Spring 5-6-2023

Blockchain Security: Double-Spending Attack and Prevention

William Henry Scott III
Stephen F. Austin State University, scottwh@jacks.sfasu.edu

Blockchain Security: Double-Spending Attack and Prevention

Creative Commons License

Blockchain Security: Double-Spending Attack and Prevention

By

WILLIAM HENRY SCOTT III, B.S.

Presented to the Faculty of the Graduate School of

Stephen F. Austin State University

In Partial Fulfillment

of the Requirements

For the Degree of

Master of Science in Cyber Security

STEPHEN F. AUSTIN STATE UNIVERSITY
May 2023

Blockchain Security: Double-Spending Attack and Prevention

By

WILLIAM HENRY SCOTT III, B.S.

APPROVED:

_____
Pushkar Ogale, Ph.D., Thesis Director

_____
Christopher Ivancic, Ph.D., Committee Member

_____
Jianjun Zheng, Ph.D., Committee Member

_____
Jonathan Mitchell, Ph.D., Committee Member

_____
Sheryll Jerez, Ph.D.
Interim Dean of Research and Graduate Studies

ABSTRACT

This thesis shows that distributed consensus systems based on proof of work are vulnerable to hashrate-based double-spending attacks due to abuse of majority rule. Through building a private fork of Litecoin and executing a double-spending attack this thesis examines the mechanics and principles behind the attack. This thesis also conducts a survey of preventative measures used to deter double-spending attacks, concluding that a decentralized peer-to-peer network using proof of work is best protected by the addition of an observer system whether internal or external.

ACKNOWLEDGMENTS

First and foremost, I would like to thank my lord and savior Jesus Christ for my intellect that I did not come by honestly. Without Him, I would be lost in darkness. I would like to thank Dr. Pushkar Ogale for his tireless belief in me that I could in fact finish this thesis and for teaching me how to administer my workflow and my time. A special thanks to Dr. Christopher Ivancic and Dr. Dae Glendowne for helping me build my foundation in both computer science and cyber security. Thanks to Dr. Robert Strader, who in my formative academic beginnings whipped me into shape. A very special thanks to Tri Nguyen-Pham of Raptoreum, for without all those late-night conversations none of my early research and experimentation would have been possible or successful. Thank you to my thesis committee, and to Larin Adams, for help editing. Thank you to all the Computer Science faculty, Mathematics faculty, and the AARC personnel at SFASU that poured into me over the years. A special thanks to my family, particularly my mother (I'm finally joining you!), father, and wife who were my greatest supporters throughout this monumental endeavor. Lastly, thank you to anyone who has ever been kind to or patient with me despite their own circumstances. I can only hope to aspire to such conduct.

TABLE OF CONTENTS

TABLE OF FIGURES

I. INTRODUCTION

The first three chapters of this thesis are a minimal introduction to general proof of work cryptocurrencies. Any readers already familiar with cryptocurrencies may wish to skip to *Chapter IV* where the experiment is discussed. Small-scale distributed consensus systems refer to proof of work cryptocurrencies which have a sufficiently small amount of hashrate on their networks. A double-spending attack is both cheaper to execute and more likely to succeed in smaller networks. This is mostly owing to majority rule which most proof of work cryptocurrencies rely on through an incentivized service known as mining.

This thesis built and deployed a private fork of a proof of work cryptocurrency called Litecoin. Through two experiments, this thesis successfully executed a hashrate-based double spending attack. The first experiment had limited results and the second experiment, made more sophisticated after the limited success of the first, was fully successful. An honest transaction was invalidated by inducing a block reorganization event in which the attacker's blockchain fork was adopted by the honest network. This blockchain fork contained a transaction that spent the same inputs as the honest transaction it invalidated.

This method of double-spending attack is vulnerable to several preventative measures, chief among them being an observer network. Whether through using a layer 1

masternode network like Dash as an internal observer or an external observer like the Komodo-Pirate integration, this thesis' research concludes that observing and reporting with a system that does not affect scalability, decentralization, security, or trustless transactions is the best protection against double-spending.

A. Literature Review

This thesis' purpose is to demonstrate a double spending attack in order to provide insight into preventative measures against this type of attack. This thesis found that current literature on cryptocurrency lacked a practical experiment executing a double spending attack and, furthermore, most preventative measures were either theoretical or did not explicitly address how the technical execution of the attack was being prevented.

This thesis found the original Bitcoin whitepaper by Satoshi Nakamoto [2] to be the most useful. After surveying the documentation and source code of numerous proof of work cryptocurrencies, it was revealed that most are software forks of Bitcoin, thusly [2] was relevant to almost every proof of work cryptocurrency that was examined. Additionally, the concepts and theories found in [2] form the foundation of most other cryptocurrencies today, proof of work or otherwise.

Following the foundational theory of cryptocurrencies is networking information. Information propagation, network resiliency, liveness, and security were discussed most clearly by [4] and [29]. As this method of double-spending attack relies heavily on producing a competing blockchain fork, understanding network information dissemination was critical and Decker & Wattenhofer in [4] did an outstanding job of

both explaining and illustrating networking within a peer to peer network. Also, Park, Im, Seol, and Paek in [29] provided nuanced insight into the technical working of Bitcoin's peer-to-peer network, especially as it pertained to macro analysis of data propagation across the network.

Lastly is the double-spending attack and its prevention or deterrence. Rosenfeld [6] adequately described a hashrate-based double-spending attack as well as provided some theoretical analysis of success probability. This work, as previously stated, lacked the literal execution of the attack and relied only on theoretical analysis. While an understanding of the attack may be attained using theory, this thesis found that a practically executed attack on a purpose-built peer to peer network has led to insights that are simply not available in theory alone.

Regarding preventative measures, works like [31] by Duffield, Schinzel, and Gutierrez provide excellent detail regarding various cryptocurrencies' security and scaling features and how they provide protection against double-spending attacks. However, the weakness of works like [31], for purposes of this thesis, is that they didn't provide a wider perspective. It was clear the authors understood their solution prevented or at least deterred double spending attacks, but a thorough and clear explanation of a double spending attack and its effects were not discussed.

B. Cryptography

The word "cryptocurrency" is derived from two words: *cryptography* and *currency*. The word *currency* is used because for users who participate in each cryptocurrency

ecosystem, they use the cryptocurrency as a financial instrument (transferring goods and services, speculative investing, value storage, etc.) according to [2]. The word *cryptography* is used because it is the foundation that enables some of the major ideals that cryptocurrency is meant to embody such as decentralization, security, and privacy.

C. Wallets and Addresses

When a prospective user of cryptocurrency wants to be a participant within the peer-to-peer network, they will typically download software known as a wallet. Typically, a wallet serves as an interface for the user to interact with the cryptocurrency's blockchain and peers (other users of the same cryptocurrency) according to [34]. Typical wallet functions include showing balances, displaying addresses (unique identifiers from which users can either send or receive funds), listing a full history of transactions for the wallet across all addresses, generating new addresses, sending funds, and receiving funds. Some cryptocurrencies such as Monero [39] or Dero [49] have anonymity as a core function, so their wallets provide some additional privacy features. Typically, when a user initializes their wallet (whether running the software wallet for the first time or reinitializing due to some other circumstance) the software first generates a private key. Various cryptocurrencies use different generation methods for private keys. A popular method is using a random seed generator [40] to produce a seed that is then used as input into data-transforming functions that introduce more randomness until enough entropy has been introduced. To produce a public key from that private key, elliptical curve cryptography is commonly used [41]. From the public key, addresses are created,

typically through using a cryptographic hashing function or multiple cryptographic hashing functions to create an output using the public key as an input [11]. Addresses can be thought of as bank accounts, though anonymous and not linked to any of the owner's identity according to [4].

D. Signatures

To discuss what an electronic coin is, digital signatures must first be understood. According to [2], cryptocurrency is meant to be a distributed peer-to-peer version of electronic cash. As there is no central authority, this model presents some unique difficulties,one of which being trustless transactions from peer-to-peer. Multiple related problems such as privacy, immutability, and authenticity also need to be addressed for the peer-to-peer model to be practical. Digital signatures solve these issues, and there is not a standard implementation across cryptocurrencies though a similar model is used. The sender, who has the private key of the cryptocurrency address that holds the funds that are going to be sent, will sign the transaction with their private key. The receiver can then check the history of signatures related to the funds in the sender's address to ensure provenance of the funds. This history, or chain, of signatures should be visible to when the funds were first minted. These signatures, in tandem with the blockchain and other features show a clear history of ownership that indicates to the recipient that the sender does in fact have funds to send.

E. Electronic Coin

Nakamoto [2] describes an electronic coin as "a chain of digital signatures. Each owner transfers the coins to the next by digitally signing a hash of the previous transaction and the public key of the next owner and adding these to the end of the coin." A hash is a form of data transformation [42]. Upon supplying a hashing function or algorithm with an input it will yield a unique output. In the case of all cryptocurrencies, the hashing algorithms used (which greatly vary amongst cryptocurrencies) are non-reversible and probabilistically give unique outputs for any given input. Coin distribution varies from cryptocurrency to cryptocurrency. Some cryptocurrency projects start like an IPO (Initial Public Offering), with an initial offering of coins for a predetermined price per coin [43]. Most proof of work cryptocurrencies distribute by minting a set number of coins for each block produced until a maximum supply is reached. The process of minting is, effectively, creating units of cryptocurrency from nothing. The coins produced from minting exist because the cryptocurrency software is coded to produce them [44] and there is a valid chain of signatures starting with the special transaction associated with the act of minting.

Langridge [28] states that tokenomics is "the topic of understanding the supply and demand characteristics of cryptocurrencies". It is a term that has been popularized within the realm of cryptocurrencies and describes several key attributes of any given cryptocurrency project such as maximum supply of coins, distribution rate, distribution schedule, burn rate, and block time. The term tokenomics is a combination of token and

economics. A token is a type of asset that can exist within a cryptocurrency, or it can be its own cryptocurrency. On websites such as bitcointalk.org, cryptocurrency owners/developers will announce the creation and launch of their projects. With these announcements, owners typically publish certain economic attributes such as the total number of coins they intend to mint, how they intend to distribute these coins and many others (typically described as supply and demand markers) [45]. The reason is to convince the audience that their project is worth investing in, mining, using, or talking about. While there is no official attributing for the origin of the term in the context of cryptocurrency, tokenomics is popular and well understood according to [47] and [48].

F. Transaction

A transaction is when one owner of coins sends funds to a recipient. The recipient can be a vendor, a friend, or even the owner themselves. Transactions form most of the data contained within a blockchain and are the primary means (and often only) by which users and the network itself transfer ownership of coins. According to [1], a Bitcoin block transaction data will contain 1000 times more data than the header making the ratio of block data 99.9%. Technical implementation of transactions varies surprisingly across the cryptocurrency landscape. Such implementation differences include chosen digital signing algorithm, transaction initiation, and transaction data obfuscation.

Transactions typically are made of several different parts including inputs, outputs, sender, recipient, transaction hash, total amount sent, transaction fee, block identification, and time sent [7]. Of important note are input and outputs of a transaction. The single

unit of any cryptocurrency is divisible, and the lowest denomination is dependent on the creators of the cryptocurrency. When a coin is initially minted, there are no user inputs as it is being created by the cryptocurrency software. However, each time its owner transfers ownership of the coins, it can either be split or combined. To handle this complication (since every unit of every coin is traceable back to its minting) transactions record all their inputs. Addresses can contain many transactions, and any user can own many addresses. For example, consider a user that wants to send 10 coins and the user has a total of 20 coins among a total of 5 addresses.

- Address A: 1 coin
    - Transaction A1: 1 coin
- Address B: 2 coins
    - Transaction B1: 0.5 coins
    - Transaction B2: 1.5 coins
- Address C: 9 coins
    - Transaction C1: 2 coins
    - Transaction C2: 5 coins
    - Transaction C3: 1 coin
    - Transaction C4: 1 coin
- Address D: 5 coins
    - Transaction D1: 5 coins

- Address E: 3 coins
    - Transaction E1: 2 coins
    - Transaction E2: 1 coin

When the user authorizes and submits the transaction to send 10 coins to another user, the wallet software cannot send the funds from a single address, much less from a single input. Wallet software tends to pick the neatest combination of inputs from however many addresses it needs to fulfill the transaction amount. The wallet software might pick the following transactions as inputs (see Fig. 1):

- Transaction B1: 0.5 coins
- Transaction C2: 5 coins
- Transaction D1: 5 coins

Assuming there is a transaction fee, these three inputs would fulfill the required amount of 10 coins and the transaction fee. Then, there would be two outputs:

- Output1: 10 coins to the recipient
- Output2: 0.5 coins to the network as a transaction fee

What is done with the transaction fee depends on the cryptocurrency. If the cryptocurrency is mineable, then the transaction fee will likely be paid in part or in total as a block reward. In some projects, the transaction fee is burned or destroyed; and in other projects the transaction fees are still paid to the developers or owners of the cryptocurrency.

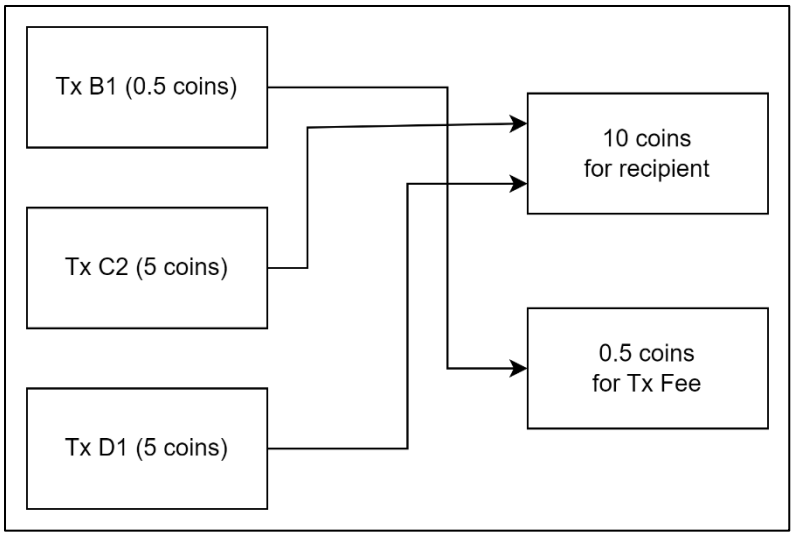

Fig. 1. Inputs from previous transactions used to create outputs for a new transaction.

The discussion of inputs will be important later as the premise of a double-spending attack occurs when an attacker is permitted to transact the same input(s) more than once. In the above example, the coin's signature history changes when the inputs are used in the transaction. Even if the user gets the same coins back from the recipient, the signature history will have two new entries. As such, at each save state of an electronic coin, it should only be able to be transacted once before having its save state updated with a new signature. A double spending attack happens when, at the same save state, the owner can transact the input more than once.

G. Blocks and Blockchains

A block, as illustrated in Fig. 2, is a segmented data structure containing both a header and a collection of transactions.

Fig. 2. Generic cryptocurrency block with header and transaction section.

The header contains various information such as a unique hash of all its transactions' hashes, which is known as the block's hash. Blocks are connected, or chained, to one another by including a hash of the previous block's hash. The block's hash provides immutability to the transactions contained in its collection. The block header containing the block of the previous hash provides immutability to the ancestry of the blockchain. Blockchains are the last data encapsulation for transactions in many cryptocurrencies. Blockchains are comparable to traditional linked-list data structures that contain blocks. In many cryptocurrencies, the blockchain is stored locally in a relational database that is accessed by the cryptocurrency software. This thesis will now provide a foundation for cryptocurrency network assets and communications.

