Institute for Data Engineering
Module Advanced Internet Computing – Lab Exercises

Project Topic C – Secure Blockchain-Based Healthcare Records Management System

Introduction

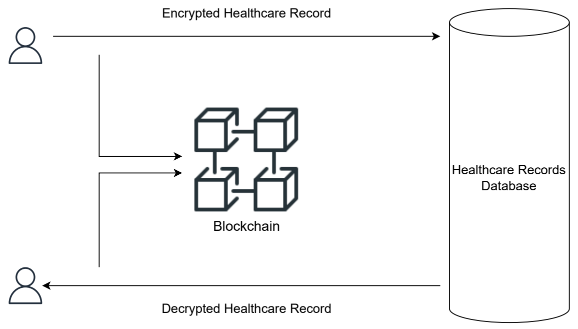
Encrypted Healthcare Record
Blockchain
Healthcare Records Database
Decrypted Healthcare Record

Figure 1: High-level overview of the health records management system

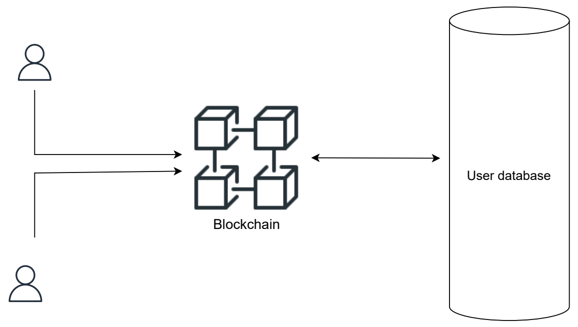

Figure 2: High-level overview of the user authentication system

In recent years, the digitization of healthcare records has brought about significant advancements in the quality and accessibility of care. However, this digital transformation also presents new data security and privacy challenges. These challenges highlight the need for innovative solutions to secure sensitive health information while facilitating efficient use. At the core of addressing these challenges is the application of cryptographic principles, which play a critical role in ensuring the confidentiality, integrity, and authentication of electronic health records (EHRs) [1]. Furthermore, the lack of interoperability between different healthcare information systems often leads to inefficiencies, increased costs, and errors in patient care. Thus, a system that ensures the utmost security of healthcare records and supports seamless data sharing and collaboration among healthcare providers, patients, and other stakeholders is urgently needed [1].

Integrating blockchain technology into managing healthcare records offers a promising solution to these issues. Blockchain's inherent properties, such as decentralization, immutability, and transparency, make it an ideal candidate for creating a secure, interoperable, and patient-centric healthcare data management system. By leveraging blockchain technology, it is possible to create a tamper-proof ledger of healthcare transactions, ensuring data integrity and enabling secure information sharing across the healthcare ecosystem [2]. Furthermore, smart contracts can automate consent and access policies, enhancing the efficiency and responsiveness of healthcare services [3].

In this project, students will be tasked with designing a Secure Blockchain-Based Healthcare Records Management System. This comprehensive project will consist of two main modules: the healthcare records management module, focused on securely managing and storing healthcare data, and the user records management and authentication module, dedicated to handling user identities, access permissions, and secure authentication processes. Through this project, students will not only apply their knowledge of cloud computing, cryptography, and blockchain technology but also tackle the development of innovative solutions to real-world problems in healthcare data management. This endeavor will equip them with critical skills and experience in creating systems that address data security, privacy, and interoperability issues in the digital healthcare domain.

For this project, we will assume that this healthcare records management system is being implemented at one specific hospital.

Overview:

- The healthcare records management system will enable users to access and modify patients' healthcare records at a hospital, controlled by roles defining specific access types.
- The system will store users' data and assign roles to them that determine which health data the users can access and modify.
- The healthcare records are stored in a database with a versioning system that allows retrieval of older versions and hence tracks changes made.
- All user and healthcare data changes are tracked using a smart contract that emits an event every time an access operation is performed.
- The smart contract is run on a private blockchain to maintain confidentiality.
- The same or a different smart contract also performs access control based on user roles.
- Users are identified using a systemwide identifier the smart contract uses to track changes and access control requests.
- The system allows users to log in, during which their systemwide identifiers are fetched or generated. All actions taken by the user are then access-controlled using that identifier and the user's role.

