# Enhancing Data Analysis with Fuzzy c-means Clustering Integration on the Blockchain Using Smart Contracts

**Mirtill Boglárka Naghi[1,2], Bence Tureczki[2], Katalin Szenes[3]**

[1]Sapientia Hungarian University of Transylvania, Calea Sighișoarei nr. 2., Târgu-Mureş, Romania, naghi.mirtill@ms.sapientia.ro

[2]Doctoral School of Applied Mathematics and Applied Informatics, John von Neumann Faculty of Informatics, Óbuda University, Bécsi road 96/B, 1034, Budapest, Hungary, naghim@stud.uni-obuda.hu, tureczki.bence@uni-obuda.hu

[3]Institute for Cyber-Physical Systems, John von Neumann Faculty of Informatics, Óbuda University, Bécsi road 96/B, 1034 Budapest, Hungary, szenes.katalin@uni-obuda.hu

*Abstract: This paper introduces a novel approach to fortify data security through the seamless integration of fuzzy clustering techniques within blockchain technology. Fuzzy clustering, known for its ability to handle uncertainties and complexities in data, synergizes with blockchain's decentralized and immutable ledger to establish a robust framework for secure data storage, analysis and retrieval. The proposed fusion not only enhances confidentiality, integrity and effectivity but also offers adaptability to the evolving dynamics of modern data landscapes. In this paper we propose a theoretical model that implements the integration of fuzzy c-means clustering on the blockchain using a cryptographically verifiable distributed computing system. By leveraging the decentralized nature of blockchain, the proposed framework ensures that data analysis processes are verifiable and tamper-resistant. Furthermore, the integration of fuzzy clustering within the blockchain not only bolsters security but also introduces a layer of transparency in the confidential data handling process.*

*Keywords: data security; blockchain technology; fuzzy clustering; smart contracts; distributed computing*

## 1 Introduction

In recent years, the convergence of cutting-edge technologies has led to advances in many areas, including data security and healthcare. The intersection of fuzzy clustering techniques and blockchain technology is a promising way to address the growing challenges of secure data management. This paper explores the synergies between fuzzy clustering and blockchain and presents a new approach that exploits the inherent strengths of both methodologies. By combining the robust data organization capabilities of fuzzy clustering with the decentralized and tamper-resistant characteristics of blockchain, our research aims to redefine the field of secure data storage and retrieval. This integration not only enhances the privacy and accuracy of information, but also lays the foundation for the development of flexible systems that can adapt to the dynamic nature of contemporary data environments. By exploring the theoretical framework and practical implications in depth, this paper contributes to the evolving discourse on the integration of fuzzy clustering within blockchain technology and its potential to revolutionize secure data management in a variety of domains.

Analyzing medical data through clustering provides valuable insights into patient profiles, particularly in scenarios such as identifying groups with shared characteristics for personalized treatment plans, delineating subtypes within specific medical conditions, detecting anomalies or outliers in test results, and segmenting medical images into meaningful regions [1, 2].

Classifying patients when they exhibit characteristics of multiple conditions concurrently poses a formidable challenge, and it is in this complex scenario that fuzzy clustering excels. Fuzzy clustering is adept at handling inherent uncertainty in cluster assignments, accommodating data points that may belong to multiple clusters with varying degrees of membership. The selection of the most appropriate clustering algorithm, however, hinges on the unique characteristics of the medical data, the objectives of the analysis, and the specific challenges inherent in the dataset under consideration.

Various variants of fuzzy c-means clustering are available and can be interchanged for data analysis. The objective of this research is to establish a system that effectively harnesses the benefits of fuzzy clustering while integrating the data integrity features of blockchain technology.

## 2 Background

### 2.1 Fuzzy c-means clustering

Fuzzy clustering is a type of clustering method used in data analysis and machine learning. Unlike traditional (hard) clustering, where each data point belongs to only one cluster, fuzzy clustering allows a data point to belong to multiple clusters to varying degrees. This is achieved through assigning membership values to data points, indicating the likelihood or degree of belonging to each cluster. Fuzzy clustering algorithms aim to optimize an objective function that considers both the minimization of intra-cluster distances and the maximization of inter-cluster distances, taking into account the membership values. Fuzzy clustering finds applications in various domains, including pattern recognition, image segmentation, and data analysis, where data points may exhibit degrees of ambiguity or uncertainty regarding their cluster assignments.

Fuzzy $c$-means (FCM) [3] is a well-known fuzzy clustering algorithm. It was introduced by James C. Dunn and later generalized by James C. Bezdek [4]. It assigns membership values (denoted by $\mathbf{u}_{ik}$ $(i = 1 \dots c, k = 1 \dots n)$) to $n$ data points (denoted by $X = \{\mathbf{x}_1, \mathbf{x}_2, \dots \mathbf{x}_n\}$) for each cluster $(c)$ and iteratively updates cluster centroids (denoted by $\mathbf{v}_i$ $(i = 1 \dots c)$) based on these memberships. In fuzzy clustering, a centroid represents the center of a cluster with a certain degree of fuzziness, reflecting the weighted contributions of data points based on their membership values. It uses a single parameter, $m > 1$ which is called the fuzzy exponent.

The FCM minimalizes the following function:

$$J_{\mathrm{FCM}} = \sum_{i=1}^{c} \sum_{k=1}^{n} u_{ik}^{m} \, ||\mathbf{x}_k - \mathbf{v}_i||^2 = \sum_{i=1}^{c} \sum_{k=1}^{n} u_{ik}^{m} \, d_{ik}^{2} \ ,$$

constrained by the probabilistic conditions

$$\sum_{i=1}^{c} u_{ik} = 1 \quad \text{and} \quad u_{ik} \in [0,1] \quad \forall k = 1 \dots n \ .$$

The iterative optimization concludes when the cluster prototypes stabilize. The formulas utilized in this process are derived from the zero gradient conditions of $J_{\mathrm{FCM}}$:

$$u_{ik} = \frac{d_{ik}^{-2/(m-1)}}{\sum_{j=1}^{c} d_{jk}^{-2/(m-1)}} \qquad \begin{matrix} \forall i = 1 \dots c \\ \forall k = 1 \dots n \end{matrix},$$

$$\mathbf{v}_i = \frac{\sum_{k=1}^{n} u_{ik}^{m}\, \mathbf{x}_k}{\sum_{k=1}^{n} u_{ik}^{m}} \qquad \forall i = 1 \dots c \ .$$

For a precise partition, set the parameter $m$ to approach $1_+$. In such instances, the membership values will exclusively be 0s or 1s.

