Decipher this!

By Ciprian Galaon

Manchester, 18th of June 2014

This article is about a new polyalphabetic ciphering method. It is therefore based on substitution, using multiple substitution alphabets, i.e. using various ciphers to encrypt the message. The method described here seeks to improve, by adding security, the existing polyalphabetic methods which evolved since 1467 when first created and used by Leon Battista Alberti. What makes this method different and safer against decryption is the fact that - apart from the alphabet of, say, 27 letters - it allows the use of an indefinite number and types of characters/symbols, with the exclusive function to divert/distract and add redundancy to the system, providing a significantly improved resistance to frequency and (table) positioning analysis.

For simplicity and clarity sake, I only considered here the extended (the most important) ASCII characters used in English language; this means that any other set of printable characters (from Russian, Greek or Arabic) - combined or not and not limited to a certain number of them -, that have a corresponding standardised computer code, can be used.

Extended ASCII symbols used in English language:

(space) ! " # $ % & ' ( ) * + , - . / 0 1 2 3 4 5 6 7 8 9 : ; < = > ? @ A B C D E F G H I J K L M N O P Q R S T U V W X Y Z [\ ] ^
_ ` a b c d e f g h i j k l m n o p q r s t u v w x y z { | } ~ ø ƒ „ … † ‡ ˆ ‰ Š ‹ Œ Ž ' ' " " o - - ˜ ™ š › œ ž Ÿ ÿ ¡ ¢ £ ¤ ¥ ¦ § ¨ ©
ª « ® ¯ ° ± ² ³ ´ µ ¶ • ¸ ¹ º » ¼ ½ ¾ ¿

Using as sample this reference key

of the English alphabet,																										a switch,	10 digits to form coordinates finders									
a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z		0	1	2	3	4	5	6	7	8	9
£	©	#	a	Q	™	7	µ	¿	*	=	@	¶	&	S	d	ž	0	»	±	G	‰	^	ƒ	'	(*space*)	§	B	†	Œ	j	+	2	/	?	¢	m

the following matrix of positional scrambling:

