

SYSTEMATIC CRYPTOGRAPHIC DESIGN

by

John E. Hershey

A Thesis Submitted to the Faculty of the
DEPARTMENT OF ELECTRICAL ENGINEERING
In Partial Fulfillment of the Requirements
For the Degree of
MASTER OF SCIENCE
In the Graduate College
THE UNIVERSITY OF ARIZONA

1 9 6 8

STATEMENT BY AUTHOR

This thesis has been submitted in partial fulfillment of requirements for an advanced degree at The University of Arizona and is deposited in the University Library to be made available to borrowers under rules of the Library.

Brief quotations from this thesis are allowable without special permission, provided that accurate acknowledgment of source is made. Requests for permission for extended quotation from or reproduction of this manuscript in whole or in part may be granted by the head of the major department or the Dean of the Graduate College when in his judgment the proposed use of the material is in the interests of scholarship. In all other instances, however, permission must be obtained from the author.

SIGNED: John E. Hershey

APPROVAL BY THESIS DIRECTOR

This thesis has been approved on the date shown below:

Frederick J. Hill 2/13/68
Frederick J. Hill Date
Associate Professor of
Electrical Engineering

ACKNOWLEDGMENT

To Professor Hansraj Gupta, DUBDC KNTLG QDCKB
LYOHX BLISP BQDDP DFCLC OQGDK CO; to Professor Fred Hill,
BEFYC UBQCO QTEQV OELYO PLORL C; to Professor Roy Mattson,
LIGSP AZLTH DGAPL CIOXL THDGA P; to Professor Granino A.
Korn, LISTC FGTHM IKCRG TFOLC DAOHB LXOIO XHQTO.TSX; and
finally to W. Martin and B. Mitchell, LIMUP LPQQB YKOHB
HWTUO IMBQK NDYCK IOSUD WQCDQ BTS.

TABLE OF CONTENTS

	Page
LIST OF ILLUSTRATIONS.	vi
LIST OF TABLES	viii
ABSTRACT	ix
 CHAPTER	
I MATHEMATICAL FUNDAMENTALS	1
Elements of the Theory of Numbers	1
Some Elements of Boolean Algebra.	7
The Concept of Information.	8
The Concept of Computation.	11
Transmission of Information	15
The Uniqueness Criterion.	18
II CLASSICAL CRYPTOGRAPHY.	20
The Substitution Cipher	20
The Autokey Cipher.	26
Accumulative Autokey Cipher	27
The Vigenere Cipher	31
The Infinite Key Vigenere Cipher.	32
III A STATISTICAL LOOK AT CLASSICAL CRYPTOGRAPHY.	35
IV A PROCEDURE FOR FIRST ANALYSIS OF COMPLETELY DETERMINISTIC CRYPTOGRAPHIC MACHINES.	52
Methodology	52
Example of the Above Procedure.	53
Development of the Switching Notation	55
Uniqueness.	55
Decipherability	57
V TRANSMISSION CHANNEL CONSIDERATIONS AND PERFECT CRYPTOGRAPHY.	58

TABLE OF CONTENTS--Continued

Chapter		Page
VI	VARIABILITY.	63
	Introductory Remarks	63
	Conditional IOBSM's	65
	Group Structure.	71
VII	SYSTEM DESIGN CONSIDERATIONS	73
	Defectors and Compromise	73
	Psychology	74
	Physical Security.	75
	Operational Security	75
	Theoretical Security	76
	Variability.	76
	Reliability and Simplicity	77
	Partition and Restriction of Critical Knowledge.	77
VIII	EXAMPLE OF A MULTI-SUBSCRIBER CRYPTO SYSTEM.	79
	The Machine.	79
	Statistics of the Device	82
	Scheme of Operation.	82
	Problems and Solutions	84
	REFERENCES.	86

LIST OF ILLUSTRATIONS

Figure	Page
1.1 The Binary Entropy Curve.	9
1.2 Graph for the Entropy Example	11
1.3 Mapping Diagram for the Case of No Computation	12
1.4 Mapping Diagram for the Case of Computation .	13
1.5 A Binary Processor.	13
1.6 Typical Time Trace of the Voltage Level of the Tapped Wire.	16
1.7 Two Unambiguous Time Traces	17
2.1 The Diagram for the $k=0$ Case of the Reverse Caesar Cipher	21
2.2 A Bernoulli Source as an Input to a Flip-Flop.	28
2.3 Two State Birth-Death Markov Process.	30
2.4 The Binary Symmetric Channel (BSC).	33
2.5 An Equivalent Representation of the BSC	34
3.1 A Tri-Symbol Concatenation.	35
3.2 PDF for Normal English.	44
3.3 Cycled PDF.	44
3.4 H_1 Computed on Different Lags of the Vigenere.	48
3.5 Example of Autokey With Lag 1	49
4.1 Basic Switching Element	53

LIST OF ILLUSTRATIONS--Continued

Figure		Page
4.2	Compound (octopus) Machine.	54
5.1	The Propagation of Error in the Autokey Cipher.	59
5.2	Noisy Channel Distortion of Transmission of Blanks	60
5.3	Cipher System Employing a Shift Register. . .	61
5.4	The PDF for the Shift Register Output	62
6.1	Series Concatenation of Two IOBSM's.	64
6.2	Parallel IOBSM Structure	65
6.3	One-to-one Mapping of s Bit Strings	66
6.4	One-to-one Mapping of s Bit Strings	68
6.5	M^X Concatenation.	69
6.6	The t Machine	70
8.1	One of the 32 Elements of the Cryptographic Machine	79
8.2	The Transition Matrix for the Cryptographic Machine	80
8.3	The PDF for the RPRNG	81
8.4	A Possible RPRNG.	81
8.5	One of the 32 Patchbays (wired)	83

LIST OF TABLES

Table	Page
1.1 The Equivalence Classes of the Integers Mod 7.	4
1.2 Probability Table for Entropy Example.	10
1.3 Operation Table for the Binary Processor of Fig. 1.5.	14
1.4 Operation Table for the "and" Function	14
1.5 Operation Table for the XOR Function	15
2.1 The Permutation Polynomials for the First Four Primes.	26
3.1 Binary Code.	37
3.2 Frequency of Letter Occurrence in Normal English.	39
3.3 Single Letter Frequency Distribution for the Vigenere Infinite Key Cipher	42
3.4 Single Letter Frequency Distribution for the Autokey Lag 1 Cipher	51

ABSTRACT

In the unclassified and declassified literature concerning cryptography there is a notable lack of theory unification. Many cryptographic schemes have been proposed and probably used from time to time but few authors have bothered to present their particular theory in such a way that it can be easily contrasted with the others. There is also a sparsity of information concerning system definition. Just what is meant by a "successful cryptographic system"?

It is the purpose of this treatise to offer a framework within which new cipher systems can be examined. First some of the old ciphers are examined and their weaknesses pointed out. With this motivation for better cipher systems apparent, the problems of a secure crypto system are brought forth followed by some plausible answers.

CHAPTER I
MATHEMATICAL FUNDAMENTALS

Before embarking upon a mathematical formulation of cryptographic processes, it is necessary to be thoroughly familiar with the basic mathematical disciplines which are to be employed in the subsequent chapters. This chapter will present some of the classical concepts from the fields of number theory, boolean algebra, and information theory. The end of the chapter will be concerned with the very important "uniqueness criterion" which applies to cryptographic processes. This chapter is not meant to teach the basic definitions and theorems but rather to refresh the reader's memory.

Elements of the Theory of Numbers

Natural Numbers.

The subset of the positive integers $\{1, 2, 3, \dots\}$ is known as the set of natural numbers (1).

Prime Numbers.

A prime number is a natural number which always has only two distinct natural number divisors. Thus, 2, 3, 5, 7, 11, 13, etc. are prime numbers whereas 1, 4, 6, 8, 9, etc. are not.

Greatest Common Divisor.

The Greatest Common Divisor (GCD) of two natural numbers a, b , denoted by (a, b) , is defined as the largest member of the intersection of the set of all divisors of a and the set of all divisors of b . The set is non-empty as 1 is always present.

Relatively Prime.

If the GCD of a and b is 1, then a and b are said to be relatively prime. As examples, 10 and 25 are not relatively prime for $(10, 25) = 5$. 14 and 33 are relatively prime for $(14, 33) = 1$.

The Division Proposition.

For any two integers a, b such that $a > 0$, there are integers q, r such that $b = qa + r$, $0 \leq r < a$. If a does not divide b exactly, then we have the stricter inequality $0 < r < a$.

Relation.

If X and Y are any two sets, then any subset of $X \times Y$ is called a binary relation between X and Y (2).

Equivalence Relation.

An equivalence relation defined on a set X is a relation R which satisfies the three conditions:

1) For every x in X , $x R x$. x is R -related to x .

(Reflexive property)

2) For every x and y in X , $x R y \implies y R x$. If x is R -

related to y , then y is related to x . (Symmetric property)

3) For every x, y , and z in X , $x R y$ and $y R z \Rightarrow x R z$.

If x is R -related to y and if y is R -related to z , then x is R -related to z . (Transitive property)

Equivalence Class.

If R is an equivalence relation on the set X , then the equivalence class containing x is the set of all elements in X that are equivalent to x under the relation R .

An equivalence relation R causes a partition of the set X into disjoint equivalence classes. Every element of X is in one and only one equivalence class.

Congruence Relation.

$$a \equiv b \pmod{m} \Leftrightarrow a = b + km \Leftrightarrow m \mid (a - b).$$

Remark.

The congruence relation is an equivalence relation.

Propositions.

If $a \equiv b \pmod{m}$ and $c \equiv d \pmod{m}$, then $a + c \equiv b + d \pmod{m}$.

If $a \equiv b \pmod{m}$ and $c \equiv d \pmod{m}$, then $ac \equiv bd \pmod{m}$.

If $a \equiv b \pmod{m}$, then $a^n \equiv b^n \pmod{m}$.

If $ka \equiv kb \pmod{m}$, then $a \equiv b \pmod{\left(\frac{m}{(m, k)}\right)}$.

*From this point on, (m) will be used for \pmod{m} .

From a previous remark we expect that a congruence relation mod m will partition the set of integers into m equivalence classes. For an example, consider the congruence relation mod 7:

Table 1.1

The Equivalence Classes of the Integers Mod 7

.
.
.
-13	-12	-11	-10	-9	-8	-7
-6	-5	-4	-3	-2	-1	0
1	2	3	4	5	6	7
8	9	10	11	12	13	14
.
.
.

The columns are the equivalence classes mod 7. Note that $-13 \equiv -6 \equiv 1 \equiv 8(7)$ but $-13 \not\equiv -12(7)$ etc.

Complete Residue System (set).

A complete residue system mod m is a set derived by taking one and only one element from each equivalence class. In the previous example, a complete residue system mod 7 could be the set $\{-13, 2, -4, 4, -2, 13, 0\}$. A

mere obvious system would be $\{0,1,2,3,4,5,6,\dots\}$. For mod m , the principal residue set shall be defined as $\{0,1,2,\dots,m-1\}$.

Remark.

The complete residue system mod m forms a finite group under congruential addition.

Theorem.

If x spans (takes on all the values of) a complete residue system mod m , then $ax+b$ also spans a complete residue set mod m if $(a,m)=1$ and if b is any integer.

Proof: It is clear that if x spans m values then $ax+b$ will also span m values. If no two of the values produced by the transformation $ax+b$ are congruent, then $ax+b$ does indeed produce m incongruent values mod m which is the definition of a complete residue set mod m . Assume that two values produced by the form $ax+b$ are congruent. Then $ax_1+b \equiv ax_2+b \pmod{m}$. By the laws of congruences we may then deduce that $ax_1 \equiv ax_2 \pmod{m}$ and then that $x_1 \equiv x_2 \pmod{\left(\frac{m}{(a,m)}\right)}$. But $(a,m)=1$ so $x_1 \equiv x_2 \pmod{m}$ which contradicts the assumption that x_1 and x_2 belong to different residue classes.

Reduced Residue System (set).

If the integers are partitioned under the congruence relation mod m and if one and only one member is taken from each equivalence class whose members are relatively

prime to the modulus, then such a collection is termed a reduced residue set. For example, a reduced residue set mod 8 might be the following: {1,3,5,7}.

Euler Phi Function.

The number of elements in the reduced residue set is denoted by $\phi(m)$ where $\phi(m)$ is known as the Euler phi function. Mathematically,

$$\phi(m) = \sum_{\substack{(a,m)=1 \\ 0 < a < m}} 1$$

Note that $\phi(p) = p-1$ if p is a prime number.

Theorem.

If x spans a reduced residue set mod m , then ax also spans a reduced residue set mod m if $(a,m)=1$. The proof is similar to the previous proof.

Euler-Fermat Theorem.

Consider the two reduced residue sets mod m : $\{r_1, r_2, \dots, r_n\}$ and $\{ar_1, ar_2, \dots, ar_h\}$ [$(a,m)=1$; $h=\phi(m)$]. Each member of the first set is congruent to one and only one member of the second and vice-versa. Hence by multiplying congruences: $(ar_1)(ar_2)\dots(ar_h) \equiv r_1 r_2 \dots r_h (m)$. Thus $a^{\phi(m)} \equiv 1(m)$. This result is due to Euler. If $m=p$ is prime, then it is obvious that $a^{p-1} \equiv 1(p)$. This less general result is due to Fermat and is known as his "little theorem."

