UNIVERSITY OF NORTH TEXAS
COLLEGE OF ENGINEERING
DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

CSCE 5050 – Application of Cryptography

Project 1

Due Date: Friday, June 3rd , 2022 at 5:00PM

1 Goals

For the first programming project you will learn how to apply security to an application, in particular, you will learn how to use cryptography to secure the data that your applications use. You are provided with a simple but working C# client/server chat system. The system as provided is totally insecure, all the messages are transmitted in the clear. You will be required to add security features on top of the provided framework, using both the cryptography and security features found in the .NET platform.

The required security features are:

1. A key exchange between the chat client and the chat server when the client connects.
2. Encryption of all chat messages with a block cipher in CBC mode.
3. Integrity check for all chat messages using Message Authentication Codes (MACs).
4. You must invent and implement a way to prevent replay attacks.

We will examine each of these features in detail in the following section

2 Security Features

2.1 Key Exchange

You may use the Diffie-Hellman key exchange protocol. Note that we have not yet discussed a method to prevent the "person-in-the-middle" MiM attack on key exchange protocols. Consequently, in this project you are only required to secure the key exchange protocol against a passive eavesdropper. We will deal with the MiM attack in a later project.

You should use the secret obtained during key exchange to generate your block cipher key, your CBC IV and your MAC key.

2.2 Message Encryption

Each message transmitted by either a client or the server must be encrypted using a block cipher. You may use any standard block cipher you like, but all messages must be encrypted using CBC mode. As mentioned above, you should generate both the cipher key and the CBC IV from the secret obtained during key exchange.

2.3 MACs

Every message that goes over the network should have a MAC to enable the detection of a malicious attacker tampering with messages en route. You should generate the key for this MAC from the secret obtained during key exchange as described above.

2.4 Preventing Replay Attacks

Cryptography itself is vulnerable to attack and is not something that can be "plugged into" an application and then forgotten. Even after you secure chat messages with encryption and MACs, there is still an obvious replay attack. Trudy captures a chat message en route to either the server or a particular client. She then repeatedly sends that message – which is still a valid chat message – flooding the intended recipient and making the chat room unusable for the participants. Your solution should prevent an attacker from replaying a user's message.

2.5 Security Holes

Even with these security features, it should be clear that there are still holes in the Chat system. Aside from the "replay attacks" and the "person-in-the-middle" attack on the key exchange protocol, the biggest remaining problem is authentication. The easiest way for an attacker to see the messages in the chat room is simply to join. Address these problems is outside the scope of this project.

3 Implementation

As mentioned in class, you will be programming this project in C#. We have provided a simple but working code for simple chatting system. Please keep the existing comments in this code without change, you may add your own comments if you wish. The description of this chat system is given in the appendix of this project.

3.1 .NET and Cryptography

.NET provides a set of cryptographic objects, supporting well-known algorithms and common uses including hashing, encryption, and generating digital signatures. These objects are designed in a manner that facilitates the incorporation of these basic capabilities into more complex operations, such as signing and encrypting a document. Cryptographic objects are used by .NET to support internal services, but are also available to developers who need cryptographic support. The .NET Framework provides implementations of many such standard cryptographic algorithms and objects. The `System.Security.Cryptography` namespace in the .NET Framework provides these cryptographic services. The Algorithm support includes: DES, TripleDES, RC2 and Rijndael as symmetric crypto, RSA and DSA public key as asymmetric crypto, MD5 and SHA1 hashing algorithms.

The SymmetricAlgorithm class allows you to configure an algorithm (select the block size, padding mode, etc.) and create instances of the classes that encrypt and decrypt data.

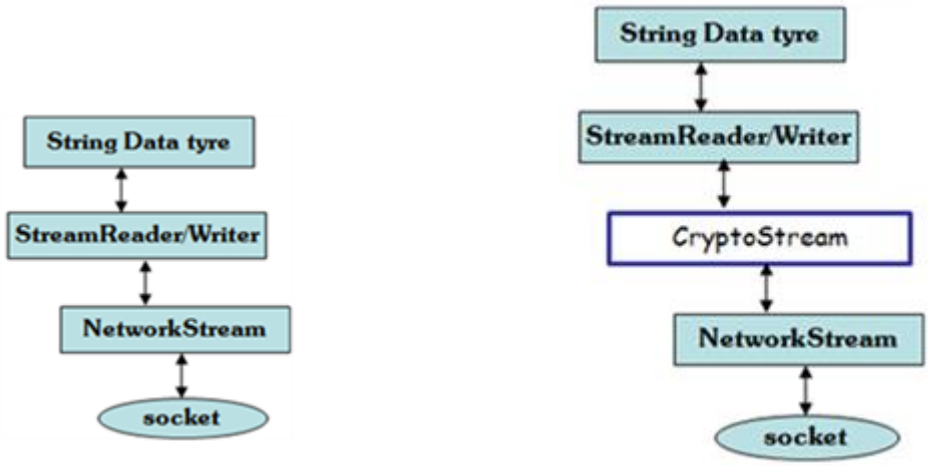
Please keep "msdn .NET Framework Developer Center" your best friend who will support you with the required cryptographic tools used in this project. It can be found at: http://msdn.microsoft.com/en-us/library/system.security.cryptography.aspx

3.2 Using IO streams

The provided Chat system is built on an IO composable streams, it utilizes the network stream, stream reader and stream writer to have the TCP socket accessible in the form of string data type stream, as it shown in Figure 1a. In this project you need to add one more stream layer, the CryptoStream need to be added on the top of NetworkStrem and before StreamReader/Writer streams, as shown in Figure 1b.

The CryptoStream class acts as a wrapper around a stream and automatically transforms blocks of data using an instance of ICryptoTransform class. The CryptoStream class transforms data read from a stream (for example, decrypting ciphertext read by the StreamReader) or written to a stream (for example, encrypting programmatically generated data and storing the result using the StreamWriter).

Please note that, your work will not be accepted unless it follows this approach.



(a) (b)

Figure 1: Using I/O Streams

4 Word about Networking

The Chat system would be rather boring if it didn't run over the network. The problem for us is that we cannot expect everyone interested in cryptography to have network programming experience. For that reason, a Chat system is given to demonstrate the network communications in our project.

5 Submission

In addition to your well-decomposed, well-commented solution to the project, you should submit a README containing the names, UNT usernames and ID, as well as a **description of the design choices you made in implementing each of the required security features**.

1. When you are ready to submit, make sure you archived all your working directories in one file named *your_groupName*_CSCE5050_project1.
2. Submit the archived file through the Canvas before the due date mentioned above.