Making symmetric block cipher meta permutating again

Heinrich Elsigan

February 2, 2026

Contents

Abstract

Hi, I'm Heinrich Elsigan I'm interested in CSharp, Java, MSSQL, .Net Core, Android and politics, society and the future; currently working as freelancer (one person company) and planning a secure endpoint 2 entpoint chat and looking to collaborate on reviews for other repositories and projects.

1. social media, development and other links:
2. personal tech and political blog blog.area23.at
3. GitHub repositories github.com/heinrichelsigan/
4. live demo .Net area23.at/net
5. StackOverflow stackoverflow.com/users/12213151/heinrich-elsigan
6. Curriculum vitae heinrichelsigan.area23.at/cv

1 Thanks to

Normally, thanks to are always at the end of each paper, but the people or organizations I benefited from while trying to make AES strong again are more important than a simple proof of concept showing that it works, except on my raw experimental test form.

1. https://bouncycastle.org/
2. https://schneier.com/
3. https://github.com/dotnet/

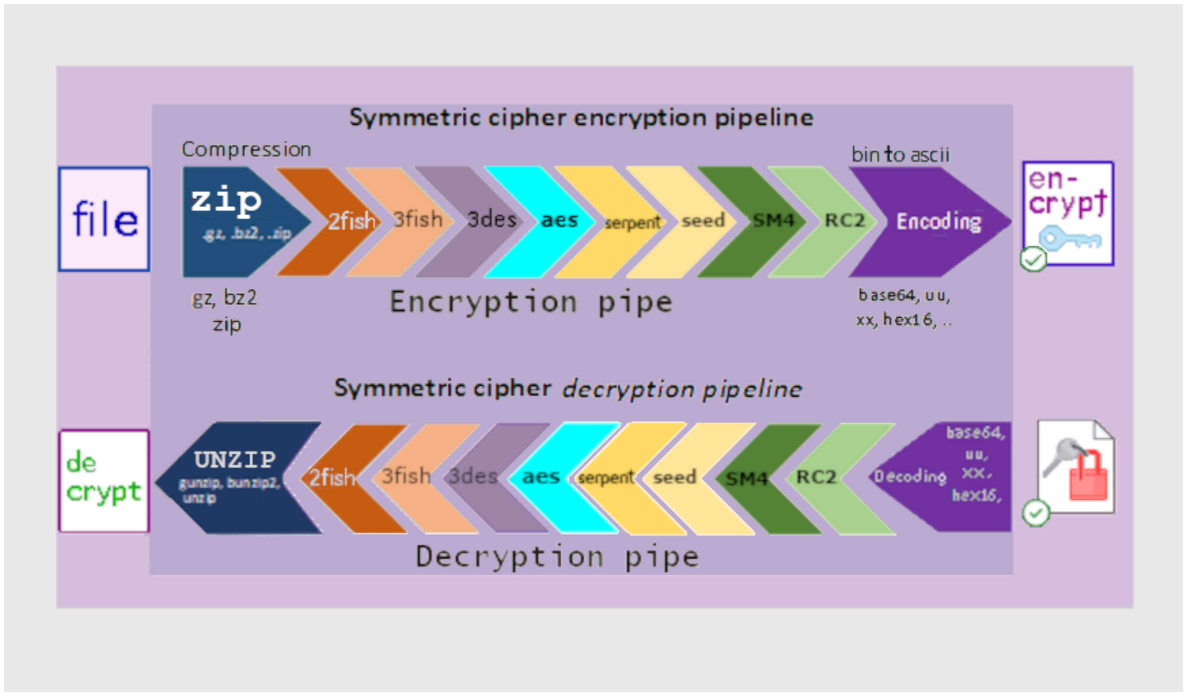


Figure 1: Cipher Pipeline

4. https://github.com/microsoft
5. https://git.lysator.liu.se/nettle/nettle (real easy to read c/c++ code)

2 Theory

2.1 8-staged symmetric block cipher pipeline

An eight staged symmetric block cipher crypto pipeline to improve advanced encryption standard based on meta DES, 3DES with P-Box S-Box.

The following image shows you an example of a symmetric cipher 8 staged encryption pipe and the corresponding decryption inverse pipe.

Before entering the encryption pipe, the file can be zipped to avoid huge amount of symmetric cipher blocks and after exiting the encryption pipe the file can be ascii encoded with base64 mime, uuencode, xxencode or hex16, because symmetric chiphered binary files might lose their block padding.

Implementation is based on my blog article: Making symmetric cipher encryption meta permutating again, including the follwing symmetric cipher algorithms:

- Aes, AesLight, Rijndael
- Bruce Schneier's BlowFish, 2-Fish, 3-Fish
- Camellia, CamelliaLight
- Cast5, Cast6
- National security agency's Des, 3-Des, SkipJack
- Dstu7624
- Ghost, Idea, Noekeon
- RC2, RC532, RC564, RC6

- SEED, SM4
- Serpent, Tnepres
- Tea, XTea
- and my own simplest symmetric block cipher alogrithms: ZenMatrix, ZenMatrix2

2.1.1 Most blockcipher algorithms break on a lot of 0 byte inside

Most blockcipher algorithms break, when you fill a > 2x BLOCKSIZE (byte)0 inside a text or some other file. You can generate such a NULL block with linux dd:

```
sudo nice -n -17 dd if=/dev/zero of=/mnt/h/zeros.txt bs=4k  count=64
```

