Investigation of Blockchain Network Security

Exploration of Consensus Mechanisms and Quantum Vulnerabilities

Jack Kelly, Michelle Lauer, Ryan Prinster, Stephenie Zhang

May 17, 2018

Contents

1 Introduction

As the value of cryptocurrencies like Bitcoin and Ethereum have risen dramatically over the past few years, public interest in investigating use cases of blockchain technology has increased as well. Blockchain technology eliminates the need for a trusted central authority, and instead relies upon consensus mechanisms to ensure the security of the network. In this investigation, we explore authentication methods common to most blockchain networks, then explore the two most popular consensus mechanisms: Proof of Work (PoW) and Proof of Stake (PoS).

Another technology that is steadily on the rise is quantum computing, particularly with companies like Google and IBM investing significant resources in development [13]. The authentication and consensus portions of blockchain networks rely upon the computational infeasibility of inverting certain functions, so quantum computers have the potential to cause significant disruption in this space. We will discuss some of the the security vulnerabilities that quantum computing would introduce to blockchain networks, how they impact PoW- and PoS-based networks differently, and potential solutions to these problems.

2 Blockchain Transactions

In the following section, we will discuss the Bitcoin transaction scheme, because this is the most popular scheme and many other networks use variations.

Keeping the blockchain authenticated and up to date is crucial to the Bitcoin architecture. Since the network is decentralized, it is especially important that every node is able to efficiently verify all transactions and blocks currently on the blockchain.

2.1 Keys

At a high level, a Bitcoin transaction consists of sending bitcoins from one wallet to another, via the address of the wallet. While this seems simple, their are many important subtleties that make the process quite complex. First off, the cryptocurrency "wallet" is really a misnomer designed to make transactions easier to understand, as there is no actual wallet anywhere. When a company like Coinbase says they are storing your Bitcoin in your wallet, this is being accomplished by storing a set of Elliptic Curve Digital Signature Algorithm (ECDSA) public and private key pairs. The public key of the pair is what is used to make your public address, which is a 58 byte number created in the following manner [31]:

- Version = 1 byte of 0 (zero); on the test network, this is 1 byte of 111
- Key hash = Version concatenated with RIPEMD-160(SHA-256(public key))
- Checksum = first 4 bytes of SHA-256(SHA-256(Key hash))
- Bitcoin Address = Base58Encode(Key hash concatenated with Checksum)

The private key of the pair is used to verify transactions.

2.2 Possession of Bitcoins

Another common misconception regarding transactions is the idea that you are actually sending Bitcoin from one wallet to another. Unlike traditional monetary transactions, where if Bob wanted to pay Alice he would take the physical money out of his wallet and hand it to her, in a Bitcoin transaction there exists no physical or even binary object that you transfer. Instead, if one were to say "I have bitcoin in my wallet", what actually is being said is that there is an unspent balance of bitcoin associated with an address that only I have the keys to.

This unspent balance, called an unspent transaction output (UTXO), is what determines how much bitcoin you possess. For example, if I have a public address that has a UTXO of 10 bitcoins, than it can be said that I have 10 bitcoins. To then transfer the bitcoin to someone else, I simply send

out a message to the network saying that I want to transfer the UTXO of my address, to a different public address. To allow other nodes to verify that this transaction is valid, I also have to sign the transaction using my ECDSA private key, and send the signature along with my public key [10]. This will allow other nodes to verify that I have the public key associated with my address and that I have the private key associated with my public key. Since all it takes to make a transaction is the public and private key, it is very important that these things be kept secure.

2.3 Executing Transactions

Another important subtlety of a Bitcoin transaction is that in every transaction, all of the UTXO of an address must be spent. That means that if Alice had 50 bitcoins of UTXO associated with an address, and she wanted to make a payment of one bitcoin to Bob, she would have to transfer all 50 bitcoins in the transaction. However, not all the UTXO must go to a single address – the transaction can be split up between multiple addresses. This means that there are two options for Alice to maintain the 49 bitcoin that she does not want to spend; she can either send it back to her original address, or she can create a new ECDSA key pair and a new public address, and send it that address. In general, it is highly recommended to do the second, especially with the possibility of quantum computing on the horizon [10]. The reasoning behind this will be discussed in section 5.3.

3 Proof of Work

We will discuss Bitcoin as an example of a digital currency which relies on the proof of work consensus algorithm. Note that while implementation details vary between currencies, the general methods and vulnerabilities discussed here apply to more than just Bitcoin.

3.1 Adding a Block

Each block on the Bitcoin chain consists of a header with block metadata, then a list of transactions. When a new block is "mined" and added to the chain, the miner receives an award of a number of newly minted Bitcoins, as well as whatever transaction fees were associated with the transactions contained in this block. These rewards and fees provide the incentive to mine despite hardware and electricity costs. The new Bitcoin reward started at 50 bitcoin, but halves every 210,000 blocks mined [29]. At the time of this paper, 12.5 bitcoins are awarded each time a new block is mined.

To mine a block, a hard cryptographic puzzle must be solved; for Bitcoin, this requires finding a double hash value that is below a specified threshold, feeding in the new block header (fixed) and a nonce as inputs. The threshold value for Bitcoin is dynamically adjusted every two weeks based on the amount of hashing power in the system, such that the estimated time between adding blocks is around 10 minutes [10]. As of May 2018, a suitable hash must start with 18 bytes of zeros.

