Cryptanalysis

Cryptography joins the twentieth century

John Manferdelli
JohnManferdelli@hotmail.com

JLM 20210227 21:00

Outline

- Measuring information
  - Quantifying uncertainty
  - Perfect cryptosystems based on information limits
  - Correlating information complexity and the difficulty of breaking a cryptosystem
- The rise of the machines
  - Enigma
  - Breaking Enigma: bad key management
  - Breaking Enigma: isolating component transformations
  - Breaking Enigma: the birth of digital computers
  - Lesson: Measuring safety is not an easy business

Measuring information: Claude Shannon

Reference: Shannon, Communication Theory of Secrecy Systems (online)

Information Theory Motivation

- How much information is in a binary string?
- Game: I have a value between 0 and 2^n-1 (inclusive), find it by asking the minimum number of yes/no questions.
  - Write the number as $[b_{n-1}b_{n-2}\ldots b_0]_2$.
  - Questions: Is b_{n-1} 1?, Is b_{n-2} 1? , … , Is b_0 1?
- So, what is the amount of information in a number between 0 and 2^n-1?
  - Answer: n bits
  - The same question: Let X be a probability distribution taking on values between 0 and 2^n-1 with equal probability. What is the information content of a observation?
  - There is a mathematical function that measures the information in an observation from a probability distribution. It's denoted H(X).
- $H(P) = \sum_{i=1}^{n} -p_i \lg(p_i)$

What is the form of H(X)?

- If H is continuous and satisfies:
  - $H\left(\frac{1}{n}, \dots, \frac{1}{n}\right) < H\left(\frac{1}{n+1}, \dots, \frac{1}{n+1}\right)$
  - $H(p_1, \dots, p_n) = H\left(p_1, \dots, qp_j, (1-q)p_j, \dots, p_n\right)$
  - $H\left(\frac{1}{n}, \dots, \frac{1}{n}\right) = 1$
- $H(p_1, \dots, p_n)$ is maximized if $p_j = \frac{1}{n}$ for all j

Information Theory

- The “definition” of H(X) has two desirable properties:
  - Doubling the storage (the bits your familiar with) doubles the information content
  - $H(1/2, 1/3, 1/6)= H(1/2, 1/2) + \frac{1}{2} H(2/3,1/3)$
- It was originally developed to study how efficiently one can reliably transmit information over “noisy” channel.
- Applied by Shannon to Cryptography (BTSJ, 1949)
- Thus information learned about Y by observing X is

  $I(Y,X)= H(Y)-H(Y|X)=H(X)+H(Y)-H(X,Y).$
- Used to estimate requirements for cryptanalysis of a cipher.

Sample key distributions

- Studying key search
  - Distribution A: 2 bit key each key equally likely
  - Distribution B: 4 bit key each key equally likely
  - Distribution C: n bit key each key equally likely
  - Distribution A': 2 bits with distribution (1/2, 1/6, 1/6, 1/6)
  - Distribution B': 4 bits with distribution (1/2, 1/30, 1/30, …, 1/30)
  - Distribution C': n bits with distribution (1/2, ½ $1/(2^n-1)$,…,½ $1/(2^n-1)$)

H for the key distributions

- Distribution A: H(X)= ¼ lg(4) + ¼ lg(4) + ¼ lg(4) +1/4 lg(4) = 2 bits
- Distribution B: H(X)= 16x(1/16) lg(16)= 4 bits
- Distribution C: H(X)= 2^nx(1/2^n) lg(2^n) = n bits
  - Expected time for key search is ~ 2^n.
- Distribution A': H(X) = ½ lg(2) + 3 x(1/6 lg(6))= 1.79 bits
- Distribution B': H(X) = ½ lg(2) + 15 x(1/30 lg(30))= 2.95 bits
- Distribution C': H(X) = ½ lg(2) + ½(2^n-1)x(1/(2^n-1) lg(2^n-1))= n/2+1 bits
  - Expected time for key search is ~ ½(2^n+1).

Coding theory and Information

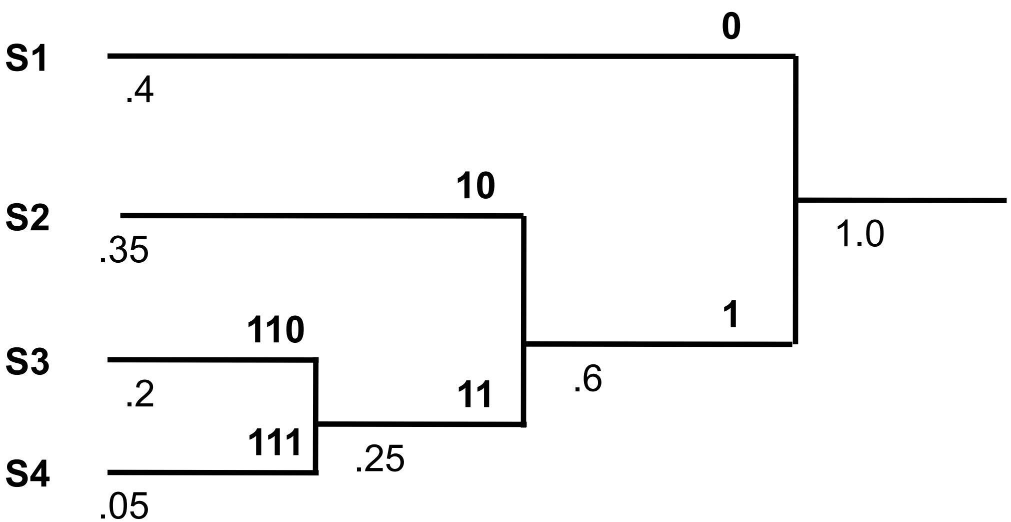
- **Shannon Source Coding:** If a memoryless source has entropy H then any uniquely decipherable code over an alphabet Σ with D symbols must have length ≥H. Further, there is a uniquely decipherable code with average length ≤ 1+H/lg(D).
- Applications to compression
- Pick 0 with probability .8 and all of the remaining (2^n-1) n bit numbers with equal probability
  - H= - .8 lg(.8) -S .2(2^n-1)$^{-1}$lg((2^n-1)$^{-1}$).
  - For n=10, we get H= 2.7 bits (Not 10!)
- How many bits on average does it take to encode n bits with the distribution above?
  - Code 0,1 with one bit, 2→3 with two bits, …, 2^{n-1}→2^n-1 with n bits.
  - For n=10, the average bit length is:
  - .8x1 + .2/1023(1x1+2x2+4x3+8x4+…+512x10)=2.7

Some Theorems

- Bayes: P(X=x|Y=y) P(Y=y)= P(Y=y|X=x) P(X=x)= P(X=x, Y=y)
- X and Y are independent iff P(X=x, Y=y)= P(X=x)P(Y=y)
- H(X,Y)= H(Y)+H(X|Y)
- H(X,Y)≤ H(X)+H(Y)
- H(Y|X)≤ H(Y) with equality iff X and Y are independent.
- If X is a random variable representing an experiment in selecting one of N items from a set, S, H(X)<lg(N) with equality iff every selection is equally likely (Selecting a key has highest entropy off each key is equally likely).
- $H(X|Y) = -\sum_x p_X(x) H(Y|X = x)$which is generally not equal to $-\sum_{x,y} p_X(y|x) \lg(p_Y(y|x)$.
- H(K|C) = H(M|C) + H(K|M, C).

Huffman Coding

- Uniquely readable
- Average length, L, satisfies
  - $H(X) \leq L < H(X)+1$



$H(X) = -(.4 \lg(.4)) + .35 \lg(.35) + .2 \lg(.2) + .05 \lg(.05))$

$H(X) = 1.74$, $\lceil H(X) \rceil = 2$. [y] means the ceiling function, the smallest integer greater than or equal to y.

Morse Code

A	. -	**N**	- .
B	- . . .	**O**	- - -
C	- . - .	**P**	. - - .
D	- . .	**Q**	- - . -
E	.	**R**	. - .
F	. . - .	**S**	. . .
G	- - .	**T**	-
H	. . . .	**U**	. . -
I	. .	**V**	. . . -
J	. - - -	**W**	. - -
K	- . -	**X**	- . . -
L	. - . .	**Y**	- . - -
M	- -	**Z**	- - . .

Long term equivocation

- $H_E = \ lim_{n \to \infty}(\frac{H(P_n)}{n})$. (“Entropy per character”)
- For random stream of letters
  - $H_R = \ \sum_{i=1}^{26} \frac{1}{26} \lg(26)$
- For English
  - H_E = 1.2-1.5 (so English is about 75% redundant)
  - There are approximately T(n)= 2^{nH} , n symbol messages that can be drawn from the meaningful English sample space.
- How many possible cipher-texts make sense?
  - $H(P^n)+H(K) > H(C^n)$
  - $nH_E + \lg(|K|) > n \lg(|S|)$
  - $n \frac{\lg(|K|)}{\lg(|S|)} - \ H_E > n$
  - $R = 1 \ - \ \frac{H_E}{\lg(|S|)}$

Unicity and random ciphers

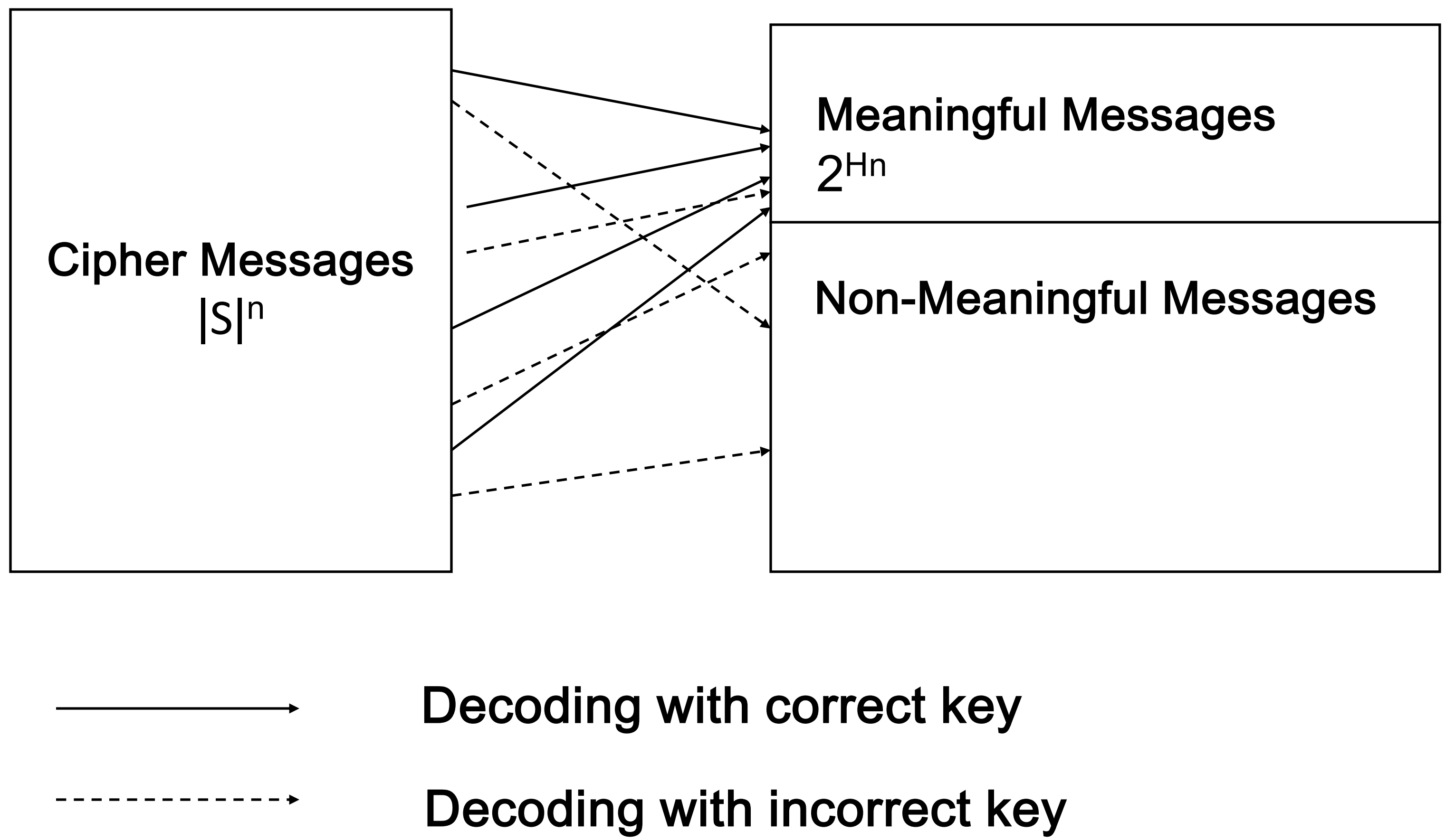
Question: How many messages do I need to trial decode so that the expected number of false keys for which all m messages land in the meaningless subset is less than 1?

Answer: The unicity point.

Nice application of Information Theory.

Theorem: Let H be the entropy of the source (say English) and let S be the alphabet. Let K be the set of (equiprobable) keys. Then $u = \frac{\lg(|K|)}{\lg(|\Sigma|) - H}$.

Unicity for random ciphers



Unicity distance for mono-alphabet

- $H_{CaeserKey}$= H_{random} = lg(26)= 4.7004
- $H_{English}$ ≅1.2.

- For Caeser, u ≅ lg(26)/(4.7-1.2)≅4 symbols, for cipher-text only attack. For known plaintext/cipher-text, only 1 corresponding plain/cipher symbol is required for unique decode.
- For arbitrary substitution, u ≅ lg(26!)/(4.7-1.2) ≅ 25 symbols for cipher-text only attack. For corresponding plain/cipher-text attack, about 8-10 symbols are required.

- Both estimates are remarkably close to actual experience.

