UNIVERSITY OF PIRAEUS

School Of Information and Communication Technologies
Department Of Digital Systems

Undergraduate Program Of Studies

Developing an Ethereum Based

Blockchain Camera

Dimitris Vagiakakos

Supervising Professor: Christos Xenakis

PIRAEUS

JULY 2022

FINAL YEAR PROJECT

Developing an Ethereum Based

Blockchain Camera

Dimitris Vagiakakos

Student ID: E18019

Abstract

Blockchain Camera comes as an idea of Developing an inexpensive device based on Raspberry Pi, using Python3 with a Blockchain backend smart-contract running in Ethereum Network written in Solidity. Blockchain Camera provides an easy and safe way to capture and guarantee the existence of videos reducing the impact of modified videos as it can preserve the integrity and validity of videos. A Blockchain Camera could be useful to private infrastructure, so as to ensure that a video is authenticate and it has not been modified since the Blockchain records the hash of each video and the time the hash has been sent to Blockchain. For validation purposes, users using a Blockchain Camera Validator Tool are able to verify a video that is genuine. Blockchain Camera is ready to be applied not only in policemen's gear in order to protect them and the civils from corruption but also it could be used of the police from the entire European Union. An another important application idea is to be used as a Car Protection Camera or as a CCTV in shops. Blockchain Camera software not only can be used as a stand-alone device, but also can be used in any GNU/Linux system running in old netbooks, laptop or computers as a Blockchain Camera. Blockchain Camera comes under GNU GPL v3.0+ software license.

SUBJECT AREA: Blockchain Camera, Blockchain, Linux, Smart-Contracts

KEYWORDS: Blockchain, Ethereum, Camera

Acknowledgements

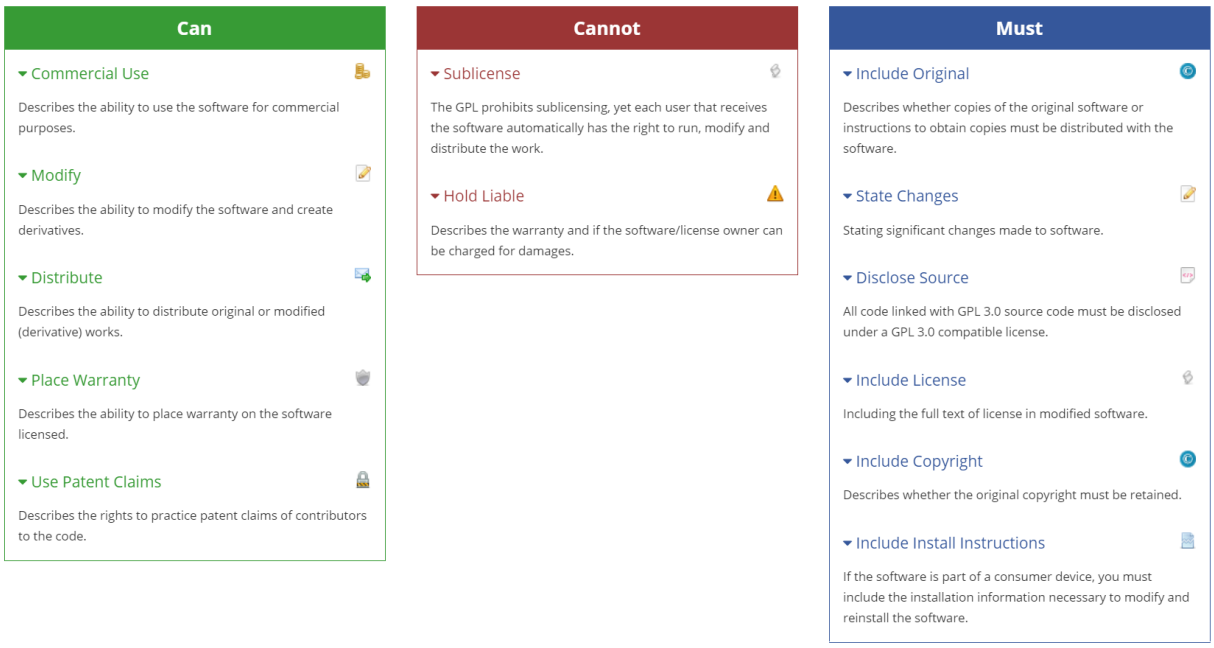
First and foremost, I would like to express my sincere gratitude to my professors Demetrios Sampson and Stefanos Gkrintzalis for their motivation as well as my supervisor, Prof. Christos Xenakis for his inspiration with the Blockchain Technology and his assistance on the Blockchain researching Projects we run in University Of Piraeus and of course to my colleague and friend Stavros Gkinos for researching together all of these years and support my idea of developing a Blockchain Camera.

Moreover, I owe my deepest gratitude to the community of Linux-User.gr and especially to Spyros Kosyvas who with his assistance I overcome a major problem with ffmpeg recording tool. Furthermore, I would like to thank Panos Vasilopoulos, Katerina Damianou and Vaios Bolgouras for their advices and ideas.

Finally, I would like to acknowledge the support of my family and friends.

Table Of Contents:

1. License



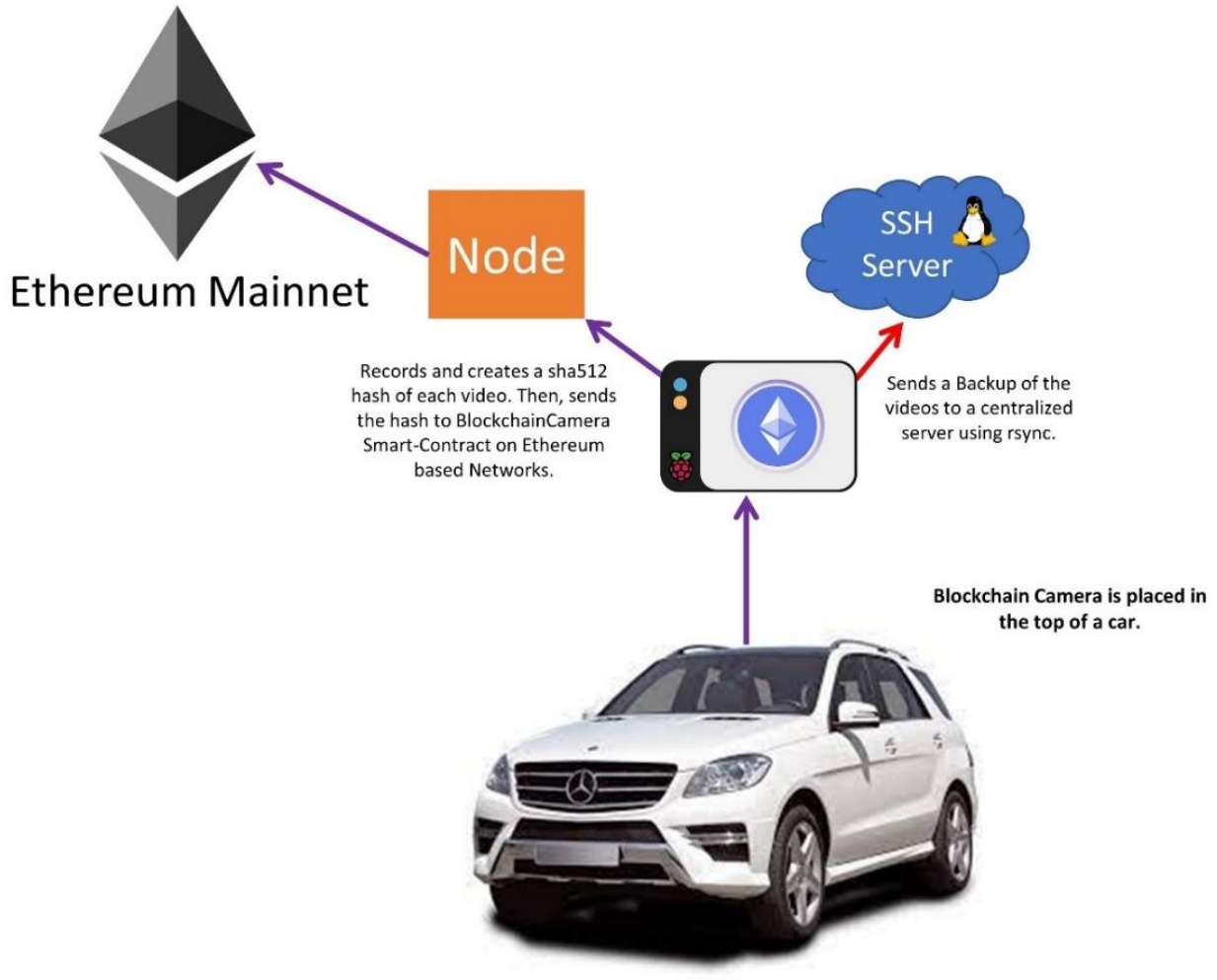
Picture 1: GNU General Public License v3.0 summary by tldrlegal.com.

2. The idea of Blockchain Camera

Back in March 2021 in Greece, the Athens police reported that when a police patrol entered New Smyrna to investigate reports of Covid-19 lockdown violations, they were subsequently attacked by a group of 30 people, injuring 2 officers, leading to the detention of 11 people. However, videos of the event emerged, showing peaceful protestors being thrown to the ground by the police and attacked with batons. Being inspired by that case, as 2 different sides had an extremely different perspective about what happened in New Smyrna, I understood that is of highly importance for police to add cameras on their gear in order, at first to protect themselves, and then to protect the citizens from police corruption. However, with the “deepfake” videos ameliorate every year and due to the fact that how easily can everyone on these days record a video and then modify it, a camera like this must be able to validate that the data recorded from there have not been modified. There comes the idea of using Blockchain Technology to store the hashes of recorded videos in order to be able to validate that a video is authentic and hasn’t been modified and be able to see the time that the hash of the video has been sent to Blockchain. We need the time because it will help us to know how many minutes have passed since the incident occurred. Therefore, if we know for example that a video has been recorded at 14.15 and its hash has been sent to Blockchain at 14.16, we know that the video is surely authentic. But, if the video will be sent at 14.50 for example, we know that they had enough time to modify the content of the video and then hash it in order to send the hash to the Blockchain, so we know that there is a higher possibility for a video to not be genuine.

2.1 Blockchain Camera Usage Ideas

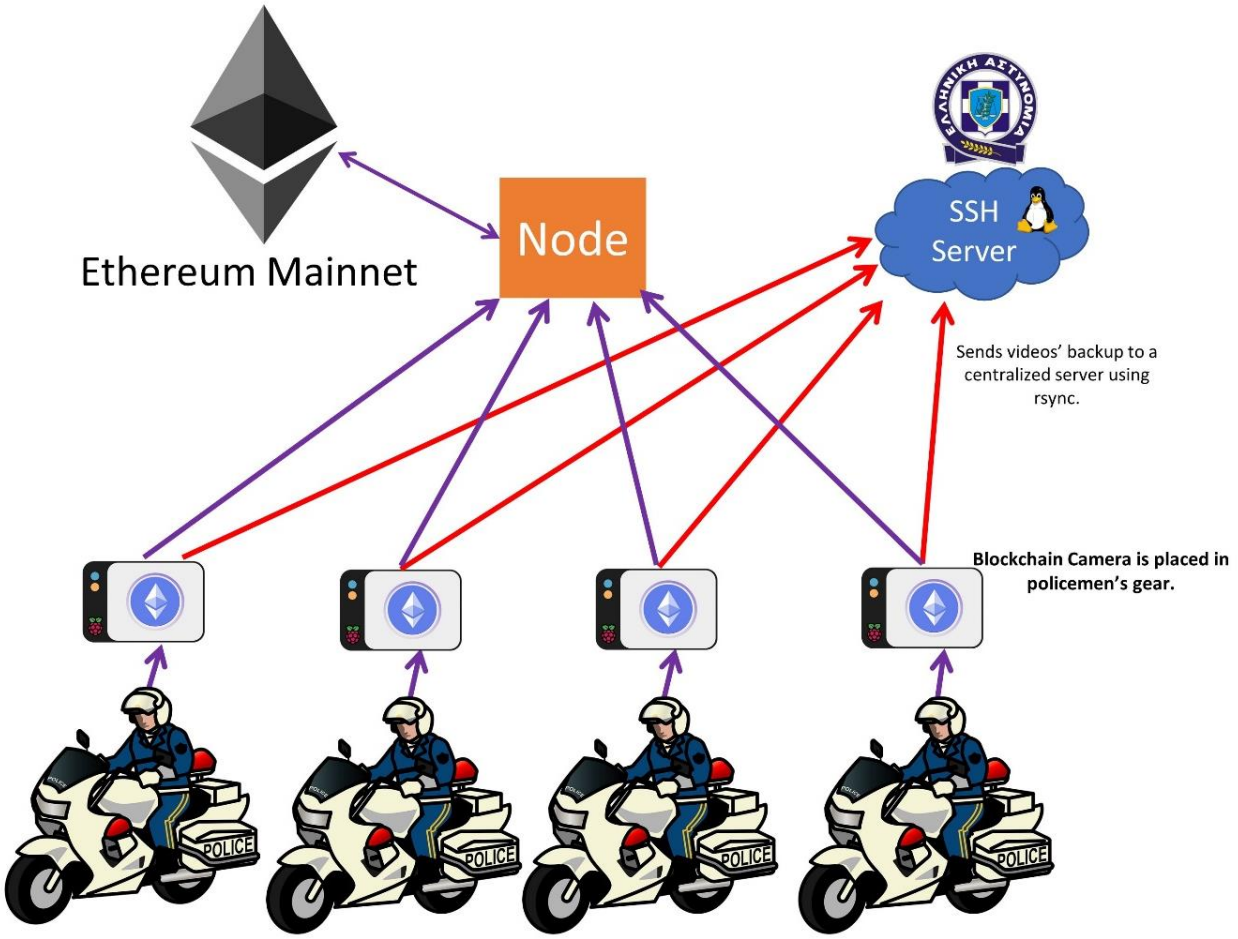
2.1.1 Blockchain Camera as Car Protection Camera



Picture 2: Car Protection Camera using Blockchain Technology to ensure the validity of its videos.