0	B	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	
1	†	£	©	#	a	Q	™	7	µ	¿	*	=	@	¶	&	S	d	ž	0	»	±	G	‰	^	ƒ	'		15
2	Œ		£	©	#	a	Q	™	7	µ	¿	*	=	@	¶	&	S	d	ž	0	»	±	G	‰	^	ƒ	'	16
3	j	'		£	©	#	a	Q	™	7	µ	¿	*	=	@	¶	&	S	d	ž	0	»	±	G	‰	^	ƒ	17
4	+	ƒ	'		£	©	#	a	Q	™	7	µ	¿	*	=	@	¶	&	S	d	ž	0	»	±	G	‰	^	18
5	2	^	ƒ	'		£	©	#	a	Q	™	7	µ	¿	*	=	@	¶	&	S	d	ž	0	»	±	G	‰	19
6	/	‰	^	ƒ	'		£	©	#	a	Q	™	7	µ	¿	*	=	@	¶	&	S	d	ž	0	»	±	G	20
7	?	G	‰	^	ƒ	'		£	©	#	a	Q	™	7	µ	¿	*	=	@	¶	&	S	d	ž	0	»	±	21
8	¢	±	G	‰	^	ƒ	'		£	©	#	a	Q	™	7	µ	¿	*	=	@	¶	&	S	d	ž	0	»	22
9	m	»	±	G	‰	^	ƒ	'		£	©	#	a	Q	™	7	µ	¿	*	=	@	¶	&	S	d	ž	0	23
10	†B	0	»	±	G	‰	^	ƒ	'		£	©	#	a	Q	™	7	µ	¿	*	=	@	¶	&	S	d	ž	24
11	††	ž	0	»	±	G	‰	^	ƒ	'		£	©	#	a	Q	™	7	µ	¿	*	=	@	¶	&	S	d	25
12	† Œ	d	ž	0	»	±	G	‰	^	ƒ	'		£	©	#	a	Q	™	7	µ	¿	*	=	@	¶	&	S	26
13	†j	S	d	ž	0	»	±	G	‰	^	ƒ	'		£	©	#	a	Q	™	7	µ	¿	*	=	@	¶	&	1*
14	†+	&	S	d	ž	0	»	±	G	‰	^	ƒ	‘		£	©	#	a	Q	™	7	µ	¿	*	=	@	¶	2
15	†2	¶	&	S	d	ž	0	»	±	G	‰	^	ƒ	‘		£	©	#	a	Q	™	7	µ	¿	*	=	@	3
16	†/	@	¶	&	S	d	ž	0	»	±	G	‰	^	ƒ	‘		£	©	#	a	Q	™	7	µ	¿	*	=	4
17	†?	=	@	¶	&	S	d	ž	0	»	±	G	‰	^	ƒ	‘		£	©	#	a	Q	™	7	µ	¿	*	5
18	†¢	*	=	@	¶	&	S	d	ž	0	»	±	G	‰	^	ƒ	‘		£	©	#	a	Q	™	7	µ	¿	6
19	†m	¿	*	=	@	¶	&	S	d	ž	0	»	±	G	‰	^	ƒ	‘		£	©	#	a	Q	™	7	µ	7
20	ŒB	µ	¿	*	=	@	¶	&	S	d	ž	0	»	±	G	‰	^	ƒ	‘		£	©	#	a	Q	™	7	8
21	Œ†	7	µ	¿	*	=	@	¶	&	S	d	ž	0	»	±	G	‰	^	ƒ	‘		£	©	#	a	Q	™	9
22	ŒŒ	™	7	µ	¿	*	=	@	¶	&	S	d	ž	0	»	±	G	‰	^	ƒ	‘		£	©	#	a	Q	10
23	Œj	Q	™	7	µ	¿	*	=	@	¶	&	S	d	ž	0	»	±	G	‰	^	ƒ	‘		£	©	#	a	11
24	Œ+	a	Q	™	7	µ	¿	*	=	@	¶	&	S	d	ž	0	»	±	G	‰	^	ƒ	‘		£	©	#	12
25	Œ2	#	a	Q	™	7	µ	¿	*	=	@	¶	&	S	d	ž	0	»	±	G	‰	^	ƒ	‘		£	©	13
26	Œ/	©	#	a	Q	™	7	µ	¿	*	=	@	¶	&	S	d	ž	0	»	±	G	‰	^	ƒ	‘		£	14

*by placing, for example, ¶ before the number that follows in the ciphertext, all the default arrangements in the matrix (starting 1-£-a, “£” is the reference key in the reference alphabet) change as indicated.

and the following redundant or distractive ASCII symbols to be used, either in solitary or in any other amount randomly mixed, in blocks with the exclusive function of representing spaces between words, switches and coordinating numbers:

! $ % ( ) , - . 1 3 4 5 6 8 9 : ; < > A C D E F H I J K L M N O P R T U V W X Y Z [\ ] _ ` b
c e f g h i k l n o p q r s t u v w x y z { | } ~ ø “ … ‡ ˆ Š ‹ Ž ' ' " " o -- ˜ š › œ Ÿ ÿ ¡
¤ ¥ ¦ ¨ ª « ® ¯ ° ² ³ ´ • ¸ ¹ º ¼ ½ ¾

The following sample of plain text: *There are three basic encryption methods: hashing, symmetric cryptography, and asymmetric cryptography.*

There is virtually no limit in the number and combinations of redundant/distractive symbols; whenever used, the entire block of redundant symbols count always as one space and their function is to make deciphering significantly more difficult. For shorter messages, however, there should be the same proportion between distractive and key symbols between switches. Using every symbol in the distractive block at least two times and close to each other also increases deciphering difficulty.

After every switch (§) the structure is always the same: [redundant symbols] - [reshuffling symbol, here it is the default one: £] - [coordinating number/s].

