

XOR-Concatenation combined with Bit Conversion

(Ernst Erich Schnoor)
eschnoor@multi-matrix.de

Abstract

I present a procedure which combines Xor-concatenated digital data with structural changes of their bits (bit conversion). "Bit conversion" is a change in the number of bits in the respective character. The main feature is a complete disconnection between plaintext and ciphertext. The procedure is intended to achieve secure solutions in cryptographic operations resistant to cryptanalysis.

Key words

XOR-concatenation, bit conversion, matrix generation, cryptographic mechanism, cipher alphabet, congruence of length, coding by pointers, dynamic "one-time-pad", ambiguous brute force

A. Introduction

Encryption systems normally use XOR operations on to be encrypted or decrypted data and key data. But, as known simple XOR-encryptions are weak and preferred objects of attacks. On principle XOR-concatenation results in an equal size of key sequences and original plain text. This means congruence of length between plaintext and ciphertext and it is laid open to all usual attacks [#1].

To complicate this an additional measure is included: "bit conversion". Relations between plaintext and ciphertext, like repetition patterns, word combinations, frequency structures and two-digit-groups, are avoided in total. The procedure forms 8 cipher characters out of 7 plaintext characters. There is no congruence of length. Usual attacks will have no success and even brute force will remain ambiguous.

The basic requirement is a mechanism - author's designation: CypherMatrix® – generating among others unlimited keys and cipher alphabets to achieve xor-concatenation and bit conversion [#2].

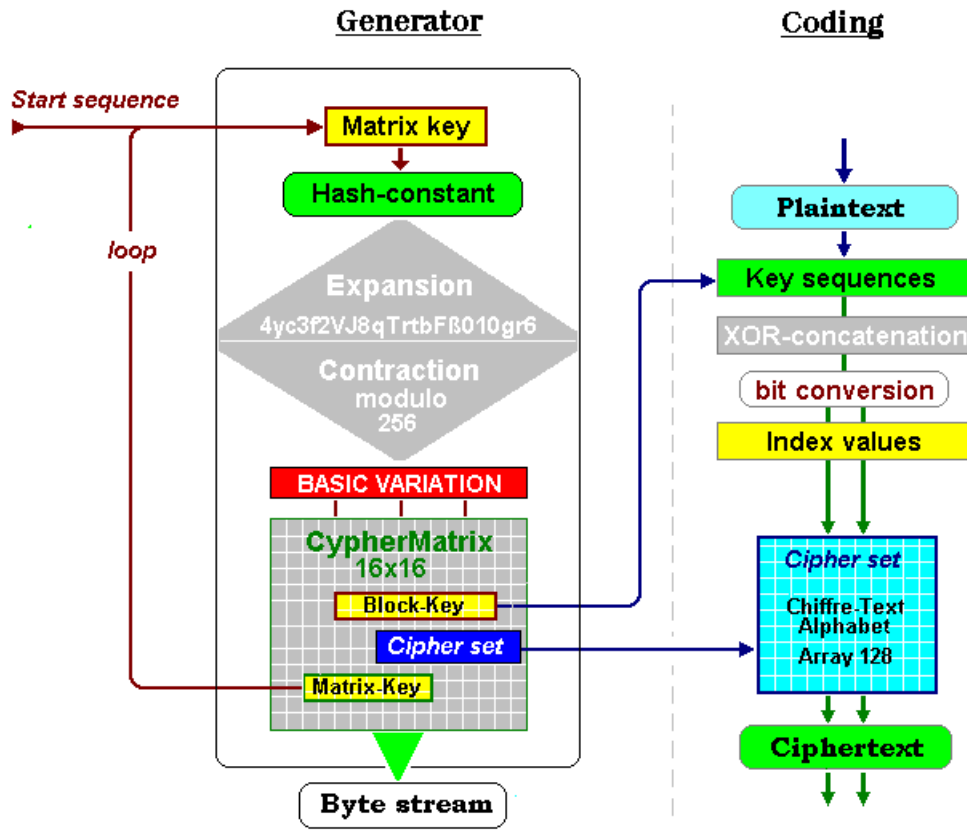
The procedure is demonstrated in three steps:

1. XOR-concatenating plaintext characters and key characters,
2. deviding 8-bit sequences of the XOR-concatenation into 7-bit sequences as indices and
3. addressing ciphertext characters by the decimal values of the 7-bit indices.

