Kostenloses eBook

LERNEN encryption

Free unaffiliated eBook created from
Stack Overflow contributors.

#encryption

Inhaltsverzeichnis

Über

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: encryption

It is an unofficial and free encryption ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official encryption.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Kapitel 1: Erste Schritte mit der Verschlüsselung

Bemerkungen

In diesem Abschnitt erhalten Sie einen Überblick über die Verschlüsselung und warum ein Entwickler sie verwenden möchte.

Es sollte auch alle großen Themen in der Verschlüsselung erwähnen und auf die verwandten Themen verweisen. Da die Dokumentation zur Verschlüsselung neu ist, müssen Sie möglicherweise erste Versionen dieser verwandten Themen erstellen.

Examples

Was ist Verschlüsselung?

> Verschlüsselung ist in der Kryptographie der Vorgang, Nachrichten oder Informationen so zu codieren, dass nur autorisierte Parteien darauf zugreifen können.
>
> Quelle: Verschlüsselung - Wikipedia

Erste Schritte mit der Verschlüsselung online lesen:
https://riptutorial.com/de/encryption/topic/4306/erste-schritte-mit-der-verschlusselung

Kapitel 2: Caesar-Chiffre

Einführung

Eine Caesar-Chiffre ist eine der einfachsten und bekanntesten Verschlüsselungstechniken. Die Namen stammen von Julius Caesar, der ihn laut Suetonius mit einer Verschiebung von drei verwendet hat, um militärisch wichtige Nachrichten zu schützen

Examples

Verschlüsselung

Die Erweckung geschieht, indem jeder Buchstabe des Alfabet durch einen anderen Buchstaben ersetzt wird. Mit diesem Kreis können die Tasten angezeigt werden. Hier wird eine Verschiebung von 8 Zeichen verwendet.

Hier wird das A durch T, B durch U, C durch V usw. ersetzt. Also wird der nächste String verschlüsselt:

Original	Verschlüsselt
HALLO WELT	AXEEH PHKEW

Entschlüsselung

Die Beschreibung erfolgt durch denselben Code, der im Verschlüsselungsbeispiel gezeigt wurde. Beispiel:

Verschlüsselt	Original
AXEEH PHKEW	HALLO WELT

Hacking

Ceasar-Chiffren sind leicht zu hacken. Wenn Sie wissen, dass ein verschlüsseltes A gleich H und ein P gleich I ist, könnte alles mit demselben Verschlüsselungsschlüssel verschlüsselt werden.

Caesar-Chiffre online lesen: https://riptutorial.com/de/encryption/topic/8823/caesar-chiffre

Kapitel 3: Text-zu-Text-Verschlüsselung

Einführung

Die moderne Kryptographie arbeitet mit Bytes und nicht mit Text. Die Ausgabe kryptographischer Algorithmen erfolgt daher in Bytes. Manchmal müssen verschlüsselte Daten über ein Textmedium übertragen und eine binärsichere Kodierung verwendet werden.

Parameter

Parameter	Einzelheiten
`TE`	Textentschlüsselung. Die Umwandlung von Text in Bytes. UTF-8 ist eine gängige Wahl.
`BE`	Binäre Kodierung. Eine Transformation, die beliebige Daten verarbeiten und eine gültige Zeichenfolge erzeugen kann. Base64 ist die am häufigsten verwendete Codierung, wobei Base16 / Hexadezimal ein guter Zweitplatzierter ist. Wikipedia hat eine Liste von Kandidatenkodierungen (halten Sie sich an die mit "Arbitrary" gekennzeichneten).

Bemerkungen

Der allgemeine Algorithmus ist:

`Encrypt` :

- Umwandlung von `InputText` in `InputBytes` durch Kodierung von `TE` (Textkodierung).
- `InputBytes` `OutputBytes` `InputBytes` in `OutputBytes`
- `OutputBytes` `OutputText` über `BE` (Binärcodierung) in `OutputText` .

`Decrypt` (umgekehrtes BE und TE von `Encrypt`):

- Umwandlung von `InputText` in `InputBytes` durch Kodierung von `BE` .
- Entschlüssele `InputBytes` zu `OutputBytes`
- Verwandeln `OutputBytes` zu `OutputText` via `TE` .

Der häufigste Fehler ist die Auswahl einer "Textcodierung" anstelle einer "Binärcodierung" für `BE` . Dies ist ein Problem, wenn ein verschlüsseltes Byte (oder ein IV-Byte) außerhalb des Bereichs `0x20` - `0x7E` (für UTF-8 oder ASCII) liegt). Da der "sichere Bereich" weniger als die Hälfte des Byte-Speicherplatzes beträgt, sind die Chancen für eine erfolgreiche Textcodierung verschwindend gering.

- Wenn die Zeichenfolge nach der Verschlüsselung `0x00` enthält, werden C / C ++ -Programme dies wahrscheinlich als Ende der Zeichenfolge falsch interpretieren.