It's important to note that fuzzy $c$-means, while effective in certain scenarios, has limitations, such as sensitivity to noise and outliers [5].

## 2.2 Blockchains

### 2.2.1 A short primer on blockchains

A blockchain is a decentralized and distributed digital ledger that records transactions across a network of computers. It consists of a chain of blocks, where each block containing transaction records, that are chained together [6]. Each block contains a multitude of well structured records. The initial root or first block of the blockchain is called the "Genesis block". The Genesis block of a blockchain is created manually by the creators of the blockchain, and is usually inscripted into the blockchain software itself. It serves as the starting point of the blockchain's chain of trust. The second block of the blockchain is attached to the Genesis block of the blockchain. Each subsequent block is connected to the previous block of the blockchain. As shown on Figure 1, a network of computers, known as miners, compete with each other to find the next block at a given time. In case multiple miners find valid candidates for the next block, there may be multiple competing blocks for the next true block, but in the end there will always only be one block that becomes the next true block. There can only be one true blockchain. In this case, the longest branch is considered as the true blockchain at any given point in time. Therefore, the blockchain itself is the longest branch of the tree of blocks.

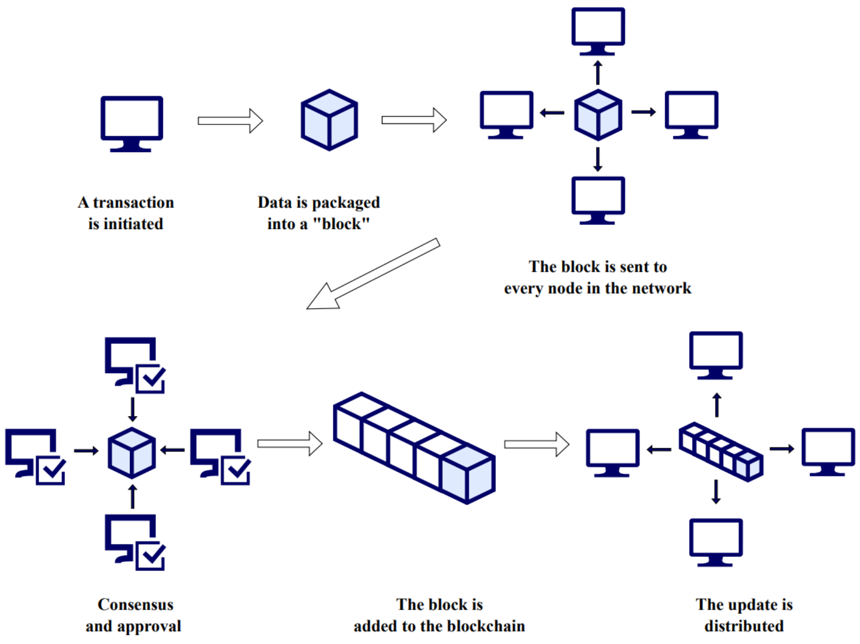

Figure 1
Structure of a distributed clustering network

Because blockchains are essentially Merkle trees (the chaining mechanism is implemented using these trees), they inherit the properties of Merkle trees as well. In other words, the blockchain can be envisioned as a large Merkle tree. The initial root of the Merkle tree, also known as the Merkle root, is called the “Genesis block” of the blockchain. Merkle trees are a fundamental component of blockchain technology, providing several important qualities contributing to the integrity and security of blockchain systems. Not only do they guarantee the data integrity within each block, but they also guarantee a form of tamper resistance. Merkle trees enable the creation of a hash chain, where each block’s hash is dependent on the previous block’s hash. Since a block’s hash changes everytime the block itself is changed, this creates a tamper-resistant data structure. Altering any block would require changing all subsequent blocks, making this form of attack computationally infeasible. The longer a block is part of a blockchain system, the harder it is for a potential attacker to modify that block of data. For example, modifying a block of data that has been created three blocks ago would require an attacker to create a sidechain that differs by at least four entire blocks, a gargantuan task by all means - considering how difficult it is to find a single block in the first place. Such dramatic changes would also be easily noticed by blockchain participants.

Because the hash of a block inside a Merkle tree always depends on the hash of the previous block, validating a subtree becomes extremely efficient computationally.

The very existence of a block guarantees that the previous blocks have already been validated. To validate a subtree of the blockchain is to validate only the blocks that are a descendant of the subtree's root. Adding new records to the blockchain does not necessitate the revalidation of the entire blockchain. The only time a blockchain participant has to revalidate the entire blockchain is when the participant first joins the network and synchronizes their version of the blockchain with the other participants.

Blockchains may be categorized by the level of access certain actors possess to the blockchain system. We may separate the nature of access into read access and write access. Participants with read access may read the contents of the blockchain, while participants with write access may also participate in the addition of new records to the next block within the blockchain.

In terms of access, there exist three types of blockchains: public, consortium and private blockchains. . Public blockchains are blockchains that are available for reading and writing for all miners. Participation is not limited, anyone can spin up a worker node and become part of the blockchain's consensus network. In comparison, consortium blockchains are only writable by a collection of participants that are previously vetted. Participation is controlled and typically limited to a specific group of known entities. These entities may include organizations, businesses or individuals with permission to join and interact with the blockchain. Consortium blockchains can differ in read permissions: some consortium blockchains allow public access to their data, while some restrict read access to a specific group of vetted entities. In private blockchains, participation is limited to members of a single organization. Public blockchains are completely decentralized, and are nearly impossible to tamper due to the sheer number of participants, while consortium are partially centralized and private blockchains are completely centralized and more easily tampered with. In addition, the consensus protocol of public blockchains is permissionless, while consortium and private blockchains are permissioned.

In the realm of public blockchains, a prominent example is Bitcoin, where anyone can participate as a miner and contribute to the decentralized consensus process. Ethereum is another noteworthy public blockchain that extends beyond simple transactions by incorporating smart contracts, enabling a broader range of decentralized applications. Moving to consortium blockchains, Hyperledger Fabric serves as an illustrative example. It is a permissioned blockchain framework that caters to business solutions, allowing vetted participants to engage in a shared ledger with controlled access. On the private blockchain front, Corda stands out as a distributed ledger platform designed for use within financial institutions, ensuring that participation is confined to authorized entities within a specific organization. These real-world examples showcase the diverse applications and governance structures across public, consortium, and private blockchains. It's important to note that some distributed ledger solutions can be used with multiple permission models. For example, Ethereum is very versatile in this regard - it can be used as both a public blockchain and a private blockchain, as it supports not only the usual Proof-of-Stake algorithm, but also provides a Proof-of-Authority model, allowing only approved signers to create new blocks [7].

