New Techniques in digital Cryptography

(Ernst Erich Schnoor)

Since centuries cryptographers used for enciphering exclusively single characters of their concerned language. When computers came up methods of ciphering changed to digital techniques. But computer on principle can distinguish between two states only : „present“ (voltage) and „not present“ (no voltage), in digits of number system on base 2: „**one**“ or „**zero**“. This connection is defined as a „**bit“,** as everybody knows.

1 Systematizing bit series

An information is multilateral. It comprises two bits, at least, in general: a **series** of **bits**. Their quantity is unlimited (1 to ∞). In order to work with bit series systematically they have to be systemized, that means: to be scaled and devided into definite segments (**units**). It is a similar phenomenon compared with the quantity of all numbers. Comparable to numerical theory bit series can be systemized in a significance order system. Hence, 8 bit sequences may be named as „**bitsystem on base 8**“.
In detail:

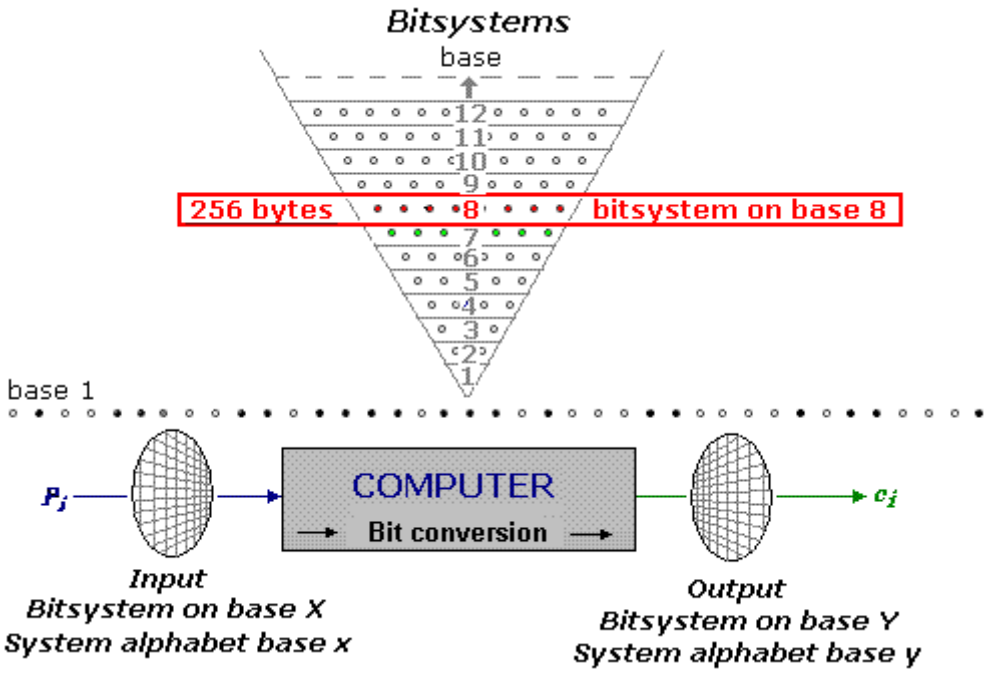
		system alphabet
bit sequences:	1-bit = bitsystem on base 1 = 2^1 signs =	2 units
	2-bit = bitsystem on base 2 = 2^2 signs =	4 units
	3-bit = bitsystem on base 3 = 2^3 signs =	8 units
	4-bit = bitsystem on base 4 = 2^4 signs =	16 units
	5-bit = bitsystem on base 5 = 2^5 signs =	32 units
	6-bit = bitsystem on base 6 = 2^6 signs =	64 units
	7-bit = bitsystem on base 7 = 2^7 signs =	128 units
	8-bit = bitsystem on base 8 = 2^8 bytes =	**256 bytes**
	9-bit = bitsystem on base 9 = 2^9 signs =	512 units
	10-bit = bitsystem on base 10 = 2^10 signs =	1024 units
	11-bit = bitsystem on base 11 = 2^11 signs =	2048 units
	12-bit = nitsystem on base 12 = 2^12 signs =	4096 units
	13-bit = bitsystem on base 13 = 2^13 signs =	8192 units
	14-bit = bitsystem on base 14 = 2^14 signs =	16384 units
	15-bit = bitsystem on base 15 = 2^15 signs =	32768 units
	16-bit = bitsystem on base 16 = 2^16 signs =	65536 units
	32-bit = bitsystem on base 32 = 2^32 signs =	4294967296 units

Up to now following bit sequences have been defined as units: bitsystem on base 1 comprising the signs: „0“ and „1“ or historically: „L“ and „H“. Bitsystem on base 4 knows the unit „nibble“ and 8-bit sequences the unit: „**byte**“. In bitsystem on base 16 are the units: „word“ and „dword“. All other bit series are marked by their number of bits (e.g. 32 bits, 64 bits) [#1].

Fundamentally connections are shown in following scheme:

System base	System-alphabet	Bit series in Units indexing	Lengths ratio
bitsystem on base 1	2	0 1 0 0 1 1 1 0 0 1 1 0 1 1 1 1 0 1 1 1 0 1 0 0 0 1 1 0 0 0 0 0.	1:8
bitsystem on base 2	4	01 00 11 10 01 10 11 11 01 11 01 00 01 10 00 01 01 10 00. 1 0 3 2 1 2 3 3 1 3 1 0 1 2 0 1 1 2 0	1:4
bitsystem on base 3	8	010 011 100 110 111 101 110 100 011 000 010 110 001 ... 2 3 4 6 7 5 6 4 3 0 2 6 1	1:2,66
bitsystem on base 4	16	0100 1110 0110 1111 0111 0100 0110 0001 0110 0010 ... 4 14 6 15 7 4 6 1 6 2	1:2
bitsystem on base 5	32	01001 11001 10111 10111 01000 11000 01011 00010 01.. 9 25 23 23 8 24 11 2	1:1,6
bitsystem on base 6	64	010011 100110 111101 110100 011000 010110 001001 ... 19 38 61 52 24 22 9	1:1,33
bitsystem on bases 7	128	0100111 0011011 1101110 1000110 0001011 0001001 ... 39 27 110 70 11 9	1:1,143
bitsystem on base 8 *operations*	256	01001110 01101111 01110100 01100001 01100010 011.. 78 111 116 97 98 4E 6F 74 61 62 N o t a b ... *streamcipher, blockcipher with congruence of length, Feistel-nets ECB. CBC. CFB. OFB. DES. IDEA. AES, RSA, differential and lineare cryptanalysis, and other programs*	1:1
bitsystem on base 9	512	010011100 110111101 110100011 000010110 001001100 156 445 419 22 76	1:1,79
bitsystem on base 10	1024	0100111001 1011110111 0100011000 0101100010 01100 313 759 280 354	1:1,6
bitsystem on base 11	2048	01001110011 01111011101 00011000010 11000100110 ... 627 989 194 1574	1:1,46
bitsystem on base 12	4096	010011100110 111101110100 011000010110 0010011001 1254 3956 1558 613	1:1,33
bitsystem on base xx	*div*	x x x x x	1: xxx

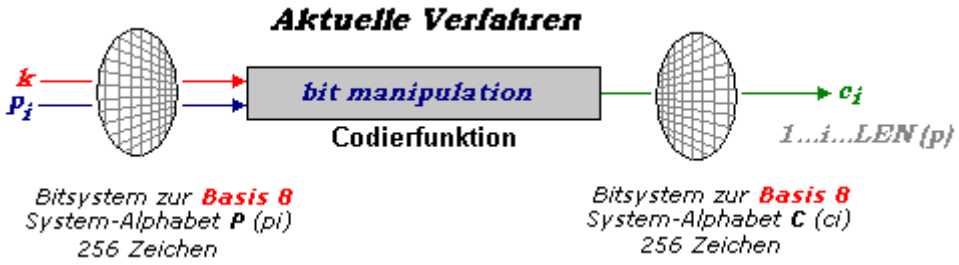
Structure of the significance order system is best described by the system pyramid turned on its head.



2 Current processing

By introduction digitally techniques the scientists appointed to further development superficially relaid upon bitsystem on base 8 and obviously rated an appropriate systemizing as subordinated. Almost all operations are performed in bit system on **base 8.**

All inputs are processed in bitsystem on base 8 and system alphabet on base 8 with 256 signs (00 to FF) and encrypted results (cipher text) as well are shown in characters of bitsystem on base 8 and system alphabet with 256 signs. In view of the fact between input and output there are manifold manipulated longer bit series that it is of no significance. Insofar – notwithstanding of manipulated bits inside coding function from input to output – all operations are performed in a uniform **order system**, in bitsystem on **base 8**.

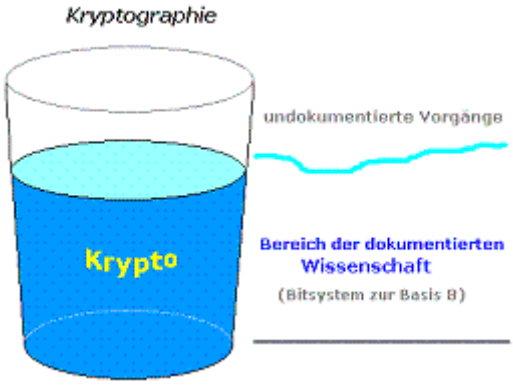


With this presuppositions and demand plaintext and ciphertext must have the same length [#2,#9] the **action range** of cryptography will be limited systematically to a more restricted extend: to bitsystem on **base 8** with a system alphabet (cipher alphabet) of **256 bytes**.

