This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826078.

Privacy preserving federated machine learning and blockchaining for reduced cyber risks in a world of distributed healthcare

D6.1
"Local blockchain mechanism"

WP6
"Blockchains and user right management"

Disclaimer

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 826078. Any dissemination of results reflects only the author's view and the European Commission is not responsible for any use that may be made of the information it contains.

Copyright message

Document information

Grant Agreement Number: 826078			**Acronym: FeatureCloud**	
Full title	Privacy preserving federated machine learning and blockchaining for reduced cyber risks in a world of distributed healthcare			
Topic	Toolkit for assessing and reducing cyber risks in hospitals and care centres to protect privacy/data/infrastructures			
Funding scheme	RIA - Research and Innovation action			
Start Date	1 January 2019	**Duration**	60 months	
Project URL	https://featurecloud.eu/			
EU Project Officer	Reza RAZAVI (CNECT/H/03)			
Project Coordinator	Jan BAUMBACH, TECHNISCHE UNIVERSITAET MUENCHEN (TUM)			
Deliverable	D6.1 “Local blockchain mechanism”			
Work Package	WP6 “Blockchains and user right management”			
Date of Delivery	**Contractual**	31/12/20	**Actual**	31/12/20
Nature	REPORT	**Dissemination Level**	PUBLIC	
Lead Beneficiary	05 SBA			
Responsible Author(s)	Walid Fdhila, Aljosha Judmayer, Nicholas Stiffter, Philipp Schindler, Andreas Kern (SBA)			
Keywords	Federated machine learning, blockchain, timestamping			

Table of Content

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 826078.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826078.

1 Objectives of the deliverables based on the Description of Action (DoA)

The associated objective of this deliverable is Objective 1, which aims at developing a **blockchain-mechanism to track the usage of data sources**.

"Most important for the global success of a machine learning platform requiring user consent is the ability of users and data owners to control the data introduced, while allowing data discovery in a privacy-preserving manner. This is especially important in order to integrate as many federated machine learning nodes as possible, while being aware of privacy rights and regulations, especially the General Data Protection Regulation (GDPR) and regulations for medical data. In order to reach these goals, we will conduct research into blockchain-based technologies, especially so-called Byzantine-Fault-Tolerance (BFT) blockchains, in order to provide user rights management, consent and data discovery mechanisms."

Objective 1 of this work package aims "to provide a local blockchain-mechanism on the individual data holder side in order to track what source data has been used for which analytical instance"

The corresponding task in this work package is *Task 1: Work out a local blockchain mechanism:*
"In this task, the main focus lies on developing a blockchain mechanism that caters for the specific needs of the FeatureCloud platform. For this, TUM and MUG will research on the exact specific requirements of information processing in different environments (hospital only, shared) and model them (with SBA) accordingly. SBA and RI will select appropriate BFT blockchain mechanisms and modify them in order to make them suitable and performant. This task will thereby also focus on the issue of modifying consent mechanisms. Based on this academic research SBA will construct a prototype that can be verified and tested with actual information. Required changes will thereby be fed back into the construction phase."

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826078.

2 Executive Summary

This deliverable (D6.1) focuses on capturing the requirements for auditing federated machine learning executions, inputs, and outputs necessary to decide on the appropriate blockchain technology and mechanisms. In particular, it takes as a basis the global FeatureCloud architecture and processes, and elaborates on mechanisms and technologies that can help facilitate the auditing tasks. Independently of specific data formats, or consent representation models (to be handled specifically in task 6.2, as described in the DoA), this first deliverable of WP6 focuses on (i) identifying stakeholders of the FeatureCloud project that are particularly involved in WP6, (ii) analyzing the requirements for the audit process, and (iii) conducting a feasibility study to evaluate whether or not blockchain technology in conjunction with other cryptographic techniques can help facilitate the audit tasks and reduce the risks related to malicious actors or behaviors. The deliverable also presents some fundamentals and use cases of blockchain technology, the most utilized consensus mechanisms, as well as useful cryptographic primitives. Furthermore, it sketches the advantages of using blockchain technology as a distributed and immutable audit-storage layer, and highlights designs, system characteristics, and assumptions that are indicative of unproven or risky approaches for integrating blockchain technologies. In this deliverable, it will be shown that FeatureCloud can notably benefit from blockchain technology for the audit purposes, but vetting the selection and implementation of candidate blockchain mechanisms for FeatureCloud will be done in future deliverables as it highly depends on other tasks and WPs, including additional requirements for consent management, data representation, and other architectural aspects.

3 Introduction (Challenge)

The possibly most pressing problem in training powerful artificial intelligence (AI) and machine learning in health informatics is that all data that is usually distributed over various hospitals needs to be accessible centrally for the training phase, e.g. in a central cloud. In the light of enormous data leaks in the recent past, the public opinion, as well as the patient trust, demand secure ways of storing and processing their data. IT solution providers are challenged to find ways of providing patients and hospitals full control over how their sensitive patient data is processed, including mechanisms to effectively revoke an earlier given consent by a patient. The primary objective of FeatureCloud is a novel kind of medical data mining technology that - unlike existing approaches - can avoid insecure client-cloud and inter-cloud communication, and that can ensure that all data remains within (legally and technically) safe harbours: the hospitals’ IT infrastructure.

Tracking consent and providing manipulation security is of vital importance for any data analysis platform dealing with sensitive personal information. This work package focuses on the application of blockchain technologies in order to provide the FeatureCloud platform with means for facilitating audits and consent tracking. This is not only important with respect to user privacy and protecting patient rights, but also allows the replication of results.

Blockchains, and related hash-chain based technologies, offer a multitude of interesting possibilities for medical sciences. Still, the field of blockchain technologies is actually quite broad and must not be reduced to Nakamoto-like (Bitcoin-like) techniques. With all the different basic technologies and designs for blockchains, a thorough requirements analysis is of vital importance. It is important to collect these requirements very carefully, as they directly shape the whole technological foundation of our blockchain approach. Based on this analysis, in WP6, we will select a suitable base technology and build a local blockchain mechanism for tracking data manipulation and managing consents.

3.1. Current Situation: “Conventional Studies”

Conducting a medical study currently usually involves the following steps:

- Specifying the exact research question
- Agreement between collaborators
- Collecting data and obtaining patient consent
- Performing analysis

We will briefly describe these steps below.

Specifying the research question. Specifying the research question seems trivial, but is the most important thing that needs to be done at the very beginning of the investigation (Krishnankutty et al., 2012). It affects all following steps, especially what data to collect and with whom, if patients are willing to consent to the usage of their data and obviously the final analysis.

Agreement between collaborators. All collaborators of a study need to agree on important aspects of the study and how to conduct it together. This mainly involves clear mandates and a common vision, strategic coordination and communication mechanisms, and formal organizational leaders (Valaitis et al., 2018). In terms of data collection, it is important to ensure a common way of collecting data to ensure homogeneous data that can later be used together. Often a common data management software is used, which allows for specifying the type of data required (Krishnankutty et al., 2012).

Requirements for this type of software, and how this relates to FeatureCloud, has been reported in Deliverable 8.1 " Report on performance benchmarks”, Section 8, Table 2.

Collecting data and obtaining consent. Managing patient consent and collecting data is often done together by study directors and patients in paper form (Krishnankutty et al., 2012). The consent document signed by study participants needs to contain purpose, benefits, risks and other study information necessary to allow the participants to make an informed decision (Nijhawan et al., 2013).

Performing analysis. After the data has been collected, reviewed and validated, it can be used for the action analysis of the study to answer the previously specified research question.

3.1.1 Differences in FeatureCloud

Conventional studies often involve collecting the data in a tailored manner, obtaining exactly the data needed for the study. In FeatureCloud, the assumption is that data is already present - either because it has been collected for personal treatments of patients, or for previously conducted studies - and can be used to answer a different research question. This means that patients don’t need to actively provide new data to a study. However, they still need to give consent for their data to be used (anew). It also means that the data can be heterogeneous between the collaborators. In conventional studies it is ensured that data collection is done in a common, identical fashion previously agreed upon. If the data has been collected before however, it needs to be preprocessed in such a way that it can be used together. This applies to the parts of the data that can be used (features selection) and transformations applied to the values themselves (feature preprocessing, e.g. normalization).

3.2 Document Structure

The remainder of this document is structured as follows. Section 4 presents blockchain fundamentals and concepts necessary for the understanding of the document. In particular, it introduces cryptographic primitives and multiple existing consensus mechanisms. While Section 5 captures and analyzes the FeatureCloud requirements for WP6, Section 6 conducts a feasibility study and Section 7 states open questions.

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 826078.

4 Blockchain Fundamentals

4.1. Overview

Cryptocurrencies have developed from a niche topic discussed by cryptography enthusiasts and hobbyists into a multi-billion dollar ecosphere in the last decade. This transformation was primarily fueled by Bitcoin and its underlying blockchain technology, which promises to replace the need for trusted third parties by facilitating a peer-to-peer based digital ledger of transactions. Interestingly, blockchain technologies do not present fundamentally new or different concepts, but rather form a new and unique symbiosis of well known and studied technological building blocks such as hash functions, Merkle trees, digital signatures, and hash-based proof-of-work.
In the context of the FeatureCloud project and work package 6, blockchain and distributed ledger technologies offer compelling characteristics and properties that can be leveraged, e.g., to develop systems that facilitate trustworthy and secure audit trails. In this section, we outline the background, different protocol designs and their components, current security research and use cases that serve to highlight both advantages as well as potential drawbacks, of these technologies. Further, we discuss different design approaches and application scenarios and show that there currently exists no single protocol design that is suitable for all use-cases.

4.2. Background

The recent hype surrounding blockchains and distributed ledgers has put pressure on a wide range of industries and businesses to integrate or otherwise leverage these new technologies, as they seemingly offer advantages over traditional system architectures. However, currently a marked and worrying gap regarding a profound understanding of the actual capabilities and limitations of concrete protocol implementations of such distributed ledger technologies (DLT), and what these systems are advertised or believed to achieve, exists. Therefore, adopters of DLT face the realistic threat of relying on technologies that are incapable of meeting the expected demands, opening up a wide range of security risks and the potential for large financial loss. Nowadays, blockchain is all too often encountered as a marketing buzzword or fuzzy umbrella term whose intended meaning is best translated to ”technologies that are loosely related to Bitcoin”. Bitcoin is a proposal and subsequent implementation of a "peer-to-peer electronic cash system“, whose novel approach promises to solve the distributed double spending problem (Hoepman, 2007; Jarecki and Odlyzko, 1997) without having to rely on a trusted third party. However, beyond the hype, the underlying concepts and technologies have managed to spark the interest of the scientific community and led to a plethora of research efforts and publications from various different disciplines, such as cryptography, distributed and fault tolerant computing, economics and legal sciences. This highlights the interdisciplinary nature of cryptocurrencies and blockchain technologies, as a new area of research is beginning to take shape. Interestingly, the term blockchain itself was not directly introduced by the pseudonymous author or authors going by the name Satoshi Nakamoto in the original Bitcoin white paper (Nakamoto, 2008), instead only the words blocks and chains are mentioned. As part of Bitcoin’s underlying data structure, transactions are grouped into blocks which are linked or chained together using hash pointers (Narayanan, Arvind and Bonneau, Joseph and Felten, Edward and Miller,

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 826078.

Andrew and Goldfeder, Steven, 2016). The combination of these words was subsequently used early on within the Bitcoin community when referring to certain concepts of this so-called cryptographic currency (or simply “cryptocurrency”).
Generally speaking, blockchain or blockchain technologies may be used to refer to the mechanisms and principles by which Bitcoin and similar systems are able to achieve some form of decentralized agreement upon a shared ledger. On the other hand, blockchain may also specifically refer to the underlying data structure of such systems. Currently, there is no broad agreement on the exact meaning of the term, and definitions are evolving as research in this field is ongoing.

4.3. Blockchain Protocol Designs

Blockchains and distributed ledger technologies (DLT) have recently received significant interest by academia and industry alike. At the center of attention is the promise that DLT can enable the decentralization of interactions between mutually distrusting parties without having to rely on a trusted third party. The outlook that using DLT could both improve security and require less trust has led many decision makers to consider adopting these technologies in a variety of different fields and use-cases that reach beyond payment mechanisms. In this regard it is often falsely assumed that adopting the protocol design of a successful cryptocurrency, such as Bitcoin, to a different use-case can yield similar guarantees. Interestingly, the purported guarantees of Bitcoin are still an ongoing topic of debate in academia and not agreed upon. In addition, there exists a vast and quickly growing number of novel and improved DLT protocol designs that promise increased performance, scalability, privacy, and sustainability. Currently, this diverse landscape of different DLT protocol (DLP) designs and implementations is opaque, complex and not sufficiently explored. Furthermore, rigorous scientific analyses regarding protocol characteristics and security guarantees are exceedingly rare. This leaves implementers and adopters of DLT in the difficult position of having to choose a particular protocol design and technology stack without sufficient information for an informed decision. More worryingly, it has been shown that many available DLT protocols have design flaws or vulnerabilities that can lead to catastrophic failures if basic system assumptions are violated, yet these assumptions may not always hold in practical settings.

While research on the Bitcoin protocol itself is relatively well established and widely published, the analysis of alternative protocol designs for cryptocurrencies and DLT is a lot more scattered and often opaque to persons not intimately familiar with the research field. Works seeking to perform a systematization and categorization of different protocol designs currently only either offer a narrowed down view of the available landscape (Bano et al., 2017), or are largely an enumeration without further classification (Xu et al., 2017), as there exist literally hundreds, if not thousands of projects that may be noteworthy of attention. Further, it is often DLT protocols whose design is strongly rooted in academic research that are categorized and analyzed in such systematization efforts, as they usually provide concise specifications and design descriptions and are peer-reviewed, thereby meeting a certain quality standard that does not demand further in-depth analysis to extract relevant information for an objective evaluation and comparison.

In practice, the choice for a particular DLT may be governed by various other deciding factors such as software availability, the utilization of particular technologies, or personal preference based on

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 826078.

past experiences. Cachin and Vukolić (Cachin and Vukolić, 2017a) observe and compare consensus protocols employed in prominent *permissioned* DLT platforms, i.e., within an environment where write access to the ledger is restricted. It is noteworthy that several of these protocols have not received an in-depth analysis before, and the authors had to result to estimates or were simply unable to assess their actual properties due to a lack of information.

