SLECP-SDN: A Secure and Lightweight Communication Protocol in a Software-Defined Network (SDN)

Aladesote Olomi Isaiah, *Member, IAENG*, Azizol Abdullah, *Member, IAENG*, Normalia Samian and Zurina Mohd. Hanapi

***Abstract*— This study presents a novel protocol, the Secure and Lightweight Communication Protocol in Software-Defined Networks (SLECP-SDN), leveraging Elliptic Curve25519 to enhance security and efficiency in SDN Southbound Interface (SBI) communication. Unlike the existing cryptographic solutions, SLECP-SDN integrates computational efficiency, robust security, and energy optimization to address vulnerabilities in SBI. Using a Lightweight Elliptic Curve Diffie-Hellman (ECDH) approach, the proposed protocol ensures secure exchange and session establishment while mitigating critical security threats, including impersonation, replay, packet injection, and Man-in-the-middle (MITM) attacks. To evaluate the system performance, the Contiki Cooja Simulator was employed to model SDN communication among 20 hosts, incorporating various mod(p) values to assess encryption/decryption performance, energy consumption, and throughput. The Automated Validation of Internet Security Protocols and Applications (AVISPA) tool was also utilized for security verification. Using the High-Level Protocol Specification Language (HLPSL), AVISPA tested the mutual authentication protocol against three attack models: On-the-Fly Model-Checker (OFMC), Constraint Logic-based Attack Searcher (CL-AtSe), and Tree Automata-based Protocol Analyzer (TA4SP). The results demonstrated that SLECP-SDN achieves a throughput of 1224.43 MBps at a 138-bit modulus, outperforming RSA and hybrid AES+RSA algorithms. Single topology configurations delivered the fastest transmission times for encrypted files. These findings validate the effectiveness of SLECP-SDN in maintaining high-security standards without compromising network performance, making it a viable option for modern SDNs.**

Index Terms: Data Security, Elliptic Curve25519*, S*oftware-defined network, Southbound Interface, Throughput, Topology.

I. INTRODUCTION

Software-defined network (SDN) has emerged as a transformative technology that decouples the control and data planes, enabling centralized network management and enhanced flexibility [1], [2]. Unlike traditional networks, control logic and data forwarding are tightly integrated [3], [4]. SDN separates these functions, allowing for programmable and scalable network architectures [5]–[7]. This integration hampers network management and restricts adaptability. The paradigm shifts facilitate rapid innovation in network management and optimization, but they also introduce new security challenges, particularly at the Southbound Interface (SBI).

SDN relies on four key interfaces: Southbound, Northbound, Eastbound, and Westbound [8], [9]. The Southbound API is essential, facilitating communication between the control and data planes [10]. Northbound APIs offer a standardized interface for application development by providing critical insights into the underlying devices [11]. Eastbound APIs manage communication between distributed controllers, while Westbound APIs integrate legacy network devices with the SDN controller. These interfaces enable SDN to deliver more flexible and manageable network operations.

The SBI, a critical component of SDN, facilitates communication between the control plane and data plane devices, such as switches and routers. While this interface is essential for SDN's functionality, it is highly vulnerable to various attacks, including unauthorized access, man-in-the-middle (MiTM), packet injection, and impersonation attacks. Securing the SBI is crucial to ensuring the integrity, confidentiality, and availability of SDN operations [12]–[14].

Existing approaches to securing SBI communication rely on cryptographic techniques such as RSA, AES, or hybrid encryption methods. However, these methods often have significant computational and energy costs, making them unsuitable for resource-constrained environments. Moreover, some solutions lack robust mechanisms for mutual authentication, leaving networks susceptible to impersonation and replay attacks. These limitations necessitate the development of a more secure and efficient protocol tailored to SDN's unique requirements.

This study proposes the Secure and Lightweight Communication Protocol in Software-Defined Networks (SLECP-SDN) to address these challenges. SLECP-SDN leverages the Elliptic Curve25519 algorithm to provide a high-security, low-overhead solution for securing the SBI. Unlike the traditional methods, the protocol incorporates a lightweight mutual authentication mechanism, ensuring trust between communicating entities and mitigating key security threats. Additionally, SLECP-SDN employs efficient encryption and decryption processes, enabling secure data

Manuscript received August 31, 2024; revised February 13, 2025.

This work was supported and funded by Universiti Putra Malaysia (UPM).

Aladesote Olomi Isaiah is a Ph.D. student in the Department of Computer Communication and Networks, Faculty of Computer Science and Information Technology, Universiti Putra Malaysia (UPM), Serdang 43400, Malaysia (phone: +2348030657156; e-mail: gs57427@student.upm.edu.my and isaaladesote@fedpolel.edu,ng)

Azizol Abdullah is an Associate Professor in the Department of Computer Communication and Networks, Faculty of Computer Science and Information Technology, Universiti Putra Malaysia (UPM), Serdang 43400, Malaysia (e-mail: azizol@upm.edu.my)

Normalia Samian is a Senior Lecturer in the Department of Computer Communication and Networks, Faculty of Computer Science and Information Technology, Universiti Putra Malaysia (UPM), Serdang 43400, Malaysia (e-mail: normalia@upm.edu.my)

Zurina Mohd. Hanapi is an Associate Professor in the Department of Computer Communication and Networks, Faculty of Computer Science and Information Technology, Universiti Putra Malaysia (UPM), Serdang 43400, Malaysia (e-mail: zurinamh@upm.edu.my)

exchange without compromising network performance. The main contributions of the study are as follows:

1. Development of SLECP-SDN, a protocol that utilizes Elliptic Curve25519 for securing SBI communication with strong cryptographic guarantees.
2. Introduction of a mutual authentication mechanism that mitigates various attacks, such as impersonation, MiTM, replay, and packet injection attacks, thereby ensuring robust device identity verification.
3. Performance evaluation of SLECP-SDN using encryption and decryption times, energy consumption, and throughput across various file sizes and modulus values.
4. Comparative analysis with traditional cryptographic algorithms, demonstrating SLECP-SDN's superior balance of security and efficiency.

The remainder of this paper is organized as follows: Section 2 reviews related work on SBI security. Section 3 presents the proposed SLECP-SDN methodology. Section 4 also introduces the experimental setup and evaluates the results. Section 5 compares SLECP-SDN's performance with existing methods. Finally, section 6 concludes the study and suggests directions for future research.

II. LITERATURE REVIEW

Securing communication in SBI in SDN is a critical challenge, as malicious switches pose significant threats by disobeying rules, colluding with other compromised entities, or falsifying information. Recent research has introduced defense mechanisms, such as encryption and authentication techniques, to address these vulnerabilities. These approaches enhance security and preserve network integrity by leveraging SDN's programmability and centralized control. This review explores these mechanisms, highlighting their effectiveness (potential) to mitigate threats and safeguard SDN environments.

Chao *et al.* [15] synthesized realistic network topologies and flow entries derived from real-world datasets to evaluate the techniques on virtual SDN networks created using Mininet. While the active probing technique effectively reduced the required number of test packets and achieved practical fault localization times, the techniques involving statistics checking and packet obfuscation require further evaluation and optimization to address their inherent weaknesses and challenges.

Ghaly and Abdullah [16] addressed the security of data transmission in software-defined networks (SDNs) by implementing robust encryption algorithms to mitigate potential security vulnerabilities arising from the separation of control and data planes, which can compromise data integrity and confidentiality. It proposes a hybrid encryption approach combining the Advanced Encryption Standard (AES) symmetric-key algorithm and the Rivest–Shamir–Adleman (RSA) asymmetric-key algorithm. The approach encrypts the original data using AES with a 256-bit key length and then encrypts the AES key using RSA with a 4096-bit public key. The hybrid approach demonstrates better encryption time and throughput compared to RSA alone. Furthermore, the single topology scenario exhibits the lowest transmission time compared to linear and tree topologies when sending encrypted files through the SDN network.

Similarly, Alemami *et al.* [17] addressed the critical issue of data security in cloud computing, where resource sharing among clients poses risks like data theft and leakage. To mitigate these risks, the study investigates encryption techniques, including AES, DES, Blowfish, RSA, and IDEA, which transform data into cipher text. The comparative analysis evaluates these algorithms based on security, encipherment capacity, memory usage, and encryption speed. The results show that AES and Blowfish are the most efficient based on speed and memory usage, while RSA and IDEA are less secure.