3.2 Hash Scalability

Since the expected difficulty of the puzzle is directly correlated with the hashing power of the system, the expected probability that a single miner will discover a correct hash is the ratio between their hashing power and the hashing power of the system. This means that as the hashing power in the system goes up, it becomes more and more difficult for any individual miner to discover the correct hash and win the next block. This is a problem that has led to exponentially increasing growth in the hashing power of the system, because as the hashing power increases, all of the individual miners have had to individually add more hashing power to maintain their probability of discovering a block, which increases the total hashing power, and so on. This cycle will continue until it is no longer profitable to run the mining equipment, which will happen when the expected value from the rewards earned from mining a block is less than the electricity cost of running the equipment. While electricity is generally very cheap, there is a massive amount of

equipment that is constantly mining. Currently, Bitcoin and Ethereum together burn an estimated 1 million dollars in energy per day [32].

Due to the exponential growth of the total hashing power, the hashing power of the entire Bitcoin system as of April 2018 was 30 exohashes per second. This means that if an individual miner was able to mine at 1 Terahash per second, the probability that they would be the one to discover a block is .003%, meaning that they would be expected to discover a block approximately every 231 days. Since most miners will not want to wait two thirds of a year between rewards, it is common for miners to join together and form mining pools. In these pools, all miners work to solve the cryptographic puzzle, and if any of them solve it, they split the reward between all members of the group. This helps to smooth out the time between rewards for miners, but also makes the network more susceptible to various attacks, and leads to a less decentralized network. For example, it is estimated that 81% of the mining pools, and thus 81% of the total hashing power of the Bitcoin network, are controlled by Chinese mining pools [28].

3.3 Block Verification

After a miner discovers a suitable hash, they broadcasts the block to the rest of the network. Any other node on the network is then easily able to tell if the block is legitimate by verifying that the header has been created correctly and that the header correctly hashes to a satisfactory hash value. If a node decides that the block is legitimate, it will broadcast that block to the rest of the network, add it to the blockchain, and start working on mining the next block. If it decides it is not legitimate, it will disregard the new block, and continue to mine on the original blockchain.

If a group of nodes decides that a block is illegitimate, but another group decides that it is legitimate, a fork will occur, causing different nodes to have different views of the blockchain. Generally, forks resolve themselves relatively quickly once one chain becomes longer than the other. Since mining on a chain that does not become the main chain is a waste of effort, it is theoretically in everyone's best interest to try and mine on the longest chain in their view, as that is most likely to become the main chain. However, the fact that there are forks and that nodes rely on a distributed consensus, open up the door for a number of different potential attacks.

3.4 Common Proof Of Work Attacks

3.4.1 Overpowering the System

An important question when working with blockchain systems is: what happens if a single entity controls over half of the system? This attack, called a 51% attack, is of particular relevance in discussions of distributed consensus systems, because the security of these systems implicitly rely upon the assumption that no single (potentially malicious) entity can control the outcomes agreed upon by the whole system.

For PoW systems, a 51% attack occurs if a single entity controls more than half of the computational power of the system. If one entity were to gain this much computational power, they would be able to choose what transactions were valid or invalid, and would also be able to double spend bitcoins. Luckily, at the scale that Bitcoin has reached today, accruing this much computational power by a single entity would be so expensive that it is infeasible in practice. However, a 51% attack does not need to be carried out by a single person, but instead can be carried out by a single mining pool. While at this point, the largest Bitcoin mining pool "BTC.com" only controls 27% of the hashing power of the system, if the four largest mining pools decided to join forces and come together, they would have significantly more than 51% of the computational power and would be able to compromise the integrity of the entire Bitcoin system [15]. Fortunately, since miners generally have a vested interest in ensuring that the price of Bitcoin continues to increase, mounting an attack on the entire system is not really in their best interest.

3.4.2 Selfish Mining

As aforementioned, since mining a block requires so much computation power, mining is often done in pools, where a group (pool) of miners will work together to mine a block, and split the rewards. In selfish mining, a pool is able to get more blocks than their fair share by acting selfishly. Essentially, if the pool discovers a block, rather than publishing it, they instead keep it private and start working to find the next block. This leads to the rest of the network wasting computational power mining on the smaller chain, which is now out of date and will not ultimately be part of the main chain. Thus, this process of selfish mining gives the selfish mining pool a head-start on the next block if they are able to accumulate enough mining power [8]. This is an ongoing problem for the Bitcoin community, as well as many other cryptocurrencies. While their have been numerous ideas proposed to address this problem for Bitcoin, none have yet come to fruition.

4 Proof of Stake

In the following section we will discuss the Proof of Stake (PoS) consensus algorithm. We will explain the algorithm in a general sense, and then will discuss the Proof of Stake algorithm Casper, which will soon be in use on the Ethereum blockchain, as an example Proof of Stake algorithm. Note that while implementation details vary between PoS algorithms, the general methods and vulnerabilities discuss here apply to more than just Ethereum.

4.1 Adding a Block

Rather than rely on miners to solve a hard cryptographic puzzle, in a PoS algorithm block creation is done by nodes known as validators. Validators ultimately determine what ends up on the blockchain, since their job is to propose and vote on the next block which will be added to the chain. This is done via a consensus algorithm between all of the validators. Any participant in the blockchain who holds currency, in Ethereum's case Ether, can become a validator by sending a transaction which locks up their currency into a deposit, which is their "stake" in the algorithm. Validators are incentivized to increase their stake because more stake gives more weight in the validation process.

