Quantum Secure Cryptology Within Cryptocurrencies

Cowen Bland
United States Coast Guard Academy
New London, USA
Cowen.H.bland@uscga.edu

Kyle Boe
United States Coast Guard Academy
New London, USA
Kyle.W.Boe@uscga.edu

Torin Kearney
United States Coast Guard Academy
New London, USA
Torin.H.Kearney@uscga.edu

Felix Hernandez-Kossick
United States Coast Guard Academy
New London, USA
Felix.I.HernandezKossick@uscga.edu

William Maxam
United States Coast Guard Academy
New London, USA
William.P.Maxam@uscga.edu

Abstract

This paper outlines the development of a quantum-secure cryptocurrency system, aimed at addressing the cybersecurity challenges posed by quantum computing for the Coast Guard. The system integrates three post-quantum cryptographic algorithms, Dilithium2 for non-repudiation, Keccak for data integrity, and NewHope for confidentiality. Dilithium2 was successfully implemented to sign and validate blocks on the blockchain, ensuring non-repudiation and preventing fraud. Keccak was used to hash data, verifying the integrity of each block and ensuring that the data remains unaltered. NewHope was employed for secure key exchange between nodes, preventing interception of sensitive communications. Additionally, a Proof of Stake (PoS) consensus mechanism was incorporated, offering a more energy-efficient and scalable alternative to traditional Proof of Work. The successful integration of these algorithms demonstrates the feasibility of a quantum-secure cryptocurrency system, showcasing its potential for non-repudiation, confidentiality, and data integrity within the Coast Guard and other similar organizations in the face of emerging quantum threats.

Keywords

Cryptocurrencies, Post-Quantum Cryptography, Blockchain Security

1 Introduction

Recent executive directives, including a Presidential Executive Order focused on maritime cybersecurity, highlight the urgent need to future-proof the cryptographic foundations protecting the U.S. Coast Guard's operations [55]. In classical encryption, the difficulty of cracking the algorithms are usually based on number factoring or solving discrete logarithms, typically making use of large enough numbers that, with current computing technology, would take centuries to solve [54].

However, algorithms such as Shor's Algorithm have been found to easily solve these two problems, leading to insecure encryption standards once sufficient quantum computing power is attained [53]. The National Institute of Standards and Technology (NIST) has recently announced that common cryptographic algorithms, such as Rivest–Shamir–Adleman (RSA) and Elliptic Curve Digital Signature Algorithm (ECDSA), will be disallowed by 2035 in anticipation of advances of quantum computing [40]. This will require all government agencies, including the U.S. Coast Guard, to transition to newer, more quantum-secure algorithms to ensure their data is continually protected.

RSA and ECDSA are two of the most widely used cryptographic algorithms today. RSA is currently beind used in systems for digital signatures, secure communication protocols, encrypting messages, and software protection [14]. Similarly, ECDSA is being used to provide secure digital signatures [19]. If these two algorithms are able to be broken in the future, it would suddenly result in processes such as signatures being unsecure. Digital signatures such as those commonly used to sign Adobe documents could be forged. This would erode confidence and trust in society, as there would be no way to verify if information is coming from who you believe it is. Digital signatures are just one example of a critical function we rely on cryptography for. Other areas such as hashes and encrypted communication would be equally effected by current algorithms becoming unsecure, undermining the trust we have placed in online applications and websites to be secure and safe.

Our project proposes a tangible response to this impending challenge by integrating post-quantum algorithms into a prototype cryptographic coin. Using a coin allows for us to impliment a wide variety of algorithms into one system, to cover a wider problem scope. It also allows for us to better prove that multiple algorithms can work concurrently within a system.

This coin leverages three quantum-secure algorithms: Dilithium2 - A lattice-based digital signature scheme selected by NIST for post-quantum standardization, ensuring non-repudiation even against quantum-capable adversaries [27]. NewHope - A lattice-based key encapsulation mechanism (KEM) providing forward secrecy and secure key exchange resistant to quantum attacks [51]. Keccak (SHA-3) - A robust, quantum-resistant hashing function standard, ensuring data integrity throughout the system's lifecycle [38]. This paper will discuss the process of implementing these three algorithms in the form of a cryptocurrency. Necessary background regarding the structure and details of the system and algorithms will be provided, followed by a discussion of our objectives, then an overview of our system design, and finished with presenting our current results and conclusions.

As part of the project results, we successfully integrated Dilithium2 for block signing, ensuring non-repudiation through the use of multiple validators across the network. We also implemented Keccak for data integrity, automatically hashing blocks after transactions and verifying the hashes to ensure consistency and correctness. Additionally, NewHope was integrated to establish secure

communication between nodes, providing encrypted messaging and forward secrecy through quantum-resistant key exchange and encryption protocols. The system demonstrated successful functionality in all areas, with testing validating the effectiveness and security of the implemented algorithms.

2 Background

Modern cryptographic systems face many critical problems, however we have narrowed it down to three core issues: ensuring data integrity, achieving non-repudiation, and maintaining confidentiality [5, 28]. Data integrity involves confirming that information has not been altered, non-repudiation ensures that parties cannot later deny actions taken, and confidentiality protects the privacy and authenticity of messages exchanged over potentially unsafe networks [12, 35]. With the onset of quantum computing, many traditional cryptographic algorithms face potential vulnerabilities, mandating the exploration of quantum-resistant techniques that can uphold these critical system necessities in a post-quantum era [36, 42].

2.1 Non-Repudiation

Non-repudiation ensures that a party cannot deny having sent a message or performed a certain action [35]. Digital signatures serve this purpose by binding a message to a signer's key, creating verifiable proof of authenticity and origin [42, 52].

2.1.1 RSA. RSA has been a cornerstone of digital signatures and key exchange due to its simplicity and strong classical security especially with larger key sizes making it computationally expensive but well-understood and widely integrated into the current infrastructure [35]. However, RSA relies on the hardness of integer factorization, and quantum algorithms like Shor's algorithm could break RSA, rendering it insecure in a post-quantum world [5, 53].

2.1.2 Dilithium2. Dilithium 2, a lattice-based post-quantum signature scheme, leverages hard lattice problems believed to be resistant to known quantum attacks, offering a stable non-repudiation solution in a quantum scenario [17, 42]. Although Dilithium 2 provides strong security guarantees, it can have larger signatures and keys compared to classical schemes like RSA, potentially impacting performance and storage requirements [17].