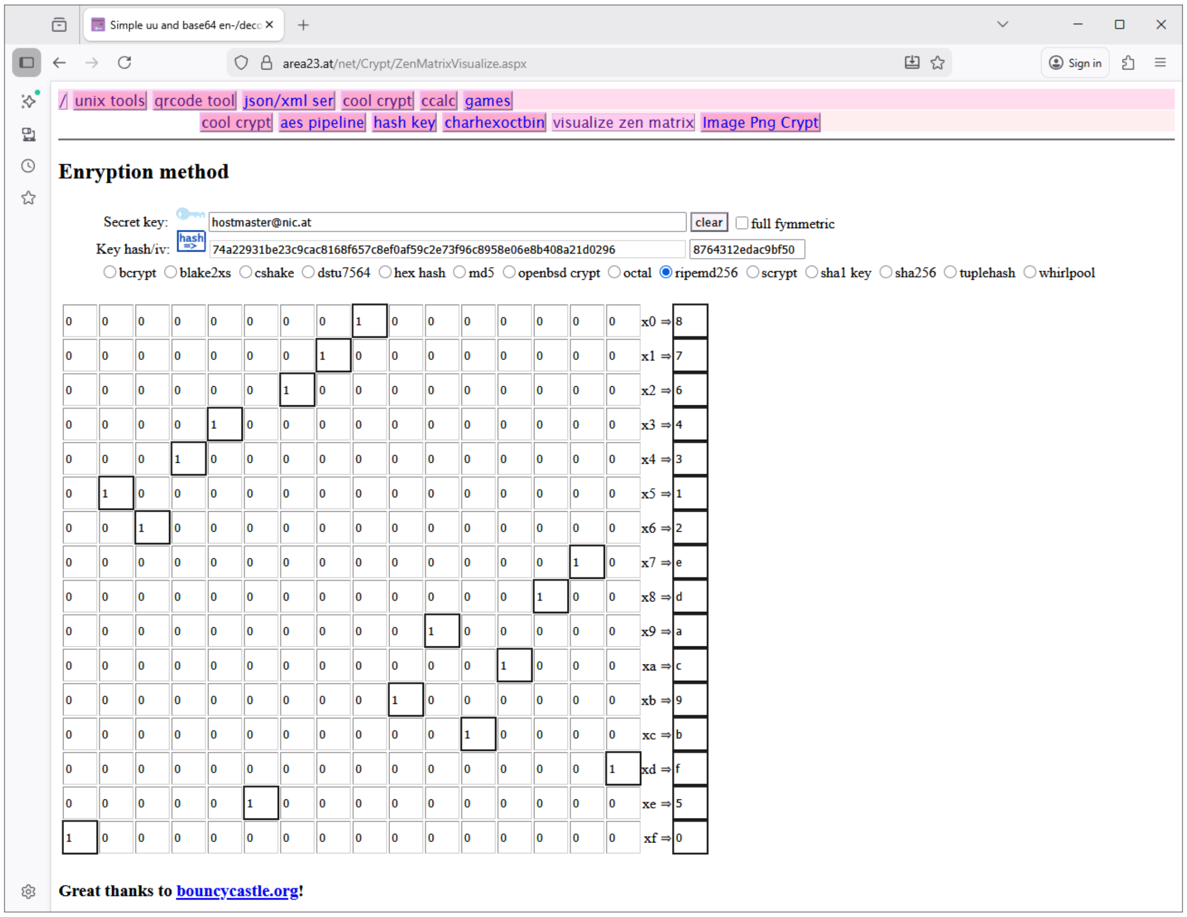
That's why we have added a gzip, bzip2, zip before to compress illegal character blocks inside the file to encrypt.

2.2 CSharp Code Example

```
/* CSharp constructing a CipherPipe */

// 1 with parameter key
string hash = KeyHash.Hex.Hash(key);
CipherPipe cPipe = new CipherPipe(key, hash,
    EncodingType.Base64, ZipType.None, KeyHash.Hex,
    CipherMode2.ECB);

// 2. with an array of ciphers, e.g. BlowFish;TwoFish;ThreeFish
string algos = "BlowFish;TwoFish;ThreeFish";
CipherEnum[] pipeAlgos = CipherEnumExtensions.ParsePipeText(algos);
cPipe = new CipherPipe(pipeAlgos, 8,
            EncodingType.Base64, ZipType.None, KeyHash.Hex,
            CipherMode2.ECB);

/* calling CipherPipe encrypt or decrypt */
string s = "Text to encrypt", key = "myKey", hashIv = KeyHash.Hex.Hash(key);
// encrypt text
string encryptext = cPipe.EncrpytTextGoRounds(s, key, hashIv,
                        EncodingType.Base64, ZipType.None, KeyHash.Hex,
                        CipherMode2.ECB);
// encrypt bytes
byte[] fbytes = System.Text.Encoding.Utf8.GetBytes(s);
byte[] cbytes = cPipe.EncryptEncodeBytes(fileBytes, key, hashIv,
                        EncodingType.Base64, ZipType.None, KeyHash.Hex,
                        CipherMode2.ECB);
// decrypt text
string dcrypt = cPipe.DecryptTextRoundsGo(encryptext, key, hashIv,
                        EncodingType.Base64, ZipType.None, KeyHash.Hex,
                        CipherMode2.ECB);
// decypt bytes
byte[] outBytes = cPipe.DecodeDecrpytBytes(cbytes, key, hashIv,
                        EncodingType.Base64, ZipType.None, KeyHash.Hex,
                        CipherMode2.ECB);
```

2.3 ZenMatrix a simplest possible algortihm to basically understand symmetric blockcipher

Starting from a no permutating 1-matrix, where projection is same as base, ZenMatrix generates a permutating mapping with blocksize 16 a matrix with only one 1 per row and column (rest is 0) to change position inside block and value offset.


Simple uu and base64 en-/deco
area23.at/net/Crypt/ZenMatrixVisualize.aspx
Sign in

/ unix tools qrcode tool json/xml ser cool crypt ccalc games
cool crypt aes pipeline hash key charhexoctbin visualize zen matrix Image Png Crypt

Enryption method

Secret key: hostmaster@nic.at clear [ ] full fymmetric

Key hash/iv: 74a22931be23c9cac8168f657c8ef0af59c2e73f96c8958e06e8b408a21d0296 8764312edac9bf50

bcrypt blake2xs cshake dstu7564 hex hash md5 openbsd crypt octal [✓ ripemd256] scrypt sha1 key sha256 tuplehash whirlpool

0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	x0 ⇒	8
0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	x1 ⇒	7
0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	x2 ⇒	6
0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	x3 ⇒	4
0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	x4 ⇒	3
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	x5 ⇒	1
0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	x6 ⇒	2
0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	x7 ⇒	e
0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	x8 ⇒	d
0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	x9 ⇒	a
0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	xa ⇒	c
0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	xb ⇒	9
0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	xc ⇒	b
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	xd ⇒	f
0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	xe ⇒	5
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	xf ⇒	0

Great thanks to bouncycastle.org!