#### 2.2.2 Consensus algorithms

Consensus algorithms lie at the heart of distributed systems, enabling a network of participants to reach an agreement on the validity and order of transactions without relying on a central authority. These algorithms play a pivotal role in technologies like blockchain, ensuring that all nodes within the network share a consistent and accurate record of events. By facilitating agreement in decentralized environments, consensus algorithms contribute to the security, reliability, and trustworthiness of distributed systems.

In a decentralized system, such as the blockchain, miners play a key role in reaching consensus. Miners are participants in the network who propose new blocks of transactions to be added to the blockchain. In the case of Proof of Work chains, they use computational power to solve complex mathematical problems. Once a miner successfully solves a problem, they gain the ability to propose a new block. Other miners then verify the validity of this block, and if they agree, the new block is added to the existing chain. This agreement among miners on the order of blocks is crucial for maintaining the integrity of the blockchain. It ensures that all participants in the network have a consistent and agreed-upon record of transactions. Attempts to manipulate or tamper with the data in a block become extremely challenging because it would require the consensus of the majority of the network, making the blockchain resistant to fraud and unauthorized alterations.

Several consensus mechanisms are used in different blockchain networks, each with its own advantages and trade-offs. Two prominent consensus algorithms are Proof of Work (PoW) and Proof of Stake (PoS).

In PoW systems, miners compete to solve complex mathematical problems, and the first one to solve it gets the right to add a new block to the blockchain. PoW is known for its robust security due to the computational work required, making it difficult and resource-intensive to attack the network. For example, Bitcoin uses PoW as its consensus algorithm.

In PoS systems, validators are chosen to create new blocks based on the amount of cryptocurrency they hold and are willing to “stake” as collateral. PoS is considered more energy-efficient than PoW since it doesn’t require the same level of computational power. Ethereum has transitioned from PoW to PoS with its Ethereum 2.0 upgrade.

Other consensus algorithms include Delegated Proof of Stake (DPoS) [8], Practical Byzantine Fault Tolerance (PBFT) [9], and Raft [10], each tailored to specific use cases and requirements. The choice of consensus algorithm has a significant impact on a blockchain’s performance, scalability, and resilience to attacks.

### 2.2.3 Evolution of blockchain

The evolution of blockchain has gone through several steps to reach the present-day technologies. The history of blockchain is closely linked to cryptocurrencies. Therefore, while discussing the evolution of blockchain, it is important to mention the major milestones in the development of cryptocurrencies. Its history dates back to the ’90s, when Stuart Haber and W. Scott Stornetta [11] introduced the concept of a cryptographically secured chain of blocks as a way to timestamp digital documents to prevent backdating or tampering. Later they upgraded their system to incorporate Merkle trees, which increase efficiency by allowing multiple documents to be collected in a single block. Nonetheless, their work laid the theoretical foundation for the technology and the world’s very first publicly available blockchain was born from it.

The New York Times served as the host for this blockchain [12]. Companies or individuals would use a proprietary tool to create a hash of a digital document, which would naturally be invalidated if the digital document was ever modified. These hashes would be sent to a server where the hash would be timestamped in order to create a digital seal. This digital seal would then be sent back to the customer, and the seal would be appended next to the digital document.

The server would keep an internal log of all timestamped digital document seals. Each week, it would collect all digital seals created in the last 7 days and create a block of data for that given week. A hash value would be calculated for this block, but not before prepending the previous hash value to the block. This construction practically implements a hash chain - a Merkle tree where each hash value depends on the previously calculated cumulative hash value. The current hash value of the chain would then be published in the New York Times in the “Lost and Found” section, beginning in 1995, creating an immutable record of all seals ever produced.

A few years later, Nick Szabó proposed “Bit Gold”, a decentralized but theoretical cryptocurrency. Although “Bit Gold” was never implemented, his work introduced many concepts that are now associated with blockchain and cryptocurrencies [13]. He is also credited for the creation of smart contracts, which play an important role in Ethereum and other contract based cryptocurrencies. In 1994, he introduced the concept, and by 1996, he delved into an examination of the potential capabilities of smart contracts [14, 15]. Nick Szabó’s visionary contributions laid the groundwork for the,  evolution of blockchain technology, influencing the development of subsequent cryptocurrencies and inspiring advancements in decentralized systems, including blockchains.

At the beginning of the millennium, Stefan Konst published a paper on cryptographically secure chains with practical implementation [16].

The crucial moment for blockchain technology occurred with the release of the white paper titled “Bitcoin: A Peer-to-Peer Electronic Cash System” at the end of 2008 [17]. This document, authored under the pseudonym “Satoshi Nakamoto” (the author’s true identity remains unknown till today), outlines the concept of a decentralized monetary system and is highly influential in the development of blockchain technology. Although the idea of decentralized money existed for a decade ("B-money" [18], "Bit Gold" [19] etc.), this event marked the birth of blockchain technology in its practical implementation. A year later, in January the first block ("genesis block") of the Bitcoin blockchain was "mined" i.e. created. It's important to note that Bitcoin is not only a blockchain but also a digital form of payment that operates independently of traditional banks.

In the years following the release of the Bitcoin white paper and the creation of its genesis block, the adoption and evolution of blockchain technology have been remarkable, jumpstarting the evolution and expansion of blockchain technology - beyond its initial application in cryptocurrencies like Bitcoin. The decentralized nature of blockchain, which relies on a distributed ledger to record transactions across a network of computers, has spurred the development of not only various cryptocurrencies, but also decentralized applications (called DApps). The introduction of Ethereum in 2013 by Vitalik Buterin is one such important milestone, expanding the capabilities of the blockchain by introducing the concept of smart contracts on the Ethereum chain, enabling programmable and self-executing agreements using a built-in programming language directly on the blockchain. Ethereum went live in 2015, enabling developers to build decentralized applications (DApps) on its blockchain [20]. This innovation further fueled the growth of the decentralized finance (DeFi) ecosystem, allowing for peer-to-peer lending, decentralized exchanges, and other financial services without the need for traditional intermediaries. As the technology continues to mature, blockchain is finding applications beyond finance, including supply chain management, healthcare, and governance, showcasing its potential to revolutionize various industries.