2.1.3 Leighton-Micali Signature. Leighton–Micali Signature (LMS) scheme, a hash-based post-quantum signature, is stateless and relatively simple, providing strong security guarantees tied to underlying hash functions [32, 34]. LMS is well suited for systems requiring many signatures, as it can efficiently generate large numbers of one-time signatures [32]. However, performance and key management might be less flexible than lattice-based counterparts, and signature sizes may be larger, impacting certain instances [34].

2.2 Confidentiality

Confidentiality ensures the security and integrity of messages exchanged between parties [44]. Key exchange mechanisms allow two parties to establish a shared secret over an insecure channel, forming the foundation of secure protocols [35]. Once the two parties have the shared secret key, they can then use a symmetric encryption algorithm to secure there communication.

2.2.1 ECDH. Elliptic Curve Diffie-Hellman (ECDH) provides a fast and efficient method for secure key exchange, offering strong security against classical adversaries with shorter key lengths and better performance than earlier discrete log systems [44]. However, ECDH's security is tied to the discrete log problem on elliptic curves, which can be solved efficiently by a sufficiently large quantum computer, threatening ECDH's long-term security [50, 53].

2.2.2 NewHope. NewHope is a lattice-based key exchange protocol designed to be secure against quantum attacks, providing a path forward for secure communications in the post-quantum era [1, 50]. While NewHope offers strong theoretical security and good performance, it can be more complex to implement, and its parameters are still undergoing refinement as standardization efforts progress [1, 42].

2.2.3 AES-256-CTR. Advanced Encryption Standard (AES) is an encryption algorithm used to secure data by converting it into an unreadable format without the proper key [24]. AES is a block cipher, which means that the algorithm takes a fixed-sixe input and produces a ciphertext of the same number of bits [25]. AES is also a symmetric encryption algorithm, meaning it uses the same key for both encryption and decryption [3].

The Counter (CTR) mode is a confidentiality mode that features the application if the forward cipher to a set of input blocks, called counters, to produce a seuqnce of output blocks that are exlusive-ORed with the plaintext to produce the ciphertext [18]. CTR mode encyrpts blocks independently, enabling parallel processing [33]. This allows for CTR mode to be more software and hardware efficient compared with other modes of AES. Figure 1 below illsutartes how CTR mode works with AES-256 to both encrypt and decrypt data.

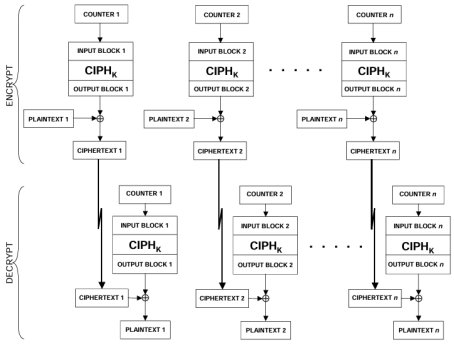

Figure 1: AES CTR Mode. Source: [18]

2.2.4 Crystals-Kyber. Crystals-Kyber is a lattice-based key encapsulation mechanism that provides quantum-safe confidentiality with efficient key generation and encapsulation/decapsulation operations [6, 43]. It was selected for standardization in the NIST Post-Quantum Cryptography Standardization Process, signaling strong

community confidence in its security and performance [43]. Despite this, ciphertext and key sizes can be somewhat larger than classical counterparts, and transitions to real-world applications must consider working implementation and performance trade-offs [6, 43].

2.3 Data Integrity

Data integrity is fundamentally about making sure that digital content remains unaltered from its intended state [21]. A primary tool for ensuring integrity is the use of hash functions, which produce fixed-size outputs to uniquely represent arbitrary data [20]. Since using these functions it is computationally infeasible to find any two distinct inputs that map to the same output, this allows for users to easily verify that data has not been manipulated. This is because if even one characters has been changed that it will result in a drastically different output of the hash function. [46]

2.3.1 SHA-256. SHA-256, a member of the SHA-2 family, is widely deployed for integrity checks, offering robust security against current classical attacks and providing efficient implementation [20]. However, SHA-256 is known to be susceptible to potential future quantum attacks, as Shor's algorithm could undermine the underlying discrete logarithm assumptions related to some associated systems, which in turn reduces its long-term security posture [36].

2.3.2 SHA-3 (Keccak). SHA-3, known as Keccak, provides a different internal structure based on sponge functions and has shown resistance to known classical cryptanalysis, offering a strong security margin and flexibility [15, 21]. Yet, while SHA-3 is considered a good candidate for post-quantum robustness, its performance can be slower in some implementations, and the general quantum threat model suggests that all classical hash functions might experience reduced security levels, although SHA-3's design may still provide stronger protection than older hashes [36].

2.3.3 Blake3. BLAKE3 is a modern hash function optimized for speed, parallelism, and simplicity, offering rapid hashing performance on various platforms [49]. It inherits cryptographic strength from the BLAKE2 family and introduces a highly parallelizable tree-based design that scales efficiently on modern hardware [49]. However, as a relatively newer version, it lacks the long-standing scrutiny of SHA-2 or SHA-3, and while no known attacks exist, further analysis over time is needed to confirm its durability against future cryptanalytic advances [2, 49].

2.4 Consensus Mechanism

A consensus mechanism is a process by which a distributed network agrees on the state of a shared ledger, such as a blockchain [13, 23]. Consensus ensures that all honest participants share a consistent view of the system's history, thereby preserving integrity, trust, and reliability [5, 8]. In a system aiming to provide quantum-resistant cryptographic assurances, maintaining a secure and tamper-proof record of transactions is essential. The consensus mechanism ensures that, even as cryptographic primitives evolve to be quantum-safe, the ledger's integrity and agreement properties remain intact [29, 41].

2.4.1 Proof-of-Stake. Proof of Stake (PoS) chooses validators based on the amount of stake they hold, reducing the need for expensive computations and promising better energy efficiency [29]. However, PoS can introduce new attack vectors, such as "nothing at stake" problems, and relies heavily on economic incentives rather than purely computational costs [4]. [13, 23].

2.4.2 Proof-of-Work. PoW requires nodes to perform energy-intensive computations, achieving strong security through economic disincentives for dishonest behavior [37]. The downside is high energy consumption, potential centralization of mining operations, and vulnerability to quantum acceleration if certain hash problems become easier with quantum computing [9].