II. CRYPTOCURRENCY NETWORK

Cryptocurrency networks are transaction tracking systems designed as a protocol and implemented as individual instances of software that, together, act as peer-to-peer networks that perform data storage, data confirmation, and data transmission according to [5].

A. Network Types

All cryptocurrencies are implemented as collections of network-connected, distributed assets. Some peers provide structure, some provide functionality, some simply use, and others provide data [3].

- **Structure**: Some cryptocurrencies such as Graph (ticker GRT) have network assets that route traffic and queries that are specific to the niche functionality of the cryptocurrency.
- **Service**: Most if not all cryptocurrencies have network assets that have full copies of the blockchain and act as a part of the network-wide consensus mechanism. Cryptocurrencies that use proof of work will have miners on the network whose sole operation is to solve hash functions.
- **User**: This type of network asset would be anything from a user with a single wallet to an online cryptocurrency exchange to an explorer or data aggregator.

- **Data**: Most cryptocurrencies have network assets that act as DNS (domain name system) [22] seeders for new clients connecting or returning clients reconnecting to the network.
    - DNS seeders are programs or functions within a program that act as DNS servers, crawling the internet looking for healthy peers. Health is defined by certain metrics such as peer connections both ingoing and outgoing and uptime.
- **Specialized**: Many cryptocurrencies offer special services such as VPN (Virtual Private Network) or data storage. Other implementations would be cryptocurrencies that use masternodes in governance, specialized operations such as smart contract execution, or as an additional layer of security for the network. Masternodes are peers that have a required amount of funds that are used as collateral to identify the peer as a masternode. The collateral is to discourage users from acting as masternodes in bad faith since the collateral is typically non-insignificant. These masternodes typically provide extra services beyond mining as previously mentioned and are incentivized because they are usually rewarded.

In summary, this thesis is concerned primarily with proof of work cryptocurrencies such as Litecoin, Bitcoin, Raptoreum and Dash. These cryptocurrencies, and many others, have 3 main types of network assets with Raptoreum and Dash both having masternodes as well. (Masternodes are instances of full nodes that, after meeting certain hardware and collateral requirements, can become incentivized by providing services to

the rest of the network.) They have miners, full nodes, and clients. The full nodes have a full copy of the blockchain stored locally, and they manage the network consensus that determines which blocks are in the blockchain and which order the transactions are recorded in. Miner nodes also have full, local copies of the blockchain, but their job is not aiding in consensus but solving blocks. Clients would be a user's wallet or a cryptocurrency exchange which don't need a full, locally stored copy of the blockchain.

B. Network Communications

Most cryptocurrencies encrypt all network communication using popular encryption methods such as TLS (Transport Layer Security) described in [21]. While many cryptocurrencies have modernized and highly efficient network communications, most follow the Bitcoin model laid forth by [2]:

1) New transactions are broadcast to all nodes.
2) Each mining node collects new transactions into a block.
3) Each mining node works on [solving or affirming] the block [based on its means of proof]
4) When a mining node finds a solution, it broadcasts the block to all nodes.
5) Nodes accept the block only if all transactions in it are valid and not already spent.
6) Nodes express their acceptance of the block by working on creating the next block in the chain, using the hash of the accepted block as the previous hash.

A transaction is broadcast to as many nodes as possible, and each mining node that receives knowledge of this new transaction will try to collect it into a new block. Then each mining node broadcasts their separate, individually produced blocks with the same new transaction to nodes on the network requesting affirmation. Whichever mining node gets a correct solution first and submits their block to the network first succeeds and all other new blocks of the same height that are being processed by other miners, regardless of whether they have been solved or not, will be rejected.

C. Event Broadcasting

The term "event broadcasting" generally means any message transmitted between peers. All cryptocurrencies broadcast at least two types of events: transaction and block creation. There are others as well, such as block reorganization, blockchain forks, and in some cases governance votes. [29] describes Bitcoin's broadcast algorithm as *recursive flooding*. [4] describe Bitcoin's broadcast algorithm as *randomized rumor spreading*. A peer cannot, by virtue of the software, broadcast to all other peers on the network at once. By default, a Bitcoin peer can only have 125 peer connections between incoming and outgoing connections. This default is hard coded in *bitcoin/src/net.h, line 77* in [33]. What this means is any given peer is connected to a subset of online and available peers and so data transmissions must make hops to propagate throughout the entire network. Hence the propagation technique [29] used, which describes valid messages being recursively broadcast to each successive peer's connected subset of the entire peer-to-peer network until all nodes are aware of the message.

Decker and Wattenhofer in [4] state that peers don't simply transmit all data at once. There is a sequence of communications between peers when broadcasting events as illustrated in Fig. 3 below replicated from [4]. The transmitting peer will send basic identifying information about the event such as a block hash or transaction hash and a timestamp. The receiving peer will check its local copy of the blockchain to determine whether it has this data. If it does, the event will simply be ignored. If the receiving peer does need the data, then it will send a response communication to the transmitting peer to send the full data. These functions can be found in *bitcoin/src/net_processing.cpp* in the Bitcoin source code. According to [13], the *inventory* function broadcasts the transmitting peer's data, one of the *getdata* functions (based upon data in the *inventory* function) such as *gettransaction* or *getblock* will respond, and then the transmitting peer will send the requested data to the receiving peer.

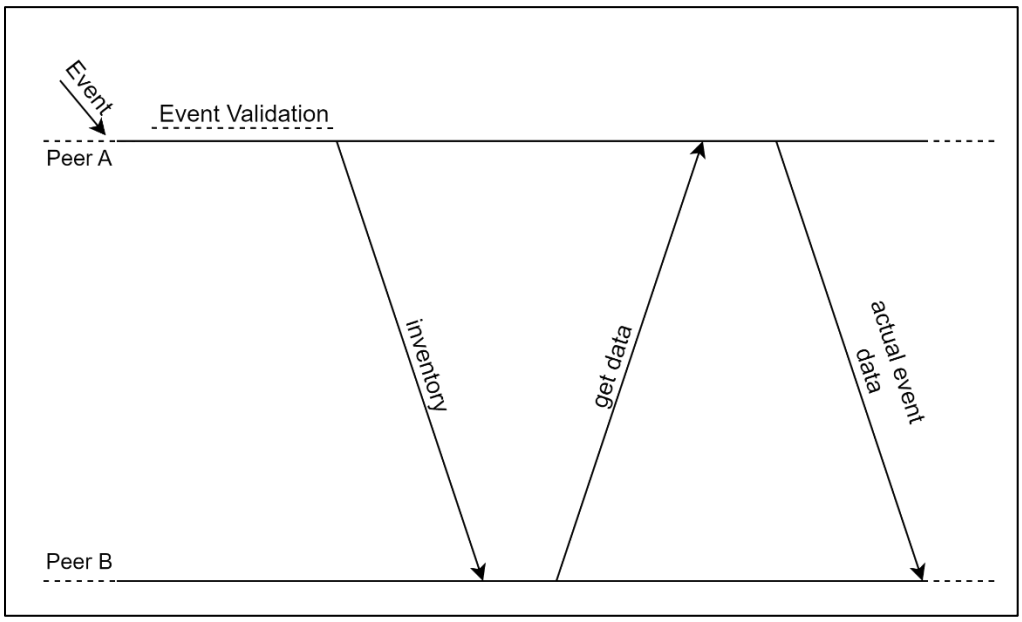

Fig. 3. Data propagation sequence from peer to peer as seen in [4].

D. Synchronization

Cryptocurrencies are a protocol describing data and a peer-to-peer network which generates and shares that data. As such, synchronization does not start with the peer-to-peer network, and its delayed propagation issues and replication inconsistencies, but instead with the data itself. Transactions are the basis for all synchronization within cryptocurrencies, as they are the smallest and most atomic unit of data and the vehicle by which users can transmit value within the blockchain. Transactions synchronize ownership of coins in two ways: who and in what order. Transactions also contain data concerning the splitting of value into outputs that will in turn be used as inputs in later transactions.

Next in the synchronization ladder there are blocks, which organize and encapsulate groups of transactions. Blocks help in synchronization by grouping transactions, being witnessed by, and propagated through peers on the network and ultimately provide a detailed chronological timeline when they are added to the blockchain. The matter of synchronization is an inherently difficult one with cryptocurrencies given their replicated and distributed nature. [35] showed that the higher the median block propagation time, the more likely blocks are to be orphaned. Orphaned blocks are those valid blocks that are produced and added to the blockchain to then be discarded in favor of another block. Each peer has either a full or partial copy of the blockchain that includes transactions, blocks, the blockchain itself, and any other data that exists within it. Disputes arise naturally concerning validity of transactions, blocks and even forks of the blockchains.

Disputes, both natural and engineered (maliciously or otherwise), are resolved through consensus.

E. Network Consensus

There is a fine yet necessary line that separates synchronization from consensus. Both synchronization and consensus have a networking aspect to their definitions and in practice, both deal with data propagation, and both are concerned with data validity. However, the sliver of difference between the two is *consistency*. Synchronization is concerned with consistency between all copies of the blockchain. Cryptocurrencies strive to, by design, maintain a singular history of the blockchain that all users of the network have. The peer-to-peer network strives to propagate all valid transactions and blocks to all peers on the network as quickly as possible and to disregard invalid transactions and blocks. Consensus on the other hand, is almost singularly concerned with which history of the blockchain is the most valid and most correct.

Due to the nature of distributed systems, data propagation through a peer-to-peer network, and the ever-present amount of entropy related to an unlimited number of users causing an unlimited amount of data to be shared across the network, there exists the possibility of alternate blockchains on the same network. These alternate blockchains are called forks whenever there is more than one on the network, and at any given time it is possible to have to multiple existing in any cryptocurrency that uses proof of work as a consensus mechanism. Blockchain forks and how they are resolved will be covered later in this thesis at length, as this mechanism is what is abused to execute a double-spending

attack. For now, it is sufficient to know that in addition to every user on the network having access to the same data, there is also a separate need to make sure the same history of data is being maintained as the correct blockchain.

F. Public Record

One of the most intentional design features of Bitcoin from [2] was that the ledger be published and freely available to all who want it for whatever reason. In cryptocurrency, ledger is a synonym for blockchain. A ledger in the classic sense is a list of financial transactions that either debit or credit an amount of money, usually recorded in a book. While the idea of identity is abstracted away with private keys, public keys, and the addresses they generate to represent an owner in cryptocurrency, the ledger in which that owner transacts is public. Any entity at any time, whether they be a government or individual, user or interested third party, can validate transactions, blocks, blockchain forks, the peer-to-peer network, and many other interesting and relevant characteristics of the blockchain. This is a major component of cryptocurrencies that makes it a trustless system. This thesis will now discuss cryptocurrency mining, as well as discuss expected consequences that arise from distributed implementations of mining.

III. MINING

A. Hashes

A cryptographic hash, also known as a signature, is a non-reversible output for a given input. For many proof of work cryptocurrencies, primarily transactions and blocks, are hashed for proof of immutability. In some cryptocurrencies, hashes may also be produced for specific message signing. In the context of mining, only the block hashes are of importance. There are several generally used inputs used when calculating a block's hash [9]:

1) Information about the proposed block such as:
    a. Timestamp
    b. Header information
    c. Transaction information
    d. Software version information
2) Previous block's hash
3) A nonce
    a. The nonce, known also as a "number used only once," is the only variable input.

Information about the proposed block is to ensure immutability to the timestamp and all data contained within the block. The previous block's hash is to preserve the

chronological order of blocks and their contained transactions being submitted to and accepted by the network.

B. Difficulty

Difficulty is a term used to express how computationally expensive it is to discover a block's hash. In most cryptocurrency software, difficulty is dynamically determined by the software based on a few factors such as block timing, current total network hashrate and average network hashrate [14]. According to [2], difficulty is adjusted to compensate for changes in hashrate over time. Block time is the average expected frequency of block's hashes to be solved and then added to the blockchain. At the time of this thesis, Bitcoin's block time is 10 minutes. Block time, to the best of this thesis' research, is to provide a baseline for predicting blocks and therefore enabling the network to adjust difficulty to meet that prediction within a margin of error. Block time seems to be arbitrary, though different cryptocurrencies state different reasons for their block times. [4] speculated that it takes 12.6 seconds on average for a message to be propagated across the peer-to-peer network based on Nakamoto's [2] best effort recursive flooding propagation algorithm. For this reason, Ethereum's developers chose a block time of 12 seconds [16] before that cryptocurrency transitioned away from proof of work to another consensus mechanism called proof of stake.

In most proof of work cryptocurrencies, the difficulty is adjusted by *requiring a specified number of leading zeroes in the proposed block's hash*, known as a target [14]. The difficulty target is said to be lowering if the number of zeroes increases, and raising

if the number of zeroes decreases from the previous blocks. The difficulty target raising or lowering is describing the subset of hashes that matches the required number of zeroes. If the difficulty target is lowering, meaning the number of required leading zeroes has increased, this means the set of valid hash outputs has been lowered. The opposite is true for a raised difficulty target, since less leading zeroes allows for that many more valid hash outputs. Take for example a made-up hashing algorithm that outputs a hash of the following form:

- Length of five characters
- Numeric characters only [0-9]

The set of hash outputs for this algorithm is limited to 10^5 members, {00000, 00001, 00002,…,99997, 99998, 99999}. That's a total of 100,000 distinct hash outputs. If the difficulty target were to be lowered to two leading zeroes, then the hash output would be of the form 00###. That would mean the set of all possible hash outputs would be lowered from 10^5 to 10^3 (10^5/10^2), in effect artificially limiting the number of hash outputs that would be acceptable for the current block's difficulty. This also means that if the previous block hash and various data from the proposed block can't change and the hash inputs were limited to those invariable inputs, then the hash output would invariably be the same and may or may not fit the difficulty target. So, then a variable, called a nonce, is introduced to adjust the inputs to achieve the target difficulty. Proof of work coins tend to iterate through nonce input sets, some auto incrementing and others using algorithms to try to predict the correct input. The faster the hardware/more hardware

working, the more likely the correct nonce will be found quickly. For more information on difficulty, how it's calculated for Bitcoin and other cryptocurrencies, and how to calculate probabilities of finding a solution and timing it based on network hashrate, please refer to [14].

C. Proof of Work

According to [2] and [8], proof-of-work is a solution implemented together with a peer-to-peer network in order to protect against double-spending threats. The problem being solved is: within a distributed system that has no central authority and thus needs to self-police, how can a payee confidently release goods or services trusting that funds transmitted are irreversible and haven't already been spent elsewhere?

1) Make the network's transaction history public.
2) Use a sufficiently irreversible hashing algorithm to provide a unique signature that incorporates all transactions, their containing blocks, and the most previous block within the blockchain.
3) Force the responsibility of executing the hashing algorithm onto the entire network.
4) Incentivize the network to execute the hashing algorithm with a reward for the first peer to provide a correct output.

Every transaction should be easily verifiable, and according to [2] any user reconnecting or connecting to the network for the first time should have unquestionable access to the blockchain and all its contents from the origin block up until the most recently accepted

block. Additionally, block contents need to be trustless and irreversible. That is, any change to an amount, a sender address, a transaction time stamp, or any other information contained in the blockchain should cause an easily detectable invalidity of information propagated from peer to peer. Consider a blockchain of length 100. If a change was made to block #2, then blocks 3-100 would have to be rehashed. This is due to the provenance provided by incorporating the hash of the previous block into the next block's hash.