Some Elements of Boolean Algebra

The Exclusive-Or.

The exclusive-or is one of 16 Boolean functions of two variables. It is denoted by \oplus and defined by the relations: $0\oplus 0=1\oplus 1=0$; $0\oplus 1=1\oplus 0=1$. Other names for the exclusive-or are XOR, the half-add, addition modulo 2 with no carry, the inequivalence operator, and addition over the bi-element finite field. The exclusive-or or XOR as we shall call it from now on, is perhaps the most important Boolean function as regards its application to cryptography. The salient points to notice about it are:

- 1) There are an equal number of 1's and 0's in its table. (it "splits" its statistics.)
- 2) changing (complementing) only one variable changes the result of the operation.
- 3) it is commutative, $b_1\oplus b_2=b_2\oplus b_1$. (b_i is a bi-valued Boolean variable.)
- 4) it is associative, $b_1\oplus(b_2\oplus b_3)=(b_1\oplus b_2)\oplus b_3$.
- 5) $b_1\oplus b_1=0$.
- 6) if $b_1=b_2$, then $b_1\oplus b_3=b_2\oplus b_3$.

The Concept of Information

Information Theory.

Information theory is an attempt to put a quantitative measure on knowledge or "surprise" gained by the various outputs or outcomes of a process for which definite a-priori probabilities have been compiled. If we know that an event has an a-priori probability p of happening and we are then told that the event has indeed occurred, we are said to gain an amount of information equal to $\log_N \frac{1}{p}$. If $N=2$, the unit of information is the "bit." If $N=3$, the "tit," if $N=10$, the "Hartley," etc. For an example, suppose we know that the probability that a binary device will output a 1 is one-half. It is then clear that the information gained by an observer every time the device puts out a bit is $\log_2 \frac{1}{\frac{1}{2}} = 1$ bit. On the other hand, if something is sure to happen then there is no information gained by observing that the event did indeed occur. The probability that the sun will rise tomorrow is 1. We will therefore gain no information by staying up all night to see if it is really so.

If a given process has N possible outcomes with respective probabilities p_1, p_2, \dots, p_N , ($p_1+p_2+\dots+p_N=1$), then the average information gain per observation of the process is

$$H = - \sum_{i=1}^N p_i \log p_i.$$

It should be noted that:

- 1) H is never negative ($0 \leq p_i \leq 1$).
- 2) H is a maximum if $p_1 = p_2 = \dots = p_N = \frac{1}{N}$.
- 3) the maximum possible value of H is also dependent upon the number of possible outcomes of the process. Indeed,

$$\max H = -\sum_{i=1}^N p_i \log p_i, \text{ such that } p_1 = p_2 = \dots = p_N = \frac{1}{N}, = \log N.$$

A final observation is that for the binary case (two possible outcomes):

$$H = -\sum_{i=1}^N p_i \log p_i = -p_1 \log p_1 - p_2 \log p_2 = -p_1 \log p_1 - (1-p_1) \log (1-p_1).$$

This of course results from the fact that, $p_1 + p_2 = 1$. Note the symmetry present: p_1 may be replaced by $1-p_1$ and vice-versa.

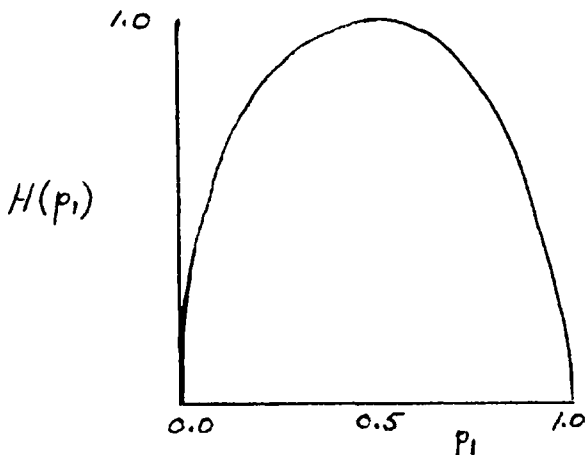


Figure 1.1 The Binary Entropy Curve

The symmetry noted above leads to some very interesting results. Consider the following example.

The President is in need of a particular adviser. The adviser must give a "YES" or a "NO" prediction on all forecasts for which he is responsible. (All forecasts are assumed to be of equal importance.) Three college graduates have applied for the job and by extensive civil service testing, the following table was compiled:

Table 1.2

Probability Table for Entropy Example

Candidate's school	Probability that candidate is correct
Harvard	0.8
Bicarb Inst. of Tech.	0.5
Canker University	0.1

At first it would seem that the Harvard graduate would be the best choice, however, the best choice is really the graduate from Canker U. This odd state of affairs arises because we are dealing with a binary case. If we always take the opposite of what the graduate from Canker U. says, we will be right 90% of the time. The answer becomes obvious if we examine the entropy curve:

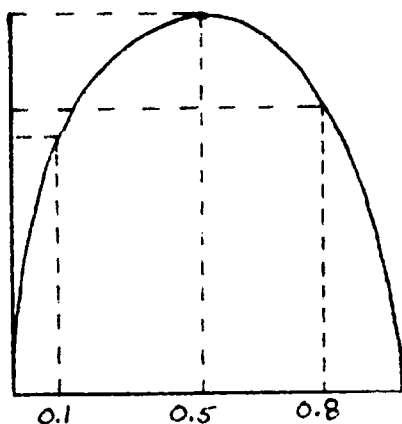


Figure 1.2 Graph for the Entropy Example

In essence we wish to gain the least amount of information from observing what actually occurs given the adviser's considered decision. In a real sense we wish to be "surprised" the least at the outcome of events if we are previously prejudiced by the adviser's pronouncement.

The Concept of Computation

Computation.

Computation is a statistical average loss of information. In information theory this loss of information may be viewed as the result of a many-to-one mapping. By way of illustration consider the following: a wire is at ground potential, 1 volt above ground, or -1 volt below

ground. (This set of signals $\{+1,0,-1\}$ could be the alphabet of a binary source with the character 0 representing a comma or space. If this wire is ready by a galvanometer to ground, we will be able to detect all three cases. A swing of the needle to the right will indicate +1 volt, null will indicate 0, and a swing to the left represent the 1- volt signal. The meter performs a one-to-one mapping of the set of signals $+1,0,-1$ onto the set of meter movements {swing right, null, swing left} (see Figure 1.3).

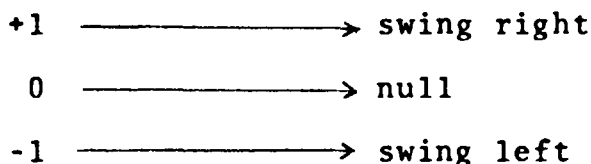


Figure 1.3 Mapping Diagram for the Case of No Computation

If then a meter is used, we have a one-to-one mapping and no computation is performed. If instead we were to place a light bulb from the wire to ground, we would note that the bulb would light whenever the output was +1 volts and would remain unlit when the wire was at ground potential. In this case it would be impossible to distinguish a +1 signal from a -1 signal. Thus, the bulb has performed the many-to-one mapping shown in Figure 1.4.

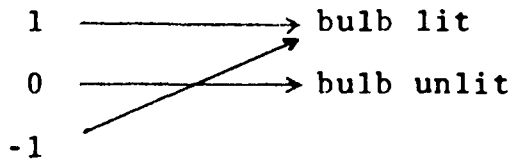
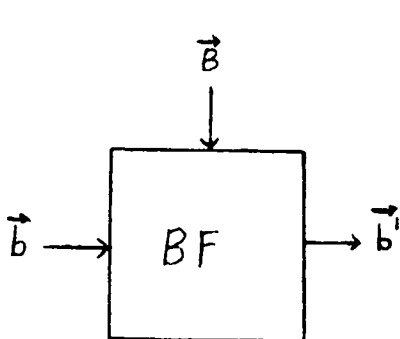


Figure 1.4 Mapping Diagram for the Case of Computation

Boolean functions also perform computation. Consider the diagram of Figure 1.5



\vec{B} is a known string of bits.
(We may observe these bits.)

\vec{b} is an unknown string of bits
which we hope to determine.

\vec{b}' is the string of bits which
results from the operation
of the Boolean function (BF)
on the two strings B and b .
We may also observe this
string of bits.

Figure 1.5 A Binary Processor

Our task is to determine the i th bit b_i in the string b given the i th bits B_i and b'_i in the respective strings B and b' . The following table may be used for any BF:

Table 1.3 Operation Table for the Binary Processor of Fig. 1.5

b_i	B_i	b'_i
0	0	
0	1	
1	0	
1	1	

If the BF is the "and" operation, the table becomes that of Table 1.5.

Table 1.4 Operation Table for the "and" Function

b_i	B_i	$b'_i = b_i B_i$
0	0	0
0	1	0
1	0	0
1	1	1

It is now clear that if $B_i = 1$ we can uniquely determine b_i . If b'_i is 0 then b_i must be 0 and if b'_i is 1 then b_i must also be 1. However, if B_i is 0, then b'_i will be 0 regardless of b_i . Thus, for the case $B_i = 0$, the BF performs a many-to-one mapping or what we have termed a computation.

If the BF is the XOR, then something very interesting happens. The table becomes:

Table 1.5 Operation Table for the XOR Function

b_i	B_i	$b_i' = b_i \oplus B_i$
0	0	0
0	1	1
1	0	1
1	1	0

It is now clear that b_i may always be uniquely determined given B_i and b_i' . This may be seen by carrying through the previous argument or by directly solving the equation

$b_i' = b_i \oplus B_i$ for b_i as a function of b_i' and B_i . The equation may be solved by XORing B_i to both sides of the equation and then using the associative property.

$$b_i' \oplus B_i = (b_i \oplus B_i) \oplus B_i = b_i \oplus (B_i \oplus B_i) = b_i \oplus 0 = b_i.$$

Transmission of Information

Parameters of Information Transmission.

It is well known that information can be transmitted by any source which employs two or more symbols in its alphabet. A source with the alphabet $\{0,1\}$ would be of the conventional binary type. What is usually not pointed

out is that continuous information transmission requires at least three parameters. These parameters in the conventional binary system are usually the two source alphabet characters and time. Usually a clock rate is assumed, that is, each symbol is transmitted for a specific, previously determined, and known time interval. Unless something is known or can be deduced about the timing or third parameter, it is impossible to transmit useful information. Consider a line which is tapped by a third party apart from the sender and his intended receiver. The eavesdropper determines that the line voltage takes on the two values 0 and 1 volt. A typical time trace might be that of Figure 1.6.

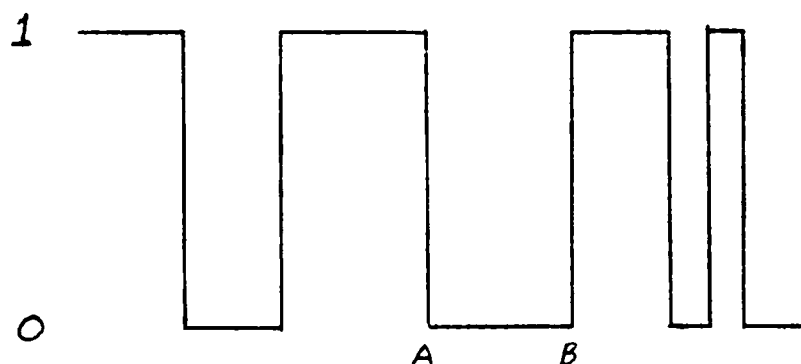


Figure 1.6 Typical Time Trace of the Voltage Level of the Tapped Wire

The Uniqueness Criterion

Decipherability.

When encoding, any cryptographic process may be thought of as a sequential machine with input being the clear (material to be enciphered) to be encoded and the output being the encoded message. When decoding, the input is, of course, the encoded message and the output is the clear. Such a process of encoding and decoding is subject to a uniqueness constraint. Let us assume that the device operates on a symbol per symbol basis - one encoded (decoded) symbol for every clear (encoded) symbol. Let us also assume that the clear and encoded symbols use the same alphabet. If the alphabet has m symbols then the cryptographic machine must, for each symbol input, decide which of the m possible symbols to output. At each step, then, it would appear that the machine would execute one of the m^m mappings of $\{S_1, S_2, \dots, S_m\}$ into $\{S_{\alpha_1}, S_{\alpha_2}, \dots, S_{\alpha_m}\}$ where $\{S_1, S_2, \dots, S_m\}$ is the set of possible input symbols for any particular instant of operation and $\{S_{\alpha_1}, S_{\alpha_2}, \dots, S_{\alpha_m}\}$ is the set of possible output symbols. There are m possible choices for each α_i and, therefore, m^m possible mappings. These are many-to-one mappings in general. On a second glance, however, we perceive that in order for us to determine the input uniquely given the output, we must restrict our attention

to the subset of one-to-one mappings of the form

$\{S_1, S_2, \dots, S_m\} \leftrightarrow \{S_{\alpha_1}, S_{\alpha_2}, \dots, S_{\alpha_m}\}$ where each S_i occurs

once and only once in $\{S_{\alpha_1}, S_{\alpha_2}, \dots, S_{\alpha_m}\}$. There are m

choices for S_{α_1} . Having assigned S_{α_1} , we note $m-1$ choices

for S_{α_2} , and so on. There are then $m!$ possible one-to-one

mappings. It must be kept in mind that the particular one-to-one mapping chosen can, depending upon the cipher design, change from one encoding/decoding operation to the next. For the simple substitution cipher, discussed in the following chapter, it is clear that a fixed one-to-one mapping is involved while for the autokey cipher, also discussed in the following chapter, it will be noted that the choice of one-to-one mapping is continually dependent upon the context of the clear.

