Module 18

"Cryptography"

TEKNOLOGISK
INSTITUT

TEKNOLOGISK
INSTITUT

Agenda

- Introducing Cryptography
- Symmetric Encryption
- Asymmetric Encryption
- Hashing
- Digital Signatures

TEKNOLOGISK
INSTITUT

Introducing Cryptography

- The discipline of
  - Encrypting
  - Decrypting
  - Hashing
  - Signing
- Mathematically sound methods and functions are devised for computers to make use of
- General principles matured through decades
  - In no way specific to .NET

TEKNOLOGISK
INSTITUT

Introducing Encryption

- Encryption
  - Encodes data from Sender to Receiver with a key
- Decryption
  - Decodes data Received from Sender with a key
- Encryption/Decryption algorithms are *ciphers*

- Symmetric encryption
  - Sender and Receiver shares secret key established upfront

- Asymmetric encryption (a.k.a. Public-Key encryption)
  - Receiver has a key pair
    - Public key used for encryption by Sender (and everyone else)
    - Private key used for decryption by Receiver (only)

Agenda

TEKNOLOGISK INSTITUT

- Introducing Cryptography
- **Symmetric Encryption**
- Asymmetric Encryption
- Hashing
- Digital Signatures

Symmetric Algorithms Keys

TEKNOLOGISK INSTITUT

- Encryptor and Decryptor share
  - Key (often derived from password + salt)
  - InitializationVector (IV)

- `SymmetricAlgorithm` properties
  - `Key`
  - `IV`
  - `KeySize`
  - `BlockSize`
  - `FeedBackSize`
  - …

```
SymmetricAlgorithm alg = new RijndaelManaged();
byte[] salt =
   Encoding.ASCII.GetBytes( "This is my sa1t" );
Rfc2898DeriveBytes key =
   new Rfc2898DeriveBytes( password, salt );
alg.Key = key.GetBytes( alg.KeySize / 8 );
alg.IV = key.GetBytes( alg.BlockSize / 8 );
```

- `Key` and `IV` could be randomly generated

Encryption/Decryption using Symmetric Algorithms

TEKNOLOGISK INSTITUT

- 1. Create **`Stream`** object to write to/read from
- 2. Create **`SymmetricAlgorithm`** as on previous slide
  - Set **`Key`** or **`IV`** or both
- 3. Obtain **`ICryptoTransform`** object via either
  - **`SymmetricAlgorithm.CreateEncryptor()`**
  - **`SymmetricAlgorithm.CreateDecryptor()`**
- 4. Create **`CryptoStream`** from
  - **`Stream`**
  - **`ICryptoTransform`**
- 5. Write to/read from **`CryptoStream`** to encrypt/decrypt

TEKNOLOGISK
INSTITUT

Symmetric Algorithms

- Symmetric algorithms
  - RijndaelManaged
  - AesManaged
  - Data Encryption Standard (DES)
  - TripleDES
  - RC2

- `ICryptoTransform` under the hood
  - `TransformBlock()` Encrypt or decrypt part
  - `TransformFinalBlock()` Encrypt or decrypt entire block

- Exam tip
  - Never change any of the default properties (except maybe Key + IV)

Agenda

- Introducing Cryptography
- Symmetric Encryption
- **Asymmetric Encryption**
- Hashing
- Digital Signatures

Asymmetric Algorithms

TEKNOLOGISK INSTITUT

- Usually much slower but safer than symmetric versions
- Public key must be exchanged upfront for Sender to be able to encrypt to Receiver

- Asymmetric algorithms include
  - RSA `RSACryptoServiceProvider`
  - Digital Signature Algorithm `DSACryptoServiceProvider`

- Asymmetric algorithms
  - Derive from **`AsymmetricAlgorithm`** class

TEKNOLOGISK INSTITUT

Assymmetric Encryption Keys

- Asymmetric keys are much more complex
  - RSA supports stored with CSP via CryptoAPI
    - `CspParameters` class
    - `RSACryptoProvider.PersistKeyInCsp` property

```
CspParameters persistantCsp = new CspParameters();
persistantCsp.KeyContainerName = "AsymmetricExample";

// ...

RSACryptoServiceProvider myRSA =
   new RSACryptoServiceProvider( persistantCsp );
myRSA.PersistKeyInCsp = true;

RSAParameters privateKey =
myRSA.ExportParameters(true);
```

Encryption/Decryption using Asymmetric Algorithms

TEKNOLOGISK INSTITUT

- 1. Create `RSACryptoProvider` object
- 2. Get/set the key with
  - `RSACryptoProvider.ExportParameters()`
  - `RSACryptoProvider.ImportParameters()`
- 3. Convert string to bytes or bytes to string
  - if necessary
- 4. Encrypt/decrypt using
  - `RSACryptoProvider.Encrypt()`
  - `RSACryptoProvider.Decrypt()`
- 5. Convert bytes to string or string to bytes
  - If necessary