Varadharajan and Tupakula [18] proposed a two-pronged security architecture to mitigate the threats posed by compromised end hosts in SDNs. This architecture aims to detect and prevent attacks targeting both the control plane (SDN controller) and the data plane (network switches) before they can reach and impact these critical components. The first part is the Security Management Application (SMA), a software component in the SDN controller. The SMA specifies and evaluates security policies leveraging the controller's global network visibility, while the second part consists of the Switch Security Components (SSCs) implemented within the network switches. The SSCs enforce the security policies the SMA defines by performing functions like flow mapping, state validation of end hosts, traffic inspection, and flow encryption if required.

Al-Hamdani and Bhaya [19] proposed a new key management scheme to address the challenges of securing communication in SDN environments due to the separation of control and data planes. This scheme ensures the secure distribution of RSA certificate keys without compromising network performance. It utilizes the RSA algorithm for key generation, a hierarchical system for key distribution, and a novel approach to prevent unauthorized access to keys. However, the proposed scheme relies heavily on the central controller for key generation and management, which could become a single point of failure or a bottleneck in larger networks.

To address the vulnerabilities from unencrypted communication channels, which allow eavesdropping and tampering between controllers and switches in OpenFlow-enabled devices, Gray et al. [20] introduced a new authentication mechanism using device fingerprinting to secure SDN environments. Experimental results show that this approach prevents unauthorized access and ensures network security. However, attackers can exploit this by mimicking static features, deceiving the SDN controller into recognizing malicious entities as legitimate switches. Mockingly examining handshake messages between the controller and switches enhanced the quality of secure sessions in the SDN data plane. This approach ensures secure communication but increases overhead, as the controller must scrutinize every message sent and received. This additional scrutiny, necessary for maintaining communication integrity and security, increases processing demands on the controller and may affect overall network performance and efficiency.

Ranjbar *et al.* [21] enhanced the quality of secure sessions in the SDN data plane by meticulously examining handshake messages between the controller and the switches, which enhanced the quality of secure sessions. The study ensures secure communication but increases overhead, as the controller must scrutinize every message sent and received.

Yigit *et al.* [22] proposed the secure distribution and management of cryptographic keys in SDN to prevent unauthorized access and maintain high performance. It uses asymmetric key generation and distribution using RSA algorithms by generating keys at a central controller and distributing them securely through SSL channels. The experimental results using an SDN testbed show that the proposed cryptography key management approach effectively secures SDN environments. However, the CPU-intensive nature of the encryption process could delay regular switch operations, and the need to store keys at the controller introduces a single point of failure.

Peng et al. [23] introduced QKDFlow, a solution that combines quantum key distribution (QKD) with a one-time pad (OTP) encryption algorithm to secure OpenFlow protocol messages. This approach is designed to prevent Man-in-the-Middle (MitM) attacks and enhance the secure communication between the control and data planes in SDN.

Adhikari et al. [24] addressed the lack of mandatory security measures like Transport Layer Security (TLS) in the OpenFlow protocol, which makes the Southbound Interface (SBI) vulnerable to MiTM attacks. They propose a combination of Elliptic-curve Diffie-Hellman (ECDH) key exchange and Advanced Encryption Standard (AES) 256 encryption to secure communication between the SDN controller and switches. The study uses Bettercap with SSLStrip to simulate MiTM attacks and validate the effectiveness of the encryption approach. However, while secure, the initial key exchange process depends on the assumption that the public keys are exchanged without interception.

The research presents the SAF-Secure Authentication framework aiming to heighten security and optimize services for entities within the SDN-IoT network. Utilizing hashing algorithms (Keccak-256) and digital certificates (Bliss-B), the study ensures the validity of entities. It assesses the proposed architecture's performance by considering computation overhead and resource utilization. The SAF architecture demonstrates enhanced security performance, improving the efficiency of message encryption. However, there is a necessity for deeper exploration into system constraints regarding authentication, particularly focusing on computation overhead and resource utilization [25]

The study in [26] addresses the lack of data plane authentication, a vulnerability that can cause controller malfunctions. Their proposed prototype, Mynah, effectively mitigates this issue with only a 4.5% increase in communication latency. Mynah introduces a novel controller and switch architecture, making it the first solution to tackle this problem.

The literature review highlights several critical research gaps in securing data planes in SDNs that need further investigation. These include optimizing statistics checking and packet obfuscation techniques, understanding the performance impact of hybrid encryption methods, and providing a tailored analysis of encryption techniques for SDNs. Centralized key management schemes present risks of single points of failure, and current authentication mechanisms are susceptible to sophisticated impersonation attacks. Enhanced security measures often increase overhead, and processing demands, affecting network efficiency, and the security of initial key exchanges relies on potentially vulnerable assumptions.

Addressing these gaps is essential for developing more effective and efficient security solutions for SDNs. Introducing Elliptic Curve25519 [27] to secure communication can address some of these gaps due to its high performance and strong security with relatively low computational overhead. Its robust cryptographic properties make it highly resistant to attacks, including impersonation, and it minimizes the additional overhead associated with enhanced security measures. Curve25519 also supports efficient key exchange management, further strengthening the security of SDNs.

III METHODOLOGY

This study introduces the Secure and Lightweight Communication Protocol for SDN (SLECP-SDN), designed to secure communication between the data plane (DP) and the control plane (CP). The proposed approach employs a pre-computed curve points strategy, enhancing computational efficiency and memory usage. It maintains robust 192-bit security while using 128-bit encrypted keys.

The protocol uses Elliptic Curve25519 for efficient, high-security encryption and key exchange, ensuring robust data transfer protection within the SDN. Additionally, integrating the GMP library boosted the performance of scalar multiplication and reduced the cost of generating large prime numbers. The study uses Elliptic Curve25519 for its cryptographic strength and efficiency. Figure 1 presents the secure communication flow of SLECP-SDN. SLECP-SDN involves the following phases: key generation, key exchange, encryption and decryption, simulation, and mutual authentication.

A). Key Generation

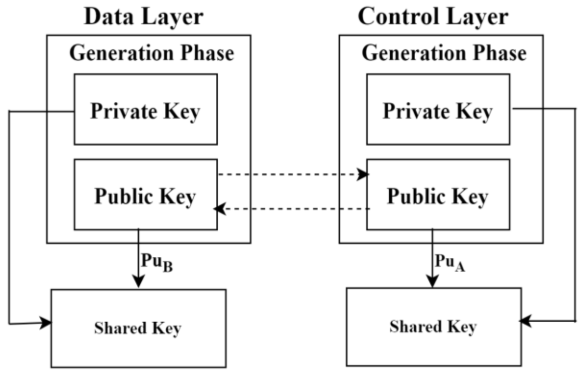

Fig. 1. Secure Communication Flow in SLECP-SDN

In this phase, both the control plane (CP) and the data plane (DP) dynamically generate private-public key pairs using Elliptic Curve Cryptography (ECC). A secure cryptographically strong pseudo-random number generator (CSPRNG), implemented via Python's os.urandom(), generates the private keys, while public keys are derived through scalar multiplication of the private key and curve base point G, as represented in equation [1]:

$$P = K * G \quad (1)$$

Where P is the public key, K is the private key, and G is the base point. The algorithm for scalar multiplication is detailed in Algorithm 1, which ensures efficient point addition and doubling using modular arithmetic.

This curve, defined by the equation $y^2 = x^3 + 486662x^2$ + x mod(p) (Montgomery, 1987), operates over a prime finite field, ensuring strong encryption using modular arithmetic, specifically:

$$P = 2^{255} - 19 \quad (2)$$

The curve's base point serves as the foundation for generating all other points on the curve, which are crucial for encryption and decryption processes. Each SDN device is assigned unique elliptic curve points generated as part of the encryption system.