4.3.1. Cryptographic Primitives in Blockchain and DLT

Cryptography plays a primordial role in achieving privacy, integrity and authenticity for digital artifacts and the processes that create and modify them. In particular cryptographic hash functions, symmetric and asymmetric encryption, and digital signature schemes are fundamental primitives that are not only crucial for realizing digital audit trails, but also form key building blocks in the majority of modern digital infrastructures, including blockchain and distributed ledger technologies. While integrity implies that data has not been altered or tampered with, authenticity signifies that the data originates from the rightful source. Non-repudiation means that a user that digitally signs the data cannot deny its existence. We hereby briefly outline these cryptographic primitives in more detail.

4.3.1.1. Cryptographic Hash functions

A hash function (H) is a mathematical deterministic function that takes a message (m) of arbitrary but finite size and outputs a fixed size hash (a.k.a. digest). In order to qualify as a *cryptographic* hash function, a hash function has to fulfill the following properties:

- Easy to compute: given a finite message m, it is computationally easy to compute its corresponding hash h = H(m).
- Pre-image resistance: given a hash value h, it is infeasible to compute in polynomial time the message m such that h = H(m).
- Second pre-image resistance: given a message m, it is infeasible to find another message m’ ≠ m such that H(m) = H(m’).
- Collision resistance: it should be infeasible to find any two messages m and m’ such that m’ ≠ m and H(m) = H(m’).

Cryptographic hash functions are useful for checking whether or not data has been modified in any way, by (1) storing the hash of the data in some tamper proof manner, (2) re-computing the hash function with the data in question and then (3) comparing whether the computed digest matches the stored hash value. The properties of cryptographic hash functions allow to uniquely identify data based on the computed hash value with high probability. Hereby, this probability relates to the specific security parameters of the hash function, in particular its collision resistance and second pre-image resistance. Furthermore, the property of pre-image resistance allows cryptographic commitments where a party commits to a secret message by only revealing its hash. At the time the secret is revealed anyone can compute the hash function with the secret as input and verify if the computed hash matches the original commitment. To ensure that pre-image resistance is achieved, the input message must be of sufficient entropy, i.e., the input space of possible messages must be large enough to render any guessing attacks infeasible. In practice this means that data with low

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 826078.

entropy, e.g. dates of birth, must be amended, usually by concatenating to the message some random secret, to prevent an adversary from being able to guess the pre-image through brute force.

4.3.1.2. Merkle Trees

A Merkle tree is a data structure in the form of a binary tree, in which leaf nodes are labeled with the values that need to be authenticated (e.g., the transactional data in the case of blockchain), and non-leaf nodes are labeled with the hashes of the child node labels (cf. Fig. 1). Merkle trees enable fast, secure and efficient verification of data.

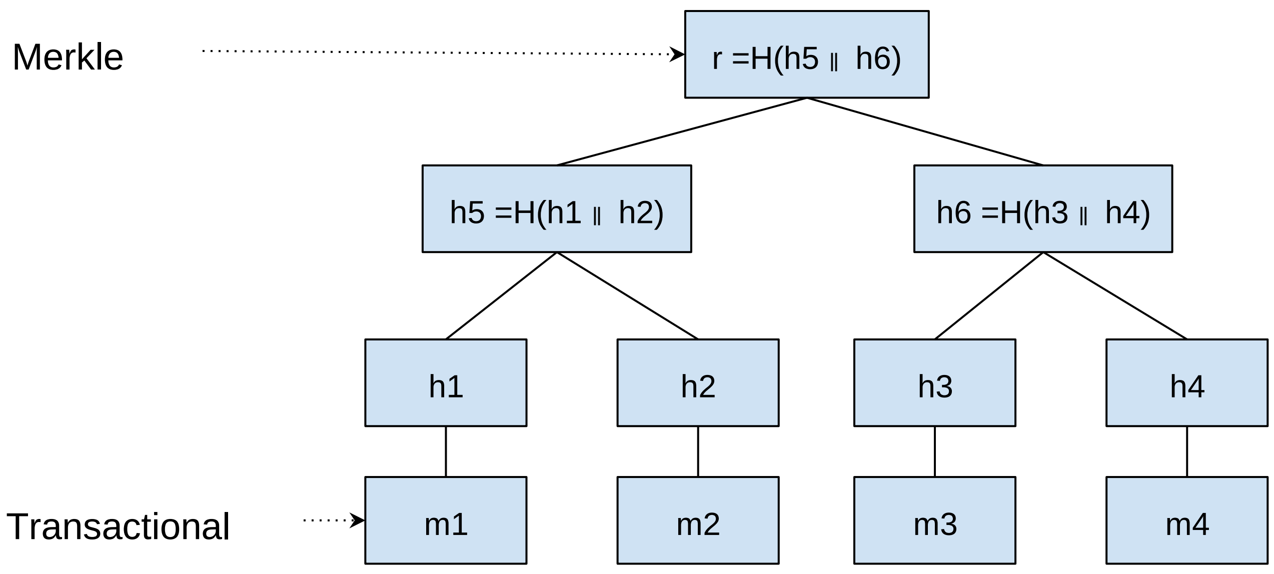


Figure 1. Merkle Tree Example

In the example above, to prove that m4 is indeed a part of the Merkle tree with root r, the hashes h3, and h5 and the message m4 are required. By computing the hash of m4 one obtains h4, by concatenating h3 and h4 and evaluating the hash function H on it, h6 is revealed. Analogously, the hash of concatenating h5 and h6 produces the root node r (cf. Fig 2). Thereby one can provide cryptographic proof that m4 is a leaf of the Merkle tree with root r. In general, it requires approximately log2(n) hash computations to prove that a message m is a leaf of a given Merkle tree.

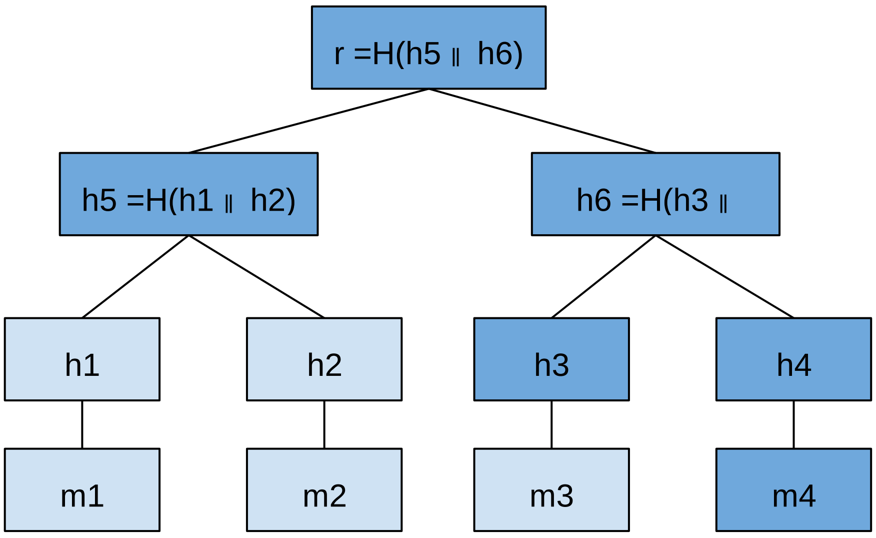


Figure 2. Proof of Inclusion Example

4.3.1.3. Asymmetric Cryptography

Asymmetric cryptography plays a key role to facilitate auditability and verifiability of data, and can also help improve privacy through encryption. It employs a combination of a public and private key generated by a cryptographic algorithm, as opposed to a shared secret key that is used in symmetric cryptography. Both keys are mathematically correlated in a way that only the private key decrypts what a corresponding public key has encrypted and vice versa. The cryptographic techniques used to implement public key encryption and digital signature schemes are generally closely related to each other and the method of achieving one scheme can be leveraged to also implement the other. The advantage of asymmetric cryptography is twofold. First, one does not have to trust that other parties keep any shared secret key secure. Second, it allows to readily and non-interactively share the necessary information, i.e. the public key, for encrypted communication, or for verification that a digital signature was created from the private key corresponding to a particular public key. Disadvantages are the higher computational complexity over symmetric encryption, and the necessity for a *public key infrastructure* (PKI), if public keys need to be reliably and securely associated with real-world identities. Establishing trustworthy and secure PKI is an ongoing research topic where the development of blockchain and distributed ledger technologies have opened up new and interesting possibilities for creating more secure and robust infrastructure.

4.3.1.4. Public Key Encryption

A *public-key encryption* scheme is defined as a triple of efficient algorithms **E** = (G,E,D) where,

- G is a key generation algorithm (pk,sk) ← G() that takes no input and outputs a key pair *(pk, sk)*, where *pk* is called public key, which can be shared publicly, and *sk* is called secret key, which should be kept private.

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 826078.

- E is a encryption algorithm $c \leftarrow E(pk,m)$ that takes as input a public-key *pk* as well as a message $m \in \mathbf{M}$ and outputs a ciphertext $c \in \mathbf{C}$ encrypted under the public-key *pk* associated with the public/secret key pair *(pk, sk)* of the intended recipient.
- D is a (deterministic) decryption algorithm $d \leftarrow D(sk,c)$ that takes as input a secret-key *sk* as well as a ciphertext $c \in \mathbf{C}$ and outputs the message $m \in \mathbf{M}$, that was encrypted under the public-key *pk* associated with sk , or $\perp$ if the wrong keys have been used.

It follows that if the respective operations are reversible $\forall(pk,sk)$ of G it holds that:

$\forall\ m \in \mathbf{M} : D(sk, E(pk,m)) = m$

4.3.1.5. Digital Signatures

A *digital signature* is a mathematical schema primarily used to authenticate a message or data by the signer identity and to verify its integrity. Digital signature schemes are generally defined as a triple of efficient algorithms
$\mathbf{S} = (G,S,V)$ where,

- G is a key generation algorithm $(pk,sk) \leftarrow G()$ that takes no input and outputs a key pair (pk,sk), where pk is called public key, which can be shared publicly, and *sk* is called secret key, which should be kept private.
- S is a signing algorithm that takes as input a secret key *sk* as well as a message $m \in \mathbf{M}$ and outputs a signature $\sigma \in \Sigma$ that can be communicated publicly together with the message. S is invoked as: $S : \sigma \leftarrow E(sk,m)$.
- V is a (deterministic) algorithm $\{\text{ accept, reject}\} \leftarrow V(pk,m,\sigma)$ that takes as input a public-key *pk* a message $m \in \mathbf{M}$ as well as a signature $\sigma \in \Sigma$ and outputs either accept or reject depending on the validity of the signature on message *m* .

It follows that a signature generated by S is accepted by V if and only if (pk,sk) is a valid public/secret key pair. So $\forall(pk,sk)$ of G it holds that $\forall m \in \mathbf{M} : V(pk,m,S(sk,m)) = \text{accept}$.
Public key encryption and digital signatures are core concepts utilized in blockchain technology.

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 826078.

4.3.2. Mining

Mining is the mechanism by which blockchain transactions are selected and gathered in block candidates, i.e., proposals for consensus, which if confirmed valid, will be added to the immutable chain of blocks. Also referred to as nodes, miners compete with each other to discover the next block of transactions and get rewarded in the form of cryptocurrency units or tokens. A consensus mechanism specific to each blockchain defines the rules by which miners agree on whether or not a block is valid. There exists a multitude of consensus protocols used within blockchain, which differ in terms of security, scalability and efficiency, for example proof of work, proof of stake, proof of authority and proof of space (Badertscher et al., 2018),(Back, 2002), (Bonneau et al., 2015). Mining often relies on game theory techniques to ensure honest nodes will preserve the blockchain integrity. Encryption, hashing, digital signatures and mining, combined with other techniques like transaction verification using Merkle trees are fundamental concepts at the core of blockchain which guarantee and maintain the security of the system. Figure 3 illustrates a blockchain structure where each block points to the hash of the previous block header. A block header contains several attributes among which the Merkle root.

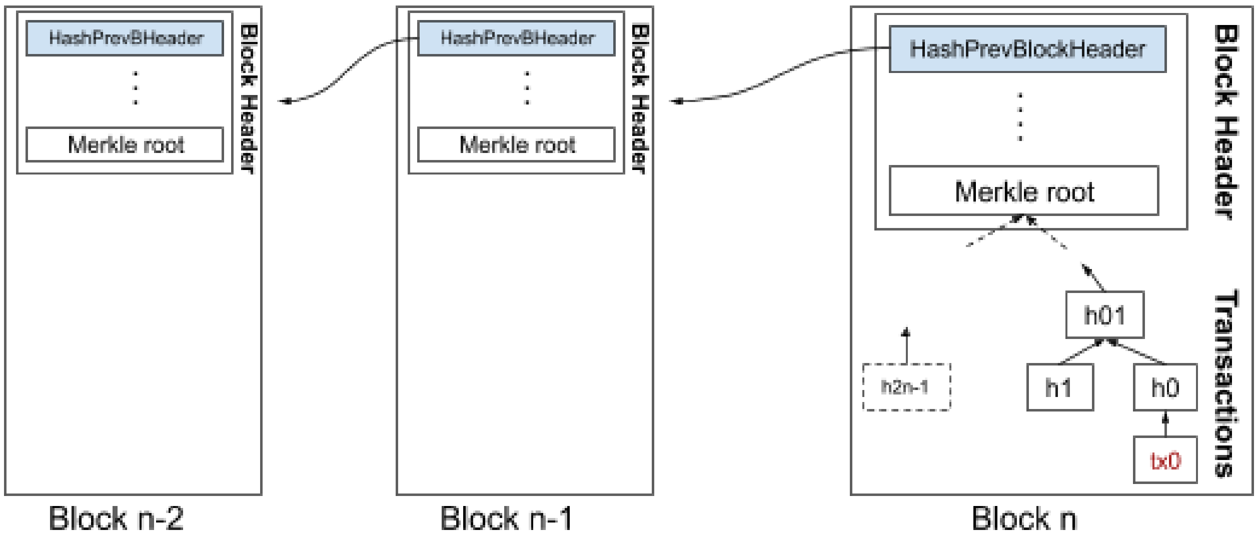


Figure 3. Blockchain Blocks

4.3.3. Consensus Mechanisms

In the following, we present the main consensus mechanisms employed for mining and discuss their security properties that need to be considered when selecting a candidate blockchain mechanism for the FeatureCloud project.

