A Model for Detecting Fake Negative Acknowledgements (NACK) in Named Data Network (NDN) Using Blockchain Technology

by

Ritwick Saha
22141054
Tahsina Shiva
19101668
Kazi Sharmin Akter Tayeeba
18101372

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science

Department of Computer Science and Engineering
BRAC University
May 2022

Declaration

It is hereby declared that

1. The thesis submitted is my/our own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

Student's Full Name & Signature:

Ritwick Saha

Ritwick Saha
22141054

Tahsina Shiva

Tahsina Shiva
19101668

Kazi Sharmin Akter Tayeeba

Kazi Sharmin Akter Tayeeba
18101372

Approval

The thesis/project titled "A Model for Detecting Fake Negative Acknowledgements (NACK) in Named Data Network (NDN) Using Blockchain Technology" submitted by

1. Ritwick Saha (22141054)
2. Tahsina Shiva (19101668)
3. Kazi Sharmin Akter Tayeeba (18101372)

Of Spring, 2022 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science on May 29, 2022.

Examining Committee:

Supervisor:
(Member)

[illegible] 29.5.22

Arif Shakil
Lecturer
Department of Computer Science and Engineering
Brac University

Program Coordinator:
(Member)

Dr. Md. Golam Rabiul Alam
Associate Professor
Department of Computer Science and Engineering
Brac University

Head of Department:
(Chair)

Dr. Sadia Hamid Kazi
Chairperson and Associate Professor
Department of Computer Science and Engineering
Brac University

Abstract

Data Networking is a new concept in current internet technology that replaces the traditional TCP/IP architecture by addressing nodes depending on user interest. Due to the massive volume of traffic generated, Internet security is an extremely demanding study subject. The built-in structure of NDN prioritizes stronger privacy and security protections, allowing for more scalable networking. Though the architecture of NDN helps in the prevention of some current TCP/IP attacks, attackers exploit some of these aspects and use them to their advantage. They have the ability to launch new types of attacks, such as the creation of fake Negative Acknowledgement (NACK). NACKs are important in the NDN architecture because they cleanse state in routers and inform consumers.NACKs appear to be beneficial in terms of security since they can assist in mitigating so-called ”Interest Flooding Attack”(IFA).

Using blockchain technology, this thesis focuses on detecting harmful Negative Acknowledgements (NACK). Our proposed framework identifies and alerts users to fake NACKs in the network. Through the paper, our main objective is to make interaction between the client and the producer simpler by applying cutting-edge technologies.

Keywords: TCP/IP architecture; NDN Architecture; Negative Acknowledgement (NACK); Blockchain technology; Interest Flooding Attack (IFA).

Acknowledgement

First and foremost, we would like to thank God for whom our thesis have been completed without any major interruption.
Secondly, to our advisor Mr. Arif Shakil sir for his kind support and advice in our work. He helped us whenever we needed help.
And finally to our parents. Without their throughout support it may not be possible. With their kind support and prayer we are now on the verge of our graduation.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

The fundamental concepts that underpin the internet were created in the late 1960s and early 1970s, when the telephone was the only example of viable and effective global-scale communications. The internet as we currently know was created to address the problem of resource sharing, which was worsened by the high cost of hardware resources in 1960s. However, we still use the same Internet today, mostly for information delivery and retrieval. According to the Cisco Visual Networking Index, video traffic will account for 80 percent of total traffic by 2021 [8], indicating that internet usage has shifted from resource sharing to content delivery. Despite the fact that the internet's applications have changed, the internet's infrastructure has not. The mismatch between architecture and usage has resulted in various IP problems, including security as it was overlooked when the present TCP/IP architecture was designed.

The idea was presented by Van Jacobson, a well-known TCP/IP architect that has prompted researchers to create a new architecture to address the security concerns. This architect's innovative concept focuses on content rather than location. Researchers named it the 'Named Data Network' (NDN), because NDN content names provide all of the information required for transmission. A port or a sequence number are not required. NDN focuses on protecting content rather than the communication route via which it is transferred.

In this thesis, we study security challenges in NDN, we identify the attacks and severity for NDN, and propose a methodology which is basically blockchain based fake NACKs detection and mitigation aimed at the flowing of legitimate traffic and reducing the malicious traffic that stores passes through the NDN router. We have implemented our mechanism and compared it to existing state-of-the-art intruder detection strategies.

We summarize the contributions of this thesis as follows:

- We create an attack detection model to detect fake Negative Acknowledgements (NACKs)
- We use blockchain technology for detection of attacks
- We provide an attack mitigation model
- We compare the performance of our model with up-to-date approaches

1.1 Named Data Networking Architecture

In this section, we will go through the strong hourglass architecture created for NDN. The hourglass form is inherited by NDN from the TCP-IP architecture. The hourglass form is shown in figure 1.1 both the TCP-IP and NDN protocol stacks are supported.

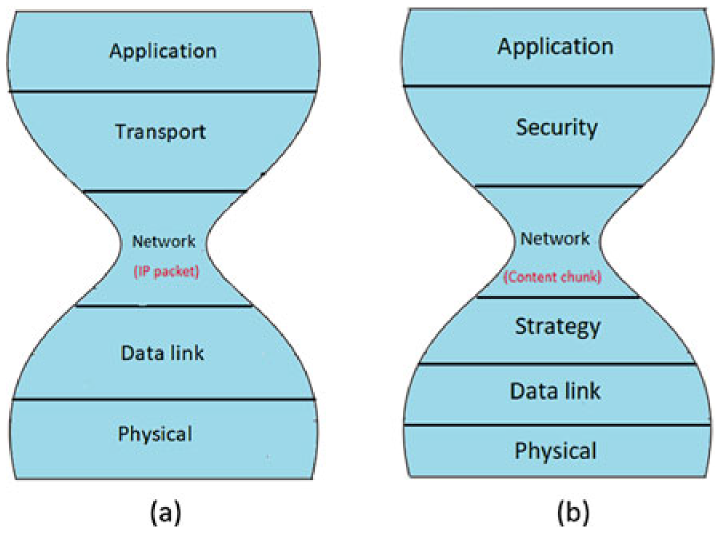


Figure 1.1: Internet and NDN Narrow-Waist Hourglass Architectures [19]

The main distinction between the two protocol stacks is that IP's narrow waist represents host-centric delivery (delivering packets to a particular destination). NDN, on the other hand, is content-centric (retrieve contents from the network based on names). Security and strategy are two additional layers added to the NDN stack. Content security is handled by the security layer. If, when, and where NDN packets are transmitted is determined by the strategy layer. As seen in Figure 1.1, the NDN stack has no transport layer. This is because, under NDN, content names contain all of the information required for transit. A port or a sequence number are not required.

Communication between hosts in NDN is performed through the generation of two separate types of packets. This is a critical component of NDN that replaces the IP address identification currently used in TCP/IP. Figure 1.2 depicts the many forms of NDN packets. Any node that sends requested data interests operates as a content consumer, whereas any node that can deliver data packets works as a producer.

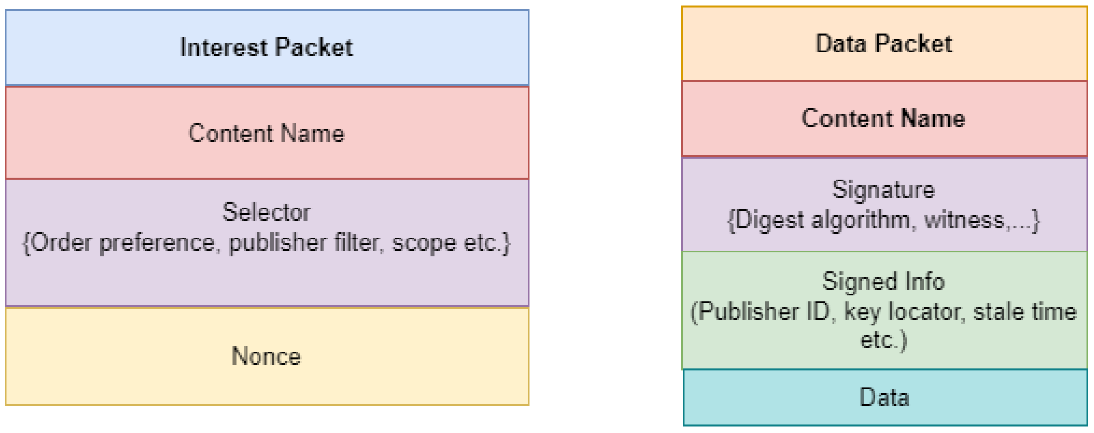

Interest Packet
Content Name
Selector
{Order preference, publisher filter, scope etc.}
Nonce
Data Packet
Content Name
Signature
{Digest algorithm, witness,...}
Signed Info
(Publisher ID, key locator, stale time etc.)
Data

Figure 1.2: NDN Interest Packet and Data Packet

Interest Packet: The customer's requested material is contained in the Interest package, which is created at the start. An Interest packet is divided into three sections:

Content Name: Content name contains the requested searched name, which identifies the desired material.

Selector: Selector field is used to refine material searching a number of fields such as order performance, scope, and so on.

Nonce: Nonce field is used to locate and identify packets with the same or equivalent interest.

Data Packet: The data packet contains the response to the requested interest packet together with its matched name and a signature by the producer's key. It links the consumer and producer. A data packet is divided into four sections: the content name, the signature, the sign Info, and the contents. The required data follows the path reversed by the interest from the asking consumer. Signed info contains information that may be used to validate the signature's authenticity and validity.

1.1.1 Naming Structure in Named Data Networking

NDN's network design is based on the content's name. NDN gives each item of material a human-readable name. It's used for both packet routing and content identification. Each name may contain a variety of elements. The character " / " is used to denote the separation of two components.

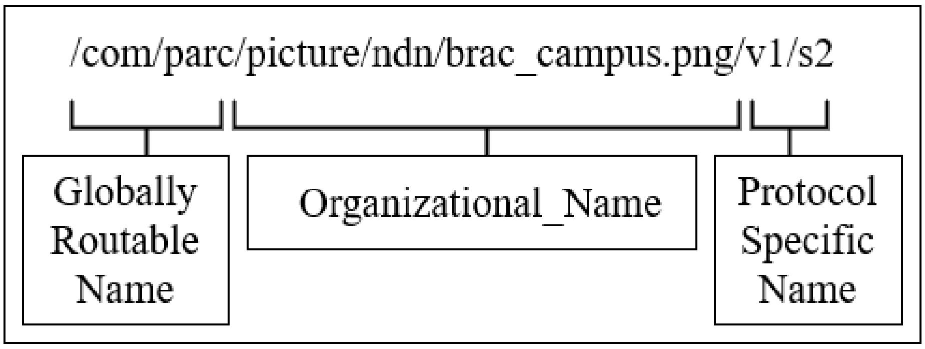


Figure 1.3: NDN Naming

Because the names are hierarchical, they can be aggregated in the same way that HTTP URLs can. A search engine helps to produce a name based on the keywords entered by the user. As shown in Figure 1.3, the name is divided into three sections. When we wish to make the data available globally, we need to use the globally routable prefix.

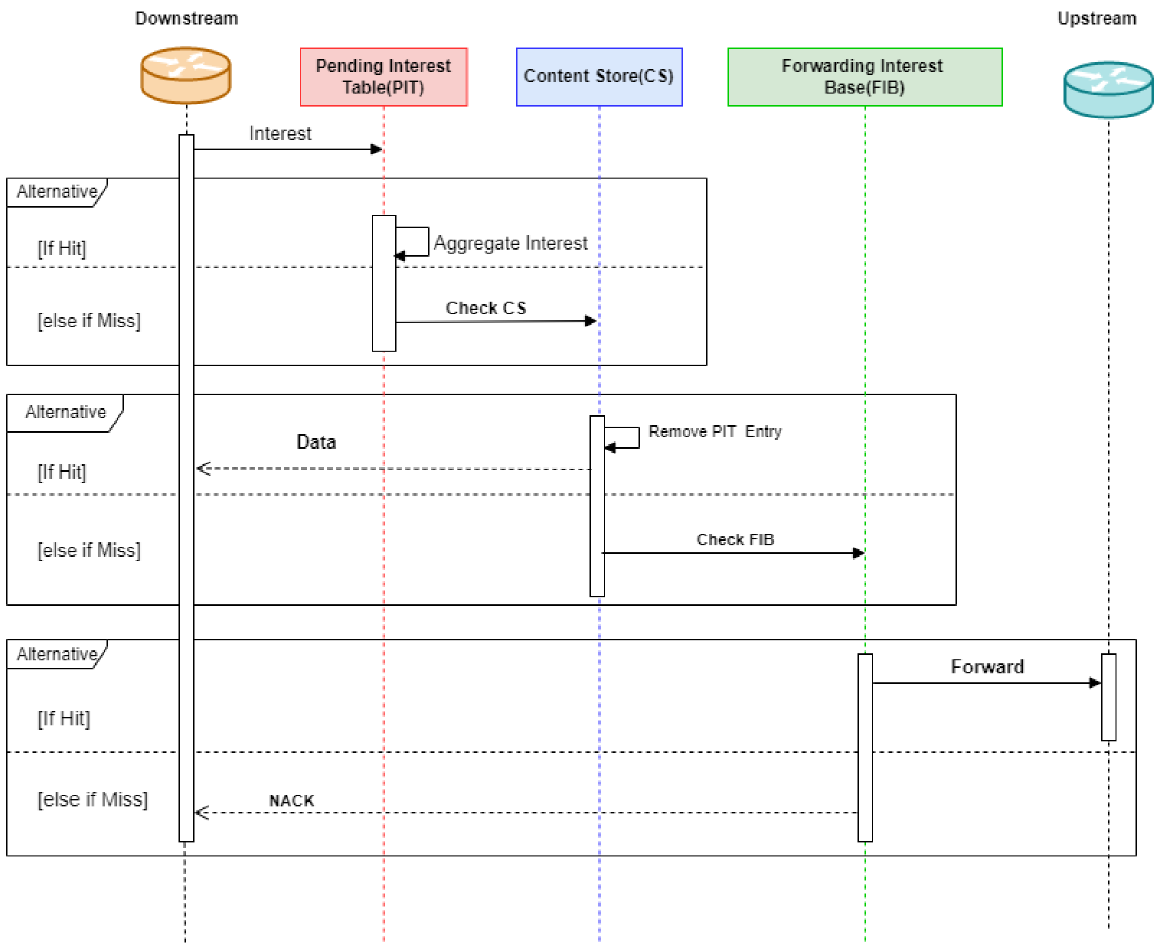
1.1.2 Packet Routing

Once generated, the interest packet takes a predefined route through a previously established router section. Every NDN object (not only routers) has three major data structures. They are as follows:

- ***Content Store (CS):*** The data packets that transit via forwarding nodes are stored in their CS. Any interest that matches the names for which they have saved data in their content repository will receive that data, eliminating the need for the interest to contact the producer.

- ***Pending Interest Table (PIT):*** PIT stores any unsatisfied interests that have been passed upstream to potential data providers. For receiving and outgoing data, each PIT entry has one or more physical interfaces. Several incoming interfaces show that the same data is being sought from multiple downstream users; different outgoing interfaces indicate that the same interest is being communicated across multiple channels.

