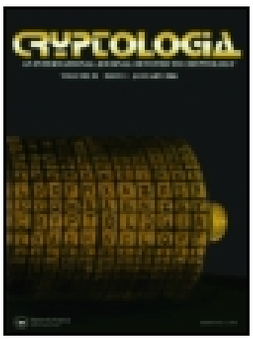

CRYPTOLOGIA

ISSN: 0161-1194 (Print) 1558-1586 (Online) Journal homepage: https://www.tandfonline.com/loi/ucry20

A PEDAGOGICAL CIPHER (This paper is the winner of the CRYPTOLOGIA First Annual Undergraduate Paper Competition in Cryptology.)

RICHARD OUTERBRIDGE

To cite this article: RICHARD OUTERBRIDGE (1982) A PEDAGOGICAL CIPHER (This paper is the winner of the CRYPTOLOGIA First Annual Undergraduate Paper Competition in Cryptology.), CRYPTOLOGIA, 6:4, 339-345, DOI: 10.1080/0161-118291857163

To link to this article: https://doi.org/10.1080/0161-118291857163

A PEDAGOGICAL CIPHER

RICHARD OUTERBRIDGE

This paper is the winner of the CRYPTOLOGIA First Annual Undergraduate Paper Competition in Cryptology.

Abstract: A method of using almost any simple calculator as a pseudo-random number generator is described in conjunction with a form of polyalphabetic substitution. The result is an interesting illustration of aperiodic stream encryption which might be useful in a course on cryptography or computer security as a tangible introduction to privacy transformations.

In an earlier article I described an adaptation of Rovner's method of key generation to a calculator with multiple constant memory registers [5][8]. Readers of Cryptologia may be familiar with the cryptographic capabilities of programmable calculators ([1],[2],[3],[7]) and it might be of interest to see how even the humble five-function calculator can be used as a cipher machine. As such, this is hardly a very impressive encryption scheme, but it possesses a certain elegance which may commend it as a pedagogically useful instance of aperiodic polyalphabetic substitution - the sort of stream encryption exemplified by many pre-WWII systems and by most of the simplest computer ciphers.

To begin with, the cipher is designed for any simple calculator with the following characteristics: it must be capable of multiplication by a constant multiplier; nonsignificant decimal zeros must be suppressed, even on overflow; when the result of a calculation exceeds the capacity of the display (i.e. overflows), it must be displayed as a mantissa in which only the excess digits are left of the desimal point; any error condition resulting from overflow must be clearable without affecting the display or the calculations in progress, i.e., you must be able to keep multiplying by the constant indefinitely. The precision the calculator maintains is crucial: you will only be able to talk to someone using a machine that maintains and displays the identical precision that yours does. Most if not all simple calculators maintain and display eight digits of precision, and that is the standard that is assumed herein. Finally, it may be easier on the eyes if the display is always right justified.

The Generator

Rovner's generator is a variant of the linear congruential multiplicative pseudo-random number generator, which can be expressed mathematically by the formula (s1 = (A * s0)mod B), where A and B are constants, s0 is the present term of the series, and s1 is the next term [4]. The difference is that rather than using the least significant portion of the previous result (modulo B), only the decimal representation of the most significant portion is retained, via the calculator's display. Unlike a conventional multiplicative generator there is no chance of the series ever degrading to zeros. To prime the generator, two seeds are needed: a constant multiplier (A) and the initial multiplicand (s0). The period of the generator depends on the value chosen for (A). Although an arbitrary value for A will not yield a series of maximal period, for the limited purposes of this cipher one does not do too badly so long as a few restrictions are observed.

