Institute of Operating Systems and
Computer Networks

Research Project

Trustworthy Recording of Media Files Using Blockchains

Hesham Hosney

Supervisor:
Prof. Dr. Rüdiger Kapitza
M.Sc. Signe Rüsch

July 11, 2019

Contents

List of Figures

Abstract

The Blockchain technology is having a huge potential since it had a huge contribution in redefining the way the data is stored, updated, and moved. It is expected that the blockchain will revolutionize different business and industry with such huge capabilities, In this report, we proposed a platform for detecting Fake news using Hypereledger fabric technology. We implemented an Android application that allows the users to share and rank nearby events.

The event is an incident which happened somewhere, for example, a car accident or urgent road maintenance leads to closing the main road. We incent users to share their stories and the stories will be displayed only to the nearby users. We proposed a reputation system for ranking the users and their stories. The algorithm ranks the users and the events they created is based on the users' location or how close the users from an event they are judging, and the users' reputation score.
By keeping a reputation score for every single user depending on his previous contributions to the platform. and how close he was from the event he is assessing furthermore we collect the last 24 hours location traces for every user to make sure that the users are not scamming the platform. using the aforementioned methods we assure the trustworthiness of the shared events. Thanks to Hyperledger fabrics' features like immutability, trustworthy, and transparency, we proposed a prototype of our application to mitigate the spread of Fake news our application leveraging the blockchain technology and enabling it as a simple use case in the industry. We assure the data confidentiality since it will be stored on a distributed ledger and only updated by the chaincode which holds the whole business logic modifying the data once it's created is not possible the data will be immutably stored and distributed on multiple nodes.

This report describes the proposed work as follow, we will start with the main motivation, a walk through the application and the use case scenario, in chapter 2 we will give a brief introduction about blockchain and Hyperledger fabric. In chapter 3 we discuss the application infrastructure we used, for instance, the network nodes, SDK implementation, and API calls. In Chapter 4 we are summarizing the Android application part and how we designed and developed the application in depth. Chapter 5 will focus on the results and insights gained and issues.

Statement of Originality

This research project has been performed independently with the support of my supervisors. To the best of the author's knowledge, this research project contains no material previously published or written by another person except where due reference is made in the text.

Chapter 1

Introduction

1.1 Problem Statement

In the era of social media, Fake news or fabricated content is exponentially growing. Social media platforms are set up to push the most viewed content, and controversial topics get way more interactions. Facebook shows an endless stream of items from all over the web. Click an interesting headline and you may end up on a Fake news site. Fake stories have no reputation to maintain and no incentive to stay honest; Social media becomes a breeding ground for Fake news, As a result of spreading fake news the public opinion could be biased and manipulated for instance in the 2016 US presidential election and how spreading Fake news influenced the presidential election and favored a specific candidate.
This raises our aim to implement an application that could mitigate Fake news by incenting users to share and verify stories and build a reputation system for the story creators and considering the user's location or how close or far away from an event in our equation. By taking into account both the users' reputation score and the location´s trustworthiness we will ensure that the users will report only the accurate stories.

1.2 Introductory Walk-Through

The spread of social media and integrating it as an essential part of our daily life has a tremendous effect on spreading fake news. Many adults prefer getting news from the social media platforms like Facebook and Twitter more than the legacy ways like newspaper and broadcast, with the advance of technology it becomes trivial to manipulate images and videos. Our mission is to mitigate fake news with our platform we incent users to publish and rank nearby events based on their reputation. By using hyperledger technology it will be hard to manipulate the data once it's stored on the blockchain.

We are using a ranking scheme to increase the trustworthiness of the users, Also the user's location is taken into consideration when calculating the ranking score. We gave more weight to the users, which their location is closer to the event they are judging. By virtue of the chain code or smart contract, which will be responsible for handling the business logic and rank the events based on users location and trustworthiness. The architecture of our application could be described in particular as below:

1. A user can capture an event or text and upload it using our android application to the blockchain.
2. The event's data will be stored permanently on the blockchain and it will be impossible to be altered.

3. Our application will fetch and display the events to the nearby users.

4. The users in the close-by area will be able to judge the events and upload comments and further media files that approve or disapprove the events. we are restricting the participation of the users in judging events, and only legitimate users that their locations are in the nearby area will be eligible to vote. [1]

5. Depending on the other users reviews on the assessed event, the chaincode will update the trustworthiness value of the event [2]

6. The chaincode will update the event creator's reputation score based on the reviews that his events received. In other words, An event which received good reviews will get a more trustworthiness score also it will increase the event creator's reputation score and gives him more credits on our platform.

In Figure 1.1 we describe the previously mentioned procedures as an overview of our platform.

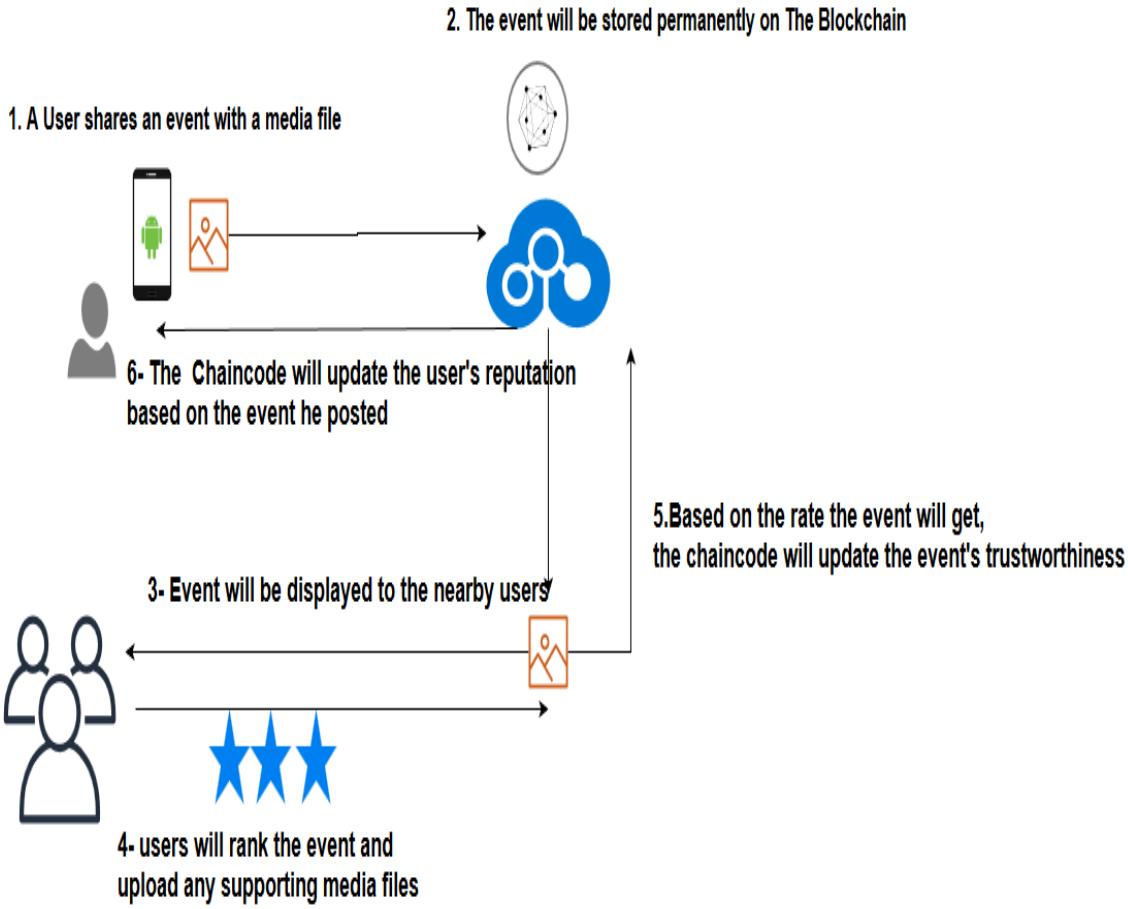

Figure 1.1: The Application Walkthrough

[1] collecting time and location evidence in order to assure that the users were in the nearby area and their location wasn't spoofed is part of a separate bachelor thesis running in parallel.

[2] Ranking the trustworthiness of the events and the users will be based on the reviews of the event in addition to the trusted code execution is part of a separate thesis running in parallel.

1.3 Architecture

The platform would be dismantled into three main components:

- The Hyperledger fabric network.
- The Backend and Fabric SDK.
- The Android Development.

In order to fully grasp the whole components of the platform. In chapter 2 we will start with the necessary background information about the Blockchain, The terminologies, And the main components of Hyperledger fabric for example peers, chaincode, transactions .etc.
In chapter 3, We will discuss the backend, API Design and how the API requests will be consumed and the way that Node.js SDK handles the requests, also how the SDK interacts with hyperledger fabric network and the chaincode.
Finally, We will discuss the Android application design, UI elements, The Main activities and classes in depth with all the implemented functionality. Figure 1.2 depicts all the architecture overview of the platform.