Insofar it may be a bit surprising that in scientific area of cryptography there are still some fields which are not yet defined and described, so to say as:

„**crypto incognito**“

In this connection "incognito" means: not yet investigated ranges of knowledge while being aware that their existence is already imaginable or presumed but not content-wise defined in their substance.



Up to now are missing: **„systematizing** of **bit series“** and definition of appropriate **„system alphabets“**.

3 New steps in coding procedures

Systematizing of bit series and creating meaningful system alphabets will lead to new solutions and techniques in digital cryptography.

3.1 System alphabet

The system alphabet is the most important component of computer techniques. It establishes foundation for visualization of bit series content. Without definition of an appropriate **alphabet** the computer will not be able to work, at all.

ASCII-character set in its respective charateristics is the fundamental system alphabet. For more voluminous character sets – especially foreign languages – **unicode** is used. Each computer output requires a concerning system alphabet (array). So, for images (pixel, graphical programs), sounds (digital/analogous converter, audio files), digital measuring instruments, bar coding and all further outputs of structured phenomenons in scale analysis and bit presentation.

3.2 Bit Conversion

Bit conversion is changing bit sequences from one bit system to another bit system. Number of bits and their order remain unchanged. No bit is added and no bit is removed. The structure will be changed, only. Decimal values of the new units serve for indexation to the assigned system alphabet. A bit conversion may transact for all bit systems from base 1 to base 14 (and higher).

4 Conversion of bitsystems

Up to now conversion of bit sequences are performed in procedure **„Coding Base 64“,** only [#3]. 8-bit sequences are converted to 6-bit segments which each gets a specific cipher character from a 64 elements alphabet (changing from bitsystem on base 8 to bitsystem on base 6). Decimal values of the 6-bit segments constitute index number of the connected system alphabet. The system alphabet is stated fix.

Bitsystem conversion presupposes that length of the bitsequence to be converted will be devisible both by number of the previous units and by number of the aimed units, as well. Otherwise single bits will be lost. For example changing from base 8 to base 7 operates as follows:

previous units (base 8):
01001110 01101111 01110100 01100001 01100010 01100101 01101110 = 7 units
aimed units (base 7):
0100111 0011011 1101110 1000110 0001011 0001001 1001010 1101110 = 8 units

Length of bit series with 56 bits is devisible by 7 and by 8 as well. As result of conversion ciphertext will be longer than plaintext by ratio **8/7**, that means: from one plaintext character results **1,143** ciphertext character. Ciphertext and plaintext have no equal length, any longer, with decisive consequences: the uniform **order system** gets lost.

4.1.1 Bit conversion from base 1 to base 8

In historical view conversion has already been developed in 1963 at beginning of digital performances when a 7-bit code combination was established for character recognition (ASCII-standard code). When system alphabet of 128 signs became to narrow an 8-bit code with an extended ASCII-character set of 256 elements (system alphabet) was introduced , which furthermore is used as a standard character representation.

By particular approach this handling already implements a coding procedure. It will scarcely be perceived as such but rather as basis of digital communication (standard code). Conversion from base 1 to base 8 in detail occurs as follows:

bit series base 1:
0100001001101100011101010110010100100000110100001100101011000010111 0110
bit series base 8:
01000010 01101100 01110101 01100101 00100000 01101000 01100101 01100001 0111
index: 66 108 117 101 32 104 101 97
system alphabet
B l u e □ h e a

system alphabet (ASCII): decimal hexa

alphabet$(32) = □ 20
alphabet$(66) = B 42
alphabet$(97) = a 61
alphabet$(101) = e 65
alphabet$(104) = h 68
alphabet$(108) = l 6C
alphabet$(117) = u 75

index (hexa) = system alphabet (hexa):
42 6C 75 65 20 68 65 61
In plaintext the output reads: „**Blue hea**ven**“**
in hexa: „**426C756520686561. . .**“

The underlying bit series in bitsystem on base 1 remain unchanged (no bit is added and no bit is removed). The example elucidates that the system alphabet only has to be changed while the unstructured bit series (number and order of sequences) on principle remain the same at each conversion.

4.1.2 Variation of system alphabet

With **generator** and start sequence „**Donky racing on the banks of San Bernadino**“ a permutated system alphabet is created, which leads to following variation of the base encryption:

```
  1  ®Žf E™□Ì"c$°'Î'#ã£□eú,²¥ áRgjÔ¦å§USTÏA#çNX,Vk#¨#JCIÈ÷õÉùGôK›.nÊ@      64
 65  0èý5¤Do tpx©€#‹:#Y[üÚº„ZWy6ï□]ÞÐÒhË½Åê%I\#(±ß´z^8×…q³□æ·rÀÁéˆ9s=  128
129  i#äÝ_O#ò1)`â##&¬##Íþša#‡»d«#□'¶ÑHµàøœb¯/Œ Æ;Q2Ó###vP—¡4ìÜ{ž¼¢Ÿ 192
193  #□!LM¾<ë3BÛw##û#*˜|#mÖÕ¿ªƒ Ø#7¹‰†ö?íF~>uŠ#¸#î}+Ãñ“Â#”ð•ÙÇ#Äó-–  256
```

new system alphabet:

```
alphabet$(32)  =  å
alphabet$(66)  =  è
alphabet$(97)  =  Ò
alphabet$(101) =  Å
alphabet$(104) =  I
alphabet$(108) =  ±
alphabet$(117) =  ³
```

Plaintext „Blue heaven“ leads to cipertext „**è±³åIÅÒ**“ **. . . . .**

This example demonstates the simpliest mode of encryption. The system alphabet only has to be changed in text blocks of e.g. 16 bytes (new round). For testing purposes you may download the program: „System08.exe“ and test it by yourself. The source code is included as well.

4.2 Bit conversion from base 8 to base 13

The conversion is performed by the program: „**System13.exe**“. Despite of converting from bitsystem on base 8 to bitsystem on base 13 number of bits and their order remain unchained. No bit ist removed and no bit is added. Only distances of the bits change. 8-bit sequences modify to 13-bit sequences. Dezimal values of the new units become index values for the signs in system alphabet on base 13. For this 2^13 signs = 8192 signs are necessary. Because of this size there are no single characters available the digits of **number system** on **base 128** – that are 16384 digits – are used as double signs. The system alphabet covers a permutated range beginning with Kappa+1, that is from digit 5193 up to 13385 = 8192 double signs.

bit series base 8:

```
01000010 01101100 01110101 01100101 00100000 01101000 01100101 01100001 011
index:   66       108      117      101      32       104      101      97
system alphabet
         B        l        u        e        □        h        e        a
```

bit series base 13:

```
0100001001101 1000111010101 1001010010000 0011010000110 0101011000010 11
index:        2125          4565          4752          1670          2754
index+1:      2126          4566          4753          1671          2755
base 13       vN            ëV            ì†            rï            &C
```

system alphabet (base 13):

```
Alphabet$(2126) =  vN
Alphabet$(4566) =  ëV
Alphabet$(4753) =  ì†
Alphabet$(1671) =  rï
Alphabet$(2755) =  &C
```

In bitsystem on base 13 „Blue heaven“ is coded as follows: **vNëVì†rï&C . . .**

4.3 Bit conversion from base 8 to base 7

Bit conversion from base 8 to base 7 is the main application for encryption.
In following example a conversion is demonstrated by program: **Crypto07.exe**.

bit series base 8:
01000010 01101100 01110101 01100101 00100000 01101000 01100101 01100001 011
index: 66 108 117 101 32 104 101 97
system alphabet
B l u e □ h e a
bit series base 7:
0100001 0011011 0001110 1010110 0101001 0000001 1010000 1100101 0110000 101
index: 33 27 14 86 41 1 80 101 48
(+1) 34 28 15 87 42 2 81 102 49
cipher alphabet base 7:
Ç 3 ù ï Ã t T ¬ ³

Plaintext „Blue heaven“ reads in bitsystem on base 7 as follows: **Ç3ùïÃtT¬³**

System alphabet on base 7 with 128 characters is created by the generator and the start sequence: „**Blue heaven over Minnesota all the day long**“. Index values are added by (+1), because the program does not recognize index „0“.

cipher alphabet (Base 7)

```
  1  P t   ¨ 1 ‹ ø Ÿ ƒ ¯ Ä [ i Â ù ¢ ¼ é % ú □ å ‘ | ö û „ 3 } ’ ¤ '     32
 33  © Ç B “ W ; Å 7 e Ã Æ , – 9 ì L ³ ü ý ã ¥ R ¶ ª > & 8 þ 6 Ë µ æ   64
 65  — m ž ( ½ ˜ f « ~ ê à u : ë Y □ T † c È î º ï g i £ * w € · <       96
 97  ¥ # - á l ¬ Î ¾ U X • ‡ D H _ Í ð … ¿ À É Ó ! + Ï ˆ Z $ E Ô Á Ì 128
```