B. Cryptographic mechanism "CypherMatrix"

A secret "start sequence" of optimal 42 characters (pass phrase) as a joint key – interchanged between sender and addressee - generates an identical cryptographic mechanism at both sites: sender and addressee. The start sequence initializes a collision-free one-way-hash function, which constitutes a specific cryptographic "hard problem" [#3].

The following scheme illustrates the functional connections:



Both areas - "Generator" and "Coding" - can be used in particular and separate manner. But in case of encryption they have to be connected, of course. The generator repeats in utmost short intervals of e.g. 63 plaintext characters and generates a permuted array of 16x16 characters (CypherMatrix). The procedure extracts the following parameters anew and different in each cycle:

1. Key sequences (e.g. 63 characters) as block keys in order to serve at XOR-concatenation,
2. an independent cipher-alphabet of 128 characters (array 0...127),
3. a matrix key of 42 characters as start sequence to initialize the next cycle (loop) and
4. all elements of the matrix to create an unlimited random "byte stream".

Portions of plaintext (e.g. 63 characters) to be encrypted and the respective block key extracted from the CypherMatrix are always of equal length. In principle, that may be denoted as a dynamic "one-time-pad" for that definite detail.

To demonstrate the steps specific to carry out the procedure we choose a "start sequence" with 42 bytes:

„Horse racing on the banks of Clearwaterbay“

As plaintext we use the following quotation stored in the file: CANNERY.TXT:

The WORD is a symbol and a delight which sucks up men and scenes, trees, plants, factories, and Pekinese. Then the Thing becomes the Word and back to Thing again, but warped and woven into a fantastic pattern. The word sucks up Cannery Row, digests it and spews it out, and the Row has taken the shimmer of the green world and the sky-reflecting seas.

John Steinbeck, Cannery Row, New York 1945

With these inputs and the following parameters

Matrix key (length sequence):	42 bytes
Block key (length sequence):	63 bytes
Expansion factor (system):	base 77
User code:	1

the >CypherMatrix< device calculates the processing Data for the first cycle:

Hash constant:	$1680 + 1 = 1681$
length (entropy):	336 bits

Control sum (Hk):	6823616
Partial hash value (Hp):	599447842887
Total hash value (Hp+Hk):	599454666503

Alpha (Hp+Hk MOD 255)+1:	189
Beta (Hk MOD 169)+1:	73
Gamma ((Hp+Code) MOD 196)+1:	133

The „start sequence“ controls the first cycle. The „matrix key“ extracted from the CypherMatrix is related back (loop) to the beginning as new start sequence to initialize the next cycle.

Alpha determines the beginning of extracting 128 elements out of the CypherMatrix as an independent “cipher alphabet” for encrypting the actual plain text block,

Beta fixes the starting point of getting a “block key” (63 elements) for concatenation with the actual plain text block and

Gamma determines the offset of extracting a new “matrix key” (42 elements) to relate back to the beginning as “start sequence” for the next cycle.

The CypherMatrix of the first cycle results as follows:

CypherMatrix (16x16)

1	97	79	87	9E	25	F6	E1	80	7C	E5	5B	BF	E8	19	DE	16	16
17	8F	F9	50	11	72	61	41	E3	AA	36	64	A2	B1	91	FC	5D	32
33	63	05	2E	4A	EC	02	B9	A9	49	F0	0F	95	21	24	FD	60	48
49	7D	3B	22	CB	5E	C8	A6	1A	C9	5C	27	52	DC	13	74	96	64
65	B0	3E	C3	75	1F	71	8C	F1	CF	F8	B8	A4	D1	0E	59	A1	80
81	AE	DF	6C	EA	28	D7	B3	D3	42	65	9F	B2	C2	43	06	12	96
97	7F	69	85	6D	2F	94	08	E0	ED	2B	BB	B6	C7	E7	6F	D6	112
113	04	23	68	89	DD	3A	81	CE	F7	AD	32	CD	4E	78	B4	4D	128
129	7A	AF	1B	99	20	7B	55	4F	62	BE	0D	82	88	1E	9B	C1	144
145	C0	57	F5	26	DB	BD	35	03	93	7E	56	83	66	6E	4C	D9	160
161	70	D2	1D	BA	33	31	3D	5F	47	E2	53	15	EE	0A	2D	54	176
177	34	E9	92	D4	DA	4B	37	09	D5	D8	AB	45	01	A0	8E	3C	192
193	84	F4	77	10	0B	58	FE	30	EB	E4	C4	73	6B	FB	FF	0C	208
209	3F	07	AC	A7	6A	CC	98	46	BC	F2	CA	44	8B	29	F3	C5	224
225	18	90	51	B5	FA	E6	8A	9C	2A	48	86	00	38	14	9D	8D	240
241	EF	76	39	B7	C6	1C	40	A3	2C	9A	17	67	A8	A5	5A	D0	256

Extracting the „Ciphertext alphabet“ from the respective CypherMatrix is achieved by the parameter Alpha at position 189 (+ omitted 4 elements = 193). Elements with hex values **00** to **19**, **22** and **2C** are skiped.

