IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received August 2, 2021, accepted August 18, 2021, date of publication September 7, 2021, date of current version September 17, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3111008

A Performant Protocol for Distributed Health Records Databases

MICAEL PEDROSA[1,2], **RUI LEBRE**[1,2], **AND CARLOS COSTA**[1]
[1]Institute of Electronics and Informatics Engineering of Aveiro, University of Aveiro, 3810-193 Aveiro, Portugal
[2]Faculty of Computer Science, University of A Coruña, 15405 A Coruña, Spain

Corresponding author: Rui Lebre (ruilebre@ua.pt)

This work was supported by the National Funds through the FCT—Foundation for Science and Technology under Project UIDB/00127/2020.

ABSTRACT Electronic Health Records (EHR) have a distributed nature and can be managed by distinct affinity domains. Sharing patient health information across distinct organisations helps to deliver a well-informed diagnosis, improving the quality of healthcare service. The federation of those information systems can take the form of a distributed database where data are partitioned and possibly replicated across distinct computational systems. However, the benefits of having a distributed system, such as consistency, availability, and data protection, are mostly absent. This article proposes a distributed database consensus protocol designed to improve the performance of EHR insertion operations, a particularly critical issue in medical imaging cases due to the data volume. It explores the personal and non-transferable nature of EHR and the proposed methodology reduces the data contention through data isolation, improving the overall retrieval performance and detection of misbehaving parties. Furthermore, the proposal follows the recent European General Data Protection Regulation (GDPR), which states that appropriate mechanisms should be used in order to protect data against accidental loss, destruction, or damage, using appropriate technical or organisational measures.

INDEX TERMS Blockchain, consensus protocol, distributed ledger, distributed databases, electronic health records.

I. INTRODUCTION

The proliferation of medical imaging acquisition equipment and the introduction of new modalities with high temporal and space resolution has resulted in a data explosion crisis in healthcare institutions [1], [2]. This presents a challenge for storage [3], data flow, security [4], interoperability [5] and document sharing. Health records are increasingly electronic but are often still trapped in silos. Teleradiology [6] is becoming an important model to access Electronic Health Records (EHR) anywhere, at any time and through any device. Providing direct access to these records will not only be a right under the most recent European legislation (i.e. GDPR[1]), but it can also improve collaboration between physicians and researchers. Computer-Aided Diagnosis (CAD) systems and annotation platforms [7]–[9] could benefit greatly from easily shareable EHR, and consequently relieve the burden on physicians already in short supply [10], especially in mass screening programs where disease coverage is low. However, for such a scenario to emerge, it is necessary to provide a good foundation for EHR to be available in many different organisations, in a GDPR-compliant manner [11].

The main goal is to lay down the foundation for high throughput, high availability and secure federation for healthcare records, and in particular for medical imaging, which represents an extreme scenario due to the volume of data produced. Fig. 1 presents the architecture overview for the main scenario of image storage and retrieval. The usual data flow is to collect images from acquisition devices (X-Ray, MRI, CT, etc.) commonly denominated as modalities, store them in a Picture Archiving and Communication System (PACS) and later retrieve them from visualisation workstations. Our goal is to provide a similar scenario under a federation of distributed parties where records of the same patient, held by different organisations, are merged in the same system. The final system can be seen as a distributed database where patient medical imaging data are partitioned and possibly replicated across distinct computational systems. In a single-client

The associate editor coordinating the review of this manuscript and approving it for publication was Gang Wang.

[1]GDPR: https://gdpr-info.eu

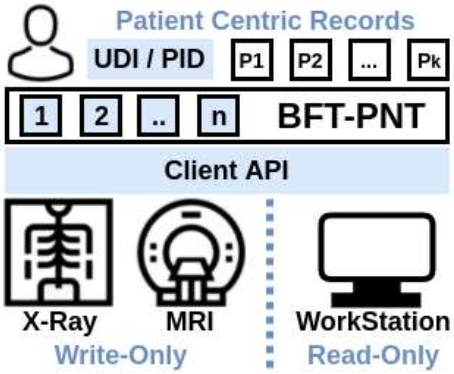

FIGURE 1. A scenario with two types of clients. Write-only X-Ray and MRI devices, and a read-only visualisation workstation. These clients write and read patient's records through a federation of BFT-PNT parties.

database, a client can modify the database without concern about other users modifying the same data at the same time. However, in a multi-client database, simultaneous transactions can try to update the same data at the same time leading to write contention or busy state (performance impacts). So, it is vital to control data concurrency and data consistency to produce meaningful and consistent results.

This article proposes and specifies a Byzantine Fault Tolerant (BFT) [12] consensus protocol for Personal and Non-Transferable records (BFT-PNT), which optimistically assumes that clients are aware of their own transactions and optimises the write contention issue in the EHR context. Under normal conditions the protocol should work as a replication mechanism with minimum overheads and, in the case of client failures, it works mostly as a protection and recovery mechanism. The BFT-PNT protocol is the primary foundation for the proposed system architecture, providing some advantages over more generic BFT protocols due to the nature of the EHR data and the specificity of the protocol. The BFT-PNT features can leverage existing PACS workflow management [13] and cloud-based frameworks [14] as a source of reliable and secure data. It is expected to be competitive due to its low contention.

In the validation section we have set feasibility tests that show the results for inserts/updates and retrieval operations, as well as a Tendermint[2] comparison of transactions throughput and latency.

II. BACKGROUND

In healthcare many small-medium institutions do not have the resources or specialists to be able to review all kinds of modalities and are forced to contract external services, resulting in EHR being spread over distinct organisational domains. Moreover, data outsourcing to the cloud alleviates institutions from the burden of local storage and maintenance while providing high availability and scalability [15].

Several reports in the literature are related to distributed PACS issues, including studies about the performance of DICOM routing and cache mechanisms [16], leveraging cloud services to reduce IT infrastructure and improving availability [17] or scaling to big data [18]. Efforts are also being made to provide cross-enterprise document sharing (XDS) in the Integrating the Healthcare Enterprise (IHE) initiative. However, there are concerns about data integrity [19] and security [20] within third-party providers. Moreover, distributed architectures may be a good fit to improve disaster response in healthcare [21]–[24] and also to apply different big data processing paradigms, such as MapReduce [25].

Cloud providers are commonly seen as "honest-but-curious" [26]. To protect data in those environments, some common methods include, searchable encryption [27]–[29] and pseudonymity [30], [31] schemes. Nevertheless, high availability through fault tolerance is only within the reach of some distributed technologies. Healthcare is evolving from traditional to more robust and distributed solutions with people encouraged to put their personal healthcare data into cloud applications [32], [33]. In parallel, distributed ledger technologies (DLT) are being increasingly adopted to exchange patient data [24], [34]; build Healthcare Data Gateways [35] prevent scattered information throughout multiple healthcare systems; integrate distributed health records [36]; secure patient records [24], [37], and turn the ledger into an automated access-control manager [38]. Solutions are being studied[3] [39]–[42] and deployed to handle the natural difficulties of decentralised EHR.