Cipher aphabet (hex)

```
  1   50 74 A0 A8 31 8B F8 9F 83 AF C4 5B 69 8F F9 A2   16
 17   BC E9 25 FA 7F E5 91 7C F6 FB 84 33 7D 92 A4 27   32
 33   A9 C7 42 93 57 3B C5 37 65 C3 C6 82 96 39 EC 4C   48
 49   B3 FC FD E3 9D 52 B6 AA 3E 26 38 FE 36 CB B5 E6   64
 65   97 6D 9E 28 BD 98 66 AB 7E EA E0 75 3A EB 59 7F   80
 81   54 86 63 C8 EE BA EF 67 A1 A3 2A 77 80 B7 3C B8   96
 97   A5 23 2D E1 6C AC CE BE 55 58 95 87 44 48 5F CD  112
113   F0 85 BF C0 C9 D3 21 2B CF 88 5A 24 45 D4 C1 CC  128
```

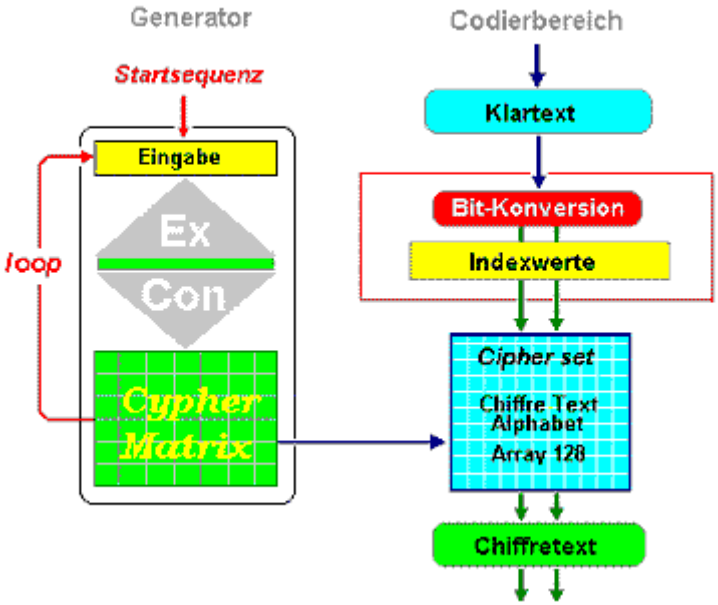
4.4 Consequences of bit conversion

As fundamental result of bit conversion turns out: Ciphertext becomes longer in lengths ratio: plaintext length in original bitsystem to ciphertext length in aimed bitsystem. Demanding plaintext and ciphertext should have the same length is principially not fulfilled.

5 The CypherMatrix procedure

CypherMatrix procedure performs encryption in a very simple way:

> **Generator** creates the necessary **system alphabet** and
> in **coding area** cipher text is written by **bit conversion**.



Both sectors are combined together, but can be used separate, as well. The essential task of the generator is to serve with all required parameters necessary for encryption. Actual writing of ciphertext takes place in the coding area.

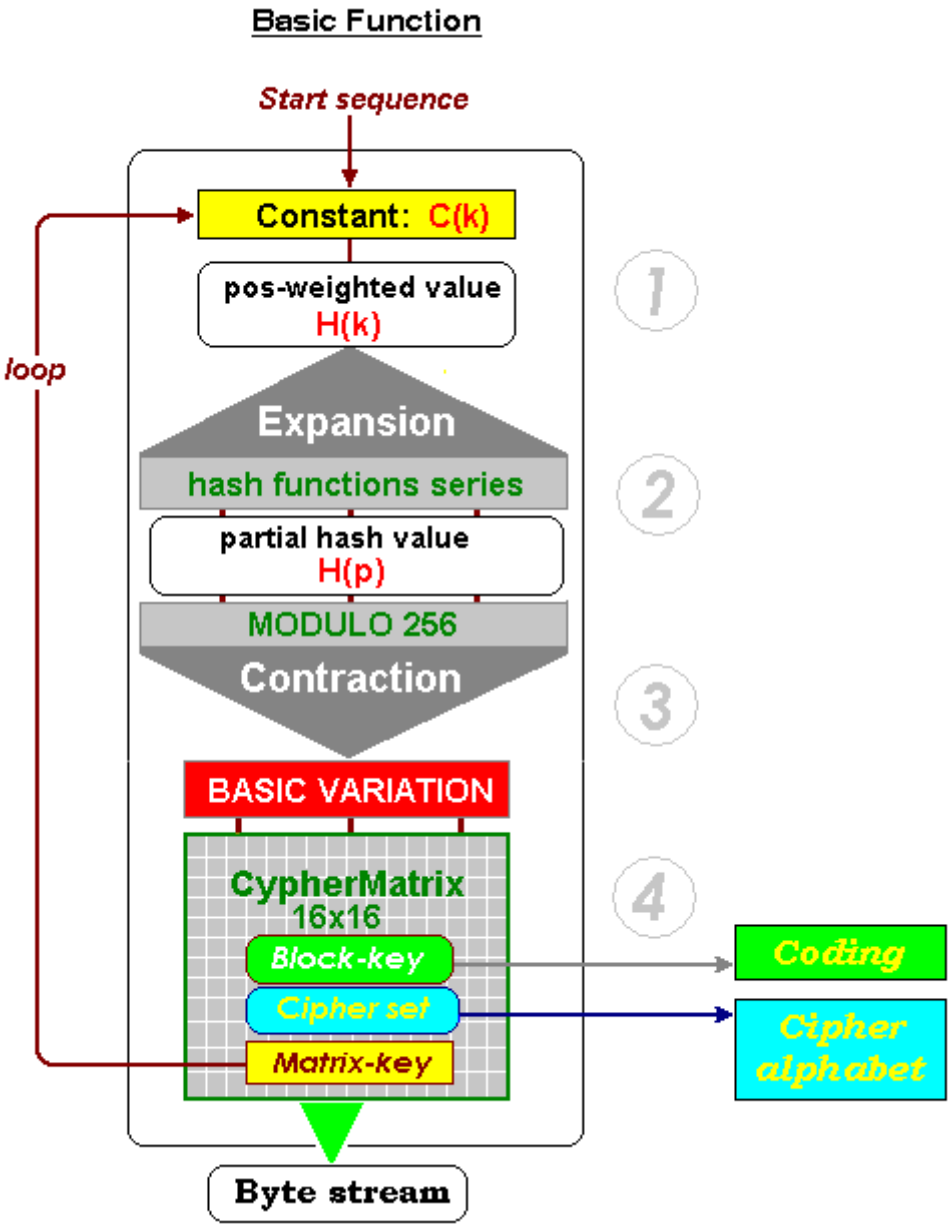
5.1 The Generator

The procedure ist arranged as block cipher. It is symmetric, because sender and recipient have to insert the same start sequence in order to initialize the generator. The procedure is dynamic, because the generator creates new control parameters for each plaintext block (e.g. 63 bytes) in each round. Any start sequence (passphrase) with at least 36 and optimal 42 characters drives and controls the whole process. Some examples:

Horse racing on the banks of San Sebastian	[42 bytes]
Koala bears are diving at Murray's Mouth	[40 bytes]
7 kangaroos jumping along the Times Square	[42 bytes]
Blue flamingos flying to Northern Sutherland	[44 bytes]

Start sequence should be easy to remember and possibly somewhat catchy and of funny words. The chosen phrase cannot be guessed, can easy be kept in mind and may not be written down, anywhere. Because of their length lexical attacks and iterative searching should be impossible. An attacker even isn't able to analyse parts of the start sequence neither apart nor successive because the passphrase could be found in total only, if at all.

Each start sequence at sender and recipient, as well, generates identical control parameters and course of the procedure.



For each round generator delivers all control parameters necessary for cipher processing:

1. Cipher alphabet (system alphabet) for the actual round,
2. block key for XOR-concatenation and
3. matrix key as start sequence for the next round.

At end of each round a **CypherMatrix** with 16x16 elements is created, which serves for all control parameters to achieve the encryption. The matrix key (42 signs) is to be led back to the beginning (**loop**) in order to initialize the next round, till an end mark is designated. Due to probability laws a repetition of identical Data will occur first in **256!** (faculty) = **8E+506** cases .

5.2 Inserting Start Sequence

The following start sequence (input) is chosen as an example:

Checkpoint Charly at Madison Square Garden [42 bytes]

43 68 65 63 6B 70 6F 69 6E 74 20 43 68 61 72 6C 79 20 61 74 20
4D 61 64 69 73 6F 6E 20 53 71 75 61 72 65 20 47 61 72 64 65 6E

The goal is to find an evident determination base for analysing the start sequence.
Input **m** is a series of definite bytes **a(i)** with lengths **n.** In order to analyse the series as state of facts the single characters have to be systematized (scaled). For this each byte **a(i)** gets an index and all bytes will be appropriate associated with each other (addition).

$$m = a_1 + a_2 + a_3 + \ldots a_i + \ldots a_n$$

(single value of „a(i)“ has to increase by (+1) because otherwise ASCII-zero (0) would not be considered)

$$m = \sum_{i=1}^{n} (a_i + 1)$$

$$m = 3991$$

In order to differentiate single bytes **a(i)** inside the series additional criterions have to be added, because otherwise no definite results will resume.

