

# Bit and Byte Techniques in Cryptography

(Ernst Erich Schnoor)

Cryptography is writing and reading of secret messages by characteristic methods (encryption). Since millenia people changed signs of their language in such a manner the legitimate recipient only was able to decipher the message. They used exclusively whole characters of their concerned alphabet. Two methods were developed: "substitution" and "transposition" (#1). Substitution replaces the character at the same position by another character while transposition transfers the character to another position of the encrypted information. In both cases the **element** of information remains the same: the changeable **character**.

This view changed when computers came up. A computer works in number system on base 2, only (binary techniques). As a result of electronic techniques one can distinguish only between two states: "present" (voltage) or "not present" (no voltage) that is in digits of number system on base 2: "**one**" or "**zero**". Generally this state is defined as "**bit**". If a comparison states no difference then either "zero" is compared with "zero" or "one" is compared with "one" and the state is "zero" otherwise the state is "one" (XOR concatenation). A bit describes a state only, and nothing more. It cannot be bearer of an information. At most it can serve as „**flag**“.

An information is multilateral. At least it comprises two bits. In general: a series (**m**) of definite bytes (**a<sub>i</sub>**) with length (**n**). In order to analyze this series (as state of facts) it has to be systematized (scaled). For this each byte **a** has to be denoted with an index (**i**) and all **n** bytes have to be linked together in a qualified manner. The extended ASCII-system offers as indexes.

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

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

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

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

With reference to **Renè Descartes** (1596 – 1650) we know that every object (fact) – which is scalable in its demensions – is exactly determined by its coordinates for **subject**, **location** and **time** (cartesian coordinate system) (#2). We set:

- object: (**m**) digital information of length (**n**)
- subject: (**a<sub>i</sub>**) elements of information, signs, bytes
- location: (**p<sub>i</sub>**) position of byte **a(i)** inside the information
- time: (**t<sub>i</sub>**) point of time of byte **a(i)** inside the information

In order to distinguish single characters each byte  $a(i)$  is **position weighted** with its location  $p(i)$ . Time will be relevant only if there exists a functionally connection between a single byte and clock frequency. Normally this connection 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  $H(k)$ :

$$m = \sum_{i=1}^n (a_i + 1) * p_i * t_i \quad t_i = 1$$

Further details are explained in file: [Cryptographic Basicfunction in Byte Techniques](#)

In technical respects bit series are generated unlimited by computers. In order to work methodically with and analyse informations it is necessary to classify bit series i.e. scaling and deviding into definite segments (bytes). By systematic arrangement a „**World of Bytes**“ will arise.

Inspired by the possibilities of binary techniques the scientists called upon further developing cryptography obviously rushed to the "**bits**" and did no appropriate apportion to the basic systematic of „**bytes**“. „**Coding base 64**“ and **Ron Rivest's RC4** seem to be the fewer works based on bytes.

Similar as number series are devided into different number systems (from base 2 up to base 256) bit series can be devided into certain segments, as well. They may constitute a regulated system itself. Segments of 4 bits – for instance - are a "nibble" and segments of 6 bits form an alphabet of 6x6 signs as already realized in procedure of "coding base 64" (#3). Almost all other procedures employed in cryptography use segmenst of 8 bits. In informatic science rules the basic principle: **1 byte = 8 bits**.

But obviously this is one-sided and too narrow. Compared with numerical theory it would read: **1 number = 10 digits**. Of course this is evident in number system on base 10, only. In number system on base 16 (hexadecimal) a number comprises 16 digits and in number system on base 62 there are 62 digits in total.