Algorithm 1: Point Addition and Point Doubling

Input: Point X = (x_1, x_2); Point Y = (y_1, y_2)
Output: Point Z= $(x_3 . y_3)$

If X is the point at infinity
  return Y
end If
If Y is the point at infinity
  return X
end if
If $x_1 = x_2$ and $y_1 \neq y_2$
  return the point at infinity.
end if
If x ≠ Y, calculate slope m
  m = $\frac{y_2 - y_1}{x_2 - x_1}$
 else if x = y, calculate
  m = $\frac{3x_1^2}{2y_1}$
 end if
 calculate the coordinates of Z
  $x_3 = m^2 - x_1 - x_2$
  $y_3 = m(x_1 - x_3) - y_1$
End

Algorithm 1 outlines the process of scalar multiplication (point addition and point doubling) on Elliptic Curve 25519 using modular arithmetic. It takes two input points, X and Y, each with two coordinates, and produces an output point, Z. Special cases are handled first: if either X or Y is at infinity, the algorithm returns the other point, and if the x-coordinates of X and Y are the same but their y-coordinates differ, it returns the point at infinity. For other cases, the slope m is calculated. If the points are different, the slope is m = $\frac{y_2 - y_1}{x_2 - x_1}$; if they are the same (point doubling), the slope is $\frac{3x_1^2}{2y_1}$. Finally, the coordinates (x, y) of the output point Z are calculated using $m^2 - x_1 - x_2$ and $m(x_1 - x_3) - y_1$ respectively.

From Fig. 1, let the data plane be A and the control plane be B. The private keys for A and B are represented as Pk_A and Pk_B, respectively. The corresponding public keys for A and B are given by:

$$Pu_A = Pk_A * G \quad (3)$$

$$Pu_B = Pk_B * G \quad (4)$$

Where G is a predefined generator point on the elliptic curve.

B). Key Exchange Phase

The protocol employs the Elliptic Curve Diffie-Hellman (ECDH) method to enable CP and DP to establish a shared secret over an insecure channel. During the exchange, Device A (DP1) sends its public key (Pu_A) to Device B (CP1), which responds by sending its public key (Pu_B) to Device A. In the shared secret exchange, A computes its shared secret (sPu_A) using B's public key (Pu_B) and A's private key (Pk_A), as shown in equation (5). Similarly, B calculates its shared secret (sPu_B) by using A's public key (Pu_A) and B's private key (Pk_B), as shown in equation (6). Both calculations result in the same shared secret key, enabling secure communication between A and B. This protocol ensures that the DP and CP can derive a shared secret, even over an insecure communication channel, as depicted in Fig. 2.

$$sPu_A = Pk_A * Pu_B \quad (5)$$

$$sPu_B = Pk_B * Pu_A \quad (6)$$

These principles also apply to DP2 and CP2, ensuring consistent exchange security. The public key is openly shared, while the private key remains confidential.

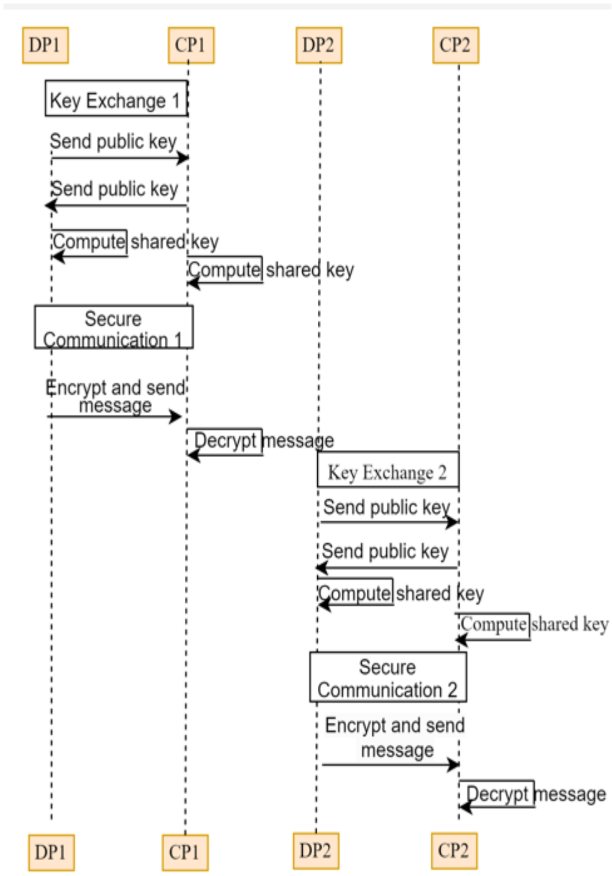

Fig. 2. Sequence diagram for public key exchange in SLCEP-SDN

C). Encryption and Decryption

Once the shared key is established, it is used for encryption and decryption. Encryption is performed as follows:

$$ECC(Char(i)) = Shared_{Key} + Char(i) \quad (7)$$

$$Encrypted\ Message = \sum_{i=1}^{n}(Shared_{key} + Char(i)) \quad (8)$$

Decryption reverses this process to recover the original plaintext:

$$Decrypted\ Message = \sum_{i=1}^{n} (Encrypted\ Message - Shared_{key}) \quad (9)$$

$$Char(i) = \left(Encrypted\ Message\ (i) - Shared_{key}\right) \quad (10)$$

The sender (DP) encrypts the plaintext using the shared key, turning it into ciphertext, and then transmits it over the network. Upon receiving the ciphertext, the recipient (CP) uses the same shared key to decrypt it and recover the original plaintext. This process ensures the confidentiality of the communication, protecting the transmitted messages from an unauthorized access.

D). Simulation and Authentication

The system was modelled using Contiki's Cooja simulator, simulating SDN communication between 20 hosts. The protocol's robustness against attacks was verified using the Automated Validation of Internet Security Protocols and Applications (AVISPA) tool, employing High-Level Protocol Specification Language (HLPSL). It ensured resistance against attack models like On-the-Fly Model-Checker (OFMC), Constraint Logic-based Attack Searcher (CL-AtSe), and Tree Automata-based Protocol Analyzer (TA4SP).

E). Proposed Mutual Authentication Technique

This section details the key features of the proposed approach. The technique aims to ensure optimal performance in constrained networks while providing the most cost-effective security for hosts in SDN networks. It addresses fundamental security components, including confidentiality, authentication, and data integrity. The data transfer process between the sender and receiver hosts is described as follows:

a. To create trust, a host must authenticate the relevant device before sending or receiving data to or from an adjacent host. The sending and receiving hosts will execute an ECDH-based authentication key agreement protocol to authenticate mutually.
b. For the ECDH process, host_1 and host_2 generate their' private keys such that Host1's private key is Prv_H1 and Host_2's private key is Prv_H2.
c. The receiving host must ensure that the data received has not been altered during transmission once the mutual authentication has been established. Similarly, data must be protected from eavesdroppers and Man in the Middle (MiTM) during transmission. End-to-end encryption is typically used to protect data from these types of assaults.

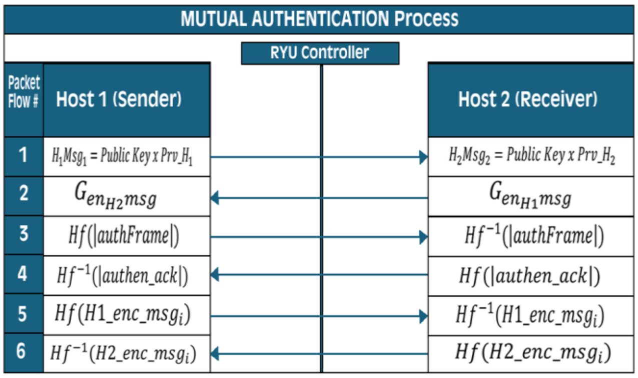

Fig. 3. Proposed Lightweight Mutual Authentication Process

d. Encryption is employed during the key exchange stages, enabling secure end-to-end encryption between the two constrained SDN hosts. The keys used in this procedure are public keys such as embedded network keys (Net_k = Network_id + Public_Key), which are the same for one network only.

The data flow of ECDH-based mutual authentication is shown in Figure 3, where the notations used in the proposed technique are shown and elaborated upon in Table 1.