5.3 Extending to Position Weighting

With reference to **René Descartes** (1596-1650) we know that every object (fact) – which is scalable in its dimensions – can be exactly determined by coordinates for **subject, location** and **time** (cartesian coordinate system). We set:

object: (**m**) digital information of length (**n**)
subject: (a_i) elements of information, signs, bytes
location: (p_i) position of byte **a**(**i**) inside the information
time: (t_i) point of time of byte **a**(**i**) inside the information

In order to distinguish single characters each byte **a**(**i**) is multiplied with its location **p**(**i**). Time will be relevant only if there exists a functionally connection between a single byte and clock frequency. Normaly this connction is constant and we may set: **t = 1**. In order to get an exact determination for series **m** we associate **subject**, **location** and **time** by multiplying its dimension values and summarize all results to a destination value **H**(**k**):

$$H(k) = \sum_{i=1}^{n} (a_i + 1) * p_i * t_i \qquad t_i = 1$$

$$H(k) = 85589$$

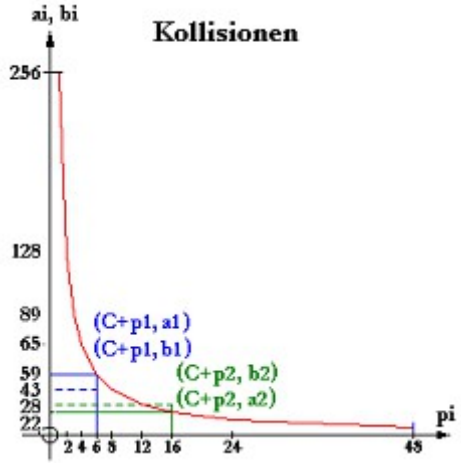
With position weighting the single bytes differ but collisions in consequence of exchanging bytes inside the series are not excluded, yet.

5.4 Excluding Collisions

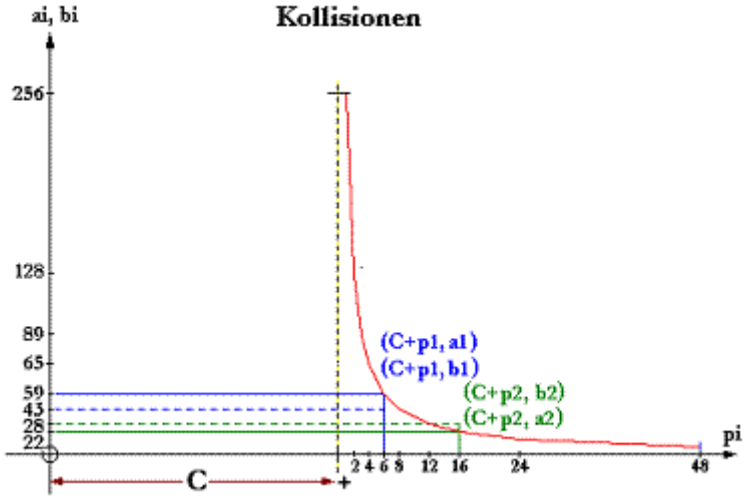
A collision occurs at following conditions:

$$\text{collision:} \quad H(k)\, a_i = H(k)\, b_i$$
$$(a_1+1) * p_1 + (a_2+1) * p_2 = (b_1+1) * p_1 + (b_2+1) * p_2$$

At position p_1 the sign a_1 is exchanged with the sign b_1 and at position p_2 the sign a_2 with b_2. The following graph illustrates the connections between single signs a_i and p_i in multiplication formula $(a_i + 1) * p_i$.



To avoid collisions the position weighting is shifted by a distance **C** to an area above length **n,** that means: p_i will be extended by a constant distance **C.**



$$(a_1+1) * (C+p_1) + (a_2+1) * (C+p_2) = (b_1+1) * (C+p_1) + (b_2+1) * (C+p_2)$$

After transforming we get the changing quotient **Q**:

$$Q = \frac{(C + p_1)}{(C + p_2)} = \frac{(b_2 - a_2)}{(a_1 - b_1)}$$

For the „changing quotient“ – here denoted with **Q –** three cases are relevant:

$$Q > 1$$
$$Q = 1$$
$$Q < 1$$

If **Q = 1** then (**C**+ p_1) and (**C**+ p_2) must be equal, as well. Because exchanging of characters occured at the same position **p** here is a collision excluded. If **Q > 1** or **Q < 1** then ($b_2 - a_2$) and ($a_1 - b_1$) are different, as well. Because a_1, a_2, b_1 and b_2 being integer values their differences will be integer values, as well.

Positions p_i in the start sequence with **N** bytes (length n) cover a range from **1** (*minimum*) to **N** (*maximum*). Such the changing quotient of above formula comprises the following range:

$$\frac{C + N}{C + 1} \; \} \; Q \; \{ \; \frac{N}{N - 1}$$

Further development is demonstrated in article:

"Determinants leading to collisionfree"

The result leads to following formula: $C = N * (N - 2)$

Factor **C** depends on length **N** of the start sequence only. **C** includes the properties being equal for all start sequences with same lengths and deviding values of position weighting in collisionfree and collision burdened segments. Because of this the factor **C** gets the name: dividing constant **C**(**k**).

In order to individualize the function an additional code is introduced – a chosen number between 1 and 99. We set code = 1:

$$C(k) = n * (n - 2) + code$$
$$C(k) = 1681$$

By including the dividing constant **C**(**k**) the destination value H_k results as follows:

$$H_k = \sum_{i=1}^{n} (a_i + 1) * (p_i + C_k + round)$$
$$H_k = 6798451$$

The result **H**(**k**) avoids collisions but is still too narrow to establish anvulnerable destination values. Probably it could serve as **MAC** for messages.

5.5 Extending to Hash Function Series

To extend the destination base an **expansion function** is introduced which widens the sequence to an extensive series in a superior number system. The number system of expansion may be chosen betwenn 64 and 96. Here the number system on **base 77** is fixed. The function calculates for each value of the inserted sequence its decimal value s_i which is changed to d_i - digits in number system on base 77. Simultaneous the function calculates the sum of all single results s_i as an additional destination value **H(p)** for creating several control parameters and accumulates the results d_i serial as hash function series (HF).

$$s_i = (a_i + 1) * p_i * H_k + p_i + code + round$$
$$s_i \rightarrow d_i \quad (base\ 77)$$

$$HF = d_1 + d_2 + d_3 + \ldots + d_i + \ldots d_m$$
(m = number of digits in system on base 77)

$$H_p = \sum_{i=1}^{n} S_i$$
$$H_p = 581872623626$$

The chosen number system on base 77 comprises the following digits:

0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz&#@àáâãäåæçèéêë
(determinated by the author, not standardized)

By generating hash function series the sequence "**Charly**" at position 12 of the inserted passphrase results to the following calculations:

char	pi		Hk	(ai+1)*pi*Hk	Si		Basis 77
	(ai+1)	(ai+1)*pi			pi+code+r		
C	68	12 816	6798451	5547536016	14	5547536030	23&YPa
h	105	13 1365	6798451	9279885615	15	9279885630	3WêáxF
a	98	14 1372	6798451	9327474772	16	9327474788	3YQ8LG
r	115	15 1725	6798451	11727327975	17	11727327992	4Pkzeç
l	109	16 1744	6798451	11856498544	18	11856498562	4TLuw&
y	122	17 2074	6798451	14099987374	19	14099987393	5G7æGU
					Summe:	581872623626	2yégr4#

The hash function series comprises 248 digits in number system on base 77:

DBlxäelGF4xDyëæ10Rjdo1RXUnp1s9Wbf1élåu929ëéæs2dFjB02çL2HuåFkrk23&YPa3W
êáxF3YQ8LG4Pkzeç4TLuw&5G7æGU1bâfVr4q7æuL5âg4ZX1v1zBN4Náxuã5oåU7P66yçfb
6ocW2n7iLroz7j##&T7&5sãU2V6B8K6PReQW8âZbAá9bKGçC89Y30Z9#DqFw8éWiOb2êw5
#R6rFä3z9RFáZeBKTLTUABOhä9AcyCOLBsl5Pi

The variables are digits (not characters). There is no way back to the start sequence (first **one-way-function**). Simultaneous the function calculates the following destination Data:

dividing-constant C(k):	1681
position weighted value (H_k):	6798451
destination value (H_p):	581872623626
total value (H_p+H_k):	581879422077

Control parameters derived from destination Data show as follows:

Variante	(H_k MOD 11) +1	=	1	begin of contraction
Alpha	((H_k + H_p) MOD 255) +1	=	148	offset cipher alphabet
Beta	(H_k MOD 169) +1	=	89	offset block key
Gamma	((H_p + code) MOD 196) +1	=	128	offset matrix key
Delta	((H_k + H_p) MOD 155) +code	=	93	dynamic bit series
Theta	(H_k MOD 32) +1	=	20	offset back factor
Omega	(H_k MOD 95)+1	=	62	parameter cipher alphabet
Kappa	(H_k MOD 16384) +1	=	8203	begin of cipher alphabet

The above control parameters serve for solution of several cryptographical tasks.