A. CONSENSUS PROTOCOLS

Associated with these DLT technologies are very important Byzantine Fault Tolerant (BFT) consensus protocols [43]. These protocols protect the underlying data from many types of faults, such as node failures, dishonest and misbehaving nodes, network partitions or even predictive failures due to maintenance. Several consensus protocols and variants are available in the historical bibliography [44]–[46]. These sometimes proved to be more difficult to implement than originally anticipated [47] and even harder if byzantine faults [12] are considered. Any byzantine consensus solution must comply with two basic conditions in the presence of t faulty processes up to $3t + 1$ total processes: the **finalisation**, where every non-faulty processes eventually chooses an irreversible decision value for the transaction; and **agreement**, where the decision value is unanimous for every non-faulty process related to the same transaction. Classic distributed consensus protocols fall into two major categories:

Leader-Based is defined when a leader is selected on the network to decide what transactions are to be committed, such as RAFT [45], [48]. In this way, concurrent transactions are resolved at one central point. Leader election is an expensive process, but it is supposed to be a rare occurrence. Networks of this type are moderately fast, around 10000 transactions per second with a latency in the order of 100ms. Yet, due to the existence of a central decision-maker, pure leader-based

[2]Tendermint: https://www.tendermint.com

[3]Medibloc: https://medibloc.org/en

byzantine protocols are not possible and the network is susceptible to denial-of-service (DoS) attacks.

Vote-Based. These have no leader election, i.e. Hashgraph [49]. In a rough definition, consensus is determined when votes from a majority (over 50%) or super-majority (over 33% for byzantine protocols) are collected. Since there is no single point of failure, it is resilient to DoS attacks, maintaining the high availability status. When every vote is known and confirmed by every other node, byzantine protection is possible. However, this normally requires $O(n^2)$ messages of n participants and multiple phases (prepare, commit) before reaching agreement, and thus, pure vote-based consensus systems have (in general) throughput and scalability issues.

Modern BFT protocols [49]–[52] consider all types of data and clients as the same, and are not designed to handle specific cases of data concurrency, storage and retrieval of big chunks of data, or to collaborate with the client in the consensus process. The characteristics of EHR, such as being personal and non-transferable, can provide some optimisation paths to these protocols. For instance, sequential consistency and linearisability [53] (cause of major performance impacts) are not always necessary if records have no change history, and the client can be trusted to control the sequence of events. In fact, this is already assumed for DICOM files where acquisition timestamps are part of the content and are provided by the acquisition device. Furthermore, patient centric records are isolated from each other and even sessions from the same patient, making certain concurrency management procedures unnecessary, unless those records need updates.

B. DATA INTEGRITY

Data integrity is the system's capacity to maintain data intact and unaltered throughout its life-cycle. Possible ways of corrupting data fall into two categories: technical issues and security flaws. Ledger integrity is maintained by using cryptography, in order to detect incorrect blocks of data (normally related to security flaws), and by decentralisation, in order to protect against technical issues. Our proposal uses both methods to maintain data integrity. In the context of this article, there are two important concepts related to integrity:

- **Data consistency** means that each user sees a consistent view of the data, including visible changes made by their transactions and those of other users.
- **Data concurrency** means that many users can access the same segment of data at the same time in a consistent view.

Maintaining consistency under data concurrency is the cornerstone of consensus protocols, and in the end, the requirement with the biggest impact on read/write contention and system performance.

C. DICOM

DICOM [54] is a standard defined by the NEMA[4] with universal level acceptance amongst imaging equipment vendors. It is used to store and transmit medical images, and includes simple workflow capabilities, such as the Modality Worklist [55]. Apart from the pixel data, DICOM files have a rich metadata structure with patient demographics and a detailed description of how the image was produced. DICOM was essentially designed for communication between image stations, archives and acquisition equipment, with simple query capabilities and no concerns about data consistency, availability and data protection.

When manipulating big chunks of data such as Whole Slide Imaging (WSI) [56] where an acquisition is split in thousands of tiles that can reach the Gigabyte scale, we can leverage the distributed nature and replication mechanisms of BFT protocols to retrieve different image blocks (or slices) from multiple parties. However, in BFT environments, each slice must be individually checked for consistency [57]. Although this methodology can be deployed using existing protocols by providing simple replication mechanisms in parallel with the consensus protocol, the proposed approach additionally simplifies the protocol by adding more elaborated client participation. A pre-serialisation mechanism was added to decide the order of some transactions, already described in Singh *et al.* [58]. The main difference is that this pre-serialisation is provided directly by the client using a round counter for each client.

III. PROTOCOL PRELIMINARIES

This article proposes the BFT-PNT consensus protocol, which starts by assuming that the client is trustworthy enough to resolve some consensus conflicts and pre-serialise the sequence of updates, improving read/write contention. Our transaction proposer is always the client, with their separate database and round counter. Each client has their independent consensus state where records are isolated from each other, defining those records as personal and non-transferable. From these assumptions, some optimisations can be made to existing BFT protocols in the context of EHR and in particular medical imaging records. Our proposal improves on other general BFT schemes by adding some semantics into its architecture, creating a distinction between two types of transactions:

- **Inserts** represent the creation of an isolated record. These transactions are idempotent and non-ordered, and there are no conflicts or write contentions. Inserts are performed by direct commits to the database.
- **Updates** have order and may generate conflicts requiring rounds of proposals and commits until a value is established.

These are reasonable assumptions due to the described nature of healthcare records. Order of inserts does not matter (we trust in the original data timestamps) and there are no interrelated data references between records. It was also assumed that the source of the data (hospitals, clinics, healthcare organisations) is responsible for the integrity of the submitted data content. Moreover, we optimistically assume that the primary workload is provided by inserts and that updates should

[4]NEMA: https://www.nema.org

be infrequently used. Other characteristics of the proposed protocol are:

- Update conflicts are explicitly resolved by the client, minimising the number of consensus messages. With the help of the client, update transactions can be completed in two rounds using a minimum of $2t + 1$ parties.
- Hash fingerprints provide integrity checks. These are also used as record identifiers. Incremental integrity checks of data slices are also provided by the core protocol, both in storage and retrieval. This can promptly identify faulty parties and discard them before the whole data block is downloaded.
- Write-only clients cannot update records without having access to the original data. Hash fingerprint verification will fail in such cases. This adds additional protection against compromised write-only clients.
- Records are not persisted as blockchain structures, and the "right to erasure"[5] can be supported via administrative procedures. However, tamper-resistance is still achieved via digital signatures.

A. DEFINITIONS

Some definitions are needed before describing the system architecture:

Definition 1: Let $H_q(D) \rightarrow \{0, 1\}^q$ define a deterministic one-way hash function with pre-image and collision resistance, outputting a q number of bits.

Definition 2 (Liveness): If t is the number of Byzantine fails allowed, the protocol can only progress with a minimum of $2t + 1$ parties.

Definition 3 (Conflict): Two or more conflicting updates are defined for the same client and update index (UDI, I_1) with different fingerprints $(D_1, D_2, .., D_k)$, independently of the round.

Definition 4 (Consistency): By consistency, we mean that the most up-to-date finalised commit is always available, and the commits are retrieved in the same order as the client submitted them.

Definition 5 (Finalisation): A commit is finalised when $n - t$ parties or more have the same commit message, where $n-t \geq 2t + 1$. The finalisation is true even if the client does not receive the set of acknowledges; this only signifies that the client does not yet know about the finalisation.