Information theoretic estimates to break mono-alphabet

Cipher	Type of Attack	Information Resources	Computational Resources
Caeser	Ciphertext only	U= 4.7/1.2=4 letters	26 computations
Caeser	Known plaintext	1 corresponding plain/cipher pair	1
Substitution	Ciphertext only	~30 letters	O(1)
Substitution	Known plaintext	~10 letters	O(1)

One Time Pad (OTP)

- The one time pad or Vernam cipher takes a plaintext consisting of symbols **p**= $(p_0, p_1, \ldots, p_n)$ and a key-stream **k**= $(k_0, k_1, \ldots, k_n)$ where the symbols come from the alphabet Z_m and produces the cipher-text **c**= $(c_0, c_1, \ldots, c_n)$ where $c_i = (p_i + k_i) \pmod{m}$.
- m=2 in the binary case and m=26 in the case of the roman alphabet.
- Unfortunately, OTP requires shared keys as long as the sum of the lengths of all plaintexts sent.
- Stream ciphers replace the 'perfectly random' sequence **k** with a pseudo-random sequence **k'** (based on a much smaller input key $\mathbf{k_s}$ and a stream generator R).

One-time pad alphabetic encryption

Plaintext +Key (mod 26)= Ciphertext

B	U	L	L	W	I	N	K	L	E	I	S	A	D	O	P	E
1	20	11	11	22	08	13	10	11	04	08	18	00	03	14	15	04

Plaintext

N	O	W	I	S	T	H	E	T	I	M	E	F	O	R	A	L
13	14	22	08	18	19	07	04	19	08	12	04	05	14	17	00	11

Key

14	8	07	19	14	01	20	14	04	12	20	22	05	17	05	15	15
O	S	H	T	0	B	U	O	E	M	U	W	F	R	F	P	P

Ciphertext

Legend

A	B	C	D	E	F	G	H	I	J	K	L	M
00	01	02	03	04	05	06	07	08	09	10	11	12
N	O	P	Q	R	S	T	U	V	W	X	Y	Z
13	14	15	16	17	18	19	20	21	22	23	24	25

One-time pad alphabetic decryption

Ciphertext+26-Key (mod 26)= Plaintext

14	8	07	19	14	01	20	14	04	12	20	22	05	17	05	15	15	*Ciphertext*
O	S	H	T	0	B	U	O	E	M	U	W	F	R	F	P	P	
N	O	W	I	S	T	H	E	T	I	M	E	F	O	R	A	L	*Key*
13	14	22	08	18	19	07	04	19	08	12	04	05	14	17	00	11	
B	U	L	L	W	I	N	K	L	E	I	S	A	D	O	P	E	*Plaintext*
1	20	11	11	22	08	13	10	11	04	08	18	00	03	14	15	04	

Legend

A	B	C	D	E	F	G	H	I	J	K	L	M
00	01	02	03	04	05	06	07	08	09	10	11	12
N	O	P	Q	R	S	T	U	V	W	X	Y	Z
13	14	15	16	17	18	19	20	21	22	23	24	25

Binary one-time pad

Plaintext $\oplus$ Key = Ciphertext

Ciphertext $\oplus$ Key = Plaintext

10101110011100000101110110110000	*Plaintext*
00101010011010110001010110010111	*Key*
10100100000110110100100000100111	*Ciphertext*
00101010011010110001010110010111	*Key*
10101110011100000101110110110000	*Plaintext*

The one time pad has perfect security

- One-time pad is perfect: E is perfect if $H(X|Y)=H(X)$ where X is a plain text distribution and Y is the cipher text distribution.

Proof:

$H(X|Y) = -\sum_{y \text{ in } Y} P(Y=y)\, H(X|Y=y)) = -\sum_{y \text{ in } Y} P(Y=y) \sum_{x \text{ in } X} P(X=x|Y=y) \lg(P(X=x|Y=y))$.

$P(X=x|Y=y)\, P(Y=y) = P(X=x, Y=y)$ and $P(X=x,Y=y) = \Pr(X=x, K=x+y) = P(X=x)P(K=k)$.

So $H(X|Y) = -\sum_{y \text{ in } Y,\, x \text{ in } X} P(X=x,Y=y)\, [\lg(P(X=x,Y=y) - P(Y=y)]$

$= -\sum_{y \text{ in } Y,\, x \text{ in } X} P(X=x, Y=y) \lg(P(X=x, Y=y)) + \sum_{y \text{ in } Y,\, x \text{ in } X} P(X=x,Y=y) \lg(P(Y=y))$

$= -\sum_{x \text{ in } X,\, y \text{ in } Y} P(X=x)P(K=x+y)\lg(P(X=x) - \sum_{x \text{ in } X,\, y \text{ in } Y} P(X=x)$ $P(Y=x+k)\lg(P(Y=x+k)$

$+\sum_{y \text{ in } Y,\, x \text{ in } X} P(X=x)\, P(Y=Y)\lg(P(Y=y))$

$= H(X)$

Shannon: Mixing cryptographic elements to produce strong cipher

- Diffusion – transposition
  - Using group theory, the action of a transposition τ on $a_1\ a_2 \ldots, a_k$ could be written as $a_{\tau(1)}\ a_{\tau(2)} \ldots, a_{\tau(k)}$.
- Confusion – substitution
  - The action of a substitution σ on $a_1\ a_2 \ldots, a_k$ can be written as

    $\sigma(a_1)\ \sigma(a_2) \ldots \sigma(a_k)$.
- Transpositions and substitutions may depend on keys. Keyed permutations may be written as $\sigma_k(x)$. A block cipher on b bits is nothing more than a keyed permutation on 2^b symbols.

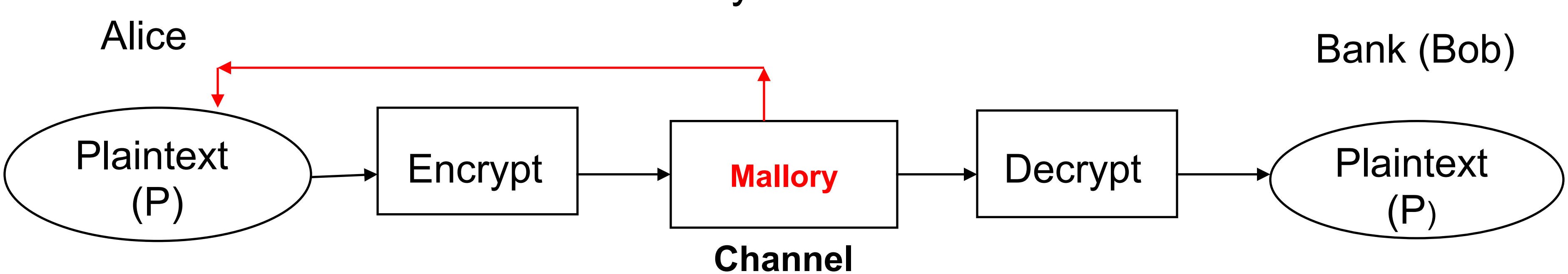
- Iterative Ciphers – staged iteration of permutation (transposition) and key dependent substitution to construct cipher. (DES, AES)

Interlude: Think like an adversary

- Alice has a bank account at the First National Bank and she does some online transactions. One of the services she can use is “Transfer money to another account.” This is done using a One Time Pad system set up by the Bank. Alice and the Bank get together and share series of one time pad keys. The Bank assured Alice this system is safe because one time pads are unbreakable. To transfer money, Alice sends a message like the one below to her Bank, encrypting it with the one time pad:

  `ALICE, ACCOUNT NUMBER, 123-456789 TRANSFERS $000020 to RECEPIENT, ACCOUNT NUMBER, 321-987654. SIGNED, ALICE.`

- Alice has a friend named Mallory, who is a “man in the middle” based attacker. Recall the Alice-Mallory-Bob adversarial model.



Interlude: The Sting

- One day, Alice and Mallory are having lunch and Alice runs out of cash. Mallory offers to loan her $20 and says: “You can repay me with an electronic transfer, my account number is 666-123456.
- Alice sends the following message to her bank and encrypts it with the one-time pad:

```
123456789012345678901234567890123456789012345678901234555
ALICE, ACCOUNT NUMBER, 123-456789 TRANSFERS $000020 to
MALLORY, ACCOUNT NUMBER, 666-123456.  SIGNED, ALICE.
```

- The numbers on the top are not sent, they just show the byte position of the characters in the message.
- Mallory intercepts the message (perhaps she controls an internet router between Alice and her Bank). She can’t decrypt it but flips the bits at character position 46 by xoring in the 8 bit string 00000001 (thus changing the ascii character “0” in this position to ascii “1”) and sends it to the Bank.

It’s bad … and worse

- When the Bank decrypts the modified message it gets:

  ```
  ALICE, ACCOUNT NUMBER, 123-456789 TRANSFERS $100020 to
  MALLORY, ACCOUNT NUMBER, 666-123456.  SIGNED, ALICE.
  ```
- Oops.
- It’s actually worse for the Bank than Alice (in the US). Alice hires a cryptographer who goes to Court and explains how easy it is to mount this attack. Alice, grief stricken, explains that the Bank told her this was “completely safe.” The Court says: “The Bank must cover the loss.”
- Oops.
- By the way, Alice and Mallory might have colluded from the beginning and shared this attack with their best 1000 friends. What happens to the Bank?
- Problem: The message was neither integrity protected or signed.

Rise of the Machines

The “Machine” Ciphers

- Simple Manual Wheel
  - Jefferson
- Rotor
  - Enigma
  - Hebern
  - SIGABA
  - TYPEX
- Stepping switches
  - Purple
- Mechanical Lug and cage
  - M209

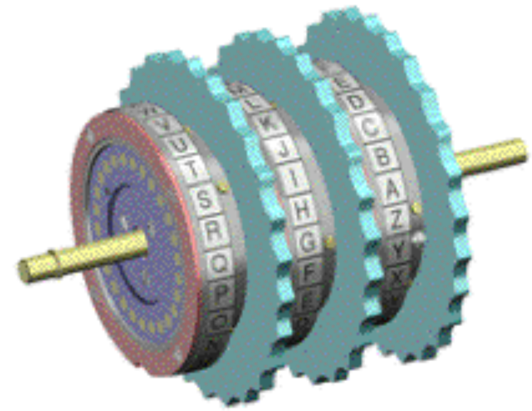
Jefferson Cipher

- The French have another name for this cipher.

Enigma

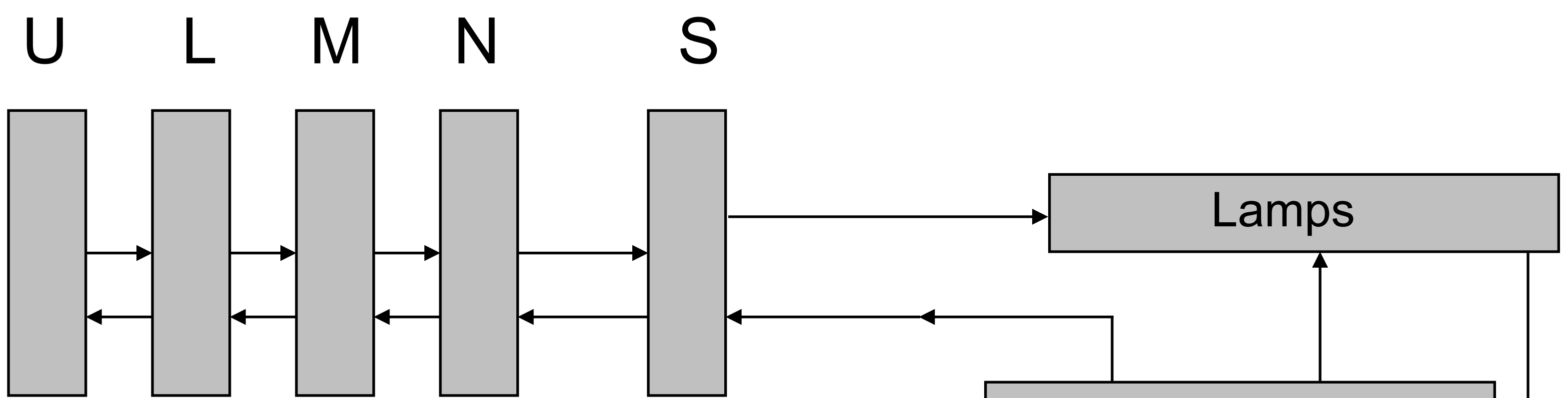
Enigma Cryptographic Elements (Army Version)

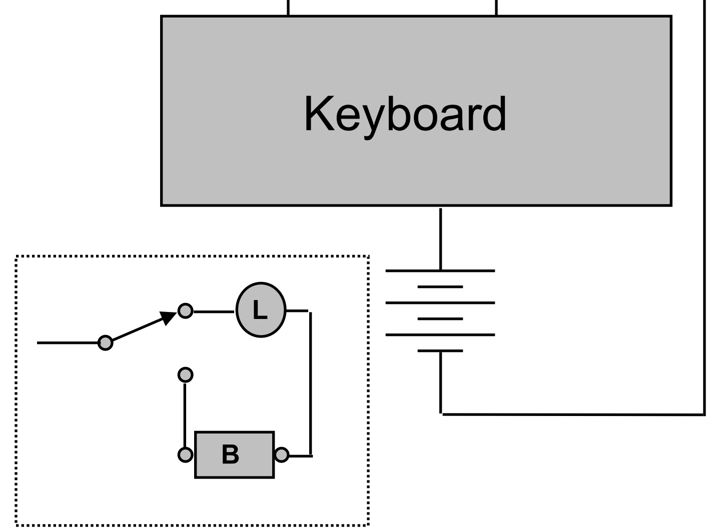
- Three moveable rotors
  - Select rotors and order
  - Set initial positions
- Moveable ring on rotor
  - Determine rotor ‘turnover’
- Plugboard (Stecker)
  - Interchanges pairs of letters
- Reversing drum (Umkehrwalze)
  - Static reflector
  - See next page



