

I. INTRODUCTION

Network security is a vital domain focused on protecting the integrity, confidentiality, and availability of data within a networked environment. It involves implementing various technologies, processes, and policies to prevent unauthorized access, misuse, or attacks on a network. Core components include firewalls, which act as the first line of defense by filtering incoming and outgoing traffic, and intrusion detection/prevention systems (IDS/IPS) that monitor for suspicious activities and mitigate threats in real time. Access control mechanisms, such as authentication (e.g., passwords, biometrics, and multi-factor authentication) and authorization, ensure that only authorized users can access specific resources or data. By controlling and managing permissions, these systems prevent attackers from gaining unnecessary access to sensitive areas.

Network security is a critical aspect of safeguarding the integrity, confidentiality, and availability of data in modern digital infrastructures. As organizations and individuals increasingly rely on interconnected networks for communication, business operations, and daily activities, the risks associated with data breaches, unauthorized access, and cyberattacks continue to grow. At its core, network security involves implementing a combination of hardware, software, policies, and procedures to prevent and monitor unauthorized access, misuse, or destruction of networked resources. The dynamic nature of cybersecurity threats makes it an ongoing challenge, requiring constant adaptation and vigilance. One of the foundational principles of network security is the concept of access control. By restricting who can access network resources and enforcing authentication mechanisms, network administrators can prevent unauthorized users from infiltrating the system. Authentication methods, such as passwords, biometric scans, or two-factor authentication, help verify the identity of users, ensuring that only authorized personnel can interact with sensitive data. Beyond authentication, encryption techniques are employed to protect data as it traverses the network, rendering it unintelligible to unauthorized interceptors. Encryption is a crucial line of defense, particularly when sensitive information is transmitted over public or unsecured networks. Intrusion detection and prevention systems (IDPS) also play a vital role in network security by actively monitoring network traffic for unusual or malicious activity. These systems analyze patterns in data flow to identify anomalies that may signal an attempted attack, such as malware propagation, denial-of-service attacks, or phishing attempts. Once detected, the system can respond by blocking traffic, alerting administrators, or automatically deploying countermeasures. The effectiveness of such systems hinges on the ability to differentiate between normal and abnormal behavior in real time, a task made more complex by the increasing sophistication of cyberattacks. Firewalls, another essential tool in network security, act as barriers between trusted internal networks and untrusted external networks, such as the internet. Firewalls filter incoming and outgoing traffic based on predetermined security rules, allowing or denying

access based on established policies. This helps to minimize the risk of external threats penetrating the network. However, as network architectures evolve, especially with the rise of cloud computing and distributed systems, traditional firewalls must be supplemented with advanced solutions that provide greater visibility and control over network traffic across multiple environments.

In recent years, the rise of cybercrime has led to a growing emphasis on the importance of regular security audits, vulnerability assessments, and penetration testing. These proactive measures help identify potential weaknesses in a network's defenses before they can be exploited by malicious actors. Security professionals simulate attacks on the network, testing its resilience against various threat scenarios. Such evaluations are critical in keeping security measures up-to-date, particularly as new vulnerabilities are discovered and as networks expand or evolve. Human error remains one of the most significant vulnerabilities in network security. Despite the most advanced technical defenses, employees can inadvertently compromise network security by falling victim to social engineering attacks, using weak passwords, or mishandling sensitive information. Therefore, user education and awareness training are vital components of a comprehensive security strategy. By fostering a culture of security consciousness, organizations can empower employees to recognize potential threats and adopt safer practices when interacting with networked systems. As the threat landscape continues to evolve, network security strategies are shifting towards a more holistic approach, encompassing not only technological defenses but also behavioral and procedural safeguards. Security frameworks such as Zero Trust, which operates under the assumption that no one inside or outside the network can be trusted by default, are gaining traction. This paradigm shift requires continuous verification of all users and devices seeking access to resources, significantly reducing the attack surface and limiting the potential impact of a breach. By adopting such comprehensive approaches, organizations can enhance their resilience against the ever-growing spectrum of cyber threats. In addition to external defenses, network security encompasses internal strategies like network segmentation, which divides a network into smaller, isolated segments to limit the spread of potential attacks. Regular monitoring and auditing help identify vulnerabilities, while patch management ensures that software and systems remain up to date against emerging threats. Security policies and procedures, including employee training and incident response planning, are also crucial for defending against human error and social engineering tactics like phishing. Moreover, compliance with industry standards and regulations, such as GDPR or HIPAA, is essential for maintaining trust and ensuring that network practices meet legal requirements. Ultimately, network security is an ongoing, dynamic process that requires constant adaptation to the evolving landscape of cyber threats.

II. PROBLEM STATEMENT

In today's digital world, where sensitive data is frequently transmitted over public networks, ensuring the privacy and security of communication has become increasingly important. Public networks are inherently insecure, leaving personal and corporate data vulnerable to interception, tampering, or unauthorized access. A Virtual Private Network (VPN) addresses this issue by creating an encrypted, secure connection between the user and the internet or private networks, protecting data during transmission. The challenge is to develop a VPN solution that provides a secure and reliable method for users to access remote resources and communicate over public networks without compromising performance or ease of use. The VPN should offer confidentiality, integrity, and authentication to ensure the safe exchange of data.

III. EXISTING SYSTEM

Virtual Private Networks (VPNs) have become essential tools in today's digital landscape, providing secure communication channels over the internet. As organizations increasingly adopt remote work policies and expand their operations across multiple locations, the need for reliable and secure connectivity has never been greater. VPNs enable users to establish encrypted connections, ensuring data privacy and security while facilitating access to internal networks. By allowing users to connect securely to remote resources, VPNs play a critical role in safeguarding sensitive information from unauthorized access and cyber threats. There are various types of VPNs, each designed to meet specific connectivity and security needs, including site-to-site VPNs and remote access VPNs. Additionally, the implementation of different protocols, such as OpenVPN, IPsec, and WireGuard, further enhances the capabilities and security of VPN solutions.

Fig 3.1: Site-to-site VPNs are designed to connect entire networks, typically used by organizations with multiple geographic locations. This system allows different offices or branches to communicate securely over the internet, functioning as a single unified network. In a site-to-site VPN setup, routers or dedicated VPN appliances at each location establish a secure tunnel, enabling encrypted data transmission between the connected sites. There are two main types of site-to-site VPNs: intranet-based, which connects various local area networks (LANs) within the same organization, and extranet-based, which connects a company's LAN to the networks of external partners or clients. The primary advantage of site-to-site VPNs is that they facilitate secure data exchange and resource sharing between multiple locations, thereby enhancing collaboration and operational efficiency.

