AMERICAN JOURNAL OF AI CYBER COMPUTING MANAGEMENT
E-ISSN:3069-0102
VOL.5, NO. 4(2025)

SECURE COMMUNICATION USING DIFFIEHELLMAN KEY EXCHANGE, AES & SHAFOR DATA INTEGRITY

S.Vamshi Krushna[1], Sanga Rethika[2], Mosam Vaishnavi[3], Sidagam Jayanth[4], Menchu Sai Kumar[5] and Kamireddy Sai[6]

[1]*Assistant Professor, Department of CSE(DS), Samskruti College Of Engineering And Technology, Kondapur (V), Ghatkesar (M), Medchal (D), Hyderabad, India.*

[2,3,4,5]*Student, Department of CSE(DS), Samskruti College Of Engineering And Technology, Kondapur (V), Ghatkesar (M), Medchal (D), Hyderabad, India.*

Corresponding email ID: csevamshi@samskruti.ac.in

ABSTRACT:

The project aims to enhance the security of communication channels by implementing a robust cryptographic framework. Leveraging the Diffie-Hellman Key Exchange for secure key establishment, Advanced Encryption Standard (AES) for confidential data transmission, and Secure Hash Algorithm (SHA) for ensuring data integrity, this initiative seeks to establish a secure and reliable communication protocol. By combining these cryptographic components, the project endeavors to protect sensitive information from eavesdropping, tampering, and unauthorized access, ensuring the confidentiality and integrity of transmitted data. In the era of ubiquitous data exchange over insecure networks, ensuring the confidentiality and integrity of transmitted data is of paramount importance. This paper presents a comprehensive approach to secure communication by leveraging the Diffie-Hellman key exchange protocol for secure key establishment, Advanced Encryption Standard (AES) for data encryption, and Secure Hash Algorithm (SHA) for ensuring data integrity. The Diffie-Hellman key exchange protocol enables two parties to securely establish a shared secret key over a public channel, mitigating the risk of eaves dropping and man-in-the-middle attacks. The protocol's security relies on the discrete logarithm problem, making it computationally infeasible for adversaries to derive the shared key. Once the shared key is established, AES, a symmetric block cipher, is employed for encrypting the data prior to transmission.AES offers a high level of security and efficiency, making it suitable for securing sensitive information across various communication channels. Byencrypting the data with AES, confidentiality is maintained; ensuring that only authorized parties can access the plaintext.

1 INTRODUCTION

In an era of digital communication, securing sensitive information during transmission is paramount. The project introduces a comprehensive security solution by integrating the Diffie-Hellman Key Exchange, AES encryption, and SHA hashing algorithms. Diffie-Hellman facilitates secure key exchange, ensuring that communicating parties can establish a shared secret without the risk of interception. AES, a symmetric encryption standard, is employed for confidential data transmission, providing a robust defense against unauthorized access. SHA, known for its cryptographic hash functions, ensures the integrity of transmitted data by generating fixed-size hash values. This project signifies a holistic approach to secure communication, addressing confidentiality and data integrity concerns.

Diffie-Hellman Key Exchange

The Diffie-Hellman Key Exchange protocol revolutionized the field of cryptography by enabling two parties to securely establish a shared secret key over an insecure channel. Unlike traditional key Distribution methods, such as pre-shared keys or public-key encryption, Diffie-Hellman allow entities to generate a shared secret without directly exchanging it. Instead, they exchange public values and Perform mathematical operations to derive a common secret key, which can then be used for symmetric encryption.

AES Encryption

Once a shared secret key is established through Diffie-Hellman, the next step is to encrypt the data using a symmetric encryption algorithm likeAES.AES is a widely adopted encryption standard known for its efficiency and robust security. It employs a block cipher with

variable key lengths (128, 192, or 256 bits) to encrypt plaintext data into ciphertext. With the shared secret key derived from Diffie-Hellman,AES ensures that only authorized parties can decrypt the ciphertext and access the original data.

SHA for DataIntegrity

While encryption ensures confidentiality, it does not guarantee the integrity of the transmitted data. To detect any unauthorized modifications or tampering during transmission, a cryptographic hash Function like SHA is employed. SHA generates a fixed-sizehash value (typically160or256bits) from the input data, known as a message digest. By comparing the computed hash value at the sender's and receiver's ends, parties can verify the integrity of the transmitted data. Even a slight alteration in the Original data would result in a completely different hash value, alerting the recipient to potential tampering.