CHAPTER II
CLASSICAL CRYPTOGRAPHY

The Substitution Cipher

One of the oldest and simplest of ciphers is the Character Substitution cipher. This cipher is exactly what its name implies as it is a set of rules which permutes the alphabet $\{A, B, \dots, Y, Z\}$. As an example, consider the permutation $\{A, B, \dots, Y, Z\} \rightarrow \{B, C, \dots, Y, Z, A\}$ where each letter is advanced by one position and the "Z" is advanced around the end to the "A". This particular substitution cipher is known as the Caesar cipher after J. G. Caesar who supposedly used it in communication with Rome. Using this cipher, the clear CUM GRANO SALIS would become DVN HSBOP TBMJT (3).

If we make the following assignment $\{0, 1, 2, \dots, 24, 25\} \leftrightarrow \{A, B, \dots, Y, Z\}$ and perform our computations modulo 26*, then it is clear that the Caesar cipher can be represented by the simple rule $C_i' = C_i \oplus k$ where C_i is the value of the ith character of the clear

*The symbol \oplus shall now stand for addition modulo 26 except where specifically noted as the XOR.

C'_i is the value of the i th character of the encoded message

k is the cycle distance ($k=1$ in the above example).

Another noteworthy example of the substitution cipher is the Reverse Caesar cipher whose rule is $C'_i \oplus C_i = k$. If $k=0$ the rule specifies the permutation $\{A,B,C,D,\dots,V,X,Y,Z\} \leftrightarrow \{A,Z,Y,X,\dots,E,D,C,B\}$. This type of permutation is often presented pictorially to aid the cryptographer. The $k=0$ case is so presented in Figure 2.1.

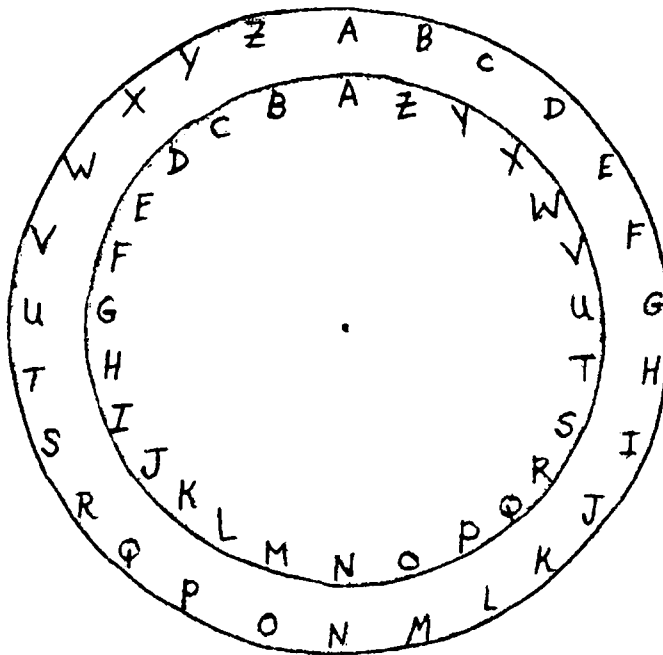


Figure 2.1 The Diagram for the $k=0$ Case of the Reverse Caesar Cipher

This "cipher wheel" is, incidentally, the emblem of the National Security Agency (NSA).

So far we have considered only two very special cases of the substitution cipher. For the most general case there is, unfortunately, no simple rule such as the Caesar rule $C'_i = C_i \oplus k$. In order to form an analytic rule for the general substitution cipher we must develop a "permutation polynomial." Such a polynomial is an algebraic form which will accomplish the required permutation. If we again assign $\{0,1,2,\dots,24,25\}$ to $\{A,B,C,\dots,Y,Z\}$ then it is obvious that the Caesar cipher's permutation polynomial is $x+k$ where x spans the complete residue set $\{0,1,2,\dots,24,25\}$ modulo 26. Thus we may write: CUM GRANO SALIS $\xrightarrow{x+1}$ DVN HSBOP TBMJT ($k=1$ in this case). The process of enciphering was:

- 1) Look at C.
- 2) Find its corresponding value which is 2.
- 3) Add 1 getting 3.
- 4) Find 3's corresponding letter which is D.
- 5) Repeat the process for U and all the remaining letters.

In general the permutation polynomial will be highly complex, but once formulated, the general substitution cipher becomes completely defined in an analytic form which is highly amenable to machine

computation. We now develop the necessary theory for constructing the permutation polynomials.

In Chapter I it was proven that if x spans a residue set mod m , then $ax+b$ where $(a,m)=1$, $b \in \text{Integers}$ also spans a residue set mod m . If we consider the principle residue set $\{0,1,2,\dots,m-1\}$ and then transform each element x by the form $ax+b$, we obtain the residue set $\{b, a+b, 2a+b,\dots, (m-1)a+b\}$. By repeated additions or subtractions of the modulus this set becomes a permutation of the principle residue set. Considering only the addition of b to each element and the subsequent congruential reductions, we obtain a simple cycling of the original set. For example, consider the principle residue set mod 5 $\{0,1,2,3,4\}$. By adding 4 to each term and reducing we obtain $\{4,0,1,2,3\}$ which is a simple cycling of $\{0,1,2,3,4\}$. Multiplication by a alone $(a,m)=1$ and subsequent reductions yield distinct permutations which are not cycles of each other. Again consider $\{0,1,2,3,4\}$. Multiply each term by 3. We obtain $\{0,3,1,4,2\}$ which is not a simple cycling of $\{0,1,2,3,4\}$ or of $\{0,a,2a,3a,4a\}$ $(a,m)=1$, $a \neq 3$. It is evident then that by using the form $ax+b$ we obtain $m\phi(m)$ permutations (ϕ is, of course, the Euler phi or totient function). Remembering that there are $m!$ permutations of m distinct elements, it is curious that the number theoretic equation

$$m \notin (m) = m! \quad [1]$$

admits $m=1,2,3$ as its solutions. Therefore all permutations of the 3 elements of the principle residue set mod 3 may be accomplished by an $ax+b$ transformation.

The question now arises as to a form which will accomplish all permutations for $m>3$. Proceeding by brute force we consider the polynomial

$$T_m(x) = r_{m-1}x^{m-1} + r_{m-2}x^{m-2} + \dots + r_1x + r_0 \quad [2]$$

Higher powers of x than x^{m-1} may be reduced by Fermat's Theorem to lower powers. By letting x span the principle residue set $\{0,1,2,\dots,m-1\} \pmod m$, we obtain m equations of the form

$$\sum_{i=0}^{m-1} r_i x_j^i = a_j \quad [3]$$

where x_j is the j th element of the principle residue set and a_j is the desired transformed value of the residue set. Symbolically

$$T_m(x_j) = a_j \quad [4]$$

These equations may now be solved for the r_i by applying a congruential Gauss-Jordan reduction. In performing the reduction we will be repeatedly concerned with divisions of the form $e|f$ where $0 < f < m$ and $0 < e < m$.

If $e \nmid m$ then the division may be performed if and only if there exists some multiple nm of m such that $e \mid f+nm$. We are therefore, interested in examining the equation $f+nm=0(e)$. It is here that we find we must restrict our problem. for $f+nm=0(e)$ will not have a solution in general. However, if we should choose $m=p$ prime, then we will always have a unique solution for $f+np$ can take on any value in the residue set mod e by proper choice of n as $(e,p)=1$. We can therefore, focus our attention on the following polynomials:

$$T_p(x) = r_{p-2}x^{p-2} + r_{p-3}x^{p-3} + \dots + r_1x + r_0 \quad [5]$$

Note that now only $p-1$ terms are necessary as $x^{p-1} \equiv 1(p)$.

The transform polynomials for the first four primes are found in Table 2.1. For an example of their use consider the problem of finding the transform polynomial for the permutation $\{0,1,2,3,4\} \rightarrow \{1,0,3,2,4\}$. We note that $a_0=1$, $a_1=0$, $a_2=3$, $a_3=2$, $a_4=4$. Thus the particular polynomial is $T_5(x) = 2x^3 + x^2 + x + 1$

$$T_5(x) = 2x^3 + x^2 + x + 1 \quad [6]$$

$T_5(0)=1$, $T_5(1)=0$, $T_5(2)=3$, $T_5(3)=2$, $T_5(4)=4$. (Note that $2x^3 + x^2 + x + 1$ is its own inverse $T_5^{-1}(x)$). By the very nature of permutations it is obvious that for any p the $T_p(x)$ polynomials for a group with identity polynomial x .

Table 2.1

The Permutation Polynomials for the First Four Primes

$$\{0, 1, 2, \dots, p-1\} \xrightarrow{T_p(x)} \{a_0, a_1, a_2, \dots, a_{p-1}\}$$

$$T_2(x) = (a_1 - a_0)x + a_0 = x + a_0$$

$$T_3(x) = (a_2 - a_1)x + a_0$$

$$T_5(x) = (4a_1 + 3a_2 + 2a_3 + a_4)x^3 + (4a_1 + a_2 + a_3 + 4a_4)x^2 + \\ (4a_1 + 2a_2 + 3a_3 + a_4)x + a_0$$

$$T_7(x) = (2a_0 + a_1 + 6a_3 + 5a_4 + 4a_5 + 3a_6)x^5 + (4a_0 + 3a_1 + 2a_3 + 2a_4 + 3a_6)x^4 + \\ (a_0 + 2a_3 + 2a_5 + 2a_6)x^3 + (2a_0 + a_1 + 5a_3 + 5a_4 + a_6)x^2 + \\ (4a_0 + 3a_1 + 6a_3 + 2a_4 + a_5 + 5a_6)x + a_0$$

The Autokey Cipher

The Autokey cipher is perhaps best described by means of an example. First assign $\{0, 1, 2, \dots, 24, 25\}$ to the alphabet $\{A, B, C, \dots, Y, Z\}$ as before. To form the encoded message, add sequentially (mod 26) to the clear first a pre-arranged phrase and then the clear itself. The number of characters in the pre-arranged phrase is

the "lag" of the Autokey cipher. Thus if the clear is:
 IT IS THE CAUSE MY SOUL and the prearranged phrase is:
 NOAH (giving a lag of four), the enciphered message is:
 VH IZBAM UTBWG UTBWG MS KSGJ (4).

IT IS THE CAUSE MY SOUL
⊕ NO AH ITI STHEC AU SEMY
 VH IZ BAM UTBWG MS KSGJ

The analytical rules for the Autokey are quite elementary. For a lag of k they are: $C_i' = C_i \oplus C_{i-k}$
 $(C_{1-k}, C_{2-k}, \dots, C_{-1}, C_0) = (k \text{ pre-arranged characters}).$

Accumulative Autokey Cipher

In addition to the many variations possible on the basic Autokey there is one that I call the Accumulative Autokey cipher. This method of enciphering involves no pre-arranged phrase and consists merely of adding (mod 26) all of the clear letters together up to and including the clear letter to be enciphered. The analytic rule is quite simply: $C_i' = C_0 \oplus C_1 \oplus C_2 \oplus \dots \oplus C_{i-1} \oplus C_i$. Using this cipher, the previous message would have come out: IB JB UBF
 HHBTX JH ZNHS.

The Accumulative Autokey cipher has some interesting statistical properties. Consider a Bernoulli source (a source whose output bits are statistically independent of

each other) with alphabet $\{0,1\}$. Let p be the probability that the source output will be one. If the source is connected as shown in Figure 2.2 the output of the circuit



(The output bit (b') is the state of the flip-flop which changes state every time the source outputs a 1 into it.)

Figure 2.2 A Bernoulli Source as an Input to a Flip-Flop

(pt. A) is now the Exclusive-Or of the k th bit of the source with all the previous bits and one initial condition bit (b_0) which is the initial setting of the flip-flop. Analytically: $b'_k = b_k \oplus b_{k-1} \oplus \dots \oplus b_2 \oplus b_1 \oplus b_0$. The probability that the k th output bit* will be a 1 is:

$$\text{Prob}(b'_k = 1) = \text{Prob}(b_k = 1) \cdot \quad [7]$$

Prob (even number of 1's in the first k bits) + Prob($b_k=0$) ·
 Prob(odd number of 1's in the first k bits).

* The first bit from the source is used to preset the flip-flop to its initial state and is not outputted. Thus the source will have outputted $k+1$ bits when the k th bit has been outputted at Pt. A.

$$= p[(1-p)^k + \binom{k}{2}(1-p)^{k-2}p^2 + \dots] + (1-p) [\binom{k}{1}(1-p)^{k-1}p + \dots]$$

↑
 Probability that k+1st bit from source is 1

↑
 probability of exactly two 1's in first k bits from source

etc.

