Preventing misinformation with a private blockchain: A case study on a news industry application

Diogo Filipe Pinto Nogueira
diogopnogueira@tecnico.ulisboa.pt

Instituto Superior Técnico, Lisboa, Portugal

November 2022

Abstract

The manipulation of news-related digital content has been on the rise. It is becoming increasingly more challenging to cope with due to the emergence of new technologies, such as artificial intelligence and deep learning, which can create various methods of compromising data authenticity. As a result, it is critical to uncover new means to combat it by providing new viable solutions. In recent years, blockchain adoption has demonstrated its ability to provide new methods of acquiring decentralization, immutability, traceability, and transparency to systems that manage sensitive information. This thesis presents a new use case that uses blockchain and Merkle trees to protect videos recorded by verified news media professionals and where anyone can detect possible forgeries in video frames. This work shows that it is possible to combat misinformation with good performance using these technologies on a real-world system ensuring trust, transparency and integrity to the data stored in a distributed ledger.
Keywords: Video Authentication; Misinformation; Blockchain; Hyperledger Fabric; Merkle tree; IPFS.

1. Introduction

Nowadays, falsified information is a huge concern that can lead to severe consequences which affect politics, economics, healthcare, and other critical sectors. A simple unintentional choice of words when sharing information on social media or, on the contrary, deliberate modification in informative content by malicious actors can have a massive impact on people's perception of world events.

Technologies like artificial intelligence and deep learning made it possible to generate the so-called *deepfakes*, which enable falsifying audio-visual content to force people to speak and act in a way that they never did.

Due to the emergence of this subject, it is becoming even more demanding for people to trust news media on the internet. Social media platforms like Twitter, Facebook, and Instagram, are used by millions of people to share any kind of information. These platforms enable sharing of altered information without proper rules, guidelines, and ways to detect fake content [1]. Therefore, as time goes by and *deepfake* technology becomes more accessible, it is generally accepted that the sharing of almost perfect manipulated videos of this kind will be at the distance of a click by the most common user on social media or other websites that follow that goal. Although sometimes it is straightforward to detect by the naked eye, this form of manipulation tends to become more and more complex to detect. As a result, there is a need to keep up with this evolution to find new ways to overcome the video authentication problem not only in social media but also in journalism organizations that make information available to everyone.

Blockchain is one of the technologies that can help overcome these difficulties and which in recent years is becoming increasingly fashionable. Based on a peer-to-peer network (P2P) maintained and verified by its network participants (nodes), it grants immutability to its data thanks to the power of cryptographic algorithms preventing content from being tampered with [2]. It can provide other properties, such as transparency, traceability, and decentralization. Recently, these properties can be applied to real-world use cases using blockchains that provide ways to create applications on top of a decentralized network.

In this thesis, a new use case for a decentralized system that interfaces with blockchain to tackle and detect the problem of falsified information sharing is proposed. Considering this issue and the state-of-art solutions that already exist to tackle it using the power of blockchain technology, the main goal of this thesis is to present a new approach to the detection and prevention of sharing untrusted news in videos using recent technologies that can

safeguard the integrity of this digital content and be functional for its users. Furthermore, it is also the objective of this work to answer the following research question:

- **Research Question**: *What are the requirements for a blockchain system to be effective in the detection and prevention of untrusted information in news videos?*

2. Background

This section introduces a background in blockchain, decentralized storages, Merkle trees and artificial intelligence in multimedia authentication.

2.1. Blockchain

Blockchain is one of the technologies considered a trend in current days. It is defined as an immutable distributed ledger running on a decentralized network where its network peers cryptographically protect transactions and verify them. Upon verification, a transaction is appended to the ledger if the network participants reach a consensus through consensus algorithms. [2] Since it is centred on decentralized P2P architecture, it does not depend on a central authority to manage its content which could be vulnerable to cyber-attacks.

Despite its advantages, there are drawbacks to using this technology as any other. Reaching consensus in some algorithms can be nefarious to the environment due to the humongous power consumption needed to append a block. An example of a consensus algorithm that can have a massive environmental impact is Proof-of-Work, the one used in Bitcoin, the most popular cryptocurrency.

Furthermore, some blockchains have different permissions regarding who can join and read its data depending on the goals a specific distributed ledger fulfils. Permissionless blockchains are the ones where anyone can participate and add data to the ledger. Participants do not require authentication to join or append data, and they get rewarded by expanding the network through the mining process of PoW. On the other hand, permissioned blockchains participants must be approved to join and operate in the network. It is generally accepted that permissioned blockchains are mainly used between organizations to enhance their businesses using the benefits of this technology.