An important note about finalisation and the FLP impossibility [59]. The FLP result states that it is impossible to reach consensus in a deterministic asynchronous setup. For that case, the BFT-PNT works in two modes. From the point of view of the parties it is non-deterministic and asynchronous. Parties have no time assumptions, and do not know if a transaction is ever finalised, not being their responsibility. From the point of view of the client it is deterministic and synchronous. The client waits for all $n - t$ responses in order to finalise the transaction.

[5]GDPR Art. 17: https://gdpr-info.eu/art-17-gdpr

B. SETUP CONDITIONS

Concerning the network conditions, the existence of a quorum of parties Q_j with pre-configured (t, n) parameters, network addresses and public keys for each party is assumed, where $n \geq 2t + 1$, and $3t + 1$ is the minimum necessary to tolerate t Byzantine failures. Each party is identified by the public key $P_i = s_i \times G$. It is assumed that clients have a unique digital identity (UDI) and an authentication key pair $P_a = s_a \times G$. The s_a/P_a key pair is used as a simplified mechanism to authenticate in the system, but we open our model to changes that would not require permanent asymmetric keys. Instead, ephemeral keys can be used with other authentication mechanisms such as user/password and multi-factor authentication (MFA) schemes.

Authorisation procedures are not part of the core protocol. However, any extensions to the protocol can be created by plugging modules capable of intercepting and interpreting the content of the data blocks. In this way, we assume that such extensions can provide authorisation procedures, data representation transformations, storage providers, custom constraint validators, indexation services, etc.

1) COMMUNICATION MODEL

The existence of open channels for broadcast and point-to-point messaging is assumed, connecting a set of parties P_i, $i \in [1, n]$ and a client C. Secure stream channels are provided by Diffie-Hellman key exchange and symmetric encryption, assuming that clients already known the public keys of the parties.

2) FAILURE MODEL

t parties may fail intermittently leaving only a quorum of usable $n - t$ parties, where $2t + 1$ is the minimum required for the protocol to progress.

3) THE ADVERSARY

It is assumed an adversary that can corrupt up to t parties of the usable quorum $n - t$, learning all the individual state of the corrupted parties, listen to all messages and deviate from the protocol in arbitrary ways. Clients are responsible for maintaining their security.

IV. SYSTEM ARCHITECTURE

This article proposes a vote-based consensus protocol for personal and non-transferable health records. BFT-PNT is based on the ideas of *Practical-BFT* [43], *Istanbul-BFT* [60] and *Conflict-free Quorum-based BFT Protocols* [58] by removing some features of the protocol and assigning those as the client's responsibility. The next sections will describe in detail the mechanism developed.

A. PROTOCOL MESSAGES

Transmitted messages are specified in the form $s\text{-}T_x\langle S\rangle$, where s represents the origin of the message corresponding to a s/P_s key pair, T_x is the message type T and sub-type x,

and S the transmitted structure representation. These are all digitally signed with the origin key s. The *UDI* is the client identification implicitly present in every message, where a_a/P_a is the client's authorisation key pair and R_a is the respective round for the client. Each message is related with a genesis data block B where $F = H_q(B)$ is the hash fingerprint that uniquely identifies the **record**. The corresponding data for F may be present at the end of the message, or a stream channel may transmit it. The public key P_s is assumed to be in the message if the respective implementation requires it (in order to have all the required fields to verify the signature).

Insert and update messages can have extended data that is not part of the message signature, such as the data block B (for small block sizes) and a replica set $\{^iR\}$, where $^iR = \langle P_i, \sigma_i \rangle$ contains the party replica signature σ_i for the message. The main messages are defined as:

Propose (Update) - $a\text{-}P_u\langle F, P_o \rangle$ with $P_o = \langle I_k, D_k, R_a \rangle$ is a proposal that must be submitted and accepted before committing an update. I_k is the update order index in relation to the original data F, $D_k = H_q(B_k)$ is the hash fingerprint for the updated data block and R_a the proposal round number of the client. B_k can only be correctly derived if the client has the most up-to-date block, where $B_k = B \cup \Delta_1 \cup .. \cup \Delta_k$ is the union of the original B with all delta changes.

Reply (Vote, No-Data, Receiving, Ack) - $i\text{-}R_{v,n,r,a}\langle F, Q_i, P_i, P_o?, \sigma_i? \rangle$. These are sent by parties P_i in the quorum index configuration Q_i in response to proposals or commits, referencing the corresponding F data. P_o is only needed for update replies, and σ_i is the replica signature of the insert or update message, only used in Ack replies.

Commit (Insert) - $a\text{-}C_i\langle F, T, S? \rangle$ with $S = \langle s, \{S_i\} \rangle$, is a direct commit to insert a data block. There are no concurrent or conflicting inserts, and so there is no need to propose the value. T is the type for the data representation, i.e. DICOM. S is an optional data slice structure defining the size of each slice s and $\{S_i\}$ an ordered set of slice fingerprints. These fingerprints are useful to validate the individual integrity of each slice.

(Update) - $a\text{-}C_u\langle F, Q_i, P_o, \{^iV\}, S? \rangle$ is an update commit where the client must collect $n - t$ vote replies to construct a set of $\{^iV\}$, where the vote structure $^iV = \langle P_i, \sigma_i \rangle$ is composed of the party P_i public key and the signature σ_i from the reply vote. The signature is verifiable since the update message has all the required information to reconstruct the vote reply. The $n - t$ value should be consistent with the reported value in the Q_i quorum configuration.

Get (Consult) - $a\text{-}G_c\langle F, I_k? \rangle$. Requested by the client to $n-t$ parties to get the most up-to-date committed data block. I_k is optional and explicitly specifies the required update index.

(Retrieve) - $a\text{-}G_r\langle F, I_k, i? \rangle$. Requested via a stream channel for the selected party that has access to the respective record F, the update index I_k and slice index i. The slice index is an optional field to balance the retrieve request through several parties; if not present, the whole data block is assumed.

Error - $i\text{-}E\langle F, E_c, I_k?, M? \rangle$. A message to report different classes of errors. I_k is only used to respond to updates. The message includes the error code E_c and an optional error message M.

Expected error codes are:

0: **Internal Server Error**. For any non-identifiable error.

1: **Unauthorised**. To report a non-existent *UDI* or an unauthorised client key P_a.

2: **Invalid Information**. To report incorrect signatures, fingerprints or parameters.

3: **Constraint Error**. To report failures of custom constraints.

Error Handling. Errors cannot usually be handled by the normal flows of the protocol, although a specified number of retries should be performed before the failure is considered unresolved. Errors should be reported to the client user interface if they make sense; otherwise, they should be part of the local log and reported to a central log repository if available. Parties should also report logs to this repository.

B. CONSENSUS RULES

Some general rules are:

- Parties should always verify the correctness of digital signatures before accepting any messages and any fingerprints of the data block.
- Parties should accept concurrent inserts since only updates can generate conflicts.
- Proposals are assigned to monotonically incremental rounds, and each client has only one designated round in an instant of time. However, a client may try to force the use of an incorrect round.
- Each record has a unique active proposal status or alternatively the corresponding vote. Proposals from different records do not interfere with each other.

