AN INTERACTIVE CRYPTANALYSIS ALGORITHM FOR THE VIGENERE CIPHER

Mehmet E. Dalkilic and Cengiz Gungor
Ege University, Intl. Computer Inst.
Bornova 35100 Izmir, TURKEY
dalkilic@bornova.ege.edu.tr

Abstract

Though it dates back centuries, Vigenere Cipher is still a practical encryption method that can be efficiently used for many applications. We have designed and implemented an interactive cryptanalysis software based on the Kasiski test, and a novel use of the Index of Coincidence (IC) concept. Our results show that cryptanalysis is possible for very short text lengths where classical cryptanalysis methods fail. Furthermore, we have observed that our software which is designed to work on English based ciphertexts, can be successfully executed on ciphertexts based on other languages. Along the way, we also compute and report the IC values for Turkish and some other languages under a different number of enciphering alphabets.

1 Introduction

The use of cryptosystems for sensitive communications for military and diplomatic purposes dates back to the time of ancient Egyptians as the story told by Kahn[1], the best known reference to the history of the secret writing, or cryptography. Traditionally cryptography has been exclusively used by governments and large corporations. However, with the widespread use of computers and the high availability of sophisticated communication networks, now most individuals and corporations need special protection of their private or business data[1] by cryptosystems.

The conventional cryptosystems are formulated by C.E. Shannon in a landmark paper at 1949[3]. We can define a cryptosystem as a five-tuple [4] $(\mathcal{P}, \mathcal{K}, \mathcal{C}, \mathcal{E}, \mathcal{D})$ where $\mathcal{P}$ is a finite set of plaintext, $\mathcal{K}$ is a finite set of keys, $\mathcal{C}$ is a finite set of ciphertexts, and for each $k \in \mathcal{K}$, there is an encryption rule $E_k \in \mathcal{E}$ and its inverse (decryption) rule $D_k = E_k^{-1} \in \mathcal{D}$.

An encryption rule is a one-to-one function that takes any plaintext message, $p \in \mathcal{P}$, and transforms it to an unintelligible form called a ciphertext message with the intend to conceal

[1] Storing data can be seen as transmission of the data in the time domain [2]. Therefore, the term transmission (or communication) refers to any situation where data are stored and/or transmitted.

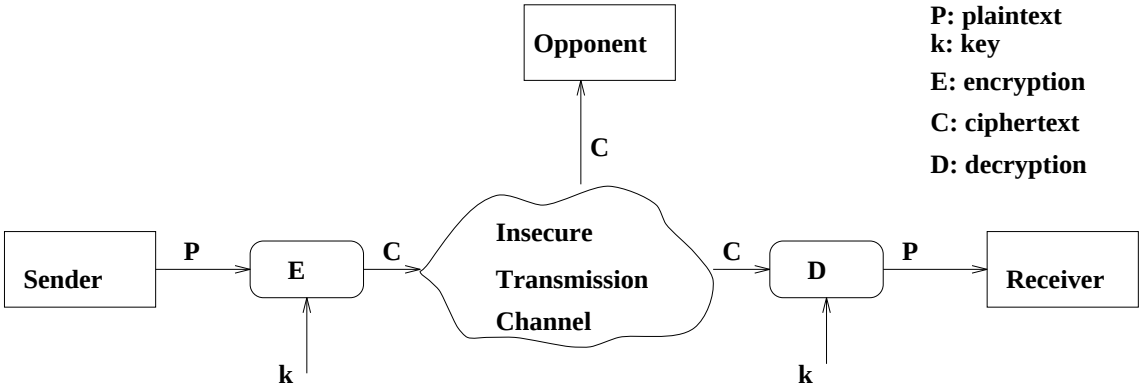

Figure 1: A Simple Model of Conventional Cryptosystem

meaning. The decryption rule recovers the original plaintext from a given ciphertext. A simple model of a crytosystem is illustrated in Figure 1.