Every blockchain solution's primary focus is preserving the integrity of digital content. The majority of these use a decentralized storage system to enhance the security properties of this technology. The purpose of this choice is to achieve complete decentralization by removing another central authority responsible to maintain and protect stored data. However, these file storage systems implement mechanisms to secure uploaded data, so applications tend to take advantage of those security properties. Currently, the state-of-the-art decentralized storage is the IPFS, but other platforms and studies can be an alternative or an enhancement to this decentralized file storage system.

IPFS is a P2P version controlled decentralized file system composed by a Merkle DAG "that seeks to connect all computing devices with the same system of files without the need for nodes to trust each other" [3]. When a file is uploaded to the IPFS network, a CID that corresponds to the stored address is generated so that it can be retrieved. Once a file is uploaded, it cannot be changed, which grants immutability to the stored content. Despite all of this, there is no such thing as a perfect system. IPFS cannot ensure the full availability of its data. A file has its access unavailable when nodes maintaining it break down.

2.2. Merkle Tree

This work uses the Merkle tree's powerful capabilities to provide high-performant verification properties. Created by the famous computer scientist Ralph Merkle [4], a Merkle tree is an authentication data structure that uses cryptography to secure data and verify its authentication in the transmission between telecommunication systems. The Merkle tree's creation consists in splitting data into nodes corresponding to the bottom level. Each node should contain the cryptographic hash of a correspondent portion of the data, and the upper-level nodes (parent nodes) are created by concatenating the hash value of their leaves. This building process continues until it reaches only one node, the root node that contains the root hash, which represents the hash of the entire data. This data structure is not only secure because it uses cryptographic algorithms, but it provides a fast mechanism of data integrity verification. If any modification occurs in a Merkle tree, the root hash is entirely different from the original one. Additionally, it is possible to verify and point out in which node of the bottom level the change happened by checking the differences between the original tree and the altered one by iterating over the tree leaves until it reaches the bottom level. This verification tree can encode data in blockchain more efficiently and securely, and it is also remarkably beneficial in other P2P networks such as IPFS, Git and Tor.

2.3. Artificial Intelligence

Most of the work done in manipulated digital content prevention focuses on automatic image detectors through sophisticated artificial intelligence and machine learning algorithms. The paper by Yu *et al.* [5] presents a survey on deepfake video de-

tection that describes numerous methods used in multiple works which use artificial intelligence algorithms such as:

- General network-based methods that use CNN;
- Temporal consistency-based methods using RNN;
- Biological signs-based methods and GAN.

Another existing detection method addressed in this survey is the camera fingerprint-based method, which detects different types of traces left by captured images.

An example of a notorious work that explores these technologies, also included in the Yu *et al.* [5] work, is the one by Li and Lyu [6]. In this paper, there is a proposed method to detect deepfake videos using CNN to capture distinctive artefacts caused by the process of deepfake content creation.

3. Related Work

In the paper by Alattar *et al.* [1], a solution to protect news-related videos from being manipulated is proposed. It uses blockchain technology as storage for information about the videos that are relevant for forensic analysis, and the distributed storage platform to save the video data. This solution presents an effective way to provide security measures to news videos against deepfake attacks. The system's architecture is enlightening for this thesis scope, which contains several components that together ensure security properties to secure digital content. However, it is not suitable for creating a solution where only organizations are the participants. Moreover, using Ethereum as a blockchain platform involves having to deal with transaction fees, which can be a huge drawback to developing a proof-of-concept for the goal of this work.

The paper by England *et. al* [7] presents AMP, a system that makes sure that media is authenticated by validating provenance. This system has a permissioned ledger, named Confidential Consortium Framework (CCF), that registers and signs manifests. Using a key-value store to save each one, where keys are the cryptographic hash of a manifest, and values are a publisher-generated signature formed by concatenating the key with the copyright string. These key-value pairs are then written to a Merkle tree, which is then copied and kept on persistent storage, offering a straightforward abstraction for the system's users to access. Through this framework, AMP ensures the accuracy and transparency of all registered manifests making all of its activities auditable while allowing a group of media providers to oversee the service. Visual components in the browser can be used to convey to the user the veracity of the material through browser extensions. The identification of fraudulent media is not addressed by AMP, which is the main focus of this thesis. The solution implemented in this paper that combats misleading information through provenance would be strengthened if it could identify manipulated content for its users.

Paper by Han *et. al* [8] provides another use for Merkle trees in multimedia. In this work, this data structure is used to securely transmit data over a P2P architecture and provide integrity checks and efficient data verification in a car live video streaming system. Instead of cryptographic hashes, the Merkle tree for this system uses perceptual hashes. These hashes get an advantage over conventional cryptographic hash algorithms since they can handle variations in format and quality, meaning that they are intended to ignore minor changes when an image is subjected to modifications such as compression, colour correction, and brightness [9]. However, using these algorithms introduce many drawbacks that affect the accuracy of a system to correctly authenticate multimedia. These systems must be implemented with the utmost precision to avoid false positives — content identified as unreliable but not manipulated — and false negatives — altered content the system can't verify to tackle malicious content.

