Uncuffed: A Blockchain-based Secure Messaging System

VASILEIOS DIMITRIADIS, University of Thessaly, Greece
LEANDROS MAGLARAS, De Montfort University, UK
NINETA POLEMI, University of Piraeus, Greece
IOANNA KANTZAVELOU, University of West Attica, Greece
NICK AYRES, De Montfort University, UK

Secure messaging is very important especially in critical applications like communication of automated cars. In this article we present a simulation of a blockchain messaging platform based on the Bitcoin protocol. The mechanism can be used for exchanging privately and securely messages and images. Each member of the blockchain can become a miner and collect rewards or just a plebeian client and send messages without actively using the platform.

Additional Key Words and Phrases: Cybersecurity, Secure Messaging, Blockchain

ACM Reference Format:
Vasileios Dimitriadis, Leandros Maglaras, Nineta Polemi, Ioanna Kantzavelou, and Nick Ayres. 2020. Uncuffed: A Blockchain-based Secure Messaging System. In *25th Pan-Hellenic Conference on Informatics (PCI 2021), November 20–22, 2020, Athens, Greece.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3437120.3437336

1 INTRODUCTION

A blockchain is a list of a continuously ever-growing list of blocks, which are linked together using a cryptographic hash. Each block contains a set of publicly viewable transactions which are stored permanently on the blockchain. Additionally, each block may also include some meta-data required for the calculations such as previous hashes, timestamps, difficulty, height etc.

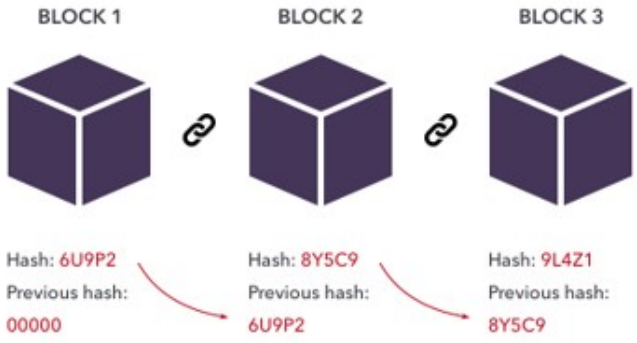

Fig. 1. How Blockchain Works

All blockchain systems implement in one way or another the following characteristics:

- Secure by Design: They are resistant to the modification of data, and become immutable when sufficient time passes as the blockchain stabilizes
- Decentralized: Designed to work in Peer-to-Peer models, with no central point of failure Permissionless: Originally, they are designed open for the world.
- No secrets. No access-control needed.

1.1 Motivation

Blockchain, a distributed append-only public ledger technology, was initially intended for the cryptocurrencies, e.g., Bitcoin. In 2008, Nakamato [Nakamoto 2008] introduced the concept of blockchain that has attracted much attention over the past years as an emerging peer-to-peer (P2P) technology for distributed computing and decentralized data sharing. Due to the adoption of cryptography technology and without a centralized control actor or a centralized data storage, the blockchain can avoid the attacks that want to take control of the system.

In the IoT devices, attackers seek to exfiltrate the data of IoT devices by using the malicious codes in malware, especially on the open source Android platform [Ferrag et al. 2018, 2020]. Moreover, n distributed P2P applications for the IoT, the IoT devices self-organize and cooperate for a new breed of applications such as collaborative movies, forwarding files, delivering messages, electronic commerce, and uploading data using sensor networks. Secure messaging is very important especially in critical applications like communication of automated cars. The intense pressure in the past years to deliver solutions quickly has resulted in varying threat models, incomplete objectives, dubious security claims, and a lack of broad perspective on the existing cryptographic literature on secure communication. Nevertheless, some of the proposed solutions look promising and can be broadly adopted in several fields.

The authors in [Shrestha et al. 2020] propose a scheme to determine the node trustworthiness and message trustworthiness in the VANET and then store them in a public blockchain that acts as the ground truth for other vehicles. As identified by the authors in [Unger et al. 2015] secure messaging involves three key challenges: trust establishment, conversation security, and transport privacy. Some of these can be covered by novel technologies like blockchain and can be supported in terms of security by well known concepts like Intrusion Detection Systems [Ferrag and Maglaras 2019] or biofeatures [Ferrag et al. 2019].