In this paper, we have studied a well-known classical cipher, Vigenere. Dating back to 16th century [5], Vigenere cipher is a polyalphabetic substitution cipher. With a random key equal to the size of the plaintext Vigenere cipher becomes a *one-time pad* which is the only proven unbreakable cipher in the history. We have developed an interactive computer program to cryptanalyze the Vigenere cipher. Our program brings together classical methods such as Kasiski Test and the Index of Coincidence with a new concept, Mutual Index of Coincidence [6].

In the next section, we briefly describe the Vigenere cipher, and explain our approach to determine the cipher period, and the keyword. Section 3 outlines our interactive cryptanalysis algorithm follwed by the experimental results in section 4. Conclusions and further research are outlined in the last section.

2 Cryptanalysis of the Vigenere Cipher

Suppose we have an n-character alphabet $A = (a_1, a_2, ..., a_n)$, an m-character key, $K = (k_1, k_2, ..., k_m)$ and a t-character plaintext, $M = (m_1, m_2, ..., m_t)$. Then, we define a Vigenere cipher $E_K(a_i) = (m_i + k_i \bmod n)$ and $D_K(c_i) = (c_i - k_i \bmod n)$.

The Vigenere cipher uses m shift ciphers where each k_i determines which of the n mono-alphabetic substitutions to be used.

The strength of the Vigenere cipher is that, as it can be seen from the above example, a plaintext letter can be represented by up to m different ciphertext letters. Nevertheless, the Vigenere cipher can not obscure the structure of the plaintext completely.

Suppose we have the following ciphertext that we believe it was encrypted using a Vigenere cipher.

```
WERXEHJVYSOSPKMUVCOGSIXFUFLTHTVYCBTWPTMCLHTRGCMGQEAGRDVFEGTDJPFPWPGVLIASCS
```

```
GABHAFDIASEFBTVZGIIIHDGIDDKAVYCCXQGJQPKMVIYCLTQIKPMWQEQDYHGEMCTPCKRAXTKVJS
PWVYJXMHNVCFNWRDCCMVQNCKXFVYCSTBIVPDYOEFBTVZGIIQXWPXAPIHWICSUMVYCTGSOPFPLA
CUCXMSUJCCMWCCRDUSCSJTMCEYYCZSVYCRKMRKMVKOJZABXHJFBXGGVRLIEMKWLTXRDV
```

2.1 Determining the Cipher Period, m

Kasiski and the Index of Coincidence tests are commonly used to determine the key size. **Kasiski test** is based on the observation that if two identical fragments of length three or more appears in the ciphertext, it is likely[2] that they correspond to the same plaintext fragment. The distance between the occurrences has to be a multiple of the period.

In our example, the segment VYC appears four times beginning at position 31, 103, 175, 211 and 253.

Starting Position	Distance from Previous	Factors
103	72	2,3,4,6,8,9,12,18,24,36,72
175	72	2,3,4,6,8,9,12,18,24,36,72
211	36	2,3,4,6,9,12,18,36
253	42	2,3,6,7,14,21,42

Common factors 2, 3, and 6 are the key length candidates to be tested by the Index of Coincidence (IC) defined below.

Def: Let $\mathbf{x} = \{x_1 x_2 ... x_n\}$ be a string of n alphabetic characters. The index of coincidence of $\mathbf{x}$, IC($\mathbf{x}$), is the probability that randomly chosen two elements of string $\mathbf{x}$ are the same[4, 5].

When applied to a natural language, the index of coincidence is like a fingerprint of the standart distribution. Therefore, the IC can be used to determine if ciphertext substrings' frequency distribution resemble standard distribution of the source language[3]. For a small sample the match would not be exact, but a close correspondence is usually sufficient. We call this test **IC-substring test**.

