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

Diogo Filipe Pinto Nogueira

Thesis to obtain the Master of Science Degree in

Information Systems and Computer Engineering

Supervisor: Prof. Kevin Christopher Gallagher

Examination Committee

Chairperson: Prof. Pedro Tiago Gonçalves Monteiro
Supervisor: Prof. Kevin Christopher Gallagher
Member of the Committee: Prof. David Rogério Póvoa de Matos

November 2022

Acknowledgments

I acknowledge with gratitude the support of all the professors involved since the beginning of this thesis. In particular, I would like to thank Prof. Kevin Gallagher for his effort that, despite having joined halfway into the team of supervisors, helped me overcome encountered limitations during the thesis.

I wish to dedicate my work, first and foremost, to my family, who always supported me all these years and offered me everything they could to reach this stage of my life. To my girlfriend, Madalena Clemente, whose love and empathy made me give my all every day. To my friends Ricardo Silveira, Diogo Matias, Filipe Capela, Carolina Lucas and Rafael Branco, for the years of continuous friendship, joy and support. To all of you, I love you, and I am grateful to have you in my life.

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; Merkle tree; IPFS; Hyperledger Fabric.

Resumo

A manipulação de conteúdos digitais relacionados com notícias tem vindo a aumentar ao longo do tempo. É um problema que tem vindo a tornar-se cada vez mais difícil de lidar devido ao aparecimento de novas tecnologias, tais como a inteligência artificial e *deep learning*, que podem criar vários métodos de violação da autenticidade dos dados. Por isso, é fundamental encontrar novas formas de a combater, fornecendo novas soluções viáveis. Nos últimos anos, a adoção de soluções *blockchain* tem demonstrado a capacidade de fornecer novos métodos de descentralização, imutabilidade, rastreabilidade e transparência a sistemas que gerem informação sensível. Esta tese apresenta um novo caso de uso que utiliza *blockchain* e *Merkle trees* para proteger vídeos gravados por profissionais dos meios de comunicação verificados, e onde qualquer pessoa pode detectar possíveis falsificações em frames de vídeo. Este trabalho mostra que é possível combater a desinformação de forma eficiente utilizando estas tecnologias num sistema do mundo real, garantindo confiança, transparência e integridade aos dados armazenados numa *ledger* distribuída.

Palavras Chave

Autenticação de vídeo; Desinformação; Blockchain; Merkle Tree; IPFS; Hyperledger Fabric.

Contents

List of Figures

List of Tables

List of Algorithms

Acronyms

API	Application Programming Interface
BFT	Byzantine-Fault Tolerance
CA	Certificate Authority
CFT	Crash Fault Tolerance
CID	Content Identifier
DAG	Directed Acyclic Graph
FR	Functional Requirement
IPFS	InterPlanetary File System
LSTM	Long Short-Term Memory
MSP	Membership Service Provider
NFR	Non-Functional Requirement
PBFT	Practical Byzantine-Fault Tolerance
PKI	Public Key Infrastructure
PoS	Proof-of-Stake
PoW	Proof-of-Work
P2P	Peer-to-Peer

1 Introduction

Contents

1.1 Motivation

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. At first, it is crucial to identify its purpose and characteristics and understand the elements that make this form of manipulation feared by reliable sources of information and those who want to inform themselves from trusted content. By doing this, we can fully understand the problem that needs to be solved and ensure that it is possible to prevent it effectively.

There are two types of false information: misinformation and disinformation. The difference between these concepts lies in the intent of whom is spreading news or any other type of information. Misinformation is the sharing of false information regardless of the intention to harm a party or entity, whereas the latter concept means the same act but to target, for instance, an organization or country, to deal reputational damage or even control masses with misleading means of propaganda. [4] Despite their purpose, both types are harmful and can deceive citizens by manipulating their beliefs and acts through falsified facts.

Disinformation is produced with malicious intent, aiming to persuade the public and give the wrong message about facts through non-ethical means of propaganda, such as creating fake news sites, bots, and fake accounts only for that purpose. This criminal activity leads to conflict and insecurity in political world activities allowing the creation and sharing of audio-visual disinformation content in an increasingly fast and effective way. The term ”fake news” became popular with the controversial 2016 United States of America presidential election of Donald Trump when he accused several media organizations of creating misleading information to attack his reputation and assault his election [5]. Today, new ways of ”fake news” have emerged to another level.

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 they never did. One of the most notorious examples of this technology in action is a video showing former U.S.A. president Barack Obama being impersonated by Jordan Peele, the famous actor, comedian, filmmaker, and also the author of this *deepfake* [6]. More recently, during the war conflict between Ukraine and Russia that emerged to another level in February 2022, the latter country was accused of trying to deceive Ukrainian citizens to make them surrender, by producing a *deepfake* video of Ukraine president Volodymyr Zelenskyy in which he declared peace between both countries [7].

This type of *deepfake* is called ”puppet-master”, in which the author mimics someone's image and voice.

Another way to create distorted facts is by producing and publishing out-of-context video clips. Sometimes, this harmful manipulation occurs in some proclaimed trusted media companies by performing subtle edits. This editing is performed by cutting part of the video that contains relevant information, in a way that people cannot perceive it occurred. Therefore, the real intention of some idea that a person who was giving a speech was sharing in that video is distorted to another with malicious intent. In November 2021, an example of this type of falsified information occurred when Fox News edited a video of Joe Biden's speech in a way that makes the current U.S.A. president sound racist [8].

1.2 Problem

Due to the emergence of the topic presented in the previous section, 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 [9]. 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. At the moment, there is no concise and flawless way to tackle disruptions of authenticity in digital artefacts while ensuring that the source of information is trusted reliably. By experimenting with better alternatives that guarantee more credibility in data handling, we can help prevent people from being deceived.

Blockchain is one of the technologies that can help overcome these difficulties and which in recent years is becoming increasingly fashionable. It is a distributed ledger based on a Peer-to-Peer (P2P) network maintained and verified by its network participants (nodes), granting immutability to its data thanks to the power of cryptographic algorithms preventing content from being tampered with [10]. Blockchain 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.

1.3 Goals and Objectives

Considering the issue presented in the previous section 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?*

This thesis reveals an extensive literature review to understand the state-of-art systems and current works. Those research papers served as a source of inspiration and knowledge for the new system implementation, which can take the best parts of the related work and contribute to the scientific community to tackle the above research question.

Moreover, this problem addresses many other technology areas, mainly artificial intelligence. It is in the scope of this work to outline how this system would benefit from those computer science areas to be applied as future work and use their properties to build a more complete and robust solution. This way, it will be possible to know how to adapt the developed procedure to other use cases that combat misinformation.

1.4 Dissertation Outline

The remaining document contains several chapters that present this work and is structured as follows.

Chapter 2 presents the background with explanations for the different concepts in the scope of this system implementation, as well as the literature review in the context of this work containing similar solutions on which this thesis was based. Furthermore, it explores the current state-of-the-art and researched technologies.

Chapter 3 exposes the system architecture design decisions and requirements to fulfil the objective of this thesis. It describes the technologies used in the implementation used to develop the system functionalities and create the interaction between the present components, which are explained in full detail in this chapter.

Chapter 4 describes the implementation details used to face the problem and develop the proof-of-concept according to the system design presented in Chapter 3.

Chapter 5 refers to the proposed system evaluation. It illustrates the methods used to evaluate the system and discusses whether it fulfils the requirements previously stated in Chapter 3.

Chapter 6 summarizes the whole document presenting the obstacles encountered during the development of this thesis. Future work is also demonstrated with possible ideas to overcome the limitations of this system.

2

Background & Related Work

Contents

This chapter presents and analyses the related work projects in the scope of this thesis, alongside the different techniques that characterize the state-of-the-art of this research. Additionally, it 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 [10]. 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.

A transaction in a blockchain is a transfer of value between two entities. Wallets are needed to perform this action, which contains the identity of a party represented by a pair of cryptographic keys: public and private. Therefore, a user with a wallet can sign transactions using their private key in a blockchain to transfer assets to other participants.

This technology contains blocks that store the transaction's timestamp, information about the participants of each transaction, and a hash for its previous block, making it possible to trace all past transactions to their origin. Moreover, everything occurring inside the blockchain is available for its participants to witness, providing transparency to the network, a property essential to ensure trust in everything contained inside the blockchain.

A transaction is successfully accepted in the network if validator nodes verify it is legal. Malicious nodes can try to append a block that is not valid to the blockchain aiming to compromise the network. Therefore, participant nodes must agree upon the addition of transactions in a process called consensus through consensus algorithms.

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 (PoW), the one used in Bitcoin, the most popular cryptocurrency. In PoW, the validator nodes create new validated blocks by solving complicated cryptographic puzzles to contribute to the network expansion. A node is rewarded with cryptocurrency if it is the first to solve that puzzle. Therefore, the one with the most computational power to create the next block gets rewarded. Although it is very resource-intensive, there is the trade-off of being incredibly secure.

Nevertheless, some public blockchain environments are switching to Proof-of-Stake (PoS), as in the case of Ethereum since the *"Merge"*[1] on September 15, 2022. It is an alternative to PoW much less damaging to the environment because there is no need to solve those complex puzzles. Instead, entities who wish to contribute to a particular network must previously own the cryptocurrency of that blockchain, staking that currency to create the next block. As a result, this consensus algorithm is less secure. If an entity has the economical power to stake a massive amount of currency, it could control the whole network.

As a consequence of the huge power consumption needed, businesses that use PoW blockchains must be prepared to deal with transaction fees and spend money on computational power, making the development expensive. Besides, although this technology presents properties that make it more secure than other alternatives, it is not thoroughly protected. It is vulnerable to attacks that can compromise the whole network and take control of its data, as previously stated while describing the PoS algorithm. If an entity controls at least 51% of the network nodes it means that "it gives the controlling parties the power to alter the blockchain" [11]. Moreover, it can also be susceptible to DDoS, compromising the network availability by congesting it with millions of requests.

As we have seen, not all blockchains are equal. They can have multiple distinguished characteristics and have different types of consensus algorithms. Furthermore, some blockchains have distinct permissions regarding who can join and read its data depending on the goals a specific distributed ledger fulfils.

2.1.1 Permissionless

Permissionless blockchains are the ones where anyone can participate and add data to the ledger. Moreover, participants do not require authentication to join or append data. If a permissionless blockchain platform uses PoW, a fashionable consensus algorithm among permissionless blockchains, participants can get rewarded by expanding the network through the mining process of PoW. Examples of permissionless blockchains are Bitcoin and Ethereum. The first one allows transactions of cryptocurrencies between two entities with a wallet, whereas the latter provides a decentralized environment to create decentralized applications within a public network. A permissionless blockchain is known for having the following characteristics:

- Fully decentralized: A blockchain is fully decentralized when no permission or authority controls the assets inside the blockchain. Everyone is equal within the network, so the participants have equal permissions and capabilities to use its features;

- Transparency: Public blockchains with no permissions are fully transparent. All network participants can verify the occurred transactions within the blockchain. However, this can also be a

[1] https://ethereum.org/en/upgrades/merge/

drawback depending on the usage that a certain blockchain solution has because it lacks privacy since everything is visible;

- Performance: This distributed ledger type tends to be more congested. As a consequence, transaction speeds are lower the larger the network becomes;
- Energy consumption: As mentioned before, some consensus algorithms consume too much energy, being harmful to the environment. Permissionless blockchains often use those consensus mechanisms like Bitcoin and Ethereum (before the *"Merge"* on September 15, 2022).

2.1.2 Permissioned

Permissioned blockchains have similarities and some massive differences in comparison to permissionless ones. They also ensure transparency to the network by granting visibility to everything that occurs inside the ledger. Furthermore, they grant immutability due to having validator nodes that preserve the integrity of the data. In opposition to the type of blockchains mentioned before, the participants of permissioned blockchains must be approved to join and operate in the network. In permissioned blockchains, the number of entities that compose the network is much lesser than the ones that exist in public or permissionless blockchains. This happens because this type of blockchain is assembled to support a determined group of participants. As it is stated in a paper where a known permissioned blockchain is described (Hyperledger Fabric), [12] "A permissioned blockchain provides a way to secure the interactions among a group of entities that have a common goal but which do not fully trust each other, such as businesses that exchange funds, goods, or information.". It is generally accepted that permissioned blockchains are mainly used between organizations to enhance their businesses using the benefits of this technology. Many characteristics define permissioned blockchains, such as:

- Not fully decentralized: Only participants with permission to join the network can operate in this type of blockchain. As a consequence, decentralization is reduced due to having lesser participants. Moreover, since there are permissions there must be a central authority that decides the restrictions within the network;
- Performance: With less congestion, there is also more performance. This type of distributed ledger offers much faster transactions speeds;
- Consensus algorithms: Permissioned blockchain use different consensus algorithms. Since it is a network composed of a certain number of known entities, they can ensure trust by using algorithms such as Byzantine-Fault Tolerance (BFT);
- Customizability: These blockchains are much more customizable. They can offer a wide range of settings depending on the use case. Smart contracts, programs that run on a blockchain, can be

written in many different programming languages. Also, consensus algorithms can vary, offering other alternatives like Practical Byzantine-Fault Tolerance (PBFT) and Raft.