To account for lack of central authority, the responsibility to perform execution of this hashing algorithm is forced onto the network itself. The peer-to-peer network itself must group transactions into blocks (make blocks), and then submit these proposed blocks to the rest of the network. The first miner to both find the correct hash for that block and submit the block to the network is the miner that will win the incentive, known as a block reward. Block rewards will be discussed at length in a few sections.

Proof of work, as a concept, is self-policing because it provides a way for the majority to determine what is most valid and most correct. Ensuring that the majority is comprised of peers acting in good faith is why proof of work is incentivized. The majority is empowered to determine what data is most correct via hash power. The network has a total hashing power of 100%. Statistically speaking, the portion of the network that controls the majority of the hashing power will find solutions to blocks more quickly than the rest of the network. For example, consider a peer-to-peer network in which there are only two miners. The first miner has 60% of the hashpower, and the other

miner controls the remaining 40% of hashpower. The miner with less power may, through the effects of entropy, find a block hash first before the other miner, but most of the time the miner with the most hash power will find the correct block hash first and thusly determine the correct blocks and the longest blockchain. This example extends directly to any partitioning of network hashrate between honest miners and any other group of miners. So long as the honest miners on the network control most of the hashing power, they should find block hashes the fastest, add blocks to the blockchain more frequently, and by doing so maintain the honest blockchain as the longest blockchain.

D. Blockchain Soft Forks

The peer-to-peer network is distributed, that is peers are not in the same physical location, geographical region or controlled by the same entity. As such, there are multiple, unavoidable delaying factors that affect data propagation. This includes network and software delays. It is not practical for any single node to connect to every other node in the network. Thus, recursive flooding and other data propagation techniques are used to notify the entire peer-to-peer network. However, this takes time to transmit data between peers, even if only a few milliseconds in-between network hops. As for software delays, it takes time for a peer to process transmitted data. Furthermore, there is often a need for a data-receiving peer to send *getdata* requests to the data-transmitting peer and so there is some delay within the software ingesting and handling data it receives. Some obscure cases cause delays and subnetworks to be cut off due to

geographical or other difficulties. For example, nodes that are located within the same country, should international internet connections be lost for some period, the nodes within that country who are still connected to one another will form their own network and not be able to update to most peers until international connectivity is restored.

It is because of these delays, both typical and atypical, that multiple, valid, blockchains can appear within the same peer-to-peer network. Consider Fig. 4 below, where a single, main blockchain exists in a peer-to-peer network. There are a total of 103 blocks that have been produced and appended to the blockchain.

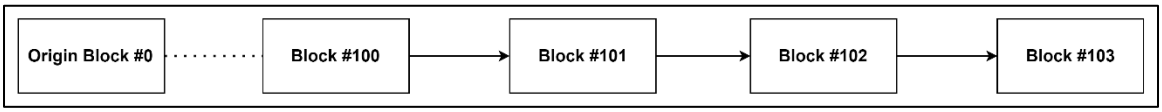

Fig. 4. Current longest blockchain on the network.

Miners can simultaneously produce blocks. The proposed blocks would all be numbered 104 and reference the current blockchain's latest block, 103. Observe in Fig. 5 that two different miners have each proposed a different block 104 at roughly the same time. These different versions will differ by time stamp, and likely differ by transactions contained in each.

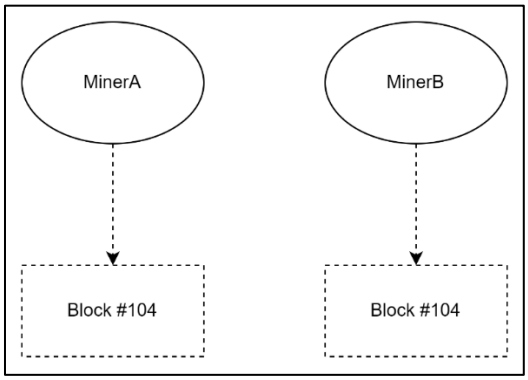

Fig. 5. Any subset of miners can propose a block simultaneously.

Since both blocks are valid, there are now two blockchain forks that exist simultaneously on the network. Fig. 6 below illustrates this forked state.

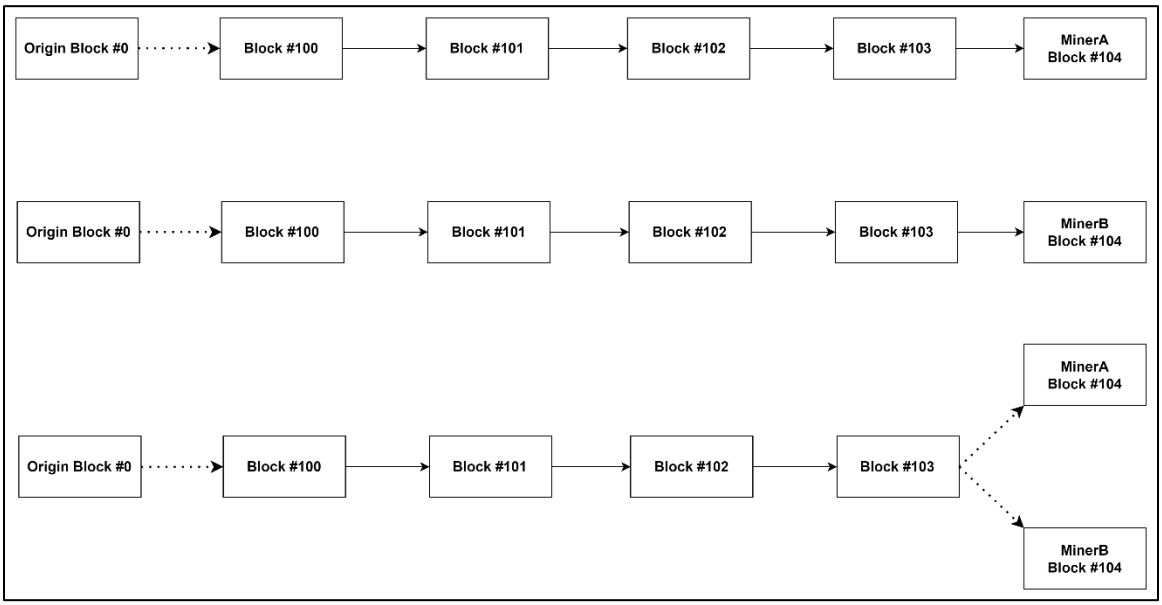

Fig. 6. The first two blockchains can be represented by the third, where there exists a common ancestor between two valid blocks of the same height.

Most proof of work cryptocurrencies resolve forks by using the longest chain consensus which relies on two factors:

1) Block height
    a. The total number of blocks in an individual fork
2) Chain work
    a. The total sum of hashing algorithm cycles used to produce each block in an individual fork. Higher chain work implies that a larger portion of the network hashrate has been used for longer to solve hashes for the blockchain. This is a determining factor for consensus because it should be

affected by majority rule, and game theory through incentivized mining by the network.

Blockchain reorganization and fork resolution will be discussed in greater detail in *Ch. IV, Section C, Subsection 7) Experimentation*. A note to the reader: the concept of soft forks is central to this thesis' experiment and as such is important to understand to fully understand the experimentation results, implications, and limitations.

E. Mining Software

Mining software is code used to implement specialized or high-powered hardware to execute as many hash functions as possible in the shortest possible time. Proof of work cryptocurrencies each implement their mining software differently, and even some third-party software developers write custom mining software. Regardless, for most proof of work cryptocurrencies, the mining software will execute the project developers-supplied core as a daemon that runs a full node that connects to the peer-to-peer network. According to [20], a daemon is "service process that runs in the background and supervises the system or provides functionality to other processes." In this case, the peer software is being run in a headless state (that is without any user interface components such as a dialog or system tray icon), listening on specific ports, and is both a server and a client to other peers in the network.

The mining software will look for transaction broadcasts, put them into a proposed block, use the current difficulty as prescribed by the network, and then themselves try to solve the block hash. More common is for mining to be operated as a pool [17]. That is

when a user or entity maintains an internet-facing server that is running mining pool software. This server will accept incoming connections from other miners, usually smaller, that are also running mining software just not a full node, only a mining client. As blocks are proposed and broadcasted to the miners on the network, the mining software will delegate sets or ranges of nonces to use in order to try to solve the block hash amongst all connected miners. If any of the miners find a block hash, then the block reward is split between all connected miners according to their contributions for that specific block. For example, Miner1, Miner2, and Miner3 all connect to the same mining pool. Of the mining pool's total hashrate, Miner1 and Miner2 each have 25%, and Miner3 has 50% for a total of 100%. If the mining pool can find a block, the mining rewards are split in proportion to the number of hashes each miner contributed. If the mining reward were 100 coins, then Miner1 and Miner2 would receive 25 coins, and Miner3 would receive 50 coins. The mining pool software would pay for itself by taking 1-5% of the block reward and then the rest would be distributed to the contributing miners according to their contributions for that specific block.

Most if not all proof of work coins run a daemon of the developer-provided core software regardless of whether it's a mining pool or a solo mining and regardless of the hardware being run (unless of course they are a miner connecting to the pool software). While the network architecture with pool mining varies, there's still a full node the pool software or solo miner must have, which necessitates that the mining software is either the vanilla developer-provided core or third-party software that complies with the

network's protocol (basically third-party software wrapped around the core, vanilla software).

F. Mining Hardware

There are 5 primary types of hardware that are used for mining in proof of work cryptocurrencies:

1) ASIC: Application-Specific Integrated Circuit
2) FPGA: Field-Programmable Gate Array
3) CPU: Central Processing Unit
4) GPU: Graphics Processing Unit
5) Storage Devices (Long term storage devices such as HDD, SSD, USB Drive)

The type of hardware used is dependent on several factors such as how well it performs the given hashing algorithm, hash density, consumer access, and cost. For example, Litecoin (LTC) is a proof of work coin that uses the hashing algorithm called Scrypt. An AMD Ryzen 9 5950X CPU can perform approximately 96 Kh/s (kilo hashes per second) [this performance was acquired during the experiment) whereas the Bitmain Antminer L7 ASIC can perform 9500 Mh/s (mega hashes per second) according to [19]. That is 96000 h/s compared to 8500000000 h/s from the ASIC. Thus, most miners who mine Litecoin use ASICs since they are much more powerful. Another example would be the algorithm Ethash (also called Dagger Hashimoto) used by Ethereum Classic (ETC) and other various proof of work cryptocurrencies. Both ASICS and GPUs are efficient at performing Ethash calculations and both are widely used. However, the cost of a GPU is

$250-2000 whereas an ASIC would cost between $2500-70000 [18]. A final example is the algorithm called Ghostrider, which is used by the Raptoreum (RTM) [38] proof of work cryptocurrency. Ghostrider was built specifically to prevent high-powered ASICs and FPGAs from mining and instead favors CPUs.

Availability, cost, and hash density are also major factors in determining which hardware is used. The previously discussed Ethash algorithm favors both ASICs and GPUs, however not everyone has tens of thousands of dollars to spend on highly specialized hardware, so many people instead decide to purchase GPUs to mine coins that use Ethash. Also, ASICs are often unavailable for immediate purchase and shipment and instead are on backorder. Hash density is also another choice that causes miners to purchase certain hardware.

Consider Ghostrider [37], a hashing algorithm that heavily favors CPUs, but that GPUs can also mine. CPUs execute hashes of Ghostrider more quickly the more L3 cache a CPU has, so AMD CPUs tend to outperform Intel CPUs by several magnitudes. Also, while GPUs can mine Ghostrider they are incredibly inefficient given how many watts they consume to perform so few hashes, so miners tend not to use GPUs unless that is all they have on hand, have low electricity costs, and have nothing more profitable to mine with their GPUs. There are less proof of work cryptocurrencies that use storage as a means to provide hash outputs. Coins like Chia (XCH), Chives (XCC), and BitcoinHD (BHD) are a few that use algorithms that are hashed with long term storage devices.

G. Block Reward

Miners mine because there is a block reward. That is, with most proof of work cryptocurrencies there is a special transaction at the beginning of each block that indicates a coin minting event often called a *coinbase transaction*. Whichever miner both A) finds a correct hash for the proposed block, B) submits their solution to the network first, and C) has their block added to the longest blockchain is rewarded with the block reward. Aside from the much smaller population of miners who have personal reasons to support the cryptocurrency they're mining; most miners purchase hardware and support cryptocurrencies through mining for financial incentive.

Many online exchanges such as Coinbase and Coinex allow miners and anyone else to trade their coins for fiat currency. After time and enough mining rewards, miners can pay their mining hardware costs and electricity costs if the cryptocurrencies they are mining are profitable. Whenever demand for cryptocurrency is up and the investment market is bullish or hopeful, miners can do very well trading their mining rewards for fiat to cover costs and to make a profit. When cryptocurrency demand is in a bear market, or electricity costs are too high and cause miners to mine at a loss, most miners will turn off their hardware until profitability increases.

There is a smaller subset of miners who mine regardless of the market state or electricity costs. There is yet a smaller subset of miners who speculatively mine. That is, some miners scour and search for newer or unreleased cryptocurrencies that don't have much hash power. When they can mine with all of their mining hardware on a

cryptocurrency with a small network, miners usually win the vast majority of block rewards while they are mining those coins. Miners do this in hopes that one day the coin will be worth selling, even though while they're actively mining the coins there is either not a market for the coin or the coin is worth very little per unit.

This concludes the introduction chapters of this thesis that hopefully provided the reader with enough basic information, and now the experimentation will immediately follow. The concepts of transaction inputs, soft forks, and hashrate are important to understand to fully appreciate the experiment , its results, and its implications.

IV. EXPERIMENTATION

This chapter will detail the experimentation phase of this thesis. *Chapter A Environment Setup* will cover the setup of the experiment including hardware, software, deployment methods, and other essential information should the reader wish to reproduce the experiments. *Chapters B* & *C* will recount two experiments conducted, the goals of which were to force a reorganization event on the peer-to-peer network that would force the honest network to adopt the attacker's soft fork which contains a transaction that double-spent the same transaction inputs as a transaction on the honest network. The first experiment was successful in forcing a block reorganization event, but failed to double-spend a transaction. The second experiment fully succeeded in both forcing the honest network to adopt the attacker's soft fork and double spending a transaction that caused the honest transaction to be orphaned.

A. Environment Setup

This experiment utilized a private fork of Litecoin 0.18. The instances of the wallet daemon were deployed on on-site hardware with Linux Ubuntu 16.04. Distribution of wallet daemon software was handled with Docker containers.

1) Network Setup: This experiment required a downscaled peer-to-peer network. It consisted of seven unique entities, of which four were physical computers, one was a router, and two were virtual machines run from a computer not on the peer-to-peer

network. The router connected the VM's host to the four physical computers. One of the VMs acted as a blockchain observer to query network status and values. The physical computers and the other VM acted as the cryptocurrency network asset, as shown in Fig. 7. One of the five was client-only software that was neither a full node nor a miner. The other four physical machines were all full nodes, and two were formally miners.