The paper by Chan *et. al* [10] proposes a framework that combines the use of blockchain security with multiple LSTM networks to secure digital content. The methodology of this work involves using Hyperledger Fabric to store video data and CNN-LSTM algorithms to ensure data integrity. Instead of using a file storage manager to store the video content, they compress the CNN-LTSM network using vector quantization to reduce storage input for the blockchain platform. In future work, the authors suggest making a performance analysis of the system to assess their methodology. To enhance their system, the authors suggest overcoming their limitations by assuring mechanisms to grant authenticity to the digital content at the point of reception.

4. Design & Architecture

This section describes the developed system's design and presents the architecture aiming to outline the key design decisions to produce a novel tamper detection system in news-related videos. This system comprises a combination of different technologies, such as Hyperledger Fabric, CouchDB, IPFS, Merkle-Tree and MPEG-7 Video Signature. The decisions made regarding the choice of these technologies are described to fulfil this thesis' require-

ments. Then, the section follows up by presenting the architecture and demonstrating the interaction between the components that comprise the system to comply with the requirements. Finally, the use cases are disclosed to specify the system functionalities that each target user can perform.

4.1. Requirements

The architecture and design of a novel system that detects manipulation in audiovisual digital content must include improvements on the limitations of existing systems as well as other general prerequisites related to this subject. To be considered feasible, a system must meet both NFR and FR. The goal of this thesis is to implement a solution that maintains the integrity of trusted videos from reliable sources in a network made up of only allowed entities. Furthermore, the protected resources stored in the system should be made available to network participants. Additionally, the targeted untrustworthy users must confirm the legitimacy of videos found on untrustworthy platforms. To achieve this, the solution must meet the following non-functional requirements:

- ***Scalability (NFR-1)***: The system should provide means to add more users and nodes without compromising the performance;
- ***Tamper-proof (NFR-2)***: The system should ensure that the stored data has its integrity protected against manipulation attempts and should be able to show that the information submitted by the network participants has not been altered;
- ***Transparency (NFR-3)***: The system should allow read-only access to all network participants so that they can certify the veracity and rightfulness of the inserted data;
- ***Availability (NFR-4)***: All data should be available to each user depending on their access permissions;
- ***Performance (NFR-5)***: Each operation should be as efficient as possible so that it is functional for every user;
- ***Trust (NFR-6)***: The data contained within the system's components should be trusted by all sources;
- ***Adaptability (NFR-7)***: The system should grant the possibility for other use cases to be implemented easily.

The implemented solution should also meet functional requirements, Table 1, which should be met with the developed functionalities. The use cases that should comply with these requirements are *Submit*, *Get video* and *Detect*, which correspond to the functional requirements FR-1.1 to FR-1.4, FR-2.1 to FR-2.4, FR-3.1 to FR-3.4, respectively..

FR-ID	Description
FR-1.1	The system must allow only data submissions by enrolled users in a Fabric network's channel
FR-1.2	The application must generate a naming convention for each submitted file, easing the process of querying the ledger by reliable users
FR-1.3	The naming convention should be sent to the user upon submitting a video file
FR-1.4	The CID for the video data added to IPFS could be sent as a response, upon submitting the video, to make it available for every other user
FR-2.1	Only trusted users can query data from the ledger
FR-2.2	Trusted users should provide the generated naming convention for the file ID to query the ledger
FR-2.3	The video data could be sent to the user as response
FR-2.4	The video file must be sent to the user
FR-3.1	The system must classify an input video as "Trusted" if its entire content is stored in the system
FR-3.2	The system should classify an input video as "Partially--trusted" if it is a portion of a video stored in the system
FR-3.3	The system should classify an input video as "Not-trusted" if it is an entire video or a portion of it containing manipulations in between
FR-3.4	The system should not evaluate the input video if it is completely different from one that is being compared

Table 1: Functional Requirements

4.2. Technology Decisions

Considering the research presented in Sections 2 and 3 and to fulfil the previously shown requirements, the technology decisions to implement the system are presented in this section. The following sections explain those choices and describe each technology displayed in Table 2.

Technology	Decision
Blockchain Platform	Hyperledger Fabric
Peer State Database	CouchDB
Storage	IPFS
Video Handling Tool	FFMPEG
Video Fingerprint Algorithm	MPEG-7 Video Signature

Table 2: Technology decisions made for this thesis developed solution

For this work, private blockchains, such as Hyperledger Fabric, Quorum or R3 Corda, are the most appropriate to build a solution so that news organizations can securely store their data. Hyperledger Fabric is the most fashionable private blockchain solution for the enterprise level due to several reasons. The one that stands out the most is the high-quality support it offers by one of the most prominent open-source communities,

the Linux Foundation. It is extremely versatile and the most efficient option among other private blockchain platforms, according to the assessment produced in paper [11]. This choice is also based on its privacy and permission levels, which grants more security and combines more means to implement applications with more use cases for different users.