2.1.3 Platform Overview

Firstly, to develop a blockchain solution that tackles the problem addressed in this thesis, it was necessary to decide which platforms provide the best procedure to define a well-grounded system for this purpose. It was also fundamental to understand what kind of trade-offs were put at stake. Thus, understanding the advantages and disadvantages of each component provides crucial information about the technologies to obtain a cohesive and well-rounded system.

This section contains a survey on different blockchain platforms researched for this solution. Each one is described, the most common use cases are enunciated, and their drawbacks and benefits are scrutinized below:

- **Ethereum**: Is the most fashionable public blockchain platform for developing software applications. Each Ethereum token is managed by employing a smart contract, which is a program with rules stored in a blockchain that is automatically executed. This blockchain platform is used to run with a proof-of-work consensus algorithm, which is one of the most secure but has the most negative environmental impact. According to [13], it is estimated that ”the Ethereum network annually uses 99.6 Terawatt-hours of electricity—more power than is required by the Philippines or Belgium. A single Ethereum transaction requires 220.05 kilowatt-hours of electricity, which is the same amount of power that an average U.S. household consumes in 7.44 days.”. However, current measures are being taken to avoid this issue with the development of an upgrade of the network, Ethereum 2.0. The so-called ”the Merge” that took place on September this year, changed its consensus algorithm to include the proof-of-stake, which is abruptly more environmental-friendly and can drastically reduce carbon emissions. Moreover, there are gas fees to perform transactions inside the network. These taxes, as it is described by the official Ethereum website [14], are ”the fuel that allows it to operate, in the same way, that a car needs gasoline to run”. It represents ”the unit that measures the amount of computational effort required to execute specific operations on the Ethereum network.”. These were becoming increasingly expensive at some point, however, recently they had their price reduced. [2]

- **Hyperledger Fabric**: As part of the Hyperledger foundation, Fabric is another platform that provides solutions using blockchain at the enterprise level and ”offers a unique approach to consensus that enables performance at scale while preserving privacy” according to this the official site of this platform [15]. It is a private permissioned blockchain consortium. Therefore, it is suitable to de-

[2] https://ycharts.com/indicators/ethereum_average_transaction_fee

velop applications for industry use cases. In the Hyperledger Fabric network, the participants must have an identity to execute a transaction. There is also a feature included in this platform that accepts transactions within a private sub-net called channelling, allowing sharing of information between two different organizations inside the network. It supports more programming languages to develop smart contracts than Ethereum. However, it is still limited to Go, Java, Javascript, and Solidity. In comparison to Ethereum, Hyperledger Fabric is an open-source free project which can use Raft, Kafka, or Solo as consensus algorithms to guarantee safety and ensure availability to the network through a crash fault tolerance mechanism. In addition, thanks to its capabilities, it can offer means to trace content to its origin. This platform is mentioned in a few papers of this thesis research, including the ones from Rashid *et al.* [3] and Chan *et al.* [16], but the state-of-the-art projects that combat misinformation in videos are developed using Ethereum blockchain.

- **R3 Corda**[3]: Is another permissioned platform available for developing decentralized applications focused on providing digital trust, especially in the finance sector, between entities in regulated markets. The procedure to reach consensus in this network to build trust in its assets is by attempting to ensure uniqueness and validity. These properties are achieved through its architecture where, to be allowed to join the network, each node that is a candidate to join must have its identity authenticated through certain nodes responsible to verify them. Moreover, upon enrolling to Corda, the nodes' transactions are validated to verify whether they achieve those properties [17].

- **Quorum**[4]: Despite being able to support other industries, similar to R3 Corda, this platform is mostly focused on building decentralized applications for the financial sector. It is based on the Golang implementation of the Ethereum blockchain with improvements in performance and privacy. Low throughputs are a result of Quorum's order-execute paradigm. Additionally, the consensus algorithm Raft enables it to perform even better.

Table 2.1 lists the blockchain platforms that were researched for this thesis, along with the characteristics of each one that were taken into account while picking the best platform for this use case. There are more platforms available that could be suitable for this work. However, these are the most fashionable ones that provide the finest tools for creating a reliable decentralized application that can accommodate a variety of development choices.

Figure 2.1 shows an overall analysis displayed in paper [1] in which the authors performed vast research on private blockchain platform comparison. In this figure, the Hyperledger Fabric show dominance in the majority of the metrics evaluated among the private blockchain paradigm.

[3] `https://www.corda.net/`
[4] `https://consensys.net/quorum/`

Platform	Open Source	Consensus Algorithm	Fees	Smart Contract Languages	Token	Support & Governance
Ethereum	Yes	PoS	Yes	Solidity, Vyper	Ether	Ethereum Developers
Hyperledger Fabric	Yes	Raft, Kafka, Solo	No	Golang, NodeJS, Java	None	Linux Foundation
R3 Corda	Yes	Raft	No	Java, Kotlin	None	R3 Consortium
Quorum	Yes	Raft Istanbul BFT	Ether	Solidity	Ether	Ethereum Developers & JPMorgan Chase

Table 2.1: Blockchain platform comparison overview based on paper [1]

2.1.4 Hyperledger Fabric

According to the platform overview of each blockchain technology presented in Section 2.1.3 and by observing Table 2.1 that summarizes their characteristics and Figure 2.1 that shows an overall assessment of each one, the chosen platform for this work was Hyperledger Fabric. This platform is the most well-known private blockchain platform for the enterprise level due to several reasons, but 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 versatile, allowing a broad range of settings that may be customized, as well as the ability to create smart contracts (in this case, *chaincodes*) in a variety of well-known programming languages. At the same time, it is very efficient providing high transaction speeds and is the most scalable option within the private blockchain platforms according to the Figure 2.1 provided in paper [1]. 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.

2.1.4.A Modular Components

As previously explained, Hyperledger Fabric is made for enterprise development in a private permissioned network of identified participants. It offers a modular architecture comprised of different components which follow the execute-order-validate paradigm of transaction flow [12]. Within this paradigm, the modular components follow distinguished roles [18]:

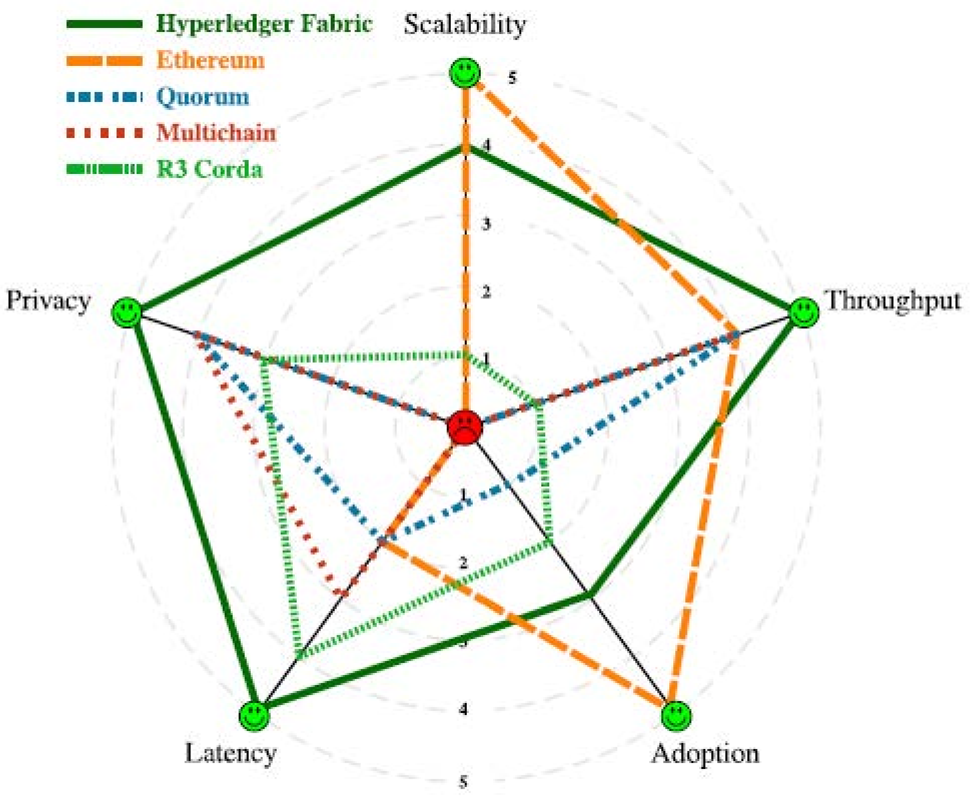


Figure 2.1: Blockchain platform overall analysis provided by paper [1]

- ***Membership Service Provider (MSP)***: When a node is added to the network, a Certificate Authority (CA) from the organization where this node belongs is attached to it. The CA provides identities to nodes using X.509 certificates that identify them within the network and prove they belong to a certain organization. Its main function is to issue digital certificates to entities so that they can sign transactions making every node know who committed data to the ledger. The Public Key Infrastructure (PKI) hierarchical model is used to generate the certificates for the nodes in the network. a unique MSP is present in each organization's network configuration and contains a list of those issued identities which are permissioned to the network;

- ***Chaincode***: The written programs that run in the Fabric network are named Chaincodes. After network administrators define the consortium (an association between two or more organizations within the network) a channel must be created so that organizations communicate with each other. It works as a private subnet where different organizations perform operations and share data confidentially. Consequently, all transactions of the network are executed within a channel by trusted peers that contain copies of the blockchain ledgers. Chaincodes are deployed to channels and installed in their peer nodes. Thus, client applications can use these chaincodes that comprehend application logic to perform operations shared by the consortium. With these programs, transactions can be executed to the whole network;

- ***Endorsement policies***: A chaincode contains endorsement policies which are guidelines on which peers should execute a transaction and approve the results so that it is considered valid. When a chaincode is deployed comprising endorsement policies to a channel, its members attached must agree on its addition before being committed. Then, the channel peers responsible to execute the transactions (endorsement peers) perform this process separately and send the results to the ordering service;

- ***Ordering Service***: A Fabric network needs an *orderer* node to be created at first. An *orderer* is a node responsible for establishing consensus on the order of transactions so that the validated blocks can be broadcasted to the peers. Along with other nodes of this type, the ordering service is created to fulfil this goal. The ordering service can have several implementations. Although there is only one implementation that is stated as the standard and that receives updates (Raft), there are others that can be used with the trade-off of being deprecated since Hyperledger Fabric version 2.x (Kafka, Solo). The Raft implementation is a Crash Fault Tolerance (CFT) ordering service that follows a model where a leader is elected per channel and the other nodes, known as followers, go along with the leader's decisions by replicating them;

- ***Validation***: After the ordering of the transaction results, the validation process is performed and consists of the verification of each block endorsed by the consortium responsible for the transaction. This procedure ensures consistency in the transaction's data as it verifies if the endorsement was executed correctly. Upon validation, the transaction is committed and appended to the ledger;

- ***P2P Gossip service***: The gossip data dissemination protocol aims to optimize blockchain performance and security. The Fabric operations of ordering and validating are decoupled. Therefore, the gossip protocol intends to broadcast the results after the ordering phase efficiently. Additionally, it is accountable for the transmission of those results to peers that recently joined or were disconnected, providing an enhancement of integrity and consistency to its data.

2.1.4.B Transaction Flow

When a Fabric network is set up, running, and prepared to receive transactions inside a channel containing multiple organizations and with a chaincode deployed, it follows a set of procedures to perform transactions successfully.

Transactions are performed according to endorsement policies that specify which peers can execute them in the network [18]. Therefore, according to those policies, the one who desires to transact data must create a transaction proposal by invoking a request to the chaincode functions using a proper SDK for a supported programming language in a client application. This proposal initiates the whole process of updating or reading the ledger and attached to it, the user's network credentials are taken to sign the proposal. Afterwards, the transaction goes through the procedures contained in the *execute-order-validate* model to be accepted and added to the ledger.

For a transaction to be executed, firstly, validations are carried out by endorser peers to verify that the submitter's identity is valid and the data is not compromised nor duplicated. If the endorsement policies are well satisfied the desired chaincode's function is executed against the chosen state database. A state database is present in every peer that contains the written value of the assets stored in the ledger. Fabric supports two types of state databases, LevelDB and CouchDB. Table 2.2 shows a comparison between both databases (Section storage will cover the decisions on this matter to develop the presented system).