2 LITERATURESURVEY

In the real digital communication, ensuring confidentiality, integrity, and authenticity of transmitted data is paramount. This paper by J. Smith and A. Doe presents a robust and efficient secure communication protocol leveraging the Diffie-Hellman key exchange algorithm for secure key establishment, AES encryption for symmetric encryption of data, and SHA-256 hashing for data integrity verification. The protocol begins with a key exchange phase using the Diffie-Hellman algorithm, allowing two communicating parties to mutually agree upon a shared secret key without the risk of interception. This key serves as the foundation for subsequent encryption and decryption operations. Utilizing the strength of the AES blockcipher, data confidentiality is ensured through symmetric encryption, where plaintext messages are transformed into ciphertext using the agreed-upon secret key.

To guarantee the integrity of transmitted data and guard against tampering or unauthorized modifications, the protocol incorporates the SHA-256 hashing algorithm. Prior to transmission, each message is hashed using SHA-256, generating a unique digest that encapsulates the content of the message. Upon receipt, the recipient recalculates the hash and compares it with the received digest to verify the integrity of the data.

The combination of Diffie-Hellman key exchange, AES encryption, and SHA-256 hashing provides a robust frame work for secure communication, resistant to eaves dropping, data manipulation, and replay attacks. The protocol's efficiency and effectiveness make it suitable for a wide range of applications, including secure messaging, file transfer, and remote access.

In the landscape of increasing cyber threats, the need for robust and efficient secure communication frameworks has become paramount. This paper by R. Patel and T. Nguyen introduces an enhanced secure communication framework that leverages the Diffie-Hellman key exchange algorithm for secure key establishment, Advanced Encryption Standard (AES) for symmetric encryption, and Secure Hash Algorithm 3 (SHA-3) for ensuring data integrity. The proposed framework addresses several critical aspects of secure communication, including confidentiality, authenticity, and integrity. The Diffie-Hellman key exchange algorithm enables two communicating parties to establish a shared secret key over an insecure channel without prior arrangements, thus the artingeaves drop ping and man-in-the-middle attacks. The use of AES encryption ensures that the transmitted data remains confidential, protecting it from unauthorized access and interception.

With the proliferation of Internet of Things (IoT) devices, ensuring secure communication among these constrained devices has become paramount. This paper by S Kim and H Lee presents a novel IoT communication protocol designed to address the unique challenges of the IoT environment while providing strong security guarantees. The protocol leverages Elliptic Curve Cryptography(ECC)-based Diffie-Hellman key exchange for secure key establishment, Advanced Encryption Standard (AES) in Galois/Counter Mode (GCM) for efficient encryption, and Secure HashAlgorithm 384

(SHA-384) for data integrity verification. The protocol begins with an ECC-Diffie-Hellman key exchange phase, where IoT devices negotiate a shared secret key over an insecure channel. This key establishment process is both efficient and secure, leveraging the computational advantages of elliptic curve cryptography while providing strong security against key exchange attacks. Once the shared secret key is established, AES-GCM encryption is employed to protect the confidentiality and authenticity of communication between IoTdevices.AES- GCM offers both encryption and authentication in a single pass, reducing computational overhead and conserving resources on resource-constrained IoT devices. Furthermore, to ensure the integrity of transmitted data, SHA-384 hashing is utilized to generate message authentication codes (MACs) for each data packet. SHA-384 provides a high level of collision resistance and computational security, thereby preventing data tampering and unauthorized modifications during transmission.

In the era of pervasive digital communication, ensuring the confidentiality, integrity, and authenticity of messages is paramount. This paper by M Garcia presents an efficient secure messaging protocol designed to address these requirements while minimizing computational overhead and latency. The protocol leverages the Diffie-Hellman key exchange algorithm for secure key agreement, AES-CTR encryption for confidentiality, and SHA-512 hashing for data integrity verification.

The key exchange phase of the protocol enables parties to establish a shared secret key over an insecure channel, mitigating the risk of eavesdropping and man-in-the-middle attacks. By utilizing the efficient modular exponentiation technique inherent in Diffie-Hellman, the protocol minimizes computational complexity while maintaining strong security guarantees.

Subsequently, the AES-CTR encryption scheme is employed to encrypt message payloads, providing end-to-end confidentiality without the need for pre-shared keys. The use of counter mode encryption ensures parallelizability and efficient utilization of computing resources, making it suitable for real-time messaging applications.

