# Cryptography

## Cryptography

- .NET Framework provides the System.Security.Cryptography namespace which provides class support for :
- the most important symmetric and asymmetric ciphers
- several secure hash algorithms
- a cryptographic quality pseudo random number generator. (PRNG)

## **Symmetric Cryptography**

- Major players here are
- DES (Digital Encryption Standard)
- Triple DES
- AES (Advanced Encryption Standard) aka Rijndael
- Uses a secret key to generate the cipher text.
- Both sender and receiver must have the key.
- Good for large blocks of data.
- Much faster than asymmetric ciphers

## **Asymmetric Cryptography**

- Uses modular arithmetic and simple number theory to make it possible to construct two distinct keys.
- One key is used for encryption, the other for decryption. Together they form a key pair.
- Although mathematically related to one another, it is exceedingly difficult to construct one from the other.
- One key (for encrypting) is made public. The other (decrypter key) is kept a closely guarded secret.
- Asymmetric cryptography is based on the idea of a one-way function that has a trap door.

## Asymmetric Cryptography

- Major player here today is RSA (Rivest, Shamir, Adleman)
- Much, much slower (1000 times) than symmetric.
- Typically used on very small pieces of plain text.

## **PRNG and the .NET Framework**

- Standard random number generators provided in Windows and UNIX (rand and srand) are not of a high enough quality to be used for cryptographic purposes.
- .NET's Random class is also not good enough.
- The framework provides the RNGCryptoServiceProvider class that delivers a high quality PRNG
- We use its GetBytes() and GetNonZeroBytes() methods which return an array of bytes containing a cryptographically strong random sequence of values.

## Cryptographic Hash Algorithms

- Takes an arbitrary amount of input data and reduces it to a fixed length (typically 128, 160, or 256 bits) output
- Output often referred to as **a message digest** or a **fingerprint**, and it is highly characteristic of the input data, just as a fingerprint is characteristic of a human.

## Cryptographic Hash Algorithms

- Ideally, a cryptographic hash function satisfies these requirements:
- it is one-way (can't get input from output)
- difficult to find two inputs that will generate same output
- difficult to find an input that will generate a particular output
- a single bit change on input changes ~ 50% of output bits

## Cryptographic Hash Algorithms

- Hash outputs can be used for:
- detecting modifications of data
- play a role in implementing digital signatures
- password transformation
- creating encryption keys

## Symmetric Cryptography

- Uses one key to do encryption and decryption.
- Thus, critical to keep it secret.
- Along with the key, Bob and Alice have to agree on
- which symmetric algorithm to use
- an initialization vector
- mode of operation
- padding convention (bytes used to pad blocks less than 64)

## Symmetric Cryptography

- You will find 2 basic categories of symmetric ciphers:
- Block ciphers that operate on 64 or 128 bit blocks of data (more secure)
- Stream ciphers that operate on a single byte (or even a bit) at a time (faster)
- The distinction is not that important because certain modes of operation allow a block cipher to behave as a stream cipher.

## DES

- Widely used symmetric cipher which used to be the US government standard.
- It has been cracked (56 hour brute force attack in 1998 and again in 22 hours with same machine working with 100,000 pcs on the internet) and has now been replaced with the Rijndel algorithm for US government.
- DES, however, is still quite fine for keeping secret all but the most sensitive info.

## Modes of Operation

- All block ciphers have to operate within the context of certain rules, and these are referred to as the modes of operation.
- The Cryptography namespace in .NET contains an enumeration called CipherMode that contains 5 modes.
- These modes describe different ways of actually doing the 16 rounds of permutation. You can look up Cipher Modes in VS.NET doc and read about them.

## Triple DES

- Adopted by the banking industry, it is a more powerful alternative to DES.
- Improvement comes from the fact that each 64 bit block of plaintext is encrypted three times, using the DES algorithm in which 3 distinct keys are used.
- Naturally, this method takes 3 times as long as DES.

## Rijndael (AES)

- Now the official US government standard.
- It can use key sizes of 128, 192, or 256 bits and work on data blocks of those sizes.
- Depending on key size, it will use 10, 12, or 14 rounds of transformation and permutation.
- Algorithm is very complex and is based on advanced algebra and polynomials.

