CHAPTER 1

Introduction

1.1 Background of Software-Defined Networking (SDN)

Traditional networking has been the backbone of communication systems for decades, relying on fixed hardware-based configurations to manage the flow of data between devices [59]. In these networks, control, and data planes are tightly coupled, meaning that decisions about how data moves through the network and the actual forwarding of that data occur within the same devices, such as routers and switches. Because control is distributed across many devices, traditional networks lack a centralized view of the entire system. This decentralized nature complicates tasks such as traffic engineering, load balancing, and security policy enforcement [36]. Traditional networks rely on complex, vendor-specific configurations that are often static and hard to change. The configuration and management of these networks typically become complex, and difficult to adapt to dynamic changes in traffic patterns or network demands. Thus, making changes to network policies or configurations typically requires manual intervention on each device, leading to a time-consuming and error-prone process. Further, as the number of devices and the amount of traffic on the network grow, traditional networks struggle to scale efficiently.

Recognizing these limitations, researchers at Stanford University began exploring a new approach to networking in 2011, leading to the development of Software-Defined Networking (SDN) [88]. It is a new paradigm of networking that redefines how networks are designed, managed, and operated [15]. The introduction of SDN changes the traditional approach by centralizing the control plane in a logically centralized controller separated from the data plane. The initial implementation of SDN was part of a project aimed at creating a more flexible and manageable Campus Area Network at Stanford University. The goal was to enable

network administrators to manage traffic flows dynamically using a centralized software-based controller, which could program the behavior of network devices via the OpenFlow protocol [61]. Therefore, the controller is responsible for making decisions about how packets should be handled, while the data plane is tasked with forwarding packets based on those decisions. This groundbreaking approach allowed for much greater control and visibility over the network, drastically reducing the complexity of managing network traffic and enabling real-time response to network conditions. Later, SDN has evolved beyond academic research and has been widely adopted in various fields due to its ability to simplify network management, enhance scalability, and optimize performance. Some key applications of SDN include Data Center Networks (DCN) [31, 18, 41], Software-Defined Wide Area Networks (SD-WAN) [110, 30], Internet-of-Things (IoT) [4, 14], Healthcare [98, 65] and Multipath Routing etc. SDN is extensively used in data centers, where it plays a crucial role in optimizing resource utilization, reducing latency, and managing complex network topologies. It helps data centers efficiently handle large volumes of traffic, automate network provisioning, and implement rapid scaling to meet dynamic workload demands. SD-WAN allows enterprises to manage wide-area networks more efficiently and securely. SD-WAN uses SDN principles to route traffic over multiple types of connections (such as MPLS, broadband, or LTE) based on real-time conditions thus reducing operational costs. Further, SDN facilitates multipath routing [75, 12, 19, 7], which allows data to be transmitted over multiple paths simultaneously, increasing redundancy, and bandwidth utilization. This approach not only improves network performance but also enhances resilience by providing alternative paths in case of link failures. Therefore, this architectural shift has paved the way for more agile and responsive network management, enabling rapid innovation and simplified network operations.

1.2 SDN Architecture

The SDN architecture consists of three main layers: the Application Layer, the Control Layer, and the Forwarding (Infrastructure) Layer (refer Figure 1.1). Additionally, SDN incorporates various interfaces that facilitate communication between these layers, including Northbound, Southbound, and East/West interfaces [60]. Below is a detailed explanation of each component:

1.2.1 Application Layer

The Application Layer is the topmost layer of the SDN architecture, consisting of network applications and services that define the behavior and functionality of the network. These applications can range from traffic engineering and load balancing to security monitoring and policy management. They communicate with the SDN controller through the Northbound API, requesting specific network behaviors or querying network states. The Application Layer leverages the programmability of SDN to dynamically adjust network behavior based on the application's needs, making the network more adaptable and responsive to changing requirements.

