factor authentication (2FA), is implemented by using a combination of something you know, you have (a key), or you are (such as fingerprints). Multifactor authentication can be achieved by combining at least two categories and is more secure than simple authentication. For example, is more difficult to both obtain a secret password and a temporarily generated code on that person’s personal device.

The second A in the mentioned model is Authorization. Authorization can be accomplished by implementing access control and limiting the resources a user can access based on their role in the organization. To achieve the best result, users should only be given the minimum necessary permissions. This rule prevents leakage of sensitive data or limits the impact of malicious activity carried out by authorized personnel, preventing privilege escalation.

The last element of the AAA model is Accounting. Keeping records of what every individual does not only holds them accountable in case a suspicious action or a cybersecurity incident occurs but also discourages users from doing anything they desire in the organization. Accounting can be realized by logging the activities of individuals that can later be accessed for forensics. [11] Accounting is also helpful in case an attack happens and professionals want to find the source of the disruption.

By implementing the above-mentioned principles, the impact of cybernetic attacks can be limited to the point of stopping them.

## Encryption

Encryption is the process of altering input data in order to make it unreadable and only allow authorized access to the data. It is a very used tool that prevents unwanted access to sensitive data.

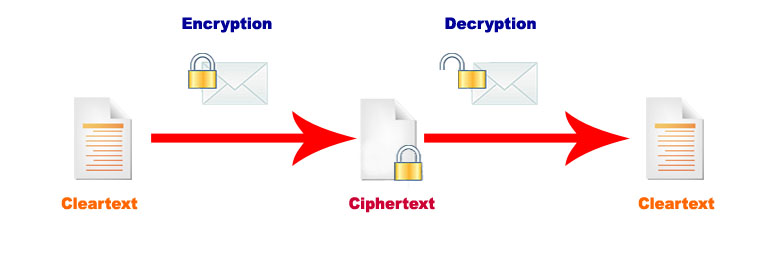


Figure 2: The encryption and decryption process [13]

There are several categories of encryption. Two main types are symmetric and asymmetric encryptions. The first one involves a key that is used both for encryption and decryption and focuses more on safely managing the existing key. It is less costly because it doesn’t involve changing information back and forth. However, this is not the case with asymmetric encryption. In the latter one, there are two keys: one public and one private. Usually, the public one is used to encrypt the information and the private one is used to decrypt it. The private key is available only for authorized users. This variation is more costly, however, but is more effective.

3DES (Triple Data Encryption Standard) is a symmetric encryption algorithm that consists of a variation of the original DES algorithm, which is no longer secure because of its small key size. 3DES is improved because it encrypts the data three times (hence the name) with a key length of 192 bits. It encrypts data in blocks. The weakness of this algorithm is that its keys should be renewed often.

Another widely used algorithm is RSA, named after its creators (Ron Rives, Adi Shamir, Leonard Adelman). It is an asymmetric algorithm considered secure because it is very expensive to decrypt it. The key uses the factorization of a product of two large prime numbers with lengths of 100-200 digits. The public keys are exchanged using the Diffie-Hellman algorithm, which is used by secure protocols such as SSL, TLS, SSH and IPSec.

AES (Advanced Encryption Standard) is a private key algorithm with a fixed block size of 128 bits, which provides good protection. The key can have a size of 128, 192 and, the most recommended, 256 bits, hence why this algorithm is considered secure. An advantage is that AES is faster than DES and 3DES, being used for the protection of sensitive data and classified government information. [14]

This type of data protection can be used in many ways: for protecting data in transit, for hiding the content of files that contains sensitive information, for making cloud storage safe, etc. One example of encryption integration in the development of the presented application is by encrypting a local binary file that contains the database backup, which is only performed manually, after the entire blockchain is verified to be valid, and only allowed to be done by the administrator.

Another type of encryption is hashing. It is an irreversible algorithm also called a digital fingerprint, which will be discussed in the next chapter.

## Integrity with hashing

A cryptographic hash function is a mathematical algorithm that takes a variable-sized input and generates a fixed-size output that represents the hash of that data. There are a few characteristics that enable this function to be used for integrity checks.

This function is irreversible, which means that by only knowing the output of the operation, the input cannot be discovered. The only solution to that would be guessing the exact same input, putting in through the hash function and getting the same output. However, if this can be successful trying the most popular passwords, it is not feasible for entire medical records or so.

Even if only one bit from the original input is changed, the entire result is significantly modified. Using this property, it is very easy to detect any alteration in the existing data. Furthermore, the hash function will always generate the same result for the same input (that means that the hash value will not change unless the data is changed). This mathematical algorithm is also collision-resistant, meaning that no two values lead to the same hash value. [15]

Knowing these properties, integrity over the internet can be achieved using a cryptographic hash function. This algorithm is only vulnerable to brute-force attacks, which involves repeatedly attempting to guess the input by comparing its hash value to the hash value of every attempted guess. A brute-force attack requires big computing power and a lot of time. The cryptographic hash algorithms have improved over time in order to make it computationally infeasible to try to guess passwords.

Moreover, an enhancement named “salting” can be used to further increase the time needed to guess a value. The process of salting involves combining another string, like the username or the email of an account, in the initial input of the function. This is effective because it prevents the hacking of the most common passwords, like “123456” or „password”. [16] If the user “John” would log in with the password “123”, choosing a very simple and random salt rule would transform the input data for the hash function from “123” to “Jo123hn”, which is dramatically harder to guess even for such a simple password., thus increasing the protection from brute-force attacks drastically and hardening the password strength. [17]

The most common hashing algorithms are MD5, SHA-1, SHA-2 and LANMAN. MD5 (Message Digest, version 5) was developed by Ron Rivest. It is a one-way hashing function that generates outputs of 128 bits; however, it was compromised in 2012 by The Flame malware and it is no longer considered an acceptable option. [18] The SHA (Secure Hash Algorithm) family includes SHA-0, SHA-1 and SHA-2, with SHA-2 being a suite of functions. SHA-1 produces a result with 160 bits but is now replaced with SHA-2, which is considered safer. Other next-generation algorithms are SHA-386 and SHA-512. [19]

Apart from data integrity, the irreversibility of the hash function also allows software developers to use it for safe authentication. In order for the application to protect its users’ credentials, like passwords, PIN codes, safe words, etc, the hash of the corresponding credential can be stored rather than the plain-text data. This prevents data theft in case the application’s database is breached. Moreover, enhancing the hash with „salt” makes it very difficult for ill-intended people to guess or find the credentials. Using this technique both external and internal threats can be mitigated.