TABLE 1
NOTATIONS IN THE PROPOSED SCHEME

Notations	Description	Generator Size	Key pair Size
H_1Msg_1	Host_1 message	64*d	128*d
H_2Msg_2	Host_2 message	64*d	128*d
$Public_{key}$	Public key	64	128
Prv_H_1	Private Key of Host_1	64	128
Prv_H_2	Private Key of Host_2	64	128
N_k	Unique Network key	32	64
G_{enH_1msg}	Generator message by Host_1	64	128
G_{enH_2msg}	Generator message of Host_2	64	128
$\|authFrame\|$	Authentication Frame string	128*d	256*d
$\|authen\_ack\|$	Authentication acknowledge Frame string	128*d	256*d
$Hf(i)$	The hash function for Encryption	64*c	128c
$Hf^{-1}(i)$	Inverse Hash for decryption	64*c	128c
$H1\_enc\_msg_i$	Sender encrypted data block	64*d	128*d
$H2\_enc\_msg_i$	Receiver encrypted data block	64*d	128*d

Note: d is the number of data block characters. c is the number of ASCII characters

Data transmission and authentication steps are stated below:
1. Host 1 sends the message, H_1Msg_1, encrypted through the private key Prv_H_1 of host 1.
2. Host 2 receives the message, makes knowledge of Prv_H_1, adds Prv_H1 with the message, H_1Msg_1 and sends a reply of H_2Msg_2.
3. Host 1 has both privates at this stage. It now encrypts and sends an authentication frame, $|authFrame|$, which is encrypted using both private keys.
4. Host 2 receives and decrypts the message using Prv_H_1, adds Prv_H_2, reads $|authFrame|$, and sends an authentication acknowledgement frame, $|authen\_ack|$.
5. After a successful acknowledgement frame, Host 1 encrypts and sends a real message G_{enH_1msg} using hash function $Hf(i)$. The encrypted block is now $H1\_enc\_msg_i$.
6. Host 2 receives the encrypted block, $H1\_enc\_msg_i$, decrypts it, G_{enH_1msg} through $Hf^{-1}(i)$ and sends $H2\_enc\_msg_i$ as completion of this session.

F). Experimental Setup

The protocol was tested on a Dell Inspiron 5402 with an 11th Gen Intel® Core™ i7-1165G7 @ 2.8GHz processor and 16GB RAM, running Ubuntu 18.04, Python 3.8, Mininet, and Ryu 4.12. Three SDN topologies—single, linear, and tree—were modelled. Performance evaluation included encryption/decryption times for varying text file sizes. The gmpy2 library (version 2.2.0) with mini-GMP was employed to optimize cryptographic operations.

IV RESULTS AND DISCUSSION

This section presents our findings on securing communication between the data plane and the control layer using the Elliptic Curve25519.

A). Reference Table

The Reference Table lists ASCII characters for message encryption, with each character mapped to a unique point on the elliptic curve. These curve points are pre-calculated to minimize runtime computation, unlike the secret key (SK). Table II includes all 128 ASCII characters, such as uppercase and lowercase English letters, digits (0–9), punctuation marks, and control characters (e.g., carriage return and line feed).

Each character is associated with a large random prime integer, which serves as a generator point (G) on the curve. These points are derived using scalar multiplication from a set generator value, which can vary depending on network settings, a public key, or periodic updates. The generator number for each ASCII character is calculated using a large random prime integer. This is essential for determining the curve's specific (x, y) coordinates, starting from G with x = 9 and y = 6248, under the modulus 1019532643. Table 2 provides data on cryptographic operations, including private key generation and elliptic curve point coordinates. It includes columns for iteration numbers, ASCII values, and the x and y coordinates of the curve points. The randomness in the data shows that the cryptographic processes were carried out securely and unpredictably. This randomness is critical for security, as any patterns in key generation or curve points could be exploited by attackers, making the system vulnerable.

B). Avalanche Effect

The avalanche effect refers to the time taken to encrypt and decrypt data blocks and the changes in bit patterns before and after encryption. Fig. 4 shows a scatter plot with ASCII values on the x-axis and PDF values on the y-axis. The data points cluster around low PDF values, indicating uniformity, unpredictability, equal probability, and randomness in the data across the ASCII range. The graph shows a near-normal distribution, where each byte has an equal chance of occurring. This uniformity is crucial in encrypted data, as it prevents patterns that attackers could exploit.

C). Computational Performance of Encryption and Decryption Time

This section highlights the computational performance of the Elliptic Curve25519 algorithm for encrypting and decrypting text files of various sizes with different mod(p) values. Table III shows that encryption time increases with file size and varies across different mod(p) values. For instance, encrypting a 106MB file takes between 375.1931µs (mod(p) = 18) and 638.905µs (mod(p) = 108). Decryption times also differ and do not always match encryption times, with the same 106MB file taking between 348.4432µs (mod(p) = 138) and 774.9815µs (mod(p) = 108), as shown in Figure 5.

D). Energy Consumption for Encryption and Decryption

This section presents the energy consumption during encryption and decryption for various input values, as Table IV depicts. The data reveals that energy usage increases with file size and longer cryptographic keys. Larger files and longer bit-length keys require more computational power and time, leading to higher energy consumption. For example, Curve25519, with a 138-bit key, offers greater security but consumes more energy for encryption than shorter keys. CPU power usage, reflected in energy consumption, generally rises with file size and mod(p) values. For instance, encrypting an 11.804MB file at mod(p) 138 uses 0.01207141 mJ/s, nearly double the 0.00651434 mJ/s at mod(p) 18. However, the relationship between mod(p) values and computational time is inconsistent. Larger mod(p) values enhance security but require more time and energy.

E). Throughput

Throughput measures how efficiently encryption operates without creating performance bottlenecks. In SDN, the controller must quickly respond to data plane events, and any delay can degrade performance. Table 5 and Figure 6 show the throughput (in MBps) for various modulus bit lengths in Curve25519 operations. The 138-bit modulus achieves the highest throughput at 1224.43 MBps, offering the best balance between security and efficiency. In contrast, the 108-bit modulus has the lowest throughput at 984.24 MBps due to more computationally intensive operations. Throughput also drops for larger and smaller bit lengths, with the 253-bit modulus at 1073.31 MBps and the 72-bit modulus at 1102.67 MBps. This indicates that bit lengths above 138 increase computational load without significant security gains, while shorter bit lengths may boost speed but weaken security.

F). Comparison of Throughput Results Across Different Algorithms

Figure 6 shows that Curve25519, with a 138-bit modulus, achieves the highest throughput at 1224.43 MBps, while RSA has the lowest at 33.52 MBps. The hybrid AES + RSA algorithm consumes more memory despite its low throughput. Curve25519's high throughput highlights its superior computational performance and fastest encryption times among the compared algorithms. Larger keys offer stronger security by requiring more computation to break the encryption. High throughput is crucial for optimal performance in high-speed networks like data centres and SDNs, which handle large data volumes. Curve25519 consistently outperforms algorithms like those used by Ghaly and Abdullah [16], providing better performance in these environments.