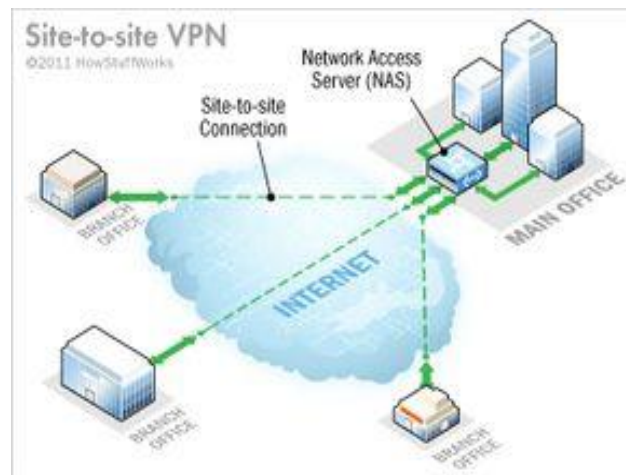


Fig 3.1 Site-to-site Virtual Private Network

Fig 3.2: Remote access VPNs provide individuals with secure connections to a private network from remote locations, which is especially beneficial for employees working from home or traveling. This type of VPN allows users to access internal company resources, such as files, applications, and databases, as if they were physically present in the office. Remote access VPNs typically involve the installation of client software on the user's device, which establishes an encrypted tunnel to the VPN server located at the corporate network. This encryption ensures that data transmitted between the user's device and the corporate network remains secure from interception. One of the key benefits of remote access VPNs is their flexibility, as users can connect from various devices, including laptops, smartphones, and tablets, without compromising security. Additionally, remote access VPNs often employ multi-factor authentication to enhance security further, ensuring that only authorized users can access sensitive corporate information.

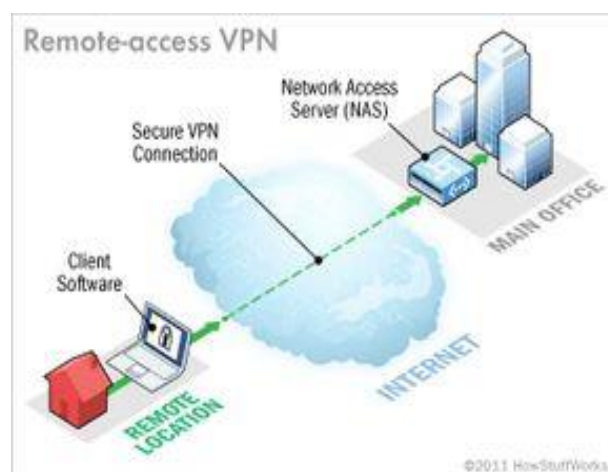


Fig 3.2 Remote access Virtual Private Network

OpenVPN is an open-source VPN protocol that has gained popularity due to its flexibility, security, and ease of deployment. Unlike some proprietary protocols, OpenVPN can be configured to work through various network configurations, including NAT and firewalls, making it suitable for a wide range of environments. It uses SSL/TLS for key exchange, providing strong encryption to protect data during transmission. OpenVPN supports multiple encryption algorithms, allowing users to choose the level of security that best fits their needs. One of the standout features of OpenVPN is its ability to be deployed in both site-to-site and remote access scenarios, making it a versatile choice for organizations. Additionally, its open-source nature means that it is continually being improved and audited by the community, ensuring that any vulnerabilities are quickly identified and addressed.

IPsec (Internet Protocol Security) is a comprehensive suite of protocols designed to secure internet communications at the IP layer. It operates by authenticating and encrypting each IP packet in a communication session, providing confidentiality, integrity, and authenticity. IPsec can be used in two primary modes: transport mode, where only the data packet is encrypted, and tunnel mode, where the entire IP packet, including the header, is encrypted. This flexibility makes IPsec suitable for both site-to-site and remote access VPN implementations. One of the key strengths of IPsec is its support for various encryption and hashing algorithms, allowing organizations to choose the level of security that meets their specific requirements. IPsec is widely adopted for its strong security features, but its complexity can pose challenges in terms of configuration and management, requiring expertise to implement effectively. Despite these challenges, IPsec remains a dominant choice for securing network communications, particularly in enterprise environments.

WireGuard is a modern VPN protocol that has gained significant traction due to its simplicity, speed, and strong security features. Unlike traditional VPN protocols, which often have complex configurations and large codebases, WireGuard is designed to be lightweight and easy to set up, with a small codebase that enhances its security by reducing the potential attack surface. WireGuard utilizes state-of-the-art cryptography, including protocols like ChaCha20 for encryption and Poly1305 for message authentication, to provide a high level of security. One of the key benefits of WireGuard is its performance; it has been shown to deliver faster connection speeds and lower latency compared to older protocols like OpenVPN and IPsec. WireGuard also supports mobile users seamlessly, maintaining stable connections as users switch between networks, such as from Wi-Fi to cellular. This combination of simplicity, performance, and security has made WireGuard an appealing choice for both individuals seeking privacy online and organizations looking for efficient VPN solutions.

IV. PROPOSED SYSTEM

The proposed VPN system is designed to offer an educational experience by enabling learners to create and understand a secure communication channel using advanced encryption techniques. This VPN implementation uses the Advanced Encryption Standard (AES) for data encryption and Elliptic Curve Diffie-Hellman (ECDH) for secure key exchange. The project provides a hands-on approach to understanding network security, giving learners an opportunity to explore how these encryption methodologies work together to protect transmitted data. By constructing this system from scratch, students can delve deeply into the underlying technical aspects of VPNs, such as the complexities of encryption, the importance of secure key exchanges, and the broader impact of these processes on data transmission security. At the heart of the VPN system is AES-256, a symmetric encryption algorithm known for its robustness and security. This algorithm encrypts all the data transmitted between the client and the server, ensuring that it cannot be intercepted or understood by unauthorized parties. AES-256, which utilizes a 256-bit key, is widely regarded as one of the most secure encryption standards, providing a strong level of protection for sensitive data. The project's focus on AES-256 demonstrates the practical implementation of a highly secure algorithm, which is commonly used in both commercial and governmental applications to safeguard communications. This gives learners direct exposure to real-world encryption methods that they will likely encounter in future cybersecurity projects or careers.

To complement AES, the system uses Elliptic Curve Diffie-Hellman (ECDH) for secure key exchange. ECDH ensures that the client and server can share a cryptographic key over an insecure channel without the risk of interception. By using elliptic curve cryptography, the key exchange process is both secure and efficient, reducing the computational overhead while maintaining a high level of security. One of the main advantages of ECDH is that the actual cryptographic key is never transmitted over the network. Instead, each party generates a shared secret independently, which is then used to encrypt and decrypt the transmitted data. This method significantly reduces the risk of key interception, as there is no point during transmission where the key itself is vulnerable to eavesdropping. The workflow of this VPN project begins with both the client and server initializing their respective environments. The client initiates the connection by generating its public and private keys using the ECDH algorithm. Simultaneously, the server generates its own key pair. Once both keys are generated, the client sends its public key to the server, and the server sends its public key to the client. At this point, both the client and server use the received public keys along with their private keys to generate a shared secret, which will be used as the key for AES encryption. This secure key exchange ensures that even if an attacker intercepts the public keys, they cannot derive the shared secret needed to decrypt the communication.