Here, ¶ function as reshuffling symbol (see the right of the matrix). To increase difficulty, the coordinating numbers could be constituted by two blocks separated by redundant symbols; for example, 45689724 and 89986. In order to obtain two digits between 10 and 26, take for example: 4+5+6+8+9+7+2+4=45, 4+5=**9** and 8+9+9+8+6+4=44, 4+4=**8**. As 98>26, a further addition is required: **9**+**8**=17 and the key (and so, the reshuffled alphabet) used from the switch on until the next switch it will be that of the row ***17***. For one digit between 2 and 9 (row 1 is for reference only), there will be only one redundant/distractive block between the switch symbol and the coordinating number.

encrypts as follows:

17 *there* *are* *three* *basic*

¥1½ ¥'|%{T2……….‡ §9Š…….. ø£†?(ˆ…. ³a0S©S 9{o…….3Y =©S P{8$8…..68a0©SS ¹1,| Š œ1………”1 @=#»¶ J®Jv9..........kc

methods(colon) *45689724* *89986* *hashing(comma)* *symmetric*

^Sa0‘&# ¶‘‰‘ ƒ Wy4W………¡¤ §³z-…….1‡¶+2/¢m?Œ+vb….•¸¢mm¢/ <Me<……;>’ ™’ž™7@Q £¶==’ z$3z….>; ž^==#0d7£eU} ÿ….58

cryptography *and* *asymmetric* *cryptography(full stop)*

£d^&0¶Qd’&™^(\b$..........l, ‘@© w Ž‡%......1!1 ‘ ž^==#0d7£tu44……..~% £d^&0¶Qd’&™^a»** ž0¶&”cA}²……………………………….T$

In order to make the message more difficult to decrypt the remaining symbols (93 in our example) - apart from the capital letters of the alphabet - will be substituted in the message by their corresponding descriptive names whenever used in the plain text. Hence, “!” it will be represented with the text “exclamation mark”, “@” with “at symbol”, “%” with percent sign, “^{2}” with “superscript two”, and so on. Whatever the switch on which the parties agree, here §, this will always be functional. That means that, where non-functional, i.e. no switching nor activating scrambling of the symbols of the alphabet, it will work like any other redundant/diverting symbol and will randomly blend with them forming blocks that, according to the aforementioned rule, always count as one space. The difference between function and non-function will come from what subsequently follows after the switch symbol in the encrypted text. If any of the coordinating digits/numbers follows, the switch becomes functional and its function rule applies; and it does not in any other case, i.e. in the case that a symbol corresponding to a letter follows.

Thus the rules, apart from the classic rule of using a secret key, are:

- the use of redundant symbols as many as possible in unitary blocks, counting each one of the blocks as nothing else that one space;
- a matrix easy to represent graphically where the alphabets are disposed using as starting point the reference alphabet (the key) and the coordinating numbers, disposed all in natural order (the order alphabet-switch-digits is optional, i.e. could well be switch-alphabet-numbers or any other);
- a switch symbol marking where the use of one alphabet ends and a distinct one starts according to the coordinating number/s that follow in the encrypted message; if a letter follows, the switch has no function and works as any other redundant symbol;
- the message always starts and ends with a block of redundant symbols adding unsurmountable complexity to frequency and table positioning analysis and
- a reshuffling letter positioned always next after the switch (plus a space) that marks the position for the first (digit 1) coordinating number in the matrix; if there is no reshuffling letter between the switch and the number, the default arrangement starting with “1-a” applies.