A stand-alone Blockchain Camera Device has many applications to many different sectors in which the integrity and the validity of the information is of utmost importance. For instance, if cars had their own Blockchain cameras integrated, in case of a car accident, we will have been able to see the truth about how an accident had really happened and we will be sure that the videos are genuine. In case of an accident, we simply get the videos from both cars and then we check their sha512 hashes. If they are valid and they have been submitted to Blockchain in about the same time and this information will be enough to ensure what really happened in the accident without the need of finding other potential witnesses. As on this implementation we use only one Blockchain Camera per car, the user can use a node infrastructure for companies like Infura in order to avoid setting up their own nodes. The same methodology will be followed for example if we would like to create a CCTV and protect a local store or our own home in case of a burglary.

2.1.2 Blockchain Camera as Police Body Camera

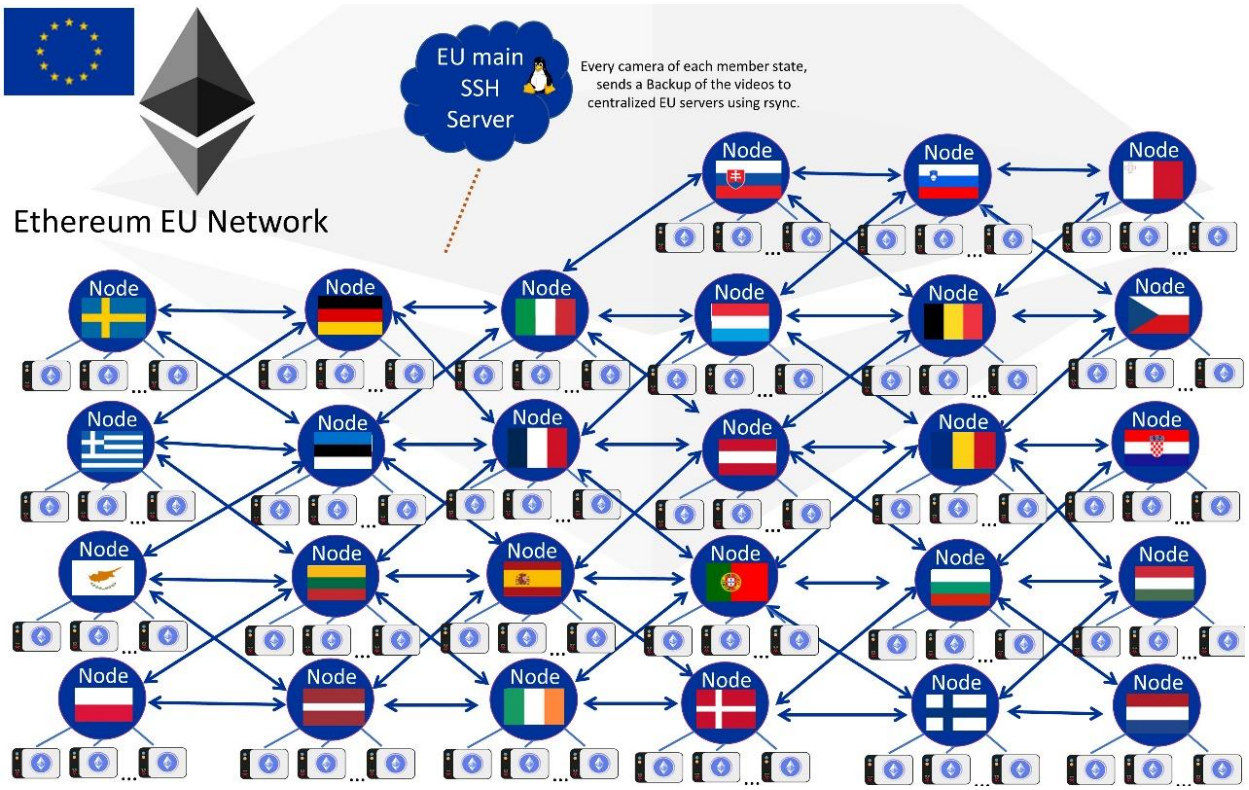


Picture 3: Police Body Camera using Blockchain Technology to ensure the validity of its videos demonstrated in Hellenic Police.

Furthermore, as it has already been mentioned, Blockchain Camera's idea came from protests and conflicts between citizens and policemen. Based on this idea, if we tried in Hellenic Police, Hellenic Republic could setup nodes to each administrative region. Policemen would use the node from their administrative region in order to be able to interact with Blockchain and use the Blockchain Camera. Each policemen's gear will have a Blockchain Camera recording continuously and then it will send the hashes of the videos to Ethereum Blockchain and the videos to each administrative region's main ssh server. This idea can be extended rapidly if Hellenic Republic creates its own Ethereum Based Blockchain and then setup to each city host its own node in order to make it more secure and decentralized. Moreover, through this implementation, Hellenic Republic will not even have to use Mainnet Ether which costs real world money. Computers from each city will be validating blocks and then will be earning their own Ether and they will be able to send Ether to each policeman's Blockchain Camera's wallet in order to have enough funds to be functional. With this implementation, we will know if a video from a policeman is authentic and we will also know the exact time in which an event really happened and its hash has been sent to Blockchain. As a result, the problem with police corruption will be more limited and the citizens will not have a point to modify videos for provocation as we will have videos with proofs about when an event really took place and the police's video is authentic and its hash has been sent to Blockchain with the time of the submit.

.

2.1.3 Blockchain Camera in European Union infrastructure



Picture 4: EU's Ethereum Based Blockchain with a lot of Blockchain Cameras to ensure the validity of the videos.

A more extended version of the example of Hellenic's Police Blockchain can be applied to entire European Union and create its own Ethereum Based Network running nodes to all countries' state administrator regions and cities. Creating and using a Pan-European Blockchain could benefit the entire EU as it creates a safe, economic and secure infrastructure to ensure the validity of information and also creates a European standard in security as the Blockchain Network became more and more decentralized as further nodes are running and validating blocks in EU. From that point of view, the corruption in European policemen or other critical locations will be more limited as there will be evidence for an incident that really happened or not, so corrupted people in critical thesis will have second thoughts before applying something inappropriate and the same will happen with corrupted citizens who intend to modify videos and create fake news and events. It should be mentioned that this study focuses about how a Blockchain Camera infrastructure to insure the validity of the data can be designed and developed. The legal obligations that arise from the duration of storing the videos for example in GDPR or other countries regulations are outside the scope of this study.

3. Introduction

3.1 What is Blockchain

To begin with, we have to explain what Blockchain is. Blockchain is a decentralized Peer-to-Peer (P2P) technology where it keeps record of the transactions, without anyone in charge controlling it, so that all transactions-information to be available to all users on the same network. This public ledger is stored on each node, building a decentralized system

without anyone who can control or manipulate the network, in contrast with a classic centralized system, where the ledger is stored for example on the central server of a bank. Each of them includes the cryptographic hash value (usually 256-bit) of the previous block in the Blockchain, resulting in the creation of a chain of blocks.

Blockchain is a revolutionary technology, as it enables to set up decentralized services which cannot be modified. As a result, network becomes more reliable than possible third-party modifications where they could occur in traditional centralized networks.

On a Blockchain Network, each Block of its Blockchain contains:

- Its data.
- Its hash value (a 256bit hash most of the times) which is being calculated by its data.
- The hash of its previous Block.

It should be mentioned that the very first block of a Blockchain Network has not any hash from a previous block (as it is the first block) and it is called Genesis Block. Furthermore, in case of a conflict (For example, Stavros spends the same amount of a cryptocurrency to 2 different wallets on the same network), Blockchain always follows the chain with the longest length.

3.2 Hashing

Hashing is the process of scrambling raw data to the point where it can no longer be reproduced on its original form. Hashing tools take a piece of information and passes it through a function that performs mathematical operations on the plaintext, ending up with a fixed-length non reversible string different from the original, called hash value or digest. The hash value is a summary of the original data. Even the smallest change (one bit) makes the whole hash distinctive. There are a large variety of hashing algorithms that are being used. The most famous examples are MD5 and SHA-2 algorithms.

Blockchain Camera hashes the videos it captures with SHA-512 algorithm.

The selection of SHA2-Family hash algorithms has been done as SHA-2 algorithm satisfies the following requirements that are of highly importance in order to create secure hashes.

- It should be fast to compute the hash value of any kind of data.
- It should be impossible to regenerate the data from its hash value.
- It should offer the lowest risk for hash collisions. Hash collision is when two pieces of data share the same hash value. The hash value in this case is derived from a hash function which takes a data input and returns a fixed length of bits.

3.3 Types of Blockchain Networks

3.3.1 Public Blockchain

Public Blockchains are the most widespread type of Blockchains since they promote transparency as they are public networks. The information of a Public Blockchain Network is available for everyone. Everyone has the ability to see, read and write data on a Public Blockchain. The data of Public Blockchains are accessible by everyone. However, that does not mean anyone can modify the data of a Public Blockchain. Moreover, as all data are publicly to everyone, the network remains Decentralized since everyone can have a copy of the Blockchain on his computer and run a node or even mine/validate blocks without a third party such as a company or an organization can control it. As a result, Public Blockchains can provide security and availability as once the information passes through the network and cannot be modified or deleted, once the entry has been validated

on its Blockchain network. On this type of Blockchain, anyone can participate in network and verify transactions. A few examples of Public Blockchain are Bitcoin, Ethereum, Cardano etc.

3.3.2 Private/Permissioned Blockchain

Unlike Public Blockchains, Private or Permissioned Blockchains have restrictions on who can read, create and validate transactions in the network. In Private Blockchains, not everyone is able to download a copy of the stored data from the Private Blockchain as well start producing or confirming blocks. In this provision, central authorities (for example, a company) controls the members/nodes of the network for the permissions they have over them in this particular Blockchain. This strategy ensures reliability of nodes, however the network is centralized. The most famous type of Private Blockchain is Hyperledger.

3.3.3 Consortium Blockchain

An another type of Blockchain is the Consortium Blockchain, which is a type of Blockchain between Private and Public Blockchains. Consortium Blockchain is a semi-private Blockchain as only default participants-users they have access to it and therefore it is not accessible to everyone but only to the above users. This type of Blockchain Networks is ideal when different companies use the same Blockchain. In addition, this kind Blockchain gives the right to read and write to multiple companies belonging to the Blockchain consortium network. The Quorum is one of the Consortium Blockchain networks.

3.4 Consensus Mechanisms

3.4.1 Proof Of Work

Proof of Work (PoW) is the algorithm that secures many Blockchain Networks, including the first version of Ethereum and Bitcoin. For the purpose of adding a new block, which bother transactions waiting for validation, into the Blockchain miners need to solve a "puzzle" using their computational power. A new block is accepted by the network each time a miner comes up with a new winning Proof Of Work, which is so difficult that miners use expensive and specialized computers. These computers require and utilize a large amount of electrical power, that provokes the environment, which means that countries providing inexpensive electric power could mine cryptocurrencies, using PoW, in much lower cost. Consequently, these states dominate each PoW Blockchain Networks leading up in a more centralized network. Proof of Work Blockchains provide adequate security only if there is a large network of miners competing for block rewards in different geographical locations. As has been mentioned before, in case of a conflict from the chains, we have always to trust and follow the longest chain of blocks because that it has the greatest computational work of all conflicted Blockchain. Suppositionally, if the network in a large area, that hosts a considerable number of miners (also known as Mining Pools), breaks down the PoW Network is losing a significant quantity of miners that make this system safe, as a result the possibility that a hacker could gain a simple majority of the network’s computational power and stage grows up, what is known as a 51% attack.

