Integrating Consent Management Techniques into Blockchain-Based Medical Data Sharing

BEN JACOBS[1] , **CHHAGAN LAL**[1] , **MAURO CONTI**[1]
[1]TU Delft

***Abstract*—The health industry is a data-intensive domain. Sharing medical data between medical facilities is necessary for providing good healthcare as well as for research in the healthcare domain. However, due to the sensitive and personal nature of health data, challenges arise when sharing this data. Consent management is a key aspect. The use of blockchain technology can enable patients to be the owner of their data, and allow them to manage who is able to view and process their data.**

In this paper, we define the requirements for an adequate consent management system for blockchain-based medical data sharing. After that, we compare the related work and identify limitations based on the requirements. Finally, we provide a highly dynamic consent management system with fine-grained access control using hierarchical structures, based on the Hyperledger Fabric permissioned blockchain. The proposed solution complies with the EU General Data Protection Regulation and addresses important privacy issues.

***Keywords*—consent management; health data sharing; blockchain; privacy**

1 Introduction

The continuous generation of large amounts of health data from a variety of sources makes the healthcare industry a data-intensive domain [1]. This data is often highly confidential since it corresponds to patients' personal data. It is important that this medical data is shared between different medical facilities for various purposes, such as in-depth analysis and collaborative research [2]. Problems arise concerning the privacy and integrity of patient information, and therefore the reliability and safety of the healthcare system as well.

As of 2018, the General Data Protection Regulation (GDPR) [3] requires the security related to the collection, sharing, and transferring of personal and sensitive information to be enhanced, with consent management being a key aspect [4]. The GDPR states that sensitive data always requires explicit consent. Due to its personal nature, health data is always deemed sensitive. Willingness to provide consent requires a large amount of trust in the entity processing the data. Trust is often deemed a measure for the willingness of an individual to be vulnerable to the actions of an entity, based on the trust that the entity will act according to expectations [5]. Although healthcare instances usually have a high degree of trust among the public, once lost it can be extremely difficult to restore [5]. Reasons for losing trust are concerns that an individual's data might be used inappropriately, or a lack of clarity [5].

The recent literature has shown that the use of blockchain technology can help mitigate these concerns through its features such as immutability, transparency and reliability [6]. Blockchain technology enables the patient to manage the distribution of their data [6], by giving the patient control over who can access what data, at what time and for which purpose. Additionally, the use of a public ledger gives the patient full transparency on the use of their data, allowing entities to be held accountable for inappropriate data usage.

Recently, research has been done on blockchain-based consent management techniques, with Albanese et al. [7] proposing a dynamic consent managing approach through the use of a private blockchain. Furthermore, Pournaghi et al. [8] suggest a technique in which symmetric cryptography is used to preserve the confidentiality of medical data. Although the recent research is promising, it is at initial stages, lacking technical detail.

In this paper, we will investigate how consent management techniques can be integrated into blockchain-based medical data sharing. We first describe the requirements for an adequate consent management system (CMS) for medical data sharing using blockchain technology. Next, we compare the current state-of-the-art work and present its limitations. Finally, we propose the design of a role-based CMS with fine-grained access control using hierarchical structures, built on the Hyperledger Fabric (HF) framework [9], [10].

The structure of the paper is as follows. In Section 2 we discuss the background information for medical data sharing, blockchain and HF specifically. We then present the requirements for an adequate system in Section 3. Section 4 compares the related work based on these requirements, and discusses its limitations. The design of the proposed CMS is described in Section 5 and its implementation in Section 6. In Section 7 the proposed CMS is evaluated based on the requirements from Section 3. Section 8 discusses unresolved challenges in the proposed work and provides future research directions.

2 Background

2.1 Medical data sharing

The sharing of medical data is vital in developing better healthcare. Digitizing, combining and using big data can lead to benefits such as detecting diseases at earlier stages, better management of individual and public health, and leaner and faster development of drugs and medical devices [11].

However, challenges emerge when sharing medical data. Due to the sensitive nature of medical data, their security, integrity and privacy must be guaranteed. Further restrictions are posed by legislation such as the GDPR [3] in Europe and the Health Insurance Portability and Accountability Act [12] in the US, which further restrict the capabilities when sharing medical data. Additionally, data incompatibility between medical facilities and data fragmentation (a single patient's data distributed across different facilities) make it difficult to gather and process patient data. The use of blockchain technology can mitigate these challenges through features such as immutability and transparency.

2.2 Blockchain

In 2009, blockchain technology emerged after Satoshi Nakamoto proposed Bitcoin [13]. The distributed ledger or *block* in blockchain technology adds a hash of the previous block in each subsequent block, thus providing an immutable chain of blocks. New blocks are validated and the ground truth is established through consensus algorithms, with the most popular algorithm being Proof of Work (PoW) [13], [14]. While Bitcoin allows for very limited scripting, newer blockchains such as Ethereum [15] enable the use of smart contracts which provide Turing-complete programming on the blockchain.

Bitcoin and Ethereum are both permissionless blockchains, which means that anyone can participate in the system. Users are pseudonymous, and the blockchain is public for everyone to view. In another type of blockchain called permissioned blockchain, users need to be accepted into the peer network. Participants know each other's identities, and the distributed ledger remains private among the participants.