Requirements and Implementation Decisions:

General implementation details:

1. Start by setting up a private blockchain (such as Ganache[1]), which can run locally on your machine and also in a VM on the cloud and will host the smart contracts you deploy.
2. Decide on the database technology(such as PostgreSQL, MongoDB) you want to use. You can use any type of data storage you want. Also look at offerings from the Google Cloud Platform.

The project outcome will have three major components: the user data storage and authentication system, the healthcare records management system, and the user interface.

[1]`https://archive.trufflesuite.com/ganache/`

The user data storage and authentication system

This system module will enable administrators to store user data on a database and use a smart contract to access control those users.

1. Decide on how you want to store user data. The database should store the details of users, including their names, addresses, and hospital designations. Each user should have an ID, which should be distinct from the system-wide identifier.
2. The user data should be encrypted in transit. You can make use of frameworks like PyCryptodome[2] to achieve this.
3. Decide how to implement user roles and perform access control based on them.
4. Think about storing mappings between roles and access permissions. This can be done in the smart contract itself or in a separate table in the database. You also need to store mappings between user IDs and roles.
5. A user should be uniquely identifiable across the system. How you identify a user is up to you.
6. Think about how you can return the yes/no values from the smart contract to the off-chain function invoking that smart contract.
7. The smart contract should emit an event for every access to user database records. For example, it should emit an event when a new user is added, a user's data are modified, or a user is deleted.
8. An event should also be emitted for logging every access request. The event should contain a timestamp, the actor's identifier, the action requested, the affected user's ID, and whether the request was approved or denied.

The healthcare records management system

This module will store and manage healthcare records along with a smart contract for auditability.

1. Decide on a schema for the healthcare records table.
2. The healthcare records table should implement a versioning system. There are multiple ways to implement this. Versions should be updated only when an existing record is modified.
3. Addition, retrieval, modification, and deletion of healthcare records are access-controlled as mentioned in the user data module.
4. You can use the same contract or a different one for handling the auditability of healthcare records.
5. Just as in the case of the user records, every access to the healthcare records table should cause the smart contract to emit an event with the same information as before: a timestamp, the actor's identifier, the identifier of the user affected and the action performed.
6. As with user data, healthcare records should be encrypted in transit.

[2] `https://www.pycryptodome.org`

The user interface

1. The user interface must offer a flow for user login, which, in the backend, should fetch the user's identifier from the Database.
2. The UI should have a search page for users. All patients, doctors, nurses, and other users should be searchable, except for roots and admins.
3. The search results can be shown as you desire, with information about users from the database.
4. There should be another page for searching health records. Health records should only be searchable using the patient's ID.
5. The UI should allow access-controlled users to modify healthcare records.

The three modules, the local blockchain and the Database Management System (if not using a Google Cloud Platform solution) should be containerized using Docker. The Docker containers should run on a VM on Google Cloud. The user interface should be accessible on a public URL.

Outcomes

The expected outcomes of this project are two-fold: (1) the actual project solution, (2) two presentations of the results.

Project Solution

The project has to be hosted on a Git repository in TUHH's GitLab instance[3] – you will get instructions about the repository setup at the lab kickoff meeting. Every member of your team has to use their own, separate Git account. *We will check who has contributed to the source code*, so please make sure you use your own account when submitting code to the repository. Furthermore, it is required to provide an easy-to-follow README that details how to deploy, start and test the solution. The best practice is to provide a README that describes "Plug-and-Play" instructions on how to use your solution.

For the submission (see below), you also have to create one or more Docker images, which contain(s) your complete implemented solution, i.e., including all dependencies.

Presentations

There are two presentations. The first one is during the consultation hours, and describes your status at that particular point of time. This presentation will not be graded, i.e., it is primarily a vehicle to check your progress and to show to other groups what you are working on and how you want to solve the problem. The second presentation is during the final meetings and contains all your results. The actual dates are announced in Stud.IP.

Every member of your team is required to present in either the first *or* the second presentation. Each presentation needs to consist of a slides part and a demo of your implementation. Think carefully about how you are going to demonstrate your implementation, as this will be part of the grading. You have 10 minutes (strict) of time for your presentation at the consultation hours, and 15 minutes (also strict) of time for your presentation at the final meeting. We recommend to use about 5 minutes at each meeting for the slides part of your presentation, and the remaining 5 / 10 minutes, respectively, for the live demonstration of your implementation results, if there is already a showcase you can make use of. After each presentation, there will be a Q&A session of max. 5 minutes. The past has shown that providing a nice use case story usually helps to present

[3] `https://collaborating.tuhh.de/`

the project outcomes. While providing such a use case story is not an absolute must, it will surely help the audience to understand your work better.