Other specific rules that the BFT-PNT protocol should strictly obey are:

- Proposals can only be overridden by other proposals of higher rounds, commits of the same or higher rounds, or a commit of a lower round with $n - t$ replicas.
- Commits can have concurrent proposals of higher rounds for the same value.
- Conflicting commits can only be overridden by other commits of higher rounds.
- An update commit is only valid if it has $n-t$ votes from different parties for the same proposal.
- Reads should consult $n-t$ parties to get the most up-to-date commit.
- If a read detects less than $n - t$ replicas for the most up-to-date commit, it should resume the commit until it reaches $n - t$ parties before accepting the read as consistent.

C. SUBMISSION PROTOCOL

All requests are idempotent, i.e. there is no additional effect if they are called more than once with the same

input parameters. The BFT-PNT protocol has the following defined procedures:

Common Replies define the state of a data block on the local storage of a party. A **no-data** reply requests to submit the block via a secure streaming channel. A **receiving** reply reports that it is receiving the data from other sources and it will reply with an **acknowledge** once it is finished. When such a response is received, the request is re-transmitted via the streaming channel followed by the data block.

1) SUBMITS

The client initiates a data submission by sending an insert commit or an update proposal to a randomly selected party (the coordinator) from the illegible client quorum. The coordinator acts as a forward proxy; however, it may be preferable to send those requests directly from the client, depending on the client's network throughput. Only the original block B or corresponding Δ_k is sent through the secure stream channel. The coordinator replicates the submission and data to other parties. The client expects to receive $n-t$ acknowledges or votes (for inserts or updates, respectively). Parties reply with:

- Insert commits are acknowledged if accepted or trigger a common reply for non-existing data.
- A proposal triggers a new reply vote $i\text{-}R_v\langle F, Q_i, P_i, P_o?, \sigma_i?\rangle$ if accepted, a reply vote for an already accepted proposal (refusing the submitted one) or returns an existing commit. It triggers common replies for non-existing data.

2) UPDATE COMMITS

Once the required votes for the proposal are gathered, the client requests a commit $a\text{-}C_u\langle F, j, P_o, \{^iV\}, S?\rangle$ from the coordinator expecting to receive $n-t$ acknowledges, finalising the commit. These commit proofs are stored with the data in all parties. If it fails in this phase, the client cannot conclude that the data is committed, and it must resume the commit procedure (a different coordinator can be selected).

3) RESOLVING CONFLICTS

When a client identifies a conflict, they should try to resolve it. Possible conflicts are in the following range:

- The client receives a vote for a different proposal than the one submitted. The client should increase the round number to be higher than any received vote and propose the value with this new round.
- The client gets an existing commit when proposing a value. If the current round is higher than the received one, the client can resume the proposal, trying to get a commit of a higher round and override the existing one. If there are no illegible $n-t \geq 2t+1$ votes the client is forced to resume the received commit if not already finalised.
- The client gets a vote for a higher round proposal when trying to commit. The client should try to propose the same value with a higher round. Commits of equal value will accept this proposal.

4) REPLICATION

Parties that already have commit messages can resend those messages to other parties with included replicas $\{^iR\}$. Each party then proceeds to update the local representation of the commit message with the new replicas of the received message and reply with an acknowledge $i\text{-}R_a\langle F, Q_i, P_i, P_o?, \sigma_i\rangle$. The reply is then used to construct a replica $^iR = \langle P_i, \sigma_i\rangle$ to update the original commit message. A replication process can receive common replies from other parties if it lacks the data block. Replication may fail due to a conflict with the local state. Conflict resolution should be made as follow:

- If it conflicts with an existing local commit of higher round it should reply with the local commit. In this process, parties having the most up-to-date commits should win the replication process.
- If it conflicts with an existing local proposal of higher round it should reply with the current vote for that proposal. If $n-t$ commit conflicts are received, the party should accept the commit as finalised and reply with an acknowledge.

D. RETRIEVING PROTOCOL

All received data should be verified against the fingerprints (F or $\{S_i\}$) of the corresponding requested commit. Invalid fingerprints should be reported with an invalid information error.

1) CONSULT

The client starts by sending a get-consult $a\text{-}G_c\langle F, I_k?\rangle$ to $n-t$ parties to get the most up-to-date commits (inserts or updates) for F, expecting to receive the same number of replies. A no-data reply or only one up-to-date commit may be received if there are byzantine failures.

2) RETRIEVE

The client selects the most up-to-date commits and opens secure stream channels to any chosen number from these chosen parties. The whole data or selected slices are requested via get-retrieve. Note that, although the client may start to retrieve the data immediately after identifying the correct party, they should not consider the commit as finalised until seeing $n-t$ consistent commits for that data.

Clients may try to force the commit finalisation to accelerate the retrieve process. Even clients that are not the original creators of the commit messages can try to force finalisation, which can be done by just replicating the already existing commits. Clients should also check the integrity of individual slices; if any of those checks fails, the client can select another eligible party to get the corresponding slices.

E. QUORUM MANAGEMENT

The quorum configuration is maintained in a special store uniquely identified by the "admin" *UDI* in a record type "quorum". This configuration is set for the first time using an insert to that store containing t and a set of party

configurations $\{\langle P_i, {}^iA\rangle\}$, with the public key P_i and the network address ${}^iA$. The size of the set corresponds to n. Changing the quorum is done by an update with a proper client authorisation, as in any other record.

Each store maintains a quorum state with a record of the same type referencing the update index of the original quorum configuration. This defines the quorum configuration of the store. A submitted proposal can only be accepted if the store is referencing the last quorum index, otherwise it is considered "out of sync". This error state is resolved by updating the quorum index; however, the proposed quorum change is only accepted if all existing commits are already replicated to $n-t$ parties. The process may lock the storage momentarily, but this is minimised since it is not a global database lock, and many stores may already have all the commits replicated.

Only proposals can disrupt data consistency by trying to create concurrent commits that are not supposed to exist. Those inconsistent commits can be created if there are enough gaps in the new quorum to obtain $n - t$ proposals. The replication process fulfils those gaps with existing commits, preventing those proposals from being accepted. In this case, it is essential that commits are still accepted in "out of sync" states and for old quorum indexes. These types of messages can be sent as part of an unfinished replication process.

V. ANALYSIS OF CORRECTNESS

This section describes a proof of correctness for the BFT-PNT protocol by proving the consistency of inserts, updates and reads with a set of lemmas and theorems. These proofs will be set first on a static quorum of n parties and then extended to handle (n, t) quorum changes.

A. UNDER A STATIC QUORUM

Lemma 1: Inserts cannot provoke inconsistent states if the one-way hash function is assumed to be preimage and collision resistant.

Proof: As previously defined, records are isolated from each other. From the lemma assumptions, if the F_n fingerprints are distinct, inserts belong to different records. Records may refer to other records but this is part of the data content (not the protocol), and it is the client's responsibility to verify if foreign references are already finalised.

Lemma 2: On a set of n parties, there cannot be two concurrent commits of the same round.