4.2.1 Storage

The storage choices used for the developed system in this thesis adhere to the emphasis placed in earlier sections on the necessity of decentralization to close the gaps in the current systems. Modern decentralized storage is utilized to accomplish this goal and to supplement a blockchain system like Hyperledger Fabric. Applications can take advantage of numerous security features provided by file storage system protocols like IPFS to build trust in the data sent across application users. A tamper-proof storage should be considered for this use case, where news-related videos need to be safeguarded from malicious modifications. Furthermore, compared to traditional databases, the data is more reliable and transparent because no central authority is trusted to manage it.

The best choice for the world state storage database among the Fabric network peers is CouchDB. To properly handle the implemented system's use cases, a database that supports rich queries is required rather than one that is more efficient. Only CouchDB can handle queries that fulfil the goals of use cases for assets submitted to the blockchain by authorized users, such as video ID naming conventions and data structures containing video frame information.

4.2.2 Video Fingerprint Algorithm

The main goal of the developed system is to detect video tampering to help users identify the veracity of information from untrusted sources. With the help of blockchain and a secure decentralized system, security properties mentioned more explicitly in Section 2.1 are assured, and data trustworthiness is granted, which will be explained in the following sections with more detail. However, to provide a fact-check feature where it is possible to point out malicious manipulations if they exist, it is necessary to use a library to process videos to accomplish this desired goal.

One of the most well-known frameworks is FFMPEG[1]. Through the use of a set of tools and libraries for application developers, it provides many operations in multimedia file content in a variety of formats. In this thesis, this framework was used to extract unique data from video frames using MPEG-7 Video Signature computing. The signature standardized by ISO/IEC MPEG outputs unique fingerprints of audiovisual content and is used to find common content in videos [12]. In the scope of this thesis, it is possible to implement the detection functionality with robustness and efficiency by extracting those fingerprints.

The reasons why no other fingerprinting algorithm was chosen are that it ensures excellent performance, and other approaches that can detect similarities in any kind of video would introduce a much higher false negative rate than this algorithm would do, which reduces the trust and security within the content. In the following sections, the architecture will be presented, where the main focus will be to ensure the most secure solution possible to build trust in the content while guaranteeing efficiency. This tool's adaptation to perform the use case will be explained in detail later in this document.

4.3. System's architecture

This section presents the system's architecture is presented, which describes the interaction and the accountability of each component to perform the desired application's features. To achieve this goal, some architectural views will be displayed and explained.

- **Module *application***: Contains the API and business logic built to allow users to perform the available use cases: *submit*, *getVideo* and *detect*. Moreover, it interacts with the module *chaincode* to execute transactions to the Fabric network.
- **Module *chaincode***: Comprehends application logic to perform the transactions in the blockchain working as an intermediary between the application module and the fabric-network module. This module is installed on the peers of the Fabric network.
- **Module *fabric-network***: The objective of this module is to set up the Hyperledger Fabric network so that transactions can be made using the chaincode when users make requests to the API in the application module. It contains all the configurations to run the network.

Figure 1, describes the interactions between each component present in the system. In this diagram, we can observe that when a user makes a request through some web browser, it is handled by the application module using a REST API. It is also possible to see that there is the IPFS component which is responsible for storing the video

[1] https://ffmpeg.org/about.html

content and other data crucial to perform the use cases that will be presented in Section 4.4. The application module interacts with the fabric network to safeguard the information about the videos. That information is shared between the peers (P1, P2, P3, P4) belonging to Channel1 which contains the chaincode module (CM) installed.

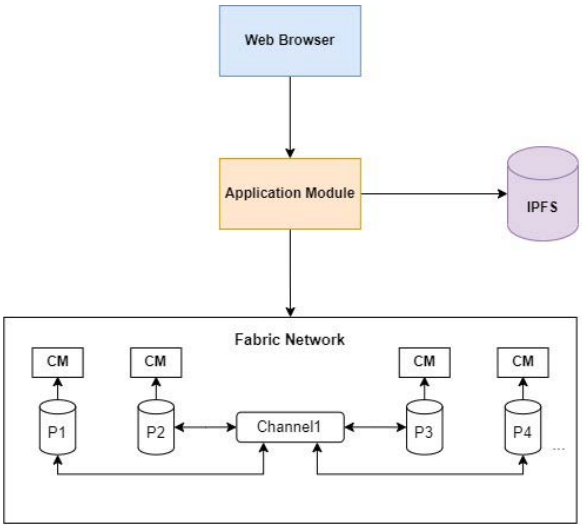

Figure 1: System's components architecture view

4.4. Use Cases

This section explains the use cases of the developed application, implemented inside the *application module*. Firstly, the use cases were designed for two distinct users.