The development of blockchain for use in a broader range of applications beyond simple peer-to-peer transactions was explored by many in the literature: applications in intellectual property protection [21], food traceability [22], healthcare data management [23], supply chain and logistics [24], dynamic support of kinematic testing [25], automating corporate tasks [26], registering students' attendance in academic settings [27] etc.

## 2.3 Ethereum and the Ethereum Virtual Machine

Ethereum is a decentralized blockchain platform that enables the creation and execution of smart contracts and decentralized applications (DApps). It operates on a cryptocurrency called Ether (ETH).

Side chains are separate blockchains that can connect to the main Ethereum blockchain, allowing for scalability and specific use cases. Some examples of Ethereum side chains include: Polygon (formerly Matic, a side chain designed to improve scalability and transaction speed on the Ethereum network), xDai (a stable chain connected to the Ethereum mainnet), providing faster and cheaper transactions, and Optimistic Ethereum (implements optimistic rollups to enhance scalability and reduce transaction fees). Ethereum’s versatility is highlighted by projects like Polygon. Polygon, being a Layer 2 scaling solution, aims to address Ethereum’s scalability issues by offering faster and more cost-effective transactions. Through its side chain architecture, Polygon achieves this by processing transactions off the main Ethereum chain, subsequently bundling them into checkpoints that are periodically anchored on the Ethereum mainnet. This approach enhances scalability without compromising on the security and decentralization inherent in Ethereum’s primary chain. At the time of writing, the gas cost of decentralized applications deployed on the Solidity chain is magnitudes lower than that of applications deployed on the Ethereum mainnet (few cents compared to tens of dollars).

The Ethereum Virtual Machine (EVM) is a runtime environment for executing smart contracts on the Ethereum network. It allows developers to write code in high-level programming languages like Solidity, which is then compiled into bytecode that the EVM can execute. The EVM ensures the consistency and security of smart contract execution across the decentralized network. One of the key features of the EVM is its isolated execution environment, ensuring that each contract operates independently. This isolation safeguards the network and other contracts from potential vulnerabilities or malicious activities in any single contract. Security is further reinforced by the EVM’s ability to manage and allocate computational resources through a mechanism known as “gas”. This system not only helps in resource management but also prevents issues like infinite loops by automatically terminating transactions that run out of gas.

Solidity is one such language used to implement smart contracts on the EVM. It is a high-level, contract-oriented programming language used primarily for developing smart contracts on the EVM. One of the key strengths of Solidity is its support for inheritance in contracts and the use of libraries, which promotes code reusability and modularity. This feature allows developers to create complex, hierarchical contract structures and to easily integrate common functionalities through libraries. Solidity is specifically designed to compile into bytecode compatible with the Ethereum Virtual Machine (EVM), ensuring seamless integration and execution of contracts on the Ethereum blockchain.

To cater to the unique needs of blockchain-based applications, Solidity includes advanced features like function modifiers for access control and events for interfacing with a decentralized application’s user interface. Safety is a paramount concern in smart contract development, and Solidity addresses this need with built-in checks for common vulnerabilities like overflow and underflow. Additionally, it supports state variables for persistently storing contract data and various control structures, such as loops and conditional statements, necessary for executing complex logic.

A distinctive feature of Solidity is the concept of “payable” functions, which allows contracts to receive and manage Ether, the native cryptocurrency of Ethereum. This capability is fundamental to the creation of financial applications and transactional functionalities within smart contracts.

## 2.4 Data within the blockchain

Using blockchain technology to encrypt medical data involves leveraging its inherent characteristics of decentralization, immutability, and transparency. While blockchain itself doesn't provide encryption in the traditional sense, it offers a secure and transparent way to manage access control and maintain the integrity of medical records. Leveraging blockchain technology offers various features that can be advantageous for the storage of medical data:

1. *Decentralized data distribution:* Instead of storing medical data in a centralized database, blockchain distributes copies of the data across multiple nodes in a network. This reduces the risk of a single point of failure or unauthorized access.
2. *The immutable ledger ensures data integrity:* Blockchain maintains an immutable ledger, meaning once data is added to the chain, it cannot be altered or deleted. This ensures the integrity of medical records, preventing unauthorized modifications.
3. *Guarantee the authenticity of data:* Blockchain uses cryptographic hash functions to secure data. Medical records can be hashed and stored on the blockchain. Even a small change in the original data would result in a completely different hash, alerting the network to potential tampering.
4. *Access control:* Smart contracts on the blockchain can define access controls for medical data. Only authorized individuals or entities with the correct cryptographic keys can access specific records. This helps in maintaining privacy and security.
5. *Secure transactions by using a consensus mechanism:* The consensus mechanism in blockchain ensures that transactions (in this case, clustering the medical records) are validated by the network. This adds an additional layer of security by requiring majority agreement before a new block is added to the chain.

While the core blockchain technology itself does not perform encryption, we can use traditional encryption methods to encrypt sensitive data before storing it on the blockchain. This way, only authorized parties with the decryption keys can access the actual medical information.

# 3 Methods

Why would it become apparent that encryption is necessary for the storage of medical data? The imperative for encryption in this context arises from the critical need to safeguard sensitive healthcare information. Various legislative requirements exist in this regard, like HIPAA. The encryption requirements outlined in the

HIPAA Security Rule mandate the establishment of a system for encrypting and decrypting electronic Protected Health Information (ePHI). This mechanism should ensure that access is restricted solely to authorized individuals or software programs with designated access rights (45 CFR § 164.312)[1].

Such a system must be implemented with rigorous security measures in mind to shield patient records from unauthorized access. We hereby propose the creation of a technique to enable the secure and verifiable analysis of medical data, using blockchain-based smart contracts acting upon the encrypted medical data of patients.

## 3.1 Verifiable, secure clustering on the blockchain

We propose the creation of a distributed computing model to implement the secure analysis of medical data, as seen on Figure 2. In this system, the participants of the blockchain work together to analyze medical data made available within the blockchain. The analyzed results are then made available through the blockchain, with the results being traceable back to the participant responsible for providing the computational resources. Multiple participants each analyze the data separately. The results are then tallied, and the final result is determined through a consensus protocol.