The implementation demonstrated here differs from Rovner's original method in four important respects. First, only the last two digits of each displayed result are used for key. I found that periodicity first appears in the most significant digits, so the most varied series is obtained by ignoring them. Second, I'm using a twenty-three, rather than twenty-five, element homophonic number table. This is because a zero rarely appears in the rightmost display position. Rather than opt for a wider but less evenly distributed range of key values I decided to use a narrower and smoother one. However, since zero does occur on the right occasionally, special-case rules are needed when it does. Third, an eight-digit constant multiplier is used instead of a seven-digit constant divisor. Fourth, the alphabet applied to the homophonic table is not mixed. This weakening is for the sake of simplicity, as the added complexity of mixing and applying a permuted alphabet does not materially add to the demonstration of the cipher. This would not be wise were one actually using the cipher for whatever security it affords: pseudo-random number generators are notoriously vulnerable to the known plaintext attack, and even monoalphabetic encipherment of a pseudo-random source will give one some protection.

The Cipher: ''Cyph0''

The cipher is quite straightforward. Take any two eight-digit numbers such that neither the first nor the last digit of either is zero, and such that the same digit occurs no more than a total of four times throughout both. Consider each as a mantissa with seven decimals. Use one as the constant and the other as the initial multiplicand. For example, take any series of words made up of at least seventeen but no more than forty letters, say from an agreed upon book.

Number the letters in this keyphrase according to the alphabetic order of appearance of the letters in the phrase, and take off two eight digit numbers from left to right. For example, if we use the words "intended to bear an inscription" as our keyphrase, then, from

intended to bear an inscription
04575586 69 3912 26 17443217308

the keying values, or seeds, would be "4.5755866" (the constant multiplier) and "9.3912261" (the first multiplicand). Enter these into the calculator.

CYPHØ Keying Tables.

ht-23

	aa	bb	cc	dd	
A	00	25	51	76	A
B	01	26	52	77	B
C	02	27	53	78	C
D	03	28	54	79	D
E	04	29	55	81	E
F	05	31	56	82	F
G	06	32	57	83	G
H	07	33	58	84	H
I	08	34	59	85	I
J	09	35	61	86	J
K	11	36	62	87	K
L	12	37	63	88	L
M	13	38	64	89	M
N	14	39	65	91	N
O	15	41	66	92	O
P	16	42	67	93	P
Q	17	43	68	94	Q
R	18	44	69	95	R
S	19	45	71	96	S
T	21	46	72	97	T
U	22	47	73	98	U
V	23	48	74	99	V
W	24	49	75	()	W
	aa	bb	cc	dd	

AZza Ciphering Alphabets

A: abcdefghijklmn azyxwvutsrqpon	B: abcdefghijklm zyxwvutsrqpon	C: abcdefghijklmz yxwvutsrqponmz
D: abcdefghijkly xwvutsrqponmz	E: abcdefghijklxy wvutsrqponmlzy	F: abcdefghijkwx vutsrqponmlzy
G: abcdefghijkvwx utsrqponmlkzyx	H: abcdefghijuvw tsrqponmlkzyx	I: abcdefghijtuvw srqponmlkjzyxw
J: abcdefghistuv rqponmlkjzyxw	K: abcdefghirstuv qponmlkjizyxwv	L: abcdefghqrstu ponmlkjizyxwv
M: abcdefghpqrstu onmlkjihzyxwvu	N: abcdefgopqrst nmlkjihzyxwvu	O: abcdefgnopqrst mlkjihgzyxwvut
P: abcdefmnopqrs lkjihgzyxwvut	Q: abcdeflmnopqrs kjihgfzyxwvuts	R: abcdeklmnopqr jihgfzyxwvuts
S: abcdejklmnopqr ihgfezyxwvutsr	T: abcdijklmnopq hgfezyxwvutsr	U: abcdhijklmnopq gfedzyxwvutsrq
V: abcghijklmnop fedzyxwvutsrq	W: abcfghijklmnop edczyxwvutsrqp	X: abefghijklmno dczyxwvutsrqp
Y: abdefghijklmno cbzyxwvutsrqpo	Z: acdefghijklmn bzyxwvutsrqpo	BJKQVWXZ ;,:?"-.P

Figure 1.

For each letter in the message a keyletter is generated by multiplying the display (the current multiplicand) by the constant, reading the last two digits of the new display, and referencing into the ht-23 homophonic number table (Figure 1). Beside that row of the table is the next keyletter, ranging from 'A' to 'W'. If the last two digits of the display happen to be '00', the ht-23 can be referenced immediately, yielding 'A'. If the last two digits are a multiple of ten though, shift left one digit and use the third- and second-to-rightmost digits of the display to reference into the table (these are the

special case rules for rightmost zeros). When the display from time-to-time overflows, use the displayed result, then clear the error condition and continue.