2.5 Cryptocurrency

A Cryptocurrency is a digital unit within the system, which is cryptographically sent from one blockchain network user to another. These units are transferred from one user to another by using digital signatures with asymmetric pairs [45].

2.6 Quantum Computer

Quantum computers are fundamentally different from classical computers. They process data using qubits, which can represent a combination of 0 and 1, a property known as superposition [7]. This enables quantum computers to handle large amounts of data more quickly, allowing them to solve the mathematical problems that current cryptographic algorithms rely on exponentially faster [56].

Figure 2 illustrates this concept. An RSA key that would take a classical computer 10^{28} hours to crack could be broken by a quantum computer in approximately one hour. This suggests that quantum computers could potentially compromise any system protected by modern cryptography within hours.

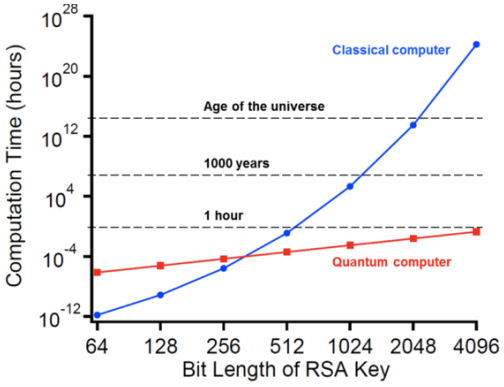

Figure 2: A comparison of how long it would take to break different RSA key lengths using classical and quantum computers. Source: [31]

2.7 Lattice-Based Cryptography

Lattice-based cryptography derives its strength from the mathematical complexity of lattice problems, which are based on multi-dimensional geometric structures.

A lattice is a set of points in an n-dimensional space with a periodic structure. This can be seen in Figure 3. Given n-linearly independent vectors $\mathbf{v}_1, \ldots, \mathbf{v}_n \in \mathbb{R}^n$, the lattice generated by these vectors is the set of all integer linear combinations of the vectors. [10, 11] A basis is a set of vectors that generate the lattice and are all linearly independent such as the two shown in Figure 3.[26]

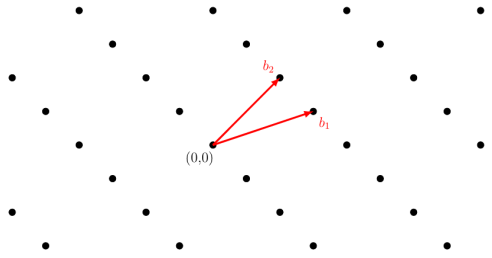

Figure 3: An example of a lattice in $\mathbb{R}^2$ with two basis vectors $\mathbf{b}_1$ and $\mathbf{b}_2$ drawn in red. Source: [26]

Using lattices as the foundation, mathematical problems can be created and used as the basis for cryptographic constructions. One such problem widely used is the Learning With Errors (LWE) problem. In the LWE problem, the goal is to find a secret vector:

$$\mathbf{s} = (s_1, s_2, \ldots, s_n) \in \mathbb{Z}^n$$

To do this, you are given a series of equations in the form:

$$a_1 \cdot s_1 + a_2 \cdot s_2 + \cdots + a_n \cdot s_n + \text{error} = \text{noisy output}$$

where $a_1, a_2, \ldots, a_n$ are known input values, $s_1, s_2, \ldots, s_n$ are secret values, and the error is a random value.

Normally determining s would be easy provided that there were enough equations. However, by adding a random error into the equations, it becomes much more difficult to solve [26, 47].

The Learning with Errors problem is used in Key Agreement algorithms like NewHope to agree on a shared key without directly transmitting it, because an eavesdropper can't solve the noisy equations to extract the shared secret. It can also be used in signature algorithms like Dilithium2 where solving involves proving knowledge of s without revealing it.

2.8 Quantum Resistant Ledger

Currently, in the realm of quantum-secure cryptocurrencies, one notable example is the Quantum Resistant Ledger (QRL) QRL utilizes the Extended Merkle Signature Scheme (XMSS) to have users sign the transactions, thus processing them in a quantum-secure manner [30]. XMSS is a hash-based signature scheme, meaning that the assumption used to guarantee the security of XMSS is that it's hash function is secure and collision resistant. Our project is different from QRL for a multitude of reasons, but the main one being a difference in goals. Our projects sole goal is to be a proof of concept for the effectiveness of quantum secure algorithms. While QRL is also focused on providing a quantum-resistant ledger, it does have monetary value attached to it. The algorithms we are using are also different from QRL. QRL uses XMSS for digital signatures and SHA-2 for hashing. Our project uses Dilithiium2 for Digital signatures and SHA-3 for hashing. Dilithium2 and XMSS are especially different as one is latticed-based while on is hashed based. We also put a larger focus on encrypting communication within our system while QRL is not focused on that area. This will be the first implementation of our selected algorithms into a cryptocurrency.

3 Objective

Our overarching goal in this project is to prove the feasibility of quantum-secure cryptography by implementing post-quantum algorithms into a network using classical resources. We aim to successfully implement Dilithium2, Keccak, and NewHope in the system, proving they can coexist and solve the problem space of ensuring non-repudiation, data integrity, and confidentiality [5, 28]. For the purpose of this project we are using a self made cyrptocurrency as the testing environment for our implementation. A cyrptocurrency is the most ideal system to test on because of the wide array of algorithms required for a functional system. In order send and receive transactions it requires cross network communication that we can attempt to secure. It also requires a hashing algorithm when blocks are created. Finally it requires a digital signature alogorithm to have validators verify the block. This allows us to demonstrate a successful implementation of multiple different algorithms in one system and prove their ability to coexist in one functional and secure system. Though we aim to provide a full proof-of-concept with all the algorithms, each algorithm can be independently selected and utilized by the sponsor or other entity to protect various forms of their data or communications. Essentially this allows for future users to be confident both in our system as a whole, but also in each individual algorithm. This allows for a wide array of possibilities for utilizing our chosen algorithms either independently or together in future projects.

3.1 Cryptocurrency

The foundation of our project is having a functional system that mirrors all the actions taken by traditional cryptocurrencies such as Bitcoin. We cannot begin to secure a system until we have all the necessary components functional. The main operations we need this crytpocurrency to do are the sending and recieving of transactions, block creation, block validation, and block distribution. As shown in Figure 4, our system begins when a user submits a transaction. This transaction will then be sent over the network to the central node. Once the central node has five transactions they will be group together into a block. This block is then hashed, validated, and redistributed to all the nodes on the system.