Proof: A commit requires $n - t$ consistent votes of the same round. In the presence of t dishonest parties there are $(n - t)-t = n - 2t$ honest proposals for the current round and commit. A second commit has $(n) - (n - 2t) = 2t$ free parties without proposals, but it will always require to increase the round to override the proposal of the missing party and acquire the minimum of $2t + 1$ votes.

Lemma 3: Once a commit is finalised, it is not possible to create a second concurrent commit of a higher round.

Proof: From lemma 2 one can also conclude that there are at most $2t$ parties without the finalised commit. Since proposals cannot override commits, there is no way to get the minimum of $2t + 1$ proposals to emit a commit of a higher round.

Lemma 4: A set of t dishonest parties cannot force the finalisation of multiple concurrent commits.

Proof: In a total set of P^n parties where $n \geq 3t + 1$, t parties can split this set in honest O and dishonest D sets of optimal sizes $O^{n-2t} + D^t + O^t = P^n$. The dishonest set D^t is only able to finalise commits for the joining partition $D^t + O^{n-2t} = A^{n-t}$, and from lemma 2 the O^{n-2t} partition must have the higher round of honest proposals. From the finalisation lemma 3, it is not possible to create a concurrent commit from the $O^t + D^t = B^{2t}$ partition, since the minimum is $2t + 1$.

Theorem 1: The client or any dishonest party cannot force an incorrect round to make the protocol progress to an inconsistent state.

Proof: From lemma 1, there are no round parameters to use in inserts. From lemmas 2, 3 and 4 one can conclude that, even if a client or party forces an incorrect round, only a commit with a higher round can win the conflict forcing the abortion of all other concurrent commits.

Theorem 2: Consistent updates are guaranteed if the client always uses n − t parties to finalise the update.

Proof: If two conflicting updates (D_1, D_2) for the same index I_1 are both to be committed, there must be at least $(2t + 1) + (2t + 1) = 4t + 2$ proposals of the same round, a value out of the range of lemma 2.

Theorem 3: Consistent reads are guaranteed for finalised commits if the client consults n − t parties.

Proof: It is easily seen that, in a configuration of $n \geq 3t + 1$ tolerating a t-partitioned network and t-dishonest parties, since the finalised commit is replicated to $n-t$ parties by removing all the failures we get $(n - t) - 2t = 3t$. At least one honest party from the $3t + 1$ set has the most up-to-date commit. That party should respond to the client get-consult and should be selected to retrieve the correct data block. It should also be able to proceed with the replication process to other working parties.

B. EXPANDING TO QUORUM CHANGES

Theorem 4: A change in n does not affect the finality of existing commits.

Proof: Let the new configuration be $n_2 = n_1 + x$. From lemma 4, t dishonest parties can divide the quorum into $O^{n_2-2t} + D^t + O^t = O^{n_1-2t} + D^t + O^{t+x}$ where O^{n_1-2t} are the honest parties that have the commits from the n_1 configuration. Dishonest parties can try to finalise an inconsistent commit for the joining partition $D^t + O^{t+x}$, resulting in $2t+x = 2t+(n_2-n_1)$. But since $2t+x \geq 2t+1$ should always be true, a value of $x < 1$ does not produce an effect. For the minimum $n_1 \geq 3t+1$ then $2t+x = n_2-t-1$ is one less than necessary to emit a conflicting commit, and so the x change does not affect the original proof of lemma 3.

Theorem 5: A change in t does not affect the finality of existing commits as long as the replication is finished to the new configuration.

Proof: Let the new configuration be $t_2 = t_1 \pm x$. In the transition phase t_1 is maintained and theorem 4 holds. The new configuration is activated when $n - t_1 \geq 2t_1 + 1$ parties vote in the migration, which assures that old commits are replicated to $n - t_2 \geq 2t_2 + 1$ before the new quorum is active, maintaining the finality of existing commits.

Theorem 6: A change in t cannot finalise multiple conflicting commits.

Proof: Let the new configuration be $t_2 = t_1 + x$. If t dishonest parties divide the quorum into two non-finalised conflicting commits $O^{n-2t_1-1} + D^{t_1+1} + O^{t_1} = O^{n-2t_2+2x-1} + D^{t_2-x+1} + O^{t_2-x}$. Then the joining partitioning should be $2t_2 - 2x + 1 \geq n - t_2$. For the minimum $n \geq 3t_2 + 1$, the last equation is reduced to $-2x \geq 0$. Then we must have $x \leq 0$ to produce an effect, but this means that the other partition with $n - 2t_2 + 2x - 1$ cannot finalise. We can generalise this to multiple conflicts since the requirements are higher than for two conflicting commits.

VI. VALIDATION

This section describes the performance evaluation that was carried out for a prototype implementation of BFT-PNT protocol. The report assumes the optimistic case where all nodes are behaving honestly. This validation aims to prove the feasibility of the protocol scheme under low contention.

Implementation details. A prototype implementation[6] of the BFT-PNT protocol was developed in Xtend/Java. We used the Ed25519 signature primitive and the SHA-256 hash function for fingerprints from the bouncycastle[7] library. Messages are delivered via UDP channels using the netty.io[8] library. Stream channels or encrypted transmissions are not implemented in this prototype.

Unit tests are available at the `bft4pnt.test` package to verify the correct implementation of the consensus rules, replication process and quorum evolution.

A. EVALUATING FEASIBILITY

We carried out validation tests using a machine equipped with a Intel i7-7700HQ CPU with 4 physical cores and 16GB of memory. Storage is simulated in memory, and therefore the validation has high memory requirements. Network latency is not taken into account. The results from this section were performed by running the scripts `testTransactions.sh <n-parties> <batch-size>` and `testRetrieves.sh <n-parties> <file-size>`.

1) TRANSACTIONS FROM INSERTS VS UPDATES

Distinctive tests were performed for inserts and updates in 4, 16 and 64 database nodes for 1024, 2048, 4096, 8192, 16384 transactions (inserts and updates). The average results of 5 runs are depicted in Fig 2 and Fig 3. These runs are performed under synchronisation constraints; transactions are performed sequentially waiting for finalisation before initiating the next transaction. This means transactions are defined as a full round trip to n servers.

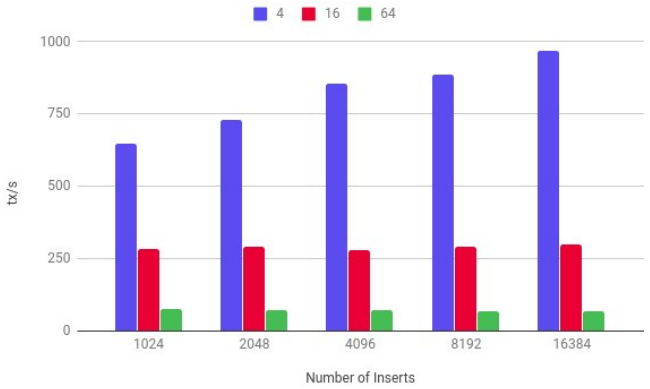

FIGURE 2. Transactions per second for the corresponding number of inserts in 3 distinct configurations: 4, 16, and 64 node instances.