Agenda

TEKNOLOGISK INSTITUT

- Introducing Cryptography
- Symmetric Encryption
- Asymmetric Encryption
- **Hashing**
- Digital Signatures

TEKNOLOGISK
INSTITUT

Introducing Hashing

- Hashing is the discipline of calculating a unique partial checksum
  - Cryptographically secure hash functions
  - Inherently a one-way process
  - Hash values always have the same length (small)

- Hashing algorithms are usually used to detect
  - Whether data has changed
  - Whether data has been tampered with

- Hashing algorithms can be keyed or non-keyed

TEKNOLOGISK INSTITUT

Non-keyed Hashing

- Non-keyed hashing algorithms
  - Derive from **`HashAlgorithm`**

- Non-keyed hashing algorithms include
  - MD5 **`MD5CryptoServiceProvider`**
  - RIPEMD160 **`RIPEMD160Managed`**
  - SHA1 **`SHA1CryptoServiceProvider`**
  - SHA*x* **`SHAxManaged`** (x = 256, 384, or 512)

Hashing using Non-keyed Hashing Algorithms

TEKNOLOGISK INSTITUT

- 1. Create the **`HashAlgorithm`** object
- 2. Convert data to be hashed to byte array
- 3. Invoke **`HashAlgorithm.ComputeHash()`** passing the byte array
- 4. Retrieve the byte array constituting the hash value using the **`HashAlgorithm.Hash`** property
- Note
  - **`ComputeHash()`** in fact returns the hash value
  - Use **`ComputeHash()`** first in order to use the **`Hash`** property afterwards

TEKNOLOGISK
INSTITUT

Keyed Hashing

- Keyed hashing algorithms
  - Derive from **`KeyedHashAlgorithm`**
- Keyed hashing algorithms include
  - HMAC using SHA1 **`HMACSHA1`**
  - MAC using TripleDES **`MACTripleDES`**
- Exam tip:
  - H ~ Hashing
  - MAC ~ Message Authentication Codes

Hashing using Keyed Hashing Algorithms

TEKNOLOGISK INSTITUT

- 1. Create the **`KeyedHashAlgorithm`** object. Either
  - Pass the key to the constructor, or
  - Use the default created random key
- 2. Convert data to be hashed to byte array
- 3. Invoke **`KeyedHashAlgorithm.ComputeHash()`** passing the byte array
- 4. Retrieve the byte array constituting the hash value using the **`KeyedHashAlgorithm.Hash`** property

Agenda

TEKNOLOGISK INSTITUT

- Introducing Cryptography
- Symmetric Encryption
- Asymmetric Encryption
- Hashing
- **Digital Signatures**

Digital Signatures

TEKNOLOGISK INSTITUT

- Digital signatures provide proof that data was sent by a specific sender
  - They do not provide encryption!

- Digital signatures use asymmetric key pairs
  - Sender can sign by
    - Computing a hash value for the data
    - Encrypting the hash value using Sender's private key
    - Sending the message along with the signature = encrypted hash
      - public key is publicly accessible
  - Receiver can verify signature by
    - Computing the hash from the received data
    - Decrypting the signature using Sender's public key
    - Compare the computed and the received hash values

Digital Signature Methods

TEKNOLOGISK INSTITUT

- Can use
  - `RSACryptoServiceProvider`
  - `DSACryptoServiceProvider`

- Methods
  - **`SignHash()`** : Generates a digital signature from a hash of data
  - **`SignData()`** : Generates a digital signature from the data itself
    - = Generate hash **`+ SignHash()`**

  - **`VerifyHash()`** : Verifies a digital signature from a hash of data
  - **`VerifyData()`** : Verifies a digital signature from data itself
    - = Generate hash + **`VerifyHash()`**

Digitally Signing Data

TEKNOLOGISK INSTITUT

- 1. Create the digital signature algorithm object
- 2. Store the data to be signed in a byte array
- 3. Call **`AsymmetricAlgorithm.SignData()`** and store the signature return value
- 4. Export the public key
  - **`AsymmetricAlgorithm.ToXmlString( false )`**

TEKNOLOGISK INSTITUT

Verifying a Digital Signature

- 1. Create the digital signature algorithm object
- 2. Import the signature and public key
  - `AsymmetricAlgorithm.FromXmlString( false )`
- 3. Store the data to be verified in a byte array
- 4. Call the `AsymmetricAlgorithm.VerifyData()` passing
  - the byte array
  - signature

TEKNOLOGISK
INSTITUT

Summary

- Introducing Cryptography
- Symmetric Encryption
- Asymmetric Encryption
- Hashing
- Digital Signatures

Question

TEKNOLOGISK INSTITUT

You are developing an application which will transfer a number of data packages from a client to a server computer.

You need to ensure that validity of the data by using a cryptographic hashing algorithm.

Which algorithm should you use?

a) `AesManaged`
b) `RNGCryptoServiceProvider`
c) `HMACSHA512`
d) `TripleDESCryptoServiceProvider`

TEKNOLOGISK
INSTITUT