5.6 Contraction to BASIC VARIATION

In order to reduce the variables back to decimal values a contract function is introduced. The digits of hash function series are assumed to be digits in number system on **base 78** (expansion base +1). Each three digits of the function series are reconverted in serial manner by **MODULO 256** to decimal numbers **0** to **255** (without repetition). The parameter **Theta** is deducted. Results are stored in BASIC VARIATION, an array of 16x16 elements. A backwards searching of foregoing Data is not possible (second **one-way-function**).

The first five reconversions beginning at variante = **1** shows as follows:

3 digits base 78	decimal	Modulo 256	- Theta	element
DBl	79997	125	20	105
Blx	70649	249	20	229
lxä	290619	59	20	39
xäe	364378	90	20	70
äel	422963	51	20	31

BASIC VARIATION (256 elements)

```
105 229 039 070 031 238 215 194 219 003 074 116 075 191 150 203
083 117 118 005 157 011 196 133 006 251 124 045 034 181 220 192
232 107 044 119 158 140 120 126 066 239 108 190 007 101 135 041
186 090 208 121 122 173 218 071 047 068 161 123 134 195 156 062
217 182 012 136 067 137 175 125 174 127 176 233 089 130 228 128
200 081 226 024 129 097 231 193 076 183 061 250 167 149 252 072
082 087 057 211 213 247 210 236 221 037 035 109 055 201 084 008
111 199 202 184 015 027 085 017 094 216 064 223 166 110 138 180
069 168 177 032 237 234 148 020 245 178 159 073 230 235 198 058
```

```
106 197 204 056 088 249 209 240 063 212 152 042 065 131 049 033
132 026 036 160 241 139 242 222 043 046 077 114 227 153 162 038
154 009 010 142 185 091 048 104 100 040 188 151 243 050 224 092
163 187 016 086 205 141 028 014 189 143 244 018 155 051 144 013
025 019 112 164 206 093 113 246 225 145 029 115 169 078 248 253
146 052 165 254 079 255 021 214 171 000 147 002 102 095 170 172
207 179 001 004 022 023 030 053 054 059 060 080 096 098 103 099
```

5.7 Calculation of „CypherMatrix“

Calculating CypherMatrix the elements of BASIC VARIATION are used directly in their distribution 16x16 to achieve the destination base. The elements values are related to index values of bytes (**0** to **255:** comparable with ASCII-set). To distribute the characters in the CypherMatrix (16x16) especially variant **D** is used:

5.8 Dynamic generating of indexes: (variant D)

All index values (16x16) are new generated from the elements of BASIC VARIATION in a special array **IndexFolge(2,16)** (another application of CypherMatric function).
In parts of source code:

```
SHARED IndexFolge(2,16), Delta, Omega

FOR B = 1 TO 2
   IF B=1 THEN X = Delta
   IF B=2 THEN X = Omega
   FOR C = 1 TO 16
      INCR X
      IF X > 256 THEN X = 1
      A = VARIATION(X) MOD 16                elements of BASIC VARIATION
      IndexFolge(B,C) = A
      IF C>1 THEN
         L = 0
         DO
           INCR L
           IF IndexFolge(B,L) = A THEN
              INCR A
              A = (A MOD 16)
              IndexFolge(B,C) = A
              L = 0
           END IF
         LOOP UNTIL L = C-1
      END IF
      IndexFolge(B,C)  =  IndexFolge(B,C) + 1    array IndexFolge (2,16)
   NEXT C
NEXT B

N = Alpha
FOR I = 1 TO 16
   FOR J = 1 TO 16
      X = IndexFolge(1,I)
      Y = IndexFolge(2,J)
      Matrix$(X,Y) = VARIATION$(N)            generation of CypherMatrix
      INCR N
      IF N > 256 THEN N = 1
   NEXT J
NEXT I
```

Final CypherMatrix (Variation: D)

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

5.9 Control Parameter

As cipher alphabet (system alphabet of bitsystem on base 7) 128 characters are taken from the actual CypherMatrix at position **148.** Certain signs (hex: 00 bis 20, 22, 2C, B0, B1, B2, D5, DB, DC, DD, DE, DF and others) are ignored because they do still their original task (e.g. **1A** = ASCII-26) and disturb the proper realization of the program.

Cipher alphabet (base 7)

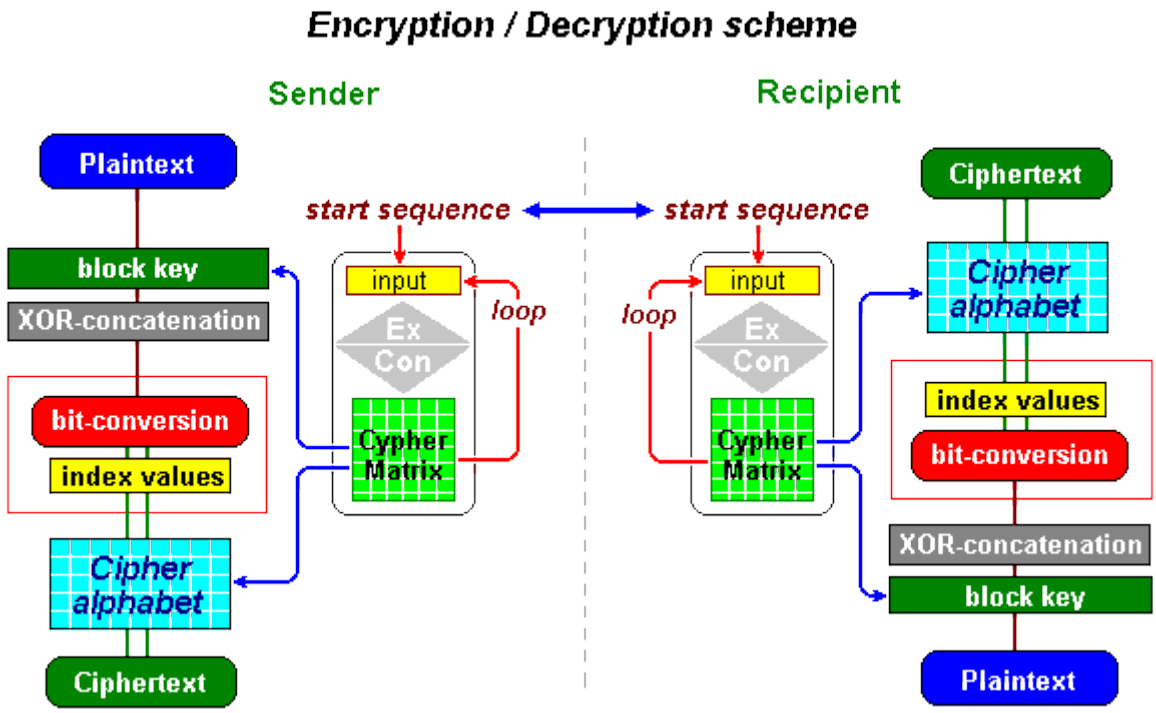
1	õ	Ò	a	C	y	Đ	k	S	c	ª	N	›	©	¼	.	?	*16*
17	U	÷	□	ˆ	Z	è	Ë	g	_	b	f	s	ô	(	+	ð	*32*
33	″	â	¶	°	À	–	¿	`	□	d	Ñ	ê	Ó	9	Q	Ù	*48*
49	)	>	‡	µ	K	P	‟	'	½	h	ò	ù	í	¸	Ê	W	*64*
65	È	R	€	œ	e	t	<	á	0	‹	X	Ç	¨	o	H	ä	*80*
81	Ã	-	J	;	«	ö	[	ñ	8	Ì	$	Å	E	ü	,	†	*96*
97	¾	|	6	Ö	q	□	¹	□	Ž	j	´	T	•	Y	{	l	*112*
113	û	5	]	Í	Î	V	„	:	Š	É	§	é	¡	ï	Â	□	*128*

Cipher alphabet (hex)

1	F5	D2	61	43	79	D0	6B	53	63	AA	4E	9B	A9	BC	2E	3F	*16*
17	55	F7	81	88	5A	E8	CB	67	5F	62	66	73	F4	28	2B	F0	*32*
33	94	E2	B6	BA	C0	96	BF	60	8F	64	D1	EA	D3	39	51	D9	*48*
49	29	3E	87	B5	4B	50	93	91	BD	68	F2	F9	ED	B8	CA	57	*64*
65	C8	52	80	9C	65	74	3C	E1	30	8B	58	C7	A8	6F	48	E4	*80*
81	C3	2D	4A	3B	AB	F6	5B	F1	38	CC	24	C5	45	FC	82	86	*96*
97	BE	7C	36	D6	71	8D	B9	A0	8E	6A	B4	54	95	59	7B	6C	*112*
113	FB	35	5D	CD	CE	56	84	3A	8A	C9	A7	E9	A1	EF	C2	FF	*128*

6 Coding area

Encryption – writing and reading of secret informations – is exclusively performed in the coding area. By inserting an identical start sequence at sender and recipient es well an equal course and identic control parameters are generated. The following schema shows the connections:



Ciphering is performed by following alternatives:

1. **Basic Coding:** bit conversion without further operations or
2. **Compound Coding:** bit conversion with additional operations ,

   a) with XOR-concatenation (foregoing or succeding),
   b) connected with further operations (see: "Combinierte Operationen")