3.4.2 Proof Of Stake

Proof of Stake (PoS) is a type of consensus mechanisms by which a Blockchain Network achieves distributed consensus. In PoS-based Blockchain Networks the creator of the next block is chosen via various combinations of random selection of wealth or age. While the overall process remains the same as Proof Of Work, the method of reaching the “target” is different. In Proof of Stake, instead of miners from Proof of Work, there are validators. The Blockchain keeps track of a set of validators, and anyone who holds the

Blockchain's base cryptocurrency (in Ethereum's case, ether) can become a validator by sending a special type of transaction that locks up their ether into a deposit. The validators take turns proposing and voting on the next valid block, and the weight of each validator's vote depends on the size of it's deposit (i.e., stake). Importantly, a validator risks losing their deposit if the block they staked it on is rejected by the majority of validators. Conversely, validators earn a small reward, proportional to their deposited stake, for every block that is accepted by the majority. The validators lock up some of their Ether as a stake in the ecosystem. The validators bet on the blocks that they will be added next to the chain. When a block gets added, the validators get a block reward in proportion to their stake. The main advantages of the Proof of Stake algorithm are energy efficiency and security. A greater number of users are encouraged to run nodes since it is easy and affordable. This along with the randomization process also makes the network more decentralized since mining pools are no longer needed to mine the blocks. Thus, PoS forces validators to act honestly and follow the consensus rules, by a system of reward and punishment. The major difference between PoS and PoW is that the punishment in PoS is intrinsic to the Blockchain (e.g., loss of staked ether), whereas in PoW the punishment is extrinsic (e.g., loss of funds spent on electricity). Some examples of PoS Blockchains are Cardano, Polkadot and the second version of Ethereum.

3.5 UTXO and ABOT

3.5.1 UTXO - Unspent Transaction Output

The UTXO (Unspent Transaction Output) model shares many similarities to the traditional banking system about how transactions are being recorded. At UTXO, the output of the unspent transaction is the result of the cryptocurrency amount the user spends and will receive in the future. Each UTXO can only be sent at once. That means once used, it cannot be reused in the future. The Validation of each transaction is necessary to maintain the security and privacy in Blockchain Networks, as at UTXO each transaction is considered as valid if it meets the following conditions:

- Each transaction input must be signed by its owner before being spent.
- If a transaction has multiple inputs, then each of the inputs must be signed with each input's owner.
- A transaction is valid only if the value of each input equals or exceeds the total value of the inputs.
- **Scalability:** UTXO allows parallel transactions to process multiple other UTXOs at the same time.
- **Privacy:** UTXO can maintain high levels of privacy as long as it uses a different address in each transaction.

3.5.2 ABOT- Account Based Online Transaction

The ABOT (Account Based Online Transaction) model appeared for the very first time in Ethereum. Compared to UTXO, ABOT is a simpler model. ABOT works exhaustively on all transactions so to improve consensus efficiently as well as to create and validate the blocks faster. Each transaction with a token is verified only if:

- The token is signed by the sender.
- The ownership of the sender's token can be verified.
- The sender's accounts have a sufficient balance to pay for the transaction.

After the transaction is validated, the sender is charged and the value is credited to the recipient's account. In Ethereum Network, the user's balance is a sum of Ether (the cryptocurrency of the Ethereum Network) where the user's private key so that a valid

signature can be generated. The advantages of using the ABOT model in Blockchain networks are the following:

- **Simplicity:** The Account/Balance model does not force transactions to hold states, while keeping the model simple.
- **Efficiency:** The Account/Balance model is efficient because the transaction needs only to prove that has a sufficient balance in the account to pay the transaction.

4. Ethereum

Ethereum is an Open-Source public Blockchain that it works as a global decentralized computer infrastructure. As a decentralized computer, Ethereum is able to execute decentralized applications, called smart-contracts. Smart-contracts are deployed in a runtime environment, called Ethereum Virtual Machine. In the same way as other popular Blockchains ensures the validity of their data, the validity of each Ether is provided by Ethereum Blockchain, which is an incessantly growing list of records called blocks, which are linked and secured using hash functions, like Keccak-256 hash. The private keys and addresses are stored in a cryptocurrency wallet. Examples of Ethereum Based wallets are Metamask and MyEtherWallet. Addresses can be used to make transactions using Ether. Due to the decentralized framework of the Ethereum, developers can build decentralized applications with built-in economic functions, as it provides high availability, auditability, transparency and neutrality, while simultaneously reduces or eliminates counter intelligence and certain counter party risks.

4.1 Ether

As previously noted, Ether is the main cryptocurrency of the Ethereum Network and is being used as a reward to the miners or validators for validating blocks in the Blockchain. The block reward together with the transaction fees is the inducement of the validators to keep the Blockchain growing and so Ether is fundamental to the operation of the network. Each Ethereum wallet with its private key has an ether balance and it is able to send ether to other wallets or smart-contracts. Ether as a measuring unit has a high value so it can be splitted into sub-units. The smallest amount of Ether which can be spent is 1 Wei which is 0,000000000000000001 Ether.

4.2 Ethereum Addresses

Ethereum addresses are consisting of the prefix "0x" concatenated with the rightmost 20 bytes of the Keccak-256 hash of the ECDSA (Elliptic Curve Digital Signature Algorithm). Ethereum implements the ECDSA encryption algorithm to ensure that funds can only be spent by their rightful owners. A byte in hexadecimal is represented by two digits, therefore addresses have 40 hexadecimal digits. The format of smart-contract addresses is the same, however they are determined by the sender and the creation transaction nonce. User accounts are indistinguishable from contract accounts, given only an address for each and no Blockchain data. Any valid Keccak-256 hash put into the described format is valid, even if it does not correspond to an account with a private key or a contract.

4.3 Gas Fees

Gas is a unit of account within the EVM used in the calculation of a transaction fee, which is the amount of ETH a transaction’s sender must pay to the miner or validator who includes the transaction in the Blockchain. Practically, Gas is a separate virtual currency with its own exchange rate with Ether. The price of gas has relevance for the time for a

transaction to be confirmed. The higher the gas price the faster the transaction is likely to be confirmed. As a matter of fact, gas prices can be set to zero as they are not prohibited by the protocol however is very unlikely to be validated before they expire.

4.4 Transactions

Transactions are data committed to the Ethereum Blockchain signed by an originating account, targeting a specific address. The transaction contains metadata such as the gas limit for that transaction. By creating transactions in Ethereum Network, not only we can send ether to other addresses, but also we can call functions from decentralized applications called smart-contracts or even deploy our own decentralized programs to Ethereum Based Blockchains.

4.5 Nodes

In Blockchain Networks, there are multiple nodes that store the complete history of a Blockchain to observe and enforce its rules. Once the nodes validate a transaction, it is shown as pending state until a miner or a validator picks and then confirms each pending transaction and then it adds the transaction to the main chain. Moreover, there are nodes in the network called supernodes which work only as communication bridges by providing the Blockchain data to other nodes. For example, Software light wallets like Metamask need a node in order to be able to create and send transactions. They don't store a full copy of the Blockchain locally, they only download the headers of each block. For instance, Metamask uses Infura's infrastructure nodes in order to communicate and interact with Blockchain. Software light wallets do not store a copy of the Blockchain locally or validate blocks and transactions. It only offers the functions of a wallet and can create and broadcast transactions. Full node wallets on the other hand, stores a full copy of Blockchain. By using a full node wallet, user can be more independent and surer that his transaction will end up to a miner or validator, making the user more independent, the network more decentralized and enhance user's privacy.

4.6 Smart Contracts

Smart contracts are decentralized computer programs which have some special features. The fact that Smart Contracts are immutable is their most extensive feature. Once deployed, the code of a smart-contract cannot be modified or change. The only way to alter the code of a smart-contract is to deploy a new instance. Moreover, the outcome after the execution of a smart-contract cannot be contradictory. The outcome of the execution is the same for everyone who runs the smart-contact. The EVM runs as a local instance of every Ethereum node, however as all the instances of the EVM operate on the same initial state and produce the same final state. That makes the Ethereum Network a Decentralized "world computer" running Decentralized Applications. Importantly, contracts only run if they are called by a transaction.

4.7 Ethash

Ethash is the Proof Of Work function in Ethereum-first generation-based Blockchain Networks. The Ethash algorithm relies on a pseudorandom dataset, initialized by the current Blockchain length. This is called a DAG and is regenerated every 30,000 blocks. Use of consumer-level GPUs for carrying out the PoW on the Ethereum Network means that more people around the world can participate in the mining process. The more independent miners there are the more decentralized the mining power is, which means situations like in Bitcoin can be avoided, where much of the mining power is concentrated in the hands of a few large industrial mining operations. In fact, there is a deliberate handicap on Ethereum's Proof of Work called the difficulty bomb, intended to gradually

make Proof Of Work mining of Ethereum more and more difficult, thereby forcing the transition to Proof of Stake.

4.8 Casper – Ethereum's Proof Of Stake Algorithm

As has already been mentioned, Ethereum Network is about to switch from Proof Of Work to Proof Of Stake. Casper is the Proof of Stake function in the second generation of Ethereum Blockchain. Casper comes to be a more scalable solution for Ethereum Network, which will be able to process a larger number of transactions per second in order to run more decentralized applications simultaneously. Another principal advantage of Casper is the Blockchain's increased security. A 51% attack is difficult to carry out with PoS thanks to validators' punishment system as validators have minimum to deposit 1500ETH. The main problem with Proof Of Work mechanism in Ethereum is that the execution of many transactions have led to an increase in transaction time and cost. Moreover, there are environmental concerns as PoW requires a large number of miners with very powerful computer equipment to compute and create blocks. As a result, Casper will make Ethereum Network more sustainable, scalable and decentralized. There are 2 implementations of Casper available. Friendly Finally Gadget (FFG) is the first step in transitioning the Ethereum Network from PoW to PoS using a hybrid PoW/PoS model in order to support an easier transition. Furthermore, in Correct by Construction (CBC) Capser relies on a safety oracle to adjust a partially constructed PoS protocol until the system be ready to switch to a full functional PoS system.

4.9 Solidity

Solidity is an object-oriented, high-level language, explicitly for writing smart-contracts with features to directly support execution on various Blockchain platforms and especially, in the decentralized environment of the Ethereum world computer. Solidity was influenced by C++, Python and JavaScript program languages and is designed to target the Ethereum Virtual Machine (EVM). The main "product" of the Solidity project is the Solidity compiler (solc) which converts programs written in the Solidity language to EVM byte code. The language is still in constant flux and things may change in unexpected ways to resolve such issues. Solidity offers a compiler directive known as a version pragma that instructs the compiler that the program expects a specific compiler (and language) version. The Solidity compiler reads the version pragma and will produce an error if the compiler is incompatible with the current version pragma. Also, it should be mentioned that minor updates of Solidity (For example, from 0.8.0 to 0.8.15) are compatible with their main version (0.8.x.). Each Pragma directives are not compiled into EVM bytecode. Version pragma is only used by the compiler to check the compatibility. In order to add version pragma, we type: Pragma solidity ^ VERSION_NUMBER;

4.10 Metamask

Metamask is a lightweight wallet available for both Android, iOS and desktop web browsers. The web browser version of Metamask comes as a browser extension that serves as the primary user interface to Ethereum Network. Once a user installs Metamask on his web browser he can store ether or other ERC-20 tokens and make transactions to any Ethereum address in any Ethereum Based Network. Furthermore, the wallet can also be used by the user to interact with Decentralized Applications or even deploy his own smart-contracts to Blockchain. Metamask can be connected to multiple Ethereum Based Blockchain Networks, for example to the official main Ethereum Blockchain or Binance's Smart-Chain or even in test networks to help the developers or testers to test and debug smart-contracts to Ethereum Test Networks, before launching them to a Mainnet Network. Examples of Ethereum Based Testnets are Rinkeby, Ropsten etc. or even we can

connect Metamask in our own testing node running to our computer with utilities like Ganache.

4.11 Remix – Ethereum.org IDE

Remix is the official online IDE provided by Ethereum.org and it can be used to create, test, compile and deploy Ethereum smart-contracts in the Solidity programming language. It has several built-in Solidity compiler versions and it runs from the web browser. Remix IDE is extremely helpful as it supports from browser to users the ability to code, test, debug and then deploy their smart-contract to Ethereum Mainnet or any Testnet, JavaScript VM via browser or even in our own testing node running to our computer with utilities like Ganache. Blockchain Camera Smart-Contract can be compiled and deployed to any Ethereum based Network by using Remix IDE. In the example on this paper, we will deploy Blockchain Camera Smart-Contract to Ropsten Test Ethereum Network.

4.12 Ganache

Ethereum smart-contracts are programs executed within the context of transactions on the Ethereum Blockchain. Ganache allows users to recreate a Blockchain environment in their own local computers and test the smart-contracts they have created. It simulates the features of a real Ethereum Network, namely it generates several addresses and each one of them owns 100 ether with their own private key. Whenever a transaction is performed, it gets added to block in order to show the current block number in that Blockchain.

4.13 Web3 development libraries

Web3 represents a new vision and focus for web applications; from centrally owned and managed applications to applications built on decentralized protocols. The main advantage of Web 3 is that it attempts to address the biggest problem that has resulted from Web 2: the collection of personal data by private networks which are then sold to advertisers. With Web 3, the network is decentralized, so no such entity controls it, and the Decentralized Applications (DApps) that are built on top of the network are open. The devoutness of the decentralized web means that no single party can control data or limit access. Anyone can build and connect with different Decentralized Applications without permission from a central company. There are multiple ways to interact with Ethereum Blockchain. The most famous choices are the libraries web3.js and web3.py. Web3.js which is a collection of libraries that allow users to interact with a local or remote Ethereum Node Using HTTP, IPC or WebSocket. Similarly, web3.py can be used for interaction with Ethereum Networks using Python3. In Blockchain Camera, web3.py will be used in the Blockchain Camera to create transactions and send the hash of each video to the Blockchain Camera's Smart-Contract. From the user's perspective, web3.py will be used in the client, Blockchain Camera Validator Tool (GUI or CLI), in order to search in the Blockchain Camera Smart Contract and find if a hash exists and the time of its validation.

