# Conceptual Architecture of Cryptonite

**Group #16**

Amanda Misek – 19aam18@queensu.ca

Benjamin Tiong – 19bkyt@queensu.ca

Bilal Imran – 20mbi@queensu.ca

Christian D'Souza – 20ceds@queensu.ca

Decheng Zhu (Eric) – 18dz6@queensu.ca

Vanshita Uthra – 19vu1@queensu.ca

***Table of Contents***

## *1.0 Abstract*

This report looks at the conceptual design of Bitcoin Core. Bitcoin Core is a software that can help users use and manage their bitcoins. It serves as a digital wallet by storing all a user's bitcoins and facilitating the sending and receiving of payments. After carefully reviewing relevant public documentation, we determined that Bitcoin Core uses a peer-to-peer architecture style. This design, which allows direct online payment between parties without the need for any financial institutions, is based on the system's decentralized trust principles. The system can be broken down into subsystems like blockchain, mining, wallets, nodes, consensus mechanisms, smart contracts. With the help of two use cases and the suggested architecture, we demonstrate the transactions of decentralized payments and the verification of those payments. Overall, this paper determines the conceptual architecture of Bitcoin Core, and will help us come up with the subsequent concrete architecture in the future.

## *2.0 Introduction and Overview*

Bitcoin is one of many cryptocurrencies created in 2009 by an anonymous group of developers called Satoshi Nakamoto, that changed the digital payment ecosystem (Investopedia). This virtual system allows peer-to-peer financial transactions without the need of any bank or third party involvement. Even today companies such as PayPal and Square have implemented this virtual network technology and have increasingly become more popular (Investopedia). Furthermore, bitcoin uses blockchain; a data structure consisting of an ordered, backlinked list of transactions. More specifically, the overarching system of bitcoin is structured to operate on the network, have the transactions validated, and then add the transactions to the world-wide ledger.

A main implementation of bitcoin is Bitcoin Core. This open-source project implements many of the aspects of bitcoin including wallets, transactions, and accessing the bitcoin network. Due to these advancements in the virtual payment industry, Bitcoin Core has become one of the most popular software used within the bitcoin network (River). Since Bitcoin Core is open-source, this allows many developers to ensure the network is constantly being improved and maintained while allowing anyone to view, copy, and suggest updates (River). Bitcoin Core is the original and most used software within the bitcoin network made up of many nodes, also known as members. Being the initial implementation, Bitcoin Core is often referred to as a template or guideline for standards and structure for other structures in the network (River).

This report showcases the architectural structure and style of Bitcoin Core while applying the concepts and materials we have studied in our course lectures. Included in this report, is the architecture of Bitcoin Core breaking down in detail the subsections of the network. The Bitcoin Core system is broken down into six subsystems including blockchain, mining, wallets, consensus, mechanisms, and contracts. Additionally, there are system diagrams, in depth explanations of external interfaces, descriptions of use cases, and a data dictionary. This is followed by our naming conventions, conclusion summarizing our findings, common lessons learned, and references. The conclusion made is that the architecture of Bitcoin Core is described as peer-to-peer or blockchain and is explained in detail in the following section.

## *3.0 Architecture*

### 3.1 Style

Peer to Peer Network

The architecture style of the Bitcoin Core system is a peer-to-peer network. In this style, the full nodes (peers) collaboratively maintain a peer-to-peer network to exchange information, and data/assets without having the involvement of a central authority. In the Bitcoin-core system, this style is applied when cryptocurrencies are being exchanged. Encryption and creation of blockchain to allow a transaction to occur in a safe manner without the involvement of a third party. You can connect to a peer by sending a version message that has the version number, block and current time to the remote node. A "verack" message is sent to another node to show that the connection has been established.