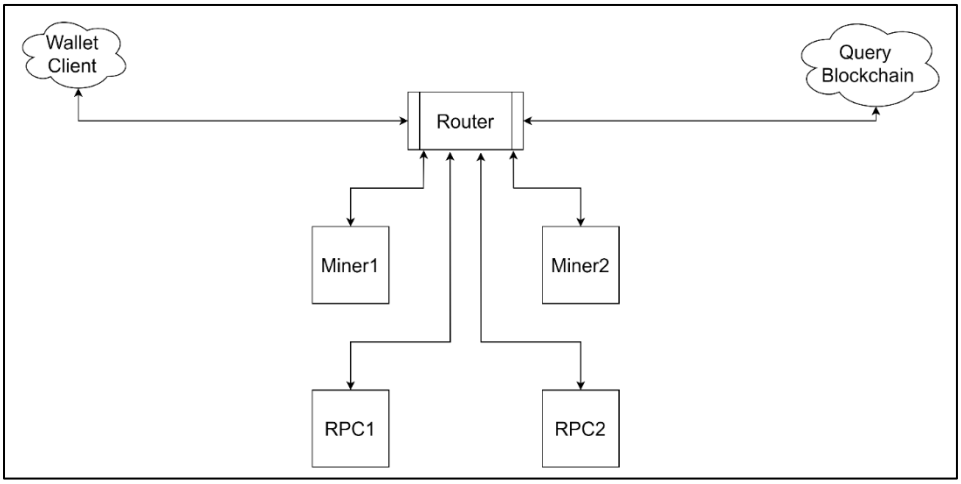

Fig. 7. Initial network architecture for experimentation.

2) Choice of Cryptocurrency: This experiment required a proof of work cryptocurrency that was open source. This thesis researched several candidate projects and several release versions within each candidate including Raptoreum, Dash, Bitcoin, Litecoin and Ethereum. This thesis determined that an early version of either Litecoin or Bitcoin would be best for the following reasons:

- Litecoin forked Bitcoin occurred very early on so the code bases would be very similar and thusly easier to understand and evaluate.

        - At the time of Litecoin's fork of Bitcoin, the only differences were the name, some marketing material assets such as images of the Litecoin logo, block time and the hashing algorithm used.
    - This thesis' research was based on Bitcoin and other proof of work cryptocurrencies that were forks of Bitcoin.
        - This would be advantageous due to acquired domain knowledge.
    - There were less features earlier on in Litecoin and Bitcoin
        - Security and otherwise, features were less numerous in the beginning. There were many features that supported scaling, data retrieval, efficiency and blockchain formatting that would have drastically increased the time it would have taken this thesis to make the fork operational.
    - Earlier versions of Litecoin and Bitcoin are heavily documented.
        - Tutorials, explanations, and other forms of documentation are plentiful and well preserved.

The final choice was Litecoin version 0.18 due to previously stated reasoning. The name of this forked version of Litecoin was called Testcoin [32].

3) Experiment Software: As the fork of Litecoin used is relatively dated, so are its host requirements. The host software used was Linux Ubuntu 16.04 LTS. Each of the physical machines in the peer-to-peer network were formatted and flashed with this version of Ubuntu, as well as the *Wallet Client* VM. The *Query Blockchain* VM used Linux Ubuntu 20.04 LTS, though this choice was arbitrary and made mostly because of

the software development dependencies that come preloaded as well as the same operating system as the other network assets would make for less disruptions during experimentation. The forked cryptocurrency software was built and distributed using Docker containers. Docker is a technology that virtualizes operating systems, as opposed to virtual machines that virtualize hardware. The basic unit of docker, the container, is an encapsulated operating system that can be used to install all dependencies and then deployed to any operating system that can run Docker. This was a premeditated choice due to experience building the fork for the first time, which was tedious, difficult, and had many configurable settings that needed to be reset each time a rebuild was required. Rapid updates to network peers were possible and the starting, stopping and modification of running peer instances became trivial tasks controllable from the *Query Blockchain* VM.

4) Experiment Hardware: The following hardware was used for the indicated network assets.

- RPC1
    - CPU: Intel Core i7-5820K @ 3.30GHz
    - MOBO: MSI X99S SLI PLUS
    - RAM: 2X4 GB DDR4 2133 MHz
    - GPU: Nvidia GeForce GT 520
- RPC2
    - CPU: Intel Core i5-9600K @ 3.70 GHz

        - MOBO: MSI Z390-A Pro
        - RAM: 4 GB DDR4 2133 MHz
        - GPU: MSI GeForce RTX 3070
- Miner1
    - CPU: AMD Ryzen 9 5950X
    - MOBO: ASUS Pro WS X570-ACE
    - RAM: 2X8 GB DDR4 3200 MHz
    - GPU: Nvidia GeForce GT 710
- Miner2
    - CPU: AMD Ryzen 5 1600X
    - MOBO: MSI B450M PRO-M2 MAX
    - RAM: 2X8 GB DDR4 3200 MHz
    - GPU: Zotac GeForce RTX 3070

Given the total network hashrate of 100%, Miner1 accounted for 96% and Miner2 accounted for 4%. With this disparity, Miner1 played the role of attacker as it controlled 51% or more of the network's hash power.

5) Wallet Creation: Upon installation of the operating system and Docker container, the instance of Testcoin software in each Docker container was manually started as a daemon. After the Testcoin instance was started, a command was issued for the daemon to erase any current wallet data and to initialize a new wallet. This would ensure that

each Docker container had its own wallet and there was no overlap of funds ownership between the nodes. This made tracking transactions across wallets simpler, such that the payee address corresponded to the wallet belonging to a different Testcoin daemon than the sender address.

B. Experiment I

The goal of this experiment was to attempt to overwrite transaction history on the blockchain by causing a block reorganization event. This attack was ultimately unsuccessful at overwriting user-to-user transactions but was successful in overwriting minting transactions via block rewards. As illustrated in Fig. 8, the attacker would attempt to overwrite a user-to-user transaction that was transmitted on the honest network while it was disconnected. Upon reconnecting the attacker after it had produced a longer blockchain, the theory being tested was whether or not the honest transaction that was sent would be orphaned or not. According to [36] a transaction is orphaned when inputs are not in the peer's mempool or local copy of the blockchain.

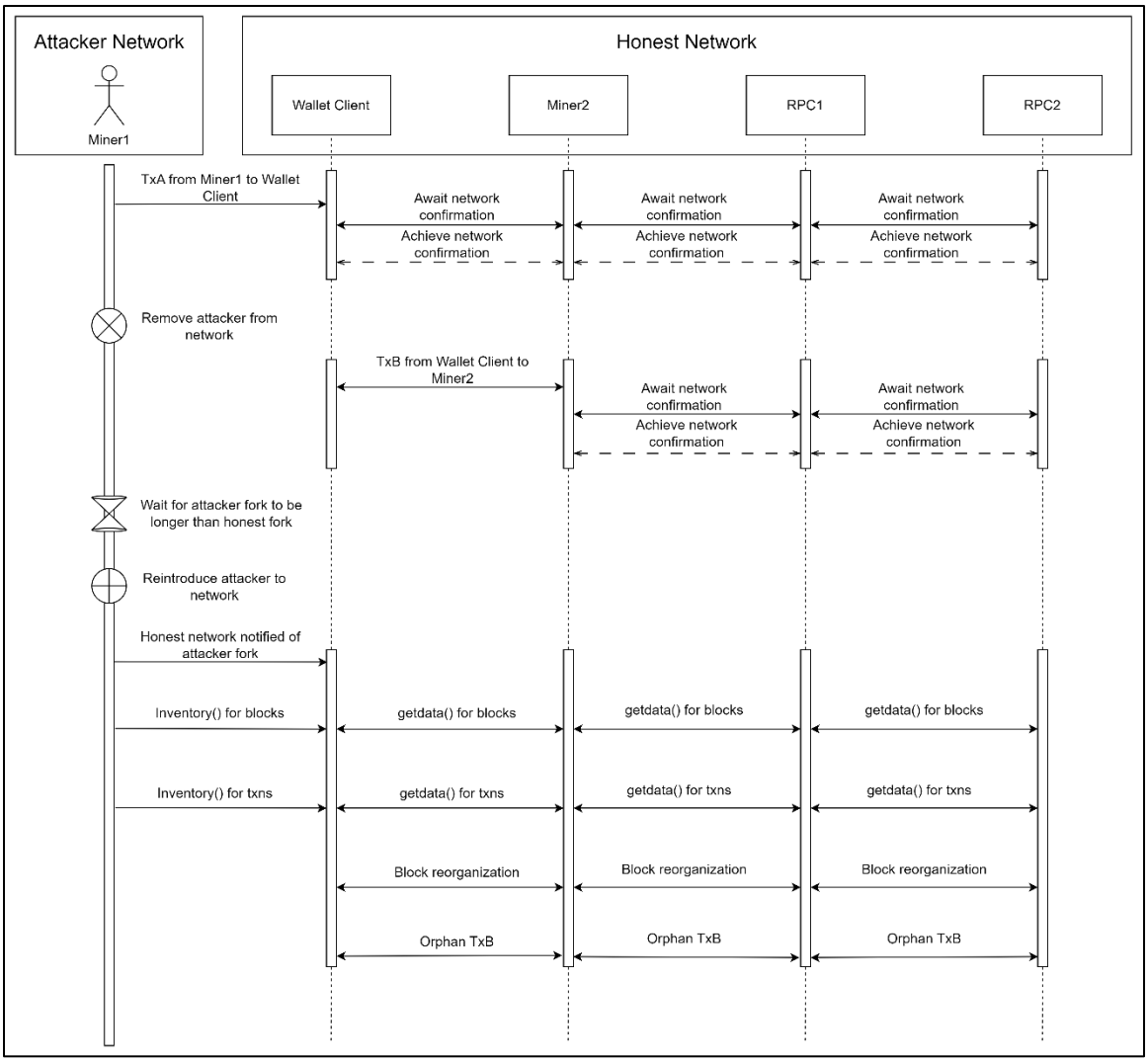

Fig. 8. Sequence diagram depicting initial experiment for experiment.

1) Order of Attack: The order of operations for this attack listed below are illustrated in Fig. 8 above:

1) The first step was to, while all peers were interconnected, submit a transaction, TxA, from Miner1 to Wallet Client.

2) The second step was to allow TxA to mature by allowing the minimum number of confirmations to accumulate. This second step simulates a payee being satisfied that their payment was successfully recorded in the blockchain at which time they would release the goods or service being paid for.
3) The third step was to remove Miner1 from the network.
4) The fourth step was to send a second transaction, TxB, from Wallet Client to Miner2.
5) The fifth step was to allow TxB to mature.
6) The sixth step was to leave Miner1 disconnected until it had mined enough blocks beyond what the honest network had mined.
7) The seventh step was to reconnect Miner1 to the rest of the network and monitor address balances, block reorganizations, and any orphaned transactions.

2) Findings: This experiment failed to overwrite the user-to-user transaction history of the honest network. During a block reorganization event, the peer software will orphan blocks. That is, the block will be disconnected from the blockchain and discarded, and the only evidence that the block ever existed is in log files. While the block itself is orphaned, the transactions contained are not so long as there are no previous uses of the inputs on the new blockchain. The experiment was fundamentally flawed because the same transactions were not being used on both the attacker and honest networks. Neither TxA nor TxB (see Fig. 8 above) were orphaned due to rebroadcasting of the transactions.

This thesis was able to determine that transaction rebroadcasting was an issue by examining the mempool. The memory pool, or mempool as it's called in Testcoin and by extension Litecoin software, is where queues of transactions are stored before being added to blocks on the blockchain. Mempool's storage of transactions is persistent until the transaction is added to a block that is on the blockchain. TxA and TxB were being rebroadcast to every node on the network after the block reorganization event, and thusly it was impossible to overwrite those transactions due to their distributed persistence in the mempools of various nodes. In Fig. 8-11 contain printouts of the Testcoin wallet software's console. In Fig. 8 and 9, the before and after details of TxA can be observed. Note that while the *time*, *timereceived*, and *details* sections have remained the same, the *blockhash* and *blocktime* sections have not. This is because the *blockhash* value in Fig. 9 belonged to an orphaned block, and the *blockhash* in Fig. 10 was the next block hashed after the reorganization event occurred.

```
gettransaction
b284b599a34633eb8e3b835ba43404967cb951e94f984bb4af052cf363e61d0b
{
"amount" : 1500.00000000,
"confirmations" : 5,
"blockhash" :
"06a98dc6497ddeefff28c83a6228d809b5591377325c1931e54269f113e9b54",
"blockindex" : 1,
"blocktime" : 1665974444,
"txid" : "b284b599a34633eb8e3b835ba43404967cb951e94f984bb4af052cf363e61d0b",
"normtxid" :
"67f6f0be2d2bc3380f46859e10d4acdf3e4ae1b56a2bd2d88a988d4a5e9c1af8",
"time" : 1665974360,
"timereceived" : 1665974360,
"details" : [
{
"account" : "",
"address" : "TWXQzj1rJtPzToZEvJYw7oXxqo1pQASMkw",
"category" : "receive",
"amount" : 1500.00000000
}
]
}
```

Fig. 9. Console output from Wallet Client of TxA before reconnecting attacker.

```
gettransaction
b284b599a34633eb8e3b835ba43404967cb951e94f984bb4af052cf363e61d0b
{
"amount" : 1500.00000000,
"confirmations" : 5,
"blockhash" :
"bc47f4c893a5996d38d8ddf3c1f4e5d10de657a65b732f045328b80cdf25b4fc",
"blockindex" : 1,
"blocktime" : 1665976430,
"txid" : "b284b599a34633eb8e3b835ba43404967cb951e94f984bb4af052cf363e61d0b",
"normtxid" :
"67f6f0be2d2bc3380f46859e10d4acdf3e4ae1b56a2bd2d88a988d4a5e9c1af8",
"time" : 1665974360,
"timereceived" : 1665974360,
"details" : [
{
"account" : "",
"address" : "TWXQzj1rJtPzToZEvJYw7oXxqo1pQASMkw",
"category" : "receive",
"amount" : 1500.00000000
}
]
}
```

Fig. 10. Console output from Wallet Client of TxA after reconnecting attacker.

In Fig. 11 and 12, the blocks containing TxA can be observed. In Fig. 11, block at height 16756 contained TxA, whereas in Fig. 12 block at height 16770 contained TxA. This is evidence of the reorganization event orphaning block 16756 and replacing it with the attacker's version of that block that didn't contain TxA. However, due to rebroadcasting to the mempools TxA was not lost and was instead included in the first block that was hashed after the reorganization event.

```
getblock 06a98dc6497ddeeffff28c83a6228d809b5591377325c1931e54269f113e9b54
{
"hash" : "06a98dc6497ddeeffff28c83a6228d809b5591377325c1931e54269f113e9b54",
"confirmations" : 5,
"size" : 17521,
"height" : 16756,
"version" : 2,
"merkleroot" :
"d7d765d3226c8c08cc00d7f5acdba00acb2d1d02e6fb2b600754d916be25a89b",
"tx" : [
"80bd84f8fb70aabe5723e0456d8faa72612bb37db81753747d55b2a3a4d401cf",
"b284b599a34633eb8e3b835ba43404967cb951e94f984bb4af052cf363e61d0b"
],
"time" : 1665974444,
"nonce" : 228597,
"bits" : "1e02f19e",
"difficulty" : 0.00132691,
"previousblockhash" :
"934c71f0ccdd440cb9197bd323e61459cfdb5c6f6f3b4ab9f896e14428898a32",
"nextblockhash" :
"bf12fa9ea3e869e35b18dff4e78147d6ddbeb7d97815536353d149a8a86e12b9"
}
```

Fig. 11. Console output from Wallet Client of block containing TxA before reconnecting attacker.

```
getblock bc47f4c893a5996d38d8ddf3c1f4e5d10de657a65b732f045328b80cdf25b4fc
{
"hash" : "bc47f4c893a5996d38d8ddf3c1f4e5d10de657a65b732f045328b80cdf25b4fc",
"confirmations" : 3,
"size" : 17521,
"height" : 16770,
"version" : 2,
"merkleroot" :
"d5459a95bdabf7e13c570565a05b1cc7c4465d6e68cd7130231c7ec0bc9c0de5",
"tx" : [
"40b334635f4d8d23f01fff7a67facdf1eb7cb6aad7497823c138e5e987cee0c0",
"b284b599a34633eb8e3b835ba43404967cb951e94f984bb4af052cf363e61d0b"
],
"time" : 1665976430,
"nonce" : 237462,
"bits" : "1e02f19e",
"difficulty" : 0.00132691,
"previousblockhash" :
"cc26aadbcbd3904611ba96822bd85ebbc8094cec1e6e66f67d98d5708393bf37",
"nextblockhash" :
"5339bcce45374423aab7d2fd0829a6c7d19a00ea9beaf5caa6efd99607ed65c7"
}
```


Fig. 12. Console output from Wallet Client of block containing TxA after reconnecting attacker.