Three Rotors on axis

Diagrammatic Enigma Structure





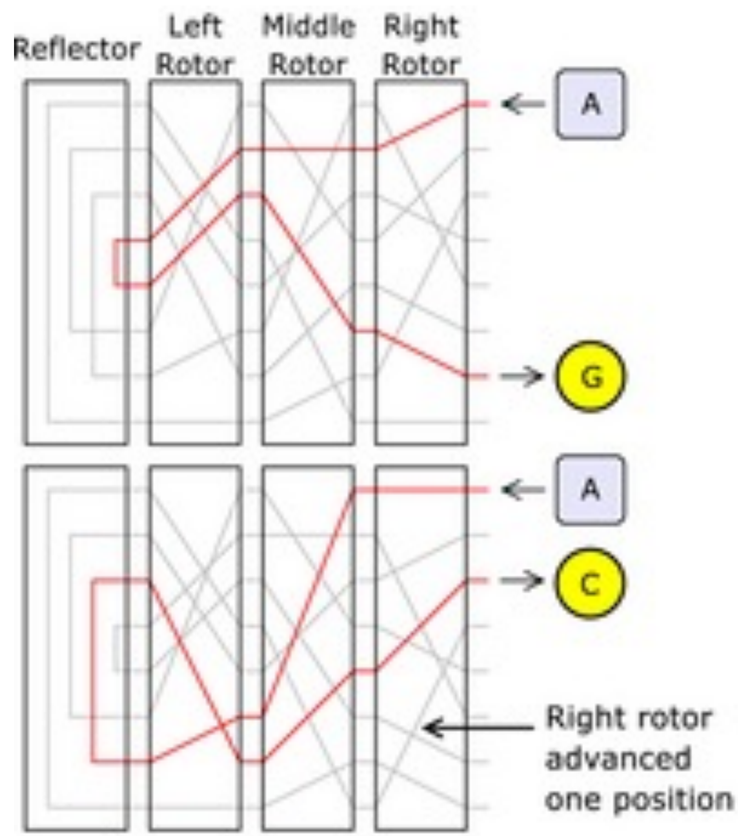
Message flows right to left

- U: Umkehrwalze (reversing drum)
- N,M,L: First (fastest), second third rotors
- S: Stecker (plugboard)

Diagram courtesy of Carl Ellison

Enigma Structure

Follow the electrons:



Original slide by Mark Stamp

Enigma Data

Rotors

Input	ABCDEFGHIJKLMNOPQRSTUVWXYZ
Rotor I	EKMFLGDQVZNTOWYHXUSPAIBRCJ
Rotor II	AJDKSIRUXBLHWTMCQGZNPYFVOE
Rotor III	BDFHJLCPRTXVZNYEIWGAKMUSQO
Rotor IV	ESOVPZJAYQUIRHXLNFTGKDCMWB
Rotor V	VZBRGITYUPSDNHLXAWMJQOFECK
Rotor VI	JPGVOUMFYQBENHZRDKASXLICTW
Rotor VII	NZJHGRCXMYSWBOUFAIVLPEKQDT

Ring Turnover

Rotor I	R
Rotor II	F
Rotor III	W
Rotor IV	K
Rotor V	A
Rotors VI	A/N

Reflector B (AY) (BR) (CU) (DH) (EQ) (FS) (GL) (IP) (JX) (KN) (MO) (TZ) (VW)

Reflector C (AF) (BV) (CP) (DJ) (EI) (GO) (HY) (KR) (LZ) (MX) (NW) (TQ) (SU)

Visualizing rotor motion

Original position (“A”)

ABCDEFGHIJKLMNOPQRSTUVWXYZ	Input
EKMFLGDQVZNTOWYHXUSPAIBRCJ	Rotor output
EKMFLGDQVZNTOWYHXUSPAIBRCJ	Final output

Rotor moves 25 position (“Z”)

ABCDEFGHIJKLMNOPQRSTUVWXYZ	Input
ZABCDEFGHIJKLMNOPQRSTUVWXY	Input to rotor (shifted)
JEKMFLGDQVZNTOWYHXUSPAIBRC	Rotor output
KFLNGMHERWAOUPXZIYVTQBJCSD	

- If rotor is R and P=(ABCD…Z), the effect at rotor position 1, writing permutations “from the right,” is $x_{in}(PRP^{-1})= x_{out}$.

First 13 settings of Rotor I

```
     abcdefghijklmnopqrstuvwxyz
Step
 0(A) EKMFLGDQVZNTOWYHXUSPAIBRCJ
 1(B) JLEKFCPUYMSNVXGWTROZHAQBID
 2(C) KDJEBOTXLRMUWFVSQNYGZPAHCI
 3(D) CIDANSWKQLTVEURPMXFYOZGBHJ
 4(E) HCZMRVJPKSUDTQOLWEXNYFAGIB
 5(F) BYLQUIOJRTCSPNKVDWMXEZFHAG
 6(G) XKPTHNIQSBROMJUCVLWDYEGZFA
 7(H) JOSGMHPRAQNLITBUKVCXDFYEZW
 8(I) NRFLGOQZPMKHSATJUBWCEXDYVI
 9(J) QEKFNPYOLJGRZSITAVBDWCXUHM
10(K) DJEMOXNKIFQYRHSZUACVBWTGLP
11(L) IDLNWMJHEPXQGRYTZBUAVSFKOC
12(M) CKMVLIGDOWPFQXSYATZUREJNBH
```

Last 13 settings of Rotor I

```
       abcdefghijklmnopqrstuvwxyz
Step
13(N)  JLUKHFCNVOEPWRXZSYTQDIMAGB
14(O)  KTJGEBMUNDOVQWYRXSPCHLZFAI
15(P)  SIFDALTMCNUPVXQWROBGKYEZHJ
16(Q)  HECZKSLBMTOUWPVQNAFJXDYGIR
17(R)  DBYJRKALSNTVOUPMZEIWCXFHQG
18(S)  AXIQJZKRMSUNTOLYDHVBWEGPFC
19(T)  WHPIYJQLRTMSNKXCGUAVDFOEBZ
20(U)  GOHXIPKQSLRMJWBFTZUCENDAYV
21(V)  NGWHOJPRKQLIVAESYTBDMCZXUF
22(W)  FVGNIOQJPKHUZDRXSACLBYWTEM
23(X)  UFMHNPIOJGTYCQWRZBKAXVSDLE
24(Y)  ELGMOHNIFSXBPVQYAJZWURCKDT
25(Z)  KFLNGMHERWAOUPXZIYVTQBJCSD
```

Military Enigma

Encryption Equation: $c= (p)P^iNP^{-i}\,P^jMP^{-j}\,P^kLP^{-k}\,U\,P^kL^{-1}P^{-k}\,P^jM^{-1}P^{-j}\,P^iN^{-1}P^{-i}$

- K: Keyboard
- P=(ABCDEFGHIJKLMNOPQRSTUVWXYZ)
- N: First Rotor
- M: Second Rotor
- L: Third Rotor
- U: Reflector. Note: $U=U^{-1}$.
- i, j, k: Number of rotations of first, second and third rotors respectively.

- Note $(p)E=c \rightarrow (c)E=p$. $E=E^{-1}$.
- Later military models added plugboard (S) and additional rotor (not included). The equation with plugboard is:
- $c=(p)S\,P^iNP^{-i}\,P^jMP^{-j}\,P^kLP^{-k}\,U\,P^kL^{-1}P^{-k}\,P^jM^{-1}P^{-j}\,P^iN^{-1}P^{-i}\,S^{-1}$

Group Theory for Rotors

- Writing cryptographic processes as group operation can be very useful. For example, if R denotes the mapping of a "rotor" and C=(1,2,…,26), the mapping of the rotor "turned" one position is CRC^{-1}.
- A prescription for solving ciphers is to represent the cipher in terms of the basic operations and then solve the component transformations. That is how we will break Enigma.
- For most ciphers, the components are substitution and transposition; some of which are "keyed".
- For Enigma, you should know the following:
  - Theorem: If $s=(a_{11}\ a_{12}\ \dots\ a_{1i})\ (a_{11}\ \dots\ a_{1j})\ \dots\ (a_{11}\ \dots\ a_{1k})$ then $dsd^{-1}= (da_{11}\ da_{12}\ \dots\ da_{1i})\ (da_{11}\ \dots\ da_{1j})\ \dots\ (da_{11}\ \dots\ da_{1k})$.
  - When permutations are written as products of cycles, it is very easy to calculate their order. It is the LCM of the length of the cycles.

Military Enigma Key Length

- Key Length (rotor order, rotor positions, plug-board)
  - 60 rotor orders. lg(60)= 5.9 bits. [Original Army cipher with 5 rotors]
  - 26*26*26 = 17576 initial rotor positions. lg(17576)= 14.1 bits of key
  - 10 exchanging steckers were specified yielding C(26,2) C(24,2)…C(8,2)/10! = 150,738,274,937,250. lg(150,738,274,937,250)= 47.1 bits as used
  - Bits of key: 5.9 + 14.1 + 47.1 = 67.1 bits
  - Moveable ring lg(26^2)= 9.4 bits
  - Note: plugboard triples entropy of key!
- Rotor Wiring State
  - lg(26!) = 88.4 bits/rotor.
- Total Key including rotor wiring:
  - 67.1 bits + 3 x 88.4 bits = 312.3 bits

Method of Batons

- Applies to Enigma
  - Without plug-board
  - With fast rotor ordering known and only the fast rotor moving
  - With a "crib"
- Let N be the fast rotor and Z the combined effect of the other apparatus, then $N^{-1}ZN(p)=c$, for first character.
- In general, $ZP^{-i}N\,P^{i}p(i)= P^{-i}NP^{i}\,c(i)$ (from left)
- Since $ZP^{-i}NP^{i}\,(p)=P^{-i}NP^{i}\,(c)$, we know the wiring of N and a crib, we can play the crib against each of the 26 possible positions of N for the plaintext and the cipher text. In the correct position, there will be no "scritches" or contradictions in repeated letters.
- This method was used to "analyze" the early Enigma variants used in the Spanish Civil War and is the reason the Germans added the plugboard. Countermeasure: Move fast rotor next to reflector.

Method of Batons: example

- Crib: RECONAISSANCE

```
r e c o n n a i s s a n c e
U P Y T E Z O J Z E G B O T
```

- Rotor I
  - 01234567890123456789012345
  - ABCDEFGHIJKLMNOPQRSTUVWXYZ
  - EKMFLGDQVZNTOWYHXUSPAIBRCJ
- P shifted by Y
  - YZABCDEFGHIJKLMNOPQRSTUVWX
  - ABCDEFGHIJKLMNOPQRSTUVWXYZ
- Rotor position A: ZN(p) over N(c), sample sritch in red

```
UFJRQNXAWBDRMH
AWCYRGUQINNDSM
```

- Rotor position Y: ZN(p) over N(c), no scritch

```
JGMGFUHRWCNSEW
UZCZBJOTAMQESA
```

Polish (Rejewski) Attack

- Rejewski exploited weakness in German keying procedure to determine rotor wiring
  - Rejewski had ciphertext for several months but no Enigma.
  - Rejewski had stecker settings for 2 months (from a spy via the French in 12/32), leaving 265.2 bits of key (the rotor wirings) to be found. He did.
- Poles determined the daily keys
  - Rejewski catalogued the characteristics of rotor settings to detect daily settings. He did this with two connected Enigmas offset by 3 positions (the “cyclotometer”).
  - In 9/38, when the “message key” was no longer in this way, Rejewski’s characteristics stopped working.

Early German Keying Procedure

- Every signal officer had a list of global Enigma keys
  - Stecker (reflector) wiring
  - Rotor position
  - Turnover settings
  - Rotor starting position
- To send a message:
  - Operator picks three letter key (ABC) called the indicator
  - Uses daily setting to encrypt the indicator twice
  - Transmits these 6 letters
  - Reset rotors to indicator setting
  - Encrypt message
  - The indicator is an ephemeral key used to reduce exposure of the daily keys.
  - Good idea, in principal. Not so good in practice.

Two Theorems

- **<u>Theorem 1</u>***:* If S= $(a_1, a_2, \ldots, a_{n1})$ $(b_1, b_2, \ldots, b_{n2})$… and T is another permutation, then the effect of $T^{-1}ST$, operating from the left, is $T^{-1}ST = (a_1T, a_2T, \ldots, a_{n1}T)$ $(b_1T, b_2T, \ldots, b_{n2}T)$.
  - Example: S= (12345) (67), T= (17)(26)(35).

    $T^{-1}ST$= (76543)(12)
- **<u>Theorem 2</u>***:* Let S be a permutation of even degree. S can be decomposed into pairs of cycles of equal length if and only if it can be written as the product of two transpositions.
  - Example: S= (1234)(8765). A= (15)(26)(37)(48), B= (25)(36)(47)(18). AB= (1234)(8765).

Note: permutations applied "from the right" here.
You never know when group theory can help!

Plan of attack

- Define $E(i,j,k)= P^iNP^{-i}\, P^jMP^{-j}\, P^kLP^{-k}\, U\, P^kL^{-1}P^{-k}\, P^jM^{-1}P^{-j}\, P^iN^{-1}P^{-i}$
- Here, N is rotor 1, M is rotor 2, L is rotor 3 and U is the reflector.
- Let A= E(1,j,k), B= E(2,j,k), C= E(3,j,k),
  D= E(4,j,k), E= E(5,j,k), F= E(6,j,k).
- Suppose the (unknown) message key is `abg`.
- A,B,C,D,E,F are involutions representing the effect of Enigma on, `abgabg`. The six letter output on abgabg is called the indicator.
- Now suppose the six letter indicator for a message is `ktz svf`.
- aA=k, aD=s; bB=t, bE=v; and gC=z, gF=f, for unknown letters a, b, g.
  - Since, $A= A^{-1}$, etc., we obtain t(AD)=s, v(BE)= z(CF).
- In the first part of the attack
  - Use message indicators to construct (AD), (BE) and (CF).
  - Then use (AD), (BE) and (CF) to find A, B, C, D, E, F.

