12/12/2021

PROGRAMMING A CRYPTO-CURRENCY

1. INTRODUCTION

Since 1977 financial institutions have used the same payments system called the Society for Worldwide Interbank Financial Telecommunications (SWIFT) for national and international payments (S.W.I.F.T. SC, n.d.). In the UK, 84% of debit cards are Visa, and 91% in Ireland are, which is an overwhelming majority (Worldpay, 2020). After discovering this, it prompted me to research modern and improved systems for payment processing which led me to discover crypto-currencies. Crypto-currencies were first conceptualised in 2008 by the pseudonymous inventor Satoshi Nakamoto (Nakamoto, 2008). Nakamoto designed his crypto-currency to allow payments between parties without going via a financial institution, allowing it to have lower fees and not rely on one central authority that could manipulate the system however it sees fit. The issue with current payment processors is that they are big companies that are majority stakeholders. This allows them to set the standard to suit them best, meaning that consumers have to put up with it. I was inspired to create my crypto-currency to learn more about how they work, as I believe that they will be an essential technology in the future and will create a shift when it comes to payment processing.

Along with this activism for fairer payment processing, I also have a passion for computer science, and by completing this project, I will have improved my programming skills and expanded my theoretical knowledge. My secondary research consisted of computational cryptography, device networking and web Application Programming Interfaces (API). Upon completion, the crypto-currency was run through multiple tests to see its performance against other crypto-currencies, how it handled errors and failure and how it compared to current payment processors. I then used these tests to see how effective and efficient my end product was and how it compared to current solutions.

2. RESEARCH

A. Hashing

Cryptographic hashing is the mathematical function that allows the entire blockchain to be verified. The hashing algorithm that I used is called Secure Hashing Algorithm 256 (SHA256), which the United States of America's National Security Agency developed (Federal Information Processing Standards, 2002). The function of a hashing algorithm is to generate a fixed-length result (hash) that is unique for each input and does not have any collisions. Collisions are where different inputs have the same output. Most hashing functions are irreversible so that, given the output, you cannot figure out what the input is, which is a requirement for it to be used in crypto-currencies as it is essential for the proof of work consensus mechanism to function.

These key attributes allow data to be verified by hashes as any changes in the data will result in a different hash so the computer can tell if something has changed. Therefore a computer does not have to trust any other computers as it can tell when the data has been changed or manipulated.

Even the slightest change in input can completely change the output. See Figure 1.

```
from Crypto.Hash import SHA256

data1 = "hello"
data2 = "hell0"
data1_hash = SHA256.new(data1.encode()).hexdigest() # Hashes data and returns hex
data2_hash = SHA256.new(data2.encode()).hexdigest()
print(data1_hash) # Prints hash value
print(data2_hash)

# Output
>>> 2cf24dba5fb0a30e26e83b2ac5b9e29e1b161e5c1fa7425e73043362938b9824
>>> bdeddd433637173928fe7202b663157c9e1881c3e4da1d45e8fff8fb944a4868
```

Figure 1 – Example of hashing text in Python using SHA256

B. Asymmetric Encryption

Encryption encodes information so that only authorised parties can read it (Hassan & Hijazi, 2016). Symmetric encryption works using a key that is used to encrypt and decrypt. Asymmetric encryption works differently from symmetric encryption as it uses two different keys, i.e. not symmetric. Usually, one key is the private key, and the other is the public key. Data encrypted with the private key can only be decrypted with the public key, and data encrypted with the public key can only be decrypted with the private key.

Asymmetric encryption is used to verify transactions and ensure they are not fraudulent. This is done by encrypting the transaction with the private key and storing the encrypted data (ciphertext) with the transaction. This means that people can verify the transaction's legitimacy by decrypting the ciphertext with the public key, and if the decrypted data is the same as the data in the transaction, then it is a legitimate transaction (R.L., et al., 1977). This can be done without the computer trusting any other computer as it can be mathematically proven similar to how hashing can be. This process is known as signing, where the encrypted data is the signature.