6.1 Programs overview

According to foregoing prinziples the Author has developed a range of programs which are listened in the following survey:

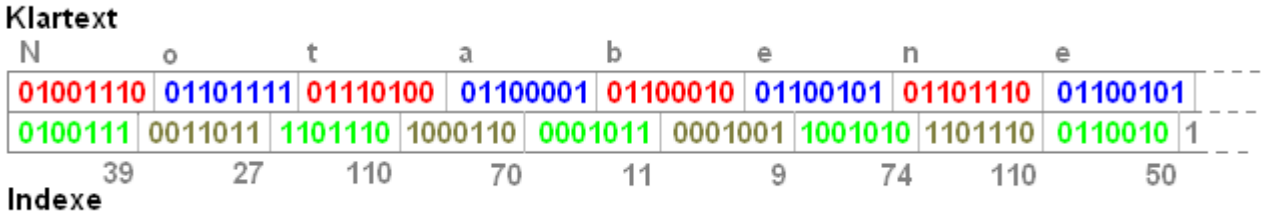
System Basis	System Alphabet	Basis-Coding (ohne XOR-Funktion) einfache Matrix	Verbund-Coding (mit XOR-Funktion) einfache Matrix	Längen-verhältnis
1	2	Crypto01	MonoCode	1:8
2	4	Crypto02	ZweiCode	1:4
3	8	Crypto03	DreiCode	1:2,66
4	16	Crypto04	VierCode	1:2
5	32	Crypto05	QuinCode	1:1,6
6	64	Crypto06	CM64Code	1:1,33
7	128	Crypto07	DataCode	1:1,143
			DynaCryp	1:1,143
			CodeData[1)]	1:1,143
			QuadCode[2)]	1:1,143
8	256	Crypto08	PlanCode	1:1
		CMCode8D	MyCode08	1:1
		System08		1:1
9	512	Crypto09	NeunCode	1:1,79
		System09	MyCode09	1:1,79
10	1024	Crypto10	ZehnCode	1:1,6
		System10	MyCode10	1:1,6
11	2048	Crypt11A	ElvaCode	1:1,46
		Crypt11B	MyCode11	1:1,46
		System11		
12	4096	Crypto12	MegaCodA	1:1,33
		System12	MegaCodB	1:1,33
			MyCode12	1:1,33
13	8192	System 13	MyCode13	1:1,23
14	16384	System14	MyCode14	1:1,143

[1)] Programm mit drei Operationen (XOR – bit conversion - exchange),
[2)] Programm mit vier Operationen (dyn24 – XOR – bit conversion – exchange).

Programs may single or in groups with or without sourcecode per e-mail requested at the author and tested or further developed within the scope of the **CMLizenz.** All programs are Dos-programs and will run under WindowsXP only. They have to be converted to **C#,** as already done under direction of Prof. **Bernhard Esslinger** (University of Singen) and his CrypTool-team (especially: **Michael Schäfer)** with the programs **DataCode** and **DynaCode** [#4]. All further programs have still to be converted.

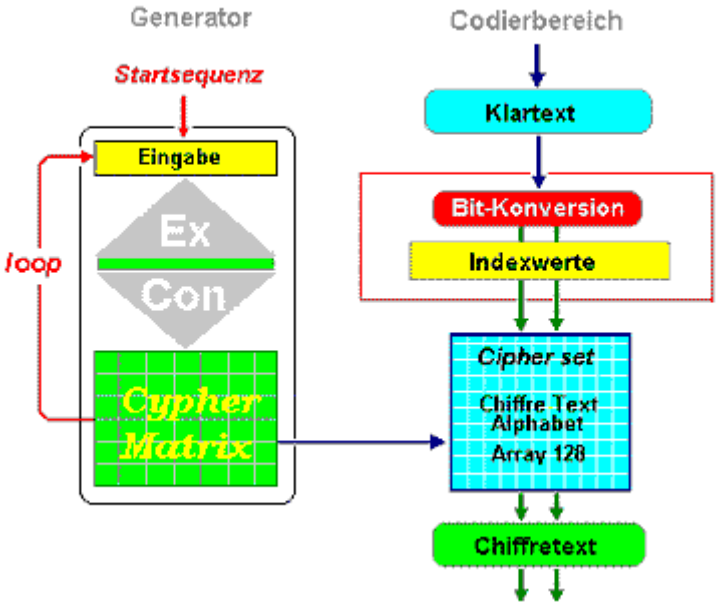
6.2 Basic Coding

In „basic coding“ modus the bit conversion is performed directly from the forgoing bitsystem to the aimed bitsystem. As example: from **base 8** to **base 7**:



Changing from plaintext to ciphertext are processed by two functions:

1. Bit conversion
   8-bit plaintext values → 7 bit index values (0 ...127)
2. Destination of ciphertext
   7-bit Index values → cipher alphabet (0...127) → ciphertext.



As an example of **Basic Coding** a text from **Mark Twain** is encrypted with program „**System12.exe**“ and start sequence: „**Donkey racing on the banks of San Bernadino**“ (file: language.txt/ content: 612 bytes).

> *In German, all the Nouns begin with a capital letter. Now that is a good idea; and a good idea, in this language, is necessarily conspicuous from its lonesomeness. I consider this capitalizing of nouns a good idea, because by reason of it you almost able to tell a noun the minute you see it. You fall into error occasionally, because you mistake the name of a person for the name of a thing. and waste a good deal of time trying to dig a meaning out of it. German names almost always do mean something, and this helps to deceive the student.*
>
> The Awful German Language, Mark Twain, A Tramp Abroad, 1880

As first plaintext block 36 bytes are inserted:

In German, all the Nouns begin with□

49 6E 20 47 65 72 6D 61 6E 2C 20 61 6C 6C 20 74 68 65
20 4E 6F 75 6E 73 20 62 65 67 69 6E 20 77 69 74 68 20

Plaintest in bitsystem on **base 8** (36x8=288) is converted into segmemts of aimed bitsystem on **base 12** (288:12=24).

base 8: I n G e r m a
01001001 01101110 00100000 01000111 01100101 01110010 01101101 01100001 . . .
base 12:
010010010110 111000100000 010001110110 010101110010 011011010110 0001 . . .

Index:	1174	3616	1142	1394	1750
(+1)	1175	3617	1143	1395	1751
ciphertext:	îi	Žs	îC	{8	}˜

system alphabet base 12:

Alphabet$(1143) = îC
Alphabet$(1175) = îi
Alphabet$(1395) = {8
Alphabet$(1751) = }˜
Alphabet$(3617) = Žs

Ciphertext reads as follows:

îiŽsîC{8}˜è4ê„å£}ˆŠs€ˆ,§éQÅ5€ÿÅ9éSé§}C…4éT€¬€ˆ,sjLX'jQX˜lèZ›b9v"
léf"lbzZg•#¦bAf—jQfZkQdZjJxDj¥#ž,êéž,î€–{xé–ƒ›|f,Réœƒ¬"™íFˆ™,‹ër
íFˆ£íG| ƒWïfƒîë£c¦a c¥alU9i¢U9s"cva¢evS¡dPo©U9Wžd–WŸdPW¤d¦a¢R}k•
X'oiNäcpX KhXrmaX¡oiWLmaX¢P™Nâb‹V¡ojX¡cZWMN‹Y5ae•'ˆN"êˆT•¦ŠW"êœT

```
EE 69 8E 73 EE 43 7B 38 7D 98 E8 34 EA 84 E5 A3 7D 88 8A 73 80
88 82 A7 E9 51 8F 35 80 98 8F 39 E9 53 E9 A7 7D 43 85 34 E9 54
80 AC 80 88 82 73 6A 4C 58 92 6A 51 58 98 6C E8 5A 9B 62 39 76
94 0D 0A 6C E9 66 94 6C 62 7A 5A 67 95 23 A6 62 41 66 97 6A 51
66 5A 6B 51 64 5A 6A 4A 78 44 6A A5 23 9E 82 EA E9 9E 82 EE 80
96 7B 78 E9 96 83 9B 7C 66 82 52 E9 9C 83 AC 94 99 ED 46 88 99
82 8B EB 72 ...
```

The encrypted file **Language.ctx** comprises 816 characters. Through bit conversion 12 characters in bitsystem on base 8 comprise 96 bits which will be assigned to 8 double signs in bitsystem on base 12. Insofar there is a ratio of 1:1,333.

The system alphabet on base 12 needs 4096 signs, which are not available by single characters. So, they are generated as double signs by digits of number system on base 128 – that are 16384 digits.