Fig 4.1: Once the shared key has been established, the client encrypts its data using AES-256. The encrypted data is then transmitted to the server, ensuring that any sensitive information remains secure during transit. Upon receiving the encrypted data, the server decrypts it using the same shared key. The server processes the request—such as fetching a webpage or handling other network protocols—and encrypts the response using the same AES key. This encrypted response is then sent back to the client, which decrypts it and processes the information as needed. This process of encrypting and decrypting data continues throughout the communication, ensuring that all transmitted information is securely protected against unauthorized access. The proposed VPN system offers significant educational value by enabling learners to observe and understand each stage of encryption and key exchange. Unlike commercial VPN services that abstract these processes, this project provides full visibility into how data is encrypted, how keys are exchanged, and how secure communication channels are maintained. This transparency allows learners to explore the practical challenges of implementing encryption algorithms and gain a deeper understanding of the complexities involved in securing data over the internet. Additionally, this hands-on experience with AES and ECDH enables learners to comprehend how encryption protects data and how key exchange protocols prevent unauthorized access to cryptographic keys.

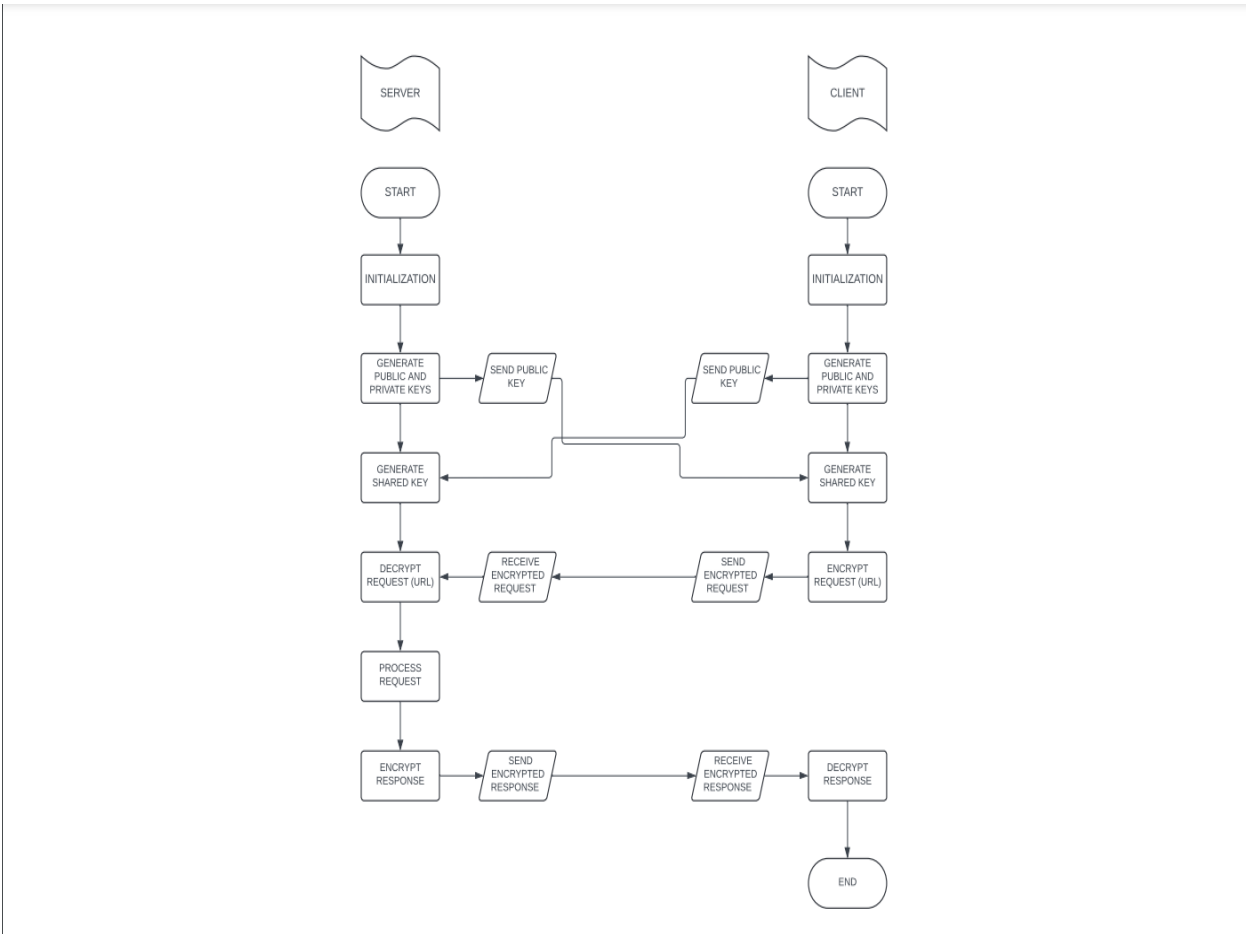


Fig 4.1 Flowchart of the proposed Virtual Private Network System

The system is also designed with flexibility in mind, allowing learners to experiment with different encryption techniques or key exchange protocols. For instance, they could replace AES-256 with ChaCha20 or explore the implications of post-quantum cryptographic methods. This encourages further experimentation and customization, offering an opportunity for learners to expand their understanding of cryptographic techniques and how they can be adapted to meet emerging security threats. Furthermore, the project provides insight into how encryption affects network performance and latency, as learners can simulate real-world traffic and explore how encryption impacts communication in high-traffic environments. By building this VPN from scratch, learners will not only gain practical skills in encryption and key exchange but also develop critical thinking skills related to security vulnerabilities and system auditing. They will be prompted to consider potential weaknesses in their implementation, such as weak key management or incorrect configuration, and develop solutions to mitigate these risks. This practical experience is essential for anyone interested in cybersecurity, as it fosters a deep understanding of how encryption works, how to secure communications, and how to proactively address potential vulnerabilities.

V. ALGORITHMS

a) Elliptic Curve Diffie-Hellman (ECDH) Key Exchange Algorithm

Elliptic Curve Diffie-Hellman (ECDH) is a widely used public-key cryptographic protocol that enables two parties to securely establish a shared secret key over an insecure communication channel. The core idea behind ECDH is based on the mathematical properties of elliptic curves, which allow for efficient computation of key pairs. Each party generates its own private key as a randomly chosen integer and computes its corresponding public key by performing scalar multiplication of a predefined base point on the elliptic curve. The exchanged public keys can then be used to compute the shared secret, which serves as the basis for secure communication.