### 3.2 Structure of The System

Block Chain

Block chain provides Bitcoin its public ledger where it can record transactions in a certain time frame. This protects people from double spending and modifying the old transactions. The nodes in the Bitcoin network store a block chain of approved blocks. The nodes are considered to be in consensus when the nodes have the same blocks in their blockchain. In order to maintain the consensus, the nodes follow these validation rules called consensus rules. First, the block of multiple transactions is collected in the transaction portion of the block. The copies of each transaction are hashed and then paired until just one hash remains which is the Merkle root of the Merkle tree. The Merkle root is then stored in the block header. The blocks are chained by the block storing the hash of the previous block's header ensuring that the transactions are not modified until the block that records it is modified. Furthermore, the Bitcoin wallet software ensures that the satoshis are sent from and to the wallet while Bitcoins move from one transaction to another. The transactions display how the previously received satoshis are spent, indicating that the one transaction's input is the previous transaction's output. Outputs of all transactions are categorized as Unspent Transaction Outputs which are used to ensure that the payment is valid. The value of the transaction's output must not exceed the value of the transaction's input as it can cause the transaction to be rejected.

## Block Height and Forking

A Bitcoin miner is responsible for hashing a block below its target threshold so it can be added to the block in the chain. These blocks are usually identified by their block height which is the number of blocks between the first block and the block header. Blocks can have the same height when two or more miners produce a block at the same time, creating a fork where each node chooses which block it wants to accept. When certain conditions are not present, the nodes accept the first block they encounter. Furthermore, the fork of one side is made stronger when a miner produces another block that is attracted to another. However, blocks that belong to shorter forks are disregarded. Block height cannot be used as a globally unique identifier as multiple blocks can have the same height.

## Transaction Data

Each block is required to have one or more transactions. The first transaction is a coinbase transaction called generation transaction which collects and spends the block reward. There is a special condition by the UTXO of a coinbase transaction stating that it cannot be spent for more than one hundred blocks. Through this, the miner is prevented from spending the transaction dees and block reward as the block can be identified as stale later. Furthermore, blocks do not need to include non-coinbase transactions but miners must include them always to collect their transaction fees.

## Consensus Rule

All nodes validate blocks using the same consensus rules in order to maintain consensus. Consensus rules are changed sometimes to add new features or prevent network abuse. When a new consensus rule is introduced to a block, the new rule is accepted by the upgraded nodes and rejected by the non-upgraded nodes. In this case, the mining software refuses to build on the same chain creating divergent chains called a hard fork. If the new consensus rules are violated by a block, it will be rejected by the upgraded nodes but accepted by the non-upgraded nodes. This is called a soft fork.

## Proof of Work

Proof of work is a consensus mechanism which involves a peer to peer interaction. When blocks are chained, it is impossible to modify the transaction inside with the block without modifying the following blocks. This results in an increase in cost when a new block is added. It takes upon an advantage of having a cryptographic hash algorithm, which converts arbitrary data into a random number. A new number is generated whenever the data is modified when running.

## Mining

Mining is a process of adding new blocks to the block chain, making the transaction history hard to modify. There are two forms of mining, solo mining and pooled mining.

## Solo Mining

Solo mining is when the miner creates a new block on his own with all the proceeds earned from the block reward and transaction fees going to himself, allowing him to have a longer time between payments. The mining software uses a component called bitcoind to acquire new transactions from the network. A template is used by the mining software to construct a block and create a block header. The block header is sent to its mining software along with a target threshold. To generate the corresponding hash, the mining hardware is required to iterate through every value of the block header nonce. The hashes are not below the threshold and then the mining hardware gets updated.

## Pooled Mining

Pooled mining is the mined “pools” resources with other miners where the proceeds are shared between the miners according to their hashing power, allowing them to have a shorter period of time between payments. New transactions are acquired from the network through the usage of bitcoin. The target threshold is set a few magnitudes higher than the network difficulty. This results in the mining hardware returning multiple block headers which do not hash to a value that is eligible to be included in the blockchain. If the hashes are below threshold, the mining hardware returns the block header along with a successful nonce to the mining software. The pool receives a copy of the information by the miner that needs to be validated by the pool to ensure that the header hashes below the target value and the block of transactions that are referred to by the header merkle root are valid.

## Stratum

A stratum gives miners the minimum information that they require to create a block on their own. A stratum gives the following information. Firstly, it gives the required information to create a coinbase transaction paying the pool. Secondly, when the coinbase updates a transaction with a new nonce, the merkle tree is required to be rehashed in order to create a merkle root. Next, a block header must contain all non-merkle root information. Lastly, it gives the current threshold target of the mining pool is accepting shares.