5 Blockchain Camera Tools and Hardware

5.1 ffmpeg

FFmpeg is a free and open-source software project consisting of a suite of libraries and programs for handling video, audio, and other multimedia files and streams. At its core is the command-line ffmpeg tool itself, designed for processing of video and audio files. It is widely used for format transcoding, basic editing (trimming and concatenation), video scaling, video post-production effects and standards compliance (SMPTE, ITU). On this project, ffmpeg will be used to record videos for a user defined duration of time and then encode the videos to mp4-h264 or mp4-h265 format. The default and recommended duration of each video is 10 minutes as the less the time duration is chosen the more hashes will be created and the device administrator will have to pay more transaction fees.

5.2 Blockchain Camera Software

5.2.1 How Solidity code works

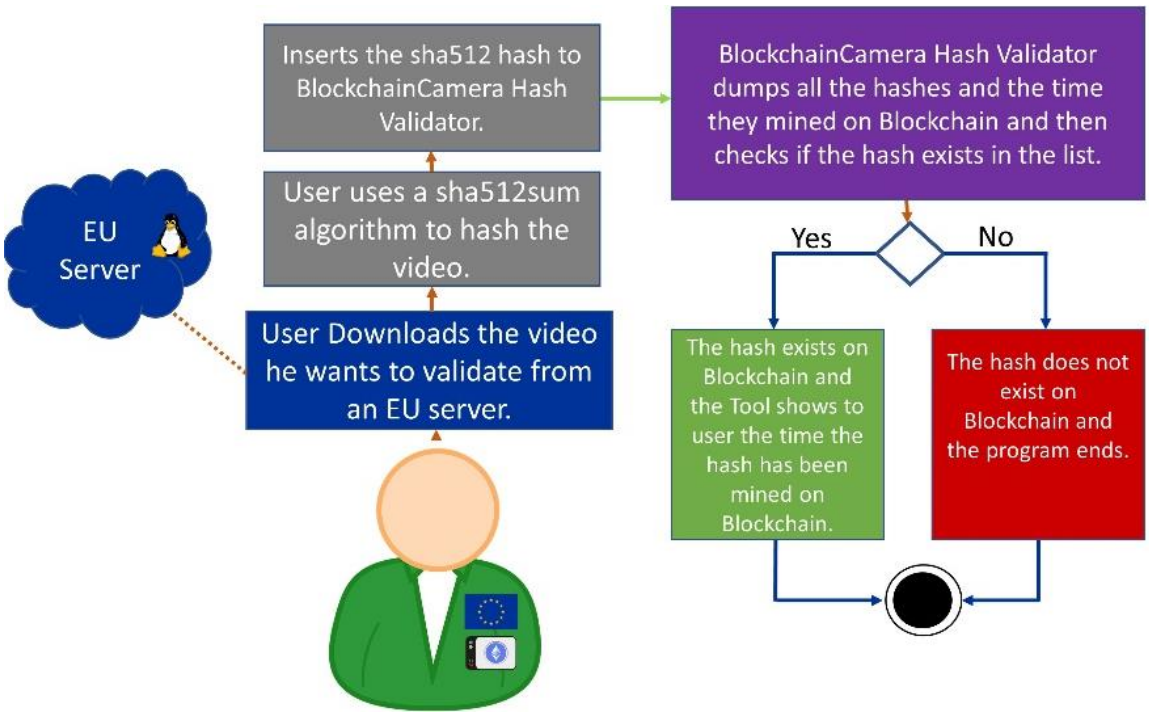
Blockchain Camera Smart-Contract has been written in Solidity and as it has been mentioned before, it is compiled and deployed using Remix IDE. Blockchain Camera Smart-Contract provides the backend Blockchain support of the Camera, as technically it guarantees the Proof of Existence on the videos that have been recorded from this Camera.

Ethereum Virtual Machine has three types of storage in order to store the data in the Ethereum Network. Each of the accounts, has its own storage which is persistent between function calls and transactions. Storage is a key-value store that maps 256-bit words to 256-bit words. Furthermore, as initializing or modifying storages in the Ethereum Network is quite expensive, especially when we want to deploy or modify. As a consequence, due to the high cost, we should minimize what we store in persistent storage to what the smart-contract really needs to run. We store the real data outside of the Blockchain and we only keep its video hash value on Blockchain and the time that the hash has been sent to Blockchain to be able to verify and prove that they are real if we want to validate the integrity of a video file. The second data area is called memory of which a smart-contract obtains a freshly cleared instance for each message call. Memory is linear and can be addressed at byte level, but reads are limited to a width of 256 bits, while writes can be either 8 bits or 256 bits wide. Memory is more costly the larger it grows as it scales exponentially. The Ethereum Virtual Machine is not a register machine but a stack machine, so all computations are performed on a data area called the stack. Stack has a maximum size of 1024 elements and contains words of 256 bits

For the above reasons, the focus of Blockchain Camera is to send the hashes from the videos to Blockchain providing us a way to validate the integrity of a video. Blockchain Camera Smart-Contract has 2 private arrays, one for storing the sha512 hashes from the videos and one for storing the time that each of the hashes, has been sent to Blockchain. It should be mentioned here that the time is not placed from the user, but the smart-contract takes the timestamp from the latest block in order to be impossible for someone to modify the timestamp from when the hash has been submitted by the Blockchain Camera. Moreover, Blockchain Camera Smart-Contract has 2 variables, one public for storing the address of the owner of this smart-contract, as the deployment wallet will be the only one which will create transactions and send hashes on this smart-contract. Moreover, an extra variable is used for storing the sum of the total hashes that are stored on that smart-contract. Constructor runs only at once, under the deployment of the smart-contract and sets the person or smart-contract who deployed the Blockchain Camera Smart-Contract, as the owner of the smart-contract. Then, it adds the string “start” as the

very first value stored in the **hashes** array. Then, **index** is increased by one and the timestamp of the block has been added to the array **time_of_hash**. For this time and now, we know exactly the time that the Blockchain Camera Smart-Contract has been deployed and it is ready to store hashes from the web3.py backend from the Blockchain Camera. Furthermore, as the variables are private, in order to view the arrays, user can call the view() functions and find whatever he wants directly from WebTools provided from Metamask or for a better experience, he can use the Blockchain Camera Validator Tools (Python Console Mode or Python GUI Mode).

5.2.2 BlockchainCamera Validator Tool



Picture 5: Activity flow about how Blockchain Camera Validation Tool Works.

Blockchain Camera Validator is a Python3 based Tool that is capable of getting hashes from the user and then check if they exist on a selected BlockchainCamera Smart-Contract. Each Blockchain Camera has its own Validator's tool that is produced from the installation script. Blockchain Camera Validator uses web3.py library to communicate with an Ethereum Network which has been chosen from the installation script. Validator Tool gets a full copy of all hashes from the Blockchain Camera Smart-Contract and then saves it as a list in Python3. Then, it creates an another list to store all the timestamps from the smart-contract too. Keeping that in our minds, it should be mentioned that each index of **hashes** and **time_of_hash** lists are at the same column. That means that the first video's hash is on the **hashes[1]** and **time_of_hash[1]** (as we explained before, the very first column ([0]) stores the word "start" and the time of the smart-contract's deployment). Then, it searches if the hash exists in that list. If the hash exists, it shows the time the block has been mined or validated. The search of the hash is being done externally from the Blockchain due to the fact that sending a hash in Blockchain in order to check if it exists costs in real world Ether. The main point of the Blockchain Camera is to provide a Proof Of Existence in videos and simultaneously cost as less as possible. To validate the hashes easier, there are 2 versions of the BlockchainCamera Validator Tool available:

- BlockchainCamera Validator CLI Tool
- BlockchainCamera Validator GUI Tool

One version is only CLI and has the feature of loading a file with many hashes (one hash per line) and then checking multiple hashes if they are valid at the same time. The second version is a GUI version of BlockchainCamera Validator Tool, and it provides a nice-looking user interface in order help user to interact and validate hashes from the Blockchain directly by copying-pasting the hashes directly in the Tool. GUI is designed by using the PySimpleGUI Python3 library.

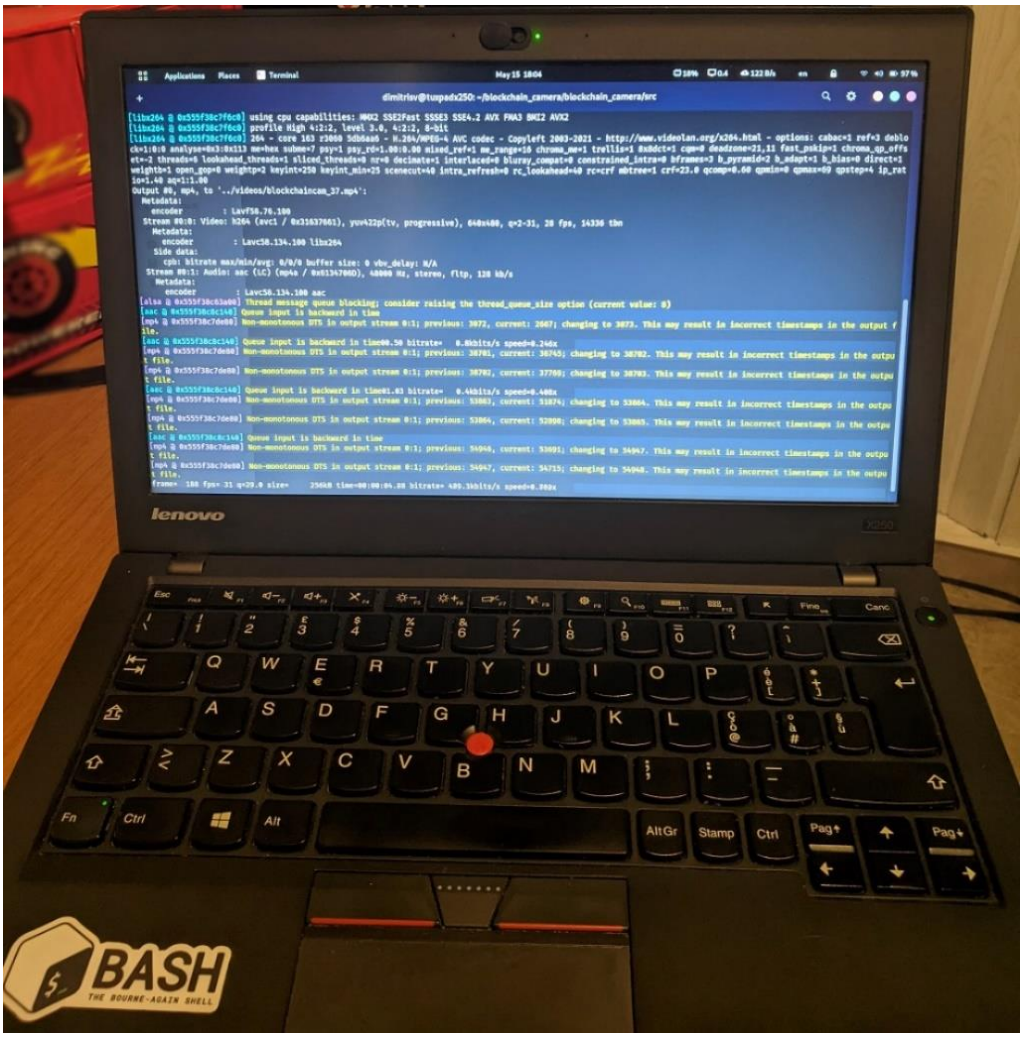
5.2.3 How Blockchain Camera Works

Picture 6: Activity flow about how Blockchain Camera Works.

Blockchain Camera software is written in Bash and Python3. First of all, it uses the ffmpeg tool with parameters of -t 00.10.00 in order to record for 10 minutes. As we can understand, we can modify the time we want to record before sending video's hash to Blockchain. For example, if we want to record videos and send them to Blockchain after every 45 seconds, we can simply modify it to 00.00.45. However, as creating transactions and sending hashes to Blockchain costs Ether, it is recommended to send the hash every 7 to 15 minutes. After the preconfigured time passes, using the sha512sum tool, Blockchain Camera creates a sha512 hash from the video which has recorded before and then it saves the hash to file "hashes". In order to protect the hash, file must be configured to be able to be opened only from the Python3 program that will send the hash to Blockchain. Python3 script uses web3.py to interact with Blockchain's Camera Smart-Contract. The private key of the Wallet that is the owner of the Smart-Contract and it is capable of sending the hash to Blockchain is encrypted by using AES256-cbc. As a key, we use the sha256 hash from the lsusb, the devices that are connected to Blockchain Camera. This encryption can easily be replaced with other methods or even been disabled (not recommended to have a private-key directly saved as plaintext). It should be mentioned here that if we connect or disconnect a device from the camera, the unlocking key from the private key will be lost forever and we will have to re-run the installation script. When the transaction will be sent to Blockchain, the program automatically will create a log file in logs. Logs with a message "Hash <HASH> has been added successfully.". The video files are saved with a sorting form with name: blockchaincam_[NUMBER_OF_COUNT].mp4. If Blockchain Camera's attempt to send the hash to Blockchain fails, an error message will be sent to logs with the message: **"Problem with the connection. Hash : <HASH> hasn't been sent to Blockchain.**". An another python3 script will check periodically if a hash has failed and then it will try to send it again to Blockchain. The time of the check occurs is configured by cronjob. More information about crons will be found to chapter 6.3.