3.2 Data Integrity

To ensure data integrity, we will implement the Keccak, or SHA-3, hashing algorithm, which was approved by NIST in FIPS 202 [21]. As mentioned above and in 4, after a block has been created, we will implement Keccak to hash that block, providing a unique output that corresponds to the contents of the block. This gives us a baseline hash output that represents the data within the block. Validators will later recompute the blocks hash once again using Keccak and verify the hashes match themselves before signing to ensure the data is accurate and does not have errors or evidence of tampering.

3.3 Confidentiality

To ensure confidentiality, we will implement the NewHope algorithm, which, though not approved by NIST, was selected as a Round 2 Finalist in the Post-Quantum Cryptography Standardization Process [39]. As shown above and in 4, everything that is sent across the network is going to be encryted to ensure no one can intercept confidential information. We will use NewHope to have the nodes agree on a shared key, which will then be used to secure the communications using AES-256-CTR. Once implemented, this will prevent man-in-the-middle attacks from snooping on the data packets being sent across the network and being able to glean credentials and transaction data for their own purposes.

3.4 Non-Repudiation

To ensure non-repudiation, we will implement the Dilithium2 signature algorithm, which has been approved by NIST in FIPS 204 [22]. After a series of transactions have been made, forming a ”block”, we will implement Dilithium2 to allow validators, or superusers, to digitally sign the block, certifying that all transactions are correct. As the digital signature will securely come from the validator and ensure that they are the ones who signed the block, not someone else who was able to crack their signature, non-repudiation will be ensured.

3.5 Resource Constraints

One of the primary resource constraints we have run into is a lack of physical hardware to efficiently build a network needed to run the system. However, we are getting around this constraint by using virtual machines, which can be safely run on the hardware we do have and can scale up to support multiple nodes as needed. We also do not have a source of outside funding for this project, so we are required to primarily utilize open-source programs. Fortunately, all three of our algorithms are free and open-source, which is also beneficial for our sponsor or another entity who desires to change their current encryption algorithms.

4 System Design

This section outlines the key design decisions made in the development of the quantum-secure cryptocurrency, focusing on the selection of algorithms for non-repudiation, data integrity, confidentiality, and consensus mechanism. One determening factor that applies to all algorithm chosen is that we wanted it to be a winner or finalist in NIST’s post-quantum Standards. This allows us not only to be confident in its ability to protect from quantum threats, but also provides us with a lot of documentation and research that can help us impliment these algorithms into our system. Another factor was that these algorithms are all based off of lattices. As mentioned in our background, this is because as you add more dimensions needed to the math needed to break algorithms, it becomes increasingly hard to quantum computers. Since lattice math uses mutli-dimensional geometric structures it is a reliable base for our algorithms. Each decision was based on a thorough comparison of available alternatives, considering both their security and practical integration into the system. We also will discuss how each alogorithm is implemented into our system to provide the needed functionality.

4.1 System Architecture

Our system functionality follows the process described in Figure 4. It begins when a user submits a transaction to another user. That transaction is securely transmitted across the network using AES-256 in CTR mode. Once five transactions are collected, they are combined into a block, which is then hashed using Dilithium2. When a block is created, three users from across all nodes are selected to be validators. Validator selection uses staking as its primary factor. The three users who have the most coin staked will be chosen as validators. If no users have any stake, or multiple users have the same amount staked, than the system randomly choses the validator. Validators subsequently sign the block, confirming the accuracy of the data. All three signatures are attached to the hashed block. Finally, the block is distributed to all nodes in the system.

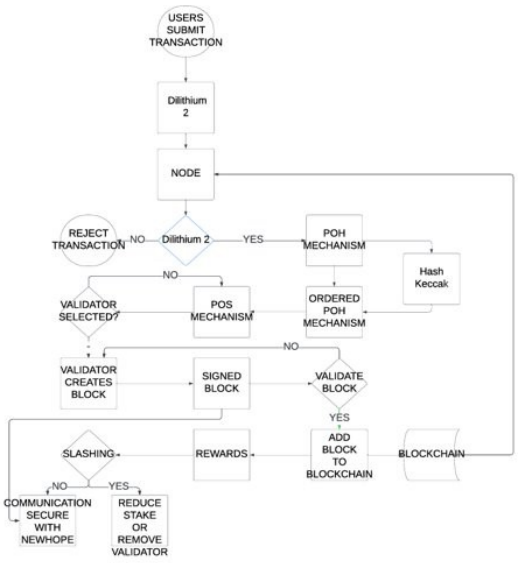

Figure 4: System Architecture Overview

4.2 Non-Repudiation

In order to ensure that blocks are properly being validated and that we can be confident in validate blocks we selected Dilithium2, a lattice-based digital signature algorithm that is recognized as secure against quantum computing threats by NIST. This will ensure that all validator signatures attacked to a validated block are coming from our trsuted users and we can be confident in their authenticity. One of the primary factors behind this choice was its compatibility with Keccak, our chosen hashing algorithm. The integration of Dilithium2 with Keccak requires minimal adjustments, ensuring a smoother implementation.

As mentioned in our background section Quantum Resistant Ledger, a quantum-resistant algorithm currently on the market uses the Extended Merkle Signature Scheme (XMSS) as its digital signature algorithm. We considered using this algorithm, but ultimately ended up chosing Dilithium2 instead. XMSS is a statefull hash-based signature scheme that relies on the security of hash

functions. Dilithium2 instead relies on lattice-based cryptography which makes it more resistant to quantum attacks compared to XMSS. Dilithium2 also produces smaller signature which allows ot to scale better for systems that require frequent signing and verification such as a cryptocurrency. We also preffered the stateless design of Dilithium2 since it allowed for us to generate signatures independitently without maintaining any state. Finally, the goal of our project is a proof of concept to advance research. Chosing algorithms that have not been integrated into a cryptocurrency already would provide more value than simply reproducing results that QRL has already been successful with.

We also considered Leighton-Micali Signatures (LMS), a hash-based signature scheme that utilizes Merkle trees for signing. While LMS is quantum-resistant and secure, its implementation would require significant changes to the codebase due to its reliance on a different hashing algorithm. Additionally, LMS generates larger signatures compared to Dilithium2, which increases computational overhead and storage requirements. Given the ease of integration and the reduced resource demands of Dilithium2, we determined it to be the most suitable choice for our project.