## Wallet

The wallet program in Bitcoin allows the creation of public and corresponding keys in order to receive and spend the satoshis. Private keys and other information related to the transaction is stored in the wallet files. Wallet programs are required to interact with the peer-to-peer network in order to receive information from the block chain and be able to broadcast new transactions. However, programs that distribute public keys are not required to interact with the peer-to-peer network.

Full Service Wallet

A full service wallet performs the following functions: generate private keys, create and distribute the corresponding public keys, monitor the outputs of the public keys, sign and create the transactions by spending those outputs, and broadcasting the signed interactions. In a full service wallet, the private keys are stored in a device which is connected to the internet. To protect it against the attackers, wallet programs encrypt wallet files which consist of the private keys.

Signing-Only Wallets

In signing-only wallets, private keys are stored in a more secure environment by a separate wallet program and work with a networked wallet in a peer-to-peer network. Signing-only wallets use deterministic key creation to create private and public keys. When the program runs, the wallet creates a parent private key and gives the networked wallet the corresponding parent public key. The signed transactions are broadcasted to the peer-to-peer network by the networked wallet. The two main variants of signing-only wallets are offline wallets, not connected to a network and the user is responsible for handling the data transfer, and hardware wallets which are the opposite, being a safe non-manual communication.

Distributing-Only Wallets

Distributing-only wallets are programs that run in “difficult to secure environments”. There are two ways to design these wallets. First, you can pre-populate the database. This can be done by populating multiple public keys or addresses and distributing them on the public key script or address. Second, child public keys are created by the parent public key.

Loose-Key Wallets

Loose-key wallets, also known as “Just a Bunch Of Keys (JBOK)”, is a deprecated form of a wallet which originated from the Bitcoin Core client wallet. The bitcoin core wallet generates multiple private/public key pairs automatically through Pseudo-Random-Number Generator (PRNG). The virtual “key pool” is used to store unused private keys and while new keys are being generated and the old keys are being used.

Contracts

Contracts are transactions which use the decentralized Bitcoin system to enforce financial agreements. One way to make sure a deal is carried out is by Escrow and arbitration. In Escrow, a neutral third party—an arbitrator—creates an "escrow contract" to make sure that all parties are fulfilling their responsibilities. The arbitrator creates a contract to keep the money until the terms of the contract are met. Arbitration allows for the resolution of conflicts without the need for court. The arbitrator considers the arguments from all sides before coming up with a verdict. The micropayment channel in Bitcoin Core is another helpful component where two people can communicate frequently and send little amounts of money without having to pay a significant transaction cost. This works by creating a bitcoin transaction that locks a specified quantity into a multi-signature and then the two parties can create a channel to send money to each other which automatically updates the amount whenever they decide to send a payment. One final transaction, showing the entire amount of bitcoin sent between them, can be broadcast to the network to close the channel. The final method for enhancing the privacy of Bitcoin Core transactions is Coinjoin. This works by mixing transactions from various users, so it is more challenging to trace back the user by looking at the public blockchain. This enables Bitcoin Core to offer its users a high level of privacy and security.

## *4.0 Use Cases*

The following are two sequence diagrams of decentralized and verified transactions for Cryptonite e-money.

### 4.1 Use Case 1: Decentralized Payments (Peer-to-peer network)

Decentralized Payments: The peer-to-peer system of Bitcoin Core can be used to support decentralized payments between individuals or businesses, removing the need for intermediaries like banks or payment systems.

The user manager sends a search Query to GRouter, which simply means that online consumers browse the merchant's search page, select the corresponding item, place an order and complete a transaction. GRouter recognizes the incoming message as a multicast message and propagates it to the specified network, where the link redirects to a secure payment server. When replies arrive, a standard server passes them to the user manager component, which presents them to the user. GRouter selects the content of the experience database and queries it using service identifiers. The reputation Manager uses the records extracted from the experience database to determine whether it trusts the service. That is, the relevant bank node checks the online consumer's ability to pay bills, etc., and sends the results back to the third party and the online consumer. After selecting a set of reliable servers, GRouter broadcasts a vote message, and the responses to the polling request are forwarded to the reputation manager, meaning that upon receiving a payment notification, the merchant ships or provides the service to the online consumer, and each bank clears with the merchant through the third-party payment platform.