The actual ciphering transformation is performed through the AZza ciphering alphabets (Figure 1), a system of reversible substitution based on the notion of radix complementation [6]. The complement of a number, to a certain modulus or radix, is simply the radix minus the number, mod the radix. For example, the complement of 10, radix 100, is 90; the complement of 90, 10; of 0,0; of 15, 85; and so on. The operation performed by the tables yields the complement, modulo 26, of the sum of any two letters. Find the alphabet indexed by either letter. In that alphabet find the other letter. Above or below it is the radix 26 complement of the sum, expressed as a letter where A = 0 and Z = 25. It will be seen that the alphabets may be entered using the keytext, plaintext or ciphertext letter as an index while still producing the same correspondence between the other two. I felt these tables would be easier for most novices to grasp than the usual tableau or slides, as well as being much easier to use in practice.

To continue the example, using the keying values derived above, say our plaintext begins "control over the flow of . . .". Then:

```
key ......:  qmfbi tgnut ehwdt ktkwu
plain.....:  contr olove rthef lowof
cipher....:  IAIGB TJZLD FAXTC FTUQB
```

where 'q' (16) plus 'c' (2) gives (26-18 = 8) 'I', with the remainder of the message, as it appears encrypted, in Figure 2. Note that a calculator displaying and maintaining eight digits of precision is assumed here. To decipher the message, simply apply the keystream to the ciphertext and reference into the AZza alphabets as though one were "enciphering" it. As always, remember, never reuse any keys.

Measures of Performance

I have neither analyzed the mathematics of the generator nor tested it exhaustively. Without a theoretical model it is impossible to make a priori predictions of performance, and in particular to refine the rules to govern the choice of the multiplier and thus the period length. While using an arbitrary multiplier is risky in that the resulting series will almost certainly be of less than maximal length (and what is worse, of unknown length) this may be balanced by the expansion of the key space that results. However, I have done some empirical testing. Starting with the same multiplier and multiplicand two generators were cycled, one performing two iterations on each loop, until

the multiplicands were again equal (the Floyd test [6]). The test procedure also watched for occurrences of the first five digits of the original multiplicand in the resultant series. The multiplicand triggering the Floyd condition was never the same as the original.

[001]	iaigb	tjzld	faxtc	ftuqb	fwuws	mvelj	gcfot	zhrfh
[002]	oddkm	mdxqa	budoh	mhkdg	noblm	yfbdq	buupx	gifgs
[003]	tqvgu	oydeh	okiif	lsnws	bhbcs	qcgfw	bruhc	mqldh
[004]	hsikf	yohaw	ydoka	tojch	ngbsf	hskji	pfvvf	yodvf
[005]	amxaa	fgxpp	edlwi	awsgy	voimx	quggs	iffqj	eovqb
[006]	turte	krmdg	leymd	zojgi	ajwvk	xfhht	gzoxp	wejgs
[007]	clfbz	goozj	xkthe	bkvwe	smsqk	wlftw	xetyl	wgazr
[008]	umdiq	vjana	lahod	eeshe	vajzz	lshii	zopmb	rltyy
[009]	gouqf	qcscn	gohzg	wmfxk	knzft	yykau	dfjvy	phlci
[010]	gxmlf	jurcw	sujwk	dtqje	wymdg	fmdkt	khbtq	rzetc
[011]	rsdjq	lteoh	wswan	igdgn	yoxrx	xnsgd	xkzee	voyue
[012]	bqlvh	xxbte	qlslf	vncwp	txfis	cdjsw	ybvyj	jcust
[013]	clwcf	njrcn	rosxg	kwosf	khhat	uelnr	bhnzf	vcwfs
[014]	qnllq	hihyq	joumv	lcrwx	aofua	fbrbx	wegdo	zwkgy
[015]	bufzs	lkmig	nmdhd	bpzgc	dxxlk	elwro	seyqu	fhrkg
[016]	awmbv	tcrhk	cnohb	udbpi	pcurp	znspv	iexud	phkkj
[017]	szdug	ddlry	rpcde					
[017]	[<131>]							

Figure 2.