The **Trusted user** is in charge of securely submitting and retrieving the video from the blockchain network. A trusted user is a member of the blockchain network and belongs to one of the network's media organizations, which form a consortium within the Fabric network. Following submission, the relevant saved data should be stored in the blockchain so that other trusted users can see and validate it.

The **Untrusted user** is the average internet user, which is not enrolled in the blockchain, and thus they cannot interact with it. This user can view or download videos as well as perform the manipulation detection functionality.

The ***Submit*** use case, which only a trusted user can perform was designed for video creators to safeguard their videos, and to do it, a user must be enrolled in the blockchain network first by belonging to a channel where their organization is present. When a creator submits a video to the application using the *submitVideo* function, a unique ID for the file is generated, *fileId* (Section 5.1). The app computes the Merkle tree of that video and gets the resulting Merkle root (the topmost node value of the tree which is the hash of the whole file) and frame hashes (a list containing all the hashes of each frame which is also the bottom level of the tree) to be stored in the IPFS repository alongside with the video and Merkle-root. As explained in previous sections, upon adding files to the IPFS a unique address to retrieve the stored data is generated, *cid*. Therefore, after receiving the address as a response, the application uses the credentials of the trusted user to submit a transaction to the fabric network using the installed chaincode on the channel's peers where the user has their organization enrolled. The submitted data includes the *merkleRoot*, *fileId* and *cid*.

Similarly to the previous one, to perform the ***Get Video*** use case, a user must be already enrolled in the blockchain. Only after creating an identity in Hyperledger Fabric can a user access resources stored in it using their credentials. A certain organization user might want to fetch the video file securely saved so that it can be published later on the organization's platform and then spread to the public. To accomplish this goal, a trusted user (in this case a publisher) must have access to the *fileId* generated for a certain file, which is an easy naming convention so that other organization members can easily access the submitted resources. When the publisher wants to have access to a video using the *getVideosByFileId* function, a query to the blockchain is made to find the resource by that naming convention. The data is retrieved containing the same values submitted in the previous use case, *merkleRoot*, *fileId* and *cid*. When the app gets this data structure, then it executes a query to the IPFS repository to find the object that contains the video data and the video content CID. With the address of the video content retrieve, the application can perform a second query to the IPFS to get it, making it available for the trusted user to download.

The ***Detection*** use case consists of an upload of an unverified video to evaluate its authenticity. Using the *Detect* function, common internet users can submit a video to be evaluated along with the CID address of the trusted video they want to compare with. The trusted video is already present in the system and was submitted by a news industry professional. As a consequence, the unreliable one uploaded by the untrusted user must be included in it, so that the application verifies whether content can be verified by performing a comparison. In most cases, the uploaded video is a portion of the whole content of another, which is called a "clip". This functionality must handle video "clips" as input and define whether they were manipulated or were simply taken out of the original video's context. During the execution of this feature, it retrieves reliable video data, by querying the IPFS using the input *CID*, containing the *merkleRoot* and the bot-

tom level of the tree, the *frameHashes*. Consequently, the application computes the Merkle tree of the untrusted video and if the video is exactly the same, meaning both Merkle trees are identical, a response is sent to the untrusted user with the message that the suspicious video can be trusted. However, if both Merkle roots are different the app performs some operations to make both Merkle trees comparable. When both trees can be compared the function compareMerkleTrees() is executed and, if the video is manipulated, the frame where the manipulation occurred is pointed out as a response, otherwise, it replies to the user that the uploaded video can be trusted and it is simply a cut from a trusted video which might have been taken out of context.

4.5. Data Model

This section enlightens the data structures involved while performing the previously presented use cases:

- ***LedgerData***: is the data structure that is stored in the blockchain when a trusted user submits a reliable video. It contains the necessary information to secure the video data to be shared with other trusted users within the fabric channel where the transaction occurred. Another trusted user with authorization to access the content can fetch the data contained in this structure. It holds the generated *fileId*, the video's *merkleRoot* and the *videoDataCID* which is the address of the *VideoData* structure of the submitted video;

- ***VideoData***: the data resulting from the Merkle-tree creation is stored in this structure. The goal of it is to be saved in the IPFS for the purposes described in the Section 4.4. Since it is safeguarded in this distributed file system, it has a *cid* attached to it. Similarly to the LedgerData, the *fileId* and *merkleRoot* are present in this structure along with the *frameHashes* (bottom of the Merkle-tree) and *videoContentCid* (the address of the actual video file);

- ***VideoContent***: the video content is saved in the IPFS in an object containing the actual file and the address to it, *cid*. By querying this resource a user can download the video.