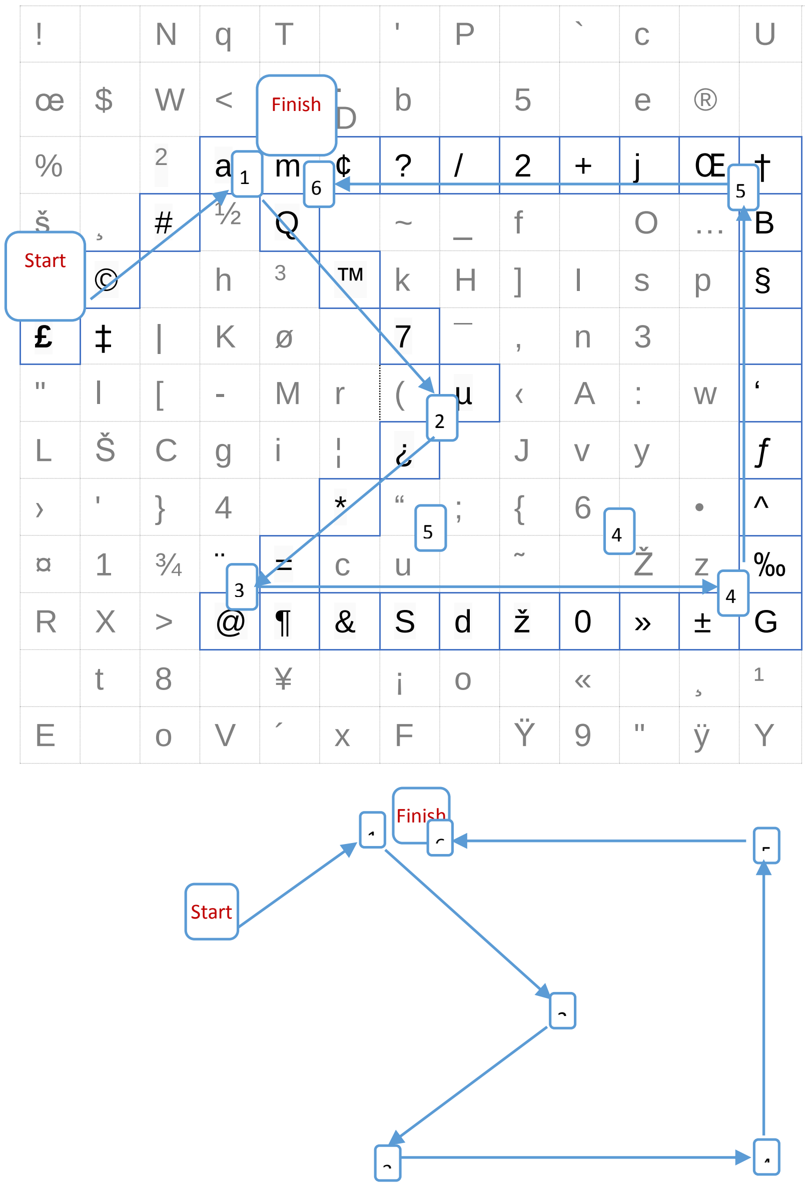
This method is more secure and have a competitive edge over any other methods because it:

- is open/public and yet secure, i.e., according to Kerchhoff's' principle: "a cryptosystem should be secure even if everything about the system, except the key, is public knowledge.";
- copes with the theorised problem of Single Point of Failure by adding, in a random way, redundancy to all potential points of failure;
- has a significantly larger ***e***, i.e. larger size of random permutations/transpositions of letters/symbols (in our example, $150! \approx 4.25 \times 10^{262}$; compare this with the number of atoms in the entire observable universe, which is estimated to be within the range of 10^{78} to 10^{82});
- does not require complex algorithms and strict and costly measures to keep the system secret and secure and
- does not necessarily requires the use of a computer or any IT applications for generating a new key or for encryption and decryption of the message, although a computer will considerably reduce the time required for operating the system and conveniently communicate.