5.3 Hardware

Blockchain Camera software can run in any computer, laptop, netbook or even tablet and smartphone if it runs a version of GNU/Linux which supports ffmpeg, Python3 and web3.py library.

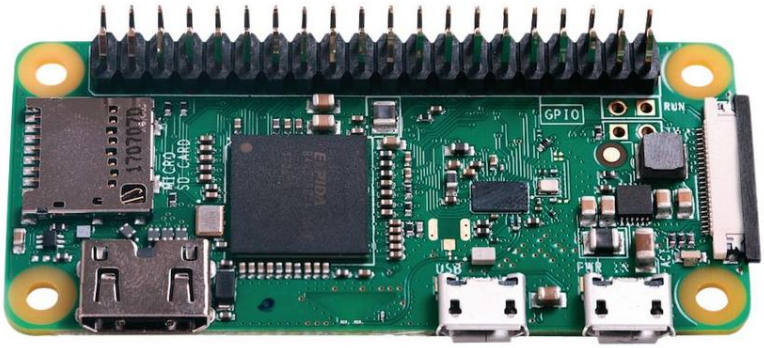


Picture 7: A Lenovo ThinkPad X250 working as a Blockchain Camera.

However, for a better experience, security and for practical use it is recommended to use the software on a stand-alone device only for using the Blockchain Camera software.

5.3.1 Raspberry Pi

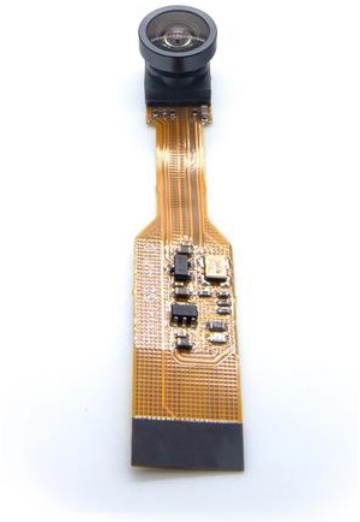
Raspberry Pi is a series of small single-board computers which are extremely inexpensive and powerful for running stand-alone projects, even small home servers. Raspberry Pi Zero series are the smallest and cheapest models from Raspberry Pi Series. The very first model of the Raspberry Pi Zero was launched in 2015 with a price of 7€. Raspberry Pi Zero is the recommended hardware of the Blockchain Camera, running Raspbian, a Debian-based Operating System designed for the Raspberry Pi and it is focused to provide a lightweight OS in order to restrict the usage of the Pi's resources from the OS. As Raspberry Pi runs Linux, we can connect a variety of hardware on it and create stand-alone devices. Blockchain Camera software works perfectly in Raspberry Pi Zero W, however it is recommended to use Raspberry Pi Zero 2 as it has more powerful processor and is capable of computing hashes faster.



Picture 8: A Raspberry Pi Zero WH.

5.3.2 Raspberry Pi Camera

As a camera module, Raspberry Pi Camera features a wide 160° fisheye lens! It's, therefore, able to deliver a clear 5MP resolution image or 1080p HD video recording at 30fps! This camera is recommended for Blockchain Camera as it costs about 20€ and is capable of recording acceptable quality of videos, even choosing s smaller video resolution from ffmpeg in order to save storage capacity.



Picture 9: Raspberry Pi Camera with fisheye Lens.

5.3.3 Microphone Module

As Raspberry Pi Cameras do not provide a built-in microphone, a tiny USB microphone can be used to Raspberry Pi of a cost of 2€, connected with a microUSB to USB adapter. Its audio gain and quality it is acceptable for its price and for the budget of the Project. As Raspberry Pi runs GNU/Linux users can connect whatever kind of microphone they want, even an external audio card and amplifier for connecting professional microphone devices.

Picture 10: A tiny USB microphone with a total cost of 2€.

5.3.4 Blockchain Camera Case

There are many implementations of Raspberry Pi Camera cases for both Raspberry Pi and Raspberry Pi Zero models in platforms such as Thingverse that anyone can print and use to his projects. There are two recommended Raspberry Zero Cases of the Blockchain Camera. These cases can be printed with a 3D Printer. Both cases are uploaded to Thingverse under MIT license. On this paper, Case 1 has been used. Click each button to check and download the .stl files in order to print the case you like the most:

Blockchain Camera Case 1 ← Click here
Blockchain Camera Case 2 ← Click here

Picture 11: Protype of Blockchain Camera stand-alone device on a 3D Printed Case.

6.Installation Process

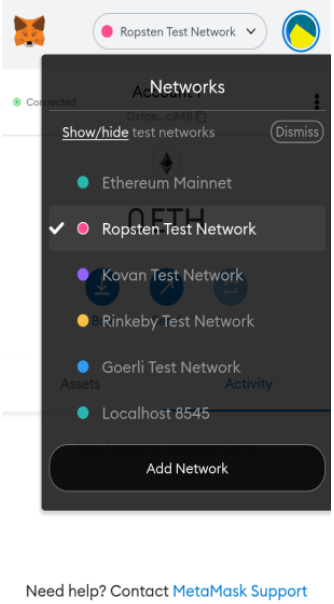
Blockchain Camera software can be installed in any GNU/Linux Device very easily. In the next subchapters, we will discuss about how we can setup a full functional Blockchain Camera to our Raspbery Pis or even to our old netbooks and computers.

6.1 Deploy BlockchainCameraSmart-Contact.sol

First of all, BlockchainCamera.sol Smart-Contract must be deployed to a Blockchain Network manually by the device administrator. The administrator with his wallet who will deploy the BlockchainCamera.sol to Blockchain will be the only one who will be able to create transactions and send data to this smart-contract from the Blockchain Camera. Note: We should consider that we need some Ether to be able to deploy the smart-contract to each Ethereum Based Blockchain Network. Before deploying Blockchain Camera to the Mainnet, it is highly recommended from the device administrator to deploy it at first to a Testnet for testing purposes and basic understanding about what is going on and when the administrator will conclude that is working full functional as It should in testnet, only then it is recommended to be deployed in the main Ethereum Network. As we have already mentioned, deploying a smart-contract to Blockchain makes it undestructive. That means if we want to make any change to the code we have to compile and deploy it again as a different smart-contract. In testnets it does not cost us real world ether to do research but in the main chain it costs real world money, that's why we must do our research at first at Testnets. On this example, we will demonstrate how a device administrator can compile and deploy BlockchainCamera.sol to Ropsten Ethereum Testnet Network.

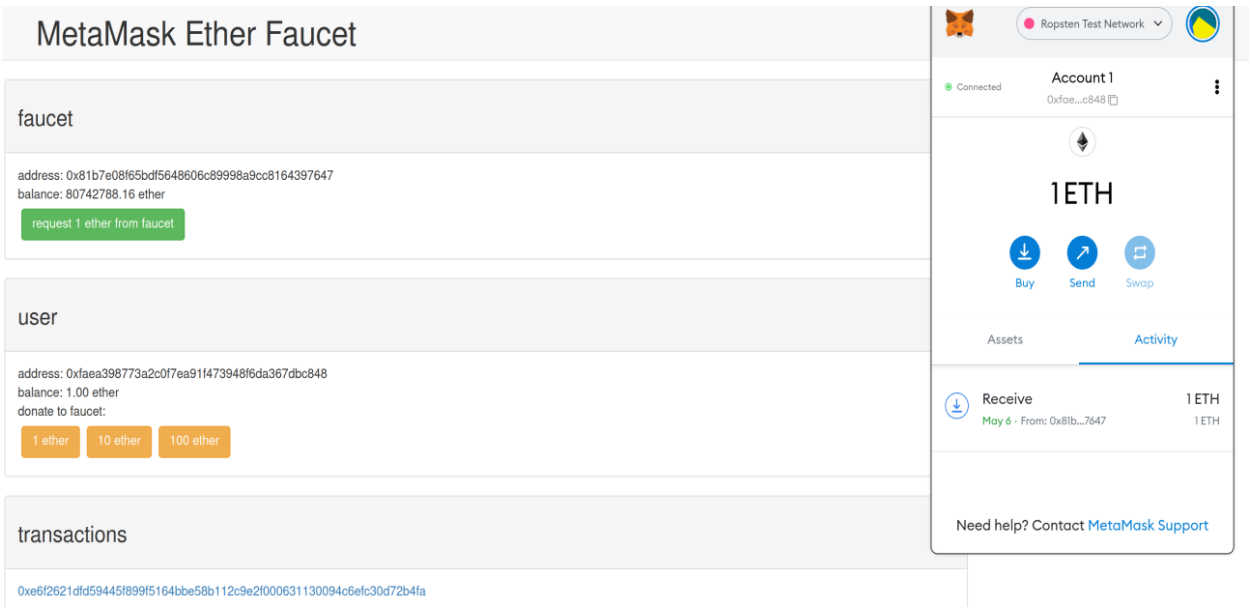
A Metamask wallet connected with Ropsten Test Network and with some ether on the wallet is required in order to be able to deploy it to Ropsten. More information about how we can create an Ethereum wallet with Metamask, can be found there: https://metamask.io/faqs/

Now, it's time to connect Metamask to Ropsten Network. In the top center of the Metamask window, we can see a button with the text of "Ethereum Mainnet". We click there and then Metamask will show us a variety of Ethereum testnets, it even offers us a connection with a potential local testing network. We choose "Ropsten Test Network". If changing Ethereum Network is locked, we should go to Settings/Advanced_Settings and then we have to enable "Show test networks".



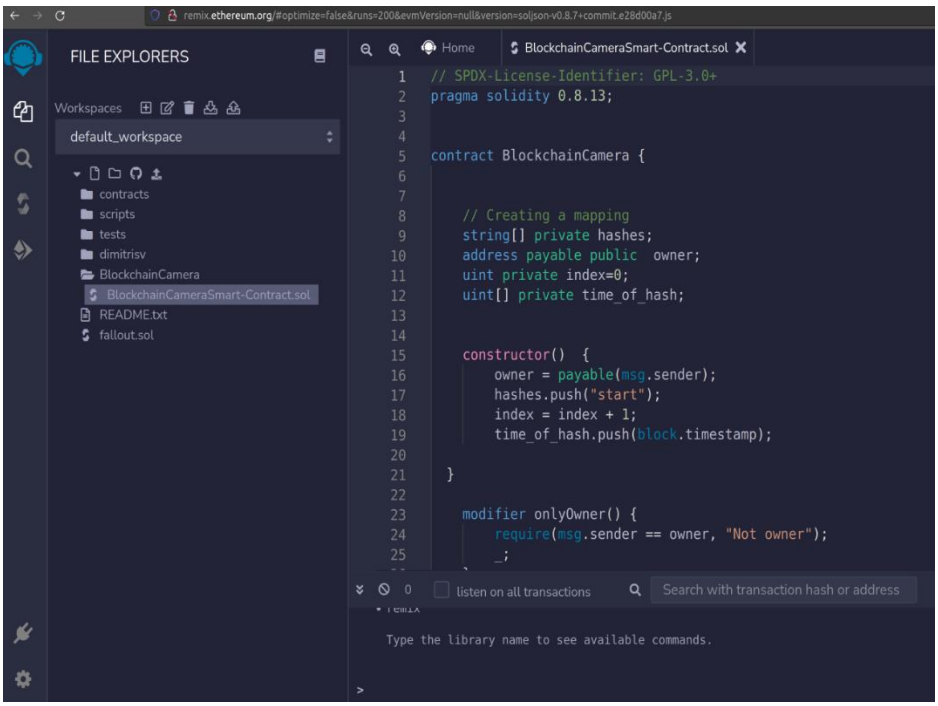
Picture 12: Choosing to connect to Ropsten Test Network.

Now, we have to request test ether for Ropsten Network. Metamask Ether Faucet can be used to achieve this very easily. We can request 1 ether from faucet by pressing the green button "request 1 ether from faucet".