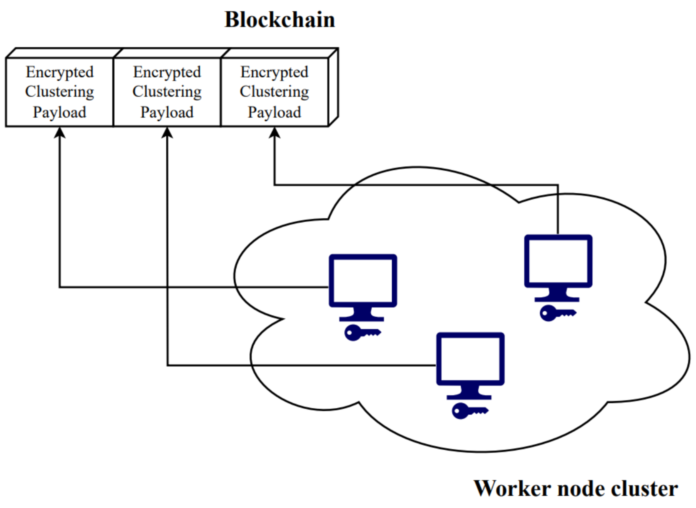
Blockchain
Encrypted Clustering Payload
Encrypted Clustering Payload
Encrypted Clustering Payload
Worker node cluster

Figure 2
Structure of a distributed clustering network

1 https://www.ecfr.gov/current/title-45/subtitle-A/subchapter-C/part-164/subpart-C/section 164.312

There are two types of actors in this system:

1. **The client**: The person initiating the analysis of the medical data. The client is responsible for initiating the actions that lead to the medical data being loaded into the blockchain. In addition to this, the client also subscribes to the smart contract responsible for the analysis of the data, awaiting the final result of the transaction.

2. **The clustering worker nodes**: The trusted blockchain network participants responsible for the actual analysis of the data. They possess the master key necessary to decrypt the sensitive medical data, as well as the necessary computational resources to handle the analysis of data on-site, without transporting the data through the network in any shape or form. Their job is to run the clustering algorithm on the medical data, and provide the results back to the blockchain. They exist within a worker node cluster.

The clustering worker nodes are subscribed, using their address, to a smart contract providing a queue for potential medical data to be analyzed. Each worker node has its own private key: $PrivSignKey$, as well as the master root key: $PrivRootKey$. Let's assume that a client wishes to analyze the medical data of a patient. After initiating any action that results in the patient's data being submitted to the blockchain, a typical clustering worker node completes the following stages:

1. **The waiting stage**: The worker node is waiting for potential medical data to be analyzed. It is subscribed to the smart contract providing the queue on the blockchain.

2. **The downloading stage**: The smart contract has notified the worker node that some medical data has to be analysed. The worker node proceeds to download the encrypted medical data. This medical data is either available on-chain (data embedded into the smart contract), or off-chain (data saved onto an external server, as a lightweight entry on-chain containing an authenticated and temporary (time-limited) URL to the medical data), as well as a cryptographic hash providing data integrity. This cryptographic hash is checked, to make sure that the data available at the external URL matches the initial medical data uploaded by the client.

3. **The decryption stage**: Potential medical data has been found. The worker node commences with the decryption of the medical data using the master root key: $PrivRootKey$. This master root key is shared between all clustering worker nodes. To derive the symmetrical decryption key, also known as the session key, $PrivRootKey$ is used to asymmetrically decrypt the $EncSessionKey$ found alongside the medical data. After $SessionKey$ is derived, the medical data is symmetrically decrypted with it. After this stage is complete, the worker node has temporary access to the medical data until it is evicted from memory following analysis.

4. **The analysis stage**: The decrypted medical data is analyzed using a fuzzy clustering algorithm. It is imperative that the fuzzy clustering algorithm used is deterministic. As such, the initial cluster prototypes must be calculated using a deterministic algorithm - the usual random approach is not sufficient. We have found that using the body diagonal of the clustering space yields a satisfactory clustering result. The artifacts of this stage are the clustering labels: the results of the analysis.
5. **The signing stage**: The artifacts of the analysis stage are signed with the working node specific signing key: $PrivSignKey$. This ensures that each analysis result can be traced back to the worker node that was responsible for creating it, using the respective public key. Each worker node has its own private signing key. This signing key is not shared with any other worker node. This signature is embedded into the result, alongside the analysis results.
6. **The submission stage**: The clustering result is submitted into the smart contract. The analysis results are embedded alongside the signature created by the worker, the address associated with the worker, the medical data identifier, as well as the current Unix timestamp. The analysis results may be embedded externally, off-chain as a simple URL in order to minimize the financial costs of the analysis: in this case, a hash of the medical data is also stored alongside the URL to ensure data integrity.

On the other hand, the client is tasked only with performing the necessary actions to load the encrypted medical data into the smart contract on the blockchain, and to subscribe to events on the smart contract to become aware when the analysis of the data has commenced. As such, the client goes through the following stages:

1. **The session key generation stage**: The client generates a cryptographically random secret session key called $SessionKey$. The client takes the public portion of the master root key: $PubRootKey$, which is known publicly, and asymmetrically encrypts the $SessionKey$ into the $EncSessionKey$. This ensures that only the worker nodes can decrypt the confidential medical data.
2. **The encryption stage**: The client initiates the necessary steps to prepare the medical data for encryption. The client uses the $SessionKey$ to symmetrically encrypt the plaintext medical data.
3. **The submission stage**: The client fills out the necessary data structure encoded in the smart contract: submitting the symmetrically encrypted medical data, the asymmetrically encrypted session key, as well as a uniquely generated identifier for the medical data to be analyzed.
4. **The waiting stage**: The client subscribes to the relevant events on the smart contract, waiting for the consensus between the worker nodes to complete. The client waits for at least $N$ worker nodes to complete the analysis process. After $N$ worker nodes have completed the analysis

process, each providing the same hash for the analysis results, the analysis is considered complete, having reached a state of consensus. Optionally, a timeout can be set in case the worker nodes fail to reach consensus within a reasonable amount of time.

5. **The downloading stage**: The client downloads the clustering labels from either on-chain or off-chain, depending on how the clustering labels were stored. If the clustering labels are stored off-chain, the client also verifies that the expected hash of the clustering labels matches the actual, calculated hash of the downloaded clustering labels. If not, the client returns to the waiting stage.