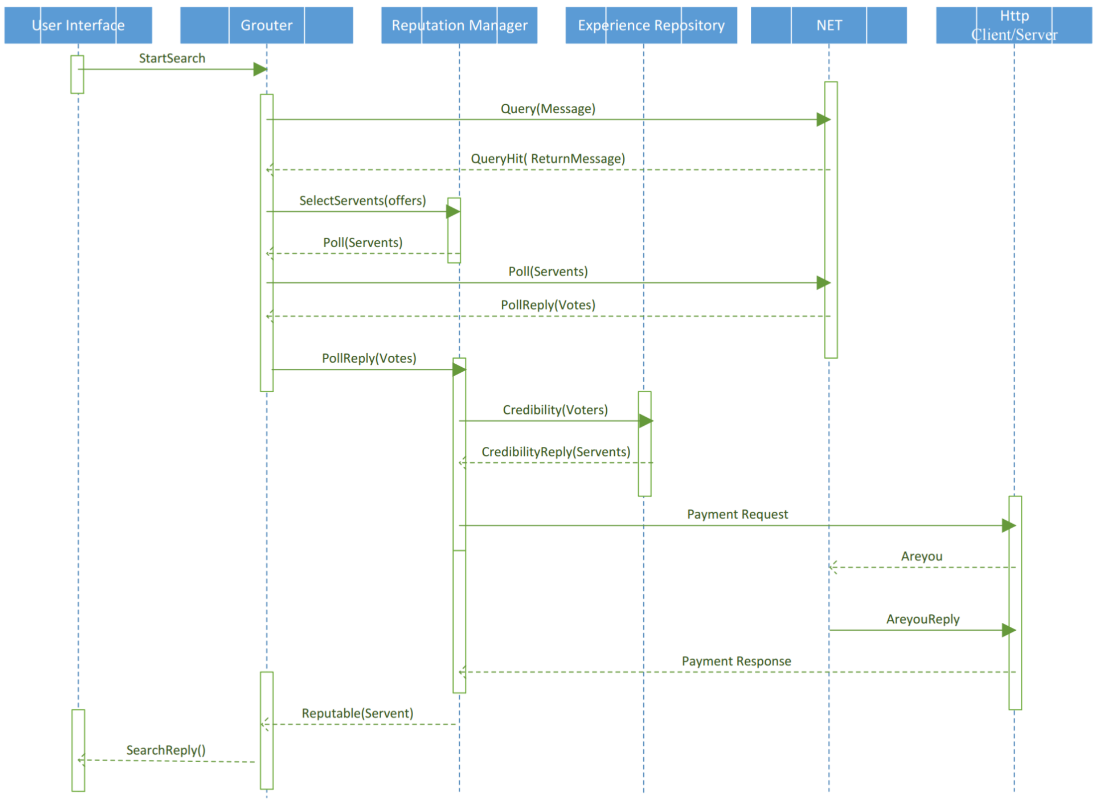

*Figure 1: Sequence Diagram of Use Case #1: Decentralized Payment*

## 4.2 Use Case 2: Verifying transactions

Verifying transactions: Bitcoin Core downloads and maintains a complete copy of the blockchain, the public ledger that records all Bitcoin transactions. Users of Bitcoin Core may make sure their transactions are legitimate and the network is secure by validating network transactions.

The peer-to-peer network payment validation step requires that the component pass a direct connection verification vote, which is a third-party verification of user information. The actual service is mainly user-related parameters. When validation is complete, the resource locator can be associated with each service, coordinating positive votes and untrusted negative votes, where a third party verifies the user's credit and other information and selects the best service to download through the HTTP client/server. Once the download is complete, users can vote on the resource, with the aim of updating the experience database and appropriately changing the reputation of the service providing the resource and the person voting on it. On the receiving end, the incoming query is extracted by GRouter and sent to Resource Locator, where Resource Locator is responsible for checking the file repository for eligible files. The information on these files is sent to GRouter and returned by QueryHit messages. When the corresponding poll reply arrives, it is sent to the reputation manager, which checks to see if there is a record of the service listed in the message. If this is the case, it passes encrypted information about them to GRouter. The latter then sends a PollHit message. The IP authentication message is sent directly to the HTTP client/server component, which responds correctly.