4.3.3.1. Byzantine Fault Tolerance

The origin of the term Byzantine failure traces back to the seminal work of Lamport et al. that introduces and addresses an agreement problem called the Byzantine generals problem (Lamport et al., 1982). In prior work, the same set of authors had first identified that ensuring consistency in

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 826078.

the presence of arbitrary failures within distributed systems is more difficult than one would intuitively expect (Pease et al., 1980). Generally speaking, the terms Byzantine and arbitrary failure are used interchangeably, even though the former more explicitly considers the possibility of adversarial behavior. If a system is allowed to exhibit arbitrary failures, it follows that there can also exist execution traces where the sequence and type of failures is indiscernible to that of any adversarial strategy. A clear distinction between the two failure models is usually not made.
Initial research on the Byzantine generals problem, or more generally how to reach agreement, i.e consensus, among a set of processes in the presence of faults, was spawned from practical engineering challenges at the time. Improvements in both microprocessors as well as networking capabilities had led to a consideration for their application in safety critical systems such as the SIFT fault-tolerant aircraft control system (Wensley et al., 1978). However, thorough analysis of a concrete design problem, namely clock synchronization among multiple clocks, revealed that synchronization algorithms become impossible for 3 clocks if one of them is faulty and can drift arbitrarily. The generalization of this problem, that is, reaching agreement upon a vector of values where each value is the private input of a participant in the agreement protocol, and the agreed upon vector must either contain the private input of each participant, or that the particular participant was faulty, is referred to as interactive consistency. Pease et al. (Pease et al., 1980) were able to show that even in a synchronous system model, i.e., where there is an a priori known upper bound Δ on computation and message transmission times, and a fully connected graph of reliable, point to point communication channels without message authentication, interactive consistency requires 3f +1 participants to arrive at a solution, where f denotes the maximum number of faulty participants that can exhibit arbitrary failures.

A few years later Fischer, Lynch and Paterson (Fischer et al., 1985) showed, what is now referred to as the FLP impossibility result, that deterministic consensus becomes impossible in an asynchronous system, if only a single process is allowed to fail in the crash-stop model, even if communication between processes is reliable. The result, however, does not extend to consensus protocols exhibiting only probabilistic guarantees for liveness or correctness. Hence, so-called randomized Byzantine consensus algorithms, first presented by Ben-Or (Ben-Or, 1983) and Rabin (Rabin, 1983), that instead eventually terminate with probability P(1), or have a non-zero probability for disagreement, are hereby not affected. Nevertheless, at the time, the take-away from these first results were that systems for reaching consensus, in particular in the presence of Byzantine failures, while in principle feasible, were largely impractical for most real-world scenarios (Castro et al., 1999). For instance, the accompanying solutions that were presented for the Byzantine generals problem and interactive consistency have an exponential message complexity in the number of participating processes and the additional computational overhead, as well as large number of replicas that are required to tolerate Byzantine failures, were simply too prohibitive for most use-cases. It would take over a decade until publications such as “Practical Byzantine Fault Tolerance“ (PBFT) by Castro and Liskov (Castro et al., 1999) showed that Byzantine fault tolerant consensus algorithms could indeed be rendered practicable under realistic system assumptions. Nevertheless, while research on the topic of BFT consensus was ongoing (Cachin and Vukolić, 2017a; Clement et al., 2009; Guerraoui et al., 2010; Veronese et al., 2013) it remained a comparatively isolated topic area, given the broad range of potential applications. In part, this may be attributed to the fact that consensus protocols are often discussed in the context of state machine replication (Lamport, 1984; Schneider, 1990),

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 826078.

and achieving active replication for services such as databases. For these scenarios, all replicas, i.e participants, may be under the control of a single entity and the more benign crash-fault tolerance can often be a tenable system model. In particular, Lamport’s crash-fault tolerant Paxos consensus algorithm (Lamport, 1998) and derivations thereof have found their way into practical applications (Chandra et al., 2007). However, even in such scenarios where crash-fault tolerance may have previously been considered acceptable, it can still be advantageous to gain the additional resilience of BFT. In particular, because these previously isolated systems are increasingly becoming interconnected, e.g. by operating in cloud environments (Vukolić, 2010), it appears sensible to be able to tolerate Byzantine failures. Even before the advent of Bitcoin and blockchain technologies, calls from the scientific community had become louder that Byzantine fault tolerant protocols could increasingly meet a wide range of practical demands and should hence be adopted (Clement et al., 2008; Liu et al., 2016). The recent hype surrounding blockchain- and distributed ledger technologies has seemingly provided a crucial stepping stone in this regard and could help achieve more widespread adoption of BFT protocols. Demand for private or consortium blockchains, as well as a quest for achieving better scalability in terms of transaction throughput and resource consumption, has put modern BFT consensus protocols at the heart of many new ledger designs (Vukolić, 2015). Further, research and newly found insights into the fundamental principles and mechanisms of Bitcoin and similar Proof-of-Work based blockchains has resulted in hybrid system models and promising new approaches for BFT protocols (e.g., (Abraham et al., 2018; Chen and Micali, 2019; Miller et al., 2016; Pass and Shi, 2018)). Together, these advancements may facilitate the deployment of BFT protocols in various systems as part of the process of exploring and familiarizing oneself how blockchain technologies could be meaningfully integrated.

4.3.3.2. Proof of Work (PoW)

The proof of work consensus (PoW) (Back, 2002; Finney, 2004), currently employed by Bitcoin and Ethereum blockchains, tries to pay particular attention to security aspects, but comes with expensive costs as miners have to use massive computing power to solve a cryptographic puzzle and win the race for the mining reward. Despite that, blockchains with PoW consensus can still be compromised if an attacker holds more than 51% of the computing power of the entire blockchain. In theory, this should not happen; however, it has been occurring several times: miners join their computing power in groups called mining pools, work collectively and share the incentives (Lewenberg et al., 2015). This collaborative mining has enabled some pools to achieve almost 51% hash power and therefore compromise the blockchain security. Another weak point in PoW is that it is prone to block withholding attacks such as selfish mining. Selfish mining is a technique in which a malicious miner can gain significant and unfair advantage by fooling other nodes into wasting time on cryptographic puzzles related to blocks that will not be included in the blockchain (Eyal and Sirer, 2014). When two parties are mining a block at the same time, the active main chain will split into two forks with the same size. The miners will then decide by majority vote which of the chains is valid. Miners on both forks will continue to validate blocks and add them to their respective chains. The longest chain becomes officially accepted, while the shorter ones will be rejected. As incentives are only awarded to blocks mined on the winning chain, miners of the chain that lost the race will move to the former. A malicious miner can exploit this by creating a stealth fork to which he/she privately adds blocks without broadcasting them to the main network. The corrupt miner would then spend all his/her coins

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 826078.

on the main chain without adding the transaction to their private and corrupted chain (where they would still own the coins). Once the private chain becomes longer than the main chain, the corrupt miner reveals it, resulting in honest miners joining this chain as it is considered the longest and, thus, valid one. As a result, the corrupted chain becomes the official blockchain, and the malicious miner is not only rewarded with the incentives for mining the blocks, but also owns the coins he/she has spent on the legitimate chain, i.e., he/she performed a “double spend attack". The attacker could also bribe honest miners to validate blocks on his/her chain. However, this can be costly as the bribe should pay more than the incentives on the main chain (Bonneau, 2016). Although selfish mining can be very profitable for the attacker, it also presents big risk, since she/he might not be able to win the race for a longer chain and lose the incentives.
In addition to selfish mining, a multitude of different security attacks exists that rely on factors such as network delay, e.g., through distributed denial of service (DDoS), Sybil or Eclipse attacks (Douceur, 2002; Heilman et al., 2015; Johnson et al., 2014) , to exhaust network resources and drive away honest miners. Furthermore, different attacking techniques that target mining pools have been exploited. This includes malicious miners within a pool who try to internally collect more reward share than they deserve, and dishonest miners that use their hashing power to conduct external attacks (Conti et al., 2017).

4.3.3.3. Proof of Stake (PoS)

The proof of stake (PoS) is a potential solution to reduce the computational resources required for PoW, i.e., by establishing a mechanism through which nodes with big stakes of cryptocoins have a better probability of being selected to validate a block (Badertscher et al., 2018). More precisely, PoS is virtual mining, whereby locked funds (stakes) are proportional to the probability of being selected as a leader. This comes from the assumption that nodes with big stakes will not cheat, since they risk losing their stakes if they do. This in turn could lead towards a more centralized blockchain where rich nodes become even richer. Furthermore, PoS is also vulnerable to the 51% attack, if a node or a group of nodes own the majority of coins in the network. The 51% attack would not only be very expensive, but also disadvantageous to the attacker, as it will decrease the value of the coins it is holding. PoS presents, however, other weaknesses, since it can be exposed to different types of attacks such as nothing at stake, fake stake and Sybil attacks (Andreina et al., 2018; Douceur, 2002; Gazi et al., 2018). PoS does not prevent miners from mining on each fork, either by accident or maliciously, and to get the rewards no matter which one wins. Unlike PoW, such procedure has marginal intrinsic cost and therefore the miner has nothing to lose, i.e., there is “nothing at stake" (Andreina et al., 2018). Note that there are different implementations of PoS, e.g., delegated proof of stake (DPoS). Ethereum has been promising for some time to switch to a hybrid proof of stake on top of a proof of work (namely Casper FFG) (Buterin and Griffith, 2019), i.e., a smart contract on top of PoW, which was replaced by plans for a pure proof-of-stake protocol which has not yet been released.

4.3.3.4. Overview of Blockchain Security Research

A general overview of research perspectives and challenges for Bitcoin is given in (Bonneau et al., 2015). The properties of the Bitcoin protocol, termed Nakamoto Consensus (NC), are first formally analyzed in (Garay et al., 2015) and an overview of related research is provided in (Garay and

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 826078.

Kiayias, 2018; Stifter et al., 2018) and (Pass et al., 2017). Hereby, a core goal is to gain a precise understanding of the achievable guarantees NC-style protocols are able to offer. A discerning characteristic between NC and most BFT protocols is a lack of *consensus finality* in the former, i.e. only eventual consistency is achieved. This is a trade-off between *liveness* and *safety* of the underlying consensus [22]. Generally speaking, NC-style blockchains only achieve consistency over a *common prefix* of blocks with a probability that increases exponentially in a security parameter *k*, which represents the number of blocks that are pruned from the end of all consensus participants' chains (Garay et al., 2015).
In practice, this means that the head of a blockchain can change, revert, or be conflicting and must first stabilize before being agreed upon with overwhelming probability. Failing to take this property into consideration may lead to incorrect system states, even within permissioned settings (Natoli and Gramoli, 2016). An analysis of parametrizations of public PoW blockchains and a framework to determine their security and performance is given in (Gervais et al., 2016). Sompolinsky and Zohar summarize and improve upon Bitcoin's security guarantees regarding *double spending attacks*, providing bounds for safely accepting transactions (Sompolinsky and Zohar, 2016). Missing consensus finality in NC also introduces the ability of block withholding attacks (Eyal and Sirer, 2014; Sapirshtein et al., 2016) that adversely affect the *chain quality* (Garay et al., 2015), i.e., the portion of blocks of miners that become part of the canonical chain relative to a miner's hash rate. Zhang and Preneel provide a comprehensive overview and analysis framework for security in PoW blockchains (Zhang and Preneel, 2019). The game-theoretic components and modeling of player incentives in NC-style blockchains is still an open research question.
Recently, Azouvi et al. provides a systematization of literature on the topic (Azouvi and Hicks, 2019). Bribing attacks constitute a particular aspect of rational players and player incentives (McCorry et al., 2018). These range from so-called *Goldfinger attacks* aimed at destroying a cryptocurrency (Bonneau, 2016), over smart contracts for bribing miners (McCorry et al., 2018), to hostile blockchain forking techniques (Judmayer et al., 2018). Ullrich et al. discusses attacking reliable power grid operation through PoW cryptocurrencies (Ullrich et al., 2018). In particular, Routing attacks have the potential to severely disrupt mining operations (Apostolaki et al., 2017) and facilitate this scenario.

Depending on the particular application scenario or use case (e.g., FeatureCloud), blockchain technologies may not be an ideal solution. Wüst and Gervais (Wüst and Gervais, 2018) and Klein et. al. (Klein and Prinz, 2018) provide frameworks for identifying appropriate use cases.
Many permissionless and permissioned blockchain designs are novel and do not provide rigorous security and correctness analyses. Cachin and Vukolic (Cachin and Vukolić, 2017b) discusses a variety of blockchain consensus protocols and analyzes their claimed characteristics. can be substantiated. Consensus protocols that are provided with official clients of permissionless blockchains for implementing local test-nets or private networks, such as Proof-of-Authority, were also shown to be flawed (Ekparinya et al., 2019; Shi, 2018).

4.4. Blockchain Use Cases

So far, we have primarily outlined the characteristics and technical challenges of blockchain technologies without expanding upon the potential use-cases that reach beyond the realm of

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 826078.

cryptographic currencies. The following examples showcase problem domains and scenarios where an application of blockchain and distributed ledger technologies can be both promising and warranted, given that their engineering goals, challenges, desirable properties of the resulting systems and also threat models have various overlaps and similarities to those encountered in the cryptocurrency space. In the following sections, we will demonstrate that these use cases can be very relevant to the FeatureCloud project as they address some of its requirements.

4.4.1 Trusted Timestamping and Data Provenance

The concept of trusted timestamping is not new, and has a wide range of useful applications, such as providing tamper resistant proofs of existence, for instance for intellectual properties such as patent applications, or to document and commit to a particular state or information (e.g. a Merkle tree root which was derived from the relevant system data, a git commit hash). In case of a system breach, where the adversary may have tampered with data, such commitments can serve as vital references to determine their integrity.

However, this of course requires that the commitment itself is safe from manipulation and ideally spread across multiple systems and media. Public Proof-of-Work (PoW) based Blockchains such as Bitcoin present an ideal platform to record such commitments as part of regular transaction data (requiring the commiting party to only pay the appropriate transaction fee). The security and manipulation resistance of such Blockchains stems from the sequential chaining of moderately hard puzzles which renders it (exponentially) increasingly unlikely for an adversary to be able to change any recorded transactions with respect to the length of newly mined blocks.