Even under the umbrella of proof of stake, there are various methods for selecting validators, the two most popular being chain-based PoS and BFT-style PoS (Byzantine Fault Tolerance). For chain-based PoS, a single validator is chosen pseudo randomly at every time step and given the right to make the next block. In BFT-style, multiple validators are chosen and given the right to propose a block; the rest of the validators in the system then vote on and choose a single block to be considered valid, and this block then gets added to the blockchain [32].

4.2 Verification Protocol

Casper is a partial consensus mechanism combining proof of stake algorithm research and Byzantine fault tolerant consensus theory. In the simple version of Casper, there is a fixed set of validators and a proposal mechanism that produces a block tree. Every validator will have a deposit that begins at the number of deposited coins and will rise and fall with rewards and penalties. Validators will broadcast a vote message that contains: [5]

- Public key of the validator
- Two checkpoints s and t
- Heights of checkpoints s and t: h(s) and h(t)
- Signature of $\langle s, t, h(s), h(t) \rangle$.

An individual validator must not publish two distinct votes for the same target height ($h(t_1) = h(t_2)$); additionally, it must not vote within the span of its other votes ($h(s_1) < h(s_2) < h(t_2) <$

$h(t_1)$). If a validator were to violate either of these in an attempt to attack the system, the validator would lose their deposit.

Although we talked about a fixed set of validators, Casper allows for the validator set to change by using what they call the dynasty of a block.

In Casper, a checkpoint is a block whose height in the block tree is an exact multiple of 100 (the genesis block is also a checkpoint). The "checkpoint height" of a block with block height $100k$ is k. The dynasty of a block b is the number of finalized checkpoints in the chain from the root to the parent of block b.

When a validator's deposit message is included in a block of dynasty d, the validator will join the validator set in the block of dynasty $d+2$ (the validator's start dynasty). When a validator wants to leave the set, the validator releases a withdraw message. If in a block of dynasty d, the validator leaves the validator set at the block of dynasty $d+2$ (end dynasty) [5].

Keeping the deposits of validators ensures that all validators remain accountable. If a validator violates a rule and is discovered, an economic penalties can be easily distributed by decreasing some or all of that validator's deposit. In Casper, the penalty for violating a rule or behaving immorally is losing the entire deposit [32].

4.3 Common Proof of Stake Attacks

4.3.1 Nothing at Stake

One concern with PoS mechanisms that is not an issue with PoW is the "nothing at stake" problem. For PoW systems, if a single entity were to try to simultaneously mine multiple chains, they would be splitting their computation power between those chains, and would thus gain no advantage over just using all of that power on a single chain. However, for PoS, there is no such similar implicit barrier disincentivizing someone from adding to multiple chains simultaneously. If validators are rewarded for adding blocks, then it is to their benefit to simultaneously add to all potential chains.

For chain-based algorithms, multiple solutions have been presented to solve this problem. Most of these solutions revolve around adding penalties which can be issued if a validator displays potentially malicious behavior. Some examples of criteria for issuing a penalty include finding a validator who has validated blocks on multiple chains, or who has validated a block on any chain other than the main chain. For BFT-style algorithms, two special types of rules called finality conditions and slashing conditions, ensure that if a given validator can be deemed beyond reasonable doubt to have misbehaved then their entire deposit is deleted.

4.3.2 Stake Grinding

Chain-based PoS algorithms utilize some sort of mechanism to randomly select the validator who makes the next block. This random selection is designed to be proportional to the amount of stake that each of the currently active validators has (i.e. 30% stake implies a 30% chance of being the next validator selected to make the next block). In Stake Grinding attacks, potential validators attempt to manipulate the system such that the theoretically random selection is biased toward giving them a higher chance of being selected. For example, with NXT, there was found to be a vulnerability is this "random" selection, because the chance of being selected at a certain block was dependent upon whether or that validator had already been selected for a previous block. Depdending on the opportunity cost of losing out on a potential block reward compared to the benefit of a higher likelihood of being selected to create the next block, validators could manipulate and optimize their behavior [32].

4.4 PoW vs PoS

There are a few large benefits of proof of stake algorithms over proof of work algorithms. One benefit is the decreased risk of centralization. In proof of stake, ten times more money gives exactly ten times more voting power; with proof of work, there are disproportional gains due to the fact that more money lets you get better equipment. Another benefit is that it is not necessary to issue as many new coins, because validators are not paying for expensive energy costs. In fact, under proof of work it is theoretically possible to even have negative fees to lower the supply of currency in circulation. In addition, the reduction of high energy costs from millions of miners, not only makes proof of work significantly more environmentally friendly, it also makes it much more scalable. Whereas proof of work is constantly being bounded by the cost of energy, proof of stake has no such limitation. Finally, where 51% attacks are a major problem with PoW as it is very possible for one group to gain more than half of the computational power in the system, a similar attack is much more difficult for currencies using PoS. While it is possible to mount a 51% on a PoS network, it requires one group or entity to have more than 51% of the currency in the system, which is much more difficult to gather than computation power.