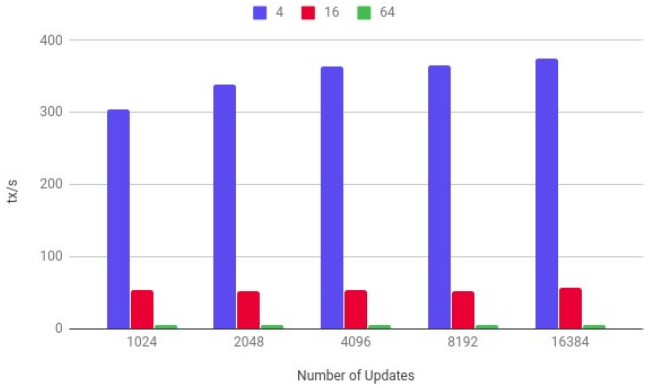

FIGURE 3. Transactions per second for the corresponding number of updates in 3 distinct configurations: 4, 16, and 64 node instances.

The results show a clear distinction in throughput between inserts and updates. The throughput does not significantly change with the number of transactions (as expected), although it is more stable at 16 and 64 nodes, most probably due to the Ed25519 signature verification impact that grows with the number of nodes. In a code profile evaluation, we found that the Ed25519 signature verification has the most significant impact on performance, occupying 36% of the running time for inserts and close to 50% for updates, with an average of 4818 ops/s, reflecting a more accentuated variance in Fig 2. Other Ed25519 implementations such as Tink[9] have reports of 5065 ops/s, and would not be much of an improvement.

2) RETRIEVES FROM SINGLE VS SLICED BLOCKS

Retrieve tests were also performed from single source vs sliced files from 4 and 8 different data sources and for different file sizes, 100, 200, 400, 800 and 1600 MB. A network

[6]Source code: https://github.com/shumy-tools/bft4pnt
[7]Bouncycastle: https://www.bouncycastle.org
[8]Netty.io: https://netty.io
[9]High-performance Ed25519 and X25519 cryptography comes to Nimbus JOSE+JWT: https://connect2id.com/blog/nimbus-jose-jwt-6

speed constraint was imposed at 10MB/s per connection to avoid congestion of the local interface and CPUs. The average results of 5 runs are shown in Table 1. There is an expected difference between sliced and non-sliced (single) block retrieves with a small distance to the theoretical limit. This is what happens in general since there are other overheads to take into account in real implementations (flow control, concurrent disk writes, etc). In our local test, we are also limited in the number of physical cores for all running threads, both for the node parties and the client, meaning we have at least $2 * n$ threads heavily processing data.

TABLE 1. Average throughput of file retrieve at MB/s for 100MB, 200MB, 400MB, 800MB, 1600MB file sizes, using single vs sliced retrieve protocol for 4 and 8 nodes. Avg. displays the average of all columns. The fold column compares the throughput to the single slice retrieve.

	100	200	400	800	1600	Avg.	Fold
Single	9.6	9.5	9.6	9.5	9.5	9.6	1
Sli. 4	40.0	38.3	35.6	35.5	37.4	37.4	3.91
Sli. 8	73.9	77.9	76.8	69.1	72.8	74.1	7.76

B. TENDERMINT COMPARISON

For a simplistic comparison with a blockchain BFT case, a benchmark was set for Tendermint version 0.31.5 on one node using `tm-bench -T 100 -r 100000`, pushing for the maximum allowed transactions in 100s. Tendermint was setup with `tendermint testnet -v <nodes>` for 1, 4 and 8 nodes. The Tendermint setup did not run for a 16 local node configuration. Transactions throughput and latency are shown in Tables 2 and 3 respectively. The BFT-PNT runs were done with the average number of transactions from the `tm-bench` output (using only inserts).

TABLE 2. BFT-PNT vs Tendermint comparison of average transactions throughput per second for 1, 4, and 8 database nodes.

	1	4	8
BFT-PNT	1820	988	548
Tendermint	2630	714	330

TABLE 3. BFT-PNT vs Tendermint comparison of transactions latency in ms for 1, 4 and 8 database nodes.

	1	4	8
BFT-PNT	0.55	1.01	1.82
Tendermint	1961	4545	7692

Existing blockchain techs use mined blocks with batches of transactions, and some benchmarks do not consider the signature overhead. This aggregation artificially raises the transaction throughput at the cost of worse latency. To be fair, our measured performance should be compared with mined blocks throughput; however, this is not always possible to control. BFT-PNT commits are set after receiving $(n - t)$ acknowledges, where Tendermint commits are only finalised after the block is completed. This difference is noted in the discrepancy of the latency results. Nonetheless, BFT-PNT achieves a gain of 1.4 times for 4 nodes and 1.7 times for 8 nodes in a scenario of low write contention.

VII. CONCLUSION

This article proposes a distributed database consensus protocol to improve the performance of EHR insertion operations. The proposed methodology is able to extend the nature of distributed EHR and improve consistency, availability, and data protection. The proposal adds custom semantics to the traditional Byzantine Fault Tolerant consensus protocols to optimise specific use-cases. Moreover, it can maintain consistency checks of individual data slices in a Byzantine setup, meaning that an inconsistent slice from a misbehaving node is promptly detected before the final aggregation and with the extra advantage of improving the performance of retrieves. The evaluation process demonstrates the feasibility of the proposed approach and moderate gains on 4 and 8 nodes in transaction throughput and considerable gains in latency. Furthermore, our data slice scheme can retrieve files close to the theoretical limit in relation to the number of slices.

REFERENCES

[1] S. Kothari, J. H. Phan, T. H. Stokes, and M. D. Wang, "Pathology imaging informatics for quantitative analysis of whole-slide images," *J. Amer. Med. Inform. Assoc.*, vol. 20, no. 6, pp. 1099–1108, 2013.
[2] S. Shilo, H. Rossman, and E. Segal, "Axes of a revolution: Challenges and promises of big data in healthcare," *Nature Med.*, vol. 26, no. 1, pp. 29–38, Jan. 2020.
[3] K. H. Lee, H. J. Lee, J. H. Kim, H. S. Kang, K. W. Lee, H. Hong, H. J. Chin, and K. S. Ha, "Managing the CT data explosion: Initial experiences of archiving volumetric datasets in a mini-PACS," *J. Digit. Imag.*, vol. 18, no. 3, pp. 188–195, Sep. 2005.
[4] M. Meingast, T. Roosta, and S. Sastry, "Security and privacy issues with health care information technology," in *Proc. Int. Conf. IEEE Eng. Med. Biol. Soc.*, Aug. 2006, pp. 5453–5458.
[5] S. Schulz and C. Martınez-Costa, "How ontologies can improve semantic interoperability in health care," in *Process Support and Knowledge Representation in Health Care*. Cham, Switzerland: Springer, 2013, pp. 1–10.
[6] P. Ross, R. Sepper, and H. Pohjonen, "Cross-border teleradiology—Experience from two international teleradiology projects," *Eur. J. Radiol.*, vol. 73, no. 1, pp. 20–25, Jan. 2010.
[7] M. Pedrosa, J. M. Silva, J. F. Silva, S. Matos, and C. Costa, "SCREEN-DR: Collaborative platform for diabetic retinopathy," *Int. J. Med. Informat.*, vol. 120, pp. 137–146, Dec. 2018.
[8] R. Lebre, R. Jesus, P. Nunes, and C. Costa, "Collaborative framework for a whole-slide image viewer," in *Proc. IEEE 32nd Int. Symp. Comput.-Based Med. Syst. (CBMS)*, Jun. 2019, pp. 221–224.
[9] R. Lebre, E. Pinho, J. M. Silva, and C. Costa, "Dicoogle framework for medical imaging teaching and research," in *Proc. IEEE Symp. Comput. Commun. (ISCC)*, Jul. 2020, pp. 1–7.
[10] D. C. Goodman and E. S. Fisher, "Physician workforce crisis? Wrong diagnosis, wrong prescription," *New England J. Med.*, vol. 358, no. 16, pp. 1658–1661, Apr. 2008.
[11] E. Chukwu and L. Garg, "A systematic review of blockchain in healthcare: Frameworks, prototypes, and implementations," *IEEE Access*, vol. 8, pp. 21196–21214, 2020.
[12] L. Lamport, R. Shostak, and M. Pease, "The Byzantine generals problem," *ACM Trans. Program. Lang. Syst.*, vol. 4, no. 3, pp. 382–401, Jul. 1982.
[13] J. R. Almeida, T. M. Godinho, L. B. Silva, C. Costa, and J. L. Oliveira, "Services orchestration and workflow management in distributed medical imaging environments," in *Proc. IEEE 31st Int. Symp. Comput.-Based Med. Syst. (CBMS)*, Jun. 2018, pp. 170–175.

