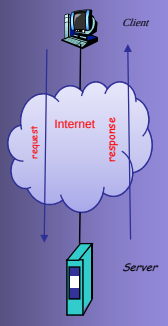


INTERNET PROTOCOLS AND CLIENT-SERVER PROGRAMMING SWE344

Fall Semester 2008-2009 (081)

Module 14: Network Security

Dr. El-Sayed El-Alfy
Computer Science Department
King Fahd University of Petroleum and Minerals
alfy@kfupm.edu.sa

Objectives

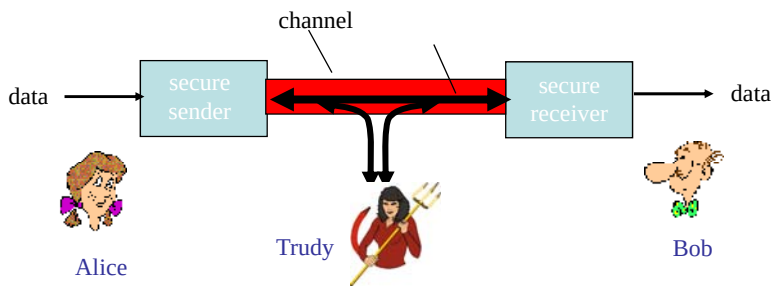
- Define basic security terminology
- Understand the principles of data encryption and network security
- Distinguish between symmetric and asymmetric data encryption
- Write simple data encryption/decryption programs using C#

What is Network Security?

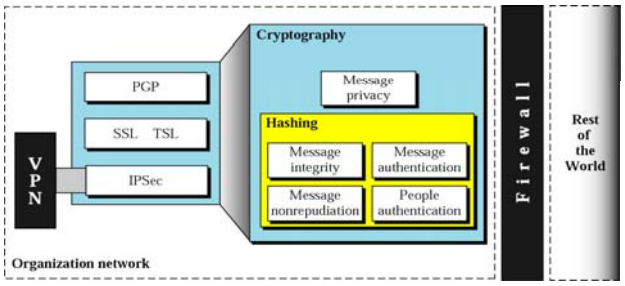
- Network security refers to all the activities that an organization undertakes in order to create a secure computing environment and to protect network assets.
- Network assets (network resources and communication) including data, systems, and applications should only available to authorized users (programs and human users)
- This implies detection and control of physical and logical threats with respect to availability, confidentiality, integrity, access control, authentication
- Other related terms: information security, data security, computer security, system security

Motivation

- Sending messages across the Internet is risky; hackers can intercept packets and look at their content



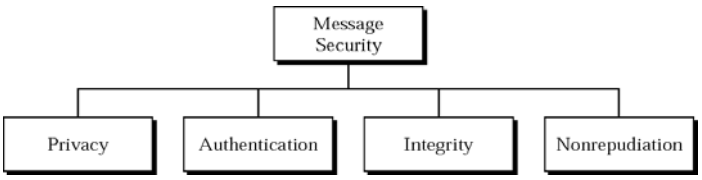
Framework for Network Security



- Network security is a broad area that includes major topics such as
  - Physical security mechanisms and security policies
  - Cryptography – the heart of security to encrypt/decrypt data
  - Firewalls and access control – to control access to organization network
  - Internet Security Protocols such as PGP, SSL, TSL, and IPSec
  - Virtual Private Networks – an organization uses the services of the Internet to connect their private networks as if the Internet were a virtual private WAN