C. Block Data Structure

A block is a data structure that contains transaction information and is immutable (cannot be changed). Each block has a hash of its data and the previous block's hash. This is a security feature; a malicious or accidental change in a historic block would result in all blocks after that being invalid as the hash would change. To ensure all participants of the P2P network have the same data, all transactions are put into blocks to be processed. A block can contain hundreds of transactions that can be verified and distributed at once, making it more efficient than verifying each transaction individually. Blocks are created at periodic intervals, allowing the nodes to reach a consensus. (Smith, 2022)

i. Attributes of a Block

1. *Timestamp – The time the block was mined at usually in UNIX timestamp form*
2. *Block Number – How many blocks have been added so far*
3. *Difficulty – Computational power required to mine the block*
4. *Hash – Unique hash for that block*
5. *Parent Hash – The hash of the previous block*
6. *Transactions – All the transactions in the block*
7. *Nonce – Number used to prove that the block has been mined*

D. The Blockchain

The blockchain is the data structure that links all blocks and transactions together. Essentially, it is an immutable database of all transactions spread across multiple computers that synchronise their data via a P2P network (The Economist, 2015).

An abstracted view of a blockchain would be that it is an immutable linked list. It is immutable due to the block above storing the previous hash, which makes the blockchain resistant to modification once it has been recorded without changing successive blocks and re-mining them. With a long enough blockchain, it becomes computationally impossible to re-mine every block. The previous hash stored in the block also makes it a linked list, as each block links to the last.

E. Multi-threading

Multi-threading is a method in which computer software can be split up and run on multiple processors. For example, if there was a four-core processor, you could split the program up to run a part on each core, making it up to 4 times faster. It can also have multiple parts of an application running simultaneously; for example, the interface and logic run on separate cores allowing them to complete their tasks without interrupting each other and providing an improved user experience (UX). See Figure 2. (Oracle Corporation and/or its affiliates, 2010)

Implementing this in my program allowed me to have numerous nodes connected as the networking module can be assigned a core so that it does not interrupt the other processes. It also allowed me to speed up the miner as the software could create batch jobs and give each core a job to complete. Speeding it up by however many cores were allocated. This increased the program's overall performance, allowing transactions to be processed more often.

```
from threading import Thread

x = lambda y: print(y) # Creates function for thread

thread1 = Thread(target=x, args=(1,)) # Creates thread with target functiom
thread2 = Thread(target=x, args=(2,))
thread3 = Thread(target=x, args=(3,))

thread1.start() # Starts thread
thread2.start()
thread3.start()

# Output
>>> 1
>>> 3
>>> 2
```

Figure 2 – Python implementation of multi-threading

F. Peer to Peer Networks

Peer to peer (P2P) networks are a type of computer network where there is no central computer orchestrating the actions of other computers (J.Alpern & J.Shimonski, 2010). Instead, all computers (nodes) on a decentralised network are of equal status and are all working towards a common goal in the context of crypto-currencies, processing transactions on the blockchain.

The benefits of a P2P network are that it is decentralised, meaning that it is not one entity; instead, it is made up of entities all independent of each other; it cannot be controlled by one entity because for changes to persist, the majority of nodes must agree on it, and P2P networks can be more powerful as they can consist of more computers and are not constrained to one system. (Lei & Kong, 2020)

Reiterating on the points from hashing and asymmetric encryption, a P2P network allows the computer to trust nothing and instead work off a majority vote from its connected nodes, filtering out malicious behaviour.

G. Proof of Work Consensus Mechanism

Proof of work is a consensus mechanism to verify and prove blocks which are done by what is called mining. Mining is where a computer will use its computational power to prove a block. It does this by hashing the data in the blocks and trying to get a hash with a specific difficulty. For example, the hash must start with ten 0s. It does this by changing a number in the block called the number used one (nonce). It will keep changing the nonce and rehashing until it meets the required difficulty. Once it has accomplished this, it sends the block to all the other nodes on the network. For this work, the miner will get a reward called a coinbase which is how new currency is minted. (Smith, 2022)

The way a node will choose which block to go with is by which took the most computational power to produce, i.e. the greatest difficulty. When a node has disconnected and needs to resync the transaction data, it cannot trust anything; it must decide whom to believe somehow, so it goes with the longest chain as it took the most computational power to produce, so it must have taken more nodes to make, i.e. a majority. (Agarwal, 2022) (Bitcoin, 2009)

H. Web Application Programming Interface

A web Application Programming Interface (API) allows data to be sent and retrieved programmatically (W3, n.d.). This allows users to log in to websites or watch their favourite cat videos. They work by sending Javascript Object Notation (JSON) data in a web request to a server. The JSON will contain a link to the information you want to fetch, e.g. the video identifier or data you want to store, e.g. a social media comment. The server will then send back a response usually containing JSON, which will have the data you want to fetch, a success message or an error message.