The definition of bytes covers only a part of all possibilities. All uniform bit segments are qualified for beeing used as a single unity: beyond 8-bit for instance even 5-bit, 7-bit, 9-bit, 10-bit and 12-bit sequences, as well. The fact that mostly procedures are created in 8-bit segments has its seed in computer techniques and the traditional alphabet extended to 256 characters. By precisely consideration about bit series and their division into segments acomparison with terms of **numerical theory** seems to be analogous, as follows:

digits	-->	bits
numbers, elements	-->	sectors, signs, bytes
straight	-->	segment, block
ordered set	-->	array, alphabet
amount	-->	index, value

Bits comply with digit, byte with number and ordered set with array and alphabet. It seems to be useful to define appropriate terms similar to terms of number series. The term "number system on base .." comply with the term "byte system on base ..", for instance "**byte system on base 8**". By this view we get the following:

bit series of 2-bit	=	byte system on base 2	=	$2^2$ bytes	=	4 bytes
6-bit	=	byte system on base 6	=	$2^6$ bytes	=	64 bytes
7-bit	=	byte system on base 7	=	$2^7$ bytes	=	128 bytes
8-bit	=	byte system on base 8	=	$2^8$ bytes	=	356 bytes
12-bit	=	byte system on base 12	=	$2^{12}$ bytes	=	4096 bytes
16-bit	=	byte system on base 16	=	$2^{16}$ bytes	=	65536 bytes
32-bit	=	byte system on base 32	=	$2^{32}$ Bytes	=	4294967296 bytes

The quantity of bytes in a byte system form the corresponding alphabet. Alphabet on base 6 contains 64 bytes (characters), alphabet on base 7 has 128 characters and alphabet on base 12 comprises 4096 signs.

The smallest byte system on base 2 comprises 4 elements (bytes): 00, 01, 10, 11. By this system – for instance – may be installed an information about series of movements: **11** = straight ahead, **01** = to the right, **10** = to the left and **00** = no motion. This example shows that one bit only cannot bear any information, it have to be at least two bits.

The alphabets are completely independent of each other. Difficulties rise up only when defining own designs (characters) for each single byte. In nowadays practice mostly the extended ASCII-character set is used: alphabet of byte system on base 8. Unicode serves for byte system on base 16 (#4). In default of own designs all other byte systems must derive the necessary characters from extended ASCII-code. In higher byte systems (i.e. 12-bit segments) it is possible to combine two signs of lesser byte systems. Due to this variety of possibilities it seems to be meaningful to systematize them to a „**Byte theory**“.

As per algebraic number theory even in „byte theory“ it must be possible to convert bits (resp. bytes) from one byte system to another byte system. But in this case number of bits have to be equal. For instance converting **63** bytes from byte system on **base 8** (504 bits) results in **72** bytes in byte system on **base 7** (504 bits). In case of double converting 84 bytes from byte system on base 8 (672 bits) will arise to 56 bytes in byte system on base 12 (672 bits) which may be converted back to 96 bytes in byte system on base 7 (672 bits).

On this basis a lot of encryption procedures may be developed. For instance: converting from plaintext alphabet with 64 characters (byte system on base 6) to byte system on base 5 with an alphabet of 36 characters.

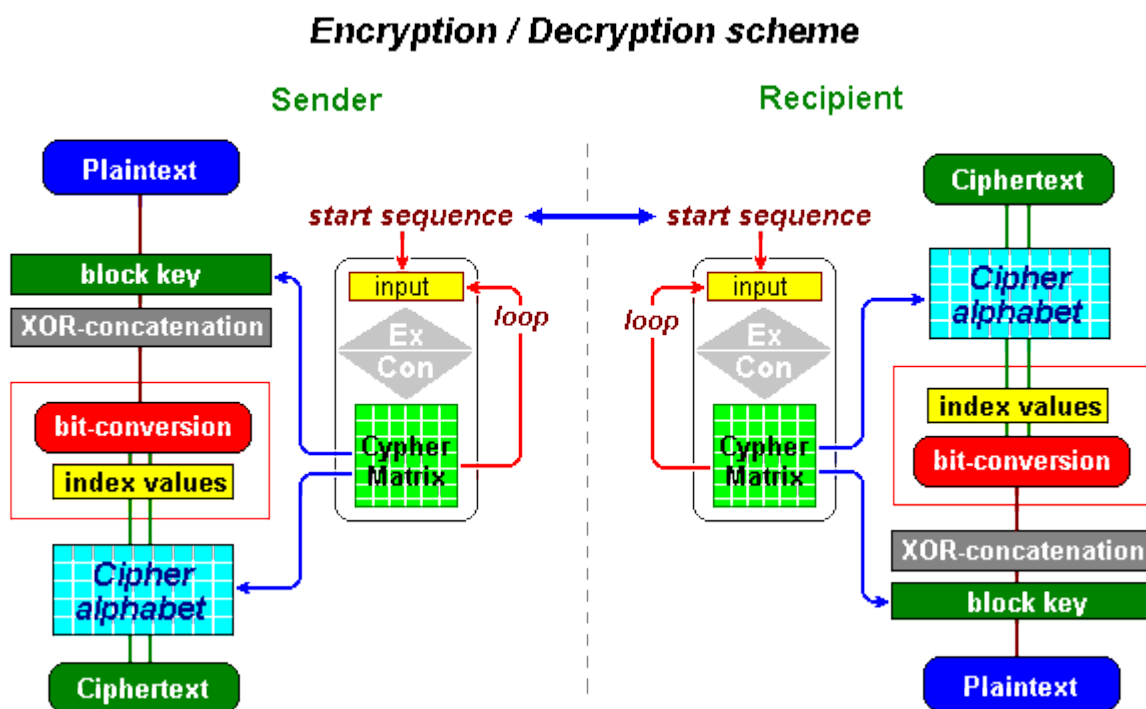
Another example: Plaintext blocks of 63 bytes in byte system on base 8 are first XOR concatenated with byte blocks (keys) of same length and then resulting 8-bit sequences are converted into 7-bit sequences (**bit conversion**). The value of this sequences (+1 = 1 to 128) as indexes get 72 cipher signs from an independent alphabet of 128 characters and combine them to the ciphertext.

