Modern Cryptanalysis of Historical Ciphers

November 1, 2019

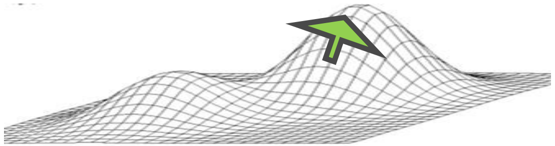
George Lasry

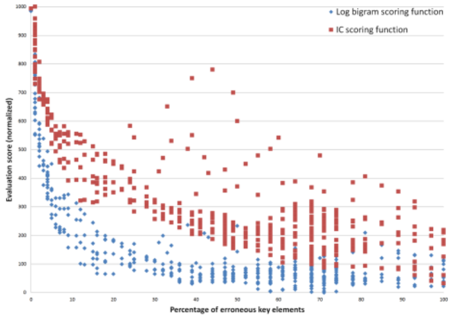
Agenda

- **Introduction**
  - Motivation
  - Difficulty
  - Generic approaches
- **Case studies**
  - Hagelin M-209
  - Playfair
  - Double transposition
  - SIGABA

Agenda

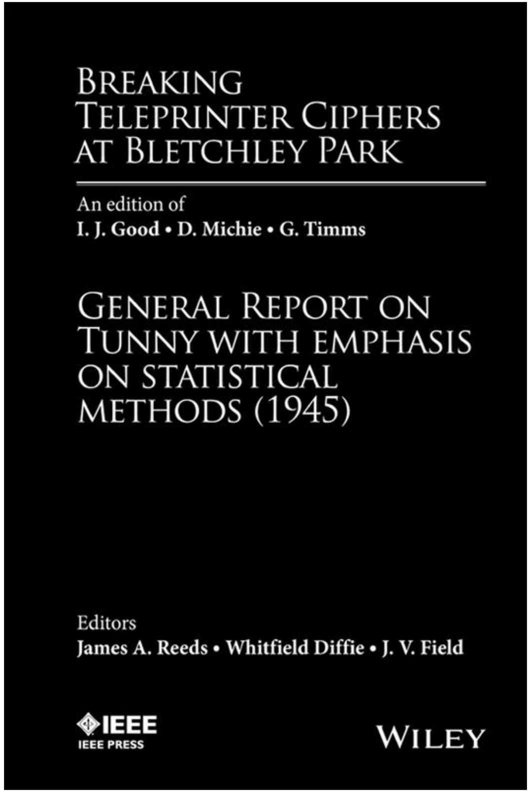
- **Introduction**
  - Motivation
  - Difficulty
  - Generic approaches
- **Case studies**
  - Hagelin M-209
  - Playfair
  - Double transposition
  - SIGABA

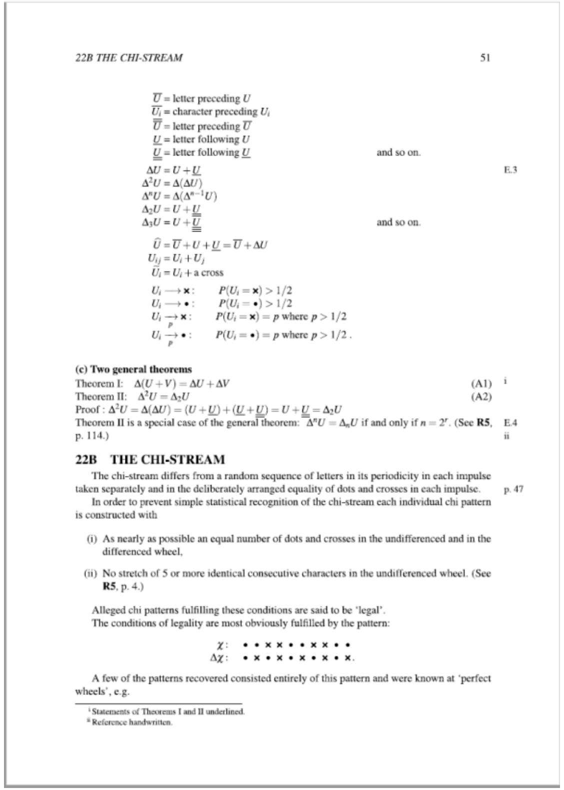




Motivation

- Historical cryptanalysis
- Undecrypted texts
- Public challenges
- Fun





22B THE CHI-STREAM 51

$\overline{U}$ = letter preceding U
$\overline{U_i}$ = character preceding U_i
$\overline{\overline{U}}$ = letter preceding $\overline{U}$
$\underline{U}$ = letter following U
$\underline{\underline{U}}$ = letter following $\underline{U}$ and so on.

$\Delta U = U + \underline{U}$ E.3
$\Delta^2 U = \Delta(\Delta U)$
$\Delta^n U = \Delta(\Delta^{n-1} U)$
$\Delta_2 U = U + \underline{\underline{U}}$
$\Delta_3 U = U + \underline{\underline{\underline{U}}}$ and so on.

$\hat{U} = \overline{U} + U + \underline{U} = \overline{U} + \Delta U$
$U_{ij} = U_i + U_j$
$\tilde{U}_i = U_i +$ a cross

$U_i \longrightarrow \times$: $P(U_i = \times) > 1/2$
$U_i \longrightarrow \bullet$: $P(U_i = \bullet) > 1/2$
$U_i \underset{p}{\longrightarrow} \times$: $P(U_i = \times) = p$ where $p > 1/2$
$U_i \underset{p}{\longrightarrow} \bullet$: $P(U_i = \bullet) = p$ where $p > 1/2$.

(c) Two general theorems

Theorem I: $\Delta(U+V) = \Delta U + \Delta V$ (A1) i
Theorem II: $\Delta^2 U = \Delta_2 U$ (A2)
Proof : $\Delta^2 U = \Delta(\Delta U) = (U + \underline{U}) + (\underline{U} + \underline{\underline{U}}) = U + \underline{\underline{U}} = \Delta_2 U$
Theorem II is a special case of the general theorem: $\Delta^n U = \Delta_n U$ if and only if $n = 2^r$. (See **R5**, p. 114.) E.4 ii

22B THE CHI-STREAM

The chi-stream differs from a random sequence of letters in its periodicity in each impulse taken separately and in the deliberately arranged equality of dots and crosses in each impulse. p. 47

In order to prevent simple statistical recognition of the chi-stream each individual chi pattern is constructed with

(i) As nearly as possible an equal number of dots and crosses in the undifferenced and in the differenced wheel,

(ii) No stretch of 5 or more identical consecutive characters in the undifferenced wheel. (See **R5**, p. 4.)

Alleged chi patterns fulfilling these conditions are said to be 'legal'.
The conditions of legality are most obviously fulfilled by the pattern:

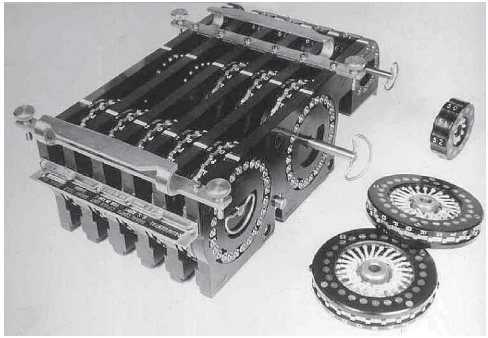
$$\chi: \bullet \bullet \times \times \bullet \bullet \times \times \bullet \bullet$$
$$\Delta\chi: \bullet \times \bullet \times \bullet \times \bullet \times \bullet \times .$$

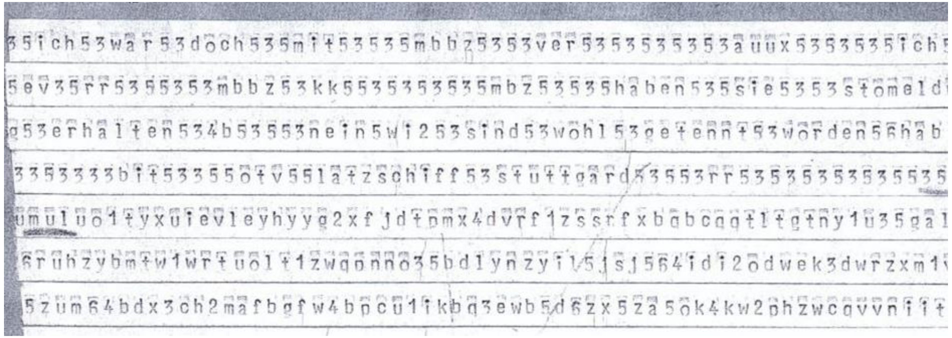
A few of the patterns recovered consisted entirely of this pattern and were known at 'perfect wheels', e.g.

[i] Statements of Theorems I and II underlined.
[ii] Reference handwritten.

Difficulty - Factors

- **System design**
  - Diffusion
  - Confusion
  - Weaknesses
- **Key**
  - Key space/length
- **Ciphertext**
  - Length
  - Language





Difficulty

Easy	Moderate	Hard	Very hard	Intractable?
Monoalphabetic substitution	Playfair (long ciphertext)	Playfair (short ciphertext)	**Playfair (very short)**	Fialka
Transposition (short key)	**Transposition (long key)**	**ADFGVX**	**Double transposition**	Double transposition (long random key)
Vigenere		Enigma (long ciphertext)	Enigma (short ciphertext)	**SIGABA (known plaintext)**
		Hagelin M-209 (long ciphertext)	**Hagelin M-209 (short ciphertext)**	
		Hagelin M-209 (known plaintext)	**Sturgeon T52 (regular stepping)**	**Sturgeon T52 (irregular stepping)**

Generic Approaches - 1

Exhaustive Search

- **Simple brute force**
- **Dictionary search**

- Match some constraints (e.g., known plaintext)
- Or optimize a scoring function

Combinatorial Search

- **Backtracking**
- **Meet in the Middle (MITM)**

- Match some constraints

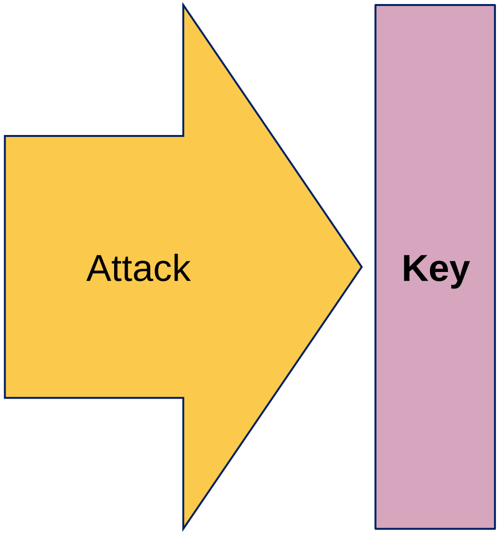
Stochastic Search

- **Hill climbing**
- **Simulated annealing**
- Hybrid (e.g., nested)
- Others (e.g., genetic algorithms)

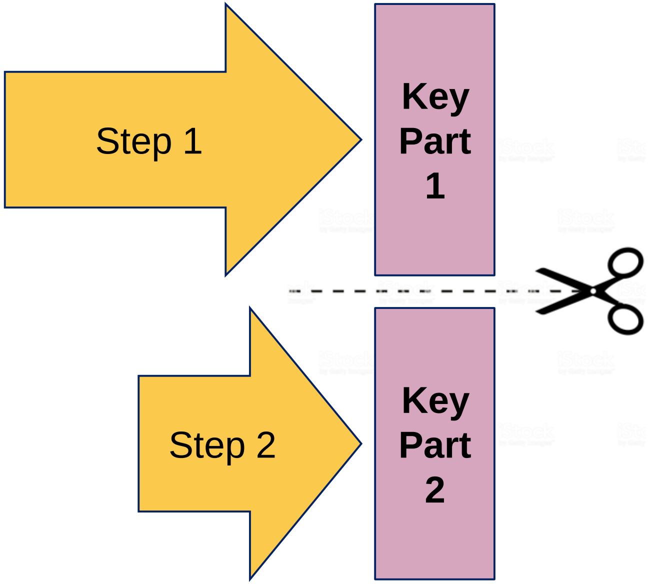
- Optimize a fitness or scoring function