Waiting for consensus in the described protocol is a crucial step to ensure the integrity and correctness of the analysis results before proceeding to the downloading stage. In this context, achieving consensus is essential to ensure that the participating nodes reach an agreement on the validity of the analysis results. Medical data analysis is a sensitive process. Waiting for a consensus among a predefined number of worker nodes ensures that the results are accurate and reliable. If there are discrepancies or malicious actions by some nodes, waiting for consensus helps identify and address such issues in a transparent fashion.

By requiring a certain number of worker nodes to agree on the analysis results, the protocol enhances security. It reduces the risk of a single compromised or malicious node influencing the final outcome. This is particularly important in scenarios involving sensitive medical data, where data integrity and confidentiality are paramount. The consensus mechanism promotes decentralization, a key feature provided by the blockchain background. It prevents a single entity or a small group of nodes from controlling or manipulating the analysis results. Decentralization increases the overall trustworthiness of the system.

Waiting for consensus also introduces a level of fault tolerance. If some worker nodes fail to provide the correct analysis results or go offline, the consensus mechanism ensures that the process can continue as long as the required number of nodes agree. This is essential for maintaining the functionality of the system in the presence of node failures. It also acts as a deterrent against tampering with the analysis results. If a malicious node attempts to manipulate the results, it would need to compromise a significant number of nodes to achieve consensus, making such attacks more difficult and less likely.

Transparency is ensured in the analysis process through the consensus protocol. Allowing multiple nodes to independently validate and agree on the results makes the process auditable. This transparency is crucial, especially in applications like medical data analysis where accountability and traceability are vital.

## 3.2 Recommendations for practical applications

In this section, we provide practical recommendations for implementing the proposed distributed computing model using Ethereum, with a focus on scalability and security. We'll explore the use of Polygon as a side chain to enhance transaction throughput, discuss suitable encryption algorithms, and suggest the use of Solidity for smart contract development, including the implementation of events for efficient communication.

Ethereum's robust and decentralized nature makes it a suitable choice for hosting sensitive medical data. The facilities Ethereum provides, in the form of smart contracts, make it an excellent choice to implement a verifiable and traceable clustering network. While the Ethereum mainnet provides a secure and immutable ledger for storing critical information, we would like to recommend the usage of Ethereum sidechains rather than directly integrating the system into the Ethereum mainnet.

To address Ethereum's scalability challenges, we recommend integrating Polygon as a side chain. Polygon offers faster transaction speeds and lower costs, making it an ideal solution for handling the high volume of transactions involved in medical data analysis. Polygon is designed to be interoperable with Ethereum. This means that assets and data can be transferred seamlessly between the Ethereum mainnet and Polygon side chain, should a migration to another blockchain be required.

For securing the session key exchange, we recommend using RSA (Rivest-Shamir-Adleman) with Optimal Asymmetric Encryption Padding (OAEP). OAEP enhances the security of RSA by introducing randomness during encryption, mitigating certain vulnerabilities associated with pure RSA encryption. ECDH may also be used to implement the session key exchange. ECDH is a variant of Diffie-Hellman that uses elliptic curve cryptography. It provides similar functionality to the traditional Diffie-Hellman key exchange but with smaller key sizes, making it computationally more efficient while maintaining a high level of security.

To implement the symmetric encryption of the sensitive medical data, we recommend the usage of AES. AES is a widely accepted and secure symmetric encryption algorithm. Alternatively, ChaCha20, a stream cipher, could be considered for its efficiency. Implementing ChaCha20 in software, on worker nodes that do not support AES decoding instructions in hardware, might also be more favorable given ChaCha20's affinity for parallel processing [28].

We recommend that the actual encryption algorithms be advertised within the smart contract, so that the system remains flexible and open for extension. Advertising the encryption algorithm types as part of the smart contracts allows new encryption protocols to be implemented at a later date. For example, if the actual implementation is using OpenSSL, then the NID of the encryption algorithms could be made part of the smart contract.

Implementing the proposed model involves the creation of smart contracts on an Ethereum-compatible blockchain to facilitate communication and consensus among participants. Solidity, Ethereum's native programming language, is recommended for smart contract development. Solidity's event system can be leveraged to streamline communication between the client and the clustering worker nodes. Events can be emitted when important milestones are met during a process. For example, when the client submits encrypted medical data, an event could trigger the nodes to begin working on the analysis process. Similarly, events could be emitted to signal the completion of the analysis by a worker node. By using events strategically, the smart contract ensures that the protocol progresses smoothly, and each participant in the network is aware of the state and progress of the process.

# 4 Background

## 4.1 About the security of the system

Excellence criteria for a blockchain-based medical data clustering system should encompass various aspects to ensure the effectivity, confidentiality, integrity, and reliability of the system. According to the literature [29], these excellence criteria may be described in the following manner:

- *Effectiveness* in an operational activity is achieved when its outcomes align with pre-established requirements that have been accepted by all relevant parties. The proposed system ensures that all parties speak the same protocol, agree on the same concepts and provide the same calculations.
- *Confidentiality* involves safeguarding sensitive information to prevent unauthorized disclosure. All sensitive medical data is sealed and is only accessible to authorized subprocessors operating on the data.
- *Integrity* is connected to the precision and entirety of information, as well as its alignment with values and expectations. Since the data subprocsssors have to agree on the clustering result, the processing of the medical data becomes an automatically peer-reviewed process.
- *Reliability* pertains to the consistent and accurate performance of the system, ensuring it operates dependably and delivers results in accordance with established standards over time. Even if certain subprocessors fail to procure an answer in a timely fashion (or at all), the system remains functional as the rest of the nodes begin to fill in the roles of the absent workers.

The system was designed in mind to comprehensively address these criteria, emphasizing security, safety, and reliability. The following criteria enlisted collectively contribute to establishing the system as a secure, reliable, and trustworthy platform for clustering medical data on the blockchain.

### 4.1.1 Restricted access

Only designated worker nodes with explicit permissions can access confidential medical data, ensuring a secure and controlled environment for data handling. The system employs robust encryption mechanisms, encompassing both symmetrical and asymmetrical encryption, to safeguard sensitive medical information stored on the blockchain.

