Making symmetric block cipher meta permutating again

Heinrich Elsigan

February 26, 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. Article written in LaTeX.

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

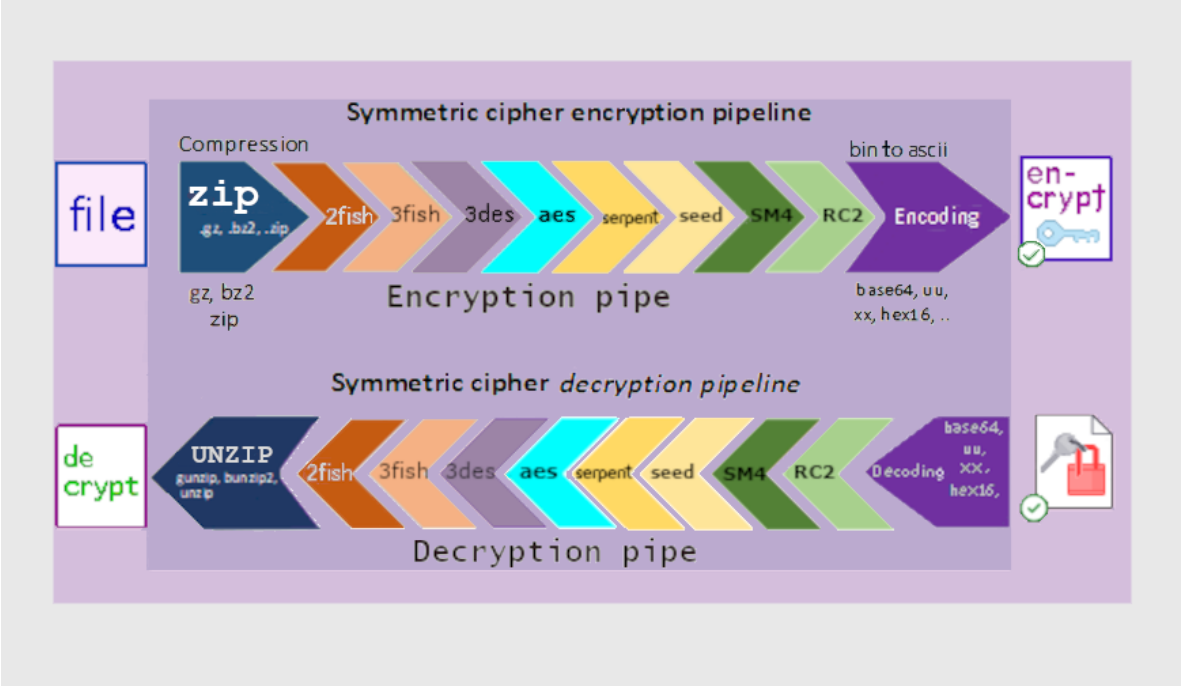

Figure 1: Cipher Pipeline

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

2 Software

Source code of PermAgainCrypt is hosted 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/.

2.1 Git: PermAgainCrypt minimalistic repository

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

2.2 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 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.

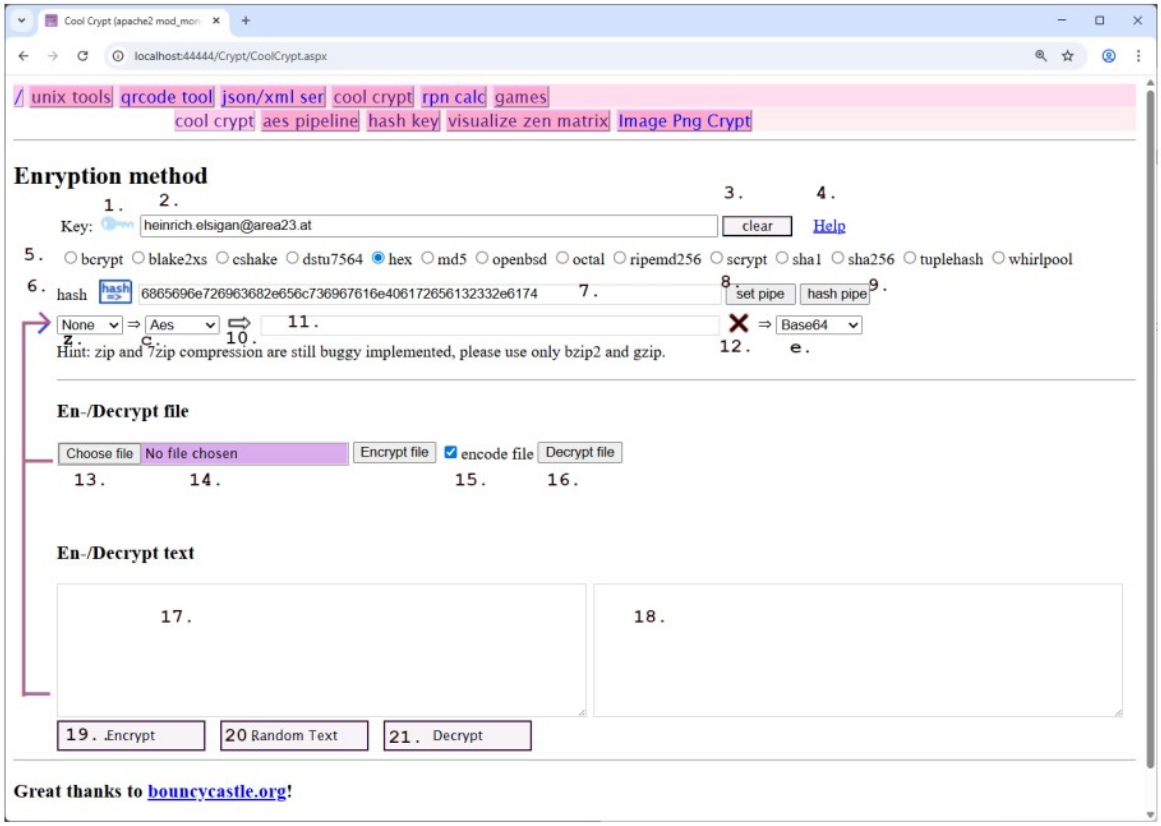

Figure 2: Symmetric Cipher WebForm

3 Different Forms

3.1 Documentation online en-/decrypt form

- cqrxs.eu/net/Crypt
- area23.at/net/Crypt

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.