The advantage of PoW-based constructions over basic signature schemes with one or multiple trusted third parties is that, unless a severe flaw is found within the cryptographic hash function, no private keys or trapdoors exist that efficiently allow for equivocation. That is, if an adversary were to gain access to the private keys used in a signature-based timestamping scheme, they could readily forge backdated commitments with very little resource requirements. However, in a PoW-based model, they would have to re-compute sequential PoWs, which imposes a highly prohibitive constraint both in terms of available time as well as computational resources. Blockchain-based timestamping has been described both in the scientific community (e.g., (Gipp et al., 2015; Szalachowski, 2018)) and is employed in commercial products (e.g., https://guardtime.com/, which is partnered with Lockheed Martin to secure systems engineering processes).

Permissioned blockchain systems that are based on BFT can also offer advantages in combination with signature schemes for timestamping, especially if they employ the use of append only authenticated data-structures such as hash-chains. As long as the signing keys for timestamping are not reused in the BFT consensus mechanism and are therefore independent, an adversary would have to compromise both systems to effectively and fully conceal its malicious activity.

4.4.2 PKI and Digital Identities

An interesting proposition is the utilization of blockchain technologies to record identity information or serve as the basis for public key infrastructures. A general problem with most identity systems is the establishment of trusted infrastructure that secures and links public keys to identities. Blockchain technologies could help augment traditional approaches such as certificate authorities. In particular, more recent developments in this area such as certificate transparency already embrace

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 826078.

authenticated data structures as a means of identifying manipulation attempts. In the context of production systems, such Blockchain-based PKI infrastructure could, for instance, help provide more robust mechanisms for establishing (and revoking) digital identities that are used for aspects such as access control or rights management, both in the development process as well as operation of the system. Another interesting application is in the area of supply chain management, where blockchain-based identity systems may help improve provenance. Proposals from the scientific community, for instance regarding certificate transparency, already exist (Wang et al., 2020), and there are currently concerted development efforts under way for establishing both standards as well as working systems for blockchain-based identity systems [1,2]. However, many of these use-cases raise several important questions regarding user privacy and compliance with legislation and regulations such as the GDPR and are, in part, still open research questions.

4.5. Trusted Timestamping

Trusted timestamping is the process of guaranteeing that the creation and modification times of a data are unable to change once it has been recorded. We differentiate between classical or historical timestamping, and digital timestamping. The former refers to information encoded as metadata, usually the date and time of day, for the purposes of recording when a paper document was sent or received. The latter is the modern expansion of the term, and refers to the timestamping of digital information attached to digital data. A distinction is made between the following two approaches based on access rights:

Centralized: In a centralized approach only the Timestamping Authority has the ability to issue a timestamp, and is therefore able to prevent a user from timestamping something.

Decentralized: In a decentralized approach everybody has the ability to timestamp something. In this approach, nobody is able to prevent a user from timestamping something if they have the possibility of appending data e.g., to a blockchain (assuming a standard which defines the data formats of timestamps exists). This process involves the use of cryptography.

The process of trusted timestamping involves the following three parties:

Prover: The prover is the first entity in the protocol. Depending on the model, it is sometimes assumed that they are an entity with bounded computing resources, and that they cannot be trusted. The prover’s goal is to convince the verifier that he possesses certain information or knowledge. In both, interactive and non-interactive schemes, it is vital to ensure that, no matter what the prover does, he/she is unable to cheat the verifier. However, there are some differences between the models when it comes to the generated proof. "In contrast to interactive timestamping, the time of proof-generation and time of proof-verification may lie far apart" (Landerreche et al., 2020).

[1] https://identity.foundation/
[2] https://www.w3.org/TR/did-core/

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 826078.

Verifier: Being the second entity in the protocol, the verifier wants to make sure that what the prover claims is in fact true, i.e. that he/she is not being cheated by the prover. As is the case with the prover, it is usually assumed that the verifier is bounded in computational power. In interactive proof protocols, these two entities continuously exchange messages until the verifier is convinced that the prover’s claim is correct, i.e. that the probability for error is negligible or acceptable to the verifier. In non-interactive proof protocols there is no requirement for interacting to verify, and it is desirable for the verification computation to be "cheap". However, the prover can also verify his own proof if desired.

Timestamping Authority (TSA): TSA represents what is known as a trusted third party (TTP). It is an entity which enables multiple parties to interact with each other, i.e. those parties that trust the TTP. In this case, the word ’trusted’ refers to the system’s requirement for the TTP to act in the verifier’s interest. It has been shown that data confidentiality can be provided with the use of TTP based Encryption Schemes (Rizvi et al., 2014). When it comes to timestamping, multiple TSAs can be used to increase security and reliability. Defined in the RFC 3161 standard, "the TSA is a TTP that creates time-stamp tokens in order to indicate that a datum existed at a particular point in time" (Adams et al., 2001).

Distributed Timestamping Schemes: In contrast to TTP schemes, in which only one Timestamping Authority (TSA) is employed, distributed schemes handle timestamping in cooperation with multiple systems or parties. This scheme has different security guarantees, compared to a single TSA. Timestamping on a public blockchain like Bitcoin would fall into this category. Note that using multiple TSAs in a TTP scheme is still different from distributed timestamping as, in the former, it would require maintaining a different timestamp for the same data on each TSA, while in the latter only one timestamp is distributedly generated.

4.6. Smart Contracts and Trusted Execution Environments

The currently established term “smart contract” is an unfortunate misnomer when it comes to succinctly describing its principal purpose or functionality, as it easily draws upon associations to legal contracts. A smart contract can be best thought of as program code that is executed in some distributed trusted execution environment. More specifically, the execution environment is generally a distributed or decentralized platform that offers both replication, and more importantly, Byzantine fault tolerance to ensure the correct execution and integrity of the smart contract code and its data storage. This is in contrast to the prevalent approach of implementing Trusted Execution Environments (TEEs) within computer hardware, such as Intel’s SGX platform (Costan and Devadas, 2016; McKeen et al., 2016), where the hardware manufacturer still acts as a single trusted third party to ensure the correctness and integrity of code execution. Permissionless blockchain-based smart contract platforms such as Ethereum (Buterin, 2014), but also permissioned counterparts such as the various incarnations of the Hyperledger platform (Cachin, 2016), offer a unique trusted execution environment for program code, where the correctness and agreement upon the result of computations can be publicly verified and is secured by Byzantine agreement. As generalized

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 826078.

computing platforms[3], the previously mentioned use-cases can readily be implemented within smart contracts, thereby allowing the contract owner to leverage the security and availability of the base platform to provide such services without having to deploy another blockchain where the desired functionality then has to be integrated.

[3] In principle such platforms offer Turing completeness for code, however, executions are generally bounded in their complexity by requiring users to pay a certain price for each operation to prevent trivial denial of service attacks.

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 826078.

5 Requirements Analysis

The purpose of this section is to identify and analyze the requirements of WP6. Consequently, a series of meetings with members of the FeatureCloud consortium has been conducted to identify the problems and understand how WP6 will interface with the other WPs. During the meetings stakeholders are identified and user stories have been gathered, considering both ideal and reasonable use cases. The gathered information lead to a series of questions that will be addressed throughout the project, among which questions that will be handled in WP6.

5.1. Stakeholder Analysis

This section presents the actors that are involved in the overall FeatureCloud system, which are specifically relevant for the user rights management. Studies in FeatureCloud are managed by global project managers (coordinators) and local project managers (participants). Coordinators can also be participants, which requires them to additionally follow the same steps as participants. Although a local project manager is a sub-role within a participant (data provider), In this deliverable, we use both terms interchangeably as no other sub-roles within participants are required for now.

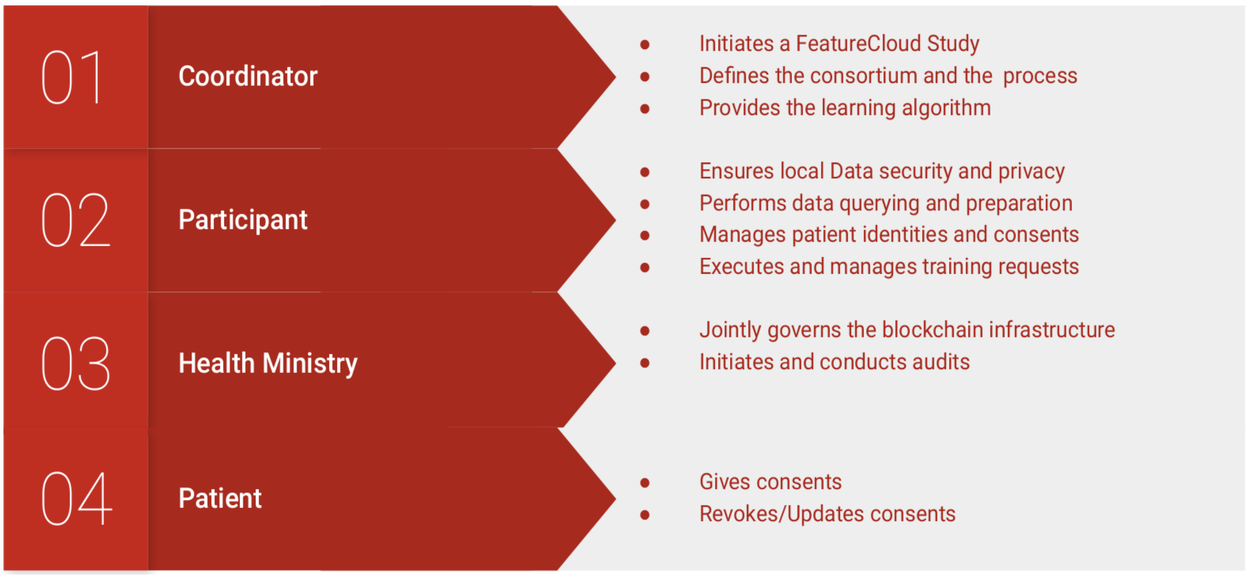

01
Coordinator
Initiates a FeatureCloud Study
Defines the consortium and the process
Provides the learning algorithm
02
Participant
Ensures local Data security and privacy
Performs data querying and preparation
Manages patient identities and consents
Executes and manages training requests
03
Health Ministry
Jointly governs the blockchain infrastructure
Initiates and conducts audits
04
Patient
Gives consents
Revokes/Updates consents

Figure 4. Stakeholders Roles

5.2. Overall User Story

As an example, let's consider the following case. A consortium or a research entity would want to conduct a study on breast cancer survival (e.g., a consortium with 10 hospitals). Therefore, gene expression data from patients are required in order to learn a model that predicts tumor recurrence. A project coordinator would select and invite participating hospitals (each of them has a local project coordinator; i.e. a participant). The project coordinator initiates and coordinates the process, and provides a data analysis (machine learning) application that can be executed locally by the participants (i.e., data providers). Before that, a participant should perform a data discovery check to compute the amount of data relevant to the study, upon which its participation in the study is either confirmed or denied.

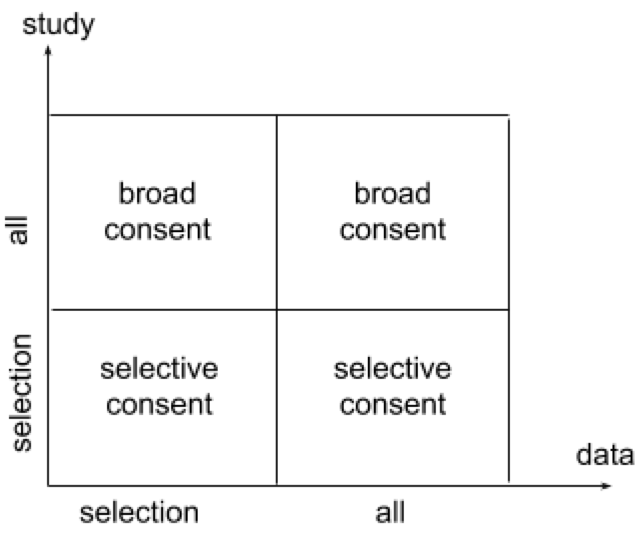


Figure 5. Consent Types

Figure 5 distinguishes between four different consents with respect to data inclusion and the number of studies in question:

A. A consent for using a selection of patient data for all studies.
B. A consent for using all patient data for all studies.
C. A consent for using a selection of patient data for a selection of studies
D. A consent for using all patient data for a selection of studies.

While A and B are broad consents, which means patients give their consent for all the studies, for C and D the consent is restricted to a selection of studies. In D, although the consent is broad with respect to data, it is still considered as selective with respect to the number of studies. Additionally, consents can be categorized according to time: (i) static, which means consent is given once for all future studies or limited with an expiry date, and (ii) dynamic, which means consent is given on the fly for each study. Note that consents are locally managed by each participant (e.g., hospital, biobank).

We can distinguish two different implementations of conducting the study:

- *Ideal world solution*: In an ideal case patients will be asked on the fly for their consent for participating in a study; i.e, dynamic consent. A client application (e.g., mobile app for patients) is then required to receive participation requests and accept or refuse to give consents. Participants (e.g., hospitals) are responsible for querying matching patients and deciding whether they can participate in a study. Only consented data can be used for the FeatureCloud app.
- *Pragmatic solution*: participants have already patient consents or delegated rights to give consents on behalf of patients. In the former case, patients would give a broad consent (not specific to a study) for using all or a specific selection of their data. Patients can also delegate consent management to participants which then accordingly selects appropriate studies for inclusion. Patients should be able to track the history of all their data inclusion in studies.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826078.

A participant is then responsible for the actual “filtering” of data that has permission. He is also responsible for ensuring the whole pre-processing pipeline (e.g., data preparation, format conversion, data standardisation / normalisation). The analysis and preprocessing are performed locally. In the ideal case, data discovery is automated and requires a federated, integrated database. In FeatureCloud, the participant is responsible for the data discovery process; he/she will compile the cohort, and then the process is automated from here. FeatureCloud provides a system that assists the participant, but the participant stays legally responsible.

Automating this process - something we intend to explore and provide where feasible - requires to understand the data in detail. More precisely, different types of data have different levels of integration and therefore require a deeper analysis in order to extract only those parts where consent is given. In the case of electronic health records (EHR), where each such record can be assigned to one patient, this would be possible. Still, formatting the data so that AIs can understand them will still require preprocessing or manual curation, mapping the data onto a common data format ready to be processed by the AI applications. In case of SNP or gene expression data however, one data file usually contains data of all available patients which would require to actively extract parts of it. Conversely, in case of MRI images, one patient usually contributes multiple files which again requires understanding the exact file and data structure. Due to the heterogeneous nature of this data and the lack of suitable standards (i.e. standards allowing for retrieving patient identifiers that could be used to check for given consent), the current goal is to provide the means to manually prepare the data.