Ciphertext alphabet (array 128 characters)

Index	1 - 16:	„ ô w X p 0 ë ä Ä s k û ? ¬ § j
Index	17 - 32:	Ì ~ F ¼ ò Ê D <) ó Å □ Q μ ú æ
Index	33 - 48:	Š œ * H † 8 □ □ i v 9 · Æ @ £ š
Index	49 - 64:	g ¨ ¥ Z Ð - y ‡ ž % ö á € å [
Index	65 - 80:	¿ è □ ù P r a A ã ª 6 d ç ` ü]
Index	81 - 96:	c . J ì ´ © I ð • ! \$ ý ` } ; Ë
Index	97 - 112:	^ È É \ ' R t - > ã u q Œ ñ Ĩ
Index	113 - 128:	ø , ¨ Ñ Y ; ® l ê (× º Ó B e Ÿ

Ciphertext alphabet: hexadecimal

84	F4	77	58	FE	30	EB	E4	C4	73	6B	FB	3F	AC	A7	6A
CC	98	46	BC	F2	CA	44	8B	29	F3	C5	90	51	B5	FA	E6
8A	9C	2A	48	86	38	9D	8D	EF	76	39	B7	C6	40	A3	9A
67	A8	A5	5A	D0	97	79	87	9E	25	F6	E1	80	7C	E5	5B
BF	E8	8F	F9	50	72	61	41	E3	AA	36	64	A2	91	FC	5D
63	2E	4A	EC	B9	A9	49	F0	95	21	24	FD	60	7D	3B	CB
5E	C8	A6	C9	5C	27	52	74	96	3E	C3	75	71	8C	F1	CF
F8	B8	A4	D1	59	A1	AE	6C	EA	28	D7	B3	D3	42	65	9F

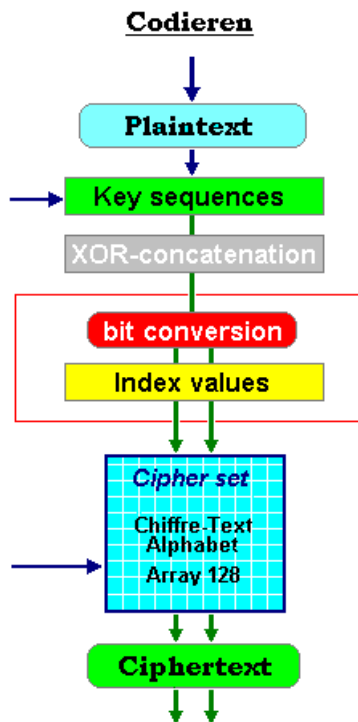
Matrix key (at offset: Gamma = 135 --> 42 bytes)

55 4F 62 BE 82 88 1E 9B C1 C0 57 F5 26 DB BD 35 03 93 7E 56 83
66 6E 4C D9 70 D2 1D BA 33 31 3D 5F 47 E2 53 15 EE 2D 54 34 E9

Gamma 133 is added by 2 positions to 135 because 2 foregoing elements are skipped.

C. XOR-concatenating plaintext and block keys

The following graph shows the sequential steps:



The first plaintext detail (block of 63 bytes) running through the procedure reads as follows:

The WORD is a symbol and a delight which sucks up men and scene

54 68 65 20 57 4F 52 44 20 69 73 20 61 20 73 79 6D 62 6F 6C 20
61 6E 64 20 61 20 64 65 6C 69 67 68 74 20 77 68 69 63 68 20 73
75 63 6B 73 20 75 70 20 6D 65 6E 20 61 6E 64 20 73 63 65 6E 65

Block key (at offset: Beta = 74 --> 63 bytes)

B8 A4 D1 0E 59 A1 AE DF 6C EA 28 D7 B3 D3 42 65 9F B2 C2 43 06
12 7F 69 85 6D 2F 94 08 E0 ED 2B BB B6 C7 E7 6F D6 04 23 68 89
DD 3A 81 CE F7 AD 32 CD 4E 78 B4 4D 7A AF 1B 99 20 7B 55 4F 62

Beta 73 is added by 1 position to 74 because 1 foregoing element is skipped.