↑
 probability of exactly one 1 in the first k bits from the source

↑
 probability that the source outputs all 0's in the first k bits

↑
 probability that the k+1st bit from the source is a 0

The above expression can be easily evaluated by constructing the appropriate linear combinations of the following two identities:

$$1 = [(1-p) + p]^k = (1-p)^k + \binom{k}{1}(1-p)^{k-1}p + \binom{k}{2}(1-p)^{k-2}p^2 + \dots$$

(Identity 1)

$$(1-2p)^k = [(1-p) - p]^k = (1-p)^k - \binom{k}{1}(1-p)^{k-1}p + \binom{k}{2}(1-p)^{k-2}p^2 - \dots$$

(Identity 2)

Thus

$$\text{Prob}(b'_k = 1) = \frac{p}{2} [1 + (1-2p)^k] + \frac{(1-p)}{2} [1 - (1-2p)^k] = \frac{1}{2} (1 - (1-2p)^{k+1}). \quad [8]$$

It is now evident that $\lim_{k \rightarrow \infty} \text{Prob}(b'_k = 1) = \frac{1}{2}$ ($0 < p < 1$).

This last result is not hard to derive if we consider the process as a simple two state birth and death Markov process and then solve the process for the steady-state probabilities. Let $\textcircled{0}$ and $\textcircled{1}$ represent the flip-flop's 0 and 1 states respectively. This results in the diagram of Figure 2.3 (5).

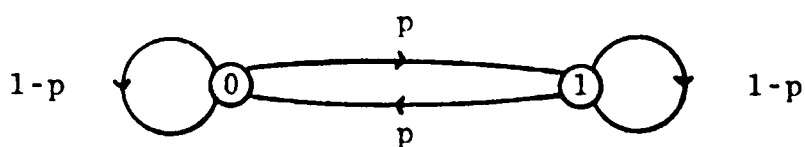


Figure 2.3 Two State Birth-Death Markov Process

The diagram in Figure 2.3 states that a 1 from the source, which occurs with probability p , will change the state of the flip-flop and a zero from the source, which occurs with probability $1-p$, will not change the flip-flop's state. If the process is stationary after it is started there will then be an equal number of traversals from the 0 state to the 1 state (births) as there are traversals from the 1 state to the 0 state (deaths). Hence we may write $P(0)b = p(1)d$ where $P(i)$ is the steady state probability of the process being in the i th state (i here is equal to 0 or 1), b is the "birth" rate, and d is the "death" rate.

Both the birth and death rates are here equal to p , hence $P(0)=P(1)$. The process must surely be in one of the possible states so $P(0)+P(1)=1$. These two equations give $P(0)=P(1)=\frac{1}{2}$. In other words, the steady state probability of the flip-flop's being in the 1 state is one-half, a result which was laboriously derived before by combinatorics.

We have then, by use of the Accumulative Autokey cipher, converted a source, whose average output of 1's was unequal to its average output of 0's, to another source (Pt. A) whose average output of 1's is equal to its average output of 0's. We have, in other words, used the Accumulative Autokey cipher to remove the first order bias of a binary source. It is important to realize that we have not increased the information rate of the source. A deterministic circuit, such as a flip-flop, is incapable of this. The output at Pt. A is still a Bernoulli source but with probability of flip-flop transition equal to p rather than probability of flip-flop state being 1 equal to p .

The Vigenere Cipher

The Vigenere cipher is perhaps the most important of all the ciphers for many ciphers can be thought of as a type of Vigenere cipher and a modern approach to cryptography depends upon the "Infinite Key Vigenere

cipher." Formally, the Vigenere cipher is a sequential addition (mod 26) of the clear text and a pre-arranged text. Analytically:

$C'_i = T_i \oplus P_i$ where C'_i is the ith encoded character.

T_i is the ith character of the clear.

P_i is the ith pre-arranged character.

As an example, consider the clear: DO UNTO OTHERS and the pre-arranged text: WORLD WITHOUT. The Vigenere cipher is then:

```

DO UNTO OTHERS
⊕ WO RLDW ITHOUT
ZC LYWK WMQSLK

```

Many times in the past the Vigenere cipher used a single repeating key for the pre-arranged text. This key might be a single word, name, phrase, or a series of nonsense words. Note that when the key consists of a single repeated letter, the cipher becomes the Caesar cipher with which we have previously dealt.

The Infinite Key Vigenere Cipher

The Infinite Key Vigenere cipher is, as its name implies, a Vigenere cipher with a non-repeating pre-arranged text or key. The cipher is of extreme importance and we will be concerned with it and the generation of its key in the following chapters.

Many non-cryptographic processes can be viewed as examples of this cipher. Take for instance the well known binary symmetric channel or BSC shown in Figure 2.4.

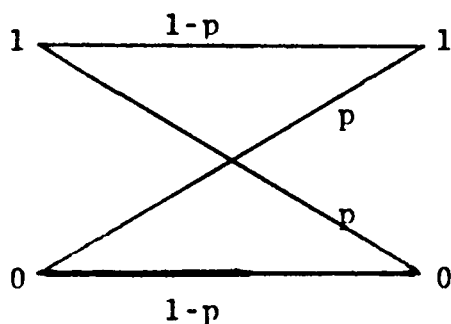


Figure 2.4 The Binary Symmetric Channel (BSC)

This model of a binary communication channel assumes that at any time a bit will be transmitted incorrectly (inverted) with probability p . We can view this process as an Infinite Key Vigenere cipher (with 0,1 and addition modulo 2) if we consider the clear as the message we are trying to send (also in binary) and the key as the output of a Bernoulli source with probability of a one output equal to p . Thus the BSC is equivalent to the device of Figure 2.5.

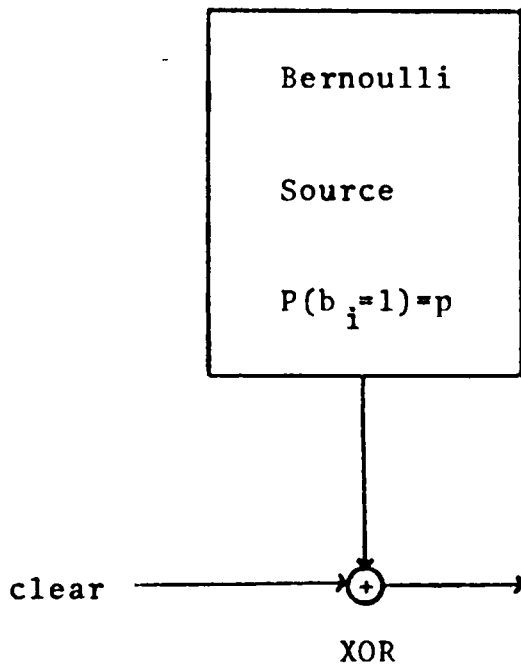


Figure 2.5 An Equivalent Representation of the BSC

CHAPTER III

A STATISTICAL LOOK AT CLASSICAL CRYPTOGRAPHY

Much of a written language and the information transmitted via the use of the language is dependent on patterns of symbols rather than upon the symbols themselves. We can learn to read English text without ever learning the English alphabet. All that needs to be taught are the mappings of the set of arrangements or patterns of distinguishable symbols onto the set of objects, relationships, actions, emotions, and other entities of the universe. When we look at the tri-symbol concatenation DOG, we do not examine each letter sequentially (Figure 3.1) and arrive at one of the 26^3 possible objects that

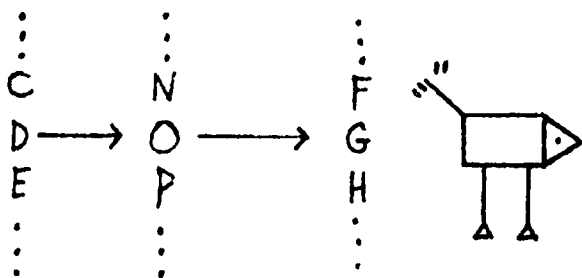


Figure 3.1 A Tri-Symbol Concatenation

three letters can specify. Rather we note the form of the word and relate it to previous experience or teaching. If we had been so trained, ⚡!♥ could also be DOG but not ⚡♥♥ for here the last two characters are not distinct and the pattern is altered.* The written language is then a set of patterns or complexions of symbols. These patterns are not usually sequentially independent for the language is very highly coupled. This is reasonable if one considers that actions in life are related. The pattern structures of the symbols representing these actions should then also be coupled.

Consider the assignment of normal binary code to the English alphabet and some special characters as shown in Table 3.1. The following 40 bit string represents the encoding of the word RAILROAD:

1000100000010000101110001011100000000011. If the channel through which the message is sent were to invert the 31st bit: 1000100000010000101110001011101000000011 the word received would be RAILROQD. Because English itself is a highly error-correcting code in its own right, we would believe that the intended word was RAILROAD as this word has the shortest 'distance' of any word in the language to RAILROQD which is certainly not in the language. We can think of this in another way; given the partial sequence

* We are assuming English form but not English characters.

Table 3.1
Binary Code

A	00000	Q	10000
B	00001	R	10001
C	00010	S	10010
D	00011	T	10011
E	00100	U	10100
F	00101	V	10101
G	00110	W	10110
H	00111	X	10111
I	01000	Y	11000
J	01001	Z	11001
K	01010	,	11010
L	01011	.	11011
M	01100	(space)	11100
N	01101	(11101
O	01110)	11110
P	01111	*	11111

Table 3.2
Frequency of Letter Occurrence in Normal English

Letter	Frequency	Letter	Frequency
A	0.0786	N	0.0728
B	0.0161	O	0.0766
C	0.0387	P	0.0210
D	0.0352	Q	0.0013
E	0.1256	R	0.0599
F	0.0238	S	0.0665
G	0.0163	T	0.0970
H	0.0522	U	0.0259
I	0.0779	V	0.0103
J	0.0008	W	0.0164
K	0.0043	X	0.0018
L	0.0399	Y	0.0161
M	0.0243	Z	0.0008

"crypt." By calculation, $H_1 = 0.8817$ crypts. It is evident that H_1 is unchanged by the simple substitution cipher, for $H_1 = H_1(f_1, f_2, \dots, f_{26}) = H_1(f_{\alpha_1}, f_{\alpha_2}, \dots, f_{\alpha_{26}})$ where $(\alpha_1, \alpha_2, \dots, \alpha_{26})$ is any permutation of $(1, 2, \dots, 26)$. If a cipher process increases H_1 , then the cipher reduces the bias in frequency of letter occurrence. It is intuitive that a cipher's worth (the resistance of the cipher to cracking procedures) should be related to H_1 . In general, the closer H_1 is to 1, the better the cipher. This rule of thumb is not always true as we shall see later. In theory, one should examine all H_i where i spans the natural numbers. H_2 , for instance, would be the entropy of the bigraph distribution (the distribution of AA, AB, ..., TH, ...).

H_1 may be found for the Vigenere cipher (discussed in Chapter II) in two ways. The first method is to actually encipher random passages of English text and then compute the f_i table and calculate H_1 . Another method however is to consider that the Vigenere cipher is formed by the equation $C_i' = T_i \oplus P_i$ where C_i' is the i th encoded character, T_i is the i th character of the clear, and P_i is the i th pre-arranged character. The pre-arranged text, as mentioned before, may be either a repeating word or phrase or non-repeating text (the infinite key Vigenere

cipher). It is the latter case that will be analyzed first as it is the most interesting and once analyzed the former case becomes obvious.

We shall assume, then, that the pre-arranged or ciphering text is English text and is chosen independently of the text to be enciphered. Using these assumptions it is clear that on the average the i th letter of the clear text has a probability f_1 of being paired with an A in the ciphering text, a probability f_2 of being paired with a B, etc. If the usual assignment is made $\{0, 1, 2, \dots, 24, 25\} \leftrightarrow \{A, B, C, \dots, Y, Z\}$, then the probability f_1' that the i th ciphered character will be an A is the sum of the probabilities that the i th clear text letter and the i th ciphering text letter will sum to 0. Thus $f_1' = P(A)P(A) + P(B)P(Z) + P(C)P(Y) + \dots + P(Y)P(C) + P(Z)P(B)$.^{*} The probability of a B, f_2' , in the ciphered text is then $f_2' = P(B)P(A) + P(C)P(Z) + P(D)P(Y) + \dots + P(Y)P(D) + P(Z)P(C)$ and so on. Thus by knowing only the frequency distribution of letters in normal English. We can derive the frequency distribution for the Vigenere infinite key cipher. The results predicted by this method are shown in Table 3.3 along with results obtained by ciphering 5000 letters from randomly selected texts. The theoretical H_1 is calculated to be 0.9948 crypts. The H_1 derived from the actual

^{*}P stands for "probability of."

Table 3.3

Single Letter Frequency Distribution for the Vigenere
Infinite Key Cipher

Letter	Theoretical Frequency	Empirical Frequency
A	0.0443	0.0394
B	0.0360	0.0344
C	0.0307	0.0260
D	0.0236	0.0270
E	0.0463	0.0458
F	0.0383	0.0376
G	0.0459	0.0442
H	0.0430	0.0436
I	0.0470	0.0472
J	0.0285	0.0320
K	0.0352	0.0396
L	0.0452	0.0446
M	0.0446	0.0452
N	0.0304	0.0268
O	0.0306	0.0290
P	0.0374	0.0382
Q	0.0297	0.0312
R	0.0415	0.0430
S	0.0436	0.0466
T	0.0426	0.0446
U	0.0324	0.0294
V	0.0502	0.0512
W	0.0477	0.0480
X	0.0390	0.0394
Y	0.0314	0.0290
Z	0.0350	0.0370

ciphering of 5000 letters of text is 0.9941 crypts. Note that H_1 for the infinite key Vigenere cipher is much nearer 1.0000 than H_1 for the substitution cipher. This is an indication that the former is a more difficult cipher to break which is most certainly true.