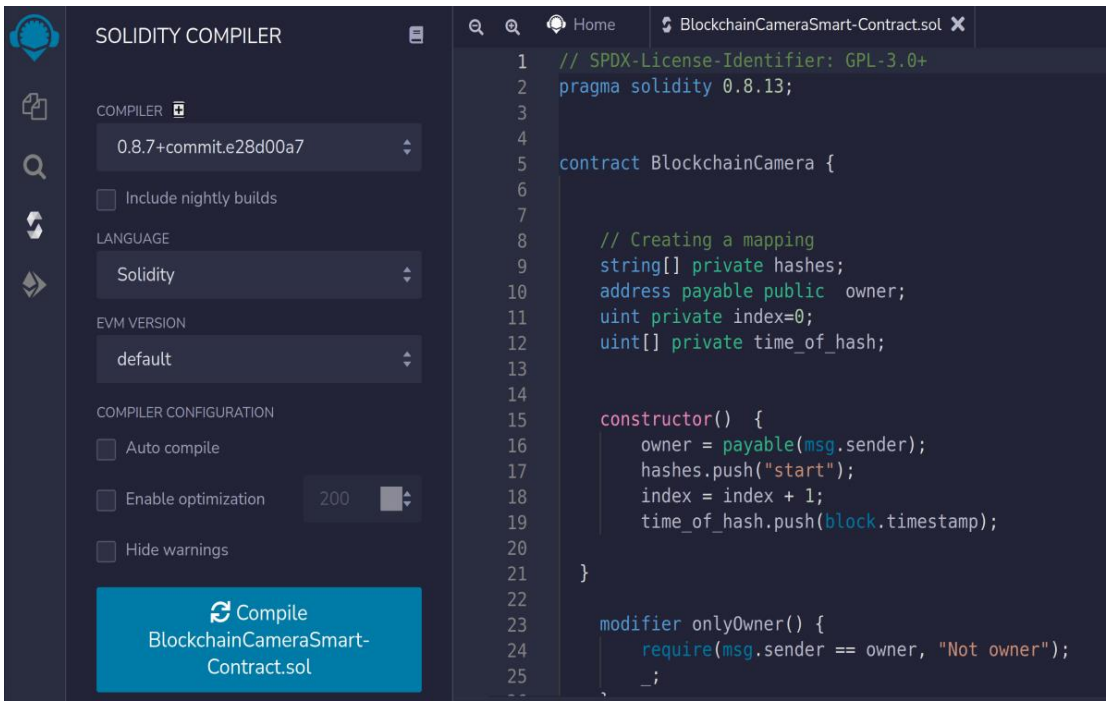


Picture 13: Metamask Ether Faucet. There we can request for test ether for free.

As we have some ether in our wallet, we are ready to compile and deploy BlockchainCameraSmart-Contract.sol to Ropsten Network. We move to https://remix.ethereum.org and then, we create a folder named “BlockchainCamera”. Inside on this folder, we create a file “BlockchainCameraSmart-Contract.sol”. We open the file and we paste there the code we can found in Blockchain’s Camera Github Repository. Link can be found in chapter 11. Find Blockchain Camera’s Code.

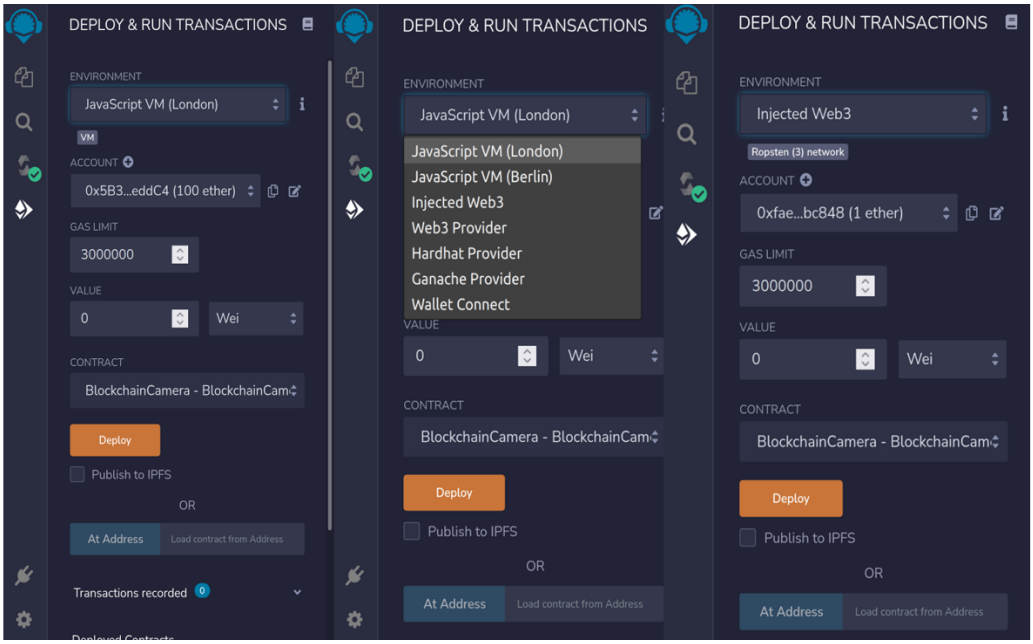


Picture 14: Remix IDE with BlockchainCamera-Contract.sol ready to be compiled.



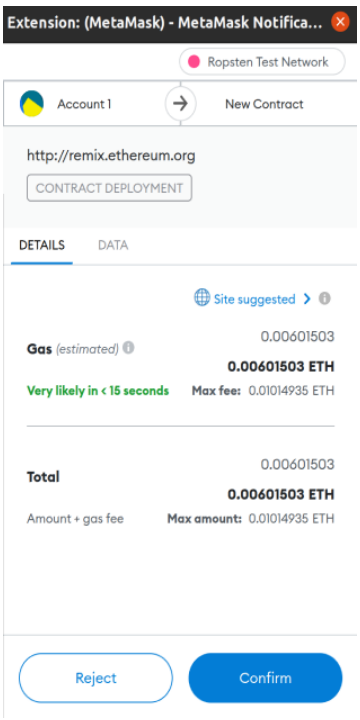
Picture 15: Compilation of BlockchainCameraSmart-Contract.

Now we move to the third process menu where we can compile the smart-contract. As we can see on the photo above, we choose the same compiler version as the version we can find it in smart-contract's pragma. Now we are ready to press Compile blue button. Let's move to the next step:



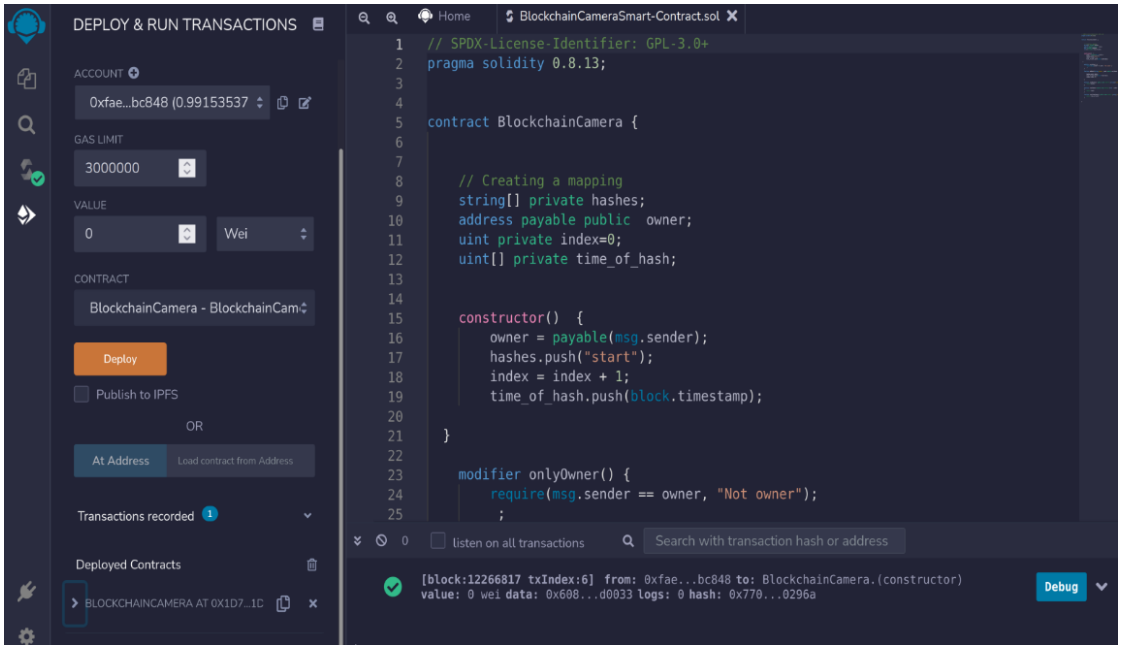
Picture 16: Connecting Remix IDE with out Metamask wallet.

As we compiled the smart-contact, now we can move to the next tab and deploy it to Ropsten Network. On deploy & run transactions tab, we have to change the environment from JavaScript VM to Injected Web3. Injected Web3 can interact with Ropsten Network through Metamask. If we haven't connected Metamask with Remix IDE before, it will request us to connect right now. After connecting it to Metamask we can see our public address in "account" tab with the amount of ether this account owns. Now we click **Deploy**. A Metamask tab will open in order to pay the transaction fees and deploy the smart-contract to Blockchain.

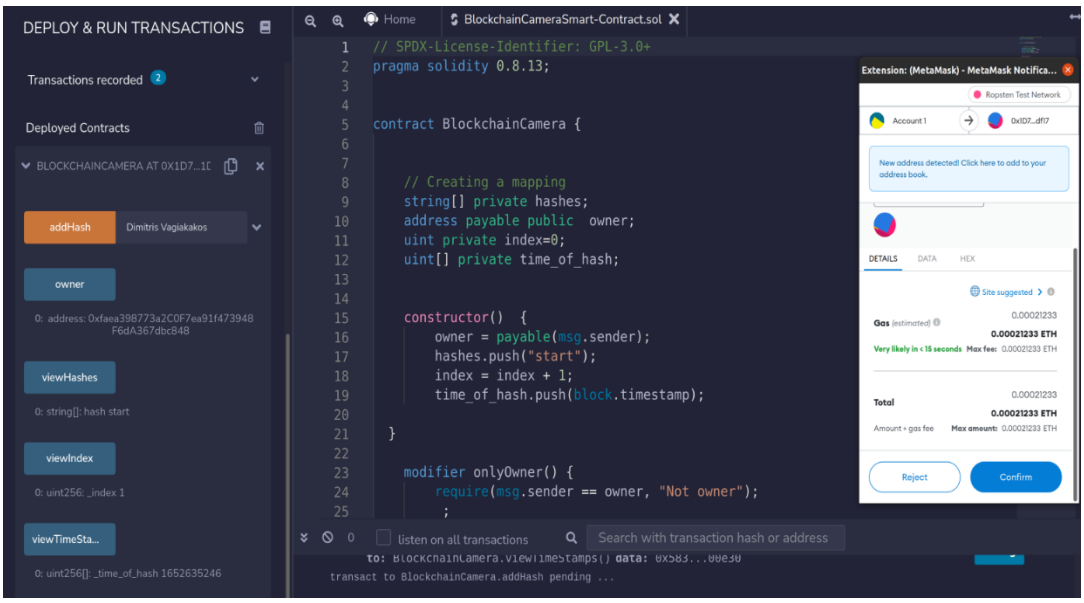


Picture 17: Time to pay the transactions fees to deploy the smart-contract to Blockchain.

We wait until the block will be validated and Blockchain Camera Smart-Contract is ready for use. Remix IDE provides us tools for easier debugging and checking. We notice that in “Deployed Contracts”, the address of the BlockchainCamera. We press the small arrow:

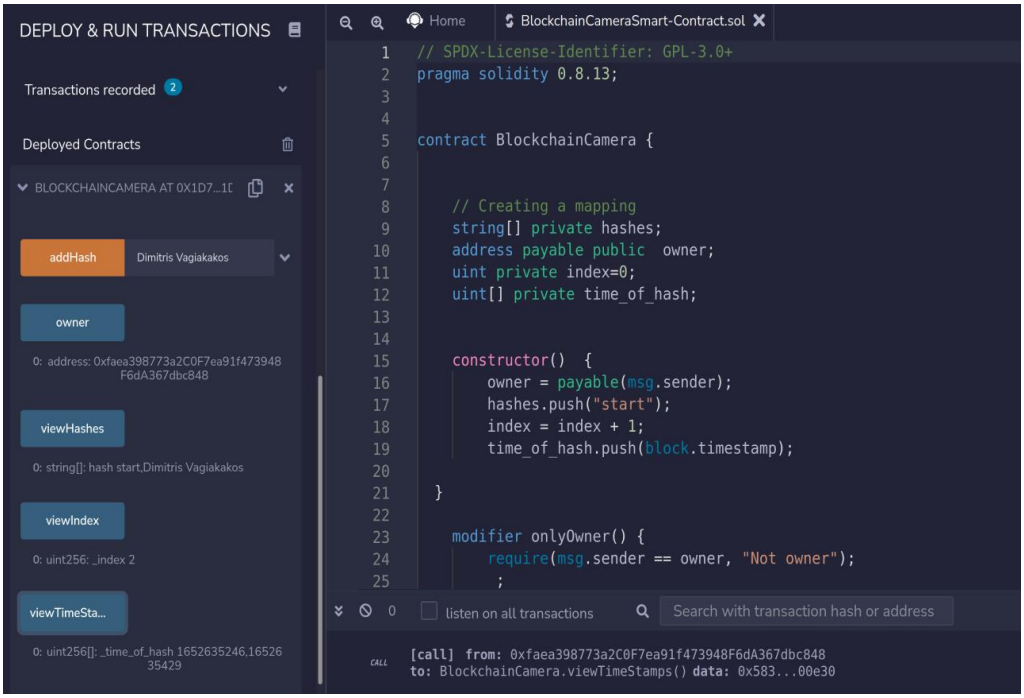


Picture 18: Blockchain Camera Smart-Contract ready for use.