- ***The Forwarding Interest Base (FIB):*** FIB works in the same way as the IP Forwarding table. However, name prefixes are retained rather than IP prefixes. Each FIB entry may provide several interfaces rather than a single ideal interface for each name prefix. The name-based routing protocol, such as NLSR, is used to create FIB



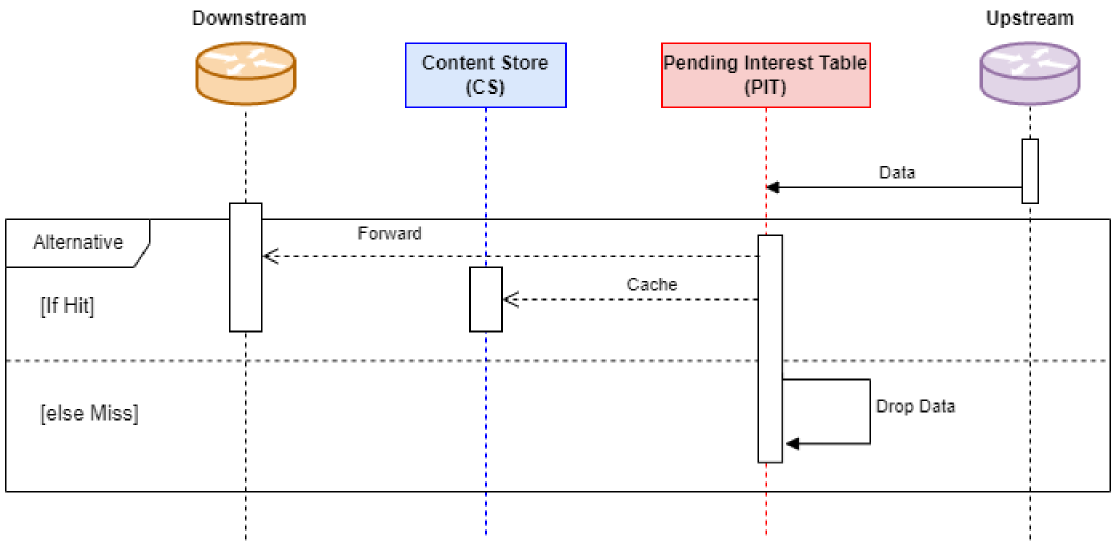


Figure 1.4: Packet forwarding in NDN

1.2 HTTP VS NDN

The basic goal of NDN is to relocate the semantics of HTTP's named data object request and response, which contains the requested item, to the network layer. NDN requests and answers function at the network packet level since it is a network layer protocol. Each request provides the data name and delivers one NDN Data packet in an NDN Interest packet. A large application data object is segmented, with the segment number contained in the data packet name. The data name is carried by both packet types; none carries an address or information about the requester. Web protocols that employ names to request material are used by the most of today's Internet apps. NDN uses this request-response communication architecture and directly uses application data names at the network layer to make network services better match for application communication patterns. In contrast to URLs used in HTTP queries, NDN data names are utilized by the network layer to fetch data, which are used exclusively by apps. As a result, namespace design is the most crucial and frequently the most difficult stage.

The functions that each must do are another key contrast among HTTP as an application protocol and NDN as a network layer protocol. HTTP uses a transport connection, such as TCP or QUIC, to ensure that packets are sent between the data

requestor and the data provider. As a consequence, a web application just has to send the request and wait for a response or a connection failure. NDN, on the other hand, transports packets over a network, which might be a local IoT network, ad hoc network of mobile devices, or the worldwide Internet. As a result, the data packet must return to the requester whereas the Interest packet has no requester information [10].
In addition to being network layer packets, NDN Data packets vary from HTTP data objects in two important aspects. First, while the underlying TCP connection connects the requesting URL to an HTTP response message implicitly, in addition to the required content, every NDN Data packet specifically contains the data name, accompanied with a signature at the moment of data generation that cryptographically connects the name to the content; the information may also be encrypted if required. Second, unlike URLs that return varying content, NDN Data packets are unchangeable: each name identifies an NDN Data packet; when a producer changes the content of a data packet, it must create a new packet with a different name to distinguish between different versions of the data.

1.3 Research Problem

The Named Data Network (NDN) was designed with the intention of controlling the increasing volume of material generated and distributed online. Its hourglass figure keeps it appropriate for current internet technology, resulting in a clear and visible progression plan [19]. NDN, like IP, can work on just about anything can work on NDN at the very same time. Although the architectural design defends NDN from several well-known TCP/IP attacks like ICMP flooding, reflection attacks, TCP SYN flood assaults, and so on, NDN's unique features like stateful forwarding plane and in-network caching encourage the introduction of new forms of attacks Injecting NACKs in an intruder control network link or DoS attack using fake Negative Acknowledgment (NACK) are such security problems.
In most communication protocols, there are two methods for determining whether a package has indeed been accepted. Negative Acknowledgements and Acknowledgments (NACKs). A receiver tells the sender of all packets which have been

successfully effectively accepted in ACK-based protocols. If a receiver detects an unfamiliar, irrational, or defective packet in a NACK-based protocol, it tells the sender. NACK is beneficial for a variety of reasons. A customer cannot distinguish between packet loss and other causes without NACK. Without NACK consumers and routers must wait till the expiry time has passed. After receiving a NACK with the reason code ”non-existent content,” there is no compelling reason to send packets to alternate paths. NACK aids in flushing out state in routers and notifying customers, freeing up valuable resources.

In Named Data Network there are two kinds of network layer NACKs. They are -

Content NACK (cNACK): A producer generates a packet known as a cNACK. It signifies that a peice of component bearing the title represented in a given request somehow doesn’t exist, has never been created, nor has never been published. A cNACK is a type of content object with the CNACK type. cNACKs provide a number of advantages. On the consumer side, they assist [5]:

1. To distinguishing between packet loss and content not found
2. Reducing consumer waiting time by notifying customers quicker than interest timeouts.

cNACKs can help routers and producers mitigate the impacts of Interest Flooding (IF) attacks. For each individual interest that it sends, a router produces a PIT (Pending Interest Table) entry. Content is not cached and transmitted downstream before a PIT record is erased. If, on the other hand, an interest requests non-existing material and the producer just ignores it, the associated PIT entries at all intermediate routers (and at the consumer) stay until they expire. A producer-generated cNACK enables routers to discard PIT entries earlier, freeing up resources. Although this method does not completely eliminate the impacts of IF attacks, it does greatly minimize them. The functioning method of cNACK is visualized in Figure 1.5

Forwarding NACK (fNACK): A router generates a network layer package known as a fNACK. Its objective is to alert downstream routers that data that is of relevance cannot be transferred due to latency or an unknown next step. The scenarios in which a router generates fNACKs, a router can generate a fNACK for two reasons:

1. No item in the FIB identifies the next-hop of the received interest,
2. All FIB-specified outbound interfaces are crowded.

Intrusion Detection System (IDS) is one of the most prominent security solutions for protecting network equipment and PCs against hostile behavior and unauthorized use. It detects a variety of threats. Individuals and network administrators can use IDS to put in place preventive protective measures.IDS generally use two distinct techniques:

- Anomaly Detection
- Signature Detection

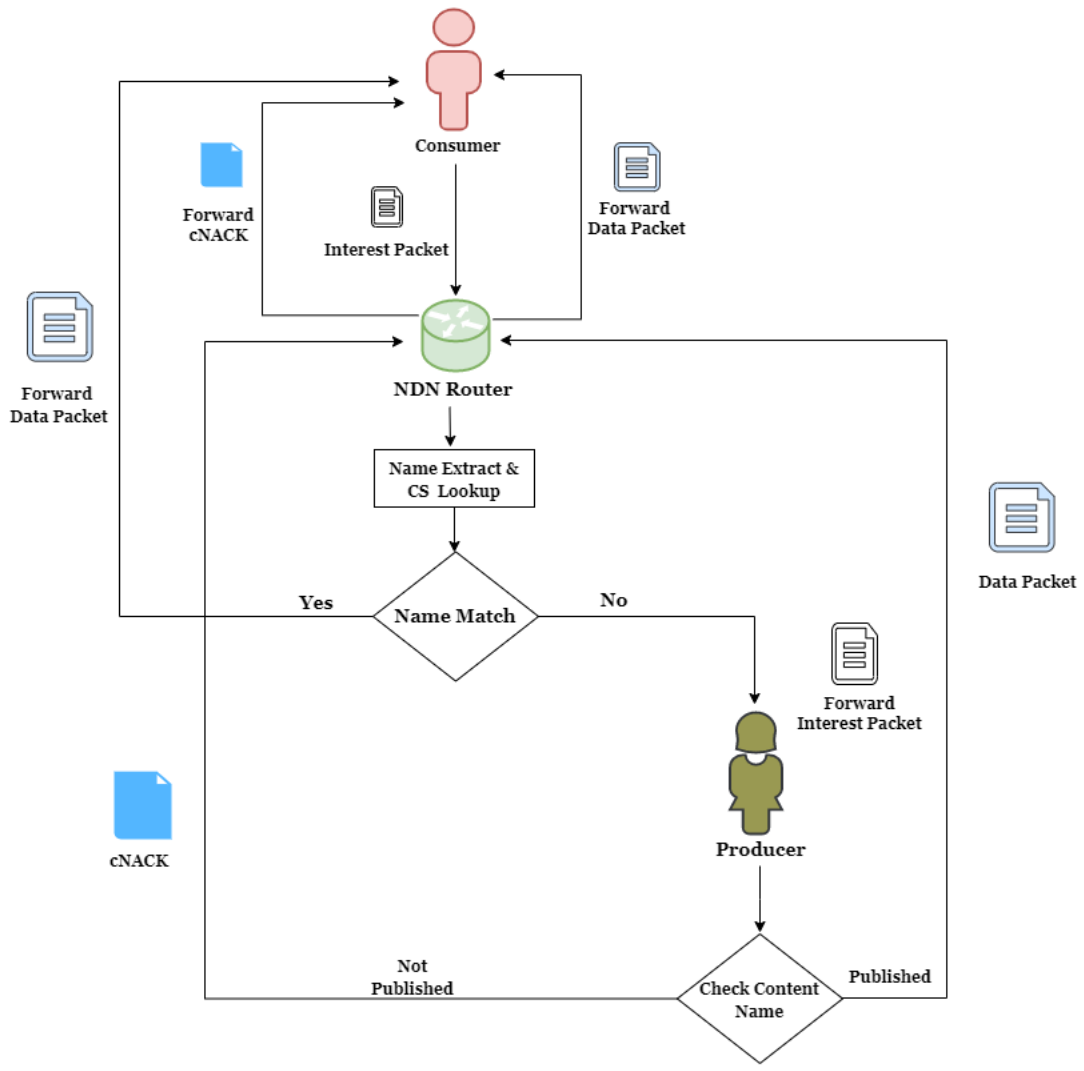


Figure 1.5: Workflow of cNACK

The primary difference between the two is that anomaly detection analyzes different factors and compares that to normal behavior, whereas signature detection searches for a specific behavior that is recognized as an attack. Additionally,Intrusion Detection Systems provides two types of classifications:

1. The Network Intrusion Detection System (NIDS)
2. The Host Intrusion Detection System (HIDS)

The Network Intrusion Detection System (NIDS) captures packets from network traffic. The header of the recorded packets is evaluated using many criteria to detect malicious behavior. The Host Intrusion Detection System (HIDS) is installed on each machine to identify unauthorized access or usage. Both sorts of systems are occasionally required to provide overall protection. A large network has multiple Network Intrusion Detection Systems spread throughout it. These networked intrusion detection systems exchange records and alarm information. A Distributed Intrusion Detection System (DIDS) is a collection of intrusion detection systems. Throughout the network, it offers improved sustained security assessment, network management, and real risk analysis. It gives admins a more in-depth understanding of the malicious activity. Because the Distributed Intrusion Detection System is built on the internet communication of individual IDS, the trustworthiness, validity, and reliability of the warnings received from the individual IDS are all issues to be concerned about. In other cases, the hacker may have hacked a specific IDS or it may be poorly designed, causing the server to be bleeped with the incorrect alert and leading the administrator astray. Coordinating several IDS and exchanging data with a central approach presents several issues. Credibility and dependability are significant issues that must be addressed. They are explored more below:

- The accuracy of individual IDS warnings is crucial. Records and alerts should be impenetrable to intruders and should never be accessed or edited by anyone.
- All participating IDS should agree on the kinds and standard of alert production.
- As the network expands and retreats, the computational requirements for the centralized server should be able to vary.
- Each IDS participating in the program ought to have access and ownership over how alert data is disclosed and accessed.

Blockchain appears to be the most dependable option for resolving these issues and making the whole system more resilient and trustworthy. For exchanging the logs and warnings created by the individual IDS, the development of a secure distributed ledger is recommended. Several studies have utilized blockchain technology in intrusion detection systems to identify malicious intrusions. These studies may be divided into two distinct categories: (1) users who employ the anomaly detection method, (2) those who employ the signature method. As Negative Acknowledgement (NACK) is a unique form of content, allowing it to be signed violates a fundamental principle of the Named Data Network (NDN). However, attackers can exploit the signing NACKs. As we all know, signing requires a large amount of processing power. Producers may consume their resources as a result of signing. It causes yet another DoS attack. According to [5] the use of blockchain technology for anomaly detection is more prevalent than for signature detection. The most essential requirement of the hour is to develop a way of detecting fake negative acknowledgements (NACK) on Named Data Networks (NDN) in real time. This paper provides a framework for addressing the issue statement.

1.4 Blockchain

The word blockchain refers to a series of blocks that were initially developed in 1991 by researchers Stuart Haber and W. Scott Stornetta as a way to securely date a digital document. Furthermore, in 1992, those researchers, working with Dave Bayer, increased the efficiency of this technique by creating the Merkle root, which efficiently checks the data's integrity. Satoshi Nakamoto released the first article introducing the term ”chain of blocks” to describe a peer-to-peer electronic payment system in 2008. The term ”chain of blocks” was replaced with the term ”blockchain”. Blockchain is a network-wide decentralized and distributed ledger that can preserve and record peer-to-peer transactions in an unchangeable and secure manner. Without the existence of a central authority, each network participant can interact directly with one another. The P2P network establishes a set of rules that must be followed by all members of the network. When transactions are carried out between peers, each participant must verify the transactions' legitimacy using certain rules before storing them in a block. The blocks form a chain when they are joined together. Each block includes the hash code of the block before it, ensuring transaction integrity and immutability. As a result, the chain gets more secure as each new block validates all prior blocks.
According to the blockchain's functioning, cryptography and hash functions are used to safeguard and preserve each user's identity unique in the network. The public and private keys are held by each peer in the network. When a peer wishes to make a transaction, he should use his private key to sign it. The remaining parties must then validate the transaction by confirming the sender's signature using its public

key. Because peers may execute transactions using their keys, the central authority has no control of the whole network. Another reason of no need of centralized authority is that each network member has a copy of the updated ledger, which contains all of the chain's blocks along with their relevant transactions.

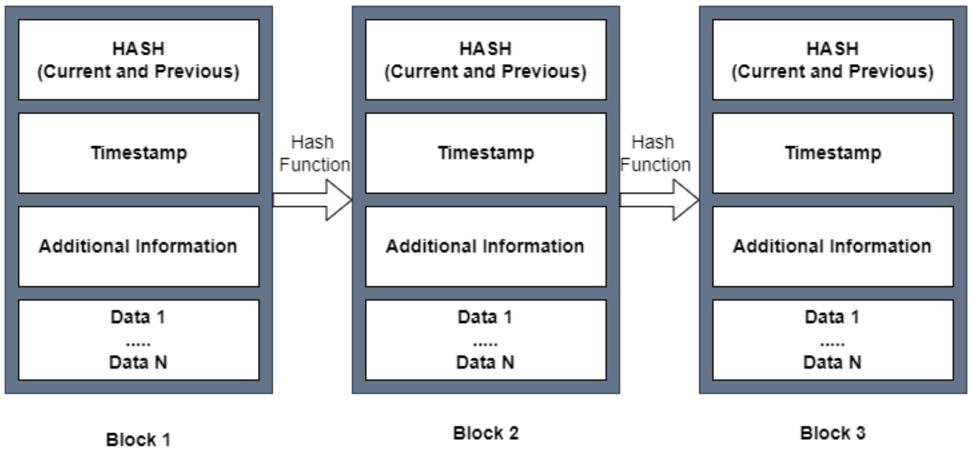


Figure 1.6: Blockchain Structure

Every peer in a peer-to-peer network has the ability to join or exit the network at any moment. The problem with a public blockchain is that the participants don't know each other, therefore peers have no way of knowing who they can trust in the network. This difficulty may be avoided by using consensus algorithms, which require all or a majority of peers to agree to the network rules in order for the blocks in the ledger to be finalized.