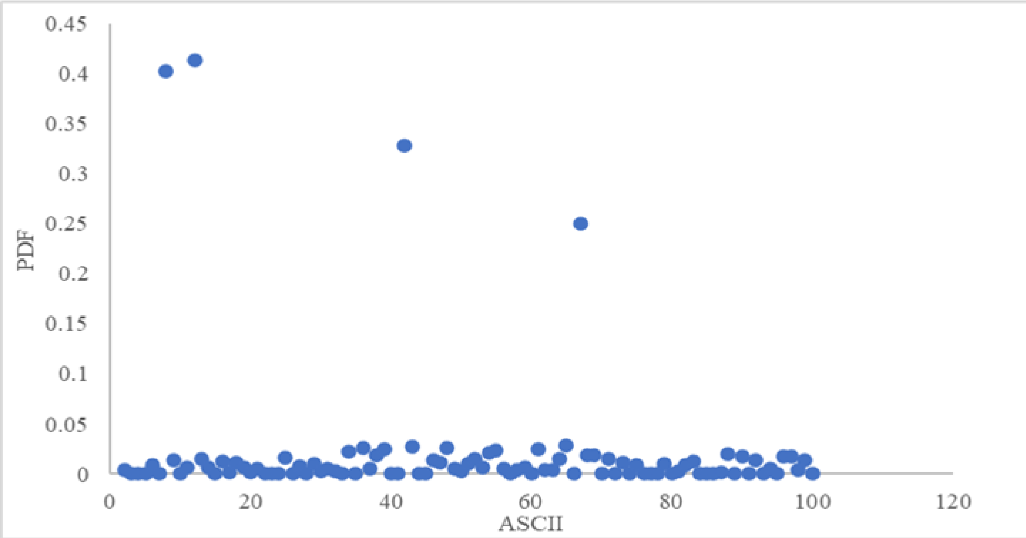

Fig. 4. Uniformity in encrypted frames

TABLE II
REFERENCE TABLE RESULTS

Ci	ASCII	Curve point (x)	Curve point (y)
1	\x00	149726810710315296553937312724301	24600957519473220769194735521699
2	\x01	163174634229599258874334183193398	85130006238113929278204390999710
3	\x02	780558091608003967686489978491883	72839741149170059733587718842234
4	\x03	137801430053945588459731715413622	158648653400952353340375015959402
5	\x04	173922987490323721209269691240318	101020736503289960385653001016359
6	\x05	157320079731207275078488946584733	121136843483945486248145127716954
7	\x06	156151758736574461367152050556143	109986854106365416332928871443683
8	\x07	191958799062696965345711303387942	132703531872930875468252346052149
9	\x08	8400387075025926071918878726382	45446136002036367512488286630135
10	\t	121743830913937608230457439173814	72403069454975549149926810602113
11	\n	197371859685415251780241189082531	55644391545652875760106396344570
12	\x0b	43287691868693190418946629180498	48203497305583214396591157548082
13	\x0c	1605531985446277787491027134457727	12493726006699614011029499893107
14	\r	98866145589130550589368806894057	132327937137858778593395070483899
15	\x0e	9071825616161890397453917942257	195010370010647726939696191003798
16	\x0f	197771975788040659340258421333968	150828200468320200456797791741440
17	\x10	159706020404494218147144263608179	107469284392674855392862852139385
18	\x11	51569769450651939011773900111107	1275543270979720175632549870553
19	\x12	34681501817146951098488246606362	117347760686959690165580149303838
20	\x13	5964526546791930670855312116629	28351474552900309024095183355180
21	\x14	24750915065416558059577050971126	118634230687814148512214991106033
22	\x15	191925054553661560390367578452552	47297171615120066510971033239849
23	\x16	48654887353283405174044353107653	189971426246448134522211951351040
24	\x17	12987369635410323020661696141373	56370959838297157328022881778406
25	\x18	58921890190933091573959672700960	131430224725100888345881575668680
26	\x19	90844289010019972680798400225182	174417308842853290691526996724895
27	\x1a	33422292471199233754465466573700	3992493444415362507375971279785
28	\x1b	819393018769986793912755029787	186589913907986804354355941385297
29	\x1c	16828623595101960755138331981848	194730759428046342543014793804
30	\x1d	43754598955246394193836544665671	157067047071810528446825274677245
31	\x1e	135551855283630705269492082093937	168102183784710518283377434124348
32	\x1f	169834897400631511353146427474768	13517088884723521426700603369916
33		67366257666644977822771863279316	12756379734990538064447403923278
34	!	1890153007922555264921718603	137838611113022894684741228810054
35	"	192928265730552011736012122413142	84764903331782183678801377123774
36	#	109844769974466109803877548361161	109407167209614069729454097758486
37	$	18311938370555316803776127250489	30606541599324377379447354480753
38	%	153015475417273965582569868444936	28011654104320990542643794927009
39	&	55048963593969599965163775948594	153617908701962226084569434909194
40	"	165825161646636817757748097178322	359174189846693826053564977105581
41	(	150867669022016994007908950114126	28246445392313803021013769527197
42	)	163098789831733149520901723958754	14112116185099818521028844627014
43	*	8069656439222140138900532800081	175941373947063280950157049086165
44	+	174131584438625841897229171361424	156877929374343254571461590487940
45	,	139189422720162185628505107465436	36076575012238035568208634535353
46	-	105275389736211994004716149622018	66489180054101744788362967225974
47	.	88102741523736263776853556938104	121204768434919897842262997584153
48	/	15094530751156508767749357961716	4256256178318410073087145156241
49	0	19665573300053478380044888782393	74235560923948052540267765007707
50	1	20518402432411406354108180054620	71702364481054182937196396316535

51	2	177509877541323754009262777497781	93913153114224778021042901992043
52	3	67270278397563451889975454207557	128525217027178677066250961415349
53	4	61601395650627617163998757183891	190393509614190102170780004198446
54	5	85223808028520395349272524121401	129662053273846207206224981909083
55	6	22873118291058051063228939523954	116313090570426320851072204819663
56	7	69476060406227291017501083516142	25833045339160927227760846401432
57	8	5346603525403728622223192329742	14340888483576693798223267923907
58	9	182726681379808148862289393529311	171282290328837196829488212123894
59	:	107599524645755463795155863770315	157534235571666312725026848591073
60	;	112060649054870949938405367040014	22866415025676792799644108987775
61	<	110069068060128478094851261088124	23226356669582304996442750285034
62	=	94648842675991353118796965944811	138517205562071362960226437824565
63	>	74935792642933298997998359868013	122262367430031813451562427079254
64	?	186547804817864498376832230696176	73965414845807207181768267176051
65	@	25447667958600840488721935076978	14152845921732743401774474241639
66	A	190030360289311768509092526491925	156480713533070104902867091671716
67	B	120793742729179040680462351806730	58448003460249953044462939394235
68	C	355900944747607790509003350995723	180374969813254870853114806192967
69	D	158193822959635885956072885135849	12834920660988076497852092205056
70	E	29442954241562060281671061401125	102647578816308278544643697662843
71	F	167779624595535581617587035526309	75428078360571754011246137627752
72	G	163036966066320695551119895927071	140416728808897714731241264758581
73	H	77287431836257600783818615663859	182397745792442612186859658857506
74	I	108686101027758476932464390592521	7247780024318848925331871461058
75	J	102331607654068877124711323416043	27548070178070857177285783050491
76	K	37649063438995812033282888101506	8676226898966166863236575493204
77	L	128876816620018786723049343831133	45054404728556752109167082660893
78	M	164857361653411862953209498478471	132517706595580004936491831506744
79	N	125461943118669953593211954318781	14700275254570640087321604268291
80	O	7009281048565051667880348070113	26432586399622891061185448632822
81	P	99713854957795943700974698049417	153829432406835011731703224913931
82	Q	14481150461500860169884091078736	161319534873142692265736012126260
83	R	24190514050144509882908182786964	69503727540034617742601551359424
84	S	165863255567946723867583381511093	192306387060460792280187761246352
85	T	171024704622051136504113568735294	96334538737065967688644328597686
86	U	119237079229441536917595561683547	44792649499058175745316344191302
87	V	20296176349580042293351379143355	10735812669345783224208082070166
88	W	73851923249152221564684885048947	169673183445614345241076592647379
89	X	42404110729088076738834892855727	165193520080956151169273085540214
90	Y	46716303657442930319591842731896	129576732878270970737326467007253
91	Z	142155738567240393113660431381586	146354843602324778212647078704776
92	[	135670151295696880609032601466280	162897270552877489690891329239163
93	\\	34516693003651721285338558617494	76226152132985243346402217357065
94	]	12828799314404994855106002747104	35800368635721308244532557541672
95	^	115555074461536010570004985062981	178697204763462838897319226170038
96	_	34798310765417988065618919219649	58414861312588046947029245178854
97	`	40311917456801293571876398123097	608582384359726646440535055977933
98	A	25201534919257264972352785531958	156943284483372127078888283435943
99	B	7853383294300904418699292881578	94966530060641782308769405025518
100	C	14257342686283904597888711491791	45202597864762652025089240987424
101	D	38079703381696206275196224663486	58400685412852677670706755377545
102	E	45754318113363907213290807399706	162804139376036548434952246241157
103	F	54603626980736375626574917973966	13452747470170595478748680068925
104	G	88310300150614603403428840866155	32061354555739246906426728504402
105	H	110324834114529365338627179220579	56350336757626433679739085552294
106	I	114962103403635324894198774726010	52106929978961837597359788332568
107	J	178352557396510142592676825983891	59328012611119213963933924806248
108	K	136893011504697995122275622579664	143415225557196245248243287418171
109	L	25644472604057487620160279983372	153746726294407072170519780611947
110	M	76284094693932509757161992494670	161479730146468807420454834406518
111	N	170958508014520504665129973870710	17147802630023397634734007394712
112	O	121196246606032260496084437330837	79058571393105006032715524873461
113	P	195168383030345801074816170176780	145951193110984762837294127256437
114	Q	186398058951962845660831462230733	155837808768653032308876190617550
115	R	174194187340190280319970993056211	114683818833633285896732555053990
116	S	109208176658095235959426144532991	136940612168538418999112306803345
117	t	186210701203574451034199896482324	29708949802675601753411873075099
118	u	53422078656993150928661280403664	151708204674122501091116616968566
119	v	16273862062814979909180619891943	7944715892388243628941844281573
120	w	91023698295672401601788723116128	20288898129341375329081035179403
121	x	139727684514397351197336984785237	193624016139554199783631283412203
122	y	194505428411659469405109674142988	125107523856880602179462363678891
123	z	116992063625517654738613317717161	82003517206186342382599394505019
124	{	60294121170902891455939443181336	87038905202294191487610086311196
125	|	2535734540927065882169831281213	153325678348891734881602883923566
126	}	130082119544515292709886022837111	150908601307678423067909547975790
127	~	11387995585517660791168964790177	101159741031841232682625855654540
128	\x7f	178929108429204537220187123714203	83115428262923441662558170926958