To try $m = 6$ we divide the ciphertext into 6 substrings so that each substring is encrypted by the same key letter i.e., $S_1 = \{c_1, c_7, c_{13}, ...\}, S_2 = \{c_2, c_8, c_{14}, ...\}, ..., S_6 = \{c_6, c_{12}, c_{18}, ...\}$. If our guess about m is correct, each of these substrings will have an IC value close to IC of the source language, in our example English. For instance, $IC(S_1) = 0,0714, IC(S_2) = 0,0672, IC(S_3) = 0,0771, IC(S_4) = 0,0745, IC(S_5) = 0,0585$ and $IC(S_6) = 0,0674$. Considering, the IC of standard English is around 0,068, $m = 6$ is correct with very high probability. If our guess is not correct, then the substrings will be more like random strings with much smaller IC values. For instance, if we have tried $m = 5$ that is $S_1 = \{c_1, c_6, c_{11}, ...\}, S_2 = \{c_2, c_7, c_{12}, ...\}, ..., S_5 = \{c_5, c_{10}, c_{15}, ...\}$. Then we would obtain $IC(S_1) = 0,0448, IC(S_2) = 0,0369, IC(S_3) = 0,0484, IC(S_4) = 0,0375$ and $IC(S_5) = 0,0478$.

[2] The likelihood of two three-letter sequences not being from the same plaintext fragment is $1/n^3 = 0,0000569$ for n=26 [7].

[3] The IC values for Turkish are computed for three different letter frequency studies and it is shown in Table 1.

Table 1: Index of Coincidence values for Turkish

	IC	IC†	First ten	First ten†	Sample size
Koltuksuz[8]	0,0582	0,073083	{a,e,i,n,r,l,ı,d,k,m }	{i,a,e,n,r,l,u,s,d,k }	5,321,885
Goksu[9]	0,0608	0,072213	{a,e,i,n,l,r,ı,k,d,t }	{i,a,e,n,l,r,u,s,k,d }	574,004
Dalkilic[10]	0,0597	0,071455	{a,e,i,n,r,l,ı,k,d,m }	{i,a,e,n,r,l,u,s,k,d }	1,115,919

†: Turkish text written in English Alphabet

Table 2: Cipher period vs. Index of Coincidence

	$m=1$	$m=2$	$m=3$	$m=4$	$m=5$	$m=6$	$m=7$	$m=10$	$m=\infty$
IC(Turkish)	0,0597	0,0470	0,0428	0,0407	0,0395	0,0386	0,0380	0,0370	0,0344
IC(French)	0,0778	0,0589	0,0526	0,0494	0,0475	0,0463	0,0454	0,0437	0,0400
IC(German)	0,0762	0,0573	0,0510	0,0478	0,0460	0,0447	0,0438	0,0422	0,0384
IC(English)	0,0667	0,0525	0,0478	0,0455	0,0441	0,0431	0,0424	0,0412	0,0384
IC(Russian)	0,0529	0,0431	0,0398	0,0382	0,0372	0,0365	0,0361	0,0352	0,0333
IC(Tr-Eng)	0,0715	0,0549	0,0494	0,0467	0,0450	0,0439	0,0431	0,0417	0,0384
IC(Tr-Eng‡)	0,0720	0,0533	0,0475	0,0473	0,0450	0,0438	0,0418	–	–

Tr-Eng: Turkish text written in English Alphabet

‡: Empirical results

Another use of the IC is that it can directly predict the period given that the amount of ciphertext is sufficiently large and the original plaintext has a normal distribution. It is possible to predict the expected IC value for a cipher of period m, and tabulate the results in a table such as Table 2 for different m values using the formula [11, 5]:

$$Expected(IC) = \frac{1}{m}\frac{S-m}{S-1}(IC(SourceLanguage)) + \frac{m-1}{m}\frac{S}{S-1}(IC(RandomText))$$

By comparing IC(C) obtained from the ciphertext at hand to the values given in Table 2, we can get an estimate of m which may support the predictions of Kasiski test. For instance, in the ongoing example $IC(C) = 0,0429$ and from Table 2 (using the IC(English) row) we obtain the closest IC value is under $m = 6$ that matches the prediction of the Kasiski test. Nevertheless, IC is a good estimator of the cipher period only for small m, but its predictions are less accurate for larger m values. We call this test **IC-predict-m test**.