1.4.1 Decentralized Network

Decentralized network avoids various drawbacks of centralized network, most notably the single point of failure. Multiple central nodes control the network, and users are linked to them. Furthermore, unlike a centralized network, where data is held exclusively in a single authority, all information is saved in all central nodes. Additionally, peers can interact directly with one another without the involvement of a third party. In the event that a central node fails, the network will continue to function since there are alternative central nodes to which users may connect to complete their transactions.

Information is kept in several users in decentralized networks, making it impossible for an attacker to change the information in all users. Furthermore, it is difficult to detect the names and addresses of users from a rogue node. Because the data is routed through many central nodes, it is hard to monitor a specific user. Another benefit of this network is that there is no bottleneck because many nodes are in charge of different implementations.

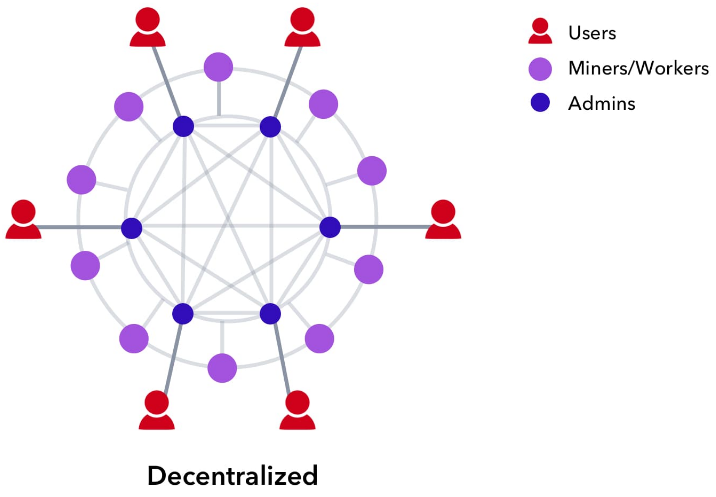


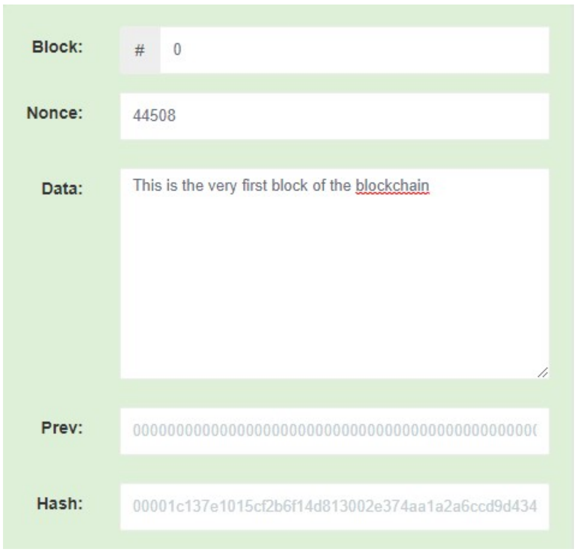
Figure 1.7: Decentralized Network

1.4.2 Genesis Block

Blocks are essentially digital capsules that store information about network interactions indefinitely. The Genesis Block, also known as Block 0, is the initial block in a blockchain and serves as the foundation for succeeding blocks.Because every block refers to the one before it, it is basically the ancestor to which every subsequent block may trace its ancestry. This started the process of verifying transactions and creating new blocks. The layers and complex history of each sequence are one of the aspects that make a blockchain-based data transmission system so safe.

1.4.3 Hash Function

Each block in our chain contains a hash that acts as a unique identifier for that block. These hashes are immutable and serve as the foundation of the blockchain, because they both represent and connect our whole network. Every block reference to the previous block's hash. These hashes are built using the timestamp, nonce,


Block: # 0

Nonce: 44508

Data: This is the very first block of the blockchain

Prev: 00000000000000000000000000000000000000000000000

Hash: 00001c137e1015cf2b6f14d813002e374aa1a2a6ccd9d434

Figure 1.8: Genesis Block

and data. We only construct one hash from the 264 integers since each one is unique, with a 1 in 3.71011 chance that the hashes will match.

1.4.4 Previous Hash Function

The preceding hash (SHA-256) connects all of the blocks in our blockchain, ensuring the integrity of the blockchain's one-way retrieval. This system employed a 64-bit SHA-256 hash that was immutable, meaning it could never be overwritten. Every block in our system refers to the block before it, and so to the genesis block. Because the blocks are dynamically formed and change on a regular basis, traveling through them will never lead to the genesis block. As a result, blockchain retains its immutability and integrity.

1.4.5 Public Key and Private Key

Each peer in the blockchain network has a pair of asymmetric keys, the public and private keys. These keys are important not just in the formation of transactions, but also in ensuring that they are legitimate. Each user has a unique private key that is kept secret. This key is used to digitally sign transactions in order to verify their validity.The public key is available to everyone on the network since it is used to hash each user's address. Furthermore, the public key is used to confirm that the transaction was sent by the correct sender. Furthermore, hashing the public key

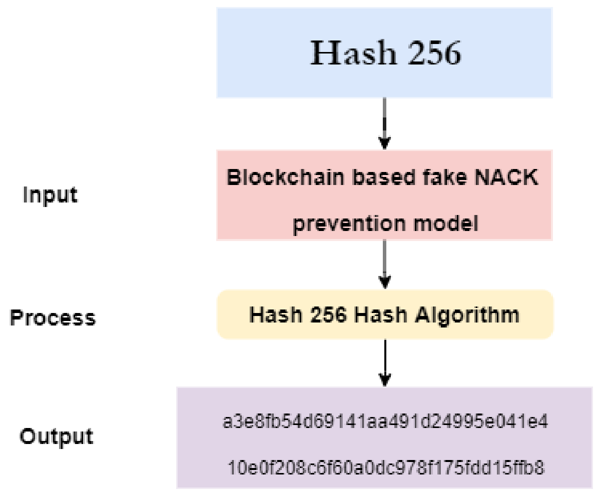

Hash 256
Input
Blockchain based fake NACK prevention model
Process
Hash 256 Hash Algorithm
Output
a3e8fb54d69141aa491d24995e041e4
10e0f208c6f60a0dc978f175fdd15ffb8

Figure 1.9: SHA-256 Hashing

provides the user with additional privacy.

The digital signature has two phases: the signing phase, during which the signature is formed using the private key, and the verification phase, during which anybody may verify the digital signature.

Alice, for example, wishes to conduct a business transaction with Bob. She must take a few procedures in order to complete this transaction.

- First Alice must need to produce a hash of all the data associated with this transaction.
- She also generates a digital signature by encrypting the hash acquired from the transaction's contents with her private key.
- Finally, she sends Bob the transaction data, which includes the digital signature. Bob may then confirm that Alice is the true sender of this transaction.

On the other hand, after Bob has received the transaction data as well as Alice's digital signature, he must do two steps to ensure that Alice's transaction is genuine.

- To obtain the hash, Bob must decrypt Alice's digital signature using her public key.
- Second, Bob must calculate the hash of the data using the same hash technique as Alice.

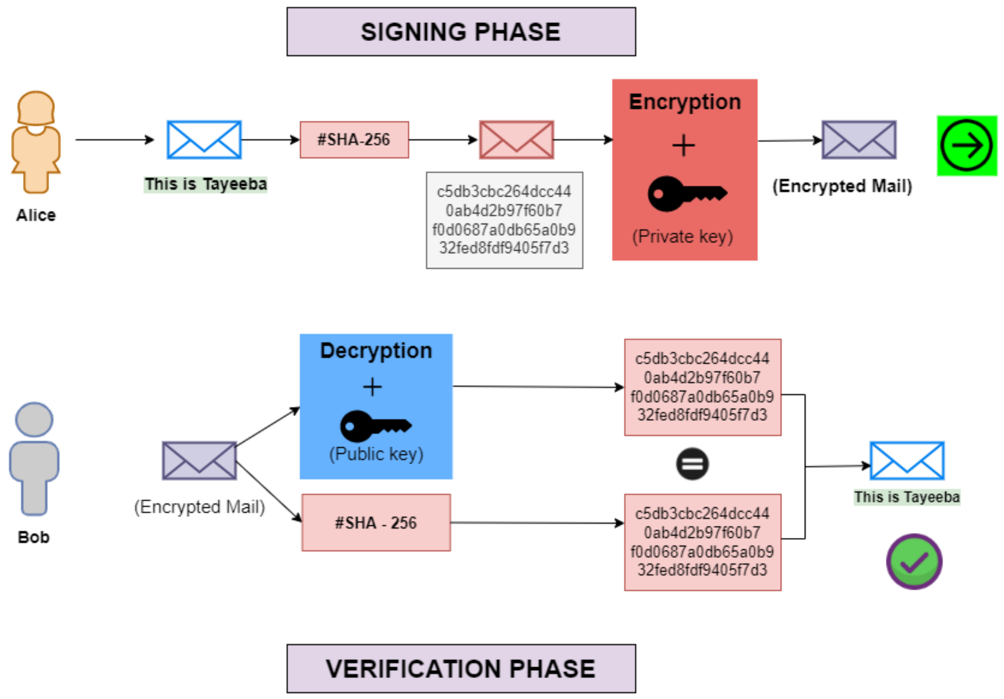


Figure 1.10: Implementation of Transaction

If the hash results match, it is safe to assume that the verification was completed without tampering with the data and that Alice is the true sender of the transaction.

1.5 Research Objectives

The goal of this research paper is to provide a system for detecting malicious NACKs utilizing blockchain technology. The primary goal of this study is to facilitate contact between the customer and the producer. The goals of this research are as follows:

1. To gain a thorough understanding of the Named Data Network and how it operates.
2. To have a thorough understanding of Blockchain Technology and how it works.
3. Create a blockchain-based model to detect false NACKs and apply it to NDN.
4. To analyze the model.
5. Make suggestions for enhancing the model.

Chapter 2

Literature Review

NDN, as previously said, is information centric rather than host focused, meaning it primarily works with content. Because of its unique properties, the network oversees discovering and returning requested content, which has shown to be the safest means to send data thus far. NDN serves the present IP based architecture with its modified functionality. Furthermore, blockchain technology has been demonstrated to be one of the most popular technologies that signifies a technological revolution. By distributing copies of documents to all participants, the blockchain approach eliminates the need for a central authority.
This section is to emphasize on previous implementation of NDN and security issues of usage of blockchain on NDN and its remedies.

2.1 Blockchain on Named-Data-Networking

The introduction of certain technologies regarding NDN and blockchain, has piqued the researcher's interest since they allow efficient data retrieval and data security. NDN's data-centric technique enables efficient record distribution and expert blockchain block synchronization. To solve the mutual trust problem between the sites,[12] proposes a blockchain-based NDN key management solution. The outcomes of this scheme's research and evolution show that it can support fewer verification numbers while enhancing verification efficiency. However, using blockchain on NDN can alleviate a variety of networking issues, it still has severe drawbacks. The main disadvantage of this strategy is that blockchains require a lot of storage. It is necessary to identify and prevent an integrated, lightweight attack technique suitable for limiting computing and storage capacity of NDN of things. Additionally, during the early stages of the concept's implementation, problems with denial of service (DoS) and peer collusion affecting the system's robustness and security were noticed. A malevolent peer may add many blocks to a ledger system, resulting in an illegal block and ruining the data integrity of the ledger [15].

2.2 Blockchain on Content Delivery in Named Data Network

With its unique, persistent, and location-independent name, the Named Data Network (NDN) guarantees immediate retrieval of content. However, this architecture could lead to major problems such as data manipulation and data broadcasting by rogue NDN nodes. In [16], a trusted blockchain-based Information Centric Network (ICN) architecture is offered as a solution to this problem. A hybrid naming strategy is used to create the trust mechanism in this suggested architecture. The publisher must create both a hierarchical and a flat name of the content to be distributed to achieve the hybrid name. The proposed model's implementation may cause issues such as content caching and generate numerous privacy concerns about content delivery.

2.3 Content Integrity in Named-Data Network

Contents are retrieved by name from network caches and the content server in Named Data Networking. This element raises serious issues about content integrity security. Due to its high computational overhead, the built-in verification technique is not practicable. Several NDN-specific hazards have been identified, according to study. The most common security attacks are interest flooding and cache poisoning. In [6] it is said that cache poisoning attacks cause routers to fill their caches with fictitious data. In terms of assault purpose, they are divided into two categories: local poisoning and content poisoning.
Content poisoning is a major problem in which valid content is isolated from the network. It starts with the placement of bogus content in the content store, which is then transferred to the receiver, contaminating all intermediary routers. To address

this problem, NDN includes a built-in security mechanism based on digital signatures that prevents worldwide poisoning by ensuring that poisoned information is appropriately removed from content store (CS). Previous study, on the other hand, has found that signature verification has too much computing cost.

2.4 Intrusion Detection using Blockchain

Anomaly and signature methods are used in blockchain-based intrusion detection models, which all have distinct constraints that blockchain technology can solve. The anomaly detection technique generates significant false alarm rate and fails to detect encoded message sent by attackers. It also has trouble creating a standard profile for complex systems, has uncategorized warnings, and requires basic training.The primary drawback of the signature detection technique, on the other hand, is that it cannot identify a new cyberattack in the system.

By studying pattern behavior for each node, the k-means algorithm is employed in [9] to distinguish between fraudulent and regular nodes in a blockchain network. It is capable of efficiently handling network nodes and transactions, as well as properly classifying nodes. However, the difficulty with this model is that it utilizes a mean value for each cluster, which may result in an incorrect cluster head being picked. Furthermore, it uses a static distance metric rather than a dynamic one.

The model in [11] detects anomalous behaviors of participants in a blockchain network using game theory and supervised machine learning methods. It calculates the likelihood of each assault depending on the value of the transaction. [14] provides a method for detecting anomalous behavior in a blockchain network. It is a useful device capable of detecting many sorts of suspicious transactions in a blockchain network. The disadvantage is that it only works well when there are repeating attacks in the network.

2.5 Related Works

This section examines previous research on NACK detection and its significance. Even though network-layer NACKs for warning downstream routers about forwarding errors have been proposed, NDN presently does not implement this feature.

Several tests have been conducted to determine how to mitigate such attacks and the importance of having these packets in the event of a smooth data transfer in order to secure cNACKs and fNACKs for all potential DoS attacks on content providers that have previously been implemented. Bloom Filter was proposed by the authors of [1]. Bloom filters are generated on a regular basis by producers, or whenever new content items are uploaded. As a result, routers will cache them and use them to satisfy outstanding interests, so clearing the PIT entries associated with them. These filters, such as BLM-FLTR, can be implemented as specific-type content objects. Bloom filter is a randomized data structure that may be used to describe a collection and facilitate membership queries. It can quickly determine if a given piece of data belongs in a specific collection. The main distinction is that Bloom filters will not be used to satisfy future interests if they are retained. The length of caching for these filters is determined by the Freshness value in their headers.Producers must fine-tune this number to match the frequency with which new material is published. Bloom filters should not be provided to consumers, despite the fact that they are content objects that follow the same route as their linked interests in reverse. This is because the presence of these filters offers malevolent users an edge in assaults.A bloom filter's disadvantage is that it allows for false positives. Each application's impact must be properly assessed to determine if the impact of false alarms is tolerable.
The Interest Key Binding (IKB) rule was proposed in [4] to secure cNACK. In order to offer routers with the trust context necessary to authenticate just one signature per piece of interest, the IKB rule requires coordination between consumers and producers. The digest of the producer's public key is included in all of the interests of consumers in particular. To secure cNACK, the Interest Key Binding (IKB) rule was suggested in [4]. The IKB rule necessitates collaboration between customers and manufacturers in favor to provide routers with the security environment required to validate just one sign each item of interest. In particular, the digest of the maker's public key is incorporated in all consumer interests. A producer, on the other hand, incorporates its public key in all of the data it distributes. Before caching or forwarding data, a router can choose to validate content authentication. When it receives a content, it checks the digest of the public key in the material prefix with the one

provided in the similar query, and it validates the material signing against the public key in the information prefix. As a result, the IKB regulation prohibits anybody other than legitimate producers from updating or creating cNACKs.

To summarize, NDN is intended to alleviate the constraints of today's IP-based Internet. The use of NACKs at the network level is a controversial element of NDN. This debate also suggests that, despite their numerous benefits, NACKs have security implications. As a result, further study should be done to solve these issues.