- Wenn für ein konsolenbasiertes Programm `0x08` angezeigt wird, kann es das vorherige Zeichen (und den Steuercode) löschen, wodurch der `InputText` Wert für `Decrypt` den falschen Wert (und die falsche Länge) hat.

Examples

C

```
internal sealed class TextToTextCryptography : IDisposable
{
    // This type is not thread-safe because it repeatedly mutates the IV property.
    private SymmetricAlgorithm _cipher;

    // The input to Encrypt and the output from Decrypt need to use the same Encoding
    // so text -> bytes -> text produces the same text.
    private Encoding _textEncoding;

    // The output text ("the wire format") needs to be the same encoding for To-The-Wire
    // and From-The-Wire.
    private Encoding _binaryEncoding;

    /// <summary>
    /// Construct a Text-to-Text encryption/decryption object.
    /// </summary>
    /// <param name="key">
    ///   The cipher key to use
    /// </param>
    /// <param name="textEncoding">
    ///   The text encoding to use, or <c>null</c> for UTF-8.
    /// </param>
    /// <param name="binaryEncoding">
    ///   The binary/wire encoding to use, or <c>null</c> for Base64.
    /// </param>
    internal TextToTextCryptography(
        byte[] key,
        Encoding textEncoding,
        Encoding binaryEncoding)
    {
        // The rest of this class can operate on any SymmetricAlgorithm, but
        // at some point you either need to pick one, or accept an input choice.
        SymmetricAlgorithm cipher = Aes.Create();

        // If the key isn't valid for the algorithm this will throw.
        // Since cipher is an Aes instance the key must be 128, 192, or 256 bits
        // (16, 24, or 32 bytes).
        cipher.Key = key;

        // These are the defaults, expressed here for clarity
        cipher.Padding = PaddingMode.PKCS7;
        cipher.Mode = CipherMode.CBC;

        _cipher = cipher;
        _textEncoding = textEncoding ?? Encoding.UTF8;

        // Allow null to mean Base64 since there's not an Encoding class for Base64.
        _binaryEncoding = binaryEncoding;
    }
```

```
internal string Encrypt(string text)
{
    // Because we are encrypting with CBC we need an Initialization Vector (IV).
    // Just let the platform make one up.
    _cipher.GenerateIV();
    byte[] output;

    using (ICryptoTransform encryptor = _cipher.CreateEncryptor())
    {
        if (!encryptor.CanTransformMultipleBlocks)
            throw new InvalidOperationException("Rewrite this code with CryptoStream");

        byte[] input = _textEncoding.GetBytes(text);
        byte[] encryptedOutput = encryptor.TransformFinalBlock(input, 0, input.Length);

        byte[] iv = _cipher.IV;

        // Build output as iv.Concat(encryptedOutput).ToArray();
        output = new byte[iv.Length + encryptedOutput.Length];
        Buffer.BlockCopy(iv, 0, output, 0, iv.Length);
        Buffer.BlockCopy(encryptedOutput, 0, output, iv.Length, encryptedOutput.Length);
    }

    return BytesToWire(output);
}

internal string Decrypt(string text)
{
    byte[] inputBytes = WireToBytes(text);

    // Rehydrate the IV
    byte[] iv = new byte[_cipher.BlockSize / 8];
    Buffer.BlockCopy(inputBytes, 0, iv, 0, iv.Length);

    _cipher.IV = iv;

    byte[] output;

    using (ICryptoTransform decryptor = _cipher.CreateDecryptor())
    {
        if (!decryptor.CanTransformMultipleBlocks)
            throw new InvalidOperationException("Rewrite this code with CryptoStream");

        // Decrypt everything after the IV.
        output = decryptor.TransformFinalBlock(
            inputBytes,
            iv.Length,
            inputBytes.Length - iv.Length);
    }

    return _textEncoding.GetString(output);
}

private string BytesToWire(byte[] bytes)
{
    if (_binaryEncoding != null)
    {
        return _binaryEncoding.GetString(bytes);
    }

    // Let null _binaryEncoding be Base64.
```

```
        return Convert.ToBase64String(bytes);
    }

    private byte[] WireToBytes(string wireText)
    {
        if (_binaryEncoding != null)
        {
            return _binaryEncoding.GetBytes(wireText);
        }

        // Let null _binaryEncoding be Base64.
        return Convert.FromBase64String(wireText);
    }

    public void Dispose()
    {
        _cipher.Dispose();
        _cipher = null;
    }
}
```

Text-zu-Text-Verschlüsselung online lesen: https://riptutorial.com/de/encryption/topic/10179/text-zu-text-verschlusselung

Credits

S. No	Kapitel	Contributors
1	Erste Schritte mit der Verschlüsselung	Community, H. Pauwelyn
2	Caesar-Chiffre	H. Pauwelyn
3	Text-zu-Text-Verschlüsselung	bartonjs