One of the primary benefits of ECDH is its ability to provide a high level of security with relatively small key sizes. For instance, a 256-bit key in ECDH offers a security level comparable to a 3072-bit RSA key. This reduced key size translates to faster computations and lower bandwidth usage, making ECDH particularly well-suited for environments with limited resources, such as mobile devices and IoT applications. The efficiency of ECDH operations allows for quicker establishment of secure connections, enhancing user experience in applications like secure web browsing and encrypted messaging. Additionally, ECDH is designed with resistance to certain types of cryptographic attacks, including those potentially facilitated by quantum computing. While not entirely quantum-proof, elliptic curve cryptography is considered more resilient against specific algorithms that could threaten traditional public-key systems. The widespread support for ECDH in various protocols, such as SSL/TLS and secure

messaging applications, further underscores its significance in contemporary cryptographic practices, ensuring secure communication across diverse platforms and applications.

Mathematical Foundations of ECDH

i) Elliptic Curve Equation

The elliptic curve is defined by the Weierstrass equation:

$$y^2 = (x^3 + ax + b) \bmod p$$

Where:

- a and b are curve parameters.
- p is a prime number that defines the finite field.
- x and y are the coordinates of points on the curve.

ii) Base Point (G)

The base point (G) is a predefined point on the elliptic curve, specified as:

$$G = (x_G, y_G)$$

iii) Private and Public Key Generation

1. Private Key Generation: Each party (Party A and Party B) generates a private key as a random integer (d) in the range $[1, n-1]$, where (n) is the order of the elliptic curve.

2. Public Key Calculation: The public key is derived from the private key using elliptic curve point multiplication:

$$P = d \cdot G$$

Where:

- (P) is the public key.
- (d) is the private key.

iv) Point Multiplication

The core operation in ECDH is point multiplication, which involves multiplying a point (G) (the base point) by a scalar (d) (the private key). This operation can be efficiently computed using the Double and Add method.

1. Point Doubling

If $P = (x_P, y_P)$ is a point on the elliptic curve, the point doubling operation is defined as:

$$\begin{aligned} 2P &= (x', y') \text{ where} \\ \lambda &= \{(3x_P^2 + a) / (2y_P)\} \bmod p \\ x' &= (\lambda^2 - 2x_P) \bmod p \\ y' &= \{\lambda(x_P - x') - y_P\} \bmod p \end{aligned}$$

Where λ is the slope of the tangent at point (P).

2. Point Addition

If you have two distinct points $P = (x_P, y_P)$ and $Q = (x_Q, y_Q)$, their sum $R = P + Q = (x_R, y_R)$ is calculated as follows:

$$\begin{aligned} \lambda &= \{(y_Q - y_P) / (x_Q - x_P)\} \bmod p \\ x_R &= (\lambda^2 - x_P - x_Q) \bmod p \\ y_R &= \{\lambda(x_P - x_R) - y_P\} \bmod p \end{aligned}$$

Pseudocode for ECDH Key Generation and Exchange

Function GenerateBasePoint():

a = [curve parameter a]

b = [curve parameter b]

p = [prime number p]

x_G = [x-coordinate of G]

y_G = [y-coordinate of G]

Return (x_G, y_G)

Function PointDoubling(P):

```
(x_P, y_P) = P
lambda = (3 * x_P^2 + a) * (2 * y_P)^(-1) mod p
x_R = (lambda^2 - 2 * x_P) mod p
y_R = (lambda * (x_P - x_R) - y_P) mod p
Return (x_R, y_R)
```

Function PointAddition(P, Q):

```
(x_P, y_P) = P
(x_Q, y_Q) = Q
lambda = (y_Q - y_P) * (x_Q - x_P)^(-1) mod p
x_R = (lambda^2 - x_P - x_Q) mod p
y_R = (lambda * (x_P - x_R) - y_P) mod p
Return (x_R, y_R)
```

Function PointMultiplication(d, G):

```
R = (0, 0)
N = G
while d > 0:
    if (d % 2) == 1:
        R = PointAddition(R, N)
    N = PointDoubling(N)
    d = d // 2
Return R
```

Function ECDH_KeyGeneration():

```
(x_G, y_G) = GenerateBasePoint()

d_A = GenerateRandomPrivateKey()
d_B = GenerateRandomPrivateKey()

P_A = PointMultiplication(d_A, G)
```

```
P_B = PointMultiplication(d_B, G)
```

```
Return (d_A, P_A, d_B, P_B)
```

```
Function ECDH_KeyExchange(privateKey, publicKey):
```

```
    SharedSecret = PointMultiplication(privateKey, publicKey)
```

```
    Return SharedSecret
```

```
Function Main():
```

```
    (d_A, P_A, d_B, P_B) = ECDH_KeyGeneration()
```

```
    SharedSecretA = ECDH_KeyExchange(d_A, P_B)
```

```
    SharedSecretB = ECDH_KeyExchange(d_B, P_A)
```

```
End Function
```

Explanation of the Functions

1. **GenerateBasePoint()**: This function defines the elliptic curve parameters and returns the base point (G).
2. **PointDoubling(P)**: This function performs point doubling on the point (P), returning the result of (2P).
3. **PointAddition(P, Q)**: This function takes two points (P) and (Q) on the curve and returns their sum ($R = P + Q$).
4. **PointMultiplication(d, G)**: This function computes the scalar multiplication ($d \cdot G$) using the Double and Add method, returning the resulting point (R).
5. **ECDH_KeyGeneration()**: This function generates private keys for both parties, computes their public keys using point multiplication, and returns the keys.
6. **ECDH_KeyExchange(privateKey, publicKey)**: This function computes the shared secret by multiplying the private key with the other party's public key.
7. **Main()**: This orchestrates the entire process, generating keys and computing shared secrets for both parties, printing the results.

b)AES-256-GCM Encryption Algorithm

Advanced Encryption Standard (AES) is a symmetric key encryption algorithm widely adopted for secure data transmission and storage. AES operates on fixed-size blocks of data (128 bits) and supports key sizes of 128, 192, or 256 bits. The "Galois/Counter Mode" (GCM) is a mode of operation for AES that provides both confidentiality and data integrity through the use of authentication tags. In the context of AES-256-GCM, the algorithm uses a 256-bit key for encryption, ensuring a high level of security that is critical for protecting sensitive information.

One of the primary benefits of AES-256-GCM is its efficiency. The algorithm is optimized for both hardware and software implementations, allowing for fast encryption and decryption processes. The GCM mode enhances performance by enabling parallel processing, making it suitable for high-throughput applications such as secure communications in cloud computing, virtual private networks (VPNs), and secure file storage. This efficiency is particularly advantageous in environments where resources are limited, such as mobile devices or Internet of Things (IoT) applications. Another significant advantage of AES-256-GCM is its robust security profile. With a 256-bit key size, it provides a level of security that is considered secure against all known practical attacks, including brute-force attacks. The GCM mode not only encrypts the data but also generates an authentication tag to ensure data integrity, which verifies that the data has not been tampered with during transmission. This dual function makes AES-256-GCM a popular choice for secure data transmission across various applications, ensuring that sensitive information remains confidential and intact.

Mathematical Foundations of AES-256-GCM

i)Galois Field Arithmetic

1. Galois Field $GF(2^8)$

AES operates over $GF(2^8)$, a field with 256 elements (0-255). Each byte is treated as an element of this field, enabling operations like addition and multiplication.