Recently several works have introduced the use of blockchain for secure messaging. In [Khacef and Pujolle 2019] the authors explain why blockchain would make communications more secure, and propose a model design for blockchain-based messaging maintaining. In [Ellewala et al. 2020] the authors developed a chat application with more secure channels of enterprise level communication. Finally, in [Singh et al. 2021] a blockchain-based E2EE framework is presented. The authors claim that their model can mitigate the contemporary vulnerabilities in messaging applications. Similar to those works, our proposed model is based on blockchain to provide secure messaging by supporting two groups of users, miners and plebeians.

2 UNCUFFED SYSTEM

In this section we present the system by explaining thoroughly its different components like the group of users, transactions, proof of work and validation of them. The project can be found on Github in: Uncuffed Code on Github

2.1 Uncuffed users

While using this project you will encounter two types of users/nodes, Clients and Miners. Lite Nodes or Clients, can be used to send or receive messages in exchange for currency, Blabbers. When receiving Transactions and Blocks by other nodes they simply store information that they may deem necessary, and echo the Transaction or Block before they discard it. This way clients help to propagate incoming messages to other nodes.

Full Nodes or Miners, are more important to the network. These nodes store the whole blockchain, validate incoming transactions or blocks and spend a considerable amount of their CPU power and storage to sustain the network. As result, these nodes are also the ones awarded currency when they successfully validate and attach blocks to the blockchain. Miners can be considered the labor-force of the network.

2.2 Block Struture

Figure 2 contains a simplified and easier to understand block structure. The following characteristics remain the same between different blockchain projects.

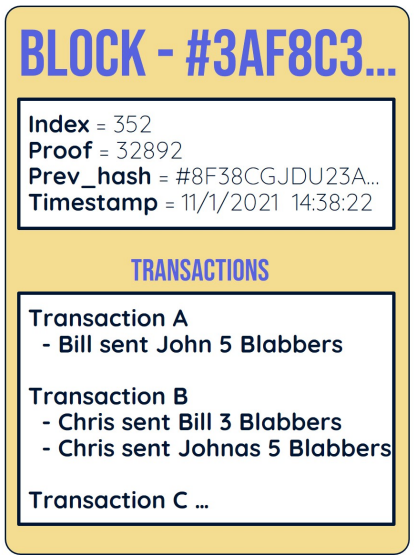

Fig. 2. A typical Block

- Every block is unique. Each block is identifiable by its unique hash or index.
- Every block follows a strict chronological order. To be validated each new block must have a timestamp higher than the previous one.
- Every block is linked with the previous one. Each block has a field containing the $previous_block_hash$. To be validated not only the index and timestamp must be higher than the previous block, but more importantly the $previous_block_hash$ must be valid!
- Every block must contain at least one transaction.
- Every block must be validated by peers.

2.3 Proof of Work

Proof of Work is the core mechanism that allows Uncuffed (and many other blockchain projects), to come to a consensus decentralized and agree on account balances, transactions, messages etc. It is a solution to a puzzle set by the blockchain to the miners. Solving this puzzle means a Miner can be allowed to append his block to the blockchain and be awarded for his contribution.