JSON consists of keys and values. You get the value by using its key. It is also language-independent, meaning that it can be used with any programming language giving me a lot of freedom when it comes to choosing one.

```
{
   "employees" :[
      {"firstName":"John",
       "lastName":"Doe"},
      {"firstName":"Anna",
       "lastName":"Smith"},
   ]
}
```

3. DEVELOPMENT

A. Initial Planning

During my initial design process, I wrote down a specification for what I wanted the crypto-currency to fulfil.

- Decentralised
- Fault-Tolerant
- Sign & Verify Transactions
- Mine Blocks
- Interact via a Web API

Once I had this list, I split the program into four modules: blockchain, P2P network, block miner, and web API. I made a flow chart for each of these modules to map out how they would work independently. This was key as it allowed me to scope the project, so I did not get carried away with implementing other features I thought of during development.

After I had an idea of the inner workings of each module, I decomposed them further and wrote each part on post-it notes colour coded to each module. These were then placed on a corkboard with different sections for done, doing, urgent, backlog and need to work on. This is known as a kanban board and is used by many companies to help develop complex applications and is what inspired me to use it for my project. Using this allowed me to track development and know what I needed to work on next whilst keeping me on track with my deadline and in scope with my specification. Throughout the development, I used the kanban board and the flow charts to ensure my program was working as I designed it to.

If I were to go back and do something different during the planning stage, I would have also created a design document for each module. This would allow me to have even finer granularity over the design and would have shortened the development. With a flow chart, you simply design the logic but not how that will be implemented. This wasted time because I had to reimplement features that did not work. Had I made design documents, I would have been able to develop the program in line with it and encounter far fewer of these incidents.

B. Blockchain Module

At this stage of the development, I developed the blockchain module, which was made up of 3 core components; transactions, blocks and the blockchain. This section will focus on each component, allowing me to detail the problems I encountered and what I would have done differently.

i. Transactions

The transaction component was the smallest part to implement whilst also being an essential part as it enables transactions to be verified and stops people from creating fraudulent transactions. This works via the asymmetric

encryption mentioned before. For this, I used the Crypto module in the python standard library. Additionally, this meant that there was excellent documentation and online discussions. As a result, I encountered very few errors or issues whilst implementing.

ii. Blocks

The block component was significant as I had to implement quite a few functions. This is where a design document would have come in handy, as I refactored the code a lot and completely redid functions. The main reason for this was that when hashing the blocks, it has to be in the data type Javascript Object Notation (JSON), but when it needs to be stored, it must be in tuple form. This caused me many headaches as I did not realise this until afterwards, so I had to rewrite a large proportion for it to work correctly.

iii. Blockchain

This was the most complex component of the blockchain module, as it required me to combine many moving parts like blocks and transactions. This added another layer of complexity because, as previously mentioned, I had issues converting between the data types, but up until this point, I had not thought that the transactions would need to be stored persistently, so I had to write some more functions to convert them into a tuple type to store in the database. Luckily, the transaction, a reasonably simple module, did not take up much time, so it allowed me to carry on with development and did not cause much delay.

For storing the blockchain persistently, I used SQLite as I did not want to redesign the wheel when it came to persistent data storage as SQLite has been around since 2000. Consequently, I did not spend too much time on the storage and was able to focus more on how the data is interacted with. I used a declarative language called Structured Query Language (SQL) when interacting with the blockchain data. This allowed me to search large amounts of data efficiently with little forework.

The most challenging part of the blockchain was block verification and making sure the blockchain was in a valid state. When you add a block, you must verify that it is correct and that all the transactions are valid. Again, this caused another headache when dealing with the different data types as transactions are signed in JSON form but stored in a tuple, so converting between is relatively inefficient. To verify each transaction, you must check if they have enough funds to transfer. With SQLite being the way it is, I had to get all the previous transactions and calculate their balance which is very inefficient. The second problem was SQLite whilst allowing you to query data; there is no simple way of extracting all the data from it for validation. This led to me having to fetch each block and validate them, which is inefficient, especially when you have a long blockchain.