2.3 Hyperledger Fabric

Hyperledger Fabric (HF) [9], [10] is such a permissioned blockchain. Similar to Ethereum, HF supports smart contracts (called *chaincode* in HF). Due to its permissioned nature, HF is able to run computationally lighter consensus algorithms such as Practical Byzantine Fault Tolerance (PBFT) [16]. HF has a membership service provider (MSP) which allows different types of users to perform different types of actions. While going against the main principles of blockchain, a centralized authority provides HF with capabilities that are desired for use in enterprises, as well as governed systems.

These features make HF a promising candidate for blockchain-based medical data sharing. It supports the granular access control we are looking for. Moreover, it allows for entities to be held accountable for data misuse, as their identities are known. According to [17], the lighter consensus algorithm permits 3000 transactions per second in HF, whereas Bitcoin and Ethereum can only handle 7 and 15 transactions per second respectively, thus providing HF with better scalability. Additionally, HF does not require cryptocurrency-based incentives, which drastically lowers transaction costs.

3 Requirements for an adequate CMS

In this section, we will present the requirements for an adequate CMS. We need to have insight into these requirements before we can evaluate the current state-of-the-art consent management techniques and identify what is missing. The goal of the proposed design is to meet all of these requirements.

In the explanation of the requirements, we will consider three entities. The **patient**: the person that owns the data. In medical data sharing, this would be the patient. **Data asset**: a piece of data owned by the patient. And a **data processor**: an entity that can request access to a patient's data for processing. This could be a doctor, a researcher or a third party wanting to access the data. Whether the data processor has access to specific data is based on whether the patient has given **consent** to the data processor to access that data asset.

We group the requirements for an adequate CMS into the following five goals: (i) legal compliance; (ii) privacy; (iii) transparency; (iv) usability; (v) accountability.

Legal compliance: According to the GDPR [3], consent must satisfy the following conditions.

- Consent must be fully informed, unambiguous and willingly given. The patient must understand to what they are giving consent, and it should be recorded in writing. Consent may not be implied through a pre-checked box or patient inactivity.
- The patient gives consent for a specific purpose. In medical data sharing, this could be something like *cancer research* or *health check*. If a data processor has already obtained consent to process a data asset for a given purpose, and they want to use the data asset for another purpose, they need to obtain additional consent for that purpose.
- Consent must be granted for a specified time period with a clear beginning and end.
- The patient can revoke consent at any time and for any reason. After consent has been revoked, the data asset may no longer be processed by the data processor. This however does not affect the processing of data that happened before revoking.
- The patient has the right to be forgotten. This means that the data processor must erase all data concerning the patient if they so request.

Additionally, the GDPR [3] states that consent can be overruled in emergency situations or situations benefiting public health. However, these will not be considered in this paper due to their complicated and ambiguous nature.

Privacy: The privacy of the patient must be guaranteed. This means that consent should be dynamic; the patient should be able to give permissions to some parts of their data while withholding access to other parts. Additionally, the permissions set by the patient should only be visible to the patient. A data processor can only see whether they have been granted permission to access the data, but not whether others might have access to the data. This holds for transactions

as well; data processors and patients should not be able to view what transactions are performed by others. Privacy can be categorized by the following three parameters: (a) *Confidentiality*: all data should remain confidential, unless granted access. This includes transactions, as well as patients' permissions and data assets. (b) *Unlinkability*: it should not be possible to link various transactions or data to the same entity, even if the identity of the entity is unknown. This is necessary because multiple transactions can be linked to the entity if their real identity is inferred by a transaction. (c) *Anonymity*: Data should remain anonymous; this is, however, infeasible as health data obviously has to be linked to a patient. The patient should however be able to withhold access to their personal data.

Transparency: The system should be transparent. The GDPR [3] requires that any patient must be able to view what personal data has been collected. Moreover, the authors of [5] state that their study reflects a greater desire for transparency in the usage of data. Therefore, the patient should be able to access their personal records, as well as audit what transactions are performed on their data. This should include who is accessing the data, at what time and for which purpose.

Usability: The system should support both primary and secondary data usage. Primary data usage is the direct use of data, where the data processor is familiar with the patient's identification, and the data is accessed by querying for a specific patient. A simplified example would be a doctor accessing a patient's medical records. Secondary data usage implies that data is requested without specifying a patient. The data that is requested can correspond to groups of patients. An example would be requesting data from cancer patients for cancer research. Moreover, the CMS should not impede the workflow of healthcare and research. It should not take a patient extra effort to give consent to multiple parties, as that would be an extra threshold resulting in less data being shared, and thus impeding research and good healthcare.

Accountability: The system should enable data processors to be held accountable for inappropriate data use. By auditing the blockchain, it can be verified whether data has been used in the correct time frame, for the specified purpose and whether the data processor was given permission to access the data. If data misuse is detected, legal consequences can apply.

Scalability: The rapid growth of technology has led to a significant increase in health data [1], and this growth is expected to continue, generating even more data in the years to come. Therefore, the system must scale well in time complexity with an increasing amount of data. Furthermore, the system must scale well with an increasing number of participants, both patients and data processors.