## SQL Injection

SQLI is a procedure that exploits input requests by writing malicious SQL code with the purpose to force users to unknowingly run SQL statements on a database. Such efforts to inject code can modify or delete database data, can enable reading sensitive information, or even shutting down a DBMS. The ill-intended code can be designed to change the purpose of the affected SQL query, cause an error or delay a command.

If a vulnerability exists, SQLI can be used to log in as an already existing user without knowing the username or the password by bypassing the requested credentials. An example of a SELECT statement that can be easily bypassed is: SELECT username, password FROM users where username = “ “ and password = “ “;. Instead of writing the actual username and password in the input-designated fields, an attacker can write, instead of the username: “ or 1=1;--. The altered query would look like this: SELECT username, password FROM users where username=”” or 1=1;. 1=1 will always return true, therefore the ill-intended user would log in as administrator without knowing their credentials.

There are other ways in which SQL injection can be used; for example, for retrieving data from the database or examining its schema. Simply adding a UNION and then another SELECT statement that requests the wanted information can return the version and the type of a database, list its tables, list the content of those tables, etc. For example, finding all the users and their rights in the database is a very high-risk attack vector and can compromise the entire system. [20] Another type of SQLI is blind SQL injection. This type of attack is more complicated but still very powerful, nonetheless. Blind SQLI does not have any visible output, so UNION attacks are not visible. However, there are ways to tell that an application is vulnerable to blind SQLI. A simple time delay included in the query that would have the result of the freezing of the application for a few seconds can expose a vulnerability. [21]

A way of dealing with these types of vulnerabilities is by placing validation policies on the input received from users. This can mean not accepting meta characters, limiting the amount of data accepted, or limiting the user to a set of allowed values. However, at some point intruders will find a way around validation policies; it is very difficult to think of all the possible vulnerabilities that a piece of code can cause and there are many ways an SQL statement can be exploited.

A more versatile and secure way of protecting your application from SQL Injection is the use of Prepared Statements (or parameterized statements). A prepared statement is a stored procedure that does not concatenate the query string and compiles it, but rather keeps the command compiled and executes the statement every time. [22] This is also efficient because the statement is only compiled once, so the overhead decreases. [23] It is more resilient as well to SQL injection because it uses placeholders in the query string and every parameter is checked if it is correct and if its type corresponds to the database column type first. [24]

## Proof of work and DoS

DoS or Denial of Service attack is a type of cybersecurity attack designed to shut down a machine or a network by repeatedly sending traffic or requests. This type of attack deprives users of accessing resources and is very used in the banking industry, as well as commerce, social media, and government organizations. [25] The cost of a successful DoS or DDoS (Distributed DoS) can rise up to $120,000 for a small business and to even more than $2 million for an enterprise. [26] However, the cost may be much higher if the target of the attack is a hospital or a chain of hospitals, because inaccessibility to a patient’s medical chart can cost a human life, besides money.

A way to protect a blockchain application from DoS or DDoS attacks is to implement a Proof of Work algorithm to slow down the process of adding transactions (blocks) to the chain. Proof of Work (PoW) is a consensus mechanism enabled for the entire blockchain that operates as a set of rules which must be met to allow a user to contribute to a blockchain. This usually requires the contributor machine to do some computer processing work, slowing down the pace at which blocks are added (therefore discouraging DoS attacks). PoW can be implemented in various forms and can increase or decrease in difficulty. [27]

Different alternatives to the Proof of Work mechanism are Proof of State (PoS), Proof of Elapsed Time (PoET), Proof of Authority (PoA), Proof of Reputation (PoR), etc. PoS differs from PoW in the way that not every entity on the network can participate in the blockchain operations, but they have to be validated by the existing validators by executing a special type of transaction. PoET is a consensus algorithm that does not require high computational power but requires participating nodes to wait a period of time chosen at random before they are permitted to contribute with another block. PoA and PoR are two fairly similar algorithms. The idea is that users are only approved to contribute to the blockchain if they become validators first. A node can become a validator if they accumulate a high score that leads to a good reputation, which is calculated using predefined formulas. [28]

## Windows Application

## .NET Framework, Windows Forms platform and C# Programming Language

.NET is a platform built for developing different types of applications. It is open-source, cross-platform, free, and can be used with different editors in different languages. It was released by Microsoft in 2002 and has reached over 3700 companies and 60.000 developers. .NET applications can be developed in programming languages like C# or Visual Basic. A widely used tool for developing .NET applications is Visual Studio.

The .NET Framework is a component of the platform with the same name and was its original implementation. Besides desktop apps, it supports alternative services and websites. The components that compose the mentioned framework are: the CLR and the Class Library.

The CLR executes and deals with the running applications. It delivers many useful services, including exception handling, garbage collection, thread management, and others. The Class Library is a set of APIs for writing and reading files, drawing, connecting to databases, etc. [29]

Windows Forms is a User Interface framework designed to build Windows desktop applications. With this technology, the graphical construction of the application is easy to deploy and update, can be worked on while online or offline and brings many functionalities like drag and drops and print previews. Windows Forms carries many different controls, from the most used ones like buttons, textboxes and date pickers to drop-down boxes, contextual menus and error providers. The flexibility of this platform also allows developers to create and design their own controls as well as drawing pie charts, histograms, etc. [30].

C# is an object-oriented programming language rooted in the C family of languages. It is a modern programming language that provides garbage collection, exception handling, lambda expressions, asynchronous operations and many more. [31] It also provides SQL language specific functions which can be used to connect to a remote database and execute commands on it.

## Microsoft Azure with SQL

Cloud storage is an accessible, safer and more affordable alternative to on-premises storage. Many companies are moving their data to the cloud because maintaining server rooms and having professionals specialized on data storage is inefficient and unnecessary.

Cloud storage is a more affordable alternative to the local version because it doesn’t require hardware equipment or a storage room for servers. The maintenance is under the control of a third party.

Another advantage of cloud storage is that it can be accessed remotely, over the internet, without even having to know where and how the data is stored. There is great flexibility to this because there is no commute necessary to access stored data and it has 24/7 accessibility; all it requires is a connection to the internet.

To provide accessibility at all times, the cloud providers store client’s data redundantly. In this way, if a power outage or a natural disaster occurs, the data is still safe. On-premises, this would be hard to assure, because servers can be permanently damaged and usually data is not stored in different geographical locations. An example for this point is Hurricane Sandy that hit New York City in the fall of 2012. Many websites had to shut down during it and one of the causes was that their data was not stored in multiple locations. [32]

Another concern is security. Concerning physical security, data centers usually have guards, fences and security cameras. On the other side, firewalls, safe authentication and many other best practices should be implemented to keep the databases safe – cloud companies usually provide this as a given, making it one of the safest places to store data. [32]

Considering these advantages to the cloud storage versus the local storage and analyzing the tools and services available, I decided that the best cloud provider for the presented application would be Microsoft Azure, which delivers the Azure SQL Database feature. This solution will be described in this chapter.