Figure 2: ZenMatrix Visualize

https://area23.at/net/Crypt/ZenMatrixVisualize.aspx

x	x0	x1	x2	x3	x4	x5	x6	x7	x8	x9	xA	xB	xC	xD	xE	xF
x0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
x1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
x2	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
x3	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
x4	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
x5	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
x6	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
x7	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
x8	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
x9	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
xA	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
xB	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
xC	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
xD	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
xE	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
xF	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

When entering hostmaster@nic.at with hash ripemd256 and not fully symmetric checked, the matrix will look like this:

x	x0	x1	x2	x3	x4	x5	x6	x7	x8	x9	xA	xB	xC	xD	xE	xF	Mx
x0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	8
x1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	7
x2	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	6
x3	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	4
x4	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	3
x5	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
x6	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	2
x7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	E
x8	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	D
x9	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	A
xA	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	C
xB	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	9
xC	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	B
xD	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	F
xE	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	5
xF	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

2.3.1 Mapping

This mean that a byte[8] e.g. 321F 0000 0000 0000 $x0 \implies x8$, $x1 \implies x7$, $x2 \implies x6$, $x3 \implies x4$, $xF \implies x0$, is mapped to 8880 8868 4888 8888.

3 Software

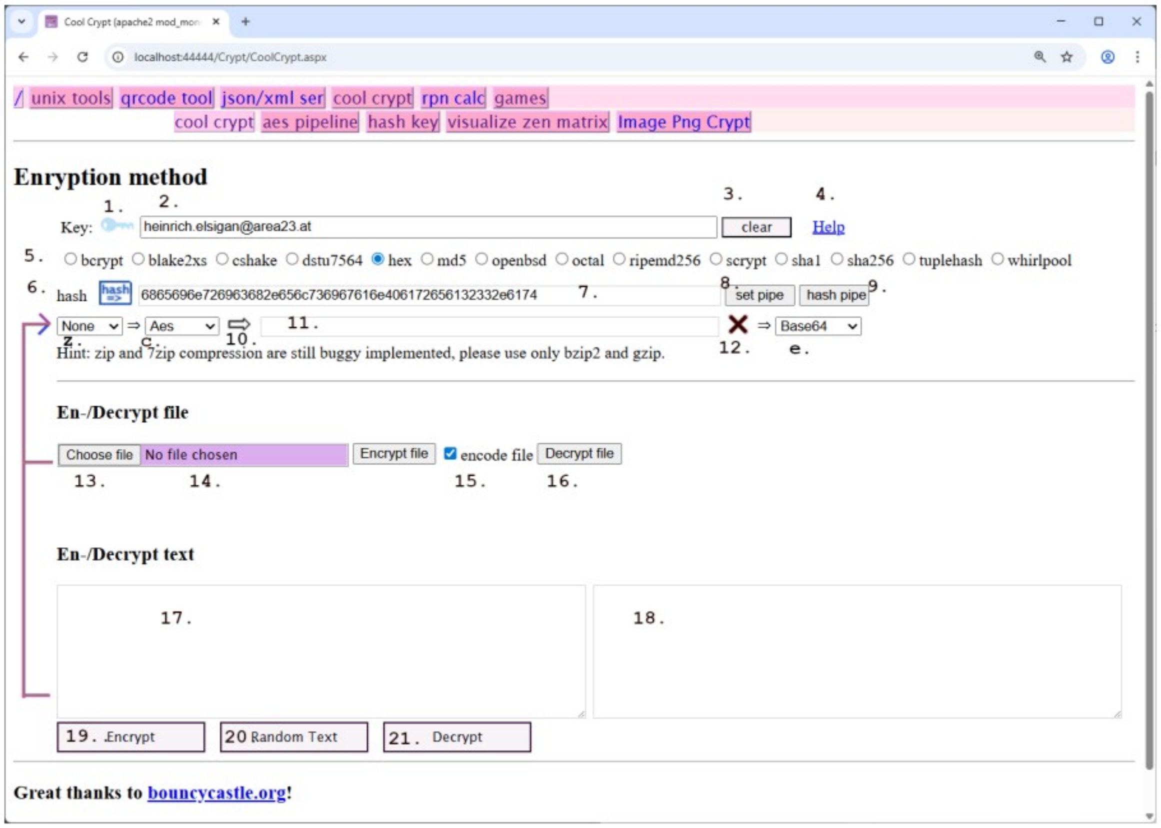
Source code of PermAgainCrypt is hoted at Github. Github releases contains compiled and linked .exe files for windows, but most users prefer download from a seperate download site https://cqrxs.eu/.

3.1 Git: PermAgainCrypt minimalistic repository

https://github.com/heinrichelsigan/PermAgainCrypt/

3.2 Download

Go to https://cqrxs.eu/ or https://io.cqrxs.eu/ and choose latest version. You can directly go to the download area https://cqrxs.eu/download/ or https://io.cqrxs.eu/download/. Attention, version


Cool Crypt (apache2 mod_mon
localhost:44444/Crypt/CoolCrypt.aspx
/ unix tools qrcode tool json/xml ser cool crypt rpn calc games
cool crypt aes pipeline hash key visualize zen matrix Image Png Crypt
1.
2.
3.
4.
clear
Help
5.
bcrypt blake2xs cshake dstu7564 hex md5 openbsd octal ripemd256 scrypt sha1 sha256 tuplehash whirlpool
6.
7.
8.
9.
set pipe
hash pipe
10.
11.
12.
z.
c.
e.
13.
14.
15.
16.
Choose file
Encrypt file
Decrypt file
17.
18.
19. Encrypt
20 Random Text
21. Decrypt

Enryption method

Key: heinrich.elsigan@area23.at

bcrypt blake2xs cshake dstu7564 [✓ hex] md5 openbsd octal ripemd256 scrypt sha1 sha256 tuplehash whirlpool

hash 6865696e726963682e656c736967616e406172656132332e6174

None ⇒ Aes ⇒ ⇒ Base64

Hint: zip and 7zip compression are still buggy implemented, please use only bzip2 and gzip.