4 Related work

Multiple designs have been proposed for a blockchain-based CMS for medical data sharing, each fulfilling part of the requirements from the previous Section.

Rouhani et al. [18] provide Medichain, a patient-centric design accessible through a browser or mobile interface. The system is built upon the HF framework using discretionary access control, meaning the patient has authority over their own data. Data processors request transactions, and patients can either accept the terms of the transaction, propose modifications or reject the transaction. However, they only support the primary access of data and do not go into details regarding consent. Also, they do not state how patients' permissions are kept private.

Azaria et al. [19] offer MedRec, a solution built on the Ethereum blockchain. They utilize smart contracts to create representations of medical records. The system is centered around patient-provider relationship contracts (PPR), which are smart contracts containing information about a relationship between the patient and the entity storing and managing the patient's data. The PPR contains pointers to where the data is stored, as well as the permissions for viewership by third parties. They use a DNS-like implementation that

Table 1: Comparison of related work based on requirements.

Citation	Legal Compliance	Privacy	Transparency	Usability	Accountability	Scalability	Implementation details[1]
Medichain [18]	○	◐	●	○	●	○	○
MedRec [19]	○	◐	●	○	●	○	◐
Medicalchain [20]	◐	◐	●	●	●	◐	◐
Consentio [21]	◐	○	●	○	●	●	●
Aldred et al. [22]	◐	◐	◐	○	○	◐	◐
Albanese et al. [7]	●	◐	●	○	●	○	◐
Proposed solution	●	●	●	●	●	○	◐

[1] Not a functional requirement. However, useful for comparison.

No: ○, Partially: ◐, Yes: ●

maps an existing form of ID to the individual's Ethereum address. They propose two ways to incentivize miners to participate in the network. MedRec however does not specify what the consent rules look like, only allows primary usage of data, and has some privacy issues. Additionally, their solution is built on Ethereum, which uses the computationally heavy PoW consensus algorithm.

Medicalchain [20] is a medical data sharing platform with a dual blockchain structure based on HF and Ethereum. Patients can manage their permissions, as well as write information to their health records. Privacy is ensured by encrypting health records using symmetric key cryptography. However, they do not go into details regarding consent, and it seems that permissions are not kept private.

Agarwal et al. [21] present Consentio, a general purpose CMS, however not excluding use in the medical sector. They propose a role-based design using the HF framework. Data processors are grouped into roles, with a trusted third party assigning the roles. Patients grant roles consent to access specific data assets. Access requests are based on the data asset type. Hence, data processors are not able to request access to a specific patient's data. By using an individual-oriented world state, the system's functions scale $O(1)$ in time complexity with increasing number of individuals. However, Consentio does not support direct data access through a patient's ID, which is difficult for healthcare. Also, consent in Consentio does not include a purpose, which is required by the GDPR [3], and they lack details on how privacy is preserved.

Similar to Consentio, Aldred et al. [22] propose a general CMS based on HF. Permissions are identified by a hash of the patient's ID and the data processor's ID, ensuring that data processors are not able to view the permissions concerning others, given that they do not know the IDs of other data processors. These permissions contain Boolean flags corresponding to whether the data processor is allowed access to the respective information. However, it is unclear whether both primary and secondary data usage are supported. Also, they do not go into specifics regarding consent, and patients are not able to see what transactions are being performed on their data, thus lacking transparency.

Albanese et al. [7] provide a dynamic CMS to be used for clinical trials, named SCoDES. The management of a patient's consent is automatically generated according to the features of a given trial. Users communicate with the blockchain infrastructure based on HF through a web application. Assets in the system correspond to trials, which contain details about the clinical trial, and contracts, which represent the official agreement containing information about the data processor, the patient and whether consent has been given. To access a patient's data, an investigator issues a contract. Patients can then give consent to all contracts that involve them. The authors however do not specify how permissions are kept private, and data can only be accessed by the people in charge of the trial. It is unclear how third parties can access data.

4.1 Limitations

Table 1 compares the related work to the requirements set in Section 3. Overall, the related work lacks technical detail regarding consent management. Additionally, almost none of the analyzed systems support both the primary and secondary usage of data. Progress can be made by ensuring the privacy of permissions. To the best of our knowledge, there is no solution that addresses all of the requirements from the previous section, as well as provides technical detail on the implementation of the system.

5 System design

This section will describe the design of the blockchain-based CMS. We propose a fine-grained, role-based CMS, built on the HF framework.

Patients have the role *patient*, whereas data processors can have a variety of roles, such as *doctor*, *nurse* or *researcher*. Note that an individual can have multiple roles, e.g. a person can be a patient, a doctor and a researcher as well. Roles are assigned by trusted third parties, hereafter called *institutions*. These institutions are entities trusted by the public, such as hospitals or government agencies. Roles are tied to the institution; this means that when *hospital X* assigns the role *doctor* to an individual, then that individual is in effect a *doctor at hospital X*. We make the assumption that institutions will not give out inappropriate roles, because this would hurt their good reputation.

5.1 Transactions

Each entity in the system is allowed to perform different transactions. The transactions are listed per entity type below.

Patient:

- *update_consent*: This transaction generates a new consent rule which can either grant or revoke access to a role, from an institution, for a purpose, to a data type and for a specific time period.
- *view_data_asset*: This transaction provides a data asset to be viewed by a patient.

