



Verschlüsselte Netzwerkkommunikation

- C# CryptoAPI für Dummies -

Gliederung



- Warum verschlüsseln?
- Was für Angriffe gibt es?
- Wie verschlüsselt man?
- Was gehört zur C# Crypto-API?
- Wie setzt man das in Code um?
- Beispiel-Klasse
- Zusammenfassung

Warum verschlüsseln?



- Schutz von Kunden- und Spieldaten
- Bezahlungssysteme implementieren
- Multiplayer Kommunikation gegen Betrug absichern

Was für Angriffe gibt es?



Unverschlüsselte Netzwerk-Pakete:

- Fremde Pakete abfangen und auslesen (Durch Wände schauen)
- Fremde Pakete abfangen und editieren (Aus 2 Gold wird 2000 Gold)
- Gefälschte Pakete senden (Alles geht...)

Was für Angriffe gibt es?



Verschlüsselte Netzwerk-Pakete:

- Angriffe auf das Passwort (Bruteforce, Dictionary)
- Angriffe auf den Algorithmus (Cryptanalysis, Implementation)
- Aufzeichnen und erneut schicken (Replay Attack)
- Überfluten mit Informationen (Denial of Service)

Wie verschlüsselt man?



Lesen der Netzwerkpakete verhindern: **Verschlüsselung**

- Verhindern, dass verschlüsselte Pakete vom Angreifer gelesen werden
- Verhindern, dass Pakete vom Angreifer frei editiert werden können
- Gefälschte Pakete erkennen und aussortieren

Wie verschlüsselt man?



Überprüfen der Netzwerkpakete auf: Zeitpunkt

- Nummerierung / Timestampen
 - Verhindern, dass verschlüsselte Pakete vom Angreifer nochmal gesendet werden
 - Verhindern, dass verschlüsselte Pakete abgefangen und zu einem anderen Zeitpunkt gesendet werden

Wie verschlüsselt man?



Überprüfen der Netzwerkpakete auf: **Herkunft**

- Digitale Unterschrift / Signing
 - Verhindern, dass verschlüsselte Pakete ohne Entschlüsselung editiert werden
 - Sicherstellen, dass der Urheber eines Pakets das Versenden nicht leugnen kann

Wie verschlüsselt man?



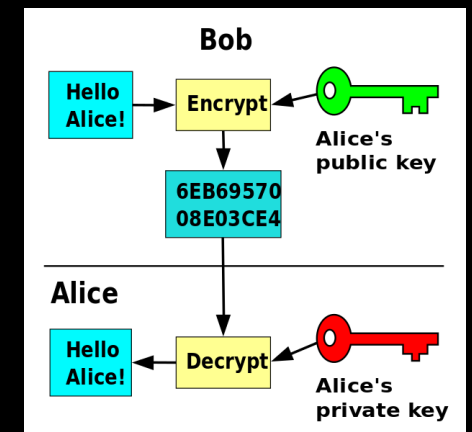
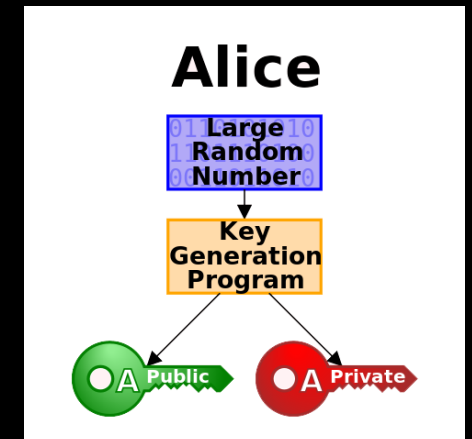
Asymmetrische Verschlüsselungen (RSA)

Zweck:

- Verschlüsselung von kleinen Datenmengen wie Passwörtern (**LANGSAM**)

Wie funktioniert:

- Verschlüsselung und Entschlüsselung mit zwei verschiedenen Schlüsseln (z.B. 4096 Bit)
- Der offene Teil des Schlüssels kann jederzeit der Gegenseite geschickt werden
- Der offene Teil des Schlüssels verschlüsselt, der geheime Teil des Schlüssels entschlüsselt



Wie verschlüsselt man?