If I were to do this again, I would probably implement my own method for persistent storage as that would allow me to implement more efficient methods for completing the tasks described above, therefore improving overall efficiency.

C. P2P Networking

The P2P network comprises two relatively simple components, but they interact with each other in a rather complex way, making this the most difficult to program overall. This is because they use multi-threading technology, which allows the program to have more connections but increases complexity. This was also a very new concept to me, so I found it even more difficult on top of the high complexity of multi-threading. Luckily, I found a lot of online resources to inspire me, such as (Snoeren, 2018). The two components of the P2P network are the connection handler and the node controller. I will talk about how I developed these and how they interact.

i. Connection Handler

The connection handler is what sends the messages between the nodes. This is done via sockets. I used the Sockets module from the Python standard library, which had lots of documentation, reiterating what I said earlier about the Crypto module. Each connected node will have a connection handler, which receives the messages and sends them back to the node controller. When receiving messages, it gets the message as just a long string of bytes which is then converted into a usable form. One of the issues is that you cannot tell when the message ends as there are no separators built-in. This meant I had to implement my own method of detecting when a message had ended. I did this using an End of Transmission Character (EOT_CHAR), an 8-bit value that you place at the end of a message. When a message is received, the program will look for the EOT_CHARs and split the messages based on their positions. This was a very effective yet simple solution.

When two nodes initially connect, they have to share information between them, like their Unique Identifier (UID) and all their clients. This proved to be quite a difficult challenge as when a socket connects, both nodes are of equal status, so neither knows if it is the `server` or `client`. Therefore they do not know who should send the data first. I had to implement a system for differentiating whether the node is the client or server. Had I created a design document, this would probably not have been as much of an issue.

ii. Node Controller

The node controller is the main component and is what receives the messages from the connection handlers to be interpreted and executed, like adding a new block or transaction. It also handles the initial connection of nodes until it is handed over to the connection handler. The node controller creates a new thread containing the connection handler when a new node connects. This allows the node to have up to 50 connections in perfect conditions with a powerful enough processor. I did this using the Python Threading library.

The node controller must keep track of all currently connected nodes. This is done via `heartbeat` messages. These are messages with no content to

check if the connection is still open. If the message fails to send, the program knows it is no longer connected. This stops the program from hanging onto closed connections and wasting resources, improving its efficiency.

iii. Cross Thread Communication

This was the most complex part of the P2P network as I had each module working independently and now had to make them work together. This was accomplished using threading events and function calls. When I closed the program due to the nature of threading, it only closed the node controller and left all the connection handlers running. I used threading events to solve this, which are boolean variables that can be switched between true and false. This is then shared between all the threads. When the program starts, the event is set to false, and when the program closes, it is set to true. All the connections close and terminate their thread when the event is true. This is a more graceful shutdown method instead of the lack of before.

For sharing data between the threads, like messages, I used functions. This may seem counterintuitive, but by calling the function of an existing object on another thread, it will then run the function on the other thread to handle that data—shown in Figure 3.

```
import threading


class Example(threading.Thread):
  def __init__(self):
    super(Example, self).__init__() # Inherit thread class
    self.msg = ""

  def run(self):
    prev_msg = ""
    while True:
      if self.msg != prev_msg:
        print(self.msg)
        prev_msg = self.msg

  def new_msg(self, msg):
    self.msg = f"From thread – {msg}"


thread = Example()
thread.start() # Run thread in background
thread.new_msg("Hello World") # Send data to thread

# Output
>>> From thread – Hello World
```

Figure 3 – A Python implementation of sharing data between threads

D. Block Miner

The block miner is the essential module when it comes to transaction clearing. Mining is what the proof of work consensus mechanism is based on. This only consists of one component as all it needs to do is hash data. The miner uses multi-threading to use multiple Central Processing Unit (CPU) cores to optimise the process allowing it to do more hashes per second. This was implemented using the Python Multiprocessing library. In the Multiprocessing module, it has a feature called worker pools, where you create a pool and allocate a set amount of CPU cores to use. Then you give it functions to run. The function is to hash the block recursively between two nonce values, e.g. 0-30000, to find one with the correct difficulty.

