HYBRID CRYPTOGRAPHY

Team Details
1. G Vishwas (20eg105411)
2. M Abhinav (20eg105426)
3. R Sai Chaitanya (20eg105437)

Project Supervisor
B Ravinder Reddy
Assistant professor

Introduction

Cryptography and Information Security: Hybrid of two algorithms ECC and diffie-hellman.

A hybrid algorithm that combines Elliptic Curve Cryptography (ECC) and the Diffie-Hellman key exchange can provide a secure way of key generation and exchanging process.

The ECC key exchange provides secrecy and protection against quantum attacks, while Diffie-Hellman offers a well-established method for secure key exchange.

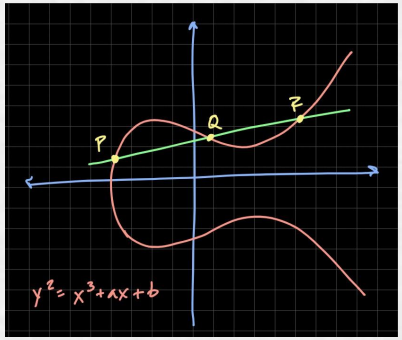
Problem Statement

The combination of Diffie-Hellman acts a secure key transmission agent for the RSA and the RSA accounts for the security of message. However, this method has a very huge key length which effects the performance for the system.

Proposed Method

ECC, an alternative technique to RSA, is a powerful cryptography approach. It generates security between key pairs for public key encryption by using the mathematics of elliptic curves.

- It makes use of elliptic curves.
- The curves are symmetric to x-axis.
- A line is drawn at any random place on the curve , the points where the line touches are taken as public and private values



Proposed Method

Start
Generate ECC Key Pair (Alice)
Select ECC Parameters (Curve, Base Point)
Generate ECC Key Pair (Bob)
Alice and Bob exchange public keys
Alice computes Shared Secret Key
Bob computes Shared Secret Key
Shared Secret Key is now established
End

Experiment Environment

We use visual studio for this project

- Visual Studio Code is a highly popular and versatile code editor developed by Microsoft.
- It boasts a rich ecosystem of extensions, making it adaptable for various programming languages and development tasks.
- VS Code's sleek design, integrated Git support, and powerful debugging tools make it a top choice for developers across the globe.

Experiment Screenshots

ANURAG UNIVERSITY

```
from cryptography.hazmat.primitives import hashes
from cryptography.hazmat.primitives.asymmetric import ec
from cryptography.hazmat.primitives.kdf.hkdf import HKDF
from cryptography.hazmat.primitives.ciphers import Cipher, algorithms, modes
from cryptography.hazmat.backends import default_backend
import base64

def generate_ecc_key_pair():
    private_key = ec.generate_private_key(ec.SECP256R1(), default_backend())
    public_key = private_key.public_key()
    return private_key, public_key

def perform_diffie_hellman(private_key, peer_public_key):
    shared_key = private_key.exchange(ec.ECDH(), peer_public_key)
    return shared_key

def derive_symmetric_key(shared_key):
    derived_key = HKDF(
        algorithm=hashes.SHA256(),
        length=32,
        salt=None,
        info=b'symmetric key',
        backend=default_backend()
    ).derive(shared_key)
    return derived_key

def encrypt_data(plaintext, symmetric_key):
    iv = b'\x00' * 16
    cipher = Cipher(algorithms.AES(symmetric_key), modes.CFB(iv), backend=default_backend())
    encryptor = cipher.encryptor()
    ciphertext = encryptor.update(plaintext) + encryptor.finalize()
    return ciphertext

def decrypt_data(ciphertext, symmetric_key):
    iv = b'\x00' * 16
    cipher = Cipher(algorithms.AES(symmetric_key), modes.CFB(iv), backend=default_backend())
    decryptor = cipher.decryptor()
    plaintext = decryptor.update(ciphertext) + decryptor.finalize()
    return plaintext
```

Experiment Screenshots

```
# Step 1: Key Generation
private_key_A, public_key_A = generate_ecc_key_pair()
private_key_B, public_key_B = generate_ecc_key_pair()

# Step 2: Key Exchange
shared_key_A = perform_diffie_hellman(private_key_A, public_key_B)
shared_key_B = perform_diffie_hellman(private_key_B, public_key_A)

# Step 3: Key Derivation
symmetric_key_A = derive_symmetric_key(shared_key_A)
symmetric_key_B = derive_symmetric_key(shared_key_B)

# Encryption
plaintext = input("Enter plain text: ").encode('utf-8')
ciphertext = encrypt_data(plaintext, symmetric_key_A)
print("Encrypted data:", base64.b64encode(ciphertext).decode('utf-8'))

# Decryption
decrypted_text = decrypt_data(ciphertext, symmetric_key_B).decode('utf-8')
print("Decrypted data:", decrypted_text)
```

Experiment Results

```
Enter plain text: Hello! Welcome to the World
Encrypted data: xHRSAjpJRFguOJygne4cWl1AUpXZsKK6P5tK
Decrypted data: Hello! Welcome to the World
```

Finding

- **Security**: ECC provides a high level of security with smaller key sizes compared to traditional algorithms like RSA. This enhances the overall security of the key generation and exchange process. The Diffie-Hellman key exchange ensures that even if an attacker intercepts the public keys exchanged during the key exchange, it is computationally infeasible for them to derive the private keys.
- **Efficiency**: ECC allows for smaller key sizes while maintaining a high level of security. This results in faster key generation and exchange compared to algorithms with larger key sizes. The use of ECC and efficient key exchange algorithms like Diffie-Hellman minimizes computational overhead, making it more efficient for devices with limited resources.
- **Secure Communication**: The derived shared secret can be used for subsequent symmetric encryption, enabling secure communication between parties. The encrypted data can only be decrypted by the intended recipient possessing the appropriate private key.
- **Ease of Integration**: The project provides a clear and structured demonstration of implementing ECC, Diffie-Hellman key exchange. This can be used as a basis for integrating these cryptographic processes into real-world applications.

Justification

The parameters improved by this method are:

- **Security:** This Hybrid algorithm ensures confidentiality and integration better than that of RSA with Diffie Hellman
- **Speed and Performance:** Using this hybrid algorithm, it gives better performance and ensures Performance Optimization and also doesn't require high computational power.
- **Key Exchange Mechanism:** Diffie Hellman is the best key exchange algorithm. ECC with Diffie Hellman makes key exchange mechanism more secure and also faster.
- **Secure Communication**: ECC and Diffie Hellman enables secure communication between parties. The encrypted data can only be decrypted by the intended recipient possessing the appropriate private key.