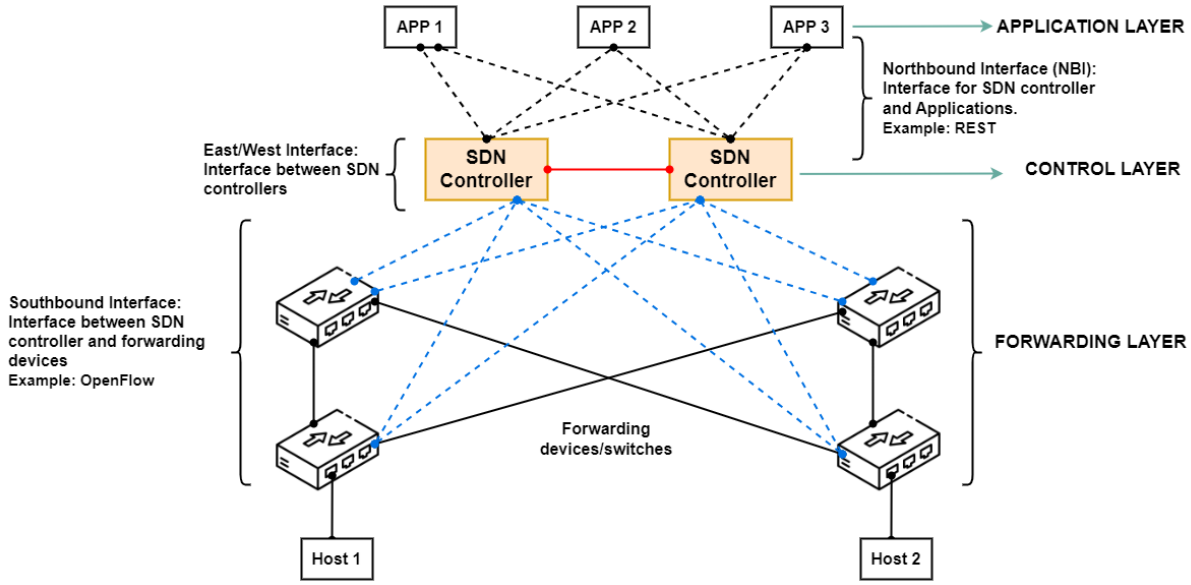

Figure 1.1: SDN Architecture

1.2.2 Control Layer

The Control Layer serves as the brain of the SDN architecture [26]. It comprises the SDN controller, which centralizes the decision-making processes and manages the overall network behavior. The controller has a global view of the network and is responsible for making decisions on how traffic flows should be handled across the network infrastructure. The controller communicates with the Data Layer using the Southbound API (e.g., OpenFlow) and interacts with network applications through the Northbound API. The Control Layer ensures that the network operates according to the policies and instructions set by the applications, adjusting flow rules dynamically based on current network conditions and demands [35].

1.2.3 Forwarding (Infrastructure) Layer

The Forwarding Layer, also known as the Infrastructure Layer, consists of the physical and virtual network devices such as switches, routers, and other forwarding elements [1]. These devices handle the actual data forwarding tasks, directing traffic based on the flow rules received from the controller via the Southbound interface. Unlike traditional network devices, SDN forwarding elements are relatively simple and do not make independent routing decisions; instead, they rely entirely on the instructions from the SDN controller [86].

1.3 OpenFlow Protocol

The OpenFlow protocol is a pivotal component of the SDN architecture, facilitating communication between the SDN controller and the forwarding devices within the network [61]. It is the most widely used Southbound protocol in SDN environments and is fundamental in implementing the centralized control paradigm. Communication between the SDN controller and the forwarding devices occurs over a secure channel, ensuring that control messages and instructions are securely transmitted. Each forwarding device maintains one or more flow tables, which contain the rules that dictate how incoming packets should be handled.