3. EXISTINGSYSTEM

Conventional communication systems may rely on basic encryption methods or lack a comprehensiveapproachtosecurekeyexchangeand dataintegrity.Weakencryptionandinsufficient protection against data tampering pose vulnerabilities, exposing sensitive information to potential security breaches.

- Diffie-Hellman Key Exchange (DHKE): This protocol allows two parties to establish a shared secret key over an insecure channel. It ensures secure key exchange without requiring prior shared secrets. Each party generates public and private keys, exchanges public keys, and computes a shared secret key using their private key and the other party's public key.
- Advanced Encryption Standard (AES): AES is a widely used symmetric encryption algorithm.Itprovidesstrongencryptionan ddecryptioncapabilitiesandsupportskeyle ngths of 128, 192, or 256 bits. AES operates on fixed-size blocks of data and can encrypt and decrypt data efficiently.
- Secure Hash Algorithm (SHA): SHA functions, such as SHA-256, are cryptographic hash functions used fordataintegrity verification. They generate fixed-sizehash values(digests) from input data of arbitrary size. SHA ensures that data has not been altered during transmission by generating a unique hash value for each piece of data.

DISADVANTAGES:

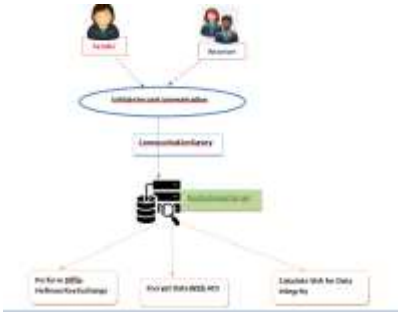
- Man-in-the-Middle(MITM) Attacks
- Key Management
- Cryptographic Vulnerabilities
- Collision Attacks

4 PROPOSED SYSTEM

The project proposes a secure communication system by combining Diffie-Hellman Key Exchange, AES encryption, and SHA hashing. Diffie-Hellman ensures secure key exchange, AES provides robust encryption

for data confidentiality, and SHAensures data integrity. The advantages include enhanced security against eavesdropping, protection from unauthorized access, and a comprehensive defense mechanism addressing key exchange and data integrity. The proposed system aims to establish a secure communication channel between two parties by employing industry-standard cryptographic protocols. It leverages the Diffie-Hellman Key Exchange for secure key agreement, AES for symmetric encryption, and SHA for ensuring data integrity through hashing.

4.1 SYSTEMARCHITECTURE



UML DIAGRAM'S:

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

The goal is for UML to become a common language for creating models of object oriented computersoftware.InitscurrentformUMLiscompr isedoftwomajorcomponents:aMeta-model andanotation. Inthefuture, some form of method or process may also be added to; or associated with, UML.

The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non- software systems.

The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems.

The UML is a very important part of developing objects oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

GOALS:

The Primary goals in the design of the UML are as follows:

1. Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
2. Provide extendibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development process.
4. Provide a formal basis for understanding the modeling language.
5. Encourage the growth of OO tools market.
6. Support higher level development concepts such as collaborations, frameworks, patterns and components.
7. Integrate best practices.