Getting the rotor

- Set
  - $Q= MLRL^{-1}M^{-1}$, $U= NP^{-1}QPN^{-1}$, $V= NP^{-2}QP^{2}N^{-1}$,
  - $W= NP^{-3}QP^{3}N^{-1}$, $X= NP^{-4}QP^{4}N^{-1}$, $Y= NP^{-5}QP^{5}N^{-1}$,
  - $Z= NP^{-6}QP^{6}N^{-1}$, $H=NPN^{-1}$.
- Use the knowledge of A and S to compute
  - $U=P^{-1}S^{-1}ASP^{1}$, $V=P^{-2}S^{-1}ASP^{2}$, $W=P^{-3}S^{-1}ASP^{3}$, $X=P^{-4}S^{-1}ASP^{4}$
  - $Y=P^{-5}S^{-1}ASP^{5}$, $Z=P^{-6}S^{-1}ASP^{6}$
- Next note
  - $UV= NP^{-1}(QP^{-1}QP)P^{1}N^{-1}$, $VW= NP^{-2}(QP^{-1}QP)P^{2}N^{-1}$.
  - $WX= NP^{-3}(QP^{-1}QP)P^{3}N^{-1}$, $XY= NP^{-4}(QP^{-1}QP)P^{4}N^{-1}$, $YZ= NP^{-5}(QP^{-1}QP)P^{5}N^{-1}$.
- Now we can calculate H.
  - $(VW)= H^{-1}(UV)H$, $(WX)= H^{-1}(VW)H$,
  - $(XY)= H^{-1}(WX)H$, $(YZ)= H^{-1}(XY)H$.
- Finally, $H=NPN^{-1}$ lets us compute N.

Calculate (AD), (BE), (CF)

$C=(p) S\ P^i N P^{-i}\ P^j M P^{-j}\ P^k L P^{-k}\ U\ P^k L^{-1} P^{-k}\ P^j M^{-1} P^{-j}\ P^i N^{-1} P^{-i}\ S^{-1}$

- Using the message indicators and:
  - $AD= SP^1NP^{-1}QP^1N^{-1}P^3NP^{-4}QP^4N^{-1}P^{-4}S^{-1}$. $(c_1)AD= c_4$.
  - $BE= SP^2NP^{-2}QP^2N^{-1}P^3NP^{-5}QP^5N^{-1}P^{-5}S^{-1}$. $(c_2)BE= c_5$.
  - $CF= SP^3NP^{-3}QP^3N^{-1}P^3NP^{-6}QP^6N^{-1}P^{-6}S^{-1}$. $(c_3)CF= c_6$.
- We can find AD, BE and CF after about 80 messages.

Calculate A, B, C, D, E, F

- Suppose
  - AD= (dvpfkxgzyo)(eijmunqlht)(bc)(rw)(a)(s)
  - BE= (blfqveoum)(hjpswizrn)(axt)(cgy)(d)(k)
  - CF= (abviktjgfcqny)(duzrehlxwpsmo)
- We know from the theorem that A maps the top line to the bottom line when the bottom is slid the correct amount

```
dvpfkxgzyodvpfkxgzy    and  bcb
thlqnumjie                   wr
```

For some bottom displacement of each line and D can then be calculated as in theorem 2 but what is the correct slide?

We can calculate B, C and E and F in a similar manner provided we know the correct displacement.

Cillies

- We can find the correct positions by trial and error but that's very time consuming.
- Frequently, operators chose the same letter for each character of the message code. These were called cillies
- Suppose this happened and we saw the indicator

  `YSG SWK` under this assumption for

  `AD=(DP)(SY)(ABQHZUIWOXL)(MNJRVTGCKEF)`

  `BE=(AJV)(HNY)(BFZSWGCIMO)(DKTLRXEQUP)`
- α`A=y`, α`B=s`, α`C=g`, a`D=s`, α`E=w`, α`F=k` for some unknown α.
- Then `A=(`α`y)(`β`s)…`, `B=(`α`s)…`, `C= (`α`g))…`, `D=(`α`s)(`β`y) …`, `E= (`α`w)…`, `F= (`α`k)…` Thus a=p, b=d or a=d, b=p.
- a cannot be `s` since then we'd have `A=(dy)(ps)…`, `B=(ds) …`, `C= (dg)…`, `D= (ds)(py)…`, `E=(sw)…`, `F=(sk)`. But this contradicts the fact that `s` and `w` are in the same cycle in `BE`.
- So `A=(yp)(sd)…`, `B=(sp)…`, `C=(gp))…`, `D=(sp)(yd) …`, `E= (wp)…`, `F=(kp)…`
- We can use these to align the alphabets.

Calculate A, B, C, D, E, F

We get

```
A=  (as)(bw)(cr)(dt)(vh)(pl)(fq)(kn)(xu)(gm)(zj)(yi)(oe)
B=  (dk)(ay)(xg)(tc)(bj)(lh)(fn)(qr)(vz)(ei)(ow)(us)(mp)
C=  (ax)(bl)(vh)(ie)(kr)(tz)(ju)(gd)(fo)(cm)(qs)(np)(yw)
D=  (as)(bw)(cr)(ft)(kh)(xl)(gq)(zn)(yu)(om)(dj)(vi)(pe)
E=  (dh)(xy)(tg)(ac)(qn)(vr)(ez)(oi)(uw)(ms)(bp)(lj)(fh)
F=  (co)(qm)(ns)(xp)(aw)(bx)(vl)(ih)(ke)(tr)(jz)(yu)(fd)
```

U, V, W, X, Y, Z

- $A = SPUP^{-1}S^{-1}$ so, $U = P^{-1}S^{-1}ASP^{1}$. This and similar equations yield:
  - $U = P^{-1}S^{-1}ASP^{1}$
  - $V = P^{-2}S^{-1}BSP^{2}$
  - $W = P^{-3}S^{-1}CSP^{3}$
  - $X = P^{-4}S^{-1}DSP^{4}$
  - $Y = P^{-5}S^{-1}ESP^{5}$
  - $Z = P^{-6}S^{-1}FSP^{6}$
- S= (ap)(bl)(cz)(fh)(jk)(qu)

- Putting this all together, we can compute U,V,W,X,Y,Z.

  U=(ax)(bh)(ck)(dr)(ej)(fw)(gi)(lp)(ms)(nz)(oh)(qt)(uy)
  V=(ar)(bv)(co)(dh)(fl)(gk)(iz)(jp)(mn)(qy)(su)(tw)(xe)
  W=(as)(bz)(cp)(dg)(eo)(fw)(gj)(hl)(iy)(kr)(mu)(nt)(vx)
  X=(ap)(bf)(cu)(dv)(ei)(gr)(ho)(jn)(ky)(lx)(mz)(qf)(tw)

Calculate (UV), (VW), (WX)

```
UV= (aepftybsnikod)(rhcgzmuvqwljy)
VW= (ydlwnuakjcevz)(ibxopgrsmtvhq))
WX= (uzftjryehxdsp)(caqvloikgnwbm)
```

(VW)= H^{-1}(UV)H, (WX)= H^{-1}(VW)H. The only consistent value of H satisfying Theorem 1 is

```
H= (ayuricxqmgovskedzplfwtnjhb)
```

Finally, H= NPN^{-1}, so

```
N:  abcdefghijklmnopqrstuvwxyz
    azfpotjyexnsiwkrhdmvclugbq
N= (a)(bzqhy)(cftvlsmieoknwu)(dpr)(gjx)
```

We have our rotor.

Turing Bombe - Introduction

- Assume we know all rotor wirings and the plaintext for some received cipher-text. We do not know plugboard, rotor order, ring and indicator.

- We need a crib characteristic that is plugboard invariant.

```
Position   123456789012345678901234
Plain Text OBERKOMMANDODERWEHRMACHT
CipherText ZMGERFEWMLKMTAWXTSWVUINZ
```

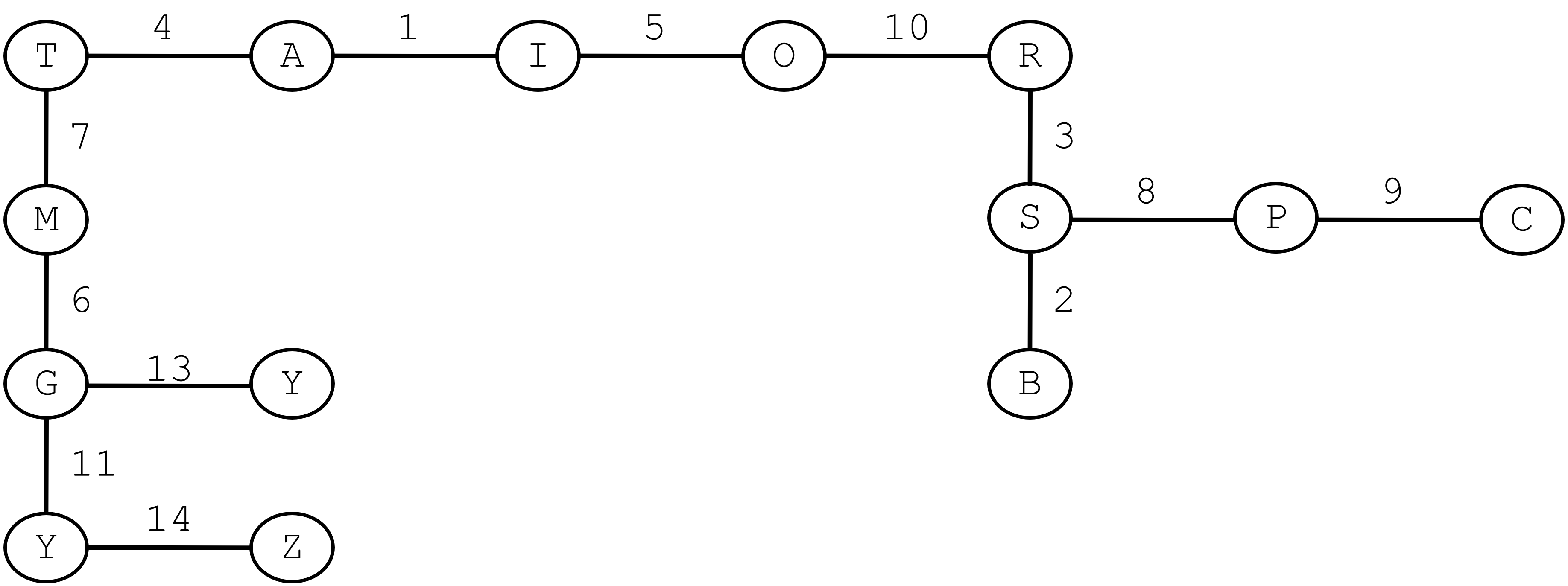
  Observe the loop A[9]→M[7]→E[14]→A.

- If M_i is the effect of the machine at position i and S is the stecker, for the above we have "E"= ("M")SM_7S and ("E")$M_7M_9M_{14}$="E". This return could happen accidently so we use another (E[4]→R[15]→W[8]→M[7]→E) to confirm as ("E")$M_4M_{15}M_8M_7$= "E".

Turing Bombe – the menu

- Want short enough text for no “turnovers”.

```
Position    123456789012345678901234
Plain text  ABSTIMMSPRUQYY
Cipher text ISOAOGTPCOGNYZ
```



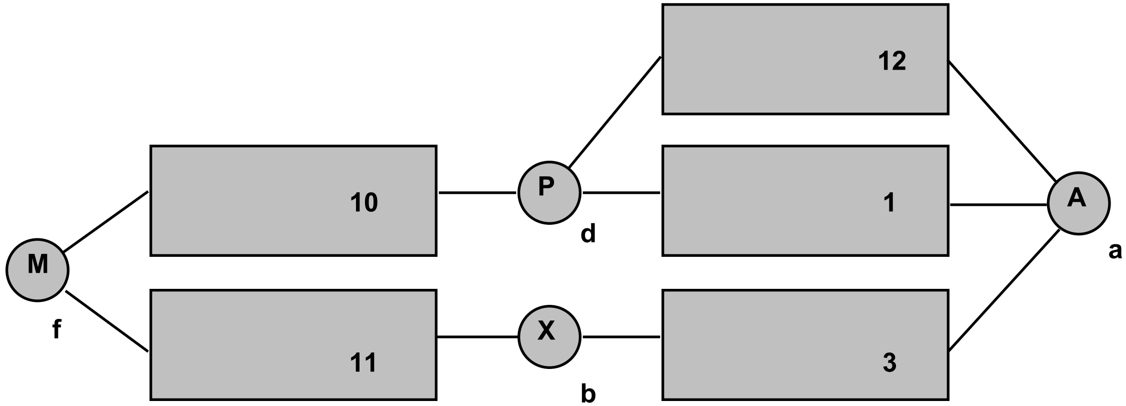
Turing Bombe -1

- Each cycle can be turned into a ring of Enigma machines.
- In a ring of Enigmas, ***all*** the S cancel each other out!
- The key search problem is now reduced from 67.5 to 20 bits !!!!
- At 10 msec/test, 20 bits takes 3 hours.
- Turing wanted ~4 loops to cut down on “false alarms.”
- About 20 letters of “crib” of know plaintext were needed to fine enough loops.
- Machines which did this testing were called “Bombe’s”.
- Built by British Tabulating Machine Company.

Courtesy of Carl Ellison

Test Register in Bombes

- In the diagram below, each circle is a 26-pin connector and each line a 26-wire cable. The connector itself is labeled with a letter from the outside alphabet while its pins are labeled with letters from the inside alphabet. Voltage on X(b) means that **X** maps to **b** through the plugboard.



X: a b c d e f g h i j k l m n o p q r s t u v w x y z

Welchman’s Improvement

- With enough interconnected loops, when you apply voltage to X(b), you will see one of three possibilities on the pins of connector X:

```
01000000000000000000000000
11101111111111111111111111
11111111111111111111111111
```

  - `01000000000000000000000000` **X** maps to **b**
  - `11101111111111111111111111` **X** really maps to **d**
  - `11111111111111111111111111` wrong Enigma key
- Gordon Welchman realized that if X(b) then B(x), because the plugboard was a self-inverse ($S=S^{-1}$).
- His diagonal board wired X(a) to A(x), D(q) to Q(d), etc.
- With that board, the cryptanalyst didn’t need loops -- just enough text
- This cut the size of the required crib in half.