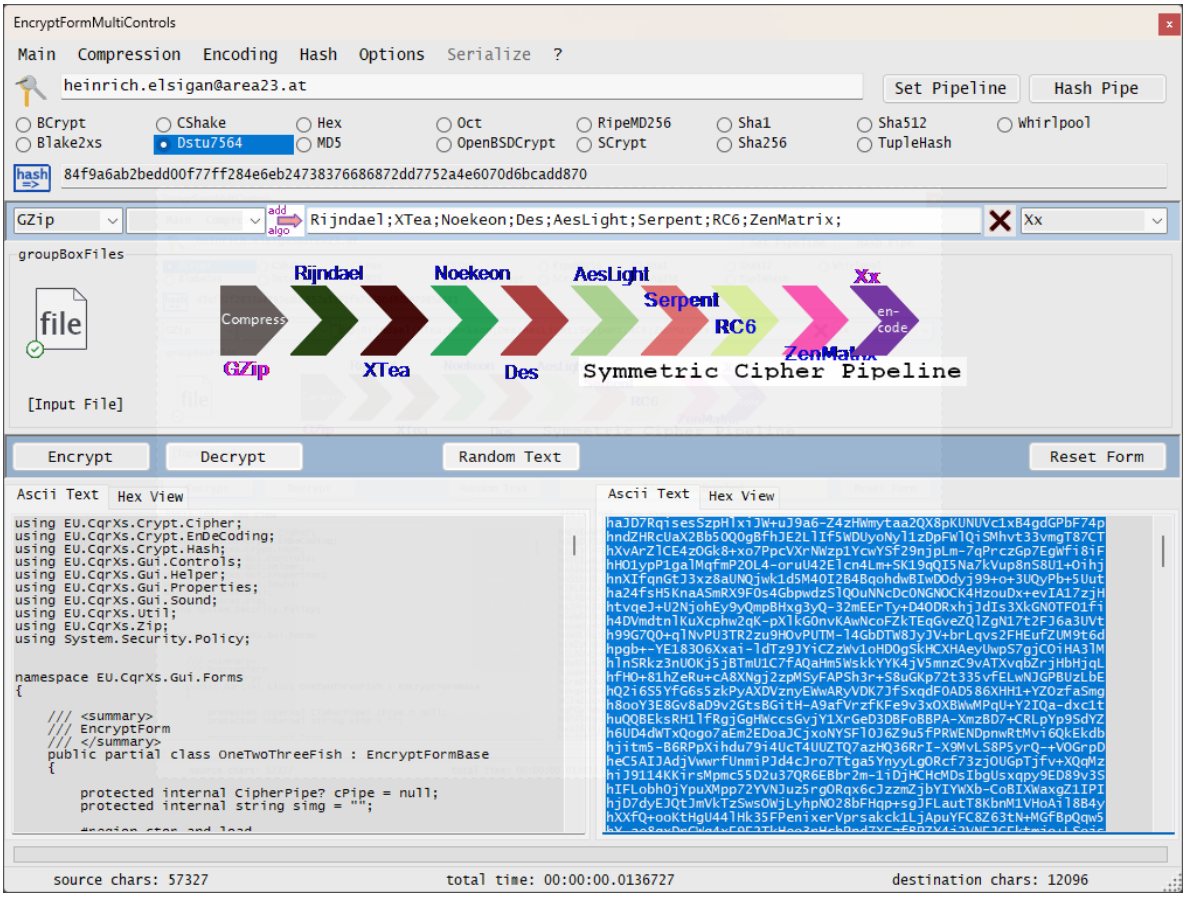

Figure 3: WinForm CSharp

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.

3.2 Documentation of WinForm

You can also go directly to github.com/heinrichelsigan/PermAgainCrypt/releases to download newest Gui Form x86 and x64. ...