Symmetrische Verschlüsselungen (AES / Rijndael, RC2, DES)

Zweck:

- Verschlüsselung von großen Datenmengen (SCHNELL)

Wie funktioniert:

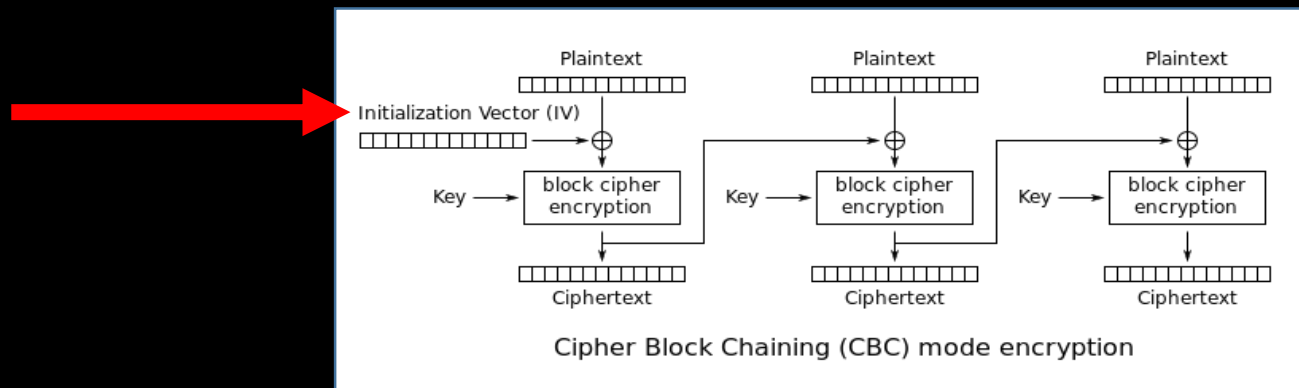
- Verschlüsselung und Entschlüsselung mit einem (1) geheimen Schlüssel (z.B. 256 Bit)
- Der Schlüssel muss irgendwie an die Gegenseite übertragen werden
- Benötigt einen Initial Vector für Cypher Block Chaining

Wie verschlüsselt man?



Cypher Block Chaining

- Aufteilen des Plaintexts in Blöcke (z.B. 16 Bytes)
- Jeder Block wird mit dem Ende des letzten Blocks verschlüsselt
- Der erste Block benötigt einen IV (Initial Vector), da es keinen letzten Block gibt
- Der IV verhindert, dass bei selbem Plaintext und gleichem Password gleicher Ciphertext herauskommt



Wie verschlüsselt man?



Hash Verschlüsselungen (SHA1, MD5, SHA256)

Zweck:

- Sicherstellen der Daten Integrität

Wie funktioniert:

- Konvertiert beliebige Daten in einen Hash bestimmter Größe
- Kleine Änderungen in den Daten verursachen große Änderungen im Hash
- Es ist rechnerisch zu kompliziert Daten zu finden, die denselben Hash haben
- Um Rainbow Table Angriffe zu verhindern benutzt man einen **SALT**

Wie verschlüsselt man?



SALT

- Gegen Rainbow Tables: Tabellen mit Hashes für bekannte Passwörter
- Eine zufällige Zeichenfolge, die an das Passwort angehängt wird
- Erzeugt zusätzliche Komplexität für das Passwort
- Kann zusammen mit den Daten offen übertragen werden oder statisch sein

Was gehört zur C# Crypto-API?