It is significant that the infinite key Vigenere cipher can indeed be broken and one can obtain both the clear and ciphering texts. This was effectively demonstrated in 1917 and in effect says that because of the great redundancy in the English language one can achieve a 2:1 compression of text. This was first noticed by Shannon (4), (6).

Let us now consider the case of a Vigenere cipher whose key is a repeating word. We shall further assume that all of the letters of the key word are distinct. Cases different from this are easily handled with a slight and obvious extension of the theory. What happens in the repeating word Vigenere is best illustrated by an example. Let us encipher a passage of English text using the repeating key word SWAN. By definition of the cipher, S will be added (modulo 26) to the first, fifth, ninth, etc. letters of the text to be enciphered. W will be added to the second, sixth, tenth, etc. letters of the text. A will be added to all letters which are in positions $3 \bmod 4$.

N to letters $O \bmod 4$. The effect of this on H_1 is easily seen if we consider that the ciphering process is in reality a combination of four Caesar cipherings. The probability density function (PDF) for letter frequency (choosing a letter at random) of normal English looks something like that of Figure 3.2. A Caesar cipher merely

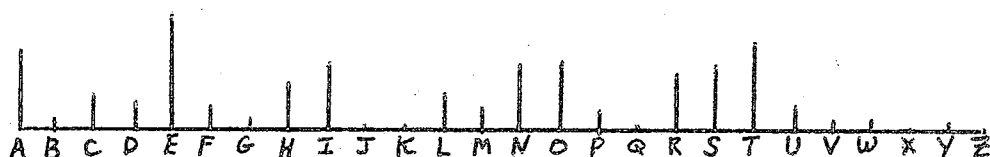


Figure 3.2 PDF for Normal English

cycles this PDF. S , for instance, cycles the distribution to that of Figure 3.3. A , the identity element in the

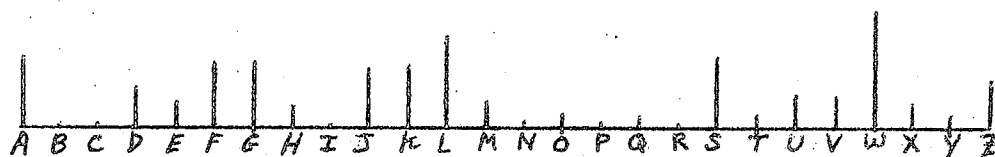


Figure 3.3 Cycled PDF

additive group, leaves Figure 3.2 as it is. W and N perform different cyclings. The result is that the PDF

of the enciphered text is obtained by adding $\frac{1}{4}$ of the PDF for S to $\frac{1}{4}$ of the PDF for W to $\frac{1}{4}$ of the PDF for A to $\frac{1}{4}$ of the PDF for N. Thus if $[f'_1, f'_2, \dots, f'_{26}]$ is the PDF matrix of the frequencies of the enciphered letters and if

$$\begin{bmatrix} f_1^S & f_2^S & \dots & f_{26}^S \\ f_1^W & f_2^W & \dots & f_{26}^W \\ f_1^A & f_2^A & \dots & f_{26}^A \\ f_1^N & f_2^N & \dots & f_{26}^N \end{bmatrix}$$

is the matrix which has as its rows the PDF's for letter

frequencies which arise from a cycling of the ordinary English frequencies by an amount equal to the superscript letter, then

$$[f'_1 \ f'_2 \ \dots \ f'_{26}] = [\alpha_1 \alpha_2 \alpha_3 \alpha_4] \begin{bmatrix} f_1^S & f_2^S & \dots & f_{26}^S \\ f_1^W & f_2^W & \dots & f_{26}^W \\ f_1^A & f_2^A & \dots & f_{26}^A \\ f_1^N & f_2^N & \dots & f_{26}^N \end{bmatrix}$$

where $[\alpha_1 \alpha_2 \alpha_3 \alpha_4]$ is the singly stochastic weighting matrix

($\sum_{i=1}^4 \alpha_i = 1$) which in this case is $[\frac{1}{4} \ \frac{1}{4} \ \frac{1}{4} \ \frac{1}{4}]$ as mentioned

before. It is clear that $H_1(f'_1 f'_2 \dots f'_{26}) > H_1(f_1 f_2 \dots f_{26})$ because the α_i 's "smooth" the distributions upon which they act. In fact, if we were to have used the entire alphabet instead of just the four letters SWAN, H_1 would go to 1.0000. This is one of the cases referred to previously in that H_1 for the infinite key Vigenere cipher is less than H_1 for the Vigenere cipher using the entire alphabet as key however the former cipher is much more difficult to break than the latter. The irregularity in this case arises from our not considering H_2, H_3 , etc. Returning to our example, we find that upon ciphering 1500 letters of text using the codeword SWAN, we obtain $H_1 = 0.9642$ crypts. Upon direct calculation of H_1 , as indicated above, we obtain $H_1(\text{theoretical}) = 0.9726$ crypts.

It is relatively simple to break the Vigenere cipher when a simple word or phrase has been used as a key word. Consider the following passage from Darwin's writings.

THE CONTROVERSY RESPECTING THE NATURE AND
 THE EXTENT OF THE DIFFERENCES IN THE
 STRUCTURE OF THE BRAIN OF MAN AND THE APES
 WHICH AROSE SOME FIFTEEN YEARS AGO HAS NOT
 YET COME TO AN END THOUGH THE SUBJECT MATTER
 OF THE DISPUTE IS AT PRESENT TOTALLY
 DIFFERENT FROM WHAT IT WAS FORMERLY

Upon enciphering this passage with codeword SWAN we obtain the following.*

(LDE) PGJTEGREEKU RRKLEPLENT (LDE) ASPUEW WNQ
 (LDE) RPPEAL KF [GZA] DVXBEEWJCRK EN [GZA]
 SGJQCGMNE BX PHR TNAVF EN ZSJ AAV PHR
 SLEF ODIPZ WRBKA SBEA FVXPERF UENJO ATG DAF
 FKT LWP CBEA TB SJ EAV PHBMCH GZA SHTFEPL
 IAGLAR BX PHR VESCMPE VK WT CJASRFP TBLWLYQ
 ZISXARRFP FEGI WUSP IG OWS SGNMRJHY

Notice that the three letter words LDE and GZA occur more than once. It is a good guess to suppose that they are the same words in clear. If this is indeed the case, then the distance from one LDE to another LDE might well be a multiple of the length of the ciphers key word or phrase. If then we take the greatest common divisor of all these differences it is a good bet that that number is the length of the cipher's key word or phrase. The distance from the first circled LDE to the second circled LDE is 24 letters. The distance from the second circled LDE to the third is 12 letters. The distance from the first boxed in GZA to the second is 16 letters. The distance from the second to the third is 92 letters. The greatest common divisor of 24,12,16,92 is 4 which is the length of our codeword SWAN.

* Spaces were inserted only to aid the reader. The statistics of spacing have not been computed.

Another method of determining the length of the cipher's key shows the great sensitivity of the entropy function H . If H is computed on any set of letters whose positions are k (k any integer) modulo the length of the key, then H will approach H_1 for regular English. When H is computed on such a set we should notice a dip in the entropy as $H_1(\text{English}) < H_1(\text{Vigenere})$. Let us denote the sample of letters from the positions $1, 1+k, 1+2k, 1+3k, \dots$ as lag k . H_1 for the different lags 1 through 60 was computed on the 1500 letters of ciphered text given above. The results are presented in the graph of Figure 3.4. The dip in entropy mod 4 shows that the key probably has a length of 4 which is correct. Note that the dip appears even in the high lags of 56 and 60. The test's great sensitivity becomes apparent when we consider that at lag 60, the sample size is a meagre 25 letters.

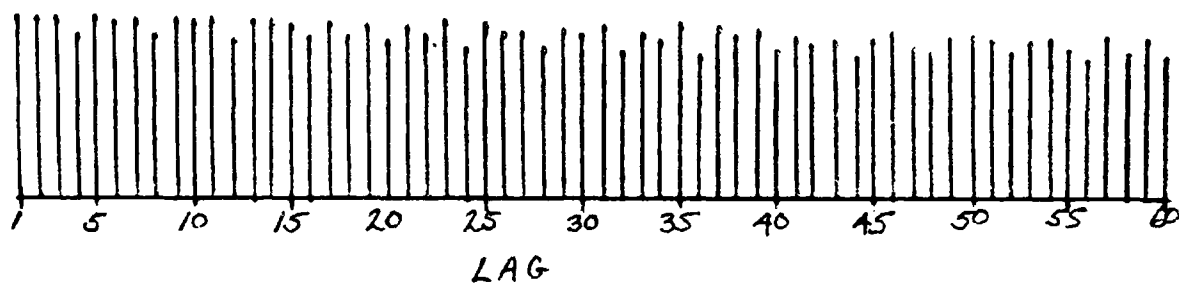


Figure 3.4 H_1 Computed on Different Lags of the Vigenere

The Autokey cipher can also be examined through its different entropy signatures. The rule for the Autokey of lag k (Chapter II) is $C'_1 = C_i \oplus C_{i-k}$ where C'_i is the i th encoded character, C_i is the i th character of the clear, and $(C_{1-k}, C_{2-k}, \dots, C_{-1}, C_0)$ are k pre-arranged characters. Let us consider the Autokey of lag 1. From the very nature of the cipher we would expect that the entropy would be strongly dependent upon the bigram distribution in English. Consider the Autokey of lag 1 with pre-arranged character X shown in Figure 3.5. Because TH is a most common bigram in English, we would

THE CAR IS GREEN	(clear)
<u>⊕ XTH ECA TI SGREE</u>	
QAL GCT BA YXVIR	(enciphered text)

Figure 3.5 Example of Autokey With Lag 1

expect a comparatively large number of A's in the enciphered text as H added to T yields A under mod26 addition with the conventional assignment given many times

before. In the limit then, we expect that

$$P'(A) = P(AA) + P(BZ) + P(CY) + \dots + P(YC) + P(ZB)$$

$$P'(B) = P(AB) + P(BA) + P(CZ) + \dots + P(YD) + P(ZC)$$

.
.
.

$$P'(Z) = P(AZ) + P(BY) + P(CX) + \dots + P(YB) + P(ZA)$$

where $P'(A)$ is the probability or frequency of A's in the enciphered text and $P(AA)$ is the probability or frequency of AA in the English language. From examining over 50,000 letters of English text a bigram count was obtained. Upon enciphering 5000 letters of text, experimental letter frequencies were derived and H_1 computed. These figures are presented in Table 3.4 along with the theoretical calculations via the equations just presented operating on the bigram table. As can be seen, the results agree quite well.

Table 3.4
 Single Letter Frequency Distribution for the
 Autokey Lag 1 Cipher

Letter	Theoretical Frequency	Empirical Frequency
A	0.0702	0.0806
B	0.0568	0.0416
C	0.0160	0.0124
D	0.0255	0.0342
E	0.0232	0.0280
F	0.0415	0.0414
G	0.0511	0.0498
H	0.0495	0.0460
I	0.0182	0.0312
J	0.0296	0.0250
K	0.0365	0.0502
L	0.0686	0.0786
M	0.0327	0.0406
N	0.0385	0.0390
O	0.0132	0.0118
P	0.0358	0.0328
Q	0.0346	0.0266
S	0.0222	0.0282
T	0.0656	0.0518
U	0.0176	0.0188
V	0.0784	0.0674
W	0.0526	0.0466
X	0.0291	0.0266
Y	0.0162	0.0158
Z	0.0350	0.0326
ENTROPY (CRYPTS)	0.9667	0.9690

CHAPTER IV
A PROCEDURE FOR FIRST ANALYSIS OF COMPLETELY
DETERMINISTIC CRYPTOGRAPHIC MACHINES

There are essentially two types of analysis that must be performed on every proposed cryptographic device. The first or type I is: Will the machine basically work? Is it possible to transmit information using it? The second or type II is: Does the machine sufficiently distort the statistics of the information transmitted, and what are its overall vulnerabilities? This chapter deals with only the first question and does so by way of an example (7).

Methodology

The general procedure for the type I analysis entails three steps. For every new piece of cryptographic gear the analyst must

- 1) develop a switching notation,
- 2) prove a one-to-one mapping between input and output,
- 3) find the decoding algorithm.

Step 1 is necessary in order to carry out steps 2 and 3 in an efficient manner. It is also often useful in gaining insight into the secrecy mechanism in an information-theoretical sense. Step 2 is performed to show uniqueness

of deciphering (the decipherability criterion). It ensures that the machine is of the non-computational or constant entropy type. (In a strict sense there are some machines that require only a many-to-one mapping to be proved, but we will not be concerned with them because they require only a trivial extension of the procedure.) Step 3 is of extreme importance, for if the machine has no easily implementable algorithm for deciphering what it has ciphered, it is next to useless.