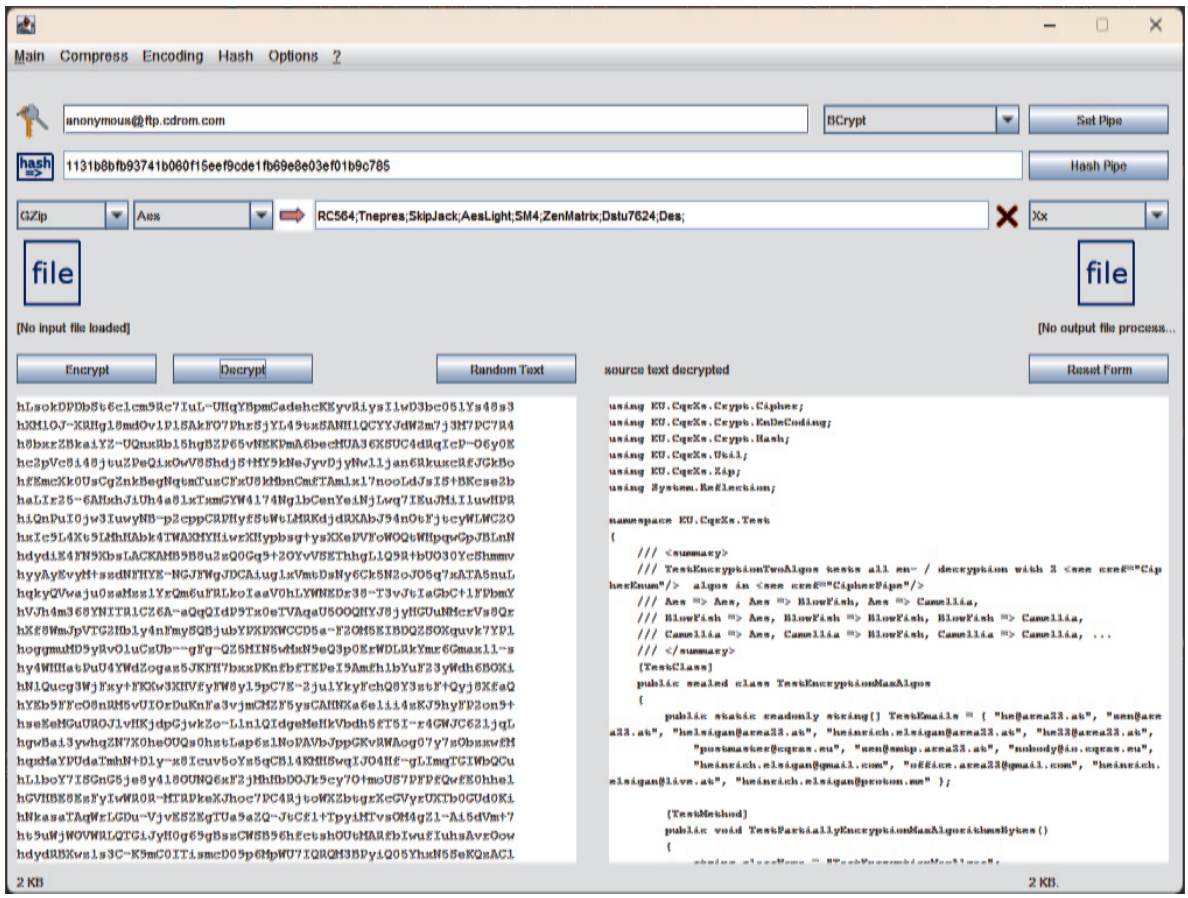

Figure 4: javax.swing.JFrame.java

3.3 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

3.4 Simple mode - SecureCipherPipe

In the simple mode SecureCipherPipe is used as CipherPipe. Each stage of the pipe will not be encrypted with the same secure key; furthermore, each pipe stage will be encrypted with a secure hashed key with a different hash.

3.4.1 SecureCipherPipe fragments in java

```
package eu.cqrxs.crypt.cipher;
import eu.cqrxs.crypt.encoding.+,

// [...]

/**
 * SecureCipherPipe is symmetric block cipher encryption and decryption pipe line
 */
public class SecureCipherPipe extends CipherPipe {
```