I would have used Graphical Processing Unit (GPU) cards to run the miner if I did this again. This is because they often have more than 3000 cores which is a lot more compared to the six cores of a CPU. Furthermore, this would significantly improve the speed of the miner leading to quicker block times hence executing more transactions per second. Also, I would have used a more efficient language as Python is not capable of computationally demanding applications like mining. My miner gets ~0.6MH/s, whereas miners used for Ethereum get ~28MH/s on the same hardware (Piotrowski, 2021).

E. Web API

The web API allows users to interact with the blockchain via desktop apps, websites and phone apps. It is built using the Flask Python module. Unlike many of the modules I used, this is not a standard library module, but it has been around since 2010 and has a community of over 58,000 people (Pallets, 2010). Along with that, it also has extensive documentation and plenty of tutorials online that helped me learn the framework. The web API for the node runs on the main thread and is blocking. Unfortunately, you must choose between running the miner or the web API. I did not see this as a problem, though, as anyone mining would probably not want to waste other resources on unnecessary processes. If I were to do this again, I would implement the ability to run them both.

4. EVALUATION

A. Did it Fulfil the Specification

The specification for my crypto-currency that I laid out was that it had to fulfil this list.

- Decentralised
- Fault-Tolerant
- Sign & Verify Transactions
- Mine Blocks
- Interact via a Web API

To my knowledge, it ticks all these boxes. It is decentralised; all the nodes are independent and do not rely on one central node, but they all work towards the same common goal. If any nodes were to go down, the state of the network would not be affected. It is fault tolerable; when an error occurs, it is logged and then handled so that it will not affect the program or any data on the blockchain. This makes my program more user friendly and overall better, as a simple error cannot take down the whole network.

Along with those, it can also sign transactions, meaning it is computationally impossible to forge a transaction on my crypto-currency without the private key. This makes it incredibly secure and trustworthy. Blocks can be mined on it; this is how the transactions are cleared and means that the longest blockchain is the correct one as it took the most computational power to produce, i.e. a majority. Users can also interact with the blockchain via a web API I created. This means users can add their transactions and make payments on the network.

B. How Can it be Improved

Due to the nature of the project, I decided to use the programming language Python. I chose Python because it has a simple syntax and is very good for quick prototyping; this allowed me to develop my crypto-currency quicker than if I used a more conventional language for crypto-currencies like Golang. Using Python did come with a trade-off, though; it is not very quick for processing. For example, the time between when a node connects to another node and when it has created a new thread and shared required information is ~2 seconds. This is slow for a computer program; it should be happening in milliseconds. Luckily this does not have much of an effect on the overall experience.

I would also implement a better system for the persistent storage of blockchain data. I used SQLite for my persistent storage in my program, but this led to a few issues like there not being an efficient way to verify data or verifying transactions taking a while because it had to fetch all previous transactions. This made the program run slower on top of the fact that it was written in a relatively inefficient language.

Given more time, I would implement a technology called smart contracts. These are pieces of code that run on the blockchain. This allows users to make programs that run in a decentralised manner, like creating their token on top of my blockchain. It could also allow Non-Fungible Tokens (NFT) to be created. Smart contracts open many doors for what could be done on the network and allow it to compete with the crypto-currencies of today.

C. How Does it Compare

Visa can handle in excess of 65,000 transactions per second (Visa, 2017). Ethereum can handle ~15 transactions per second, and Bitcoin can handle ~7 (Coinbase, 2021) (River Financial, n.d.). My crypto-currency can handle ~2

transactions per second. It may seem like my crypto-currency is not far off from others, but it is challenging to increase this whilst still maintaining decentralisation and not letting the size of the blockchain grow out of control as people need to be able to store it. However, I was not expecting to be near any of these since they have teams of more than 500 people working on them, at least (Macro Trends, 2021) (Buterin, 2014) (Bitcoin, 2009).

