SECURED TRANSMISSION OF DATA

USING A RANDOMN ENCRYPTION ALGORITHM

CIS-628 Introduction to Cryptography

Submitted by  
Chaitanya Lakshmi Chava   
Under

Vir Virander Phoha

**Abstract:**

In this paper, we propose a new approach to secure data transmission that involves the use of a randomly selected encryption algorithm. Our method involves encrypting the message with a different algorithm each time it is transmitted and encrypting the name of the used algorithm and concatenating it with the message. This approach has the potential to enhance security by making it more difficult for attackers to intercept and decrypt the message, as they would not know in advance which algorithm was used.

Through our research, we aim to evaluate the effectiveness of this approach in improving the security of data transmission and addressing the limitations of existing methods. We also aim to identify and address any technical or implementation challenges that may arise during the development and testing of our system. Our results show that our method is effective in improving the security of data transmission and is resistant to man-in-the-middle attacks and other types of attacks.

**Problem Statement:**

Despite the availability of various encryption algorithms and protocols, the security of data transmission remains a significant concern. Traditional methods that rely on a single algorithm or protocol are vulnerable to many attacks, that can compromise the confidentiality and integrity of the transmitted data. So, we are going to develop an inter communication secured channel where every message will be encrypted using a different algorithm every time it is sent. Our problem statement is to develop the channel where the messages can be encrypted.

**Execution:**

The way we executed our idea is that we have chosen python as our programming language as it has wide variety of libraries available. Next, we started implementing our idea starting with random selection of cryptographic technique. That is done in a python file named techniqueSelector.py.

**TechniqueSelector.py:**

This python code fetches a public key that is stored locally in a file name public\_key.pem and randomly picks an encryption algorithm. It then encrypts the name of that randomly picked algorithm using RSA algorithm with the fetched public key. That encypted encryption algorithm name is stored in a file called encrypted\_technique.txt.

**ConversationDisplay.py:**

Now we moved to conversationDisplay.py that helps display the conversations of users. This code generates public and private key pair using python’s rsa library and stores the public key in a pem file which is used in the above-mentioned python code. Then this code goes into an infinite loop that continuously reads a file called conversation.txt, decrypts the messages and prints the conversations in it.

**Sender.py:**

Now this piece of code reads the encryption technique name from a file called encrypted\_technique.txt and asks for the name of the user that is going to start the chat. Then it goes to an infinite loop which takes in the current timestamp, username, and the message that the user had entered and concatenates all the information. Which is then encrypted using the randomly picked encrypted algorithm and stored in a file called conversation.txt.

**Significance:**

The main motto of our team is to develop a secure chat application that is less prone to man in the middle attacks and other hacking techniques. Our algorithm handles it very well by not allowing the intruders to know hoe the messages are encrypted by randomly picking a new cryptographic technique every time the application is run. This makes it so difficult for the intruders to breach.

That is what makes our application so strong and less prone to these kinds of attacks.

**Novelty:**

In our project we increased security of data during the transmissions. We accomplished our thoughts as proposed. In our future, upon accomplishing this successfully, we would like to work this in accordance with the input i.e., based upon the type (text, audio, video) of message, the algorithm would select the random algorithm for encryption among the set of algorithms suitable to it.

**Literature Search:**

"Research and implementation of RSA algorithm for encryption and decryption"

The main goal of using encryption techniques is to ensure the integrity, certainty of the information and confidentiality. In practice, symmetric key algorithms and public key cryptography algorithms are always combined to achieve the maximum efficiency. The security or the result obtained for the algorithms depends on its internal structure and the difficult mathematics. In this the mode n, play a major role and hence the decomposition of n can be the easiest way to attack. A hash function is used to create the digital signature for the RSA algorithm. Digital signatures have the advantage of representing the file's characteristics. If the file changes, the value of the digital signature changes. Different files receive digital signatures. The accumulation of a series of binary codes is one of the most fundamental hash functions.

"Research on Diffie-Hellman key exchange protocol"

The Diffie-Hellman protocol allows two users to securely exchange a secret key, which can then be used for subsequent message encryption. The protocol itself is limited to the exchange of keys. However, due to the lack of an entity authentication mechanism, the Diffie-Hellman protocol is vulnerable to man-in-the-middle and impersonation attacks in practice. Although DSA and RSA have nearly identical cryptographic strengths, they each have distinct performance advantages. DSA is faster at decryption and signing than RSA is at encryption and verification. So, if you frequently encounter performance issues, it might be a good idea to look at the source of the problem (i.e., whether it's client- or server-based) and base your key algorithm selection on that.