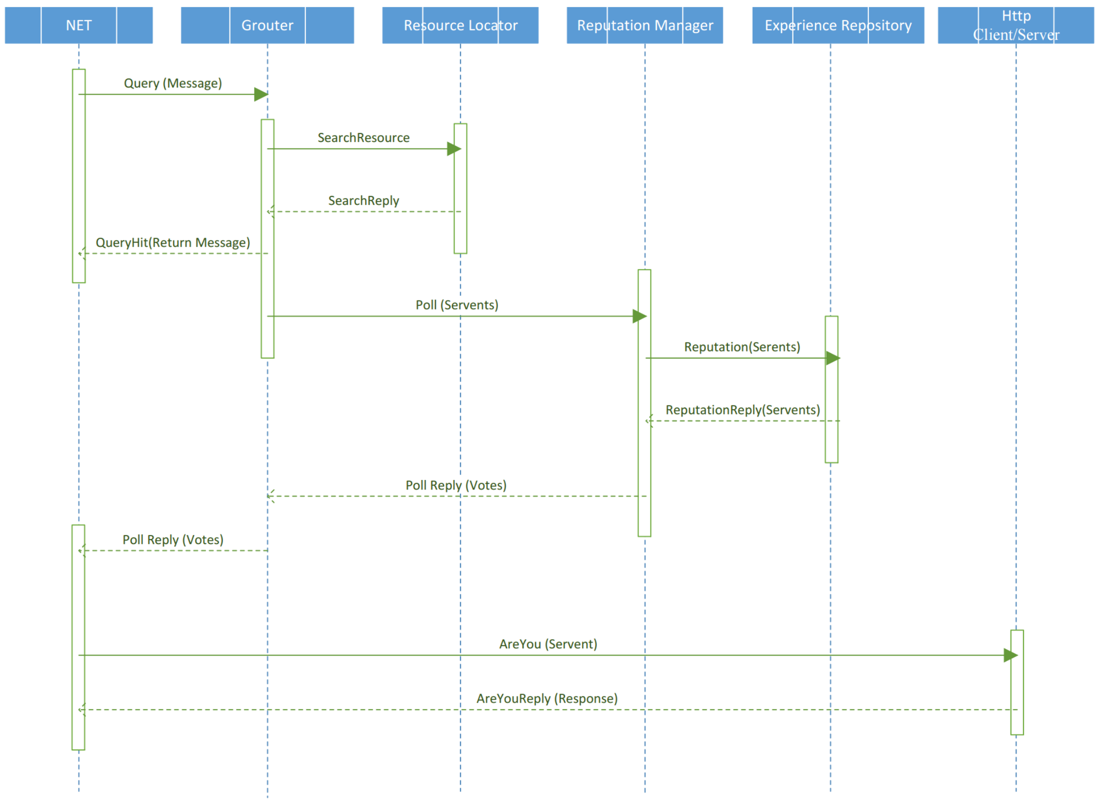

Figure 2: Sequence Diagram of Use case #2: Verifying Transactions

## *5.0 External Interfaces*

The external interfaces perform the role of transmitting or receiving information, There are a couple of components that Bitcoin Core's system interacts with: RPC, LevelDB, and Bitcoin Core's Network.

1) RPC
   Bitcoin Core's RPC (Remote Procedure Call) is a method that executes various commands on remote servers. RPC allows a program to request a service from another program that is remotely located on a different network in a client-server type of manner. It does not require the network's details in order to access it. In Bitcore's context, the RPC interface allows Bitcoin Core to be controlled by other programs, in manners such as reading data, spending wallet funds, alongside various operations. One of the main functions for an RPC is to allow transactions to be sent to several networks, while allowing users to read data.

2) LevelDB
   Bitcoin Core's need for a database is covered by LevelDB, a Google provided library that stores an ordered mapping of key-value pairs. The throughput for the read and write are both enhanced with the use of LevelDB. For each transaction that is made, its id is stored as the key in a key-val pair. As a database, LevelDB outperforms several other databases in terms of various operations such as reading and writing. In terms of concurrency, Bitcoin Core follows the one process at a time rule when sending information to LevelDB, where in an instance of multiple writers, the first writer gets to write while the other processes get blocked from writing to the database. This allows LevelDB to successfully save the index of the transaction.

3) Bitcoin Network
   The Bitcoin peer-to-peer protocol has tons of nodes running. These nodes need to interact with each other through networks, and they most commonly communicate through the internet with TCP and IP connections, or even through onion servers. The network is used to transmit various transactions and blocks.