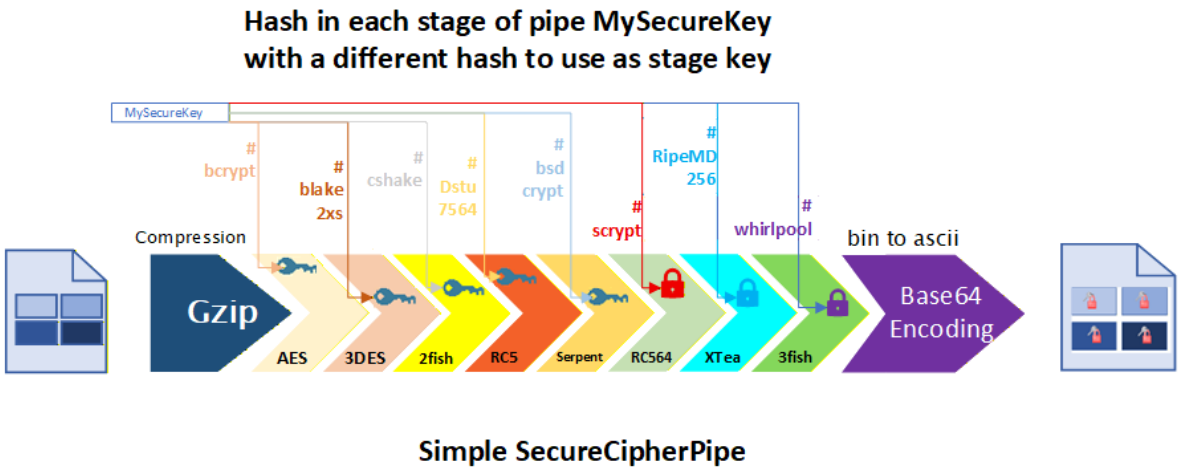

Figure 5: Simple SecureCipherPipe

```
String cipherKeyHash = "";
final static KeyHash[] secureHashes = {
   KeyHash.BCrypt, KeyHash.Blake2xs,  KeyHash.CShake, KeyHash.Dstu7564,
   KeyHash.OpenBSDCrypt, KeyHash.SCrypt, KeyHash.RipeMD256, KeyHash.Whirlpool };

/**
 * parameterless constructor of SecureCipherPipe
 */
public SecureCipherPipe() {
    cipherKeyHash = "";
    inPipe = new CipherEnum[0];
    encodeType = EncodeEnum.Base64;
    zType = ZipType.GZip;
    cMode2 = CipherMode2.CFB;
}

/**
 * SecureCipherPipe constructor with following parameters
 * @param cipherEnums an array of {@link CipherEnum}
 * @param maxpipe maximum pipeline size {@link eu.cqrxs.util.Constants}.MAX_PIPE_LEN
 * @param encType {@link EncodeEnum}
 * @param zpType {@link ZipType}
 * @param cmode2 {@link CipherMode2}
 */
public SecureCipherPipe(CipherEnum[] cipherEnums, int maxpipe, EncodeEnum encType,
                                        ZipType zpType, CipherMode2 cmode2) {
    // What ever is entered here as parameter, maxpipe has to be not greater 8, because of no such agency
    maxpipe = ((maxpipe > Constants.MAX_PIPE_LEN) ? Constants.MAX_PIPE_LEN : maxpipe); // if somebody wants

    int isize = Math.min(((int)cipherEnums.length), ((int)maxpipe));
    inPipe = new CipherEnum[isize];
    for (int ib = 0; (ib < cipherEnums.length && ib < isize); ib++) {
        inPipe[ib] = cipherEnums[ib];
    }
    cMode2 = cmode2;
    encodeType = encType;
    zType = zpType;
}

// [...]
```

```
/**
 * SecureCipherPipe ctor with array of user key bytes
 * @param keyBytes user key bytes
 * @param maxpipe maximum length {@link Constants}.MAX_PIPE_LEN
 * @param encType {@link EncodeEnum}
 * @param zpType {@link ZipType}
 * @param cmode2 {@link CipherMode2}
 */
public SecureCipherPipe(byte[] keyBytes, int maxpipe, EncodeEnum encType,
                                ZipType zpType, CipherMode2 cmode2) {
    // What ever is entered here as parameter, maxpipe has to be not greater 8, because of no such agency
    maxpipe = ((maxpipe > Constants.MAX_PIPE_LEN) ? Constants.MAX_PIPE_LEN : maxpipe); // if somebody wants
    // [...]
}

/**
 * Constructs a SecureCipherPipe from key and hash
 * @param key users secret key per default email address
 * @param encType {@link EncodeEnum}
 * @param zpType {@link ZipType}
 * @param cmode2 {@link CipherMode2}
 */
public SecureCipherPipe(String key, EncodeEnum encType, ZipType zpType, CipherMode2 cmode2) {
    this(CryptHelper.getKeyBytesSingle(key, 16), Constants.MAX_PIPE_LEN, encType, zpType, cmode2);
    cipherKeyHash = key;
}

/**
 * SecureCipherPipe constructor with single users key argument
 * all other parameters are set to default
 * @param key only users secret key
 */
public SecureCipherPipe(String key) {
    this(key, EncodeEnum.Base64, ZipType.GZip, CipherMode2.CFB);
    cipherKeyHash = key;
}

/**
 * encryptBytesFast generic encrypt bytes to bytes
 * @param inBytes array of bytes
 * @param cipherAlgo {@link CipherEnum}
 * @param hashKey users secret key for encryption
 * @param cmode2 {@link CipherMode2}
 * @return byte array of encrypted bytes
 */
public static byte[] encryptBytesFast(byte[] inBytes, CipherEnum cipherAlgo, String hashKey,
                                      CipherMode2 cmode2) throws InvalidCipherTextException {
    if (hashKey == null || hashKey.length() < 1)
        throw new IllegalArgumentException("hashKey");
    byte[] encryptBytes = inBytes;
    CryptParams cpParams = new CryptParams(cipherAlgo, hashKey, hashKey, cmode2);

    switch (cipherAlgo) {
        case ZenMatrix:
            encryptBytes = (new ZenMatrix(hashKey, hashKey, false, KeyHash.Hex)).encrypt(inBytes, true);
            break;
        default:
            CryptBounceCastle cryptBounceCastle = new CryptBounceCastle(cpParams, true);
            encryptBytes = cryptBounceCastle.encrypt(inBytes);
            break;
    }
```

```
    return encryptBytes;
}

/**
 * Generic decrypt bytes to bytes
 * @param cipherBytes enrypted bytes
 * @param cipherAlgo {@link CipherEnum}
 * @param hashKey users secret key
 * @param cmode2 {@link CipherMode2}
 * @return decrypted bytes for one cipher algo
 */
public static byte[] decryptBytesFast(byte[] cipherBytes, CipherEnum cipherAlgo, String hashKey,
                                      CipherMode2 cmode2) throws InvalidCipherTextException {
    if (hashKey == null || hashKey.length() == 0)
        throw new IllegalArgumentException("hashKey");
    byte[] decryptBytes = cipherBytes;
    CryptParams cpParams = new CryptParams(cipherAlgo, hashKey, hashKey, cmode2);

    switch (cipherAlgo) {
        case ZenMatrix:
            decryptBytes = (new ZenMatrix(hashKey, hashKey, false, KeyHash.Hex)).decrypt(cipherBytes);
            break;
        default:
            CryptBounceCastle cryptBounceCastle = new CryptBounceCastle(cpParams, true);
            decryptBytes = cryptBounceCastle.decrypt(cipherBytes);
            break;
    }
    return decryptBytes; // TODO: EnDeCodeHelper.GetBytesTrimNulls(decryptBytes);
}

/**
 * merryGoRoundEncrpyt starts merry to go arround from left to right in clock hour cycle
 * @param inBytes plain byte[] to encrypt
 * @param hashKey user secret key to use for all symmetric cipher algorithms in the pipe
 * @param cmode2 {@link CipherMode2}
 * @return encrypted byte[]
 */
public byte[] merryGoRoundEncrpyt(byte[] inBytes, String hashKey, CipherMode2 cmode2)
                throws InvalidCipherTextException {
    if (inPipe.length == 0)
        return inBytes;
    if ((hashKey == null && cipherKeyHash == null) ||
        (hashKey.length() == 0 && cipherKeyHash.length() == 0))
            throw new IllegalArgumentException("hashKey");
    int merry = 0;
    byte[] encryptedBytes = new byte[inBytes.length];
    System.arraycopy(inBytes, 0, encryptedBytes, 0, inBytes.length);
    cipherKeyHash = (hashKey != null && hashKey.length() > 0) ? hashKey : cipherKeyHash;

    for (CipherEnum cipher : inPipe) {
        String cipherHashKey = secureHashes[(merry % secureHashes.length)].hash(cipherKeyHash);
        if ((++merry) > (secureHashes.length -1)) merry = 0;
        encryptedBytes = encryptBytesFast(inBytes, cipher, cipherHashKey, cmode2);
        inBytes = encryptedBytes;
    }

    return encryptedBytes;
}

/**
 * decrpytRoundGoMerry against clock turn -
 *    starts merry to turn arround from right to left against clock hour cycle
```