Crypto Algorithms:

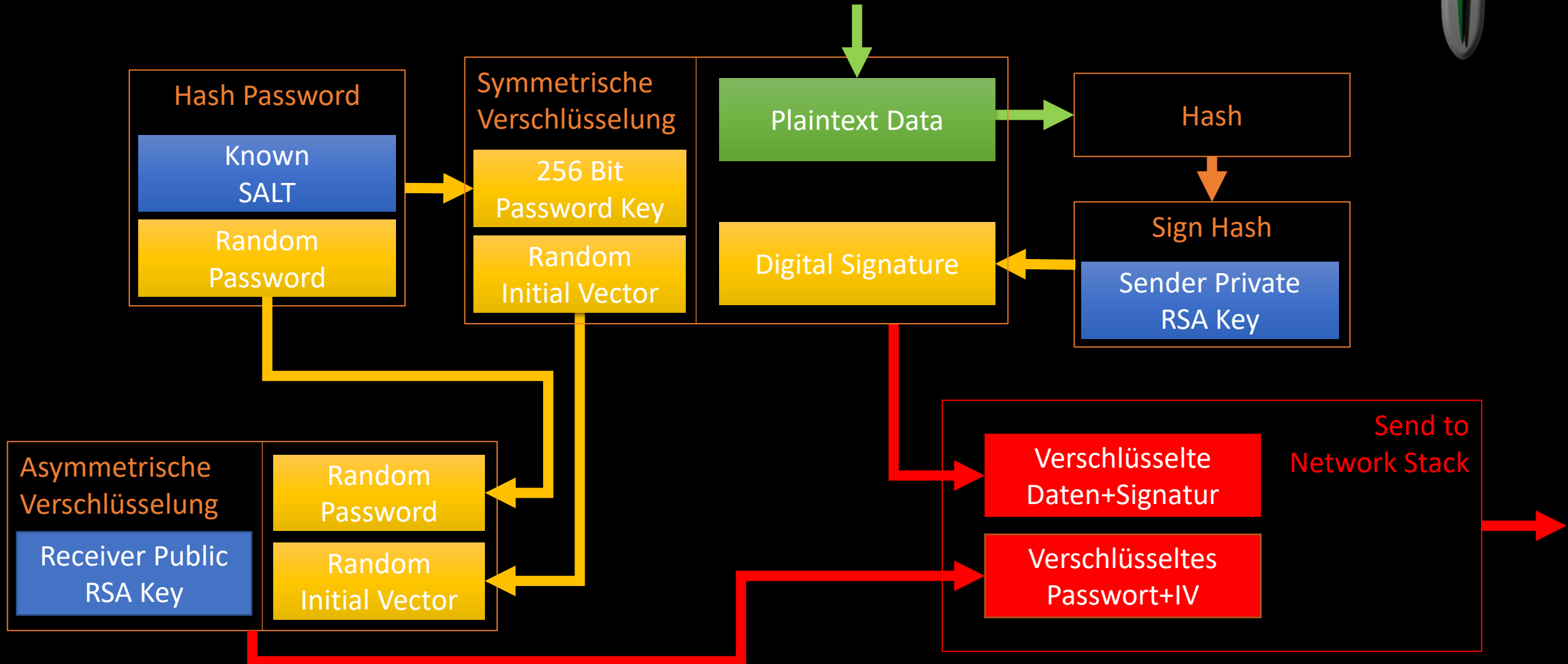
- AsymmetricAlgorithms
 - DSA, ECDiffieHellman, ECDsa, **RSA**
- SymmetricAlgorithms
 - DES, RC2, Rijndael, TripleDES, **AES**

Tools:

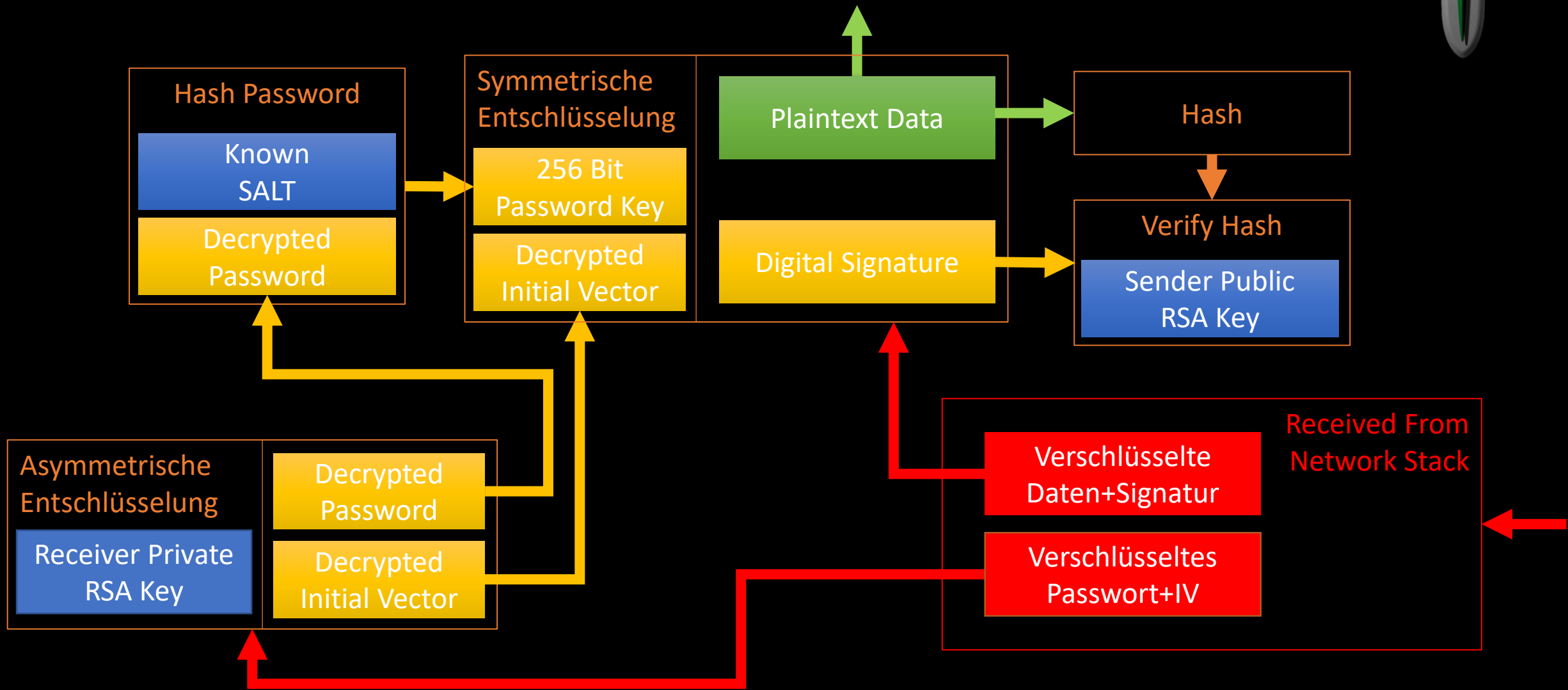
- **Crypto Random Number Generator**
 - Create Random IV, Random Password
- HashAlgorithms
 - MD5, SHA1, RIPEMD160, **SHA256**



Wie setzt man das in Code um?



Wie setzt man das in Code um?