Repetitions of the first five digits of the starting multiplicand were periodic, with a period that varied from multiplier to multiplier. This period was never a factor of the prime period (i.e., the length of the series as a whole). Such repetitions did not occur with every multiplier I tested. Using the constant multiplier of the example, 4.5755866, I found that the prime series had a period of 192,179 with no repetition of the first five digits. The constant 8.6132447 yielded a period of 9054. The constant 5.3147919 yields a prime series period of 288,634 with a subsidiary first-five period of 24,884. These are the lowest and highest I've found: not remarkable by any means, but probably adequate for pencil and paper cryptography. I should mention that tests with a generator maintaining sixteen digits of precision, but only using the least significant eight bits of the first nine digits (as one might adapt the system to computers) for keystream output yielded spectacularly higher prime series period lengths. It would seem that the greater the precision, the longer the obtainable periods are likely to be.

As far as I can tell, the keystream seems "quite random." When the keystream of the example is taken to 30,000 iterations it produces the following statistics (remember, without series repetition). As I expected, 'A' and 'W' were about 0.75 as frequent as the other letters, and so the chi-square test failed with a value of 160.975. Considering only the twenty-one letters from 'B' to 'V' I measured a chi-square of 14.853. I measured a one-letter entropy of 4.519423 bits (as compared to log2(23) = 4.523562 bits) and a two-letter entropy of 9.014288 bits per digram (as compared to log2(529) = 9.047124 bits. As Knuth demonstrates though [4], these tests are inadequate measures of the randomness of a generator and should not be taken at face value.

Conclusion

Compared to the state-of-the-art, ciphers such as this are already obsolete. Stream encryption is too unwieldy for some of the more important computer applications, and the security of one-letter polyalphabetic substitution without transposition function leaves much to be desired - especially when using a pseudo-random number generator to provide the keystream. Yet as a way of introducing novices to the intricacies of modern cryptography the cipher's simplicity is an asset. Only a few steps above pencil-and-paper systems, most students should have an easy time understanding how it works and what it is doing. At the same time, it sets the stage for modern cryptography by showing what it is not and yet where it evolved from. The mathematics is simple enough to readily admit of analysis. It will work with practically any calculator and if nothing else might make a marvelous educational toy. Moreover, it delimits such systems as the DES and public-key methods from the systems of the preceding era - many of which, on computers, were very much like the cipher described here.

Annotated References

1. Costas, J. P. 1981. The hand-held calculator as a cryptographic machine. Cryptologia. 5.2: 89-94.

2. Eckler, A. Ross. 1980. Some comments on the use of the HP 67/97 as a cryptograph. Cryptologia. 4.1: 51-53.

3. Ford, J. R. 1979. The HP 67/97 cryptograph. Cryptologia. 3.1: 43-50.

4. Knuth, D. E. 1969. The Art of Computer Programming: Volume 2. Seminumerical algorithms. Reading, MA: Addison-Wesley. Section 3.2.1.2. pp 18-20.

Almost all of chapter three, if not explicitly devoted to the testing and analysis of generators, is interlaced with ways to measure their performance by a number of criteria.

5. Outerbridge, R. 1980. Some cryptographic and computing applications of the Toshiba LC836MN memo note 30 pocket calculator. Cryptologia. 4.2: 89-94.

6. Pierce, Clayton C. 1977. Secret and Secure: Privacy, Cryptography and Secure Communication. Ventura, CA. pp 9-10.

In particular, the AZza alphabets are derived directly from the table notated ''Radix 26, Class 3, Count 001'' on page 74.

7. Richter, Michael. 1980. A note on public-key cryptography. Cryptologia. 4.1:20-22.

8. Rovner, L. 1974. Una Nueva Posibilidad Encryptografia. Buenos Aires: Editorial Cuarto Mundo.