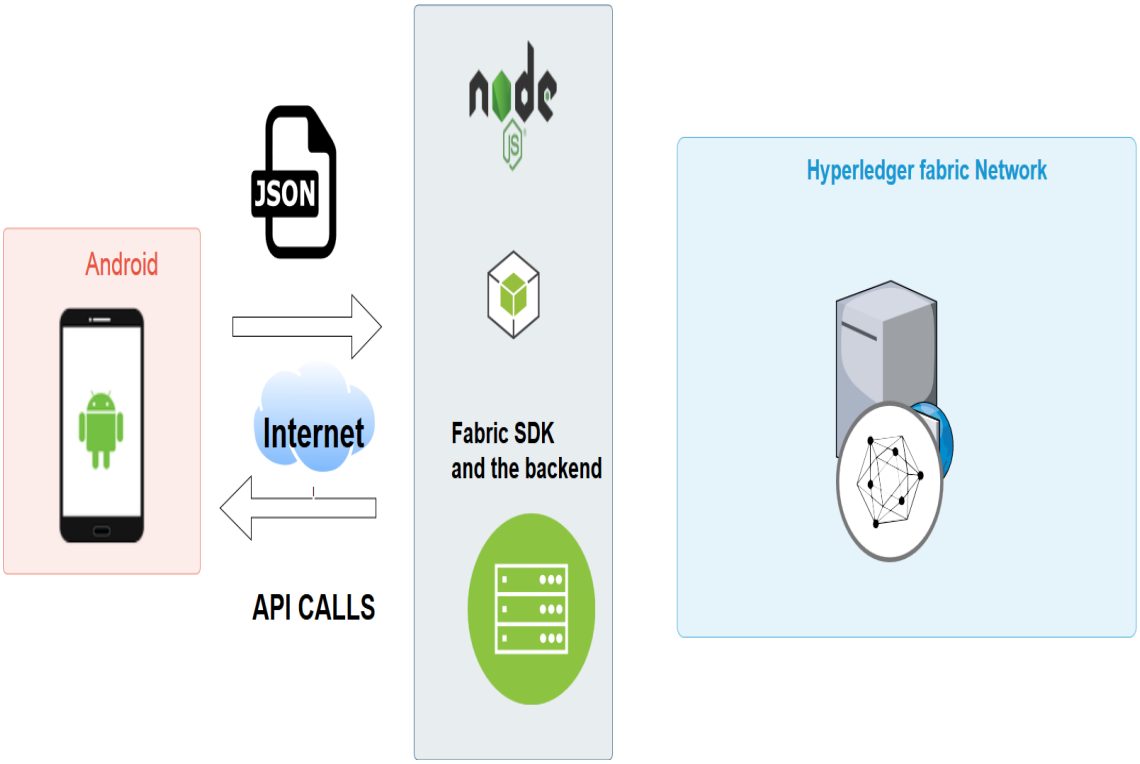

Figure 1.2: Architecture Overview

Chapter 2

The Blockchain And Hyperledger Fabric Background

2.1 The Blockchain

Blockchain[9] has become one of the most hyped technology trends. it provides a new way of managing and storing data, in addition, it provides capabilities for solving complex computing solutions. Taken further blockchain could be a cornerstone for secure data transfer and exchange on the internet. Trust on the internet considered one of the most stubborn problems, and here blockchain came to play. unlike the traditional database that requires a central authority which authorizes the access to it and determines what is inserted and deleted. The traditional database has more two major flaws[10]. First, it has a single point of failure, in addition, it will have only one major central authority.
Central authority between many stakeholder levies additional overhead. For instance processing time and cost, vulnerability and limited participation between parties. Blockchain introduces a new way of storing data instead of a central server the data will be stored on every single node on the network in a distributed fashion.

It's worth mentioning that Blockchain was based on a collection of related technologies, for instance, The public key infrastructure or PKI. Public Key Infrastructure provides an infrastructure for digital certificate management.
PKI will add Trustworthiness of the keys, thus we can avoid a situation when Anybody could have uploaded their own keys and claimed they belong to an email address (masquerading). After the organization generates the public and private key it will share the public key and here comes the role of a digital signature, which is equivalent of a handwritten signature, but offering far more inherent security, a digital signature is intended to solve the problem of tampering and impersonation in digital communications.
Digital signatures can provide the added assurances of evidence to an origin, identity, and status of an electronic document, transaction or message, as well as acknowledging informed consent by the signer.

in addition, it's also beneficial to cover some terms. A nonce is a number that is used once and never used again it helps to avoid duplicate transactions in digital transmission, for example, sometimes the data entered the database may have the same identifier adding nonce making it a bit harder for occasional replication. A Hash function is a mathematical process that takes data of any size and returns a unique hash with a fixed size no matter the size of the input data, however, it's only a one-way function thus it's hard to generate the original text from the hashed value. one major advantage of hashing is to keep the database small, as storing only the hash value save a tremendous

amount of disk. in blockchain, hashes are used as an identifier for blocks and transactions.

Blockchain[11] is a distributed database and its entries are immutable in other words it's not living on one server it's distributed and the data stored in it can't be modified or deleted. Blockchain is a sequence of blocks, which holds a complete list of transaction records like a conventional public ledger. namely that the information is packaged into blocks, which link to form a chain with other blocks of similar information. It is this act of linking blocks into a chain that makes the information stored on a blockchain so trustworthy. Once the data is recorded in a block it cannot be altered without having to change every block that came after it, making it impossible to do so without it being seen by the other participants on the network. Normally, each block contains the data it is recording, for example, transaction details as well as timestamps of when that information was recorded. It will also include a digital signature linked to the account that made the recording and a unique identifying link, in the form of a hash to the previous block in the chain. It is this link that makes it impossible for any of the information to be altered or for a block to be inserted between two existing blocks. In order to do so, all following blocks would need to be edited too. As a result, each block strengthens the previous block and the security of the entire blockchain because it means more blocks would need to be changed to tamper with any information. When combined, all of these create unquestionable storage of information, one that cannot be altered.

In traditional database data is stored in tables, rows and, columns usually in every table the stored data is related. For instance, a table might contain a customer's name and address. It will also contain a unique key for every record to link one table to another. In a client-server model, the database usually lives on a single logical server and it's queried as request and the server runs the query and sends back the result to the client. Another aspect of this is in web architecture there are usually other tiers like a client, web server and load balancer, web servers and database servers are distributed among physical servers, however, logically they are governed by the same set of rules and said to be centralized even if they are physically distributed. the traditional database is useful however it's also vulnerable. A blockchain database is structured in a way that it comprises of specific blocks of data organized in a specific structure, However, there is a huge difference in how the database is hosted a copy of a blockchain database resides on every node participate in the network there is no central server and it's not hierarchical it's a peer to peer. For example, if a user joins a blockchain environment all the database will be downloaded to his computer next each transaction happens on the blockchain will be recorded in every instance of the blockchain database in this way it's called a distributed Ledger. for every transaction, consensus must be reached by all participating nodes. Lastly, a blockchain distributed ledger is immutable entries made will never be edited once a transaction occurred it will always be there. there is no remedy.

The ledger is updated by a smart contract. A smart contract[12] is a computer code running on top of a blockchain containing a set of rules under which the parties to that smart contract agree to interact with each other. If and when the pre-defined rules are met, the agreement is automatically enforced. The smart contract code facilitates, verifies, and enforces the negotiation or performance of an agreement or transaction. It is the simplest form of decentralized automation.
Figure 2.1 depicts the main characteristics of the blockchain and the differences between the traditional databases.

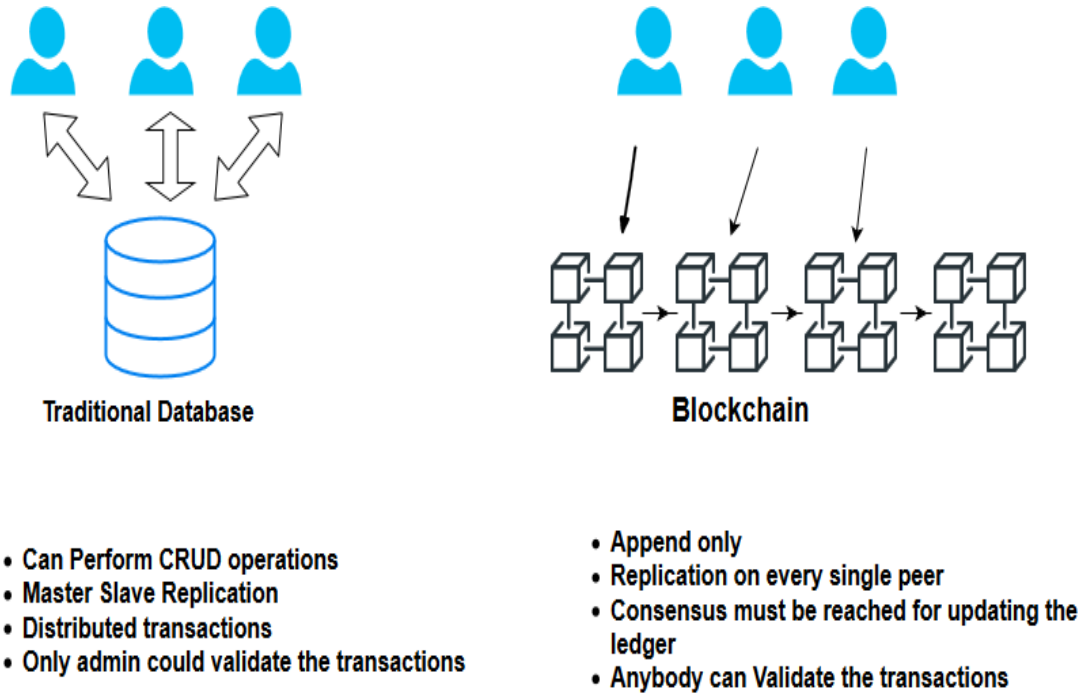

Figure 2.1: Blockchain VS Traditional Database

2.2 Hyperledger Fabric

2.2.1 Introduction

Hyperledger Fabric[13] is an Open-source permissioned Blockchain platform it was profounded under the Linux Foundation. Hyperledger Fabric became one of the fastest growing projects in the Linux Foundation. Hyperledger Fabric is the first platform that supports developing the smart contracts("chaincode")[12] in general-purpose programming languages such as Java, Go and Node.js. In a public or permissionless blockchain anyone can participate without a specific identity. Hypeperledger Fabric is a permissioned blockchain in a simplified way this means the participants will be known to each other instead of being anonymous and fully untrusted. It provides a way to secure the interactions between different parties, which they not particularly trusting each other having that it reinforces the trust between different entities. Hypeledger Fabric is an enterprise distributed ledger based on GO programming language that uses smart contracts. The architecture of Fabric follows an execute-order-validate paradigm for distributed execution of untrusted code in an untrusted environment. It separates the transaction flow into three steps, which may be run on different entities in the system: (1)execute where Transactions are executed(using chaincode) in any order, possibly even in parallel. (2)ordering through a consensus protocol When peers reach an agreement on the results of a transaction, it's added to the ledger and spread to all peers. (3) transaction validation by the peers which can check whether a later transaction was invalidated by an earlier transaction. For example, this prevents one item from being sold two times (called double-spending)[14].