Beispiel-Klasse Übersicht



```
using System;
using System.IO;
using System.Security.Cryptography;
using System.Text;

namespace Tacdev.Crypto
{
    public static class Encryptor
    {
        #region Settings
        private static RNGCryptoServiceProvider rand = new RNGCryptoServiceProvider();
        private static int KeySize = 256; // Increase size as required for your safety concerns
        private static int PasswordLength = 16; // This is not really safe but a tradeoff vs. UDP size
        private static int IVLength = 16; // Increase size as required for your safety concerns
        private static string Hash = "SHA1";
        private static byte[] Salt = new byte[] { 0x53, 0xA8, 0x29, 0x41, 0x12, 0xA3, 0x92, 0x19, 0x15, 0xc3, 0xd1, 0xa1, 0x42, 0x52, 0xe2, 0xf1 };
        #endregion

        /// This is what you call to encrypt a payload. It applies a symmetric encryption to the message and encrypts the password
        /// used for that encryption with the RSA asymmetric cypher
        public static byte[] EncryptPayload(byte[] message, RSACryptoServiceProvider SenderKey, RSACryptoServiceProvider ReceiverKey)
        {
            /// The symmetrical part of the encryption, that returns a password and IV that then can be
            /// encrypted using asymmetrical encryption
            private static byte[] EncryptSymmetric<T>(byte[] value, out byte[] password, out byte[] IV) where T : SymmetricAlgorithm, new()
            {
                /// This is what you call to decrypt a payload. It decrypts the password used for the symmetric encryption using the RSA
                /// asymmetric cypher and then decrypts the message and signature with the the symmetric decryption routine
                public static byte[] DecryptPayload(byte[] message, RSACryptoServiceProvider SenderKey, RSACryptoServiceProvider ReceiverKey)
                {
                    /// The symmetrical part of the decryption, that accepts a password and IV received
                    /// from the asymmetrical decryption
                    private static byte[] DecryptSymmetric<T>(byte[] value, byte[] password, byte[] IV) where T : SymmetricAlgorithm, new()
                    {

```

Beispiel-Klasse



```
/// This is what you call to encrypt a payload. It applies a symmetric encryption to the message and encrypts the password
/// used for that encryption with the RSA asymmetric cypher
public static byte[] EncryptPayload(byte[] message, RSACryptoServiceProvider SenderKey, RSACryptoServiceProvider ReceiverKey)
{
    // We sign the data so the receiver can verify it is from us. The signature is inside the encrypted container.
    byte[] signature = SenderKey.SignData(message, HashAlgorithm.Create("SHA256"));

    // The complete container to be encrypted has the signature and the actual message
    byte[] complete = null;

    // The size of the signature depends on the key size so we store its length to get it back out
    using (MemoryStream ms = new MemoryStream())
    {
        using (BinaryWriter binWriter = new BinaryWriter(ms))
        {
            binWriter.Write(signature.Length);
            binWriter.Write(signature);
            binWriter.Write(message.Length);
            binWriter.Write(message);
            complete = ms.ToArray();
        }
    }

    // Max length of RSA encryption = [ReceiverKey.KeySize / 8 - 2 - (2 * SHA1 Hash Size = 20)]: 1024 bit key = 86 bytes data

    // We encrypt with a symmetric cypher and get back a random password and IV that have to be encrypted with the asymmetric cypher
    byte[] RandomPasswordFromAES;
    byte[] RandomIVFromAES;
    byte[] encPayload = EncryptSymmetric<AesManaged>(complete, out RandomPasswordFromAES, out RandomIVFromAES);
}
```

Beispiel-Klasse



```
// Max length of non-encrypted (ReceiverKeySize / 8) = (8 - 1) * 1024 * 1024 / 8 = 1024 * 1024 / 8 = 128 * 1024 = 131072 bytes data

// We encrypt with a symmetric cypher and get back a random password and IV that have to be encrypted with the asymmetric cypher
byte[] RandomPasswordFromAES;
byte[] RandomIVFromAES;
byte[] encPayload = EncryptSymmetric<AesManaged>(complete, out RandomPasswordFromAES, out RandomIVFromAES);

// We assemble everything that has to be asymmetrically encrypted.... the password and the IV of the symmetric encryption
complete = new byte[RandomPasswordFromAES.Length + RandomIVFromAES.Length];
Buffer.BlockCopy(RandomPasswordFromAES, 0, complete, 0, RandomPasswordFromAES.Length);
Buffer.BlockCopy(RandomIVFromAES, 0, complete, RandomPasswordFromAES.Length, RandomIVFromAES.Length);

// The actual asymmetrical encryption
byte[] encAESKey = ReceiverKey.Encrypt(complete, true);

// Assemble encrypted key and encrypted container into byte array
using (MemoryStream ms = new MemoryStream())
{
    using (BinaryWriter binWriter = new BinaryWriter(ms))
    {
        binWriter.Write(encPayload.Length);
        binWriter.Write(encPayload);
        binWriter.Write(encAESKey.Length);
        binWriter.Write(encAESKey);
        complete = ms.ToArray();
    }
}

return complete;
}
```