The data selected for a particular study will be used as input for the ML program, which is locally executed (e.g., in a container). The result is an output model that will be aggregated with the resulted models of the other participants.

5.3. Studies in FeatureCloud

This section describes how studies can be conducted in FeatureCloud from a user perspective and a high level description how it integrates or interacts with the overall FeatureCloud system. For a more in-depth technical description, including a complete walkthrough, see deliverable 7.2.

5.3.1. User Interaction Workflow

For an overview of the user roles and how they interact, see figure 6.

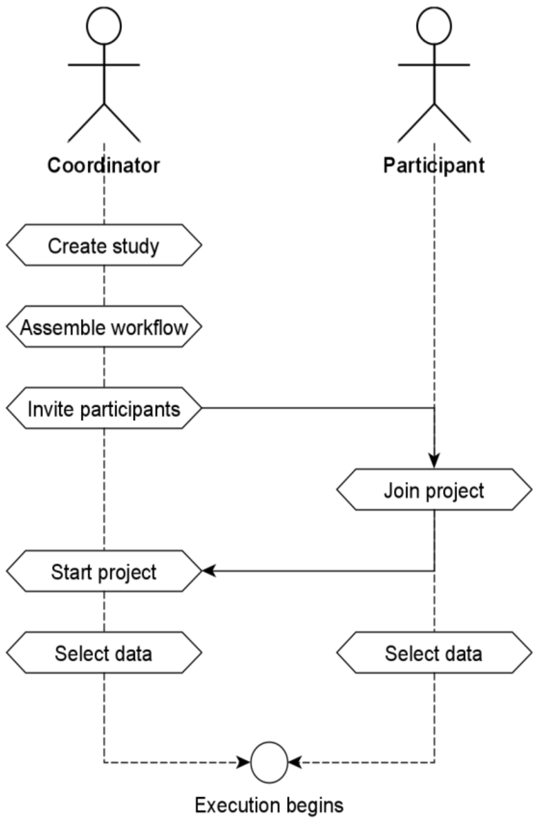


Figure 6. User interaction between coordinator and participant. *Coordinator and participant collaborate in a project by passing a join token.*

Study creation and workflow assembly.
Since all participants need to run the same workflow, its properties, particularly all involved apps, need to be specified on a global level. This is done by the coordinator in the FeatureCloud interface.

Invitation of participants.
Once the workflow is final, other participants can be invited. Changes to the workflow are no longer possible. To invite a participant, a unique token is created and passed to potential collaborators. This token is tied to the project and allows for joining a single time (no re-use of tokens) making the joining party a participant of the project.

Join project.
The token received from the coordinator needs to be entered in the FeatureCloud frontend. It shows a preview of the workflow so that a potential participant can decide based on the selection of apps whether to join or not. This is also related to patient consent as the participant needs to ensure that consent is present for all involved steps of the workflow (i.e. type of required data).

Start project.
After reviewing all parties who joined, the coordinator can start the workflow execution from the FeatureCloud frontend.

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 826078.

Select data.
All participants (usually including the coordinator) select the data they have consent for.

5.3.2. System Integration / Overall FC Architecture

A participant needs to run the so-called FeatureCloud controller. It manages the local execution. On the coordinator side, the controller also orchestrates execution and instructs the participants’ controllers to ensure a globally synchronous execution. Since this deliverable is focused on data use audit and blockchain applicability, it includes a simplified view on the architecture. For a detailed description of the system components and their implementation, see Deliverable 7.2 “ App store ready and extendible by developers”, Section 4.

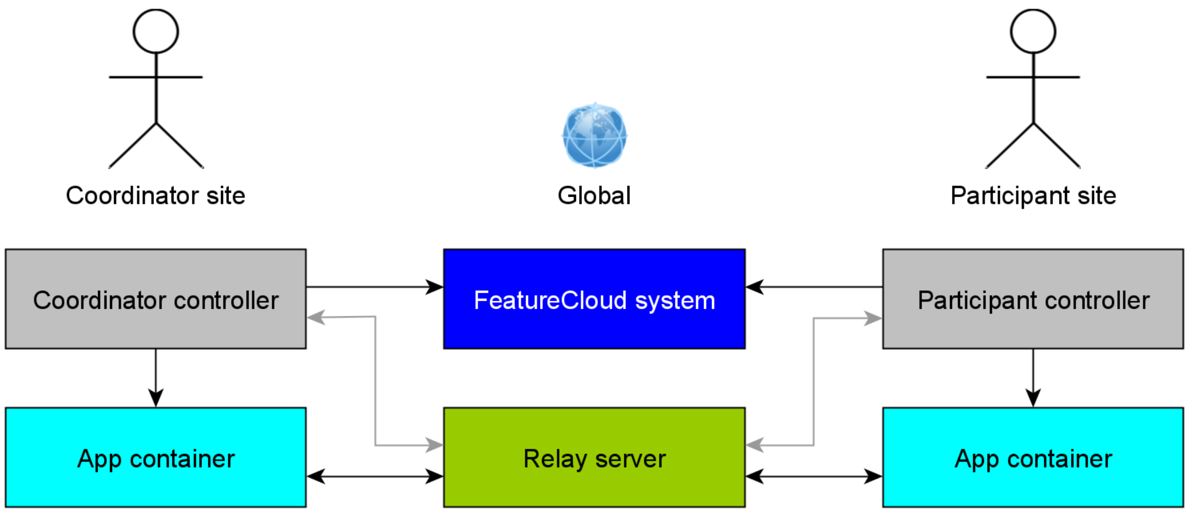


Figure 7. Interaction between participant and coordinator controllers. *Grey connections denote a technical connection (implementation), black a logical connection.*

As shown in Fig.7, app containers cannot communicate directly with each other to restrict Internet access for security reasons. Instead, the controllers pass through the traffic, which use a separate relay server. The FeatureCloud system includes the global API where project details are stored, and the AI store where FeatureCloud apps can be fetched from. Apart from aggregated model parameters, nothing related to patients, their consent or, most importantly, their raw data, leaves a participant’s or coordinator’s site. As mentioned in the previous section, all data the apps have access to had been selected by the participants before.

5.4. Requirements specification

Summarizing the general requirements which needs to be considered for WP6 are as follows:

- The system is expected to work in a minimal trust environment where some of the actors can behave maliciously (excluding the auditor).
- The system is expected to reduce trust assumptions on centralized services, and is able to tolerate or minimize the effects from maliciously acting or compromised participants.

- The system should make the audit process easier for detecting wrong doings.
- The system should ensure traceability, confidentiality and integrity of healthcare data used for studies.
- It should be analyzed if and how patient consents and identities could be managed digitally in a secure way.

To study whether blockchain technology can improve data use audit in the healthcare domain, the following questions will be addressed:

- What are the advantages and disadvantages of different solutions in terms of trust, security, privacy, and cost?
  - What blockchain type (e.g., permissionless, permissioned, public) is more suitable for FeatureCloud?
- Which techniques (e.g., cryptographic algorithms) can be employed as an alternative to, or in conjunction with blockchain to fulfill the aforementioned requirements?

Threat model of data use audit. What are the things that need to be checked during an audit:

- How to prove that the data used for machine learning are the same data output from the discovery phase?
- How to prove that the result output model corresponds to the input data?
- How to prove that a participant does not claim more data than he actually has?
- How to prove that a participant did not exclude eligible data from the study?
- How does the non-determinism of machine learning algorithms impact the proposed solution?
- Is it possible to reduce the impact of non-determinism on the solution?

The following questions have not yet been addressed:

- Which specific blockchain solutions can be employed in the context of FeatureCloud (e.g., Hyperledger, Quorum, Ethereum)? Although a feasibility study on the general applicability of blockchain technology to FeatureCloud is conducted, a decision on a specific implementation(s) has yet to be made in future tasks based on additional requirements for consent management, data representation, and other architectural aspects.
- What is the overall threat model (including threats on consent management)?
- How is blockchain governance conducted? In particular, if a local blockchain mechanism is selected, then an analysis on who will run, add or remove nodes has to be discussed.

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 826078.

6 Feasibility Study

Based on our requirement analysis, we engaged into a feasibility study to evaluate if and how blockchain-mechanisms can be used to benefit the overall FC architecture and workflow. In this Section we describe our overall approach and how we identified time stamping as a method to achieve a pragmatic solution to data usage audit.

6.1. Applicability of blockchain-based solutions

Before designing our architecture, we first evaluated whether or not a blockchain-based approach is needed, or provides benefits compared to blockchain-less solutions. This is of particular importance as the addition of using a blockchain, e.g., compared to a traditional (centralized) database system, may increase the overall system complexity or impose certain design limitations, e.g., the number of operations which can be performed in a particular time frame. Unlike the hype around the topic may suggest, blockchains are certainly not the answer to every possible problem.

We followed the methodology by Wüst and Gervais (Wüst and Gervais, 2018) as a first step towards the question of whether a blockchain, or which kind of blockchain, can be beneficial in our use case scenario. The original flowchart summarizing the methodology is depicted in Fig. 8. Based on this methodology, we describe the analysis of our use case below.

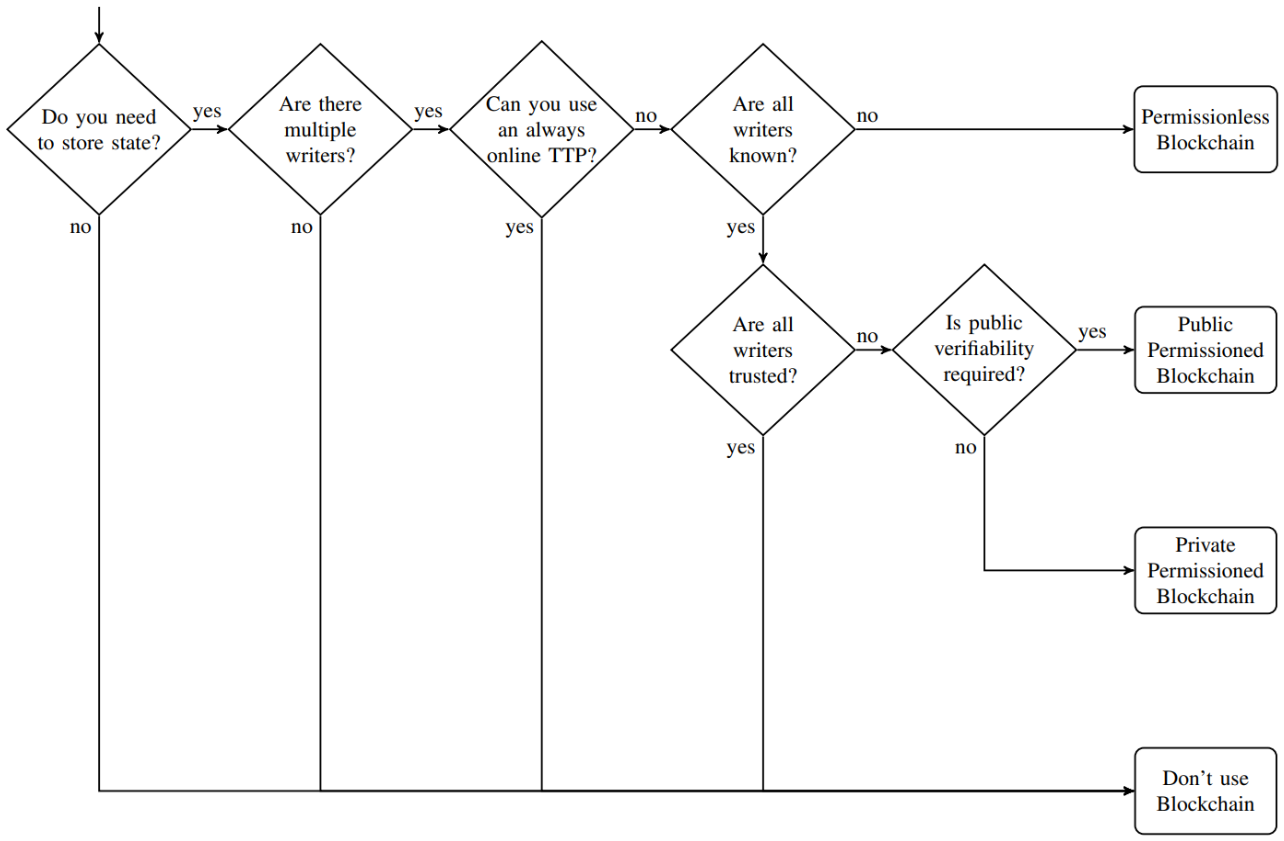

Do you need to store state?
yes
no
Are there multiple writers?
yes
no
Can you use an always online TTP?
no
yes
Are all writers known?
no
yes
Permissionless Blockchain
Are all writers trusted?
no
yes
Is public verifiability required?
yes
no
Public Permissioned Blockchain
Private Permissioned Blockchain
Don’t use Blockchain

Figure 8. A flow chart to determine whether a blockchain is the appropriate technical solution to solve a problem. (Wüst and Gervais, 2018)

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 826078.

Do you need to store state?
Yes, in our application we want to create accountability in the form of an audit-log. This requires certain data (e.g. meta-information about the performed study, consent approvals…), and events to be permanently and securely recorded, i.e., stored.

Are there multiple writers?
Yes, there are a range of different roles (e.g. the Participant Controller (PC)) at multiple institutions (e.g., hospitals participating in a study, the coordinating body setup up the study) which require write access.

Can you use an always online TTP [Trusted Third Party]?
Given that we work with highly sensitive data, it is desirable to minimize trust assumptions as much as possible. Therefore, it would be undesirable to collect / aggregate the data at a centralized trusted third party. However, as we describe in more detail in Section 4, we intend to use cryptographic techniques (hash functions, Merkle tree, cryptographic proofs) to remove the necessity of recording sensitive information. In this way, the impact on privacy in the case of a compromise, or data leaks at a TTP is minimized. However, the cryptographic techniques are *not able to prevent many kinds of manipulation* of the records in case a TTP is compromised. So while it would be possible to rely on a TTP in principle, the additional trust assumptions and risks associated with a potential hack of misbehaving TTP make the use of a TTP undesirable in our scenario.

Are all writers to the ledger known?
Yes, in our case all writers are typically known (initially the participants). However, it is very likely that some actors will not be known when the platform is first launched, but will be later added to the system. Thus, we expect the set of writers to be changing over time. At this decision point in the flowchart, it is not certain in which direction to proceed. A yes answer would indicate public/private *permissioned* blockchain as appropriate technical solutions, whereas no would indicate a *permissionless* system. To follow the methodology, we continue with the last question, and postpone a more detailed discussion on the different options in Section 6.4.