Hyperledger is distributed by design there is no single point of failure or a single place that the

information is stored. All the participants are holding the same data, moreover, the data cannot be altered once it's stored, therefore Hyperledger fabric reinforces the trust among the participants.

2.2.2 Hyperledger Fabric Components

Hyperledger Fabric comprises of three main components:

- Hyperledger Fabric CA (Certificate Authority)
- Hyperledger Fabric Peer
- Hyperledger Fabric Orderer

Hyperledger Fabric CA: every single operation on hyperledger fabric must be signed with certificates. Fabric CA is a high-quality tool that pursues cryptographic standards to generate certificates these certificates are per user and attributes could be added while generating those certificates, therefore specific rules and permissions could be enforced per specific group of users. It is a place to generate certificates and where the account would be created this process called enrollment you provide username and password and the fabric CA does all the magic by the Issuance of Enrollment Certificates used for signing and identifying users identities. In addition, Fabric-CA could be attached to an existing LDAP and all attributes would be fetched from there. Fabric CA registers and enrolls the users, then it provides the certificate to SDK and the SDK signs all requests that will be executed inside Hyperledger fabric.

Peer: is the place where the ledger blockchain is stored there could be more than one peer especially in production they contain the same ledger. the request is sent from the SDK to peers, in order to do operations on the ledger like insert or query. The main role of the peer is to endorse and update the ledger. all peers are synced automatically adding more peers in the network the other peer will automatically update the state of the newly joined peer, and this is facilitating the scalability of hyperledger fabric.

Orderer: is coming into the heart of the consensus algorithm its main role is to provide order of operation before anything committed to the ledger it should be first validated and verified by the order and orderer creates the blocks so all transactions are stored into blocks and once blocks are full it will be sent to peers to be committed into the ledger. namely, the orderer is responsible for determining the order of operations. There are several different implementations of ordering the first type is Solo which is mainly for development and this is only one instance if this instance fails all the operations for updating the ledger will fail, and the second type is Kafka which is a Crash fault tolerance (CFT) implementation is suitable for the production environments.

It's also worth mentioning **the channel** concept channels are a method developed in hyperledger fabric for data isolation More broadly it's a completely isolated instance of hyperledger fabric. every channel is completely independent and they never exchange data. when building network peers must be a part of the channel when creating peers, the peers should join a channel. with channel concept, the privacy and confidentiality of the data will be assured. the channel should be created and configured by a specific tool from hyper ledger fabric. in addition, one peer could be a part of different channels.

Chaincode is a smart contract which is a program that reads and updates the ledger data, usually, all the business logic is written inside the chaincode. namely, the only way of interacting with the ledger will be only via the chaincode. The chaincode could be written in general purpose programming languages like Go, Java, and Javascript. The chaincode could adapt to any desired

complex business logic.

One important note that the chaincode must be apart of a channel inside one channel there could be as many as chaincodes. it could be possible to define all the business logic on one chaincode or split it amongst many chaincodes. The chaincode has to be installed and later instantiated. when writing chaincode it must be installed on every single peer that is a part of that channel this could be done using SDK or by using the peer itself.

Installing the chaincode on the peers is a must once it's the only way to interact with the ledger which resides on the peers. Once the chaincode is installed it should be then instantiated. Instantiating the chaincode will start a container that runs the chaincode while instantiating the chaincode the endorsement policy would be applied which will be responsible for validating and matching the rule on every single operation. For instance, applying a rule in a way that for every transaction every peer must agree on adding the record otherwise the transaction will be rejected.

Memebership Service Provider MSP: is one of the most fundamental concepts of hyperledger fabric is a setup of cryptographical materials where the organizations will be defined. every single on hyperledger fabric must have its cryptographic certificates that define if a specific node belongs to specific organization generating the cryptographic material is trivial by hyperledger fabric tool cryptogen. The most essential part is that only peers that are part of the same MSP could communicate with each other. other peers couldn't the peer must be a part of the organization and this is defined by the peer's certificate later when configuring the channel all the public certificates will be shared on the configuration thus only the configured peer with a shared certificate could communicate and be a part of this channel.

The State Database: In the blockchain concept, the state of the data would be tracked from the transactions history, for instance, assume that there is a simple balance transfer network. Person A sends 100 euros to person B and person C sends another 100 euros to person C. the current balance of person B should be 200 euros, however The blockchain consisted of serious of transactions, in order to calculate the latest state of the accounts all transactions history should be reviewed and validated. For more efficient way Hyperledger fabric uses a key-value store database to update the latest state of the data in an efficient way. Chaincode invocations execute transactions against the current state data. To make these chaincode interactions extremely efficient, the latest values of all keys are stored in a state database. The state database is simply an indexed view into the chain of transactions log, it can, therefore, be regenerated from the chain at any time. The state database will automatically get recovered (or generated if needed) upon peer startup before transactions are accepted.

World state or state database are options include LevelDB and CouchDB. LevelDB is the default state database embedded in the peer process and stores chaincode data as key-value pairs. CouchDB is an optional alternative external state database that provides addition query support when your chaincode data is modeled as JSON, permitting rich queries of the JSON content. Leveraging CouchDB as Hyperledger fabric state database[15].

2.2.3 Under The hood Hyperledger Fabric

In this part, we will try to explain how hyperledger fabric works under the hood, Assume that a user wanted to exchange the ownership of some assets, and The client wants to make a transaction. The first step is that the user has to register and enroll in the organization using Fabric CA and get back the proper cryptographic materials that authorize him to interact with the network. Then the SDK will be responsible for preparing the transaction as a transaction proposal. The transaction proposal will be sent to one or more peers. At this point the peer accepts this proposal and simulates it, the result of this simulation will be called read and write sets. The read set contains a list of unique keys and their committed versions that the transaction reads during the simulation. The write set contains a list of unique keys and their new values that the transaction writes. At this phase, the ledger is not updated and the assets values remaining the same. The endorsing peers which are special kind of peers that its role is to verify and approve the transaction. this is will be done by validating the format and the signature of the transaction, and if the application user is authorized to make this transaction. Then the peers are sending back those transaction responses to the SDK. The SDK collects all these transaction responses and signs it and sends it to the orderer, the orderer node verifies the transaction signature and the endorsement policy. no matter if the transaction approved or rejected it will be stored on the blockchain, however, the state of the ledger will not be updated in case of the failure. Once the orderer node receives the transaction responses from the SDK, it will verify the read and write sets collected from every single peer and assuring that all transaction responses are the same. Finally, in case the transaction is valid the orderer will send the transaction to the committing peers to be committed and updating the ledger. Figure 2.2 depicts Hyperledger fabric transaction flow [1].

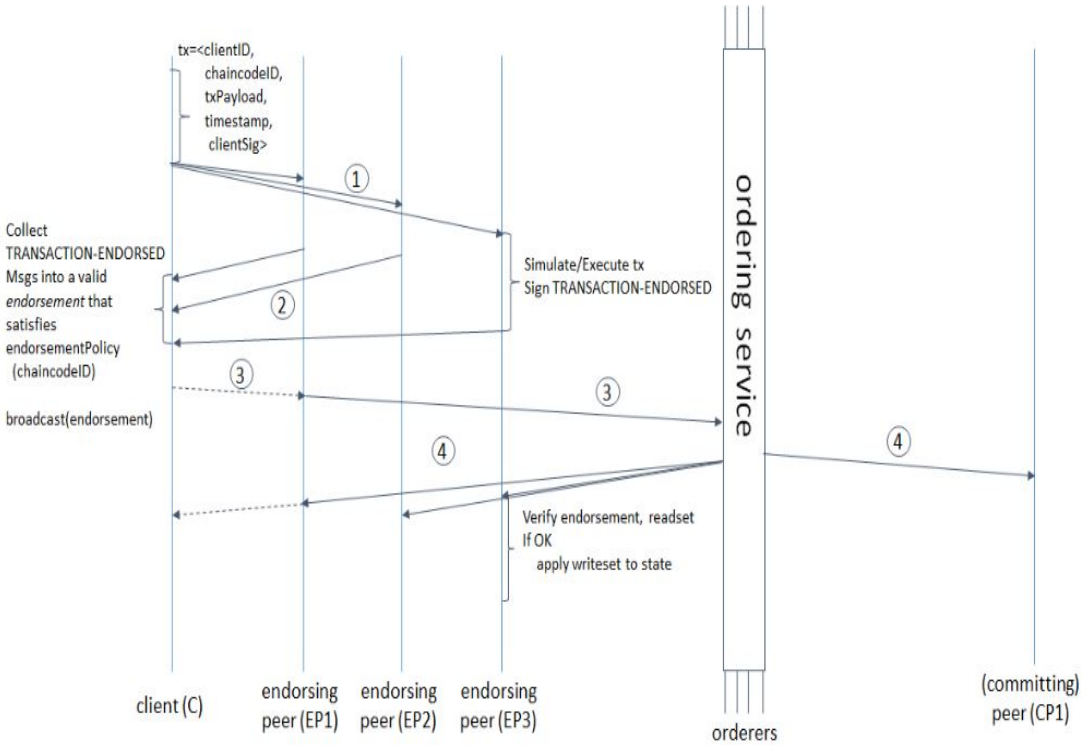

Figure 2.2: Hyperledger Fabric Transaction Flow Diagram

Chapter 3

The Application Infrastructure

3.1 System Architecture

The Fake news detection system architecture is shown in Figure 3.1 it was inspired by balance transfer network [2]