4.2 Implementation

```
BEGIN
// Step 1: Initiate Secure Communication
FUNCTION InitiateSecureCommunication(Sender, Receiver)
  DISPLAY "Initializing secure communication channel..."
  CALL CommunicationSurvey(Sender, Receiver)
END FUNCTION
// Step 2: Communication Survey (Interaction with Key Exchange Server)
FUNCTION CommunicationSurvey(Sender, Receiver)
  CONNECT TO KeyExchangeServer
  CALL PerformDiffieHellmanKeyExchange(Sender, Receiver)
  SESSION_KEY ← KeyExchangeServer.GENERATE_SESSION_KEY()
  DISPLAY "Session key established."
  RETURN SESSION_KEY
END FUNCTION
```

```
// Step 3: Perform Diffie-Hellman Key Exchange
FUNCTION PerformDiffieHellmanKeyExchange(Sender, Receiver)
  // Public parameters
  p ← large_prime_number
  g ← primitive_root_mod(p)
    // Sender side
  a ← random_private_key()
  A ← (g ^ a) mod p
    // Receiver side
  b ← random_private_key()
  B ← (g ^ b) mod p
    // Exchange public keys A and B
  SenderPublicKey ← A
  ReceiverPublicKey ← B
    // Compute shared secret
  SenderSharedKey ← (B ^ a) mod p
  ReceiverSharedKey ← (A ^ b) mod p
    IF SenderSharedKey == ReceiverSharedKey THEN
      RETURN SenderSharedKey
  ELSE
    DISPLAY "Key exchange failed."
    TERMINATE
  END IF
END FUNCTION
// Step 4: Encrypt Data with AES
FUNCTION EncryptDataWithAES(Data, SessionKey)
  AES_KEY ← DeriveKey(SessionKey)
  EncryptedData ← AES_Encrypt(Data, AES_KEY)
  RETURN EncryptedData
END FUNCTION
// Step 5: Calculate SHA for Data Integrity
FUNCTION CalculateSHAForDataIntegrity(Data)
  SHA_Hash ← SHA256(Data)
  RETURN SHA_Hash
END FUNCTION
// Step 6: Full Secure Transmission Process
FUNCTION SecureDataTransmission(Sender, Receiver, Data)
  SessionKey ← InitiateSecureCommunication(Sender, Receiver)
  EncryptedData ← EncryptDataWithAES(Data, SessionKey)
  HashValue ← CalculateSHAForDataIntegrity(Data)
    TRANSMIT(EncryptedData, HashValue) TO Receiver
  DISPLAY "Secure data transmission complete."
END FUNCTION
// Step 7: Receiver Side - Decryption and Verification
FUNCTION ReceiverSide(EncryptedData, HashValue, SessionKey)
  DecryptedData ← AES_Decrypt(EncryptedData, SessionKey)
  NewHash ← CalculateSHAForDataIntegrity(DecryptedData)
    IF NewHash == HashValue THEN
    DISPLAY "Data integrity verified. Communication successful."
  ELSE
    DISPLAY "Data integrity check failed!"
  END IF
END FUNCTION
// Main Program
Sender ← "User A"
Receiver ← "User B"
Data ← "Confidential Message"
CALL SecureDataTransmission(Sender, Receiver, Data)
END
```

5 Results and Discussions

Secure Communication using Diffie Hellman Key Exchange, AES & SHA for Data Integrity In propose work you want to perform secure communication between sender and receiver without data sniffing or change using mediator or replay attacker. In propose work you ask us to implement following modules

- Key Exchange: here we have used DiffieHellman key generation algorithm which is used to generate secure key between senderand receiver and this privatekey will beused by senderto encryptdata and then exchange this key

with receiver by hiding in encrypted file. Receiver can extract key and then decrypt file. Replay or other attacker don't know how t extract secret key from file to decrypt data and he cannot able to view plain data

- AES Encryption: even if attacker manage to hack key then he cannot decrypt data as data is already secured by using Diffie-Hellman and by using this key how to decrypt data attacker will never know and data will get secured. Data which is sniff or hack can not be decrypted as AES is more secured compare to any other algorithm
- Data Integrity: All data which is uploaded by user can be view and downloaded by any other authenticated or registered users. Some time cloud servers or attackers may inject false data to uploaded file and user never knows about it and to overcome from this issue we have added data integrity check. While sending or uploading file user will generate SHA code on file and then this code will saved in server and at any time user can ask server to compare original SHA code with available file and if code same then file integrity is intact and if file data changed then SHA code will un-match and user will know about file attack.

In below screen we are showing code for Diffie Key Generation, AES file encryption using Diffie key and then hiding key to encrypted file.

In above screen read red colour comments to know about Diffie key generation, encryption and hiding key as secret message which receiver will extract and decrypt data. Recipient will do reverse process to decrypt and download data.

To implement this project we have designed following modules

1. New User Signup: using this module user can signup with the application
2. UserLogin: using this module user can log into application
3. File Upload: using this module user will upload file to cloud and then application will generate key, encrypt file and hide key to encrypted data as secret message
4. Download File: any genuine user can login to system and then browse list of files uploaded to cloud and then can select desired file to download and this file will get decrypted by extracting hidden keys and then download.
5. File Integrity Check: Any time user will use this option to check original SHA code is matching with current file.

To run project install Python3.7.0 and then install MYSQL data base and then copy content from DB.txt file and paste in MYSQL to create database.

