Fall 2015

Modern Crypto-Analysis of Polyalphabetic Ciphers Using a Genetic Algorithm

Benjamin E. Heald
Gettysburg College

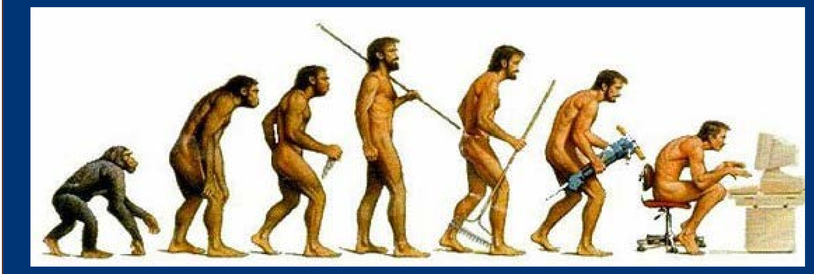
Modern Crypto-Analysis of Polyalphabetic Ciphers Using a Genetic Algorithm

Abstract
This project involved implementing a genetic algorithm to help automate the crypto-analysis of the Vigenere cipher.

Keywords
Cryptography, Genetic, Algorithm, Vigenere

Disciplines
Mathematics

Comments
This poster was made for Professor Darren Glass's First Year Seminar, *FYS 146-2: Cryptography: The Science of Secrecy*, Fall 2015. It was presented as part of the first CAFE Symposium, 2016.



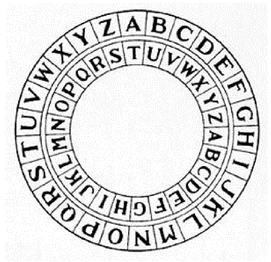
Modern Crypto-Analysis of Polyalphabetic Ciphers Using a Genetic Algorithm

Benjamin Heald

Gettysburg COLLEGE

The Vigenere Cipher

- Invented in 1553 by Giovan Battista Bellaso, in his book “La cifra del. Sig. Giovan Battista Bellaso”.
- Later misattributed to Blaise de Vigenère
- Was considered for hundreds of years to be “unbreakable”
- Broken in 1854 by Charles Babbage.
- Is a combination of many so-called mono-alphabetic ciphers.



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

Automating the Process

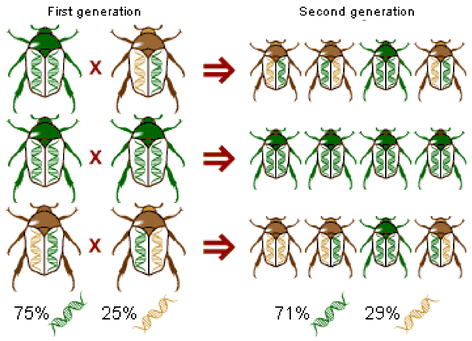
Step One: Finding the Key Length

- Kasiki Test to find the most likely candidates, I.E. 2, 4, 8.
- These candidates are then analyzed by the Friedman Method to determine which most resembles english text.
- The standard “Index of coincidence” for english text is 1.73
- The candidate with the “IC” closest to this ideal is most likely the key length
- This combination of the two algorithms makes it much easier for a basic computer program to find the most likely key length.

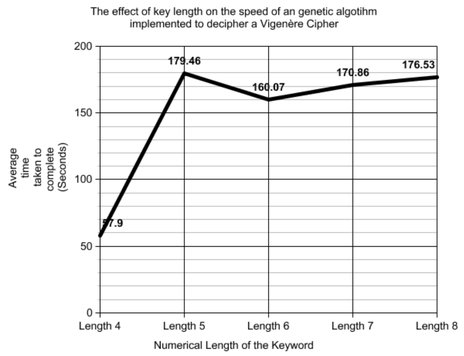
Step Two: Finding the Key Text

- With a known key length, a variety of attacks may now be used.
- Without human interaction however, it is very difficult for the computer to know if it has arrived at a solution.
- **Method One: Brute Force Attack**
- **Method Two: Common-Word Dictionary Attack**
- **Method Three: Markov Chain**
- **Method Four: A Genetic Algorithm**