5. RESEARCH REVIEW

A. Ethereum.org

Ethereum.org is the official website for the Ethereum crypto-currency. Ethereum is currently the second most popular crypto-currency and has been around since July 2015. One of the sections on the website shows how crypto-currencies work from a more general view allowing you to grasp the key concepts from an abstracted perspective. Along with this, they have a section dedicated to documenting how Ethereum works and what all the sub-systems do. These two sections allowed me to grasp the inner working of crypto-currencies and Ethereum itself. Along with teaching how Ethereum worked, it also gave me insight into how to implement certain technologies, why they did something in a specific way, and why it was not implemented differently.

This proved to be my most valuable source as it gave me a foundation to build off, along with information on how I can build off of it. Not only was I able to read the information, but they had an interactive forum in which I was able to ask questions and converse with developers working on Ethereum. This was helpful, especially when I got stuck understanding a particular concept. One of the significant strengths of this source is that it is community-run and moderated, meaning that it is always up to date, and any incorrect information is usually removed swiftly as not to pollute any of the factual information. A potential limitation of this is that a lot of the people contributing to the website also own Ethereum, meaning that it is in their best interest to make it look good and seem like a robust system with no faults. This can lead to them leaving out details where things went wrong or could make Ethereum look bad, potentially driving the price down. However, I believe this is not an issue as Ethereum has always been very open about flaws or vulnerabilities found in the past.

B. Bitcoin: A Peer-to-Peer Electronic Cash System

The Bitcoin whitepaper was the first scientific paper to introduce a crypto-currency concept that could be implemented. It talks about the theory behind crypto-currencies and the problems that they aim to solve. Since it contained a

lot of the theory and was the first concept of crypto-currencies, it allowed me to create a scope of my project and what goals I wanted it to fulfil.

The whitepaper was written by Satoshi Nakamoto, who, to this day, no one is sure who they are. Luckily, this impact on the source's credibility is negligible as all of the points are cited and proven with either mathematical equations or computer code. Also, it is a scientific paper, so there are little to no opinions to be argued, and instead, the majority of the content is sharing facts and specifications.

C. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems

This source was another scientific paper and has stood the test of time, being cited over 24000 times showing the influence this paper had on further studies. It has also been fully referenced, improving reliability as each piece of information can be traced back to its origin. The authors of this paper are also very well known researchers and are highly respected in their field of research, cryptography. RSA is the method of encryption I used to sign transactions so that they could not be forged. Since RSA is such a well-established algorithm, I knew I could trust the paper for my research, and in my program, due to the rigorous testing it has had over the years.

D. Scaling Ethereum & Crypto for a Billion Users

Coinbase.com is a crypto-currency exchange founded in 2012. They have extensive knowledge of crypto-currencies, as shown by their success in the crypto exchange market. This solidifies the credibility of the source. However, they are a publicly-traded company, so they are legally obliged to make money as it is in their shareholder's best interest. Furthermore, they will also hold a large amount of crypto-currency, so any good press they release could increase its value due to their high profile position in the crypto-currency world. This could tempt them to write articles with inflated or optimistic opinions and facts. However, many of their articles are published by industry leaders who would not want to tarnish their reputations.

E. Visa About Visa Factsheet

Visa is an international payment processing company with a majority stakeholder in debit and credit card payment processing (Worldpay, 2020). This means that it is in their best interest to stay dominant in their field, entice merchants to use them, and get banks to use them. However, the source I used is a factsheet that they released to promote themselves and inform businesses, so any lies on there would be unlikely as they would not want to acquire new clients fraudulently.

6. REFERENCES

Agarwal, A., 2022. *Proof of Work (POW).* [Online]
Available at: https://ethereum.org/en/developers/docs/consensus-mechanisms/pow/
[Accessed December 2021].

Bitcoin, 2009. *bitcoin/bitcoin: Bitcoin Core integration/staging tree.* [Online]
Available at: https://github.com/bitcoin/bitcoin
[Accessed December 2021].

Buterin, V., 2014. *Official Golang Implementation of the Ethereum Protocol.* [Online]
Available at: https://github.com/ethereum/go-ethereum
[Accessed December 2021].

Coinbase, 2021. *Scaling Ethereum & crypto for a billion users.* [Online]
Available at: https://blog.coinbase.com/scaling-ethereum-crypto-for-a-billion-users-715ce15afc0b
[Accessed December 2021].