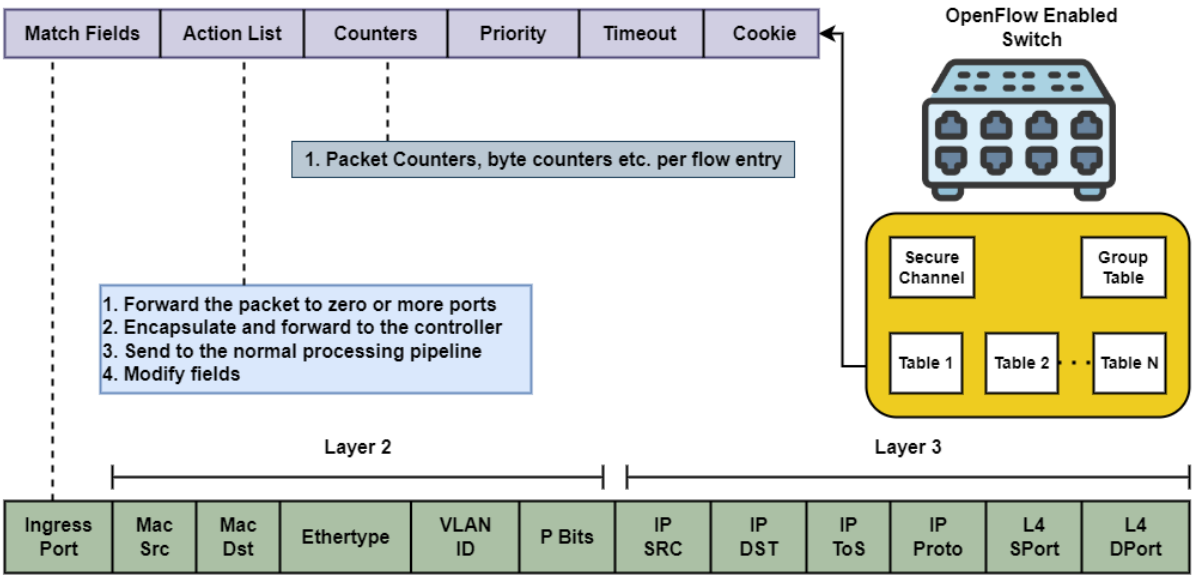

Figure 1.2: OpenFlow Structure

When a packet arrives at an OpenFlow-enabled switch, the switch matches the packet's

header fields against its flow table entries. If a match is found, the specified action (e.g., forward to a port, modify the packet, drop the packet) is executed. If no match is found, the packet can be sent to the controller for further processing. The OpenFlow message structure is shown in Figure 1.2. OpenFlow protocol messages are used to manage and monitor network devices. These messages include:

- **Flow Modifications:** Instructions to add, modify, or delete flow entries in the switch's flow table.
- **Packet-In:** A message sent from a switch to the controller when a packet does not match any existing flow entries.
- **Packet-Out:** A command from the controller to the switch to handle specific packets.
- **Stats Request/Reply:** Used to gather statistics about the flows, ports, and other elements of the network devices.

1.4 Multi-Controller Architecture

To overcome the limitations of having a single point of control, SDN often employs a multi-controller architecture [112, 39, 69]. This setup distributes the control tasks among several controllers, enhancing scalability, fault tolerance, and network performance. In multi-controller SDN environments, different roles are assigned to controllers to manage the interactions with forwarding devices and to ensure redundancy and failover capabilities. These roles are master, slave, and equal [111]. We briefly discuss these controller roles in the following:

a) **Master Controller:** The master controller is the primary controller with full control and authority over a set of network devices. The master controller installs flow rules, collects device statistics, handles packet-in messages, and makes real-time decisions that affect the network's operation. If the master controller fails or becomes unreachable, the system may promote a slave or equal controller to assume the master role, ensuring the continuity of network operations.

b) **Slave Controller:** A slave controller acts as a backup to the master and does not have direct control over the forwarding devices. It operates in a passive role, receiving updates

and maintaining synchronization with the network state but not actively managing the devices. While in slave mode, the controller can query the network for information but cannot issue control commands to modify the network's behavior.

c) **Equal Controller:** An equal controller shares authority with other controllers over a specific set of network devices. Unlike the strict master-slave relationship, equal controllers collaboratively manage devices, allowing them to concurrently read and write flow rules. This setup can improve load balancing and fault tolerance. Since multiple controllers can issue commands simultaneously, there must be mechanisms in place to resolve conflicts and ensure consistency. Controllers often rely on synchronization protocols or predefined rules to prevent conflicting flow modifications.

1.5 Security Challenges

The separation of control plane from the forwarding devices introduce significant flexibility and programmability, but it also brings a set of challenges that need to be addressed to ensure network security. Key challenges in SDN networks include maintaining flow rule integrity, mitigating the risk of a single point of failure, ensuring cross-domain flow integrity, and securely authenticating users within the network. We briefly discuss each of these challenges in the below subsection.

a) **Forwarding rule integrity:** One of the central components of SDN is the flow table, which resides in the data plane switches and dictates how packets should be handled. Flow rule integrity can be compromised in several ways, such as through malicious attacks where an adversary injects unauthorized flow rules, or through accidental misconfigurations that lead to security breaches or network outages [106, 80, 102, 103, 92, 56]. Traditional methods, such as static analysis and formal verification, can be used to check flow rules, but these approaches often struggle with scalability, particularly in large and dynamic networks where flow rules are frequently updated.

b) **Single point of failure:** The centralized nature of the SDN control plane, where a single controller or a small set of controllers manage the entire network, introduces the risk of a single point of failure. If a controller fails due to a malicious attack, the entire network

could become unmanageable, leading to service disruptions or complete network outages. To address this issue, researchers have proposed distributed controller architectures, where multiple controllers are deployed across the network [95, 2, 47, 107]. However, this introduces new challenges, such as ensuring consistent state synchronization among controllers and minimizing the latency introduced by distributed control.

c) **Cross-domain flow integrity:** In large-scale networks, particularly those spanning multiple administrative domains, maintaining flow integrity across domain boundaries is a significant challenge. Each domain may have its policies, protocols, and security requirements, leading to potential conflicts or vulnerabilities when flows traverse multiple domains [72, 109, 105]. Ensuring that flow rules are consistently enforced across these boundaries is crucial to maintaining overall network security and performance.

d) **Authentication of Users:** In SDN networks, where the control plane is centralized, authenticating users becomes even more critical, as unauthorized access to the control plane could allow an attacker to manipulate flow rules and compromise the entire network [20, 27, 43, 48].