Is public verifiability required?
Considering the use case, where a patient, who has previously given consent for its data being used, wishes to withdraw / revoke its consent, is desirable for the patient to be able to verify that this revocation is indeed recorded. Otherwise, one could always claim that the revocation was never received. As the user (patient) in this case does certainly not take part in running the platform's infrastructure, we consider this as a form of a public verifiability requirement. Although a publicly recorded revocation does not prevent the usage of the corresponding patient data by a participant (will not be detected by the patient), it can still be detected by an audit. The auditor will have to check that consents of data used for a study have not been revoked using the public record. We note, however, that this form of verifiable is quite different to, e.g., transaction verification in public permissionless blockchains like Bitcoin, where each user is able to check the transactions of all users. In our use case, we are more focussed on providing the parties with ability to verify records with concern specifically to them.

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 826078.

Following the analysis, we see that public permissionless and public permissioned blockchains are indeed a viable possible solution for the problem scenario at hand. We note that, interestingly, private (permissioned) blockchain, although at a first glance maybe appealing for privacy reasons, are not on the shortlist of possible solutions. This is due to their lack of verifiability capabilities, in particular for the patients as users of the systems. Also on a first glance, using a public blockchain for data in the medical field seems to raise a range of concerns regarding the confidentiality and privacy of the data being stored. However, our solutions circumvents these issues by a careful design on which data is actually stored. The details in this regard are given in the following section. Essentially no personal, or identifying information is recorded publicly.

6.2. A blockchain based audit-log

As the *local project manager,* as well as the hospitals themself technically have access to all patient data for obvious reasons, there is no direct technical way[4] to prevent them using this data in an unlawful or unauthorized way for which the patient has not given its consent to. Therefore, the main goal of the current design is to identify wrongdoing or misbehaviour of the local project manager *retrospectively* during an audit.
We set out to implement an architecture that allows us to create a permanent record of the machine learning activities managed by the platform. For this purpose, in particular the local manager executing the machine learning algorithms is required to create a new log entry whenever a new study is to be performed. The key elements for such entries include:

- input data (training set) for the machine learning model used
- matching consent confirmations for all patient in the input dataset
- intermediate states
- output data
- meta-information about the study itself, including a unique identifier
- specification of the machine learning algorithm and the corresponding hyper-parameters

To hide all sensitive information, the above elements are not stored directly. Rather the record is composed of fingerprints of the data calculated using cryptographic hash functions in the form of a Merkle tree (as described in Section 4). The properties of the cryptographic hash function ensure that it is not possible to derive any of the information using the publicly stored fingerprint, while the record serves as a binding commitment. Upon request by an auditor, the local manager discloses the requested record(s) (identified by its fingerprints) to the audit, which first verifies that record against the publicly stored fingerprint, and then checks the validity of the consent confirmations to ensure only data which have given consent are actually used to train the machine learning model. Notice that the data is only required/extracted to be managed by the auditor during the duration of the actual audit. This drastically reduces the attack surface, as there is no single party where all data

[4] At least without significant changes to internal data processing processes within hospitals as well as established laws.

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 826078.

is required to be stored. The separation ensures that, in case of a data leak or compromise of one of the parties, the other parties are unaffected.

To hold the local project manager accountable, each record (fingerprint) is further associated with the manager's identity. This identity is established by using asymmetric cryptography, in particular digital signatures. Prior to the use of the system, the local manager generates a cryptographic keypair and registers the public key at the auditor. Upon writing a new record to the blockchain, the record is signed using the corresponding private key. In most blockchain platforms, this functionality is directly available without additional implementation effort as records are stored by executing transactions, which are always authorized by digital signatures.

Figure 9 provides a global overview on the FeatureCloud timestamping architecture. While a participant controller acts as a writer to the DLT for logging the necessary data timestamps, an auditor is granted reading access to be able to check timestamps against the corresponding actual data used for the study. It is also possible for the auditor to log the auditing results. In an ideal case, patient consents are digitized and stored on blockchain. A patient can therefore define consent expiry date or revoke a given consent, which in turn, can be checked by the auditor for detecting the use of non-consented data (this is not the focus of this deliverable). The coordinator collects and aggregates all output models. As the latter are also timestamped, it becomes possible to prove that a coordinator used the rights models for the aggregation. Each participant is responsible for querying and preparing data, executing the machine learning algorithms, and managing consent and identities. Healthcare data is stored locally and securely within each participant and is not transferred between participants during the study.

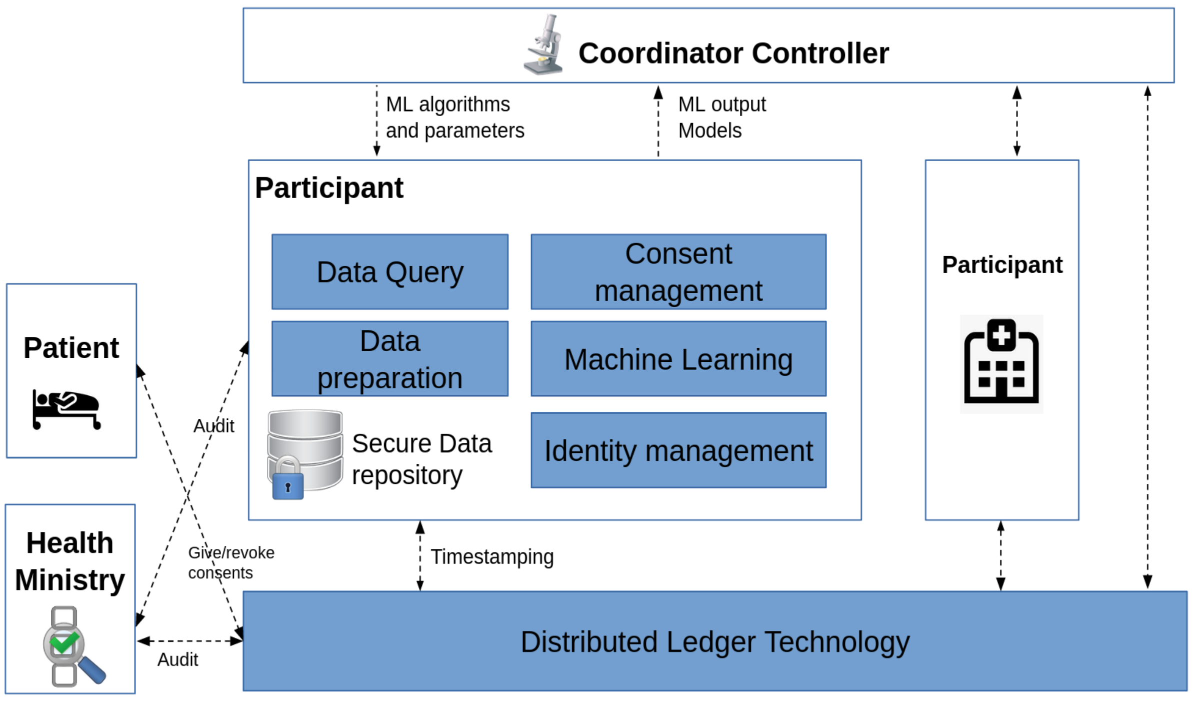


Figure 9. Overall Architecture

6.3. Prevention of selective use of patient data (exclusion)

Similar to the problem of including additional fake data in the input data for machine learning, a malicious participant may also exclude specific eligible and consented data from being used as input. This can be useful to influence the resulting output models (e.g., eliminating data that influence negatively a recommendation for a specific drug). This clearly represents a more challenging problem as it not only requires maintaining the state of the database when the selection query is carried, but also ensuring its integrity over time (history of all states and applied CRUD operations). Two alternatives are possible for the attacker: (i) to modify the selection query to exclude specific items, or (ii) apply the correct selection query to a modified or a view of the database. Alternative (i) can be addressed by timestamping the query, which in turn, will be rerun during the audit. As in this scenario the database is not modified, a replay of the query should output the same data selection. In (ii) a timestamp of the query will not be sufficient because a rerun of the latter on the modified database will not show any malicious behavior. The problem shifts to ensuring that the database state before the query has not been maliciously changed, which requires rolling over all the database state history. One possible solution requires a regular timestaming (on a daily or monthly basis) of the database state. An auditor can therefore replay the selection query on the database state at the query time (Pröll and Rauber, 2013). To prevent modifying the database before the query and reestablishing its state right after it, within the timestamp interval (daily or monthly), an auditor can replay the query on several time-stamped states surrounding the query time (random depth) and compare the results by studying their variance factor. However, it has to be decided on an acceptable state change (the variance between results) and what to be considered as a malicious modification.

6.4. Technical requirements for the used blockchain

At this stage of the project, we decide not to narrow the focus of our work to a specific blockchain, but rather analyze the requirements a blockchain has to fulfill as the basis for our audit-storage layer of our platform. We consider a layered design approach, where our solution can in principle be set up on top of any blockchain platform with suitable security and performance guarantees. This allows for greater flexibility in the later progress of the project, as the underlying blockchain platform can be easily changed in case new requirements emerge.

6.4.1 Transaction Throughput and Size

Considering the current feature set, where the application commits the fingerprints of the created records into the selected blockchain system instead of writing the actual data, the performance (throughput) requirements for the underlying blockchain are significantly reduced. As fingerprints (e.g. using SHA2-256 or SHA3-256 algorithms) are only 32 bytes, the sizes for individual transactions are very small and comparable to typically financial transactions processed in common blockchain solutions. Including considerations for logging intermediate states during the training of the machine learning models, we only expect new audit records to be published in hourly intervals. However, widely available blockchain solutions already support tens to hundreds to thousands of transactions per second. Given that both the rate of transactions as well as transaction sizes are

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 826078.

low/small, all of these solutions are equally applicable considering our requirement regarding transaction throughput.

6.4.2 Immutability

The underlying blockchain solution has to ensure that after an audit-log record is recorded in the blockchain, it cannot be tampered with, be altered, or removed from the blockchain data structure anymore. When using a common public permissionless blockchain, like, e.g. Ethereum or Bitcoin, this requirement is achieved after a sufficient waiting period (e.g. typically an hour for Bitcoin) for each record. When considering to use a (BFT-based) permissioned blockchain instead, immutability of a record is achieved as soon as a supermajority (typically more than ⅔ of the nodes) agrees on the inclusion of the record. In this case, immutability depends on the trust assumption of the institutions running the blockchain. Therefore, it is crucial that the blockchain infrastructure is, e.g., not run by a single local manager, or by the same local-manager acting at different hospitals.

6.5. Technical challenges and possible problematic blockchain designs

Based on our prior overview of blockchain security research, we highlight designs, system characteristics, and assumptions that are indicative of unproven or risky approaches for integrating blockchain technologies. Systems that exhibit such properties are not necessarily vulnerable, however there exists considerably less research and experience, requiring additional diligence and careful protocol analysis to ensure correctness and security.

Permissioned + Proof-of-Work: An application of PoW in a permissioned setting is indicative of a less than ideal system configuration. The security guarantees of PoW are derived from an honest computational majority (Nakamoto, 2008). PoW can be readily outsourced to other hardware, and mechanisms to prevent this generally rely on game-theoretic incentives that do not apply in most permissioned systems. Securing a permissioned blockchain by PoW hence requires a large amount of computational resources to ensure ample security against an adversary compromising a mining node and increasing its mining capacity through outsourcing. Further, PoW-based consensus generally only offers eventual consistency and not consensus finality.

Unproven Protocol Components and Primitives: This includes consensus protocols and cryptographic algorithms that have not been formally analyzed and rigorously studied. Further, protocol compositions should also be re-evaluated as their security guarantees may change.

Additional Correctness Requirements: Components or participants that are required to be correct at all times, beyond the defined thresholds for BFT, may present a single point of failure and can introduce further security risks. Different failure tolerance thresholds or permissible types of failures within the same system design also warrant closer inspection.

Permissioned + Dynamic Consensus Membership: Achieving dynamic group membership % i.e. the ability for consensus nodes to dynamically change their configuration in a Byzantine setting is a

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 826078.

difficult problem (Stifter et al., 2018). Bitcoin and Nakamoto consensus NC presents a particular solution in the permissionless setting, however general solutions for a permissioned environment with realistic system assumptions remain an open research question. Protocols that intend to achieve such guarantees should be carefully analyzed.

Permissioned + Incentives: The assumption of additional game-theoretic incentives, such as monetary reward structures, can be problematic, as there is still considerably less security research on protocols employing these mechanisms (Azouvi and Hicks, 2019). Utilizing monetary incentives may furthermore introduce unwanted new attack surfaces through bribing and as a direct bounty for successful attacks.

Consistency Requirements + No Consensus Finality: Special care needs to be taken if consensus safety guarantees are weakened, e.g., by a lack of finality for blockchain and distributed ledger proposals that do not offer consensus finality, i.e. have weakened consensus safety guarantees. Unless confirmation times are carefully and appropriately chosen, various attacks become possible (Sompolinsky and Zohar, 2016). Proposals and use cases where this is the case should explicitly acknowledge and address the possible effects of forks, long chain reorganizations, and state reversions.

Sensitive Data in Transactions and Ledger: If data contained within the ledger or any transactions may be used by an adversary for attacks, the design should be closely examined for possible vulnerabilities. An issue related to this topic is the prevention of unwanted content within permissionless ledgers (Matzutt et al., 2018) and possible approaches for redacting already committed data (Deuber et al., 2019). In FeatureCloud, no actual sensitive data will be stored on the ledger, but rather the corresponding timestamps and possibly digital consents required for the audit.

The above overview highlights that key characteristics of novel blockchain protocols are still the topic of discussion and their provided guarantees are not always clear. Many of the outlined attacks against NC-style PoW blockchains may also be adopted by an adversary in a permissioned PoW setting, once mining nodes are compromised to assume its identity. If a modest set of (permissioned) consensus participants is acceptable, relying on well-established BFT protocols currently remains a prudent approach, as the well researched, and often stronger, security guarantees and characteristics offer a clear advantage (Vukolić, 2015).