2. Addition in $GF(2^8)$

Addition is performed using the XOR operation:

$$a \oplus b = c$$

Where c is the result of adding a and b .

3. Multiplication in $GF(2^8)$

Multiplication is defined as:

$$a \cdot b = \text{reduce}(a \cdot b \bmod m(x))$$

Where $m(x) = x^8 + x^4 + x^3 + x + 1$ is the irreducible polynomial used in AES.

ii) AES Encryption Process

1. Key Expansion

The original key is expanded into multiple round keys using a key schedule that involves:

- RotWord: Circularly rotates a word (4 bytes).
- SubWord: Applies a substitution from a predefined S-box.
- XOR: Combines the result with the previous round key.

2. The AES Round Function

Each round consists of several transformations:

i) SubBytes

Each byte is substituted using the S-box:

$$S(b) = S\text{-box}[b]$$

ii). ShiftRows

Rows of the state are shifted left by varying offsets.

iii) MixColumns

Each column is mixed using a matrix multiplication over $GF(2^8)$:

$$\text{Output} = \text{Matrix} \cdot \text{Input}$$

The mixing matrix is:

[2 3 1 1]

[1 2 3 1]

[1 1 2 3]

[3 1 1 2]

iv) AddRoundKey

Each byte of the state is XORED with the corresponding byte of the round key:

$$\text{State}_i = \text{State}_i \oplus \text{RoundKey}_i$$

iii). AES Decryption Process

1.Initial AddRoundKey

The ciphertext undergoes an initial round where the round key is applied:

$$\text{State}_0 = \text{Ciphertext} \oplus \text{RoundKey}_N$$

Where N is the total number of rounds.

2.Round Function (Reversed) for Each Round

For each round i from N-1 down to 1:

i)AddRoundKey:

$$\text{State}_i = \text{State}_{i+1} \oplus \text{RoundKey}_i$$

ii) Inverse MixColumns(only for rounds 1 to N-1):

Each column is mixed using the inverse of the mixing matrix:

$$\text{Output} = \text{InverseMatrix} \cdot \text{Input}$$

The inverse mixing matrix is:

[14 11 13 9]

[9 14 11 13]

[13 9 14 11]

[11 13 9 14]

iii) Inverse ShiftRows:

Each row of the state is shifted back to the right.

iv) Inverse SubBytes:

Each byte is substituted back using the inverse S-box:

$$(S^{-1})b = S^{-1}[b]$$

3. Final AddRoundKey

The last round key is applied to produce the final plaintext:

$$\text{Plaintext} = \text{State}_1 \oplus \text{RoundKey}_0$$

iv)Galois/Counter Mode (GCM)

1.Counter Mode

GCM uses a counter value (IV combined with a nonce) to encrypt plaintext blocks:

$$C_i = P_i \oplus \text{AES}(K, \text{Counter}_i)$$

Where P_i is the plaintext block and C_i is the ciphertext block.

2.Authentication Tag Calculation

GCM provides integrity through an authentication tag computed as follows:

i).Compute the tag** from the ciphertext and additional authenticated data (AAD):

For AAD, compute:

$$H = \text{AES}(K, 0) \text{ (where } K \text{ is the AES key)}$$

For each block of AAD and ciphertext:

$$T = (T \oplus \text{AAD}_i) \cdot H$$

$$T = (T \oplus C_i) \cdot H$$

ii) Final Tag:

The final authentication tag is generated from T:

$$\text{Tag} = T \oplus \text{len_AAD} \oplus \text{len_C}$$

v)Decryption Process

The decryption is similar to encryption:

$$P_i = C_i \oplus \text{AES}(K, \text{Counter}_i)$$

Verification involves checking if the computed tag matches the received tag:

$$T_{\text{received}} = T_{\text{computed}}$$

Workflow of AES-256-GCM Encryption

1. Key Generation: A shared secret key is derived using an appropriate key exchange method, such as ECDH.
2. Nonce Generation: Generate a unique nonce for each encryption operation.
3. Encryption: Encrypt the plaintext using AES with the shared key, nonce, and GCM mode.
4. Authentication Tag Generation: An authentication tag is generated to ensure the integrity of the ciphertext.
5. Decryption: The ciphertext can be decrypted using the same nonce, key, and the authentication tag for verification.

Pseudocode for AES-256-GCM Encryption Algorithm

Function AES256_GCM_Encrypt(plaintext, sharedKey):

```
    nonce = GenerateUniqueNonce()
    cipher = AES_GCM(sharedKey, nonce)
    ciphertext = cipher.Encrypt(plaintext)
    authTag = cipher.GetAuthTag()
    Return (nonce, ciphertext, authTag)
```

Function AES256_GCM_Decrypt(nonce, ciphertext, authTag, sharedKey):

```
    cipher = AES_GCM(sharedKey, nonce)
    plaintext = cipher.Decrypt(ciphertext, authTag)
    Return plaintext
```

Function Main():

```
    sharedKey = ECDH_KeyExchange(privateKeyA, publicKeyB)
    plaintext = "Hello, World!"
    (nonce, ciphertext, authTag) = AES256_GCM_Encrypt(plaintext, sharedKey)
    decryptedText = AES256_GCM_Decrypt(nonce, ciphertext, authTag, sharedKey)
```

End Function

Explanation of the Functions

1. AES256_GCM_Encrypt():

- Generates a unique nonce and initializes the GCM mode with the shared key.
- Encrypts the plaintext and generates the authentication tag.
- Returns the nonce, ciphertext, and authentication tag.

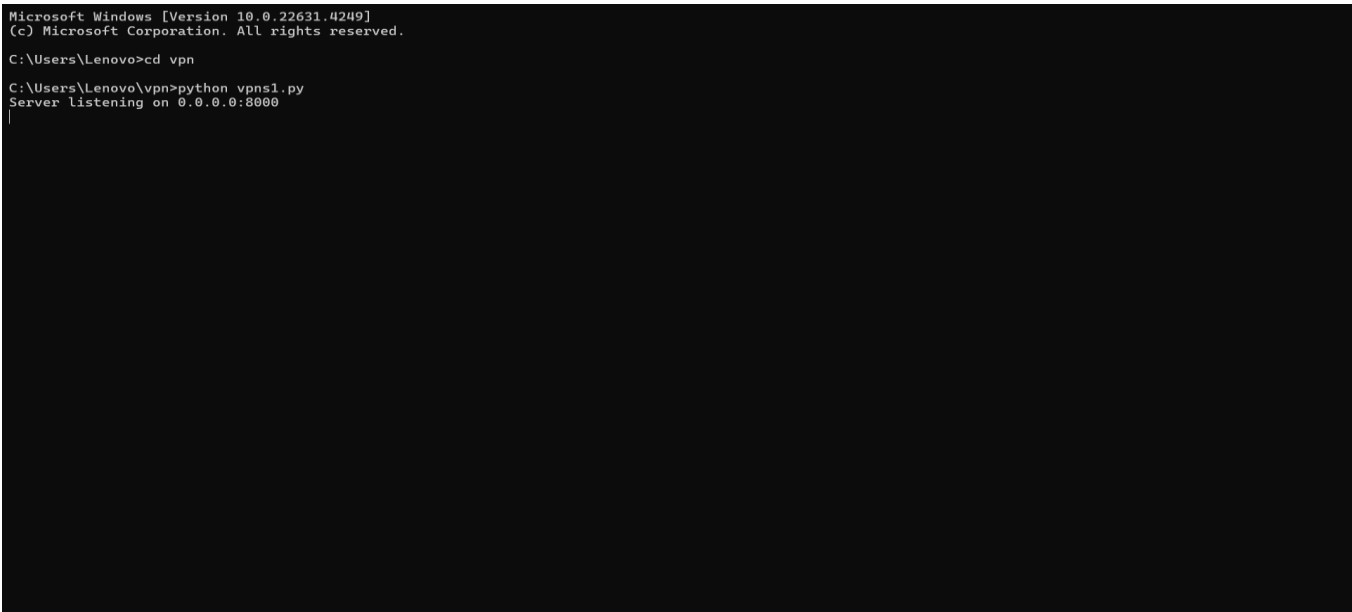
2. AES256_GCM_Decrypt():