Grading

A maximum of 60 points are awarded in total for the project. Of this, 70% are awarded for the implementation (taking into account both quality and creativity of the solution as well as code quality and documentation), and 30% are awarded for the final presentation (taking into account content, quality of slides, presentation skills, and discussion).

A strict policy is applied regarding plagiarism. Plagiarism in the source code will lead to 0 points for the particular student who has implemented this part of the code. If more than one group member plagiarizes, this may lead to further penalties, i.e., 0 points for the implementation of the whole group.

Deadline

The hard deadline for the project is **July 7th, 2024**, 23:59. Please submit the presentations via Stud.IP. The Docker images need to be uploaded to DockerHub and made available to the lecturer team. For this, please write a mail on how to access the containers to avik.banerjee@tuhh.de. The deadline for this is also July 7th, 2024, 23:59. Late submissions will not be accepted.

Test Cloud Infrastructure

This year, the Google Cloud Platform is supporting the lecture with an Education Grant, which you can use for Virtual Machines (VMs) and other cloud resources. The computational resources can be used for free, however, you need to own a Google account to use them. To access the resources, please visit the following URL (please note that this is a masking URL, i.e., you need to click it, simply copy/pasting it will *not* help): http://google.force.com/. The full URL is also provided in Stud.IP.

Please note that you need to request the resources until August 2nd, 2024; the coupons are then valid until April 2nd, 2025. Afterwards, they will become void and you will not be able to access the budget.

The e-mail address you provide in this form does *not* have to be your Google account, but it *must* be an address within the TU Hamburg domain, i.e., `@tuhh.de`. By providing such an address, you can claim a coupon code, which you can then use to redeem a $50.00 voucher for the Google Cloud Platform. Detailed instructions are provided at this URL and subsequent e-mails you will receive from Google.

You can redeem one coupon code per e-mail address at a time. The voucher of $50.00 is in most cases sufficient to conduct the implementation tasks of the lab exercises. However, if this is not the case, please send a mail to avik.banerjee@tuhh.de. Please also take into account that there are the free tier resources of the Google Cloud Platform, which are usually sufficient to test and run your solutions.

Note that exceeding this budget will lead to immediate shutdown of the project by Google, and we have no means of influencing this, or providing extensions. We therefore recommend you to monitor your resource usage, in order to avoid unnecessary spending, for instance, by VMs left running.

You are allowed to use the Google Cloud Resources only for the purpose of this course. Both Google and TU Hamburg are monitoring the use of the resources, and any misuse (hosting of illegal data, cryptocurrency mining, etc.) will be punished. In case of any questions regarding the organization of the Google Cloud Platform, please send an e-mail to avik.banerjee@tuhh.de.

Please take care that you do not publish any credentials or (ssh) keys on the Internet. For your implementation, please use dedicated credential files and add these credential files and the keys to the .gitignore file or put them on a system path outside of your Git repository. If you accidentally leak any credentials or ssh keys, inform us immediately. Google is pretty good at identifying leaked credentials, and will then shut down your account at least temporarily. If this happens just before the deadline, there is a good chance that your account will not be released again in due time.

In case of noncompliance, the person(s) responsible will fail the lab!

References

[1] A. A. Mamun, S. Azam, and C. Gritti, "Blockchain-based electronic health records management: A comprehensive review and future research direction," *IEEE Access*, vol. 10, pp. 5768–5789, 2022.

[2] R. P. Sarode, Y. Watanobe, and S. Bhalla, "A blockchain-based approach for audit management of electronic health records," in *10th International Conference on Big Data Analytics*, vol. 13830 of *Lecture Notes in Computer Science*, pp. 86–94, Springer, 2022.

[3] X. Mao, C. Li, G. Zhang, X. Wei, and C. Xing, "Application of blockchain in trusted data provenance," in *2022 IEEE 22nd International Conference on Software Quality, Reliability, and Security Companion*, pp. 106–112, 2022.