XOR - concatenation before bit-conversion

```
EC CC B4 2E 0E EE FC 9B 4C 83 5B F7 D2 F3 31 1C F2 D0 AD 2F 26
73 11 0D A5 0C 0F F0 6D 8C 84 4C D3 C2 E7 90 07 BF 67 4B 48 FA
A8 59 EA BD D7 D8 42 ED 23 1D DA 6D 1B C1 7F B9 53 18 30 21 07
```

The XOR concatenation is a standard operation. As there is no real protection with the simple XOR concatenation yet [#4] the procedure adds a further step: "bit conversion".

D. "Bit Conversion" indices addressing the ciphertext alphabet

Data-technically, the "bit conversion" is a change in the numbers of bits in the XOR-concatenated elements. Best known is the procedure "Coding Base 64" which converts 8-bit sequences into a series of 6-bit sequences. Decimal values of these sequences are indices to a cipher alphabet of 64 characters. Compared with the plaintext the length of the ciphertext expands at a ratio of 6:8 [#5].

The here presented procedure is based on cipher alphabets of 128 characters extracted out of the respective "CypherMatrix". Thus the cipher alphabet changes with each cycle.

In order to perform the bit conversion the ciphertext characters are stored in an array of 128 elements. The series of 8-bit sequences of the XOR concatenations (0...255) are divided into series of 7-bit sequences (0...127) as indices (+1) to address the characters stored in the ciphertext array (1...128).

The selected plaintext is processed as follows:

XOR-concatenation of plaintext and block key

```
EC CC B4 2E 0E EE FC 9B 4C 83 5B F7 D2 F3 31 1C F2 D0 AD 2F 26
73 11 0D A5 0C 0F F0 6D 8C 84 4C D3 C2 E7 90 07 BF 67 4B 48 FA
A8 59 EA BD D7 D8 42 ED 23 1D DA 6D 1B C1 7F B9 53 18 30 21 07
```

demonstrated in 8-bit XOR-sequences

```
11101100 11001100 10110100 00101110 00001110 11101110 11111100
10011011 01001100 10000011 01011011 11110111 11010010 11110011
00110001 00011100 11110010 11010000 10101101 00101111 00100110
01110011 00010001 00001101 10100101 00001100 00001111 11110000
01101101 10001100 10000100 01001100 11010011 11000010 11100111
10010000 00000111 10111111 01100111 01001011 01001000 11111010
10101000 01011001 11101010 10111101 11010111 11011000 01000010
11101101 00100011 00011101 11011010 01101101 00011011 11000001
01111111 10111001 01010011 00011000 00110000 00100001 00000111
```

converted into 7-bit sequences

```
1110110 0110011 0010110 1000010 1110000 0111011 1011101 1111100
1001101 1010011 0010000 0110101 1011111 1011111 0100101 1110011
0011000 1000111 0011110 0101101 0000101 0110100 1011110 0100110
0111001 1000100 0100001 1011010 0101000 0110000 0011111 1110000
0110110 1100011 0010000 1000100 1100110 1001111 0000101 1100111
1001000 0000001 1110111 1110110 0111010 0101101 0010001 1111010
1010100 0010110 0111101 0101011 1101110 1011111 0110000 1000010
1110110 1001000 1100011 1011101 1010011 0110100 0110111 1000001
0111111 1101110 0101010 0110001 1000001 1000000 1000010 0000111
```

No bit is added and no bit is removed. In the series of "0" and "1" the order remain unchanged, only another grouping has been arranged (8-bit --> 7-bit). But the number of sequences change in a ratio of 7 plaintext characters to 8 index values. And because each index is assorted to a definite character the number of cipher characters change as well.

$$7 * (8\text{-bit}) = 56 \text{ bit} = 8 * (7\text{-bit})$$
$$x * (8\text{-bit}) = x*8 = (x*8)/7 (7\text{-bit})$$

Hexadecimal 7-bit values after > bit conversion <

```
76 33 16 42 70 3B 5D 7C 4D 53 10 35 5F 5F 25 73 18 47 1E 2D 05
34 5E 26 39 44 21 5A 28 30 1F 70 36 63 10 44 66 4F 05 67 48 01
77 76 3A 2D 11 7A 54 16 3D 2B 6E 5F 30 42 76 48 63 5D 53 34 37
41 3F 6E 2A 31 41 40 42 07
```