Partial sourcecode:

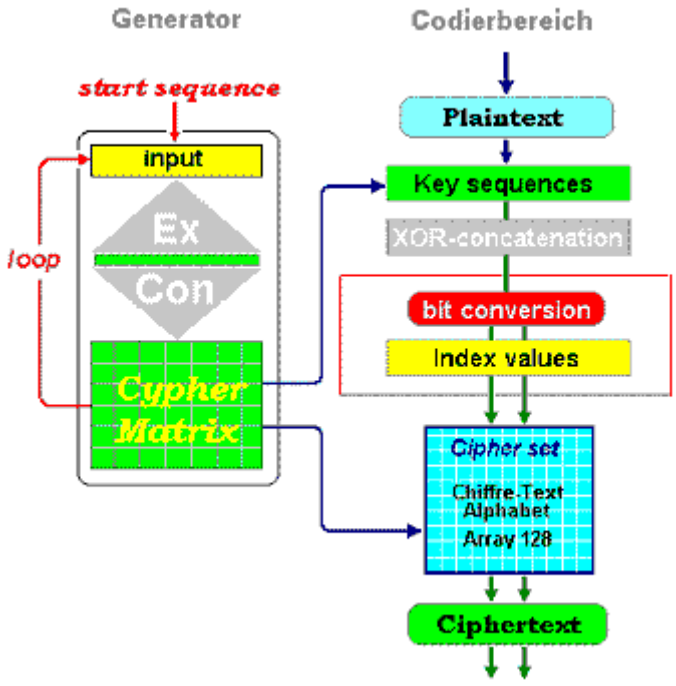
```
SUB Alphabet
    SHARED Alphabet$(),  Kappa
    Kappa = 7979, 1202, 142, 8330, 1428  (anew generated in each round)

    FOR C=1 TO 4096
      X## = C + Kappa
      CALL DezNachSystem (128, X##, Zeichen$)
      Digit$ = „00“+Zeichen$
      Digit$ = RIGHT$(Digit$,2)
      Alphabet$(C) = Digit$
   NEXT C
END SUB
```

6.3 „Compound Coding“

By „Compound coding“ several operations are combined in succession with bit conversion and further operations (e.g. XOR concatenation, dyn24, exchange).

XOR concatenation is installed before bit conversion. Encryption then works in three functions:

1. partial dynamic „one-time-pad“
   plaintextblock → block key → 8-bit XOR concatenation
2. bit conversion
   8-bit XOR concatenation → 7 bit index values (0 ...127)
3. destination of ciphertext
   7-bit index values → cipher alphabet (0...127) → ciphertext.



Moduls are passing the procedure in succession.

```
. . . . . .
CALL  XORGenerator (InputData$,TransData$)          for „XOR“
CALL BitConversion (TransData$,DataTrans$)          for „bit conversion“
CALL Substitution (DataTrans$,ReportData$)          for „dyn24“
CALL ByteWechsel (ReportData$,OutputData$)          for „exchange“
. . . . . .
```

6.3.1 Encryption from base 8 to base 7

The following example demonstrates encryption steps of file **Museum.txt** with program **Datacode.exe** and the start sequence „**Yellowstoneparkbridge is under construction**“ (43 bytes)**.** Plaintext file contains 596 bytes.

> *The science of bird study in all its aspects is known as ornithology. A major task of the ornithology is to describe and name the birds of the world and to arrange them into species, genera, families, and higher categories of kinship. About 8700 species are known. There is still much to learn concerning the evolutionary relationships of the families and orders of birds. New methods in systematics, as applied to populations of closely related birds, are constantly leading to a better understanding of the process of evolution.*
>
> The American Museum of Natural History, New York 1972

In each round a plaintext block of 42 bytes (42x8 = 336 bits) is XOR concatenated with a block key of equal length. As first plaintext block the program reads:

The science of bird study in all its aspec (42 bytes)

54 68 65 20 73 63 69 65 6E 63 65 20 6F 66 20 62 69 72 64 20 73
74 75 64 79 20 69 6E 20 61 6C 6C 20 69 74 73 20 61 73 70 65 63

The block key extracted at position 115 from current CypherMatrix comprises:

†9ÉóB£WkÒ¥®”æ@#Ðë‡Í`Î‹#ÛIÏO5#ÑYd'ZVÇXTiû\c

86 39 C9 F3 42 A3 57 6B D2 A5 AE 94 E6 40 1D D0 EB 87 CD 60 CE
8B 1C DB 49 CF 4F 35 12 D1 59 64 27 5A 56 C7 58 54 69 FB 5C 63

plaintext base 8:
01010100 01101000 01100101 00100000 01110011 01100011 01101001 01100101 . . .
block key base 8:
10000110 00111001 11001001 11110011 01000010 10100011 01010111 01101011 . . .
XOR contatenation base 8:
11010010 01010001 10101100 11010011 00110001 11000000 00111110 00001110 . . .

As result arises a partial dynamic „**one-time-pad**“. Plaintext and block key are of equal length and the key will not be repeated. Each round gets another key from the respective CypherMatrix.

6.3.2 Bit conversion

The result of XOR-concatenation in bitsystem on base 8 (42x8 = 336 bits) is converted to characters of bitsystem on base 7 (336 bits = 48x7) . Decimal values of the changed signs (base 7) are index values to positions of cipher characters in the system alphabet on base 7 (cipher alphabet of 128 elements). Indexes have to be added with (+1) because the cipher alphabet array does not recognize the value „**0**“

XOR base 8:
11010010 01010001 10101100 11010011 00110001 11000000 00111110 00001110 . . .
conversion base 7:
1101001 0010100 0110101 1001101 0011001 1000111 0000000 0111110 0000111 0 . . .

index:	105	20	53	77	25	71	0	62	7
(+1)	106	21	54	78	26	72	1	63	8
cipher:	[	™	N	“	M	‘	i	¦	e

System alphabet on base 7

1	**i** û \ c ^ ³ o **e** Ž % ¨ Ú ˜ f Ë © v t ð ™ E) × • **M** Å Ä á „ » è	32
33	s Ì b í , n Á ÷ a Ù ¸ A Å C ‰ À (H · ¥ = **N** I J ¶] ~ m Ÿ Â **¦** Q	64
65	Ø / ¼ « Œ º ¾ ‘ ô D Ö ½ ª “ ¡ q à ƒ 7 Ã Ì Ó ú ; ý . w ù R ø Ô	96
97	x ï { þ ˆ y z □ â **[** | : } 2 S # ! < □ 4 $ 8 F □ G È ¬ ã u ? î ž	128

System alphabet (hex)

1	69	FB	5C	63	5E	B3	6F	65	8E	25	A0	A8	DA	98	66	CB	16
17	A9	76	74	F0	99	45	29	D7	95	4D	C5	C4	E1	84	BB	E8	32
33	73	8D	62	ED	82	6E	C1	F7	61	D9	B8	41	8F	43	89	C0	48
49	28	48	B7	9D	3D	4E	6C	4A	B6	5D	7E	6D	9F	C2	A6	51	64
65	D8	2F	BC	AB	8C	BA	BE	91	F4	44	D6	BD	AA	93	A1	71	80
81	E0	83	37	C3	CC	AD	D3	FA	3B	FD	2E	77	F9	52	F8	D4	96
97	78	EF	7B	FE	88	79	7A	7F	E2	5B	7C	3A	7D	32	53	23	112
113	21	3C	7F	34	24	38	46	90	47	C8	AC	E3	75	3F	EE	9E	128

Encryption of the plaintext file **Museum.txt** results in ciphertext file **Museum.ctx** with 720 characters, as follows:

Ciphertext (base 7)

[™N“M‘i¦eÀ•}R7tÁ»}àÀCnûÂž.J4eŸªwM□oàØázb.»o*ƒ*Ì□□i
ùe%§Š·¾-%û…ö*3í\„4T3ä□¨*ƒ*T,©–*ƒ*˜¾ •ä×lÎsG5□/&¥é³å0
?Çµ€Í?ržZ»¶ÒÒÃØõ¦lAæ{D¦;ëx#ÒYØ«nfñÂ>³6W½¡€€#}ÒèÄ
ª/´Ú”3¹Vû0ªþŠoÈ4Žú9Ë2\Š´‰·êKHÍˆ0ê□Òn¿þ1&è?Ž”’õÅn
l¥%FãÑD÷UcîÌOªü•ZE»Mh¸Øè„úØK\÷£NÎíÄDˆ×®nÍeÔªÄ´O. . . . . .

Cipher file (hex)

5B 99 4E 93 4D 91 69 A6 65 C0 95 7D 52 37 74 C1 BB 7D E0 C0 43 6E FB C2 9E
2E 4A 34 65 9F AA 77 4D 8F 6F E0 D8 E1 7A 62 2E BB 6F 83 CC 8F 7F 69 F9 65
25 A7 8A B7 BE 2D 25 FB 85 F6 2A 33 ED 5C 84 34 54 33 E4 7F A8 83 54 82 A9
96 83 98 BE A0 95 E4 D7 49 CE 73 47 35 8F 2F 26 A5 E9 B3 E5 30 3F C7 B5 80

CD 3F 72 9E 5A BB B6 D2 D2 C3 D8 F5 A6 6C 41 E6 7B 44 A6 3B EB 78 23 D2 59
D8 AB 6E 66 F1 C2 3E B3 36 57 BD A1 80 80 23 7D D2 E8 C4 AA 2F B4 DA 94 33
B9 56 FB 30 AA FE 8A 6F C8 34 8E FA 39 CB 32 5C 8A B4 89 B7 EA 4B 48 CD 88
30 EA 90 D2 6E BF FE 31 26 E8 3F 8E 94 92 F5 C5 6E 6C A5 25 AD 46 E3 D1 44
F7 55 63 EE CC 4F AA FC 95 5A 45 BB 4D 68 B8 D8 E8 84 FA D8 4B 5C F7 A3 4E
CE ED C4 44 88 D7 AE 6E CD 65 D4 AA C4 B4 4F . . . . . .