4.3 Data Integrity

To ensure our blocks are accurate and maintain data integrity, we chose Keccak, a member of the SHA-3 family and one of the most trusted post-quantum secure hashing algorithms. Keccak provides a robust defense against attacks such as hash collisions and preimage attacks. It is efficient and has been widely adopted in cryptographic systems, including blockchain implementations, making it an ideal choice for our cryptocurrency.

An alternative we evaluated was BLAKE3, which offers faster performance than Keccak in certain contexts. However, integrating BLAKE3 into our system would require significant changes to the codebase. Additionally, while BLAKE3 is secure, its adoption is not as widespread as Keccak, and it did not offer enough advantages in terms of efficiency to justify the extra complexity of integration. Thus, Keccak remained the optimal choice for ensuring data integrity in our system.

4.4 Confidentiality

To ensure confidentiality, we selected NewHope, a key encapsulation mechanism (KEM) based on the Ring Learning With Errors (RLWE) protocol. NewHope offers secure key exchange and is efficient to implement. Its security is well-established in the post-quantum cryptographic community, and it integrates well with our chosen algorithms. This makes it an ideal candidate for enabling secure communication within the blockchain network.

We also considered CRYSTALS-KYBER, another post-quantum KEM based on the Module Learning With Errors (MLWE) protocol. While CRYSTALS-KYBER offers slightly better efficiency than NewHope, it requires more complex implementation and integration into the existing system. Given that the performance gains were minimal and that NewHope already provides strong security with a simpler implementation, we opted to use NewHope for our secure communication needs.

4.5 Consensus Mechanism

For the consensus mechanism, we selected Proof of Stake (PoS), a mechanism employed by many modern blockchains, including Ethereum. In PoS, validators are chosen to create new blocks based on the amount of cryptocurrency they ”stake” as collateral. Validators who act honestly are rewarded, while dishonest validators risk losing their staked tokens. PoS is more energy-efficient and scalable compared to Proof of Work (PoW), which is used by Bitcoin. Additionally, PoS can be more easily adapted to quantum-secure cryptography, making it a good fit for our quantum-resistant cryptocurrency. Although PoW provides strong security, it requires significant computational resources for mining, which leads to high energy consumption. Given the environmental concerns and scalability issues associated with PoW, PoS was selected for its efficiency and adaptability to the quantum-secure ecosystem.

4.6 System Architecture

Our system is currently a centralized network comprised of multiple Debian virtual machines that are acting as our nodes. There is a central node which all other nodes communicate with. Logging of transactions, block creating, and user account information is all stored on this central node. Additionally, we use a centralized key repository on our central node. Every user on our system will have a secret private key that is stored on whichever node they are utilizing. However, they will also have an associated public key that anyone can access and view. These public keys for all users are stored on a repository on the central node. This allows for all public keys to be stored together for ease of use and access. Currently we have five additional nodes used for sending and recieving transactions, as well as storing our blockchain. Our consensus mechanism is Proof of Stake (PoS). In PoS, validators are selected based on how much cryptocurrency they are willing to stake as collateral. This method is responsible for validating transactions and adding new blocks to the blockchain. Together, these technologies ensure that our system is secure, energy-efficient, and resilient to quantum computing.

5 Results

Throughout the project we created fully functional cryptocurrency with all essential components including blockchaining and a consensus mechanism. The main priority of our project however was to be a proof-of-concept that it is possible to secure a system from quantum threats, with only using classical resources. All three of our chosen algorithms were successfully implemented into our system and tested to ensure functionality.

5.1 Non-Repudiation

Dilithium2, a digital signature algorithm, solved the issue of non-repudiation within our system. For this algorithm to work every user has to have their own unique public key, private key, and signature. To test this, we looked at each user and ensured that had the appropriate keys and signature associated with their account. In Appendix 7.1, you can see what these values look like when printed. This appendix shows the values for ’user11’ within our system, but all accounts will look similar. Then we compared the

values of these keys and signature to other users on our system using an online difference checker, which verified that each user had unique values. We also looked at the bit length of these values and compared them to the expected lengths that Dilithium2 generates. Appendix 7.1, also shows the bit length of the keys and siganture. On the official GitHub of Open Quantum Safe, the foundation we got the open source Dilithium2 code from, there is a chart showing what the bit lengths of these values for Dilithium should be [48]. This table can be seen in Appendix 7.2. Since the lengths of keys and signatures for users on our system matched Open Quantum Safe's expected lengths, we were confident that Dilithium2 was properly generating keys and signatures within our system.

Once we have a block that is ready to be signed, three validators will be selected across all accounts on all nodes using the method described in our system design section. All three validators are able to concerrently verify and sign the block if they compute the hash as accurate. We tested this by signing multiple blocks with a different order of validators. These validator signatures are then attached to the block and visible to everyone of the network. To test that user signatures are properly being appended to our blocks we ran our program and had all three validators verify it. In this process all three validator's signatures got appended to the original block. We were able to print the value of the block with the signatures attached, showing that all three signatures are being appended. We then compared the signatures attached to the block to the original signature of the users using a difference checker, confirming that the signtaures are accurate.

5.2 Data Integrity

Keccak, a hashing algorithm, solved the issue of data integrity within our system. To test that this algorithm worked we utilized a set of Keccak test vectors that are published. The two test vectors we used were the string 'abc' and the empty string ''. In Appendix 7.3, you can see the output we recieved for those two inputs. We then compared our outputs to the expected outputs seen in Appendix 7.4. Both of the two test vectors were identical, proving that Keccak was generating the correct hashes.

We then had to verify that Keccak was properly functioning once fully implemented. Since Keccak is used to sign blocks, we had Keccak has the same block multiple times. We compared the outputted hashes and ensured that they were identical. This allowed us to be confident that Keccak was consistently generating the correct hash. We also verified this by modifying the block being hashed, and ensuring that the resulting hash was different.

5.3 Confidentiality

NewHope, a key agreement algorith, was the foundation of solving confidentiality within our system. To test NewHope we began by starting with a simple program on two nodes that would have the nodes agree on a key. We ensured this worked properly by comparing the key generated on the sending node, with the key on the client node. We then repeated this process, switching which node began the interaction. When both nodes were consistently agreeing on the correct key we were confident NewHope worked on a small scale.