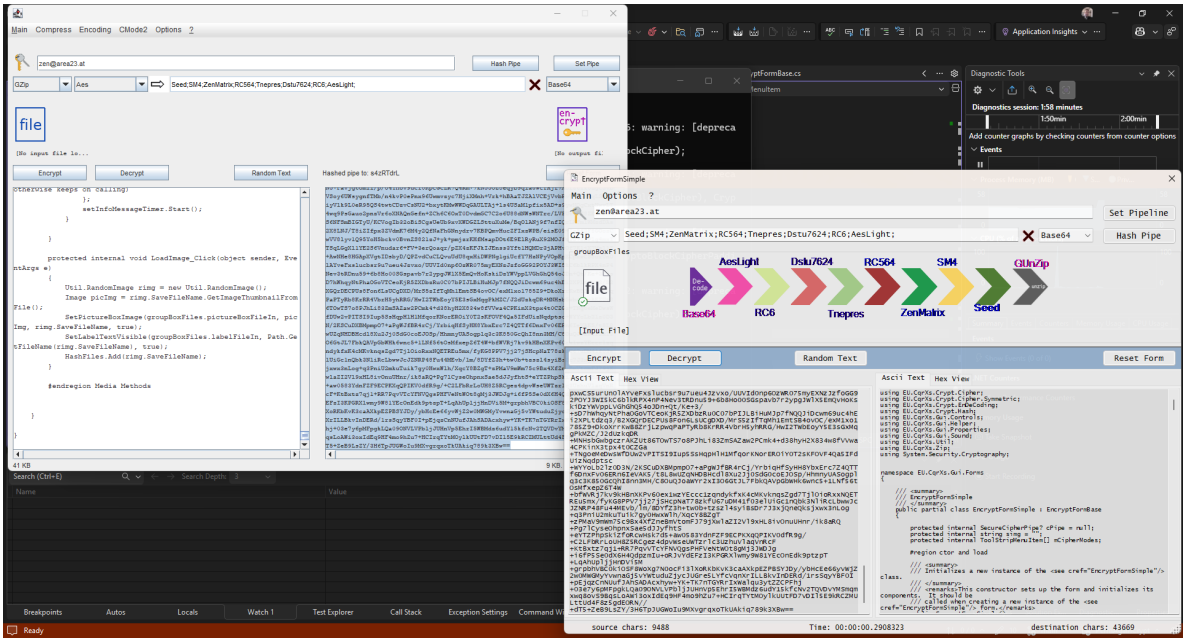

Figure 6: En-/Decryption in simple mode between C# and Java

```
     * @param cipherBytes encrypted byte array
     * @param hashKey user secret key, normally email address
     * @param cmode2 {@link CipherMode2}
     * @return byte[]
     */
    public byte[] decrpytRoundGoMerry(byte[] cipherBytes, String hashKey, CipherMode2 cmode2)
                    throws InvalidCipherTextException {
        if (inPipe.length == 0)
            return cipherBytes;
        if ((hashKey == null && cipherKeyHash == null) ||
            (hashKey.length() == 0 && cipherKeyHash.length() == 0))
                throw new IllegalArgumentException("hashKey");
        cipherKeyHash = (hashKey != null && !hashKey.isEmpty()) ? hashKey : cipherKeyHash;
        int roundsGo = secureHashes.length - 1;
        byte[] decryptedBytes = new byte[cipherBytes.length];

        for (CipherEnum cipher : getOutPipe()) {
            String cipherHashKey = secureHashes[(roundsGo % secureHashes.length)].hash(cipherKeyHash);
            if ((--roundsGo) < 0) roundsGo = secureHashes.length - 1;
            decryptedBytes = decryptBytesFast(cipherBytes, cipher, cipherHashKey, cmode2);
            cipherBytes = decryptedBytes;
        }

        return decryptedBytes;
    }
    // [...]
}
```

3.4.2 En-/Decryption between C# Java in Simple mode

Here we can see, that simple mode is intercompatible between java and C#. Here you see a text encrypted in simple mode in java and decrypted in C#.