Generic Approaches - 2

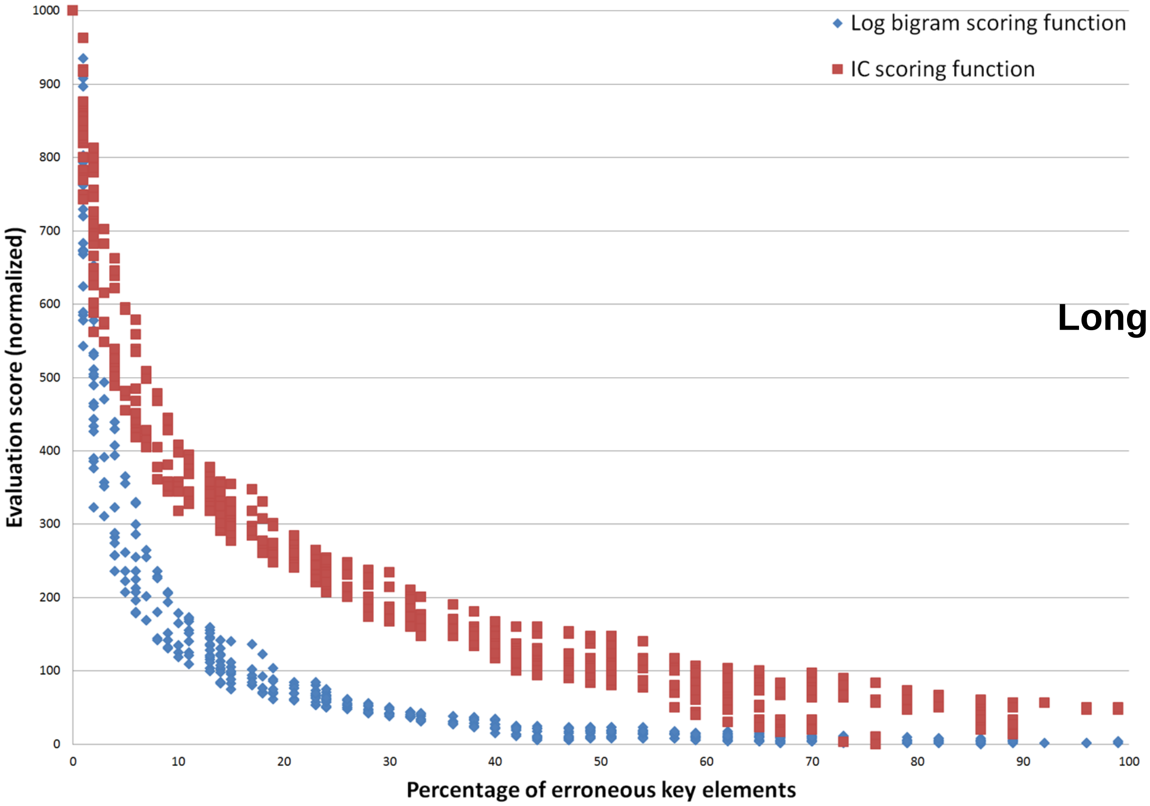
Frontal Attack



Divide and Conquer

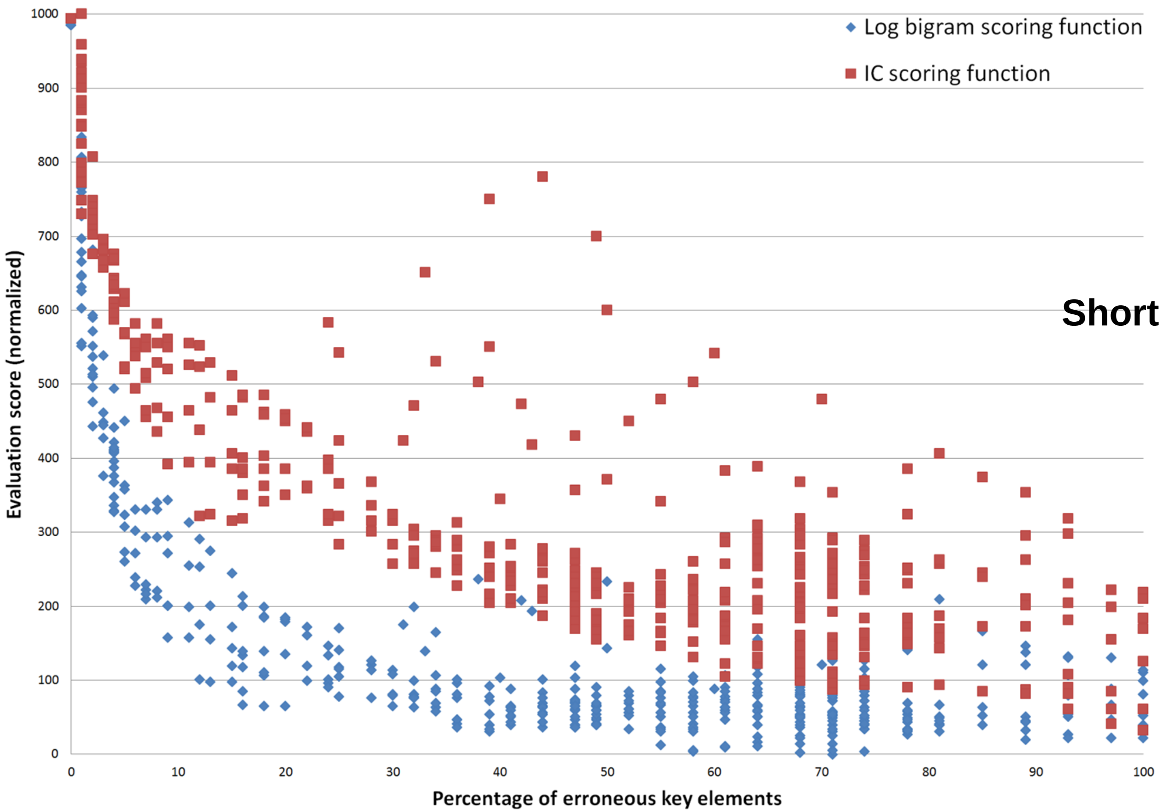


Scoring Functions - Resilience to Errors vs. Selectivity



Long ciphertexts

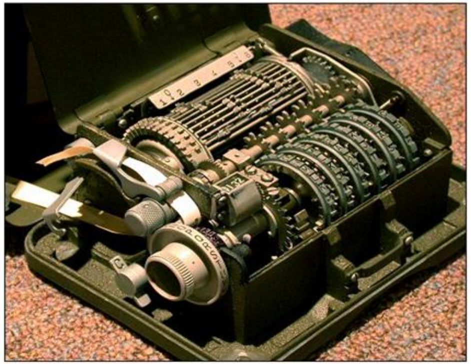
Scoring Functions - Resilience to Errors vs. Selectivity



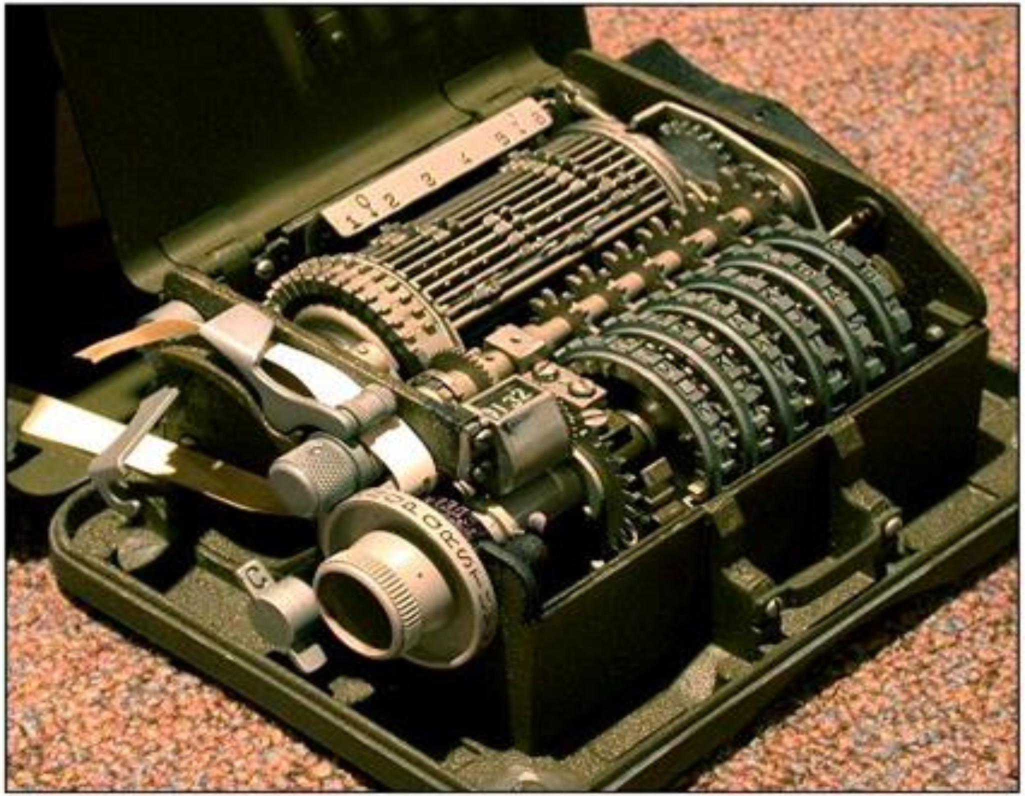
Short ciphertexts

Agenda

- **Introduction**
  - Motivation
  - Difficulty
  - Generic approaches
- **Case studies**
  - Hagelin M-209
  - Playfair
  - Double transposition
  - SIGABA

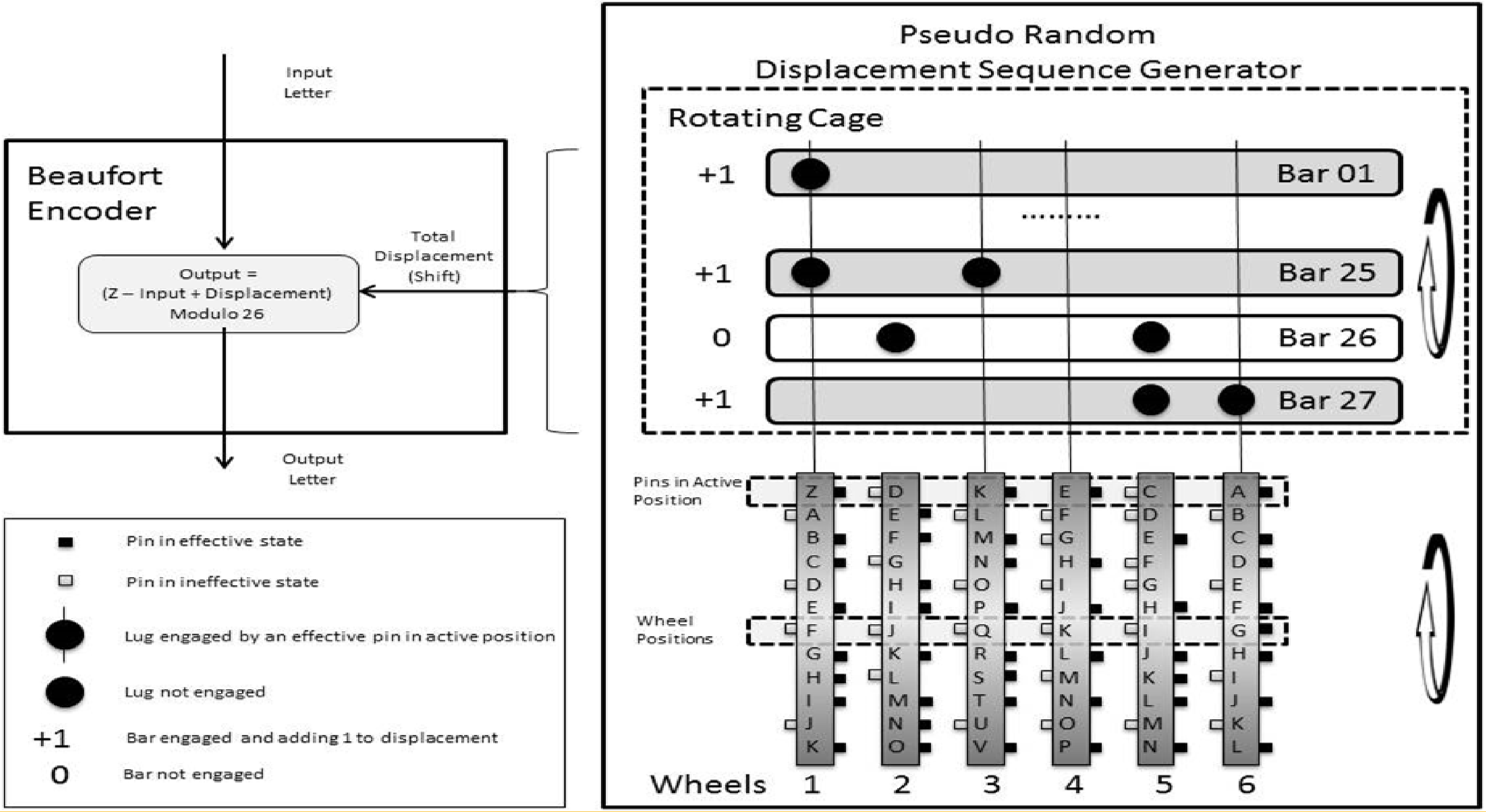


Hagelin M-209



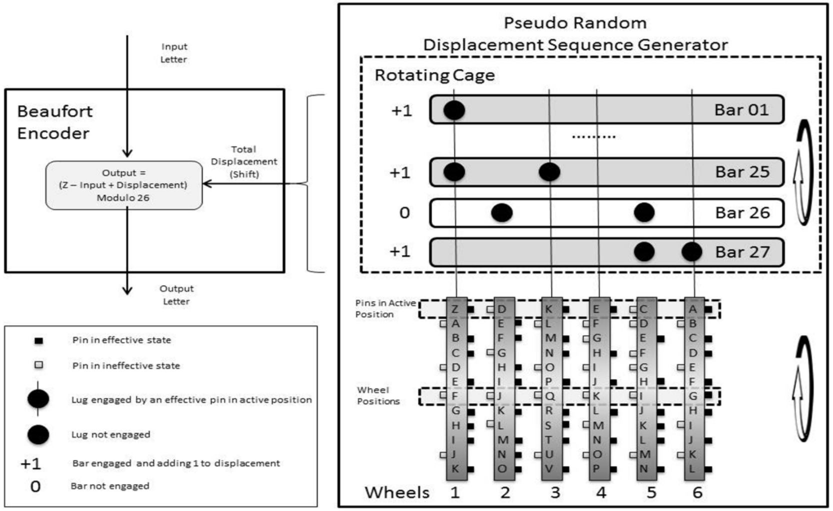
Boris Hagelin
1892-1983

Hagelin M-209 – Functional Diagram



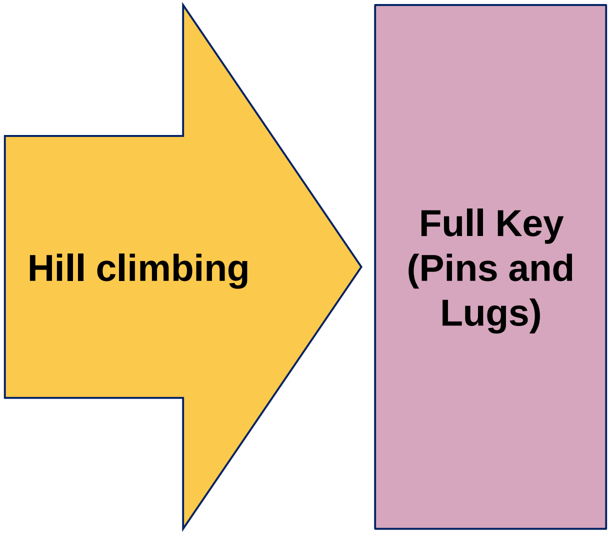
Hagelin M-209 - Key Space

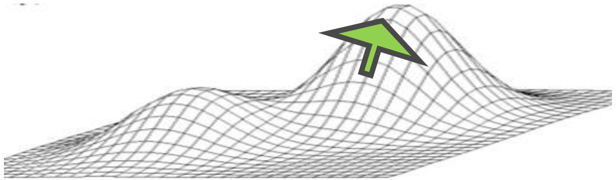
- **Wheel pins**
  - 2^{131} options
- **Lugs**
  - 2^{38} options
- **Total keyspace**
  - 2^{169}