Example of the Above Procedure

We now present an application of the above rules to a hypothetical binary cryptographic device. Let us define the switching element s shown in Figure 4.1. The element has two states and operates as follows:

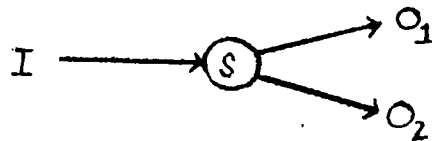


Figure 4.1 Basic Switching Element

a bit enters s along line I and leaves uncomplemented along line O_1 if s is in the \uparrow or 0 state. The bit leaves complemented along O_2 if s is in the \downarrow or 1 state.

s changes state after passing the bit if and only if the bit entering on line I is a 1. Our cryptographic device is constructed using these elements in the octopus arrangement shown in Figure 4.2. For $n=2$, and for the initial states of s_{11}, s_{12}, s_{22} all \uparrow ,

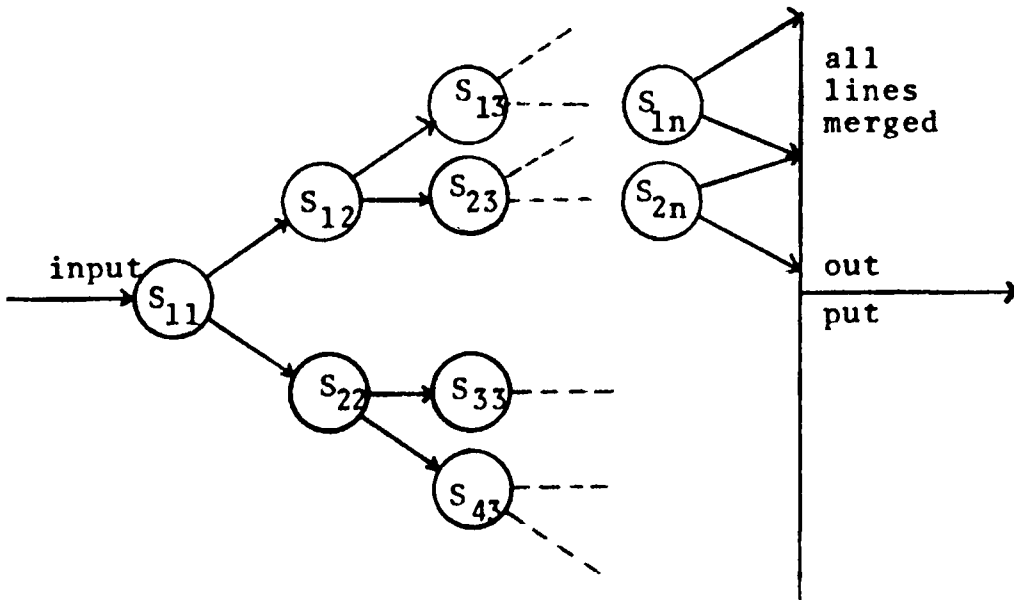


Figure 4.2 Compound (octopus) Machine

it is easy to verify the following enciphering:

$$\begin{array}{ccccccc}
 \text{---}b_3 & b_2 & b_1 & \longrightarrow \text{machine} \longrightarrow & \text{---}b'_3 & b'_2 & b'_1 \\
 0 & 1 & 1 & & 1 & 0 & 1
 \end{array}$$

(b' indicates enciphered bit).

Development of the Switching Notation

We denote the k th state of the (n,r) element by ϕ_{nr}^k . If the $(k+1)$ st processed or input bit does not pass through s_{nr} in its path through the machine, then

$\phi_{nr}^{k+1} = \phi_{nr}^k$. However, if the bit should pass through s_{nr} , then the next state is $\phi_{nr}^{k+1} = \phi_{nr}^k \oplus b_e$, where b_e

is the bit entering s_{nr} and \oplus is the exclusive-or Boolean function. If b_ℓ is the bit leaving s_{nr} on one of its branches, then it is also clear that $b_\ell = \phi_{nr}^k \oplus b_e$.

In this example the switching notation is relatively simple. In general this is not the case.

Uniqueness

We must now show that there is a one-to-one correspondence of the set of all possible input strings to the set of all possible output strings.

Proof: Consider two identical machines of the type just described. Let us call them S and S^* . Let S and S^* have the same initial states. Assume that we apply two distinct input bit sequences $F(b_1, b_2, b_3, \dots)$ and $F^*(b_1^*, b_2^*, b_3^*, \dots)$ to S and S^* respectively. Because the sequences are distinct, they will differ in a non-zero number of bits. Let us assume that the first bit in which F and F^* differ is bit i . Clearly the machines will have identical outputs and states up to bit i . Bits b_i and b_i^* of F and F^*

will follow identical paths through both S and S^* , for the switching of the individual s and s^* elements of S and S^* is performed after the bit has passed through the element. Thus the i th output of S is $b_i \oplus [\vartheta_{11}^{i-1} \oplus \dots]$ where $[\vartheta_{11}^{i-1} \oplus \dots]$ is the XOR sum of states through which b_i passes. The i th output of S^* is $b_i^* \oplus [{}^*\vartheta_{11}^{i-1} \oplus \dots]$. But $[\vartheta_{11}^{i-1} \oplus \dots] = [{}^*\vartheta_{11}^{i-1} \oplus \dots]$ and $b_i^* = \overline{b_i}$; hence the i th encoded bits will differ. The one-to-one mapping between input and output lists is now clearly evident.

It is interesting to note also an apparent dualism between states and bits. Consider again the two machines S and S^* . It is impossible for S and S^* to attain identical states if they process the same binary string, but differ in one or more initial states. The proof is quite simple. Let us assume that S and S^* have different initial states and that they can attain identical states after processing the same binary sequence. Consider ϑ_{11} and ϑ_{11}^* . If these states differ initially, it is clear that they will always differ under the same binary sequence. Hence, in order to fulfill our premise, we must choose $\vartheta_{11}^0 = \vartheta_{11}^{*0}$ and have other initial states differ. However, if we choose $\vartheta_{11}^0 = \vartheta_{11}^{*0}$, we can now extend our argument to ϑ_{12}^0 and ϑ_{22}^0 . By continued reduction, we exhaust the machine's states and contradict the premise.

Decipherability

Once we have a coded sequence $(b'_1, b'_2, b'_3, \dots)$ we need only know the machine's initial states $(\theta_{11}^0, \theta_{12}^0, \theta_{22}^0, \dots)$ in order to decode the sequence. The first bit encoded will be $b'_1 = b_1 \oplus [\theta_{11}^0 \oplus \dots]$, where $[\theta_{11}^0 \oplus \dots]$ is the XOR sum of states through which the uncoded bit passes to become the coded bit. If we then process b'_1 by sending it through the machine set to its initial states, we shall recover b_1 ; for $b_1 = b_1 \oplus [\theta_{11}^0 \oplus \dots] \oplus [\theta_{11}^0 \oplus \dots]$. Now if $b_1 = b'_1$, we may go ahead and recover b_2 by sending b'_2 through the machine; for the machine's states will now be the same as they were when b_2 was encoded. However, if $b_1 = \overline{b'_1}$, we must reset the machine to its state immediately preceding the processing of b_1 --in this case $(\theta_{11}^0, \theta_{12}^0, \theta_{22}^0, \dots)$ --and run b_1 through before processing b'_2 . In general then, we process or decode bit b'_i . If the processed bit b_i is equal to b'_i , we continue on and process b'_{i+1} . However, if $b_i = \overline{b'_i}$, we must reset the machine to its state preceding the processing of b'_i , namely $(\theta_{11}^{i-1}, \theta_{12}^{i-1}, \theta_{22}^{i-1}, \dots)$, and update the states to the correct states $(\theta_{11}^i, \theta_{12}^i, \theta_{22}^i, \dots)$ by running through $\overline{b'_i}$. We are then ready to process b'_{i+1} .

The above algorithm is easily implemented, and the machine has passed the type I analysis. The type II analysis may now be performed.

CHAPTER V
TRANSMISSION CHANNEL CONSIDERATIONS
AND PERFECT CRYPTOGRAPHY

In Chapter I, under the decipherability criterion, it was mentioned that a cryptographic machine is essentially a sequential machine which performs the one-to-one mapping φ on the set of possible unencoded or clear symbols onto itself. It was also mentioned that φ may or may not be dependent upon the clear. In the case of the simple substitution cipher, φ is independent of the clear. In the case of the Autokey cipher, however, φ is heavily dependent on the clear. φ 's dependence or independence upon the clear text has important consequences if we are dealing with a noisy transmission channel. If, for instance, the 5th character of the encoded message is distorted in transmission, then this error will affect only the 5th decoded character at the receiver if the cipher employed is that of simple substitution. If however the cipher used is the Autokey of lag 1, then the error will affect not only the 5th decoded character but all characters following the 5th. Consider the message: THE CAT IS GREEN with pre-arranged character X. Let the transmission

channel change the 5th encoded character C to D. Figure 5.1 shows the effects. It is clear then that if ϕ is

THE CAT IS GREEN	(clear)
<u>⊗ XTH ECA TI SGREE</u>	
QAL GCT BA YXVIR	(enciphered text)
	(channel-distorts 5th symbol)
QAL GDT BA YXVIR	(received message)
<u>XTH ECB SJ RHQFD</u>	
THE CBS JR HQFDO	(deciphered message)

Figure 5.1 The Propagation of Error in the Autokey Cipher

text dependent then the crypto decipherer will be plagued with the annoying problem of tedious and perhaps difficult corrections necessitated by a noisy channel. For instance, let us assume that our crypto alphabet consists of the characters of Table 3.1. If the channel distorts symbols with a probability $p=0.1$ then the operator will have to perform an average of one correction for every ten characters deciphered. A vivid way of demonstrating this is to simulate the transmission of four rows of 40 blanks. If the channel were perfect ($p=0.0$) there would be no distortions and we would receive 160 blank characters.

whose φ is independent of everything. This cipher is so exceedingly trivial to break that it is next to useless in modern cryptography. \mathcal{P} does not have to be constant, however, to be text independent. Consider $\mathcal{P} = \varphi(i)$ where i is the number of the i th encoded/decoded character. If the sender and the various subscribers can be synchronized (system acquisition) and if $\mathcal{P} = \varphi(i)$, then it is clear that a channel error will affect only a single character.

An example of a system whose $\mathcal{P} = \varphi(i)$ and the nature of "perfect cryptography."

Consider the following case suggested by Golomb. A shift register of length 15 is used in the manner shown in Figure 5.3. The clear is represented in

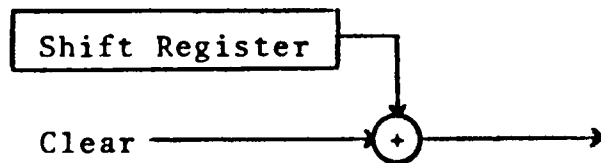


Figure 5.3 Cipher System Employing a Shift Register

5 bit/symbol binary code according to Table 3.1. Each successive bit of clear is XORed to each successive output bit from the shift register.

The shift register behaves as a Bernoulli source with probability of a 1 output equal to $\frac{1}{2}$. It is then clear that any one of the 32 possible characters is equally probable and appears statistically independent of any outputs preceding it. The PDF for the 5 bit output of the shift register is shown in Figure 5.4. Because

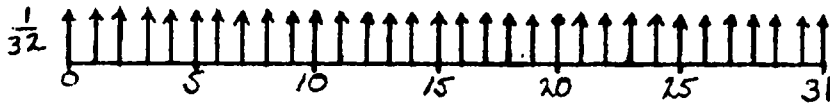


Figure 5.4 The PDF for the Shift Register Output

of the nature of the PDF of the 5 bit samples from the shift register, the PDF of the enciphered text must be identical to the PDF shown in Figure 5.4 no matter what the PDF of the material to be enciphered. Therefore, the enemy can tell nothing about what was sent as the a-posteriori probabilities are all equal. This type of situation is known as "perfect cryptography" (8).

CHAPTER VI

VARIABILITY

Introductory Remarks

The shift register example of Chapter V, while feasible, would not be a very good crypto system for it suffers from a lack of variability of parameters. The only parameter that can be changed from message to message is the initial complexion of the shift register. As the shift register is of length 15 bits, once any three consecutive letters (15 bits) of the clear are compromised or guessed (such as the word THE) the whole message becomes compromised.

In order to add greater variability to crypto processes, often one may add sequence distorters or "scramblers" at strategic points of the basic machine. These distortions must usually be performed in a manner that is reversible. This chapter concerns itself with just one such type of machine as an example (9).

The sequential circuits discussed in this chapter will be assumed to be machines with fixed initial states. The circuits are in their initial states at the beginning of the binary input strings in all of the following discussions.

Definition 1. An instantaneous one-to-one binary sequential machine [IOOBSM] is a binary device with one input line and one output line that outputs one bit for every input bit subject to the uniqueness condition that if $\vec{b} (=b_k b_{k-1}, \dots, b_2 b_1)$ and \vec{b}^* represent any two input strings and \vec{b}^1 and \vec{b}^{1*} represent the two corresponding output strings then if \vec{b} and \vec{b}^* are processed by identical machines $\vec{b}^1 = \vec{b}^{1*}$ if and only if $\vec{b} = \vec{b}^*$.

Lemma 1: If M_a and M_b are two IOOBSMs, then their series concatenation (see Figure 6.1) is also an IOOBSM.