Therefore, the attacker can exploit vulnerabilities in the control plane to modify or redirect network traffic by changing the output port, leading to security breaches and data leakage. This creates a potential vulnerability where malicious actors can deceive security applications, leading to the installation of malicious flow rules on OpenFlow switches. Therefore, the integrity of these flow rules is essential to maintaining the intended operation of the network and must be protected from unauthorized modifications.

1.6 Blockchain Technology

Blockchain is a decentralized and distributed ledger technology that records transactions across multiple nodes in a secure, transparent, and tamper-resistant manner [114]. It operates on a peer-to-peer network where each node maintains a copy of the ledger, ensuring data consistency and integrity without the need for a central authority. Blockchain's architecture makes it highly suitable for applications requiring transparency, traceability, and security, such as financial transactions [46, 96], supply chain management [25, 83], Healthcare [37, 97, 94], Industry

[87, 11, 45], and digital identity verification [82, 24, 33, 113]. At its core, blockchain technology relies on cryptographic techniques to secure data. The concept behind blockchain is to record the transaction history in a chain of blocks across its peer-to-peer network. Blocks are connected in a chain, with each block containing a hash value that corresponds to the previous block. Therefore, mutating one block will make the entire chain invalid. Each node in the network takes a copy of the blockchain and performs a validation of transactions. Therefore, a consensus is made among the nodes before creating a new block, which makes it difficult for the eavesdropper to tamper with any block (once a block has been tampered with, it is visible to all nodes involved in the consensus). The consensus algorithms tell how the nodes agree to add the block to the chain. After the nodes have verified the transaction, it is approved, and the distributed ledger is updated. The updated ledger is then broadcasted to all nodes on the blockchain, ensuring that every node has an accurate and current copy of the data (see Figure 1.3). This immutability, combined with the decentralized nature of blockchain, provides a robust platform for trustless interactions among parties.