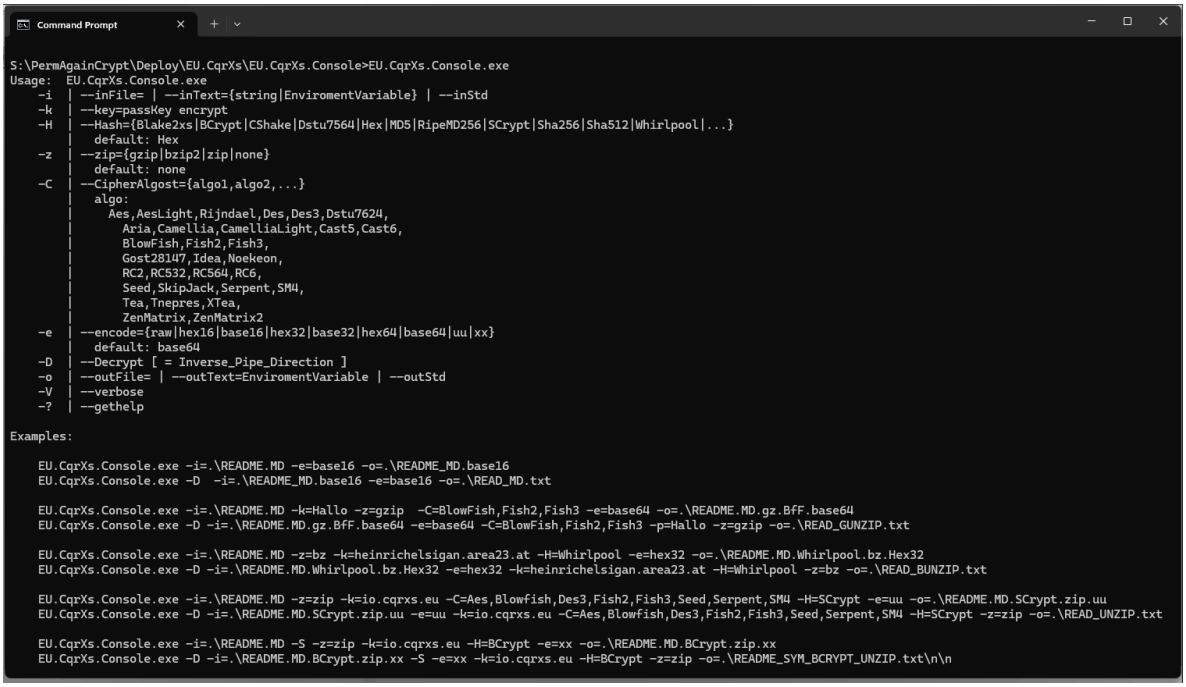

Figure 7: Windows EU.CqrXs.Console.exe

4 Console and cmd programs

4.1 Windows Console

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

4.1.1 Usage: 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|Hex|MD5|RipeMD256|SCrypt|Sha256|Sha512|Whirlpool|...}
        |   default: Hex
    -z  | --zip={gzip|bzip2|zip|none}
        |   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
    -e  | --encode={raw|hex16|base16|hex32|base32|hex64|base64|uu|xx}
        |   default: base64
    -D  | --Decrypt [ = Inverse_Pipe_Direction ]
    -o  | --outFile= | --outText=EnviromentVariable | --outStd
    -S  | -simpleMode
    -V  | --verbose
    -?  | --gethelp
```

4.1.2 Examples: EU.CqrXs.Console.exe