- Initializes the GCM mode with the shared key and nonce.
- Verifies the authentication tag and decrypts the ciphertext to retrieve the original plaintext.

3. Main():

- Derives a shared key using ECDH key exchange.
- Encrypts and decrypts a sample plaintext.

VI RESULT ANALYSIS



```
Microsoft Windows [Version 10.0.22631.4249]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Lenovo>cd vpn
C:\Users\Lenovo\vpn>python vpns1.py
Server listening on 0.0.0.0:8000
```

Fig 6.1 Server Initialization and Listening for Connections

Fig 6.1: The system workflow begins when the server starts listening for incoming client connections on the specified IP address and port (0.0.0.0:8000). The server is now ready to accept connections and displays the message: "Server listening on 0.0.0.0:8000." This indicates that the server has been successfully initialized and is waiting for a client to connect. The IP 0.0.0.0 allows the server to accept connections from any client on the network.

```
Microsoft Windows [Version 10.0.22631.4249]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Lenovo>cd vpn

C:\Users\Lenovo\vpn>python vpns1.py
Server listening on 0.0.0.0:8000
Connection from ('192.168.137.1', 55901)
```

Fig 6.2 Client Connection Established

Fig 6.2: When a client initiates a connection, the server accepts the connection from the client and prints a message, displaying the IP address and port of the client, such as "Connection from ('192.168.137.1', 55901)." This confirms that the client has successfully connected to the server. The server and client are now engaged in an active session, allowing them to exchange messages and data. The server begins by sending its public key to the client for key exchange, allowing both parties to establish a secure communication channel.

```
Microsoft Windows [Version 10.0.22631.4249]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Lenovo>cd vpn

C:\Users\Lenovo\vpn>python vpnc8.py
Connected with server
Shared key: b61970cb81fb80d5ee27155da8720bab3a2b4d7810bde0177e67a66cad28fca9
```

Fig 6.3 Shared key generation

Fig 6.3: On the client side, after establishing the connection, the client prints "Connected with server," indicating that the client has successfully connected to the server. The client then generates a shared key based on the server's public key and its own private key, and this shared key is printed on the client's terminal, such as "Shared key: [hex value]." This shared key will be used for encrypting and decrypting messages between the client and server. The server also generates the exact same shared key based on the client's public key and its own private key

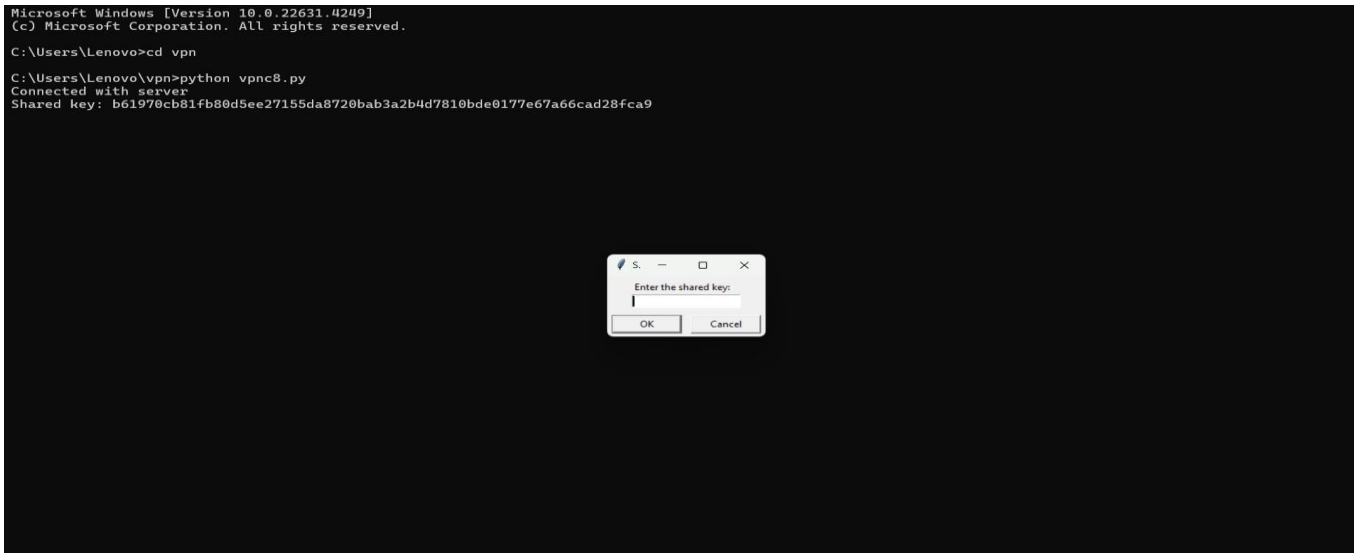


Fig 6.4 Shared Key Entry Dialog

Fig 6.4 : Prompts the user to enter the shared key through a dialog box that appears on the client's screen. This box is used to verify whether the key provided matches the one derived during the key exchange process.