2.2 Determining the Keyword

We employ a new technique suggested by Dan Velleman [6] which uses mutual index of coincidence (MIC) in a very smart fashion.

The mutual index of coincidence of **x** and **s**, denoted MIC(x,s), is defined to be the probability that two random elements, one from each, are identical. Let **x** is a string of standard source language e.g. English. Let string **x** has length l and frequency distribution $r_1, r_2, ..., r_n$ where

First ten letters with highest frequency are also listed and almost perfectly matches in all three study.

Shift	String 1	String 2	String 3	String 4	String 5	String 6	Key
26	0,0314	0,0382	0,0344	0,0433	0,0356	0,0422	A
25	0,0358	0,0434	0,0315	0,044	0,0309	0,0385	B
24	0,0698	0,0429	0,0411	0,0461	0,0353	0,0317	C
23	0,0419	0,0331	0,0266	0,0262	0,0344	0,0499	D
22	0,038	0,0503	0,0364	0,0439	0,0498	0,0418	E
21	0,026	0,0463	0,0331	0,0335	0,0393	0,0409	F
20	0,0375	0,0389	0,0278	0,0336	0,0487	0,026	G
19	0,034	0,0358	0,0338	0,0285	0,04	0,0406	H
18	0,0409	0,0321	0,0416	0,0388	0,048	0,0394	I
17	0,0401	0,0265	0,0518	0,0377	0,0363	0,0307	J
16	0,036	0,0402	0,0422	0,0431	0,0388	0,0369	K
15	0,0329	0,0392	0,0412	0,0467	0,0332	0,0415	L
14	0,0315	0,0344	0,0332	0,0349	0,04	0,0361	M
13	0,0491	0,043	0,039	0,0302	0,0303	0,0293	N
12	0,0413	0,0337	0,0412	0,0444	0,0257	0,0645	O
11	0,0399	0,0342	0,0399	0,0686	0,0394	0,0469	P
10	0,0381	0,0449	0,0351	0,0364	0,0357	0,0303	Q
9	0,053	0,0645	0,0373	0,0334	0,0376	0,0239	R
8	0,0376	0,0353	0,0336	0,0263	0,0375	0,0504	S
7	0,0285	0,0292	0,0319	0,0368	0,0663	0,0402	T
6	0,0274	0,0444	0,048	0,0331	0,04	0,0336	U
5	0,0457	0,04	0,0374	0,0367	0,0364	0,0343	V
4	0,0368	0,0322	0,0322	0,0318	0,0313	0,0381	W
3	0,0319	0,0358	0,0416	0,039	0,041	0,0241	X
2	0,0435	0,0369	0,0704	0,0443	0,0336	0,0347	Y
1	0,0315	0,0248	0,0376	0,0387	0,035	0,0543	Z
	C	R	Y	P	T	O	

Figure 2: Mutual Indices of Coincidences

n is the size of the alphabet e.g. 26. Clearly, for large l, probability distributions of n letters r_i/l will be similar to the standard probability distributions p_i of the source language. Now, let string $\mathbf{y}$ has length l' and frequency distribution $r'_1, r'_2, ..., r'_n$. Then,

$$MIC(\mathbf{x}, \mathbf{s}) = \frac{\sum_{i=1}^{n} r_i r'_i}{ll'} \simeq \frac{\sum_{i=1}^{n} p_i r'_i}{l'}$$