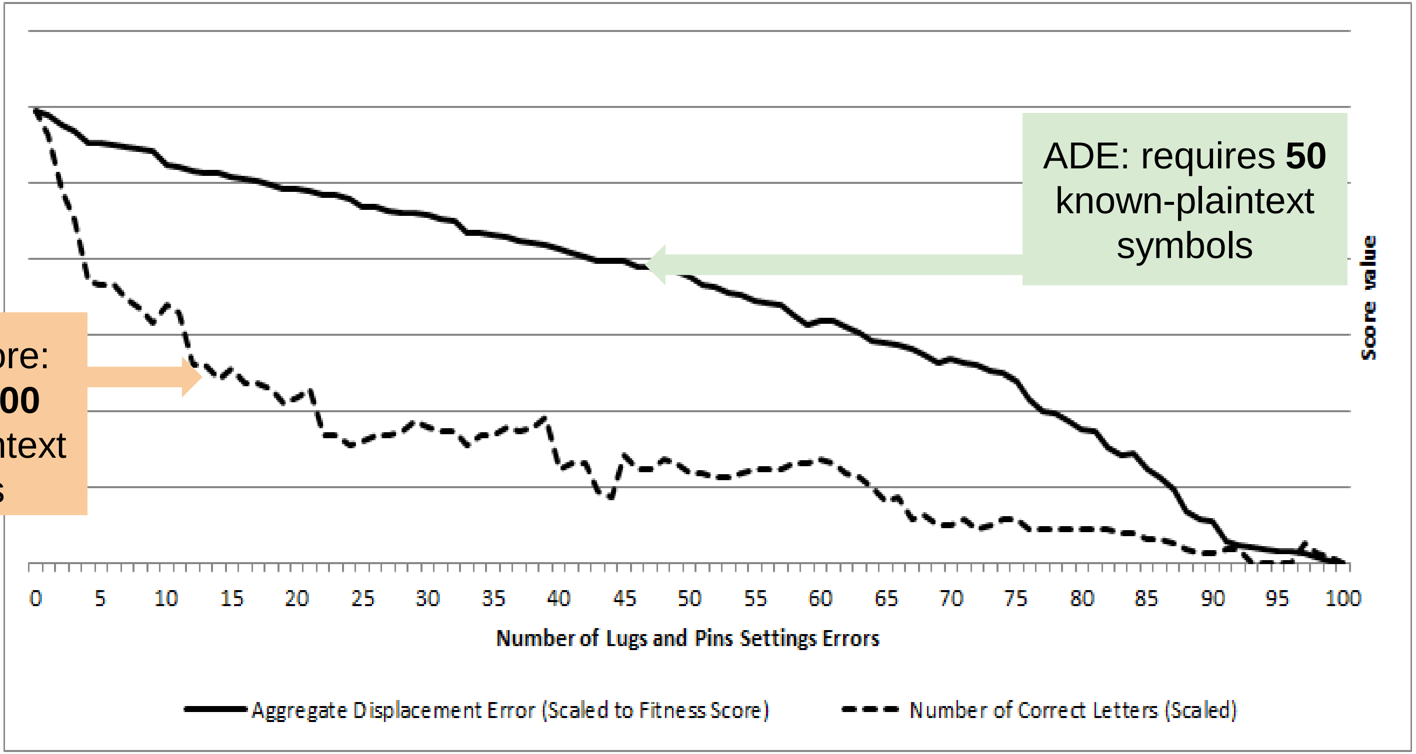
Known-Plaintext Attack

- **Frontal attack**
  - On full key space - pins and lugs
- **Hill climbing**
- **Specialized score**
  - ADE - Aggregate Displacement Score





Known-Plaintext Attack - ADE Scoring Function

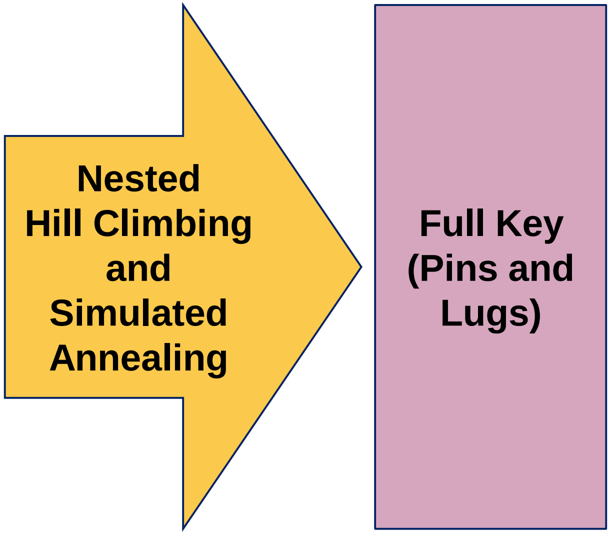


ADE: requires **50** known-plaintext symbols

Simple score: requires **300** known-plaintext symbols

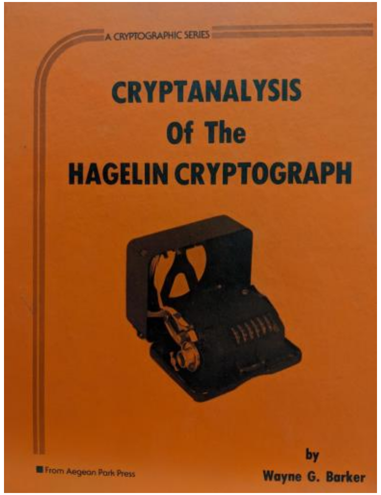
M-209 - Ciphertext-Only Attack

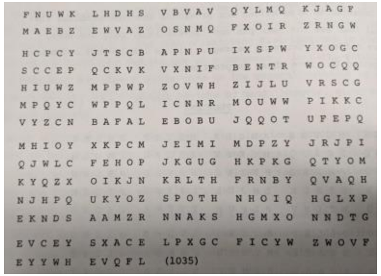
- **Frontal <u>nested</u> attack**
- **Outer hill climbing - lugs**
  - Inner simulated annealing - pins
- **Log monograms**
- **Requires only 500 letters**
  - Vs. 1500 with previous attacks



M-209 - Ciphertext-Only Attack

- Frontal nested attack
- Outer hill climbing - lugs
  - Inner simulated annealing - pins
- Log monograms
- Requires only 500 letters
  - Vs. 1500 with previous attacks
- **Challenges solved**
  - 1035 letters - 1977
  - 500 letters - 2012

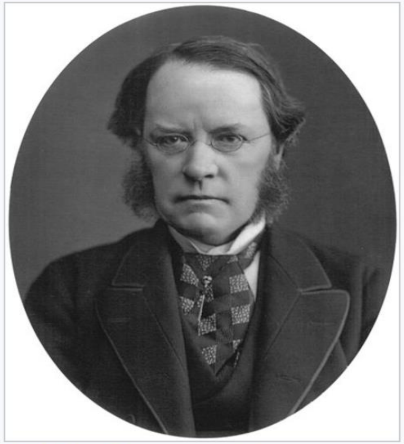




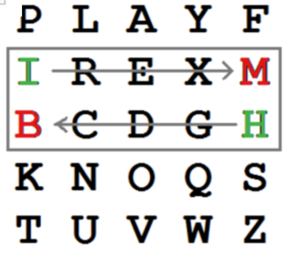
Agenda

- Introduction
  - Motivation
  - Difficulty
  - Generic approaches
- **Case studies**
  - Hagelin M-209
  - **Playfair**
  - Double transposition
  - SIGABA

The Playfair system was invented by Charles Wheatstone, who first described it in 1854.

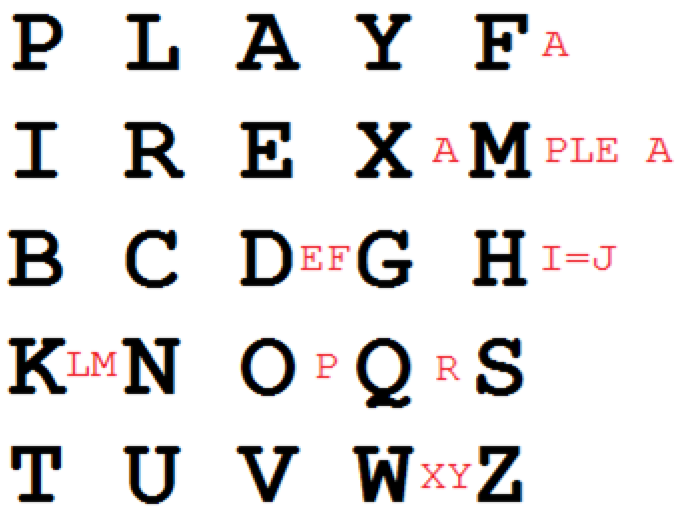


Lord Playfair, who heavily promoted its use.