Federal Information Processing Standards, 2002. *SECURE HASH STANDARD.* [Online]
Available at:
https://csrc.nist.gov/csrc/media/publications/fips/180/2/archive/2002-08-01/documents/fips180-2.pdf
[Accessed December 2021].

Hassan, N. A. & Hijazi, R., 2016. *Data Hiding Techniques in Windows OS.* 1st ed. s.l.:s.n.

J.Alpern, N. & J.Shimonski, R., 2010. *Eleventh Hour Network+.* 1st ed. s.l.:s.n.

Lei, J. & Kong, L., 2020. 2 - Fundamentals of big data in radio astronomy . In: *Big Data in Astronomy.* s.l.:s.n., pp. 29-58.

Macro Trends, 2021. *Visa: Number of Employees 2010-2021 | V.* [Online]
Available at: https://www.macrotrends.net/stocks/charts/V/visa/number-of-employees
[Accessed December 2021].

Nakamoto, S., 2008. *Bitcoin: A Peer-to-Peer Electronic Cash System.* [Online]
Available at: https://bitcoin.modeapp.com/bitcoin-white-paper.pdf
[Accessed December 2021].

Oracle Corporation and/or its affiliates, 2010. *Understanding Basic Multithreading Concepts.* [Online]
Available at: https://docs.oracle.com/cd/E19455-01/806-5257/6je9h032e/index.html
[Accessed December 2021].

Pallets, 2010. *pallets/flask: The Python micro framework for building web applications..* [Online]
Available at: https://github.com/pallets/flask
[Accessed December 2021].

Piotrowski, D., 2021. *Best Ethereum Mining Software for Nvidia and AMD.* [Online]
Available at: https://2miners.com/blog/best-ethereum-mining-software-for-nvidia-and-amd/
[Accessed December 2021].

R.L., R., A., S. & L., A., 1977. *A Method for Obtaining Digital Signatures and Public-Key Cryptosystems.* [Online]
Available at: https://people.csail.mit.edu/rivest/Rsapaper.pdf
[Accessed December 2021].

River Financial, n.d. *How Bitcoin Can Scale.* [Online]
Available at: https://river.com/learn/how-bitcoin-can-scale/#:~:text=Scaling%20the%20Bitcoin%20Blockchain,-The%20reason%20Bitcoin's&text=In%202010%2C%20Satoshi%20Nakamoto%20established,around%201.3MB%20in%20size.
[Accessed December 2021].

S.W.I.F.T. SC, n.d. *SWIFT history | SWIFT - The global provider of secure financial messaging services.* [Online]
Available at: https://www.swift.com/about-us/history
[Accessed December 2021].

Smith, C., 2022. *Blocks.* [Online]
Available at: https://ethereum.org/en/developers/docs/blocks/
[Accessed December 2021].

Smith, C., 2022. *Mining.* [Online]
Available at: https://ethereum.org/en/developers/docs/consensus-mechanisms/pow/mining/
[Accessed December 2021].

Snoeren, M., 2018. *Python Implementation of a Peer-to-Peer Decentralised Network.* [Online]
Available at: https://github.com/macsnoeren/python-p2p-network
[Accessed December 2021].

The Economist, 2015. *The great chain of being sure about things.* [Online]
Available at: https://www.economist.com/briefing/2015/10/31/the-great-chain-of-being-sure-about-things
[Accessed December 2021].

The Python Software Foundation, 2001. *Python 3.10.3 documentation.* [Online]
Available at: https://docs.python.org/3/
[Accessed December 2021].

Visa, 2017. *About Visa Factsheet.* [Online]
Available at:
https://www.visa.co.uk/dam/VCOM/download/corporate/media/visanet-technology/aboutvisafactsheet.pdf
[Accessed December 2021].

W3, n.d. *Web APIs - Introduction.* [Online]
Available at: https://www.w3schools.com/js/js_api_intro.asp
[Accessed December 2021].

Worldpay, 2020. *Market share of international and domestic payment card schemes in 15 countries in Europe in 2020.* [Online]
Available at: https://www.statista.com/statistics/1116580/payment-card-scheme-market-share-in-europe-by-country/
[Accessed December 2021].