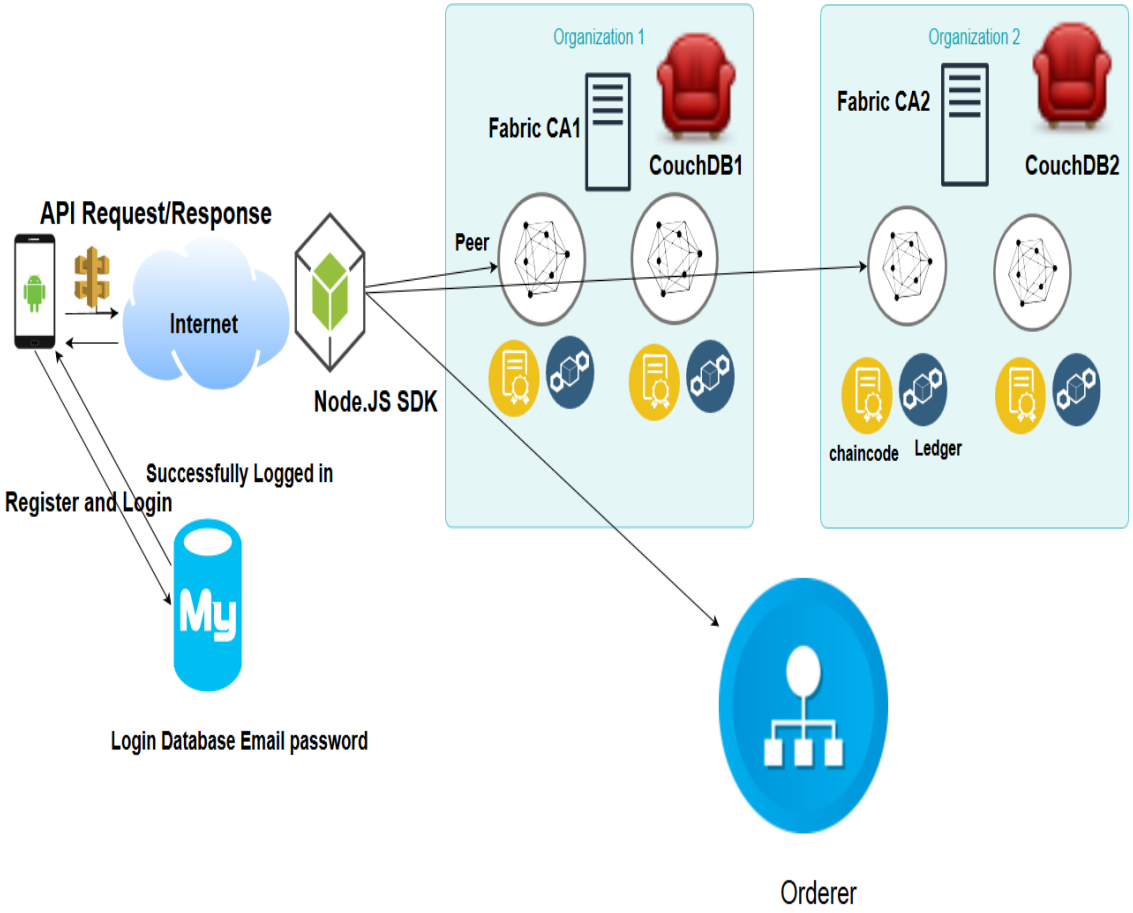

Figure 3.1: The Application Infrastructure Overview

As shown, The system comprises of two organizations organization 1 and organization 2 with two peers in each organization. With a solo orderer node. There are also two Fabric CA servers one for each organization, Two **Couch-DB** instances as world state databases for more efficient and rich queries. In addition, we are using node.js SDK in order to interact with the hyperledger fabric network and abstracting all the network infrastructure as a standard model, thus the network could be scalable horizontally and vertically. All nodes are running on Docker containers for our testing purpose we implemented all the infrastructure on one Virtual Machine, However, in a production scenario every node should be an independent physical instance.

A layer of authentication using a traditional database was introduced so that the users could sign up on our application using the traditional way we used email and password method for registration and login, however, integrating a phone authentication with a one-time pad could be trivial. Once the user successfully login he could send requests from the Android device using our application to the SDK. and the SDK will handle the interaction with hyperledger fabric network and the chaincode.

The Fabric CA server will be responsible for the registration of identities, Issuance of enrollment certificates for signing and identifying the application users, Issuance of transaction certificates and providing both anonymity and unlinkability when transacting on a Hyperledger fabric blockchain. All these functions are provided. The network was the adoption of Balance transfer [2] A sample Node.js app to demonstrate fabric-client, fabric-ca-client, and Node.js SDK APIs. the main idea of the network is to demonstrate a simple use case app using hyperledger fabric in order to transfer a balance from a user to another one and query the ledger. we modified the network, added couch database instances and updating it with our chaincode. we modified the node.js application to suit our application.

Crypto materials have been generated using the cryptogen tool provided by hyperledger fabric and mounted to all peers to identify different network components. The genesis block (genesis.block) and the channel configuration(mychannel.tx) have been pre-generated using the configtxgen tool provided by hyperledger fabric and all these configurations have been placed inside the artifacts folder.

3.2 Building the Network

Hyperledger fabric came with many tools for facilitating the development process. As hyperledger fabric is modular by design many modules interacting together to make the application up and running. Once all prerequisites packages are installed on the platform where the application supposed to be installed. Hyperledger fabric community provided fully annotated scripts that simply launches the network and generate the cryptographic material.

Initially, All the network topology should be pre-described in a yaml file, for instance, the number of organizations and peers, and the organizations name.
Taking into consideration that, every single operation must be signed and verified. as a result, every single network entity must have cryptographic materials (x509 certs and signing keys) those certificates will be used as identities and expedite the authentication process to take place as entities communicate and transact.
One of those tools is Cryptogen tool which consumes the Yaml file, which describes the network and generates all required cryptographic materials, after running Cryptogen tool. The generated certificates and keys will be saved to a folder titled crypto-config.

Figure 3.2 describes the crypto config directory structure.

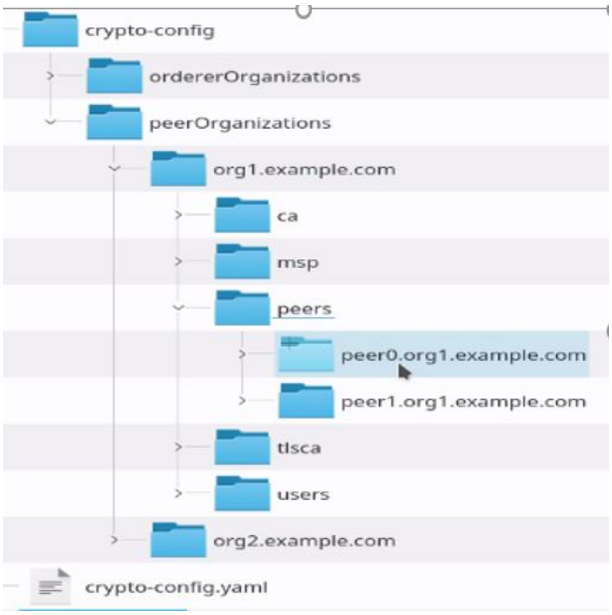

Figure 3.2: Crypto Config Directory structure

Configtxgen is used to generate the main configuration artifacts. such as genesis block, the channel and the anchor peers. Finally, docker-compose will be used to spin up the network.

3.3 Node.js App Backend and Node.js Fabric SDK

In our application we adopted one of the sample network provided by hyperledger fabric called balance transfer as a backend, this was done after modifying the main scripts to cope with our application requirements. **Balance Transfer** is a simple network with nodejs app that demonstrates fabric-client, fabric-ca-client, and Node.js SDK APIs.

- Allows the communication with the hyperledger fabric network by using fabric node.js Libraries.
- Allows different operations using GET/POST requests, such as channel's creation, joining the channel, installing and instantiating the chaincode.

Node.js Fabrik SDK: [3] eases the communication with hyperledger fabric blockchain using APIs. The Hyperledger Fabric SDK allows the interaction with the network on behalf of the users by simply calling APIs the following operations are supported:

- Creating channels.
- Allowing peers to join channels.
- Installing chaincodes in the peers.
- Instantiating chaincodes in the a channel.

- Querying the ledger.
- Invoking transactions.

We used two main modules provided by the SDK:

- fabric-client: Provides APIs to interact with the core components of a Hypreledger Fabric-based blockchain network, namely the peers, orderers and event streams.
- fabric-ca-client: Provides APIs to interact with the optional component, fabric-ca, that contains services for membership management.