```
EU.CqrXs.Console.exe -i=.\README.MD -e=base16 -o=.\READMD.base16
EU.CqrXs.Console.exe -D  -i=.\READMD.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
```

5 Theory

5.1 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[Els24], 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

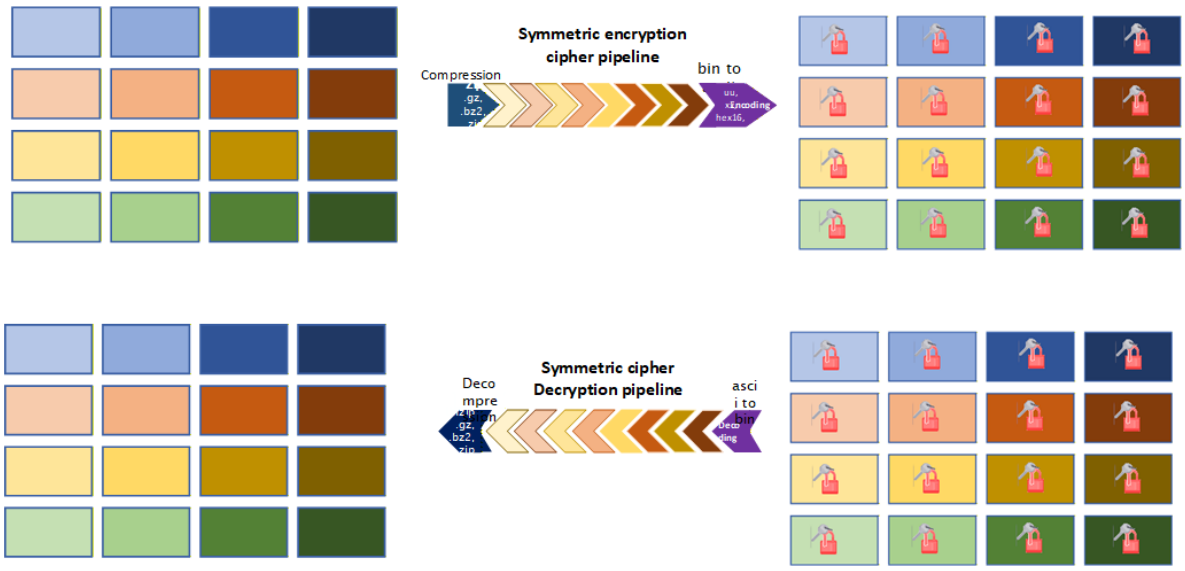

Figure 8: Symmetric BlockCipher Structure

5.2 What are advantages and disadvantages of Symmetric Block Cipher

5.2.1 Advantages

Since symmetric block cipher ciphers each block in the same encrypting way

parallel processing can be implemented quiet easy with average performance bust on huge multi-processore machines.

5.2.2 Disadvantages

Since we know often concrete structure of a file header, because of more static tableized structure inside file header and the sepecific binary format signature (MIT magic cookie)

REPLAY attacs could be used by trying encode only the 1st symmetric cipher block with some heuristic headers and possible keys.

5.2.3 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.

5.3 Mathematical theory

Consider that there are full bijective desterministic inversible functions, where inverse function

to $y_x = F(x, ...)$

is $x_y = f(y, ..)$

then inverse function

to $y_x = F(G(H(I(J(K(L(M(N(x, ...)))))))))$

is $x_y = n(m(l(k(j(i(h(g(f(y, ...)))))))))$.

You can see a mappings from ascii-8 $\Longrightarrow ascii - 8$,

also always as Matrix from R256 $\Longrightarrow R256$

or hexadecimal from Rx100 $\Longrightarrow Rx100$

you can see mappings from UTF-8 $\Longrightarrow UTF - 8$

...

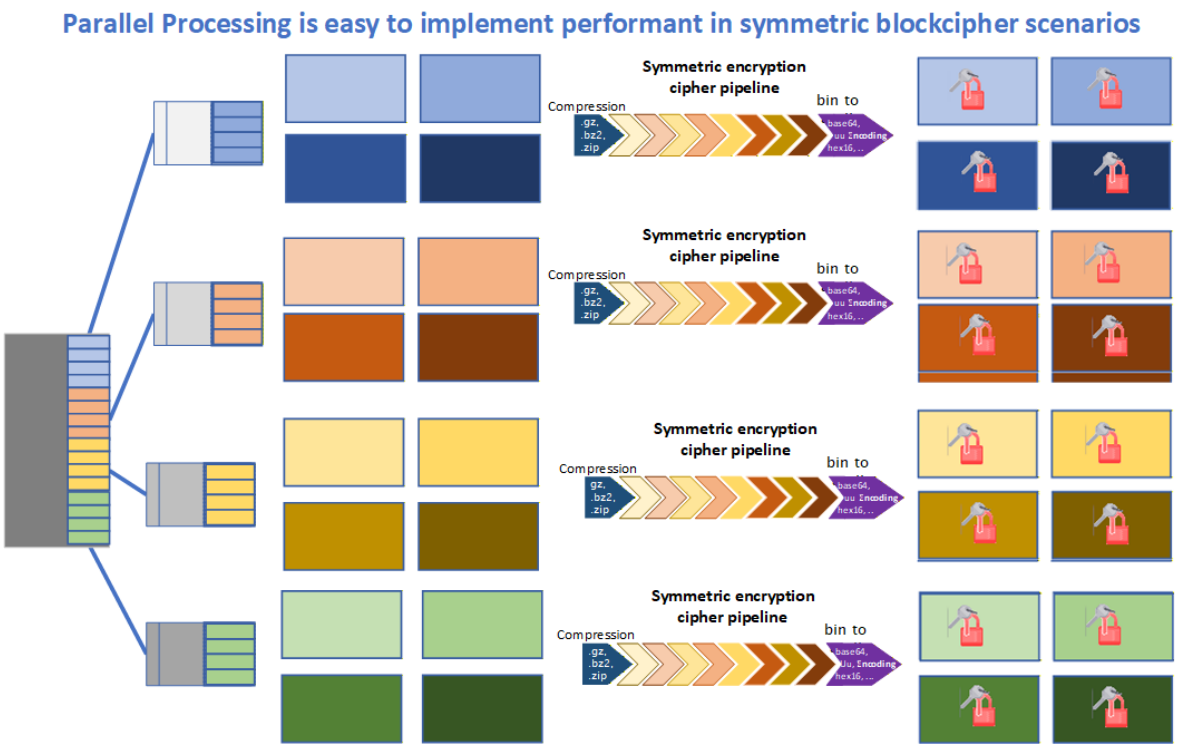

Figure 9: Symmetric BlockCipher Parallelism

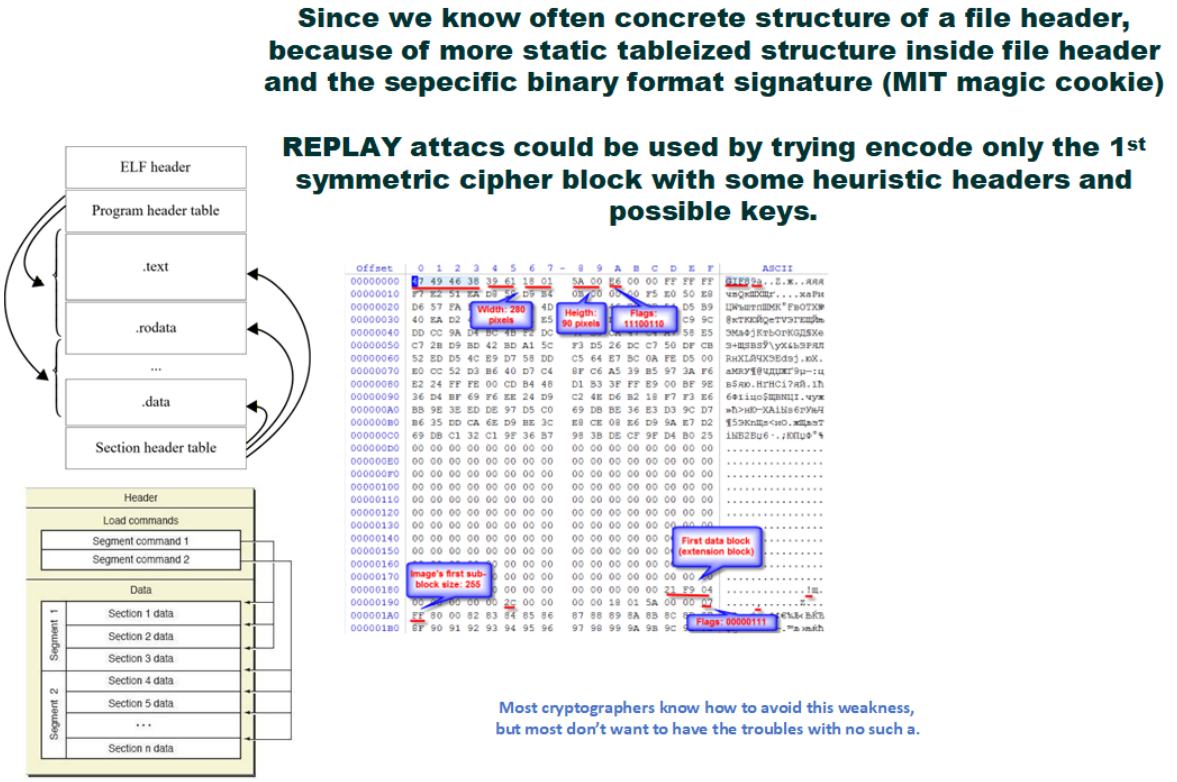

Figure 10: Symmetric BlockCipher Header 1st Block

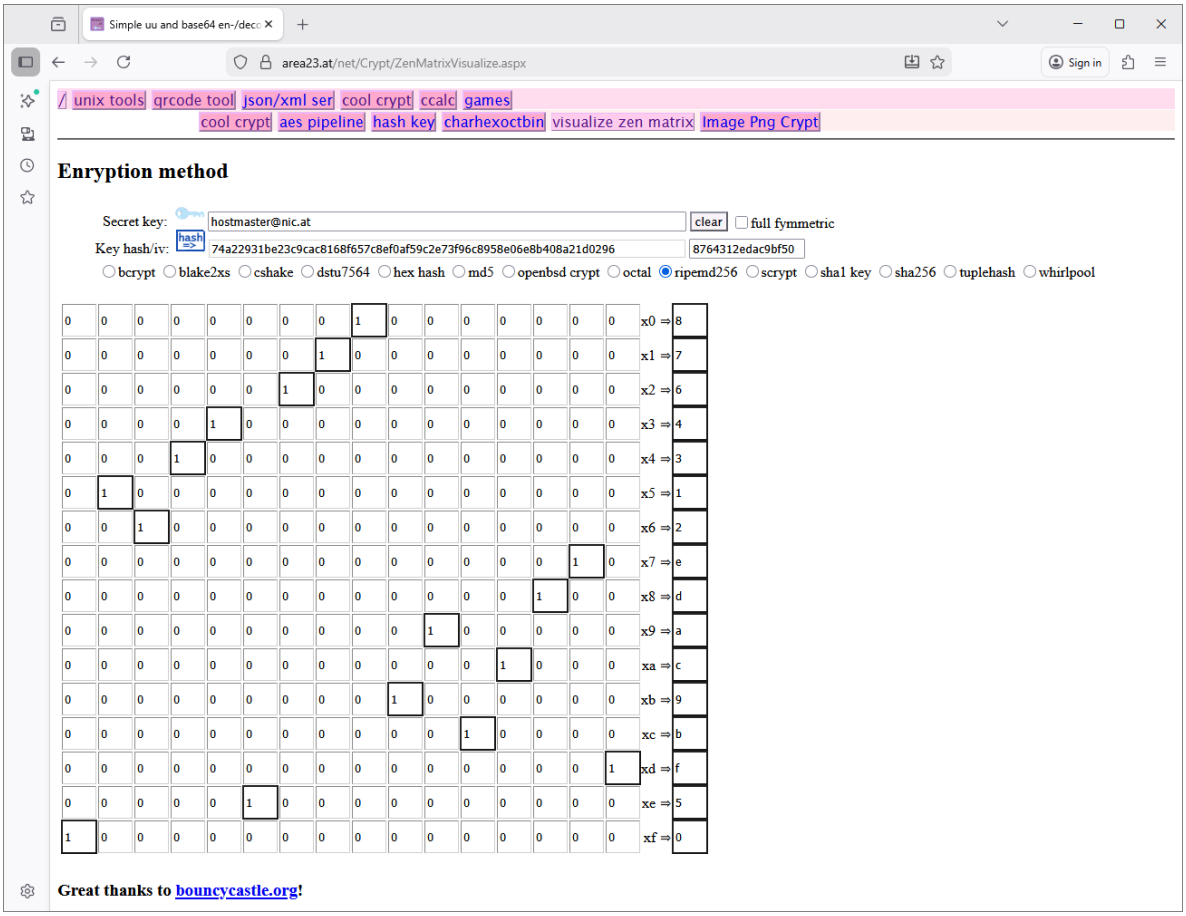

Figure 11: ZenMatrix Visualize

5.4 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.

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

6 Code Examples

6.1 C# 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.CFB);

// 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.CFB);

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

6.2 Java Example