6.6. Exploration of blockchainless solutions

As stated in Section 4, we distinguish between two timestamping architectures: (i) centralized that employs a trusted timestamping authority (TSA), and (ii) distributed timestamping schemes, into which fall public blockchains. In this regard, it is worth noting that it is also possible to envisage a blockchainless solution to address the timestamping of FeatureCloud data (e.g., using one or multiple TSAs). Although such a solution can be practical, it obviously presents lesser security guarantees in terms of availability, immutability, central point of failure, and flexibility. As the scope of FeatureCloud goes beyond the simple use of timestamping, and requires more complex

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 826078.

operations for managing consents and identities (as part of next deliverables), blockchain technology presents itself as a more promising solution.

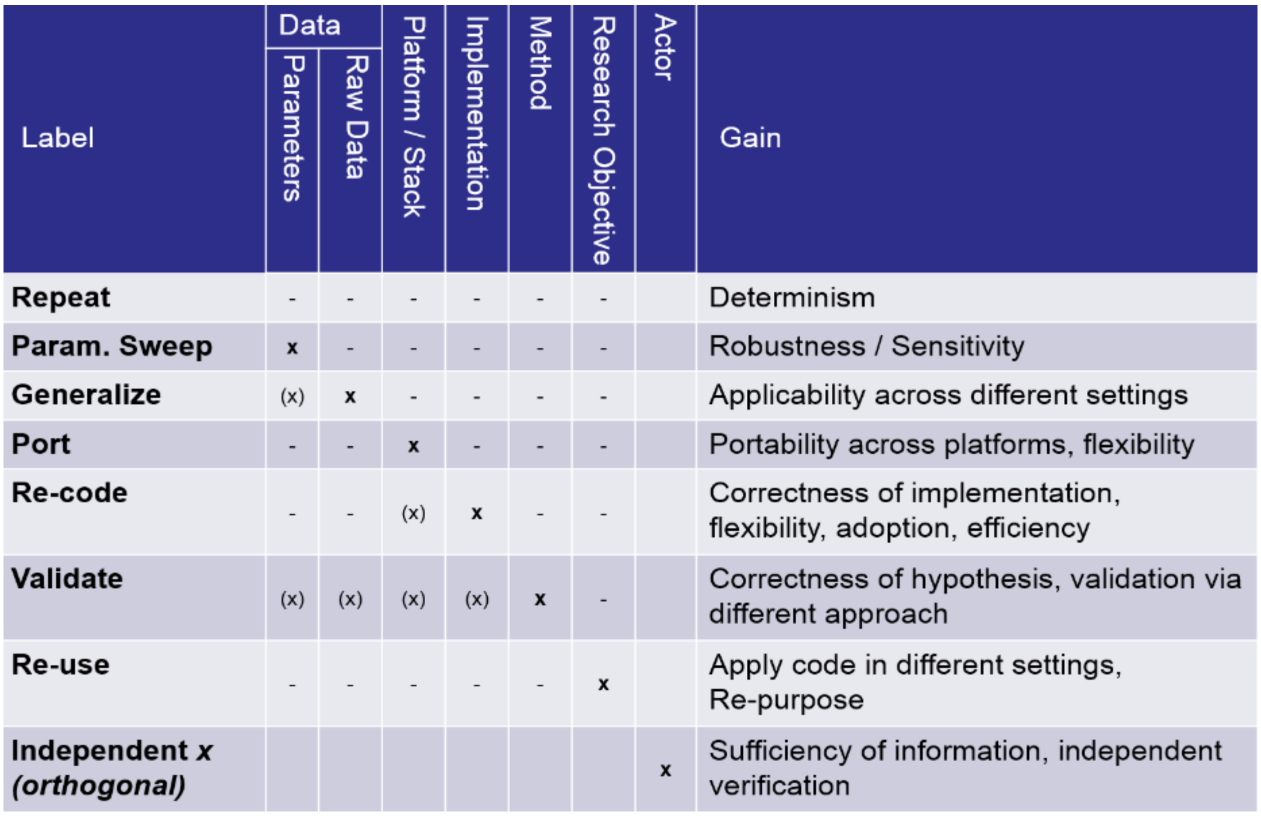
7 Open issues

The following issues remain open and will be further investigated in future deliverables.

7.1. Determinism and repeatability in data analysis

Repeatability of data analysis processes, such as learning a machine learning model, is generally considered a desired property. This would be the basis for allowing data analysis processes to be **deterministic**, which is an important **basis for auditing** such data analysis processes. However, repeatability is notoriously difficult to achieve. Several recent works have analysed the problem of repeatability and more challenging tasks such as replicability, reproducibility, and tried to identify reasons for non-determinism and repeatability.

Terminology is not uniform, and several attempts have tried to define these terms - e.g. (Roure, 2014) lists 21 different terms, and groups them into 6 categories, and (Ivie and Thain, 2018) provides a recent survey focussed on reproducibility. We consider the model given in (Freire et al., 2016), which identifies potential sources of changes in process execution, given in Figure 10.



Label	Data		Platform / Stack	Implementation	Method	Research Objective	Actor	Gain
	Parameters	Raw Data						
Repeat	-	-	-	-	-	-		Determinism
Param. Sweep	x	-	-	-	-	-		Robustness / Sensitivity
Generalize	(x)	x	-	-	-	-		Applicability across different settings
Port	-	-	x	-	-	-		Portability across platforms, flexibility
Re-code	-	-	(x)	x	-	-		Correctness of implementation, flexibility, adoption, efficiency
Validate	(x)	(x)	(x)	(x)	x	-		Correctness of hypothesis, validation via different approach
Re-use	-	-	-	-	-	x		Apply code in different settings, Re-purpose
Independent *x* *(orthogonal)*							x	Sufficiency of information, independent verification

Figure 10: PRIMAD model of reasons why process executions differ (Freire et al., 2016)

Regarding the setting of the FeatureCloud project, the main question to identify is which aspects can be guaranteed to be unmodified, to enable a repeatable process execution, and thus determinism. We can assume that the columns towards the right, i.e. the actor, research object, and method are kept unchanged - these are in general more relevant when the process is to be reproduced, and thus performed by stakeholders external to the initial execution. Therefore, rather the aspects given in the first columns, such as data, platform / stack and implementation are to be considered. It is thus important for FeatureCloud to ensure that the implementation and platform/stack are kept constant. For the implementation, this entails e.g. versioning and archiving of the applications that are executed. For the platform/stack, using the approach of containers that FeatureCloud adopts enables such versioning and archiving of **parts** of the platform/stack as well, namely those layers specifically utilised in the respective containers. However, the parts of the platform/stack that are outside the containers, i.e. the hosts, still need to be considered specifically.
Regarding the data, the model distinguishes between **raw data** and parameters. While raw data is in the FeatureCloud case to be considered the input data that the platform provides, and for which mechanisms to retrieve the same exact input at even a later stage should be retrievable, other aspects are in the control of the application provider, or influenced by other aspects. As such, the application provider will generally choose the (hyper-)parameters that are required for e.g. training the machine learning models. The FeatureCloud platform can here provide guidelines and rules on how to make this aspect deterministic, and respective applications could be marked as conforming. A further aspect is randomisation during the process, e.g. for choosing initialisation values, determining the sequence of training samples, etc. Also, these can be made deterministic in most cases (e.g. by having a fixed initialisation to the pseudo-random number generators). Another category of potential data changes could be if applications also retrieve data from other, external, third-party sources, and these influence the data analysis process. Internalising such services by the application providers could be a valid strategy in some cases.

In overall, however, FeatureCloud must assume that not all processes will be deterministic, and consider the impacts of this for the auditing process.

7.2. Fuzzy Hashing to identify near-duplicates

With the above outlined issue of determinism in executing data analysis processes, a potential solution to address the problem is to change the underlying mechanism to identify whether two executions of a process are identical, to a more relaxed approach of verifying whether they produce similar results. The assumption behind this approach is that, even with different initializations or other random factors in the process, the result of the process should be *similar* in most cases. E.g. when training a neural network, maybe the learned parameters are not exactly identical in two instances of the (somewhat randomised) training process, but they will be to some extent similar.
We outlined above that we want a cryptographic hash function to produce a rather different digest for even just marginally different inputs (the so-called avalanche effect), to render making predictions about the input, given only the output, difficult.

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 826078.

A slightly contrary concept is the one of fuzzy hashing (Baier and Breitinger, 2011), sometimes also called piecewise hashing, which aims at creating similar hash values for similar input values, and is thus a potential solution to determine whether two inputs are similar, rather than identical, without having to analyse said inputs themselves. Many fuzzy hashing algorithms include two complementary algorithms - the hash algorithm itself, and the comparison algorithm for determining whether hashes "match."
Fuzzy hash algorithms often partition the input data into a number of blocks (either fixed-size or a size dependent on the input data) which are then hashed. This set of hashes, the "fuzzy hash," is then compared to other fuzzy hashes using a metric or distance by which it can be decided whether the inputs are similar or not. Fuzzy hashing is frequently employed in settings where a fast comparison of data is needed, e.g. malware or spam detection, but also file search, and digital forensics in general (Roussev, 2011).

Utilizing fuzzy hashing might alleviate the issue of non-determinism to some extent, even though the amount by which the results of the data analysis processes are allowed to vary is very likely specific to the processes itself, and might need to be defined on a case-by-case basis by the process provider; it further remains to be seen whether such varying degree of expected divergence in results can be utilised as a parameter to fuzzy hashing comparisons.

Finally, employing a similarity instead of identity check for results (e.g. via fuzzy hashing, but also other methods) entails that there might be an increased attack surface for a malicious local project manager or other adversaries, as it becomes easier to determine a set of data analysis process outputs that would be potentially based on malicious use of input data, but still be considered acceptable due to the inherent non-determinism of the process. This attack potential needs to be investigated before such solutions should be deployed.

8 Conclusion

Blockchains, and related hash-chain based technologies, offer a multitude of interesting possibilities for medical sciences. Still, the field of blockchain technologies is actually quite broad and must not be reduced to Nakamoto-like (Bitcoin-like) techniques. With all the different basic technologies and designs for blockchains, a thorough requirements analysis is of vital importance. In this deliverable, we have conducted a primary requirement analysis, in which we have identified relevant stakeholders, and gathered and analyzed requirements related to audit processes. A feasibility study proved that blockchain technology can indeed improve and facilitate the audit process while providing better security guarantees. In particular, a blockchain based audit-log has been presented, and different protocol designs as well as technical challenges have been discussed. However, a decision on which specific blockchain implementation (e.g., Ethereum, Hyperledger) will be adopted is a future work and depends on additional requirements for consent and identity management, data representation, and other architectural aspects. Finally, a discussion on open issues is presented, where we outlined (i) how the non-determinism of ML programs can impact the proposed solution, and (ii) how fuzzy hashing can address this problem by identifying near-duplicate executions.

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 826078.

9 References

Abraham, I., Gueta, G., Malkhi, D., 2018. Hot-Stuff the Linear, Optimal-Resilience, One-Message BFT Devil.

Adams, C., Cain, P., Pinkas, D., Zuccherato, R., 2001. Internet X. 509 public key infrastructure time-stamp protocol (TSP). RFC3161 8.

Andreina, S., Bohli, J.-M., Karame, G.O., Li, W., Marson, G.A., 2018. PoTS - A Secure Proof of TEE-Stake for Permissionless Blockchains (No. 1135).

Apostolaki, M., Zohar, A., Vanbever, L., 2017. Hijacking bitcoin: Routing attacks on cryptocurrencies, in: 2017 IEEE Symposium on Security and Privacy (SP). IEEE, pp. 375–392.

Azouvi, S., Hicks, A., 2019. SoK: Tools for Game Theoretic Models of Security for Cryptocurrencies, arXiv preprint arXiv:1905.08595.

Back, A., 2002. Hashcash - A Denial of Service Counter-Measure.

Badertscher, C., Gazi, P., Kiayias, A., Russell, A., Zikas, V., 2018. Ouroboros Genesis: Composable Proof-of-Stake Blockchains with Dynamic Availability (No. 378).

Baier, H., Breitinger, F., 2011. Security Aspects of Piecewise Hashing in Computer Forensics, in: 2011 Sixth International Conference on IT Security Incident Management and IT Forensics. Presented at the 2011 6th International Conference on IT Security Incident Management and IT Forensics (IMF 2011), IEEE, Stuttgart, Germany, pp. 21–36. https://doi.org/10.1109/IMF.2011.16

Bano, S., Sonnino, A., Al-Bassam, M., Azouvi, S., McCorry, P., Meiklejohn, S., Danezis, G., 2017. Consensus in the Age of Blockchains.

Ben-Or, M., 1983. Another advantage of free choice (extended abstract): Completely asynchronous agreement protocols, in: Proceedings of the Second Annual ACM Symposium on Principles of Distributed Computing. ACM, pp. 27–30.

Bonneau, J., 2016. Why buy when you can rent? Bribery attacks on Bitcoin consensus, in: BITCOIN '16: Proceedings of the 3rd Workshop on Bitcoin and Blockchain Research.

Bonneau, J., Miller, A., Clark, J., Narayanan, A., Kroll, J.A., Felten, E.W., 2015. SoK: Research Perspectives and Challenges for Bitcoin and Cryptocurrencies, in: 2015 IEEE Symposium on Security and Privacy. Presented at the 2015 IEEE Symposium on Security and Privacy (SP), IEEE, San Jose, CA, pp. 104–121. https://doi.org/10.1109/SP.2015.14

Buterin, V., 2014. Ethereum: A next-generation smart contract and decentralized application platform.

Buterin, V., Griffith, V., 2019. Casper the Friendly Finality Gadget. ArXiv171009437 Cs.

Cachin, C., 2016. Architecture of the Hyperledger Blockchain Fabric.

Cachin, C., Vukolić, M., 2017a. Blockchain Consensus Protocols in the Wild.

Cachin, C., Vukolić, M., 2017b. Blockchain Consensus Protocols in the Wild.

Castro, M., Liskov, B., others, 1999. Practical Byzantine fault tolerance, in: OSDI. pp. 173–186.

Chandra, T.D., Griesemer, R., Redstone, J., 2007. Paxos made live: an engineering perspective, in: Proceedings of the Twenty-Sixth Annual ACM Symposium on Principles of Distributed Computing. pp. 398–407.

Chen, J., Micali, S., 2019. Algorand: A secure and efficient distributed ledger. Theor Comput Sci 777, 155–183. https://doi.org/10.1016/j.tcs.2019.02.001

Clement, A., Marchetti, M., Wong, E., Alvisi, L., Dahlin, M., 2008. BFT: the time is now, in: Proceedings of the 2nd Workshop on Large-Scale Distributed Systems and Middleware. pp. 1–4.