3.4 Understanding Node.js App

Node.js App[4] comprises of java scripts which simplify the process of the communication with hyperledger fabric network. it obscures all the network and exposes REST APIs for communication. the main files are highlighted as below.
app.js will route the requests in a modular way by avoiding the repetitive tasks. express node.js framework used to do all of those functionality.
users.js will validate if the user enrolled successfully on the organization or not. If not it will enroll the user by calling fabric-ca-client and generate a json web token JWT, Finally the application user will receive this JWT to make subsequent requests.
helper.js will load the network configuration settings, where all the keys are stored, thus the fabric-ca library could interact with the different hyperledger fabric entities. Furthermore, it will check if the user enrolled, and if the user was not enrolled. It will enroll by loading admin credentials and calling fabric-ca-client to enroll the user.
install-chaincode.js contains the methods for calling fabirc-client to install the chaincode on the peers.

instantiate-chaincode.js contains methods to instantiate the chaincode on the channel.

invoke-transaction.js will invoke transaction proposals.

3.4.1 Json Web Token(JWT)

For securing the requests a JWT which is a signed token will be generated after enrolling the user and it will be sent to the application user to make subsequent requests. The token can be sent over in the request's header when making API calls for invoking transactions or querying the ledger. The application then validates the token and, if it's valid, the request will be forwarded to the fabric network. As per the below code snippet, the JWT will be expired after 10 hours then it should be renewed from the application side by pressing on the reload button.

```
const token = jwt.sign({
    exp: Math.floor(Date.now() / 1000) + parseInt(
        hfc.getConfigSetting('jwt_expiretime'), 10),
    username,
    orgName
}, app.get('secret'));

const response = await helper.getRegisteredUsers(username, orgName);

if (response && typeof response !== 'string') {
    res.json({
        success: true,
        token
    });
} else {
    res.json({
        success: false,
        message: response
    });
}
```

Listing 3.1: JWT Generation

3.4.2 runApp.sh Script

Using runApp.sh from balance transfer network[3] will spin up the network and expose a URL. All docker containers will be up and running. In addition, it will launch an instance of the app listening on port 4000.
Figure 3.3 displays a simple output after running runApp.sh script.

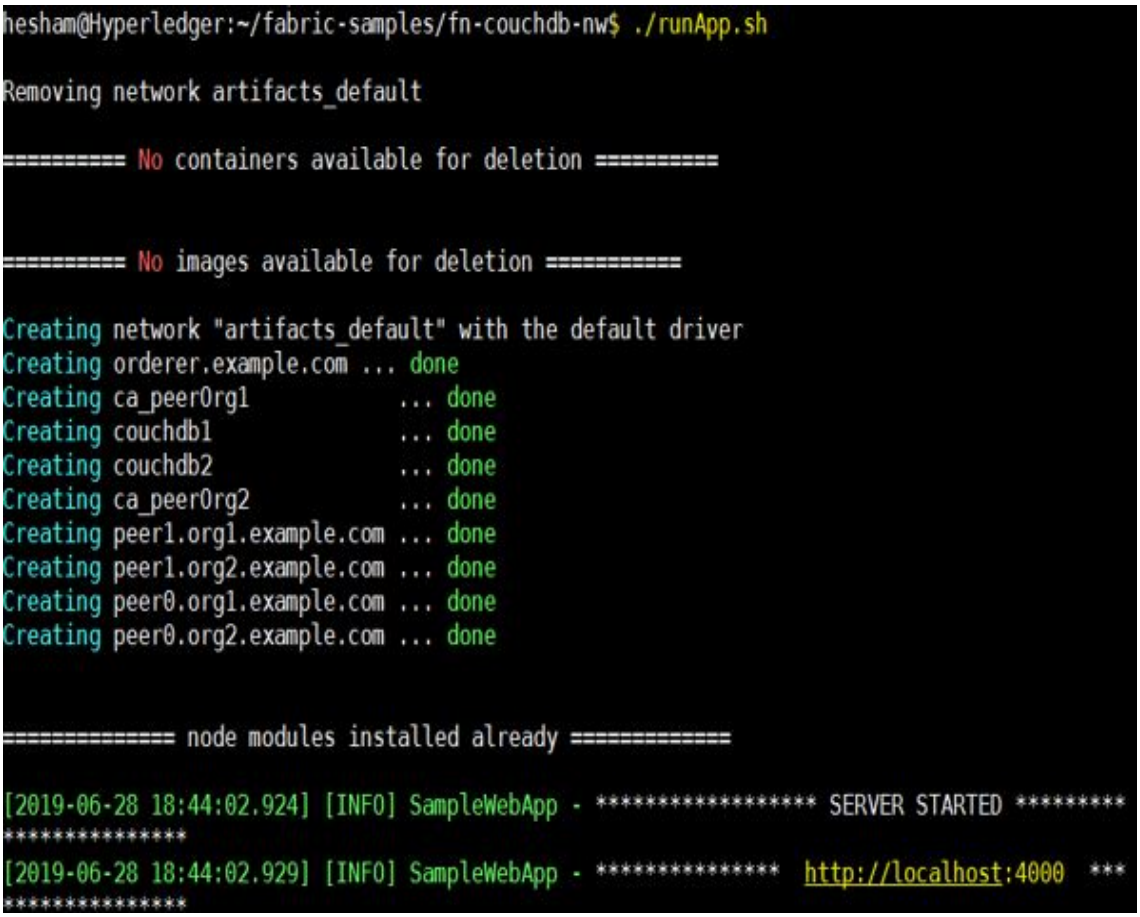

Figure 3.3: runapp.sh Script Simple Output

3.4.3 testApi.sh Script

After runApp.sh script successfully run and all network components will be up and running. testApi.sh script will be responsible for creating a channel, allowing peers to join the channel, installing chaincode on the peers. last but not least instantiating the chaincode on the channel.

3.5 Mysql Login Database

In order to allow the application users to register to our application on a traditional way using email and password. We created a traditional relational MySQL database named "fakenews" to store the users' email and hashed password with one table "login". We are making the process of users' registration independent from the blockchain network. Although, We have only one server that hosts both the Xampp server and the fabric network. However, this is only for the development purpose.

We created a simple XAMPP Server consists of:

- Mysql Database: to store the login data information.
- Apache as a webserver to serve the requests over the network.
- Php as a scripting language to consume the requests and making different CRUD(Create, Update, Read, Delete) operations on the database.
- PhpMyadmin for simplifying the management of the database using a web interface.

The user will send a registration request using an email and password from the android device. The request will be sent over the network, the Apache server is listening on port 80. We are calling register.php script for registering users. The register.php script check if the email already existed if not it will insert a record in the login table with the user's email and the hashed password. and return a success response to the android as a confirmation. Similarly, the login procedures, the user will send a login request using his email and hashed password by calling login.php and it will be checked against his record on the database if the entered credentials are correct. The access will be granted to the user.
Figure 3.4 describes the simple Xampp server and the fakenews database schema.

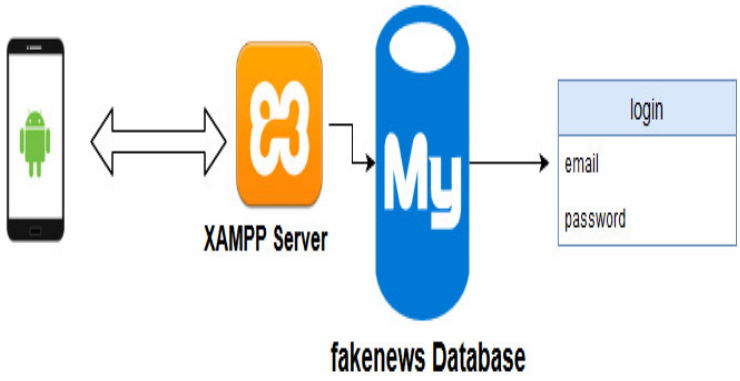

Figure 3.4: Xampp Server Overview

3.6 The Enrollment Procedures

This section describes the enrollment procedures as the following:

1. The user will register and login to the application using an email and password.
2. Once the user successfully logged-in the application will send an enrollment request to the SDK using the user's email.
3. The SDK forwards the enrollment request to the Fabric-CA.
4. Fabric-CA generates the required ecerts and keys for the user and enrolls the user on the network.
5. Once the user enrolled successfully the SDK generates a JWT for the user and sends it to the application.
6. Finally, The application stores the JWT on the android device and includes it on the subsequent requests' header.

Figure 3.5 depicts the whole operation.