Chapter 3

Problem Domain

In this chapter, we will go through the specifics of the issue formulation as well as the framework of the suggested solution. It is critical to recognize intruders and their created fake NACKs in the NDN network for our problem statement though It is simple to verify the origin and integrity of NACK. In fact, one of NDN's fundamental ideas is that every material must be signed by the creator. However, introducing a specific category of material that is not required to be signed would be a violation of this concept. Before caching or forwarding material, a router might choose to check content signatures. However, for numerous reasons explained in [4], this procedure is not required.

Assume that an intruder has control of a network link and has the ability to insert cNACKs for all interests passing across that link. In this instance, intruders may be able to block customers from accessing legitimate content. Intruders only need to inject cNACKs on one path to succeed in the attack, even if there are numerous paths to the producer; this is a different type of cNACK invasion. Another form of cNACK attack is, intruder declares an interest for material that has not yet been produced, anticipating the name of content that has not yet been generated, and obtains a real cNACK from the true producer. This cNACK is cached by the path's routers. Even if the real material is released soon after, further requests for it will be met by a cached cNACK, resulting in a DoS attack.

The following are the conditions for securing cNACKs, according to [5]:

1. **Signature:** A cNACK, like any other piece of work, must be signed by the creator.

2. **Timestamps:** Not per interest, but per time period, a cNACK should be

created.

3. **Expiration:** The expiration period should be included in a cNACK for plausible material (i.e. not yet published).

We offer a blockchain architecture that may appropriately solve these difficulties while keeping the essential guarantees of blockchains and NDN networks in the following sections.

3.1 Our Approach

3.1.1 Fake cNACK Prevention

We presented a solution in which producers are trusted based on their reliability to the network's other producer nodes. The producers' reliability will be judged by how long they have been in the network and how much content they have published. This is the foundation of our authentication system. We referred to it as *Proof-of-Reliability* (PoR). A producer that wants to verify the legitimacy of its created cNACK must submit a properly prepared transaction, known as a Proof-of-Reliability claim, to the blockchain network. The transaction includes-

- The hash of the cNACK to authenticate.
- On the claim, there is a signature and an encrypted message that confirms how long the producers have been in the network and how many items they have released.
- The public key of the producer is used to validate the message and identify the sender.
- The previous block's hash is used to avoid transaction pre-signing.
- The maximum age in the block header that denotes the cNACK's validity.

The producer sends the transaction to the blockchain network. The producer can construct the next Proof-of-Reliability transaction referring to the previous block's hash after all producer nodes have approved the block using the consensus process outlined below. The max-age in the header will be based on how frequently the producer publishes a content. Max-age necessitates worldwide synchronization systems, such as a secure NTP [3]. The router that forwarded the interest packet to the producer will only cache the cNACK block in its Content Store (CS) if the block is verified by the consensus protocol. After the max-age in the header, the cNACK will be immediately deleted from CS. In the next sections, we'll go over our blockchain-based solution in more depth.

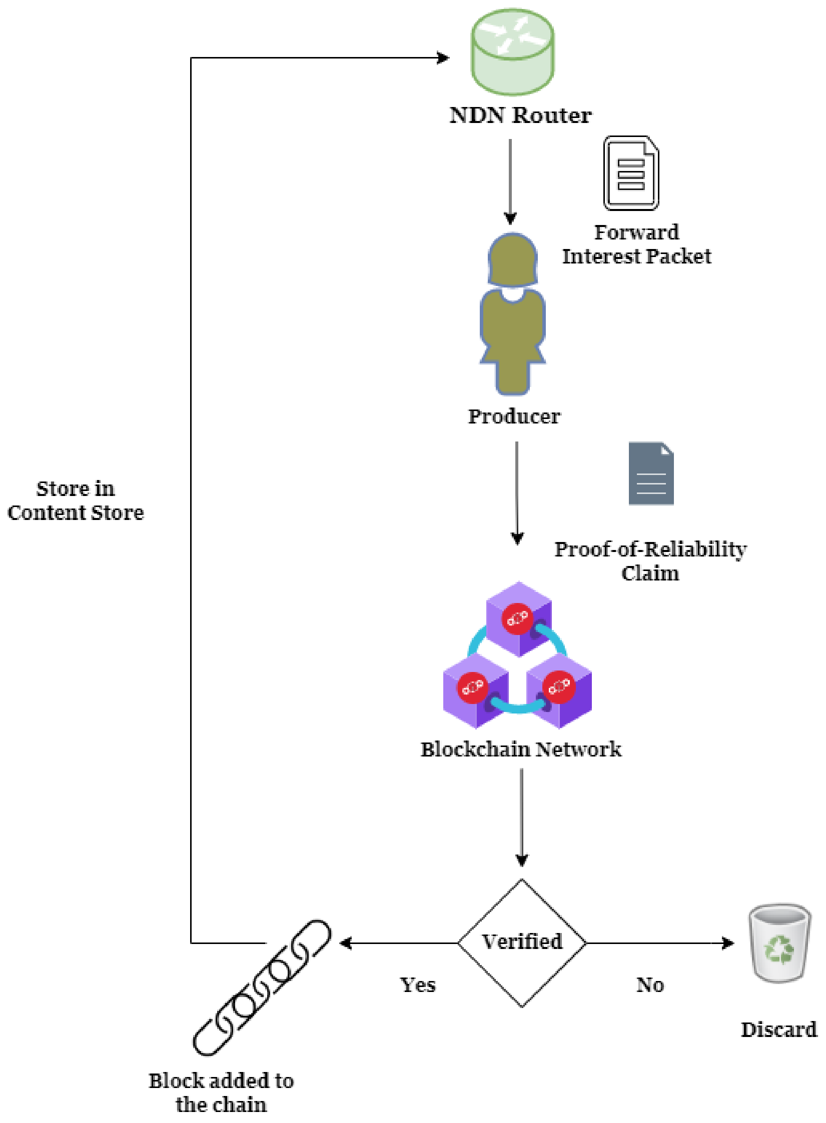


Figure 3.1: Workflow of the cNACK in the Proposed Model

3.1.2 Fake fNACK Prevention

In an NDN network, if a router receives a fNACK on every upstream interface specified in the FIB, it must forward fNACKs on all downstream interfaces (where interests were received). Before transmitting the fNACKs to the downstream routers, each router in our proposed framework must submit them to the trusted blockchain network. A fNACK producer must provide a properly prepared transaction, Proof-of-Reliability (PoR) claim to the blockchain network, just like a cNACK producer has to submit a properly prepared transaction, PoR claim to the blockchain network to validate the legality of its generated fNACK.The transaction consists of the following:

- The hash of the fNACK to authenticate
- There is a signature and an encrypted message on the claim that certifies how long the routers have been in the network and how many packets they have forwarded.
- The router's public key is used to validate and identify the messages.
- To avoid transaction pre-signing, a hash of the prior block is utilized.
- The maximum age in the block header that denotes the fNACK's validity.

After verifying the transaction, the blockchain network will provide reliability score and will add the transaction into the chain. Once the blockchain network has accepted the fNACKs, they will be sent to the downstream router. The downstream router will issue a Reliability Score request to the blockchain network, together with the hash fNACK and the upstream router's public key, before accepting the fNACK. The downstream router will next determine if the Reliability Score is high enough to trust the fNACK.

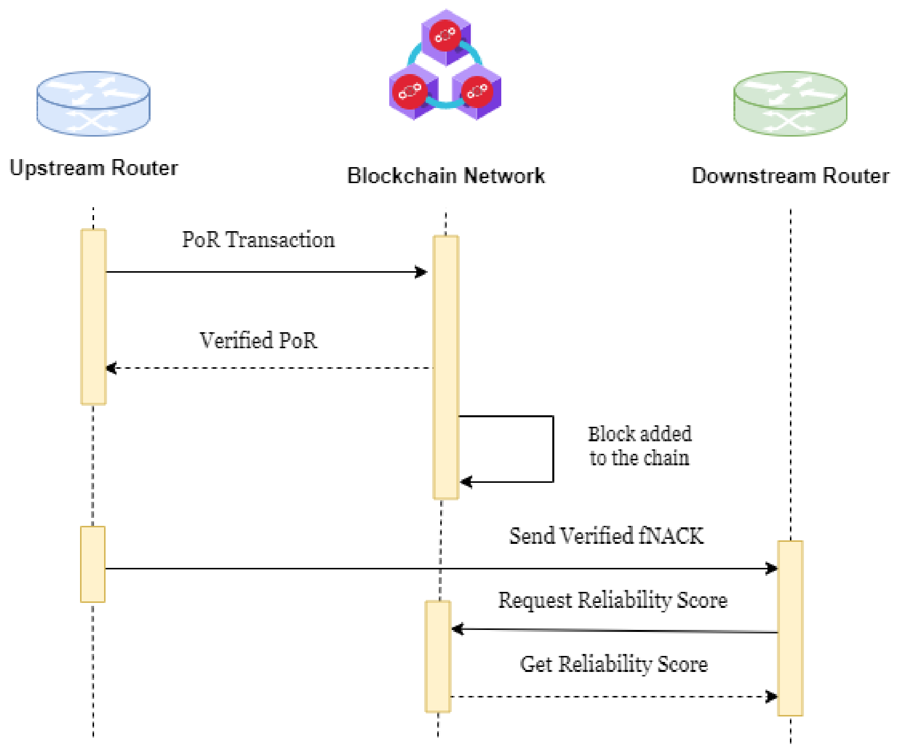


Figure 3.2: Workflow of the fNACK in the Proposed Model

3.2 Blockchain Layer

Only producers can provide a Reliability Score to a producer in our proposed model, and only producers would participate in the consensus procedure. We limit the number of participants in the consensus process by restricting access rights to producers, resulting in speedier transaction finality and increased throughput while reducing the end user's ledger maintenance expense. Because NDN nodes will be free of the trust network overhead, we can assume that our blockchain layer and transport layer will be separated. This will reduce unnecessary complexity. However, we'll assume for the rest of the article that an NDN node will serve as both an NDN and a blockchain node.The implementation of the network will determine whether one node should perform both functions or one of them.

3.3 Reliability Score:

There are two states of content authenticity in the graph infection protocol [3], authenticated and unauthenticated, which is quite restrictive. As a result, we included a Reliability Score in our suggested model, which enhances authentication accuracy. The score will be determined by the number of prior blocks created by that particular producer, as determined by the other producer nodes in the blockchain network.

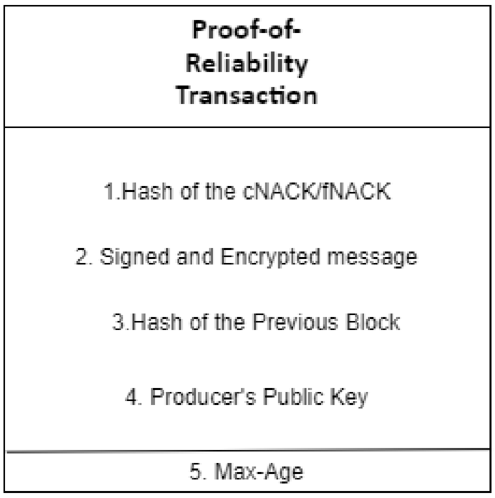

Proof-of-Reliability Transaction
1.Hash of the cNACK/fNACK
2. Signed and Encrypted message
3.Hash of the Previous Block
4. Producer's Public Key
5. Max-Age

Figure 3.3: Structure of Proof-of-Reliability

In the network no one can modify the prior transaction history and every node has the same blockchain database stored in them. We anticipate that an intruder will not be able to stay in the network for long. He/she also won't be able to generate as much content as a true producer. There will be a threshold value for trusting a cNACK, which will be determined by the network's deployment. The consensus algorithm will suspend the authentication process if an attacker attempts to construct a bogus cNACK.

3.4 Consensus Protocol

In a centralized network, the central authority is in charge of transaction validation, which ensures the network's security and trust. In a dispersed and decentralized network, however, the peers only engage with one another because there is no central authority. In this situation, for better network maintenance, all peers must adhere to a set of network-defined norms and reach an agreement on a single block that will be added into the chain. Consensus algorithms are used in the blockchain to establish this agreement. There are several consensus algorithms available today, each of which is employed by a distinct form of blockchain.In blockchain applications, for example, Proof-of-Work (PoW), Proof-of-Stake (PoS), Proof-of-Authority (PoA), and multiple derivations of the Byzantine Fault Tolerance (BFT) protocol are all employed. Every protocol has its own set of benefits and drawbacks. PoW, for example, is a decentralized way of confirming transactions that also pays miners in cryptocurrency. However, the difficulty with PoW is that it consumes a lot of energy and frequently necessitates expensive equipment. Furthermore, PoW has several restrictions in terms of performance. The first is that PoW has a poor throughput since the average number of transactions that may be committed per second is 7. PoW also has a high latency because a block can only be declared genuine once at least 6 verifed blocks have been added to the chain. PoS, on the other hand, allows for quick and low-cost transaction processing, but validators with big holdings may have undue influence on transaction verification. This is unjust since the wealthier participants have a higher chance of becoming forgers than the others, and PoS only strengthens the network's wealthier participants. The Federated Byzantine Agreement (FBA) and its blockchain implementation, the Stellar Consensus Protocol (SCP) [17], appear to be the best fit for our requirements.

3.5 Byzantine Fault Tolerance

The Fault Tolerance of Byzantium is the capacity of a computer system to continue operating even if some of its nodes fail or act maliciously. The name derives from the hypothetical Byzantine Generals' Problem. For a group of Byzantine generals, this was the logical problem. Each general is assigned an army and a location near a citadel, and they must choose whether to attack or retreat as a unit. They will be victorious if they all make the same decision. The battle, on the other hand, is lost if a misunderstanding or betrayal compels certain generals to charge while others retire.This kind of challenges are referred to as Byzantine flaws. Each node in a multi-node computer system might be called a "general." The Byzantine Fault

Tolerance of a system relates to whether it can continue to function even if some nodes fail or try to fool it.

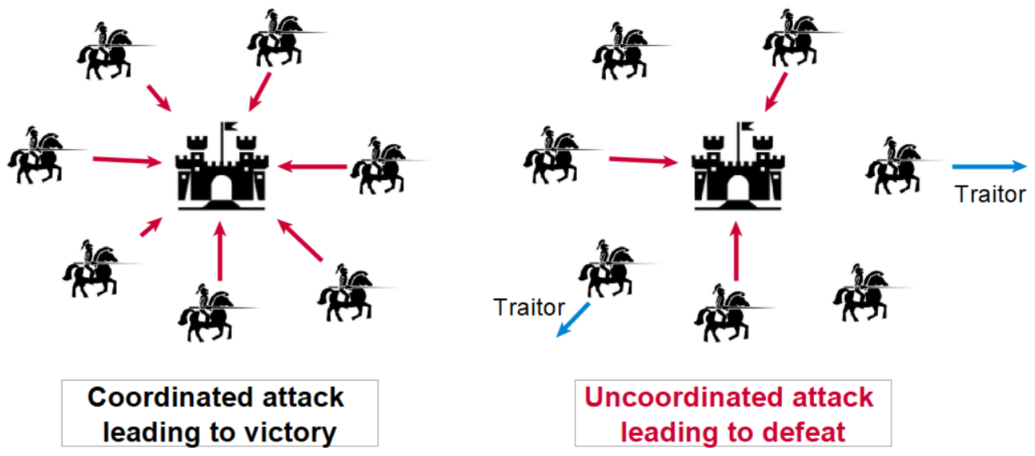

Traitor
Traitor
Coordinated attack leading to victory
Uncoordinated attack leading to defeat

Figure 3.4: Byzantine Generals' Problem

3.6 Federated Byzantine Agreement (FBA)

Federated Byzantine Agreements (FBA) are a type of Byzantine fault tolerance in which each byzantine general is in charge of their own blockchain. Because of its high throughput, network scalability, and cheap transaction costs, a Federated Byzantine Agreement (FBA) is deployed. Before users seek any performance from the FBA, nodes must be known and validated ahead of time. Individual nodes in the FBA network make judgments about who they trust, and quorums of nodes develop as a result of those decisions. The FBA makes use of quorum slices, which are subsets of quorums that can convince certain network nodes to agree with them.

