無料電子ブック

学習

encryption

Free unaffiliated eBook created from
Stack Overflow contributors.

#encryption

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

1: の

このセクションでは、のと、がそれをするについてをします。

また、されたきなテーマについてもし、するトピックにリンクするがあります。のためのドキュメンテーションはしいので、これらのトピックのバージョンをするがあります。

Examples

とはですか

> では、は、されたのみがアクセスできるようにメッセージまたはをエンコードするプロセスです。
>
> - ウィキペディア

オンラインでのをむ https://riptutorial.com/ja/encryption/topic/4306/の

2: シーザー

き

シーザーは、もでもくられているの1つです。 SuetoniusによればJulius Caesarのは、のメッセージをするために3つのシフトでそれをした

Examples

アルファベットのアルファベットをのアルファベットにきえることでエンサイクロペディアがこります。このでキーをすることができます。ここでは、された8のシフトがあります。

ここでAはT、BはU、CはVなどにきえられますので、のはされます

の	された
こんにちは	AXEEH PHKEW

のでされているのとじキュークルによってがわれます。

された	の
AXEEH PHKEW	こんにちは

ハッキング

Ceasarはにハックすることができます。されたAがHにしく、PがIにしいことをっていれば、じキーをつすべてをすることができます。

オンラインでシーザーをむ https://riptutorial.com/ja/encryption/topic/8823/シーザー

3: テキストからテキストへの

き

のは、テキストではなくバイトでするため、アルゴリズムのはバイトです。には、されたデ―タをテキストをしてしなければならず、バイナリセ―フなエンコ―ディングをするがあります。

パラメ―タ―

パラメ―タ	
TE	テキストエンコ―ディング。テキストからバイトへの。 UTF-8はなです。
BE	バイナリエンコ―ディング。のデ―タをしてなをする。 Base64はもにされるエンコ―ディングで、Base16 / 16は2のです。ウィキペディアにコードのリストがあります「」とラベルけされたものにする。

なアルゴリズムはのとおりです。

`Encrypt`

- `TE` テキストエンコ―ディングをエンコ―ドすること`InputBytes InputText`を`InputBytes`します。
- `InputBytes`を`OutputBytesInputBytes`する
- `BE`をして`OutputBytes`を`OutputText`し`OutputBytes` バイナリエンコ―ディング 。

`Decrypt Encrypt`からのBEおよびTE

- `BE`をエンコ―ドすること`InputBytes` 、 `InputText`を`InputBytes`します。
- `InputBytes`を`OutputBytesInputBytes`する
- `TE`をして`OutputBytes`を`OutputText`します。

もないは、わりのための「バイナリエンコ―ディング」の「テキストエンコ―ディング」をすることである`BE`のされたバイトまたはのIVバイトがであればである、 `0x20 - 0x7E` UTF-8またはASCIIのために 。 「」はバイトスペ―スのよりもさいので、するテキストエンコ―ディングのはにさい。

- のに`0x00`がまれていると、C / C ++プログラムはそれをのとしてってするがあります。
- コンソ―ルベ―スのプログラムで`0x08`が`InputText`と、のおよびコ―ドがされ、 `InputText`が`Decrypt`にったおよびったさがけられることがあります。

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;
    }
}
```

オンラインでテキストからテキストへのをむ https://riptutorial.com/ja/encryption/topic/10179/テキストからテキストへの

クレジット

S. No		Contributors
1	の	Community, H. Pauwelyn
2	シーザー	H. Pauwelyn
3	テキストからテキストへの	bartonjs