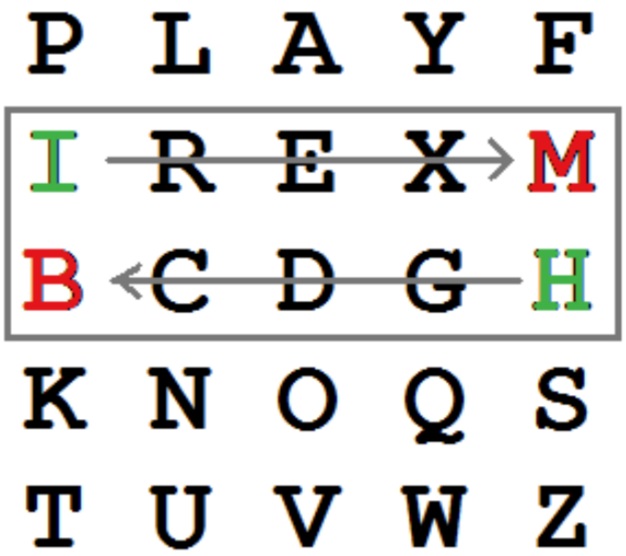


The Playfair Cipher – Key Square

- **Keyword:**
  - PLAYFAIREXAMPLE



The Playfair Cipher – Encryption Rule 1

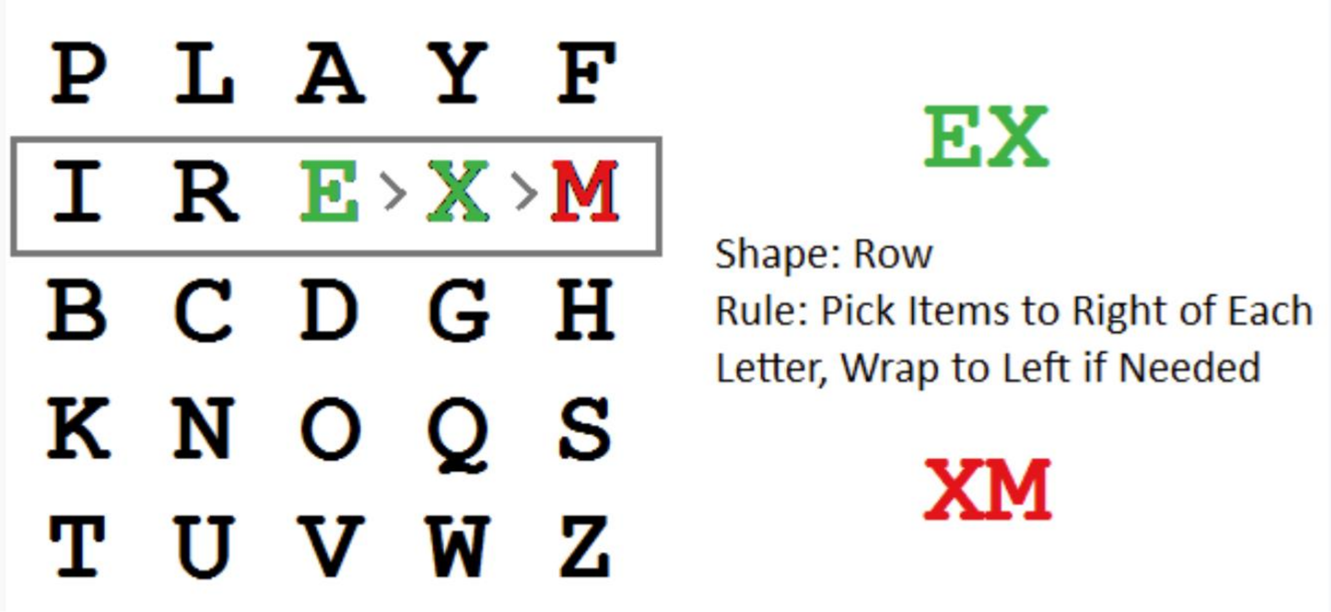


HI

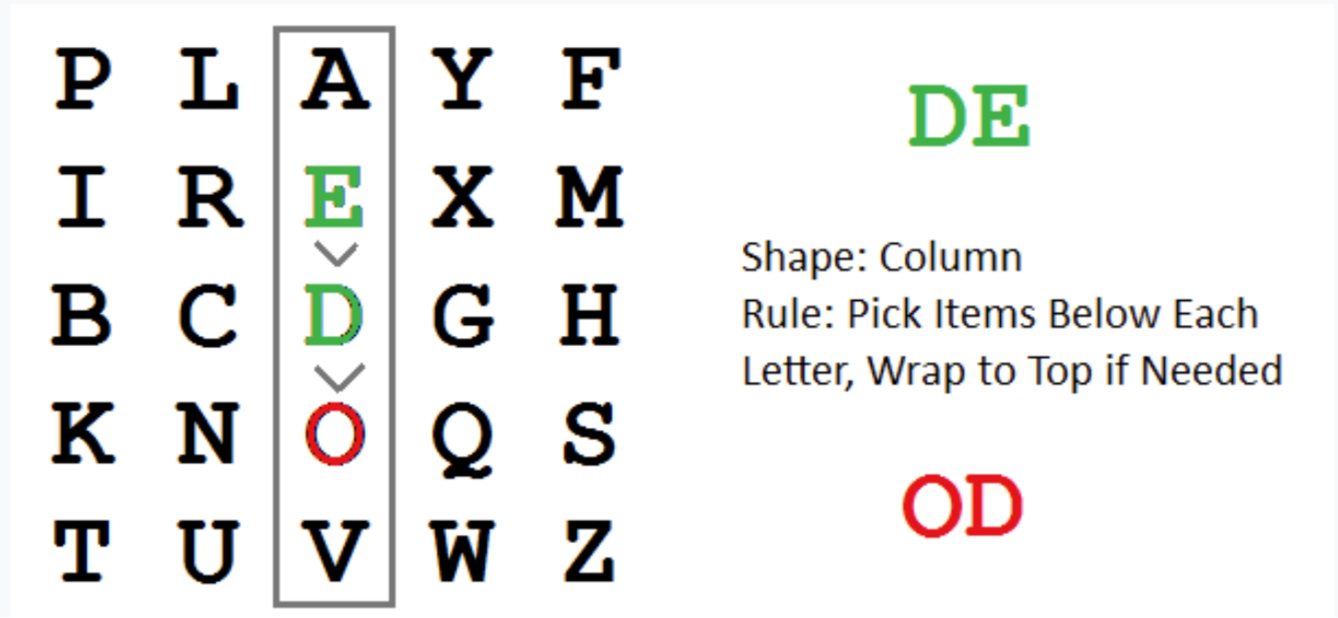
Shape: Rectangle
Rule: Pick Same Rows, Opposite Corners

BM

The Playfair Cipher – Encryption Rule 2



The Playfair Cipher – Encryption Rule 3



The Playfair Cipher – Example

```
Original plaintext:  Hide the gold in the tree stump

Formatted plaintext: HI DE TH EG OL DI NT HE TR EX ES TU MP

Ciphertext:          BM OD ZB XD NA BE KU DM UI XM MO UV IF
```

Prior Attacks

- **Historical attacks**
  - Ciphertext only: 800 letters (Mauborgne, 1918)
  - Key from keyword: 30 letters (Monge, 1936)
  - From crib
- **Modern attacks**
  - Hillclimbing: hundreds of letters
  - Simulated annealing: 80 letters (Cowan, 2008)
    - 4-grams, logarithmic scale
  - Compression-based: 60 letters (Al-Kazaz et al., 2018)
    - Order 5, equivalent to 6-grams on log. scale

Chief Signal Officer, U.S. Army

Joseph Oswald Mauborgne

Born	February 26, 1881 New York, New York
Died	June 7, 1971 (aged 90) Atlanta, Georgia

SOLUTION OF A PLAYFAIR CIPHER [1] *

By Private ALF MONGE, *Ninth Signal Service Company*

Breaking Short Playfair Ciphers with the Simulated Annealing Algorithm

MICHAEL J. COWAN

Abstract Describes adaptation of simulated annealing to solve short playfair ciphers (80–120 letters) without using a probable word.

Keywords classical ciphers, cryptanalysis, Playfair, short ciphers, simulated annealing (SA)

An Automatic Cryptanalysis of Playfair Ciphers Using Compression

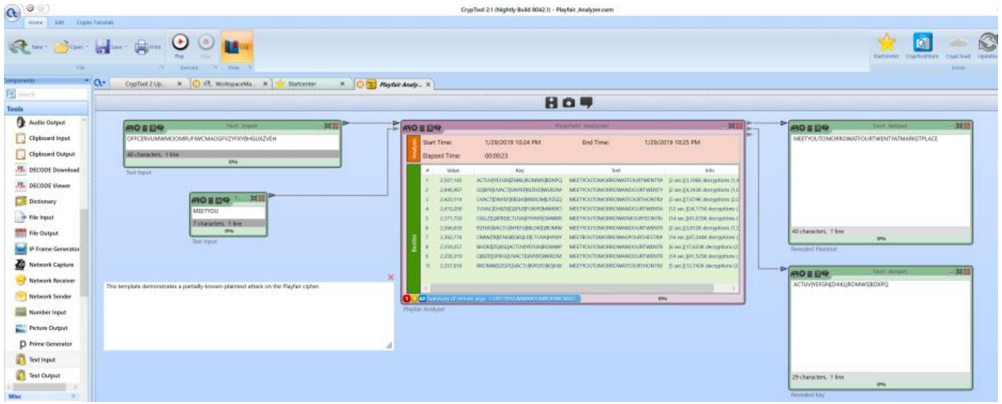
Noor R. Al-Kazaz[1]
School of Computer Science
Bangor University
Bangor, UK
n.al-kazaz@bangor.ac.uk
noor82.nra@gmail.com

Sean A. Irvine
Real Time Genomics
Hamilton, New Zealand
sairvin@gmail.com

William J. Teahan
School of Computer Science
Bangor University
Bangor, UK
w.j.teahan@bangor.ac.uk

Ciphertext-Only Attack - Short Ciphertexts

- **Integrated into CrypTool 2**
  - Java code
  - Analysis Connector API
- **Simulated Annealing**
  - Enhanced
- **6-grams**
- **Rich transformations**
  - Swaps of any 2 elements/rows/columns
  - Permutations of the rows/columns, inside row/column



Simulated Annealing - Variable Temperature

Algorithm 4 Simulated annealing algorithm - variable temperature

procedure SIMULATEDANNEALING(C, N, T_0, α) ▷ N = SA rounds, α = cooling factor
  $BestKey \leftarrow CurrentKey \leftarrow RandomKey()$
  $T \leftarrow T_0$
  for $I = 1$ to N **do**
    for $Transformation \in PossibleTransformations$ **do**
      $CandidateKey \leftarrow Apply(Transformation, CurrentKey)$
      $D \leftarrow S(CandidateKey, C) - S(CurrentKey, C)$ ▷ Degradation
      $P_a \leftarrow e^{-\frac{|D|}{T}}$ ▷ Acceptance probability
      if $D > 0$ or $Random(0..1) < P_a$ **then**
        $CurrentKey \leftarrow CandidateKey$ ▷ New key accepted
        if $S(CurrentKey, C) > S(BestKey, C)$ **then**
          $BestKey \leftarrow CurrentKey$ ▷ Found a better global key
    $T \leftarrow \alpha \cdot T$ ▷ Reduce temperature
  return $BestKey$

Simulated Annealing - Fixed Temperature

Algorithm 5 Simulated annealing algorithm - fixed temperature

procedure SIMULATEDANNEALING(C, N, T) ▷ T = fixed temperature
    $BestKey \leftarrow CurrentKey \leftarrow RandomKey()$
    for $I = 1$ to N **do**
        for $Transformation \in PossibleTransformations$ **do**
            $CandidateKey \leftarrow Apply(Transformation, CurrentKey)$
            $D \leftarrow S(CandidateKey, C) - S(CurrentKey, C)$ ▷ Degradation
            $P_a \leftarrow e^{-\frac{|D|}{T}}$ ▷ Acceptance probability
            if $D > 0$ or $Random(0..1) < P_a$ **then**
                $CurrentKey \leftarrow CandidateKey$ ▷ New key accepted
                if $S(CurrentKey, C) > S(BestKey, C)$ **then**
                    $BestKey \leftarrow CurrentKey$ ▷ Found a better global key
    return $BestKey$

Simulated Annealing - Minimal Acceptance Probabililty

Algorithm 6 Simulated annealing algorithm - with minimal acceptance probability

procedure SIMULATEDANNEALING(C, N, T, P_{min}) ▷ P_{min} = min. acceptance probability
  $BestKey \leftarrow CurrentKey \leftarrow RandomKey()$
  for $I = 1$ to N **do**
    for $Transformation \in PossibleTransformations$ **do**
      $CandidateKey \leftarrow Apply(Transformation, CurrentKey)$
      $D \leftarrow S(CandidateKey, C) - S(CurrentKey, C)$ ▷ Degradation
      $P_a \leftarrow e^{-\frac{|D|}{T}}$ ▷ Acceptance probability
      if $D > 0$ or $(Random(0..1) < P_a$ and $P_a > P_{min})$ **then**
        $CurrentKey \leftarrow CandidateKey$ ▷ New key accepted
        if $S(CurrentKey, C) > S(BestKey, C)$ **then**
          $BestKey \leftarrow CurrentKey$ ▷ Found a better global key
  return $BestKey$

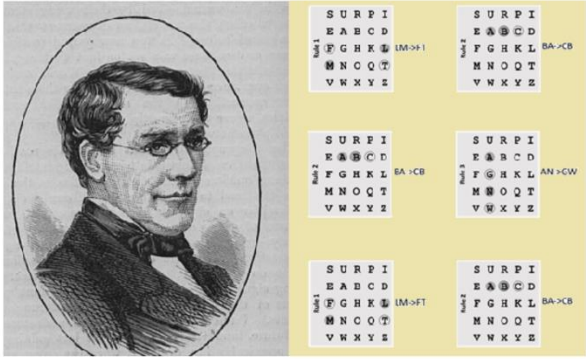
Klaus Schmeh's Challenges

KLAUS IS KRYPTO KOLUMNE

Playfair cipher: Is it unbreakable, if the message has only 50 letters?

Von Klaus Schmeh / 7. April 2018 / 15 Kommentare / Seite 1 von 2 / Auf einer Seite lesen

Gefällt mir 11 Twittern Mehr



The Playfair cipher is an encryption method from the 19th century. Some say that a Playfair-encrypted message of 50 or less letters is still secure today, if the method is used properly. Let's put this claim to the test.

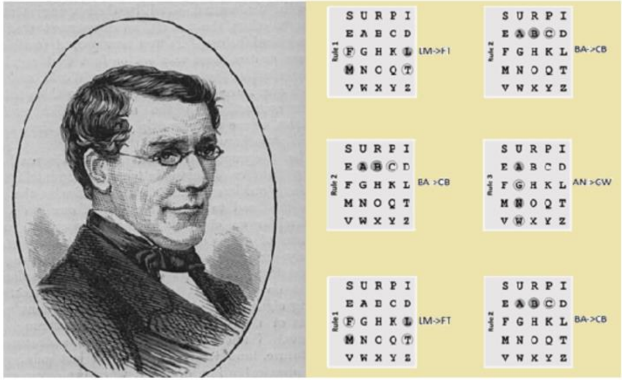
SOLVED

KRYPTO KOLUMNE

Playfair cipher: Is it breakable, if the message has only 40 letters?

Von Klaus Schmeh / 8. Dezember 2018 / 11 Kommentare / Seite 1 von 2 / Auf einer Seite lesen

Gefällt mir 11 Twittern Mehr



My readers have shown that a Playfair cryptogram consisting of only 50 letters can be broken. Here's a Playfair challenge with only 40 letters. Can you break it, too?

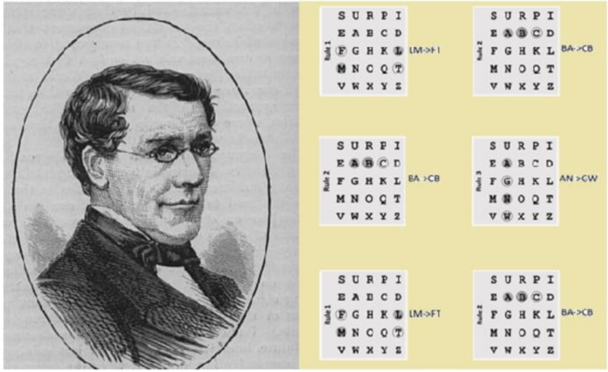
SOLVED

KLAUS IS KRYPTO KOLUMNE

Playfair cipher: Is it breakable, if the message has only 30 letters?

Von Klaus Schmeh / 15. April 2019 / 7 Kommentare / Seite 1 von 2 / Auf einer Seite lesen

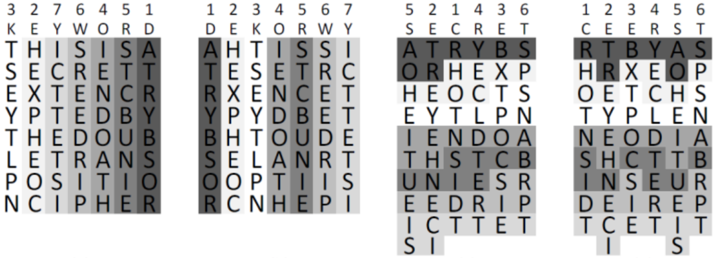
Gefällt mir 5 Twittern Mehr



My readers have shown that a Playfair cryptogram consisting of only 40 letters can be broken. Here's a Playfair challenge with only 30 letters. Can you break it, too?