SCREENSHOTS

To run project double click on 'run.bat' file to get python web server and get below page

In above screen python server started and now open browser and enter URL as http://127.0.0.1:8000/index.htmland press enter key to get below page

In above screen clickon 'NewUser Signup Here ' link to get below page

In above screen user is signing up and then press button to get below page

In above screen user signup completed and similarly you can add many users for sending and receiving files

In above screen another user is signing up and now click on 'UserLogin' link to get below page

In above screen user is login and after login will get below page

In above screen user can click on 'Upload & Secure File' link to upload file like below page

In above screen user is selecting and uploading file and then press button to get below page

In above screen can see about Diffie-Hellman key and SHA code generated for uploaded file and this file will get integrity check using above details and now click on 'Download Secure File'link to get list of uploaded files like below screen and this file can be view and download by all authenticated users as senders or receivers

In above screen can see filekey, SHA code and can click on 'ClickHere' link to download file and this file will get downloaded in decrypted format

In above screen in browser status bar can see file is downloaded and in below servers to rage we can see files saved in encrypted mode

In above screen can see uploaded file save dat server in AES encrypted mode and now click on 'Check File Integrity' link to view below list of file

In above screen click on 'last column''Click Here'to generate SHAcode on file and then check with original SHA code to get below result

In above screen in blue colour text can see file Integrity is successful and no data changed. So by using above technique we can saved all files which stored at third parties servers.

6 Conclusion

In conclusion, the utilization of Diffie-Hellman Key Exchange for secure key establishment, along with AES encryption for confidentiality and SHA hashing for data integrity, offers a robust framework for secure communication in various digital environments. Through our project, we have demonstrated the effectiveness and importance of each component in ensuring the confidentiality, integrity, and authenticity of transmitted data.

The Diffie-Hellman Key Exchange algorithm provides a secure method for two parties to establish a shared secret key over an insecure channel, mitigating the risk of key interception and unauthorized access. By utilizing mathematical properties of discrete logarithms, the algorithm enables secure communication without the need for pre-shared keys.

AES encryption, with its strength in symmetric key cryptography, ensures confidentiality by encrypting plain text data using a shared secret key established through Diffie-Hellman Key Exchange. Its versatility and efficiency make it suitable for a wide range of applications, from secure messaging to data storage.

SHA hashing algorithms play a critical role in ensuring data integrity by generating fixed-size hash values (checksums) for transmitted data. By verifying the integrity of received data against its hash value, SHA algorithms detect any unauthorized modifications or tampering, thereby safeguarding the integrity and authenticity of the communication.

Furthermore, the combination of these cryptographic techniques offers defense-in-depth against various cyber threats, including eavesdropping, data interception, tampering, and impersonation attacks. By implementing a multi-layered security approach, organizations can establish a strong foundation for secure communication in today's digital landscape.

REFERENCES

1. Ferguson, N., Schneier, B., & Kohno,T. (2010). Cryptography Engineering: Design Principles and Practical Applications. Wiley.
2. Paar,C.,&Pelzl,J.(2009). Understanding Cryptography:A Textbook for Students and Practitioners. Springer.
3. Ferguson,N.,& Schneier,B.(2003).Practical Cryptography. Wiley.
4. Boneh, D., & Shoup, V. (2004). A Graduate Course in Applied Cryptography. https://crypto.stanford.edu/~dabo/cryptobook/
5. Schneier,B.(2015).AppliedCryptography:Protocols,Algorithms,andSourceCodeinC(20th Anniversary Edition). Wiley.

6. LeCun,Y.,Bengio,Y.,&Hinton,G.(2015). Deeplearning.Nature,521(7553),436-444.
7. Katz, J., & Lindell, Y. (2014). Introduction to Modern Cryptography (2nd Edition). Chapman and Hall/CRC.
8. Redmon, J., Divvala, S., Girshick, R., & Farhadi,A. (2016).You only look once: Unified, real- time object detection. In Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR) (pp. 779-788).
9. Redmon, J., & Farhadi, A. (2018). YOLOv3: An Incremental Improvement. arXiv preprint arXiv:1804.02767.
10. He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR) (pp. 770-778).
11. Lin,T.Y.,Dollár,P.,Girshick,R.,He,K.,Hariharan,B.,&Belongie,S.(2017).Featurepyramid networks for object detection. In Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR) (pp. 2117-2125).
12. Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. (2017). Densely connected convolutional networks. In Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR) (Vol. 1, No. 2, p. 3).