Suppose $\mathbf{s_1}, \mathbf{s_2}, .., \mathbf{s_m}$ be the m substrings of the ciphertext $\mathbf{s}$. Each of these substrings are obtained by shift encryption of the unknown plaintext. Though plaintext is unknown at the moment, its probability distribution and therefore its IC is known. Consider, substring $s_j (1 \leq j \leq m)$ is encrypted by unknown key k_j. Suppose we shift s_j by b $(1 \leq b \leq n)$ and obtain n different s_j^b each of which corresponds to a decryption with a different key value. If s_j has frequency distribution $r''_1, r''_2, ..., r''_n$ and length l'', then

$$MIC(x, s_j^b) \simeq \frac{\sum_{i=1}^{n} p_i r''_{i-b}}{l''}$$

It is expected that $MIC(x, s_j^b) \simeq IC$(source language) if b is the additive inverse of the correct key modulo n; otherwise we obtain a much smaller MIC value. For instance if $n = 26$, and maximum MIC value for a substring is obtained when $b = 5$, then the probable key for that substring is $-5 \bmod 26 = 21$. For the example ciphertext our prediction is $m = 6$. So, we divide the ciphertext into 6 substrings, and we shift each substring by one, two, up to 26 and compute the MIC values shown in Table 2. By simply selecting the maximum MIC value for each substring we get a probable keyword, in our example it is **crypto**. To see whether the keyword we have obtained is correct we decrypt the ciphertext, and we get[4] a passage from Tanenbaum[12].

[4]Spaces are added to ease the reading.

Table 3: The Kasiski Test Results

No Prediction	Correct Prediction	Wrong Prediction
10	40	2
19.23%	76.92%	3.84 %

```
Until the advent of computers one of the main constraints on cryptography
had been the ability of the code clerk to perform the necessary transform
ations often on a battlefield with little equipment. However the danger of
a code clerk being captured by the enemy has made it essential to be able
to change the cryptographic method instantly if need be.
```

3 Interactive Cryptanalysis Algorithm

Figure 3 outlines the basic algorithm used in our interactive cryptanalysis program. For a given ciphertext, first apply the Kasiski test, and then run the IC-predict-m test. Then, using the predictions about the key length, m, select a likely m value. Note that, the results of the Kasiski test are sorted giving priority to those obtained from the occurrence of the largest segment size. For instance, suppose possible m values 6, 9 obtained from a segment of size three and 5, 11 obtained from a segment size of four. In that case, we give priority to 5, 11 values. Next, apply IC-substring test; If the results are promising, IC values of substrings are close to the IC of standard plaintext distribution, then we continue with the Mutual Index of Coincidence (MIC) test; otherwise, go back and select another m value.

After the MIC test, one gets a key string. If the key is correct then we have a **perfect run**; that is, the cryptanalysis program found the key in a single shot. If the key is not correct (i.e., the plaintext does not seem right) but there seems to be off by several positions (e.g., small segments of text are recognizable) then interactively modify the key with the help of the substring key alternatives provided by the cryptanalysis program. For instance, in Figure 2 if the key "C" were not the right one for substring S_1, then the second choice would be the key "N" which has the second highest MIC value at $0,0491$. Our program displays on the screen the first six choices for each substring. If you reach the solution either after trying multiple m values or test multiple key alternatives, that run is marked as **success** but not a perfect run. However, if you think we are not getting any closer to a solution, then go back to select another m. This process of trial and error continues until either a solution is found or all m values are exhausted. If the ciphertext could not be broken within a preset time limit, mark it as **failure**. Fortunately, as the following section demonstrates, most trials are either perfect runs or leads to success in a few try.