	LevelDB	**CouchDB**
Option	Default Embedded in the peers	Alternative and external
Storage Type	Any binary data	Any binary Data
Chaincode Operations	Simple	Complex
Querying	Simple Key-pair	Simple and rich queries (JSON queries)
Performance	More efficient	Less efficient
Latency	Less latency	More latency

Table 2.2: Hyperledger Fabric state databases comparison: LevelDB vs CouchDB

At this point, if the transaction aims to update the ledger, then it is inspected and redirected to the ordering service. As a consequence, when this service gathers proposals it orders them in chronological order by channel, creating blocks to be submitted to all peers belonging to those channels. Nevertheless, in case the transaction proposal is a query that intends to read the ledger, it doesn't need to be broadcasted to the ordering service. After the ordering service ends its procedure, the validation process begins on the created blocks containing transactions to check whether the endorsement policies are satisfied. Upon this process is finished successfully with the proposal being validated, the transaction is committed to the channel's ledger so that each peer contains the update registered in its state database and ledger. Lastly, the client application, which was the source of the transaction request, receives a message depending on the result of the transaction.

2.1.5 Blockchain-based Multimedia Authentication

Blockchain-based solutions in the context of multimedia authentication have been an early adoption in recent years. As a result, there are not many papers that address this issue with this technology and the ones applied only for the news content use case are even fewer.

In the paper by Alattar *et al.* [9], a solution to protect news-related videos from being manipulated is proposed. It focuses on building a system based on watermarking the audio and video using appropriate technologies to detect deepfakes made with face-swaps and lip-syncs. It also uses blockchain technology as storage for information about the videos that are relevant for forensic analysis, and the distributed storage platform (the InterPlanetary File System (IPFS), which will be described in later sections) to save the video data. This solution presents an effective way to provide security measures to news videos, ensuring protection against the ”copy attack”, an attack that provides watermarking extraction from one video to another. It uses a proof-of-authority-based Ethereum blockchain network which is a public permissionless network. 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 H.R. Hasan *et al.* [2] has the main goal to provide traceability and provenance to creators' or publishers' digital content by producing a system operating with a distributed ledger platform. Using the Ethereum blockchain network, a smart contract written in Solidity language is created and attached to a video file, audio file, or any other type of content. This program in the blockchain contains rules to enable requests to manage, share and restrict content from other users. A remarkable feature the authors decided to include in their work is the Ethereum Name Service. It is used to manage the identity of a content's owner by linking the Ethereum address of an asset stored in the blockchain to human-readable texts containing relevant information to describe the publisher or creator. Creators have an associated profile in the system, which is connected to a decentralized reputation system that manages reviews and ratings given by other users to build trust. Additionally, the authors utilize an IPFS, a decentralized storage system to save the video content and metadata in JSON format, providing a fully decentralized system. In Fig. 2.2 it is possible to observe this paper's proposed solution architecture and understand the different roles of each component. In future work, the authors proposed to enhance their reputation system and planned to develop DApps in the context of video protection to automate traceability in web browsers and proof-of-authenticity consensus to digital content. Similarly to the previous work, it uses a state-of-the-art architecture for preserving digital content integrity and authentication. Since the Ethereum blockchain is a public network where anyone can join and update the ledger, the built reputation system to build trust in the stored content is noteworthy. Even though private blockchains do not require this kind of approach to achieve trust in the content, due to having only allowed entities joined in the network, it is a powerful feature to enhance the trust if needed, for instance, to rate the content each organization stores in the ledger.

The paper provided by Rashid reflects on the current deepfake detection and creation background, explores ongoing research related to blockchain digital content protection and presents a framework to prevent deepfakes by using smart contracts, IPFS distributed storage and Ethereum blockchain. In

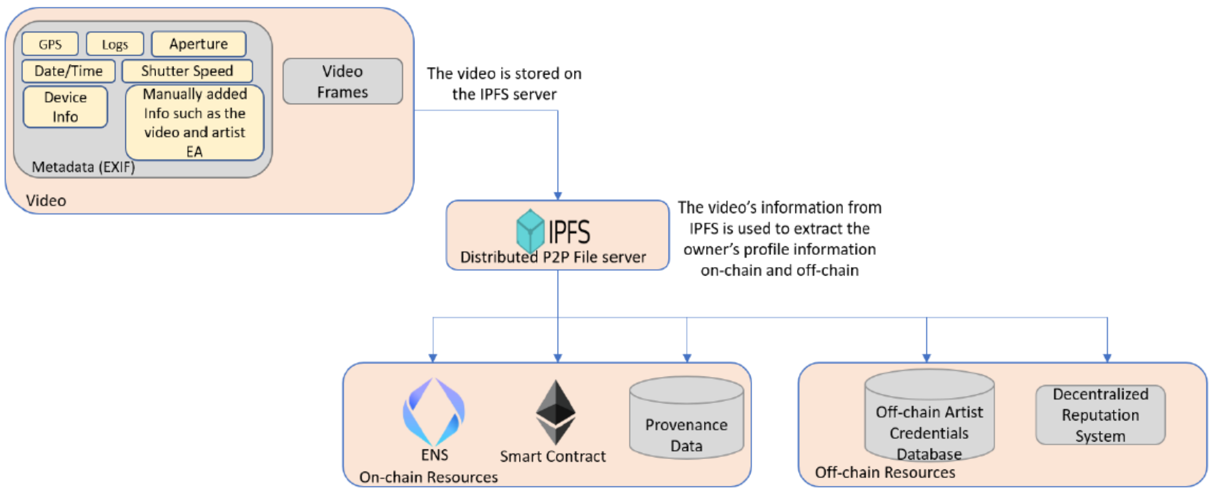

GPS
Logs
Aperture
Date/Time
Shutter Speed
Device Info
Manually added Info such as the video and artist EA
Metadata (EXIF)
Video Frames
Video
The video is stored on the IPFS server
IPFS
Distributed P2P File server
The video's information from IPFS is used to extract the owner's profile information on-chain and off-chain
ENS
Smart Contract
Provenance Data
On-chain Resources
Off-chain Artist Credentials Database
Decentralized Reputation System
Off-chain Resources

Figure 2.2: Hasan *et. al* [2] proposed solution architecture

Figure 2.3 the architecture of this solution is described, and we can notice how the state-of-the-art components communicate with each other. This project focuses entirely on securing video and image copyright integrity and malicious dissemination. As a result of this work, the conclusions do not differ from the previously mentioned papers since the developed solution is quite similar.

Regarding blockchain-based journalism systems, the solution described in the paper by Teixeira *et. al* [19] improves crowdsourcing journalism by providing a decentralized marketplace targeting live-made news videos created by citizens, allowing them to collaborate with news organizations in a transparent system. For that purpose, the marketplace had to be built on the organization-level permissioned blockchain Hyperledger Fabric to ensure trust in the shared data by granting permissions only to a set of individuals. This paper contributed to enhancing the known news systems by creating a new way to make a profit by distributing news videos and rewarding those who share the content. It also proves that journalism organizations can benefit from permissioned blockchains to secure their content and provide trustworthy sources of information. The contribution provided by this paper is an example of a similar system used for a different use case for a common goal, which is to protect news videos. This solution could be complemented with the proposed one in this thesis since it is an excellent example of a different use case that would benefit from a manipulation detection system that provides video integrity validation.

2.2 Decentralized Storage

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 mech-

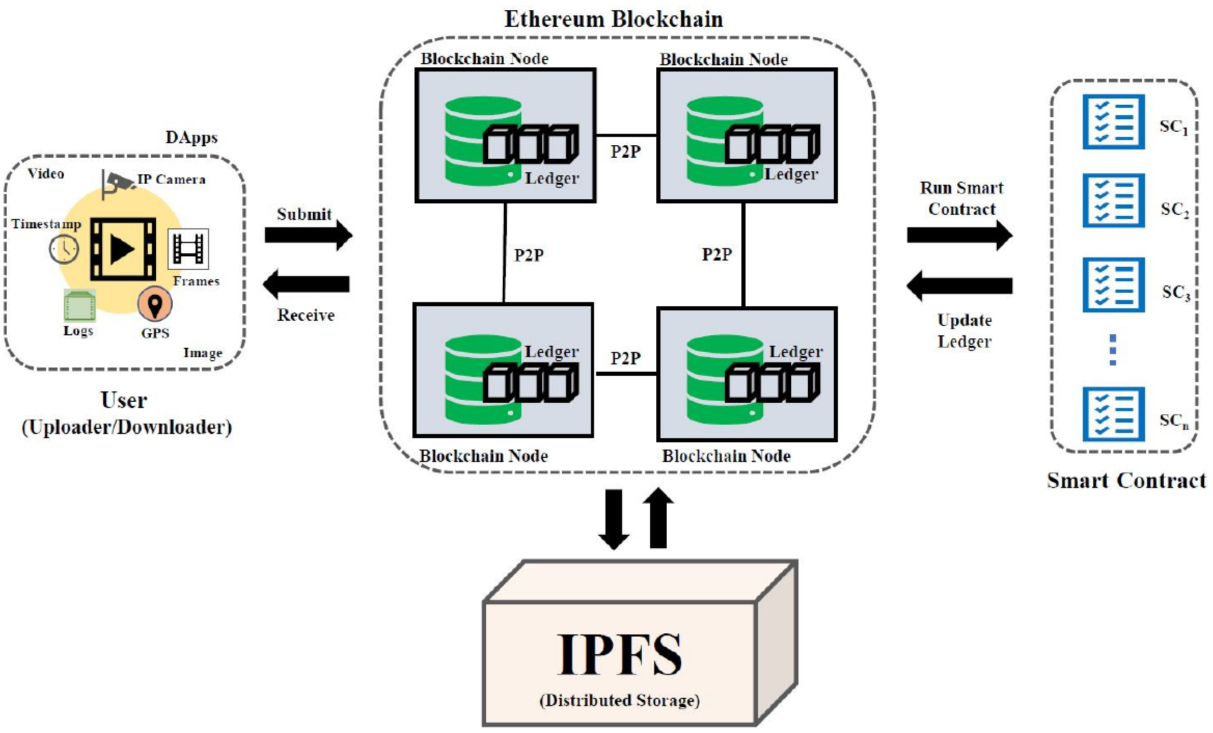


Figure 2.3: Rashid *et. al* [3] proposed solution architecture

anisms 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 ”that seeks to connect all computing devices with the same system of files without the need for nodes to trust each other” [20]. It uses a Merkle Directed Acyclic Graph (DAG), a junction between Merkle Tree and DAG. It is composed of nodes with identifiers formed by hash functions of the content using the SHA256 cryptographic function. Similarly to Merkle Trees, which will be covered in the next section, it provides the ability to check whether a change has occurred within this data structure since any change in a parent node of the graph would alter all the children nodes' content producing a different Merkle DAG.

When a file is uploaded to the IPFS network, a Content Identifier (CID) that corresponds to the stored address is generated so that it can be retrieved. The CID is a unique hash of the file in the Merkle DAG, hence the owner of that resource, upon querying to IPFS can verify if it was tampered with by checking whether the hash matches with the one received. Also, when a file is uploaded to the network it is only created once, meaning that it can not have duplicates of that resource ensuring efficiency to the network. This feature is called deduplication, so once a file is uploaded, it cannot be changed granting immutability to the stored content. A resource is stored in an IPFS object, which can take up to 256kB of data. If a certain file is larger than this limit, it is partitioned into smaller parts and a single IPFS object is created containing the addresses of every single part of that file.

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 the nodes maintaining it break down. Filecoin [5] brings up a possible solution for this problem. It is a decentralized storage that runs on top of IPFS that aims to provide more availability to its data by granting crypto-economic incentives. Clients provide storage to other people by making deals, while the network verifies the correctness of the storage process and rewards the client with Filecoin cryptocurrency during the period of that storage deal.

2.3 Merkle Tree

In the previous section, Merkle Tree was mentioned in the scope of IPFS in which it was complemented with DAG. Created by the famous computer scientist Ralph Merkle [21], 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.

2.3.1 Merkle Tree-based Multimedia Authentication

Some research works have concluded that Merkle Trees can also be used to authenticate and secure multimedia content. Furthermore, this data structure integrated with a blockchain architecture has revealed an improvement in integrity verification in multimedia.

The paper by Chen *et. al* [22] presents a scheme to verify tampering in images by revealing and restoring the manipulated pixel area. It uses the same architecture as the other blockchain systems from the literature, but with the usage of Merkle trees, it provides reliable storage and verification. The Merkle tree in this work is generated by taking the image's most-significant bit of each pixel of the original

[5] https://docs.filecoin.io/about-filecoin/what-is-filecoin/

image and then creating a new one. The new image is sliced to a non-overlapping block and each pixel is encrypted with to be stored in the IPFS. The hash computed by IPFS when the file is stored is used to build the Merkle tree as it corresponds to one of the leaf nodes. By storing the built Merkle tree containing all IPFS fingerprints for each uploaded file this system can ensure improved integrity of its data. This data structure can identify the modified stored files and the tampered area through the ciphered image blocks. This system is focused on protecting image content rather than videos which could introduce performance issues due to having to deal with every frame in the video.