## RC2

- Invented by RSA Data Security Incorporated, it was a temporary replacement for DES.
- It can use a variable key size (1 – 128 bits) and can be made more secure than DES with good key choices.
- Runs about twice as fast as DES.

## Programming Symmetric

- Base abstract class is *SymmetricAlgorithm*.
- Descending from it are other abstract classes named
- DES TripleDES Rijndael and RC2
- and each of these have concrete classes that descend from them. The concrete classes are the ones we use. These are:
- DESCryptoServiceProvider
- TripleDESCryptoServiceProvider
- RC2CryptoServiceProvider
- RijndaelManaged

## Programming symmetric

- Parties involved have to use the same
- Key
- IV (initialization vector)
- Mode of operation
- Padding byte

## Programming Symmetric

- DESCryptoServiceProvider csp = new DESCryptoServiceProvider();
- This default constructor will generate for you a KEY and IV, a padding byte (PKCS7) and a mode (CBC)
- You can access these via public properties csp.Key and csp.IV (get/set)
- byte[] Key = csp.Key; byte[] IV = csp.IV;
- Or you can generate a new Key and IV with:
- byte[] Key = csp.GenerateKey()
- byte[] IV = csp.GenerateIV();

## Programming Symmetric

- You can also call a static method of the DES class to return a concrete instance for you:
- DES des = DES.Create();
- This is true for all of the 4 symmetric types.
- SymmetricAlgorithm sa = Rijndael.Create();

## ASYMMETRIC CRYPTOGRAPHY

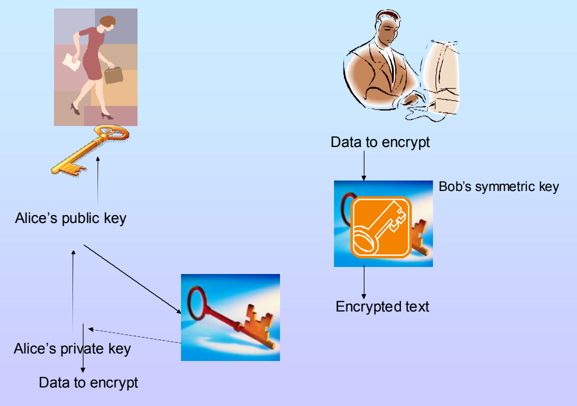
- Basic motivation for its development was the problems of the sharing of secret keys.
- These techniques use a public and private key pair that are mathematically related.
- I let anyone use my public key to encrypt data that they send to me.
- Only my private key (which only I have) can decrypt the data.

## ASYMMETRIC CRYPTOGRAPHY

- As mentioned previously, these algorithms are based on trap door one way functions.
- A one way function is a mathematical function that is highly asymmetric in terms of its computational complexity with respect to its inverse function.
- It is easy to compute in its forward direction but extremely difficult to compute in the reverse direction.
- But with a trap door, you have an additional piece of information that allows you to easily compute the reverse direction.
- There are many one way functions to choose from in mathematics, but finding one that allows you to incorporate the all important trap door is not so easy.

## ASYMMETRIC CRYPTOGRAPHY

- It is tempting to just say “OK, let’s just always use asymmetric algorithms and we don’t have any problems”
- But this would abandon the superior speed and security offered by symmetric algorithms, so one common protocol combines the 2 classes like this.



## ASYMMETRIC CRYPTOGRAPHY

- There are several asymmetric algorithms in existence today, including
- RSA (Rivest, Shamir, Adleman) – can be used both for confidentiality and for digital signatures
- DSA (Digital Signature Algorithm) – can be used for digital signatures but not confidentiality
- ElGanal – not supported by .NET -- does both
- ECC (Elliptic Curve Cryptography) – not supported by .NET

## ASYMMETRIC CRYPTOGRAPHY

- Again, these techniques are much slower and less secure than symmetric algorithms.
- Should use a large key size to be effective and should only encrypt small amounts of data.