"Integrated design of AES (Advanced Encryption Standard) encrypter and decrypter"

In an AES system, which is a symmetric-key system, both the sender and the recipient use the same key to encrypt and decode the message. A key or message's data length can be set to 128, 192, or 256 bits. The AddRoundKey, SubBytes, ShiftRows, MixColumns, and KeyExpansion procedures make up the AES encryption algorithms five fundamental operations. Data encryption and decryption processes are fundamentally the same thing. The digital signature (EC) protocol additionally requires integer multiplication, inverse operation, modular operation, and a hash function in addition to elliptic curve operations like scalar multiplication, field multiplication, and field inverse multiplication.

"A comparison of the standardized versions of ECIES"

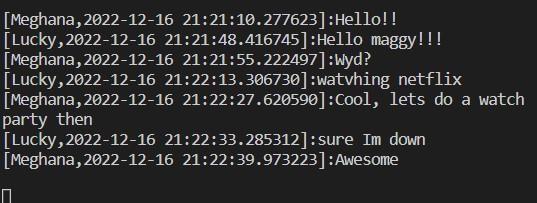
Even a quantum computer is expected to have difficulty solving the well-known NPcomplete problem known as the Knapsack problem. Because of this, post-quantum cryptography favors this type of cryptosystem over all others. Recently, many fresh knapsack-based cryptosystems have been put out. Like the Merkle Hellman cryptosystem, the fundamental operations of each of these systems are super expanding sequences and modular multiplications. The basic Merkle Hellman cryptosystem is the subject of Shamir's attack in this article, and we also present an improved version of it. This new idea will be useful in assessing the security of the new knapsack-based cryptosystems. The orthogonal lattice technique is our primary weapon.

**Results:**

Project directory contains three python scripts namely, techniqueSelector.py, sender.py and conversationDisplay.py.

**ConversationDisplay.py:**

This python code is responsible for displaying the chat box which contains users conversation where each message has a sender name and the timestamp the message has been sent will be attached to the message.



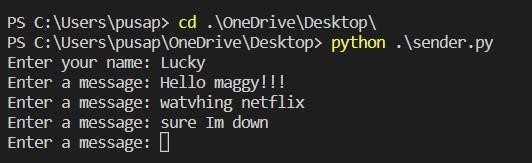
**TechniqueSelector.py**

This python script is responsible for selecting a random encryption algorithm from set of algorithms. This code randomly picks a cryptographic technique and stores it in text file which is also encrypted using RSA algorithm.

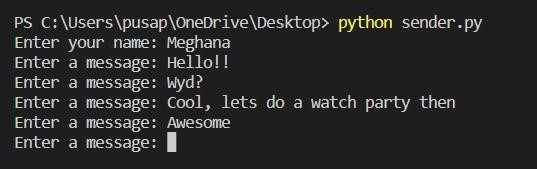
**Sender.py**

This python file is responsible for encryption of a message with the randomly chosen encryption technique if the user is sending a message and also for the decryption of the same message that is received by the receiver.

**User1:**



**User2:**



**Execution instructions:**

* Go to the directory where all the above three files are present
* Run conversationDisplay.py
* Run techniquSelector.py in a separate command prompt window
* Run sender.py as many times you want depending on the number of users you want to get involved in the conversation. Note: each sender.py must be executed in different windows to allow different users to chat at the same time in different windows.

**Challenges Faced:**

The main that we have faced while executing our idea is that we faced a lot of trouble while storing an encrypted technique name and referencing it in another python file to access it the same way it was stored.

Same challenge was faced while store public key is .pem file.

The way we overcame this challenge is by using repr() and eval() functions of python. Repr() function is used to store the exact representation of the given string in a file and eval is used to evaluate the given string exactly as compiler does.

This way we can develop a chatting application that can take as users as it can and allow users to chat with each other simultaneously. Yet there is lot of room to improve by scaling this application even more and moving this environment from local to cloud.

**References:**

1. X. Zhou and X. Tang, "Research and Implementation of RSA algorithm for encrption and decryption," in *Proceedings of 2011 6th International Forum on Strategic Technology*, 2011.
2. N. LI, "Research on Diffie-Hellman key exchange protocol," in *2010 2nd International Conference on Computer Engineering and Technology*, 2010.
3. S.-M. Yen and C.-S. Laih, "Improved Digital Signature algorithm," *IEEE Transitions on Computers,* vol. 44, pp. 729-730, 1995.
4. C.-C. Lu and S.-Y. Tseng, "Integrated design of AES (Advanced Encryption Standard) encrypter and decrypter," in *Proceedings IEEE International Conference on Application - Specific Systems, Architectures and Processors*, 2002.
5. G. M. V, H. E. F and S. A. v. C, "A Comparison of the standardized versions of ECIES," in *2010 Sixth International Conference on Information Assurance and Security*, 2010.
6. L. J, B. J and X. S, "An Improved Attack on the Basic Merkle-Hellman Knapsack Cryptosystem," *IEEE Access,* vol. 7.