Data processor:

- *request_access_by_patient_id*: This transaction requests access to a specific patient's data asset. The data processor must provide their role, institution and the ID of the patient whose data they are requesting access to. Additionally, they must provide for what purpose they are requesting access, to what data type and for what time period. This transaction corresponds to the primary usage of data as described in Section 3.
- *request_access_by_data_type*: This transaction requests access to data based on the type of data. The data processor does not specify a patient; they request access to multiple patients' data. The data processor provides their role and institution, as well as the purpose, data type and time period they are requesting access for. This transaction corresponds to the secondary usage of data as described in Section 3. Note that labeling data with a type is essential to be able to request data access this way.
- *add_data_asset*: This transaction uploads a patient's data to the blockchain. The data processor labels parts of the data with the correct data type. The data is split into multiple

data assets based on data type. Additionally, the data processor provides a pointer that can be used to retrieve the actual data.

After performing any of these actions, if permission is granted, the data processor will be able to retrieve the data. Note, however, that the mechanics of pointer creation and data retrieval will not be considered, as they are beyond the scope of this paper.

Institution:

- *assign_role*: This transaction assigns a role to a data processor.
- *revoke_role*: This transaction revokes a role from a data processor.

These transactions are validated and the outcomes are stored in the distributed ledger.

5.2 Privacy

To ensure privacy as described in Section 3, we define what the different entities are able to view in terms of transactions and data. In Section 6.3, we describe how this is achieved.

Transactions in the distributed ledger are only viewable by the entities that are involved in them. Patients are able to view what transactions they have performed, as well as what transactions data processors have performed on their data. Data processors are able to view what transactions they have performed on patients' data, as well as the roles that have been granted to or revoked from them. Note that to protect the patient, data processors are not able to view any of the permissions, even if they are assigned the role that is specified in the permission rule. Institutions are only able to view what roles they have granted or revoked.

The data that is stored in HF's world state, which is used to execute chaincode, cannot be viewed by anyone. This ensures that not even the peers that validate the transactions are able to view a patient's permissions or data pointers.

5.3 Hierarchical roles, institutions, purposes and data types

A consent rule created by a patient must contain the following: role, institution, purpose, data type and time period. For highly dynamic consent, it is necessary that the patient can be as specific as they want. A trusting patient will likely want to give consent to all medical staff, for multiple purposes and to the majority of their data. Contrarily, a distrustful patient may want to give consent to only their own doctor, for few purposes and to little data. Data from trusting patients is necessary for good healthcare and research, however, it is a patient's right to choose precisely what they share. Problems arise when increasing the specificity of consent rules, because it might discourage a trusting patient from sharing data if they have to give additional consent to each individual doctor and researcher. To enable distrustful patients, without discouraging trusting patients, we propose that the role, institution, purpose and data type fields of the consent rule are structured hierarchically, in a tree-like fashion.

Figure 1 presents a simplified example of such a hierarchical role structure. Every parent node in the tree encapsulates its children. Only the leaves of the tree are assigned as roles, whereas consent can be given to any of the nodes in the tree. Thus, patients are able to be extremely specific in whom they are giving consent, while still being able to give multiple roles consent with one consent rule. After the tree is constructed, the leaves are numbered in increasing order. For each parent node a range is computed which contains all the leaves for that node. Hence, the system is able to check whether the node is the parent of a given role without having to traverse the entire tree, limiting the computation needed. Note that the participating entities need to decide on the hierarchical structure of the roles before the system is able to operate. The structures of institution, purpose and data type are constructed in a similar fashion.

6 System implementation

This section will explain how the design described in Section 5 will be implemented. As mentioned in Section 2, the HF framework best fits the needs of our system, and therefore we will be basing our implementation on HF.

6.1 Identification

In HF, the certificate authorities (CA) provide identification for the entities participating on the network. Additionally,

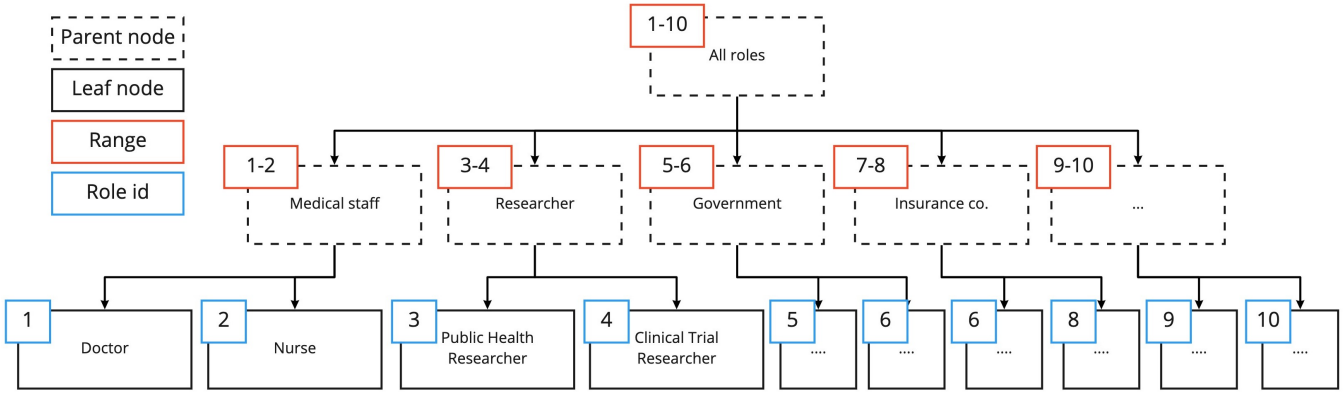

Figure 1: Hierarchical structure of roles.