The paper by England *et. al* [23] presents AMP, a system that makes sure that media is authenticated by validating provenance. For each media instance that a provider uploads, AMP creates publisher-signed manifests, which are then stored in a database so that programs like browsers may efficiently search for them. 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. CCF maintains a private key that the service guards and occasionally uses it to sign the Merkle root in the distributed ledger to make sure that any manipulation can be discovered. 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* [24] 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 [25]. 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. For this purpose, this type of hashing algorithm and other ones that can only assess the actual content of a multimedia file, such as robust hashing and deep hashing described in papers [26] and [25] respectively, must be tested in future work for systems like the one presented in this thesis.

2.4 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.* [27] presents a survey on deepfake video detection 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 work that explores these technologies, also included in the Yu *et al.* [27] work, is the one by Li and Lyu [28]. 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.

The paper by Chan *et. al* [16] proposes a framework that combines the use of blockchain security with multiple Long Short-Term Memory (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.

In the research performed in the paper by Qureshi *et. al* [29], a proof-of-concept for detecting deep-fakes is shown that can identify voice-impersonation-produced fake news video clips. The suggested scheme employs a hybrid speech watermarking technique to implant digital watermarks in a video's audio track. The implanted watermark's resistance to typical signal processing and video integrity threats is assessed through simulations in this work.

3 Design & Architecture

Contents

This chapter 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' requirements. To specify those requirements they were split into two categories: Functional Requirements (FRs) and Non-Functional Requirements (NFRs). The FRs relate to the technical functionality of the system, whereas NFRs correspond to a set of key drivers, which were necessary to follow as guidelines to produce a feasible solution. Then, the chapter 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.

3.1 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 NFRs and FRs. 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.

Table 3.1 shows the functional requirements the implemented solution should meet with its developed functionalities. The use cases that, will be discussed later in this Chapter, 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 3.1: Functional Requirements

3.2 Technology Decisions

Considering the research presented in Chapter 2 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 3.2.

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

Table 3.2: Technology decisions made for this thesis developed solution

3.2.1 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.

By comparing the qualities of each database in Table 2.2, 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.

3.2.2 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

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

audiovisual content and is used to find common content in videos [30]. Figure 3.1 shows an example of a video frame signature extracted to a *.xml* file containing relevant data to uniquely define a frame. 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.

```
<VideoFrame>
  <MediaTimeOfFrame>0</MediaTimeOfFrame>
  <FrameConfidence>27</FrameConfidence>
  <Word>7  233  218  132  40 </Word>
  <FrameSignature>  1  0  0  0  1  1  0  0  0  0  1  1  0  0  0  0  0  1
  1  0  1  0  1  0  0  0  1  1  0  0  0  0  0  0  2  0  2  2  2  1  0  2
  2  1  2  1  1  1  0  2  2  2  0  2  2  2  2  2  1  1  0  0  1  1  2  1
  0  2  1  1  2  2  2  2  0  0  2  0  1  0  1  1  2  1  1  1  1  2  2  1
  0  1  1  1  0  2  0  0  0  2  1  0  1  0  0  0  0  2  1  1  1  1  1  1
  1  1  1  1  1  1  2  0  0  0  2  0  0  0  1  1  1  1  2  1  1  1  2  2
  0  2  1  1  2  0  1  1  1  1  2  0  2  0  0  2  0  2  2  2  1  1  0  0
  2  1  2  0  1  1  1  1  1  0  1  2  1  2  2  2  1  2  2  0  2  2  1  1
  1  2  2  2  2  2  2  2  1  1  0  0  1  1  0  0  1  0  0  1  1  2  2  1
  0  2  1  1  0  2  1  0  2  0  2  0  1  1  1  2  1  0  2  1  2  0  2  1
  2  2  2  0  0  0  0  0  1  1  2  0  0  1  2  1  2  1  1  0  2  2  2  1
  1  2  2  1  0  1  0  2  2  0  2  2  2  2  2  2  2  1  0  0  2  1  2  2
  0  2  2  1  0  0  1  1  1  1  1  2  2  2  1  1  2  0  1  0  2  0  1  1
  0  2  2  1  0  0  2  0  2  0  0  0  0  0  1  2  1  2  2  2  0  2  0  2
  1  1  1  1  1  1  1  2  1  2  1  2  1  2  1  2  1  1  1  1  0  0  2  0
  0  0  2  0  1  0  0  2  1  1  0  0  2  2  0  0  2  1  1  1  2  1  2  2
  0  1
  </FrameSignature>
</VideoFrame>
```

Figure 3.1: A MPEG-7 video frame signature extracted to a *.xml* file using FFMPEG

3.3 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.

Figure 3.2 shows the decomposition style view of the developed system representing the modules created to implement the solution.

- **Module *application***: Contains the Application Programming Interface (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.

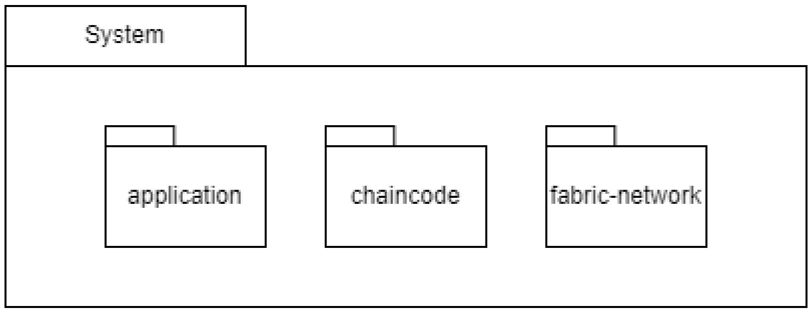


Figure 3.2: System's architectural decomposition view

Figure 3.3, 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 content and other data crucial to perform the use cases that will be presented in Section 3.4. 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.

3.4 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.

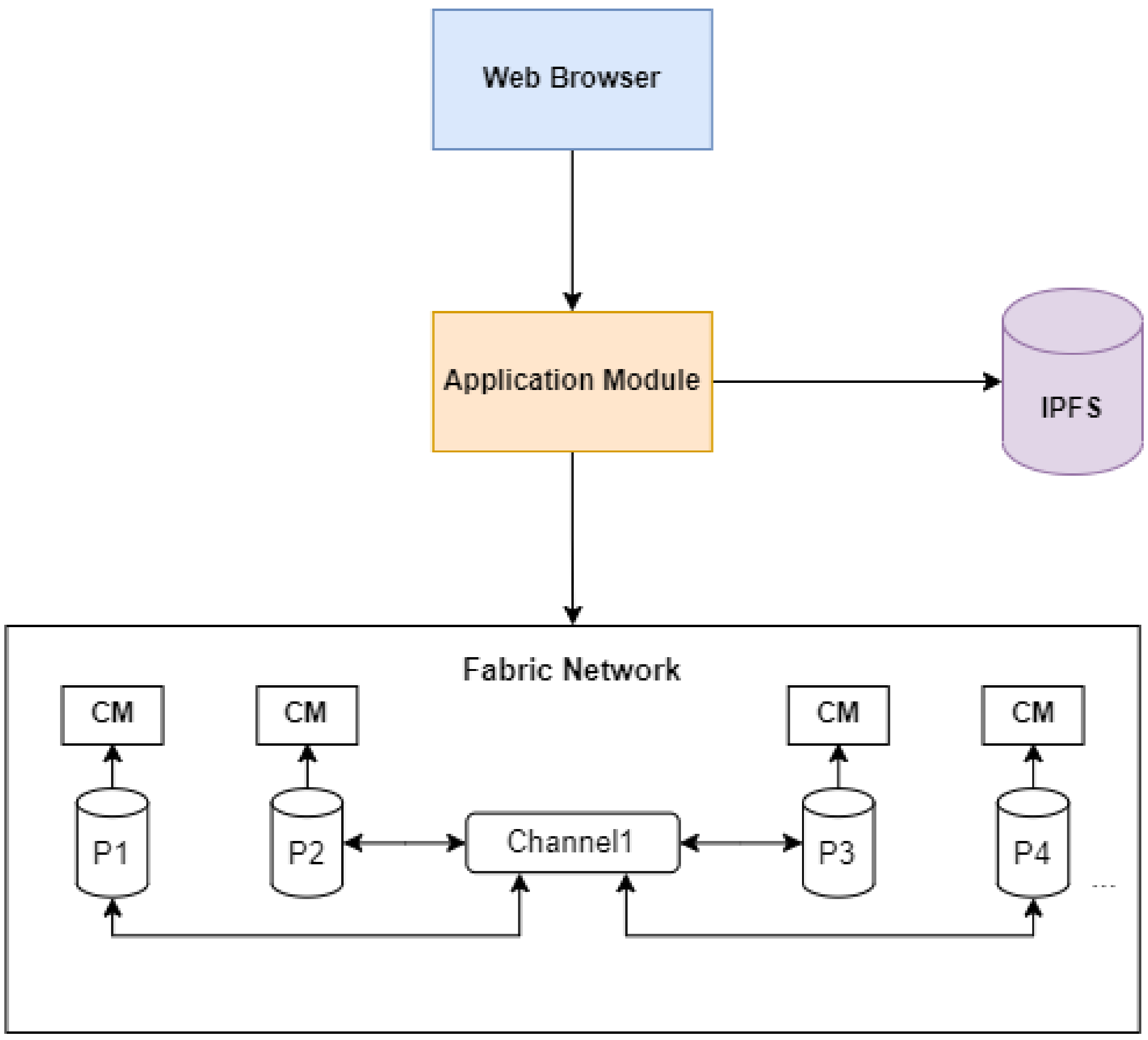


Figure 3.3: System's components architecture view

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 following figures display in detail the features each user can perform. Figure 3.4 describes the *Submit* use case, which only a trusted user can perform. This functionality 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 trusted user submits a video to the application using the *submitVideo* function, a unique ID for the file is generated, *fileId*. The *fileId* creation will be explained later in Section 4. 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*.

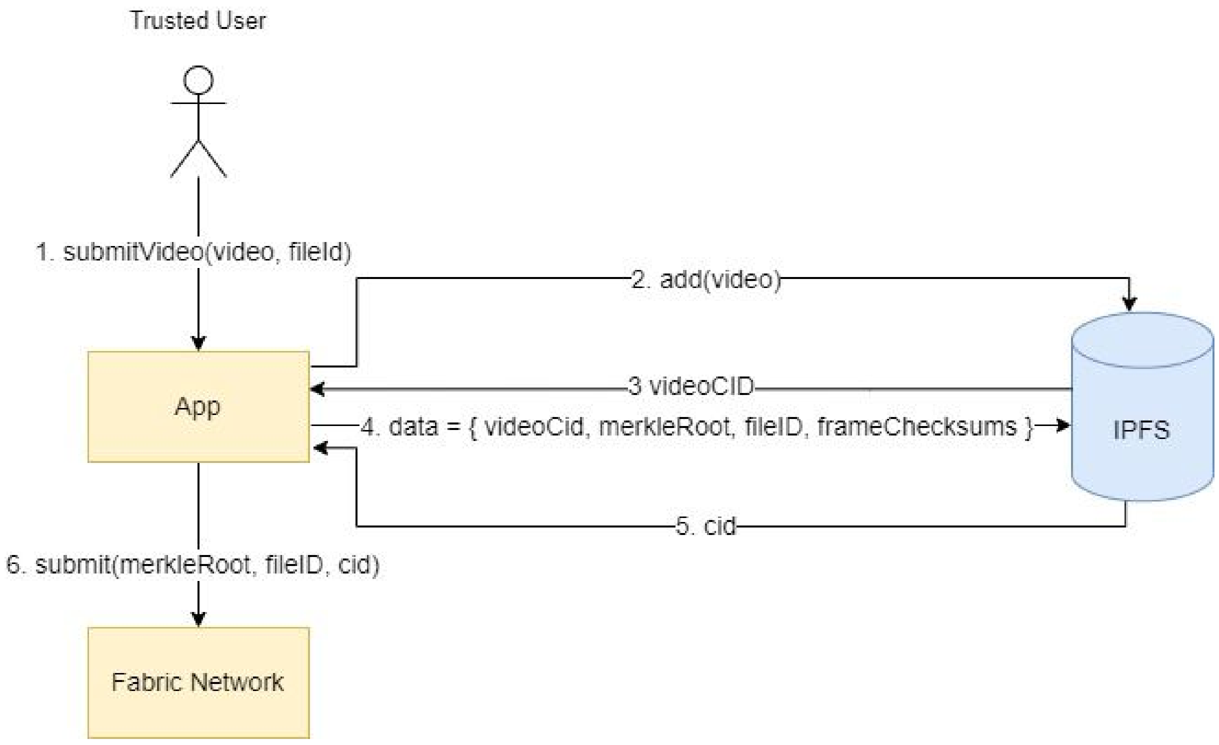

Trusted User
1. submitVideo(video, fileId)
2. add(video)
App
3 videoCID
4. data = { videoCid, merkleRoot, fileID, frameChecksums }
IPFS
5. cid
6. submit(merkleRoot, fileID, cid)
Fabric Network

Figure 3.4: *Submit* trusted video use case

Figure 3.5 presents another use case targeted for trusted users. 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.