Agenda

- Introduction
  - Motivation
  - Difficulty
  - Generic approaches
- **Case studies**
  - Hagelin M-209
  - Playfair
  - **Double transposition**
  - SIGABA



(a) (b) (c) (d)

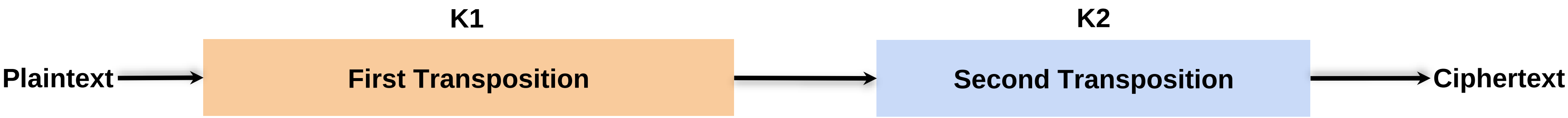
Double Transposition Cipher - The "Spy Cipher"

3	2	7	6	4	5	1
K	E	Y	W	O	R	D
T	H	I	S	I	S	A
S	E	C	R	E	T	T
E	X	T	E	N	C	R
Y	P	T	E	D	B	Y
T	H	E	D	O	U	B
L	E	T	R	A	N	S
P	O	S	I	T	I	O
N	C	I	P	H	E	R

1	2	3	4	5	6	7
D	E	K	O	R	W	Y
A	H	T	I	S	S	I
T	E	S	E	T	R	C
R	X	E	N	C	E	T
Y	P	Y	D	B	E	T
B	H	T	O	U	D	E
S	E	L	A	N	R	T
O	O	P	T	I	I	S
R	C	N	H	E	P	I

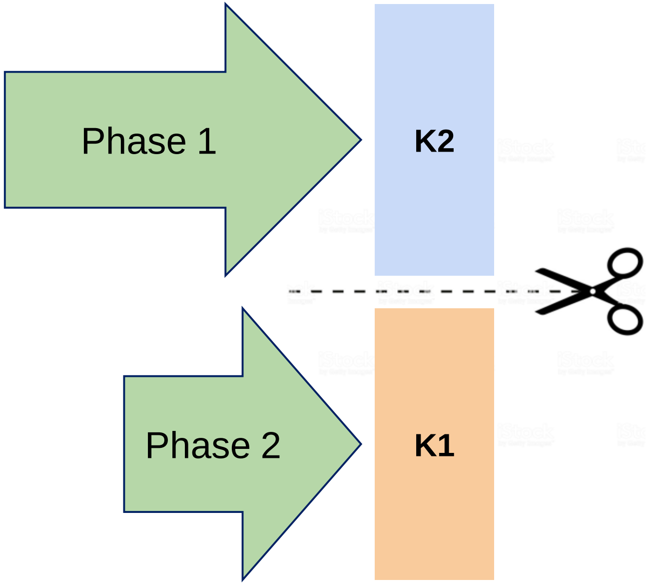
5	2	1	4	3	6
S	E	C	R	E	T
A	T	R	Y	B	S
O	R	H	E	X	P
H	E	O	C	T	S
E	Y	T	L	P	N
I	E	N	D	O	A
T	H	S	T	C	B
U	N	I	E	S	R
E	E	D	R	I	P
I	C	T	T	E	T
S	I				

1	2	3	4	5	6
C	E	E	R	S	T
R	T	B	Y	A	S
H	R	X	E	O	P
O	E	T	C	H	S
T	Y	P	L	E	N
N	E	O	D	I	A
S	H	C	T	T	B
I	N	S	E	U	R
D	E	I	R	E	P
T	C	E	T	I	T
	I			S	

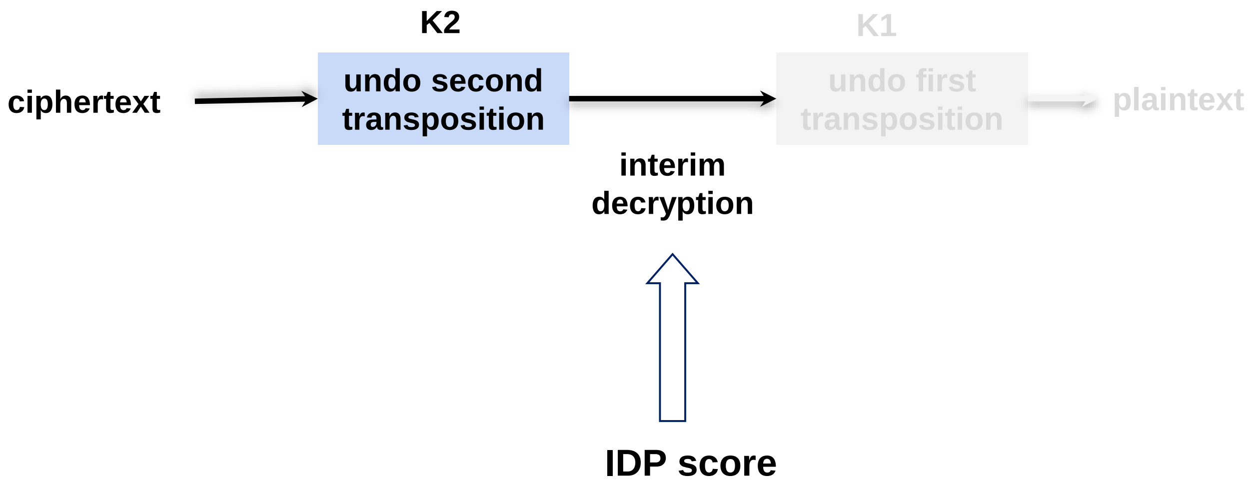


Double Transposition - Attack

- **Divide and Conquer**
- **Phase 1**
  - Find K2
  - Hillclimbing
  - ***Specialized scoring - IDP***
- **Phase 2**
  - Undo K2 and find K1
  - Hillclimbing, 4-grams



IDP - Index of Digraphic Potential



- Hillclimbing
- Dictionary attack

The Double Transposition Cipher Challenge, 2007

- **Otto Leiberich**
- **Klaus Schmeh**
- **Secure parameters**
  - Different K1 and K2
  - Key lengths 20 to 25
  - Cryptogram length

VESINTNVONMWSFEWNOEALWRNRNCFITEEICRHCODEEAHEACAEOHMYTONTDFIFMDANGTDRVAONRRTORMTDHE
OUALTHNFHHWHLESLIIAOETOUTOSCDNRITYEELSOANGPVSHLRMUGTNUITASETNENASNNANRTTRHGUODAAAR
AOEGHEESAODWIDEHUNNTFMUSISCDLEDTRNARTMOOIREEYEIMINFELORWETDANEUTHEEEENENTHEOOEAUEA
EAHUHICNCGDTUROUTNAEYLOEINRDHEENMEIAHREEDOV...

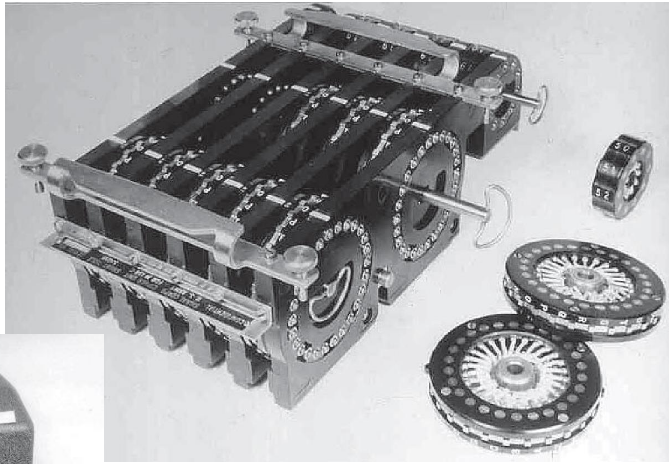
Solving the Challenge

What exciting news, deciphering the "Doppelwürfel"!
I congratulate you to this great success.

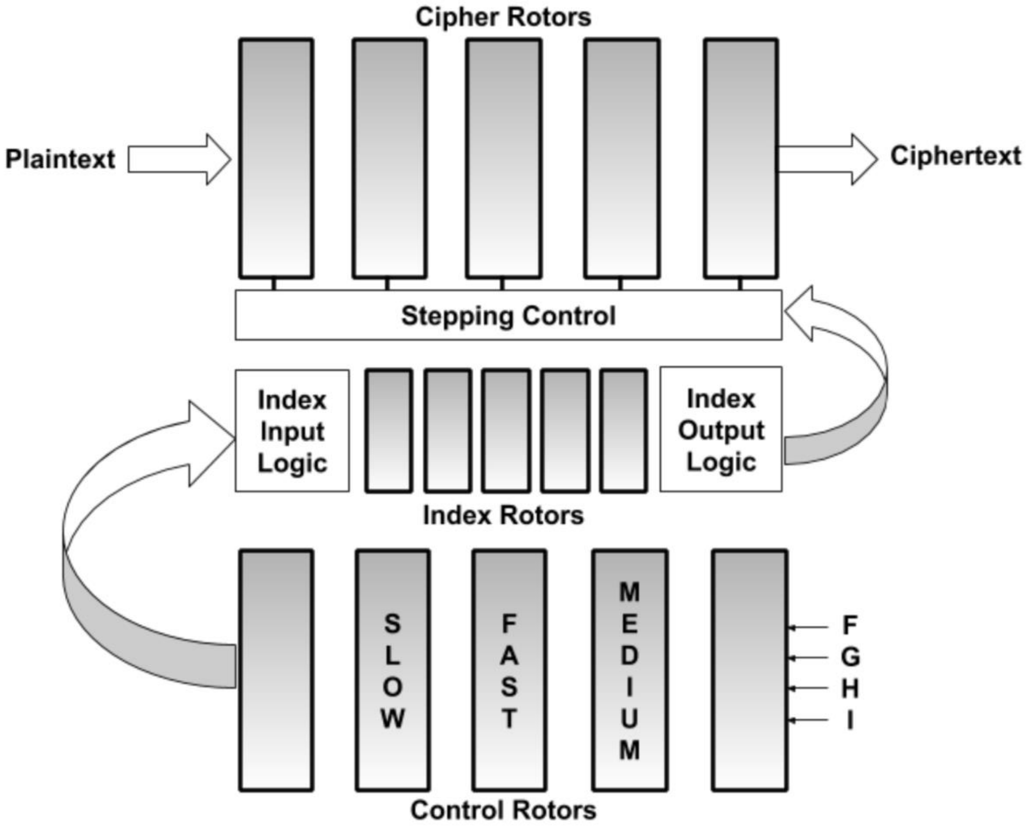
Otto Leiberich, December 29, 2013

Agenda

- Introduction
  - Motivation
  - Difficulty
  - Generic approaches
- **Case studies**
  - Hagelin M-209
  - Playfair
  - Double transposition
  - SIGABA

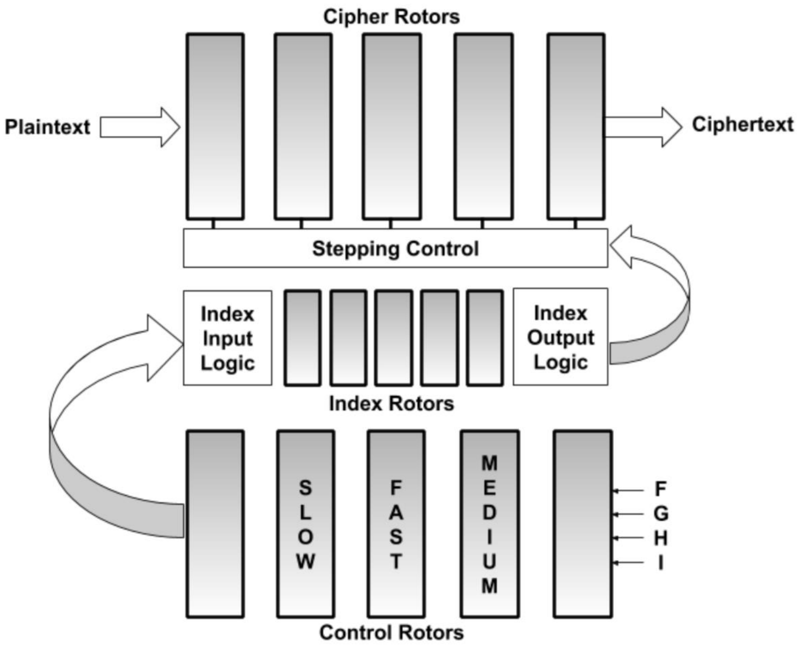


Design of SIGABA



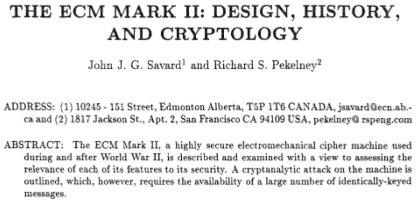
SIGABA - Key Space

- **Cipher and control rotors**
  - $2^{78.8}$ options
- **Index Rotors**
  - $2^{16.8}$ options
- **Total keyspace**
  - $2^{95.6}$



Prior Attacks

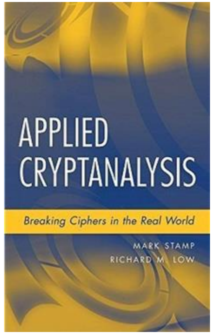
- **WW2**
  - "U.S. 5-letter traffic: Work discontinued as unprofitable at this time."
- **Savard and Pekelney – 1999**
  - Attack on messages "in depth"
  - Unrealistic operational scenario
- **Stamp and Chan – 2007, Stamp and Low – 2007**
  - Known-plaintext attack
  - $2^{86.7}$ vs. $2^{95.6}$ for brute-force attack