Picture 19: Calling all Blockchain Camera Smart-Contract's Functions and then sending my name to Blockchain.

All smart-contract functions we can call are shown. Instead of using CLI to interact with BlockchainCameraSmart-Contract, we can directly press the buttons and send each transaction we want. On this time, we should remind that reading from Blockchain is free of cost. As a result, we can press the blue buttons and read the data of each function very easily for free. A successful deployment of BlockchainCameraSmart-Contarct must include as the **owner**, the administrator's wallet address. Moreover, **viewHashes** must return a **"start**" message as the very first string in the array hashes from the deployment and the **viewTimeStamp** returns the time the smart-contract deployed to Blockchain in UNIX time. Now we are ready to try and send data to Blockchain for the very first time. We can type a random text to **addHash** orange button, and then press it. **addHash** button will create and send a transaction with the string/hash to Blockchain. We wait until the block will be validated and then we click in all blue buttons again to check if the new hash/string we added has successfully added to BlockchainCameraSmart-Contract.



Picture 20: My name and the time of the transaction has been successfully added to Blockchain.

It should be mentioned that when the transaction of the deployment will be successful, we have to note the address of the Blockchain Camera Smart-Contract, we will need it later. Now it's time to setup the BlockchainCamera to our device.

6.2 Install Blockchain Camera to your device

6.2.1 Dependencies

Blockchain Camera code runs in any GNU/Linux distribution and needs ffmpeg and Python3 to be functional. Python3 uses the web3 library. Web3 can be installed very easily with the command:

```
pip install web3
```

Furthermore, ffmpeg installation varies as each GNU/Linux distribution use a different package manager. For this reason it is recommended to search to your distribution's forum about how it can be installed to your system. For example for Debian/Ubuntu based distributions, ffmpeg can be installed with the command:

```
sudo apt install ffmpeg
```

In Fedora, CentOS and Red Hat Linux, ffmpeg can be installed with the command:

```
sudo dnf install ffmpeg
```

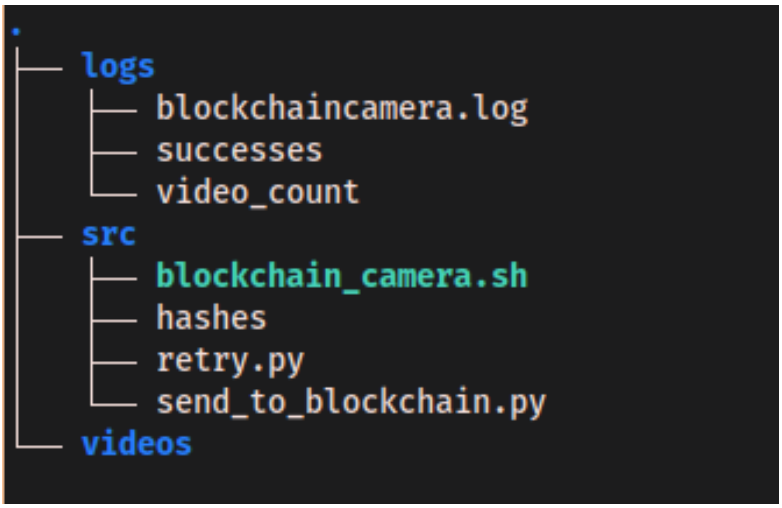
6.2.2 Install Blockchain Camera software

Blockchain Camera software provides an automated installation script which will create and configure all the files are necessary in order to have a full functional Blockchain Camera. At first, we have to download the entire installer folder from Github and then run the setup.py. Then, the installer will ask about the BlockchainCamera's Smart-Contract address, the private key of the wallet, and the http link from the node that it will be used to interact with Blockchain.

On my own example, I used the infrastructure from Infura. Infura provides a Blockchain node infrastructure service that allows apps and developers to get data from, and broadcast transactions to the Ethereum Blockchain. Infura provides access to the Mainnet as well as to the test Networks. For example, I connected my own Blockchain Camera to Ropsten Testnet Ethereum Network, in which I deployed the Blockchain Camera Smart-Contract too.

When we add the private key from the wallet we deployed the Blockchain Camera Smart-Contract, system will automatically hash the lsusb and then encrypt the data key with aes256-cbc encryption cipher of the lsusb the devices that are connected to Blockchain Camera. It is mandatory to remove all the devices that they will not be connected on the camera when the installer will ask us to enter the private key.

The installer will make our system ready to record videos and then send the hashes to Blockchain.

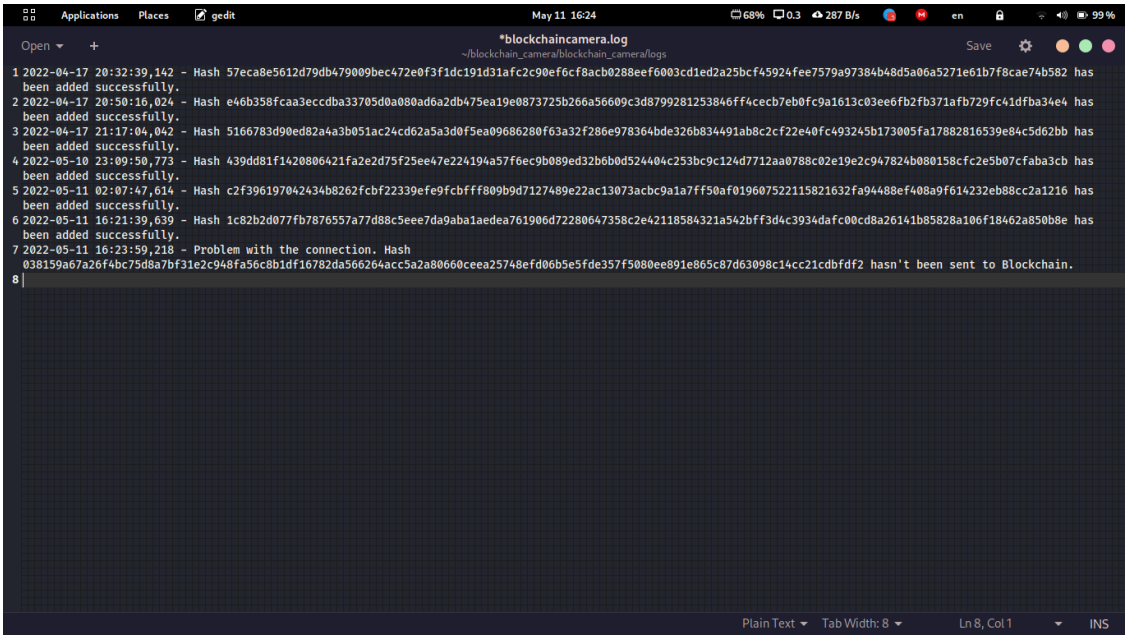


Picture 21: The contents of the Blockchain Camera installation *folder after running the installation script.*

Blockchain Camera files are divided into 3 subfolders:

- logs: where the log files are stored.
- src: where the Blockchain Camera applications are stored.
- videos: where the videos are stored.

Moreover, installer will produce 2 Validation Tools, one CLI version and one GUI version, which can connect and read data only from the Smart-Contract address the administrator has entered before, due to installation process. If user wants to connect the Validator Tools to a different Blockchain Camera, it is recommended to run the installer again and then enter as an input only the smart-contract address he wants to be able to connect.



Picture 22: Logs from the Blockchain Camera. When a hash failed to be sent to Blockchain, the camera can be configured to try to resend it again after a couple of minutes using the retry.py program.

In blockcain_camera.sh, device administrator must configure ffmpeg to record from its own camera or/and microphone and choose the minutes he wants to record before

hashing the video. The Python software send_to_blockchain.py is called from blockcain_camera.sh or retry.py in order to send a hash to Blockchain.

For example, Raspberry Pi ffmpeg configuration for only video recording with audio for 10 minutes is:

```
ffmpeg  -f video4linux2  -r 28 -i /dev/video0 -f alsa -ac 2 -i hw:1,0  -t 00:10:00 ../videos/blockchaincam_$name.mp4
```

Furthermore, to activate the Blockchain Camera to run every 10minutes we have to create a cronjob to force it re-run. On the next subchapter we will discuss about how cron works and how to use it.

Retry.py is an optional script that can be used in order to check from the log files if a hash failed to be sent to Blockchain and if not, it will call send_to_blockchain.py to send it again. To activate it, it is recommended to create a cronjob to check for failures every 7-15 minutes.

6.3 Job Scheduling Blockchain Camera to run automatically

The cron command-line utility, also known as cron job, is a job scheduler on Unix-like operating systems. Users who set up and maintain software environments use cron to schedule jobs, commands or scripts to run periodically at fixed times, dates, or intervals. As we want Blockchain Camera to work periodically at fixed times (for example capturing 10minute videos every 10 minutes or retrying to submit hashes that failed to be submited every 15minutes). The actions of cron are driven by a crontab (cron table) file, a configuration file that specifies shell commands to run periodically on a given schedule.
Each line of a crontab file represents a job, and looks like this:

```
# ┌──────────────────── minute (0 - 59)
# │ ┌──────────────────── hour (0 - 23)
# │ │ ┌──────────────────── day of the month (1 - 31)
# │ │ │ ┌──────────────────── month (1 - 12)
# │ │ │ │ ┌──────────────────── day of the week (0 - 6) (Sunday to Saturday)
# │ │ │ │ │
# │ │ │ │ │
# │ │ │ │ │
# * * * * * <command to execute>
```

To create a cronjob we open a terminal and we type:

```
crontab -e
```

And then in order to make Blockchain Camera to start capturing videos every 10 minutes, we add a new line on this file as:

```
*/10 * * * * su username -c “<LOCATION>/src/blockchain_camera.sh”
```

If we want to capture videos for a different timestamp, we can just modify the */10.

For example if we want to capture videos every 5 minutes, we can configure the cron to:

```
*/5 * * * * su username -c “<LOCATION>/src /blockchain_camera.sh”
```

Except blockchain_camera.sh, if we want Blockchain Camera to check if all hashes have successfully sent to Blockchain periodically and if not to submit them again, we should also add retry.py as a cronjob. The recommended time is after 13 minutes. We add retry.py to a new line:

```
*/13 * * * * su username -c “<LOCATION>/src /retry.py”
```

7. User's Manual

7.1 Depedencies

Blockchain Camera Validator Tools in order to be functional need Python3 to be installed with the library of web3. Web3 can be installed very easily with the command:

```
pip install web3
```

Moreover, Blockchain Camera GUI needs PySimpleGUI library too. PySimpleGUI can be installed very easily with the command:

```
pip install PySimpleGUI
```

7.2 Hashing a Video

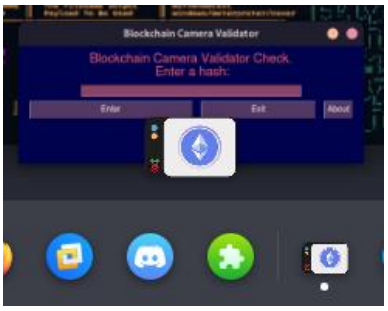
As has already been mentioned before, Blockchain Camera hashes are created by using a SHA-512 hash function and then stored to Blockchain. As users, when we want to validate that a video is genuine, at first we have to download the video from a centralized center with the events that we want to validate if they really happened.

We can hash a video by using sha512sum tool from GNU/Linux or other Unix like Operating Systems by typing:

```
sha512sum $NAME_OF_FILE.mp4
```

sha512sum tool will return a hash we can copy and then paste to a Blockchain Camera Validator Tool (GUI or CLI) in order to check if that hash really exists in Blockchain.