Azure SQL Database is a Microsoft Azure service that provides relational database storage for applications. It is a cloud Platform as a Service (PaaS) engine with a scalable architecture and enterprise-needed features such as advanced security, monitoring and alerting, elastic pools and different price tiers to serve different needs. [33] The security capabilities integrated into the Azure SQL Database service provide a layered defense using TLS (Transport Layer Security), advanced threat protection, server firewall and others.

Other security features can be easily enabled from the Microsoft Azure Portal. Vulnerability assessment data discovery and classification, compliance (a functionality that allows the database to participate in regular audits) and others are just a few of the utilities supplied by Microsoft Azure to help deploy a secure application that uses Microsoft SQL Database. [34]

## Using Blockchain to create, view and manage EHRs – an application

The solution presented in this paper is a user-friendly application created to store sensitive data in a secure environment and assure data confidentially. It has blockchain-based storage which does not allow the alteration of medical records nor does it permit unauthorized access. The scope of this application is to provide a feasible solution for managing electronic health records for a hospital using the blockchain technology.

The application has three types of users: the patients, the doctors (along with nurses and other medical staff members) and the administrator.

The patients have to firstly register in the presence of the application administrator. The patient has to input their data: name, surname, patient ID (social security number), birthdate, password, email and confirm their password as well. After this is accomplished, the user receives an email with their PIN code, which changes once every 30 days. This is a simple form of Two Factor Authentication that adds another layer of security. After logging in, they can see a list of their records along with the doctor names and the details about every appointment. Patients can also print their records.

The doctors register similarly but also have their specialization included. Their possible actions are adding a new patient to their list (by knowing their patient ID), adding a new record (by knowing their PIN code), viewing patient’s records and printing them.

The administrator can, besides registering the users, see statistics on the database and perform an encrypted, local database backup. This can only be done after the validity of the blockchain is assured with a function that calculates all the hashes of the blocks from the beginning. The administrator can also overwrite the database in case of blockchain failure. However, this is a very critical measure and should only be performed in order to replace false data that was added to the database in the case of an attack.

It is important to be noted that every action on the application is logged. Every time the user wrongly inputs a password or a PIN code 5 times consecutively, they are locked out and the event is logged in a file. Logging also happens whenever the doctor adds a new patient, adds a new record, the administrator performs a local backup, etc.

Regarding the blockchain aspect of the application, every patient’s record is constructed as a block which then is attached to the hospital chain. At every insert operation, the entire blockchain’s validity is checked. This measure is put in place to make sure that no records are added on top of false records. In the case the database was compromised, instead of inserting, the doctor is informed that an error occurred, the event is logged and the administrator is immediately alerted. Otherwise, if the blockchain is valid, the new block is added to the database. The hash of the previous block is also stored in the new block, along with its own hash. If any block were to be altered, the corresponding hash would have to be recalculated, along with all the hashes of the next blocks. This would require very high computational power and speed, factors which can also be increased by setting the difficulty of the hash as high as possible (with the trade-off in mind, as this might affect the application performance if not done correctly).

# 5.1 The classes

The defined classes for this project are: Patient, Doctor, Block, Email, Hash and SqlBuilder. All the classes contain private attributes and public constructors, getters and setters (except the SqlBuilder and Email classes, which are implemented using the Design Pattern Singleton).

The Patient class contains the following attributes, along with their types: lastName (string), firstName (string), patientID (long), PINCode (byte[]), hashedPassword (byte[]), birthdate (DateTime), emailAddress (string), doctors (ArrayList<Doctor>). The default constructor is required in order to connect using SqlConnection and different constructors with parameters are implemented (for different use cases). The „doctors” list contains the Doctor instances of every doctor that patient has visited.

The Doctor class contains the following attributes, along with their types: lastName (string), firstName (string), doctorID (long), PINcode (byte[]), hashedPassword (byte[]), birthdate (DateTime), emailAddress (string), patients (ArrayList<Patient>), specialisation (string). The list of patients contains all the patients of that doctor. The implemented constructors are the default ones and others with a mix of parameters depending on the case (for example, not in all situations a doctor needs a list of patients).

The Block class represents the medical records and contains the following attributes: title (string), description (string), doctorID (long), patientID (long), date (DateTime), timestamp (Timestamp), nounce (int), index (int), hashOfPrevBlock (string), hashOfCurrBlock (string). The date attribute represents the actual date of the appointment, while the timestamp field represents the date the block was added to the database. Nounce is a dummy variable incremented with 1 each time the hash of the block is calculated. This variable assures the hash changes every time it doesn’t meet the difficulty required. The index of the block represents the index that records has in the database. The last two attributes contain the hashes of the previous block, respectively the current block, which are necessary to maintain the blockchain technology.

The Email class is a Singleton. The Singleton design pattern only allows the creation of one instance of a class that is used every time. This is an efficient solution for this application because source email, the password and other settings never change, so initializing an instance every time would be a waste of time and resources. The Email class has the following attributes: client (SmtpClient), message (MailMessage), smtpCredentials (NetworkCredential), sourceEmailAddress (string), sourceEmailPassword (string). SMTP (Simple Mail Transfer Protocol) is a reliable and efficient application-level protocol that is used to send emails. It operates over TCP and uses protocol number 25. [35] The SmtpClient class enables applications to send emails over SMTP. Using the client object we can set required properties such as host (Gmail), port (25), or user credentials. The MailMessage attribute type class facilitates the creation of email messages to be sent. The useful properties of this class are the subject, the body, the source and the destination of the email that is to be sent. [36] The NetworkCredential class provides the credentials for sending an email because it requires authentication. One of the constructors provided by this class only requires the email and the password of the sender, which is exactly what is needed for sending an Email in this particular case of the application. [37]

Another Singleton used for this application is MySqlBuilder. It only creates one instance because the user ID, password, data source and database name do not change and do not have to be set every time an instance is created. It has two attributes: the builder (SqlConnectionStringBuilder) and the instance (MySqlBuilder). The builder string is a simple solution to creating a connection string suitable for the SqlConnection class used to connect to the Microsoft Azure database.

The last class mentioned is the Hash class. The attributes present in this class are theString (string), theHash (string) and the nounce (int). This class provides the method that computes the hash (theHash) using the data inputted (theString) and the nounce. The method by which the hash is computed was detailed previously in this chapter.