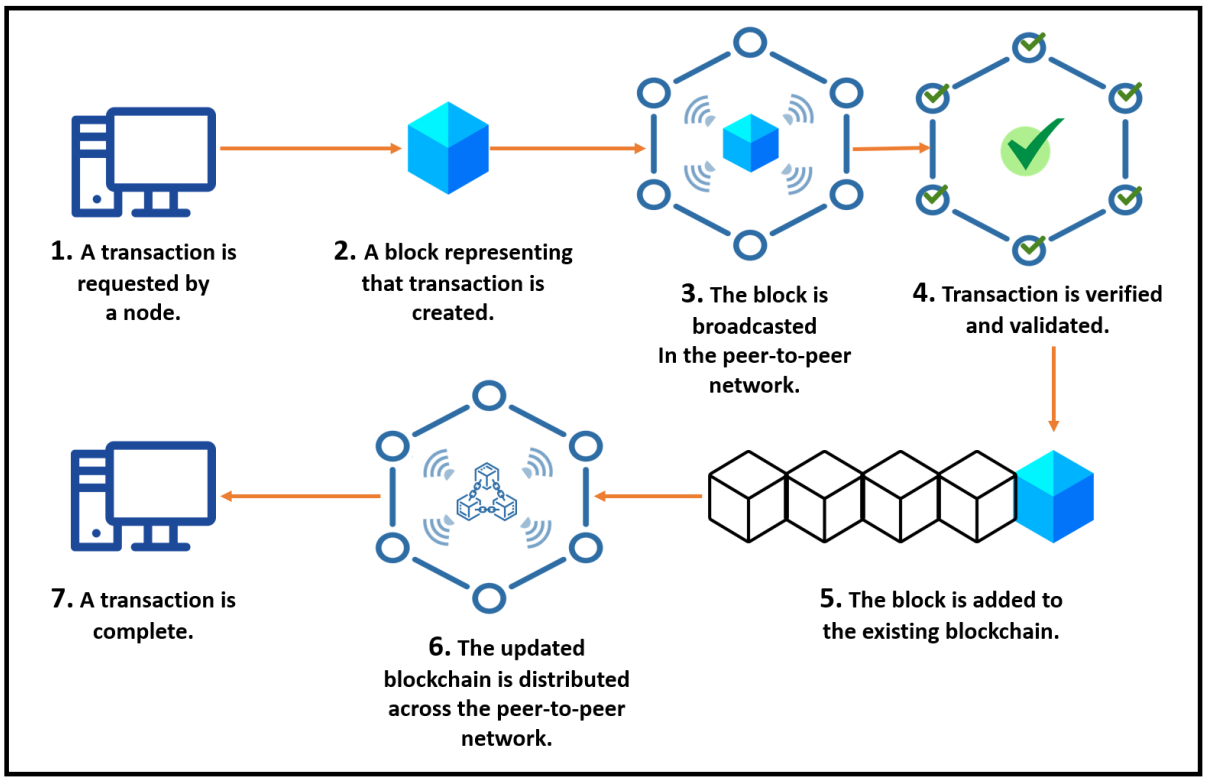

Figure 1.3: Transaction Processing

1.6.1 Blockchain Consensus

Consensus algorithms are protocols that allow nodes in a blockchain network to agree on the state of the ledger, ensuring that all participants have the same version of the truth [104]. Different consensus algorithms have been developed to meet various needs, balancing security, speed, and energy efficiency. We briefly discuss some of these protocols below.

a) **Proof of Work (PoW):** PoW is one of the earliest consensus algorithms used in blockchain, notably in Bitcoin. It requires participants (miners) to solve complex cryptographic puzzles to validate transactions and add new blocks to the chain. It provides high security and resistance to attacks. However, it consumes high energy and is slower in transaction processing.

b) **Proof of Stake (PoS):** PoS replaces the energy-intensive mining process with a system where validators are chosen based on the number of tokens they hold and are willing to stake as collateral. Validators create new blocks and receive rewards based on their stake. It consumes lower energy consumption compared to PoW but poses potential centralization risks if a few entities hold large stakes.

c) **Practical Byzantine Fault Tolerance (PBFT):** This protocol is designed to tolerate Byzantine faults and is often used in permissioned blockchains. It operates through a series of voting rounds where nodes agree on the order of transactions.

1.6.2 Ethereum Smart Contracts

Similar to real-world contracts, a Smart Contract (SC) is a digital contract in the form of a computer program that runs on the Blockchain network. SC is a self-executed program and is triggered when certain conditions are satisfied without consulting any central authority. It is mainly used to automate the operations in the network [73]. Ethereum provides an implementation of SC to automate the network characteristics. Since SC is stored on the blockchain, they are also immutable and accessible publicly. This makes the SC tamper-proof. Solidity and Vyper are the two programming languages used to write SC on the Ethereum platform. The Ethereum SC contains few functions and machine states. The functions are used to change the machine state in the blockchain. Therefore, it is useful for automating the verification of flow

rules on the blockchain without the need for intervention. Furthermore, the decentralized architecture of blockchain technology makes it suitable for the integrity of SDN flow tables. The immutable nature of blockchain prevents the malicious manipulation of flows by the attacker and makes it potential for SDN integrity. So, it can be used to strengthen the integrity of SDN and make it more robust.

1.7 Motivation

This research is motivated by the need to develop a comprehensive security framework that not only addresses the existing vulnerabilities in SDN but also leverages the strengths of blockchain technology to build a more resilient and trustworthy network environment. Our work seeks to bridge the gap between SDN's potential and its security challenges, contributing novel solutions that enhance flow integrity, controller resilience, cross-domain collaboration, and secure authentication in SDN-based IoT ecosystems.

1.8 Problem Statement

The architectural shift in the SDN network has introduced critical security challenges that threaten the integrity, reliability, and scalability of SDN deployments. One of the primary issues is the integrity of flow rules within SDN-enabled networks, as unauthorized modifications to these rules can lead to severe security breaches. Additionally, the centralized nature of the SDN controller creates a single point of failure, making the network vulnerable to targeted attacks and operational failures. As SDN scales into multi-domain environments, ensuring cross-domain flow integrity becomes increasingly complex, requiring robust coordination and verification mechanisms. Moreover, the integration of SDN with IoT introduces additional security concerns, particularly the authentication of users and devices, which is essential to prevent unauthorized access and ensure reliable communication. Existing solutions lack the comprehensive security framework needed to address these multi-faceted challenges effectively, particularly in dynamic and large-scale SDN environments. Therefore, there is a pressing need for innovative approaches that combine SDN with blockchain technology to enhance flow rule integrity, mitigate single points of failure, secure cross-domain interactions, and authenticate devices and users in a trustworthy manner.

1.9 Research Objectives

The existing literature techniques often struggle to maintain flow integrity, leaving networks vulnerable to attacks that can manipulate or disrupt data flows. Similarly, the reliance on centralized controllers can create single points of failure, making networks susceptible to outages or attacks. Managing flow integrity across multiple domains adds another layer of complexity, as does ensuring robust authentication of users and devices. Therefore, this thesis primarily focuses on enhancing the security and resilience of SDN-enabled networks.