3.7 Stellar Consensus Protocol (SCP)

The Federated Byzantine Agreement (FBA) consensus protocol is used by the Stellar Consensus Protocol (SCP). The membership of FBA systems is open and control is decentralized. Nodes have the ability to pick who they trust. Individual nodes make decisions that lead to system-wide quorums. A quorum is the minimum number of nodes necessary for a system to attain consensus. To avoid network partitioning, any two quorums must have at least one node overlap, which is known as the quorum-intersection condition. If there is no quorum-intersection, the network will come to a standstill until the nodes' administrators modify it. When the nodes come together to agree on a single version of the ledger, they begin a voting procedure inside their quorum, eventually achieving a consensus. SCP is quick and light, and a node with a lot of processing power doesn't gain anything. A new node cannot contribute to the protocol unless it is trusted by another node in the network.

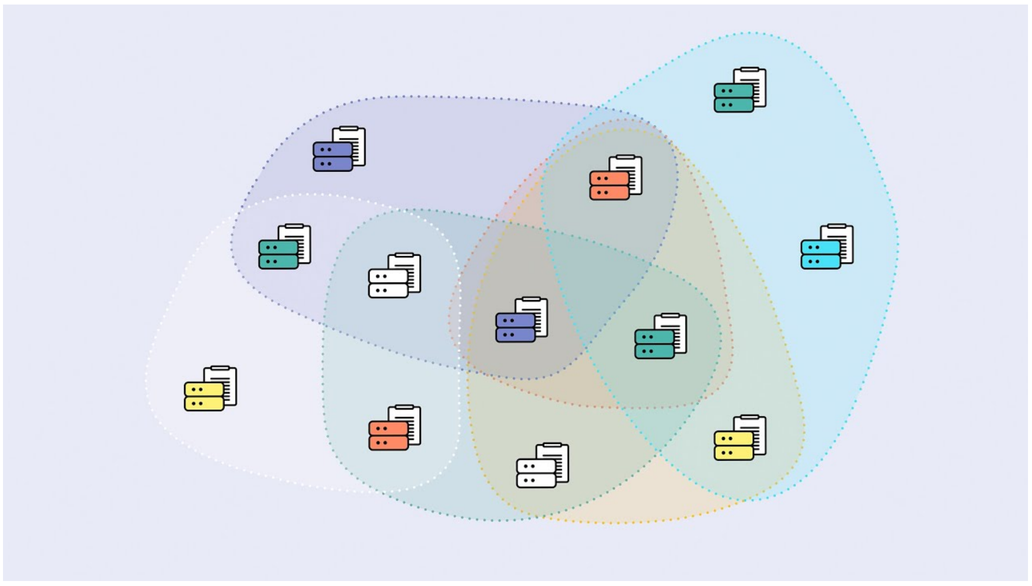


Figure 3.5: Stellar Consensus Protocol

According to FLP impossibility [2], all of those consensus protocols face the trilemma, where only two of three qualities can be attained. Fault tolerance, Safety, and Liveness are the three qualities. Fault tolerance, which is defined as the capability to handle node failure, is chosen by the majority of consensus protocols. Safety prevents nodes from dividing if they cannot agree on a transaction. The nodes wait for the dispute to be resolved before proceeding. Liveness, on the other hand, ensures that the system will always respond to new transactions and accept them. When a conflict arises, the system is split into two states, with the intention that one of them will be rejected at some time, restoring consistency. The majority of Nakamoto Consensus protocols prioritize Liveness over Safety. However, there is an issue with the confirmation period. Although it is difficult to alter and recalculate a huge number of blocks, it is doable, and the attack may be carried out effectively [18]. Make certain that the chain in which our transaction was included will not be forked in favor of a longer chain. In fact, many systems demand a certain number of confirmations to trust a transaction. 2–3 confirmations are often required for Bitcoin, with each confirmation taking on average 10 minutes (20–30 total). The number of confirmations in Ethereum is larger, about 15–30, but each confirmation takes on average 15 seconds (3–7.5 minutes overall). Safety is chosen via SCP and other vote-based methods. The ledger is closed and cannot be reversed once the network's quorum agrees on the state. Furthermore, the confirmation time is significantly reduced. The ledger in Stellar closes in 3–5 seconds.

Not all blockchain consensus algorithms are viable as Internet-wide solutions that must operate on resource-constrained devices and reach consensus on a millisecond time frame. For example, [13] claims that Stellar consensus is too slow for IoT devices. Nonetheless, in our instance, Proof-of-Reliability assertions are based on the length of time content creators have been in the network and the amount of content they have created. As a result, we feel our answer is enough. Furthermore, we discover that a recent virtual-voting version of FBA [17] may establish agreement with nearly little communication overhead, which is highly encouraging.

3.8 A Transaction's Lifecycle in Steller

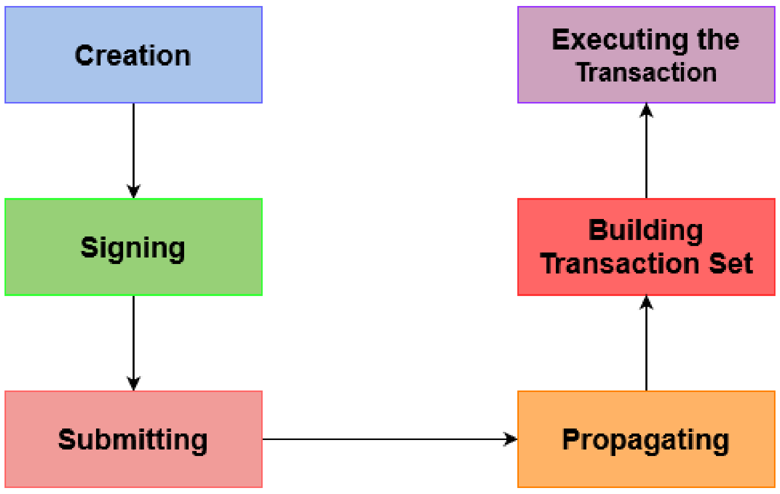


Figure 3.6: Lifecycle of Steller

Creation: The client creates a transaction, fills in all of the data, assigns it the correct sequence number, and adds any operations it wants.
Signing: Transactions must be signed using the sender's public keys, and many parties' signatures may be necessary in complicated transactions.
Submitting: The transaction should now be valid, and it may be signed and uploaded to the Stellar network. Transactions are usually transmitted using horizon, but they can also be delivered to a stellar-core instance directly.
Propagating: When stellar-core gets a transaction from a customer or another stellar-core, it runs preliminary checks to see whether it's valid. It guarantees that the transaction is properly formed and that the issuing account has adequate cash to cover the transaction charge, among other things.
Building Transactions Set: When the moment comes to close the ledger, stellar-core creates a transaction collection from all of the transactions that it has received since the previous one had ended. If it receives any new transactions now, it files them away until the next ledger close. The transaction set that Stellar-core has gathered is named. SCP is in charge of settling conflicts among the many transaction sets and chooses the network's preferable transaction set.
Executing the Transactions: The ledger is updated when SCP has approved on a transaction set. At this stage, a charge is debited from the origin account for each transaction in the set. The activities are carried out in the transaction's sequence of appearance. If any transaction fails, the transaction as a whole fail, and the transaction's previous actions' effects are undone.A new ledger is produced when all of the transactions in the set have been applied, and the process begins again.

3.9 Producer's Workflow (cNACK)

A producer does the following to publish cNACKs on the NDN network and authenticate it using the blockchain node (Figure: 3.7).

1. Generate a cNACK and sign using the producer's key pair.
2. Compute hash of the cNACK.
3. Create a Proof-of-Reliability transition that includes the hash of the cNACK

and preceding blockchain block's hash, as well as a signed and encrypted message proving the producer's trustworthiness.

4. The encrypted message will include the total number of pieces of content the producer has published as well as the length of time the producer has been a member of the network.

5. Add a timestamp (max-age) in the header of the block.

6. Publish the Proof-of-Reliability transaction on the network.

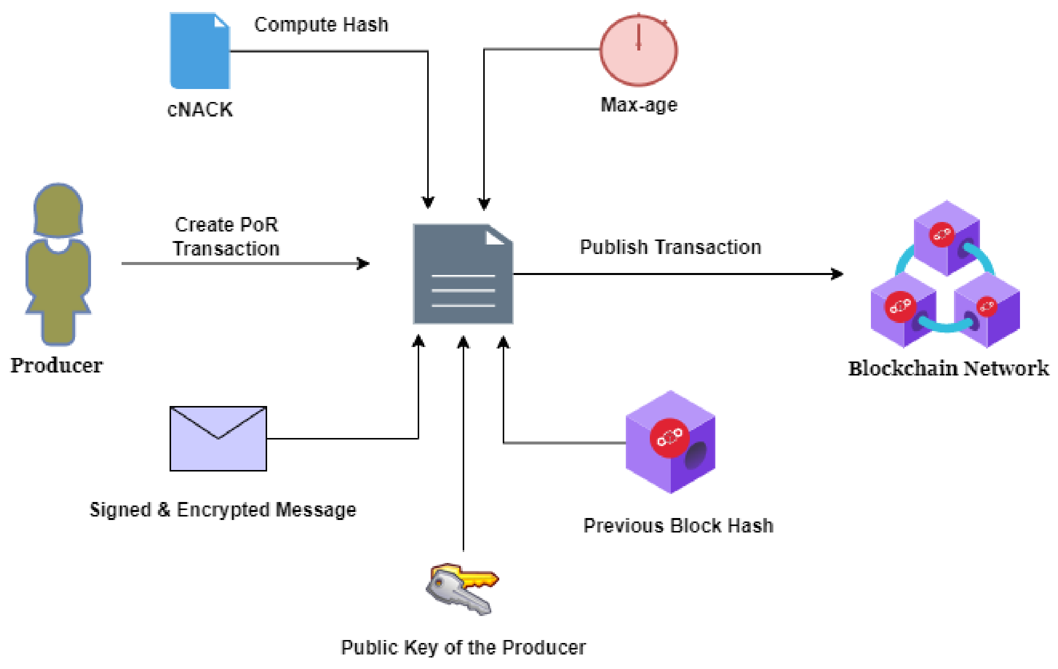


Figure 3.7: Producer's Workflow

3.10 Consumer's Workflow (cNACK)

A consumer does the following to get material from the NDN network and authenticate it using a blockchain node (Figure 3.9):

1. Send an interest packet to an NDN router. The NDN router will return a cNACK if the content is not available to the producer or for some reason the producer does not want to share it.

2. Send a Reliability Score request along with the hash cNACK and the producer's public key to the NDN nodes and receive it.

3. Decide if the Reliability Score is sufficient to trust the cNACK.

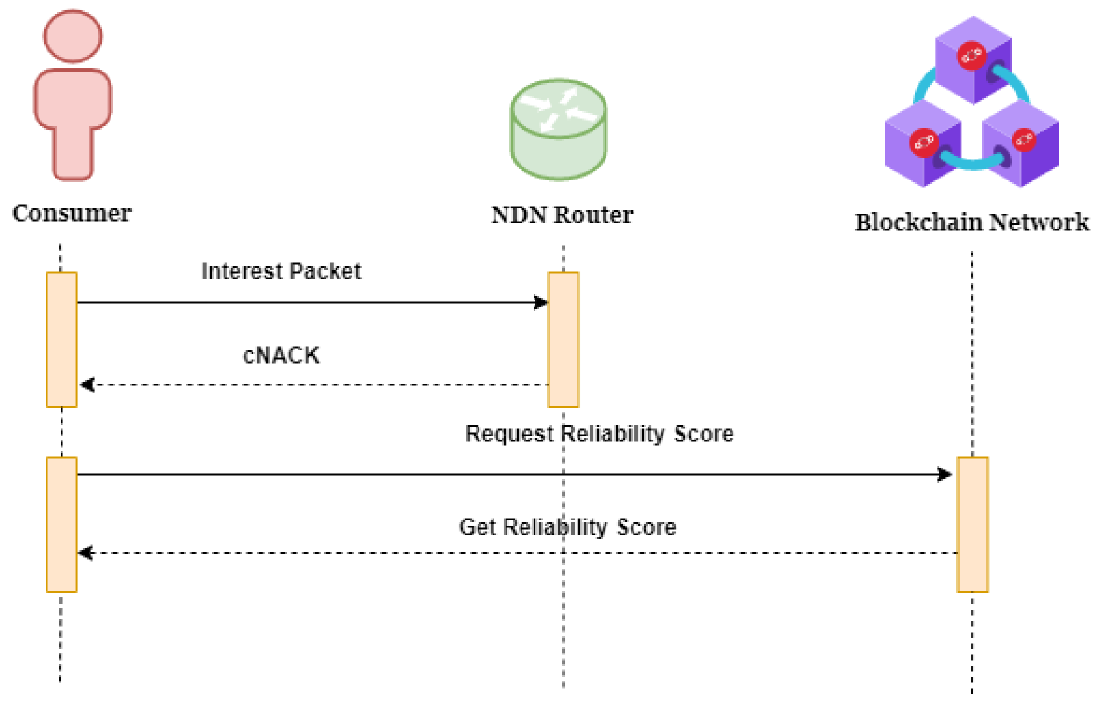


Figure 3.8: Consumer's Workflow

Chapter 4

Implementation

We designed a test topology and a test blockchain to implement our proposed model. In our Linux operating system, we used ndnSIM 2.8 to create the test topology. Python SDK was used to generate the test blockchain. The core SDK, as well as any dependencies, will be installed using this utility. For Horizon endpoints, the Stellar Python SDK provides a networking layer API. It also lets us to create and sign transactions, connect with a Stellar Horizon instance, and examine network history.
We've included the code and a diagram of the test architecture we designed below.