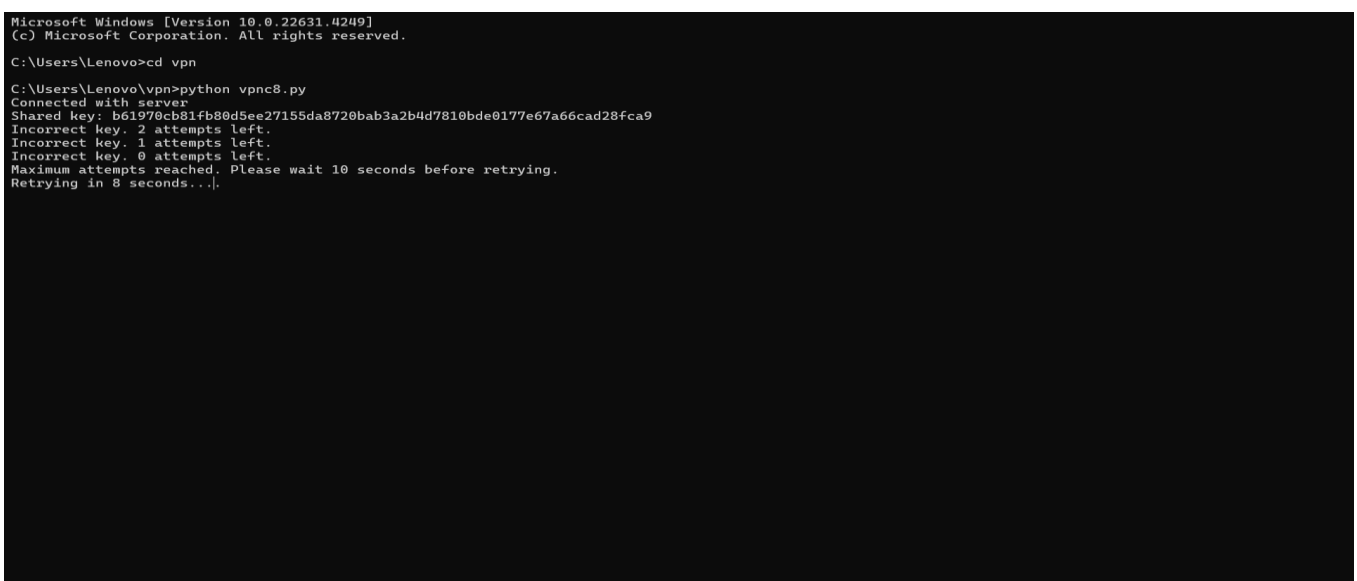


Fig 6.5 Incorrect Key Attempts and Countdown

Fig 6.5: If the client enters an incorrect shared key, the system will notify them with a message indicating that the key was wrong and the number of attempts left. Once the maximum number of attempts is reached, the client is temporarily blocked from entering the key again, triggering a 10-second countdown before retrying. During this countdown, a message is displayed in the terminal, counting down the time: "Retrying in [n] seconds...". This adds a security measure, preventing rapid brute-force attempts to guess the correct key.