5 Quantum

5.1 Proof of Work

5.1.1 Mining

In PoW systems, mining is a crucial part of the consensus mechanism which provides security and consistency to the chain. The hash function h that is currently used for Bitcoin mining is a composite function where $h(x)$ =SHA-256(SHA-256(x)). As of now, there is no efficient algorithm, classic or quantum, that can invert SHA-256. However, quantum enables more efficient mining by enabling more random nonces to be checked in parallel. At a high level, bitcoin mining requires a 256-bit outputted random number to be less than a certain threshold value. Quantum computing provides an advantage in finding this nonce because while regular bits are set to either 0 or 1, quantum bits allow values within this range to be explored simultaneously.

5.1.2 About Grover's Algorithm

One known quantum algorithm that can be used to explore a space of possible options is the Grover Algorithm, which results in quadratic quantum speedup over traditional non-quantum computers [25]. For example, if we want to explore a space of 16 bits, we would need to check 2^{16} options on a classical computer; on a quantum computer using Grover search, we could input 16 qubits, and then would need to check only $\sqrt{2^{16}}$ possibilities. This is done by leveraging superposition of quantum states. One condition of Grover's algorithm is that it is most efficient at exploring a finite unordered set. Since we are working with a set of 80 byte inputs to SHA-256 for mining, this condition holds.

Let's say we are using Grover's algorithm with n inputs (n qubits), which serves to explore a search space of size 2^n. The system is initialized using Hadamard transform such that all 2^n possible states are in equal superposition, which means that each possible state occurs with equal probability. Additionally, there is an amplitude of $\frac{1}{\sqrt{2^n}}$ associated with every possible configuration of the system. Using quantum superposition, every possible element in the search space can be considered simultaneously by manipulating the amplitudes such that each element occurs with probability above some threshold.

When using Grover's algorithm, it is necessary to assign some function f where $f(x) = 1$ if the search has been satisfied, and $f(x) = 0$ otherwise. With that in mind, to use Grover's algorithm for PoW mining, the function to use will be $f(x) = 1$ if $h(x) \leq t$ for some threshold t (where $h(x)$ =SHA-256(SHA-256(x), t = the hash threshold mentioned in section 3.1 below which a block is valid), and $f(x) = 0$ otherwise. The search space for the algorithm is all the possible nonce

values that combined with the previous blocks header and other block information, forms the 80 byte block header.

5.1.3 Grover Computation Analysis

If we assume the time for generating and checking an element of our search space is the same between classical and quantum computers, then a quantum computer will be able to explore a fixed-size space in the square root of the amount of time required for a classical computer. While this is a significant speedup, this can be accounted for by setting a more challenging threshold value. Let's say that we have a threshold set for a classical computer set such that the lower 2^{n_C} bits must be 0, and that this generally takes T_C time to solve; on a quantum computer, this would take only $T_Q = \sqrt{(T_C)}$ time. To make $T_Q = T_C$, we will need to increase the number of lower bits in our threshold such that $\sqrt{(2^{n_Q})} = T_Q = sqrt(T_C) = 2^{n_C}$. Thus, in the event of a quantum computer we can adjust the Bitcoin mining procedure such that for a given threshold requiring n_C bits, we will instead increase this to $n_Q = 2n_C$ bits.

However, note that this calculation is missing all of the constant-factor considerations about how fast quantum computers will be able to execute calculations relative to classical computers. For one thing, it is very unlikely that the time to generate and check elements in Bitcoin mining will be the same between classical and quantum computers, as there are extremely specialized classical computers that can do 13.5 TH/s. Further, quantum computers require more extensive error correction than classical ones, which adds more computation overhead to these computations [2]. While this latter problem can theoretically be solved by adding more qubits, it is a major obstacle for quantum computers in the near term.

5.1.4 Interim Period (mixed classical and quantum)

In this section, we have analyzed quantum effectiveness assuming the existence of quantum computers. There are two points to consider here: 1) quantum computing may not be viable, particularly as a consumer product, anywhere in the near future (see section 7), and 2) assuming that quantum computing does become a practical option for consumers, there will be some interim period where some people have quantum computers and others do not. At first glance it might seem that quantum computers will make classical computers seemingly ineffective, thereby disrupting the entire infrastructure that is in place. However, we believe that even if there is a time that quantum computers exist, the expense for one will be so high that the expected return from mining for a classical computer will still be greater than that of a quantum computer.

Another concern is that during this hypothetical period when quantum computers are entering the market, PoW blockchain infrastructures will be at an elevated risk for 51% attacks. Contrary to this, we find that given that the hashrate at the time of this paper already exceeds 30,000 TH/s on the Bitcoin network, even with a quadratic speedup, any malicious party would still need an unrealistic amount of quantum computing power to mount a 51% attack [14]. Overall, while the potential quadratic speedup that quantum computers may enable does pose an interesting problem of how to best maintain the hashing threshold for the network, we find that after taking into account hardware and economic constraints, we do not expect quantum computing to have a significant impact on the efficacy or security of Bitcoin mining.

5.2 Proof of Stake

5.2.1 Staking

As mentioned above, in contrast to Proof of Work schemes, Proof of Stake lacks the mining component and insteads boasts a staking component that may be privy to quantum attacks. As above, we will be looking specifically at Casper for our Proof of Stake scheme.

The standard Proof of Stake scheme is more vulnerable to quantum attacks than the Proof of Work scheme mentioned above. As indicated in section 4.2, to cast a vote, a validator must publish his

or her public key alongside every vote to ensure that the validator is indeed a validator and has the authority to publish a vote. If the public key of the validator is not in the validator set, the vote is considered invalid. As a result, a user's public key is effectively revealed as long as that user sits in the validator set and is voting on transactions (if the user stops voting, Casper institutes an "inactivity leak" which drains the deposit of any validator that is not voting for checkpoints). Thus, in Casper, as long as the user is voting, the public key is vulnerable to a private key reverse engineering attack [5].