We then scaled up the implimentation, including NewHope in our program that inlcuded other coin functions. Similar to before we esnured that keys were still being agreed upon properly. We then addded our encryption algorithm, AES-256-CTR, into the system. Using the agreed upon shared key from NewHope, AES-256-CTR would encrypt messages being sent across our network.

We began by adding this functionality into our login and create account functionlity. We believed that these were the most crucial areas since secure account credentials were bing transmitted. To test NewHope and AES-256-CTR were both working, we would create a new account on one node with a new username and password. These credentials would then be encrypted using the NewHope keyand sent over the network, being decrypted on the reciving node using the shared key from NewHope. We confirmed that the username and password decrypted properly by checking what credential was added to /etc/passwd, where we store credentials.

We also tested to ensure that while in trasmission, the data remained encrypted. To do this we ran Wireshark while sending credentials, and looked at the packets captured. As seen in Apendix 7.5, the data remained properly encyrpted while being sent over the network.

We then scaled up the implementation, utilizing NewHope in all network traffic in order to ensure full confidentiality. All communication across the network including sending transactions, sending validator signatures, and distribution of blocks relies on these algorithm. Every time we expanded NewHope into new functions within our system we tested using a similar process as above, checking to see if the data encyrpted properly, was encytpted in transmission, and decrypted accurately.

5.4 Consensus Mechanism

The Consensus Mechanism was in the background compared to the algorithms, but also a vital part of this project. For the time we had during this portion of the project, only one of the proposed two consensus mechanisms was able to be implemented into our system. We were able to successfully implement proof-of-stake into our system and have it fully compatible with the other algorithms used in the system.

5.5 Design and Engineering Standards

By integrating NIST's Module-Lattice-Based Digital Signature Algorithm standard (FIPS 204) and the Federal Information Processing Standard for cryptographic modules (FIPS 140-3), this project achieves both quantum-resilient authenticity and rigorously validated module security. FIPS 204's ML-DSA scheme—built on module-lattice mathematics—ensures that every block and validator message is signed with a quantum-safe algorithm, defending against future large-scale quantum attacks while maintaining acceptable performance on our node hardware. At the same time, designing our wallet and validator components to meet FIPS 140-3 Level 2 requirements enforces role-based authentication, secure key storage, automated self-tests, and operational logging, delivering quantifiable assurance to users that our cryptographic modules resist both logical and physical attack vectors .

Looking ahead, early compliance with these standards positions our blockchain ecosystem for seamless integration into active environments like communications, information security, critical infrastructure—and for straightforward third-party validation or certification processes. As FIPS 204 and FIPS 140-3 (and their underpinning ISO/IEC 19790 controls) become mandated or recommended across both U.S. federal and international contexts, our architecture's alignment with these frameworks not only secures current operations but also future-proofs the system against evolving cryptanalytic capabilities and regulatory requirements .

6 Conclusions

Being quantum-secure is a vital part of being ready for the future and being able to carry out the presidential order given to the Coast Guard. By implementing these post-quantum algorithms into a proof-of-concept through the cryptocurrency we will able to show that they are able to be implemented in a complex system effectively and operate simultaneously. With the demonstration that shows that they can operate, we believe that the Coast Guard would be able to implement these algorithms into less complicated systems, although on a larger scale. The use of consensus mechanisms in this project, although not the main focus of this project, showed us that we were able to take the advantages of the proof-of-stake mechanism and implement it into the project in the hopes of in the future being able to successfully implement the proof-of-help mechanism as well. This combination of consensus mechanisms is a new way for cryptocurrencies to go about implementing them into their system.

References

[1] Erdem Alkim, Léo Ducas, Thomas Pöppelmann, and Peter Schwabe. 2016. NewHope without Reconciliation: Learning with Errors Key Exchange with Simple Error Reconciliation. In *Proceedings of the 25th USENIX Security Symposium*.

[2] Jean-Philippe Aumasson. 2015. The Hash Function BLAKE2. https://blake2.net.

[3] Annie Badman and Matthew Kosinski. 2024. What Is Symmetric Encryption? https://www.ibm.com/think/topics/symmetric-encryption Accessed: 2025-04-22.

[4] Iddo Bentov, Charles Lee, Alex Mizrahi, and Meni Rosenfeld. 2014. Proof of Activity: Extending Bitcoin's Proof of Work via Proof of Stake. In *ACM Workshop on Hot Topics in Security (HotSec)*.

[5] Daniel J. Bernstein, Johannes Buchmann, and Erik Dahmen (Eds.). 2009. *Post-Quantum Cryptography*. Springer.

[6] Joppe W. Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. 2017. CRYSTALS-Kyber: A CCA-Secure Module-Lattice-Based KEM. https://pq-crystals.org/kyber.

[7] BTQ. 2023. How Will Lattice-Based Cryptography Protect Us from Quantum Computers? https://www.btq.com/blog/how-will-lattice-based-cryptography-protect-us-from-quantum-computers Accessed: 2025-04-09.

[8] Vitalik Buterin. 2013. Ethereum White Paper. https://ethereum.org/en/whitepaper/.

[9] Rainer Böhme, Nicolas Christin, Benjamin Edelman, and Tyler Moore. 2015. Bitcoin: Economics, Technology, and Governance. *Journal of Economic Perspectives* 29, 2 (2015), 213–238.

[10] Christian Cachin, Klaus Kursawe, Anna Lysyanskaya, and Reto Strobl. 2008. Asynchronous Verifiable Secret Sharing and Proactive Cryptosystems. In *Advances in Cryptology - CRYPTO 2002*, Moti Yung (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 271–286. https://doi.org/10.1007/978-3-540-88702-7_5

[11] Christian Cachin, Klaus Kursawe, and Victor Shoup. 2005. Secure and Efficient Asynchronous Broadcast Protocols. In *Advances in Cryptology - CRYPTO 2005*, Victor Shoup (Ed.). Lecture Notes in Computer Science, Vol. 3621. Springer Berlin Heidelberg, Berlin, Heidelberg, 524–541. https://doi.org/10.1007/11818175_8

[12] D. A. Cooper et al. 2020. Internet Security Glossary, Version 2 (RFC 4949). https://tools.ietf.org/html/rfc4949.

