DOCUMENTATION

Diffie-Hellman Key Exchange Algorithm combined with AES Symmetric Encryption between two communicating parties

Table of Contents

Degenbaev Iskender

Introduction

In today's digital world, ensuring secure communication over insecure channels is of paramount importance. Cryptographic protocols play a crucial role in achieving this goal.

Background

The Diffie-Hellman key exchange algorithm is a cornerstone of modern cryptography. It allows two parties to securely establish a shared secret key over an insecure communication channel. AES (Advanced Encryption Standard) is a symmetric encryption algorithm widely used for encrypting and decrypting data securely.

Implementation Details

This project is a simple implementation created by a student for educational purposes. It involves generating Diffie-Hellman keys for two parties, exchanging public keys, computing a shared secret key, and using AES encryption and decryption for secure communication.

Environment Setup

Before running the program, ensure you have Python installed on your system. Additionally, you need to install Cryptography library using the following command: pip install cryptography

```
degen@Isken MINGW64 /c/Projects/SecureCommunication
$ pip install cryptography
```

Diffie-Hellman Key Exchange

The Diffie-Hellman key exchange algorithm involves several steps:

- Generating Diffie-Hellman keys for both parties.
- Exchanging public keys securely.
- Computing a shared secret key using each party's private key and the received public key.

AES Encryption and Decryption

AES encryption and decryption are performed using the shared secret key obtained from the Diffie-Hellman key exchange. This ensures the confidentiality of the communication between the parties.

Communication Simulation

The simulation involves two parties, Alice and Bob, exchanging messages securely:

- They generate Diffie-Hellman keys and exchange public keys.
- They compute a shared secret key.
- They use the shared secret key for AES encryption and decryption of messages.

Running the Program

To run the program, execute the Python script "secure_communication.py". The program will simulate secure communication between Alice and Bob.

```
degen@Isken MINGW64 /c/Projects/SecureCommunication
$ python secure_communication.py
```

Source Code Explanation

Alice and Bob generate their Diffie-Hellman key pairs

```
alice_dh = DiffieHellman()
bob_dh = DiffieHellman()
```

Exchange public keys

```
alice_public_key_bytes = alice_dh.get_public_key_bytes()
bob_public_key_bytes = bob_dh.get_public_key_bytes()
```

Load public keys

```
alice_public_key = DiffieHellman.load_public_key(alice_public_key_bytes)
bob_public_key = DiffieHellman.load_public_key(bob_public_key_bytes)
```

Generate shared keys

```
alice_shared_key = alice_dh.generate_shared_key(bob_public_key)
bob_shared_key = bob_dh.generate_shared_key(alice_public_key)
```

Encrypt and decrypt a message using the shared key

```
message = "Hello, Bob! This is Alice."
aes_cipher = AESCipher(alice_shared_key)
```

Running Program

```
degen@Isken MINGW64 /c/Projects/SecureCommunication
$ python secure_communication.py
Alice's public key: -----BEGIN PUBLIC KEY-----
MIICJDCCARcGCSqGSIb3DQEDATCCAQgCggEBAJ92yNXqK0E7SMsaVzJcCiXkZRiP
5dyRhzp4K2FV7iTN3kiOylnNGf5uyWOYqe4liySeDtfS06r5Mt+FqNms0Gu1NtNQ
qJq3c2QVimVTAHr24JV8wjKQ0WCgakMNJ0Yfv/2Pr86rP3YvJ5GkX60Qr/vvk470
DEPpH0lx4h4Xhn29WwjKVY8FJBUQXjDZUr3FQfxto1TlC2td/pyb8KhUDXERNLan
qZyaerzfITpc+SWdRWXB86uifDzASMRHWL2LmVDlPRU+HiMfFz4yGnfLd+RovUry
5g/dtJtEjMUE+tpYDkouz4GIWrUzIsBZhYicCjJ5OmrkxABz+cn+BII6mi8CAQID
ggEFAAKCAQBe5sB1SSaBt1eYJySkv3sqJrf2wSmOXujvc79jz3s1TPmT5Bst06ry
a/jwKAtxskICXdGbK2hq6pcTP40t7HQh0z33jBiZ5jUVa05wbX30hPsp0X2pwAua
Ju0tg3t2Kh1+KPLDmCncLMg2k7f/4xGt8Pm9FB+qp5wGIQEYmCTUklMorbfw85Js
v5IWSXnaqJTCGch8tfq5Dc0fbjEx8eJajn66Cbs1poFkLjzmpedqIxBPAbEZK7gX
WIzUe57r1IBI+8rJ+cQTxrVjUP7HA7ko369PyrmCjqnQgo2+IO92wWYGtA25T6kX
iw0rvnRVfriED0i/6oT12u+0Y/0lhIuW
-----END PUBLIC KEY-----

Bob's public key: -----BEGIN PUBLIC KEY-----
MIICJDCCARcGCSqGSIb3DQEDATCCAQgCggEBAJ92yNXqK0E7SMsaVzJcCiXkZRiP
5dyRhzp4K2FV7iTN3kiOylnNGf5uyWOYqe4liySeDtfS06r5Mt+FqNms0Gu1NtNQ
qJq3c2QVimVTAHr24JV8wjKQ0WCgakMNJ0Yfv/2Pr86rP3YvJ5GkX60Qr/vvk470
DEPpH0lx4h4Xhn29WwjKVY8FJBUQXjDZUr3FQfxto1TlC2td/pyb8KhUDXERNLan
DEPpH0lx4h4Xhn29WwjKVY8FJBUQXjDZUr3FQfxto1TlC2td/pyb8KhUDXERNLan
qZyaerzfITpc+SWdRWXB86uifDzASMRHWL2LmVDlPRU+HiMfFz4yGnfLd+RovUry
5g/dtJtEjMUE+tpYDkouz4GIWrUzIsBZhYicCjJ5OmrkxABz+cn+BII6mi8CAQID
ggEFAAKCAQBdCL0xYh7uZ/FtQVQxkbEukcZFWCT3sdsHx3IO2VXdd2lIINlpZle5
GnN8s+L2yoUXh7jPW8MWGebYuS1eueG0yoLWl+7G05Z2o9o0R2EZOPcfkTSeuKza
PxMfgf52Usq0uaftsZa0Y6nHTIQgF0iYlcSSFT2rnChnSK5QZE9/+2mFF/cBoeoS
vGxnamyJavEX8zqaLs0dP9B6zwxsbmyQXYM0RJXnsefXl3t1ceqWccgjt6/GqftK
ft5gBwWfPNij6TgHdmgomSSrqyzq22uwyh+bW58HauYakLe+7VLsiTGlTstJ3iYY
vxyrl4dvcizpfsPsrrCC5Y0q8951c27x
-----END PUBLIC KEY-----
```

```
Encrypted message: b"9\x83\xd6\xb6$\x8d\x9aB)\x1e\x10\xaa\x80\x86
\x04\xf8\xb9\xb0\xcd\x97At\xdcY\xfe\x9b\x00\xe1'\x17\x02\xbbt\x08
\x81\x94Ie\x00:5c\xde\xee\xcb\xc5\xb6\x11"
Decrypted message: Hello, Bob! This is Alice.

degen@Isken MINGW64 /c/Projects/SecureCommunication
$
```

Conclusion

This project demonstrates the basic implementation of secure communication using the Diffie-Hellman key exchange algorithm and AES encryption. While simple, it highlights the fundamental concepts of cryptographic protocols and their role in ensuring secure communication.

I've added everything I could to the documentation. I might have missed something, probably because I haven't slept in three days. LOL