Courtesy of Carl Ellison

Stream Ciphers

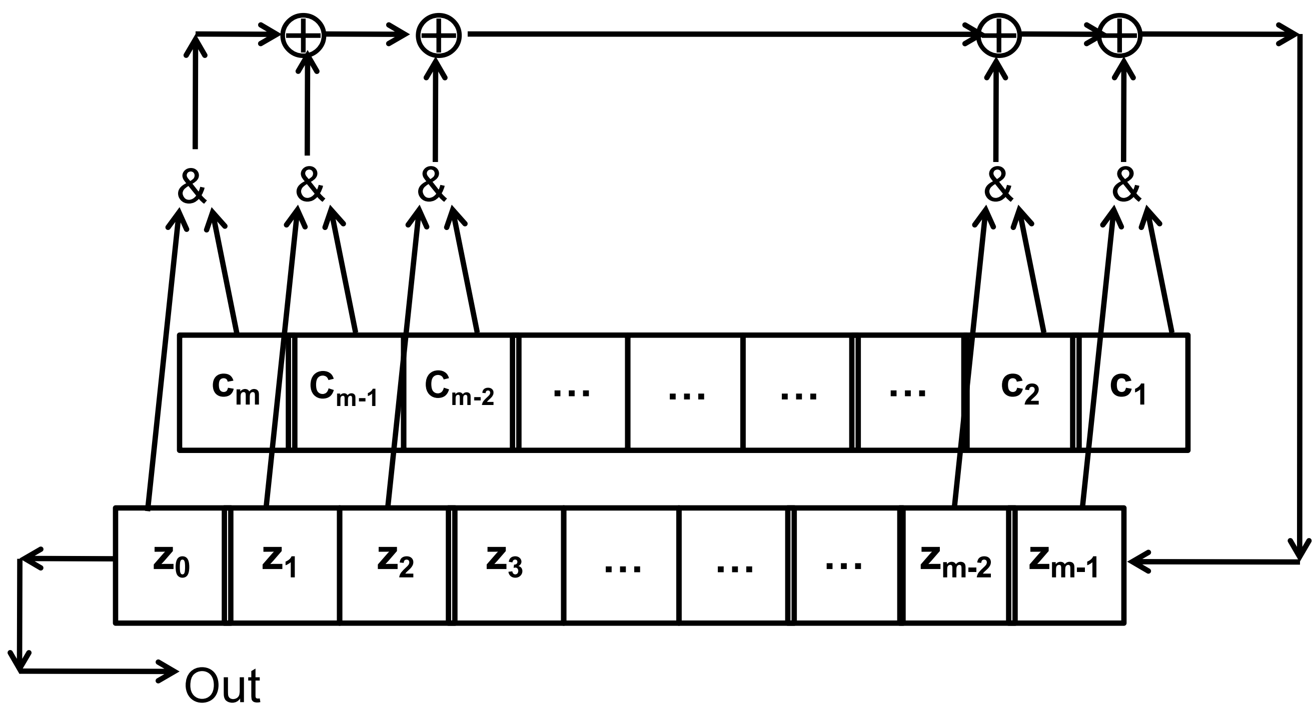
Binary one-time pad

Plaintext $\oplus$ Key = Ciphertext

Ciphertext $\oplus$ Key = Plaintext

10101110011100000101110110110000	*Plaintext*
00101010011010110001010110010111	*Key*
10100100000110110100100000100111	*Ciphertext*
00101010011010110001010110010111	*Key*
10101110011100000101110110110000	*Plaintext*

Linear Feedback Shift Registers (LFSR)



- State at time t: $S(t) = \langle z_0, z_1, \ldots, z_{m-1} \rangle = \langle s_t, s_{t+1}, \ldots, s_{t+m-1} \rangle$.
- Recurrence is $s_{j+1} = c_1 s_j + \ldots + c_m s_{j-m-1}$,
- At time t, LFSR outputs $z_0 = s_t$, shifts, and replaces z_{m-1} with $c_1 z_{m-1} + \ldots + c_m z_0$.

LFSR performance metrics

- The output sequence of and LFSR is periodic for all initial states. The maximal period is 2^m-1.
- A non-singular LFSR with primitive feedback polynomial has maximal period of all non-zero initial states
- A length m LFSR is determined by 2m consecutive outputs
- Linear complexity of sequence z_0, z_1, …, z_n is the length of the smallest LFSR that generates it
- Berlekamp-Massey: $O(n^2)$ algorithm for determining linear complexity

Linear Complexity, simple $O(n^3)$ algorithm

- There is a non-singular LFSR of length m which generates $s_0, s_1, \ldots, s_k \ldots$ iff there are $c_1, \ldots, c_m$ such that:

$$s_{m+1} = c_1 s_m + c_2 s_{m-1} + \ldots + c_m s_1$$
$$s_{m+2} = c_1 s_{m+1} + c_2 s_m + \ldots + c_m s_2$$
$$\ldots$$
$$s_{2m} = c_1 s_{2m-1} + c_2 s_{2m-2} + \ldots + c_m s_{m+1}$$

- To solve for the c_i's just use Gaussian Elimination (see math summary) which is $O(n^3)$.
- But there is a more efficient way!

Example: Breaking a LFSR

- $z_{n+1} = c_1 z_n + \ldots + c_m z_{n-m-1}$. m=8.
- Plain: 1 0 0 1 1 1 0 1 0 1 1 1 0 0 1 0 1 1 1
- Cipher: 1 1 1 1 0 0 1 0 1 0 1 0 1 1 0 0 0 1 0
- LFSR Output: 0 1 1 0 0 1 1 1 1 1 0 1 1 1 1 0 1 0 1

	c_8	c_7	c_6	c_5	c_4	c_3	c_2	c_1	
i	Z_0	Z_1	Z_2	Z_3	Z_4	Z_5	Z_6	Z_7	S_{i+8}
0	0	1	1	0	0	1	1	1	1
1	1	1	0	0	1	1	1	1	1
2	1	0	0	1	1	1	1	1	0
3	0	0	1	1	1	1	1	0	1
4	0	1	1	1	1	1	0	1	1
5	1	1	1	1	1	0	1	1	1
6	1	1	1	1	0	1	1	1	1
7	1	1	1	0	1	1	1	1	0

- GE gives solution $(c_1, c_2, \ldots, c_8)$: 10110011

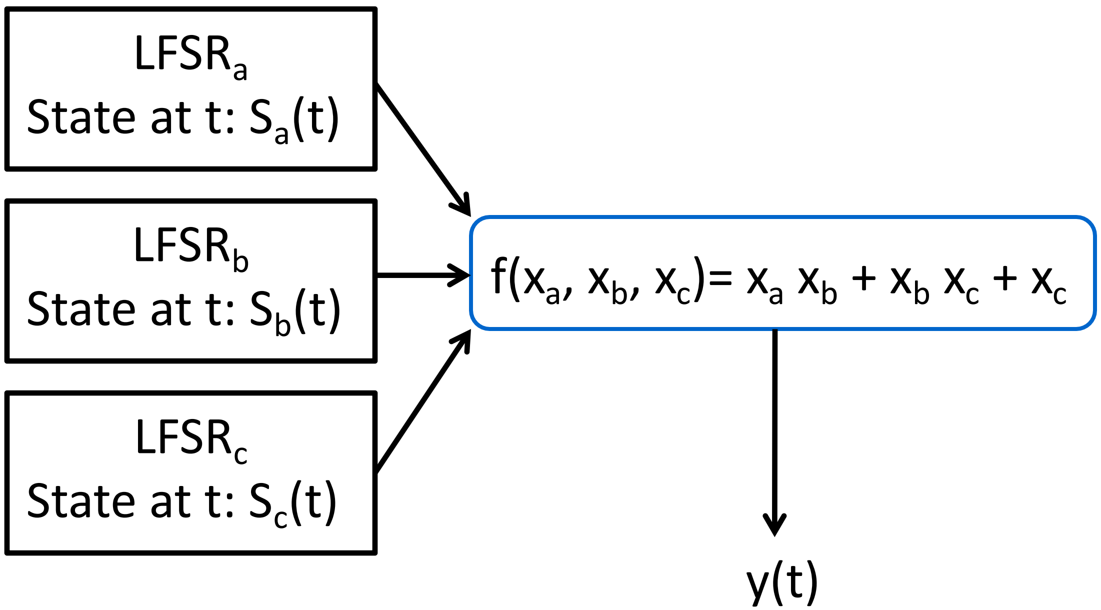
LFSR as linear recurrence

- $G(x)$ is power series representing the LFSR, coefficients are outputs.
- $G(x)= a_0 + a_1 x + a_2 x^2 + \ldots + a_k x^k + \ldots$
- Let $c(x)= c_1 x + \ldots + c_m x^m$.
- Because of the recurrence, $a_{t+m} = \sum_{0<i<m+1} c_i a_{t+m-i}$,
  - $G(x)= a_0 + a_1 x + a_2 x^2 + \ldots + a_{m-1} x^{m-1} + x^m (c_1 a_{m-1} + \ldots+c_m a_0)+ x^{m+1} (c_1 a_m + \ldots+c_m a_1)+ x^{m+2} (c_1 a_{m+1} + \ldots+c_m a_2)+\ldots$
  - After some playing around, this can be reduced to an equation of the form $G(x)= K/(1-c(x))$, where K is a constant that depends on initial state only. Let $f(x)= 1-c(x)$ be the called the connection polynomial. [$1-c(x)=1+c(x) \pmod 2$, of course].
  - If the period of the sequence is p, $G(x)= (a_0 + a_1 x + \ldots + a_{p-1} x^{p-1})+ x^p(a_0 + a_1 x + \ldots + a_{p-1} x^{p-1}) + \ldots= (a_0 + a_1 x + \ldots + a_{p-1} x^{p-1})(1+x^p +x^{2p} + \ldots)$
- We get $(a_0 + a_1 x + \ldots + a_{p-1} x^{p-1})/(1-x^p)= K/(f(x))$ so $f(x) \mid 1-x^p$ and $f(x)$ is the equation for a root of 1. If $f(x)$ is a primitive root of 1 p will be as large as possible, namely, $p=2^m-1$.

Geffe Generator

- Three LFSRs of maximal periods (2^a-1), (2^b-1), (2^c-1) respectively.
- Output filtered by $f(x_a, x_b, x_c)= x_a x_b + x_b x_c + x_c$
- Period: $(2^a-1)(2^b-1)(2^c-1)$
- Linear complexity: ab+bc+c
- Simple non-linear filter.

Geffe Generator



x_a	x_b	x_c	$f(x_a, x_b, x_c)$
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

- Note that x_c and $f(x_a, x_b, x_c)$ agree 75% of the time.

Correlation attack: breaking Geffe

- Guess $S_c(0)$ and check the agreement of $S_c(t)_{out}$ and $y(t)$.
  - If guess is right, they will agree much more often than half the time
  - If guess is wrong, they will agree about half the time
  - In this way, we obtain $S_c(0)$.
- Now guess $S_b(0)$.
  - Compare $y(t)$ and $x_a\, S_b(t)_{out}+S_b(t)_{out}S_c(t)_{out}+S_c(t)_{out}$.
  - If guess is right, they will agree much more often than half the time.
  - If not, they will agree about half the time.
  - In this way, we obtain $S_b(0)$.
- Now guess $S_a(0)$.
  - $y(t)$ and $S_a(t)\, S_b(t)_{out}+S_b(t)_{out}\, S_c(t)_{out}+S_c(t)_{out}$ will be the same as $y(t)$ for the correct guess.
- Complexity of attack (on average) is about $2^{a-1}+ 2^{b-1}+ 2^{c-1}$ rather than about $2^{a+b+c-1}$ which is what we'd hoped for.

Berlekamp-Massey

- Given output of LFSR, $s_0, s_1, \ldots, s_{N-1}$, calculate length, L, of smallest LFSR that produces $<s_i>$. Algorithm below is $O(n^2)$. In the algorithm below, the connection polynomial is: $c(x) = c_0 + c_1 x + \ldots + c_L x^L$ and c_0=1 always.

```
c(x)=1; L= 0; m=-1; b(x)=1;
for(n=0; n<N; n++)
    d= sn + ∑i=1^L-1 ci sn-i     //  d is the "discrepency"
    if(d!=0) {
        t(x)= c(x);
        c(x)= c(x) + b(x) x^(n-m);
        if(L<=n/2)) {
            L=n+1-L;
            m= n;
            b(x)= t(x);
            }
    }
}
```

$$d = s_n + \sum_{i=1}^{L-1} c_i\, s_{n-i}, \qquad c(x) = c(x) + b(x)\, x^{n-m}$$

Berlekamp-Massey example

- $s_0, s_1, \ldots, s_{N-1}$ = 001101110, N=9

n	s_n	t(x)	c(x)	L	m	b(x)	d
-	-	-	1	0	-1	1	-
0	0	-	1	0	-1	1	0
1	0	-	1	0	-1	1	0
2	1	1	$1+x^3$	3	2	1	1
3	1	$1+x^3$	$1+x+x^3$	3	2	1	1
4	0	$1+x+x^3$	$1+x+x^2+x^3$	3	2	1	1
5	1	$1+x+x^2+x^3$	$1+x+x^2$	3	2	1	1
6	1	$1+x+x^2+x^3$	$1+x+x^2$	3	2	1	0
7	1	$1+x+x^2$	$1+x+x^2+x^5$	5	7	$1+x+x^2$	1
8	0	$1+x+x^2+x^5$	$1+x^3+x^5$	5	7	$1+x+x^2$	1

Linear complexity and linear profile

- "Best" (i.e.-highest) linear complexity for $S_N = s_0, s_1, \ldots, s_{N-1}$ is L=N/2.
- Complexity profile for S is the sequence of linear complexities $L_1, L_2, \ldots, L_{N-1}$ for $S_1, S_1, \ldots, S_N$.
- For a "strong" shift register, we want not just large L but large L_k for subsequences (thus hug the line L= N/2).
- $E(L(< s_0, s_1, \ldots, s_{N-1}>)) = N/2 + (4+(\sum_{i=0}^{N-1} s_i) \pmod 2)/18 - 2^{-N}(N/3+2/9)$

Shrinking Generator

- Two LFSRs of maximal periods (2^s-1), (2^a-1) respectively. $(a,s)=1$.
- Output is output of A clocked by S.
- Period: $(2^{s-1}-1)(2^a-1)$.
- Linear Complexity: $a2^{s-2}<c<a2^{s-1}$
- SEAL cipher from Coppersmith.