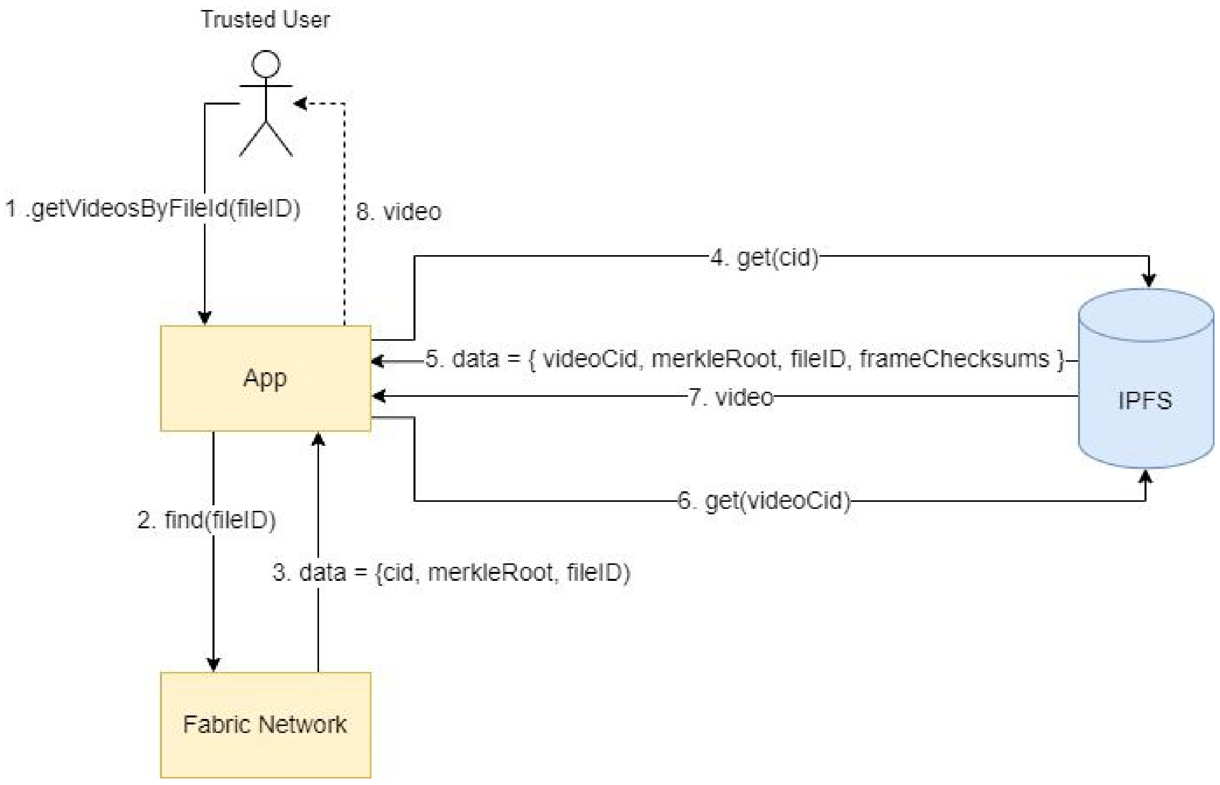

Trusted User
1 .getVideosByFileId(fileID)
8. video
4. get(cid)
App
5. data = { videoCid, merkleRoot, fileID, frameChecksums }
7. video
IPFS
6. get(videoCid)
2. find(fileID)
3. data = {cid, merkleRoot, fileID)
Fabric Network

Figure 3.5: *Get* trusted video use case

Figure 3.6 shows the use case intended for untrusted users. The *Detection* use case consists of an upload of an unverified video to evaluate its authenticity. Using the *Detect* function, common internet users can upload 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 bottom 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.

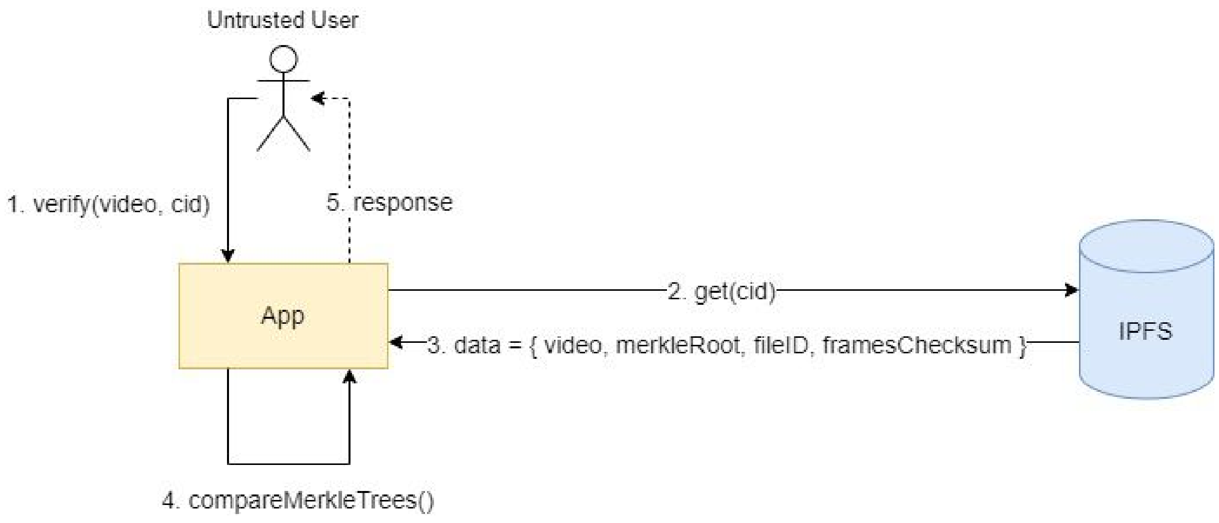


Figure 3.6: *Detect* video fraud use case

3.5 Data Model

This section enlightens the data structures involved while performing the previously presented use cases. Figure 3.7 exhibits the tables containing information that is stored in the ledger or in IPFS.

- ***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 3.4. 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.

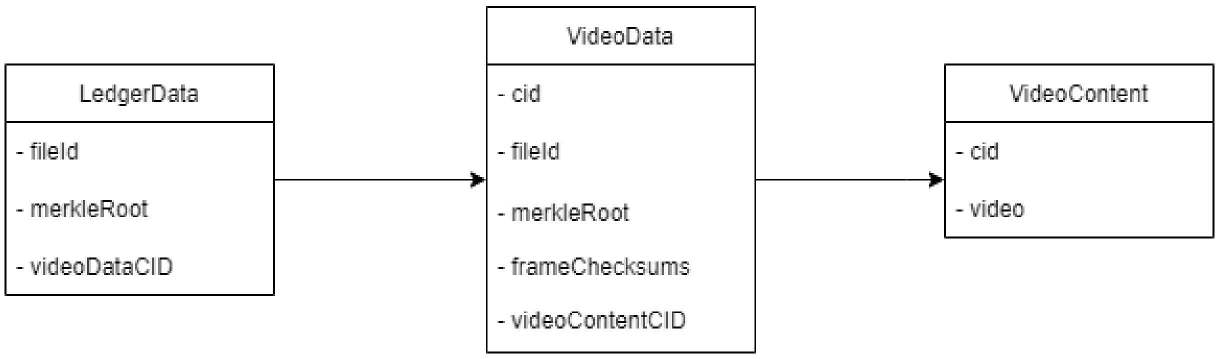


Figure 3.7: System's data model

4 Implementation

Contents

This chapter 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.

4.1 Application module

As it was mentioned before in Chapter 3, the application module contains the API that allows users to perform the system's functionalities. Figure 4.1 defines the folder structure containing the relevant directories that comprise this module. There is an ***api*** folder containing the endpoints and application logic for each user's features. The ***trustedUser*** folder is composed of *.js* files that allow the realization of the *submit* and *getVideo* use cases, whereas the ***untrustedUser*** directory covers the *detect* use case. The directory named ***helper*** has all the files that provide tools to help with the API implementation and interface with other components. The folder ***models*** contains the data structures used to implement the Merkle tree for the videos. When users are enrolled on the Hyperledger Fabric network, their identities are stored in a ***wallet*** folder to be used for the functionalities' purpose.

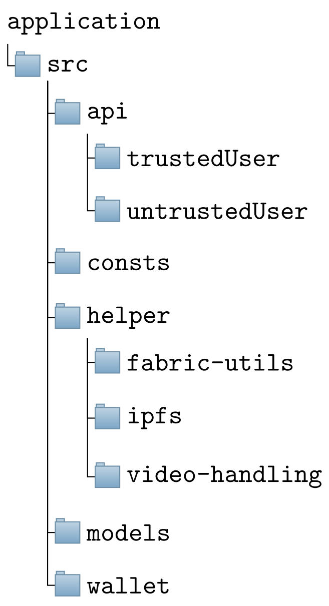


Figure 4.1: Application module's directory structure

4.1.1 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 command below takes an input video and extracts the whole video frames' signature to a *.xml* file.

```
$  ffmpeg -i inputVideo -vf \
signature=detectmode=full:format=xml:filename=outputDirectoryPath/signature.xml \
-map 0:v -f null - \
```

The MPEG-7 video signature, as explained in Section 3.2.2, 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 [31]. 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:

Algorithm 4.1: Merkle Tree creation algorithm

```
Input : $frameHashList$
begin
1   $merkleRoot \longleftarrow []$
2   $nextLevelNodes \longleftarrow ComputeBottomLevel(frameHashList)$
3   $treeHeight \longleftarrow 2$
4   while $nextLevelNodes.Length > 1$ do
5       $nextLevelNodes \longleftarrow ComputeMiddleLevel(nextLevelNodes, treeHeight)$
6       $treeHeight \longleftarrow treeHeight + 1$
7   if $nextLevelNodes.length == 1$ then
8       $merkleRoot \longleftarrow nextLevelNodes[0]$
```

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.

4.1.2 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.

4.1.3 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 for validation, 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 using the Algorithm 4.1 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. The Algorithm 4.2 is performed to check whether this possibility might be confirmed, using the video frames retrieved from the IPFS.

Algorithm 4.2: Trusted frame hash list re-scaling algorithm

```
   begin
1      $index \longleftarrow 0$
2      if $trustedFirstHash \neq untrustedFirstHash$ then
3          $index \longleftarrow trustedHashes.IndexOf(untrustedFirstHash)$
4          if $index == -1$ then
5              return
6      $trustedHashes \longleftarrow Slice(index, index + untrustedHashesLength)$
7      if $trustedHashes == untrustedHashes$ then
8          return
9      else
10         $trustedMerkleTree \longleftarrow BuildMerkleTree(trustedHashes)$
```

The Algorithm 4.2 displays that 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 goes as described in Algorithm 4.3.

Algorithm 4.3: Merkle tree comparison algorithm

```
Input   : $trustedTree, untrustedTree$
Output: $untrustedFrameNumber$
begin
    if $untrustedLeftNode == undefined$ then
        return $untrustedFrameNumber$
    else
        if $untrustedRightNode == undefined$ then
            return $untrustedFrameNumber$
    if $trustedLeftHash \neq untrustedLeftHash$ then
        return $CompareMerkleTrees(trustedLeftNode, untrustedLeftNode)$
    else
        return $CompareMerkleTrees(trustedRightNode, untrustedRightNode)$
```

The Merkle tree comparison focuses on searching for the first element in the untrusted tree that is different from the trusted one. The Algorithm 4.3 ensures that the trees are traversed equally from top to bottom whenever a node is different. This algorithm 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.

4.2 Chaincode module

This module contains the main JavaScript file with the contract necessary to install in the Fabric network's peers and submit transactions. The folder structure of this module is presented in Figure 4.2.