THE ECM MARK II: DESIGN, HISTORY, AND CRYPTOLOGY

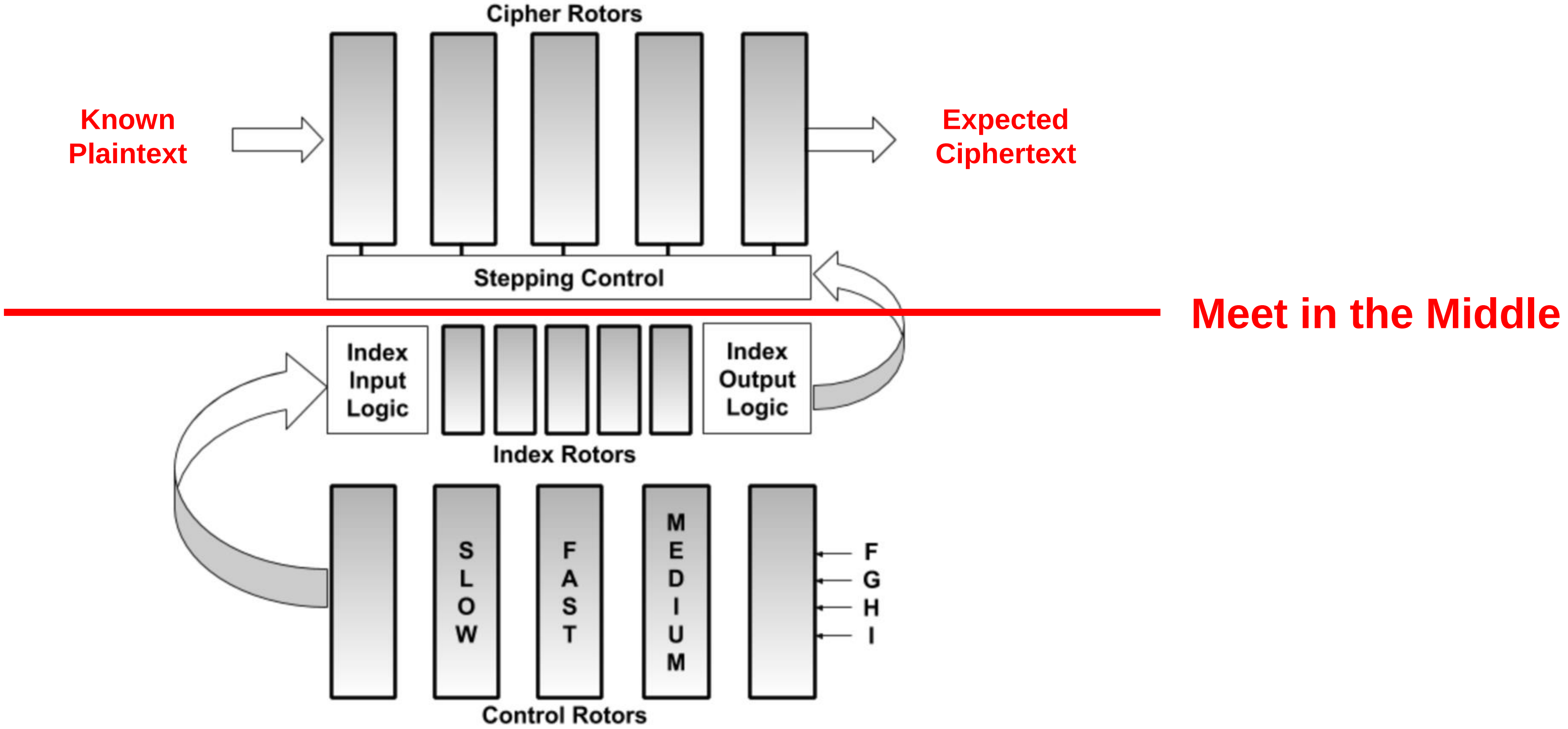
John J. G. Savard[1] and Richard S. Pekelney[2]

ADDRESS: (1) 10245 - 151 Street, Edmonton Alberta, T5P 1T6 CANADA, jsavard@ecn.ab.ca and (2) 1817 Jackson St., Apt. 2, San Francisco CA 94109 USA, pekelney@rspeng.com

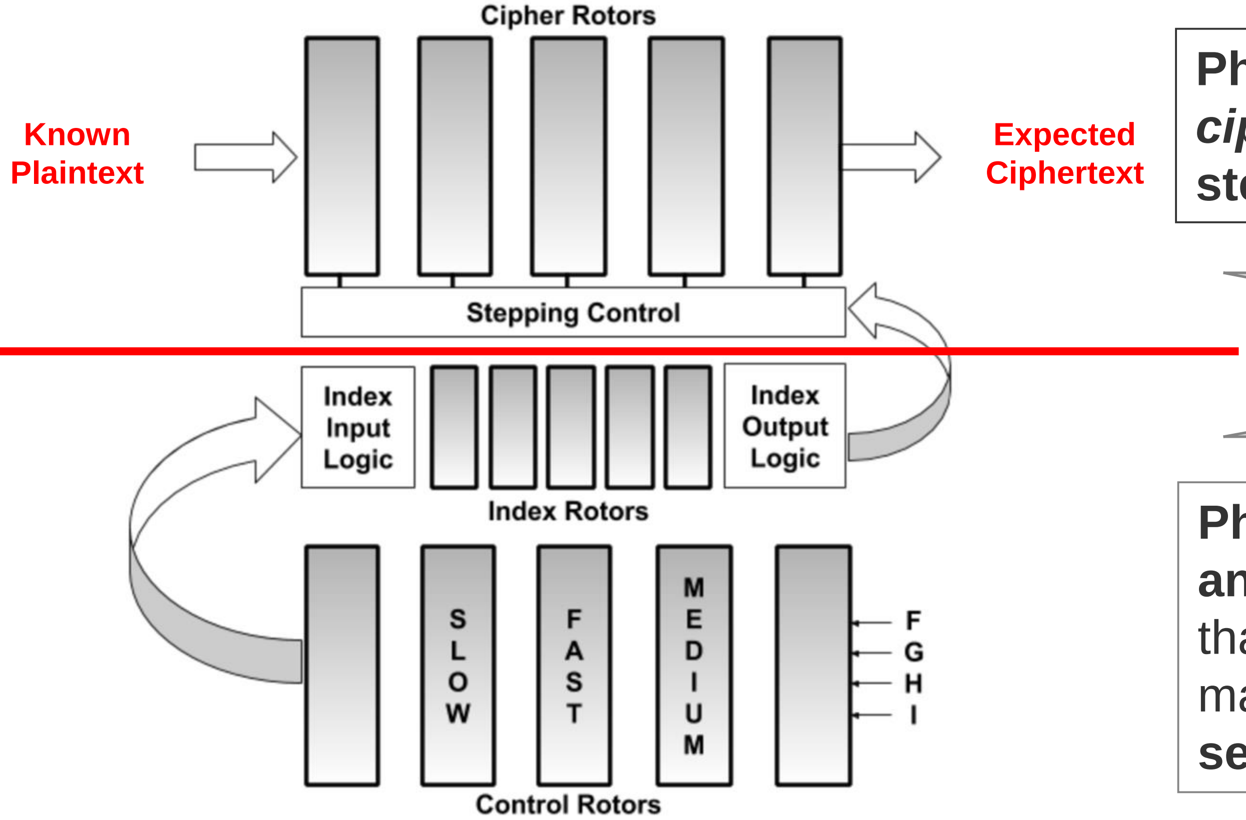
ABSTRACT: The ECM Mark II, a highly secure electromechanical cipher machine used during and after World War II, is described and examined with a view to assessing the relevance of each of its features to its security. A cryptanalytic attack on the machine is outlined, which, however, requires the availability of a large number of identically-keyed messages.



New Known-Plaintext Attack – Concept



New Known-Plaintext Attack – Outline



Phase 1: Find matching ***cipher*** **rotor settings** and **stepping sequences**

Meet in the Middle

Phase 2: Find ***control*** **and** ***index*** **rotor settings** that generate one of the matching ***cipher*** **stepping sequences**

Hash Table for Meet-in-the-Middle Attack

Stepping Sequence (Hash Key)		(Maps to) **Cipher Rotor Settings**					
		Rotor Selection 1	**2**	**3**	**4**	**5**	**Starting Positions**
01011 01000 11001 00111 00111 11110 00010	⇛	8R	0	4R	7	1	H Y J N H
	⇛	1	7R	0	8R	4	T U A L M
01111 01100 11100 11001 00010 10101 10001	⇛	1R	4	8R	7	0	K H J N M
11010 10011 00111 10101 00111 00110 10011	⇛	0	8R	7R	4	1R	E Q A M B

Meet-in-the-Middle Attack – Complexity

- **Processing**
  - Phase 1: $2^{47.1}$
  - Phase 2: $2^{60.2}$
  - Overall: max($2^{47.1}$, $2^{60.2}$) = $\mathbf{2^{60.2}}$
  - Comparison:
    - $\mathbf{2^{95.6}}$ for brute-force attack
    - $\mathbf{2^{86.7}}$ for best prior attack (Stamp & al. 2007)
    - $\mathbf{2^{56}}$ for DES – cracked in 1997
- **Space**
  - About 80GB RAM for hash table
- **Feasible with modern technology**

Solving MysteryTwisterC3 Challenges

New Challenges

	Ciphertext	First 100 Plaintext Letters	Hint
#1	GSZQEMAGFULNFZHHRVUTCUEXU FBMPDGOROJRPMAUDOZMJWJCVH YCBZDELOWKVLYJLSZBQJXWXLR WOIMBVUTBAVRHPPPYQDTIURLV IQGIZSEVGXOYCMGESFOXDLPFT UQQCRDSRNFDTBDDULFJKQGXZB XKKIMSBSIUZSZNOOLCFRRVTOD XFQRRXLDEMSLORKXUCGDKCZKY ULDORUGEGDLTTROBUIVWJTBVH YWOKANYJCGQUYGPHSMWJRILZP SQJOXKKMEGMWQKXWVKF	AHZFOULZSHREWDZNEWSZBESHR EWZTHYZVERYZHEARTZIZDIDZN OTZTHINKZTOZBEZSOZSADZTON IGHTZASZTHISZHATHZMADEZME	All *cipher* and *control* rotors are at position A.
#2	ZMJHMLJTJSSHZBBMYXJRVZCUS PMETNBPZQCAHGYJDHJNQNMTHY EJAOOQYFSURONLTOGQVKOMABX QXGKRAVBZYWBRWYWGLBYFZNNA XIVJVOJYYBQGTWJIIZESYBRAN XEWYDRMYAINJWWDFWBVCTHRGL ZCTNHWWBRYSJSZSYMSSLUXBLZ STDBARVGCSMTJOWIRFXYIBZCF CCYRUXMUCISNUIFLCOJYZQTBY DWVFJDHZBJNSAPYAUYWQGFPYO ZJYWPCWVRSVCQTPHTFPGHCJAM CFZRHYNFXJVWWNNN	WOULDSTZTHOUZNOTZBEZGLADZ TOZHAVEZTHEZNIGGARDLYZRAS CALLYZSHEEPBITERZCOMEZBYZ SOMEZNOTABLEZSHAMEZFABIAN	The last 4 *cipher* rotors and the last 4 *control* rotors are at position A.
#3	HYYQUSBFHVDVKSLKSGUQIVZAR QKCQZBLLGCTCLQHZNBEQVUOJH BROKUKRYXWPGSPDJSWLLTDASB MTTPRPFHMSXPLBDENAYJWAQZD JDXGBJCWXNARABTTSEZBJDYHT NEIQCQRTFUAZDTTVBNHJGWQHF UHAPPBPYJAIXGELTILPULVSNC BJJIGFJNYDURTIVWYHTNKFSLS ALTHLBHYQBYXUK	TISZWONDERFULZWHATZMAYZBE ZWROUGHTZOUTZOFZTHEIRZDIS CONTENTZNOWZTHATZTHEIRZSO ULSZAREZTOPFULZOFZOFFENCE	The last 3 *cipher* rotors and the last 3 *control* rotors are at position A.
#4	CEXZZGZOYLDYPAGJQTFJSEYZP ORHMSTYLQVSJARJLCDBYXFPKB NREAEYVOPBQKYFYETXOUQNMAT CBWIIFKJWZJFWZHMJYQALVNXV UDUVEJGJNBWZRCVMIHDHLPOSD LSBPTFNEGWIAIRZZPIPPVEBWV VBGLNCGBKWFUUCVGTTGKGEHJQ XGEHVPLDDLALNWVNDOXTPPWCQ HNAWFTXVOWIZFVRWXBIIJDFAU TMCNWDHLSCHNOBQRURVLCXLVB YXDXKMPYIWPYOXPFXBNESBUCR WZECWXOUDTVVNRGGHPTE	IZWILLZBESPEAKZOURZDIETZW HILESZYOUZBEGUILEZTHEZTIM EZANDZFEEDZYOURZKNOWLEDGE ZWITHZVIEWINGZOFZTHEZTOWN	The last 2 *cipher* rotors and the last 2 *control* rotors are at position A.
#5	JJJWJZMPUKYDGRHSPIXTYPAPA IVGFOTXMFWRZLBRXQPNRYLCPF WNMZFHFSMVIEEDAHWZOMBIVPA RTAOWYOWRFACGAITUAFDFCTEV YZAQIQXVHZFCIBSVSQJAMYPTS YNWXBFBKDKVDOXZQQEVVGAAWI LRFYRGIPJCKVVPMQAEIAIMOPY XCSJFDAUHYZYVQJXGGZTMCAGW BEICRYROYCPNGEZQFVVQTSZBP SZYWCONNWMUBCNYQX	HOWZMIGHTZWEZSEEZFALSTAFF ZBESTOWZHIMSELFZTONIGHTZI NZHISZTRUEZCOLOURSZANDZNO TZOURSELVESZBEZSEENZPOINS	The last *cipher* rotor and the last *control* rotor are at position A.
#6	FWEYNOPSTLFMWXQITVTMRVHOL YDEIROBXPPVZVBLCSJPSYIXIY IJHJMCHAWSWAQBHSUVASAGYLR DJREKIFQUXBEJZUFVIJBJMWVT VSPHOQTRAECHEEJLBRCDTGXRP OVSJKDYYWNWIUTPXKVXSHDCBC WVYDGBVJLMCPZJROXKDPTDTMC PHXGCTHPDLVHYQHHFRTTKSOTE IWAXEDMUOVBLSLZUWFTYGNCQY YPHZRNJRBXYVVSNPYWAEMXOIV UQWAXAECBOODIPLWGCVQJVDCX GKCBXHCUK	TOZHAVEZNOZSCREENZBETWEEN ZTHISZPARTZHEZPLAYDZANDZH IMZHEZPLAYDZITZFORZHEZNEE DSZWILLZBEZABSOLUTEZMILAN	No hint given.