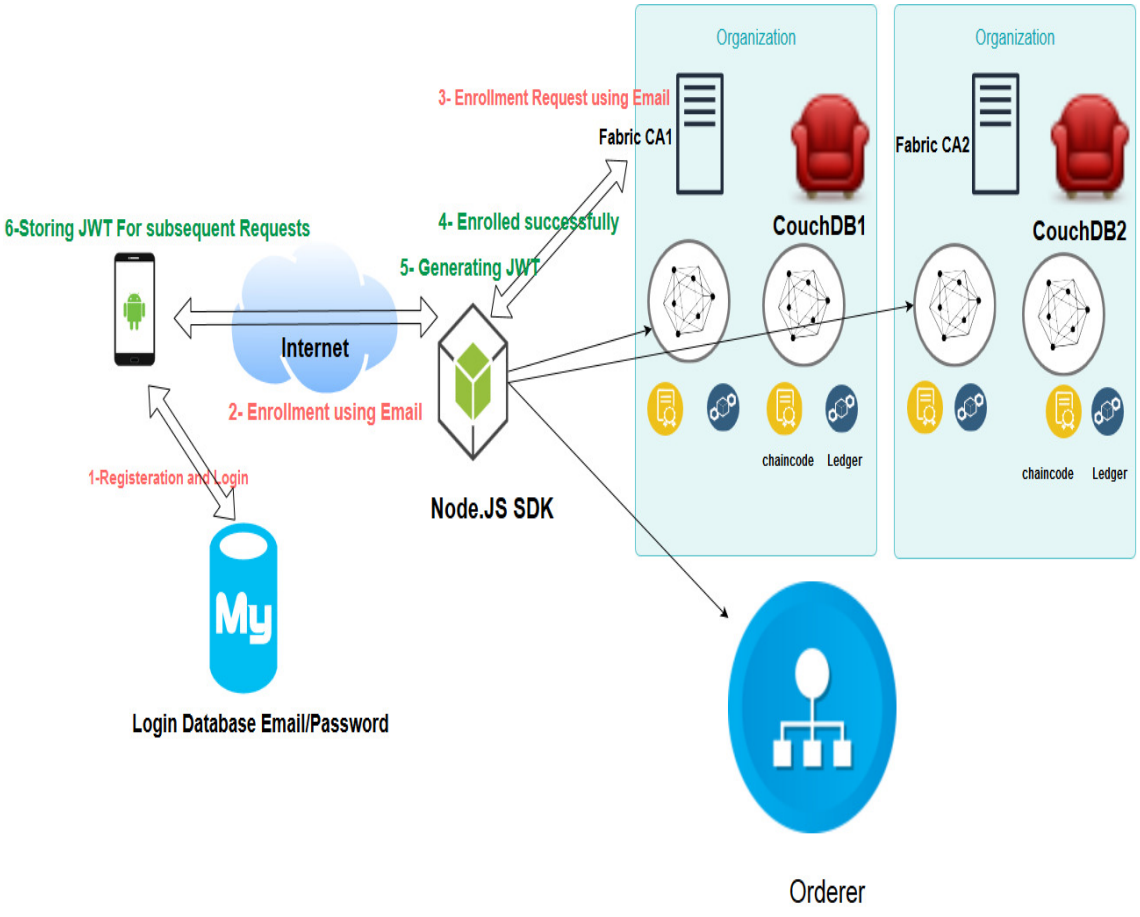

Figure 3.5: The Enrollment Procedures

3.7 Calling The Rest API

The node.js App obscuring the hyperledger fabric network and exposes a rest API endpoint to call, this will allow the application users to do different operations on the network. The application users could send a request by calling the URL endpoint using Post or Get Methods and passing the parameters, for instance, the chaincode name and arguments. In addition, The JWT must be included in the request header. Once the request is made and sent over the internet the node.js app receives the request and forward it to SDK to make different operations on the ledger like invoke transactions or querying the ledger. Once the transaction is made the response will be sent back to the Android application as a JSON response. The below listing illustrates an example of making an API call using curl command. calling the endpoint, passing chaincode name and parameters in the Post request and attaching the JWT for authorization in the request header

```
curl -s -X POST \
  http://localhost:4000/channels/mychannel/chaincodes/mycc \
  -H "authorization: Bearer $JWT" \
  -H "content-type: application/json" \
  -d '{
        "peers": ["peer0.org1.example.com","peer0.org2.example.com"],
        "fcn":"addEvent",
        "args":["{\"title\": \"foo\", \"location\": { \"latitude\": 52,
           \"longitude\": 10 }, \"description\": \"bar\" }"]
}'
```

Listing 3.2: Simple REST API Call Using Curl

The following Table shows the different functions provided by the chaincode for full API calls documnetation review [5].

Chaincode API Calls		
CC Function name	Usage	Request Type
addEvent	Invoking an event to the chaincode	Post
queryEvents	Query The ledger	Get
judgeEvent	Invoking an assessment on a specific event	Post
getFullEvent	Query a specific event with all it's assessments	Get

Chapter 4

Android Application

4.1 Overview

The application was designed to allow the users to interact with the blockchain. The Application UI was made similar to modern social networks, displaying the nearby events as a scrollable home feed Figure 4.1 shows the different screens design of the Android app. Authentication is required for the user to login to the blockchain network and to use the app. Once the user successfully authenticated the application will display the nearby events and the app is self-explanatory.

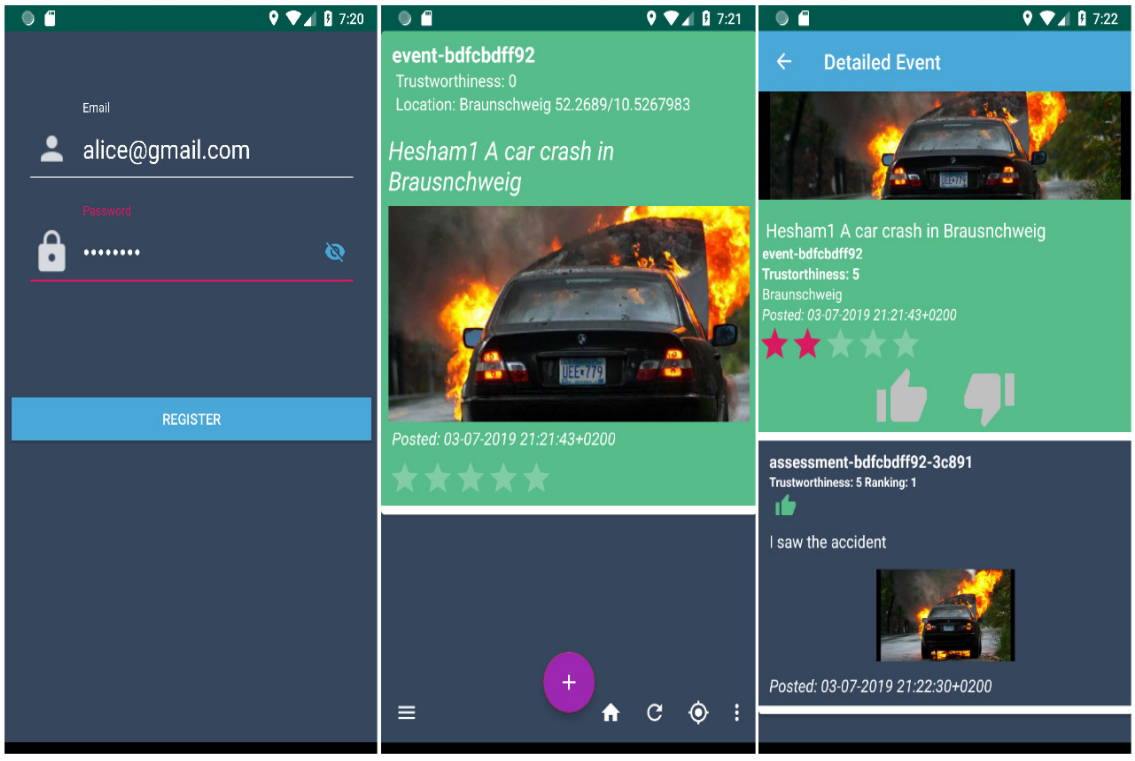

Figure 4.1: App Screens Represent The Application Design

4.2 The Main Classes

In this section, we describe the main classes we used in the application. Every class provided a different functionality.

4.2.1 Mainactivity

MainActivity is the first activity launched when the application started. it checks if the user logged-in then it will direct the user to registration and login activities. Once the user login the email address will be stored in the shared-preference in order to keep the user login. The location access permission will be checked as it's one of the essential parts to continuously fetch the location of the users. A warning message will be displayed to the user if the location access wasn't granted. As long as the location access permission is granted. An enrollment request will be sent using the user's email address as discussed in 3.6 following the enrollment procedures and storing the JWT for making the upcoming requests to the blockchain network. Finally, the application will use the JWT and the user's location to query the ledger and fetch all the nearby events and displaying them one by one chronologically in the news feed. Displaying the feed is nontrivial we used RecyclerView adapter which is an android class used to represent the data dynamically in a separate view and provides the ability to implement both horizontal and vertical layouts. The RecyclerView adapter used to display the data collections whose elements change at runtime based on user action or network events. In other words every time we query the ledger the user get a JSON response containing all events list. Those events change at run time based on the user's location. The events will be displayed one by one in a separate view similar to Facebook or Instagram posts. We created a simple view holder for every event object and we instantiated an object of this view holder and binding the event data to it. The same methodology used to display the assessments list. Figure 4.2 Describes the flow of the Mainactivity.