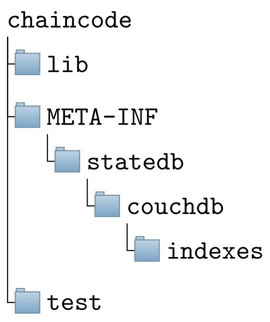


Figure 4.2: Chaincode module's directory structure

Using the Fabric SDK for NodeJS, a class *VideoData* is created, which extends the *Contract* class provided by the *fabric-contract-api* NodeJS package. This class is present in the **lib** folder and contains a variety of functions (Figure 4.3) 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 [18].

```
class VideoDataContract extends Contract {

  async SubmitVideoData(ctx, merkleRoot, fileID, cid) { ... }

  async QueryVideoDataByFileID(ctx, fileID) { ... }

  async VideoDataExists(ctx, id) { ... }

  // Internal functions
  async _GetAllResults(iterator, isHistory) { ... }

}
```

Figure 4.3: The functions available in the Chaincode

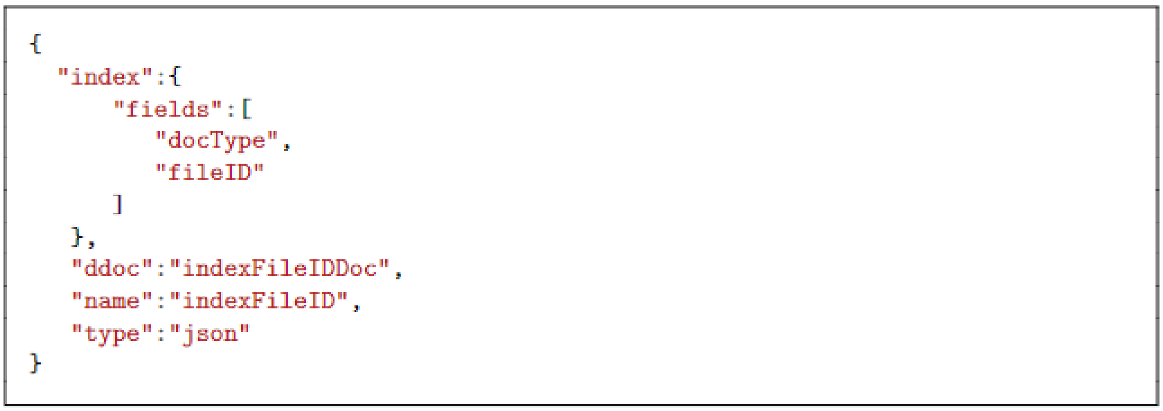
The trusted user interacts with two distinguished functions present in the chaincode deployed in the channel, which are run throughout the execution of the features intended for them. The first function is ***Submit(ctx, merkleRoot, fileID, cid)***. This function is used in the *submit* use case when 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. Before implementing this function, it is necessary to create an index so that the CouchDB query can be performed more efficiently. To do this, a *.json* file must be created, Figure 4.5, to allow those types of database queries. The ”index” JSON object has two attributes, in addition to the index ”name” and ”type”, which are the other two pertinent ones: The ”docType” attribute distinguishes the various sorts of stored objects in the world state, and ”fileID” is the field that will be queried. The *”ddoc”* element is a best practice that enables updating the index and specifying the index to use on a query. 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

```
class VideoDataContract extends Contract {

    ...

    async Submit(ctx, merkleRoot, fileID, cid) {
        const exists = await this.VideoDataExists(ctx, merkleRoot, cid);
        if (exists) {
            throw new Error(`The video data with merkle root hash ${merkleRoot}"
            + " already exists`);
        }

        let id = merkleRoot;
        let videoData = {
        docType: "videoData",
        merkleRoot: id,
        fileID: fileID,
        cid: cid,
        };

        await ctx.stub.putState(id, Buffer.from(JSON.stringify(videoData)));

        let indexName = "merkleRoot~fileID";
        let indexKey = await ctx.stub.createCompositeKey(indexName, [
        videoData.merkleRoot,
        videoData.fileID,
        ]);

        await ctx.stub.putState(indexKey, Buffer.from("\u0000"));
        return JSON.stringify(videoData);
    }

    ...

}
```

Figure 4.4: Chaincode's *Submit* 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.



```
{
  "index":{
      "fields":[
         "docType",
         "fileID"
      ]
   },
   "ddoc":"indexFileIDDoc",
   "name":"indexFileID",
   "type":"json"
}
```

Figure 4.5: The *indexOwner.json* file stored in the Chaincode module

```
class VideoDataContract extends Contract {

...

  async QueryVideoDataByFileID(ctx, fileID) {
    let queryString = {};
    queryString.selector = {};
    queryString.selector.docType = "videoData";
    queryString.selector.fileID = fileID;

    let resultsIterator =
        await ctx.stub.getQueryResult(JSON.stringify(queryString));
    let results = await this._GetAllResults(resultsIterator, false);

    return JSON.stringify(results);
  }

...

}
```

Figure 4.6: Chaincode's *QueryDataByFileID* function

5

Evaluation

Contents

In this Chapter, the evaluation of the implemented system is presented. Primarily, the assessment methodology is demonstrated and then the results are shown. This Chapter ends with a discussion and analysis of the achieved results.

5.1 Evaluation Methodology

The assessment methodology for this thesis is divided into three categories:

- **Soundness Evaluation**: The purpose of this evaluation method was 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.
- **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;
- **Requirements Evaluation**: To conclude the evaluation process the decisions made throughout the development of this thesis are evaluated to check if the system fulfils the requirements. We assessed all FRs, and for NFRs, we evaluated everything we could with the prototype implementation, but NFR-1 and NFR-3 require a deployed system in the wild;

Every experiment was performed using a desktop computer with an Intel i5-8600k CPU 3.60GHz and 16GB 3000MHz RAM, and the application was run inside a virtual machine. The Hyperledger Fabric network tested for this environment was formed by two organizations *Org1* and *Org2* with a created channel shared between each one. This setup was chosen to build confidentiality between the two news organizations. Moreover, each peer present in this channel contains a CouchDB as a world state database. Every video of the sample was taken from the *YouTube* platform and each one has the same dimensions with a frame rate of 30 frames per second. For these experiments, it was assumed that every aspect of the input video file uploaded to be validated by an untrusted user is identical to those stored in the system by trusted users, except for the potentially altered frames.

5.2 Soundness Evaluation

The effectiveness of the system to identify video tampering is examined and discussed to conclude whether the *Detect* feature complies with the functional requirements stated in Chapter 3.

Before describing the evaluation procedure, it is necessary to first present the sample for this experiment, which aims to validate if the various requirements of this functionality are met. Furthermore, a user of this system may wish to compare a video found on an untrustworthy internet platform with the same content of a video already stored in the system. There are also some cases where the beginning of the video was deleted but the rest of the content was not, which is referred to as a ”clip” in this context. The suspicious video, on the other hand, could have been maliciously altered using ”frame swapping” or subtle cuts in between to distort the message it contains. Table 5.1 illustrates the outcomes that this functionality aims to achieve by accepting a variety of input values.

Input-ID	FR-ID	Input video’s description	*Detect* Result
I-1	FR-3.1	The same content as a system-stored video	Trusted
I-2	FR-3.2	A ”clip” containing all content present in the system with no editing	Partially Trusted
I-3	FR-3.3	An entire video with the content present in the system but with some type of editing in between	Not-Trusted
I-4	FR-3.3	A ”clip” with the content present in the system but with some type of editing in between	Not-Trusted
I-5	FR-3.4	A completely different video or ”clip”	Not-Evaluated

Table 5.1: *Detect* functionality input types

Beginning with **I-1**, when the application receives a video as input, it computes its Merkle tree after retrieving from the IPFS the required trusted one for the comparison. The validation result, which is depicted in Figure 5.1, found that the input is authentic because both Merkle root hashes are equal.

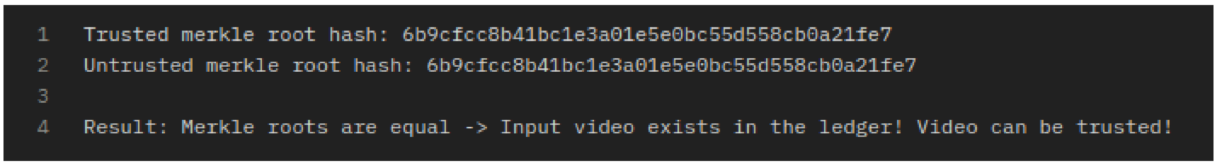
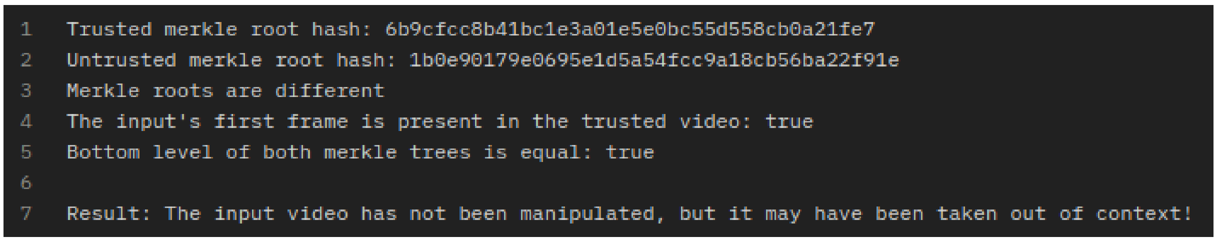

```
Trusted merkle root hash: 6b9cfcc8b41bc1e3a01e5e0bc55d558cb0a21fe7
Untrusted merkle root hash: 6b9cfcc8b41bc1e3a01e5e0bc55d558cb0a21fe7

Result: Merkle roots are equal -> Input video exists in the ledger! Video can be trusted!
```

Figure 5.1: I-1: Output when the input is trusted

I-2 denotes the case in which the input video was taken from a stored one in the system and was not modified. As a result, when the untrusted Merkle tree is constructed, the resulting Merkle root hashes differ. Therefore, the application performs an integrity check by comparing the bottom level of both trees, and as shown in Figure 5.2, both bottom levels are identical. The result of **I-2** is that the input cannot be completely trusted because there may have been malicious intent by removing the ”clip” from its context, resulting in misinformation.

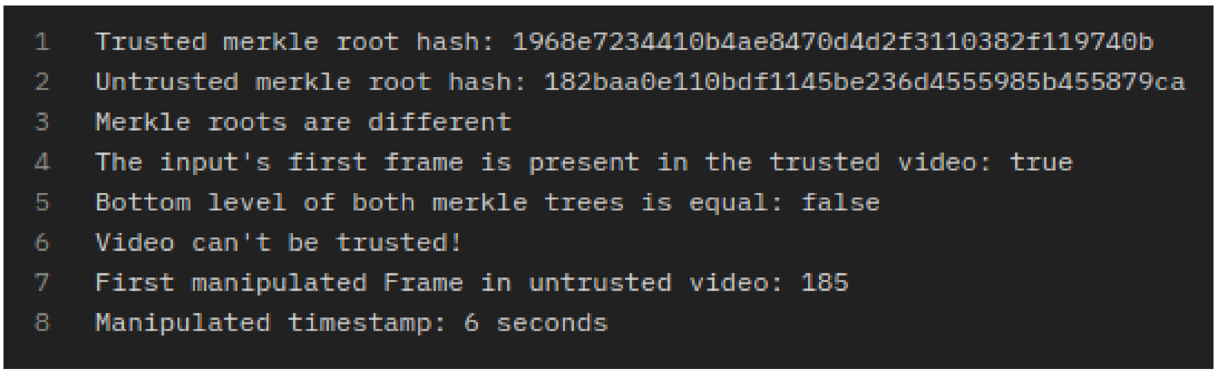
Another requirement mentioned in Table 5.1 and one of the main ones in this thesis is to provide a system that allows users to detect unauthenticated videos. As a result, the previously mentioned real-world example of manipulation in news videos was recreated to perform the **I-3** [8]. Assuming that a similar Joe Biden speech video is already in the system, the application must prompt that it is an



```
Trusted merkle root hash: 6b9cfcc8b41bc1e3a01e5e0bc55d558cb0a21fe7
Untrusted merkle root hash: 1b0e90179e0695e1d5a54fcc9a18cb56ba22f91e
Merkle roots are different
The input's first frame is present in the trusted video: true
Bottom level of both merkle trees is equal: true

Result: The input video has not been manipulated, but it may have been taken out of context!
```

Figure 5.2: I-2: Output when the input is partially trusted

unreliable one containing manipulated information by providing as input one that was edited exactly like Fox News did (cutting in between). To achieve this goal, after computing the Merkle trees and confirming that both root hashes differ, the bottom levels are compared. However, the list of hashes corresponding to that tree level on the input video is different this time. As a consequence, a new Merkle tree of the same size as the one created for the input is computed for the trusted video, and the Algorithm 4.3 from Chapter 4 is run. The output of this process indicates to the user the first encountered manipulated frame, which is frame number 185 of the input in Figure 5.3.