En-/Decrypt file

No file chosen [✓ encode file]

En-/Decrypt text

Great thanks to bouncycastle.org!

Figure 3: Symmetric Cipher WebForm

before March 2025 have a 3-fish over Aes-Engine encryption bug, because 3-fish uses AES default block size and key length and Bouncy Castle Aes-Engine parameters. It works, but settings AES default engine for 3-fish, isn't so serious and please download version after March 30.

3.3 Documentation online en-/decrypt form

1. Key: When clicking key your entered key will be stored temporary in session.
2. Textbox secret key: Enter your Email address and secret key.
3. Buton Clear: Clear and reset the entire form.
4. Hyperlink Help: Show this help
5. RadioButtonList KeyHashes Choose the hash method to hash your secret key.
6. ImageButton Hash: Clicking will hash your key and display hashed key in textbox.
7. TextBox Hash (readonly): Displays your hashed key.
8. Button "Set Pipe": sets symmetric cipher pipe, dependent only on your entered key
9. Button "Hash Pipe": sets symmetric cipher pipe, dependent primary on calculated hash and secondary on your entered key.
10. DropDown ZipTypes Choose encryption type (Please only GZip or Zip or None)
11. DropDown CipherTypes Choose a symmetric cipher to add it to the symmetric cipher pipe.

12. ImageButton „add algo“: Clicking on will add the in c. DropDown CipherTypes selected symmetric cipher algorithm to CipherPipe.

13. TextBox CipherPipe (readonly): Displays the current Cipher Pipe algorithms.

14. ImageButton “Clear Pipe”: Clicking on will clear only the entire Cipher Pipe

15. DropDown EncodingTypes: Choose the final binary to ascii encoder, default is Base64. Beware of using uuencode in Web, because <> will be interpreted as possible html injection.

16. TextArea Source: paste or enter Text here.

17. TextArea Destination: (readonly) After clicking Encrypt or Decrypt processed text will appear in the text area.

18. Button Encrypt: Encrypts the text from TextArea Source and displays encrypted text in TextArea Destination.

19. Button “Random Text”: Adds a short fortune to TextArea Source.

20. Button Decrypt: Decrypts text in TextArea Source and display decrypted text in TextArea Destination.

3.4 Documentation of WinForm

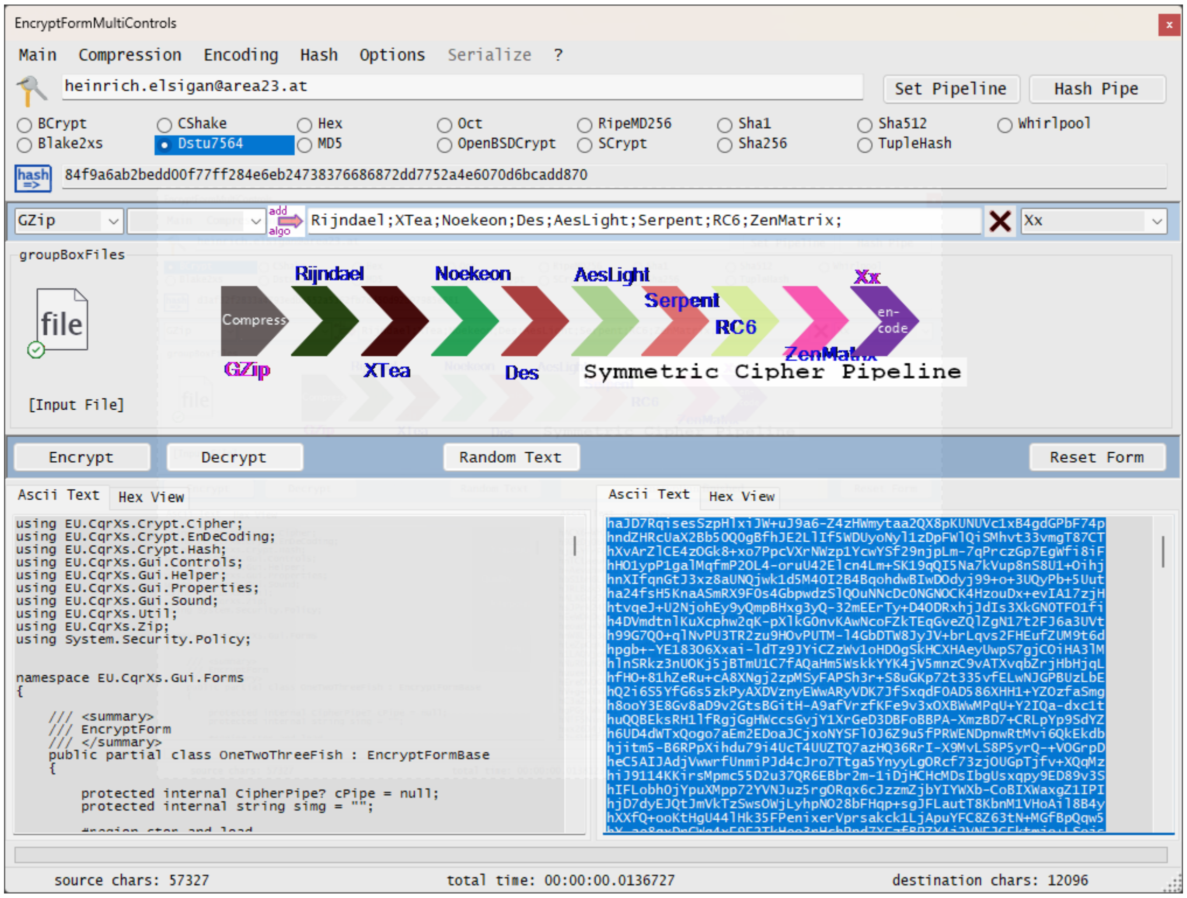
You can also go directly to github.com/heinrichelsigan/PermAgainCrypt/releases to download https://github.com/heinrichelsigan/PermAgainCrypt/releases/download/2026-01-25-Hex64-Console-JFrame-Latex-release-v2.26.125/2026-01-25_EU.CqrXs.Gui_x64.7z . . .