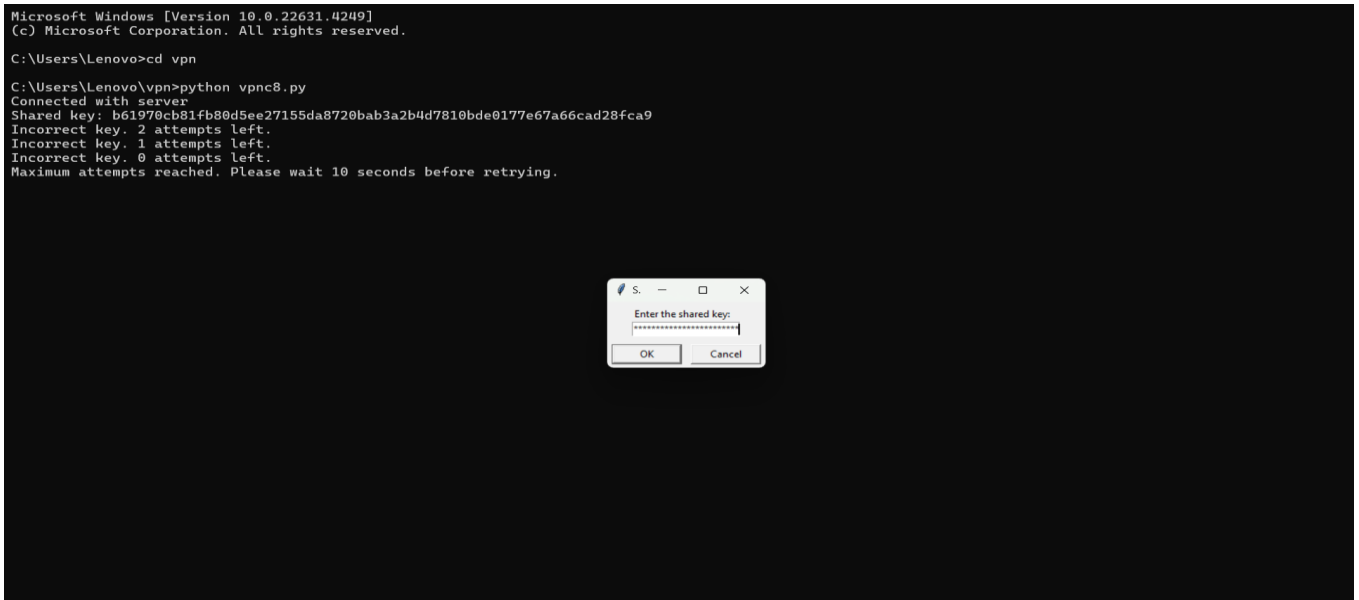


Fig 6.6 Correct Key Entry and Successful Key Exchange

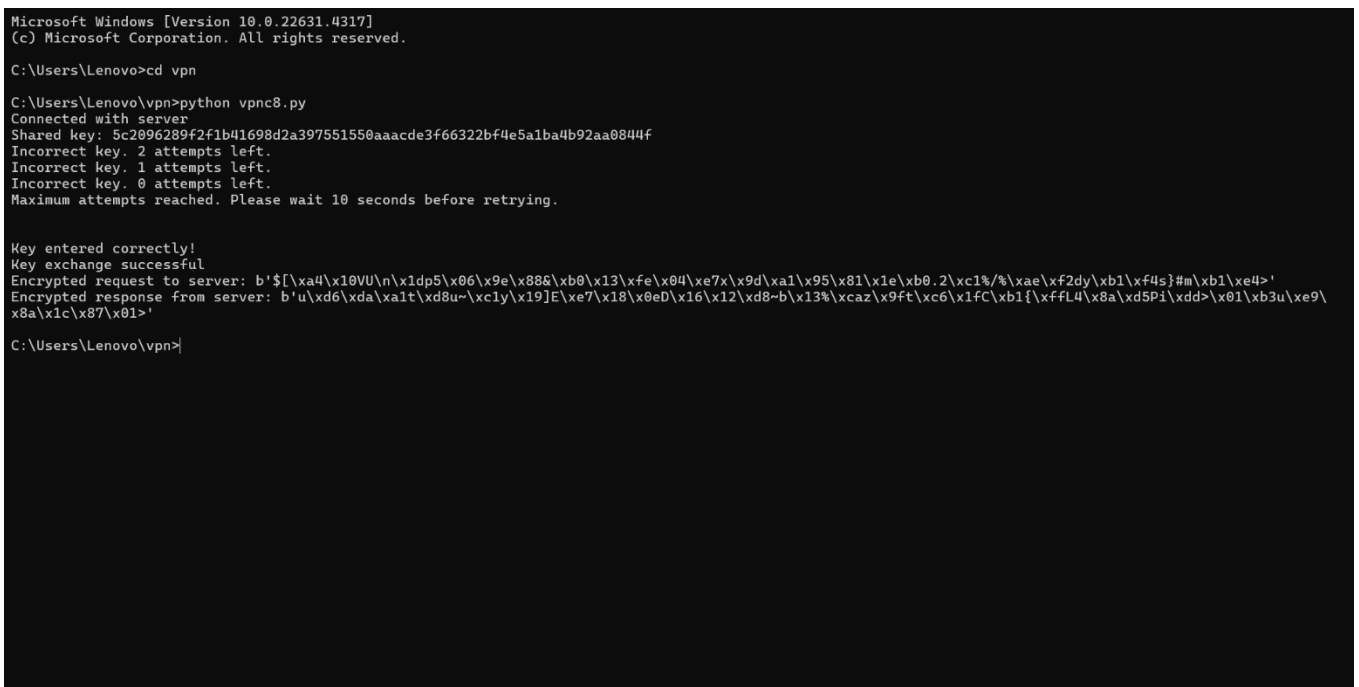


Fig 6.7 Client Sends Encrypted Request to Server

Fig 6.6 and Fig 6.7: After the countdown finishes, the client is presented with the shared key entry dialog box once again. The user is given the chance to re-enter the correct key. Once the correct key is entered, the client prints "Key entered correctly!" followed by "Key exchange successful," confirming that the key exchange process has been successfully completed. This allows the client to encrypt the request to the server, which is then printed as "Encrypted request to server: [encrypted data]." The encrypted request contains the URL that the client wishes to access, ensuring that the communication is protected from eavesdropping.

```
Microsoft Windows [Version 10.0.22631.4317]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Lenovo>cd vpn

C:\Users\Lenovo\vpn>python vpns1.py
Server listening on 0.0.0.0:8080
Connection from ('127.0.0.1', 64880)
Encrypted request from client: b'[\xa4\x10VU\n\x1dp5\x06\x9e\x88&\xb0\x13\xfe\x04\xe7\x9d\xa1\x95\x81\xe\x0.2\xc1%/%\xae\xf2dy\xb1\xf4s}#\xb1\xe4>'
Decrypted request from client: http://amazon.com
```

Fig 6.8 Server Receives and Processes Encrypted Request

Fig 6.8: At the server end, after receiving the encrypted request from the client, the server prints "Encrypted request from client: [encrypted data]," confirming that it has successfully received the encrypted message. The server decrypts the message using the shared key, retrieves the requested URL, and processes the request. The server then encrypts the response and sends it back to the client.

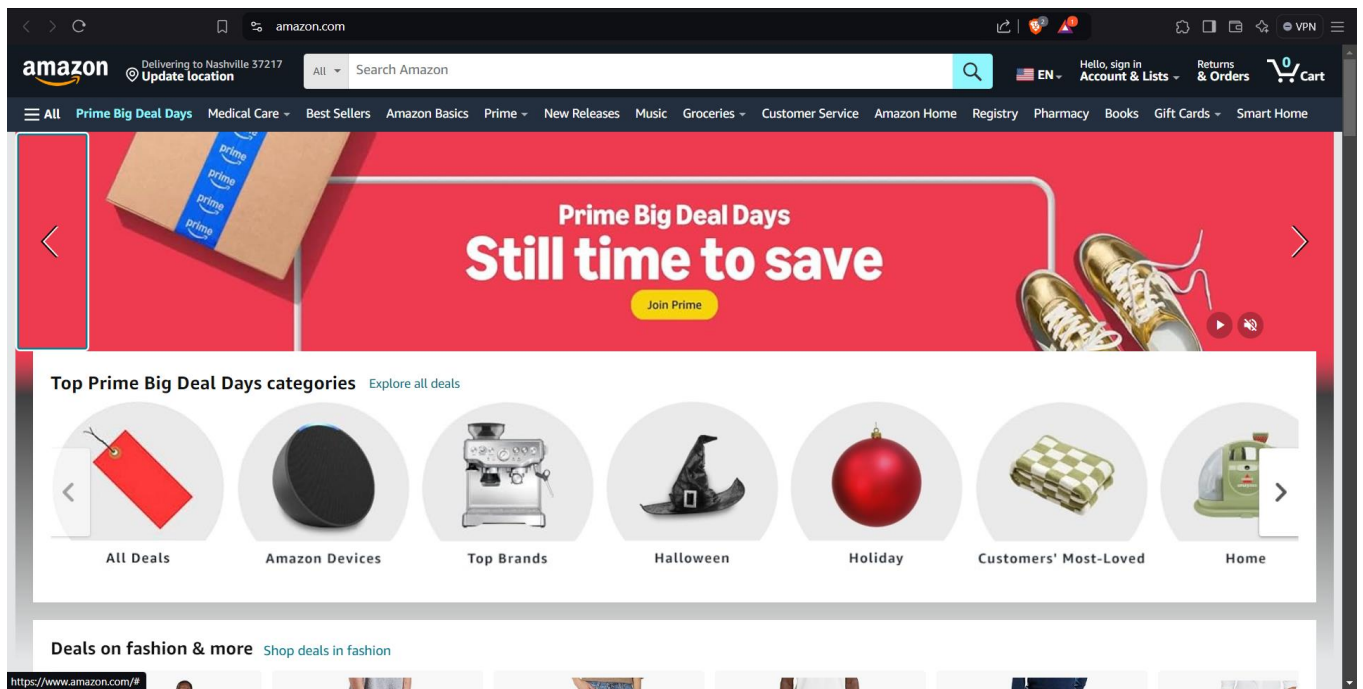


Fig 6.9 Client Receives Encrypted Response and Opens Webpage

Fig 6.9: Once the client receives the encrypted response from the server, it prints "Encrypted response from server: [encrypted data]." The client then decrypts the response using the shared key. If the decrypted response is valid, the client automatically processes the information. In this case, since the client requested the Amazon homepage, the client opens the actual amazon.com webpage after successful decryption. The process concludes with the client successfully displaying the requested webpage, confirming the successful end-to-end encryption and decryption workflow between the client and server.

VII CONCLUSION

In conclusion, this VPN project demonstrates the effective use of modern cryptographic techniques, such as AES-256-GCM encryption and Elliptic Curve Diffie-Hellman (ECDH) key exchange, to create a secure communication channel between a client and server. By implementing these robust protocols, the project ensures that data transmitted over the network is encrypted, protecting it from potential eavesdroppers and malicious actors. The system efficiently handles key exchange, encryption, decryption, and secure transmission of data, providing a reliable and secure framework for educational purposes. While designed for small-scale use and educational exploration, this VPN prototype showcases the essential principles of modern VPNs, emphasizing security, confidentiality, and integrity in data transmission.