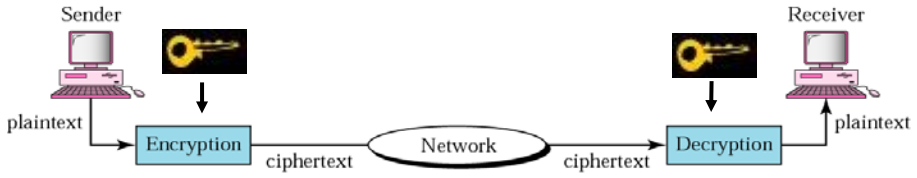
Components of Message Security

- The security measures applied to each single message include
  - privacy (confidentiality): only the indented user can understand the message content by exchanging encrypted messages
    - sender encrypts message
    - receiver decrypts message
  - authentication: sender and receiver confirm identity of each other (e.g. using a technique called digital signature)
  - message integrity: ensure message is not altered (in transit, or afterwards) without detection (e.g. using digital signature)
  - Non-repudiation: to be able prove that a received message came from a specific user (the sender must not be able to deny sending); can be performed using digital signature



Cryptography

- Cryptography (data encryption) means secret writing (in Greek); today it refers to the art and science of changing the appearance and representation of messages to make them secure and immune to attacks
- Data encryption can provide a basic level of privacy to sensitive data sent by your application (although it is not safe 100%)
- Before sending the data, encrypt it. After receiving, decrypt it.
- Many encryption algorithms are in use; they fall into two categories: symmetric data encryption (also called private key encryption or shared key encryption) and asymmetric data encryption (public key encryption)

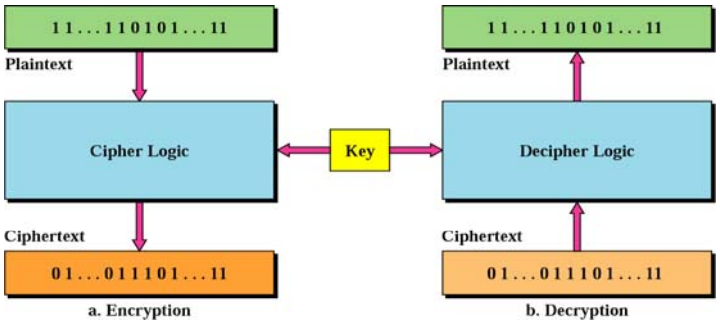


Symmetric Key Encryption

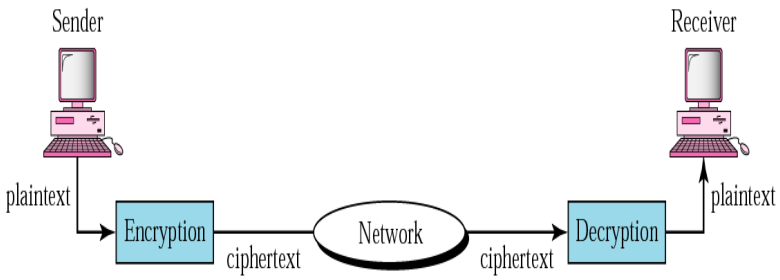
- Use a single private key to encrypt and decrypt a message
- Symmetric encryption algorithms encrypt data in blocks
- Symmetric-key cryptography is often used for large messages.
- **Advantages** – more efficient, less time to encrypt, smaller (size of) keys
- **Disadvantages** – required large number of private keys and difficult distribution of the keys

Symmetric Key Encryption …

- Block encryption/decryption



Asymmetric Key Encryption



- Each application has two keys: one public and one private.
- If application A wants to send a message to B, A uses the public key of B to encrypt the message. Then B uses its private key to decrypt the message.
- One common application of asymmetric encryption is to encrypt the symmetric private key as it is sent to the remote machine. Then, use symmetric key to encrypt data.

Using Data Encryption in .NET Framework

- Lots of classes for encrypting data can be found in **System.Security.Cryptography** namespace
- .NET Symmetric Encryption Classes

Class	Description
DESCryptoServiceProvider	Implements the DES encryption algorithm
RC2CryptoServiceProvider	Implements RC2 encryption algorithm
RijndaelManaged	Implements Rijndael Managed encryption algorithm
TripleDESCryptoServiceProvider	Implements the triple DES encryption algorithm

- .NET Asymmetric Encryption Classes

Class	Description
DSACryptoServiceProvider	Implements the DSA encryption algorithm
RSACryptoServiceProvider	Implements the RSA encryption algorithm