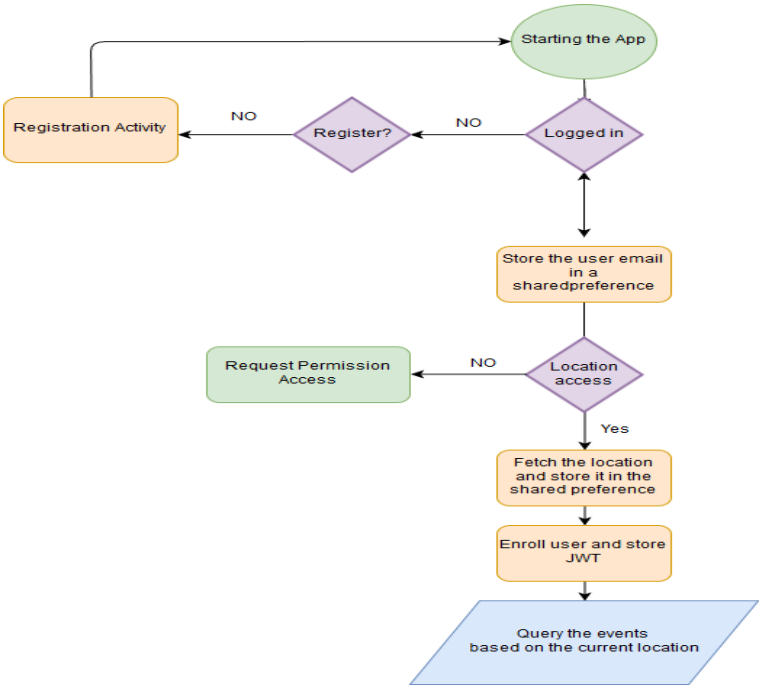

Figure 4.2: Mainactivity Flowchart

4.2.2 AddEvent

AddEvent is the activity that runs when the user wanted to create a new event. It will prompt the user to upload a photo and an event description. the activity will check the storage's permission access, and if it's granted it will allow the user to select an image from the gallery, then the image will be captured as a bitmap and converted into a string. This string will be hashed and sent to the Xampp server which will rebuild the media file and storing the file with its hash name. In other words, the picture will be stored with its hash and only the hash will be stored on the blockchain for fetching the picture and displaying it directly from the server, not from the blockchain. lastly, the transaction will be invoked by calling addEvent chaincode function name and passing all event's details as parameters. If the transaction successfully invoked a confirmation message will be displayed to the user that the event is online, Otherwise, an error will be displayed. Storing the hash instead of the entire media file will be more efficient as the file will be replicated many times on the peers. It's similar to using A content delivery network CDN which is a standard way to deliver content more quickly and efficiently, based on their geographic location. A content delivery network CDN facilitates the quick transfer of assets needed for loading media content including images, and videos. using this approach will increase the performance and decrease the latency. In addition, storing the hash only will save a lot of disk space and assuring that the ledger size will be only a few megabytes instead of hundreds of terabytes.

Figure 4.3 Describes the flow of AddingEvent Activity.

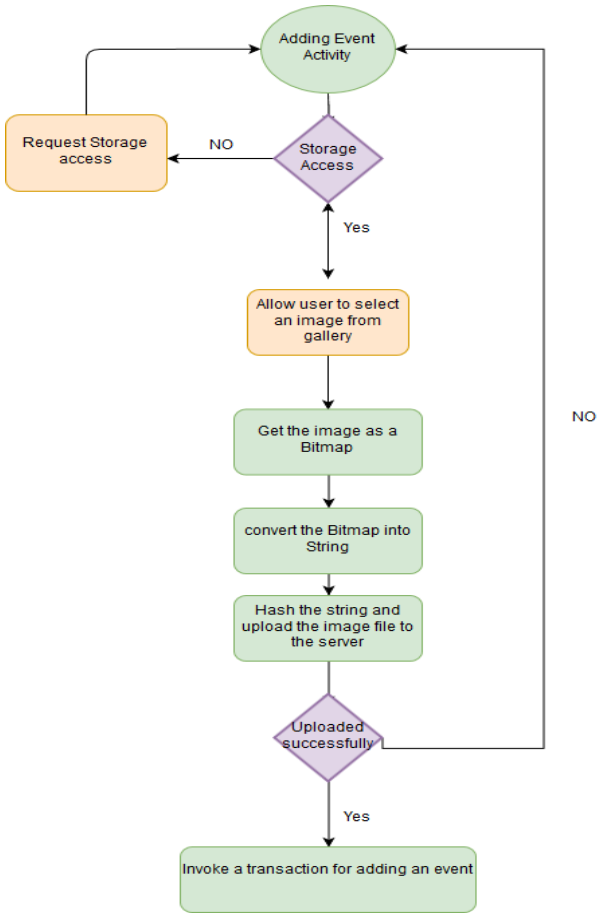

Figure 4.3: Addevent Flowchart

4.2.3 The Hashing Method

For a more efficient way of storing data on the blockchain, only the media file is hashed and only the hash of it will be stored on CDN like S3 we store the data on a simple Xampp server for development purpose. the media file will be easily fetched and displayed. **PBKDF2** Algorithm is implemented. this hash function is slow enough to impede attacks, but still fast enough to not cause a noticeable delay for the users. A simple implementation of the hashing algorithm in listing 4.1.

```
import javax.crypto.SecretKeyFactory;
import javax.crypto.spec.PBEKeySpec;
import java.math.BigInteger;

public class Hashing {

    private static final int ITERATIONS = 1000;
    private static final int KEY_LENGTH = 192; // bits
    public static final String SALT = "Random Salt" ;

    public static String hashPassword(String password, String salt){
        char[] passwordChars = password.toCharArray();
        byte[] saltBytes = salt.getBytes();
        PBEKeySpec spec = new PBEKeySpec(
                passwordChars,
                saltBytes,
                ITERATIONS,
                KEY_LENGTH
        );
        SecretKeyFactory key = null;
        try {
            key = SecretKeyFactory.getInstance("PBKDF2WithHmacSHA1");
        } catch (NoSuchAlgorithmException e) {
            e.printStackTrace();
        }
        byte[] hashedPassword = new byte[0];
        try {
            hashedPassword = key.generateSecret(spec).getEncoded();
        } catch (InvalidKeySpecException e) {
            e.printStackTrace();
        }
        return String.format("%x", new BigInteger(hashedPassword));
    }
}
```

Listing 4.1: PBKDF2 Simple Implementation

4.2.4 Voting

When the user selects on a specific event a new activity will be created, and the event will be separately displayed with more details about the event, and the application will allow the user to judge or assess the event by up-voting or down-voting, also it would be possible to attach a description or a media file. The voting activity is similar to adding an event the data will be stored captured media file will be hashed and uploaded to the server and finally, a transaction will be invoked and judgeEvent chaincode function will be invoked and all assessments details will be passed as parameters. in the end, a confirmation message will be displayed in case of success and an error in case of any failure or network error. Assessing events will credit or discredit a specific event and directly reflects the creator of the event's reputation score. The voting is similar to adding the event activity just different in calling the judgeEvent.

4.2.5 Login and Register

In order to allow the users to log-in and register, we implemented two separate activities. the user will input an email and password, then we are validating the user input. and an error will be displayed if the email address is not valid or empty or in use by another user. If the data was valid we submit a request over the network and store it to the relational database. The email will be stored in a plain text, however, the password will be stored hashed. Similarly, the login activity once the user inputs his email and password it will be checked against the database if the credentials were correct the access is granted to the user and the email will be saved to keep the user logged in.

4.2.6 Ranking and Trustworithness

One of the main features we introduced is ranking the events and calculating users' reputation. The user's reputation will depend on the user's interaction on the platform. how frequent he participated, how many good reviews he received on the event he created. Every user starts with 0 reputation score. Adding an event will increase his score by +1.
The algorithm considers the users rank and location's trustworthiness or how close is the user from the event. Assume a user with score 510 and was about 10 meters away from an event he is going to assess. First, we convert the score into stars as the user's reputation score is 510 then the stars function will return 5. Second, we calculate how far is the user and return a value from 1 (almost in the scene of the event he is assessing) to .1 (so far away from the event) in this case the closeness function will emit 1 so the final score the user is 25. This score will increase/decrease in the assessed event's trustworthiness. Depending on the vote type if it's up/down vote this will credit or discredit the event's trustworthiness and increase or decrease the overall reputation score of the event creator. Once the assessment is made the chaincode updates the assessed event's trustworthiness and the event creator's reputation score. In summary, What easily influences the user's reputation and the event's trustworthiness is getting good/bad reviews from the highly ranked users who are very nearby to the assessed event. Figure 4.4 describes the ranking algorithm.