3.5 Documentation of Windows Console

Download the lastest Console from io.cqrxs.eu/download/EU.CqrXs.Console/

3.6 Usage: EU.CqrXs.Console.exe

```
Usage:  EU.CqrXs.Console.exe
    -i  --inFile= | --inText={string|EnviromentVariable} | --inStd
    -k  --key=passKey encrypt
    -H  --Hash {BCrypt|CShake|Dstu7564|Hey|MD5|RipeMD256|SCrypt|Sha1|Whirlpool|TupleHash}
            default: Hex
    -z  --zip={gzip|bzip2|zip}
            default: none
    -C  --CipherAlgost={algo1,algo2,...}
            algo:
                Aes,AesLight,Rijndael,Des,Des3,Dstu7624,
                Aria,Camellia,CamelliaLight,Cast5,Cast6,
                BlowFish,Fish2,Fish3,
                Gost28147,Idea,Noekeon,
                RC2,RC532,RC564,RC6,
                Seed,SkipJack,Serpent,SM4,
                Tea,Tnepres,XTea,
                ZenMatrix,ZenMatrix2
    -S  --SymmCipher
    -e  --encode={raw|hex16|hex32|base32|base64|uu}
            default: base64
    -D  --Decrypt [ = Inverse_Pipe_Direction ]
    -o  --outFile= | --outText=EnviromentVariable | --outStd
    -V  --verbose
    -?  --gethelp
```



```
heinrich.elsigan@area23.at
```

```
84f9a6ab2bedd00f77ff284e6eb24738376686872dd7752a4e6070d6bcadd870
```

GZip

```
Rijndael;XTea;Noekeon;Des;AesLight;Serpent;RC6;ZenMatrix;
```

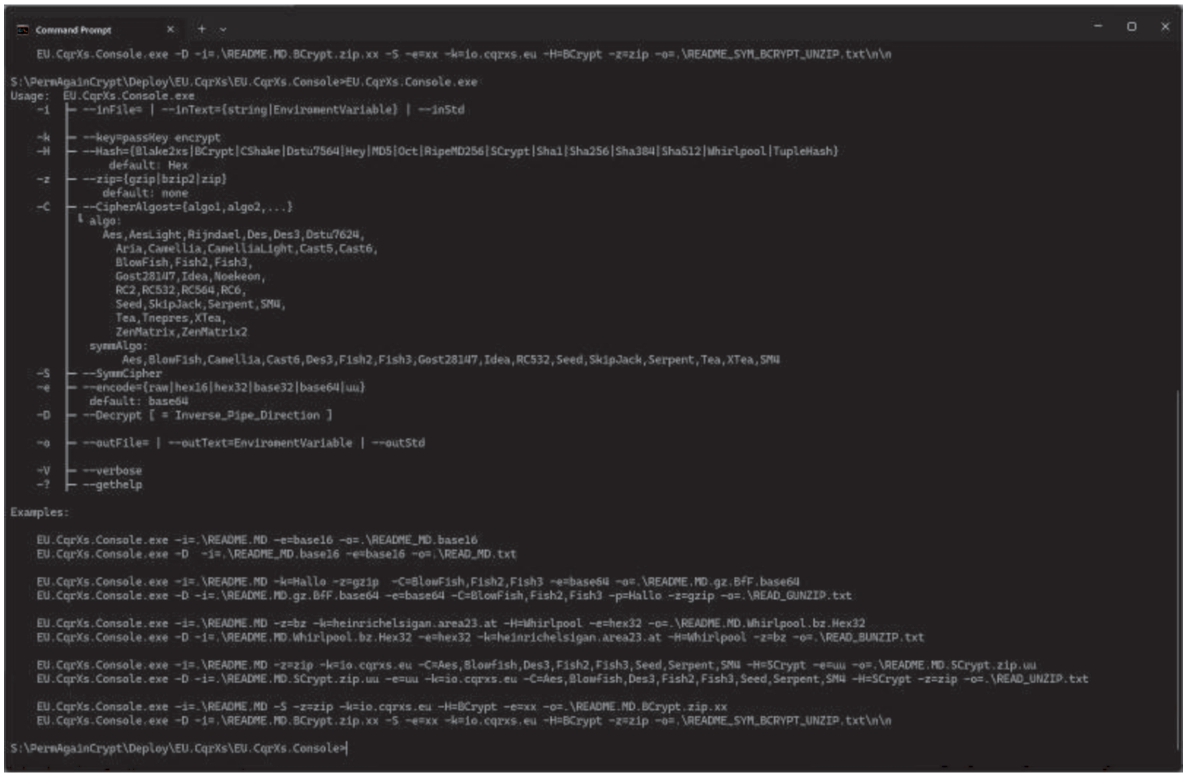
Xx

```
using EU.CqrXs.Crypt.Cipher;
using EU.CqrXs.Crypt.EnDeCoding;
using EU.CqrXs.Crypt.Hash;
using EU.CqrXs.Gui.Controls;
using EU.CqrXs.Gui.Helper;
using EU.CqrXs.Gui.Properties;
using EU.CqrXs.Gui.Sound;
using EU.CqrXs.Util;
using EU.CqrXs.Zip;
using System.Security.Policy;


namespace EU.CqrXs.Gui.Forms
{

    /// <summary>
    /// EncryptForm
    /// </summary>
    public partial class OneTwoThreeFish : EncryptFormBase
    {

        protected internal CipherPipe? cPipe = null;
        protected internal string simg = "";

        #region ctor and load
```

```
haJD7RqisesSzpHlxiJW+uJ9a6-Z4zHWmytaa2QX8pKUNUVc1xB4gdGPbF74p
hndZHRcUaX2Bb5OQ0gBfhJE2LlIf5WDUyoNyl1zDpFWlQiSMhvt33vmgT87CT
hXvArZlCE4zOGk8+xo7PpcVXrNWzp1YcwYSf29njpLm-7qPrczGp7EgWfi8iF
hHO1ypP1galMqfmP2OL4-oruU42Elcn4Lm+SK19qQI5Na7kVup8nS8U1+Oihj
hnXIfqnGtJ3xz8aUNQjwk1d5M40I2B4BqohdwBIwDOdyj99+o+3UQyPb+5Uut
ha24fsH5KnaASmRX9FOs4GbpwdzSlQOuNNcDcONGNOCK4HzouDx+evIA17zjH
htvqeJ+U2NjohEy9yQmpBHxg3yQ-32mEErTy+D4ODRxhjJdIs3XkGNOTFO1fi
h4DVmdtnlKuXcphw2qK-pXlkGOnvKAwNcoFZkTEqGveZQlZgN17t2FJ6a3UVt
h99G7QO+qlNvPU3TR2zu9HOvPUTM-l4GbDTW8JyJV+brLqvs2FHEufZUM9t6d
hpgb+-YE183O6Xxai-ldTz9JYiCZzWv1oHDOgSkHCXHAeyUwpS7gjCOiHA3lM
hlnSRkz3nUOKj5jBTmU1C7fAQaHm5WskkYYK4jV5mnzC9vATXvqbZrjHbHjqL
hfHO+81hZeRu+cA8XNgj2zpMSyFAPSh3r+S8uGKp72t335vfELwNJGPBUzLbE
hQ2i6S5YfG6s5zkPyAXDVznyEWwARyVDK7JfSxqdFOAD586XHH1+YZOzfaSmg
h8ooY3E8Gv8aD9v2GtsBGitH-A9afVrzfKFe9v3xOXBWwMPqU+Y2IQa-dxc1t
huQQBEksRH1lfRgjGgHWccsGvjY1XrGeD3DBFoBBPA-XmzBD7+CRLpYp9SdYZ
h6UD4dWTxQogo7aEm2EDoaJCjxoNYSFlOJ6Z9u5fPRWENDpnwRtMvi6QkEkdb
hjitm5-B6RPpXihdu79i4UcT4UUZTQ7azHQ36RrI-X9MvLS8P5yrQ-+VOGrpD
heC5AIJAdjVwwrfUnmiPJd4cJro7Ttga5YnyyLgORcf73zjOUGpTjfv+XQqMz
hiJ9114KKirsMpmc55D2u37QR6EBbr2m-1iDjHCHcMDsIbgUsxqpy9ED89v3S
hIFLobhOjYpuXMpp72YVNJuz5rgORqx6cJzzmZjbYIYWXb-CoBIXWaxgZ1IPI
hjD7dyEJQtJmVkTzSwsOWjLyhpNO28bFHqp+sgJFLautT8KbnM1VHoAil8B4y
hXXfQ+ooKtHgU44lHk35FPenixerVprsakck1LjApuYFC8Z63tN+MGfBpQqw5
```

```
source chars: 57327          total time: 00:00:00.0136727          destination chars: 12096
```