# 5.3 Navigating through the application

The presented solution contains, aside from the 6 classes described above, 11 Windows Forms. In alphabetical order, the forms are the following: AdminInterface, AdminLogin, BlockchainApp, CheckRecord, ConfirmOverwrite, DoctorInterface, DoctorLogin, MedicalRecordInterface, PatientInterface, PatientLogin, ResetPassword. I will be detailing every interface according to the natural flow of the application.

The first form that appears when the application is launched is the BlockchainApp form. Here the user is presented with the logo and the name of the application. The user can click on the button corresponding to their intention. The following buttons are visible: „I’m a doctor/nurse”, „I’m a patient” and „Admin”, each for the type of user interacting with the application. There is also a clickable region if the user forgot their password.

Graphical user interface, application, Teams

Description automatically generated

Figure 3: First form of the application

With a click on the doctor/nurse options, the DoctorLogin form opens and either a doctor or a nurse can input their credentials and continue to the interface. The necessary credentials are doctor/nurse ID, password and PIN code. The PIN code has the purpose of sustaining 2-Factor-Authentication. All of the fields have validations using the ErrorProvider class, therefore the Validating and the Validated events are implemented for them. In the Validating function, the inputted credentials are checked: the password needs to be at least 9 characters (Length > 9) in order to prevent usage of passwords that are too weak. The password also needs to contain at least one upper character (char.IsUpper), one lower character (char.IsLower), one letter (char.IsLetter), one number (char.isNumber) and one special symbol (char.IsPunctuation). If at least one of these requirements is not met, an error using the ErrorProvider class appears and the event is canceled, therefore not reaching the Validated event. Instead, if these conditions are not broken, the Validated event occurs and the errorProvider’s error description for the password field is set to null. The same process repeats for the PIN code (where the only conditions are that the value introduced in the corresponding field is exactly 4 characters and is a number) and the doctor/patient ID, which has to be a number and have a length of exactly 7 characters. Once the validated event occurs for all of the validated input, the login is successful and the user authenticates.

If the user inputs wrong credentials (either they are not abiding by the implemented rules aforementioned or they do not match the database records) for 5 times consecutively, the administrator of the database is informed. They receive an email with the title „Too many login attempts” as well as a body with information about the doctor ID used to log in (this may be relevant if the person trying to log in used a valid ID and the IP of the machine the attempt is made on). A message appears on the screen warning the user that they have too many failed attempts and the application freezes for 30 seconds as a brute-force attack threat mitigation. In this way, an attacker cannot brute-force their way into the application. If the administrator notices that this action is repeatedly coming from the same IP it can be manually added to the firewall with a deny rule. The described event is also logged in a file, as well as all of the actions of the users.

In order to identify the IP of the machine that is trying to connect to the application, the hostname is retrieved with the method GetHostName from the Dns class, as a string. [38] Then, the Dns.GetHostByName(hostname) is called with the parameter the hostname identified previously and returns an object of type IPHostEntry which contains information about the local computer. [39] One of the properties of the object returned is AddressList, which contains the list of IP addresses associated with the host. Simply retrieving the first element of that array and casting it to string retrieves the IP address that can be written in the text file to accomplish logging for the user. [40]

# 5.3 Logging

NLog is a logging library for .NET platforms. It is a very useful tool because it has different levels of alerts for the kind of event logged (Trace, Debug, Info, Warn, Error, Fatal). Using the default settings, the exact time of the event is stored in a text file, along with the triggering class or form, the alert level and the desired message. This is an example of a logging message for the described case: **“2021-03-13 20:38:29.8125|WARN|BlockchainApp.DoctorLogIn|The doctor with IP 192.168.56.1 is repeatedly trying to log in.”** The filename, minimum logged level, the destination of the log and other settings can be written in the Nlog.config file attached to the project. In order to parse through the current day’s logs as well as browse through older logs easily, the events are stored in a file will all of the other events as well as a file created on that day which then is replaced the next day.

If the log-in is successful for a doctor or a nurse, the next Windows Forms is opened, the Doctor/Nurse Interface. The first thing that appears on the page is the list of the current user’s patients. The list can be sorted alphabetically by first name, last name or in ascending order by the patient ID by clicking on the column of the ListView.

Graphical user interface, application

Description automatically generated

Figure 4: The list of patients of a doctor shown in the application

In this form, a hospital employee can add a new patient to their list of patients, knowing their patient ID. If the inserted ID is valid, the patient is immediately added to a ListView control, where you can see the name, last name and ID, along with all of their records corresponding to the logged-in medical staff member, if clicked on the patient. The details of an already-existent record can be viewed if here as well, clicking on the row on the Listbox control containing the records and then selecting the button „Select record”. Then, a new interface appears, MedicalRecordInterface, which contains the patient ID, last name, first name, title and description of the selected appointment. At this point, the chart can be printed by clicking the Print button (and selecting all of the desired settings from the PrintDialog) or it can be closed, clicking the „Done” button. From the parent form, the doctor can also add a new record to the database. To do this, the medical staff needs to input the PIN code of the patient. This feature of the application was applied because no medical chart should be introduced without the patient’s knowledge.

The same methodology as with the password is applied to the PIN code: if the hospital employee tries to input a wrong PIN code more than 5 times, the user is warned, the application freezes for 30 seconds, the administrator is notified and the event is logged.

If the PIN is correct, the necessary fields to add an appointment appear. The doctor or the nurse can now add the title and the description of the appointment, as well as change the date (which is by default that day) if necessary. When clicking „OK”, a new Windows Forms appears which displays all of the information asking the user if they are sure that every detail is correct.

These are all the actions that an authenticated hospital employee (other than the administrator) can take.

Moving to the next type of user of the presented solution, the patient log-in section is implemented in the same way as the doctor/nurse.

After successfully logging in the application, the user can view a list of their records along with the caretaker that added them. With a click on the record, the details appear in the same form, on the right. With the record still selected, a user can print it and set desired settings with the PrintDialog.

Graphical user interface, application

Description automatically generated

Figure 5: The patient interface with a selected record

In case one of the users forgot their password, it can be changed. Clicking on the „Forgot your password” section, a form pops up with the following fields: ID, PIN and email address. After correctly filling in these details, a verification code is emailed to them. The client inputs the code and, if it is correct, they can change their password. The same rules still apply. After they fill in the password, the database record is updated and the password is successfully changed.

The last button clickable on the main form is the „Admin” button. A simple login form appears and the administrator has to fill in their password. We can see that there is also a message: „Unauthorized access is prohibited. Every action is logged.”. This type of warning has two roles: informing any possible intruder that they are not allowed to access the system as well as bring the attention to legitimate users that their actions are logged and they should not do anything that is not allowed in a contract. [41] After successfully introducing the administrator password, the AdminInterface appears with 3 different windows.