7 Deciphering

For deciphering the generator performes an identic course equal to the encryption process. Deciphering is performed in the coding area, but in reverse order, only.

1. Analysing the ciphertext
   ciphertext → cipher alphabet (0...127) → 7 bit index values
2. bit conversion
   7 bit index values (0 ...127) → 8-bit XOR concatenation
3. XOR concatenation
   8-bit XOR concatenation → block key → plaintext block

The program searches in blocks of **48** cipher characters the decimal index values of single characters in an identical created cipher alphabet (system alphabet) and connects them to a series of 336 bits. This series will be devided into **42** 8-bit sequences (336 bits) in bitsystem on base 8 and concatenated with the respective block key. The original plaintext becomes visible.

As an example deciphering the ciphertext file **Museum.ctx:**

At start the program reads the first ciphertext block with 48 characters:

[™N“M‘i¦eÀ•}R7tÁ»}àÀCnûÂž.J4eŸªwM□oàØázb.»o*ƒ*Ì□□i

Index:	106	21	54	78	26	72	1	63	8
(-1)	105	20	53	77	25	71	0	62	7

base 7:
1101001 0010100 0110101 1001101 0011001 1000111 0000000 0111110 0000111 . . .
conversion to base 8:
11010010 01010001 10101100 11010011 00110001 11000000 00111110 00001110 . . .
block key base 8:
10000110 00111001 11001001 11110011 01000010 10100011 01010111 01101011 . . .
XOR concatenation base 8:
01010100 01101000 01100101 00100000 01110011 01100011 01101001 01100101 . . .
Index base 8:

	84	104	101	32	115	99	105	101
ASCII:	T	h	e	□	s	c	i	e

As plaintext results: **The scie**nce of bird . . . .

8 Cryptanalysis

As is known cryptanalysis encompasses all attempts to draw any conclusion out of the ciphertext intended to recovery the plaintext. To noticeable events of any language belong statistically repetition patterns and word combinations, frequency structures and two-digit-groups and bigramms [#5]. But this circumstances require that plaintext and ciphertext are in a ratio of 1:1.

The most known attacks are analysing structures, „known plaintext attack“ and „chosen plaintext attack“, possibly still „differential“ and „linear“ analysis [#6]. By this attacks it is intended to find conspicuous connections in order to discover a way to the plaintext, possibly. To analyse this features it is necessary that ciphertext and plaintext share certain structures which can be compared really. Therefore a unit **order system** must exist which works in plaintext and ciphertext as well.

8.1 Current procedures

Nearly all current procedures have an equal length between plaintext and ciphertext "Congruence of Length" [#6]. Thus follows that there must exist for each single plaintext character a specific ciphertext character. Both areas – input and output - work within the same system alphabet (ASCII-character set), and therefore in the same bitsystem on **base 8**. Insofar a uniform order system exists and no bitsystem changes. Particularities in plaintext must be effective in ciphertext as well and might be found.

8.2 „CypherMatrix“ procedure

This conditions are not implemented in CypherMatrix procedures. Because of **bit conversion** the uniform order system between plaintext and ciphertext is not applicable. Consequently both areas can not be compared any longer. Congruence of length is missing. Due to conversion for each plaintext character there is a ciphertext character, which is by factor **1,143** (8/7) longer than the causative plaintext character. Besides the independent ciphertext alphabet contains only 128 elements, and the plaintext alphabet comprises 256 characters.

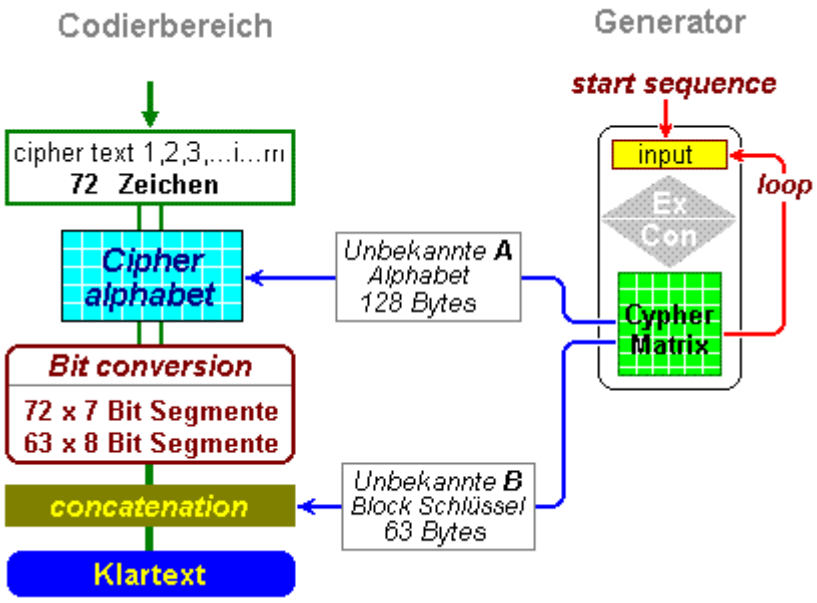
Cryptanalysis on the basis of certain distinguishing marks like repetition patterns, frequency structures and all other special features mentioned above require length of plaintext and ciphertext to be in a ratio of 1:1. But, uniform order system for both areas is missing and by this the base for all special attacks, as well. They will have **no effect** and we may forget them. In order to test this you may download some with CypherMatrix programs encrypted messages as ZIP-file and try to break the cipher with todays usual attacks [#7].

8.3 Security of the procedure

At least there is still the possibility to break the cipher by a „brute force attack“. An attacker in principle knows the **ciphertext** and the CypherMatrix **procedure**. But the respective program and single control parameters, including start sequence he does not know.

He may try an iteration of all possibilities. An attack on the start sequence with **42 bytes** length results to an entropie of **336** and an exponential complexity of **O(2^336) = 1.4E-101** An attacker using the ciphertext may then try to find single parts of encryprion steps. But there are no starting points which give some expectation for success.

Decryption happens in each round as follows:



The procedure includes three functions:

1. Plaintext block --> **block key** --> -8 bit XOR sequences
2. 8-bit XOR sequences --> 7-bit index values
3. 7-bit index values --> **chipher alphabet (128)** --> ciphertext

In this functions the parameters **block key** and **cipher alphabet** are two variables independent from each other. Effective are:

cm = f [f1 (an, **k1**), f2 (b1, b2), f3 (b2, **k2**)]
an = f [f3 (cm, **k2**), f2 (b2, b1), f1 (b1, **k1**)]

fx = functional connection
an = plaintext
k1 = **block key**
b1 = 8-bit sequence
b2 = 7-bit index value
k2 = **cipher alphabet (128)**
cm = ciphertext

Ascertaining the ciphertext (**cm**) and retrograde searching for the plaintext (**an**) points out as an equation with two unknown veriables: **k1** und **k2.** That leads to a definite solution, only, if one unknown variable can be derived from the other unknown variable or if there are two equations with the same unknown variables.

But between the respective block key = k1 and the cipher alphabet (128) = k2 generated in the same round there are no connections. Nevertheless, even if both are extracted out of the current CypherMatrix yet they have no functional connection: (**k1** --> (**Hk MOD 169**) **+1**) and **k2** --> (**Hk +Hp**) **MOD 255**)**+1**). The current CypherMatrix itself is derivated from the initial start sequence only. But thereunto is no way back (two one-way-functions prevent this). Hence, there are many couples of cipher alphabets / block keys which result from an „brute force“ attack and deliver any legible texts, but one does not know which is the right one: Angriff mit "brute force" . Thus „brute force“ cannot have success, too.

9 Summary

The foregoing explanations show some alternative principles:

1. An unity order system of bit series (structure and system alphabets) is not defined up to now,
2. Description of cryptography are confined to coding in bitsystem on base 8 (input and output) and
3. almost all attacking scenarios presuppose a comparable order system for plaintext and ciphertext (bitsystem on base 8 and system alphabet with 256 characters).

More details of **CypherMatrix** procedure under: www.telecypher.net/ [#8]
Who wants to deal with the procedures more intensively may request single programs – with or without source code – from the author per e-mail and work with them under the CMLizenz **.** (eschnoor@multi-matrix.de).

Munich, in March 2013

Notes

[#1] Bräkling, Andre, http://www.braekling.de/web-development/6-bits-und-bitfolgen.html
[#2] Schneier, Bruce, Angewandte Kryptographie (dt. Ausgabe), Bonn … 1996, S.229
[#3] Morin, Charles, www-lehre.inf.uos.de/~rspier/referat/feist.html
[#4] Esslinger, Bernhard, Uni Siegen, http://www.cryptool.org/de/
[#5] Bauer, Friedrich L., Entzifferte Geheimnisse – Methoden und Maximen der Kryptologie. Berlin Heidelberg New York, 1995, S. 186 ff.,
[#6] Strukturvergleich Klartext und Geheimtext, www.telecypher.net/Equilang.pdf
[#7] Download, www.telecypher.net/ZUSENDEN.HTM
[#8] Der Kern des CypherMatrix Verfahrens, www.telecypher.net/CYPHKERN.HTM
[#9] Schmeh, Klaus, Safer Net, Kryptografie im Internet und Intranet, Heidelberg 1998, S.61,