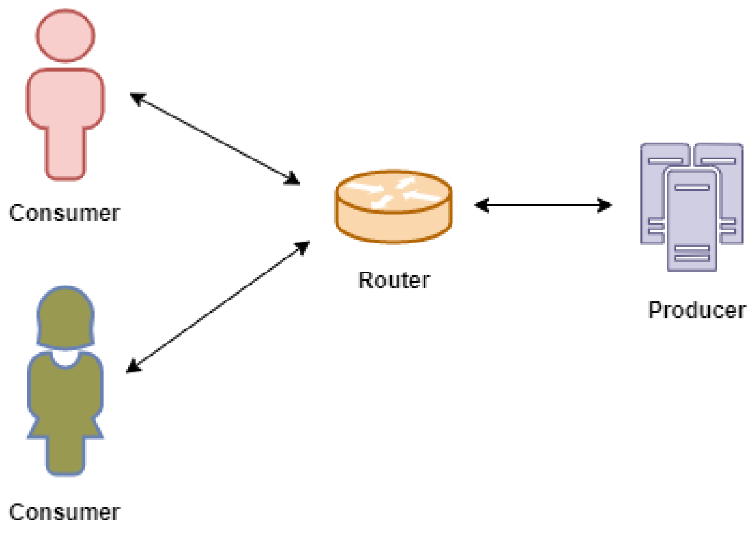


Figure 4.1: Diagram of the Test Topology

```
#include ”ns3/core-module.h”
#include ”ns3/network-module.h”
#include ”ns3/point-to-point-module.h”
#include ”ns3/ndnSIM-module.h”

namespace ns3 {

int
main(int argc, char* argv[]){

Config::SetDefault(”ns3::PointToPointNetDevice::DataRate”, StringValue(”2Mbps”));
Config::SetDefault(”ns3::PointToPointChannel::Delay”, StringValue(”5ms”));
Config::SetDefault(”ns3::DropTailQueue<Packet>::MaxSize”, StringValue(”30p”));

CommandLine cmd;
cmd.Parse(argc, argv);

NodeContainer nodes;
nodes.Create(4);

PointToPointHelper p2p;
p2p.Install(nodes.Get(0), nodes.Get(2));
p2p.Install(nodes.Get(1), nodes.Get(2));
p2p.Install(nodes.Get(2), nodes.Get(3));

ndn::StackHelper ndnHelper;
ndnHelper.SetDefaultRoutes(true);
ndnHelper.InstallAll();

ndn::StrategyChoiceHelper::InstallAll(”/prefix”, ”/localhost/nfd/strategy/multicast”);

// Consumer
ndn::AppHelper consumerHelper(”ns3::ndn::ConsumerCbr”);
consumerHelper.SetPrefix(”/prefix”);
consumerHelper.SetAttribute(”Frequency”, StringValue(”30”));
auto con1 = consumerHelper.Install(nodes.Get(0));
auto con2 = consumerHelper.Install(nodes.Get(1));
con1.Stop(Seconds(30.0));
con2.Stop(Seconds(30.0));

// Producer
ndn::AppHelper producerHelper(”ns3::ndn::Producer”);

// Producer will reply to all requests starting with /prefix
producerHelper.SetPrefix(”/prefix”);
producerHelper.SetAttribute(”PayloadSize”, StringValue(”1224”));
producerHelper.Install(nodes.Get(3));

Simulator::Stop(Seconds(500.0));
Simulator::Run();
Simulator::Destroy();
return 0;
}
}
int
main(int argc, char* argv[])
{
return ns3::main(argc, argv);
}
```

Table 4.1: Code of the Implemented Topology

Each node in our test topology must create a transaction account in the Stellar blockchain network in order to send and receive packets. In Stellar, accounts are the most important data structure. They keep track of sign transactions and distribute assets. A given account owns all entries that remain on the ledger. Each node needs its own valid keypair before generating an account. To maintain the security of each transaction, Stellar employs public key cryptography. A keypair consists of a public key and a secret key for each stellar account. Nodes need the public key to identify themselves to the other nodes in the network, hence sharing it is always safe. We only require the secret key because nodes in NDN have a built-in public key. Stellar Laboratory was used to create accounts and conduct transactions. The Stellar Laboratory is a suite of tools that allows individuals to experiment with the Stellar network and learn more about it. The laboratory can create, sign, and submit transactions to the network. It can also communicate with any Horizon endpoint.

```
from stellar_sdk import Keypair
pair = Keypair.random()
print(f"Secret Key: {pair.secret}")
#Secret Key: SC4XGDPCDEK6L3J7SRKIPD5KKJYHDIF4W7TCCJWJLBEIB2ATBUUJXQNP
```

Table 4.2: Secret Key Generation Code

```
import requests
public_key = "GC4NSMMEVMKWGWJM5XSNNIAGDJ5336X67BOOQOTSRB7HWINTNXHA6LEF"
response = requests.get(f"https://friendbot.stellar.org?addr={public_key}")
if response.status_code == 200:
    print(f" NEW ACCOUNT Created :)\n{response.text}")
else:
    print(f"TRY AGAIN Response: \n{response.text}")
```

Table 4.3: New Account Creation Code

A valid keypair, on the other hand, does not establish an account. The next stage is to create an account. To do so, nodes must provide the public key they generated to Friendbot, , a tool for supporting friendly accounts. It'll make a new account with the public key as the account ID.

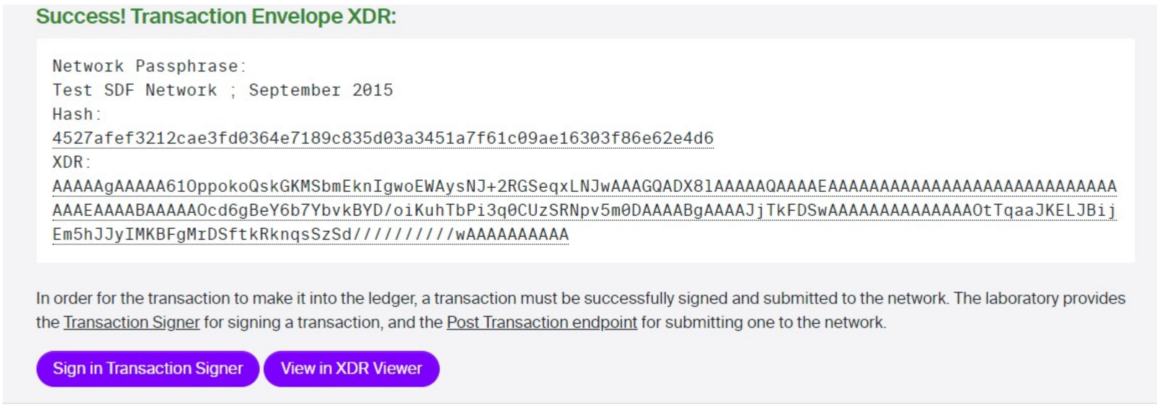
Nodes can begin transacting on the network after creating an account.

Operations are the actions that take occur on Stellar, such as sending and receiving packets or buying and selling offers. Participants bundle operations into transactions to submit them to the network. A transaction is a collection of one to 100 operations plus some extra information, such as the account that is initiating the transaction and a cryptographic signature to verify that it is genuine. In our case, this transaction is known as PoR.

Stellar stores and transmits transaction data in a binary format called XDR, which is optimized for network efficiency but unreadable to the naked eye. Fortunately, Horizon, the Stellar API, and the Stellar SDKs convert XDRs into more user-friendly formats.

The final step is to obtain the account's information and check its status. Multiple information can be stored in a single account.

The next step is to build a trustline, which indicates that the distribution account has faith in the issuing account. For that we used Stellar Laboratory. Stellar Laboratory was used to create accounts and conduct transactions. The Stellar Laboratory is a suite of tools that allows individuals to experiment with the Stellar network and learn more about it. The laboratory can create, sign, and submit transactions to the network. It can also communicate with any Horizon endpoint. To create a trustline, go to the Build Transaction page and fill in the source account box with the public key for the distribution account. Then, click on “Fetch next sequence number for account starting with..” After clicking, we will get a transaction sequence number. We can keep the next three fields: Memo, Base Fee and Time Bounds blank and move to field and set that to “Change Trust.” After selecting operation type as “change trust”, set the asset to alphanumeric 12 based. We are creating a token for the cNACK/fNACK asset, so we selected Alphanumeric 12 and enter ‘cNACK’ in the asset code field and finally added issuing accounts public key in the “Issuer Account ID.”

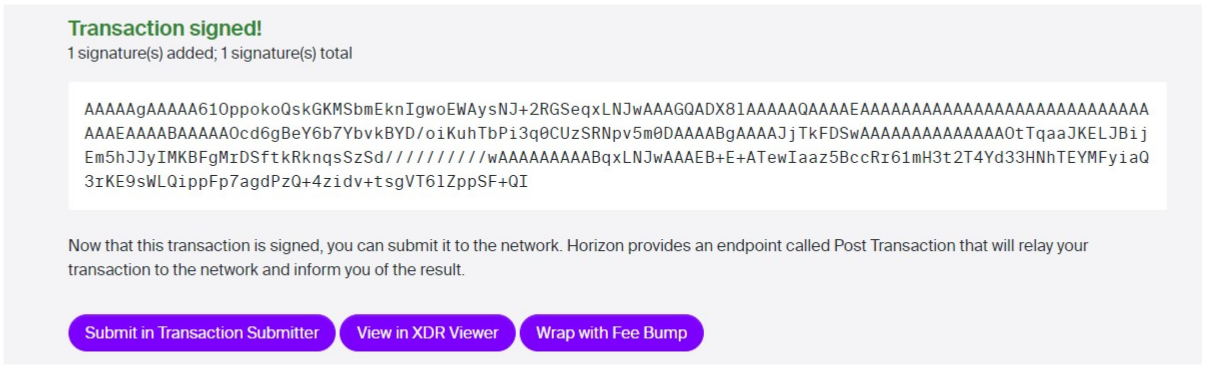

Success! Transaction Envelope XDR:

```
Network Passphrase:
Test SDF Network ; September 2015
Hash:
4527afef3212cae3fd0364e7189c835d03a3451a7f61c09ae16303f86e62e4d6
XDR:
AAAAAgAAAAA61OppokoQskGKMSbmEknIgwoEWAysNJ+2RGSeqxLNJwAAAGQADX8lAAAAAQAAAAEAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAEAAAABAAAAAOcd6gBeY6b7YbvkBYD/oiKuhTbPi3q0CUzSRNpv5m0DAAAABgAAAAJjTkFDSwAAAAAAAAAAAAAAAOtTqaaJKELJBij
Em5hJJyIMKBFgMrDSftkRknqsSzSd//////////wAAAAAAAAAAAA
```

In order for the transaction to make it into the ledger, a transaction must be successfully signed and submitted to the network. The laboratory provides the Transaction Signer for signing a transaction, and the Post Transaction endpoint for submitting one to the network.

Figure 4.2: Transaction Envelope XDR Screenshot

At the bottom of the page, click the ”Sign in Transaction Signer” button to complete the transaction. It will go to the signatures section’s next page. In the ”Add Signer” area, we’ll enter the distribution account secret key. If we were doing it in real life, we would use a ledger, but because this is only an example, we would paste the key as indicated before. Scroll to the bottom of the page and click ”Submit in transaction submitter” once the secret key has been pasted.


Transaction signed!

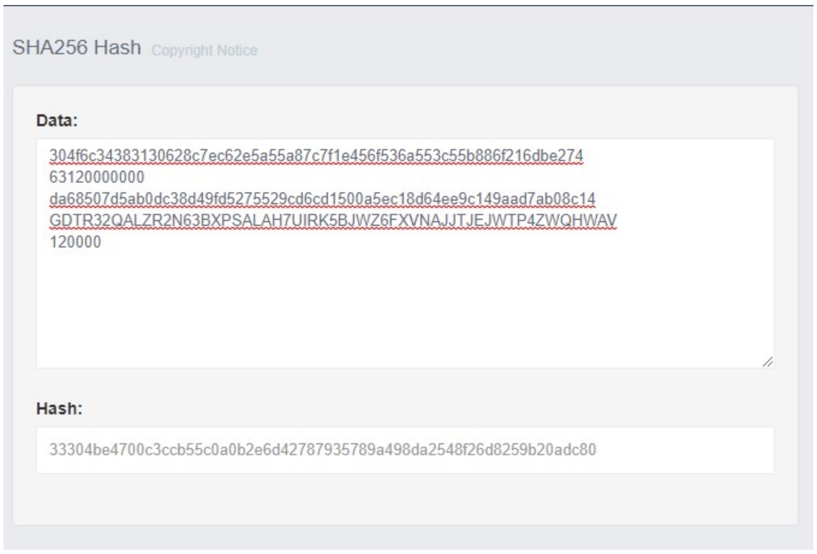
1 signature(s) added; 1 signature(s) total

```
AAAAAgAAAAA61OppokoQskGKMSbmEknIgwoEWAysNJ+2RGSeqxLNJwAAAGQADX8lAAAAAQAAAAEAAAAAAAAAAAAAAAAAAAAAAAAA
AAAEAAAABAAAAAOcd6gBeY6b7YbvkBYD/oiKuhTbPi3q0CUzSRNpv5m0DAAAABgAAAAJjTkFDSwAAAAAAAAAAAAAAOtTqaaJKELJBij
Em5hJJyIMKBFgMrDSftkRknqsSzSd//////////wAAAAAAAAABqxLNJwAAAEB+E+ATewIaaz5BccRr61mH3t2T4Yd33HNhTEYMFyiaQ
3rKE9sWLQippFp7agdPzQ+4zidv+tsgVT6lZppSF+QI
```

Now that this transaction is signed, you can submit it to the network. Horizon provides an endpoint called Post Transaction that will relay your transaction to the network and inform you of the result.

Figure 4.3: Screenshot of Successful Transaction Sign

Backend Workflow: We encrypted the data to reduce the amount of data kept for each PoR transaction. For each batch of data, the data creates a unique hash.That hash is stored in the Block, which saves space and time.We utilized SHA256 for our implementation.



Data:

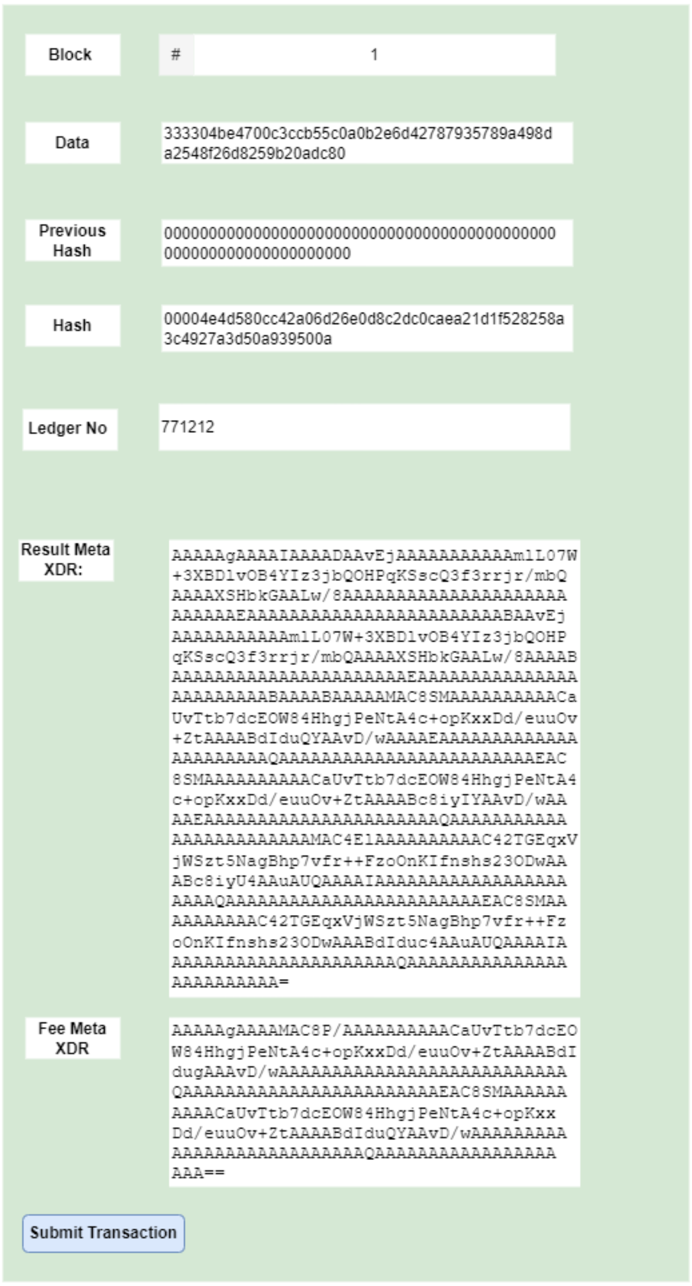
```
304f6c34383130628c7ec62e5a55a87c7f1e456f536a553c55b886f216dbe274
63120000000
da68507d5ab0dc38d49fd5275529cd6cd1500a5ec18d64ee9c149aad7ab08c14
GDTR32QALZR2N63BXPSALAH7UIRK5BJWZ6FXVNAJJTJEJWTP4ZWQHWAV
120000
```

Hash:

```
33304be4700c3ccb55c0a0b2e6d42787935789a498da2548f26d8259b20adc80
```

Figure 4.4: Generated SHA256 Hash from PoR

Once the hashed PoR is submitted to the blockchain network, the trustworthy producer nodes perform the consensus process to generate the transaction hash, ledger number, result, and fee meta in XDR. The following block uses the hash of the previous block to form the blockchain. The chain becomes longer as additional blocks are placed to the end. The hash is reset if any value in the block is altered, and the block must be created again to receive a new hash. Any update to a block in the center of the blockchain invalidates all future blocks since the new hash will be included in the next block. The block's hash will be reset when a new hash is added, and it will need to be validated again. Because the blockchain is a decentralized ledger, every client will have a copy of the original blockchain, making any node with a modified block easily identifiable. The neighbors must agree on which blockchain is valid before any data can be verified.