This thesis aims to address the identified challenges through the following four objectives:

a) **To develop a methodology to maintain flow integrity in the OpenFlow-based network:** In an SDN network, OpenFlow rules dictate how traffic should navigate the network, making the flow table a prime target for security threats like DoS attacks. Therefore, in this thesis, we propose a technique to ensure the integrity of OpenFlow rules in SDN networks through the blockchain smart contract.

b) **To develop a Blockchain-enabled SDN Multi-Controller architecture that provides high availability of network services:** The reliance on a single controller in network architecture presents a significant vulnerability, as the failure of this central controller can lead to a complete network outage. Therefore, we employ the multi-controller architecture of SDN to make the network more agile and scalable.

c) **To develop a solution for ensuring flow integrity across multiple domains:** Deploying SDN across multiple domains presents distinct security challenges, particularly regarding the integrity and security of flow modification requests. Furthermore, various malicious activities, including rogue controller attacks, replay attacks, and Distributed Denial of Service (DDoS) attacks, create vulnerabilities that can compromise flow integrity. In the literature, similar flow rules are often installed on each domain, and rogue flow modifications are checked. However, in practice, each domain maintains separate flow rules. Therefore, our work emphasizes real practical scenarios for preserving the integrity of flow in a multi-domain network.

d) **To develop a blockchain token-based authentication of users in SDN networks that ensures the authenticity of users accessing the network:** Authentication plays a crucial

role in securing IoT networks by ensuring that only authorized users and systems can access and control the devices connected to the network. Furthermore, the centralized architecture, in which all the data and resources are controlled by a single entity, can be a drawback in the modern world because it can lead to a lack of security, scalability, and decentralization.

1.10 Methodologies/Approaches

The research work presented in this thesis is aimed at enhancing the security and resilience of Software Defined Networking (SDN) through the integration of blockchain smart contracts. The thesis is divided into four primary contributions. First, the preservation of flow table integrity in a single SDN controller architecture through the smart contract. Second, the development of a multi-controller architecture to prevent network collapse due to controller failure. Third, the development of multi-stage flow verification of OpenFlow rules in a multi-domain SDN network that uses the digital signature and consensus among the controllers for the acceptance of flow modification proposal. Finally, the development of token-based authentication of users to access the network devices in an SDN-enabled network. The methodologies used in this research include system design, blockchain integration, implementation of smart contracts, and experimental validation. Further, we adopt a hybrid approach combining theoretical modeling with experimental validation.

1.10.1 Blockchain Integration

A private Ethereum blockchain is integrated into the SDN environment to serve as a decentralized ledger for storing and verifying flow rules. Smart contracts are developed using Solidity to automate the process of recording flow rules on the blockchain. The smart contracts are also tested using the Remix IDE independently. We integrated the Hyperledger Fabric in the third contribution to utilize the advantage of modular architecture.

1.10.2 Experimental Validation

The experiments are conducted using Mininet, a network emulator that provides a realistic environment for testing SDN architectures. The Ethereum blockchain is deployed on a private network, and the smart contracts are deployed on the blockchain to perform the flow verification task. We utilize the Web3.py library to communicate with the blockchain network. The proposed methodologies are validated through a series of controlled experiments in a simulated SDN environment. We tested the proposed technique in both normal and various attack scenarios. We compared the results obtained in our method with the latest existing methods including FRChain [101], BlockFlow [53], BlockSDSec [16], BCS [8], and BMC-SDN [22]. Further, we also conducted experiments on SDN hardware switches (Allied Telesis) using Ryu controller to demonstrate the effectiveness of our approach.

1.10.3 Performance Measure

We measure the performance of the proposed method using various parameters including latency for flow verification, verification success rate, consistency of flow rules, transaction cost for execution of smart contract functions, controller overhead, and complexity of the proposed techniques. Further, we tested the performance under various conditions to assess the effectiveness of the proposed solutions in maintaining flow table integrity and ensuring network resilience.

1.11 Thesis Contributions

As part of our research, four different approaches have been developed to secure the OpenFlow rules in an SDN-enabled network using blockchain technology. Each of these methods is briefly explained with results in the following subsections.