Figure 4: WinForm CSharp



```
    EU.CqrXs.Console.exe -D -i=.\README.MD.BCrypt.zip.xx -S -e=xx -k=io.cqrxs.eu -H=BCrypt -z=zip -o=.\README_SYM_BCRYPT_UNZIP.txt\n\n

S:\PermAgainCrypt\Deploy\EU.CqrXs\EU.CqrXs.Console>EU.CqrXs.Console.exe
Usage:  EU.CqrXs.Console.exe
    -i  ├─ --inFile= | --inText={string|EnviromentVariable} | --inStd
        │
    -k  ├─ --key=passKey encrypt
    -H  ├─ --Hash={Blake2xs|BCrypt|CShake|Dstu7564|Hey|MD5|Oct|RipeMD256|SCrypt|Sha1|Sha256|Sha384|Sha512|Whirlpool|TupleHash}
        │     default: Hex
    -z  ├─ --zip={gzip|bzip2|zip}
        │    default: none
    -C  ├─ --CipherAlgost={algo1,algo2,...}
        │ └ algo:
        │     Aes,AesLight,Rijndael,Des,Des3,Dstu7624,
        │       Aria,Camellia,CamelliaLight,Cast5,Cast6,
        │       BlowFish,Fish2,Fish3,
        │       Gost28147,Idea,Noekeon,
        │       RC2,RC532,RC564,RC6,
        │       Seed,SkipJack,Serpent,SM4,
        │       Tea,Tnepres,XTea,
        │       ZenMatrix,ZenMatrix2
        │   symmAlgo:
        │        Aes,BlowFish,Camellia,Cast6,Des3,Fish2,Fish3,Gost28147,Idea,RC532,Seed,SkipJack,Serpent,Tea,XTea,SM4
    -S  ├─ --SymmCipher
    -e  ├─ --encode={raw|hex16|hex32|base32|base64|uu}
        │   default: base64
    -D  ├─ --Decrypt [ = Inverse_Pipe_Direction ]
        │
    -o  ├─ --outFile= | --outText=EnviromentVariable | --outStd
        │
    -V  ├─ --verbose
    -?  ├─ --gethelp

Examples:

    EU.CqrXs.Console.exe -i=.\README.MD -e=base16 -o=.\README_MD.base16
    EU.CqrXs.Console.exe -D  -i=.\README_MD.base16 -e=base16 -o=.\READ_MD.txt

    EU.CqrXs.Console.exe -i=.\README.MD -k=Hallo -z=gzip  -C=BlowFish,Fish2,Fish3 -e=base64 -o=.\README.MD.gz.BfF.base64
    EU.CqrXs.Console.exe -D -i=.\README.MD.gz.BfF.base64 -e=base64 -C=BlowFish,Fish2,Fish3 -p=Hallo -z=gzip -o=.\READ_GUNZIP.txt

    EU.CqrXs.Console.exe -i=.\README.MD -z=bz -k=heinrichelsigan.area23.at -H=Whirlpool -e=hex32 -o=.\README.MD.Whirlpool.bz.Hex32
    EU.CqrXs.Console.exe -D -i=.\README.MD.Whirlpool.bz.Hex32 -e=hex32 -k=heinrichelsigan.area23.at -H=Whirlpool -z=bz -o=.\READ_BUNZIP.txt

    EU.CqrXs.Console.exe -i=.\README.MD -z=zip -k=io.cqrxs.eu -C=Aes,Blowfish,Des3,Fish2,Fish3,Seed,Serpent,SM4 -H=SCrypt -e=uu -o=.\README.MD.SCrypt.zip.uu
    EU.CqrXs.Console.exe -D -i=.\README.MD.SCrypt.zip.uu -e=uu -k=io.cqrxs.eu -C=Aes,Blowfish,Des3,Fish2,Fish3,Seed,Serpent,SM4 -H=SCrypt -z=zip -o=.\READ_UNZIP.txt

    EU.CqrXs.Console.exe -i=.\README.MD -S -z=zip -k=io.cqrxs.eu -H=BCrypt -e=xx -o=.\README.MD.BCrypt.zip.xx
    EU.CqrXs.Console.exe -D -i=.\README.MD.BCrypt.zip.xx -S -e=xx -k=io.cqrxs.eu -H=BCrypt -z=zip -o=.\README_SYM_BCRYPT_UNZIP.txt\n\n

S:\PermAgainCrypt\Deploy\EU.CqrXs\EU.CqrXs.Console>
```