[14] W. Lin, W. Dou, Z. Zhou, and C. Liu, "A cloud-based framework for home-diagnosis service over big medical data," *J. Syst. Softw.*, vol. 102, pp. 192–206, Apr. 2015.

[15] G. Hayes, K. El-Khatib, and C. McGregor, "Supporting health informatics with platform-as-a-service cloud computing," in *Advanced Technologies, Embedded and Multimedia for Human-Centric Computing*. Dordrecht, The Netherlands: Springer, 2014, pp. 1149–1158.

[16] T. M. Godinho, C. Viana-Ferreira, L. A. B. Silva, and C. Costa, "A routing mechanism for cloud outsourcing of medical imaging repositories," *IEEE J. Biomed. Health Informat.*, vol. 20, no. 1, pp. 367–375, Jan. 2016.

[17] L. A. B. Silva, C. Costa, and J. L. Oliveira, "A PACS archive architecture supported on cloud services," *Int. J. Comput. Assist. Radiol. Surg.*, vol. 7, no. 3, pp. 349–358, May 2012.

[18] C. T. Yang, J. C. Liu, and S. T. Chen, "Implementation of a big data accessing and processing platform for medical records in cloud," *J. Med. Syst.*, vol. 41, no. 10, p. 149, 2017.

[19] L. Zhou, A. Fu, S. Yu, M. Su, and B. Kuang, "Data integrity verification of the outsourced big data in the cloud environment: A survey," *J. Netw. Comput. Appl.*, vol. 122, pp. 1–15, Nov. 2018.

[20] S. Subashini and V. Kavitha, "A survey on security issues in service delivery models of cloud computing," *J. Netw. Comput. Appl.*, vol. 34, no. 1, pp. 1–11, 2011.

[21] H. Bala, V. Venkatesh, S. Venkatraman, and J. Bates, "If the worst happens: Five strategies for developing and leveraging information technology-enabled disaster response in healthcare," *IEEE J. Biomed. Health Informat.*, vol. 20, no. 6, pp. 1545–1551, Nov. 2016.

[22] A. Shahnaz, U. Qamar, and A. Khalid, "Using blockchain for electronic health records," *IEEE Access*, vol. 7, pp. 147782–147795, 2019.

[23] S. Tanwar, K. Parekh, and R. Evans, "Blockchain-based electronic healthcare record system for healthcare 4.0 applications," *J. Inf. Secur. Appl.*, vol. 50, Feb. 2020, Art. no. 102407.

[24] Y. Zhuang, L. R. Sheets, Y.-W. Chen, Z.-Y. Shae, J. J. P. Tsai, and C.-R. Shyu, "A patient-centric health information exchange framework using blockchain technology," *IEEE J. Biomed. Health Informat.*, vol. 24, no. 8, pp. 2169–2176, Aug. 2020.

[25] A. Pashazadeh and N. J. Navimipour, "Big data handling mechanisms in the healthcare applications: A comprehensive and systematic literature review," *J. Biomed. Informat.*, vol. 82, pp. 47–62, Jun. 2018.

[26] C. Fang, Y. Guo, N. Wang, and A. Ju, "Highly efficient federated learning with strong privacy preservation in cloud computing," *Comput. Secur.*, vol. 96, Sep. 2020, Art. no. 101889.

[27] Y. Zheng, R. Lu, and J. Shao, "Achieving efficient and privacy-preserving k-NN query for outsourced eHealthcare data," *J. Med. Syst.*, vol. 43, no. 5, p. 123, May 2019.

[28] D. Suresh and M. L. Florence, "Securing personal health record system in cloud using user usage based encryption," *J. Med. Syst.*, vol. 43, no. 6, p. 171, Jun. 2019.

[29] L. S. Ribeiro, C. Viana-Ferreira, J. L. Oliveira, and C. Costa, "XDS-I outsourcing proxy: Ensuring confidentiality while preserving interoperability," *IEEE J. Biomed. Health Informat.*, vol. 18, no. 4, pp. 1404–1412, Jul. 2014.

[30] J. M. Silva, A. Guerra, J. F. Silva, E. Pinho, and C. Costa, "Face de-identification service for neuroimaging volumes," in *Proc. IEEE 31st Int. Symp. Comput.-Based Med. Syst. (CBMS)*, Jun. 2018, pp. 141–145.

[31] B. Riedl, T. Neubauer, G. Goluch, O. Boehm, G. Reinauer, and A. Krumboeck, "A secure architecture for the pseudonymization of medical data," in *Proc. 2nd Int. Conf. Availability, Rel. Secur. (ARES)*, Apr. 2007, pp. 318–324.

[32] M. R. Patra, R. K. Das, and R. P. Padhy, "CRHIS: Cloud based rural healthcare information system," in *Proc. 6th Int. Conf. Theory Pract. Electron. Governance*, Oct. 2012, pp. 402–405.

[33] C. O. Rolim, F. L. Koch, C. B. Westphall, J. Werner, A. Fracalossi, and G. S. Salvador, "A cloud computing solution for patient's data collection in health care institutions," in *Proc. 2nd Int. Conf. eHealth, Telemed., Social Med.*, Feb. 2010, pp. 95–99.

[34] M. Mettler, "Blockchain technology in healthcare: The revolution starts here," in *Proc. IEEE 18th Int. Conf. e-Health Netw., Appl. Services (Healthcom)*, Sep. 2016, pp. 1–3.

[35] X. Yue, H. Wang, D. Jin, M. Li, and W. Jiang, "Healthcare data gateways: Found healthcare intelligence on blockchain with novel privacy risk control," *J. Med. Syst.*, vol. 40, no. 10, p. 218, 2016.

[36] M. Misbhauddin, A. Alabdulatheam, M. Aloufi, H. Al-Hajji, and A. AlGhuwainem, "MedAccess: A scalable architecture for blockchain-based health record management," in *Proc. 2nd Int. Conf. Comput. Inf. Sci. (ICCIS)*, Oct. 2020, pp. 1–5.