Table 1: New SIGABA Challenges

Reference Source Code for SIGABA Simulator

- **Used to create the challenges**
- **Validated against:**
  - Pekelney (1998)
    - Itself validated against real machine
  - Sullivan (2002)

5 Appendix – Source Code and Challenges

Listing 1: SIGABA Simulator Source Code

```
package simulator;

class Sigaba {
    private Rotor cipherBank[] = new Rotor[5];
    private Rotor controlBank[] = new Rotor[5];
    private IndexRotor indexBank[] = new IndexRotor[5];
    Sigaba(String cph, String ctl, String idx,
           String cphP, String ctlP, String idxP) {
        for (int i = 0; i < 5; i++) {
            cipherBank[i] =
              new Rotor(cph.charAt(i * 2) - '0',
                        cph.charAt(i * 2 + 1) == 'R',
                        cphP.charAt(i) - 'A') ;
            controlBank[i] =
              new Rotor(ctl.charAt(i * 2) - '0',
                        ctl.charAt(i * 2 + 1) == 'R',
                        ctlP.charAt(i) - 'A');
            indexBank[i] =
              new IndexRotor(idx.charAt(i) - '0',
                             idxP.charAt(i) - '0');
        }
    }
    String encryptDecrypt(boolean decrypt, String in) {
        String outString = "";
        for (char c : in.toCharArray()) {
            outString +=
              (char)(cipherPath(decrypt, c - 'A') + 'A');
            cipherBankUpdate();
            controlBankUpdate();
        }
        return outString;
    }
    private void controlBankUpdate() {
        if (controlBank[2].pos == (int) 'O' - 'A') {
            // medium rotor moves
            if (controlBank[3].pos == (int) 'O' - 'A') {
                // slow rotor moves
                controlBank[1].advance();
            }
            controlBank[3].advance();
        }
        // fast rotor always moves
        controlBank[2].advance();
    }
    private static final int INDEX_IN[] =
                {9, 1, 2, 3, 3, 4, 4, 4, 5, 5, 5, 6, 6,
                 6, 6, 7, 7, 7, 7, 7, 8, 8, 8, 8, 8, 8};
    // rotor stepping magnet
    private static final int INDEX_OUT[] =
            {1, 5, 5, 4, 4, 3, 3, 2, 2, 1};
    private void cipherBankUpdate() {
        boolean move[] = new boolean[5];
        for (int i = (int) 'F' - 'A';
             i <= (int) 'I' - 'A';
             i++) {
          int indexIn = INDEX_IN[controlPath(i)];
          move[INDEX_OUT[indexPath(indexIn)] - 1] = true;
        }
        for (int i = 0; i < 5; i++) {
          if (move[i]) cipherBank[i].advance();
        }
    }
    private int cipherPath(boolean decrypt, int c) {
        if (decrypt) {
          for (int r = 4; r >= 0; r--)
            c = cipherBank[r].rightToLeft(c);
        } else {
          for (int r = 0; r <= 4; r++)
            c = cipherBank[r].leftToRight(c);
        }
        return (c);
    }
    private int controlPath(int c) {
        for (int r = 4; r >= 0; r--)
          c = controlBank[r].rightToLeft(c);
        return (c);
    }
    private int indexPath(int c) {
        for (int r = 0; r <= 4; r++)
          c = indexBank[r].indexPath(c);
        return (c);
    }
    static class Rotor {
        private static final String[] WIRINGS = {
                "YCHLQSUGBDIXNZKERPVJTAWFOM",
                "INPXBWETGUYSAOCHVLDMQKZJFR",
                "WNDRIOZPTAXHFJYQBMSVEKUCGL",
                "TZGHOBKRVUXLQDMPNFWCJYEIAS",
                "YWTAHRQJVLCEXUNGBIPZMSDFOK",
                "QSLRBTEKOGAICFWYVMHJNXZUDP",
                "CHJDQIGNBSAKVTUOXFWLEPRMZY",
                "CDFAJXTIMNBEQHSUGRYLWZKVPO",
                "XHFESZDNRBCGKQIJLTVMUOYAPW",
                "EZJQXMOGYTCSFRIUPVNADLHWBK"};
        // Index for left to right.
        private static final int TO_RIGHT = 0;
        // Index for right to left.
        private static final int TO_LEFT = 1;
        private int wiring[][] = new int[2][26];
        int pos;
        private boolean reversed;
        Rotor(int wiringIndex, boolean reversed, int pos) {
            for (int i = 0; i < 26; i++) {
              wiring[TO_RIGHT][i] =
                WIRINGS[wiringIndex].charAt(i) - 'A';
              wiring[TO_LEFT][wiring[TO_RIGHT][i]] = i;
            }
            this.reversed = reversed;
            this.pos = pos;
        }
        void advance() {
            if (reversed) {
                pos = (pos + 1) % 26;
            } else {
                pos = (pos - 1 + 26) % 26;
            }
        }
        int leftToRight(int in) {
            if (!reversed) {
              return
                (wiring[TO_RIGHT][(in+pos)%26]-pos+26)%26;
            }
            return
              (pos-wiring[TO_LEFT][(pos-in+26)%26]+26)%26;
        }
        int rightToLeft(int in) {
            if (!reversed) {
              return
                (wiring[TO_LEFT][(in+pos)%26]-pos+26)%26;
            }
            return
              (pos-wiring[TO_RIGHT][(pos-in+26)%26]+26)%26;
        }
    }
    static class IndexRotor {
        private static final int WIRINGS[][] = {
                {7, 5, 9, 1, 4, 8, 2, 6, 3, 0},
                {3, 8, 1, 0, 5, 9, 2, 7, 6, 4},
                {4, 0, 8, 6, 1, 5, 3, 2, 9, 7},
                {3, 9, 8, 0, 5, 2, 6, 1, 7, 4},
                {6, 4, 9, 7, 1, 3, 5, 2, 8, 0}};
        private int wiring[] = new int[10];
        private int pos;
        IndexRotor(int wiringIndex, int pos) {
            System.arraycopy(WIRINGS[wiringIndex], 0,
              wiring, 0, 10);
            this.pos = pos;
        }
        int indexPath(int in) {
          return (wiring[(in + pos) % 10] - pos + 10) % 10;
        }
    }
    public static void main(String[] args) {
        Sigaba sigaba =
         new Sigaba("0R1N2N3N4R", "5N6N7R8N9N",
                    "01234", "ABCDE", "FGHIJ", "01234");
        String out = sigaba.encryptDecrypt(false,
          "AAAAAAAAAAAAAAAAAAAA");
        System.out.printf(
          "%s (expecting JTSCALXDRWOQKRXHKMVD) \n", out);
        sigaba =
          new Sigaba("0R1N2N3N4R", "5N6N7R8N9N",
                     "01234", "ABCDE", "FGHIJ", "01234");
        String in = sigaba.encryptDecrypt(true, out);
        System.out.printf(
          "%s (expecting AAAAAAAAAAAAAAAAAAAA)\n", in);
    }
}
```

Other Projects

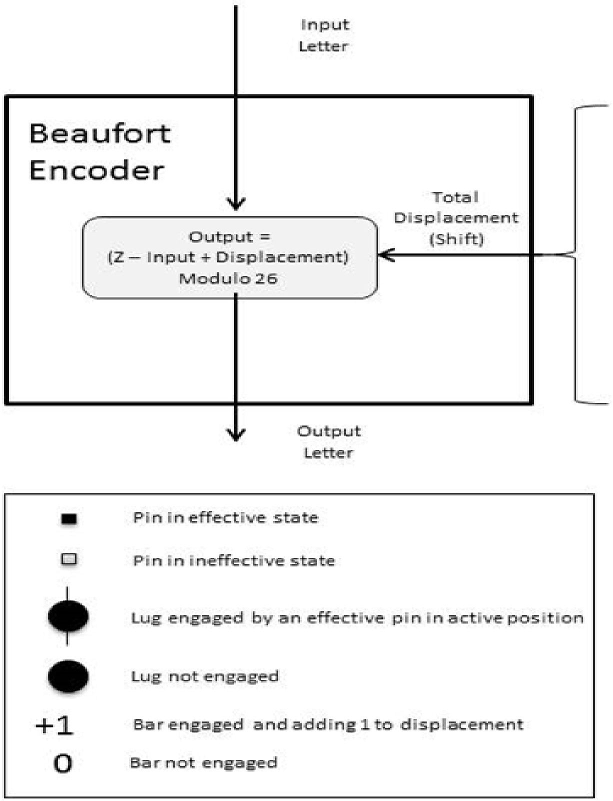
Project	Method	Results
ADFGVX	Divide and conquer, hillclimbing, IC and ngrams	600 original Eastern Front German cryptograms, 1918
Sturgeon T52	Divide and conquer, two phases, specialized scoring and monograms Backtracking for known-plaintext attack	Original German cryptograms from 1942
Vatican ciphers	Manual and computerized methods (e.g. simulated annealing)	Homophonic and polyphonic ciphertexts, from 16-18th cent.
WW1 Diplomatic codes	Mostly manual methods	1913-1915 German messages
Enigma - double indicators	Hillclimbing, specialized scoring	5-10 indicators required
Single transposition with long keys	Hillclimbing, two phases, specialized scoring and 4-grams	Key length up to 1000

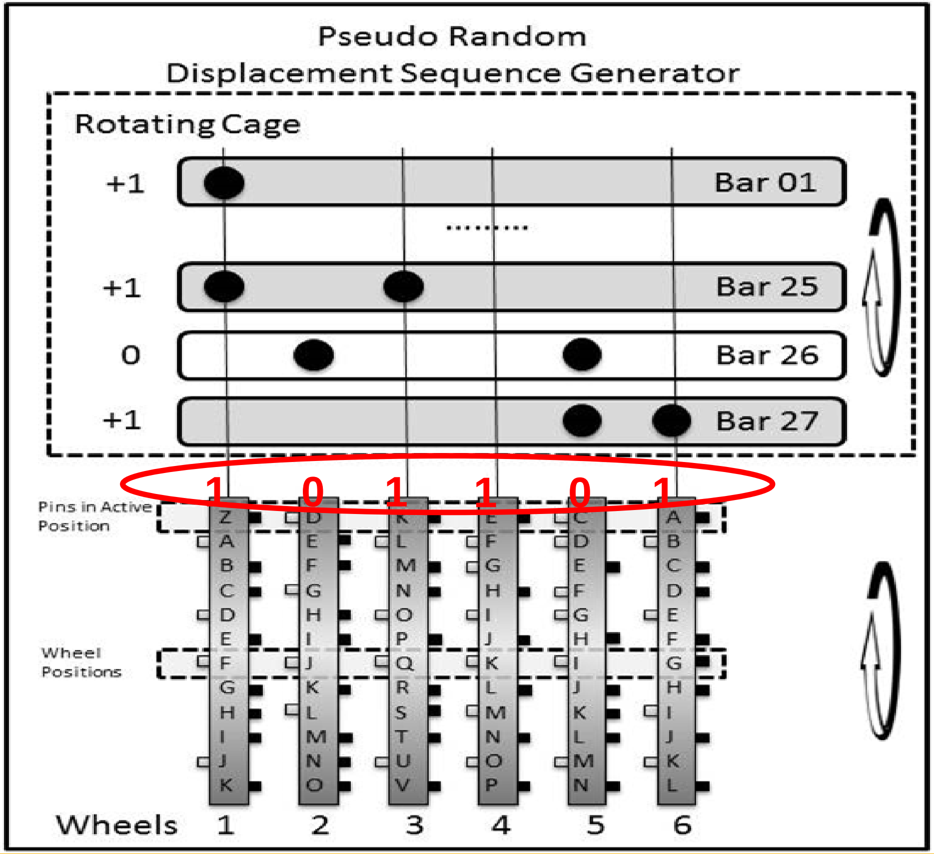
Thank You

November 1, 2019
George Lasry, Ph.D.
george.lasry@gmail.com

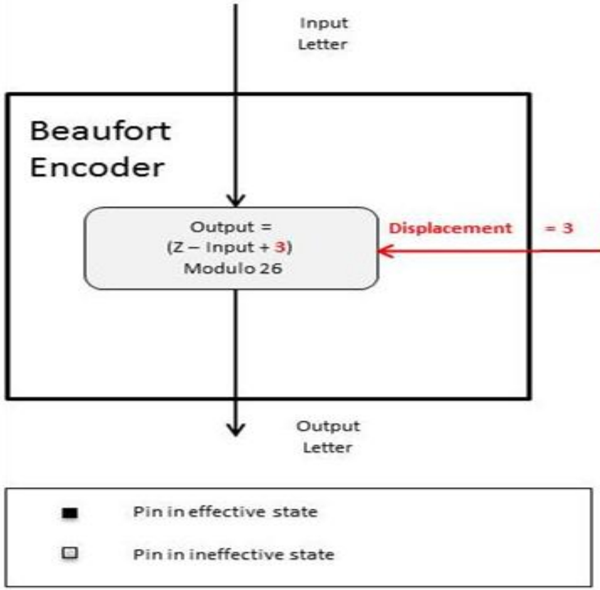
- George Lasry, Solving a 40-Letter Playfair Challenge with CrypTool 2, Proceedings of the 2nd International Conference on Historical Cryptology, HistoCrypt 2019, June 23-26, 2019, Mons, Belgium, fulltext
- George Lasry, A Practical Meet-in-the-Middle Attack on SIGABA, Proceedings of the 2nd International Conference on Historical Cryptology, HistoCrypt 2019, June 23-26, 2019, Mons, Belgium, fulltext
- George Lasry, A Methodology for the Cryptanalysis of Classical Ciphers with Search Metaheuristics, Kassel University Press, Ph.D. Thesis, 2018, fulltext
- Full list of publications