The third problem was mining timeout due to connectivity. Testcoin peers and many other cryptocurrencies keep a collection of connected peers that are verified and updated periodically. That is, the software will examine each peer in its collection and attempt to send and receive data. If the attempt is unsuccessful, a log event will be recorded. These attempts to connect occur periodically. After several failed attempts to connect to peers in its collection, the software will remove that node from its collection. This is important to note because the software will only mine as long as it has at least a non-empty collection of connected peers. This was a problem in this experiment for a few reasons. The first reason was the attacker was separated from every other peer while it was disconnected. This can be observed in Fig. 13.

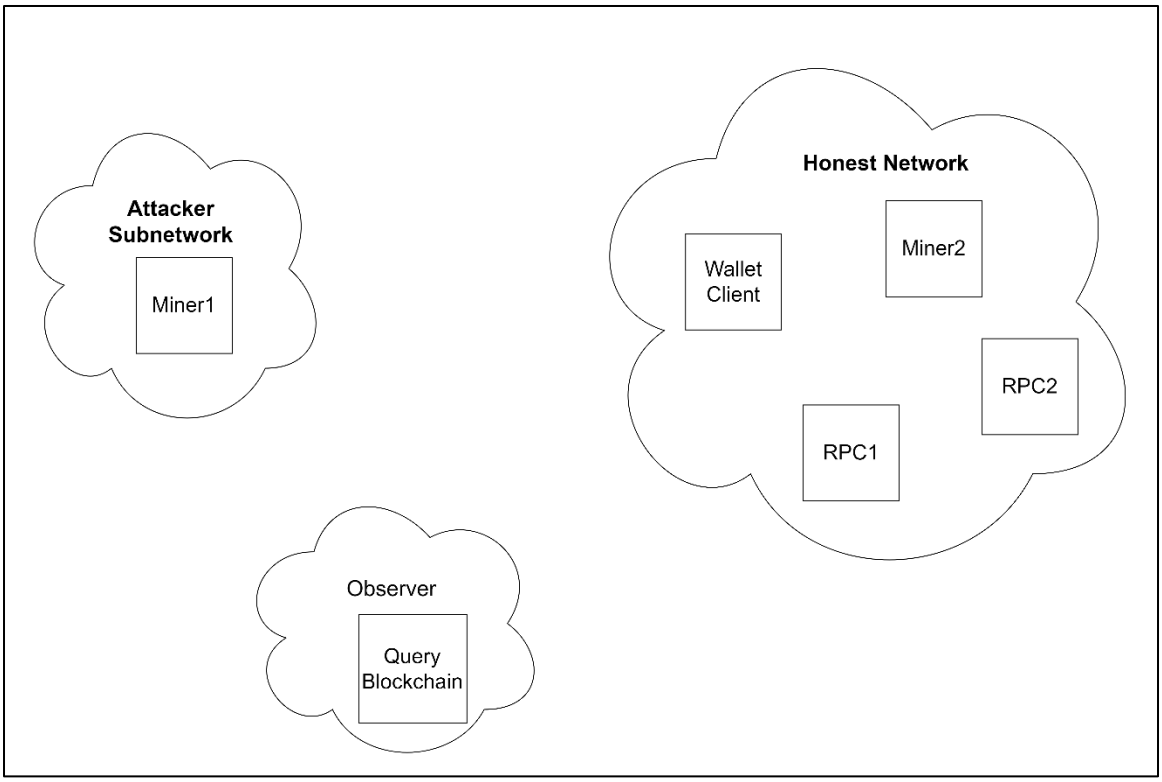

Fig. 13. Seperation of network assets after attacker disconnected from peer-to-peer network during initial experiment.

The second reason this was a problem is there was a time limit on how long the attacker could mine while disconnected. During several attempts to execute this experiment, the miner would stop mining before producing a longer blockchain than the honest network. An attacker could theoretically remain unconnected indefinitely (or at least until a major update for the peer software is released) and overwrite any honest blocks that were produced during that time. The attacker would need to be permitted to remain disconnected as long as required to out-hash the honest network.

3) Progress Achieved: While there were many issues with this experiment, it also bore fruit. The fact that a reorganization event was not only possible but somewhat trivial to reproduce using this experiment was a major success. If the attacker cannot force the honest network to accept its blockchain, then the double-spending attack will not be successful as whatever transaction the honest network has recorded will be kept. Secondly, the minting transactions were overwritten. With the attacker off the network, Miner2 and Wallet Client were being awarded mining rewards for solving blocks. However, when Miner1 (attacker) was reconnected to the network the mining rewards for orphaned blocks were removed from the honest miners and instead awarded to Miner1. Fig. 14 shows the transaction list of Wallet Client before the block reorganization event. All transactions with "Mined" in the third column and between 19:33-19:46 are mining reward transactions. The red clock icon to the left of each indicates that the transactions are still being confirmed by the network and cannot be spent yet, though they are still valid.

	10/16/22 19:46	Mined	(TM9HtS1iU7zK6744iNsmgk1xuiSyaEernD)	[5.00]
	10/16/22 19:43	Mined	(THoDCKZLraL9yvevPodKqDWi8xcSV3padQ)	[5.00]
	10/16/22 19:43	Mined	(TNUpTSdi1jVjDuGR7uFEjEoHVswUAGkDXd)	[5.00]
	10/16/22 19:42	Mined	(TBpGZY5HsasiXCehHimFXvsW8EmafNx5qR)	[5.00]
✓	10/16/22 19:39	Received with	(TWXQzj1rJtPzToZEvJYw7oXxqo1pQASMkw)	1500.00
	10/16/22 19:38	Mined	(THM4KQSRf3ZfiUSq75Uy3cmLgSJ12Hexa6)	[5.00]
	10/16/22 19:38	Mined	(TV4uh2Yw5Jaf2o5shswDRzHRu2qaSXsWJQ)	[5.00]
✓	10/16/22 19:33	Received with	(TWXQzj1rJtPzToZEvJYw7oXxqo1pQASMkw)	1000.00
?	10/16/22 19:32	Mined	(TXnXBxnezQKtETTQgyrYWTvYZBQmin5s8o)	[5.00]
?	10/16/22 19:30	Mined	(TYeP7SRCQR6t7uRbqrwcaqL7WquCpo4Go8)	[5.00]
✓	10/16/22 19:19	Received with	(TWXQzj1rJtPzToZEvJYw7oXxqo1pQASMkw)	100.00
✓	10/16/22 19:07	Received with	(TWXQzj1rJtPzToZEvJYw7oXxqo1pQASMkw)	100.00

Fig. 14. Output from Wallet Client software showing mining rewards transactions as valid and being confirmed by network.

In Fig. 15, those same mining reward transactions between 19:33 and 19:46 have been voided after the attacker was reconnected and the block reorganization event occurred. Also, note that the transaction submitted at 19:39 (TxA) was reordered after it was orphaned from block 16756 and readded to the blockchain at block 16770, which was a higher block height than the original mining rewards.

	10/16/22 19:39	Received with	(TWXQzj1rJtPzToZEvJYw7oXxqo1pQASMkw)	1500.00
?	10/16/22 19:46	Mined	(TM9HtS1iU7zK6744iNsmgk1xuiSyaEernD)	[5.00]
?	10/16/22 19:43	Mined	(THoDCKZLraL9yvevPodKqDWi8xcSV3padQ)	[5.00]
?	10/16/22 19:43	Mined	(TNUpTSdi1jVjDuGR7uFEjEoHVswUAGkDXd)	[5.00]
?	10/16/22 19:42	Mined	(TBpGZY5HsasiXCehHimFXvsW8EmafNx5qR)	[5.00]
?	10/16/22 19:38	Mined	(THM4KQSRf3ZfiUSq75Uy3cmLgSJ12Hexa6)	[5.00]
?	10/16/22 19:38	Mined	(TV4uh2Yw5Jaf2o5shswDRzHRu2qaSXsWJQ)	[5.00]
✓	10/16/22 19:33	Received with	(TWXQzj1rJtPzToZEvJYw7oXxqo1pQASMkw)	1000.00
?	10/16/22 19:32	Mined	(TXnXBxnezQKtETTQgyrYWTvYZBQmin5s8o)	[5.00]
?	10/16/22 19:30	Mined	(TYeP7SRCQR6t7uRbqrwcaqL7WquCpo4Go8)	[5.00]
✓	10/16/22 19:19	Received with	(TWXQzj1rJtPzToZEvJYw7oXxqo1pQASMkw)	100.00
✓	10/16/22 19:07	Received with	(TWXQzj1rJtPzToZEvJYw7oXxqo1pQASMkw)	100.00

Fig. 15. Output from Wallet Client software showing mining reward transactions being invalid.

C. Experiment II

This experiment was an improvement on the previous iteration. Changes were made to the experiment with the limitations encountered in mind.

1) Order of Attack: The order of operations, illustrated in Fig. 16, for this attack were as follows:

1) Disconnect attacker subnetwork;
2) Send TxA on attacker subnetwork from Miner1 to RPC3;
3) Send TxB on honest network from Wallet Client to Miner2;
4) Wait for attacker blockchain to be longer than honest blockchain;
5) Reconnect attacker subnetwork to honest network;
6) Monitor address balances, block reorganizations, and any orphaned transactions.

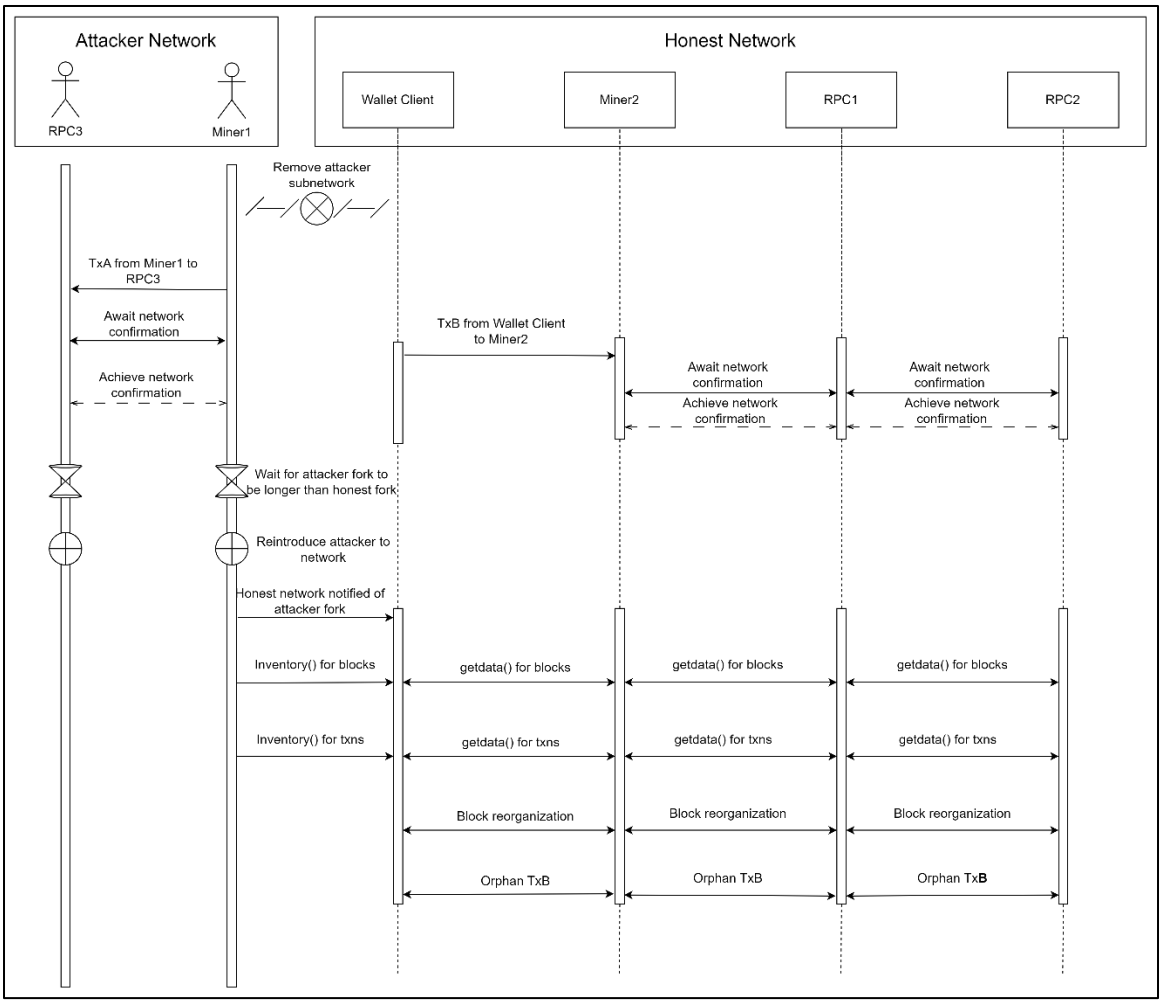

Fig. 16. Sequence diagram depicting revised experiment procedure after initial limitations.

2) Network Architecture Change: To counteract the mining timeout due to lack of network connectivity, RPC3 was created and installed on the same hardware as Miner1. When the physical CAT cable was removed from the motherboard, Miner1 and RPC3 would still be able to communicate and act as a network. This was accomplished by deploying another Docker container copied from RPC2 and adjusted to be a unique peer called RPC3. Additionally, this required some networking consideration. Miner1 was

exposed on the default network interface IP and ports. This change can be observed in Fig. 17.

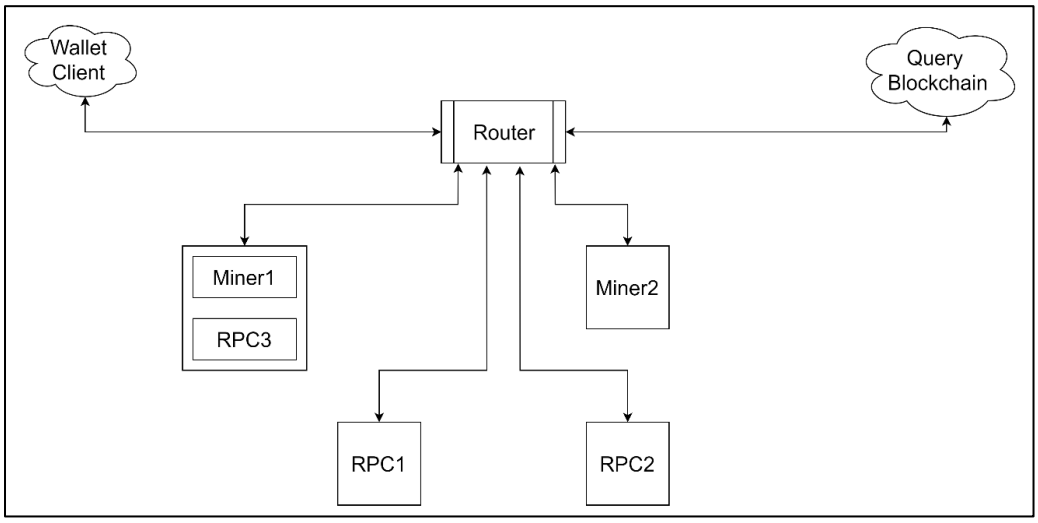

Fig. 17. Network architecture revision, adding a peer called RPC3 to the hardware running the attacker, Miner1.