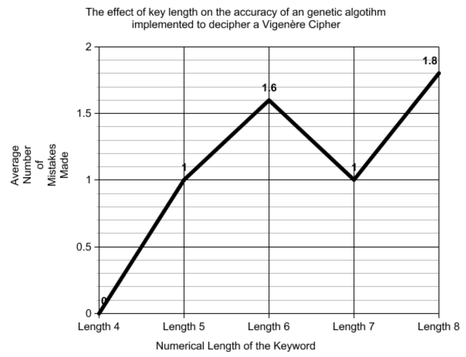
The Genetic Algorithm



Key Length and Speed



Key Length and Accuracy



Experimental Data

TRIAL #	CHAR. COUNT	KEY LENGTH	TIME	RESULT	ACTUAL KEY	LETTERS INCORRECT	MAX GENERATIONS
1	1500	4	45.11	TEXT	TEXT	0	500
2	1500	4	73.13	TEXT	TEXT	0	500
3	1500	4	55.42	TEXT	TEXT	0	500
4	1500	4	60.74	TEXT	TEXT	0	500
5	1500	4	55.13	TEXT	TEXT	0	500
AVERAGE			***57.9***			***0***	
1	1500	5	179.45	TEXTP	TEXTB	1	500
2	1500	5	179.19	TEXTP	TEXTB	1	500
3	1500	5	179.43	TEXTP	TEXTB	1	500
4	1500	5	179.55	TEXTP	TEXTB	1	500
5	1500	5	179.67	TEXTP	TEXTB	1	500
AVERAGE			***179.46***			***1***	
1	1500	6	180.15	TEBTBO	TEXTBO	1	500
2	1500	6	179.57	TEXSPO	TEXTBO	2	500
3	1500	6	86.51	TEBTBO	TEXTBO	1	500
4	1500	6	180.02	TEXPSO	TEXTBO	2	500
5	1500	6	172.1	TEXPSO	TEXTBO	2	500
AVERAGE			***160.07***			***1.6***	
1	1500	7	171.38	TEXTBSO	TEXTBOO	1	500
2	1500	7	171.59	TEXTBZO	TEXTBOO	1	500
3	1500	7	170.67	TEXTBSO	TEXTBOO	1	500
4	1500	7	169.64	TEXTBSO	TEXTBOO	1	500
5	1500	7	171.04	TEXTBZO	TEXTBOO	1	500
AVERAGE			***170.66***			***1***	
1	1500	8	174.36	TEMTBFOK	TEXTBOO	2	500
2	1500	8	149.53	TCXTCOOK	TEXTBOO	2	500
3	1500	8	186.91	TEMTBDOK	TEXTBOO	2	500
4	1500	8	186.16	TPMTBOOK	TEXTBOO	2	500
5	1500	8	185.71	TEXTBZOK	TEXTBOO	1	500
AVERAGE			***176.53***			***1.8***	

Data Conclusions

- This algorithm finds almost 70% of the correct letters in keywords of length greater than four.
- With further optimization however, this number reaches 98%
- This experiment was carried out on a older model PC, with newer hardware, much faster speeds could be achieved.

Summary

- Through combining many different algorithms, the key length can be found very reliably.
- The Genetic Algorithm is best used when combined with the Friedman Method.
- The Genetic Algorithm, once properly implemented, is the best and most accurate way to automate the decipherment process.
- Without human interaction, the entire process becomes much harder.
- Though not perfect, this program is relatively accurate for all key lengths.
- With greater key lengths, the more genetic “generations” are needed to arrive at the solution.
- Automating the crypto-analysis makes finding the key much faster than when done by hand.
- With newer hardware, the analysis could be done even faster.
- When incorporated into a web interface, the program is very accessible.

Acknowledgments

• Professor Glass for his First-Year seminar on the science and history behind Cryptography