eBook Gratuit

# APPRENEZ encryption

eBook gratuit non affilié créé à partir des **contributeurs de Stack Overflow.**

#encryption

# Table des matières

# À propos

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

# Chapitre 1: Démarrer avec le cryptage

## Remarques

Cette section fournit une vue d'ensemble de ce qu'est le chiffrement et des raisons pour lesquelles un développeur peut vouloir l'utiliser.

Il devrait également mentionner tous les sujets importants du chiffrement, et établir un lien avec les sujets connexes. La documentation pour le chiffrement étant nouvelle, vous devrez peut-être créer des versions initiales de ces rubriques connexes.

## Examples

### Qu'est-ce que le cryptage?

> En cryptographie, le cryptage consiste à coder des messages ou des informations de manière à ce que seules les parties autorisées puissent y accéder.
>
> Source: Cryptage - Wikipedia

Lire Démarrer avec le cryptage en ligne: https://riptutorial.com/fr/encryption/topic/4306/demarrer-avec-le-cryptage

# Chapitre 2: César

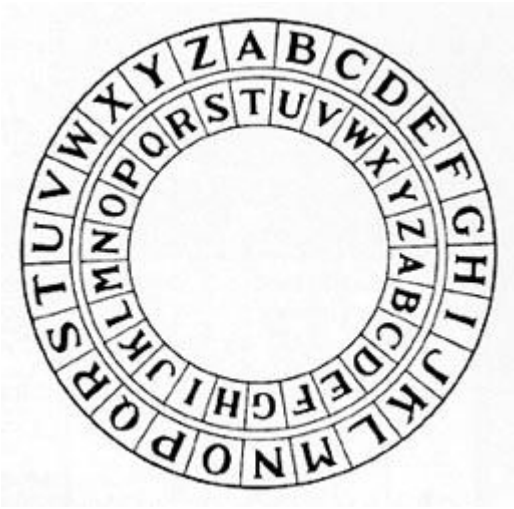
## Introduction

Un chiffrement César est l'une des techniques de chiffrement les plus simples et les plus connues. Les noms viennent de Jules César, qui, selon Suétone, l'a utilisé avec un décalage de trois pour protéger les messages d'importance militaire

## Examples

### Cryptage

Encyption arrive en remplaçant chaque lettre de l'alfabet par une autre lettre. Les clés peuvent être montrées avec ce cercle. Voici un décalage de 8 caractères utilisés.




Ici, le A remplacé par T, B par U, C par V etc.

| Original | Crypté |
|---|---|
| BONJOUR LE MONDE | AXEEH PHKEW |

### Décryptage

Decription se produit par le même exemple que dans l'exemple de chiffrement. Exemple:

| Crypté | Original |
|---|---|
| AXEEH PHKEW | BONJOUR LE MONDE |

### Le piratage

Les chiffres de Ceasar sont faciles à pirater. Si vous savez qu'un A chiffré est égal à H et qu'un P est égal à I, tout ce qui a la même clé de chiffrement peut être chiffré.

Lire César en ligne: https://riptutorial.com/fr/encryption/topic/8823/cesar

# Chapitre 3: Chiffrement du texte en texte

## Introduction

La cryptographie moderne fonctionne sur des octets, pas sur du texte, de sorte que la sortie des algorithmes cryptograhic est octet. Parfois, les données cryptées doivent être transférées via un support texte et un codage binaire sécurisé doit être utilisé.

## Paramètres

| Paramètre | Détails |
|---|---|
| `TE` | Codage du texte. La transformation du texte en octets. UTF-8 est un choix courant. |
| `BE` | Codage binaire. Transformation capable de traiter des données arbitraires et de produire une chaîne valide. Base64 est l'encodage le plus couramment utilisé, avec Base16 / hexadécimal un bon second. Wikipedia a une liste de codages candidats (coller à ceux étiquetés "arbitraire"). |

## Remarques

L'algorithme général est:

`Encrypt` :

- Transformez `InputText` en `InputBytes` via le codage `TE` (encodage de texte).
- Crypter des `InputBytes` en `OutputBytes`
- Transformez les `OutputBytes` en `OutputText` via `BE` ( codage binaire ).

`Decrypt` (inverser BE et TE de `Encrypt` ):

- Transformer `InputText` en `InputBytes` via le codage `BE` .
- Déchiffrer les `InputBytes` en `OutputBytes` en `OutputBytes`
- Transformez les `OutputBytes` en `OutputText` via `TE` .

L'erreur la plus courante consiste à choisir un "codage de texte" au lieu d'un "codage binaire" pour `BE` , ce qui pose problème si un octet chiffré (ou tout octet IV) est en dehors de la plage `0x20` - `0x7E` (pour UTF-8 ou ASCII). ). Comme la "plage de sécurité" est inférieure à la moitié de l'espace octet, les chances qu'un codage de texte soit réussi sont extrêmement faibles.

- Si la chaîne de post-cryptage contient un `0x00` C / C ++ risquent de ne pas interpréter cela comme la fin de la chaîne.
- Si un programme basé sur une console voit `0x08` il peut effacer le caractère précédent (et le code de contrôle), ce qui fait que la valeur `InputText` à `Decrypt` a la mauvaise valeur (et la

mauvaise longueur).

# 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;
    }
}
```

Lire Chiffrement du texte en texte en ligne:
https://riptutorial.com/fr/encryption/topic/10179/chiffrement-du-texte-en-texte

# Crédits

| S. No | Chapitres | Contributeurs |
|---|---|---|
| 1 | Démarrer avec le cryptage | Community, H. Pauwelyn |
| 2 | César | H. Pauwelyn |
| 3 | Chiffrement du texte en texte | bartonjs |