RPC3, however, was only exposed to Miner1 on the Docker default network interface bridge. RPC3 was always exposed to Miner1 although no other peer on the network could connect to it. Certain ports are hardcoded in the Testcoin software that instructs peers which ports to broadcast themselves on for test and production. Because of this, it was difficult to expose both Miner1 and RPC3 to the rest of the network. An attempt was made to create another network interface on the host OS; however, it was an unsuccessful one. As such, this closed loop between Miner1 and RPC3 was created to account for this issue. In Fig. 18, the attacker subnetwork is illustrated as being a loop so that Miner1 would have as much time as required to out-hash the honest network.

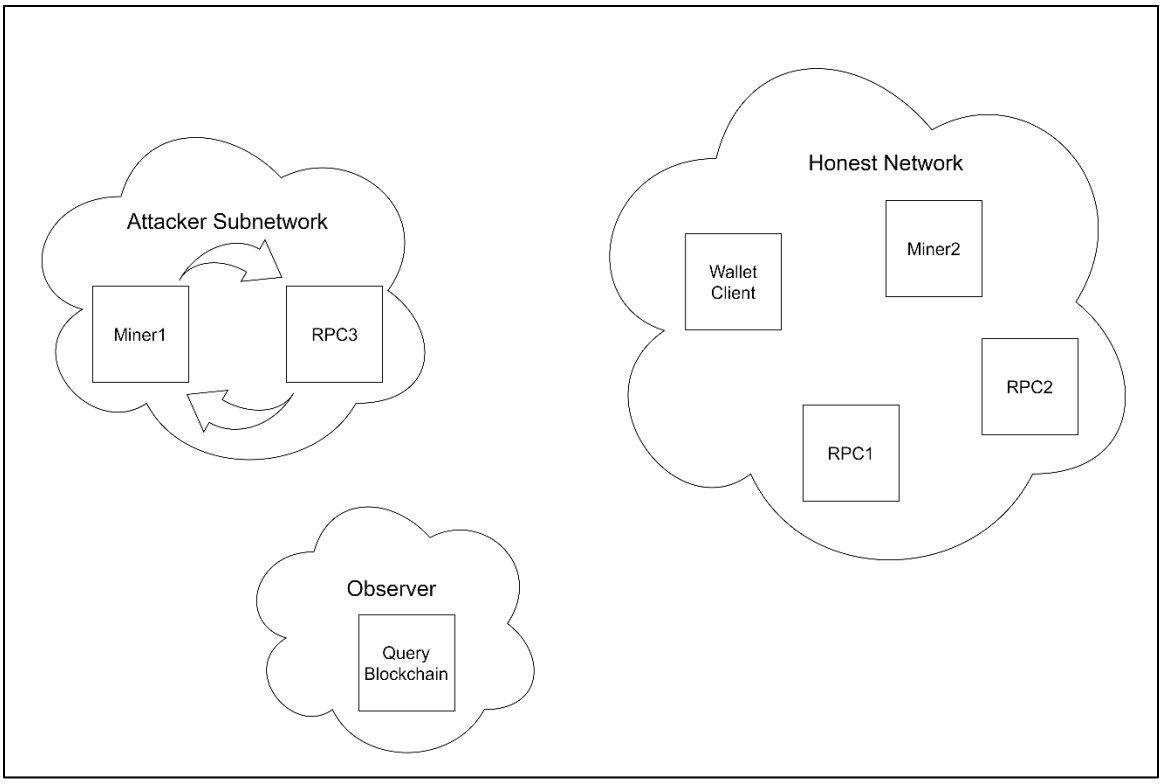

Fig. 18. Seperation of network assets after attacker disconnected from peer-to-peer network revised.

This created a sub-network within the overall peer-to-peer network that would ensure Miner1 would continue mining regardless of how long it was disconnected from the honest network. The connection between Miner1 and RPC3 can be described as a sub-network of the peer-to-peer group comprising Testcoin. To address the inputs issue that was also allowing transactions to be rebroadcast, two copies of the same wallet were installed. That is, when a Testcoin daemon (and most other cryptocurrency wallet software) initializes a wallet, it creates a private key/seed file from which all keys and addresses are generated. Miner1 was reinitialized, and then the private key/seed file was copied to Wallet Client. In effect, both Miner1 and Wallet Client were the same funds

owner, and each could make transactions that would directly impact the other's ability to do so.

3) Disconnect Attacker Sub-Network: The first step in this experiment was to remove the attacker sub-network from Testcoin peer-to-peer network. This will accomplish some objectives:

- The attacker can begin hashing a blockchain fork that will certainly be longer than the honest network's blockchain.
    - When two or more blocks have the same block height, forking the block chain. Typically occurs when two or more miners find blocks at nearly the same time. Can also happen as part of an attack according to [10].
- The attacker can send a transaction on its subnetwork without the rest of the network discovering the transaction until it is reconnected.

4) Double-spend on Attacker Network: Immediately after disconnecting the attacker sub-network, a transaction, TxA, was sent on the honest network from Miner1 to itself. This was accomplished by sending funds to an address that Miner1 owns. The hash for TxA was 44a13080d7ac089a5dcc062af75b77e1537c1135141e95dc36d2795c104d2007. This transaction can be shortened to **Tx44a** for brevity.

5) Send Transaction on Honest Network: Immediately after submitting TxA on the attacker sub-network, TxB was submitted on the honest network from Wallet Client to Miner2 using at least one of the same inputs as Tx44a. The hash for TxB was

8b81f320313f04685c88fff8c9ab82bc9cf7ffbf11d9676fa4355a63e35c590f. This transaction can be shortened to **Tx8b8** for brevity.

6) Increasing Hashpower on Honest Network: An unforeseen issue was that difficulty had not adjusted quickly enough to accommodate the massive reduction in hashpower from the honest network when the attacker sub-network was disconnected. While the attacker was still generating enough hashpower to generate blocks approximately on-schedule, the honest network was struggling greatly. As a reminder, Miner2 only had 4% of the network hashrate. As such, RPC1 and RPC2 were forced to mine to approximately triple the hashpower of the honest network. In effect, this temporary bump in hashrate increased the honest network's share of the original hashrate from 4% to 12%. This became an issue because TxA could not get confirmation. To ensure this experiment's success, the honest transaction needed to be confirmed to simulate a payee being confident enough to release goods or services. As soon as the requisite confirmations were attained, RPC1 and RPC2 had their mining turned off and they returned to simple peers.

7) Reconnect Attacker: Between Tx8b8 and Tx44a, each has 151 inputs. Of those 151 inputs for each transaction, ninety-six were shared. The log file entry for Wallet Client creating Tx8b8 can be observed in Fig. 19.

```
3846: 2022-10-31 03:32:00 CommitTransaction:
3847: CTransaction(hash=8b81f320313f04685c88fff8c9ab82bc9cf7ffbf11d9676fa4355a63e35c590f, ver=1, vin.size=151, vout.size=2, nLockTime=0)
3848:     CTxIn(COutPoint(63e99811e3290fadbee90a096d531e5db258dbe290d4ac7f60fc6b27ad0f0007, 0), scriptSig=3045022100a3952b2d0739b3)
3849:     CTxIn(COutPoint(89f61a673f911ccc0bbdf69d830150fc3333d45508c68a68ca3c9320d33b0409, 0), scriptSig=3045022200408f3dbd2736c26)
3850:     CTxIn(COutPoint(93bdf31722841aa86467a53768b38dc42359cc48eaee7d3caf3a43d54474070b, 0), scriptSig=304402204fc0329955f6d3b8)
3851:     CTxIn(COutPoint(3d629c52213e13ce0c2cc8c4337ebce9018c274a77749fdfa02667524bcc0d0b, 0), scriptSig=30440220497a5429668c670e)
3852:     CTxIn(COutPoint(f7c0672bc590c6d932e410e31c069facffe22172ae106207f1c020df05ea4e05, 0), scriptSig=3045022007d7362a51ffbf3d)
3853:     CTxIn(COutPoint(b50935eb26acdc1a895189638997804808073188699a02eb5477b01b16abe00e, 0), scriptSig=30440220497da0819d43786a)
3854:     CTxIn(COutPoint(6c10a43a952c5c5244edf533e63a73092519c6fea156eb053355af1d4143ad0f, 0), scriptSig=3046022100a5aa8b60388d9e)
3855:     CTxIn(COutPoint(f42371dfec562b179a3d1464b789aa0225bd72d93f2725306ed1672ae2992112, 0), scriptSig=304502200ffce71e4c66ccb7)
3856:     CTxIn(COutPoint(14566c894942dca42a292e4b0eb235a9cd3f3037dde63d81e9ade414d5c08f12, 0), scriptSig=3045022100910dbaf8cc0b9a)
3857:     CTxIn(COutPoint(3a22a500d21179420ac5cca0f2f713b82e158d2e665c64291e442bfd1465c517, 0), scriptSig=304402202599bd1cd57496aa)
3858:     CTxIn(COutPoint(6988735f277cef93d89719f7b10519f9b1246d6962514e6cdacfc49cdf13f92a, 0), scriptSig=3046022100b705a4cea8b5ff)
3859:     CTxIn(COutPoint(8cb0a60b992b19cd6e1f1e39a112700ab6542f20eb47ee6eb9a3c9c5c9837632, 0), scriptSig=3044022033d98bfeb687e2ca)
3860:     CTxIn(COutPoint(5633c1a775a4832b2997d9b5f28acbaac5c54973b1ede1abc5d209c404be9034, 0), scriptSig=304502206a7b1e99fc561495)
3861:     CTxIn(COutPoint(fb3806d7279a657850a5f42f7e71bdd3b32c6393c312d701f8afe5526dbbc135, 0), scriptSig=30460221009181b67068cd15)
3862:     CTxIn(COutPoint(b919eb1cfdb1c06f09b259030f41576a7d1ab99f691bb87d9e05ed21b972ce48, 0), scriptSig=304502202426fc4d6aaefe1a)
3863:     CTxIn(COutPoint(e10fb17ab7818f2e69a773b3e6c433001679be2ac8d4f0db5beba8fedd1d674b, 0), scriptSig=3045022100f8b18711c15209)
3864:     CTxIn(COutPoint(2887f929b36dc4792e85f72d4848666d7e859f93bf53123d190f9b927f2764455, 0), scriptSig=3046022100fdaa3d32293dd9)
3865:     CTxIn(COutPoint(3fe51444ec0c0b268b406fcf57d3a0fa9653976ac204ee797d0acd67472f6856, 0), scriptSig=3046022100ed05593189dabb)
3866:     CTxIn(COutPoint(456c3953c8ea968a93151427d35e1249aa12c213267344d52be331bbcb8e635e, 0), scriptSig=3045022020ce88de5e6b28b9)
3867:     CTxIn(COutPoint(48e346db367dffc0134cc37f8e0791e148b28ceacbc0d5e8ed57e60c6a2d6a60, 0), scriptSig=3046022100f8a56259451828)
3868:     CTxIn(COutPoint(48cc427eeaa25f17a07cfc19e3259b5767659e9654dcb2cb736da5c305151361, 0), scriptSig=3046022100995b7b8ab05d1b)
3869:     CTxIn(COutPoint(62a588c035b4f6a97ba261a5efc682d640ad7d844c8245888b4ed5072ab90866, 0), scriptSig=3046022100929e985dc12a3e)
3870:     CTxIn(COutPoint(bfdc8018db8dfd561340194c6ae19eba932d3cc1cd3f4845e80f4b625280e671, 0), scriptSig=3046022100dfd02432fb9c23)
3871:     CTxIn(COutPoint(db0ef6337bd1fe97598ef000675f5ac93f781272df6ce1508edbb3a1a1c38f7b, 0), scriptSig=304402203cff0cb4922706ea)
3872:     CTxIn(COutPoint(a363e6ee0d22b7689ad4c706fc9874562f32d953d983c1864d6537c34f84217c, 0), scriptSig=304502205e0c143ee724db7d)
-------------------------------------------------------------------------------------------------------------------------------------
3969:     CTxIn(COutPoint(ac502ca9de1768dc527e510151a479a3bb03b00bc237f04b2351bffc603360ec, 0), scriptSig=30450221009226a8fc107794)
3970:     CTxIn(COutPoint(e046fa6ff8625edf4998b45c9b198d465f8edb0b6081c6732acc5178ad0f25cc, 0), scriptSig=3045022100944cf0f5858617)
3971:     CTxIn(COutPoint(0673a4373619ddcc6b283e4438c29501b979d4565aa97a78488a2a92a8aef9e4, 0), scriptSig=3045022100db445f2892649d)
3972:     CTxIn(COutPoint(4bb19b57b6098a23242ea70b3d938919a8303fd7e67563976894bca8c6c9eb50, 0), scriptSig=304402206dfba6b095224140)
3973:     CTxIn(COutPoint(f84ba2fbe94cfba929680d5e5e9fcf82a2c9cf3f3cf64a6ced0cdd3c4f50c057, 0), scriptSig=3046022100d4d4db0dde4fd3)
3974:     CTxIn(COutPoint(25a9a34602f6ce7b4950c4e9e0f0312423e2c1072ee804d816a027519e2dd3d4, 0), scriptSig=3045022100ff5b80a7226df7)
3975:     CTxIn(COutPoint(4bd79969268da2deae4f8edc5b31d77479756927 53b286e9b2def80055a0a8d1, 0), scriptSig=3044022059d7c79e99fd786e)
3976:     CTxIn(COutPoint(139ac91c17e2f891065a876be481d9b1c379cbdf1198684e3543168ac876d605, 0), scriptSig=3046022100e4aa4e72c25827)
3977:     CTxIn(COutPoint(36dc9faaf17dbed71509896afaa58722a7bddd2433979cbbc3941b74b24ed31e, 0), scriptSig=3045022200d94a3eb4aa5379b)
3978:     CTxIn(COutPoint(2e09d7f5efc8360ff323f4f8ec930ab8c67a6a99ca840f0d99de8f6252af2b38, 0), scriptSig=3046022100ce7fa1d66f75d6)
3979:     CTxIn(COutPoint(1ef01c1bb1c5d2303467232 93effb5bac649aeae24119c444dab90dcde1f25bc, 0), scriptSig=3045022100 9ebbd30801c994)
3980:     CTxIn(COutPoint(e00587e5e2184be528eebd0cbbf9d8ebee184238432cb7b66cfc8af02475aa88, 0), scriptSig=304402201d6bb136f3c1e23f)
3981:     CTxIn(COutPoint(ad930a79a3794fe6e52bbcfe9b49c4ccc50007aaad4ae92497e592e809e3f983, 0), scriptSig=3045022061bf5774670e7898)
3982:     CTxIn(COutPoint(5d3f97a156e8094c5b7637b9e775e8936a39f3b1b8e8bc2a904fd68419aa6662, 0), scriptSig=30460221009 3aea2dbf69e92)
3983:     CTxIn(COutPoint(c654918ee5ebbfd974de7b9905dc37d6923c30527fad9bd153526a22bfc91d0b, 0), scriptSig=30460221009c17251a9966c4)
3984:     CTxIn(COutPoint(a7f25b0fd53f88a6d6e45936313d709f1a72f5961523eff20297dcae0506f6e7, 0), scriptSig=3046022100a3618fcdc0a28f)
3985:     CTxIn(COutPoint(7396b648161e1d1784acc5d14b9f5e438f8ddd8cfb9d1d8e828e1c32e0e5c6fb, 0), scriptSig=304502205597d39b649c0ebc)
3986:     CTxIn(COutPoint(c566086b9f7a5cd11c8315fb2f99645de81cee4b8ae1eb088cfd7128e3b523a0, 0), scriptSig=3046022100cef2290e4d95d5)
3987:     CTxIn(COutPoint(3ae375c9e098c4c3517d157f8e713425d1620ed2c834d695e95061504ae0c743, 0), scriptSig=3045022100952e29d668700e)
3988:     CTxIn(COutPoint(7f8ddd933a437c8c4d6ef7ec4eb9f1fbe5cb4f75b6a705bd6df50dc7f8cd1692, 0), scriptSig=304402202162e4ebe0813b7f)
3989:     CTxIn(COutPoint(bdc77e676415babd8ea617d9639a2b67e9b501d70ea66041ce81569f3485f405, 0), scriptSig=3045022100a3ee477bb62325)
3990:     CTxIn(COutPoint(cb7affb080f763db015ea288ec0345250539bc5e343ad995bced1102f3136ba3, 0), scriptSig=304502206dcacafb31bf9b57)
3991:     CTxIn(COutPoint(98f5f4e718195d76f0d76bef6be4417708d24b5170744ba67e6e80b74318c99d, 0), scriptSig=3045022100d2fc79f897d37d)
3992:     CTxIn(COutPoint(3c688234caa246c53db5253619e4dfd1fa7ec27bb4ffe7de3aa8f65cc96cd324, 0), scriptSig=3045022079d1355a15ef30b3)
3993:     CTxIn(COutPoint(3fc6de54e2e4a66c4e786f9c37054a5cbe257df7ee43d62c10c8c65db6104c26, 0), scriptSig=3045022100c898753c3cccdd)
3994:     CTxIn(COutPoint(77596a22239ac6102bc6ccaffe7ce79d81f57bfdccbad8292f9760d460265f83, 0), scriptSig=3046022100e414d17ac17442)
3995:     CTxIn(COutPoint(e735fb81f0d8d267ba9be8d827a1bfa4fdda903cab361c01978547743700 6603, 0), scriptSig=3045022075a5d33c39f03964)
3996:     CTxIn(COutPoint(e869fc073d63c66bd2ed5fe251a5ccb73f0c1b8d2a2832b8228431754379d28a, 0), scriptSig=3045022059bfde02625f797e)
3997:     CTxIn(COutPoint(6b7f81ldd9728af383acb13b5b62a7a54ddad350b0aa0c8730906b502d9b88829, 0), scriptSig=304602210088b406a11aa344)
3998:     CTxIn(COutPoint(25ef00416c5aeda6d26c8ef485296eeb32aafdf9e9cd4e08abbdff727bbedd81, 0), scriptSig=30450220280d33b77d306ee1)
3999:     CTxOut(nValue=9.98200000, scriptPubKey=OP_DUP OP_HASH160 9e44974da1eb)
4000:     CTxOut(nValue=1500.00000000, scriptPubKey=OP_DUP OP_HASH160 639847d52279)
4001: keypool keep 20
```