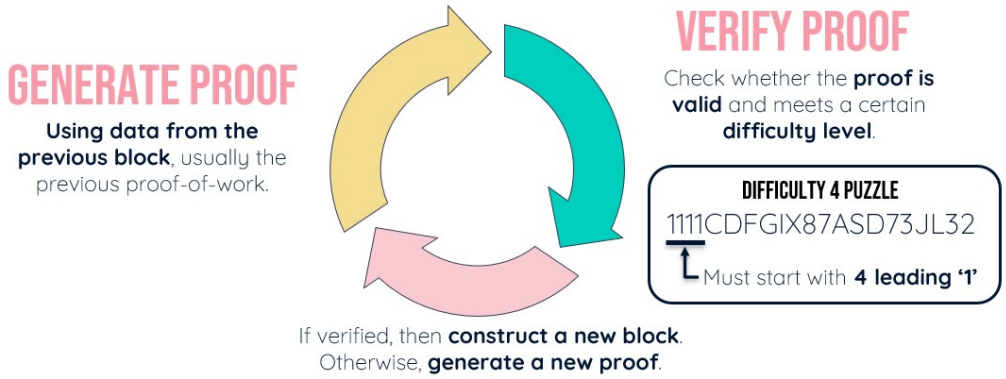

Fig. 3. Proof of Work

As soon as a new valid block is received in a Miner's blockchain he starts generating a new Proof of Work. The Miner uses the data from the previous block, usually the previous proof-of-work to solve the puzzle.

What is the puzzle? To generate a random set of characters, the proof of work, where the first X (where X is difficulty) characters must match a specific pattern. In the example above we can see a difficulty 4 puzzle, meaning the first 4 characters must match a specific pattern, in this case 1111.

This concept works fantastically because in order to generate those random characters, using the data from the previous block (in this case the previous proof), it may take many thousands or millions of calculations depending on the difficulty.

To verify though that this proof is actually correct, we only need two variables; The previous proof and the proof a miner claims to have found. Meaning, a Miner may take a huge amount of processing time in order to find a valid proof-of-work that solves the puzzle, but as soon as he does any miner or client can verify that proof with a single and fast calculation. In our case the puzzle is solved with fixed difficulty of 2, which is quite easy to be fair. But easy is what we are looking for in order to update the user's chat faster.

2.4 Transactions

Usually, transactions signal the transfer of wealth from a user to another. This remains the case for Uncuffed, but with the additional payload of PLAINTEXT or ENCRYPTED messages.

Similarly to a block, each transaction is uniquely identifiable by its hash. This hash is generated using the transactions contents, such as the sender, receiver, value and timestamp.

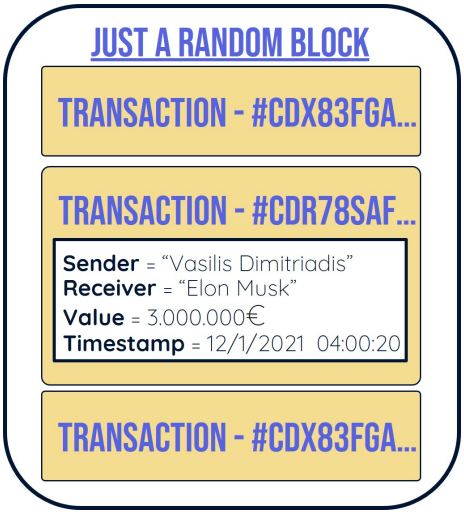

Fig. 4. A typical transaction

3 UTXOS (UNSPENT TRANSACTION OUTPUTS)

The term UTXO refers to the amount of digital currency someone has left after executing a transaction. Each single transaction entity contains two lists; Transaction Inputs and Transactions Outputs

The first one, Transaction Inputs, is easier to understand. It can be considered as a Pointer to a specific Transaction Output.

The Transacton Output stores the transfer of wealth, meaning it is a "coupon" saying that the X User has received Y amount of money. Each transaction output can be considered either SPENT or UNSPENT, aka UTXO (Unspent Transaction Output) or STXO (Spent Transaction Output). The simplest way to demonstrate this is with an example.

3.1 UTXO Example

In a classical banking system, Vasilis may have 3.005.000 Euros, which were acquired by many years of hard work. That money is stored as a single variable, money, in a database of a centralized bank. If Vasilis wanted to transfer 3.000.000 Euros to Elon Musk he could just write a check of 3.000.000 Euros and immediately his bank account would update that money variable with his new balance.

