LEARNING encryption

#encryption

Table of Contents

About

Chapter 1: Getting started with encryption

Remarks

This section provides an overview of what encryption is, and why a developer might want to use it.

It should also mention any large subjects within encryption, and link out to the related topics. Since the Documentation for encryption is new, you may need to create initial versions of those related topics.

Examples

What is encryption?

> In cryptography, encryption is the process of encoding messages or information in such a way that only authorized parties can access it.
>
> Source: Encryption - Wikipedia

Read Getting started with encryption online: https://riptutorial.com/encryption/topic/4306/getting-started-with-encryption

Chapter 2: Caesar cipher

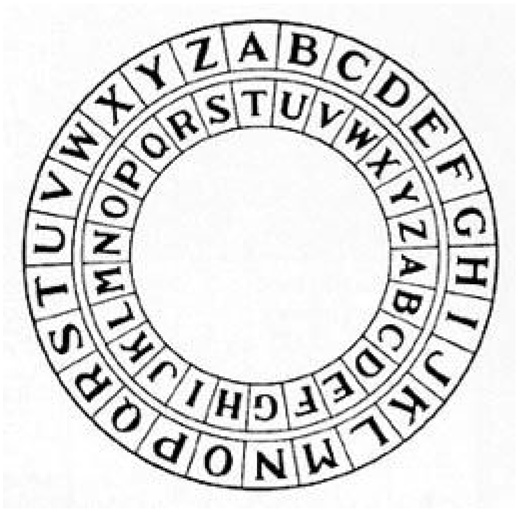
Introduction

A Caesar cipher is one of the simplest and most widely known encryption techniques. The names comes from Julius Caesar, who, according to Suetonius, used it with a shift of three to protect messages of military significance

Examples

Encryption

Encyption happens by replacing each letter of the alfabet with an other letter. The keys can be showed with this circle. Here is a shift of 8 chars used.



Here will the A replaced by T, B by U, C by V etc. So will next string encrypted:

Original	Encrypted
HELLO WORLD	AXEEH PHKEW

Decryption

Decription happens by the same cicle showed in the encryption example. Example:

Encrypted	Original
AXEEH PHKEW	HELLO WORLD

Hacking

Ceasar ciphers are easy to hack. If you know that an encrypted A is equal to H and a P is equal to I, everything with the same encryption key could be encrypted.

Read Caesar cipher online: https://riptutorial.com/encryption/topic/8823/caesar-cipher

Chapter 3: Text-to-text encryption

Introduction

Modern cryptography operates on bytes, not text, so the output of cryptograhic algorithms is bytes. Sometimes encrypted data must be transferred via a text medium, and a binary-safe encoding must be used.

Parameters

Parameter	Details
`TE`	Text Encoding. The transformation from text to bytes. UTF-8 is a common choice.
`BE`	Binary Encoding. A transform which is capable of processing any arbitrary data and producing a valid string. Base64 is the most commonly used encoding, with Base16/hexadecimal a good runner-up. Wikipedia has a list of candidate encodings (stick to ones labelled "Arbitrary").

Remarks

The general algorithm is:

`Encrypt`:

- Transform `InputText` to `InputBytes` via encoding `TE` (text encoding).
- Encrypt `InputBytes` to `OutputBytes`
- Transform `OutputBytes` to `OutputText` via `BE` (binary encoding).

`Decrypt` (reverse BE and TE from `Encrypt`):

- Transform `InputText` to `InputBytes` via encoding `BE`.
- Decrypt `InputBytes` to `OutputBytes`
- Transform `OutputBytes` to `OutputText` via `TE`.

The most common mistake is to choose a "text encoding" instead of a "binary encoding" for `BE`, which is a problem if any encrypted byte (or any IV byte) is outside the range `0x20`-`0x7E` (for UTF-8 or ASCII). Since the "safe range" is less than half of the byte space the chances of a text encoding being successful are vanishingly small.

- If post-encryption string contains a `0x00` then C/C++ programs will likely misinterpret that as the end of the string.
- If a console-based program sees `0x08` it may erase the previous character (and the control code), making the `InputText` value to `Decrypt` have the wrong value (and the wrong length).

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;
    }
}
```

Read Text-to-text encryption online: https://riptutorial.com/encryption/topic/10179/text-to-text-encryption

Credits

S. No	Chapters	Contributors
1	Getting started with encryption	Community, H. Pauwelyn
2	Caesar cipher	H. Pauwelyn
3	Text-to-text encryption	bartonjs