G). Securing Data

Safeguarding sensitive data is very crucial with the evolving cybersecurity threats. One of the most robust and efficient methods of securing data, especially in SDN networks, requires a reliable algorithm, which ECC is one which offer strong encryption while minimizing computational overhead. The elliptic curve 25519 has exceptional security properties and performance characteristics compared to the various ECC curves. Implementing elliptic curve25519 to secure data in SDN requires striking the right balance between security and computational efficiency. Analysis, as presented in Table V and Fig. 7, reveals that adopting a 138-bit mod(p) value with curve25519 balances robust encryption and efficient computation. The transmission of files encrypted with a 138-bit mod(p) value of the ECC curve25519 occurs through the SDN network using the Ryu controller with three topologies: single, linear, and tree. Table 6 presents the transmission times (milliseconds) for encrypted text files across network topologies.

TABLE III
COMPUTATIONAL TIME PERFORMANCE OF ENCRYPTION AND ENCRYPTION

Text file	P-value	Bit Size	Encryption Time	Decryption Time
915KB	137849	18	28.17	17.33
	5171003929967	43	24.83	15.50
	3044861653679985063343	72	13.71	14.48
	198211423230930754013084525763697	108	12.85	13.07
	276602624281642239937218680557139826668747	138	15.07	14.97
	1447401115466452442794637312608598848160326 3447650325797860494125407373907997	253	15.16	16.67
5.384MB	137849	18	30.25	35.12
	5171003929967	43	29.10	29.22
	3044861653679985063343	72	25.92	49.47
	198211423230930754013084525763697	108	30.87	30.21
	276602624281642239937218680557139826668747	138	29.12	27.24
	1447401115466452442794637312608598848160326 3447650325797860494125407373907997	253	28.77	31.56
11.804MB	137849	18	65.14	64.18
	5171003929967	43	71.99	66.51
	3044861653679985063343	72	70.03	66.29
	198211423230930754013084525763697	108	84.37	62.67
	276602624281642239937218680557139826668747	138	120.71	76.51
	1447401115466452442794637312608598848160326 344765032579786049412540373907997	253	57.20	58.05
35.350MB	137849	18	229.99	334.31
	5171003929967	43	241.77	206.49
	3044861653679985063343	72	206.49	234.83
	198211423230930754013084525763697	108	160.30	172.52
	276602624281642239937218680557139826668747	138	171.81	230.41
	1447401115466452442794637312608598848160326 344765032579786049412540373907997	253	245.98	228.07
59.809MB	137849	18	360.77	333.67
	5171003929967	43	346.11	349.67
	3044861653679985063343	72	410.10	303.10
	198211423230930754013084525763697	108	405.74	410.98
	276602624281642239937218680557139826668747	138	319.92	472.83
	1447401115466452442794637312608598848160326 344765032579786049412540373907997	253	245.65	316.53
106MB	137849	18	375.19	383.95
	5171003929967	43	440.82	402.22
	3044861653679985063343	72	438.37	459.07
	198211423230930754013084525763697	108	638.90	774.98
	276602624281642239937218680557139826668747	138	417.70	348.44
	1447401115466452442794637312608598848160326 344765032579786049412540373907997	253	632.83	654.76

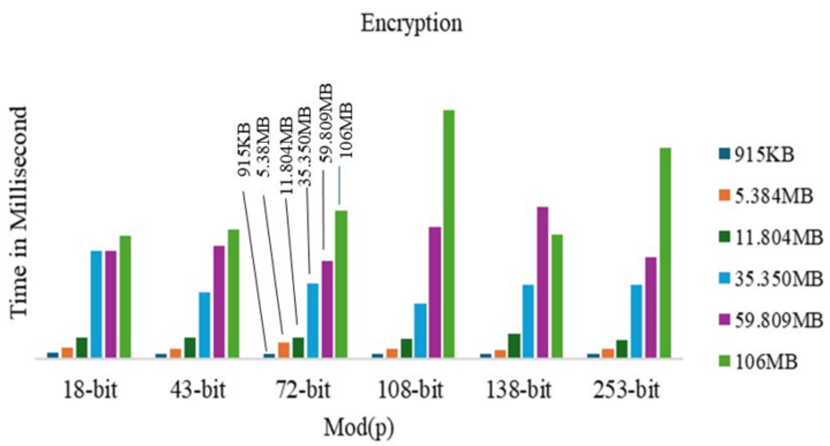

Fig. 5. Graphical Analysis of Encryption Computational Time Performance

TABLE IV
ENERGY CONSUMPTION OF ECC CURVE25519 FOR ENCRYPTION AND DECRYPTION

Text File	P-value	Bit Size	Encryption Energy	Decryption Energy
	137849	18	0.0028	0.0017
	5171003929967	43	0.0025	0.0015
	3044861653679985063343	72	0.0014	0.0014
	198211423230930754013084525763697	108	0.0013	0.0013
	276602624281642239937218680557139826668747	138	0.0015	0.0015
915KB	14474011154664524427946373126085988481603263447650325797860494125407373907997	253	0.0015	0.0017
	137849	18	0.0030	0.0035
	5171003929967	43	0.0029	0.0029
	3044861653679985063343	72	0.0026	0.0049
	198211423230930754013084525763697	108	0.0031	0.0030
	276602624281642239937218680557139826668747	138	0.0029	0.0027
5.384MB	14474011154664524427946373126085988481603263447650325797860494125407373907997	253	0.0029	0.0032
	137849	18	0.0065	0.0064
	5171003929967	43	0.0072	0.0067
	3044861653679985063343	72	0.0070	0.0066
	198211423230930754013084525763697	108	0.0084	0.0063
	276602624281642239937218680557139826668747	138	0.0121	0.0077
11.804MB	14474011154664524427946373126085988481603263447650325797860494125540373907997	253	0.0057	0.0058
	137849	18	0.0230	0.0334
	5171003929967	43	0.0242	0.0206
	3044861653679985063343	72	0.0235	0.0160
	198211423230930754013084525763697	108	0.0164	0.0173
	276602624281642239937218680557139826668747	138	0.0172	0.0230
35.350MB	14474011154664524427946373126085988481603263447650325797860494125540373907997	253	0.0246	0.0228
	137849	18	0.0361	0.0334
	5171003929967	43	0.0346	0.0350
	3044861653679985063343	72	0.0410	0.0303
	198211423230930754013084525763697	108	0.0406	0.0411
	276602624281642239937218680557139826668747	138	0.0320	0.0473
59.809MB	14474011154664524427946373126085988481603263447650325797860494125540373907997	253	0.0246	0.0317
	137849	18	0.0375	0.0384
	5171003929967	43	0.0441	0.0402
	3044861653679985063343	72	0.0439	0.0459
	198211423230930754013084525763697	108	0.0639	0.0775
	276602624281642239937218680557139826668747	138	0.0418	0.0348
106MB	14474011154664524427946373126085988481603263447650325797860494125540373907997	253	0.0633	0.0655

TABLE V
ELLIPTIC CURVE25519 THROUGHPUT

	18-bit Mod(p) Total Time (µs)	43-bit Mod(p) Total Time (µs)	72-bit Mod(p) Total Time (µs)	108-bit Mod(p) Total Time (µs)	138-bit Mod(p) Total Time (µs)	253-bit Mod(p) Total Time (µs)
915KB	28.14	24.83	13.71	12.85	15.07	15.16
5.384MB	30.24	29.10	25.92	30.87	29.12	28.77
11.804MB	65.14	71.99	70.03	84.37	120.71	57.20
35.350MB	229.99	241.77	234.83	163.76	171.81	245.98
59.809MB	360.77	346.11	410.10	405.74	319.92	245.65
106MB	375.19	440.82	438.37	638.91	417.70	632.83
Throughput	1207.41	1139.28	1102.67	984.24	1224.43	1073.31