Patients		
Patient	client_id	existing_id

Roles			
Role	client_id (data processor)	role_id	institution_id

Tree nodes				
Role	role_id	tag	is_leaf	range | integer
Institution	institution_id	tag	is_leaf	range | integer
Purpose	purpose_id	tag	is_leaf	range | integer
Data type	data_type_id	tag	is_leaf	range | integer

Data					
Data Asset	asset_id	data_type_id	client_id (patient)	client_id (data processor)	pointer

Permissions					
Permission	client_id (patient)	role_id	institution_id	purpose_id	data_type_id
	time_from	time_to			

Figure 2: Objects in Hyperledger Fabric's world state.

a membership service provider (MSP) governs which identities are allowed to perform what transactions. Identities that are performing transactions are called *clients*. In our system, there will be three types of clients; these are patient, data processor and institution, as mentioned in Section 5.1. When a client performs a transaction, they sign the transaction with their certificate. The MSP then ensures that these three types of clients are only allowed to perform the transactions corresponding to their type, as specified in Section 5.1.

6.2 World state

In order not to have to traverse all the transactions to retrieve the state of an entity, as is done in Bitcoin [13], HF maintains a world state. The world state is stored in a database that each of the peers maintains. Once transactions are validated and added to the blockchain, the world state is updated, based on the chaincode in the transaction. In our design, the world state will consist of the following aspects, which can also be seen in Figure 2:

Patients: A mapping of existing identification, such as social security number, to patients' IDs. This is necessary for primary data usage.

Roles: Contains mappings of a data processor to a role and the institution that granted it. Note that a data processor can possess multiple roles.

Tree nodes: Contains the tree nodes, as described in Section 5.3. For example, the nodes of the tree corresponding to roles are structured as follows. Each role has an ID and a tag corresponding to the type of role, for example *DOCTOR*. Additionally, it indicates whether it is a leaf or a node in the tree. If the role is a leaf, it contains an integer; if it is a node it contains a range. The tree nodes of institution, purpose and data type are structured in a similar fashion.

Data: Contains all the data assets. Each asset has an ID and the ID of the data type it corresponds to. Additionally, it contains the ID of the patient that owns the data asset, the ID of the data processor who uploaded the data asset, and a pointer to the location of the actual data.

Permissions: Contains all the permissions. Each permission contains an ID corresponding to the patient, the ID of the role that is granted access and the ID of the institution it is issued by. Additionally, it contains the ID of the purpose for which permission is given and the ID of the data type. It also contains the time from and to which the permission is issued. Note that the role, institution, purpose and data type can correspond to nodes or leaves in the permissions.

6.3 Privacy

To provide the privacy on transactions as described in Section 5, we use trusted execution environments (TEE) [23], [24], as well as cryptographic methods. In other blockchain platforms, zero-knowledge proofs (ZKP) [25], [26] are often used to ensure privacy; they are however not applicable for our system, as ZKPs do not support chaincode with unknown data from multiple parties [27].

Trusted Execution Environment

As described in Section 5, not even the validating peers should be able to see the inputs of a transaction, nor the data in the world state. Therefore, each peer will execute the chaincode in a TEE. The authors of [23] define a TEE as "a tamper-resistant processing environment that runs on a separation kernel. It guarantees the authenticity of the executed code, the integrity of the runtime states (e.g. CPU registers, memory and sensitive I/O), and the confidentiality of its code, data and runtime states stored on a persistent memory." TEEs have been developed by multiple parties, such as [28–30]. The world state of HF will be encrypted using a public key that is shared by all TEEs. When chaincode is executed on the TEEs, world state data is accessed and decrypted using

the private key which is stored inside the TEE and therefore unaccessible by the peers.

Encryption

All outputs from the chaincode executed on the TEEs is encrypted as well. Due to the properties of encryption, the validating peers in the network will still be able to compare outputs based on the encrypted text. The outputs are encrypted using a symmetric key, which is sent to all entities involved in the transaction. This way, only the entitled entities are able to view the outputs of the transaction.

6.4 Messaging

In HF, transactions can trigger messages to inform participants of events. In our system, patients receive messages when new data is uploaded that concerns them. They are then able to view the data, and provide new consent rules based on the data. Additionally, they receive a message when access has been requested to their data. Data processors receive messages when consent has been revoked for data they have requested access to. When the data processor receives this message, they are obligated to delete the data if they have it stored in their own database. This way, the right of erasure is guaranteed if a patient revokes consent. Patients, data processors and institutions all receive messages stating whether their transactions have succeeded or failed.