RC4 Initialization

- Array `key` contains N bytes of key
- Array `S` always has a permutation of 0,1,…,255

```
for i = 0 to 255
        S[i] = i
        K[i] = key[i (mod N)]
next i
j = 0
for i = 0 to 255
        j = (j + S[i] + K[i]) (mod 256)
        swap(S[i],S[j])
next i
i = j = 0
```

Slide adapted from Mark Stamp

RC4 Keystream

- For each keystream byte, swap elements of array `S` and select a byte from the array:

```
i = (i + 1) (mod 256)
j = (j + S[i]) (mod 256)
swap(S[i], S[j])
t = (S[i] + S[j]) (mod 256)
keystreamByte = S[t]
```

- Use keystream bytes like a one-time pad
  - XOR to encrypt or decrypt

Slide adapted from Mark Stamp

End

RC4 Weakness

- RC4 Weakness: Let S_i *be the state at time i, N = 2n (n = 8, usually). Let* $\langle z_i \rangle$ *be the output sequence.*

  $P(z_2) = 0) = 2N$.

  Proof: Suppose $S_0[2] = 0$, $S_0[1] \neq 2$, $S_0[1] = X$, $S_0[X] = Y$.

  Round 1: $i = 1$, $X = S_0[1] + 0$. *Exchange* $S_0[1]$ *and* $S_0[Y]$.

  Round 2: $i = 2$, $j = X + S_1[2] = X$, *Output* $S_1[S_1[2] + S_1[X]] = S_1[X] = 0$. *So* $P(z_j = 0) \approx 1$

  $N + 1 N (1 - 1N) \approx 2$

  N *. So by Bayes, if* $z_2 = 0$*, we can extract* byte of state with probability 1/2 .

Slide adapted from Mark Stamp

Enigma: method of batons

- Encryption equation with no Stecker: $c = (p)R_1{}^{i}R_2{}^{j}\ R_3{}^{k}\mathrm{Z}R_3{}^{-k}\ R_2{}^{-j}\ R_1{}^{-i}$
- Trial encrypt plaintext with fast rotor in each possible position.
- Correct position will produce isomorphic text.

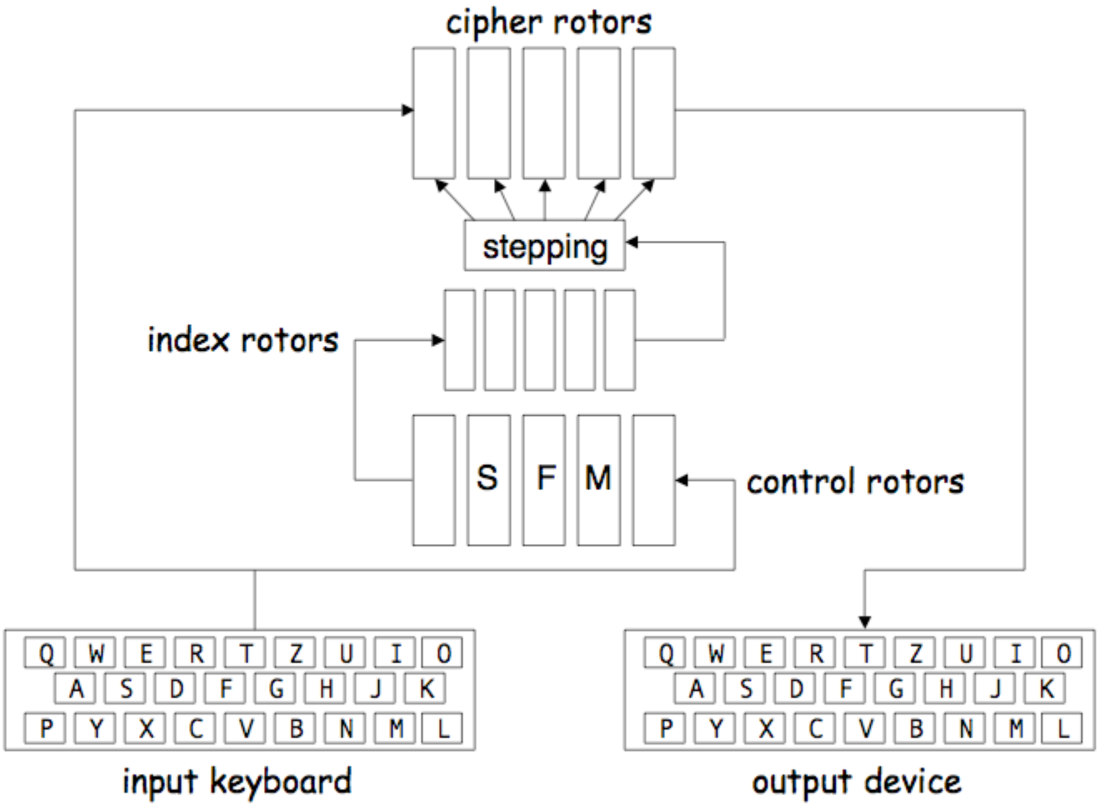
Visualizing rotor motion

	step	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
	0	E	K	M	F	L	G	D	Q	V	Z	N	T	O	W	Y	H	X	U	S	P	A	I	B	R	C	J
	1	J	L	E	K	F	C	P	U	Y	M	S	N	V	X	G	W	T	R	O	Z	H	A	Q	B	I	D
	2	K	D	J	E	B	O	T	X	L	R	M	U	W	F	V	S	Q	N	Y	G	Z	P	A	H	C	I
	3	C	I	D	A	N	S	W	K	Q	L	T	V	E	U	R	P	M	X	F	Y	O	Z	G	B	H	J
	4	H	C	Z	M	R	V	J	P	K	S	U	D	T	Q	O	L	W	E	X	N	Y	F	A	G	I	B
R	5	B	Y	L	Q	U	I	O	J	R	T	C	S	P	N	K	V	D	W	M	X	E	Z	F	H	A	G
O	6	X	K	P	T	H	N	I	Q	S	B	R	O	M	J	U	C	V	L	W	D	Y	E	G	Z	F	A
T	7	J	O	S	G	M	H	P	R	A	Q	N	L	I	T	B	U	K	V	C	X	D	F	Y	E	Z	W
O	8	N	R	F	L	G	O	Q	Z	P	M	K	H	S	A	T	J	U	B	W	C	E	X	D	Y	V	I
R	9	Q	E	K	F	N	P	Y	O	L	J	G	R	Z	S	I	T	A	V	B	D	W	C	X	U	H	M
	10	D	J	E	M	O	X	N	K	I	F	Q	Y	R	H	S	Z	U	A	C	V	B	W	T	G	L	P
I	11	I	D	L	N	W	M	J	H	E	P	X	Q	G	R	Y	T	Z	B	U	A	V	S	F	K	O	C
	12	C	K	M	V	L	I	G	D	O	W	P	F	Q	X	S	Y	A	T	Z	U	R	E	J	N	B	H
	13	J	L	U	K	H	F	C	N	V	O	E	P	W	R	X	Z	S	Y	T	Q	D	I	M	A	G	B
	14	K	T	J	G	E	B	M	U	N	D	O	V	Q	W	Y	R	X	S	P	C	H	L	Z	F	A	I
	15	S	I	F	D	A	L	T	M	C	N	U	P	V	X	Q	W	R	O	B	G	K	Y	E	Z	H	J
	16	H	E	C	Z	K	S	L	B	M	T	O	U	W	P	V	Q	N	A	F	J	X	D	Y	G	I	R
	17	D	B	Y	J	R	K	A	L	S	N	T	V	O	U	P	M	Z	E	I	W	C	X	F	H	Q	G
	18	A	X	I	Q	J	Z	K	R	M	S	U	N	T	O	L	Y	D	H	V	B	W	E	G	P	F	C
	19	W	H	P	I	Y	J	Q	L	R	T	M	S	N	K	X	C	G	U	A	V	D	F	O	E	B	Z
	20	G	O	H	X	I	P	K	Q	S	L	R	M	J	W	B	F	T	Z	U	C	E	N	D	A	Y	V
	21	N	G	W	H	O	J	P	R	K	Q	L	I	V	A	E	S	Y	T	B	D	M	C	Z	X	U	F
	22	F	V	G	N	I	O	Q	J	P	K	H	U	Z	D	R	X	S	A	C	L	B	Y	W	T	E	M
	23	U	F	M	H	N	P	I	O	J	G	T	Y	C	Q	W	R	Z	B	K	A	X	V	S	D	L	E
	24	E	L	G	M	O	H	N	I	F	S	X	B	P	V	Q	Y	A	J	Z	W	U	R	C	K	D	T
	25	K	F	L	N	G	M	H	E	R	W	A	O	U	P	X	Z	I	Y	V	T	Q	B	J	C	S	D
		a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z

Hebern machines

- **Hebern equation:** $c = R_5[p_5(t)]R_4[p_4(t)]R_3[p_3(t)]R_2[p_2(t)]R_1[p_1(t)](p)$
  - For idealized Hebern: $p_1(t) = t, p_2(t) = \left[\frac{t}{26}\right], p_3(t) = [\frac{t}{676}]$
- **Idea:** If a plaintext is enciphered several times with the fast rotor in the same position, the resulting ciphertexts are isomorphic, that is, the ciphertexts transform into each other using monoalphabetic substitution. Same is true for same ciphertext decrypted with fast rotor in same position.
- Assumption: Suppose we know wiring of fast rotor. We can trial encipher plaintext to obtain an isomorphic ciphertext.
  - We can figure out first rotor wiring from a lot of ciphertext
- If a plaintext has no repeated letters, the number of ways two strings can be enciphered is $(\prod_{i=0}^{n-1} \frac{26-i}{26})^2$. There are fewer ways to do this with when characters are repeated.

Sigaba Wiring Diagram



- Control and index rotors determine stepping of cipher rotors

M209

- Also called C-48
- Six wheels with $(26, 25, 23, 21, 19, 17)$ pins respectively
- Cage with 27 bars. Each bar has one or two lugs. Each lug can be positioned at one of the six wheel positions
- Each of the 131 pins can be in one of two positions, “active” or “inactive.”
- “Keyword” that sets offset of each wheel at start, usually sent with message.

M209

- Machine generates keystream $< k_i >$.
- Plain-text $< p_i >$ generates cipher-text $< c_i >$ where $c_i = 27 + k_i - p_i \ (mod\ 26)$. Decryption is reciprocal.
- When an "active pin" comes into contact with a lug, it contributes 1 to the keystream. For each letter, each of the 27 bars rotates to contact the current pins at each current wheel offset.
- After each letter is enciphered, each wheel advances one position.
- For example, if $p_1 = 1(A)$, and $k_1 = 9$, $c_1 = 35 = 9\ (mod\ 26)\ or\ I$
- The "current" wheel position can present from from 0 to 6 active pins (i.e.- 64 possible states)
- In the following example, we specify the lug settings on each bar.

M209

- Consider the lug settings, where “1” indicates lug presence

Bar	Lugs	Bar	Lugs	Bar	Lugs
1	100000	10	001000	19	000010
2	100000	11	000110	20	000010
3	100000	12	000110	21	000010
4	100000	13	000010	22	000010
5	100000	14	000010	23	000011
6	100000	15	000010	24	000001
7	100000	16	000010	25	000001
8	110000	17	000010	26	000001
9	010010	18	000010	27	000001

- With these settings, wheel 1 can contribute up to 8 ticks, wheel 2, 2 ticks, wheel 3, 1 tick, wheel 4, 3 ticks, wheel 5 8 ticks and wheel 6, 4 ticks, for a maximum count of 26.

M209

- If the active pins are 101101 at the initial position, for example, $k_1 =$ 8+1+3+4=16.
- The “key” settings consist of the lug positions and the active pin settings on each of the six wheels.
- If we are given corresponding known and cipher text, we are essentially given the keystream $k_i, i = 1, \dots, n$. There is some ambiguity because 0= 26 (mod 26) so a “displacement of 0 and 26 are cryptographically indistinguishable.
- The analysis problem is: Given $k_i, i = 1, \dots, n$ and the starting wheel positions, determine the lug positions on each bar and the active pins on each wheel.

M209 example

M209 simulator

27 lugs in total

Machine state:

wheel 1: 01010101000010100101010001

wheel 2: 0000110000101100110000101

wheel 3: 00111000001101100011010

wheel 4: 101011101010110010101

wheel 5: 0101000001001100010

wheel 6: 11000101100001101

lugs on wheel: 04 05 03 07 02 06

wheel positions: 0 0 0 0 0 0

keystream: 13 12 10 09 15 22 07 10 13 02 15 03 18 23 13 00 18 17 16 07 09 14 13 07 11 20 10 14 02 16 11 19 02 20 16 11 05 15 09 15 04 23 21 04 10 04 07 24 16 08 18 12 13 04 12 18 09 07 15 13 17 00 12 12 13 16 11 21 13 19 02 14 08 20 07 13 13 06 00 21 08 17 09 14 13 17 13 08 17 09 20 05 22

M209 example

Message is 93 letters long

Plain :

HELLOTHERETHISISAMUCHLONGERMESSAGEFORBILLYFRIEDMANABRAHAMSIN
KOVSOLOMONKULLBACKANDFRANKROWLETT

Cipher :

GIZYBDAGWYWWKFFISFWFCDZUFQTCYYTTWQLXOOBETZQNCAEMQVSLWEFSX
PHAHMRUZFXJZGSUXJGNLWAIFMSOAHWUVYQMD

Decrypted:

HELLOTHERETHISISAMUCHLONGERMESSAGEFORBILLYFRIEDMANABRAHAMSIN
KOVSOLOMONKULLBACKANDFRANKROWLETT

M209 –basic attack

- Consider wheel 1 which returns to its starting position every 26 letters of key stream.
- Write the keystream in columns as

$$
\begin{array}{lllll}
k_{26\times0+1} & k_{26\times1+2} & k_{26\times0+3} & \ldots & k_{26\times0+26} \\
k_{26\times1+1} & k_{26\times1+2} & k_{26\times1+3} & \ldots & k_{26\times1+26} \\
\ldots & \ldots & \ldots & \ldots & \ldots \\
k_{26\times m+1} & k_{26\times m+2} & k_{26\times m+3} & \ldots & k_{26\times m+26}
\end{array}
$$

- For simplicity we assume 26(m+1)=n, the size of the known keystream
- If we average each column, a column with a high number of lugs in position 1 will contribute a larger count (by the number of lugs) when the pin at position 1 is active that when it is inactive.