The administrator is responsible for adding new patients and doctors to the system. This is necessary in order to make sure that only legitimate individuals are allowed to use the application. It also is very important for the fields to be correctly filled in, since the personal data will be used for official medical records. The doctors need to write their doctor ID, last name, first name, specialization, email address, password (and confirm their password again). The patients fill in the same details plus the birthdate and excepting the specialization. The ID and password fields have validations as mentioned above.

Graphical user interface

Description automatically generated

Figure 6: The window for registering a new patient from the administrator side

A third section of the administrator page is the backup window. Here, the user can see the last manual, local database backup that was performed and other statistics (the number of patients, doctors and records existent in the database). By clicking the „Backup” button, after the validity of the blockchain is checked, the administrator copies all of the electronic health records to a binary encrypted file.

# 5.4 File encryption

It is very important to encrypt a file with a strong algorithm. I used AES (Advanced Encryption Standard), detailed more in chapter 3.c, with a key size of 256 bytes, the recommended size for the most secure usage of the algorithm. To use the AES algorithm in the file encryption, Aes class needs to be instantiated. Using the method Create() from the Aes class, a cryptographic object of type Aes is created. [42] In order for the algorithm to work, only the sender and the receiver need to know the key. In this particular case, since we are only storing a file, the sender and the receiver are the application itself, which uses and processes the information inside the file. The key should be stored in a secure location, because using it, an attacker could gain access to the information. In this application’s case, the key is stored in the source code; this is not a best practice and should be avoided; a solution would be storing it in the Azure Key Vault, which is a Microsoft service that stores strings and keys in a safe space. [43] Obfuscating the source code of the application can be a small fix that could help to deter an attacker from accessing it. More on obfuscating source code will be found later in this chapter.

The secret key for the encryption algorithm can be set using the Key property of an Aes object.

Another element of the encryption process is an initialization vector (IV). This tool’s purpose is to assure that no two strings have the same encryption value; it is very similar to a nounce used in blockchains. It changes the output of the encryption so that two identical strings are encrypted differently. [44]

Having only the IV or only the key is not enough to decrypt a file; this is why it is very important to store them in different ways and locations. In this application, the IV is created every time, using the GenerateIV() method from the Aes class, and it is written to a file with the name "IV.bin", separately. It is serialized with a BinaryFormatter instance and then printed to the file using a FileStream. After the IV has been generated and saved, the database content needs to be backed up. The list of records is stored in a List<Block> which is also serialized with a BinaryFormatter object. The first block or record, the genesis block, is not stored in the file, because its value never changes and is hardcoded. A FileStream which writes to the file "backup.bin" is created and encapsulates a CryptoStream. This type of stream associates the data to its encrypted form. A CryptoStream needs a FileStream, an Aes object and a file mode (in this case, it is Write) to be created and used. All of the streams are used enclosed in a using block in order to be sure they are disposed of correctly and no stream remains open. Otherwise, the developer would have to manually close the stream every time they do not want to use it anymore.

In case of a database failure or an attack, the database can be restored using this file. The process of decryption is very similar; firstly, the initialization vector is read from the "IV.bin" file. Then, the records are deserialized and decrypted from the "backup.bin" file using the FileStream, CryptoStream and BinaryFormatter and every record is inserted into the database.

# 5.6 The database

The database contains 4 tables, Patient, Doctor, Associations and Block, one sequence, block\_indexes, and one trigger, onUpdate.

The Patient table has 8 columns. patient\_id is the primary key and is represented as an integer; as the primary key of the table, it has the constraints to be not null and unique. Patient\_last\_name, patient\_first\_name, hashed\_pass, hashed\_PIN, email are columns that contain data as varchar type and cannot be null. Last\_login and birthday are of type date. The column last\_login is necessary in order to keep track if a new personal PIN should be issued again and contains the date the person last logged in; if the account has not been used for more than 30 days, a new PIN is issued and sent on the email.

The Doctor table is very similar to the patient table; the only difference is that it doesn’t contain the column birthday and it contains the column specialization, which is represented as varchar and has the constraint to be not null.

The Associations table is used to solve a many-to-many relationship between doctors and patients, because any patient can have multiple doctors and many doctors can have multiple patients. The two columns in this table are patient\_id and doctor\_id, both represented as integers and not null. Together they form a composite primary key for the table and any pair of two has to be unique.

The block table stores the medical records of the patients and forms the blockchain. There are 10 columns in this table, which are all not null: patient\_id (int), doctor\_id (int), appointment\_date (date), appointment\_title (varchar), appointment\_description (varchar), nounce (int), block\_timestamp, (date), block\_index (int), hash\_of\_prev\_block (varchar), hash\_of \_curr\_block (varchar). The primary key of this table is composite and contains the first four columns mentioned.

The trigger onUpdate raises an error every time an update query is applied on the Block table. This does not allow the modification of a block as it would cancel the immutability characteristic of a record in a blockchain. The error message “No updating” is displayed when an update is tried on the table.

The sequence block\_indexes contains the record index and starts at 0. It automatically increments with 1 when a block is added. It has no maximum value and it has the “no cycle” value set, because the block\_index value cannot be the same for two records.

The .NET framework requires the namespace System.Data.SqlClient in order to make a connection with an SQL server and execute procedures. This namespace contains necessary classes like SqlConnection, SqlCommand and SqlException, among many others.

To connect to a database, an instance of the SqlConnection class needs to be created. This object represents a session with the SQL Server and requires a connection string. The connection string contains details of the server like the database name and other settings. An example of a valid connection string (which is constructed with an instance of the SqlConnectionStringBuilder class) is the following: "Persist Security Info=False;Integrated Security=true;Initial Catalog=Northwind;server=(local)". [45] The connection is opened using the Open() method.

The database connection needs to be closed by calling Close or Dispose. However, if the connection is created inside a “using” block, it is closed automatically after the block is finished executing. [46]

The class used to execute commands in the .NET framework is SqlCommand. An instance of this class executes a statement or a stored procedure. A stored procedure is a statement or a set of statements that have already been compiled and are only required to be called in order to be executed. [47]

A SqlCommand object requires, when created, as parameters, a SqlConnection object and a string that contains the query. The query must be a Transact-SQL or T-SQL statement, which is Microsoft’s extension of the SQL language and is very similar to SQL. After initializing a SqlCommand object, the database connection is opened using the Open() method.

Useful methods of the SqlCommand class that execute commands include ExecuteReader and ExecuteNonQuery.