5. Implementation

This section presents the implementation details of the solution carried out in this thesis. The application was developed locally using NodeJS version 16.14.2, IPFS version 0.15.0 and Hyperledger Fabric version 2.4.6.

5.1. Submit implementation

To perform the submit functionality a user must send a video as input to the application. Firstly, a user must be enrolled on the Hyperledger Fabric network to accomplish this goal. When the network is up and running with the chaincode already deployed and with a certain trusted user already enrolled, the application asks for the identity of that user and validates whether it is present in the wallet folder. Using the Fabric SDK for Nodejs, a connection between the blockchain and the client is made by creating a gateway using the identity and the wallet. If it succeeds the user is trusted and can upload the video. When the video is received, a file unique id is generated. The goal of this identification is to ease the process for trusted users to query the blockchain for the right video. For this implementation a simple naming convention comprised of the following information:

- **Organization name**: the news organization where the trusted user works;

- **User name**: the name of the user submitting the video;

- **Category**: the topic addressed in the video (politics, sports, environment, etc...);

- **Submission Timestamp**: the exact timestamp when the video submission occurred.

Once the naming convention is generated, an FFMPEG command is run to create the digital signatures of each video frame. The MPEG-7 video signature, as explained in Section 4.2.2, provides a means to uniquely identify video frames as well as other relevant information that composes this signature. As a result, by digesting those values with a cryptographic hash algorithm, it is possible to construct Merkle trees of video frames. For this case, the SHA-256 was the choice for the hashing algorithm. In comparison to other cryptographic algorithms, it is not the fastest algorithm for hashing strings but its performance doesn't have much negative impact and it is currently one of the safest options. Regardless of the choice of the cryptographic hash function, the system wouldn't be compromised if an attacker obtained access to the original content of the video frames signature because the data is publicly posted on IPFS and is protected by the deduplication feature described in earlier sections of this article.

When the digest is completed, a list containing the resulting hashes is returned. This list corresponds to the bottom level of the Merkle tree of the trusted video. As a consequence, the tree is built with the following algorithm:

The Merkle tree is built from the bottom up, with nodes attached to the tree using the functions *ComputeBottomLevel* and *ComputeMiddleLevel*. A node contains references to both its left and right child nodes. A list variable *nextLevelNodes* keeps track of the previously created level's resulting upper-level nodes. A next-level (or parent) node contains the hash of its left and right child nodes concatenated. As a result, each time the Merkle tree height is increased, a new level of nodes is added until it reaches a single node, the *merkleRoot*.

Following the generation of the Merkle tree, the video is uploaded to IPFS separately from the other data about the Merkle tree and the file naming convention. The decision was made due to performance concerns and to avoid data corruption, as putting everything in the same IPFS object causes errors when parsing the object to JavaScript. To accomplish this, a local IPFS daemon was launched, and the video was saved to the repository using the *add* function from the NodeJS package *ipfs-http-client*. When this operation is completed successfully, the CID is returned, and the other data is added to a different IPFS object, yielding a different CID. The data saved to that object consists of the file identifier, the hash of the *merkleRoot*, the frames list, and the video CID.

To complete this procedure, the client initiates a transaction to submit the required data to the distributed ledger using the appropriate chaincode function. In this case, the data to be submitted consists of the CID returned from the previous upload to the IPFS repository, the file ID, and the Merkle root's hash. The values are sent as String parameters to the chaincode, and if successful, the data is appended to the ledger.

5.2. Get video implementation

To obtain a video from the implemented system, one must be trusted within the Fabric network, similar to how the Submit feature was previously explained. As a result, before they can make requests to the endpoint responsible for this use case, users must be enrolled. After verifying the user's authenticity, the application establishes a connection with the blockchain to use the chaincode in the same manner as in the previous functionality. This time, the application queries the blockchain using the file's naming convention to retrieve the desired video data using the chaincode function *QueryDataByFileID*. If the transaction is successful, the fabric network returns the CID of the video data, and the application then queries the IPFS for that data, where the file content's CID is stored. When the *Uint8Array* data chunks are converted into JavaScript objects, the file content's address is obtained, and the app downloads the video from IPFS if the procedure is successful.

5.3. Detect implementation

Concerning the detection feature's implementation, an untrusted user must be able to assess the trustworthiness of videos found on the internet from untrusted sources. To perform this functionality, the trusted video to be compared with the one that is going to be validated must have been already posted in a way everyone can see it. Even though the front end of this application has not been implemented, it is assumed that a user has access to the CID of the video data. The video data is retrieved from the IPFS local node using the function from the NodeJS package *ipfs-http-client* named *cat*.