6.5 Chaincode

Chaincode, which is called *smart contracts* in other blockchains like Ethereum [15], manipulates the world state. Transactions can trigger chaincode, and the transaction and its inputs are added to the ledger. The pseudocode for the transactions to be performed by the patient, data processor and institution can be seen in Listings 1, 2 and 3 respectively. Note that the pseudocode is meant to be explanatory, and it is not optimized for time complexity.

Encryption and decryption, as specified in Section 6.3, are left out of the pseudocode for the sake of simplicity. Moreover, the identity of the client that calls the chaincode is verified by the MSP before chaincode execution, and is thus also omitted below.

Listing 1: Pseudocode for transactions performed by patient.

```
Input: patient_id, role_id, institution_id, purpose_id, data_type_id,
time_from, time_to, is_granted
Transaction UpdateConsent:
    if (is_granted = True)
        then add permission with input as fields to world state

    else if (permission with input as fields exists)
        then remove the permission from world state
        send message to data processors


Input: patient_id, asset_id
Transaction ViewDataAsset:
    if (data_asset with asset_id exists &
        data_asset.patient_id == patient_id)
        then return pointer to data
```

Listing 2: Pseudocode for transactions performed by data processor.

```
Input: data_processor_id,
patient_id, purpose_id, data_type_id, time_from, time_to
Transaction RequestAccessByPatientId:
    forall permissions p of patient with patient_id
        if (p.role_id == data_processor_role_id &
            p.institution_id == data_processor_institution_id &
            p.purpose == purpose_id &
            p.data_type == data_type_id &
            p.time_from < time_from &
            p.time_to > time_to)
                then if (data processor has role
                and institution specified in permission)
                    then send message to patient
                    pointers = getDataPointers(patient_id, data_type_id)
                    return pointers

Input: data_processor_id,
purpose_id, data_type_id, time_from, time_to
Transaction RequestAccessByDataType:
    pointers = []
    forall patients:
        forall permissions p of patient
            if (p.role_id == data_processor_role_id &
                p.institution_id == data_processor_institution_id &
                p.purpose == purpose_id &
                p.data_type == data_type_id &
                p.time_from < time_from &
                p.time_to > time_to)
                if (data processor has role
                and institution specified in permission)
                    then send message to patient
                    pointers += getDataPointers(patient_id, data_type_id)
    return pointers

Input: patient_id, data_type_id, data_pointer
Transaction AddDataAsset:
    add data asset with data_type_id, patient_id
    and data_pointer to world state
    send message to patient with patient_id
```

Listing 3: Pseudocode for transactions performed by institution.

```
Input: data_processor_id, role_id, institution_id
Transaction AssignRole:
    add role with role_id and institution_id to data processor
    with data_processor_id

Input: data_processor_id, role_id, institution_id
Transaction RevokeRole:
    if (hasRole(data_processor_id, role_id, institution_id))
        then remove role with role_id and institution_id
        from data processor with data_processor_id
```

7 Evaluating the CMS

To evaluate the proposed CMS, we go over the requirements from Section 3 to check whether they have been met. How the proposed CMS compares to the related work from Section 4 can be viewed in Table 1.

Legal compliance: The proposed CMS meets all the requirements for legal compliance. The patient has full control over who is given consent, this consent is willingly and explicitly given, and this consent is recorded in written form. Permission rules contain a purpose and time period, which are enforced when requesting access. Also, revoking consent is as easy as granting consent and the patient has the right of erasure, thus meeting the requirements set by the GDPR.

Privacy The privacy requirement is met by the CMS, as the consent is highly dynamic. Also, due to the use of encryption no entity is able to view transactions that do not involve them. The use of TEEs ensures the confidentiality of the data in the world state. This achieves the confidentiality metric as well as unlinkability, as no participant is able to link transactions or data to an entity unless the transaction involves them. Additionally, anonymity is preserved, as the patient is able to withhold access to whatever data they please.

Transparency: The system meets the transparency requirement, since the patient is able to view all data that is collected, as well as receives a notification when new data is added. Additionally, patients receive a message when their data is accessed, and are able to view the parameters of the transaction.

Usability: Usability is achieved in the system, because data processors are able to retrieve data for a specific patient as well as by searching for a specific data type. Thus, both primary and secondary usage of data are supported. Furthermore, the way the roles, institutions, purposes and data types are structured enables patients to share information with multiple parties without any extra effort, while still being able to be specific. Therefore, patients are not discouraged from sharing data for research or improved healthcare.

Accountability: The system supports accountability as patients are able to audit all the transactions that are performed on their data, including timestamps. Because the granting and revoking of consent is also stored in the distributed ledger, misuse of data can result in legal consequences, for example if a data processor were to process data after the patient has revoked consent.

Scalability: Unfortunately, due to time constraints, we have not been able to test the scalability of the CMS. However, the use of HF provides a promising prospect, as HF is capable of handling 3000 transactions per second, according to the authors of [17]. The authors of [21] even report being able to handle upward of 6000 transactions per second.

Implementation details: We provide details on how to integrate the CMS into the HF framework; however, the actual implementation of the system is missing. Thus, we satisfy the implementation details metric in Table 1 partially.