While revealing a public key is currently not a problem, as there is no known classical algorithm to efficiently reverse engineer the private key from the public key, there is a known quantum algorithm that can do just that. This algorithm, known as Shor's algorithm and discussed in more detail in section 5.3, poses a significant threat to PoS schemes. If Alice were able to recover the private keys from the validator's public keys, then she would be able to create a valid signature and thereby a valid vote from any validator. By controlling the voting, she would have complete control over block creation, and what gets put on the block chain. With this power, she could easily double spend coins, as well as block any transactions that she does not find favorable. Further, she could break the validator rules, and cause any and all validators to lose their staked deposits. Finally and potentially worst of all, after Alice had control of the system, it would be incredibly hard for the honest nodes to take back the system, as she would be able to user her voting power to forge transactions and gain currency, which would allow her to stay in power.

5.2.2 Cold-Staking

To protect the public key, some Proof of Stake schemes, such as Particl, have implemented a feature called cold-staking. Cold staking consists of several components:

- Staking node
- Wallet (with coins)
- Cold-staking script

Whereas in the original scheme, validators give away their public key when they stake (validators sign their vote, so the public key needs to be there to validate), in cold-staking, they would set up a staking node. The staking node would sign transactions with its private key. The cold staking script allows the private key from the staking node to sign the transaction for cold staking transactions. As a result, during staking, only the public key of the staking node is revealed, not the public key of the wallet [22].

The wallet containing the coins remain offline, whereas the staking node will remain online (and contain 0 coins). Cold-staking thus makes determining the private key of the wallet nearly impossible. However, adversaries may be able to reverse engineer the private key of the staking node. While the staking node remains active, the adversary may potentially be able to vote on behalf of the wallet, thus the problems mentioned in the previous subsection are still applicable.

5.3 Authentication

5.3.1 Shor's Algorithm

Authentication is likely the largest area of the blockchain security that is vulnerable to quantum attacks. A majority of the security behind existing authentication mechanisms relies on problems which are difficult to solve, but easy to verify. Two of these problems, factoring and the discrete logarithm problem, are very widely used in different security algorithms. However, these problems are only considered to be "hard" using a classical computation paradigm. Theoretically, the advent of a quantum computer could make these problems computationally feasible, allowing an adversary to break any scheme that relies on the difficulty of these problems. The ECDSA authentication that Bitcoin uses (discussed in Section 2.1) is one such scheme, and is based upon the classical difficulty of the discrete logarithm problem. Therefore a vulnerability to the discrete log problem could potentially create large vulnerabilities to those using Bitcoin.

While quantum computers don't inherently let computations be done more quickly, specialized algorithms are able to take advantage of qubit superposition to do things impossible on classical computers. One such algorithm is Shor's algorithm. First, let us outline the principles behind Shor's algorithm, which is a quantum-based algorithm that is able to factor numbers and compute discrete logs in sub-polynomial time. To solve both of these problems, Shor's algorithm relies on a quantum sub-routine for finding the period (r) of a function $f(\cdot)$, such that $f(x) = f(x + r) = f(x + 2r)$ etc. The classical portion of Shor's algorithm reframes the problem of finding a discrete log or factorization of a number to be a period-finding problem [23] . With this in mind, the discrete log problem can be solved as follows: [7]

Let g be a generator of group $\mathbb{G}$ of prime order q. Given $y = g^k \in \mathbb{G}$, find the value of k.

- Consider the bivariate function $f : (x_1, x_2) \mapsto g^{x_1} y^{x_2}$.
- The period finding routine finds a pair (ω_1, ω_2) such that $f(x_1 + \omega_1, x_2 + \omega_2) = f(x_1, x_2)$.
- The solution to the discrete logarithm problem is then given by $k = -\omega_1/\omega_2 \pmod q$. One has $f(x_1 + \omega_1, x_2 + \omega_2) = f(x_1, x_2) \Leftrightarrow g^{\omega_1} y^{\omega_2} = 1_{\mathbb{G}} \Leftrightarrow g^{\omega_1 + k\omega_2} = 1_{\mathbb{G}}$ and thus $\omega_1 + k\omega_2 \equiv 0 \pmod q$.

Intuitively, the ability to quickly find a period makes it easy to find the modulus of a group, as the modulus is essentially the period of cyclic group. Similarly, it makes it easy to find factors of numbers, as the period of a number must be a factor if you take x to be zero.

The runtime of Shor's algorithm for discrete logarithm is $O(n^3)$, which is bottlenecked by quantum-classical modular exponentiation, rather than the quantum period-finding subroutine which runs significantly more quickly [20].

5.3.2 Quantum Resource Estimates

Pertaining to the actual reversal of the public key, though the computational "quantum advantage" is larger for EC discrete logarithms than for integer factoring problem, both require qubit and speed demands beyond the capabilities of current and imminent quantum computers. With a given number of bits n, the number of qubits needed is of order $O(n)$, and the number of gates is of order $O(n^3)$ [17].