When an untrusted user uploads an unreliable video, the *Detect* function extracts its video signature extraction using the same procedure as in the *Submit* implementation. A list containing the hashes of the MPEG-7 video signature is computed corresponding to the bottom level of the untrusted Merkle tree. Then, the Merkle tree is built so that the comparison is made.

At this point, if the video is exactly the same, the untrusted Merkle root hash is equal to the one stored in the IPFS, which means the video can be trusted. However, if there's a mismatch, the video might be a clip which was cut from a reliable source. Then, an algorithm is performed to check whether this possibility might be confirmed using the video frames retrieved from the IPFS.

If the first entry of the untrusted frame hash list is present in the trusted list, it scales down the trusted frame hash list to match the size of the untrusted one. Therefore, this algorithm allows checking if the user can rely on the clip returning a response in case both hash lists are equal. Otherwise, the video is determined as being manipulated. Hence, a new Merkle tree using the re-scaled hash list is built to be compared with the uploaded one by the user.

The Merkle tree comparison focuses on searching for the first element in the untrusted tree that is different from the trusted one. The function ensures that the trees are traversed equally from top to bottom whenever a node is different. This computation prioritizes the leftmost node in case both branches of a certain node are different from the trusted tree one so that when the traverse is completed, the first manipulated node is reached, and the corresponding frame number is returned.

5.4. Chaincode

This module contains the main JavaScript file with the contract necessary to install in the Fabric network's peers and submit transactions.

Using the Fabric SDK for NodeJS, a class *Video-*

Data is created, which extends the *Contract* class provided by the *fabric-contract-api* NodeJS package. This class contains a variety of functions available to perform operations in the ledger. Every function receives a transaction context *ctx* as a parameter which provides access to the Fabric API operations to update and query the ledger [13].

The trusted user interacts with two distinguished functions present in the chaincode deployed in the channel, which are run throughout the execution of the features targeted for them. The first function is ***Submit(ctx, merkleRoot, fileID, cid)***. This function is used in the *submit* use case when a trusted user wants to add video data to the ledger. The chaincode first checks to see if the *merkleRoot* provided by the client application is present among the values that have been recorded in the ledger. If it is unique, the Fabric API is used to marshal the video data to JSON and update the CouchDB world state database with a key-value pair matching the *merkleRoot*. The final step involves building a composite key in the world state that enables performing rich queries to the ledger.

On the other hand, the function ***QueryVideoDataByFileID(ctx, fileID)*** is used during the *Get video* use case. The process of this chaincode method consists of the creation of a rich query selector to retrieve the video data using the file naming convention. Regarding this function's implementation, a parameterized query is created with the "docType" set to *videoData* and the *fileId* given by the client filling out the other corresponding field. The rich query is run against the state database using the Fabric API function *getQueryResult*, which returns a *StateQueryIterator*, a data structure that can be iterated over using the internal function *_GetAllResults*. The client receives the return value as a JSON string after it has been unmarshaled.

6. Results & discussion

In this Section, the evaluation of the implemented system is presented. The assessment methodology is demonstrated, and then the results are discussed. The assessment methodology for this thesis was divided into three categories:

- **Soundness Evaluation** - The purpose of this evaluation method is to assess the system's functionalities. Evaluating how well the system's features operate is vital to ensure that the requirements are met. In this instance, the evaluation concentrates on the *Detection* use case to evaluate the precision in manipulating video detection. We assessed all functional requirements, and for non-functional requirements, we evaluated everything we could with the prototype implementation, but NFR-1 and NFR-3 require a deployed system in the wild. The results showed the application can perform validations on untrusted input, building trust in the content with the presented architecture. It can correctly identify and point out different types of anomalies found in videos that were manipulated, using as sources the ones stored in the system. However, the *Detection* feature currently works on the assumption that every input the user uploads to be assessed contains the same features (video codec, resolution, video quality, compression, etc.).
- **Performance Evaluation** - After testing the soundness of the system, it is also crucial to validate whether the functionalities are efficient to perform. To accomplish this objective, the latency of each functionality is measured and discussed. For this assessment, single-threaded execution was assumed in order to represent the worst-case scenario for machines with just one core available. The performance evaluation outcomes show that Merkle trees are viable data structures to be used in video signature hashing for video authentication. Additionally, the most prominent advantage of this data structure is the ability to perform comparisons with other trees at great speed with a time complexity of $O(\log_2(N_f))$. This operation's latency only depends on the tree's height, which is always a small number. This would not be the case if arrays were utilised for the comparison, which would have a time complexity of $O(N_f^2)$. The interactions with the components that encompass the system revealed an expected assessment because they have proven to be highly performant within systems similar to the one developed in this thesis. Nevertheless, the latency of the signature disclosed that, when an input video uploaded to perform the *Detection* functionality contains a high number of frames, the operation may be impracticable, jeopardizing the application's usability. This occurs because the application needs to calculate the video signature for each frame to complete the operations. However, this issue is not considered a critical risk because, this computation is executed in a single thread, and as the experiment demonstrates, it can process a two-hour duration input video at a doable latency. Moreover, as stated in the soundness evaluation discussion, this could be mitigated by replacing the MPEG-7 video signature with another type of fingerprint algorithm with more performance.
- **Requirements Evaluation** - The decisions