7.3 Blockchain Camera Validator Tool GUI



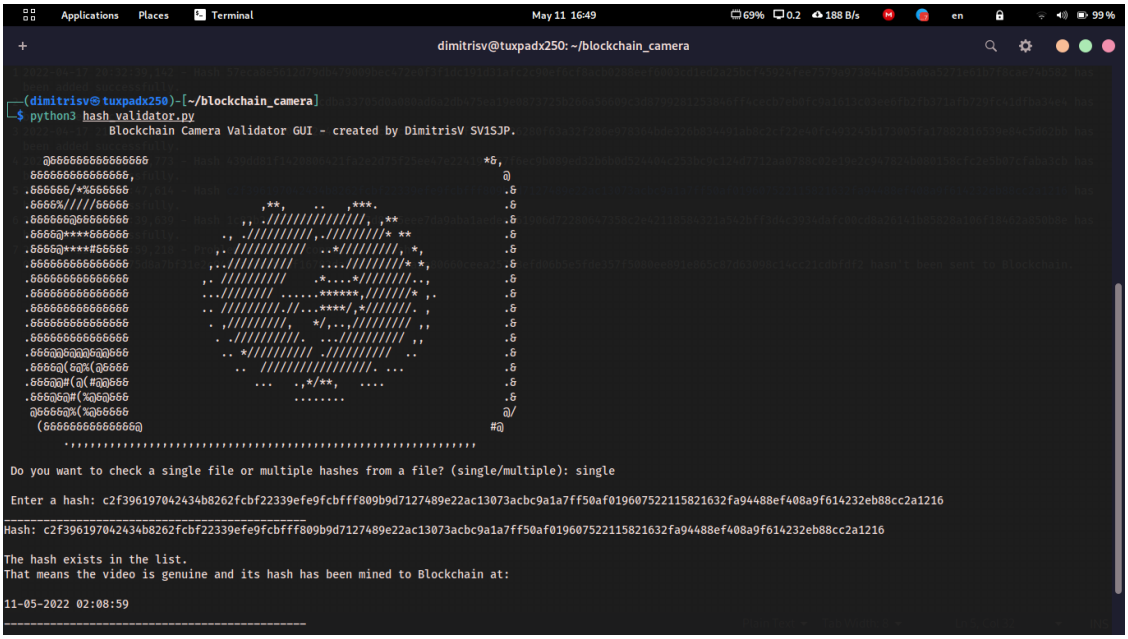
Blockchain Camera Validator Tool GUI is recommended to be used from normal users as it offers a nice-looking interface where users can interact with Blockchain Camera Smart-Contract very easily. However, in order to be functional, keep in mind that the script must be edited and a node must be added by user. Without a node, Blockchain Camera Validator Tool cannot interact with Blockchain. If you do not want to create your own Ethereum Node you can use a company's node infrastructure, for example from Infura, and then paste the appropriate HTTPProvider link to **w3** variable.

Furthermore, user can very easily copy the video's hash from the sha512sum command and paste it to Blockchain Camera Validator GUI and then press Enter. Blockchain Camera will download a full copy of the Blockchain Camera Smart-Contract's and using basic Python function it will find out if the hash exists in the list. If the hash exists, the tool will return the message:
"The hash exists in the list. That means the video is genuine and its hash has been mined to Blockchain at: <UNIX_TIME_CONVERTED_TO_LOCAL>" and then it will show the time the hash has been sent to Blockchain.

Picture 23: Hash exists in Blockchain. Blockchain Camera Validator Tool GUI shows the time the hash has been sent to Blockchain.

If the video's hash has not been sent to Blockchain, The Blockchain Validator Tool returns a message of "This hash is not genuine.".

7.4 Blockchain Camera Validator Tool CLI



Picture 24: Hash exists in Blockchain. Blockchain Camera Validator Tool CLI shows the time the hash has been sent to Blockchain.

Blockchain Camera Validator CLI is the backend version of the GUI version of the app. Blockchain Camera Validator Tool CLI comes as an easier command line-based tool in order to be implemented to other projects and extend its futures much easier. As the GUI version for being functional, user must add a node to the program. Blockchain Camera Validator Tool CLI works exact the same as the GUI. All the user has to do is to paste a hash as an input and the application will return if the hash exists to Blockchain Camera Smart-Contract. However, CLI mode has some special futures which are missing from GUI version.

7.5 Check Multiple Hashes at the same time with Blockchain Camera Validator Tool CLI

Blockchain Camera Validator Tool CLI supports multi hash validation. All we have to do is to create a new file at the same file with the Blockchain Camera Validator Tool CLI and paste one hash on each line. Then, we run the tool and we choose multiple and we enter as an input the name of the file. File's location must be on the same folder with the Blockchain Camera Validator CLI Application. Blockchain Camera Validator tool will check each hash if exists in the Blockchain or not and it will return the result.

Picture 25: Checking if multiple hashes are valid using the Blockchain Camera Validator CLI Tool.

8. The Cost Of Blockchain Camera

One of the main goals of Blockchain Camera is to be built at low cost in order to be more accessible from anyone without a huge budget. Using an old netbook, laptop or computer, Blockchain Camera can be built at extremely low cost as we will use older computer components which are not capable of doing modern day usage like office, work, browsing etc. and instead of recycling it, Blockchain Camera will extend its lifespan for some years. However, older generation of multiprocessors consume a large amount of energy than newer hardware. To be more specific, older processors, for example Intel Pentium 4 or the very first models of Intel Core 2 Duo series or even AMD FX series may be spending far more electricity that the hardware is worth. Of course electricity cost changes from country to country due to different electricity cost per country, so its on your hands to do your research if it is actually worth using old hardware in your country. Keeping this in our minds, creating a stand-alone Blockchain Camera device is recommended because:

- It's more secure to use hardware specific for one only usage.
- It's more convenient to use a small stand-alone device.
- It consumes less than 5W per hour electricity.
- Blockchain Camera recommended hardware and 3D case can be wearable.

The basic implementation of Blockchain Camera hardware costs as we can see on the board (in average cost of EU based markets):

Component	Cost
Raspberry Pi Zero W WIFI	12€
Raspberry Pi Camera Fisheye	20€
64GB MicroSD Card	6€
3D Printed Case	25€ (printed by a third party company)
Overall Cost	63€

Depending on the setup or the needs to each situation, these extra components can be beneficial to extend the features of Blockchain Camera:

Component	Cost
PowerBank 20.000mAh	18€
Tiny USB Microphone	2€
Power On/Off Button	5€
Waveshare SIM7600E-H 4G HAT communication module	80€

9. Extra implementation ideas

9.1 Save the videos externally from the BlockchainCamera

As RaspberryPi has a limited storage capacity, if we want to use the BlockchainCamera on a daily basis, it is recommended to use an external server on the network in order to save the videos externally there. To achieve this, we can use the rsync tool. Rsync is a tool that is capable of transferring and synchronizing the videos between the Blockchain Camera and a storage drive and across networked computers by comparing the modification times and sizes of files. Especially, if we want to use more than 1 Blockchain Camera to our setup, rsync is extremely beneficial. By configuring an ssh connection with the server and creating a cronjob for a backup, for example, every 6 hours and then, share the videos to an https server or sftp in order to be available for everyone to download, watch and then validate the genuineness of these videos. Moreover, we can use the BitTorrent protocol in order to peer-to-peer distribute these videos. This method is better in terms of availability because less hardware is needed to protect and store the recorded videos. However due to law limitations about data and privacy protection for example in GDPR is a very controversial solution as videos cannot be removed from peer-to-peer networks.

To create backups to an ssh server using rsync every 20 minutes, we can create a new cronjob:

```
*/20 * * * * su username -c “rsync –r <LOCATION>/videos username@IP://<BACKUP_FOLDER_NAME
```

9.2 Permissions for standalone devices

For standalone devices, it is recommended to create a new limited user account, which will have only permissions to execute the Python3 and Bash code and not be able to modify or delete anything else except the logs. This will help Blockchain Camera to stay secured. Chmod is a command and system call used to change the access permissions of file system objects (files and directories) sometimes known as modes.

Python and Bash scripts permissions should be configured to be only executable by the owner. From the root user, after running the installation script, we move the **blockchain_camera** folder to the new user’s home directory. Then, we create a new group named blockchaincamera:

```
groupadd -g group-ID blockchaincamera,root
```

To add the new user to the blockchain camera group we enter the command:

```
usermod -G blockchaincamera user-name
```

Then, we change the permissions of the Blockchain Camera scripts to be only executable from the same group by using the chmod command:

```
chmod -R 710 blockchain_camera/src
```

Now we set the permissions of the videos to make blockchain camera only capable of capturing videos and not able to read or execute them by using chmod command:

```
chmod -R 760 blockchain_camera/videos
```

Finally, we configure the logs to have read and write access but not being executable:

```
chmod -R 720 blockchain_camera/logs
```

10. Conclusion

Blockchain Camera has a great potential to upgrade the way trust works in our daily lives where we use people as witnesses to validate and figure out what really happened in accidents or other events or even having to trust third party issuers. Creating a secure infrastructure to check the validity of videos or photos is of highly importance on a modern digital world and we need new ways to secure and validate information. Blockchain Technology as an upgraded version of the Web and can improve the way trust works. As an aftermath, corruption can be reduced to many sectors because of a widespread use of Blockchain Cameras as we will know the truth in sensitive issues. Blockchain Camera is full functional for daily use to provide a way to trust information. Blockchain Camera is a new Blockchain Project and its usage will make it more applicant in real world. Changes and upgrades to the main smart-contract code as well as to backend scripts to extend their functionality will be released the next months and the main point is to extend the Blockchain Camera with more functionalities and more security. Blockchain Camera released under GPL v3.0+ in order to be accessible by everyone and to ensure that the future updates, patches and code will remain open-source forever. There are many ideas to be implemented in the future to extend its functionality and security. For example, full device encryption can be used in Blockchain Camera in order to protect the sensitive information from malicious third parties or because Blockchain Camera has been stolen. Furthermore, Blockchain Camera Validator Tool can be more extended and integrate a sha512sum tool in order to be easier for the end user to validate videos or even creating Docker images for easier deployments. Blockchain Technology is the future of Internet and investing in the feature technologies is extremely helpful not only for new the ideas to be born but also to extend the already existing ideas and creating a better future for the next generations with less violence and corruption.

11. Find Blockchain Camera's Code

Blockchain Camera code can be found on Github's Blockchain Camera Repository:
https://github.com/sv1sjp/Blockchain_Camera

You can try to run it in your computer. Read the instructions above. Moreover, read the README.md for some extra instructions. If you find any bug or you have any idea for the Blockchain Camera don't hesitate to contact me on my email or any social media.

12. References

[1] Developing a Blockchain eVoting Application using Ethereum, Dimitris Vagiakakos, Konstantinos Karahalis, Stavros Gkinos (2021) : https://github.com/sv1sjp/eVoting_Elections_Decentralized_App/blob/main/eVoting_Smart_Contract_paper.pdf

[2] Smart Contract Security, Dimitris Vagiakakos, Stavros Gkinos, Ioannis Karvelas (2022) :https://github.com/sv1sjp/eVoting_Elections_Decentralized_App/blob/main/smartcontract_security_paper.pdf

[3] Privacy in Blockchain and Web3.0 Second Edition, Dimitris Vagiakakos, Stavros Gkinos, Konstantinos Karahalis (2022): https://github.com/sv1sjp/eVoting_Elections_Decentralized_App/blob/main/privacy_in_blockchain_and_web3.0_second_edition.pdf

[4] Antonopoulos, Andreas M.; Wood, Gavin (2018). Mastering Ethereum: building smart contracts and DApps (First ed.). Sebastopol, CA: O'Reilly Media, Inc.

[5] Antonopoulos, Andreas M,; (2017). Mastering Bitcoin: programming the open blockchain (Second ed.). CA: O'Reilly Media, Inc.

[6] Token Economy: How the Web3 Reinvents the Internet (2020) Shermin Voshmgir

[7] Vitalik Buterin. "Merkling in Ethereum". Ethereum.org.

[8] Eth.Research: https://ethresear.ch/latest

[9] The Etheraut - A Web3 wargame played in Ethereum Virtual Machine - https://ethernaut.openzeppelin.com/

[10] Smart Contract Security - https://ethereum.org/en/developers/docs/smartcontracts/security/

[11] Dapp University - https://www.dappuniversity.com/

[12] ETHEREUM: A SECURE DECENTRALISED GENERALISED TRANSACTION LEDGER - DR. GAVIN WOOD : http://gavwood.com/paper.pdf

[13] Types of wallets in blockchain: https://academy.binance.com/en/articles/crypto-wallet-types-explained

[14] Importance of nodes in blockchain and what are they: https://www.seba.swiss/research/Classification-and-importance-of-nodes-in-a-blockchain-network

[15] Infura Ethereum Nodes: https://infura.io/

[16] Metamask Guide: https://docs.metamask.io/guide/

[17] Solidity 0.8.0 Documentation : https://docs.soliditylang.org/en/v0.8.0/

[18] Solidity Github Documentation and examples: https://github.com/ethereum

[19] web3.js Documentation: https://web3js.readthedocs.io/en/v1.3.0/

[20] Metamask Guide: https://docs.metamask.io/guide/

[21] Ropsten Ethereum Testnet: https://ropsten.etherscan.io/

[22] PySimpleGUI Cookbook: https://pysimplegui.readthedocs.io/en/latest/cookbook/

[23] Web3.py - Read The Docs: https://web3py.readthedocs.io/en/stable/

[24] Raspberry Pi OS: https://www.raspberrypi.com/software/

[25] ffmpeg documentation: https://ffmpeg.org/ffmpeg.html

[26] cron – Linux-User.gr: https://linux-user.gr/t/cron-trechoyme-aytomata-entoles-kai-efarmoges-me-th-vohtheia-toy/1908

[27] rsync – Linux-User.gr: https://linux-user.gr/t/sygchronismos-kai-backup-se-fakeloys-me-rsync/3451

[28] Remix IDE: https://remix.ethereum.org/

[29] The police department in Nea Smyrni under siege – Greek City Times: https://greekcitytimes.com/2021/03/10/police-nea-smyrni-under-siege/

[30] GNU GPL v3.0: https://www.gnu.org/licenses/gpl-3.0.en.html