M209 –basic attack

- The resulting averages, after binning, with be distributed bimodally and difference between the values of the peaks of the bins will be close to the number of lugs in position 1 corresponding to wheel 1. Further, a high average at an offset indicates that the pin is active at that offset.
- This lets us determine the number of lugs at each wheel position and the active pin settings on each wheel. It also allows us to disambiguate the 26, 0 coincidence. If we compute all these averages for each wheel (with periods 26, 25, 23, 19, 17, depending on the wheel), the wheels with the most pronounced bimodal peaks will have the largest number of lugs.
- After "guessing" the active pins (based on the larger keystream values in a column) and the number of lugs (based on the difference in the difference between the peaks), we can subtract out the effect of the wheel to make the effect of the other wheels more obvious.
- This "trial and error" guessing and correcting inconsistency determines the lugs and pins when $n \sim 250$

M209 – basic attack

- Here is what the initial data might look like, when we pick the “obviously” bimodal spread at wheel 4:

```
M209 analysis, key length is 250
Global average:   12.09
Wheel 4
  pin  0:   14.50
  pin  1:    8.08
  pin  2:   14.58
  pin  3:    8.58
  pin  4:   14.83
  pin  5:   14.75
  pin  6:   15.58
  pin  7:    8.33
  pin  8:   15.42
  pin  9:    6.42
  pin 10:   16.33
  pin 11:    7.42
  pin 12:   16.00
  pin 13:   14.25
  pin 14:    8.50
  pin 15:    7.67
  pin 16:   16.08
  pin 17:    7.58
  pin 18:   15.25
  pin 19:    8.36
  pin 20:   15.27
```

M209 – basic attack

- After binning:

```
 0:
 1:
 2:
 3:
 4:
 5:
 6: x
 7: xxx
 8: xxxxx
 9:
10:
11:
12:
13:
14: xxxxx
15: xxxx
16: xxx
17:
18:
19:
20:
```

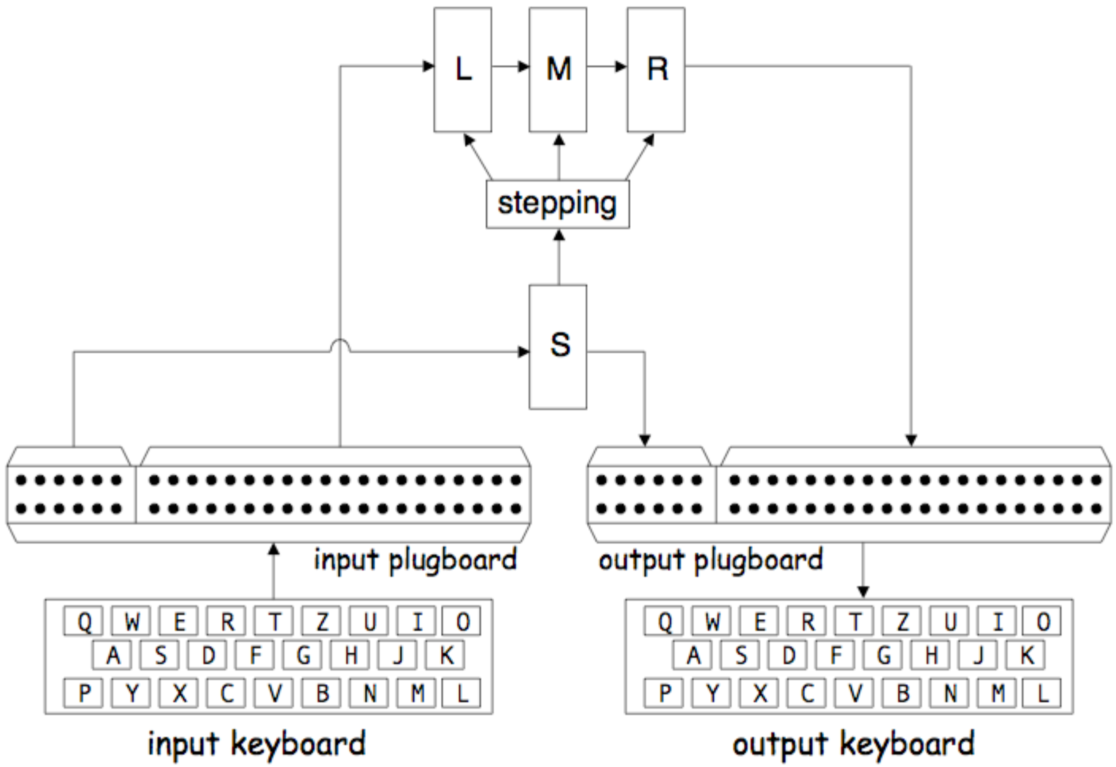
- This indicates that there are about 6 lugs in wheel position 4 and pins *0,2,4,5,6,8,10,12,13,16,18,20* are likely active while the remaining pins are not.
- This is right, almost. There are actually 7 lugs at wheel 4.

M209 cipher-text only cryptanalysis

- We can extend this to a ciphertext only attack.
- Let $f_{i\lambda}$ be the number of occurrences of cipher-text letter λ with pin position i.
- Hypothesis $f_{i\lambda} = x_i c_\lambda + (1 - x_i) b_\lambda$ where c_λ and b_λ are ideal frequency distributions. $x_i = 1$ if pin i is in class 1 and $x_i = 0$ if pin i is in class 0. Since the distributions are not ideal, x_i will become a fraction.
- We optimize $\Phi = \sum_{i=1}^{N} \sum_{\lambda=0}^{L-1} (f_{i\lambda} - x_i c_\lambda + (1 - x_i) b_\lambda)^2$ by least squares.
- $\frac{\partial \Phi}{\partial b_\lambda} = -2 \sum_{i=1}^{N} f_{i\lambda} - x_i a_\lambda + b_\lambda = 0$ and $b_\lambda = \sum_{i=1}^{N} \frac{f_{i\lambda}}{N} - a_\lambda \sum_{i=1}^{N} \frac{x_i}{N}$, $a_\lambda = c_\lambda - b_\lambda$
- Putting $g_{i\lambda} = f_{i\lambda} - \sum_{i=1}^{N} \frac{f_{i\lambda}}{N}$, $\Phi = \sum_{i=1}^{N} \sum_{\lambda=0}^{L-1} (g_{i\lambda} - x_i a_\lambda + \ a_\lambda \sum_{j=1}^{N} \frac{x_j}{N})^2$
- Now put $y_i = x_i - \sum_{j=1}^{N} \frac{x_j}{N}$ and now $\Phi = \sum_{i=1}^{N} \sum_{\lambda=0}^{L-1} (g_{i\lambda} - y_i a_\lambda)^2$
- Eigenvector with the largest eigenvalue corresponding to $\gamma_{ij} = \sum_{\lambda=0}^{L-1} g_{i\lambda} g_{j\lambda}$
- This requires a longer ciphertext

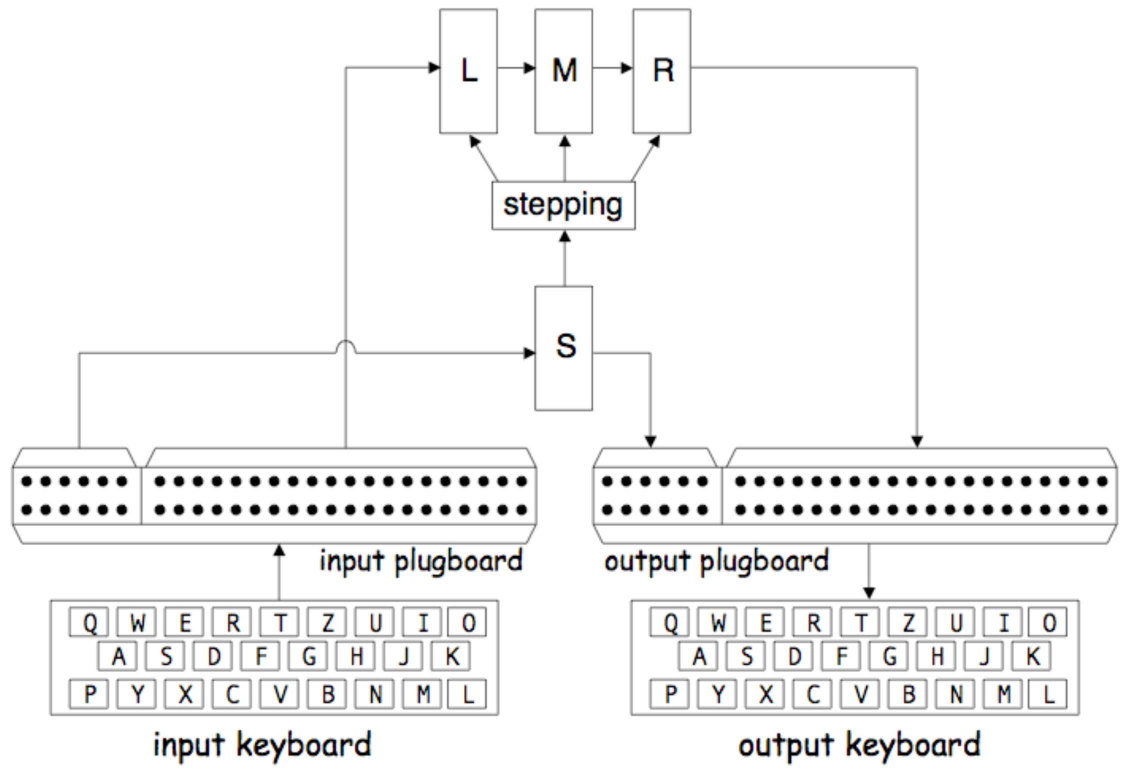
Purple

- Input letter permuted by plugboard.
- Vowels and consonants sent thru different switches.
- The “6-20 split”



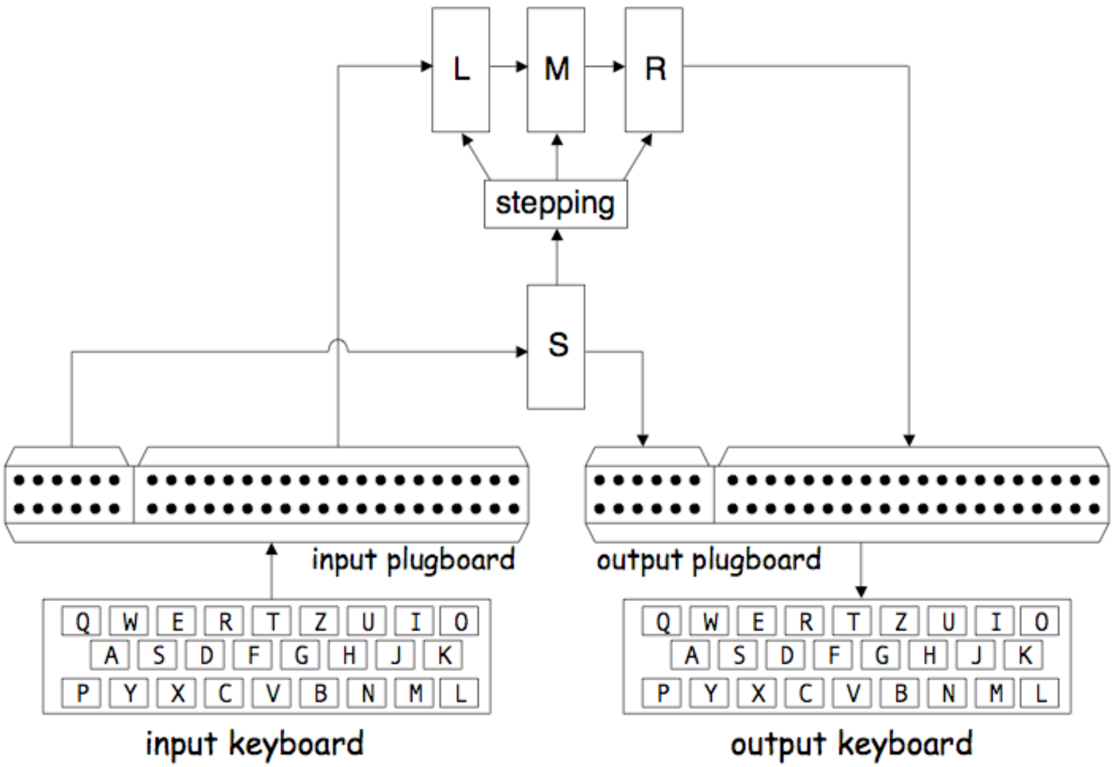
Purple

- Switch S
  - Steps once for each letter typed
  - Permutes vowels
- Switches L,M,R
  - One of these steps for each letter typed
  - L,M,R stepping determined by S