[13] Kyle Croman, Christian Decker, Ittay Eyal, Adem Efe Gencer, Ari Juels, Ahmed E. Kosba, Andrew Miller, Prateek Saxena, Elaine Shi, Emin Gün Sirer, Dawn Song, and Roger Wattenhofer. 2016. On Scaling Decentralized Blockchains. In *Financial Cryptography and Data Security (FC)*. 106–125.

[14] CyberTalents. 2022. RSA Encryption and RSA Algorithm: A Comprehensive Overview. https://cybertalents.com/blog/rsa-encryption Accessed: 2025-04-22.

[15] Joan Daemen and Gilles Van Assche. 2012. The Keccak Reference. https://keccak.team/files/Keccak-reference-3.0.pdf.

[16] DI Management Services. 2022. Test Vectors for SHA-1, SHA-2 and SHA-3. https://www.di-mgt.com.au/sha_testvectors.html. Accessed: 2025-04-21.

[17] L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, P. Schwabe, G. Seiler, and D. Stehlé. 2018. CRYSTALS-Dilithium: A Lattice-Based Digital Signature Scheme. https://pq-crystals.org/dilithium.

[18] Morris Dworkin. 2001. *Recommendation for Block Cipher Modes of Operation: Methods and Techniques*. NIST Special Publication 800-38A. National Institute of Standards and Technology. https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38a.pdf Accessed: 2025-04-22.

[19] Encryption Consulting. 2022. What is ECDSA Encryption? How Does It Work? https://www.encryptionconsulting.com/education-center/what-is-ecdsa/ Accessed: 2025-04-22.

[20] FIPS PUB 180-4. 2015. Secure Hash Standard (SHS). https://doi.org/10.6028/NIST.FIPS.180-4.

[21] FIPS PUB 202. 2015. SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions. https://doi.org/10.6028/NIST.FIPS.202.

[22] FIPS PUB 204. 2024. Module-Lattice-Based Digital Signature Standard. https://doi.org/10.6028/NIST.FIPS.204.

[23] Juan A. Garay, Aggelos Kiayias, and Nikos Leonardos. 2015. The Bitcoin Backbone Protocol: Analysis and Applications. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT)*. 281–310.

[24] GeeksforGeeks. 2025. Advanced Encryption Standard (AES). https://www.geeksforgeeks.org/advanced-encryption-standard-aes/ Accessed: 2025-04-22.

[25] GeeksforGeeks. 2025. Block Cipher Modes of Operation. https://www.geeksforgeeks.org/block-cipher-modes-of-operation/ Accessed: 2025-04-22.

[26] Stephen Harrigan. 2017. Lattice-Based Cryptography and the Learning with Errors Problem. https://mysite.science.uottawa.ca/mnevins/papers/StephenHarrigan2017LWE.pdf. Accessed: 2025-04-16.

[27] K.A. Jackson, C.A. Miller, and D. Wang. 2024. Evaluating the Security of CRYSTALS-Dilithium in the Quantum Random Oracle Model. In *Advances in Cryptology – EUROCRYPT 2024 (Lecture Notes in Computer Science)*, M. Joye and G. Leander (Eds.), Vol. 14656. Springer, Cham. https://doi.org/10.1007/978-3-031-58751-1_15

[28] Jonathan Katz and Yehuda Lindell. 2014. *Introduction to Modern Cryptography*. Chapman and Hall.
[29] Sunny King and Scott Nadal. 2012. PPCoin: Peer-to-Peer Crypto-Currency with Proof-of-Stake. https://peercoin.net/assets/paper/peercoin-paper.pdf.
[30] Quantum Resistant Ledger. 2024. The Quantum Resistant Ledger (QRL). https://theqrl.org/. Accessed: 2024-09-23.
[31] Georg Legler. 2022. Quantum Computing - The End of Modern Cryptography? https://users.ph.nat.tum.de/ge39leg/post/quantum-computing/ Accessed: 2025-04-22.
[32] F. Thomson Leighton and Silvio Micali. 1995. Large Provably Fast and Secure Digital Signature Schemes from Structured Abstracts. MIT/LCS/TM-505.
[33] Helger Lipmaa, Phillip Rogaway, and David Wagner. 2000. *Comments to NIST concerning AES Modes of Operations: CTR-Mode Encryption*. Technical Report. National Institute of Standards and Technology. https://csrc.nist.rip/groups/ST/toolkit/BCM/documents/proposedmodes/ctr/ctr-spec.pdf Accessed: 2025-04-22.
[34] David A. McGrew and Michael Curcio. 2019. Hash-Based Signatures (RFC 8554). https://tools.ietf.org/html/rfc8554.
[35] Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone. 1996. *Handbook of Applied Cryptography*. CRC Press.
[36] Michele Mosca. 2013. Cybersecurity in an Era with Quantum Computers. https://eprint.iacr.org/2013.
[37] Satoshi Nakamoto. 2008. Bitcoin: A Peer-to-Peer Electronic Cash System. https://bitcoin.org/bitcoin.pdf.
[38] National Institute of Standards and Technology. 2015. *SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions*. Federal Information Processing Standards Publication FIPS PUB 202. U.S. Department of Commerce. https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.202.pdf Accessed: 2025-04-22.
[39] NIST IR 8309. 2020. Status Report on the Second Round of the NIST Post-Quantum Cryptography Standardization Process. https://doi.org/10.6028/NIST.IR.8309.
[40] NIST IR 8547. 2024. Transition to Post-Quantum Cryptography Standards. https://doi.org/10.6028/NIST.IR.8547.ipd.
[41] NIST PQC FAQ. 2020. Post-Quantum Cryptography FAQ. https://csrc.nist.gov/projects/post-quantum-cryptography/faqs.
[42] NIST PQC Project. 2021. Post-Quantum Cryptography Standardization Project. https://csrc.nist.gov/projects/post-quantum-cryptography.
[43] NIST PQC Project. 2023. Third Round Candidates. https://csrc.nist.gov/projects/post-quantum-cryptography/round-3-submissions.
[44] NIST SP 800-56A Rev. 3. 2018. *Recommendation for Pair-Wise Key-Establishment Schemes Using Discrete Logarithm Cryptography*. Technical Report. NIST.
[45] National Institute of Standards and Technology. 2025. Cryptocurrency. https://csrc.nist.gov/glossary/term/Cryptocurrency. Accessed: 2025-04-09.
[46] National Institute of Standards and Technology. 2025. Hash Function. https://csrc.nist.gov/glossary/term/hash_function. Accessed: 2025-04-09.
[47] Author(s) of the paper. Year of publication. Title of the Paper. *CiteSeerX* (Year of publication). https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=4cb8ebc5bc8adc450c40a1bed072a607a287f2bb
[48] Open Quantum Safe Project. 2024. CRYSTALS-Dilithium. https://openquantumsafe.org/liboqs/algorithms/sig/dilithium.html. Accessed: 2025-04-21.
[49] D. A. Osvik et al. 2020. BLAKE3: One function, fast everywhere. https://github.com/BLAKE3-team/BLAKE3-specs.
[50] Chris Peikert. 2014. Lattice Cryptography for the Internet. https://eprint.iacr.org/2014/070.
[51] Thomas Pöppelmann. 2019. NewHope Round 2 Presentation. https://csrc.nist.gov/CSRC/media/Presentations/new-hope-round-2-presentation/images-media/newhope-poeppelmann.pdf. Presented at the Second PQC Standardization Conference, August 24, 2019, University of California, Santa Barbara.
[52] Ronald L. Rivest, Adi Shamir, and Leonard Adleman. 1978. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Commun. ACM* 21, 2 (1978), 120–126.
[53] Peter W. Shor. 1997. Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. *SIAM J. Comput.* 26, 5 (1997), 1484–1509.
[54] Emil Sjöberg. 2022. Discrete Logarithms, Integer Factorization and their Role in Classical and Quantum Cryptography. https://www.diva-portal.org/smash/get/diva2:1729425/FULLTEXT01.pdf. https://www.diva-portal.org/smash/get/diva2:1729425/FULLTEXT01.pdf U.U.D.M. Project Report 2022:27, Uppsala University.
[55] The White House. 2023. Fact Sheet: Executive Order on Strengthening and Promoting Innovation in the Nation's. https://www.presidency.ucsb.edu/documents/fact-sheet-executive-order-strengthening-and-promoting-innovation-the-nations Accessed: 2025-04-21.
[56] Intro to Quantum. 2023. Quantum Essentials. https://introtoquantum.org/essentials/quantum/ Accessed: 2025-04-09.