8 Discussion and future research directions

Despite satisfying most of the requirements set in Section 3, the proposed work remains incomplete. The solution we provide is merely theoretical, and the actual implementation of the CMS is missing due to time constraints. Additional work is required to implement and test the actual CMS.

To implement the proposed CMS, additional research needs to be done. Firstly, HF's world state is pluggable, thus supporting different databases. Therefore, the compatibility as well as the performance of different databases needs to be explored. The different available TEEs need to be investigated as well. Another research direction is the creation of the data pointer. An important consideration here is that a data processor must not be able to retrieve data after access is revoked.

Additionally, the scalability of the system should be tested, analyzing performance with an increasing number of nodes in the system, as well as an increasing number of patients, assets and permissions in the world state.

Furthermore, different attacks on the system should be explored. The usage of TEEs is a promising prospect for achieving complete privacy in the system; however, they are prone to some attacks, as explained by the authors of [24], [31]. Other well-known attacks on HF must be explored as well. The authors of [32] provide a survey of the known attacks.

Lastly, to maximize the potential of the hierarchical structures throughout the system, additional research needs to be done on appropriate structures for roles, data type, purpose and institutions.

9 Responsible research

In this section, we discuss the integrity and reproducibility of our research.

9.1 Integrity

We have not done any experiments throughout the research, therefore discussing data fabrication, falsification and data trimming is irrelevant to the integrity of this research. Moreover, no experimentation on human subjects has been done. All data that has been gathered is in the form of publications, which are properly referenced throughout this paper. Noteworthy, however, is that to ensure integrity we first set the requirements for a consent management system before comparing the related work. This has been done so that the requirements remain unbiased toward related work; else requirements might have been selected based on what the related work is missing. Additionally, requirements have remained the same throughout the research. We did not remove any requirements if our proposed solution did not meet them, although the proposed solution might have seemed more impressive if we had.

9.2 Reproducibility

The absence of an experiment or results based on data makes it difficult to discuss the reproducibility of the research. However, a way to improve on the reproducibility might be related to the comparison of related work. The gathering of related work can be done in a more systematic manner, for example by using a specified search query and exclusion criteria. However, because the area of consent management for medical data sharing using blockchain technology is so recent and diverse, it might be difficult to find all the relevant work based on a single query. In this paper, we picked some of the more well-known work. The research could be extended upon by comparing additional related work based on the set requirements.

10 Conclusion

In this paper, we proposed the design of a role-based consent management system with fine-grained access control using hierarchical structures. The system provides dynamic consent management for blockchain-based medical data sharing, enabling the patient to be the owner of their own data. The system will be implemented in Hyperledger Fabric, a permissioned blockchain.

The proposed design satisfies the requirements for legal compliance, privacy, transparency, usability and accountability. We provided details regarding the integration of the system into the Hyperledger Fabric framework by describing the interaction between the different components. Additionally, we presented pseudocode for the available transactions by the clients in the system. We diverged from the state-of-the-art work by enabling patients to give consent to multiple parties

without any extra effort through the use of hierarchical structures. Moreover, we provided additional privacy through the integration of trusted execution environments, and the proposed design supports both primary and secondary data usage. Open challenges remain regarding the scalability of the system.

References

[1] R. Fang, S. Pouyanfar, Y. Yang, S.-C. Chen, and S. S. Iyengar, "Computational health informatics in the big data age: A survey," *ACM Comput. Surv.*, vol. 49, no. 1, pp. 1–36, 2016.

[2] Y. Yu, M. Li, L. Liu, Y. Li, and J. Wang, "Clinical big data and deep learning: Applications, challenges, and future outlooks," *Big Data Min. Anal.*, vol. 2, no. 4, pp. 288–305, 2019.

[3] "The european parliament and of the council," [Online] https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32016R0679, 2016, [Accessed]: 2021-4-30.

[4] X. Larrucea, M. Moffie, S. Asaf, and I. Santamaria, "Towards a GDPR compliant way to secure european cross border healthcare industry 4.0," *Comput. Stand. Interfaces*, vol. 69, no. 103408, p. 103408, 2020.

[5] K. Spencer, C. Sanders, E. A. Whitley, D. Lund, J. Kaye, and W. G. Dixon, "Patient perspectives on sharing anonymized personal health data using a digital system for dynamic consent and research feedback: A qualitative study," *J. Med. Internet Res.*, vol. 18, no. 4, p. e66, 2016.

[6] E. J. De Aguiar, B. S. Faiçal, B. Krishnamachari, and J. Ueyama, "A survey of blockchain-based strategies for healthcare," *ACM Comput. Surv.*, vol. 53, no. 2, pp. 1–27, 2020.

[7] G. Albanese, J.-P. Calbimonte, M. Schumacher, and D. Calvaresi, "Dynamic consent management for clinical trials via private blockchain technology," *J. Ambient Intell. Humaniz. Comput.*, vol. 11, no. 11, pp. 4909–4926, 2020.

[8] S. M. Pournaghi, M. Bayat, and Y. Farjami, "MedSBA: a novel and secure scheme to share medical data based on blockchain technology and attribute-based encryption," *J. Ambient Intell. Humaniz. Comput.*, vol. 11, no. 11, pp. 4613–4641, 2020.