Figure 6.1 Series Concatenation of Two IOOBSM's

Proof: Consider two different binary strings \vec{b} and \vec{b}^* applied to two identical M_a - M_b concatenations. Because the strings are different they will differ in a non-zero number of bits. Let i be the number of the first bit which is different. After the strings have been processed by M_a , we have the strings \vec{b}^1 and \vec{b}^{1*} which are fed into M_b to produce the final strings \vec{b}^{11} and \vec{b}^{11*} . Because $b_i \neq b_i^*$, $b_i^1 \neq b_i^{1*}$ and so $b_i^{11} \neq b_i^{11*}$ thus proving the lemma.

Let r IOOBSM's be arranged in parallel as shown in Figure 6.2. The bit distribution follows an arbitrary prearranged pattern, which is independent of the incoming bit string. The merging of bits at the output follows the same pattern as the input distribution. It can be shown that the device in Figure 6.2 is also an IOOBSM.

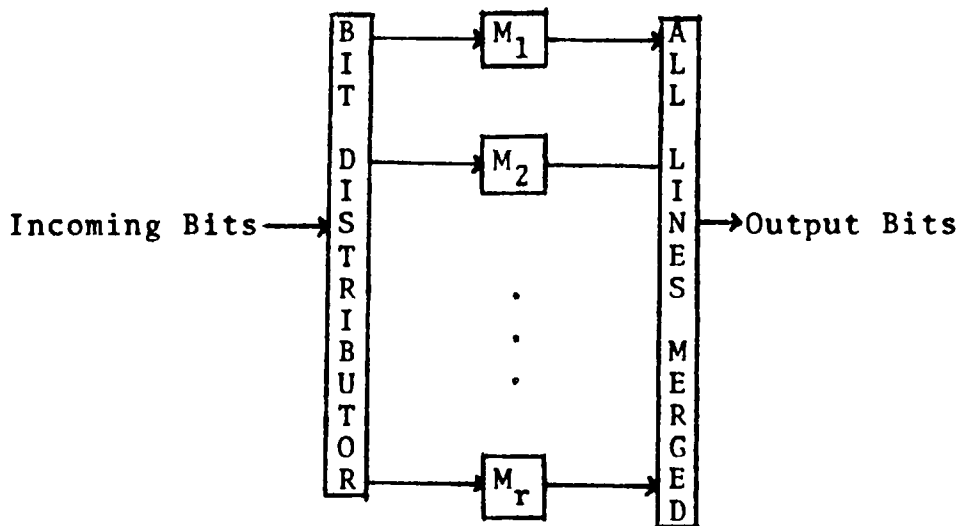


Figure 6.2 Parallel IOOBSM Structure

Conditional IOOBSM's

Definition 2: A conditional s -bit instantaneous one-to-one binary sequential machine [CIOOBSM(s)] behaves as an IOOBSM for at least the first s bits.

All CIOOBSM(s)s are then a subset of the $(2^s)!$ one-to-one mappings of the form shown in Figure 6.3.

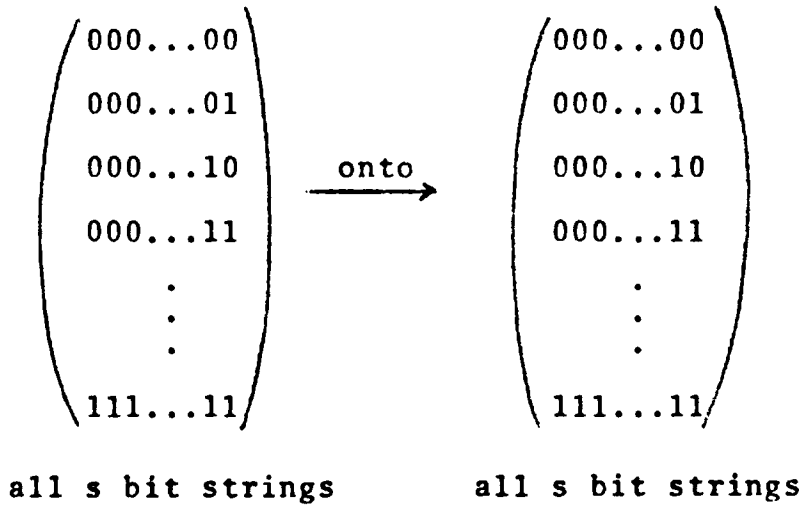


Figure 6.3 One-to-one Mapping of s Bit Strings

Theorem 1: There are $2^{(2^s-1)}$ CIOOBSM(s)s.

Proof: a) Basis. There are two CIOOBSM(1)s. They are

$$\begin{pmatrix} 0 \\ 1 \end{pmatrix} \longrightarrow \begin{pmatrix} 0 \\ 1 \end{pmatrix} \quad \text{identify}$$

and

$$\begin{pmatrix} 0 \\ 1 \end{pmatrix} \longrightarrow \begin{pmatrix} 1 \\ 0 \end{pmatrix} \quad \text{complement.}$$

b) $2^{(2^{s-1})}$ CIOOBSM(s)s may be formed out of a CIOOBSM($s-1$). This is so because each string of length ($s-1$) in the CIOOBSM($s-1$) appears twice in the CIOOBSM(s) post fixed in one case by a 0 and in the other by a 1.

To make this clearer, consider the string $b_{s-1}b_{s-2}\dots b_2b_1$

and the corresponding processed string $b'_{s-1}b'_{s-2}\dots b'_2b'_1$. The extension to strings of length $(s+1)$ bits may be accomplished by either of the two assignments:

$$0b_{s-1}b_{s-2}\dots b_2b_1 \longrightarrow 0b'_{s-1}b'_{s-2}\dots b'_2b'_1 \quad [1]$$

or

$$0b_{s-1}b_{s-2}\dots b_2b_1 \longrightarrow 1b'_{s-1}b'_{s-2}\dots b'_2b'_1 \quad [2]$$

The mapping of a zero input is, of course, just opposite the mapping of a 1 input. As there are 2^{s-1} strings of length $s-1$ there are 2^{s-1} such assignments, which must be made. Thus there are $2^{(2^{s-1})}$ possible mappings of output bit s as a function of the last s inputs. The choice of mapping of the s bit is independent of the mappings of previous bits so the total number of mappings of the first s bits (the number of CIOBSM(s)'s) is given by

$$\begin{aligned} 2 \cdot 2^2 \cdot 2^{(2^2)} \cdot 2^{(2^3)} \cdot \dots \cdot 2^{(2^{s-1})} &= 2^{1+2+4\dots 2^{s-1}} \\ &= 2^{(2^s-1)} \quad [3] \end{aligned}$$

It will be convenient to use the notation of permutation groups to identify particular CIOBSM(s)'s throughout the remainder of the chapter. Let the rows of the matrices in Figure 6.3 be numbered as shown in Figure 6.4. Then we may designate a CIOBSM(s), which implements a mapping of the form of Figure 6.4 by some

$M = \begin{pmatrix} 1 & 2 & 3 & \dots & 2^s \\ \alpha_1 & \alpha_2 & \alpha_3 & \dots & \alpha_{2^s} \end{pmatrix}$ where the α_i 's are a permutation

of the elements of the top row. In particular a circuit

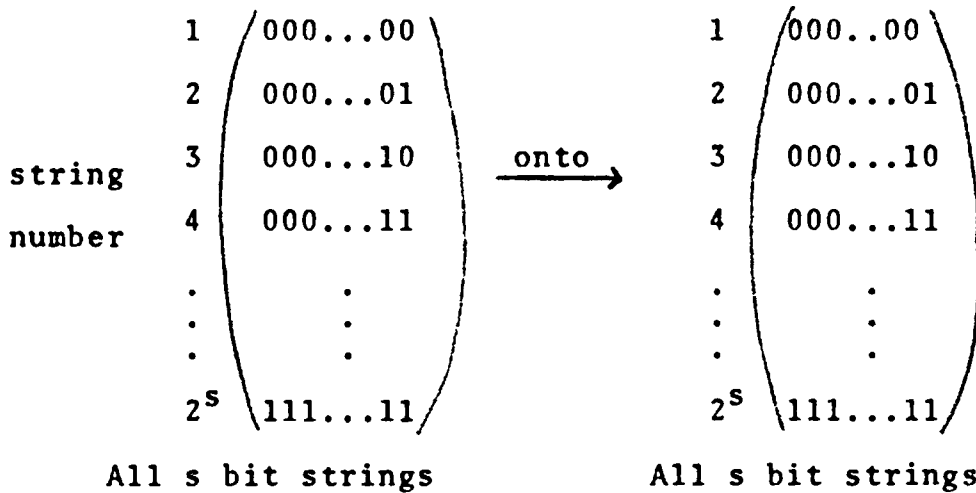


Figure 6.4 One-to-one Mapping of s Bit Strings

which carries out the identity mapping of all strings onto themselves may be designated by

$$I = \begin{pmatrix} 1 & 2 & 3 & \dots & 2^s \\ 1 & 2 & 3 & \dots & 2^s \end{pmatrix} \quad [4]$$

Several elementary theorems on permutation groups will be stated without proof. To distinguish these from the theorems, which are unique contributions of this chapter, we shall refer to the theorems of group theory as propositions. The group operation in permutation

groups is merely the successive application of two mappings, denoted $M_1 M_2$. The successive application of M r times, is denoted by M^r .

Proposition 1. Let M be a permutation of 2^s binary strings. Then there exists an integer r such that $M^r = I$. The smallest such r is called the order of the permutation M .

It is, thus, impossible to distinguish the concatenation in Figure 6.5, implementing M^r , from a straight wire for strings of s or fewer bits.

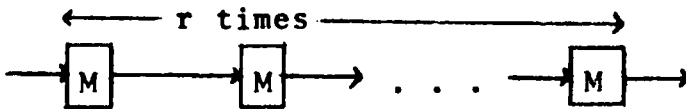


Figure 6.5 M^r Concatenation

Example 1

Consider the simple machine, t , shown in Figure 6.6. A bit enters t on line x and leaves t on line y uncomplemented if t is in the \uparrow state and complemented if t is in the \downarrow state. The machine, changes state after passing the bit if and only if the bit entering on line x is a 1. The circuit of Figure 6.6 can be shown to be an IOBBSM. A concatenation of these elements is then also an IOBBSM.

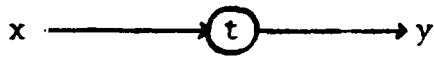


Figure 6.6 The t Machine

If we choose t in the initial state \downarrow as our machine M , then it is easy to verify the following mapping for strings of length 2.

$$\begin{array}{l} 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{pmatrix} 00 \\ 01 \\ 10 \\ 11 \end{pmatrix} \longrightarrow \begin{array}{l} 4 \\ 1 \\ 2 \\ 3 \end{array} \begin{pmatrix} 11 \\ 00 \\ 01 \\ 10 \end{pmatrix}$$

In symmetric group notation: $M = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 4 & 1 & 2 & 3 \end{pmatrix}$. If these machines are cascaded we have $M^2 = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 3 & 4 & 1 \end{pmatrix}$,

$$M^3 = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 3 & 4 & 1 \end{pmatrix}, \quad M^4 = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 1 & 2 & 3 & 4 \end{pmatrix}. \quad \text{Thus the}$$

order of M is 4. Alternately, in cyclic notation, we may write $M = (1432)$. Therefore

$$M^2 = (13)(24), \quad M^3 = (1234)$$

$$\text{and } M^4 = I.$$

The permutation $M^2 = (13)(24) = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 3 & 4 & 1 & 2 \end{pmatrix}$ is said to be a product of two disjoint cycles, that is one cycle of the form $1 \rightarrow 3 \rightarrow 1$ and another of the form

$2 \rightarrow 4 \rightarrow 2$. Any permutation may be uniquely expressed as a product of disjoint cycles.

Group Structure

Lemma 2: The set of mappings $\{M, M^2, \dots, M^{r-1}\}$ forms a cyclic group G_m of order r .

Proof:

- 1) Machine concatenation is a valid binary operation and closure is present (Lemma 1). Associativity is obvious.
- 2) There exists a right identity namely I .
- 3) There exists a right inverse for any M^i namely M^{r-i} .

Theorem 2: The set of all $2^{(2^s-1)}$ CIOBSM(s)'s forms a group G_s of order $2^{(2^s-1)}$.

Proof:

- 1) The existence of a valid binary operation, closure, and associativity are guaranteed as before in Lemma 2.
- 2) The identity mapping is a CIOBSM(s).
- 3) If M is any CIOBSM(s) then by Lemma 2 its inverse, M^{r-1} , exists and is also a CIOBSM(s).

Proposition 2: The order of a finite group is a multiple of the order of every one of its subgroups.

Theorem 3: The order of any mapping M generated by a CIOBSM is of the form 2^α .

Proof: The theorem is an immediate consequence of theorem 2, lemma 2, and proposition 2.

Proposition 3: The order of any permutation, M , is the least common multiple of the lengths of its disjoint cycles.

Theorem 4: If 2^α is the order of a CIOBSM(s), then $\alpha \leq s$.

Proof: By proposition 3, all cycles of a permutation, M , representing a CIOBSM(s) must have lengths which are powers of 2. Therefore the order of M is the length of the longest cycle. As the degree of the permutation is only 2^s , no cycle can be longer than 2^s . Thus, no permutation may have order greater than 2^s .