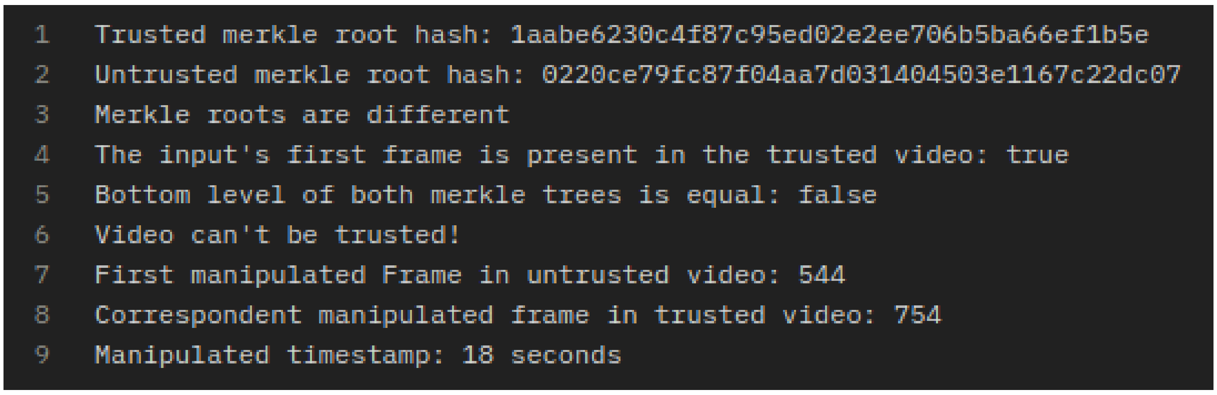


```
Trusted merkle root hash: 1968e7234410b4ae8470d4d2f3110382f119740b
Untrusted merkle root hash: 182baa0e110bdf1145be236d4555985b455879ca
Merkle roots are different
The input's first frame is present in the trusted video: true
Bottom level of both merkle trees is equal: false
Video can't be trusted!
First manipulated Frame in untrusted video: 185
Manipulated timestamp: 6 seconds
```

Figure 5.3: I-3: Output when the input is present in the system but contains editing in between

I-4 is similar to **I-3**, except that the input is a "clip" taken from a video stored in the system. This is a common type of misinformation spread through untrustworthy information sources such as social media platforms. People typically post "clips" with cuts or other types of editing in between. To address this issue, the application follows the same steps as **I-3**. When it uses the Merkle tree comparison to find the first manipulated frame, it returns the first manipulated frame number in the untrusted video, as well as the correspondent frame number in the trusted video where the manipulation occurred, Figure 5.4.

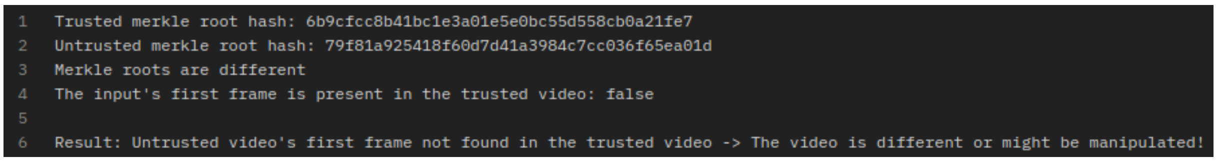
The detection feature's primary goal, as previously stated, is to identify the first occurrence of a manipulated frame. To comply with **I-5**, the system should notify the user if it receives a completely different video or one with a different starting frame. To attain this, the app computes the Merkle tree and notifies the user if the first frame of the manipulated video is missing from the reliable Merkle tree's bottom level. As a result, the output shown in 5.5 guarantees that the video has different content because



```
Trusted merkle root hash: 1aabe6230c4f87c95ed02e2ee706b5ba66ef1b5e
Untrusted merkle root hash: 0220ce79fc87f04aa7d031404503e1167c22dc07
Merkle roots are different
The input's first frame is present in the trusted video: true
Bottom level of both merkle trees is equal: false
Video can't be trusted!
First manipulated Frame in untrusted video: 544
Correspondent manipulated frame in trusted video: 754
Manipulated timestamp: 18 seconds
```

Figure 5.4: I-4: Output when the input is a clip with the content present in the system but contains editing in between

the suspicious first occurrence's starting frame is distinct.



```
Trusted merkle root hash: 6b9cfcc8b41bc1e3a01e5e0bc55d558cb0a21fe7
Untrusted merkle root hash: 79f81a925418f60d7d41a3984c7cc036f65ea01d
Merkle roots are different
The input's first frame is present in the trusted video: false

Result: Untrusted video's first frame not found in the trusted video -> The video is different or might be manipulated!
```

Figure 5.5: I-5: Output when the input corresponds to a different video

5.3 Performance evaluation

In this Section, the latency of the functionalities and the application's interactions with the various system components were tested to determine if the system meets the performance requirements. The functionalities aimed at reliable users are presented first, followed by the results for untrustworthy users. Table 5.2 contains information about the videos submitted for this simulation by trusted users. The sample includes a wide range of videos with varying durations and, consequently, frame amounts.

ID	Duration	Number of frames	File size
A_T	18 s	543	3.93 MB
B_T	3 min 02 s	5471	35.71 MB
C_T	26min 03s	35568	116.39 MB
D_T	1h 00min 28s	108846	414.8 MB
E_T	2h 06min 40s	228000	361.9 MB

Table 5.2: Trusted video samples submitted by news organizations

Table 5.3 displays the results of the *Submit* and *Get video* use cases for each input video in Table 5.2. This table shows that, despite not being low-performing, the MPEG-7 signature has a significant impact on the latency of this procedure in comparison to the other metrics, being directly proportional to the number of frames the input video has. As it was expected, the IPFS *add* and *get* operations for video content take longer as the file size grows.

The measured values that were constant and independent of the input are displayed in Table 5.4. The Merkle tree node values were calculated using the signature's SHA-256 hash, and the more frames there are, the more times a hash must be calculated.

ID	MPEG-7 Signature extraction	Merkle tree creation	IPFS *add* video	IPFS *get* video video
A_T	3.598s	5.066ms	78.757ms	195.108ms
B_T	6.501s	22.896ms	409.963ms	453.787ms
C_T	31.264s	150.694ms	930.746ms	928.185ms
D_T	2min 42s 361ms	380.016ms	2.722s	3.217s
E_T	5min 25s 926ms	719.128ms	2.488s	2.581s

Table 5.3: *Submit* and *Get video* functionalities latency experiment results

Average time for calculating a frame signature SHA-256 hash	Average time for submitting a transaction in *Submit* use case	Average time for querying data from the blockchain in *Get video* use case
0.01995ms	2.2028s	26.0280ms

Table 5.4: Constant measured values during *Submit* and *Get video* functionalities

The first three videos from Table 5.2 were used as the source for the *Detect* latency tests, and Table 5.5 contains information about this experiment's sample videos that have been altered and classified as ”Not-trusted” by the application. Assuming that most people will use this feature to validate small selected pieces of information they found on untrustworthy sources, A_M and B_M represent small videos received as input that reflect those cases, whereas C_M represents a larger input to compare the application's behaviour with a different sized payload.

Name	Frame amount	Manipulation description
A_M	248	The A_T video with a cut in between
B_M	1077	A ”clip” of B_T with a cut in between
C_M	30956	The C_T video with a cut in between

Table 5.5: Input samples to test the latency of the *Detect* functionality

The results of this experiment are presented in Table 5.6. Similarly to *Submit*, the signature computation is the operation that takes significantly longer in comparison to the other ones. However, the comparison process shows that it only depends on the height of the Merkle tree instead of the number of frames, N_f, having a time complexity of $O(h)$, being h the height of the tree for every iteration, regardless of the location where the tampered with frame takes place in the bottom of the tree. Upon the creation of a Merkle tree for comparison purposes, the height is given by the following formula with the result being rounding down to the nearest integer:

$$h = \log_2(N_f)$$

Name	Signature extraction latency	Merkle tree height	Merkle tree comparison latency
A_M	1.098s	9	0.015ms
B_M	1.567s	12	0.024ms
C_M	35.270s	16	0.104ms

Table 5.6: Input samples to test the latency of the *Detect* functionality

5.4 Requirements Evaluation

In this section, the requirements from Chapter 3 will be evaluated to determine whether the system's implementation, as well as the various architectural and technological choices made, comply with the requirements. For this purpose, two tables will be presented containing the cross-checked requirements for each type, FRs and NFRs.

Table 5.7 shows that all the requirements were fulfilled with a more detailed description contained in Appendix A for FR-1.x and FR-2.x. Section 5.2 provided a detailed assessment of the FR-3.x, since it is the main functionality of this system.

ID	Status	Reference
FR-1.1	✔	A.1
FR-1.2	✔	A.2
FR-1.3	✔	A.3
FR-1.4	✔	A.4
FR-2.1	✔	A.5
FR-2.2	✔	A.6
FR-2.3	✔	A.7
FR-2.4	✔	A.8
FR-3.1	✔	Section 5.2
FR-3.2	✔	Section 5.2
FR-3.3	✔	Section 5.2
FR-3.4	✔	Section 5.2

Table 5.7: Requirements Verification (Functional Requirements)

Table 5.8 shows that almost all the requirements were fulfilled in their fullness, showing that only **NFR-1** and **NFR-3** were marked with **-** meaning that despite being addressed in the system's design, they required more specific evaluation tests and need further investigation. A more detailed description is contained in Appendix A for all NFRs except NFR-5, which is described in Section 5.3.

5.5 Discussion

By following the assessment methodology performed for the implemented system described in previous sections of this chapter, it is evident that it showed promising results for a novel system that is capable

ID	Status	Reference
NFR-1	-	A.9
NFR-2	✓	A.10
NFR-3	-	A.11
NFR-4	✓	A.12
NFR-5	✓	Section 5.3
NFR-6	✓	A.13
NFR-7	✓	A.14

Table 5.8: Requirements Verification (Non-Functional Requirements)

of detecting tampering from untrustworthy sources.

Starting with the soundness evaluation of the *Detect* functionality, as the results show, the application can perform validations on untrusted input, building trust in the content with the architecture that interfaces with the blockchain used in the implementation of the system. 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.). Therefore, Section 6.2 future work addresses this issue and provides suggestions based on the related work to provide more accuracy with different methods to assess content in videos using the developed architecture for this thesis.

The performance evaluation outcomes show that Merkle trees are viable data structures to be used in video signature hashing for video authentication. Merkle tree creation for a two-hour video with 228000 frames takes less than a second, demonstrating that it has no adverse effects on latency, despite a large number of frames, N_f, it may have. 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 $O(\log_2(N_f))$. As previously shown, 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 in order 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.

To conclude the evaluation discussion for this system, the implementation displayed that almost every requirement was met. 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.

6 Conclusions

Contents

This chapter presents the thesis's conclusions by summarizing the research and its significant contributions. The purpose of this study was to answer the research question, "What are the prerequisites for a blockchain system to be effective in the identification and prevention of untrusted material in news videos?"

Towards addressing the previous question, Chapter 2 provides background information on the fundamentals of blockchain and the characteristics of the most suitable blockchain platforms for this subject. Furthermore, decentralized storages were frequently associated with blockchain solutions that address this subject, which enhance decentralization in systems by removing trust from a central authority and security to extensive data that are unfeasible to store on a blockchain's ledger. The studied Merkle tree use cases outlined their advantages in multimedia integrity verification, and also, artificial intelligence implied to be the standard of sophisticated video manipulation detection. Likewise, in this chapter, the presented related work revealed that the field of video tampering using blockchain is not yet mature but provided a wealth of knowledge on the benefits of employing a blockchain-based system. Much of the literature provided an analysis of the current state of those technologies and solutions, which the approach proposed in this thesis incorporates.

Chapter 3 discussed the system design and architecture, where the architecture of a novel tamper detection system in news-related videos was given and produced using a set of technological choices to satisfy the requirements. After that, it examined the system's use cases and architectural views to attain the objectives of establishing a system capable of building trust and safeguarding news industry content.

Chapter 4 outlined the necessary steps employed to implement the system, which consisted of an application that can interact with a private permissioned blockchain and a decentralized storage system to perform custom actions tailored for the particular use case at hand.

Chapter 5 presents the system's evaluation to verify how well the system performs its features and how efficiently it is. A cross-check of the requirements was also conducted to determine whether the system could meet them. The overall assessment was favourable because it met almost all the prerequisites. The unsatisfied ones were declared to be addressed in the future with more detail. 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.

6.1 Work challenges

During the development of this thesis, some obstacles needed to be surpassed, which made this thesis challenging. Firstly, defining the target users and functionalities addressed to them was a process that was only possible through background research on state-of-the-art video content integrity preservation systems. It was crucial to address this challenge before going through a more developed process because it directly affected the technology choices carried out to fulfil the objectives, mainly in the blockchain platform choice. By including both journalists and citizens as users in a system that builds trust in video content created by those who are trusted, the choice of this technology was soon narrowed to be the most suitable one for the idealized features. Another solved issue was the choice of the fingerprint algorithm that uniquely identifies video content. Some libraries were taken into consideration and tested but revealed unsatisfactory performance results and were discarded for this thesis. Furthermore, this work's purpose is to develop a trustworthy solution where untrusted users may verify content they find on unreliable internet platforms through trusted content created by experts in the news industry. Therefore, it didn't use sophisticated algorithms to detect manipulation with greater precision. After some research, the used digital fingerprint method explained in Chapter 3 was the most suitable one and was revealed to have good performance, which fulfils the system's requirements.

6.2 Future Work

Future work outlined in this section supports the assertions made in the analyzed use case and confirms the broader subject of blockchain-based misinformation detection in news-related videos.