7 Apendices

7.1 Dilithium2 Keys and Signature

This image shows the public key, private key, and signaure of a user on our system. It also shows the size of each in bytes.

7.2 Expected Dilithium2 Keys and Signature Size

This image shows the expected size of the public key, private key, and signature using Dilithium2.

Parameter set summary

Parameter set	Parameter set alias	Security model	Claimed NIST Level	Public key size (bytes)	Secret key size (bytes)	Signature size (bytes)
Dilithium2	NA	SUF-CMA	2	1312	2528	2420
Dilithium3	NA	SUF-CMA	3	1952	4000	3293
Dilithium5	NA	SUF-CMA	5	2592	4864	4595

Figure 5: Source: [48]

7.3 Keccak Output for Test Vectors

This image shows the output of inputting "abc" and "" into the Keccak hashing function.

7.4 Expected Keccak Output for Test Vectors

This image shows the expected output for multiple test vectors for various versions of Keccak. We are focused on SHA-3-512.

Input message: "abc", the bit string (0x)616263 of length 24 bits.

Algorithm	Output
SHA-1	a9993e36 4706816a ba3e2571 7850c26c 9cd0d89d
SHA-224	23097d22 3405d822 8642a477 bda255b3 2aadbce4 bda0b3f7 e36c9da7
SHA-256	ba7816bf 8f01cfea 414140de 5dae2223 b00361a3 96177a9c b410ff61 f20015ad
SHA-384	cb00753f45a35e8b b5a03d699ac65007 272c32ab0eded163 1a8b605a43ff5bed 8086072ba1e7cc23 58baeca134c825a7
SHA-512	ddaf35a193617aba cc417349ae204131 12e6fa4e89a97ea2 0a9eeee64b55d39a 2192992a274fc1a8 36ba3c23a3feebbd 454d4423643ce80e 2a9ac94fa54ca49f
SHA-3-224	e642824c3f8cf24a d09234ee7d3c766f c9a3a5168d0c94ad 73b46fdf
SHA-3-256	3a985da74fe225b2 045c172d6bd390bd 855f086e3e9d525b 46bfe24511431532
SHA-3-384	ec01498288516fc9 26459f58e2c6ad8d f9b473cb0fc08c25 96da7cf0e49be4b2 98d88cea927ac7f5 39f1edf228376d25
SHA-3-512	b751850b1a57168a 5693cd924b6b096e 08f621827444f70d 884f5d0240d2712e 10e116e9192af3c9 1a7ec57647e39340 57340b4cf408d5a5 6592f8274eec53f0

Input message: the empty string "", a bit string of length 0.

Algorithm	Output
SHA-1	da39a3ee 5e6b4b0d 3255bfef 95601890 afd80709
SHA-224	d14a028c 2a3a2bc9 476102bb 288234c4 15a2b01f 828ea62a c5b3e42f
SHA-256	e3b0c442 98fc1c14 9afbf4c8 996fb924 27ae41e4 649b934c a495991b 7852b855
SHA-384	38b060a751ac9638 4cd9327eb1b1e36a 21fdb71114be0743 4c0cc7bf63f6e1da 274edebfe76f65fb d51ad2f14898b95b
SHA-512	cf83e1357eefb8bd f1542850d66d8007 d620e4050b5715dc 83f4a921d36ce9ce 47d0d13c5d85f2b0 ff8318d2877eec2f 63b931bd47417a81 a538327af927da3e
SHA-3-224	6b4e03423667dbb7 3b6e15454f0eb1ab d4597f9a1b078e3f 5b5a6bc7
SHA-3-256	a7ffc6f8bf1ed766 51c14756a061d662 f580ff4de43b49fa 82d80a4b80f8434a
SHA-3-384	0c63a75b845e4f7d 01107d852e4c2485 c51a50aaaa94fc61 995e71bbee983a2a c3713831264adb47 fb6bd1e058d5f004
SHA-3-512	a69f73cca23a9ac5 c8b567dc185a756e 97c982164fe25859 e0d1dcc1475c80a6 15b2123af1f5f94c 11e3e9402c3ac558 f500199d95b6d3e3 01758586281dcd26

Figure 6: Source: [16]

7.5 AES-256-CTR Wireshark Packet

This image shows a Wireshark packet being sent over our network. The image shows the content is encrypted in transport as expected.