Figure 5: Windows EU.CqrXs.Console.exe

3.7 Java JFrame based

- https://docs.oracle.com/javase/8/docs/api/javax/swing/JFrame.html
- https://github.com/openjdk-mirror/jdk7u-jdk/blob/master/src/share/classes/javax/swing/JFrame.java

3.8 How to write Mathematics

LaTeX is great at typesetting mathematics. Let $X_1, X_2, \ldots, X_n$ be a sequence of independent and identically distributed random variables with $\mathrm{E}[X_i] = \mu$ and $\mathrm{Var}[X_i] = \sigma^2 < \infty$, and let

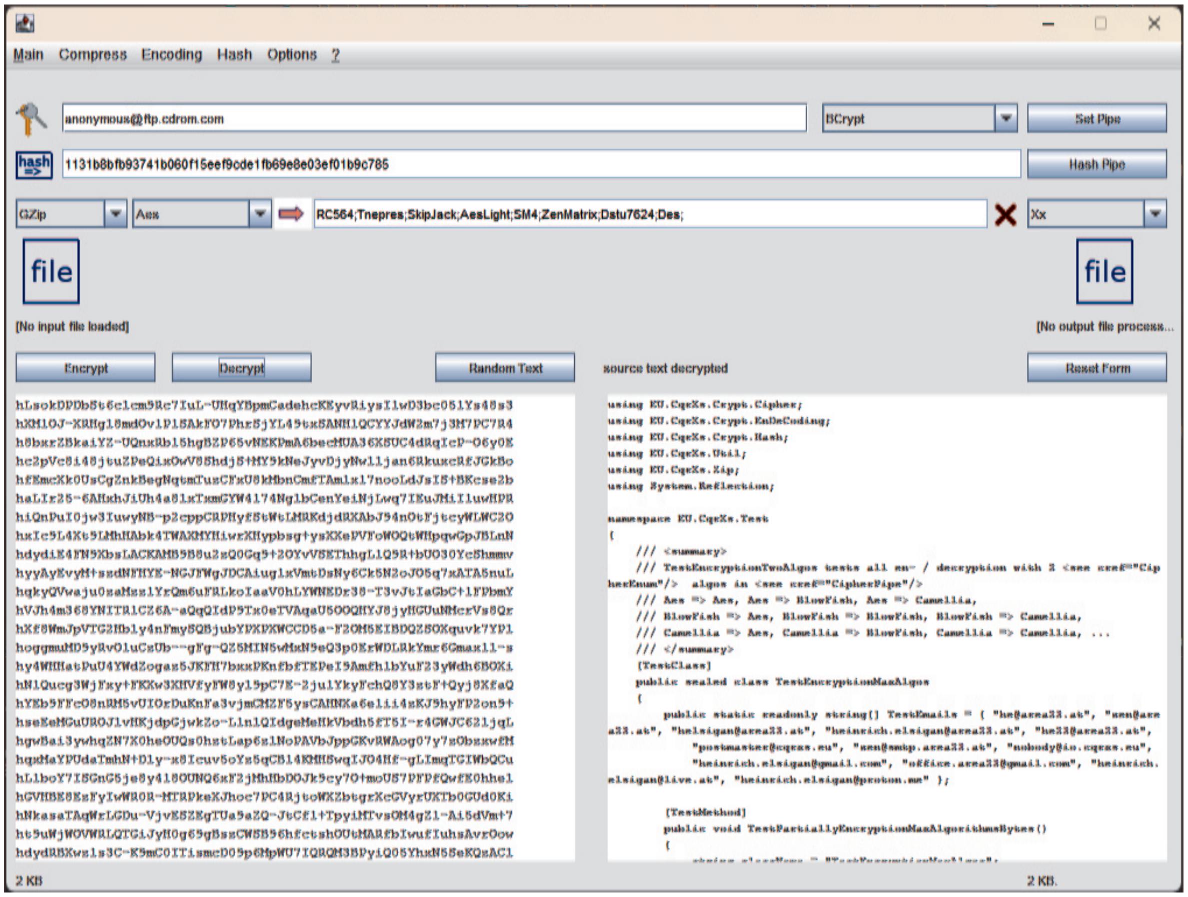
$$S_n = \frac{X_1 + X_2 + \cdots + X_n}{n} = \frac{1}{n}\sum_{i}^{n} X_i$$

denote their mean. Then as n approaches infinity, the random variables $\sqrt{n}(S_n - \mu)$ converge in distribution to a normal $\mathcal{N}(0, \sigma^2)$.