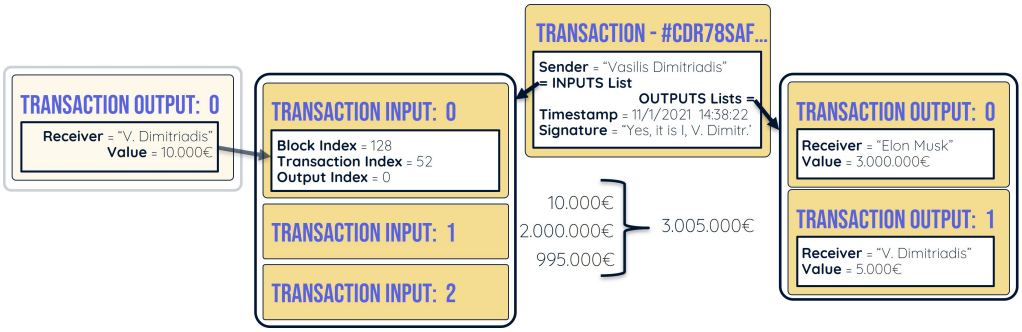

Fig. 5. UTXO Example

In our decentralized banking system, this concept wouldn't be sustainable. As such lets consider we know the exact transactions where Vasilis got his 3.005.000 balance. Specifically Vasilis received as single transactions:

- 2.000.000 from an Index Fund
- 995.000 from his job
- 10.000 from a friend

Now for Vasilis to send that amount of money to Elon Musk he would have to collect all these 'coupons' that say he has X amount of money (a.k.a Unspent Transaction Outputs) in a single list (Input List).

Now Vasilis wants to send 3.000.000 to Elon Musk, but he has 3.005.000 Euros. As such he creates a new list (Output List), where he sends 3.000.000 to Elon Musk and returns 5.000 to himself. The transaction is complete.

Although this might seem a complicated system, in actuality, this system allows every Miner to keep track of the UTXOs and quickly verify if a single transaction is valid or invalid without having to ask a central authority!

4 VALIDATION

Validations occur on each element of the Blockchain, from the Blockchain itself all the way to each single Transaction Input and Output. We demonstrate the whole validation process starting from the whole Blockchain. If someone wanted to validate a single element presented here, all he would need to do is follow the graph and validate the ones right to it.

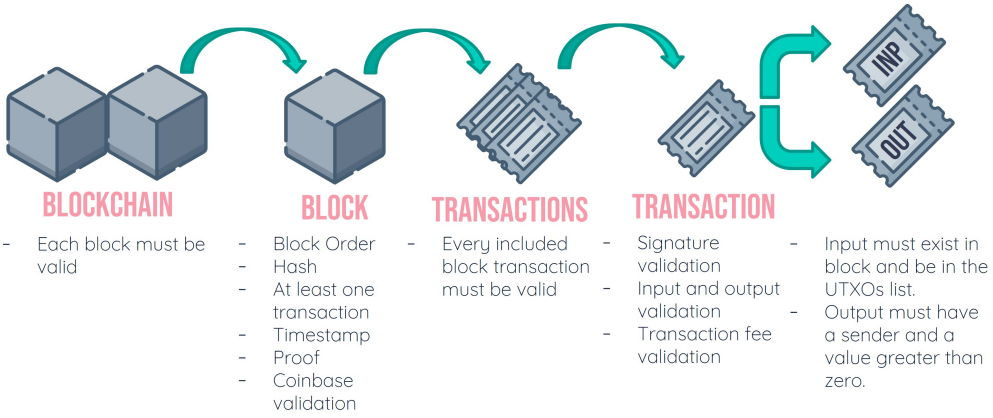

Fig. 6. Validations

To be valid,

- A Blockchain, must validate all each blocks
- A Block:
  - Must have the correct ascending block order in the blockchain
  - Must have a valid hash
  - Must have at least one transaction
  - Must follow a strictly chronological order with its timestamps
  - Must have a valid proof
  - Must have a valid miner reward, which is the sum of the transaction fees + mining reward. This transaction is called a Coinbase Transaction
- Each Block's Transactions must be valid.
- Each Transaction:
  - Must have a valid signature, signed by the sender to verify ownership
  - Must have valid transaction Inputs and Outputs, meaning you can't send more than you have
  - Must have a valid transaction fee going to a miner (0 or above)
- Each Transaction Input must exist in a block and in order to be valid it must also exist in the UTXO list
- Each Transaction Output must have a sender and a positive wealth exchange value

