EBook Gratis

APRENDIZAJE encryption

Free unaffiliated eBook created from
Stack Overflow contributors.

#encryption

Tabla de contenido

Acerca de 1

Capítulo 1: Comenzando con el cifrado 2

Observaciones 2

Examples 2

¿Qué es el cifrado? 2

Capítulo 2: Cifrado César 3

Introducción 3

Examples 3

Cifrado 3

Descifrado 3

Seco 3

Capítulo 3: Encriptación de texto a texto 5

Introducción 5

Parámetros 5

Observaciones 5

Examples 6

DO# 6

Creditos 9

Acerca de

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

Capítulo 1: Comenzando con el cifrado

Observaciones

Esta sección proporciona una descripción general de qué es el cifrado y por qué un desarrollador puede querer usarlo.

También debe mencionar cualquier tema grande dentro del cifrado y vincular a los temas relacionados. Dado que la Documentación para el cifrado es nueva, es posible que deba crear versiones iniciales de los temas relacionados.

Examples

¿Qué es el cifrado?

> En la criptografía, el cifrado es el proceso de codificación de mensajes o información de manera que solo las partes autorizadas pueden acceder a él.
>
> Fuente: Encriptación - Wikipedia.

Lea Comenzando con el cifrado en línea:
https://riptutorial.com/es/encryption/topic/4306/comenzando-con-el-cifrado

Capítulo 2: Cifrado César

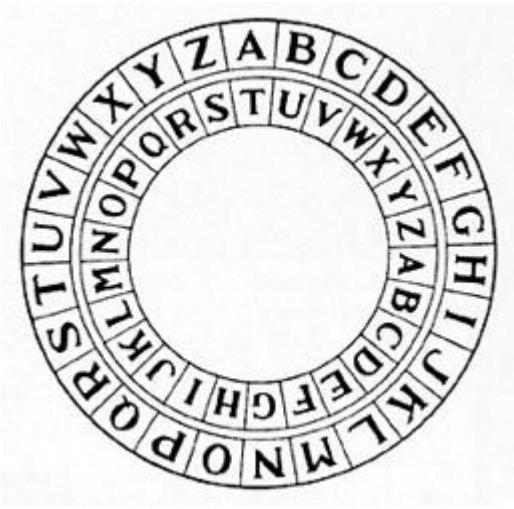
Introducción

Un cifrado César es una de las técnicas de cifrado más sencillas y conocidas. Los nombres provienen de Julio César, quien, según Suetonio, lo usó con un turno de tres para proteger mensajes de importancia militar.

Examples

Cifrado

La encriptación ocurre al reemplazar cada letra del alfabeto con otra letra. Las llaves se pueden mostrar con este círculo. Aquí hay un cambio de 8 caracteres utilizados.



Aquí se reemplazará la A por T, B por U, C por V, etc. La siguiente cadena estará encriptada:

Original	Cifrado
HOLA MUNDO	AXEEH PHKEW

Descifrado

La descripción ocurre por el mismo ciclo que se muestra en el ejemplo de cifrado. Ejemplo:

Cifrado	Original
AXEEH PHKEW	HOLA MUNDO

Seco

Los cifrados de César son fáciles de hackear. Si sabe que una A cifrada es igual a H y una P es igual a I, todo lo que tenga la misma clave de cifrado podría estar cifrado.

Lea Cifrado César en línea: https://riptutorial.com/es/encryption/topic/8823/cifrado-cesar

Capítulo 3: Encriptación de texto a texto

Introducción

La criptografía moderna opera en bytes, no en texto, por lo que la salida de los algoritmos criptográficos es de bytes. A veces, los datos cifrados se deben transferir a través de un medio de texto y se debe utilizar una codificación segura para archivos binarios.

Parámetros

Parámetro	Detalles
`TE`	Codificacion de texto. La transformación del texto a bytes. UTF-8 es una opción común.
`BE`	Codificación binaria. Una transformación que es capaz de procesar cualquier dato arbitrario y producir una cadena válida. Base64 es la codificación más utilizada, con Base16 / hexadecimal un buen subcampeón. Wikipedia tiene una lista de codificaciones candidatas (se adhieren a las etiquetadas como "Arbitrarias").

Observaciones

El algoritmo general es:

`Encrypt`

- Transforme `InputText` en `InputBytes` través de la codificación `TE` (codificación de texto).
- Cifrar `InputBytes` a `OutputBytes`
- Transforme `OutputBytes` en `OutputText` través de `BE` (codificación binaria).

`Decrypt` (revertir BE y TE de `Encrypt`):

- Transforme `InputText` en `InputBytes` través de la codificación `BE` .
- Descifrar `InputBytes` a `OutputBytes`
- Transformar `OutputBytes` a `OutputText` través de `TE` .

El error más común es elegir una "codificación de texto" en lugar de una "codificación binaria" para `BE` , que es un problema si cualquier byte encriptado (o cualquier byte IV) está fuera del rango `0x20` - `0x7E` (para UTF-8 o ASCII). Como el "rango seguro" es menos de la mitad del espacio de bytes, las posibilidades de que una codificación de texto sea exitosa son muy pequeñas.

- Si la cadena posterior al cifrado contiene un `0x00` , es probable que los programas C / C ++ lo malinterpreten como el final de la cadena.

- Si un programa basado en consola ve `0x08` , puede borrar el carácter anterior (y el código de control), haciendo que el valor de `InputText` para `Decrypt` tenga el valor incorrecto (y la longitud incorrecta).

Examples

DO#

```
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;
    }
}
```

Lea Encriptación de texto a texto en línea:
https://riptutorial.com/es/encryption/topic/10179/encriptacion-de-texto-a-texto

Creditos

S. No	Capítulos	Contributors
1	Comenzando con el cifrado	Community, H. Pauwelyn
2	Cifrado César	H. Pauwelyn
3	Encriptación de texto a texto	bartonjs