8-bit XOR-sequences rearranged into 7-bit sequences (byte system on base 7):

8-bit: **111011111000111100110101010111110101001110111101000100** ....  
 7-bit: **111011111000111100110101010111110101001110111101000100** ....

The separated 7-bits are only **indexes** (pointers) to positions in array **Cipher Alphabet** of 128 bytes which have been extracted from the respective CypherMatrix. The process is similar to **coding base 64** where 8-bit segments are converted to 6-bit segments (byte system on base 6) which each get a specific cipher character from a 64 elements alphabet.

The following scheme shows the combined working of basic function and coding area:



There is only one problem left: from where to take key sequences (block key) and independent alphabets (cipher alphabet)?

A basic function, named "**CypherMatrix**" procedure, serves this problem. A start sequence of optimal 42 bytes initializes the function. This input is expanded into a higher number system (e.g. base 77), then again reduced by MODULO 256 to

BASIC VARIATION (256 elements) which are converted into byte system on base 8 and stored in a matrix of 16x16 bytes (CypherMatrix). Controlled by the start sequence all destination Data and parameters necessary for encrypting (block keys and cipher alphabets) are taken from the current CypherMatrix. At principle the basic function can be applied in any byte system.

Mostly encryptions are performed as "substitutions". That is evident especially because all applications are grounded on byte system on base 8, both plaintext and ciphertext as well. It is explicit required that plaintext and ciphertext should be of equal length. Consequently following of equal length there exists for each plaintext character one definite ciphertext character - even at the same position as in plaintext: that means: "substitution".

Working with byte system on base 8 implies a restriction: "transpositions" as an essential part of classical cryptography will barely (or even not) be realized in todays applications. Because for this converting of byte systems should be necessary: for instance for converting from 8-bit sequences into 7-bit sequences (**bit conversion**). Of course the ciphertext will extend at a ratio of 7 to 8 length units. Each plaintext character has an adequate sign of  $8/7 = 1.143$  ciphertext character. Equal positions of plaintext characters and ciphertext characters are not possible. Even in many other situations converting of byte systems would be appropriate.

More details about bytes used in cryptographic solutions you will find in internet:

[www.telecypher.net/INDENG.HTM](http://www.telecypher.net/INDENG.HTM)

From there you may download several DEMO programs and test all by yourself.

(#1) Bauer, Friedrich L., Entzifferte Geheimnisse, Berlin Heidelberg New York.  
1995, p. 36 and 78

(#2) level of abstraction for each scaleable fact, derived from  
cartesian coordinate system (as to René Descartes).

(#3) "coding base 64"

(#4) unicode

Munich, in January 2010  
**Ernst Erich Schnoor**

---