Block # 1

Data 333304be4700c3ccb55c0a0b2e6d42787935789a498da2548f26d8259b20adc80

Previous Hash 0000000000000000000000000000000000000000000000000000000000000000

Hash 00004e4d580cc42a06d26e0d8c2dc0caea21d1f528258a3c4927a3d50a939500a

Ledger No 771212

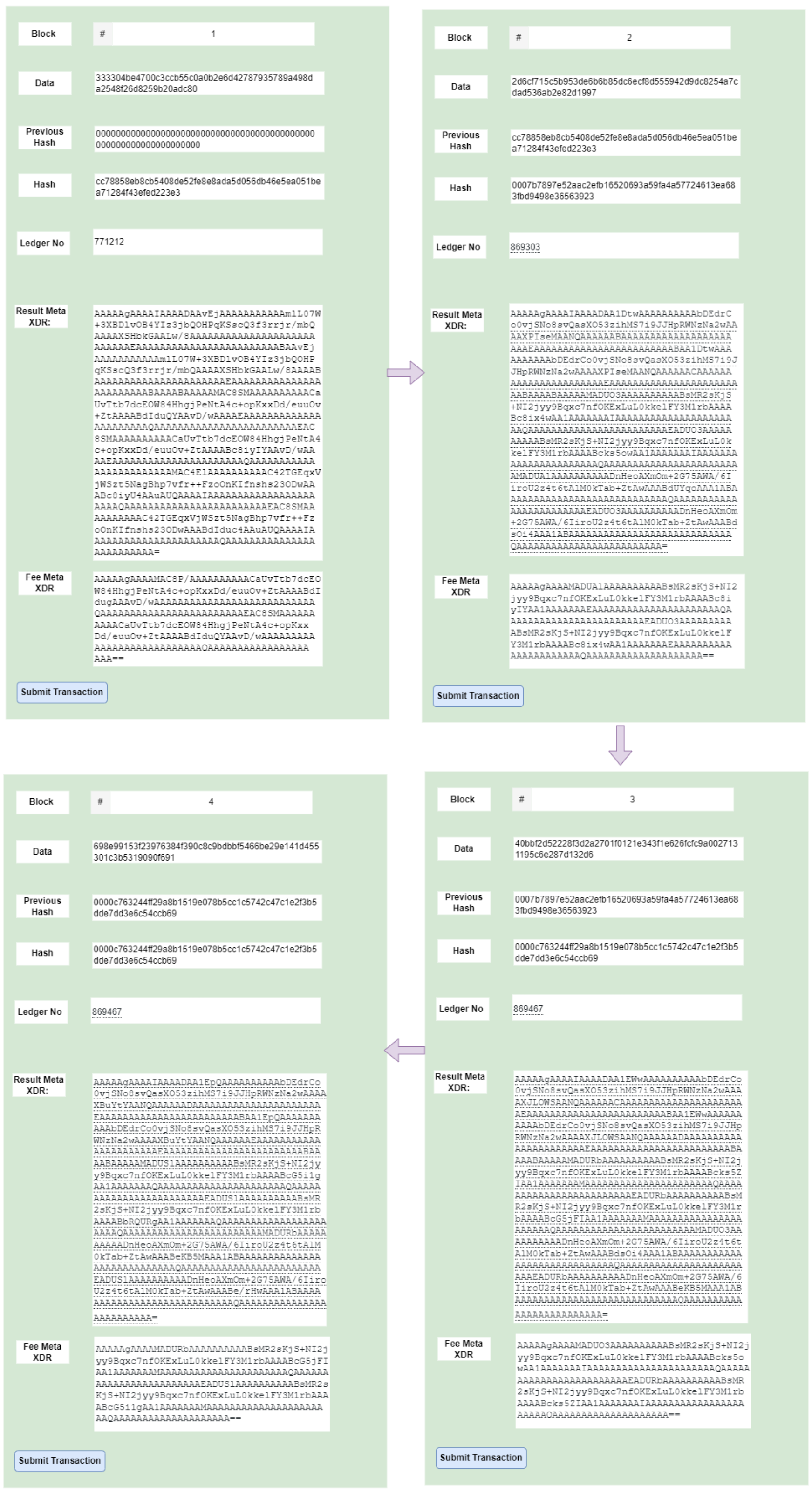
Result Meta XDR:

```
AAAAAgAAAAIAAAADAAvEjAAAAAAAAAAAAmlL07W
+3XBDlvOB4YIz3jbQOHPqKSscQ3f3rrjr/mbQ
AAAAXSHbkGAALw/8AAAAAAAAAAAAAAAAAAAAA
AAAAAAEAAAAAAAAAAAAAAAAAAAAAAAABAAvEj
AAAAAAAAAAAAmlL07W+3XBDlvOB4YIz3jbQOHP
qKSscQ3f3rrjr/mbQAAAAXSHbkGAALw/8AAAAB
AAAAAAAAAAAAAAAAAAAAAAEAAAAAAAAAAAAAAA
AAAAAAAAABAAAABAAAAAMAC8SMAAAAAAAAAAAACa
UvTtb7dcEOW84HhgjPeNtA4c+opKxxDd/euuOv
+ZtAAAABdIduQYAAvD/wAAAAEAAAAAAAAAAAAAA
AAAAAAAAAQAAAAAAAAAAAAAAAAAAAAAAAAEAC
8SMAAAAAAAAAAAACaUvTtb7dcEOW84HhgjPeNtA4
c+opKxxDd/euuOv+ZtAAAABc8iyIYAAvD/wAA
AAEAAAAAAAAAAAAAAAAAAAAAAQAAAAAAAAAAAA
AAAAAAAAAAAAAMAC4E1AAAAAAAAAAAAC42TGEqxV
jWSzt5NagBhp7vfr++FzoOnKIfnshs23ODwAA
ABc8iyU4AAuAUQAAAAIAAAAAAAAAAAAAAAAAAA
AAAAQAAAAAAAAAAAAAAAAAAAAAAAAAEAC8SMAA
AAAAAAAAC42TGEqxVjWSzt5NagBhp7vfr++Fz
oOnKIfnshs23ODwAAABdIduc4AAuAUQAAAAIA
AAAAAAAAAAAAAAAAAAAAAQAAAAAAAAAAAAAAAAA
AAAAAAAAAA=
```

Fee Meta XDR

```
AAAAAgAAAAMAC8P/AAAAAAAAAAACaUvTtb7dcEO
W84HhgjPeNtA4c+opKxxDd/euuOv+ZtAAAABdI
dugAAAvD/wAAAAAAAAAAAAAAAAAAAAAAAAAAAA
QAAAAAAAAAAAAAAAAAAAAAAAAAEAC8SMAAAAAA
AAAACaUvTtb7dcEOW84HhgjPeNtA4c+opKxx
Dd/euuOv+ZtAAAABdIduQYAAvD/wAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAQAAAAAAAAAAAAAAAAAAAA
AAA==
```

Figure 4.5: A Block in the Blockchain



Block # 1

Data
333304be4700c3ccb55c0a0b2e6d42787935789a498d
a2548f26d8259b20adc80

Previous Hash
0000000000000000000000000000000000000000
00000000000000000000

Hash
cc78858eb8cb5408de52fe8e8ada5d056db46e5ea051be
a71284f43efed223e3

Ledger No
771212

Result Meta XDR:

```
AAAAAgAAAAIAAAADAAvEjAAAAAAAAAAAml1L07W
+3XBDlvOB4YIz3jbQOHPqKSscQ3f3rrjr/mbQ
AAAAXSHbkGAALw/8AAAAAAAAAAAAAAAAAAAAA
AAAAAAEAAAAAAAAAAAAAAAAAAAAAAAAAAABAAvEj
AAAAAAAAAAAml1L07W+3XBDlvOB4YIz3jbQOHP
qKSscQ3f3rrjr/mbQAAAAXSHbkGAALw/8AAAAB
AAAAAAAAAAAAAAAAAAAAAAAEAAAAAAAAAAAAAAAA
AAAAAAAAABAAAABAAAAAMAC8SMAAAAAAAAAAACa
UvTtb7dcEOW84HhgjPeNtA4c+opKxxDd/euuOv
+ZtAAAABdIduQYAAvD/wAAAAEAAAAAAAAAAAAAA
AAAAAAAAAQAAAAAAAAAAAAAAAAAAAAAAAAAAAEAC
8SMAAAAAAAAAACaUvTtb7dcEOW84HhgjPeNtA4
c+opKxxDd/euuOv+ZtAAAABc8iyIYAAvD/wAA
AAEAAAAAAAAAAAAAAAAAAAAAAAQAAAAAAAAAAAA
AAAAAAAAAAAAAMAC4ElAAAAAAAAAAC42TGEqxV
jWSzt5NagBhp7vfr++FzoOnKIfnshs23ODwAA
ABc8iyU4AAuAUQAAAAIAAAAAAAAAAAAAAAAAAA
AAAAQAAAAAAAAAAAAAAAAAAAAAAAAAAEAC8SMAA
AAAAAAAAC42TGEqxVjWSzt5NagBhp7vfr++Fz
oOnKIfnshs23ODwAAABdIduc4AAuAUQAAAAIA
AAAAAAAAAAAAAAAAAAAAAAQAAAAAAAAAAAAAAAA
AAAAAAAAAA=
```

Fee Meta XDR

```
AAAAAgAAAAMAC8P/AAAAAAAAAACaUvTtb7dcEO
W84HhgjPeNtA4c+opKxxDd/euuOv+ZtAAAABdI
dugAAAvD/wAAAAAAAAAAAAAAAAAAAAAAAAAAAA
QAAAAAAAAAAAAAAAAAAAAAAAAAAAEAC8SMAAAAAAA
AAAACaUvTtb7dcEOW84HhgjPeNtA4c+opKxx
Dd/euuOv+ZtAAAABdIduQYAAvD/wAAAAAAAAA
AAAAAAAAAAAAAAAAAQAAAAAAAAAAAAAAAAAAAA
AAA==
```

Submit Transaction

Block # 2

Data
2d6cf715c5b953de6b6b85dc6ecf8d555942d9dc8254a7c
dad536ab2e82d1997

Previous Hash
cc78858eb8cb5408de52fe8e8ada5d056db46e5ea051be
a71284f43efed223e3

Hash
0007b7897e52aac2efb16520693a59fa4a57724613ea68
3fbd9498e36563923

Ledger No
869303

Result Meta XDR:

```
AAAAAgAAAAIAAAADAA1DtwAAAAAAAAAAbDEdrC
o0vjSNo8svQasXO53zihMS7i9JJHpRWNzNa2wAA
AAXPIseMAANQAAAAAABAAAAAAAAAAAAAAAAAAAA
AAAEAAAAAAAAAAAAAAAAAAAAAAAAAAAABAA1DtwAAA
AAAAAAAbDEdrCo0vjSNo8svQasXO53zihMS7i9J
JHpRWNzNa2wAAAAXPIseMAANQAAAAAACAAAAAA
AAAAAAAAAAAAAAAAEAAAAAAAAAAAAAAAAAAAAAAAA
AABAAAABAAAAAMADUO3AAAAAAAAAABsMR2sKjS
+NI2jyy9Bqxc7nfOKExLuL0kkelFY3M1rbAAAA
Bc8ix4wAA1AAAAAAAIAAAAAAAAAAAAAAAAAAAAA
AAQAAAAAAAAAAAAAAAAAAAAAAAAAAEADUO3AAAAA
AAAAABsMR2sKjS+NI2jyy9Bqxc7nfOKExLuL0k
kelFY3M1rbAAAABcks5owAA1AAAAAAAIAAAAAAA
AAAAAAAAAAAAAAAQAAAAAAAAAAAAAAAAAAAAAAAA
AMADUA1AAAAAAAAAAADnHeoAXmOm+2G75AWA/6I
iroU2z4t6tAlM0kTab+ZtAwAAABdUYqoAAA1ABA
AAAAAAAAAAAAAAAAAAAAAAAAAAAQAAAAAAAAAAA
AAAAAAAAAAAAAEADUO3AAAAAAAAAAADnHeoAXmOm
+2G75AWA/6IiroU2z4t6tAlM0kTab+ZtAwAAABd
sOi4AAA1ABAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
QAAAAAAAAAAAAAAAAAAAAAAAAAA=
```

Fee Meta XDR

```
AAAAAgAAAAMADUA1AAAAAAAAAABsMR2sKjS+NI2
jyy9Bqxc7nfOKExLuL0kkelFY3M1rbAAAABc8i
yIYAA1AAAAAAAEAAAAAAAAAAAAAAAAAAAAAAAAQA
AAAAAAAAAAAAAAAAAAAAAAAAEADUO3AAAAAAAAAA
ABsMR2sKjS+NI2jyy9Bqxc7nfOKExLuL0kkelF
Y3M1rbAAAABc8ix4wAA1AAAAAAAEAAAAAAAAAAA
AAAAAAAAAAAAQAAAAAAAAAAAAAAAAAAAAA==
```

Submit Transaction

Block # 3

Data
40bbf2d52228f3d2a2701f0121e343f1e626fcfc9a002713
1195c6e287d132d6

Previous Hash
0007b7897e52aac2efb16520693a59fa4a57724613ea68
3fbd9498e36563923

Hash
0000c763244ff29a8b1519e078b5cc1c5742c47c1e2f3b5
dde7dd3e6c54ccb69

Ledger No
869467

Result Meta XDR:

```
AAAAAgAAAAIAAAADAA1EWwAAAAAAAAAAbDEdrCo
0vjSNo8svQasXO53zihMS7i9JJHpRWNzNa2wAAA
AXJLOWSAANQAAAAAACAAAAAAAAAAAAAAAAAAAAA
AEAAAAAAAAAAAAAAAAAAAAAAAAAAABAA1EWwAAAAAA
AAAAbDEdrCo0vjSNo8svQasXO53zihMS7i9JJHp
RWNzNa2wAAAAXJLOWSAANQAAAAAADAAAAAAAAAA
AAAAAAAAAAAAEAAAAAAAAAAAAAAAAAAAAAAAAAABA
AAABAAAAAMADURbAAAAAAAAAABsMR2sKjS+NI2j
yy9Bqxc7nfOKExLuL0kkelFY3M1rbAAAABcks5Z
IAA1AAAAAAAMAAAAAAAAAAAAAAAAAAAAAAQAAAA
AAAAAAAAAAAAAAAAAAAAEADURbAAAAAAAAAABsM
R2sKjS+NI2jyy9Bqxc7nfOKExLuL0kkelFY3M1r
bAAAABcG5jFIAA1AAAAAAAMAAAAAAAAAAAAAAAA
AAAAAAQAAAAAAAAAAAAAAAAAAAAAAAAAMADUO3AA
AAAAAAAADnHeoAXmOm+2G75AWA/6IiroU2z4t6t
AlM0kTab+ZtAwAAABdsOi4AAA1ABAAAAAAAAAAA
AAAAAAAAAAAAAAAAQAAAAAAAAAAAAAAAAAAAAAAA
AAAEADURbAAAAAAAAAADnHeoAXmOm+2G75AWA/6
IiroU2z4t6tAlM0kTab+ZtAwAAABeKB5MAAA1AB
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAQAAAAAAAAAAAA
AAAAAAAAAAAAAAA=
```

Fee Meta XDR

```
AAAAAgAAAAMADUO3AAAAAAAAAABsMR2sKjS+NI2j
yy9Bqxc7nfOKExLuL0kkelFY3M1rbAAAABcks5o
wAA1AAAAAAAIAAAAAAAAAAAAAAAAAAAAAAAAQAAAAA
AAAAAAAAAAAAAAAAAAAEADURbAAAAAAAAAABsMR
2sKjS+NI2jyy9Bqxc7nfOKExLuL0kkelFY3M1rb
AAAABcks5ZIAA1AAAAAAAIAAAAAAAAAAAAAAAAAA
AAAAAQAAAAAAAAAAAAAAAAAA==
```