CHAPTER VII

SYSTEM DESIGN CONSIDERATIONS

A cryptographic system must be designed with not only the mathematics and electronics in mind but also the human being that will interact with it. Failures of all types in either sphere must be reckoned with in advance. The design of a crypto system must, therefore, incorporate careful technological foresight and some of the 'paranoia' of supreme security. The following is a list of basic criteria which should be applicable to overall crypto system design.

Defectors and Compromise

If a nation's cryptographic system is to be widely employed, then many people must have access to it. In the case of military operators these people are perhaps crypto officers who, after a tour of duty, may return to civilian life and then must be replaced. As a result, the membership in the crypto community expands from year to year. It is inevitable that security compromises will occur either through premeditated defection or negligence and therefore it is mandatory that the system be so

designed that occasional compromise will inflict no more than a minimal predetermined amount of damage to overall security.

Psychology

Many times in the past the security of secret communications depended upon the ability to outsmart or outthink the enemy. A notable example of this is the celebrated message of World War I:

PRESIDENT'S EMBARGO RULING SHOULD HAVE
IMMEDIATE NOTICE. GRAVE SITUATION
AFFECTING INTERNATIONAL LAW. STATEMENT
FORESHADOWS RUIN OF MANY NEUTRALS.
YELLOW JOURNALS UNIFYING NATIONAL
EXCITEMENT IMMENSELY (10).

The above is a typical, believable wartime message of little strategic importance if taken at face value. The Allies who authored it hoped it would pass for such for if the first letter of each word is taken, one gleans a message which is of much greater strategic value. This message was in fact received and deciphered in Britain's famous Room 40 by Sir Alfred Ewing's group. Another noteworthy example is that of Nauen's 'Lightning' discussed in Pratt's Secret and Urgent. The latter method comes under the heading of steganography, a most fascinating branch of cryptography (3).

The trend of thought in cryptography has followed the general trend in military thinking: Base your course of action on what the enemy CAN do rather than on what you think he WILL do. Thus ingenious concealment practices have been phased out and replaced by analytical rather than psychological methods for secure transmission practices.

Physical Security

In any highly classified operation there is a need for sound security practices. This is especially true in cryptographic work for the cryptographic network handles classified material relating to many secret matters at all levels of classification. The system is charged with the safe transmission of the nation's military plans and her sensitive diplomatic communiqués. Just as a document must bear as its overall classification the highest classification contained anywhere within, so must a crypto system assume the classification of the most sensitive material transmitted.

Operational Security

High speed electronics has had a great effect on the cryptographic field. The old cipher systems readily fall to pieces under the high speed and exhaustive analysis which is afforded by a large computation system.

It has thus become necessary to devise electronic systems that will so disguise the statistics of the language that analysis will confound the giant computers.

A good crypto system should have its security independent of all basic algorithms. In other words, the system should be designed so that the enemy will not be aided should they discover how the system works. The security of such a system should lie with a number of easily variable and periodically changed parameters or 'code sheets.'

Theoretical Security

For efficient operation, the crypto system will most likely employ a conventional mode of transmission and all transmissions should therefore be assumed to be available to the enemy. The cryptography must be of such a caliber that it will frustrate any reasonable attempt made by the enemy to decipher messages on the sole basis of information gained by interception and subsequent analysis of ciphered transmissions.

Variability

The variable parameters or 'code sheets' of the crypto system should be changed often and the changes should be easy to effect on the system. The number of parameters should be high, but not excessively so in order that the crypto officer would not become

unnecessarily burdened with operations. The larger the number of parameters and their ranges, the more time and cost to the enemy to do an exhaustive simulation to 'crack the code.'

Reliability and Simplicity

The crypto system must, of course, be reliable in all conditions under which it will be used. These conditions then include normal peace-time operations and those accompanying the various degrees of hostilities of war. Reliability is not to be construed as relating only to the cryptographic device but also to the communication channel used and to the human operator. The device must be easy to operate and the operation routine must be standardized and thoroughly ingrained in the operator for it is conceivable that some crypto systems might be compromised by a faulty enciphering procedure.

Ease of maintenance must also be a factor in design. The employment of non-standard electronic components and components that must be classified should be avoided.

Partition and Restriction of Critical Knowledge

A cryptographic system's 'code sheets' are certainly sensitive material but the methods used in generating the 'code sheets' are of greater sensitivity, they are critical. There is no need for a crypto officer to know these methods

and the community of those who must know the methods should be kept as small as possible. When possible this critical knowledge should be partitioned so that no single defector from the 'critical community' may do irreparable damage to the system and result in the compromise of all previous messages handled by it. Again it is worth emphasizing that general ideas are more susceptible to compromise than rafts of numbers. Therefore it is desirable that the security of the generation of the 'code sheets' also be dependent upon highly variable and often changed parameters rather than upon secret algorithms. Many of the security guidelines that apply to the design of the infield portion of the crypto system apply equally well to the system's smaller, critical crypto control and supply portion of which we have been speaking.

If all of the above rules are respected then it is clear that a total system compromise cannot affect more than a limited number of messages. This is of extreme importance.

CHAPTER VIII

EXAMPLE OF A MULTI-SUBSCRIBER CRYPTO SYSTEM

Now that the rules and methods for designing a successful cryptographic system have been developed, let us apply them in an example of a multi-subscriber secure communications network.

First let us design a cryptographic machine to serve as the heart of the system.

The Machine

Our (again hypothetical) cyptographic device is constructed of 32 of the elements one of which is shown in Figure 8.1. A reproducible pseudo random noise generator (RPRNG) "drives" the machine from state to state as will be shown. The machine can be in any one

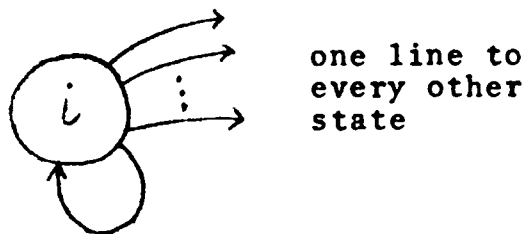


Figure 8.1 One of the 32 Elements of the Cryptographic Machine

of the 32 states. If it is in the i th state ($0 \leq i < 31$) then i is added modulo 32 to the clear character being processed: It is evident that this machine is of the

$\mathcal{P} = \mathcal{P}(i)$ type discussed in Chapter V. The clear text, as before, consists of the alphabetic and special characters A,B,...,Z,COMMA,..,SPACE,(,),* and their numerical equivalents 0,1,...,31. After the machine has processed or encoded one clear character it changes state according to the transition matrix of Figure 8.2. The numbers (0 to 31) along the vertical side of the matrix represent the machine's present state, while the

	0	1	2	...	31	(Input from the RPRNG)
0	3	2	0	...	20	
1	4	6	2	...	13	
2	1	0	8	...	6	
.	
.	
.	
31	2	1	3		30	

Figure 8.2 The Transition Matrix for the Cryptographic Machine

numbers (0 to 31) along the top represent the possible discrete outputs of the RPRNG. The numbers in the matrix represent the next states of the machine. Thus, if the

machine is in state 2 and receives input 0 from the RPRNG it will go into state 1. If, while the machine was in state 2, it received the clear character B, it would respond with the enciphered character D. For decoding, the inverse operation (modular subtraction) is performed.

The RPRNG might be any device that produces a flat (uniformly distributed) and statistically independent variable that ranges discretely between 0 and 31 inclusive. The PDF of such a variable would be the familiar set of delta functions shown in Figure 8.3. Such a device might be constructed out of two shift registers say of lengths

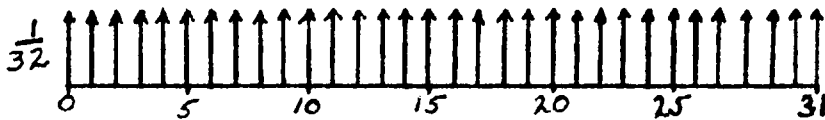


Figure 8.3 The PDF for the RPRNG

17 and 19 XORed together as shown in Figure 8.4. The

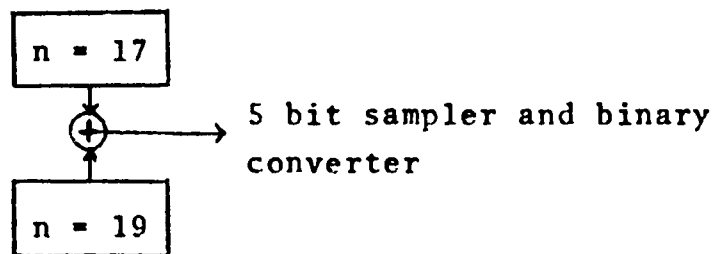


Figure 8.4 A Possible RPRNG

random variable could be created by forming the numbers which would be formed from 5 bit samples or pulses of the RPRNG. If for instance this particular RPRNG put out ...10110, this would correspond to 22.

Statistics of the Device

Because of the nature of the PDF and the device, it is clear that the ciphered text will have the same PDF as the RPRNG no matter what the statistics of the clear. This was discussed in Chapter V and mentioned as "perfect cryptography." Thus the enciphered text will appear as flat discrete noise and the different H_i entropies or signatures will all be maximized.

Scheme of Operation

Before a message can be deciphered, the decipherer must know three things about the sender:

- 1) the transition matrix used
- 2) the initial complexion (36 bits in our example) of the RPRNG shift registers
- 3) the initial or starting state of the crypto device.

In total then, the receiver must know $36+32 \times 32+32=1092$ different variables. It would be impractical to set all of these variables each time a subscriber wished to communicate, however, the requirement of variability demands that a sufficient number be changed each message.

A feasible plan of operation would be to change the basic transition matrix every week. This is the biggest job of all as it would require 32^2 or 1024 connections. It could conceivably be patched on 32 separate patchbays of the type shown in Figure 8.5.

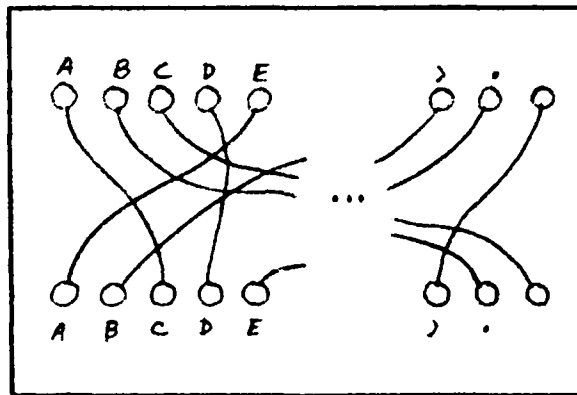


Figure 8.5 One of the 32 Patchbays (wired)

By simple calculation it is clear that there are $(32.)^{32}$ possible matrices from which to choose the transition matrix. It would also be reasonable to assign a given starting state for each subscriber to use for every message that he originates. These states could be distributed on a list to all subscribers.

Thus, every time a message is sent, the variable changed will be the initial states or complexions of the shift registers.

Problems and Solutions

At this point we may foresee a problem. If any two subscribers were to use the same initial shift register states, then the enemy, by modularly subtracting one message from the other, could remove the pseudo random noise from the transmissions and thus reduce the problem to an infinite key Vigenere cipher which is relatively simple to break. The first modification to help correct for this would be to change the shift register complexion every 5 minutes. A new codebook containing these complexions for every 5 minute interval could be issued each week along with the new transition matrix. A line from the codebook for 5:35 P.M. might read:

```
1735 U D D D U D U D D D U U D U U U D U D U U D U D D
      U D D U D D D U D U U
```

This notation merely stands for the switch positions of the 36 initial bits of the shift registers. U (UP) indicates a 1 and D (DOWN) indicates 0. However, suppose two subscribers wish to send a message in the same 5 minute interval. We are then still faced with the previous problem. The answer here is to provide each subscriber with a unique "flip-code." A flip-code would be something like:

```
1 4 6 11 19 28 33 35
```

This would mean to "flip" or change switches 1,4,6,11, 19,28,33,35 to the opposite positions from what the codebook says. Thus the operator would first set up:

U D D D U D U D D D U U D U U U D U D U U D U D D U D D
 U D D D U D U U

and then flip the indicated positions to get:

D D D U U U U D D D D U D U U U D U U U U D U D D U D U
 U D D D D D D U*

The security of such a system does not depend upon any algorithms which must be kept secret. If a compromise of the week's codebook should occur, it will not affect the security of the previous weeks or the following weeks.

*Care must of course be taken to ensure (either electronically or otherwise) that the initial complexion of a shift register is not all 0's.

REFERENCES

1. Gupta, H. Class Notes.
2. Harrison, M., Introduction to Switching and Automata Theory, McGraw-Hill, 1965.
3. Pratt, F., Secret and Urgent.
4. Shannon, C., "Communication Theory of Secrecy Systems," Bell Systems Technical Journal, 28/1949.
5. Drake and Sachs, Probability Theory.
6. Kahn, D., "Modern Cryptology," Scientific American.
7. Hershey, J., "A Procedure for First Analysis of Completely Deterministic Cryptographic Machines," IEEE Transactions on Information Theory, Jan., 1968.
8. Rayfield, M., Private Communication.
9. Hershey, J. and Hill, F., "Noncomputational Sequential Machines."
10. Explorations in Science.