The number of logical qubits for the controlled elliptic curve point addition in a simulation conducted by Microsoft Research [19] is:

$$9n + 2\log_2(n) + 10$$

and the number of Toffoli gates (in the quantum circuit) is:

$$448n^3 \log_2(n) + 4090n^3$$

Thus, to reverse a 256-bit ECDSA key, approximately 2330 qubits would be needed, and 1.26×10^{11} Toffoli gates. The simulation ran in 3848 seconds, which is approximately 1.075 hours. If the ECDSA key were to be 521-bits, approximately 4719 qubits would be needed, and 1.14×10^{14} Toffoli gates. The simulation ran in 42888 seconds, which is approximately 11.913 hours. [20]

With regards to the current progress on quantum computing processors, Google recently unveiled a 72 qubit quantum computer processor, which was considered a significant accomplishment for quantum computing [13]. As a result, because of the number of qubits necessary to run an effective attack, current cryptocurrency protocols remain secure in the near-term. However, assuming that one day there are quantum computers of 2000+ qubits, Shor's algorithm will pose many problems for cryptocurrencies, further discussed in the following subsection.

5.3.3 Quantum Vulnerabilities

Though theoretically Shor's algorithm would allow an adversary to efficiently reverse public keys to determine the corresponding private keys, the current utilization of Bitcoin has some inherent protections against coin theft.

As mentioned in Section 2.1, Bitcoin reveals a user's address for the user on the receiving end of a transaction, and that address is generated by taking the public key, and running SHA-256 and RIPEMD-160. Because the public key is run through these functions to generate the address, a user's public key is not revealed by the address. Instead, the public key is only revealed when the bitcoins associated with the public key are spent, making that the only timespan during which the private key is vulnerable to a quantum attack.

Most wallets, including the most used wallet Coinbase, generate a new address after every transaction. As a result, the coins move from a private, public key combination with a published public key to a new private, public key combination with a non-published public key, rendering the new private key secure against potential ECDSA attacks.

Because of safeguards such as automatic key generation, private keys in Bitcoin for the most part remain protected against public key reversals. However, during the interim between when a public key is revealed in a pending transaction and when the transaction is added to a block, a quantum adversary may be able to reverse the public key to discover the private key. Though typically a new block is added every ten minutes, after a transaction is placed it usually sits in a transaction pool for a few hours before actually making it into a block [26]. During this entire time, the private key is vulnerable to a quantum attack. If an adversary were able to recover this private key in that time, they would be able to submit another transaction from the original wallet that has a very high transaction fee, which would incentivize the miners to put that transaction in over the original, thus effectively stealing any currency from the hacked address.

What's worse, any transactions made at any point on the blockchain that didn't use automatic key generation, will be susceptible to a retroactive quantum attack. What this means is that if two years ago Bob made a transaction, and then reused his same address, even if powerful quantum computers don't exist for the next 20 years, if they do one day have the power to break ECDSA then at that point they will be able to steal any money that Bob has at the address that he reused. Since the entire blockchain history is publically available, an adversary with a powerful quantum computer will be able to look back and steal money from every single public address that has ever been reused. For Bitcoin, despite the recommendation that users always make a new address after every transaction, there would be thousands of addresses at risk. This is one of the strongest arguments that supports that cryptocurrencies, due to their public nature, should begin to switch to quantum resistant algorithms as soon as possible.

6 Quantum Resistant Signatures

6.1 QR Schemes

In anticipation of the arrival of quantum computing, quantum resistant cryptography has become a more extensively studied field. Current study is focused on a few specific fields.

- Lattice-Based Cryptography - These constructions are based on the presumed hardness of the lattice problem and are believed to be secure against quantum computers. These lattices problems study optimization problems in lattice groups, such as finding the shortest non-zero vector in the space given some set of basis vectors (SVP) or finding the closest vector to a given vector given some set of basis vector. (CVP)[11, 34]. There are currently no known quantum algorithms for solving lattice problems that perform significantly better than the best known classical (i.e., non-quantum) algorithms. [16]
- Code-Based Cryptography - An example of code-based cryptography is based on the hardness of the Learning with Errors (LWE) problem. There are no quantum attack against LWE are known (unlike the other major cryptographic hardness assumptions such as factoring or discrete logarithm). LWE typically leads to efficient implementations, involving low complexity operations. [18]
- Multivariate Polynomial Cryptography - Multivariate Public-Key Cryptography (MPKC) depends on the problem of solving a system of multivariate quadratic polynomials over a finite field (MQ problem). The most promising multivariate encryption scheme is the Simple Matrix encryption scheme where computations are done over one finite field and the

decryption process is the solution of linear systems; this scheme is very efficient. Other multivariate encryption schemes tend to be inefficient (due to guessing during decryption to ensure security). [1, 21]

- Elliptic Curves Cryptography - Another version studies a special class of elliptic curves called supersingular elliptic curves, which is over a field with large endomorphism rings, which gives the groups significantly different properties, yet could still be used similarly to the way elliptic curves are used today [3, 11]
- Hash-Based Signatures - Hash based algorithms have also been considered as a security mechanism against quantum driven attacks. As explained above, hash based one-way functions are considered to be secure against quantum algorithms, with an increase in the number of bits used.

6.2 Example: Lamport Signatures

A clear example of this is the Lamport signature [6, 33]. This is a candidate signature scheme for replacing vulnerable signature schemes in Bitcoin today.

6.2.1 The Scheme

A Lamport signature generates a preimage by randomly generating 2*256 numbers of 256 bits each. These comprise the private key. Hash each of these numbers, and together these comprise the public key.