### 4.1.2 Immutability

The concept of immutability within the system is a crucial assurance, providing a robust safeguard against unauthorized modifications by potential attackers. This inherent characteristic plays a pivotal role in upholding the integrity of both the medical data and the derived analysis results across the entire lifecycle of the blockchain. The immutability is enforced through the inherent properties of blockchain technology, such as cryptographic hashing and decentralized consensus mechanisms. By enforcing immutability, the system ensures that once data is committed to the blockchain, it remains unaltered and tamper-resistant, instilling confidence in the reliability and consistency of the clustered information over time. This commitment to immutability not only fortifies the security posture of the system but also establishes a trustworthy foundation for the long-term storage and retrieval of critical medical insights.

### 4.1.3 Consensus safety

The concept of consensus safety within the system is a pivotal component contributing to the overall security and reliability of the clustering process. The consensus protocol, a fundamental mechanism of the blockchain, acts as a safeguard against a rare yet impactful occurrence known as cosmic bit flips.

Cosmic bit flips refer to the exceedingly improbable scenario where multiple bits within the system's computational elements, or workers in the context of this system, undergo simultaneous and identical alterations due to external factors such as cosmic radiation. The sheer magnitude of workers, denoted by $N$, significantly diminishes the likelihood of such synchronous and uniform bit flips occurring across the entire system.

The consensus safety mechanism leverages the decentralized nature of the blockchain network, where each worker independently contributes to the validation and agreement on the integrity of transactions. In the context of clustering, this means that the chance of cosmic bit flips causing simultaneous alterations to the clustering results across multiple workers is astronomically small.

By relying on the decentralized consensus protocol, the system mitigates the potential impact of cosmic bit flips, ensuring that the integrity of the clustering results remains robust and resilient against rare, external influences. This introduces an additional layer of security, reinforcing the reliability of the clustered outcomes and fortifying the system against the impact of unexpected events on the accuracy and consistency of the medical data analysis.

### 4.1.4 Verifiability

The system prioritizes verifiability as a cornerstone feature, offering a mechanism to confidently authenticate results generated through the clustering process. This emphasis on verifiability serves to enhance transparency and foster trust within the system. By facilitating the scrutiny of outcomes, this feature ensures alignment between the final analysis results, the original dataset, and the methodologies employed by the clustering algorithms. Stakeholders, including healthcare professionals and system users, can rely on the verifiability aspect to validate the accuracy and legitimacy of the clustered data, thereby reinforcing the credibility of the system and the insights derived from the medical data.

Verifiability is enhanced by the following aspects of the system: first of all, each worker node signs the submitted results using its assigned cryptographic key. Digital signatures verify the authenticity of the data and confirm that it originated from the legitimate worker. Verification can be performed by comparing the signature with the worker's public key. Second of all, the consensus mechanism within the blockchain involves multiple nodes reaching an agreement on the validity of transactions. This consensus ensures that the majority of nodes confirm the legitimacy of the clustered results, providing an additional layer of verification against potential manipulation. Lastly, hash functions are applied to each block of data within the blockchain. Any modification to the data, no matter how minor, would result in a completely different hash. Regular integrity checks of these hashes enable verification of the data's consistency and detect any unauthorized changes.

### 4.1.5 Forgery prevention

The system incorporates measures to prevent the submission of fake records by unauthorized or malicious workers. This helps maintain the authenticity and accuracy of the clustered medical data, safeguarding against potential fraudulent or misleading activities. This is due to the fact that each submitted result is authenticated (signed) by the worker using the key assigned to them and stored in the manifest.

### 4.1.6 Liability traceability

The emphasis on liability traceability within the system is a critical feature that enhances accountability and transparency throughout the medical data clustering process. This functionality is achieved by meticulously documenting the date of analysis, providing a clear chronological record of when clustering operations occur.

The visibility into the timing of the analysis serves as a crucial tool for attributing responsibility and tracing liability in the event of discrepancies or issues arising during the clustering process. By establishing a direct link between the actions performed and the specific instances in time, the system offers a comprehensive audit trail.

In practice, liability traceability becomes particularly valuable when investigating and addressing any anomalies, errors, or disagreements related to the clustered results. Stakeholders, including healthcare professionals, administrators, or regulatory bodies, can refer to the timestamped records to identify the precise moment when the analysis occurred. This capability not only facilitates the identification of potential sources of error but also supports a more informed and accountable resolution of any arising issues.

## 4.2 Further remarks about security

No system nor protocol is completely infallible, especially not when theoretical aspects are put into practice inside actual, practical deployments. As such, there are certain aspects that we must keep in mind: aspects that must be known, aspects that may require further investigation if they are deemed important enough to take action upon.

Regarding the clustering outcome, the analysis results of the system are available on the blockchain in a plaintext form. This is to ensure that the results are easily traceable and verifiable at a moment's glance. However, it would be feasible to enhance security by encrypting the output using either a block cipher in ECB mode or initializing a cipher using an initialization value derived deterministically from the data and employed in CBC (or GCM) mode. Another approach that could be taken is asymmetrical encryption, where the client would send its own public key as part of the data, where the worker nodes would asymmetrically encrypt the results so that only the client can decrypt the result. Encrypting the analysis results, however, would reduce the traceability of the process. If both properties of the system are deemed equally important, then further investigation must be done in order to find a way to construct a protocol that can ensure the coexistence of both properties.

The outlined system introduces a few concerns that necessitate additional research to address its shortcomings. Primarily, there is a need to explore secure methods for constructing and distributing the manifest of worker node keys among the clients. Creating such a key distribution mechanism is not within the scope of our research. However, creating a manifest might not be required in the context of a blockchain, as transactions are already signed by default. Similarly, how do the cluster nodes share the $PrivRootKey$ securely among them? There exists secure key exchange mechanisms, but a comprehensive exploration of the topic is beyond the scope of the current discussion. Continuing our exploration, an additional concern arises in establishing a robust mechanism for periodic key rotation (perhaps through the introduction of ephemeral keys), ensuring long-term security and minimizing potential vulnerabilities through unintentional key disclosure.

**Conclusions**

The analysis of medical data through clustering has demonstrated its significance in various healthcare scenarios, from personalized treatment plans and disease subtyping to anomaly detection and meaningful image segmentation. Fuzzy clustering, by accommodating uncertainty in cluster assignments and enabling membership in multiple clusters, fuzzy clustering addresses the intricate challenge of clustering (grouping) patients with overlapping characteristics.