Encrypting Data

- The .NET symmetric encryption classes pass data in streams.
- Assume we want to use the TripleDESCryptoServiceProvider for encryption and decryption
- The CryptoStream class passes the blocks of encrypted data through to a Stream object. The stream can be anything including FileStream, MemoryStream, and NetworkStream.
- The CryptoStream constructor requires three parameters:

```
CryptoStream(Stream stream,
    ICryptoTransform transform,
    CryptoStreamMode mode)
```

Encrypting Data …

- *stream*,
  - represents the stream of data to encrypted to or decrypted from.
- *transform*,
  - represents the encryption/decryption algorithms. To create this value, you must use appropriate method of the TripleDESCryptoServiceProvider class

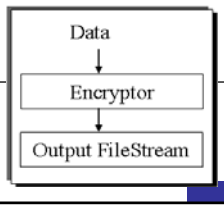
    CreateEncryptor(Key *key*, IV *iv*)

    CreateDecryptor(Key *key*, IV *iv*)
  - To ensure that each block of encrypted data is unique, a second data value called initialization vector (IV) is used along the private key.
  - Both of these values are 16-byte arrays, which must be the same for both sides of the encryption transaction.
  - Often these values can be transferred in a secure snail-mail message or via a separate encrypted message.
- *mode*,
  - defines whether the CryptoStream will write to the associated *stream* (CryptoStreamMode.Write) or read from it (CryptoStreamMode.Read)

Encrypting Data …

- Sample code snippet for encrypting data with symmetric encryption:

```
FileStream fs =
  new FileStream("test.enc", FileAccess.Create);
TripleDESCryptoServiceProvider tdes =
  new TripleDESCryptoServiceProvider();
CryptoStream csw = new CryptoStream(fs,
        Tdes.CreateEncryptor(key, iv),
        CryptoStreamMode.Write);
byte[] data =
   Encoding.ASCII.GetBytes("Phrase to encrypt");
csw.Write(data, 0, data.Length);
csw.Close();
fs.Close();
```



Decrypting Data

- The decryption process works similar to the encryption process: data is passed through the decryptor stream to a specified stream object.

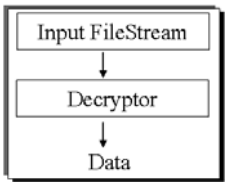
```
CryptoStream csr =
  new CryptoStream(
    memstrm,
    tdes.CreateDecryptor(key, iv),
    CryptoStreamMode.Read);
```


Decrypting Data …

- A simple example of symmetric decryption looks like this:

```
1.  FileStream fs = new FileStream("test.enc", FileAccess.Open);
2.  TripleDESCryptoServiceProvider tdes =
3.        new TripleDESCryptoServiceProvider();
4.  CryptoStream csr = new CryptoStream(fs,
5.      tdes.CreateDecryptor(key, iv),
6.          CryptoStreamMode.Read);

7.  byte[] data = new byte[1024];
8.  int recv = csr.Read(data, 0, data.Length);
9.  string phrase = Encoding.ASCII.GetString(data, 0, recv);
10. csr.Close();
11. fs.Close();
```



Example 1

```
using System;
using System.IO;
using System.Security;
using System.Security.Cryptography;
using System.Text;
namespace TestEncryption {
    class Program {
        static void Main(string[] args) {
            Console.Write("Enter phrase to encrypt: ");
            string phrase = Console.ReadLine();
            MemoryStream memstrm = new MemoryStream();
            byte[] Key =
                   Encoding.ASCII.GetBytes("MAASSALAAMSHABAB");
            byte[] IV =
                   Encoding.ASCII.GetBytes("abcdefgh12345678");
            TripleDESCryptoServiceProvider tdes =
                          new TripleDESCryptoServiceProvider();
            CryptoStream csw = new CryptoStream(memstrm,
                               tdes.CreateEncryptor(Key, IV),
                               CryptoStreamMode.Write);
            csw.Write(Encoding.ASCII.GetBytes(phrase), 0,
                                              phrase.Length);
            csw.FlushFinalBlock();
```