made throughout the development of this thesis are evaluated to check if the system fulfils the requirements. The results show that every requirement was met, except NFR-1 and NFR-3. Those that were not fully achieved must be better evaluated in the future with additional tests. The implemented system demonstrated good adoption prospects. However, a more detailed evaluation of the presented architecture is required to develop more use cases to give power to this idea.

7. Conclusions

This document presented a blockchain-based misinformation detection system in news-related videos. This approach, in particular, enables regular internet users to verify the videos they come across with the potential to establish trust in the content by removing trust from untrustworthy source platforms. Concerning the general research question, another step was taken to keep up with the technological progress of new emerging procedures that create untrustworthy content. However, more applications of similar prototypes must be investigated and applied to a real-world system to determine more limitations and explore the impact they can make, not only in the journalism industry but in citizens' perception of real-world events.

References

[1] Adnan M. Alattar, Ravi K. Sharma, and John Scriven. A system for mitigating the problem of deepfake news videos using watermarking. *electronic imaging*, 2020.

[2] Adnan Qayyum, Junaid Qadir, Junaid Qadir, Muhammad Umar Janjua, Muhammad Umar Janjua, Falak Sher, and Falak Sher. Using blockchain to rein in the new post-truth world and check the spread of fake news. *IT Professional*, 2019.

[3] Juan Benet. Ipfs - content addressed, versioned, p2p file system, 2014.

[4] Ralph C. Merkle. Secrecy, Authentication, and Public Key Systems. Master's thesis, Stanford University, June 1979.

[5] Peipeng Yu, Zhihua Xia, Jianwei Fei, and Yujiang Lu. A survey on deepfake video detection. *IET Biom.*, 10(6):607–624, November 2021.

[6] Yuezun Li and Siwei Lyu. Exposing deepfake videos by detecting face warping artifacts. *arXiv: Computer Vision and Pattern Recognition*, 2018.

[7] Paul England, Henrique S. Malvar, Eric Horvitz, Jack W. Stokes, Cédric Fournet, Rebecca Burke-Aguero, Amaury Chamayou, Sylvan Clebsch, Manuel Costa, John Deutscher, Shabnam Erfani, Matt Gaylor, Andrew Jenks, Kevin Kane, Elissa M. Redmiles, Alex Shamis, Isha Sharma, John C. Simmons, Sam Wenker, and Anika Zaman. Amp: Authentication of media via provenance. In *Proceedings of the 12th ACM Multimedia Systems Conference*, MMSys '21, page 108–121, New York, NY, USA, 2021. Association for Computing Machinery.

[8] Dongqi Han, Jindong Zhang, Yang Liu, Peibin Wu, and Yiding Sun. Live video streaming authentication for vehicle multimedia based on merkle tree. In *Proceedings of the 2018 International Conference on Network, Communication, Computer Engineering (NCCE 2018)*, pages 530–534. Atlantis Press, 2018/05.

[9] Li Weng and Bart Preneel. A secure perceptual hash algorithm for image content authentication. In *Proceedings of the 12th IFIP TC 6/TC 11 International Conference on Communications and Multimedia Security*, CMS'11, page 108–121, Berlin, Heidelberg, 2011. Springer-Verlag.

[10] Christopher Chun Ki Chan, Vimal Kumar, Steven Delaney, and Munkhjargal Gochoo. Combating deepfakes: Multi-LSTM and blockchain as proof of authenticity for digital media. In *2020 IEEE / ITU International Conference on Artificial Intelligence for Good (AI4G)*. IEEE, September 2020.

[11] Julien Polge, Jérémy Robert, and Yves Le Traon. Permissioned blockchain frameworks in the industry: A comparison. *ICT Express*, 7(2):229–233, 2021.

[12] Stavros Paschalakis, Kota Iwamoto, Paul Brasnett, Nikola Sprljan, Ryoma Oami, Toshiyuki Nomura, Akio Yamada, and Miroslaw Bober. The mpeg-7 video signature tools for content identification. *IEEE Transactions on Circuits and Systems for Video Technology*, 22(7):1050–1063, 2012.

[13] Hyperledger. A blockchain platform for the enterprise - hyperledger-fabricdocs main documentation. Technical report, Hyperledger, 2020-2022. Last time accessed on October 2022.