1.11.1 Development of a Blockchain-enabled Flow Integrity in the OpenFlow-based network

The separation of the control plane from the forwarding devices in SDN enhances the flexibility. However, it also brings the vulnerability that can be exploited by adversaries to manipulate flow rules on SDN switches. To address this challenge, we developed FTISCON, a method that preserves the integrity of flow rules by storing a copy of the flow rule on the blockchain. Our threat model addresses three main risks: flow modification attacks, false flow insertion attacks, and flow deletion attacks. Using blockchain smart contracts, we streamline the flow verification process to protect against malicious alterations. In our approach, a Blockchain Agent continuously verifies flow rules through the decentralized blockchain network. This module operates independently from the SDN controller, reducing its processing load and accelerating the flow rule verification process. Experimental results demonstrate that our method effectively identifies all three attack types and significantly reduces execution time compared to existing methods.

1.11.2 Development of a Blockchain-enabled Multi-Controller architecture in SDN network.

To ensure resiliency in the event of a master controller failure, we have developed a Blockchain-enabled SDN Multicontroller architecture (SDBlock-IoT) designed to eliminate the single point of failure. In this system, multiple equal controllers are employed, any of which can take over network control if needed. The process begins with the detection of a master controller failure, where equal controllers continuously monitor its status by sending PING requests. If the master controller is active, it responds with an ICMP message. If no response is received, the equal controllers then vote via a smart contract to confirm the failure of the master controller. Next, the equal controllers perform a consensus process considering the response time and resource utilization of all equal controllers to appoint a new master controller. This architecture ensures increased resilience by incorporating a mechanism that dynamically selects the equal controller with optimal performance as a master controller in the event of failure, thus minimizing disruptions and optimizing network performance. The experimental result shows that the proposed model significantly enhances the responsiveness of network service availability by providing a minimum update time compared to existing methods.

1.11.3 Development of a blockchain-enabled multi-stage proposal verification in multi-domain SDN network

Without robust security mechanisms, multi-domain SDN networks are susceptible to a wide range of attacks that can have severe consequences, including network downtime, data loss, and compromised user privacy. Further, various malicious activities, such as rogue controller attacks, replay attacks, and Distributed Denial of Service (DDoS) attacks introduce vulnerabilities that can compromise flow integrity, leading to unauthorized flow manipulation and service disruptions. To address these challenges, we propose Cross-DistBlock, a multi-stage flow verification in multi-domain SDN networks, implemented through smart contracts. This method ensures that flow modification proposals undergo three security checks before a final decision is made (digital signature, selecting the non-rogue controllers, and voting consensus). All proposals are digitally signed using the private key of the proposing domain. SDN controllers then independently verify the digital signatures of these proposals, and a proposal is accepted only if the majority of controllers within the domain verify it successfully. If a controller casts an invalid vote, its trust score is reduced, and it is added to the rogue list if the score drops below a certain threshold. Additionally, we introduced a novel framework for sharing security events across SDN domains through smart contracts. An adaptive policy enforcement mechanism is also designed to proactively address potential security threats by isolating infected network components and inserting proactive flow rules to mitigate threats before they escalate. A thorough security analysis confirms that the integrated use of blockchain and adaptive policy enforcement significantly strengthens the overall security posture of multi-domain SDN networks.

1.11.4 Development of a blockchain token-based authentication of users in SDN network

In this work, we developed a distributed Token-based user authentication through Blockchain to provide secure access control of network services in an SDN-IoT network. By verifying both the device and user identities, the authentication process helps prevent fraudulent activities and ensures data integrity. The process is automated through a smart contract that issues a token for each user request. This authentication process is divided into four phases: registration,

authentication, token distribution, and validation. When a user requests authentication, the smart contract checks whether the IoT device is registered on the blockchain. Upon successful authentication, a digital token is generated for each device access request via the smart contract. These tokens, created by network peers, serve as identification for users or devices in subsequent system interactions. The smart contract verifies the token's signature by decoding it into a specific format to ensure it has not been tampered with. Additionally, the smart contract can revoke tokens upon unsuccessful authentication attempts to prevent attacks on the system. The performance of the authentication process is evaluated based on success rate, latency, security, and the gas cost associated with authentication. Experimental results indicate that the proposed system effectively resists tested attack scenarios, demonstrating a high level of security.