3.9 Good luck!

We hope you find PermAgainCrypt useful, and good luck. help library for more tutorials and user guides! Please also let us know if you have any feedback using the **Contact us** link at the bottom of the Overleaf menu — or use the contact form at https://heinrichelsigan.area23.at.

References



anonymous@ftp.cdrom.com

BCrypt

1131b8bfb93741b060f15eef9cde1fb69e8e03ef01b9c785

GZip

Aes

RC564;Tnepres;SkipJack;AesLight;SM4;ZenMatrix;Dstu7624;Des;

Xx

[No input file loaded]

[No output file process...

source text decrypted

```
hLsokDPDb5t6clcm5Rc7IuL-UHqYBpmCadehcKEyvRiysIlwD3bc05lYs48s3
hXM1OJ-XRHgl8mdOv1P15AkFO7Phr5jYL45tx5ANH1QCYYJdW2m7j3M7PC7R4
h8bxrZBkaiYZ-UQnxRb15hgBZP65vNEKPmA6becMUA36X8UC4dRqIcP-O6y0E
hc2pVc8i48jtuZPeQixOwV85hdj5+MY5kNeJyvDjyNw11jan6RkuxcRfJGkBo
hfEmcXk0UsCgZnkBegNqtmTuzCFxU8kMbnCmfTAmlx17nooLdJsI5+BKcse2b
haLIr25-6AHxhJiUh4a81xTxmGYW4174NglbCenYeiNjLwq7IEuJMiIluwHPR
hiQnPuI0jw3IuwyNB-p2cppCRPHyf5tWtLMRKdjdRXAbJ94nOtFjtcyWLWC2O
hxIc5L4Xt5LMhHAbk4TWAXMYHiwrXHypbsg+ysXXePVFoWOQtWHpqwGpJBLnN
hdydiE4FN5XbsLACKAMB5B8u2sQ0Gq5+2OYvV5EThhgL1Q5R+bUO30Yc5hmmv
hyyAyEvyM+ssdNFHYE-NGJFWgJDCAiuglxVmtDsNy6Ck5N2oJO5q7xATA5nuL
hqkyQVwaju0saMsslYrQm6uFRLkoIaaV0hLYWNEDr38-T3vJtIaGbC+1FPbmY
hVJh4m368YNITR1CZ6A-aQqQIdP5Tx0eTVAqaU5OOQHYJ8jyHGUuNMcrVs8Qr
hXf8WmJpVTG2Hbly4nFmy5QBjubYPXPXWCCD5a-F2OM5EIBDQZ5OXquvk7YP1
hoggmuMD5yRvOluCzUb--gFg-QZ5MIN5wMxN5eQ3p0ErWDLRkYmr6Gmax11-s
hy4WHHatPuU4YWdZogaz5JKFH7bxxPKnfbfTEPeI5AmfhlbYuF23yWdh6BOXi
hN1Qucg3WjFxy+FKKw3XHVfyFW8yl5pC7E-2julYkyFchQ8Y3stF+Qyj8XfaQ
hYEb5FFcO8nRM5vUIOrDuKnFa3vjmCMZF5ysCAHNXa6elii4sKJ5hyFP2on5+
hseEeMGuUROJlvHKjdpGjwkZo-L1n1QIdgeMeHkVbdh5fT5I-r4GWJC621jqL
hgwBai3ywhqZN7X0heOUQs0hstLap6slNoPAVbJppGKvRWAog07y7sObsxwfM
hqxMaYPUdaTmhN+Dly-x8Icuv5oYs5qCB14KMH5wqIJO4Hf-gLImqTGIWbQCu
hLlboY7I5GnG5je8y418OUNQ6xF2jMhHbDOJk5cy7O+moU57PFPfQwfE0hhel
hGVHBE8EsFyIwWR0R-MTRPkeXJhoc7PC4RjtoWXZbtgrXcGVyrUXTb0GUd0Ki
hNkasaTAqWrLGDu-VjvE5ZEgTUa5aZQ-JtCf1+TpyiMTvsOM4gZ1-Ai5dVm+7
ht5uWjWOVWRLQTGiJyH0g65gBssCW5B56hfctshOUtMARfbIwufIuhsAvrOow
hdydRBXwzls3C-K5mC0ITismcD05p6MpWU7IQRQM3BPyiQ05YhxN55eKQsAC1
```

```
using EU.CqrXs.Crypt.Cipher;
using EU.CqrXs.Crypt.EnDeCoding;
using EU.CqrXs.Crypt.Hash;
using EU.CqrXs.Util;
using EU.CqrXs.Zip;
using System.Reflection;

namespace EU.CqrXs.Test
{
    /// <summary>
    /// TestEncryptionTwoAlgos tests all en- / decryption with 2 <see cref="Cip
herEnum"/>  algos in <see cref="CipherPipe"/>
    /// Aes => Aes, Aes => BlowFish, Aes => Camellia,
    /// BlowFish => Aes, BlowFish => BlowFish, BlowFish => Camellia,
    /// Camellia => Aes, Camellia => BlowFish, Camellia => Camellia, ...
    /// </summary>
    [TestClass]
    public sealed class TestEncryptionMaxAlgos
    {
        public static readonly string[] TestEmails = { "he@area23.at", "zen@are
a23.at", "helsigan@area23.at", "heinrich.elsigan@area23.at", "he23@area23.at",
            "postmaster@cqrxs.eu", "zen@smtp.area23.at", "nobody@io.cqrxs.eu",
            "heinrich.elsigan@gmail.com", "office.area23@gmail.com", "heinrich.
elsigan@live.at", "heinrich.elsigan@proton.me" };

        [TestMethod]
        public void TestPartiallyEncryptionMaxAlgorithmsBytes()
        {
```

2 KB

2 KB.

Figure 6: javax.swing.JFrame.java