Beispiel-Klasse



```
/// The symmetrical part of the encryption, that returns a password and IV that then can be
/// encrypted using asymmetrical encryption
private static byte[] EncryptSymmetric<T>(byte[] value, out byte[] password, out byte[] IV) where T : SymmetricAlgorithm, new()
{
    // Random IV
    IV = new byte[IVLength];
    rand.GetBytes(IV);

    // Random Password
    password = new byte>PasswordLength; // Sorry UDP size restrictions. Its not a nuclear factory.
    rand.GetBytes(password);

    byte[] encrypted;

    using (T cipher = new T())
    {
        // From the password we derive a key with the correct length.... the password can be much shorter than the key length
        // The salt is fixed and does not get transmitted. The attacker can get it from this application though.
        PasswordDeriveBytes _passwordBytes = new PasswordDeriveBytes(password, Salt, Hash, 2);
        byte[] keyBytes = _passwordBytes.GetBytes(KeySize / 8);

        cipher.Mode = CipherMode.CBC;

        // Send to encryption stream
        using (ICryptoTransform encryptor = cipher.CreateEncryptor(keyBytes, IV))
        {
            using (MemoryStream to = new MemoryStream())
            {
                using (CryptoStream writer = new CryptoStream(to, encryptor, CryptoStreamMode.Write))
```

Beispiel-Klasse



```
using (T cipher = new T())
{
    // From the password we derive a key with the correct length.... the password can be much shorter than the key length
    // The salt is fixed and does not get transmitted. The attacker can get it from this application though.
    PasswordDeriveBytes _passwordBytes = new PasswordDeriveBytes(password, Salt, Hash, 2);
    byte[] keyBytes = _passwordBytes.GetBytes(KeySize / 8);

    cipher.Mode = CipherMode.CBC;

    // Send to encryption stream
    using (ICryptoTransform encryptor = cipher.CreateEncryptor(keyBytes, IV))
    {
        using (MemoryStream to = new MemoryStream())
        {
            using (CryptoStream writer = new CryptoStream(to, encryptor, CryptoStreamMode.Write))
            {
                writer.Write(value, 0, value.Length);
                writer.FlushFinalBlock();

                encrypted = to.ToArray();
            }
        }
    }
    cipher.Clear();
}

return encrypted;
}
```

Beispiel-Klasse



```
/// This is what you call to decrypt a payload. It decrypts the password used for the symmetric encryption using the RSA
/// asymmetric cypher and then decrypts the message and signature with the the symmetric decryption routine
public static byte[] DecryptPayload(byte[] message, RSACryptoServiceProvider SenderKey, RSACryptoServiceProvider ReceiverKey)
{
    // Read the (symmetric) encrypted payload and the (asymmetric) encrypted key from the byte array
    byte[] encPayload = null;
    byte[] encAESKey = null;

    using (MemoryStream ms = new MemoryStream(message))
    {
        using (BinaryReader binReader = new BinaryReader(ms))
        {
            {
                int num = binReader.ReadInt32();
                encPayload = binReader.ReadBytes(num);
                num = binReader.ReadInt32();
                encAESKey = binReader.ReadBytes(num);
            }
        }
    }

    // We decrypt the Password and the IV for the AES container
    byte[] decAESKeyAndIV = ReceiverKey.Decrypt(encAESKey, true);

    // We take the two apart
    byte[] Password = new byte[PasswordLength];
    byte[] IV = new byte[IVLength];
    Buffer.BlockCopy(decAESKeyAndIV, 0, Password, 0, Password.Length);
    Buffer.BlockCopy(decAESKeyAndIV, Password.Length, IV, 0, IV.Length);

    // We decrypt the AES container with the decrypted password and IV
```