Decimal index values (+1) to address the array > Ciphertext alphabet <

```
119 52 23 67 113 60 94 125 78 84 17 54 96 96 38
116 25 72 31 46 6 53 95 39 58 69 34 91 41 49
32 113 55 100 17 69 103 80 6 104 73 2 120 119 59
46 18 123 85 23 62 44 111 96 49 67 119 73 100 94
84 53 56 66 64 111 43 50 66 65 67 8
```

Encrypted > Cipher text < derivated from > Ciphertext alphabet <

®ZD[øá}Ó‘iÏ—ËË8Ñ)Aú@0Đ;[Pœ\$igæøÿÉÏPR]0tãôl®ö@~×¹D|·ñËg[ªÉ}iĐ‡è[ñ9”è¿ä

```
AE 5A 44 8F F8 E1 7D D3 91 EC CC 97 CB CB 38 D1 29 41 FA 40 30
D0 3B 9D 25 50 9C 24 EF 67 E6 F8 79 C9 CC 50 52 5D 30 74 E3 F4
6C AE F6 40 98 D7 B9 44 7C B7 F1 CB 67 8F AE E3 C9 7D EC D0 87
E8 5B F1 39 A8 E8 BF 8F E4
```

The decryption works like this:

At the addressee the joint key as "start sequence" generates identically parameters and matrixes. The procedure searches in the array of the respective ciphertext alphabet for the received cipher characters, ascertains the index of the characters [1...128], converts each index (-1) into a binary 7-bit sequence and combines it to a 7-bit series. This series will be separated into 8-bit sequences. In one cycle 72 7-bit sequences result into 63 8-bit sequences. Each 8-bit value corresponds with the XOR-concatenation generated at the sender and will be reconverted to the initial plaintext by concatenating with the identically generated block key.

A fundamental effect of "bit conversion" becomes clear by this small example: The length of cipher text expands, in this case at a ratio of 7:8. The initial 7 plaintext characters have now become 8 converted ciphertext characters. Thus, the functional connection between plaintext and ciphertext is interrupted.

E. Cryptanalysis of the procedure

Compared with conventional bit-encryption the here presented procedure indicates a remarkable difference: in the first case the structure of the bits are changed, only. The number of bits in the plaintext characters and in the respective ciphertext character are principally equal (8-bit). A specific ciphertext character exist for a definite plaintext character. Insofar there is a uniform digital order system in the whole encryption process: $c_i = f(p_i)$. This is the reason why the encrypted text bears certain informations of the respective plaintext [#6].

1. „Coding by pointers“ is insignificant

In the second case with including "bit-conversion" there exists no uniform order system. The connection between plaintext (**p**) and ciphertext (**c**) is interrupted. At least two separate functions are necessary to realize the connection:

$$c = f [f_2 (f_1)]$$

$f_1 = f(p) =$ Converting XOR-concatenated plaintext bits into index values and
 $f_2 = f(f_1) =$ Index values to point to characters in the permuted ciphertext alphabet.

But the ciphertext characters as results of the second function are only "pointers" leading to the respective positions in the ciphertext array at the addressee.

However, the "pointers" do not contain any information about the plaintext. Therefore, if the encrypted message is not carrying any information about the initial plaintext the secret text is actually of no value to any attacker. In data-technical sense this is a "coding by pointers" [#7]. The only possible access is via the start sequence and by reproducing the correct "CypherMatrix" at the addressee.

The best known attacks on encrypted messages include structure analysis, "known plaintext attack", "chosen plaintext attack" and "brute force attack", and possibly also "differential" and "linear" cryptanalysis [#8]. These attacks are meant to filter out statistically acquirable regularities from the ciphertext which may possibly show a lead to the plaintext. However,

all these methods presuppose that plaintext and ciphertext are of an identical length.

Because of the bit-conversion and thereby caused the extension of ciphertext this necessary condition is not relevant here, at all. All possible attacks dispenses with a fundamental basis: the congruence of length. Thus, related to our procedure, we may forget all these attacks.

2. Searching backwards is ambiguous

But, an attempt of brute force attack could still be possible. There are three points of entry:

1. Searching for the "start sequence" at the beginning,
2. taking advantage of certain points during running procedure and
3. backwards searching from ciphertext to plaintext.

Performance of brute force first depends on length of the "start sequence", here about 42 bytes. Because all 256 ASCII-characters are available for the start sequence the key range extends maximal up to $256^{42} = 1.34E+101$ combinations. That results in a key length of 336 bits (entropy = 335.93). A definite success will take approximately $4.25E+90$ years to occur.