* ExecuteReader builds a SqlDataReader and returns rows so it is recommended to be used with SELECT queries. A SqlDataReader provides a way of reading rows from an SQL Server database. This object obstructs the connection and must be closed before further operations can be performed. [48]
* ExecuteNonQuery does not return any rows and executes the following commands: INSERT, DELETE, UPDATE, SET.

Command parameters are a safe alternative to string concatenation in C# because they prevent SQL injection by verifying supplied input data.

In C#, a SqlCommand has attached to it a collection of parameters (a SqlParameterCollection object). Adding to this list of parameters the user-provided data (of course, after validating it) is a good practice for dynamic SQL. [49] The methods used for parameter insertion are Add, AddWithValue, AddRange.

The function Add is straightforward; it receives as an argument a parameter and it adds it to the collection. The function AddWithValue, which was used in this application, receives a String and an Object as arguments and replaces the location specified in the String with the Object received. [50] The function AddRange inserts a list of parameters to the end of the SqlParameterCollection. [51]

When an error or a warning occurs, an instance of SqlError is created and, along with that, a SqlException is also triggered. The SqlConnection may remain open if the severity of the error is not high. [52]

An exception can be raised in case a constraint is violated (such as not null or unique), an invalid cast is made, an error occurs during the transmission of the data over a Stream, etc. An exception handler such as try-catch should be used in the development of an application written in C# to prevent the application from crashing.

# 5.6 Design

To improve the Windows Forms design I used a NuGet package called MaterialSkin, which provides a series of controls and changes the aspect of the form. In order to implement MaterialSkin, the form needs to inherit MaterialForm and not Form. To obtains a consistent design, all of the application forms should implement MaterialForm, which can be found in the MaterialSkin.Controls collection, along with many other controls. Then, an instance of the MaterialSkinManager should be created and the form should be added to the manager with the method AddFormToManage. After this, the Theme can be set (i used MaterialSkinManager.Themes.LIGHT) and the ColorScheme property by mentioning the colours of the forms. In this application, I used different shades of Indigo. [53]

# 5.7 Code obfuscation

Code obfuscation is a technique that modifies executable code to prevent hackers from reading the source code. Code obfuscation is not encryption and cannot completely prevent someone from reading code – it simply makes it more difficult and time-consuming, intending to discourage attackers from attacking the software. The obfuscator, for instance, can change variable names to nonsense, making complicated code harder to comprehend. Coders will rename variables in accordance with clean code principles, and an obfuscator will change those names. Another method for concealing keys, passwords, and strings is to hide them from compilers by encoding them at runtime. An obfuscator can also add useless code or completely replace your code with objects that do the same job but are tougher to interpret. [54]

ConfuserEx is an open-source tool designed for .NET applications that obfuscates code. It uses protection methods like symbol renaming, method reference hiding, anti-memory dumping, constant encryption and others. [55] It has a simple GUI and makes it very easy to obfuscate code even if you are not a security expert. This is a tool that can obfuscate .dll code, which may contain any sensitive strings, relevant variable, functions names, etc. [56]

## Conclusions and future work

## Conclusions

To sum up everything that has been stated so far, with the growing amount of cybernetic threats putting pressure on the medical information system, there should be taken more and more measures to prevent successful attacks, as well as implementing security good practices such as the ones presented in this paper. Hackers find ingenious ways and craft new tools to attack institutions and disrupt normal activity, so the security measures must evolve with them in order to keep up with the attacks and even prevent them from happening. Starting with the COVID-19 pandemic and not stopping to it, a call of action should be made regarding hospitals’ cybernetic systems, because the stakes are more than material damages, as attacks can impact human lives to the degree of putting them in danger.

Patients’ data stored in electronic health records should be treated very seriously, because it is extremely sensitive information and a successful attack on it can have devastating effects; if a temporary ransomware attack caused a patient to die in transit, we can agree that, for example, if medical charts were to be wiped off with no backup, it can have unimaginable consequences. This would be critical damage that should never be allowed to happen, but it most definitely is not out of reach for attackers in the future.

This paper has the goal of providing a simple, secure blockchain-based solution for managing electronic health records that is easily accessible for patients and doctors and does not require many resources. The permissioned blockchain technology has the immutability feature which prevents anyone from changing or deleting records from the database. It brings good practices to the table, as they are described in this paper and they come down to the most important principles in the cybersecurity domain: CIA and AAA. Confidentiality, Integrity and Availability prevent data theft, data alteration and disruption of services. Authentication does not allow unauthorized individuals to have access to data, Authorization ensures there are different levels of authority in the application and a user can only do what they are permitted to do, and Accounting keeps track of what every user does to keep them responsible.

The presented solution incorporates a variety of good practices, like 2FA authentication, data encryption, data hashing, SQL injection prevention, source code obfuscation, logging, brute-force attack prevention and others. To accomplish these described features, it uses the powerful programming language and tool C#, paired with the Microsoft Azure service Azure SQL Databases which stores data on the cloud, with redundancy, high accessibility and data protection, all while providing a last-resort database backup locally but encrypting it with a 256-bit key AES encryption algorithm.

While there is future work necessary for this project, the described application manages to store a hospitals’ sensitive data safely in a blockchain-based fashion while still assuring good functionality and without affecting its performance. The blockchain element adds data immutability, which is necessary for the healthcare sector as losing medical charts can have catastrophic effects.

Securing hospitals’ information systems should be a future direction taken by the government and specialists. The COVID-19 pandemic should be a call to action to strengthen the medical information infrastructure and modernize it, especially in Romania, which, as stated in the before-mentioned article from CERT-RO, has many vulnerabilities in many hospitals and clinics all over the country. I believe there should be a nationwide solution that would tie all hospitals together in a secure environment, without exposing patient’s data to risks. Taking all of the precautions to protect the medical information system is a very important step in protecting patients, because nowadays physical care is not enough anymore. Healthcare is not a very popular domain for IT specialists to work in – however, this lack of investments and attention should be stopped now – the healthcare system needs IT professionals.

## Future work

The application presented in this paper is functional and could be used to store patient’s data. However, a few improvements should be added that would overall increase its utility.

The first item on the list for future work for this project is allowing users to save other types of data, such as images, files, PDF documents and others, which at the moment of writing this paper, limits the usage of the application. For example, X-Rays and other image-based medical tests cannot be stored in the database.

In terms of source code, an application of this size should implement design patterns. For example, the creational pattern Builder would be very good in case of instantiating the records, patients and doctors as their constructors require multiple parameters.

In terms of security, even with obfuscation, encryption keys should not be stored in source code but rather in a secure place, like Azure Key Vault, provided in the suite of services from Microsoft Azure, once this solution would receive the necessary funding.