Fig. 19. Wallet Client log file excerpt of Tx8b8 inputs. Line numbers are to the left and dashed line indicates remove of text for brevity.

Upon reconnecting the attacker sub-network to the honest network, the peers begin to communicate block height (number of blocks) and block hashes for each block. When the network finds a difference between the honest blockchain and attacker blockchain, it begins to compare blockchains to determine which one is most valid/correct. The software considers two variables: block height and chainwork. Chainwork is the cumulative number of hashes it took the network to solve each block. For example,

consider the same block height of two competing blockchains, one produced by Miner1 and the other by Miner2. If Miner1 solved the block hash with 1000 hash cycles and Miner2 solved the block hash in 10000 block cycles, then Miner2 would have more chainwork for that block.

When the software is comparing blockchains, it is both comparing the total sum of all hash cycles for each block in each blockchain as well as the number of blocks in each blockchain. In the case of this experiment, the attacker network mined an additional 79 blocks (block height = 353) which can be observed in Fig. 20, and the honest network mined an additional 11 blocks (block height 274), which can be observed in Fig. 21, after the attacker was disconnected from the network. The chainwork for the attacker network was approximately 371201680 cycles and the chainwork for the honest network was approximately 299897239 cycles. So not only was the attacker at a higher block height but also had significantly more chainwork completed. As such, the attacker's blockchain was considered the valid fork and adopted by the blockchain.

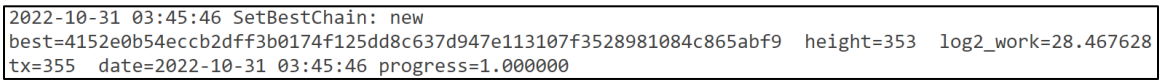

Fig. 20. Last block mined by attacker network before reconnecting to honest network.

Fig. 21. Last block mined by honest network before reconnecting attacker network.

The chainwork values were calculated by inverting the log function pictured in Fig. 22 below.

```
printf("SetBestChain: new best=%s  height=%d  log2_work=%.8g  tx=%lu  date=%s
progress=%f\n",
  hashBestChain.ToString().c_str(), nBestHeight, log(nBestChainWork.getdouble())/log(2.0),
  (unsigned long)pindexNew->nChainTx,
  DateTimeStrFormat("%Y-%m-%d %H:%M:%S", pindexBest->GetBlockTime()).c_str(),
  Checkpoints::GuessVerificationProgress(pindexBest));
```

Fig. 22. Code snippet of calculating convenient chainwork representation, from the function SetBestChain() in main.cpp.

The variable called "nBestChainWork" is the total cycles for the current blockchain. In the codebase for Testcoin, the most valid blockchain fork is often described as the best chain, and so by extension the chainwork for the valid blockchain fork is called the best chainwork. Simply raise the natural base to the power of the product of the "log2_work" variable from Fig. 20 & 21 and ln(2). Essentially:

$$e^{x*ln(2)}, x = "log2work"$$

8) Reorganization Event: During a reorganization event, the peer-to-peer network will orphan all blocks that were produced beyond the common ancestor of competing blockchains. In this experiment, the attacker was disconnected at block height 274. The Testcoin network orphaned 11 blocks that belonged to the honest blockchain as seen in Fig. 23. This is easily done since both blockchains contained the hash of block 274 as the previous block in each of their 275th blocks. Also, during this reorganization event, Tx8b8 was orphaned also as the attacker blockchain spent the same inputs as seen in Fig. 24.

```
2022-10-31 03:46:03 REORGANIZE: Disconnect 10 blocks;
27d48f0dfc25c2d05d2c84b4406458575b67169913c0e8c3b60ca1ddbebc27bf..
2022-10-31 03:46:03 REORGANIZE: Connect 11 blocks;
..c55eac2d0aae9b3345d1fcd30ca9baa66baf66ac21931eb29515b9144f9d75e1
```

Fig. 23. Log event recording reorganization event.

```
2022-10-31 03:46:02 ignoring large orphan tx (size: 17300, hash:
8b81f320313f04685c88fff8c9ab82bc9cf7ffbf11d9676fa4355a63e35c590f)
```

Fig. 24. Tx8b8, the honest transaction, being orphaned by Miner1.

In Fig. 25 it's evident that RPC1, just prior to the block reorganization event, detected the invalidity of Tx44a and attempted to orphan it. However, this change was overwritten when the attacker's blockchain was swapped. Only peers that had consensus of the honest blockchain were forced to reorganize, so Miner1 and RPC3 had no such event take place as they were the attacker subnetwork.

```
2022-10-31 03:46:02 ignoring large orphan tx (size: 17299, hash:
44a13080d7ac089a5dcc062af75b77e1537c1135141e95dc36d2795c104d2007)
2022-10-31 03:46:02 received block 0876e6fdd20fe4403c0931aafe954204c369f8314477922035620f4e886aa5b3
2022-10-31 03:46:02 ProcessBlock: ACCEPTED
2022-10-31 03:46:03 received block 47379cb8bebf1a74723b416faa38718a165a63eb594a1ba6634472c1aa99d961
2022-10-31 03:46:03 ProcessBlock: ACCEPTED
2022-10-31 03:46:03 received block 7b0624efbc77d6cf13a2d82c989fda75c1c978187e29b1847af1ee1ca0bd1f6f
2022-10-31 03:46:03 ProcessBlock: ACCEPTED
2022-10-31 03:46:03 received block 913c1bf0f1fbf45999603e5c41edb454eb907c9ac33adeba206eb3d24475e686
2022-10-31 03:46:03 ProcessBlock: ACCEPTED
2022-10-31 03:46:03 received block 9db10700a11ed07564a264158103b109761adc7a6770c586fc8147906c59f759
2022-10-31 03:46:03 ProcessBlock: ACCEPTED
2022-10-31 03:46:03 received block abea35de4374beebf67a21c9d253fb97c4fa7193978f26f981368eb7e78a115b
2022-10-31 03:46:03 ProcessBlock: ACCEPTED
2022-10-31 03:46:03 received block bfd6d67f11182821959c67c12a62602de7dd832ed82bf1194adb875ba3a671f6
2022-10-31 03:46:03 ProcessBlock: ACCEPTED
2022-10-31 03:46:03 received block b7694ff668df214dc498c99f68e7f917a0bb816aabc26902ba47b8fe699a5bd3
2022-10-31 03:46:03 ProcessBlock: ACCEPTED
2022-10-31 03:46:03 received block 36a9123a575a8f85401d7de4790a3d01e7a51e990b2404a82a2d6de4cad35061
2022-10-31 03:46:03 ProcessBlock: ACCEPTED
2022-10-31 03:46:03 received block ac3be28feaa8fef7e15322f65c702d870f2e59bcbf399c965b5b9c3514d9ca6e
2022-10-31 03:46:03 ProcessBlock: ACCEPTED
2022-10-31 03:46:03 received block c55eac2d0aae9b3345d1fcd30ca9baa66baf66ac21931eb29515b9144f9d75e1
2022-10-31 03:46:03 REORGANIZE: Disconnect 10 blocks; 27d48f0dfc25c2d05d2c84b4406458575b67169913c0e8c3b60ca1ddbebc27bf..
2022-10-31 03:46:03 REORGANIZE: Connect 11 blocks; ..c55eac2d0aae9b3345d1fcd30ca9baa66baf66ac21931eb29515b9144f9d75e1
2022-10-31 03:46:03 Committing 229 changed transactions to coin database...
2022-10-31 03:46:03 SetBestChain: new best=c55eac2d0aae9b3345d1fcd30ca9baa66baf66ac21931eb29515b9144f9d75e1  height=286
log2_work=28.164929  tx=288  date=2022-10-31 03:33:43 progress=0.999970
```

Fig. 25. Log file snippet from RPC1 attempting to orphan Tx44a just before the reorganization event.

This concludes the experimentation portion of this thesis. The following chapter will discuss preventative measures that the attack in this chapter is vulnerable to. These preventative measures have all been implemented in real-world cryptocurrencies and most have been effective in either completely discouraging attacks or successfully repelling actual attacks on their respective cryptocurrency integrations. There are other

preventative techniques such as increasing required transaction confirmations before making funds available, but this method and others like it are inadequate given they affect some important tenant of cryptocurrencies such as scalability or decentralization.

V. Discussion of Results & Preventative Measures

Experiment II successfully executed a double-spending attack because it 1) used an attacker that controlled the majority of the blockchain and thusly was able to out-hash the honest network, 2) spent inputs more than once on the attacker and honest forks, and 3) was able to force a block reorganization event that saw the honest transaction orphaned. Hashrate-based double-spending attacks like Experiment II primarily rely on those three concepts to execute successfully and most preventative measures this thesis has researched that are worth noting will address at least one of those concepts.

A. Observer Network

A network outside of the peer-to-peer network that observes the network and provides security either actively through providing security services or passively through observation alone.

1) Masternode Network: This section will discuss masternodes as they are implemented in Dash [31] and its clones such as Raptoreum [38]. Masternodes are instances of a cryptocurrency's peer software that provide additional services to the network after meeting certain requirements. In both Dash and Raptoreum, masternodes are used to implement features such as transaction locking and chain locking. Temporarily established subsets of the masternode network are formed into quorums, which are used to verify both transactions and blocks. How masternodes deter double-

spending attacks is through collateral and quorum selections. In order to confirm a transaction or a block, a subset of masternodes must first be deterministically selected. If an attacker wanted to perform a double spending attack, they would need to control two things:

1. 51% or more of the total network hash rate
2. A high enough percentage of masternodes to control 75% of a given quorum.
    a. For Dash specifically, 75% of the LLMQ needs to agree on any messages being processed such as transactions and blocks, although voting is present on the Dash network along with other unrelated features.

To own that many masternodes, the attacker would also have to own enough collateral to fill those masternodes. Consider Raptoreum, which as of March 6, 2023, has a price of $0.001663 per coin according to [27]. The collateral for a Raptoreum masternode is 1.8M coins, which would cost approximately $2994.14 USD (United States Dollar). Generously assuming that controlling 51% of masternodes would be sufficient to perform a double-spending attack, as of March 6, 2023, there are 838 masternodes on the network. That would mean $2994.14 * (838 * 0.51) which comes out to $1,279,635.55 worth of Raptoreum as collateral. That is not to mention how much it would cost to control most of the network hashrate.

This thesis' experiment methodologies would be vulnerable to a masternode network as the attack would have to be more sophisticated. An attacker would not only need to control the majority of hashpower, but also control a high enough percentage of

masternodes to secure majority control of a quorum. In initiating the attack, the possibility of not successfully double-spending a specific transaction is much higher. That is to say, an attacker is more likely to lose those funds spent on the honest network if a majority quorum is attained. An attacker would likely need to perform a sustained attack or repeated attacks to ensure success which adds another layer of sophistication to the attack and also raises the likelihood of discovery and manual intervention by the network or core developers.

2) Block Notarization: Block notarization is the act of preserving blockchain states at regular intervals or manually. Komodo [23] offers a paid integration service to other cryptocurrencies in which it regularly records block hashes, transaction hashes, and any other blockchain data on its own publicly available blockchain. For example, the Pirate cryptocurrency [24] is currently integrated with Komodo's notarization service and it advertises double-spending attack protection because of this integration. When blocks and their transactions are preserved on the Komodo blockchain, they are considered final and any block reorganization events or transaction orphaning that attempt to overwrite this data are ignored.

Block notarization is effective against the experiments formed in this thesis due to the simplicity of implementation. Any peer receiving a message for block reorganization would need only check the external block notarization to determine validity of the incoming fork. The attacker would need to greatly reduce the amount of time its attacker spent offline, to only a few block production cycles. This would likely require the

attacker to own overwhelming majority of network hash power, so much so that the honest network would be incapable of producing a valid block while the attacker was disconnected due to how short of an attack window is available.

B. Chain Locking

Chain locking is the act of periodically preserving the blockchain. In Bitcoin and many other cryptocurrencies, checkpoints are taken by the core developers and released with new peer software updates [12]. Technically speaking, block hashes are hardcoded into the peer source code and are therefore set in stone. In this way, every block whose hash is included in the checkpoint is considered permanent and irrevocable. However, there is a more decentralized method that Dash employs called chain locking [25]. This is where the previously discussed LLMQ will periodically sign the most recent block of the blockchain, marking it as permanent [26]. This way, any attempts to cause a block reorganization for the block that was signed or any block before it on the block chain will be ignored.