5 MESSAGE PAYLOAD

One issue that remains to be clarified is how are these messages sent all over the network. Each message is inserted as a message payload inside a Transaction Output.

```
def __init__(self, recipient_address: str, value: int, message: Optional[AMessage]):
    self.recipient_address: str = recipient_address
    self.value: int = value
    self.message: Optional[AMessage] = message
```

Fig. 7. Message Payload

The bigger the message, the more Blabbers you are going to need in order to transmit the message. This is done by the following simple calculation, which in essence Maximum between 1 and (message size in bits) / 64:

The message payload is the combination of an enumerator EMessageType and the actual string message message. The project supports the following messages: Plain Text Message, Encrypted Message, Image Message and Encrypted Image Message. Each one of these has its own dedicated class.

- The Plain Text Message simple attaches the user's message in the message variable.
- The Encrypted Message encrypts using the RSA Algorithm (with ECB style) prior to attaching the user's message in the message variable. While this is not optimal for an actual blockchain, it is more than enough for this presentation.
- The Image Message takes the uploaded image and converts it to a base64 string prior to sending the message, while the Encrypted Image does exactly the same thing but also uses the RSA Algorithm to encrypt it prior to sending it.

Upon receiving a message, a Client checks if the message is for him. If the message is plaintext it just stores it directly. In the case where the message is Encrypted, the client users his private key to decrypt the message and then stores it to his chat.

6 CONCLUSIONS

Secure messaging is very important especially in critical applications like communication of automated cars. In this article we presented a simulation of a blockchain messaging platform based on the Bitcoin protocol. The mechanism can be used for exchanging privately and securely messages and images. Each member of the blockchain can become a miner and collect rewards or just a plebeian client and send messages without supporting the platform.

REFERENCES

UP Ellewala, WDHU Amarasena, HV Sachini Lakmali, LMK Senanayaka, and AN Senarathne. 2020. Secure Messaging Platform Based on Blockchain. In *2020 2nd International Conference on Advancements in Computing (ICAC)*, Vol. 1. IEEE, 317–322.

Mohamed Amine Ferrag, Makhlouf Derdour, Mithun Mukherjee, Abdelouahid Derhab, Leandros Maglaras, and Helge Janicke. 2018. Blockchain technologies for the internet of things: Research issues and challenges. *IEEE Internet of Things Journal* 6, 2 (2018), 2188–2204.

Mohamed Amine Ferrag and Leandros Maglaras. 2019. DeliveryCoin: An IDS and blockchain-based delivery framework for drone-delivered services. *Computers* 8, 3 (2019), 58.

Mohamed Amine Ferrag, Leandros Maglaras, Ahmed Ahmim, Makhlouf Derdour, and Helge Janicke. 2020. Rdtids: Rules and decision tree-based intrusion detection system for internet-of-things networks. *Future internet* 12, 3 (2020), 44.

Mohamed Amine Ferrag, Leandros Maglaras, and Abdelouahid Derhab. 2019. Authentication and authorization for mobile IoT devices using biofeatures: Recent advances and future trends. *Security and Communication Networks* 2019 (2019).

Kahina Khacef and Guy Pujolle. 2019. Secure Peer-to-Peer communication based on Blockchain. In *Workshops of the International Conference on Advanced Information Networking and Applications*. Springer, 662–672.

Satoshi Nakamoto. 2008. Bitcoin: A peer-to-peer electronic cash system. *Decentralized Business Review* (2008), 21260.

Rakesh Shrestha, Rojeena Bajracharya, Anish P Shrestha, and Seung Yeob Nam. 2020. A new type of blockchain for secure message exchange in VANET. *Digital communications and networks* 6, 2 (2020), 177–186.

Raman Singh, Ark Nandan Singh Chauhan, and Hitesh Tewari. 2021. Blockchain-Enabled End-to-End Encryption for Instant Messaging Applications. *arXiv preprint arXiv:2104.08494* (2021).

Nik Unger, Sergej Dechand, Joseph Bonneau, Sascha Fahl, Henning Perl, Ian Goldberg, and Matthew Smith. 2015. SoK: secure messaging. In *2015 IEEE Symposium on Security and Privacy*. IEEE, 232–249.