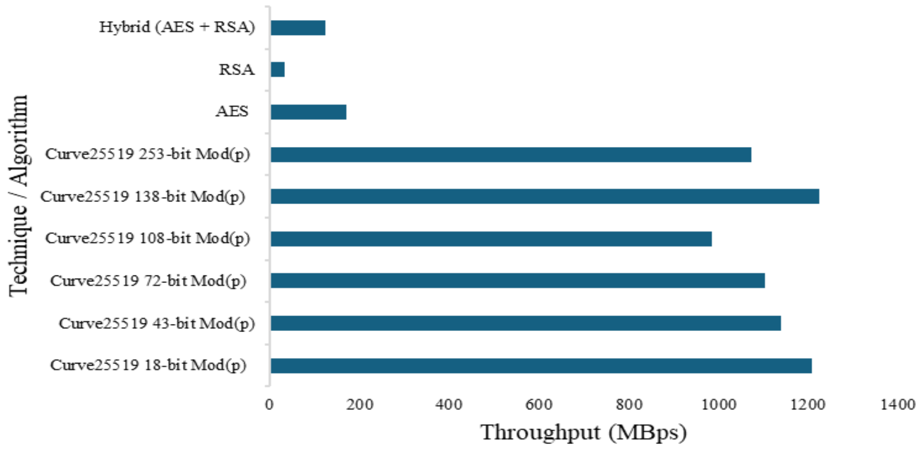

Fig. 6. Comparison of Throughput Across Different Security Algorithms

Analysis, as presented in Table V and Figure 6, reveals that adopting a 138-bit mod(p) value with Curve25519 balances robust encryption and efficient computation. The transmission of files encrypted with a 138-bit mod(p) value of the ECC Curve25519 occurs through the SDN network using the RYU controller with three topologies: single, linear, and tree. Table 6 presents the transmission times (milliseconds) for encrypted text files across network topologies.

Table VI
TRANSMISSION TIMES OF ENCRYPTED TEXT FILES OVER DIFFERENT NETWORK TOPOLOGIES

Text Files	Single Topology	Linear Topology	Tree Topology
Text1	0.0036	0.0042	0.0099
Text2	0.0052	0.0055	0.0156
Text3	0.0053	0.0076	0.0205
Text4	0.1013	0.1045	0.2450
Text5	0.1115	0.1165	0.2250
Text6	0.1301	0.1367	0.3540
Total Time (µs)	0.357	0.375	0.87

Fig. 7 shows that it takes the least Time for a single topology to transmit encrypted files compared to other topologies, which implies that a single topology transmits encrypted files faster than other topologies.

H). Analysis of Mitigating Attack Vectors

Integrating robust encryption and mutual authentication techniques poses a significant challenge for attackers trying to bypass security measures. Consequently, prevalent attack routes, such as Man-in-the-Middle (MiTM) attacks, packet injection, and unauthorised access attempts, are effectively mitigated, thereby enhancing the overall security posture of the SDN environment as detailed as follows:

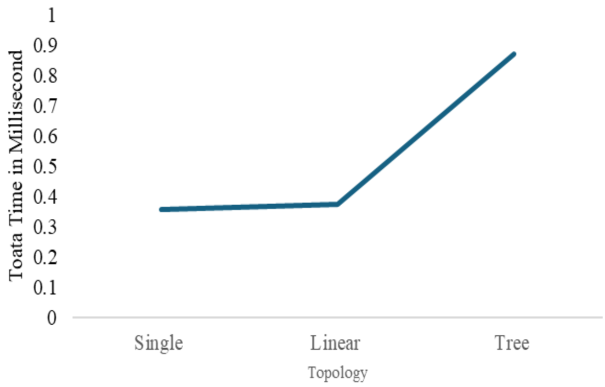

Fig. 8. Graphical Analysis of Transmission Times of Encrypted Text Files over Different Network Topologies

1). Attack by Impersonators from Host 1

Assume an attacker has gained possession of H1. This means the attacker has access to the network key and the public keys. The attacker then replaces H1 with a counterfeit device, Fake_H1. Fake_H1 uses a random number generator to compute the private key of H1 and determine curve points before initiating communication with H2. H2 computes H1_Msg1 and returns H2_Msg2. However, for the impersonator (Fake_H1) to verify authenticity, they need to know |aF|. Since |aF| is embedded in the firmware, it is impossible for the impostor to obtain this value.

2). Attack by Impersonators from Host 2

When the attacker obtains H2, they replace it with a fake device known as Fake_H2, and they are aware of all the previously mentioned parameters. Similarly, fake host gets

H_1Msg_1 from H1 and computes Prv_H_2. The H2 then computes $G_{en_{H2}msg}$ and returns Prv_H_2 . Following the receipt of |aF| from H1, Fake_H2 will have to wait for the authentication acknowledgment frame from H1. In this case, the authentication procedure won't start in the absence of an authentication acknowledgment frame. The session key will expire since H1 will not be able to validate authentication and will not be able to send actual data to Fake_H2 because the authentication acknowledgment frame is not available. It demonstrates the resilience of our approach against impersonation attacks.

3). Man-in-the-Middle (MiTM) Attack

The attacker continuously watches the transmission channel for messages sent back and forth between H1 and H2. With time, MiTM can comprehend and imitate the data that is transmitted. Likewise, data spoofing allows for the analysis of H1 and H2 messages over an extended period, which allows for the prediction of keys and the discovery of |authentication Frame string| and |acknowledge frame string|. However, authentication frames are encrypted when end-to-end encryption is used. MiTM finds it very challenging to decrypt and analyze the sent data because of the elliptic curve-based end-to-end encryption. Moreover, the encrypted data block's 1-bit mutation will result in points at infinity. Consequently, in the event of a single-bit mutation, H2 will not calculate points at infinity and the entire data block will produce null values.

4). Device Anonymity and Privacy

Once an attacker gains possession of a host controller, they attempt to retrieve data from a certain host. Even with complete authorization via H1 or H2, the impostor will not be able to extract the data or status of other devices since the hosts have a very particular communication pattern. Because of the way the plan is set up, H1 will only ever meet H2 once while H1 and H2 will need to mutually re-authenticate once the session terminates. Additionally, without knowledge of H1 and H2's private keys, the network administrator is unable to decrypt specific transmission data blocks. All hosts in the network have high privacy and anonymity as a result.

5). Dos / Replay Attack

The compromised host must first be connected to the network by the attacker to use DoS or replay assaults. To compute the precise amount for each devoted session, an attacker must also know a set of parameters such as the network, public and private keys. This is a challenging and computationally demanding procedure. As a result, H1 or H2 will compute curve points at infinity, preventing the ECDH process from continuing. Replay attacks work similarly, with each step’s message pattern changing. The session will stop, and the message will be refused if the attacker repeatedly transmits the same message patterns since the relevant device will calculate curve points at infinity.

V CONCLUSION

This study introduced SLECP-SDN, a novel protocol designed to secure communication between SDN's data and control planes using the Elliptic Curve25519 algorithm. The protocol integrates /lightweight encryption and robust authentication to effectively address vulnerabilities in the SDN Southbound Interface (SBI). The results demonstrate the efficacy of SLECP-SDN in achieving high throughput, energy efficiency, and enhanced resistance to security threats, such as impersonation, replay, man-in-the-middle, device anonymity and packet injection attacks. By utilizing pre-computed curve points and optimizing cryptographic operations, the protocol offers a practical solution that balances security and performance, making it highly suitable for resource-constrained environments. The implementation of Curve25519 establishes a new benchmark for securing SDN environments, ensuring data confidentiality, integrity, and availability without compromising network efficiency.

ACKNOWLEDGEMENT

We appreciate Universiti Putra Malaysia (UPM) for providing an enabling environment for conducting the research work. In addition, O. I. Aladesote would like to thank the Management of Federal Polytechnic, Ile Oluji, Ondo State, Nigeria for their support in pursuing his postgraduate studies.