[37] D. Ivan, "Moving toward a blockchain-based method for the secure storage of patient records," in *Proc. ONC/NIST Use Blockchain Healthcare Res. Workshop*. Gaithersburg, MD, USA: ONC/NIST, 2016, pp. 1–11.

[38] G. Zyskind, O. Nathan, and A. Pentland, "Decentralizing privacy: Using blockchain to protect personal data," in *Proc. IEEE Secur. Privacy Workshops*, May 2015, pp. 180–184.

[39] T. McGhin, K.-K.-R. Choo, C. Z. Liu, and D. He, "Blockchain in healthcare applications: Research challenges and opportunities," *J. Netw. Comput. Appl.*, vol. 135, pp. 62–75, Jun. 2019.

[40] S. Khezr, A. Yassine, and R. Benlamri, "Blockchain technology in healthcare: A comprehensive review and directions for future research," *Appl. Sci.*, vol. 9, no. 9, p. 1736, 2019.

[41] H. Kaur, M. A. Alam, R. Jameel, A. K. Mourya, and V. Chang, "A proposed solution and future direction for blockchain-based heterogeneous medicare data in cloud environment," *J. Med. Syst.*, vol. 42, p. 156, Aug. 2018.

[42] H. Tian, J. He, and Y. Ding, "Medical data management on blockchain with privacy," *J. Med. Syst.*, vol. 43, no. 2, p. 26, Feb. 2019.

[43] M. Castro and B. Liskov, "Practical Byzantine fault tolerance," in *Proc. OSDI*, vol. 99, 1999, pp. 173–186.

[44] L. Lamport, "Paxos made simple," *ACM SIGACT News*, vol. 32, no. 4, pp. 18–25, 2001.

[45] D. Ongaro and J. K. Ousterhout, "In search of an understandable consensus algorithm," in *USENIX Annu. Tech. Conf.*, 2014, pp. 305–319.

[46] B. Temkow, A.-M. Bosneag, X. Li, and M. Brockmeyer, "PaxonDHT: Achieving consensus in distributed hash tables," in *Proc. Int. Symp. Appl. Internet (SAINT)*, Jan. 2006, p. 9.

[47] T. D. Chandra, R. Griesemer, and J. Redstone, "Paxos made live: An engineering perspective," in *Proc. 26th Annu. ACM Symp. Princ. Distrib. Comput. (PODC)*, 2007, pp. 398–407.

[48] D. Kim, I. Doh, and K. Chae, "Improved raft algorithm exploiting federated learning for private blockchain performance enhancement," in *Proc. Int. Conf. Inf. Netw. (ICOIN)*, Jan. 2021, pp. 828–832.

[49] L. Baird, "The Swirlds hashgraph consensus algorithm: Fair, fast, Byzantine fault tolerance," Swirlds, College Station, TX, USA, Tech. Rep. SWIRLDS-TR-2016-01, 2016.

[50] H. Huang, W. Kong, S. Zhou, Z. Zheng, and S. Guo, "A survey of state-of-the-art on blockchains: Theories, modelings, and tools," *ACM Comput. Surg.*, vol. 54, no. 2, pp. 1–42, Apr. 2021.

[51] P. Chevalier, B. Kaminski, F. Hutchison, Q. Ma, S. Sharma, A. Fackler, and W. J. Buchanan, "Protocol for asynchronous, reliable, secure and efficient consensus (PARSEC) version 2.0," *CoRR*, vol. abs/1907.11445, 2019. [Online]. Available: http://arxiv.org/abs/1907.11445

[52] S. Zhang and J.-H. Lee, "Analysis of the main consensus protocols of blockchain," *ICT Exp.*, vol. 6, no. 2, pp. 93–97, Jun. 2020.

[53] H. Attiya and J. L. Welch, "Sequential consistency versus linearizability," *ACM Trans. Comput. Syst.*, vol. 12, no. 2, pp. 91–122, May 1994.

[54] J. Lennerz, D. A. Clunie, A. Fedorov, S. W. Doyle, S. Pieper, V. Klepeis, L. P. Le, G. L. Mutter, D. S. Milstone, T. J. Schultz, and R. Kikinis, "Implementing the DICOM standard for digital pathology," *J. Pathol. Informat.*, vol. 9, no. 1, p. 37, 2018.

[55] M. E. Gale and D. R. Gale, "DICOM modality worklist: An essential component in a PACS environment," *J. Digit. Imag.*, vol. 13, no. 3, pp. 101–108, Aug. 2000.

[56] T. M. Godinho, R. Lebre, L. B. Silva, and C. Costa, "An efficient architecture to support digital pathology in standard medical imaging repositories," *J. Biomed. Informat.*, vol. 71, pp. 190–197, Jul. 2017.

[57] S. Wan, M. Li, G. Liu, and C. Wang, "Recent advances in consensus protocols for blockchain: A survey," *Wireless Netw.*, vol. 26, no. 8, pp. 5579–5593, Nov. 2020.

[58] A. Singh, P. Maniatis, P. Druschel, and T. Roscoe, "Conflict-free quorum-based BFT protocols," Max Planck Inst. Softw. Syst., Saarbrücken, Germany, Tech. Rep. 2007-1, 2007.

[59] M. J. Fischer, N. A. Lynch, and M. S. Paterson, "Impossibility of distributed consensus with one faulty process," *J. ACM*, vol. 32, no. 2, pp. 374–382, 1985.

[60] Y.-T. Lin. (2017). *Istanbul Byzantine Fault Tolerance*. [Online]. Available: https://github.com/ethereum/EIPs/issues/650

MICAEL PEDROSA received the master's degree in computer engineering from the University of Aveiro, Portugal. He is currently pursuing the Ph.D. degree with the University of A Coruña, Spain.

He is currently a Researcher at IEETA, Portugal. His research interests include data privacy, healthcare, medical information systems, authorization policies, data protection, public-key cryptography, and security of data in medical contexts. He was a recipient of the Best Student Paper Award at the 19th International Conference on BioInformatics and BioEngineering, given by the BIBE2019 Steering and Awards Committee.

RUI LEBRE received the master's degree in computer engineering from the University of Aveiro, Portugal, in 2017. He is currently pursuing the Ph.D. degree in information technology with the University of A Coruña, Spain.

Later that year, he turned himself as a Researcher at IEETA, Portugal. During the past few years, his research topics include medical imaging, medical informatics, and information technologies. Simultaneously, he is the Co-Maintainer of Dicoogle Open-Source PACS. His research interests include the decentralization of the storage and index of healthcare solutions and providing high-fidelity architectures in the healthcare field.

CARLOS COSTA received the Ph.D. degree in medical informatics area. He is currently an Associate Professor with the Department of Electronics, Telecommunications and Informatics, University of Aveiro. He is also a Researcher with the Institute of Electronics and Telematics Engineering of Aveiro (IEETA) and a Partner of BMD Software Company. He is the author or coauthor of more than 150 publications in this area. His research interests include PACS-DICOM (medical imaging systems and networks), telemedicine, healthcare information systems, and other areas of research, such as machine learning, databases, security, and access control.