Since the system runs in a single thread to simulate and assess the performance of the worst-case scenario, in the future, a multi-threaded system's implementation would improve the efficiency of the features, which is more suitable to be executed in real-world systems. For instance, parallel execution of video signature computing would enhance the efficiency of that functionality. Moreover, this system requires more assessments of the presented requirements, especially those not completely satisfied, to thoroughly prove it can tackle the current misinformation concern.

The current system would benefit from replacing the video signature that uniquely identifies video frames (MPEG-7 Video Signature) with one more robust and precise, such as the ones stated in Section 2.3.1. As an alternative, artificial intelligence deepfake detection methods would also be suitable for this system since the literature proved to be great combined with the components that compose the presented solution. With this transition, the system could handle any video taken from any internet platform to have its content verified and would enhance it to detect deepfakes.

Additionally, some works from the literature revealed to be complementary to the ones presented in this thesis, as it was stated in Chapter 2. As a result, it would benefit these systems if combined with this work. Furthermore, this solution could incorporate some features from the analyzed related works. A reputation system could be developed in the future to distinguish the most trusted news organizations enrolled in the private blockchain from those that are less trustworthy. This feature would increase the trust of the involved parties within the network, and any user could rate the trustworthiness of organizations.

Bibliography

[1] J. Polge, J. Robert, and Y. Le Traon, "Permissioned blockchain frameworks in the industry: A comparison," *ICT Express*, vol. 7, no. 2, pp. 229–233, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2405959520301909

[2] H. R. Hasan and K. Salah, "Combating deepfake videos using blockchain and smart contracts," *IEEE Access*, 2019.

[3] M. M. Rashid, S.-H. Lee, and K.-R. Kwon, "Blockchain technology for combating deepfake and protect video/image integrity," *Journal of Korea Multimedia Society*, vol. 24, no. 8, pp. 1044–1058, 08 2021.

[4] "Misinformation vs disinformation: Get informed on the difference," last time accessed on October 2022. [Online]. Available: https://www.dictionary.com/e/misinformation-vs-disinformation-get-informed-on-the-difference/

[5] S. Huckle and M. White, "Fake news: A technological approach to proving the origins of content, using blockchains," *Big Data*, 2017.

[6] Bloomberg, "How faking videos became easy - and why that's so scary," June 2021, accessed on October 2022. [Online]. Available: https://fortune.com/2018/09/11/deep-fakes-obama-video/

[7] J. Wakefield, "Deepfake presidents used in russia-ukraine war," last time accessed on October 2022. [Online]. Available: https://www.bbc.com/news/technology-60780142

[8] "Fox news edits video of biden to make it seem he was being racially insensitive," November 2021, accessed on January 2022. [Online]. Available: https://www.theguardian.com/media/2021/nov/13/fox-news-edits-biden-video-negro-leagues-satchel-paige

[9] A. M. Alattar, R. K. Sharma, and J. Scriven, "A system for mitigating the problem of deepfake news videos using watermarking," *electronic imaging*, 2020.

[10] A. Qayyum, J. Qadir, J. Qadir, M. U. Janjua, M. U. Janjua, F. Sher, and F. Sher, "Using blockchain to rein in the new post-truth world and check the spread of fake news," *IT Professional*, 2019.

[11] Investopedia. (2022) 51% Attack: Definition, Who Is At Risk, Example, and Cost. [Online]. Available: https://www.investopedia.com/terms/1/51-attack.asp

[12] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. De Caro, D. Enyeart, C. Ferris, G. Laventman, Y. Manevich, S. Muralidharan, C. Murthy, B. Nguyen, M. Sethi, G. Singh, K. Smith, A. Sorniotti, C. Stathakopoulou, M. Vukolić, S. W. Cocco, and J. Yellick, "Hyperledger fabric: A distributed operating system for permissioned blockchains," in *Proceedings of the Thirteenth EuroSys Conference*, ser. EuroSys '18. New York, NY, USA: Association for Computing Machinery, 2018. [Online]. Available: https://doi.org/10.1145/3190508.3190538

[13] N. Reiff, "What's the environmental impact of cryptocurrency?" last time accessed on October 2022. [Online]. Available: https://www.investopedia.com/tech/whats-environmental-impact-cryptocurrency/

[14] "Gas and fees," last time accessed on October 2022. [Online]. Available: https://ethereum.org/en/developers/docs/gas/

[15] "Hyperledger - open source blockchain technologies," available at https://www.hyperledger.org/ accessed on October 2022.

[16] C. C. Ki Chan, V. Kumar, S. Delaney, and M. 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, Sep. 2020.

[17] R. G. Brown, "The corda platform: An introduction," May 2018.

[18] Hyperledger, "A blockchain platform for the enterprise - hyperledger-fabricdocs main documentation," Hyperledger, Tech. Rep., 2020-2022, last time accessed on October 2022. [Online]. Available: https://hyperledger-fabric.readthedocs.io/en/latest/

[19] L. Teixeira, I. Amorim, A. U. Silva, J. a. C. Lopes, and V. Filipe, "A new approach to crowd journalism using a blockchain-based infrastructure," in *Proceedings of the 18th International Conference on Advances in Mobile Computing &amp; Multimedia*, ser. MoMM '20. New York, NY, USA: Association for Computing Machinery, 2021, p. 170–178. [Online]. Available: https://doi.org/10.1145/3428690.3429159

[20] J. Benet, "Ipfs - content addressed, versioned, p2p file system," 2014. [Online]. Available: https://arxiv.org/abs/1407.3561

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

[22] Y.-C. Chen, Y.-P. Chou, and Y.-C. Chou, “An image authentication scheme using merkle tree mechanisms,” *Future Internet*, vol. 11, no. 7, 2019. [Online]. Available: https://www.mdpi.com/1999-5903/11/7/149

[23] P. England, H. S. Malvar, E. Horvitz, J. W. Stokes, C. Fournet, R. Burke-Aguero, A. Chamayou, S. Clebsch, M. Costa, J. Deutscher, S. Erfani, M. Gaylor, A. Jenks, K. Kane, E. M. Redmiles, A. Shamis, I. Sharma, J. C. Simmons, S. Wenker, and A. Zaman, “Amp: Authentication of media via provenance,” in *Proceedings of the 12th ACM Multimedia Systems Conference*, ser. MMSys ’21. New York, NY, USA: Association for Computing Machinery, 2021, p. 108–121. [Online]. Available: https://doi.org/10.1145/3458305.3459599

[24] D. Han, J. Zhang, Y. Liu, P. Wu, and Y. 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)*. Atlantis Press, 2018/05, pp. 530–534. [Online]. Available: https://doi.org/10.2991/ncce-18.2018.84

[25] L. Weng and B. 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*, ser. CMS’11. Berlin, Heidelberg: Springer-Verlag, 2011, p. 108–121.

[26] M. Tanaka and H. Kiya, “Fake-image detection with robust hashing,” in *2021 IEEE 3rd Global Conference on Life Sciences and Technologies (LifeTech)*, 2021, pp. 40–43.

[27] P. Yu, Z. Xia, J. Fei, and Y. Lu, “A survey on deepfake video detection,” *IET Biom.*, vol. 10, no. 6, pp. 607–624, Nov. 2021.

[28] Y. Li and S. Lyu, “Exposing deepfake videos by detecting face warping artifacts,” *arXiv: Computer Vision and Pattern Recognition*, 2018.

[29] A. Qureshi, D. Megías, and M. Kuribayashi, “Detecting deepfake videos using digital watermarking,” in *2021 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, 2021, pp. 1786–1793.

[30] S. Paschalakis, K. Iwamoto, P. Brasnett, N. Sprljan, R. Oami, T. Nomura, A. Yamada, and M. Bober, “The mpeg-7 video signature tools for content identification,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 7, pp. 1050–1063, 2012.

[31] L. Latinov, “Md5, sha-1, sha-256 and sha-512 speed performance,” last time accessed on November 2022. [Online]. Available: https://automationrhapsody.com/md5-sha-1-sha-256-sha-512-speed-performance/#:~:text=SHA%2D1%20is%20fastest%20hashing,and%2023.4%25%20for%20longer%20strings.

A

Requirement Evaluation

In this appendix, all requirements are discussed, presenting a table containing the requirements description, compliance status and the arguments to support them.

FR-1.1	
Status	✓
Problem/issue	The system must allow only data submissions by enrolled news professionals in a Fabric network channel
Arguments	The current system implementation employs the private permissioned blockchain Hyperledger Fabric, which only grants permissions to write the ledger to users who belong to an organization within a channel and are that enrolled in it

Table A.1: Evaluation of FR-1.1

FR-1.2

Status	✓
Problem/issue	The application must generate a naming convention (file ID) for each submitted file, easing the process of querying the ledger by reliable news professionals
Arguments	The current system creates a naming convention (file ID) for each video file using data such as the user's name and organization nomenclature, the category designation the video is inserted and the timestamp of the submission

Table A.2: Evaluation of FR-1.2

FR-1.3

Status	✓
Problem/issue	The naming convention (file ID) should be sent to the user upon submitting a video file
Arguments	The current implementation of the system sends as a response the naming convention (file ID) generated for a video after submitting it, allowing trusted news media professionals to query the ledger

Table A.3: Evaluation of FR-1.3

FR-1.4

Status	✓
Problem/issue	The CID of the video data added to IPFS could be sent as a response upon submitting the video to make it available for every other user
Arguments	The current system makes available the CID of the inserted IPFS data so that it is shared with every user who desires to perform the *Detect* functionality

Table A.4: Evaluation of FR-1.4

FR-2.1

Status	✓
Problem/issue	Only news professional users can query data from the ledger
Arguments	The current system implementation employs the private permissioned blockchain Hyperledger Fabric, which only grants permissions to read the ledger to users who belong to an organization within a channel and are that enrolled in it

Table A.5: Evaluation of FR-2.1

FR-2.2

Status	✓
Problem/issue	Trusted users should provide the generated naming convention (file ID) to query the ledger
Arguments	In the current system implementation, only a user considered trusted holding a file ID should query the ledger

Table A.6: Evaluation of FR-2.2

FR-2.3	
Status	✓
Problem/issue	The video data should be sent to the user as response
Arguments	In the current system, when a trusted news professional user performs the *GetVideo* operation provided by the implemented API, the application retrieves the information about the video from the decentralized storage containing data such as the computed Merkle root, the bottom level of the Merkle-tree, the generated file ID and the CID for the video content to be downloaded

Table A.7: Evaluation of FR-2.3

FR-2.4	
Status	✓
Problem/issue	The video file must be sent to the user
Arguments	In the current system, when a trusted news professional user performs the operation *Get video* provided by the implemented API, the application downloads the desired video from the decentralized storage

Table A.8: Evaluation of FR-2.4

NFR-1	Scalability
Status	-
Problem/issue	The system should provide means to add more users and nodes without compromising the performance
Arguments	This requirement can be validated through the capabilities of Hyperledger Fabric outlined in Section 2.1.4. However, more network tests must be run to change the number of nodes required to support such a system scale for this requirement to be proven.

Table A.9: Evaluation of NFR-1

NFR-2	Tamper-proof
Status	✓
Problem/issue	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.
Arguments	This requirement is assessed on a theoretical level due to the capabilities of the system by using the Hyperledger Fabric network and the secure decentralized storage IPFS explained in 2.1.4 and 2.2, which have proven to preserve data integrity.

Table A.10: Evaluation of NFR-2

NFR-3	**Transparency**
Status	**-**
Problem/issue	The system could allow read-only access to all network participants so that they can certify the veracity and rightfulness of the inserted data
Arguments	This requirement is assessed on a theoretical level due to the capabilities of the system by using the Hyperledger Fabric network, which has proven to ensure transparency to the transactions within the ledger. However, this requirement is not a priority in comparison to others, and future work could assess it in more detail

Table A.11: Evaluation of NFR-3

NFR-4	**Availability**
Status	✔
Problem/issue	All data should be available to each user depending on their access permissions
Arguments	The system provides components that provide high availability to its data, such as Hyperledger Fabric. However, a drawback outlined in Section 2.2 is that IPFS cannot completely guarantee this property. Therefore, tests could be made using other emerging decentralized storages such as *Filecoin* that provide more data availability.

Table A.12: Evaluation of NFR-4

NFR-6	**Trust**
Status	✔
Problem/issue	The data contained within the system components should be trusted by all sources
Arguments	By limiting who can contribute audiovisual content to authorized news professionals, the system use cases offer a way to ensure the data's reliability. Additionally, users can validate content using the *Detect* functionality when they encounter untrustworthy content from dubious sources. Future implementation of a reputation system to gauge the credibility of any organization could enhance this requirement.

Table A.13: Evaluation of NFR-6

NFR-7	**Adaptability**
Status	✔
Problem/issue	The system's implementation should make it simple to add improvements
Arguments	By creating a data structure for data verification like a Merkle-tree, the application is ready to be enhanced by replacing MPEG-7 digital signature with other hashing mechanisms or A.I. algorithms presented in Chapter 2

Table A.14: Evaluation of NFR-7