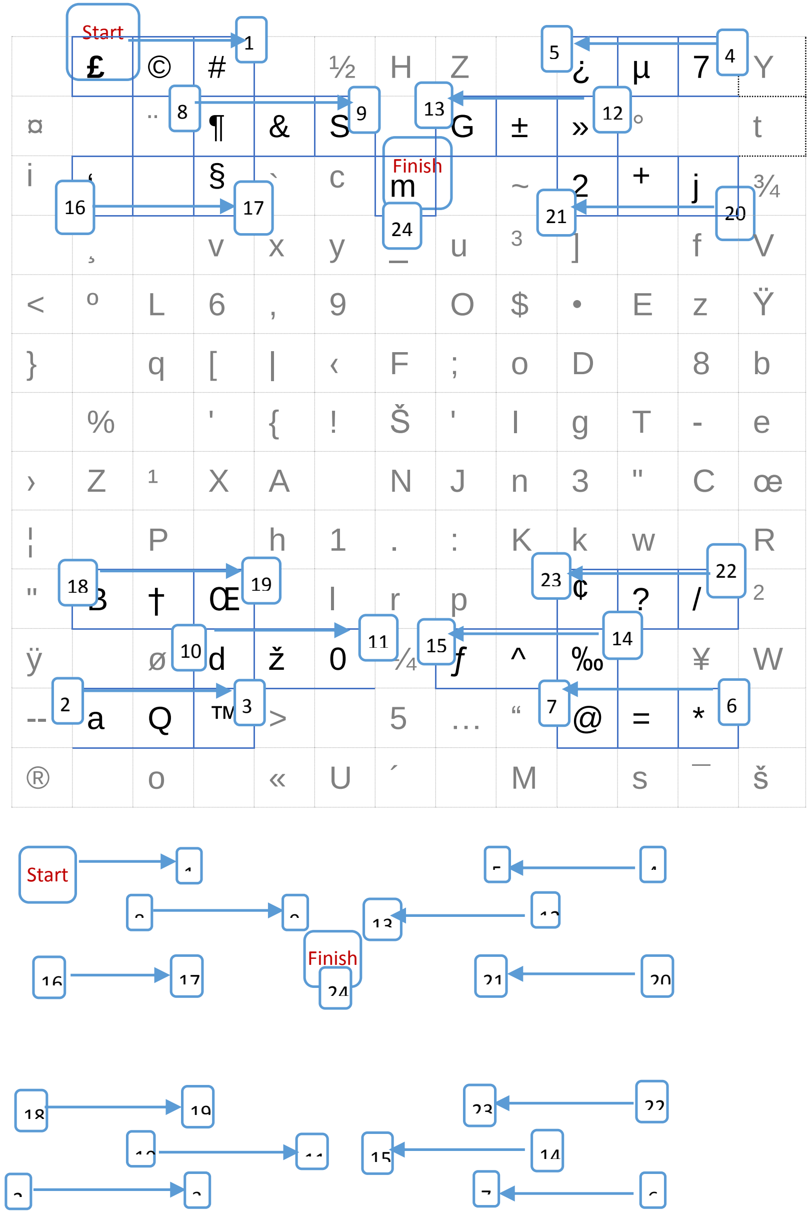
  In other words, the system has the potential of operating entirely on an analogic basis operated completely by humans, relying fundamentally on spatial/graphic positioning of the selected symbols and following minimal easy to remember rules and common knowledge, i.e. the natural order of the letters in the alphabet and the natural sequence of the (natural) numbers.

To keep the key secret, while open, on sight and always at hand, the following method could be used. Take a certain set of symbols (in this example the 150 symbols of the extended ASCII) and dispose them in a table of, say, 13 x 13 cells starting with the first of the 37 symbols (the alphabet, the switch symbol and the coordinating digits, in this precise agreed consecutive order). Dispose the symbols in an easy to remember pattern, in some respects, as the patterns used in un/blocking the screen of a modern smartphone. Then randomly fill the remaining cells with each of the remaining (113 in our example) agreed symbols. Although, in our example, 19 of the cells will remain blank, this will not affect in any way the effectiveness and security of the method; in any case, more symbols could be incorporated to fill the gaps, leaving nevertheless some of them blank in order to be confused with " " (space). The pattern chosen by the user could be a continuous line or not; it will always depend on the easiness of memorising it. Obviously, the more interrupted and multidirectional it is, provided it conserves its logic that makes it memorable, the more difficult it will be to find, intuitively of systematically. It will resemble the mine finder computer game, except that this mines never explode and give up their position. The next two examples give a sense of it:

Example one



Example two



The level of security that could be reached with this method makes it suitable both for private and official safe communications and data transfers/storage. Also, this method potentially constitutes the missing piece for secure, both wireless and through wire/cable, communications and transactions. Up-to-date, any of the technologies proposed by, for example, Apple or Google for electronic transactions were able to tackle the most important security issues, despite the huge amounts of capital invested in R&D to this end. The weaker link of this technologies stands virtually always on the users' end, on levels of security with which the memory of the device operates. Any smartphone software based on the method proposed here will have its safety guaranteed by the simple fact that it will not have to include by any means in its memory the key for en/decryption. A table as proposed above can be used. At the beginning of each session, either to communicate or to transact using electronic money or sensitive data, the user will introduce on the screen the pattern corresponding to the secret key. During the operation, the application will save the key in its volatile memory and it will always delete it (formatting/purging the RAM) in the exact instant when the session ends; simple and secure!

I honestly hope that the method proposed here have the potential to bring an extra amount of security to our communications and make our lives better.

Ciprian Galaon
Lawyer-linguist

email: cgalaon@hotmail.com