Another improvement of the storage system would be backing up data in a second cloud environment, such as Oracle databases, for example. This would eliminate the single point of failure vulnerability and assure that the database backup is not performed locally, on the host machine, which can as well be affected by a cybernetic attack, but rather on a second trusted platform. However, this can raise the costs of the application a lot, as a second service would have to be bought.

Another solution to this problem is adopting the public blockchain technology and storing the data transparently, on all of the trusted nodes. This improvement idea should be approached carefully, as public blockchains are not appropriate mediums to store sensitive information on, such as healthcare data.

One more improvement idea is enhancing multifactor authentication. Currently, it uses the 2FA version, where users have to input their secret password and their generated PIN code. Nonetheless, this has some downs to it, because the PIN code is also accessed using personal credentials, which may very well be the same for both this application and the email in case of an inexperienced user. Adding a physical element such as a token that generates PIN codes on the spot (like some banks use) or integrating biometrics would be more effective.

Finally, the scope of the presented application is confined to a single hospital, restricting its reach and impact. If it were to be extended across the country, it would provide a solution for safely managing medical records at a national level.

# Bibliography

|  |  |
| --- | --- |
| [[1] | CERT-RO, “Informare referitoare la vulnerabilități și atacuri cibernetice privind spitale și clinici din România,” CERT-RO, Bucharest, 2020. |
| [[2] | CERT-RO, “Informare referitoare la vulnerabilități și atacuri,” 8 November 2020. [Online]. Available: https://cert.ro/vezi/document/prezentare-sesiune-online-spitale-octombrie-2020. |
| [[3] | Kaspersky, “Kaspersky,” [Online]. Available: https://usa.kaspersky.com/resource-center/definitions/what-is-social-engineering. |
| [[4] | L. Conway, “Blockchain Explained,” 17 November 2020. [Online]. Available: https://www.investopedia.com/terms/b/blockchain.asp. |
| [[5] | T. K. Sharma, “PUBLIC VS. PRIVATE BLOCKCHAIN : A COMPREHENSIVE COMPARISON,” [Online]. Available: https://www.blockchain-council.org/blockchain/public-vs-private-blockchain-a-comprehensive-comparison/. |
| [[6] | BlockGeeks, “Blockchain in healthcare: The Ultimate use case?,” [Online]. Available: https://blockgeeks.com/guides/blockchain-in-healthcare/. |
| [[7] | “Blockchain Tutorial for Beginners: Learn Blockchain Technology,” [Online]. Available: https://www.guru99.com/blockchain-tutorial.html. |
| [[8] | N. Heath, “What’s the difference between a private and public blockchain?,” 05 September 2018. [Online]. Available: https://www.intheblack.com/articles/2018/09/05/difference-between-private-public-blockchain. |
| [[9] | Euromoney, “The rise of private blockchains,” [Online]. Available: https://www.euromoney.com/learning/blockchain-explained/the-rise-of-private-blockchains. |
| [[10] | PurpleSec, “2021 Cyber Security Statistics,” 2021. [Online]. Available: https://purplesec.us/resources/cyber-security-statistics/. |
| [[11] | L. O. Nweke, “Using the CIA and AAA Models to Explain Cybersecurity Activities,” *PM World Journal,* p. 3, 2017. |
| [[12] | CYBER EDU, “What is the CIA Triad?,” [Online]. Available: https://www.forcepoint.com/cyber-edu/cia-triad. |
| [[13] | Any Difference Between, “Difference Between Encryption and Decryption,” [Online]. Available: https://anydifferencebetween.com/difference-between-encryption-and-decryption/. |
| [[14] | Cisco Systems, Inc., “What Is Encryption?,” [Online]. Available: https://www.cisco.com/c/en/us/products/security/encryption-explained.html. |
| [[15] | Synopsys Editorial Team, “What are cryptographic hash functions?,” 10 December 2015. [Online]. Available: https://www.synopsys.com/blogs/software-security/cryptographic-hash-functions/. |
| [[16] | R. Picheta, “How hackable is your password?,” 23 April 2019. [Online]. Available: https://edition.cnn.com/2019/04/22/uk/most-common-passwords-scli-gbr-intl/index.html. |
| [[17] | I. M., “Three random words or #thinkrandom,” 27 October 2016. [Online]. Available: https://www.ncsc.gov.uk/blog-post/three-random-words-or-thinkrandom-0. |
| [[18] | D. Rountree, “Hashing Algorithm,” 2011. [Online]. Available: https://www.sciencedirect.com/topics/computer-science/hashing-algorithm/pdf. |
| [[19] | D. Eastlake and P. Jones, “US Secure Hash Algorithm 1 (SHA1),” RFC Editor, 2001. |
| [[20] | PortSwigger, “PortSwigger,” [Online]. Available: https://portswigger.net/web-security/sql-injection. |
| [[21] | PortSwigger, “Blind SQL injection,” [Online]. Available: https://portswigger.net/web-security/sql-injection/blind. |
| [[22] | “SQL Injection,” [Online]. Available: https://owasp.org/www-community/attacks/SQL\_Injection. |
| [[23] | “Prepared statements and stored procedures,” [Online]. Available: https://www.php.net/manual/en/pdo.prepared-statements.php. |
| [[24] | “SQL Injection,” [Online]. Available: https://www.w3schools.com/sql/sql\_injection.asp. |
| [[25] | Paloalto Networks, “What is a denial of service attack (DoS) ?,” [Online]. Available: https://www.paloaltonetworks.com/cyberpedia/what-is-a-denial-of-service-attack-dos. |
| [[26] | Bulletproof, “Bulletproof Annual Cyber Security Report 2019,” Bulletproof, 2019. |
| [[27] | D. Karaivanov, “Proof of Work Explained in Simple Terms,” 27 October 2019. [Online]. Available: https://chainbulletin.com/proof-of-work-explained-in-simple-terms/. |
| [[28] | R. Zhang, R. Xue and L. Liu, “Security and Privacy on Blockchain,” *ACM Computing Surveys,* p. 35, 2019. |
| [[29] | “.NET Framework documentation,” [Online]. Available: https://docs.microsoft.com/en-gb/dotnet/framework/. |