REFERENCES

[1] Z. Latif, K. Sharif, F. Li, M. M. Karim, S. Biswas, and Y. Wang, “A comprehensive survey of interface protocols for software defined networks,” *J. Netw. Comput. Appl.*, vol. 156, no. July 2019, p. 102563, 2020, doi: 10.1016/j.jnca.2020.102563.

[2] T. Bakhshi, “State of the art and recent research advances in software defined networking,” *Wirel. Commun. Mob. Comput.*, vol. 2017, 2017, doi: 10.1155/2017/7191647.

[3] A. Mohamed *et al.*, “Software-defined networks for resource allocation in cloud computing: A survey,” *Comput. Networks*, vol. 195, no. December 2020, p. 108151, 2021, doi: 10.1016/j.comnet.2021.108151.

[4] S. Khorsandroo, A. G. Sánchez, A. S. Tosun, J. M. Arco, and R. Doriguzzi-Corin, “Hybrid SDN evolution: A comprehensive survey of the state-of-the-art,” *Comput. Networks*, vol. 192, p. 107981, 2021, doi: 10.1016/j.comnet.2021.107981.

[5] M. H. Rehmani, A. Davy, B. Jennings, and C. Assi, “Software Defined Networks-Based Smart Grid Communication: A Comprehensive Survey,” *IEEE Commun. Surv. Tutorials*, vol. 21, no. 3, pp. 2637–2670, 2019, doi: 10.1109/COMST.2019.2908266.

[6] J. H. Cox *et al.*, “Advancing software-defined networks: A survey,” *IEEE Access*, vol. 5, pp. 25487–25526, 2017, doi: 10.1109/ACCESS.2017.2762291.

[7] O. I. Aladesote and A. Abdullah, “Efficient and Secure Topology Discovery in SDN: Review,” *Lect. Notes Data Eng. Commun. Technol.*, vol. 127, pp. 397–412, 2022, doi: 10.1007/978-3-030-98741-1_33.

[8] Z. Latif, K. Sharif, F. Li, M. M. Karim, S. Biswas, and Y. Wang, “A comprehensive survey of interface protocols for software defined networks,” *J. Netw. Comput. Appl.*, vol. 156, no. February, p. 102563, 2020, doi: 10.1016/j.jnca.2020.102563

[9] S. Ahmad and A. Hussain Mir, “SDN Interfaces: Protocols, Taxonomy and Challenges,” *Int. J. Wirel. Microw. Technol.*, vol. 12, no. 2, pp. 11–32, 2022, doi: 10.5815/ijwmt.2022.02.02.

[10] D. Comer and A. Rastegarnia, “Toward Disaggregating the SDN Control Plane,” *IEEE Commun. Mag.*, vol. 57, no. 10, pp. 70–75, 2019, doi: 10.1109/MCOM.001.1900063.

[11] S. Algarni, F. Eassa, K. Almarhabi, A. Algarni, and A. Albeshri, “BCNBI: A Blockchain-Based Security Framework for Northbound Interface in Software-Defined Networking,” *Electron.*, vol. 11, no. 7, pp. 1–27, 2022, doi: 10.3390/electronics11070996.

[12] A. H. Abdi *et al.*, “Security Control and Data Planes of SDN: A Comprehensive Review of Traditional, AI, and MTD Approaches to Security Solutions,” *IEEE Access*, vol. 12, no. May, pp. 69941–69980, 2024, doi: 10.1109/ACCESS.2024.3393548.

[13] M. S. Farooq, S. Riaz, and A. Alvi, “Security and Privacy Issues in Software-Defined Networking (SDN): A Systematic Literature Review,” *Electron.*, vol. 12, no. 14, 2023, doi: 10.3390/electronics12143077.

[14] M. Iqbal, F. Iqbal, F. Mohsin, M. Rizwan, and F. Ahmad, “Security issues in software defined networking (SDN): Risks, challenges and potential solutions,” *Int. J. Adv. Comput. Sci. Appl.*, vol. 10, no. 10, pp. 298–303, 2019, doi: 10.14569/ijacsa.2019.0101042.

[15] T. W. Chao *et al.*, “Securing data planes in software-defined networks,” *IEEE NETSOFT 2016 - 2016 IEEE NetSoft Conf. Work. Software-Defined Infrastruct. Networks, Clouds, IoT Serv.*, pp. 465–

470, 2016, doi: 10.1109/NETSOFT.2016.7502486.

[16] S. Ghaly and M. Z. Abdullah, "Design and implementation of a secured SDN system based on hybrid encrypted algorithms," *Telkomnika (Telecommunication Comput. Electron. Control.*, vol. 19, no. 4, pp. 1118–1125, 2021, doi: 10.12928/TELKOMNIKA.v19i4.18721.

[17] Y. Alemami, A. M. Al-Ghonmein, K. G. Al-Moghrabi, and M. A. Mohamed, "Cloud data security and various cryptographic algorithms," *Int. J. Electr. Comput. Eng.*, vol. 13, no. 2, pp. 1867–1879, 2023, doi: 10.11591/ijece.v13i2.pp1867-1879.

[18] V. Varadharajan and U. Tupakula, "Counteracting Attacks from Malicious End Hosts in Software Defined Networks," *IEEE Trans. Netw. Serv. Manag.*, vol. 17, no. 1, pp. 160–174, 2020, doi: 10.1109/TNSM.2019.2931294.

[19] H. Al-Hamdani and W. S. Bhaya, "A Proposed Cryptography Key Management in Software-Defined Networking (SDN)," *6th Iraqi Int. Conf. Eng. Technol. its Appl. IICETA 2023*, pp. 22–28, 2023, doi: 10.1109/IICETA57613.2023.10351402.

[20] N. Gray, T. Zinner, and P. Tran-Gia, "Enhancing SDN security by device fingerprinting," *Proc. IM 2017 - 2017 IFIP/IEEE Int. Symp. Integr. Netw. Serv. Manag.*, pp. 879–880, 2017, doi: 10.23919/INM.2017.7987393.

[21] A. Ranjbar, M. Komu, P. Salmela, and T. Aura, "An SDN-based approach to enhance the end-to-end security: SSL/TLS case study," *Proc. NOMS 2016 - 2016 IEEE/IFIP Netw. Oper. Manag. Symp.*, no. July 2017, pp. 281–288, 2016, doi: 10.1109/NOMS.2016.7502823.

[22] B. Yigit, G. Gur, B. Tellenbach, and F. Alagoz, "Secured Communication Channels in Software-Defined Networks," *IEEE Commun. Mag.*, vol. 57, no. 10, pp. 63–69, 2019, doi: 10.1109/MCOM.001.1900060.

[23] Y. Peng, C. Wu, B. Zhao, W. Yu, B. Liu, and S. Qiao, "QKDFlow: QKD based secure communication towards the openflow interface in SDN," *Commun. Comput. Inf. Sci.*, vol. 699, pp. 410–415, 2017, doi: 10.1007/978-981-10-3969-0_45

[24] T. Adhikari, M. Kule, and A. K. Khan, "An ECDH and AES Based Encryption Approach for Prevention of MiTM in SDN Southbound Communication Interface," *2022 13th Int. Conf. Comput. Commun. Netw. Technol. ICCCNT 2022*, pp. 1–5, 2022, doi: 10.1109/ICCCNT54827.2022.9984509.

[25] D. S. Sahana and S. H. Brahmananda, "Secure Authentication Framework for SDN-IoT network using Keccak-256 and Bliss-B algorithms," *International Journal of Information Technology (Singapore)*, vol. 15, no. 1. pp. 335–344, 2023. doi: 10.1007/s41870-022-01074-w.

[26] J. W. Kang, S. H. Park, and J. You, "Mynah: Enabling lightweight data plane authentication for SDN controllers," *Proc. - Int. Conf. Comput. Commun. Networks, ICCCN*, vol. 2015-Octob, pp. 1–6, 2015, doi: 10.1109/ICCCN.2015.7288433.

[27] D. J. Bernstein, "Curve25519 : new Diffie-Hellman speed records," vol. 25519, 2006.

[28] P. L. Montgomery, "Speeding the Pollard and Elliptic Curve Methods of Factorization," *Math. Comput.*, vol. 48, no. 177, p. 243, 1987, doi: 10.2307/2007888.