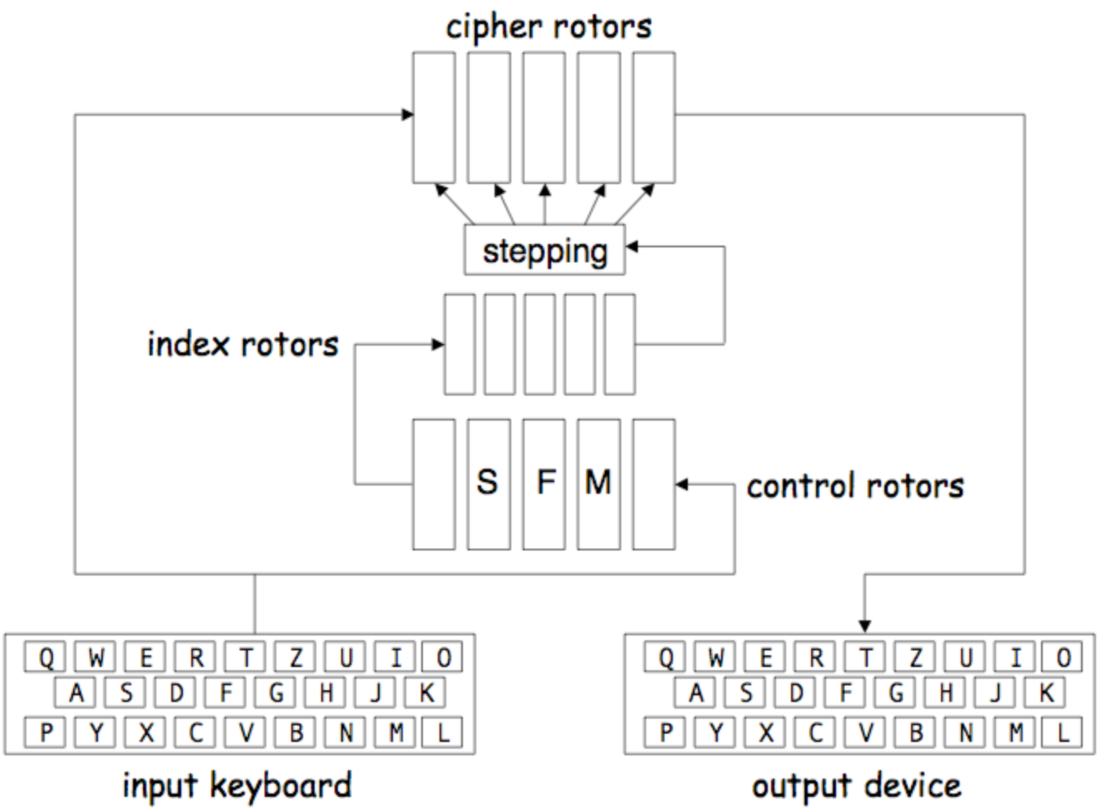


Purple

- Switched permutations
  - **Not** rotors!!!
- S,L,M, and R are switches
  - Each step, one of the perms switches to a different permutation



Sigaba



From Stamp

Sigaba Wiring Diagram

- Index rotors do not step
- Control rotors
  - Middle 3 step as: slow, fast, medium
  - Outside rotors don’t step
- Cipher rotors
  - At least 1, at most 4 step each time
- When a letter is typed
  - 4 inputs to **control rotors** activated
    - These are: F,G,H,I
    - Then 4 (scrambled) letters output
  - Outputs of control rotors combined
    - Then fed into index rotors
  - Outputs of control rotors combined
    - Result(s) go into index rotors
    - From 1 to 4 inputs to index rotors
  - Index rotor outputs combined in pairs
    - Active index rotor outputs determine which cipher rotors step

From Stamp

Cipher Rotor Stepping

- F,G,H,I input to control rotors
- Let $I_0, I_1, \ldots, I_9$ be inputs to index rotors
  - I_0 is always inactive, and…
  - A,B,C,…, Z are control rotor **outputs**

$I_1 = B$	$I_2 = C$	$I_3 = D \vee E$
$I_4 = F \vee G \vee H$	$I_5 = I \vee J \vee K$	$I_6 = L \vee M \vee N \vee O$
$I_7 = P \vee Q \vee R \vee S \vee T$	$I_8 = U \vee V \vee W \vee X \vee Y \vee Z$	$I_9 = A$

- Let $O_0, O_1, \ldots, O_9$ be index rotor outputs
  - If $C_i == 1$, cipher rotor i steps
  - Cipher rotors numbered left-to-right

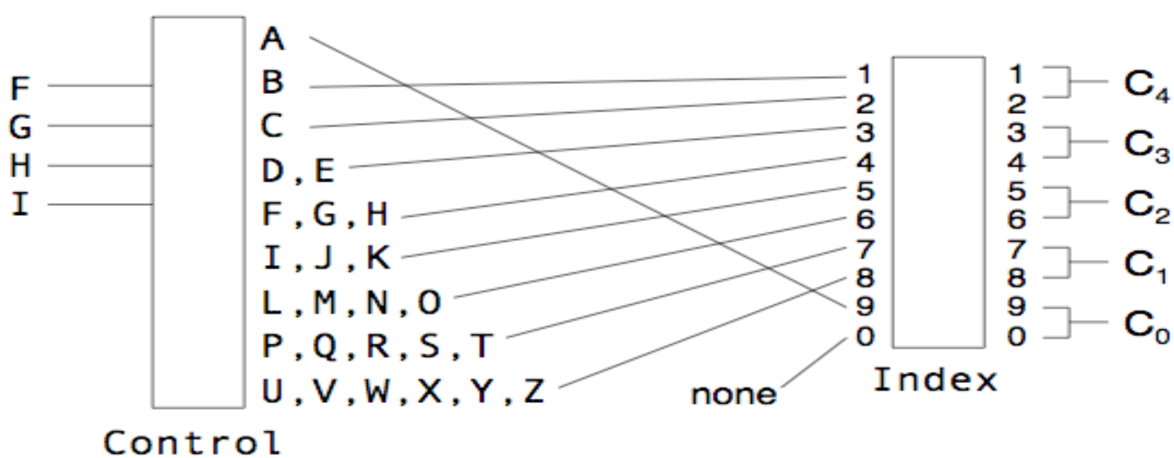
$C_0 = O_0 \vee O_9$	$C_1 = O_7 \vee O_8$	$C_2 = O_5 \vee O_6$	$C_3 = O_3 \vee O_4$	$C_4 = O_1 \vee O_2$

- Note that 1 to 4 of the O_i are active
- Implies that 1 to 4 of C_i are active

From Stamp

Stepping Maze

- If cipher/control rotors all set to “A”
- And index rotors all set to “0”
  - Select 5 cipher rotors: $(26!)^5 = 2^{442}$
  - Select 5 control rotors: $(26!)^5 = 2^{442}$
  - Select 5 index rotors: $(10!)^5 = 2^{109}$
- Keyspace is enormous: 993 bits!
- Model control permutations as random
- Probabilities of the C_i are not uniform



From Stamp

Sigaba Attack

- Assumptions
  - Full 95.6 bit WWII key-space is used
  - Trudy has a Sigaba machine (so Trudy knows rotor permutations)
  - Trudy has some known plaintext
  - Goal: use as little-known plaintext as possible
- **Primary phase**
  - Find all cipher rotor settings that are consistent with known plaintext
  - Requires some amount of known plaintext
- **Secondary phase**
  - Find control rotor settings, index perm
  - May require more known plaintext

From Stamp

Primary Phase

- Select and initialize cipher rotors
  - Binomial(10,5) · 5! · 2^5 · 26^5 = $2^{43.4}$ settings
- For each of these $2^{43.4}$ settings, how many cipher perms are possible at next step?
  - From 1 to 4 cipher rotors can step
  - 30 ways that 1 to 4 (out of 5) rotors can step
- Given a putative cipher rotor setting…
- Check whether it matches known plaintext. If not, discard it
- If a match, try all 30 steps and keep any that match with next known plaintext
- Repeat until either
  - No matches (putative setting is discarded)
  - Used all known plaintext (save for secondary)

From Stamp

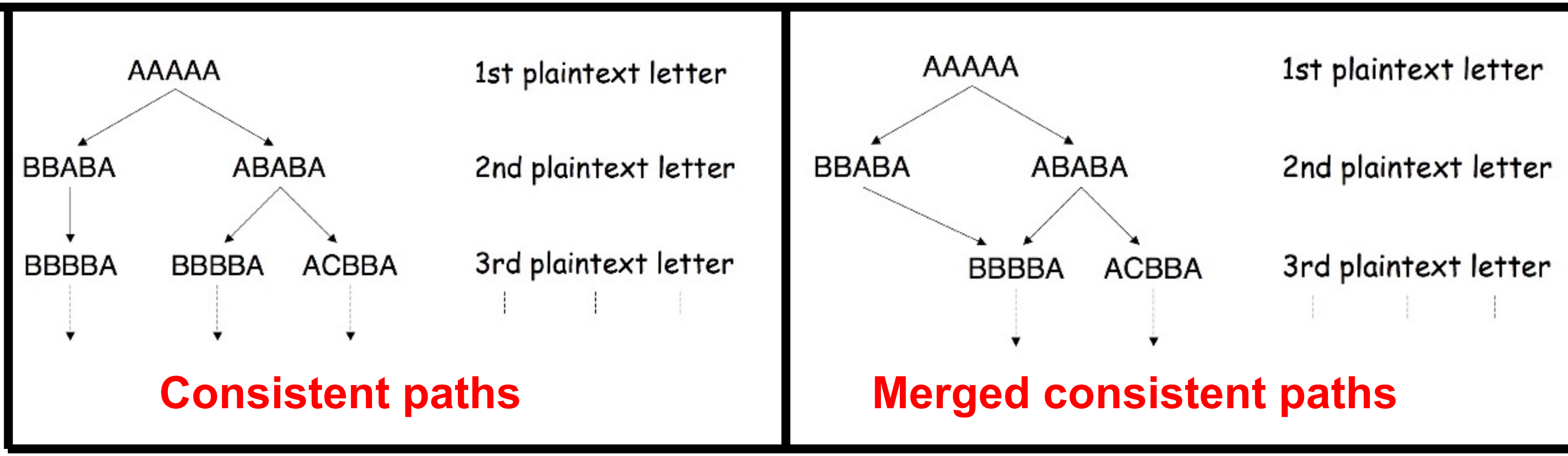
Primary Phase

- Correct rotor setting is said to be **causal**
  - All incorrect settings are **random**
- First letter matches with probability 1/26
- If it matches, then must try all 30 steps
  - Each matches with probability 1/26
  - Binomial distribution with $p = 1/26$ and $n = 30$
- Expected matches is $30/26 = 1.154$
- If first letter matches, then after n steps, we expect about $(1.154)^n$ surviving paths
- If first plaintext matches, about $(1.154)^n$ surviving paths after n steps
- We want to eliminate putative cipher rotor settings that are incorrect
  - The random settings
- How to do this when we get more paths!

From Stamp

Primary Phase: Merging Paths

- Suppose first 3 plaintext match
  - With initial settings AAAAA



- Can merge paths since only the rotor settings (not path) needed for next step

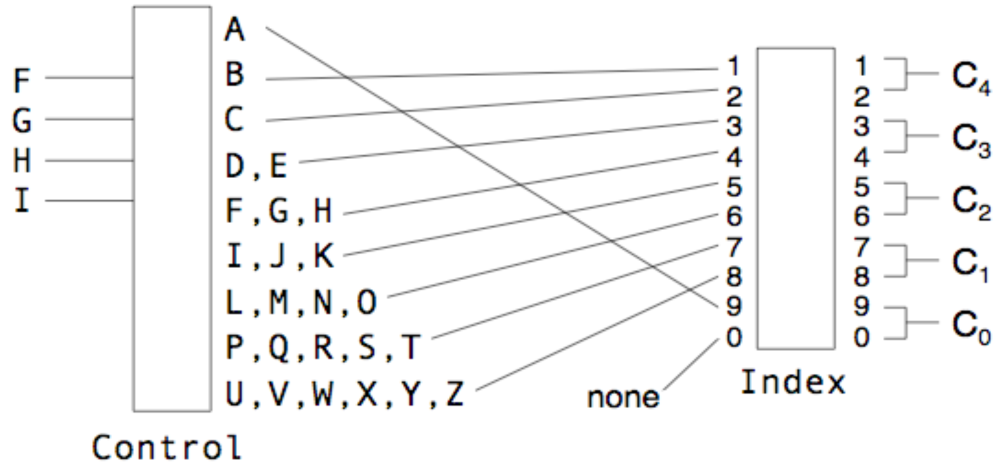
From Stamp

Secondary Phase

- Discussion here applies to each primary phase survivor
- Each primary survivor gives putative cipher rotors and settings
- Obvious secondary test is to…
  - Try all control and index settings: $10!/32 \cdot 5! \cdot 2^5 \cdot 26^5 = 2^{52.2}$ of these
  - Work of more than **2^{52} per primary survivor**
- Primary work is about 2^{43}
  - With about 2^{20} survivors
- Obvious secondary has total work about 2^{72}
  - Since 2^{52} work for each of 2^{20} survivors
- Can we improve secondary phase?
- Cipher rotor motion is not uniform
  - Recall that from 1 to 4 steps for each letter
  - Also, index permutation is fixed for a message

From Stamp

Example



- Consider index perm 0123456789 → 5479381026
  - C_4 connected to 10 control letters
  - C_2 connected to 1 control letter
- C_4 will step much more frequently than C_2

From Stamp

Index Perm Inputs

number of letters	count	pairs
1	3	(0,1) (0,2) (0,9)
2	4	(0,3) (1,2) (1,9) (2,9)
3	5	(0,4) (0,5) (1,3) (2,3) (3,9)
4	7	(0,6) (1,5) (2,5) (5,9) (1,4) (2,4) (4,9)
5	6	(0,7) (1,6) (2,6) (6,9) (3,4) (3,5)
6	6	(0,8) (1,7) (2,7) (7,9) (3,6) (4,5)
7	6	(1,8) (2,8) (8,9) (3,7) (4,6) (5,6)
8	3	(3,8) (4,7) (5,7)
9	3	(4,8) (5,8) (6,7)
10	1	(6,8)
11	1	(7,8)

- Can use this table to determine input pairs
  - Count number of times each cipher rotor steps (using the known plaintext)

From Stamp

Improved Secondary

- About 100 to 200 known plaintexts
- Reduces number of index perms to……about 2^7
- This reduces secondary work from $10!/32 \cdot 5! \cdot 2^5 \cdot 26^5 = 2^{52.2}$ to about $2^7 \cdot 5! \cdot 2^5 \cdot 26^5 = 2^{42.4}$
- Note: this work is per primary survivor
- WWII Sigaba had a readily available keyspace of 95.6 bits
- Our attack has work factor of about 2^{60} under reasonable assumptions
- Sigaba as generally used in WWII had exhaustive key search work of $2^{47.6}$

From Stamp