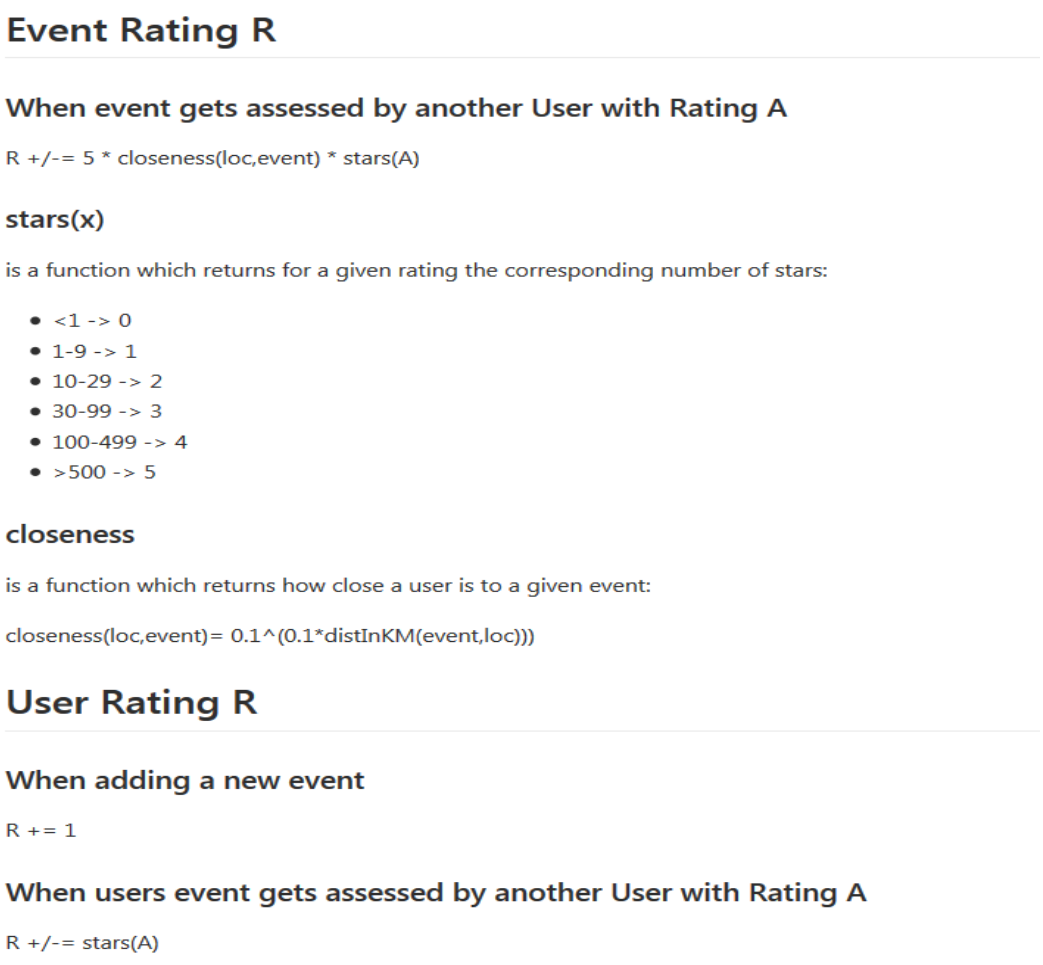

Figure 4.4: The Ranking Algorithm

Chapter 5

Issues and Future Work

5.1 Issues

One of the design issues faced during the development is where should the SDK be installed from Architecture point of view? 1-On the server side where the hyperledger fabric lives? or 2-On the client side (Android side) taking into consideration there is no official support for an Android SDK. The only supported SDKs are on GO, javascript, and java. Building and integrating an Android SDK from scratch is nontrivial it will require such time and effort and a dedicated team to achieve this goal. We implemented the SDK on the server side where Hyperledger fabric lives, thus we are obscuring the network and only exposing the node.js App with all scripts that consume the requests and make SDK calls to interact with the ledger. Designing the backend that way by totally obscuring the hyperledger fabric network will have the following advantages:

1. Since there is no officially supported SDK for Android it will require much time and effort to redevelop an Android SDK, as there is no point for reinventing the wheel.

2. Standardizing the communication using APIs. this way all the APIs could be easily standardized and manifested, Meaning, the application will only call a fixed URL, chaincode function name and passing arguments. No matter on which language the application was written if the client was written on android, web, or ios. It will be the same network rest API call made over the internet a example discussed in 3.7. However, if the SDK lives on the client side the application developer will be responsible for customizing the requests to match the SDK's language, and the way it interacts with the hyperledger fabric network.

3. Avoiding the latency and restricting all the transaction flow on the internal network. so the communication between the SDK and the peers will be done on the internal network and we avoid the delay will be introduced if the requests were made on the internet with many hobs

4. The communication will be encrypted using SSL certificate. All the header and payload will be encrypted.

5. The Whole hyperledger fabric network will be isolated and couldn't be directly accessed from the internet as we only exposing the end point for communication and no direct interaction with the peers, orderer or Fabric CA They are hidden behind the node.js App. in case of the node.js app compromised the data and all the hyeprledger fabric network will not be affected.

6. Finally, Avoiding the unnecessary SDK Library size which will expected to be around 30-50Mb on the smartphone taking into consideration the whole application size is 5Mb.

One main **disadvantage** of this approach is the data could be viewed by the server admin and this raises **privacy concerns**, as discussed on 3.7 the Android application is calling the exposed URL and passing the parameters required in order to make transactions. Parameters passed such as the chaincode function name and the events details. The node.js app scripts consumed those data and use the node.js SDK to do different operations on the fabric network such as querying the ledger or invoking a transaction. At the point, the data could be leaked by the server admin. even hyperledger fabric didn't explicitly discussed this concern and they implemented the Hyperledger Composer[8] that simply exposes the network on the same way using server and generating a REST API from a deployed blockchain business network that can be easily consumed by HTTP or REST clients. This was the reason we wanted to install the SDK on the client side and making the requests directly to fabric network or run the node.js on an Intel SGX enclave. However, to modify the data directly will be nontrivial. and could be easily investigated. Since the data on the ledger is append-only and the business logic is reinforced by the chaincode. Furthermore introducing a simple cryptographic scheme to cipher the data would resolve this issue. Figure 5.1 depicts how the network is obscured.

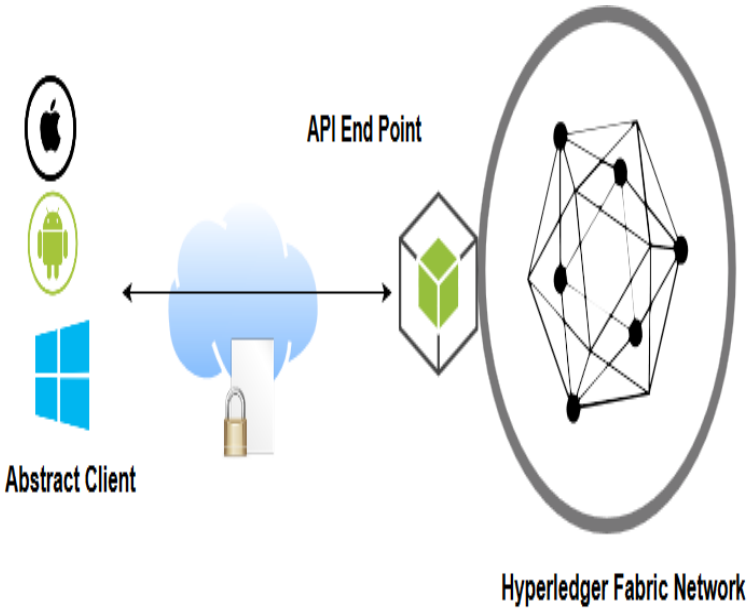

Figure 5.1: Obscuring the Network by Using the Node.js App

5.2 Future Work

Based on our reusable and modular design we could adopt any complex business logic by simply modifying the chaincode. Furthermore, we could introduce more roles such as, Researchers and verifiers who could have more privileges like banning other users who scam the platform and prevent any malicious behavior.

Chapter 6

Conclusion

In this chapter, we conclude the work has been done in the research project and all tasks that have been accomplished.

6.1 Summary

During the research project, We aimed to implement a real solution leveraging the blockchain features such as decentralization and data immutability. In addition, Providing a solution for a real problem like Fake news. As Fake news in the digital era is one of the latest issues that has raised many concerns and has many negative effects for instance, biasing the public opinion, spreading the hate speech and destroying innocent people's reputation.

We started by studying the blockchain and hyperledger fabric. We designed a case study scenario for a mobile application that incents users to share and rank stories and storing them on the blockchain. Hyperledger fabric was used as a permissioned blockchain that restricting the access into the network and only the verified entities could participate.

We built a fully functioned development network using Docker containers and adopt one of the provided node.js APP, which abstracts the whole network and exposes Rest APIs for different chaincode operations on the hyperledger fabric,The node.js App also serves as a back-end to the whole infrastructure after highly modifying the code to match our case. We built a simple Xampp development server in order to allow the users to register in a traditional way using email and password, Furthermore we introduced an efficient way for storing data on the blockchain by storing only the hash of the media files, and serving the media files as content delivery network CDN approach for more efficient, low latency and high performance. In addition, We implemented a score and reputation system using the chaincode and location trustworthiness. We designed an Android application for the end user that facilitates the interaction and allows different operations. The application designed to be user-friendly, self-explanatory and easy to use. Finally, we introduced a modular design thus more features with the minimum required modifications will be easy introduced and integrated into an inter-operable environment.

Bibliography

[1] Hyperledger Fabric architecture Hyperledger-fabric.readthedocs.io.\T1\textquotedblleftArchitectureExplained\T1\textquotedblright, Hyperledger-fabricdocsmasterdocumentation,2017

[2] Hyperledger Fabric Balance transfer network https://github.com/hyperledger/fabric-samples/tree/release-1.4/balance-transfer

[3] Hyperledger Fabric Node.js SDK https://fabric-sdk-node.github.io/release-1.4/index.html

[4] Node.js APP Source Code https://gitlab.ibr.cs.tu-bs.de/ds-thesis/2018-pa-hesham-mohamed-media-blockchain/tree/network-configurations/app

[5] Chaincode Rest API calls https://gitlab.ibr.cs.tu-bs.de/ds-thesis/2018-pa-hesham-mohamed-media-blockchain/wikis/API-Calls-Documentation

[6] A guide for setting up the infrastructure https://gitlab.ibr.cs.tu-bs.de/ds-thesis/2018-pa-hesham-mohamed-media-blockchain/wikis/How-to-setup-the-infrastructure.

[7] Android Documentation https://developer.android.com/docs

[8] Hyperledger Composer REST Server https://hyperledger.github.io/composer/v0.19/reference/rest-server

[9] Bitcoin White Paper https://bitcoin.org/bitcoin.pdf

[10] Blockchain vs Tranditional Database https://www.researchgate.net/publication/327483781_Blockchain_Versus_Database_A_Critical_Analysis

[11] Blockchain Overview https://www.researchgate.net/publication/318131748_An_Overview_of_Blockchain_Technology_Architecture_Consensus_and_Future_Trends

[12] Hawk: The Blockchain Model of Cryptography and Privacy-Preserving Smart Contracts https://www.computer.org/csdl/proceedings-article/sp/2016/0824a839/12OmNs59JYD

[13] Hyperledger Fabric: A Distributed Operating System forPermissioned Blockchains https://arxiv.org/pdf/1801.10228.pdf

[14] All roads lead to Rome:Many ways to double spend your cryptocurrency https://arxiv.org/pdf/1811.06751.pdf

[15] Using CouchDB on Hyperledger Faric https://hyperledger-fabric.readthedocs.io/en/release-1.4/couchdb_tutorial.html