Submit Transaction

Block # 4

Data
698e99153f23976384f390c8c9bdbbf5466be29e141d455
301c3b5319090f691

Previous Hash
0000c763244ff29a8b1519e078b5cc1c5742c47c1e2f3b5
dde7dd3e6c54ccb69

Hash
0000c763244ff29a8b1519e078b5cc1c5742c47c1e2f3b5
dde7dd3e6c54ccb69

Ledger No
869467

Result Meta XDR:

```
AAAAAgAAAAIAAAADAA1EpQAAAAAAAAAAbDEdrCo
0vjSNo8svQasXO53zihMS7i9JJHpRWNzNa2wAAAA
XBuYtYAANQAAAAAADAAAAAAAAAAAAAAAAAAAAAA
EAAAAAAAAAAAAAAAAAAAAAAAAAAABAA1EpQAAAAAA
AAAbDEdrCo0vjSNo8svQasXO53zihMS7i9JJHpR
WNzNa2wAAAAXBuYtYAANQAAAAAAEAAAAAAAAAAA
AAAAAAAAAAAEAAAAAAAAAAAAAAAAAAAAAAAAAABAA
AABAAAAAMADUS1AAAAAAAAAABsMR2sKjS+NI2jy
y9Bqxc7nfOKExLuL0kkelFY3M1rbAAAABcG5i1g
AA1AAAAAAAQAAAAAAAAAAAAAAAAAAAAAAQAAAAA
AAAAAAAAAAAAAAAAAAAEADUS1AAAAAAAAAABsMR
2sKjS+NI2jyy9Bqxc7nfOKExLuL0kkelFY3M1rb
AAAABbRQURgAA1AAAAAAAQAAAAAAAAAAAAAAAAAAA
AAAAQAAAAAAAAAAAAAAAAAAAAAAAAAMADURbAAAAA
AAAAADnHeoAXmOm+2G75AWA/6IiroU2z4t6tAlM
0kTab+ZtAwAAABeKB5MAAA1ABAAAAAAAAAAAAAAA
AAAAAAAAAAAAAQAAAAAAAAAAAAAAAAAAAAAAAAAA
EADUS1AAAAAAAAAADnHeoAXmOm+2G75AWA/6Iiro
U2z4t6tAlM0kTab+ZtAwAAABe/rHwAAA1ABAAAA
AAAAAAAAAAAAAAAAAAAAAAAAQAAAAAAAAAAAAAAAA
AAAAAAAAA=
```

Fee Meta XDR

```
AAAAAgAAAAMADURbAAAAAAAAAABsMR2sKjS+NI2j
yy9Bqxc7nfOKExLuL0kkelFY3M1rbAAAABcG5jFI
AA1AAAAAAAMAAAAAAAAAAAAAAAAAAAAAAAQAAAAAA
AAAAAAAAAAAAAAAAAAAEADUS1AAAAAAAAAABsMR2s
KjS+NI2jyy9Bqxc7nfOKExLuL0kkelFY3M1rbAAA
ABcG5i1gAA1AAAAAAAMAAAAAAAAAAAAAAAAAAAAA
AAQAAAAAAAAAAAAAAAAAAAA==
```

Submit Transaction

Figure 4.6: Blockchain with Four Blocks

```
{
”_links”: {
”self”: {
”href”: ”https://horizon-testnet.stellar.org/accounts/GCNFF5HNN63VYEHFXTQHQYEM66G3IDQ47KFEVRYQ3X66XLR276M3I7RZ”
},
”transactions”: {
”href”: ”https://horizon-testnet.stellar.org/accounts/GCNFF5HNN63VYEHFXTQHQYEM66G3IDQ47KFEVRYQ3X66XLR276M3I7RZ/transactions{?cursor,limit,order}”,
”templated”: true
},
”operations”: {
”href”: ”https://horizon-testnet.stellar.org/accounts/GCNFF5HNN63VYEHFXTQHQYEM66G3IDQ47KFEVRYQ3X66XLR276M3I7RZ/operations{?cursor,limit,order}”,
”templated”: true
},
”payments”: {
”href”: ”https://horizon-testnet.stellar.org/accounts/GCNFF5HNN63VYEHFXTQHQYEM66G3IDQ47KFEVRYQ3X66XLR276M3I7RZ/payments{?cursor,limit,order}”,
”templated”: true
},
”effects”: {
”href”: ”https://horizon-testnet.stellar.org/accounts/GCNFF5HNN63VYEHFXTQHQYEM66G3IDQ47KFEVRYQ3X66XLR276M3I7RZ/effects{?cursor,limit,order}”,
”templated”: true
},
”offers”: {
”href”: ”https://horizon-testnet.stellar.org/accounts/GCNFF5HNN63VYEHFXTQHQYEM66G3IDQ47KFEVRYQ3X66XLR276M3I7RZ/offers{?cursor,limit,order}”,
”templated”: true
},
”trades”: {
”href”: ”https://horizon-testnet.stellar.org/accounts/GCNFF5HNN63VYEHFXTQHQYEM66G3IDQ47KFEVRYQ3X66XLR276M3I7RZ/trades{?cursor,limit,order}”,
”templated”: true
},
”data”: {
”href”: ”https://horizon-testnet.stellar.org/accounts/GCNFF5HNN63VYEHFXTQHQYEM66G3IDQ47KFEVRYQ3X66XLR276M3I7RZ/data/{key}”,
”templated”: true
}
},
”id”: ”GCNFF5HNN63VYEHFXTQHQYEM66G3IDQ47KFEVRYQ3X66XLR276M3I7RZ”,
”account_id”: ”GCNFF5HNN63VYEHFXTQHQYEM66G3IDQ47KFEVRYQ3X66XLR276M3I7RZ”,
”sequence”: ”3311724727894016”,
”subentry_count”: 0,
”last_modified_ledger”: 771071,
”last_modified_time”: ”2022-05-02T07:29:56Z”,
”thresholds”: {
”low_threshold”: 0,
”med_threshold”: 0,
”high_threshold”: 0
},
”flags”: {
”auth_required”: false,
”auth_revocable”: false,
”auth_immutable”: false,
”auth_clawback_enabled”: false
},
”balances”: [
{
”balance”: ”10000.0000000”,
”buying_liabilities”: ”0.0000000”,
”selling_liabilities”: ”0.0000000”,
”asset_type”: ”native”
}
],
”signers”: [
{
”weight”: 1,
”key”: ”GCNFF5HNN63VYEHFXTQHQYEM66G3IDQ47KFEVRYQ3X66XLR276M3I7RZ”,
”type”: ”ed25519_public_key”
}
],
”data”: {},
”num_sponsoring”: 0,
”num_sponsored”: 0,
”paging_token”: ”GCNFF5HNN63VYEHFXTQHQYEM66G3IDQ47KFEVRYQ3X66XLR276M3I7RZ”
}
```

Table 4.4: Code of Account Details JSON

Chapter 5

Result Analysis

Security is the greatest difficulty in today's Internet, despite breakthroughs in encryption, secure protocols, and system-level protections. NDN approaches this problem in a very different way than current procedures. NDN allows one to directly safeguard data by having data producers cryptographically sign each data packet to connect the data name, producer, and content. When data secrecy is required, they can also be encrypted [7]. Authenticity and confidentiality are security features of data that are independent of data containers and communication routes. Despite the fact that NDN is considered more secure than contemporary TCP/IP, it has several drawbacks, such as the inability to get original cNACKs and fNACKs. Our suggested blockchain based false NACK prevention methodology is more secure, dynamic, and practical than existing alternatives. Also it works for both static and dynamic contents.

5.1 Cost Analysis

Permissioned blockchain is used in our system. Permissioned blockchains offer a variety of economic benefits, the majority of which stem from the technical advantages they provide. They are often smaller than public blockchains. As a result, they react faster. The blockchain data is smaller and there are less transactions than on a public blockchain. They can confidently identify their consumers. There are virtually few chances of identity theft because each consumer is given both public and private keys. Hard forks in permissioned blockchains can happen extremely quickly. Hard forks, unlike public blockchains, do not require community consent. As a result, administrators having authority over the private network can carry them out.They do not waste energy in wasteful ways. They don't employ a proof-of-work system like mining. They normally use a multi-party voting system instead. The cost of network maintenance is also reduced since the blockchain community can 'self-maintain' in many ways. Permissioned blockchain networks are expected to save financial institutions between 15 billion and 20 billion by 2022, according to estimates. The price of cryptocurrency is skyrocketing, as is its utility. We utilized Stellar and Lumen as test coins in our implementation.Purchasing cryptocurrencies may appear to be too expensive in the current market, but in our proposed model, we used permissioned blockchain, which allows the creation of a new coin for its use. As a result, network managers may create their own money for this permissioned blockchain rather than purchasing current cryptocurrencies, lowering the cost significantly. Moreover, the digital coin and blockchain may be created in whatever way the developers choose to reduce transaction costs and times.

5.2 Latency Analysis

The time delay between an input and the received output is referred to as latency in computing. It's present at every level of computing, from user-to-computer IO delay to network latency when data and information flow from a computer to servers all over the world. Latency can refer to two separate temporal delays in a blockchain network. The first is the latency in a blockchain network, while the second is the delay in a data or packet exchange.

Network latency is the time between submitting a transaction to a blockchain network and the network's initial confirmation of acceptance. The transaction gets more definitive after the first confirmation as more blocks are added beyond the original confirmation. Low network latency is critical for a payments system that expects to garner widespread acceptance. If the delay between paying at the cash register and receiving payment confirmation becomes too long, it becomes a source of user friction.

The capacity to process and executes huge quantities of transactions in their order books is measured by the latency of a data or packet exchange. Day traders frequently use bots to automate the majority of their trading activity, which means the

bots are placing and canceling a massive number of orders per second. An exchange with low latency and high throughput can execute these orders quickly, allowing the day trader (through the bot) to take advantage of price movements more effectively. In contrast, a high-latency exchange will execute orders with a time lag behind the changing asset price, resulting in incorrectly priced orders and missed chances for the day trader.
Every 10 minutes or so, miners add new blocks to public blockchains. Because that procedure cannot be sped up due to the underlying technology, and because each block can only hold a finite amount of transaction records, there is a backlog of transactions that must be joined to blocks and verified when there is a lot of activity.Only the producers participate in the consensus mechanism in our suggested approach. Producer nodes also serve as both an NDN and a blockchain node in our model. Each Proof-of-Reliability (PoR) is only delivered to the trustworthy blockchain network once, and the NACKs are sent to the routers once the packet is verified and added to the ledger. We don't have any delay in our suggested architecture because we're employing Stellar Consensus Protocol (SCP) and the ledger in Stellar closes in 3–5 seconds.

Chapter 6

Conclusion and Future Work

We started our research by looking for and securing fake negative acknowledgement packets (NACK) of Named Data Network. Then, using blockchain technology, we started looking for and securing those packets. We proposed a workflow for consumers and producers, as well as a mechanism for achieving safe data transfer over the NDN network, in our research. We believe it will be the most secure solution for combating fraudulent NACK creation, rather than the previously stated technique. Steller and the Steller Consensus Protocol are the foundations of our suggested paradigm and implementation (SCP). However, other cryptocurrencies and blockchain systems can readily be used to achieve our suggested approach. Better consensus procedures can be used, such as proof of stake (PoS), proof capacity (PoC), or Ripple. These consensus processes are currently being researched and refined, hence they are relatively new to the business. Participants in Ripple are either the network's server or its clients. Clients transmit transactions to the server, which calculates the percentage of agreement using a unique node list, which is subsequently added to a ledger if it exceeds 80 percent. Using newer consensus mechanisms, our approach can be implemented considerably more effectively in the near future.

The architecture we proposed is based on a consensus method that allows producers to prove their reliability to other producers. It is determined by the length of time a server has been in the network and the number of contents it has published. If a new producer wants to join the network, proving their reliability will be challenging, even if they are original. Furthermore, Stellar's ledger closes in every 3–5 seconds, which is relatively slow in terms of data transmission. If we could reduce the closing time, communication would be faster. More research is needed to discover a solution to these problems.

Instead of waiting for issued interests to expire, customers may rely on Negative Acknowledgements (NACKs) to quickly discover non-existent material. It also enables customers to distinguish between non-existent content and packet (interest) loss. It must be carefully secured because it plays such an important role in NDN. We can do this using blockchain since once they are released to the main blockchain, they cannot be changed and consumers may trust them based on their reliability score. Through our demo model, we attempted to illustrate the bulk of the model's capabilities. It is highly recommended that additional time and effort be put into establishing a reliable model. There are already NDN and blockchain specialists working on this, and they must be contacted before the model is deployed in production.

Bibliography

[1] B. H. Bloom, “Space/time trade-offs in hash coding with allowable errors,” *Communications of the ACM*, vol. 13, no. 7, pp. 422–426, 1970.

[2] M. J. Fischer, N. A. Lynch, and M. S. Paterson, “Impossibility of distributed consensus with one faulty process,” *Journal of the ACM (JACM)*, vol. 32, no. 2, pp. 374–382, 1985.

[3] D. Mills, J. Martin, J. Burbank, and W. Kasch, “Network time protocol version 4: Protocol and algorithms specification,” 2010.

[4] C. Ghali, G. Tsudik, and E. Uzun, “Network-layer trust in named-data networking,” *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 5, pp. 12–19, 2014.

[5] A. Compagno, M. Conti, C. Ghali, and G. Tsudik, “To nack or not to nack? negative acknowledgments in information-centric networking,” in *2015 24th International Conference on Computer Communication and Networks (ICCCN)*, IEEE, 2015, pp. 1–10.

[6] D. Kim, S. Nam, J. Bi, and I. Yeom, “Efficient content verification in named data networking,” in *Proceedings of the 2nd ACM Conference on Information-Centric Networking*, 2015, pp. 109–116.

[7] Y. Yu, A. Afanasyev, and L. Zhang, “Name-based access control,” *Named Data Networking Project, Technical Report NDN-0034*, 2015.

[8] V. Cisco, “Cisco visual networking index: Forecast and methodology, 2016–2021,” *CISCO White paper*, 2017.

[9] R. KUMARI and M. CATHERINE, “Anomaly detection in blockchain using clustering protocol,” *International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)*, vol. 4, no. 12, 2017.

[10] A. Afanasyev, J. Burke, T. Refaei, L. Wang, B. Zhang, and L. Zhang, “A brief introduction to named data networking,” in *MILCOM 2018-2018 IEEE Military Communications Conference (MILCOM)*, IEEE, 2018, pp. 1–6.

[11] S. Dey, “Securing majority-attack in blockchain using machine learning and algorithmic game theory: A proof of work,” in *2018 10th computer science and electronic engineering (CEEC)*, IEEE, 2018, pp. 7–10.

[12] J. Lou, Q. Zhang, Z. Qi, and K. Lei, “A blockchain-based key management scheme for named data networking,” in *2018 1st IEEE international conference on hot information-centric networking (HotICN)*, IEEE, 2018, pp. 141–146.

[13] M. Salimitari and M. Chatterjee, “A survey on consensus protocols in blockchain for iot networks,” *arXiv preprint arXiv:1809.05613*, 2018.

[14] M. Signorini, M. Pontecorvi, W. Kanoun, and R. Di Pietro, “Advise: Anomaly detection tool for blockchain systems,” in *2018 IEEE World Congress on Services (SERVICES)*, IEEE, 2018, pp. 65–66.

[15] Z. Lei, C. Feng, Y. Liu, D. S. Lee, T. Tsang, J. Liang, Z. Xiong, Y. Liu, and G. Chen, “Next generation blockchain network (ngbn),” in *2019 20th IEEE International Conference on Mobile Data Management (MDM)*, IEEE, 2019, pp. 452–456.

[16] H. Li, K. Wang, T. Miyazaki, C. Xu, S. Guo, and Y. Sun, “Trust-enhanced content delivery in blockchain-based information-centric networking,” *Ieee Network*, vol. 33, no. 5, pp. 183–189, 2019.

[17] N. S. Rowan and N. Usher, “The flare consensus protocol: Fair fast federated byzantine agreement consensus,” Tech. rep, Tech. Rep., 2019.

[18] S. Sayeed and H. Marco-Gisbert, “Assessing blockchain consensus and security mechanisms against the 51% attack,” *Applied Sciences*, vol. 9, no. 9, p. 1788, 2019.

[19] M. Buragohain and S. Nandi, “Demystifying security on ndn: A survey of existing attacks and open research challenges,” in *The Essence of Network Security: An End-to-End Panorama*, Springer, 2021, pp. 241–261.