Moreover the key sequences must be found in one straight succession process. Results from foregoing attempts are helpless. Because of their length (about 42 bytes) dictionary attacks (on base of literary or colloquial quotations) will be beyond all hope. Best solution is choosing funny and unlikely expressions which are easy to remember and it's not necessary to write them down.

During the running procedure no inputs and no interruptions are possible.

Backward searching with "brute force" on base of the ciphertext will even have no success by following reasons:

Principally the attacker knows the ciphertext and the respective length of the encrypted message. Further he may know the method "CypherMatrix" with its three functions:

Plaintext --> key --> XOR-concatenation
8-bit XOR-concatenation --> bit-conversion --> 7-bit index values
7-bit index values --> ciphertext array --> ciphertext

First have to be found out the ciphertext array with 128 characters and their distribution. The values of the ciphertext characters (hex or binary) are not qualified for this task, because they are no functional part of the encryption procedure. The 7-bit values necessary for reconvertig to 8-bit sequences of the XOR-concatenation will be given only by the positions of the characters in the ciphertext array. Thus the distribution of cipher characters in the array have to be reconstructed first. The correct distribution can be found only by iterative selection of all involved characters.

To achieve this all possible distributions have to be tried out. But that alone is not sufficient, the respective "block key" has to be found out, as well. Iteration is the only possible way.

Nevertheless, if both unknown data - **distribution** and **block key** - will yet be found the efforts will have no success, which may be proven mathematically by the following facts:

As already pointed out the method includes three functions:

1. Plaintext --> key --> 8-bit XOR-sequences
2. 8-bit sequences --> 7-bit index-values
3. 7-bit index-values --> array (128) --> cipher text

Parameters **key** and **array** (128) are two variables independent from each other.

$$\begin{aligned} \text{cm} &= f [f_1 (\text{pn}, k_1), f_2 (b_1, b_2), f_3 (b_2, k_2)] \\ \text{pn} &= f [f_3 (\text{cm}, k_2), f_2 (b_2, b_1), f_1 (b_1, k_1)] \end{aligned}$$

f_x = function
pn = plaintext
 k_1 = key
 b_1 = 8-bit sequence
 b_2 = 7-bit index-value
 k_2 = array (128)
cm = cipher text

Finding out the cipher text **cm** and searching retrograde for the plaintext **pn** means forming equations with two unknown variables: k_1 and k_2 . This results in a definite solution only if one of the unknown can be derived from the other unknown variable or if there are two equations with the same variables. There are a lot of pairs **array/key** which result in readable plaintext but they don't match the correct plaintext [#9].

The reason is why there is no connection between the respective key and the cipher text alphabet generated in the same cycle. Their common root is the initial start sequence only. But to that position there is no way back (one way function). Insofar it is made evident that in this case "brute force" does not attain any definite result at all. Apparently this are comparable mathematical facts such as to the proof that an "one-time-pad" cannot be broken [#10].

F. Quintessence

If all conventional attacks against the CypherMatrix method do not lead to any success in consequence of bit conversion and missing "congruence of length" and by this reasons the only remaining way - "brute force" attack - will be hopeless then the definitely quintessence will be: The method is secure and unbreakable. Opposition against this statement will be accepted at every time and checked at length.

G. Appendix

You may read an appendix to this paper in which insignificant modifications of the start sequence are analysed and described. See at:

<http://www.telecypher.net/ConvertX.pdf>

H. References

- [# 1] Congruence of length, see: www.telecypher.net/EQLENGTH.HTM
- [# 2] Core of the CypherMatrix Method, see: www.telecypher.net/CORECYPH.HTM
- [# 3] CypherMatrix as dynamic Hash function, see:
www.telecypher.net/DYNAHASH.HTM and CMHash.pdf
- [# 4] Schneier, Bruce, Angewandte Kryptographie (German edition), Bonn 1995, p. 15, 16
- [# 5] Morin, Randy Charles, "<http://www.kbcafe.com/articles/HowTo.Base64.pdf>"
- [# 6] see: <http://www.telecypher.net/CORECYPH.HTM#Z21>
- [# 7] Meganet Corporation, Virtual Matrix Encryption
- [# 8] Wobst, Reinhard, Abenteuer Kryptologie, Bonn 1997, p. 56
- [# 9] Schneier, Bruce, a.a.O., p. 281, 282
- [#10] Schneier, Bruce, a.a.O., p. 17 and Wobst, Reinhard, a.a.O., p. 53

Munich, in June 2006