```
boolean reverseDirection = false;       // variables we need later for cipher pipe
ZipType zipType = ZipType.None;
EncodeEnum encodingType = EncodeEnum.None;
KeyHash keyHash = KeyHash.Hex;
String inString = "Hallo, byte[] inside String will converted soonly!", outString = "";
String passKey = "MySecureKey", optCryptAlgos = "Aes,Des,Des3,Blowfish,Fish2,Fish3";
String[] algos = optCryptAlgos.split(",;:");
byte[] outBytes = null, inBytes = inString.getBytes(StandardCharsets.UTF_8);

CipherPipe pipe;                        // Create cipher pipe 4 en-/decrypting
if (passKey == null || passKey.isEmpty() || algos.length > 0) {
    pipe =  new CipherPipe(algos, Constants.MAX_PIPE_LEN,
            encodingType, zipType, keyHash, CipherMode2.CFB);  // CFB is default CipherMode
    verbout("Created pipe without passkey: " + pipe.getPipeString());
} else {
    pipe = new CipherPipe(passKey, keyHash.hash(passKey),
            encodingType, zipType, keyHash, CipherMode2.CFB); // CFB instead old ECB
    verbout("Created pipe with passkey=" + passKey + " pipe=" + pipe.getPipeString());
}
if (!reverseDirection) {                // encrypt
    PrintPipe(pipe, reverseDirection);
    try {                               // CipherPipe encrypt encode
        passKey = (passKey.length() == 0) ? " " : passKey;
        outBytes = pipe.encryptEncodeBytes(inBytes,
                    passKey, keyHash.hash(passKey),
                    encodingType, zipType, keyHash, CipherMode2.CFB);
    } catch (Exception exi) {
        exi.printStackTrace();
    }
    outString = new String(outBytes);
} else {                                // decrypt
    String inString = new String(inBytes);
    PrintPipe(pipe, reverseDirection);
    try {                               // CipherPipe decode decrypt
        passKey = (passKey == null || passKey.isEmpty()) ? "" : passKey;
        outBytes = pipe.decodeDecrpytBytes(inBytes,
                    passKey, (passKey.isEmpty() ? "" : keyHash.hash(passKey)),
                    encodingType, zipType, keyHash, CipherMode2.CFB);
    } catch (Exception exi) {
        exi.printStackTrace();
```

```
    }
}
```

6.3 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

[Els24] Heinrich Elsigan. Making symmetric cipher encryption meta permutating again. *Google Blogger*, 2024.