Example 1 …

```
        byte[] cryptdata = memstrm.GetBuffer();
            Console.WriteLine("Encrypted: {0}",
                              Encoding.ASCII.GetString(cryptdata, 0,
                              (int)memstrm.Length));

            memstrm.Position = 0; //reset the memory stream
            byte[] data = new byte[1024];
            CryptoStream csr = new CryptoStream(memstrm,
                                   tdes.CreateDecryptor(Key, IV),
                                   CryptoStreamMode.Read);
            int recv = csr.Read(data, 0, data.Length);
            string newphrase = Encoding.ASCII.GetString(data,0,recv);
            Console.WriteLine("Decrypted: {0}", newphrase);

            csr.Close();
            csw.Close();
            memstrm.Close();
            Console.ReadLine();
        }
    }
}
```

```
Enter phrase to encrypt: salam shabab
Encrypted: ??gfj??↑?♦z??G??
Decrypted: salam shabab
```

Network Data Encryption

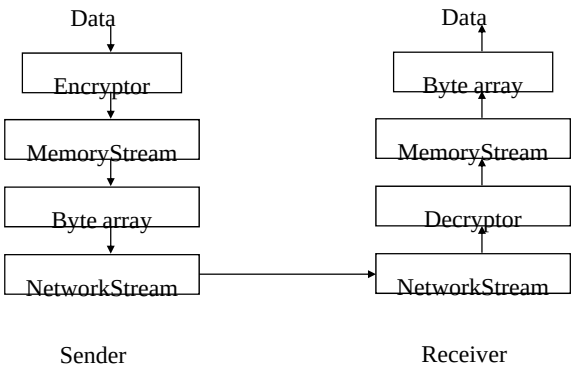
- It is convenient to use the CryptoStream class with FileStream and MemoryStream objects,
- But it is often difficult to use it with NetworkStream objects due to lack of data boundaries within the data stream
  - When data is encrypted and passed directly to a NetworkStream object, there is no easy way of determining the end of the data.
  - If the decryptor attempts to decrypt the data before the end of the stream has arrived, an error will occur.
- The easiest solution is to determine the size of the encrypted message and send it before the actual message.
- The receiving program can extract the message size from the NetworkStream and will know how many bytes of data to read from the network to properly form the message.

Network Data Encryption …

- How to determine the size of the encrypted message?
  - the message must be stored in a buffer after being encrypted and before being sent out on the NetworkStream.
  - Once the encrypted message is in the buffer, it's easy to determine its size and send both the size and the message to the NetworkStream.
- A simple appraoch to achieve this
  - use the MemoryStream class, as demonstrated in the last example.
  - By forwarding the CryptoStream output to a MemoryStream, you can use the GetBuffer() method to create a byte array of the encrypted message.
  - The array can then be sent out on the network just like any other byte array of data.
- On the receiver side, the opposite steps are performed.
  - Data read from the NetworkStream is fed for decryption into the CryptoStream object, which points to a MemoryStream object.
  - Once the entire decrypted message is in the MemoryStream object, it can be extracted using the GetBuffer() method.

Network Data Encryption…

- Encrypting/Decrypting data for the network



Example 2

```
1.  using System;
2.  using System.IO;
3.  using System.Net;
4.  using System.Net.Sockets;
5.  using System.Security;
6.  using System.Security.Cryptography;
7.  using System.Text;
8.  namespace CryptoSender {
9.      class Program {
10.         static void Main(string[] args) {
11.             TcpClient client = new TcpClient("127.0.0.1", 9050);
12.             NetworkStream stream = client.GetStream();
13.             byte[] Key = Encoding.ASCII.GetBytes("MAASSALAAMSHABAB");
14.             byte[] IV = Encoding.ASCII.GetBytes("abcdefgh12345678");
15.             String msg;
16.             do {
17.                 Console.Write("Enter message to send <Enter> to quit: ");
18.                 msg = Console.ReadLine();
19.                 SendData(stream, Encoding.ASCII.GetBytes(msg), Key, IV);
20.             } while (msg != "");
21.             stream.Close();
22.             client.Close();
23.         }
```