1.12 Thesis Organization

Figure 1.4 presents the flow of chapters, highlighting key components and their interrelationships. The rest of the dissertation is organized as follows:

a) **Chapter 1** provides an overview of traditional networking and introduces the concepts of Software-Defined Networking (SDN) and blockchain technology. It defines the research problem, outlines the motivation behind this study, states the objectives, and highlights the key contributions.

b) **Chapter 2** presents a comprehensive review of the literature related to Software-Defined Networking (SDN), blockchain integration, existing security solutions and finding research gap.

c) **Chapter 3** describes the proposed smart contract–based solution for preserving flow table integrity in OpenFlow networks, including its design, implementation, and performance evaluation.

d) **Chapter 4** presents a distributed SDN controller architecture to mitigate the risk of single points of failure and improves fault tolerance, ensuring robust operation even in the face of controller failures or attacks.

e) **Chapter 5** presents a framework for maintaining flow integrity across multi administrative domain in SDN environments including its components, workflow, and security

mechanisms.

f) **Chapter 6** presents an authentication method using blockchain tokens to secure user and device authentication in SDN-IoT networks, detailing its design, implementation, and security evaluation.

g) Finally, **chapter 7** concludes the thesis with a summary of key findings, a discussion of their implications, and potential directions for future research.

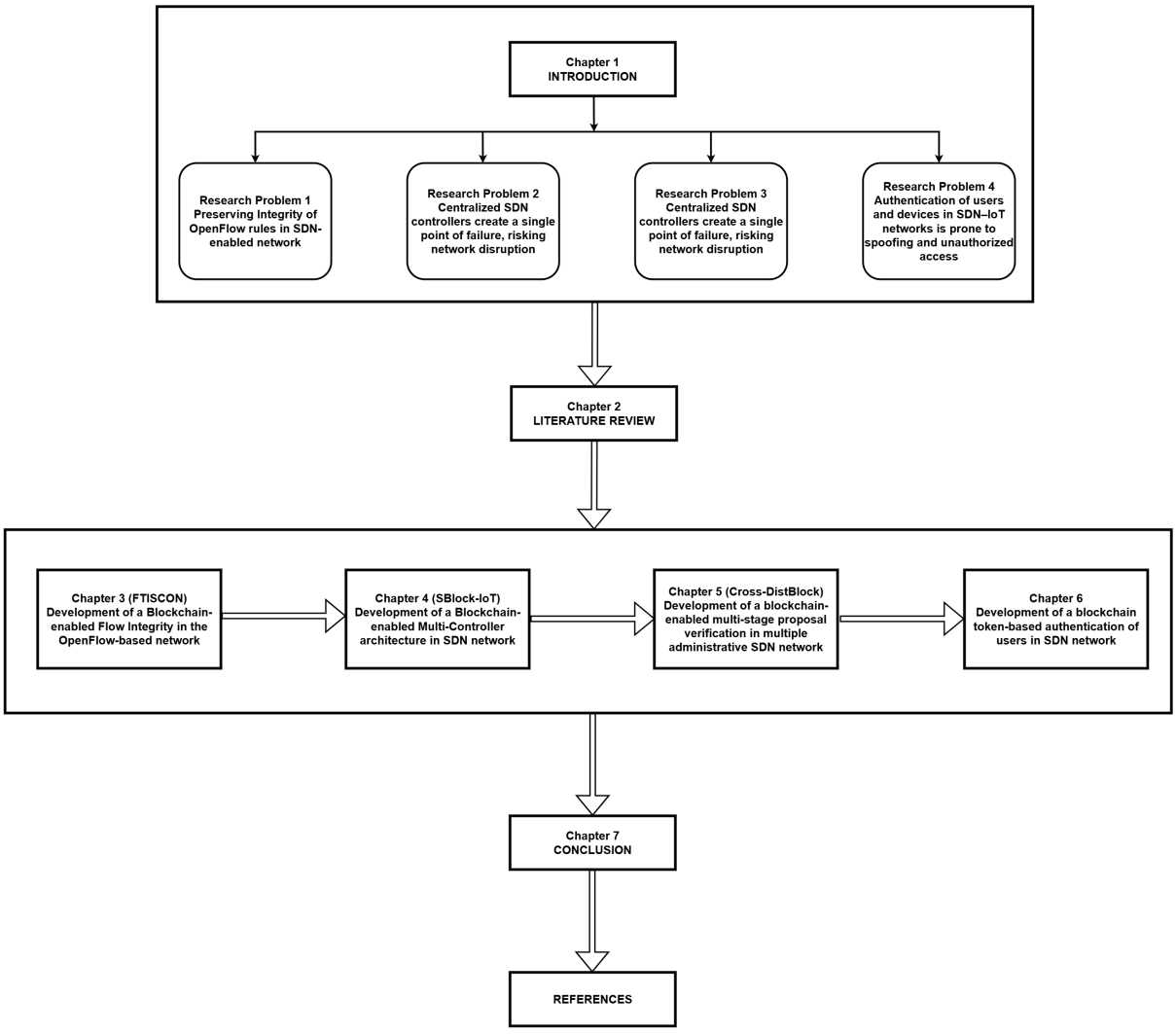

Figure 1.4: Block diagram of thesis organization.