Classes of Active Pins

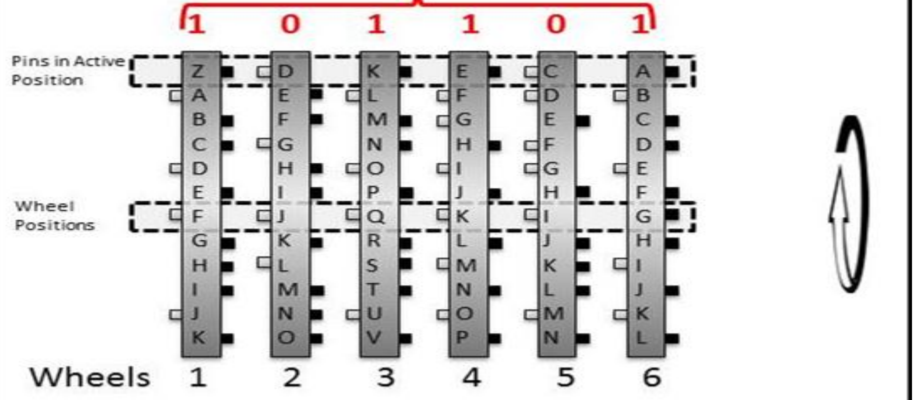




Classes of Active Pins

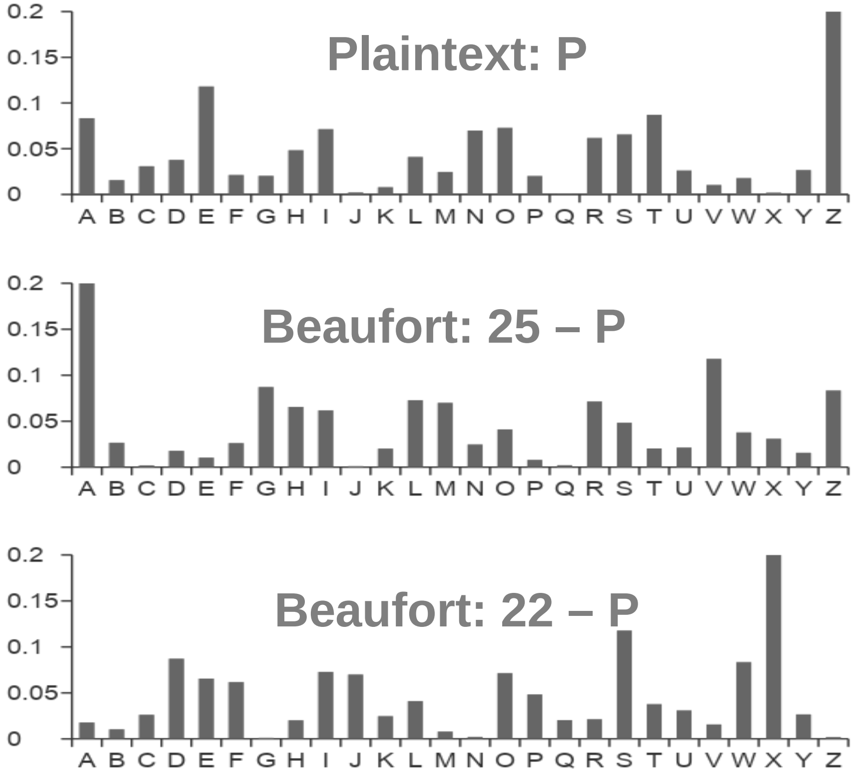


Displacement	Class
0	000000
7	000001
…	…
1	101100
3	101101
…	…
12	111110
25	111111

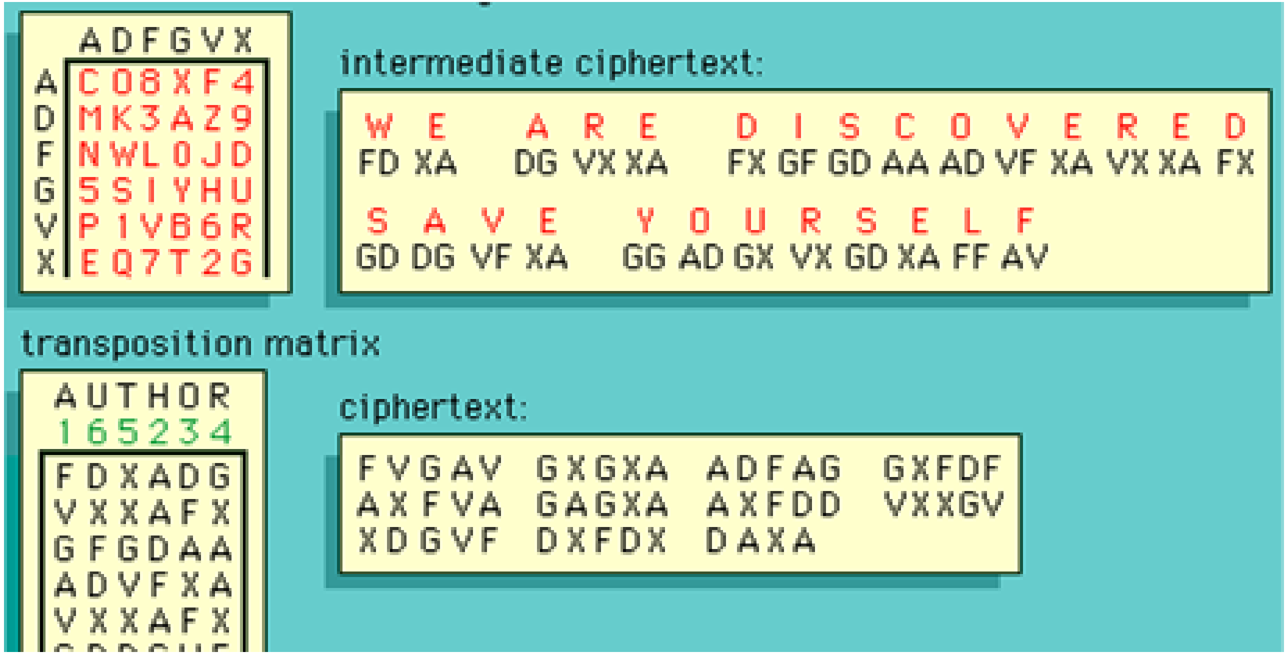


Classes of Active Pins

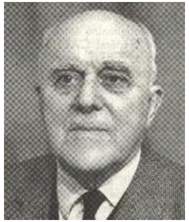
Class	Displacement D	Beaufort Encryption 25 – P + D Modulo 26
000000	0	25 – P
000001	7	18 – P
…	…	…
101100	1	24 – P
101101	3	22 – P
…	…	…
111110	12	13 – P
111111	25	0 – P



ADFGVX



Substitution + Fractionation + Columnar Transposition

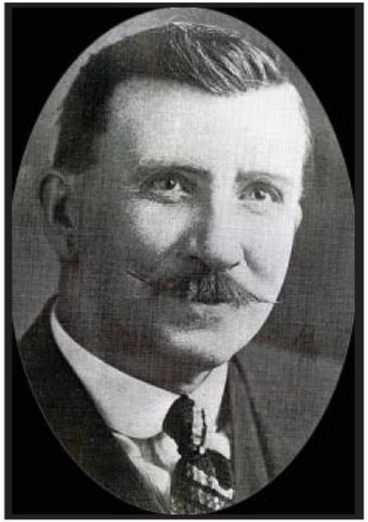


Fritz Nebel
1891-1967

Georges Painvin
1886-1980

Before SIGABA – Hebern Cipher Machines – 1920s

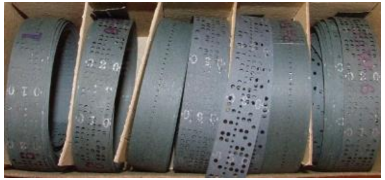
- 5 cipher rotors
- Regular stepping

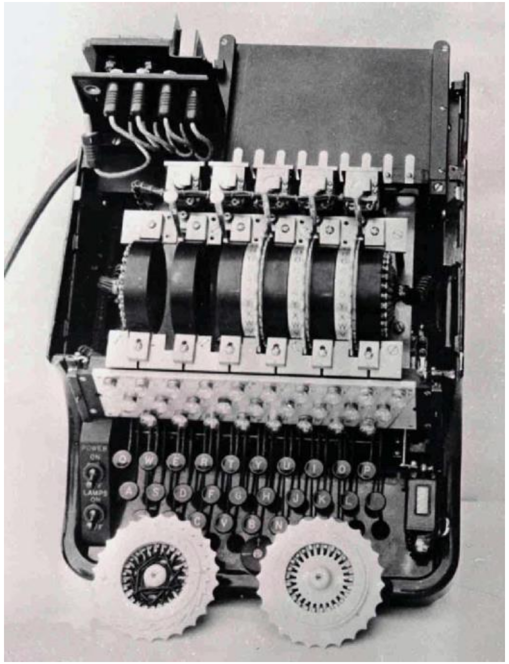


Edward Hebern and his electromechanical rotor cipher machine

History of SIGABA – William Friedman's Design

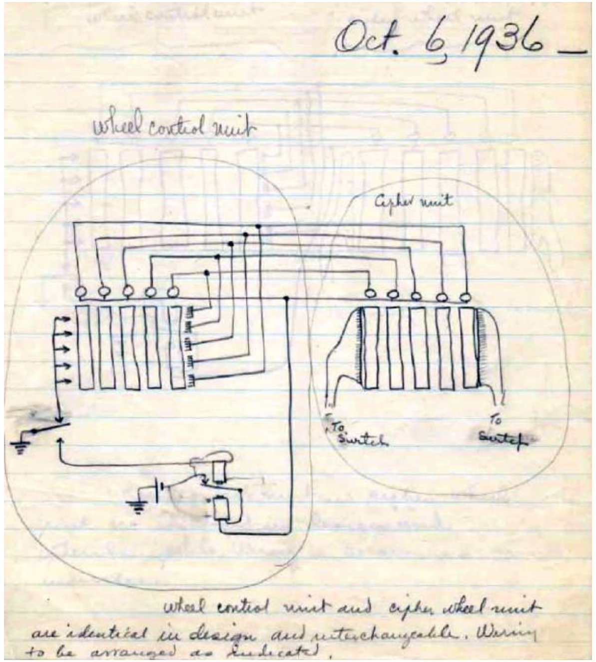
- **5 cipher rotors**
- **Irregular stepping**
  - Punched taps
  - Plugboard





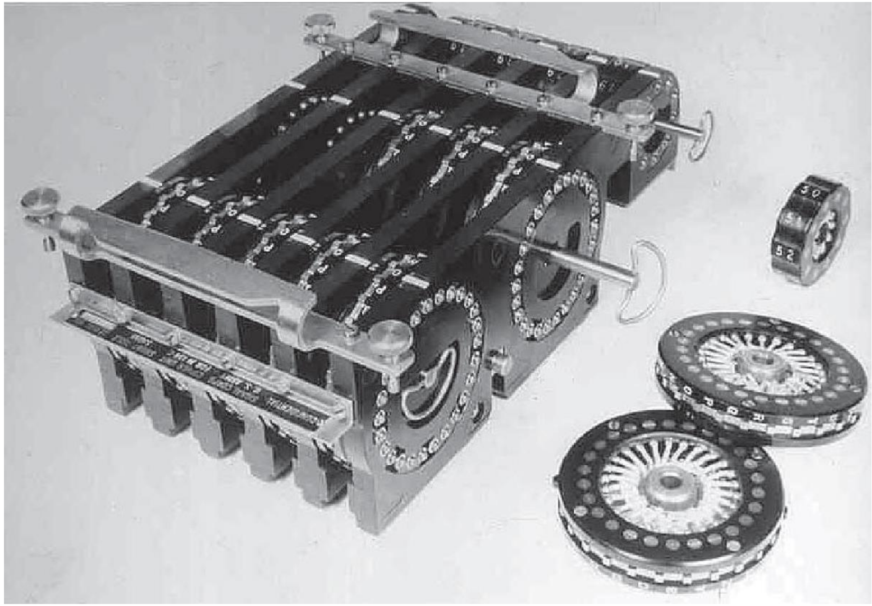
SIGABA

- **5 cipher rotors**
- **Irregular stepping**
  - 5 „control“ rotors
  - Plugboard



History of SIGABA – Final Design – US Navy

- **5 cipher rotors**
- **Irregular stepping**
  - 5 „control“ rotors
- **No plugboard**
  - 5 „index“ rotors



Time-Memory Trade-off

- **Process only 8 known-plaintext symbols**
  - Rotors step up to 7 times
- **Less than 8 - slower**
  - More false positives
- **More than 8 - more memory**
  - More matching sequences
- **Pruning false positives**
  - Use additional known-plaintext symbols
  - Or Index of Coincidence after decrypting ciphertext

Stepping Sequence (Hash Key)		(Maps to) **Cipher Rotor Settings**					
		Rotor Selection					**Starting**
		1	**2**	**3**	**4**	**5**	**Positions**
01011 01000 11001 00111 00111 11110 00010	⇛	8R	0	4R	7	1	H Y J N H
	⇛	1	7R	0	8R	4	T U A L M
01111 01100 11100 11001 00010 10101 10001	⇛	1R	4	8R	7	0	K H J N M
11010 10011 00111 10101 00111 00110 10011	⇛	0	8R	7R	4	1R	E Q A M B