To sign a message, hash the message m to give you a 256 bit hash. Recall our private keys, which are 2 sets of 256 random numbers. Call first 256 numbers are our "0" preimage, and the second are our "1" preimage. For each bit of this hashed message, take the corresponding number from the private key. For instance, if the first bit is a "0", take the first number in the "0" preimage, else take the first number in the "1" preimage. Do this for all bits of the hashed message, and this comprises our signature. So, we have 256 of 2*256 or half of the numbers available to the public, available in the signature.

To validate the signature, we simply hash each of these numbers and verify that they hash to the correct block of the public key, given a message. Intuitively, this is secure as a one-time signature because replicating it would require creating a hash collision. Similarly, we can see that this is again not secure with reused signatures, as replicating a signature would only require creating a fabricated message that hashes to something close to the hash of the original message, but not exactly it. However, not reusing keys between transactions is something already heavily recommended in Bitcoin, so it would not be a significant jump to go from recommended to enforced.

6.2.2 Implementation

The following code shows relevant methods for the Lamport signature scheme, consistent with Section 6.2.1 The full code is in the citations. [9]

```
// GenerateKey takes no arguments, and returns a keypair.
func GenerateKey() (SecretKey, PublicKey, error) {
    var sec SecretKey
    var pub PublicKey
    for i := 0; i < 256; i++ {
        token_zero := make([]byte, 32)
        token_one := make([]byte, 32)
        rand.Read(token_zero)
        rand.Read(token_one)
        block_zero := BlockFromByteSlice(token_zero)
        block_one := BlockFromByteSlice(token_one)
        sec.ZeroPre[i] = block_zero
        sec.OnePre[i] = block_one
```

```
        pub.ZeroHash[i] = sec.ZeroPre[i].Hash()
        pub.OneHash[i] = sec.OnePre[i].Hash()
    }
    return sec, pub, nil
}

// Sign takes a message and secret key, and returns a signature.
func Sign(msg Message, sec SecretKey) Signature {
    var sig Signature
    for i := 0; i < 256; i++ {
        bit := (msg[uint(i)/8]>>(7-(uint(i)%8)))&0x01
        if bit == 1{
            sig.Preimage[i] = sec.OnePre[i]
        } else {
            sig.Preimage[i] = sec.ZeroPre[i]
        }
    }
    return sig
}

// Verify takes a message, public key and signature, and returns a boolean
func Verify(msg Message, pub PublicKey, sig Signature) bool {
    for i := 0; i < 256; i++ {
        i := uint(i)
        bit := (msg[i/8]>>(7-(i%8)))&0x01
        if uint(bit) == 1{
            if sig.Preimage[i].Hash() != pub.OneHash[i] { return false }
        } else {
            if sig.Preimage[i].Hash() != pub.ZeroHash[i] { return false }
        }
    }
    return true
}
```

6.2.3 Bitcoin Scalability

A primary reason why Lamport signatures are not used today despite security benefits are due to scalability concerns. Using the current ECDSA, Bitcoin uses a 33-byte public key, and a maximum signature size of 73 bytes [30]. Using Lamport signatures instead requires 16KiB of storage for public key data, and 8KiB for signatures [33]. This means that the Lamport signature requires more than 200 times more space to store. This poses a significant barrier to its adoption since many argue that Bitcoin is already unscalable due to limitations in on-chain bandwidth.

However, these scalability concerns have motivated research into areas of speedup. Some of these improvements include modifications to the original protocol like Segregated Witness (SegWit), or off chain layer two additions like the Lightning Network [24, 27]. Additionally, Lamport Signatures have inspired other more compressed signature schemes with similar security mechanisms like the Winternitz One-Time Signature Scheme [4]. These simultaneous advancements will hopefully combat the additional strain that a more secure signature scheme would place on the network.

7 Conclusion

Due to the expectation of quantum computing, there has been a focus on the development of quantum-resistant cryptography as mentioned prior. Since its inception, there has been doubt as to the possibility of scalable quantum technology, with experts believing quantum states were too fragile and error-prone. With the introduction of quantum error correcting codes and threshold theorems, the concern of reliability and fault tolerance has gone down at the expense of requiring

more qubits for quantum computation. Although scientists have managed to develop quantum gates below a fault tolerance threshold of around 1%, there is undoubtedly uncertainty in the transition between laboratory experiments to scalable quantum computers. The resources used to control a qubit are diverse and classical; to create a scalable quantum computer, these classical resources must also be scalable, leading to complex engineering and infrastructure issues [12].

The future of scalable quantum computers is uncertain, with some of the most aggressive estimates predicting quantum computers powerful enough to break 2000-bit RSA by 2030 that would cost around a billion dollars [11]. This threat of powerful quantum computing poses serious problems for Proof of Work algorithms, as well as for any transaction algorithms that rely on a non quantum secure signature scheme. With this in mind, we feel that it is important that cryptocurrencies start addressing the risks that revealing non quanutm secure public keys presents. If action is not taken, then in the event that the quantum age comes sooner rather than later, millions of cryptocurrency coins owned by people across the world will be at risk.

References

[1] European Telecommunications Standards Institute White Paper No 8. *Quantum Safe Cryptography and Security; An introduction, benefits, enablers and challenges.* URL: https://portal.etsi.org/Portals/0/TBpages/QSC/Docs/Quantum_Safe_Whitepaper_1_0_0.pdf.