Beispiel-Klasse



```
// We take the signature
byte[] Password = new byte[PasswordLength];
byte[] IV = new byte[IVLength];
Buffer.BlockCopy(decAESKeyAndIV, 0, Password, 0, Password.Length);
Buffer.BlockCopy(decAESKeyAndIV, Password.Length, IV, 0, IV.Length);

// We decrypt the AES container with the decrypted password and IV
byte[] decPayload = DecryptSymmetric<AesManaged>(encPayload, Password, IV);

// We take apart the signature and the message
byte[] decSignature = null;
byte[] decMessage = null;

using (MemoryStream ms = new MemoryStream(decPayload))
{
    using (BinaryReader binReader = new BinaryReader(ms))
    {
        {
            int num = binReader.ReadInt32();
            decSignature = binReader.ReadBytes(num);
            num = binReader.ReadInt32();
            decMessage = binReader.ReadBytes(num);
        }
    }
}

// We use the senders key to verify that the message was not altered
if (!SenderKey.VerifyData(decMessage, HashAlgorithm.Create("SHA256"), decSignature))
    return null;

return decMessage;
}
```


Beispiel-Klasse



```
/// The symmetrical part of the decryption, that accepts a password and IV received
/// from the asymmetrical decryption
private static byte[] DecryptSymmetric<T>(byte[] value, byte[] password, byte[] IV) where T : SymmetricAlgorithm, new()
{
    byte[] decrypted;
    byte[] decMsg;
    int decryptedByteCount = 0;

    using (T cipher = new T())
    {
        // From the password we derive a key with the correct length.... the password can be much shorter than the key length
        // The salt is fixed and does not get transmitted. The attacker can get it from this application though.
        PasswordDeriveBytes _passwordBytes = new PasswordDeriveBytes(password, Salt, Hash, 2);
        byte[] keyBytes = _passwordBytes.GetBytes(KeySize / 8);

        cipher.Mode = CipherMode.CBC;

        // Send to encryption stream
        try
        {
            using (ICryptoTransform decryptor = cipher.CreateDecryptor(keyBytes, IV))
            {
                using (MemoryStream from = new MemoryStream(value))
                {
                    using (CryptoStream reader = new CryptoStream(from, decryptor, CryptoStreamMode.Read))
                    {
                        decrypted = new byte[value.Length];
                        decryptedByteCount = reader.Read(decrypted, 0, decrypted.Length);
                    }
                }
            }
        }
    }
}
```


Beispiel-Klasse



```
// Send to encryption stream
try
{
    using (ICryptoTransform decryptor = cipher.CreateDecryptor(keyBytes, IV))
    {
        using (MemoryStream from = new MemoryStream(value))
        {
            using (CryptoStream reader = new CryptoStream(from, decryptor, CryptoStreamMode.Read))
            {
                decrypted = new byte[value.Length];
                decryptedByteCount = reader.Read(decrypted, 0, decrypted.Length);
            }
        }
    }
}
catch (Exception)
{
    return null;
}

decMsg = new byte[decryptedByteCount];
Buffer.BlockCopy(decrypted, 0, decMsg, 0, decryptedByteCount);

cipher.Clear();

return decMsg;
}
```

Zusammenfassung?



- Wichtige Kommunikation sollte immer verschlüsselt sein
- Asymmetrische Algorithmen verschlüsseln Passwörter
- Symmetrische Algorithmen verschlüsseln Daten
- Niemals Abkürzungen nehmen (statische IV, kein SALT, nur RSA, etc.)
- Use CryptoRandomNumbers to create SALT, Random Password und Random IV



FRAGEN?

Beantworte ich in der Pause...

Alle noch fit?



Diskussion



- Wer würde ein Spiel eher ohne Verschlüsselung machen?
- Wer kennt Multiplayer Spiele bei denen man cheaten kann?
- Wie wichtig ist Verschlüsselung eurer Daten für euch persönlich?