C. Hybrid Consensus

The use of masternodes can fall under this category depending upon implementation. Also, there are other proofs of authenticity such as proof of stake, proof of authority and proof of capacity. Proof of stake is like masternodes in that users with more than a certain threshold of funds in their wallets can leave their software running and be used to authenticate transactions and blocks while receiving an award. Proof of stake is similar in proof of work in operation but utilizes funds in the process and much less electricity.

Proof of authority is typically implemented as trusted validators authenticating every transaction and block broadcast to the network. The Polkadot cryptocurrency uses proof of authority as part of its consensus mechanism. Proof of capacity is a means of using long term storage such as hard drives and solid-state drives to authenticate transactions and blocks. Chia is a cryptocurrency that uses proof of capacity.

The idea with hybrid consensus is that an attacker would need to control the majority of hash power as well as the majority of the other consensus mechanism(s) being deployed. For proof of stake, the attacker would need to control over half of the circulating supply of coins. Proof of authority would be much more difficult to attack since the attacker would, for a time, be required to work and work well within the confines of honest validation. Lastly, the attacker would need to control the majority of all long-term storage devices on the network. The biggest deterrent in hybrid consensus is cost prohibitive attacks and the infrastructure to support a multi-consensus network are so high that it is not worth it to do so.

D. Limitations of Experiment Results

This thesis' scope was limited to hash-based double-spending attacks. More sophisticated double-spending attacks exist such as integration with a Sybil network attack [46]. As such, further research could be conducted to find a solution that more wholly protects a cryptocurrency from any form of double-spending attack. To continue this thesis research would require implementation and integration of one or more preventative measures into a double-spending-vulnerable cryptocurrency and then

conducting more experiments to determine the best methods based upon observation. Further than prevention integration, continued research into what preventions have the least effect of decentralization, scaling, trustless transactions, and security would be ideal.

VI. CONCLUSION

Most proof of work cryptocurrencies are exceedingly difficult to attack through means represented in this thesis mostly due to the peer-to-peer network's size. To attack the network successfully would cost so much money that it's prohibitively expensive. After which point, if the attack is successful public opinion of the coin would fall, users would move their value off-chain to other cryptocurrencies, and the price of the coin would fall [15]. In short, it would cost the attacker a lot of money to execute and if they succeed the value of the coin would drop significantly if not kill the cryptocurrency entirely due to loss of market value, users, and community.

A double-spending attack is a much more serious threat to younger, smaller cryptocurrencies. Attackers will often target smaller cryptocurrencies around the time they first are listed on an exchange. It is this thesis' opinion that all preventative measures researched and mentioned are only effective with scale and that observer networks are the most intuitive. The larger the network, the more expensive it becomes to attack with a double-spending attack. There are other solutions, such as increasing the transaction confirmations required to before an input can be spent [6], but this makes cryptocurrencies difficult to use since it extends the amount of time for a simple transaction to hours or days. Other proposed solutions such as the Multistage Secure Protocol offered by [30] to detect and prevent double-spending attacks would likely only hinder network performance further.

Regarding distributed consensus, this thesis finds two important aspects: liveness and safety. Liveness is how likely the chain will continue moving forward no matter the conditions. Safety is guaranteed with the finality of blockchain data. This can be expressed in terms of blockchain reorganization events and transaction permanence. Whatever solution is deployed to prevent or discourage hashrate-based double spending attacks needs to ensure blockchain liveness and safety, while not treading on decentralization, security, trustless transactions, or to a lesser degree scalability is incredibly important in terms of cryptocurrency performance and application development.

REFERENCES

[1] A. M. Antonopoulos, *Mastering Bitcoin: Unlocking Digital Cryptocurrencies.* Sebastopol, CA, USA: O'Reilly Media, 2014.

[2] S. Nakamoto, "Bitcoin: A Peer-To-Peer Electronic Cash System," www.bitcoin.org, 2008. [Online]. Available: https://bitcoin.org/bitcoin.pdf (accessed Oct. 8, 2020).

[3] R. Skudnov, "Bitcoin Clients," B.S. Thesis, Inf Tech Dept, TUAS, Turku, FI, 2012. [Online]. Available: https://cryptochainuni.com/wp-content/uploads/Rostislav-Skudnov-Bitcoin-Clients.pdf

[4] C. Decker and R. Wattenhofer, "*Information propagation in the Bitcoin network*," IEEE P2P 2013 Proceedings, Trento, Italy, 2013, pp. 1-10, doi: 10.1109/P2P.2013.6688704.

[5] J. A. D. Donet, C. Perez-Sola and J. Herrera-Joancomarti, "The Bitcoin P2P Network," in *Proceedings of the 1st Workshop on Bitcoin Research (in Assc. with Financial Crypto 14),* 2014, pp. 87-102, doi: 10.1007/978-3-662-44774-1_7.

[6] M. Rosenfeld, "Analysis of Hashrate-based Double-spending," 2014. [Online]. Available: arXiv:1402.2009v1

[7] Bitcoin Developer. "Transactions — Bitcoin." Bitcoin Developer. https://developer.bitcoin.org/devguide/transactions.html?highlight=signature (accessed Dec. 6, 2022).

[8] Whiteboard Crypto, USA. *What is Proof of Work? (Cryptocurrency Explanation).* (Apr. 11, 2021). Accessed: Sept. 14, 2022. [Online Video]. Available: https://www.youtube.com/watch?v=XLcWy1uV8YM

[9] Bitcoin Developer. "Mining — Bitcoin." Bitcoin Developer. https://developer.bitcoin.org/devguide/mining.html (accessed Feb. 2, 2023).

[10] Bitcoin Developer. "Glossary — Bitcoin." Bitcoin Developer. https://developer.bitcoin.org/glossary.html#term-Fork (accessed Jan. 5, 2023).

[11] Bitcoin Wiki. “Bech32.” Bitcoin Wiki. https://en.bitcoin.it/wiki/Bech32 (accessed Feb. 18, 2023).

[12] Bitcoin Wiki. “Checkpoint Lockin.” Bitcoin Wiki. https://en.bitcoin.it/wiki/Checkpoint_Lockin (accessed Feb. 2, 2023).

[13] Bitcoin Wiki. “Protocol documentation.” Bitcoin Wiki. https://en.bitcoin.it/wiki/Protocol_documentation#getblocks (accessed Jan. 18, 2023).

[14] Bitcoin Wiki. “Difficulty.” Bitcoin Wiki. https://en.bitcoin.it/wiki/Difficulty#What_is_the_formula_for_difficulty.3F (accessed Jan. 25, 2023).

[15] Caleb & Brown. “The Game Theory of Cryptocurrency.” Caleb & Brown. https://calebandbrown.com/blog/the-game-theory-of-cryptocurrency/#:~:text=Basics%20of%20Game%20Theory,the%20reliability%20of%20distributed%20databases (accessed on Feb. 3, 2023).

[16] V. Buterin. “Toward a 12-second Block Time.” Ethereum Foundation Blog. https://blog.ethereum.org/2014/07/11/toward-a-12-second-block-time (accessed Dec. 18, 2022).

[17] Raptoreum Developers. “Mining Pool Setup – Raptoreum Docs.” Raptoreum. https://docs.raptoreum.com/docs/mining/miningpoolsetup/ (accessed Mar. 1, 2023).

[18] CoinWarz. “EtHash ASIC Miners.” CoinWarz. https://www.coinwarz.com/mining/hardware/ethash (accessed Feb. 1, 2023).

[19] CoinWarz. “Scrypt ASIC Miners.” CoinWarz. https://www.coinwarz.com/mining/hardware/scrypt (accessed Feb. 1, 2023).

[20] Freedesktop.org. “daemon(7) - Linux manual page.” Linux Manual. https://man7.org/linux/man-pages/man7/daemon.7.html (accessed Jan. 3, 2023).

[21] T. Dierks and E. Rescorla. “RFC 5246: The Transport Layer Security (TLS) Protocol Version 1.2.” RFC Series. https://www.rfc-editor.org/rfc/rfc5246 (accessed Dec. 3, 2022).

[22] P. Mockapetris. “RFC 1034: Domain names - concepts and facilities.” RFC Series. https://www.rfc-editor.org/rfc/rfc1034 (accessed Dec. 4, 2022).

[23] Komodo Platform. “Komodo Platform Overview.” Komodo Platform. https://developers.komodoplatform.com/basic-docs/start-here/about-komodo-platform/about-komodo-platform.html (accessed Jan. 23, 2023).

[24] Pirate Developers. “PirateNetwork/pirate: Pirate Chain (ARRR) - Untraceable, Anonymous, Private Cryptocurrency.” Pirate Github Repository. https://github.com/PirateNetwork/pirate (accessed Jan. 23, 2023).

[25] A. Block. “Chainlocks.” Dash Improvement Proposals. https://github.com/dashpay/dips/blob/master/dip-0008.md (accessed Oct. 23, 2022).

[26] A. Block. “Long-Living Masternode Quorums.” Dash Improvement Proposals. https://github.com/dashpay/dips/blob/master/dip-0006.md (accessed Oct. 23, 2022).

[27] Coin Market Cap. “Raptoreum Price (RTM).” Coin Market Cap. https://coinmarketcap.com/currencies/raptoreum/ (accessed Mar. 6, 2023).

[28] S. Langridge. “What Is Tokenomics.” Coin Market Cap. https://coinmarketcap.com/alexandria/article/what-is-tokenomics (accessed Mar. 1, 2023).

[29] S. Park, S. Im, Y. Seol and J. Paek, "Nodes in the Bitcoin Network: Comparative Measurement Study and Survey," IEEE Access, vol. 7, pp. 57009-57022, 2019, doi: 10.1109/ACCESS.2019.2914098.

[30] K. Nicolas and Y. Wang, "A novel double spending attack countermeasure in blockchain," 2019 IEEE 10th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON), New York, NY, USA, 2019, pp. 0383-0388, doi: 10.1109/UEMCON47517.2019.8992991.

[31] E. Duffield, H. Schinzel and F. Gutierrez, “Transaction Locking and Masternode Consensus: A mechanism for Mitigating Double Spending Attacks,” Dash Pay, Sept. 22, 2014. [Online]. Available: https://github.com/dashpay/docs/blob/master/binary/Dash%20Whitepaper%20-%20Transaction%20Locking%20and%20Masternode%20Consensus.pdf

[32] *Testcoin*. (2022). Testcoin Developers. Accessed: Oct. 1, 2022. [Online]. Available: https://github.com/QuodSumusHocEritis/Testcoin

[33] *Bitcoin*. (2023). Bitcoin Developers. Accessed: Oct. 1, 2022. [Online]. Available: https://github.com/bitcoin/bitcoin

[34] Coinbase. “What is crypto wallet.” Coinbase. https://www.coinbase.com/learn/crypto-basics/what-is-a-crypto-wallet (accessed Dec. 2, 2022).

[35] D. Boscovic, N. Chawla and D. Tapp, “Block Propagation Applied to Nakamoto Networks,” Dash Pay, Jun. 28, 2018. Accessed: Feb 20, 2023. [Online]. Available: https://www.darrentapp.com/pdfs/Block_Propagation_Applied_to_Nakamoto_Networks.pdf

[36] M. A. Imtiaz, D. Starobinski and A. Trachtenberg, "Characterizing Orphan Transactions in the Bitcoin Network," 2020 IEEE International Conference on Blockchain and Cryptocurrency (ICBC), Toronto, ON, Canada, 2020, pp. 1-9, doi: 10.1109/ICBC48266.2020.9169449.

[37] T. Nguyen-Pham, “Ghostrider Mining Algorithm,” Raptoreum Developers, 2016. Accessed: Dec. 8, 2020. [Online]. Available: https://docs.raptoreum.com/raptoreum-docs/raptoreum-papers/ghostrider-whitepaper/

[38] D. O. Morris and T. Nguyen-Pham, “Raptoreum: A Flexible System For the Creation and Transfer Of ‘Futures,’ Assets and Contracts,” Raptoreum Developers, 2020. Accessed: Sep. 1, 2020. [Online]. Available: https://docs.raptoreum.com/raptoreum-docs/raptoreum-papers/raptoreum-litepaper/

[39] N. van Saberhagen, “CryptoNote v 2.0,” 2013. Accessed: Mar. 7, 2023. [Online]. Available: https://github.com/monero-project/research-lab/blob/master/whitepaper/whitepaper.pdf

[40] Ledger. “Quality of Randomness.” Ledger. https://support.ledger.com/hc/en-us/articles/360010073520-Quality-of-randomness?docs=true (accessed Oct. 8, 2022).

[41] T. K. Sharma. “How Does Blockchain Use Public Key Cryptography.” Blockchain Council. https://www.blockchain-council.org/blockchain/how-does-blockchain-use-public-key-cryptography/#:~:text=Bitcoin's%20protocol%20uses%20what's%20called,to%20send%20and%20receive%20funds (accessed Oct. 15, 2022).

[42] A. Zola. “Definition: Hashing.” Tech Target. https://www.techtarget.com/searchdatamanagement/definition/hashing#:~:text=Hashing%20is%20the%20process%20of,or%20employ%20the%20original%20string (accessed Mar. 8, 2023).

[43] si14bet. “[ANN] [ICO] [Si14] Si14Bet - International Sports Betting Exchange P2P.” Bitcoin Forum. https://bitcointalk.org/index.php?topic=5237883.0 (accessed Mar. 8, 2023).

[44] CFI Team. “Minting Crypto.” CFI. https://corporatefinanceinstitute.com/resources/cryptocurrency/minting-crypto/#:~:text=Minting%20crypto%20is%20the%20process,can%20be%20minted%20this%20way (accessed Mar. 8, 2023).

[45] Raptoreum. “[ANN] Raptoreum - POW (GhostRider) | ASIC And FPGA Resistant | Mainnet Is Live.” Bitcoin Forum. https://bitcointalk.org/index.php?topic=5065848.0 (accessed Mar. 8, 2023).

[46] S. Zhang and J. H. Lee, "Double-Spending With a Sybil Attack in the Bitcoin Decentralized Network," in IEEE Transactions on Industrial Informatics, vol. 15, no. 10, pp. 5715-5722, Oct. 2019, doi: 10.1109/TII.2019.2921566.

[47] N. Eliason. “Tokenomics 101: The Basics of Evaluationg Cryptocurrencies.” Every Media, Inc. https://every.to/almanack/tokenomics-101 (accessed Mar. 22, 2023.)

[48] B2B In Pay. “What is Tokenomics? – How Does It Work?.” B2B In Pay. https://b2binpay.com/en/what-is-tokenomics-how-does-it-work/ (accessed Mar. 22, 2023).

[49] DERO Community, “DERO Project White Paper Build It, They Will Come,” DERO Developers 2018. [Online]. Available: https://github.com/deroproject/documentation/blob/master/Dero_Whitepaper.pdf

VITA

William Henry Scott III enlisted into active-duty United States Air Force immediately after high school and spent 6 years serving include two deployments to Iraq and Afghanistan. After separating from the military, he entered Stephen F. Austin State University and completed his Bachelor of Science in Computer Science with a minor in Mathematics in 2018. He earned his Master of Science in Cyber Security in the Spring of 2023.

Permanent Address: STEM Building, Suite 312
Nacogdoches, TX 75962

IEEE Reference Guide (v. 11.12.2018) regarding citations and in partial compliance with the *IEEE Editorial Style Manual for Authors (v. 11.12.2018)* regarding general style, superseded in part by SFASU Thesis Standards and Guidelines.

This thesis was typed by William Henry Scott III.