[2] Divesh Aggarwal et al. *Quantum attacks on Bitcoin, and how to protect against them.* URL: https://arxiv.org/pdf/1710.10377.pdf.

[3] D. Bernstein. *Introduction to post-quantum cryptography.* URL: http://www.pqcrypto.org/www.springer.com/cda/content/document/cda_downloaddocument/9783540887010-c1.pdf.

[4] Johannes Buchmann, Erik Dahmen, and Sarah Ereth. *On the Security of the Winternitz One-Time Signature Scheme.* URL: https://eprint.iacr.org/2011/191.pdf.

[5] V. Buterin and V. Griffith. *Casper the Friendly Finality Gadget.* URL: https://arxiv.org/abs/1710.09437.

[6] T. Dryja. *Lecture 3: Signatures.* URL: https://github.com/mit-dci/mas.s62/blob/master/slides/lec03-tadge.pdf.

[7] Crypto stack exchange. *Using Shor's algorithm to solve the discrete logarithm problem.* URL: https://crypto.stackexchange.com/questions/37018/using-shors-algorithm-to-solve-the-discrete-logarithm-problem.

[8] Ittay Eyal and Emin G¨un Sirer. *Majority is not Enough: Bitcoin Mining is Vulnerable.* URL: https://www.cs.cornell.edu/~ie53/publications/btcProcFC.pdf.

[9] *Go Lamport Sigs.* URL: https://github.mit.edu/prinster/LamportSigs.

[10] Bitcoin Developer Guide. *Block Chain.* URL: https://bitcoin.org/en/developer-guide#block-chain.

[11] S. Jordan L. Chen et al. *Report on Post-Quantum Cryptography.* URL: https://nvlpubs.nist.gov/nistpubs/ir/2016/NIST.IR.8105.pdf.

[12] T. D. Ladd et al. *Quantum computers.* URL: https://www.nature.com/articles/nature08812.

[13] Frederic Lardinois. *Google's new Bristlecone processor brings it one step closer to quantum supremacy.* URL: https://techcrunch.com/2018/03/05/googles-new-bristlecone-processor-brings-it-one-step-closer-to-quantum-supremacy/.

[14] Blockchain Luxembourg. *Hash Rate.* URL: https://blockchain.info/charts/hash-rate.

[15] Blockchain Luxembourg. *Hashrate Distribution.* URL: https://blockchain.info/pools?timespan=4days.

[16] D. Micciancio and O. Regev. *Lattice-based Cryptography.* URL: https://cims.nyu.edu/~regev/papers/pqc.pdf.

[17] J. Proos and C. Zalka. *Shor's discrete logarithm quantum algorithm for elliptic curves.* URL: https://arxiv.org/pdf/quant-ph/0301141.pdf.

[18] O. Regev. *The Learning with Errors Problem.* URL: https://cims.nyu.edu/~regev/papers/lwesurvey.pdf.

[19] Martin Roetteler et al. *Quantum Resource Estimates for Computing Elliptic Curve Discrete Logarithms*. URL: https://arxiv.org/pdf/1706.06752.pdf.
[20] M. Roetteler et al. *Quantum Resource Estimates for Computing Elliptic Curve Discrete Logarithms*. URL: https://arxiv.org/pdf/1706.06752.pdf.
[21] K. Sakumoto, T. Shirai, and H. Hiwatari. *Public-Key Identification Schemes Based on Multivariate Quadratic Polynomials*. URL: https://link.springer.com/chapter/10.1007/978-3-642-22792-9_40.
[22] P. Schmitzer. *Primer: Cold Staking on Particl*. URL: https://particl.news/primer-cold-staking-on-particl-98adbd4a0cac.
[23] P. Shor. *Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer*. URL: https://epubs.siam.org/doi/pdf/10.1137/S0036144598347011.
[24] Jimmy Song. *Understanding Segwit Block Size*. URL: https://medium.com/@jimmysong/understanding-segwit-block-size-fd901b87c9d4.
[25] Emma Strubell. *An Introduction to Quantum Algorithms*. URL: https://people.cs.umass.edu/~strubell/doc/quantum_tutorial.pdf.
[26] L. Tessler and T. Byrnes. *Bitcoin and Quantum Computing*. URL: https://arxiv.org/pdf/1711.04235.pdf.
[27] *The Bitcoin Lightning Network*. URL: https://lightning.network/lightning-network-summary.pdf.
[28] Jordan Tuwiner. *Bitcoin Mining Pools*. URL: https://www.buybitcoinworldwide.com/mining/pools/.
[29] Bitcoin Wiki. *Block*. URL: https://en.bitcoin.it/wiki/Block.
[30] Bitcoin Wiki. *Elliptic Curve Digital Signature Algorithm*. URL: https://en.bitcoin.it/wiki/Elliptic_Curve_Digital_Signature_Algorithm.
[31] Bitcoin Wiki. *Protocol Documentation*. URL: https://en.bitcoin.it/wiki/Protocol_documentation.
[32] Ethereum Wiki. *Proof of Stake FAQ*. URL: https://github.com/ethereum/wiki/wiki/Proof-of-Stake-FAQ.
[33] Wikipedia. *Lamport signature*. URL: https://en.wikipedia.org/wiki/Lamport_signature.
[34] Wikipedia. *Lattice-based cryptography*. URL: https://en.wikipedia.org/wiki/Lattice-based_cryptography.