Clement, A., Wong, E.L., Alvisi, L., Dahlin, M., Marchetti, M., 2009. Making Byzantine Fault

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826078.

Tolerant Systems Tolerate Byzantine Faults., in: NSDI. pp. 153–168.
Conti, M., E, S.K., Lal, C., Ruj, S., 2017. A Survey on Security and Privacy Issues of Bitcoin.
Costan, V., Devadas, S., 2016. Intel SGX Explained. IACR Cryptol EPrint Arch 2016, 1–118.
Deuber, D., Magri, B., Thyagarajan, S.A.K., 2019. Redactable Blockchain in the Permissionless Setting. 2019 IEEE Symp. Secur. Priv. SP 124–138. https://doi.org/10.1109/SP.2019.00039
Douceur, J.R., 2002. The sybil attack, in: International Workshop on Peer-to-Peer Systems. Springer, pp. 251–260.
Ekparinya, P., Gramoli, V., Jourjon, G., 2019. The Attack of the Clones Against Proof-of-Authority. ArXiv190210244 Cs.
Eyal, I., Sirer, E.G., 2014. Majority is not enough: Bitcoin mining is vulnerable, in: Financial Cryptography and Data Security. Springer, pp. 436–454.
Finney, H., 2004. Reusable Proofs of Work (RPOW).
Fischer, M.J., Lynch, N.A., Paterson, M.S., 1985. Impossibility of distributed consensus with one faulty process, in: Journal of the ACM (JACM). ACM, pp. 374–382.
Freire, J., Fuhr, N., Rauber, A., 2016. Report from Dagstuhl seminar 16041: reproducibility of data-oriented experiments in e-science. Dagstuhl Rep. 6, 108–159.
Garay, J., Kiayias, A., 2018. SoK: A Consensus Taxonomy in the Blockchain Era.
Garay, J., Kiayias, A., Leonardos, N., 2015. The bitcoin backbone protocol: Analysis and applications, in: Advances in Cryptology-EUROCRYPT 2015. Springer, pp. 281–310.
Gazi, P., Kiayias, A., Russell, A., 2018. Stake-Bleeding Attacks on Proof-of-Stake Blockchains, in: Crypto Valley Conference on Blockchain Technology, CVCBT 2018, Zug, Switzerland, June 20-22, 2018. IEEE, pp. 85–92. https://doi.org/10.1109/CVCBT.2018.00015
Gervais, A., Karame, G.O., Wüst, K., Glykantzis, V., Ritzdo rf, H., Capkun, S., 2016. On the security and performance of proof of work blockchains, in: Proceedings of the 2016 ACM SIGSAC. ACM, pp. 3–16.
Gipp, B., Meuschke, N., Gernandt, A., 2015. Decentralized Trusted Timestamping using the Crypto Currency Bitcoin. ArXiv150204015 Cs.
Guerraoui, R., Knežević, N., Quéma, V., Vukolić, M., 2010. The next 700 BFT protocols, in: Proceedings of the 5th European Conference on Computer Systems. ACM, pp. 363–376.
Heilman, E., Kendler, A., Zohar, A., Goldberg, S., 2015. Eclipse Attacks on Bitcoin’s Peer-to-Peer Network, in: 24th USENIX Security Symposium (USENIX Security 15). pp. 129–144.
Hoepman, J.-H., 2007. Distributed Double Spending Prevention., in: Security Protocols Workshop. Springer, pp. 152–165.
Ivie, P., Thain, D., 2018. Reproducibility in Scientific Computing. ACM Comput Surv 51. https://doi.org/10.1145/3186266
Jarecki, S., Odlyzko, A., 1997. An efficient micropayment system based on probabilistic polling, in: Financial Cryptography. Springer, pp. 173–191.
Johnson, B., Laszka, A., Grossklags, J., Vasek, M., Moore, T., 2014. Game-theoretic analysis of DDoS attacks against Bitcoin mining pools, in: International Conference on Financial Cryptography and Data Security. Springer, pp. 72–86.
Judmayer, A., Stifter, N., Schindler, P., Weippl, E., 2018. Pitchforks in Cryptocurrencies: Enforcing rule changes through offensive forking- and consensus techniques (Short Paper), in: CBT’18: Proceedings of the International Workshop on Cryptocurrencies and Blockchain Technology.
Klein, S., Prinz, W., 2018. A use case identification framework and use case canvas for identifying and exploring relevant blockchain opportunities, in: Proceedings of 1st ERCIM Blockchain Workshop 2018. European Society for Socially Embedded Technologies (EUSSET).
Krishnankutty, B., Bellary, S., Kumar, N.B.R., Moodahadu, L.S., 2012. Data management in clinical research: An overview. Indian J. Pharmacol. 44, 168–172. https://doi.org/10.4103/0253-7613.93842

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 826078.

Lamport, L., 1998. The part-time parliament, in: ACM Transactions on Computer Systems (TOCS). ACM, pp. 133–169.
Lamport, L., 1984. Using Time Instead of Timeout for Fault-Tolerant Distributed Systems., in: ACM Transactions on Programming Languages and Systems (TOPLAS). ACM, pp. 254–280.
Lamport, L., Shostak, R., Pease, M., 1982. The Byzantine generals problem, in: ACM Transactions on Programming Languages and Systems (TOPLAS). ACM, pp. 382–401.
Landerreche, E., Stevens, M., Schaffner, C., 2020. Non-interactive cryptographic timestamping based on verifiable delay functions, in: International Conference on Financial Cryptography and Data Security. Springer, pp. 541–558.
Lewenberg, Y., Bachrach, Y., Sompolinsky, Y., Zohar, A., Rosenschein, J.S., 2015. Bitcoin mining pools: A cooperative game theoretic analysis, in: Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems. International Foundation for Autonomous Agents and Multiagent Systems, pp. 919–927.
Liu, S., Viotti, P., Cachin, C., Quema, V., Vukolic, M., 2016. XFT: Practical Fault Tolerance beyond Crashes, in: 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16). USENIX Association, Savannah, GA, pp. 485–500.
Matzutt, R., Henze, M., Ziegeldorf, J.H., Hiller, J., Wehrle, K., 2018. Thwarting Unwanted Blockchain Content Insertion.
McCorry, P., Hicks, A., Meiklejohn, S., 2018. Smart Contracts for Bribing Miners, in: 5th Workshop on Bitcoin and Blockchain Research, Financial Cryptography and Data Security 18 (FC). Springer.
McKeen, F., Alexandrovich, I., Anati, I., Caspi, D., Johnson, S., Leslie-Hurd, R., Rozas, C., 2016. Intel® software guard extensions (intel® sgx) support for dynamic memory management inside an enclave, in: Proceedings of the Hardware and Architectural Support for Security and Privacy 2016. pp. 1–9.
Miller, A., Xia, Y., Croman, K., Shi, E., Song, D., 2016. The honey badger of BFT protocols, in: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. ACM, pp. 31–42.
Nakamoto, S., 2008. Bitcoin: A Peer-to-Peer Electronic Cash System.
Narayanan, Arvind and Bonneau, Joseph and Felten, Edward and Miller, Andrew and Goldfeder, Steven, 2016. Bitcoin and Cryptocurrency Technologies, Bitcoin and Cryptocurrency Technologies.
Natoli, C., Gramoli, V., 2016. The Blockchain Anomaly. ArXiv160505438 Cs.
Nijhawan, L.P., Janodia, M.D., Muddukrishna, B.S., Bhat, K.M., Bairy, K.L., Udupa, N., Musmade, P.B., 2013. Informed consent: Issues and challenges. J. Adv. Pharm. Technol. Res. 4, 134–140. https://doi.org/10.4103/2231-4040.116779
Pass, R., Seeman, L., Shelat, A., 2017. Analysis of the Blockchain Protocol in Asynchronous Networks, in: Coron, J.-S., Nielsen, J.B. (Eds.), Advances in Cryptology – EUROCRYPT 2017, Lecture Notes in Computer Science. Springer International Publishing, Cham, pp. 643–673. https://doi.org/10.1007/978-3-319-56614-6_22
Pass, R., Shi, E., 2018. Thunderella: Blockchains with Optimistic Instant Confirmation, in: Nielsen, J.B., Rijmen, V. (Eds.), Advances in Cryptology – EUROCRYPT 2018, Lecture Notes in Computer Science. Springer International Publishing, Cham, pp. 3–33. https://doi.org/10.1007/978-3-319-78375-8_1
Pease, M., Shostak, R., Lamport, L., 1980. Reaching agreement in the presence of faults, in: Journal of the ACM (JACM). ACM, pp. 228–234.
Pröll, S., Rauber, A., 2013. Scalable data citation in dynamic, large databases: Model and reference implementation, in: 2013 IEEE International Conference on Big Data. Presented at the 2013 IEEE International Conference on Big Data, pp. 307–312. https://doi.org/10.1109/BigData.2013.6691588

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 826078.

Rabin, M.O., 1983. Randomized byzantine generals, in: Foundations of Computer Science, 1983., 24th Annual Symposium On. IEEE, pp. 403–409.
Rizvi, S., Cover, K., Gates, C., 2014. A trusted third-party (TTP) based encryption scheme for ensuring data confidentiality in cloud environment. Procedia Comput. Sci. 36, 381–386.
Roure, D.D., 2014. The future of scholarly communications. Insights 27, 233–238.
Roussev, V., 2011. An evaluation of forensic similarity hashes. Digit. Investig. 8, S34–S41. https://doi.org/10.1016/j.diin.2011.05.005
Sapirshtein, A., Sompolinsky, Y., Zohar, A., 2016. Optimal selfish mining strategies in bitcoin, in: International Conference on Financial Cryptography and Data Security. Springer, pp. 515–532.
Schneider, F.B., 1990. Implementing fault-tolerant services using the state machine approach: A tutorial, in: ACM Computing Surveys (CSUR). ACM, pp. 299–319.
Shi, E., 2018. Analysis of Deterministic Longest-Chain Protocols (No. 1079).
Sompolinsky, Y., Zohar, A., 2016. Bitcoin’s Security Model Revisited. ArXiv Prepr. ArXiv160509193.
Stifter, N., Judmayer, A., Schindler, P., Zamyatin, A., Weippl, E., 2018. Agreement with Satoshi - On the Formalization of Nakamoto Consensus.
Szalachowski, P., 2018. (Short Paper) Towards More Reliable Bitcoin Timestamps, in: 2018 Crypto Valley Conference on Blockchain Technology (CVCBT). IEEE, pp. 101–104.
Ullrich, J., Stifter, N., Judmayer, A., Dabrowski, A., Weippl, E., 2018. Proof-of-Blackouts? How Proof-of-Work Cryptocurrencies Could Affect Power Grids, in: International Symposium on Research in Attacks, Intrusions, and Defenses. Springer, pp. 184–203.
Valaitis, R., Meagher-Stewart, D., Martin-Misener, R., Wong, S.T., MacDonald, M., O’Mara, L., 2018. Organizational factors influencing successful primary care and public health collaboration. BMC Health Serv. Res. 18. https://doi.org/10.1186/s12913-018-3194-7
Veronese, G.S., Correia, M., Bessani, A.N., Lung, L.C., Verissimo, P., 2013. Efficient byzantine fault-tolerance, in: IEEE Transactions on Computers. IEEE, pp. 16–30.
Vukolić, M., 2015. The quest for scalable blockchain fabric: Proof-of-work vs. BFT replication, in: International Workshop on Open Problems in Network Security. Springer, pp. 112–125.
Vukolić, M., 2010. The Byzantine empire in the intercloud. ACM Sigact News 41, 105–111.
Wang, Z., Lin, J., Cai, Q., Wang, Q., Zha, D., Jing, J., 2020. Blockchain-based Certificate Transparency and Revocation Transparency. IEEE Trans. Dependable Secure Comput. 1–1. https://doi.org/10.1109/TDSC.2020.2983022
Wensley, J.H., Lamport, L., Goldberg, J., Green, M.W., Levitt, K.N., Melliar-Smith, P.M., Shostak, R.E., Weinstock, C.B., 1978. SIFT: Design and analysis of a fault-tolerant computer for aircraft control. Proc. IEEE 66, 1240–1255. https://doi.org/10.1109/PROC.1978.11114
Wüst, K., Gervais, A., 2018. Do you Need a Blockchain?, in: Crypto Valley Conference on Blockchain Technology, CVCBT 2018, Zug, Switzerland, June 20-22, 2018. IEEE, pp. 45–54. https://doi.org/10.1109/CVCBT.2018.00011
Xu, X., Weber, I., Staples, M., Zhu, L., Bosch, J., Bass, L., Pautasso, C., Rimba, P., 2017. A Taxonomy of Blockchain-Based Systems for Architecture Design, in: Software Architecture (ICSA), 2017 IEEE International Conference On. IEEE, pp. 243–252.
Zhang, R., Preneel, B., 2019. Lay down the common metrics: Evaluating proof-of-work consensus protocols’ security, in: 2019 IEEE Symposium on Security and Privacy (SP). IEEE.

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 826078.

10 Table of acronyms and definitions

API	Application programming interface
BFT	Byzantine Fault Tolerance
concentris	concentris research management GmbH
DDoS	Distributed denial of service
DLT	Distributed Ledger Technology
DoA	Description of Action
DPoS	Delegated proof of stake
EHR	Electronic Health Records
GND	Gnome Design SRL
ML	Machine Learning
MRI	Magnetic resonance imaging
MS	Milestone
MUG	Medizinische Universitaet Graz
NC	Nakamoto Consensus
Patients	In this deliverable, we use the term “patients” for all research subjects. In FeatureCloud, we will focus on patients, as this is already the most vulnerable case scenario and this is where most primary data is available to us. Admittedly, some research subjects participate in clinical trials but not as patients but as healthy individuals, usually on a voluntary basis and are therefore not dependent on the physicians who care for them. Thus to increase readability, we simply refer to them as “patients”.
PKI	Public Key Infrastructure
PoS	Proof of stake
PoW	Proof of Work
RI	Research Institute AG & Co. KG
SBA	SBA Research Gemeinnutzige GmbH
SDU	Syddansk Universitet
SNP	Single-nucleotide polymorphism
TEE	Trusted Execution Environment
TSA	Time Stamping Authority

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 826078.

TTP	Trusted Third Party
TUM	Technische Universitaet Muenchen
UM	Universiteit Maastricht
UMR	Philipps Universitaet Marburg
WP	Work package

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 826078.