## *6.0 Data Dictionary*

*Bitcoin's P2P Network* - The Bitcoin network protocol allows peers to collaboratively maintain a peer-to-peer network for block and transaction exchange.

*Digital Wallet* - A Bitcoin wallet, refers to either a wallet program or a wallet file.

*Blockchain* - Bitcoin's public ledger, which is an ordered and timestamped record of transactions.

*Mining* - The miner generates new blocks on his own, with the proceeds from the block reward and transaction fees going to himself.

*Contracts* - Transactions that use the decentralized Bitcoin system to enforce financial agreements.

*Consensus* - A shared agreement (called consensus) that allows people to only accept valid bitcoins, enforcing Bitcoin's rules against even the most powerful miners.

## *7.0 Naming Conventions*

User Interface: The User Interface, which also includes User Manager.

Grouter: Router that can send multicast messages.

Reputation Manager: Judging bank user credit, etc.

Experience Repository: A database of user payment credit records, etc.

NET: specified network, transfer data to third-party nodes.

Http: the client.

## *8.0 Conclusion*

In conclusion, our group's report summarizes Bitcoins Core's conceptual architecture and the fundamental components used throughouts its system. Bitcoin Core utilizes a peer-to-peer network as its software architecture, mainly to enable the base functionality of direct online payments between parties without the use of financial intermediaries. The Bitcoin Core software is mainly based around two use cases, decentralized payments which are focused between individuals or businesses for an easier way to broadband users together and verifying the transactions between them to ensure every block and transaction is valid, preventing miners and banks from taking control of Bitcoin. Other components of Bitcoin Core are also split into smaller subsystems that include blockchain, mining, wallets, consensus, mechanisms, and contracts. While still being a large open source developer project, Bitcoin Core provides a service that is privately secure while being one of the most popular Bitcoin software’s in the bitcoin network. Looking forward, Bitcoin Core may not need any major changes or heavy overhauls to its architecture, as it has already implemented its core functionality of decentralized payments with full validation on the blockchain. That being said, there will always be a possibility for loss as Bitcoin softwares in general is still an experimental system and buying bitcoins remains a risky investment. As a suggestion for future features, adding support for a commercial hardware wallet can help increase safety and risk towards possible bitcoin loss.

## *9.0 Lessons Learned*

There were a lot of lessons learnt over the course of this group assignment. One of the biggest lessons we learned was to communicate often and effectively. As we were all doing our own part, it was a bit difficult to maintain the flow of the assignment, resulting in us needing to make some minor modifications for several parts. A solution to this would be to have more group meetings to go over any new updates or help each other in clearing any doubts. Another lesson learned was to have a better understanding of Bitcoin Core’s architecture, and just software architecture in general. While working on parts such as external interfaces, there was trouble at the beginning on understanding what components involved with the architecture would be defined as an external interface. Looking back at how our group divided the assignment, it is reasonable to say that we could have divided the work more equally, so some group members wouldn’t have to work on an important part of the assignment alone. In terms of time management, we learned that we needed to be better paced with our work, as it has been difficult sometimes to get all the work in as soon as possible. This can be fixed as well with more group meetings and creating more deadlines.

## ***10.0 References***

Antonopoulos, A. M. (n.d.). *Introduction*. Introduction · GitBook. Retrieved February 19, 2023, from https://cypherpunks-core.github.io/bitcoinbook/

Frankenfield, J. (2022, November 23). *What is bitcoin? how to mine, buy, and use it*. Investopedia. Retrieved February 19, 2023, from https://www.investopedia.com/terms/b/bitcoin.asp

*Helping you keep Bitcoin decentralized*. Bitcoin Core. (2016, October 26). Retrieved February 19, 2023, from https://bitcoin.org/en/bitcoin-core/

*What is Bitcoin Core?: River Learn - Bitcoin Technology*. River Financial. (2021). Retrieved February 19, 2023, from https://river.com/learn/what-is-bitcoin-core/

*P2P Network.* Retrieved February 19, 2023, from https://developer.bitcoin.org/devguide/p2p_network.html

*Sequence Diagram Tutorial – Complete Guide with Examples* (12 December 2022). Retrieved February 19, 2023, from https://creately.com/guides/sequence-diagram-tutorial/