[9] C. Cachin *et al.*, "Architecture of the hyperledger blockchain fabric," in *Workshop on distributed cryptocurrencies and consensus ledgers*, vol. 310, no. 4. Chicago, IL, 2016.

[10] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. De Caro, D. Enyeart, C. Ferris, G. Laventman, Y. Manevich, S. Muralidharan, C. Murthy, B. Nguyen, M. Sethi, G. Singh, K. Smith, A. Sorniotti, C. Stathakopoulou, M. Vukolić, S. W. Cocco, and J. Yellick, "Hyperledger fabric: A distributed operating system for permissioned blockchains," in *Proceedings of the Thirteenth EuroSys Conference*. New York, NY, USA: ACM, 2018.

[11] S. Kumar and M. Singh, "Big data analytics for healthcare industry: impact, applications, and tools," *Big Data Min. Anal.*, vol. 2, no. 1, pp. 48–57, 2019.

[12] "Hipaa adminstrative simplification," [Online] https://www.hhs.gov/sites/default/files/ocr/privacy/hipaa/administrative/combined/hipaa-simplification-201303.pdf, 2013, [Accessed]: 2021-5-15.

[13] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," Manubot, Tech. Rep., 2019.

[14] M. Jakobsson and A. Juels, "Proofs of work and bread pudding protocols," in *Secure information networks*. Springer, 1999, pp. 258–272.

[15] V. Buterin *et al.*, "A next-generation smart contract and decentralized application platform," *white paper*, vol. 3, no. 37, 2014.

[16] M. Castro and B. Liskov, "Practical byzantine fault tolerance and proactive recovery," *ACM Trans. Comput. Syst.*, vol. 20, no. 4, pp. 398–461, 2002.

[17] C. C. Agbo and Q. H. Mahmoud, "Comparison of blockchain frameworks for healthcare applications," *Internet Technol. Lett.*, vol. 2, no. 5, p. e122, 2019.

[18] S. Rouhani, L. Butterworth, A. D. Simmons, D. G. Humphery, and R. Deters, "MediChainTM: A secure decentralized medical data asset management system," in *2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*. IEEE, 2018.

[19] A. Azaria, A. Ekblaw, T. Vieira, and A. Lippman, "MedRec: Using blockchain for medical data access and permission management," in *2016 2nd International Conference on Open and Big Data (OBD)*. IEEE, 2016.

[20] "Medicalchain," http://medicalchain.com/en/whitepaper, Oct. 2017, accessed: 2021-6-21.

[21] R. R. Agarwal, D. Kumar, L. Golab, and S. Keshav, "Consentio: Managing consent to data access using permissioned blockchains," in *2020 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*. IEEE, 2020.

[22] N. Aldred, L. Baal, G. Broda, S. Trumble, and Q. H. Mahmoud, "Design and implementation of a blockchain-based consent management system," 2019.

[23] M. Sabt, M. Achemlal, and A. Bouabdallah, "Trusted execution environment: what it is, and what it is not," in *2015 IEEE Trustcom/BigDataSE/ISPA*, vol. 1. IEEE, 2015, pp. 57–64.

[24] V. Costan and S. Devadas, "Intel sgx explained." *IACR Cryptol. ePrint Arch.*, vol. 2016, no. 86, pp. 1–118, 2016.

[25] J. Partala, T. H. Nguyen, and S. Pirttikangas, "Non-interactive zero-knowledge for blockchain: A survey," *IEEE Access*, vol. 8, pp. 227 945–227 961, 2020.

[26] H. Kang, T. Dai, N. Jean-Louis, S. Tao, and X. Gu, "FabZK: Supporting privacy-preserving, auditable smart contracts in hyperledger fabric," in *2019 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. IEEE, 2019.

[27] F. Benhamouda, S. Halevi, and T. Halevi, "Supporting private data on hyperledger fabric with secure multiparty computation," *IBM Journal of Research and Development*, vol. 63, no. 2/3, pp. 3–1, 2019.

[28] A. Fitzek, F. Achleitner, J. Winter, and D. Hein, "The andix research os—arm trustzone meets industrial control systems security," in *2015 IEEE 13th International Conference on Industrial Informatics (INDIN)*. IEEE, 2015, pp. 88–93.

[29] S. Johnson, V. Scarlata, C. Rozas, E. Brickell, and F. Mckeen, "Intel® software guard extensions: Epid provisioning and attestation services," *White Paper*, vol. 1, no. 1-10, p. 119, 2016.

[30] N. Santos, H. Raj, S. Saroiu, and A. Wolman, "Using arm trustzone to build a trusted language runtime for mobile applications," in *Proceedings of the 19th international conference on Architectural support for programming languages and operating systems*, 2014, pp. 67–80.

[31] A. Moghimi, G. Irazoqui, and T. Eisenbarth, "Cachezoom: How sgx amplifies the power of cache attacks," in *International Conference on Cryptographic Hardware and Embedded Systems*. Springer, 2017, pp. 69–90.

[32] S. Brotsis, N. Kolokotronis, K. Limniotis, G. Bendiab, and S. Shiaeles, "On the security and privacy of hyperledger fabric: Challenges and open issues," in *2020 IEEE World Congress on Services (SERVICES)*. IEEE, 2020, pp. 197–204.