Example 2 …

```
24.         private static void SendData(NetworkStream netStream,
25.                                      byte[] data, byte[] Key, byte[] IV
26.         {
27.             MemoryStream memStream = new MemoryStream();
28.             TripleDESCryptoServiceProvider tdes =
29.                                 new TripleDESCryptoServiceProvider();
30.             CryptoStream csw = new CryptoStream(memStream,
31.                                    tdes.CreateEncryptor(Key, IV),
32.                                    CryptoStreamMode.Write);
33.             csw.Write(data, 0, data.Length);
34.             csw.FlushFinalBlock();
35.             byte[] encyptedData = memStream.GetBuffer();
36.             int encyptedDataSize = (int)memStream.Length;
37.             byte[] size = BitConverter.GetBytes(encyptedDataSize);
38.             netStream.Write(size, 0, size.Length);
39.             netStream.Write(encyptedData, 0, encyptedDataSize);
40.             netStream.Flush();
41.             csw.Close();
42.             memStream.Close();
43.         }
44.     }
45. }
```


Example 2 …

```
1.  using System;
2.  using System.IO;
3.  using System.Net;
4.  using System.Net.Sockets;
5.  using System.Security;
6.  using System.Security.Cryptography;
7.  using System.Text;
8.  namespace CryptoReceiver {
9.      class Program (
10.         static void Main(string[] args) {
11.             TcpListener server = new TcpListener(IPAddress.Any, 9050);
12.             server.Start();
13.             Console.WriteLine("Waiting for a client...");
14.             TcpClient client = server.AcceptTcpClient();
15.             NetworkStream stream = client.GetStream();
16.             byte[] Key = Encoding.ASCII.GetBytes("MAASSALAAMSHABAB");
17.             byte[] IV = Encoding.ASCII.GetBytes("abcdefgh12345678");
18.             byte[] data; int size;
19.             while (true) {
20.                 size = ReceiveData(stream, Key, IV, out data);
21.                 if (size == 0) break;
22.                Console.WriteLine(Encoding.ASCII.GetString(data, 0, size));
23.             }
24.             stream.Close(); server.Stop(); }
```

Example 2 ...

```
25.         private static int ReceiveData(NetworkStream netStream,
26.                         byte[] Key, byte[] IV, out byte[] result) {
27.             MemoryStream memStream = new MemoryStream();
28.             TripleDESCryptoServiceProvider tdes =
29.                             new TripleDESCryptoServiceProvider();
30.             CryptoStream csw = new CryptoStream(memStream,
31.                                 tdes.CreateDecryptor(Key, IV),
32.                                 CryptoStreamMode.Write);
33.             byte[] data = new byte[2048];
34.             netStream.Read(data, 0, 4); //read the size
35.             int size = BitConverter.ToInt32(data, 0);
36.             int sofar = 0, recv;
37.             while (sofar < size) {
38.                 recv = netStream.Read(data, 0, data.Length);
39.                 csw.Write(data, 0, recv);
40.                 sofar += recv;
41.             }
42.             csw.FlushFinalBlock();
43.             result = memStream.GetBuffer();
44.             return (int)memStream.Length;
45.         }
46.     }
47. }
```


Resources

- MSDN Library
  - http://msdn.microsoft.com/en-us/default.aspx
- Books
  - Richard Blum, C# Network Programming. Sybex 2002.
- Lecture notes of previous offerings of SWE344 and ICS343
- Some other web sites and books; check the course website at
  - http://faculty.kfupm.edu.sa/ics/alfy/files/teaching/swe344/index.htm