This study has explored the intersection of fuzzy clustering techniques and blockchain technology, presenting a novel approach that capitalizes on the synergies between these methodologies. Blockchain technology ensures secure and tamper-resistant data storage, while fuzzy clustering, by accommodating uncertainty in cluster assignments and allowing membership in multiple clusters, adeptly tackles the complex task of grouping patients with overlapping characteristics.

The primary objective of this paper was to create a system that seamlessly incorporates the advantages of fuzzy clustering with the data integrity features of the blockchain, offering a theoretical foundation for the development of a secure and flexible framework for the analysis of medical data. We have successfully laid the theoretical groundwork for a system that not only meets these criteria for excellence but also allows for further flexibility and growth in the domain.

**Acknowledgement**

The work of Mirtill Boglárka Naghi was supported in part by the Collegium Talentum 2023 Programme of Hungary.

## References

[1] Wen-Liang Hung and Yen-Chang Chang. A modified fuzzy c-means algorithm for differentiation in mri of ophthalmology. In Modeling Decisions for Artificial Intelligence: Third International Conference, MDAI 2006, Tarragona, Spain, April 3-5, 2006. Proceedings 3, pages 340–350. Springer, 2006.

[2] Laszlo Szilagyi, Laszlo Lefkovits, and Balazs Benyo. Automatic brain tumor segmentation in multispectral mri volumes using a fuzzy c-means cascade algorithm. In 2015 12th international conference on fuzzy systems and knowledge discovery (FSKD), pages 285–291. IEEE, 2015.

[3] J. C. Dunn. A fuzzy relative of the isodata process and its use in detecting compact well-separated clusters. Journal of Cybernetics, 3(3):32–57, 1973.

[4] James C. Bezdek, Robert Ehrlich, and William Full. Fcm: The fuzzy c-means clustering algorithm. Computers & Geosciences, 10(2):191–203, 1984.

[5] Salar Askari. Fuzzy c-means clustering algorithm for data with unequal cluster sizes and contaminated with noise and outliers: Review and development. Expert Systems with Applications, 165:113856, 2021.

[6] Zibin Zheng, Shaoan Xie, Hongning Dai, Xiangping Chen, and Huaimin Wang. An overview of blockchain technology: Architecture, consensus, and future trends.

In 2017 IEEE international congress on big data (BigData congress), pages 557–564. Ieee, 2017.

[7] Cyril Naves Samuel, Severine Glock, Fran¸cois Verdier, and Patricia Guitton-Ouhamou. Choice of ethereum clients for private blockchain: Assessment from proof of authority perspective. In 2021 IEEE International Conference on Blockchain and Cryptocurrency (ICBC), pages 1–5. IEEE, 2021. 19

[8] Daniel Larimer. Delegated proof-of-stake (dpos). Bitshare whitepaper, 81:85, 2014.

[9] Miguel Castro, Barbara Liskov, et al. Practical byzantine fault tolerance. In OsDI, volume 99, pages 173–186, 1999.

[10] Diego Ongaro and John Ousterhout. In search of an understandable consensus algorithm (extended version). In Proceeding of USENIX annual technical conference, USENIX ATC, pages 19–20, 2014.

[11] Stuart Haber and W Scott Stornetta. How to time-stamp a digital document. Springer, 1991.

[12] Daniel Oberhaus. The world's oldest blockchain has been hiding in the new york times since 1995. https://www.vice.com/en/article/j5nzx4/ what-was-the-first-blockchain, 2018. [Online; accessed 2-December2023].

[13] Nick Szabo. Bit gold proposal. Decentralized Business Review, page 21449, 2008.

[14] Nick Szabo. Formalizing and securing relationships on public networks. First monday, 1997.

[15] Nick Szabo. Smart contracts: building blocks for digital markets. EXTROPY: The Journal of Transhumanist Thought,(16), 18(2):28, 1996.

[16] S Konst. Secure log files based on cryptographically concatenated entries. Technische Universitat Braunschweig, 2000.

[17] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. Available at SSRN 3440802, 2008. [Online; accessed 2-December-2023].

[18] Wei Dai. B-money. http://www.weidai.com/bmoney.txt, 1998.

[19] Nick Szabo. Bit gold. https://unenumerated.blogspot.com/2005/12/bit-gold.html, 2005. [Online; accessed 2-December-2023].

[20] Vitalik Buterin et al. A next-generation smart contract and decentralized application platform. white paper, 3(37):2–1, 2014.

[21] Wei Chen, Kun Zhou, Weidong Fang, Ke Wang, Fangming Bi, and Biruk Assefa. Review on blockchain technology and its application to the simple analysis of intellectual property protection. International Journal of Computational Science and Engineering, 22(4):437–444, 2020.

[22] Marina Creydt and Markus Fischer. Blockchain and more-algorithm driven food traceability. Food Control, 105:45–51, 2019.

[23] Dimiter V Dimitrov. Blockchain applications for healthcare data management. Healthcare informatics research, 25(1):51–56, 2019.

[24] Davor Dujak and Domagoj Sajter. Blockchain applications in supply chain. SMART supply network, pages 21–46, 2019. 20

[25] Bence Tureczki, Henriette Steiner-Komoróczki, and Katalin Szenes. A blockchain-based dynamic support of kinematic testing. In 2022 IEEE 22nd International Symposium on Computational Intelligence and Informatics and 8th IEEE International Conference on Recent Achievements in Mechatronics, Automation, Computer Science and Robotics (CINTIMACRo), pages 000329–000334. IEEE, 2022.

[26] Katalin Szenes and Bence Tureczki. Ai assistant in a smart cloud. In 2022 IEEE 20th Jubilee World Symposium on Applied Machine Intelligence and Informatics (SAMI), pages 000311–000316. IEEE, 2022.

[27] Krisztián Bálint. Possibilities for the utilization of an automatized, electronic blockchain-based, students' attendance register, using a universities' modern security cameras. Acta Polytechnica Hungarica, 18(2):127–142, 2021.

[28] Radu Velea, Florina Gurzău, Laurențiu Mărgărit, Ion Bica, and Victor Valeriu Patriciu. Performance of parallel chacha20 stream cipher. In 2016 IEEE 11th International Symposium on Applied Computational Intelligence and Informatics (SACI), pages 391–396, 2016.

[29] K Szenes. Enterprise governance against hacking. In 3rd IEEE International Symposium on Logistics and Industrial Informatics, pages 229–234. IEEE, 2011.