| [[30] | “Desktop Guide (Windows Forms .NET),” 26 10 2020. [Online]. Available: https://docs.microsoft.com/en-us/dotnet/desktop/winforms/overview/?view=netdesktop-5.0. |
| [[31] | “A tour of the C# language,” 06 08 2020. [Online]. Available: https://docs.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/. |
| [[32] | L. Rathnam, “Why Is Cloud Storage a Good Idea?,” 06 July 2020. [Online]. Available: https://www.cloudwards.net/cloud-storage-good-idea/. |
| [[33] | Microsoft, “What is Azure SQL Database?,” 21 September 2020. [Online]. Available: https://docs.microsoft.com/en-us/azure/azure-sql/database/sql-database-paas-overview. |
| [[34] | Microsoft, “An overview of Azure SQL Database and SQL Managed Instance security capabilities,” 26 October 2020. [Online]. Available: https://docs.microsoft.com/en-us/azure/azure-sql/database/security-overview. |
| [[35] | J. Klensin, “RFC 5321,” October 2008. [Online]. Available: https://tools.ietf.org/html/rfc5321. |
| [[36] | Microsoft, “MailMessage Class,” [Online]. Available: https://docs.microsoft.com/en-us/dotnet/api/system.net.mail.mailmessage?view=net-5.0. |
| [[37] | Microsoft, “NetworkCredential Class,” [Online]. Available: https://docs.microsoft.com/en-us/dotnet/api/system.net.networkcredential?view=net-5.0. |
| [[38] | Microsoft, “Dns.GetHostName Method,” [Online]. Available: https://docs.microsoft.com/en-us/dotnet/api/system.net.dns.gethostname?view=net-5.0. |
| [[39] | Microsoft, “Dns.GetHostByName(String) Method,” [Online]. Available: https://docs.microsoft.com/en-us/dotnet/api/system.net.dns.gethostbyname?view=net-5.0. |
| [[40] | Microsoft, “IPHostEntry.AddressList Property,” [Online]. Available: https://docs.microsoft.com/en-us/dotnet/api/system.net.iphostentry.addresslist?view=net-5.0#System\_Net\_IPHostEntry\_AddressList. |
| [[41] | The University of Tennessee System, “The University of Tennessee System,” [Online]. Available: https://security.tennessee.edu/login-banners/. |
| [[42] | Microsoft, “Microsoft,” [Online]. Available: https://docs.microsoft.com/en-us/dotnet/api/system.security.cryptography.aes.create?view=net-5.0#System\_Security\_Cryptography\_Aes\_Create. |
| [[43] | Microsoft, “CA5390: Do not hard-code encryption key,” 05 October 2020. [Online]. Available: https://docs.microsoft.com/en-us/dotnet/fundamentals/code-analysis/quality-rules/ca5390. |
| [[44] | TechTarget Contributor, “Initialization vector (IV),” March 2011. [Online]. Available: https://whatis.techtarget.com/definition/initialization-vector-IV. |
| [[45] | Microsoft, “SqlConnection.ConnectionString Property,” [Online]. Available: https://docs.microsoft.com/en-us/dotnet/api/system.data.sqlclient.sqlconnection.connectionstring?view=dotnet-plat-ext-5.0. |
| [[46] | Microsoft, “Microsoft,” [Online]. Available: https://docs.microsoft.com/en-us/dotnet/api/system.data.sqlclient.sqlconnection?view=dotnet-plat-ext-5.0. |
| [[47] | Microsoft, “Microsoft,” [Online]. Available: https://docs.microsoft.com/en-us/dotnet/api/system.data.sqlclient.sqlcommand?view=dotnet-plat-ext-5.0. |
| [[48] | Microsoft, “Microsoft,” [Online]. Available: https://docs.microsoft.com/en-us/dotnet/api/system.data.sqlclient.sqldatareader?view=dotnet-plat-ext-5.0. |
| [[49] | Microsoft, “Microsoft,” [Online]. Available: https://docs.microsoft.com/en-us/sql/relational-databases/security/sql-injection?view=sql-server-ver15. |
| [[50] | Microsoft, “Microsoft,” [Online]. Available: https://docs.microsoft.com/en-us/dotnet/api/microsoft.data.sqlclient.sqlparametercollection.addwithvalue?view=sqlclient-dotnet-core-2.1. |
| [[51] | Microsoft, “SqlParameterCollection.AddRange Method,” [Online]. Available: https://docs.microsoft.com/en-us/dotnet/api/microsoft.data.sqlclient.sqlparametercollection.addrange?view=sqlclient-dotnet-core-2.1. |
| [[52] | Microsoft, “Microsoft,” [Online]. Available: https://docs.microsoft.com/en-us/dotnet/api/system.data.sqlclient.sqlerror?view=dotnet-plat-ext-5.0. |
| [[53] | I. Maes, “IgnaceMaes/MaterialSkin,” [Online]. Available: https://github.com/IgnaceMaes/MaterialSkin. |
| [[54] | A. Heddings, “What Is Code Obfuscation, and Should You Use It?,” 15 December 2020. [Online]. Available: https://www.cloudsavvyit.com/6923/what-is-code-obfuscation-and-should-you-use-it/. |
| [[55] | “ConfuserEx,” [Online]. Available: https://yck1509.github.io/ConfuserEx/. |
| [[56] | TutorialsPoint, “DLL - How to Write,” [Online]. Available: https://www.tutorialspoint.com/dll/dll\_writing.htm. |
| [[57] | Cisco Systems, Inc., “Next Generation Cryptography,” October 2020. [Online]. Available: https://tools.cisco.com/security/center/resources/next\_generation\_cryptography. |
| [[58] | Cloudflare, “What is TLS (Transport Layer Security)?,” [Online]. Available: https://www.cloudflare.com/en-gb/learning/ssl/transport-layer-security-tls/. |
| [[59] | Purplesec, “2021 Cyber Security Statistics,” [Online]. Available: https://purplesec.us/resources/cyber-security-statistics/. |
| [[60] | Kaspersky, “What is a Trojan? - Definition and Explanation,” [Online]. Available: https://www.kaspersky.com/resource-center/threats/trojans. |
| [[61] | J. Strickland, “How Cloud Storage Works,” [Online]. Available: https://computer.howstuffworks.com/cloud-computing/cloud-storage.htm. |
| [[62] | Accenture Security, “The cost of cybercrime,” 2019. |

# List of figures

[Figure 1: My own data processing based on data collected from Purplesec [59] - 11 -](file:///D:\Facultate%20an%202%20semestru%202\Windows\BlockchainApp\BlockchainApp\Sabrina%20Lupsan's%20Bachelor%20thesis.docx#_Toc75196111)

[Figure 2: The encryption and decryption process [13] - 13 -](#_Toc75196115)

[Figure 3: First form of the application - 23 -](#_Toc75196116)

[Figure 4: The list of patients of a doctor shown in the application - 25 -](#_Toc75196117)

[Figure 5: The patient interface with a selected record - 27 -](#_Toc75196118)

[Figure 6: The window for registering a new patient from the administrator side - 28 -](#_Toc75196119)