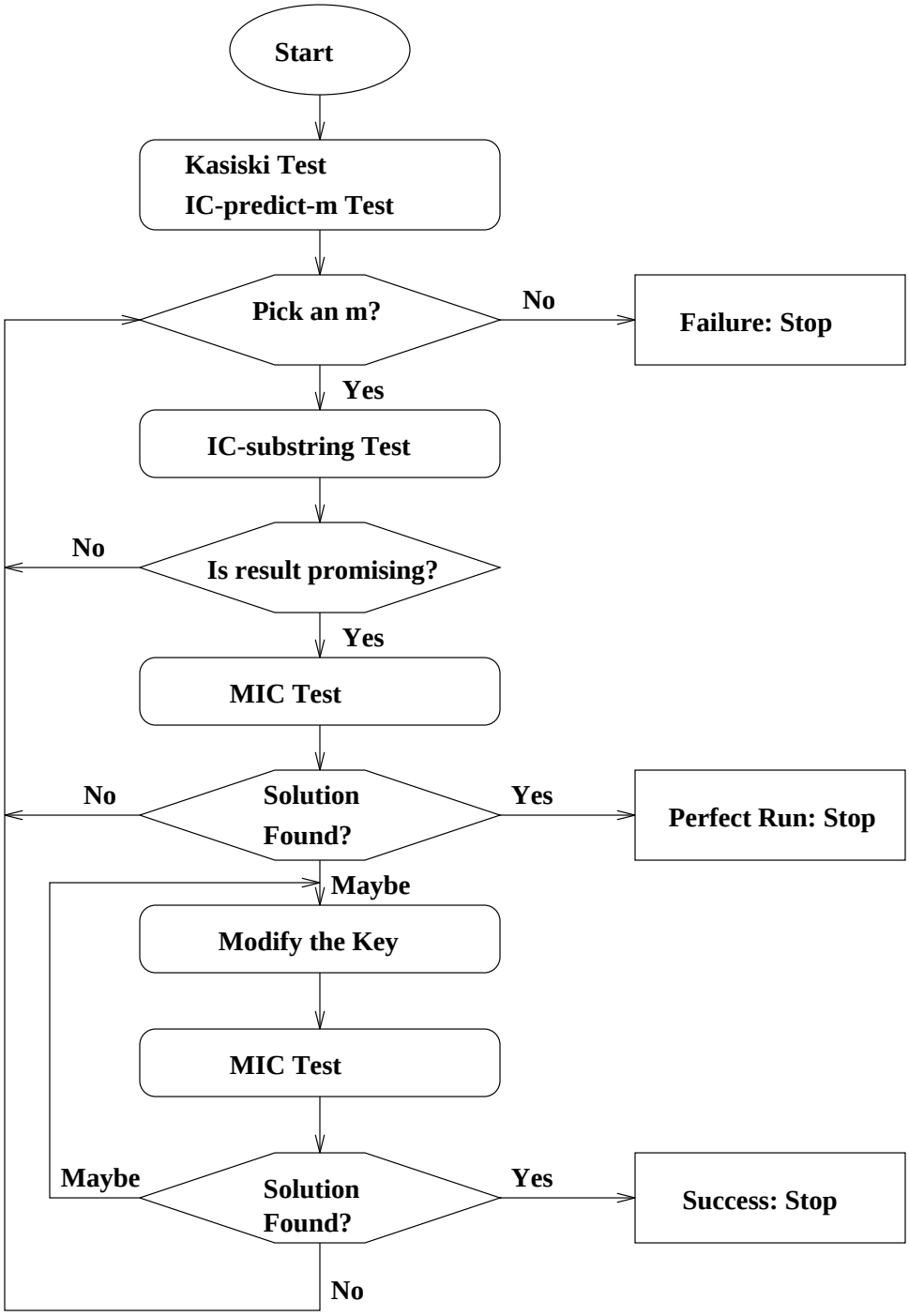

Figure 3: The Basic Flowchart of the Program

4 Experimental Results

To test our interactive cryptanalysis program we have used the following procedure. Each author prepared thirteen plaintext files of length 50, 75, 100, 125, 150, 200, 250, 300, 400, 500, 600, 800, 1000. We allowed up to 5% deviation on the size of the files. Then, each plaintext file is encrypted with two different keys. Key lengths vary 2 to 11 characters. Thus, a total of fifty-two different ciphertext files are attained. Then, the authors exchanged the ciphertext files but no plaintext files or key information. Next, the cryptanalysis program run on each of the ciphertext files. While runs on larger ciphertexts were almost always ended in perfect runs, the smallest ciphertext files were harder to break; five of them could not be broken within half an hour of time and left alone. Both authors have acquired similar success rates with their share of the test cases.

Kasiski test (see Table 3) for about one fifth of the test cases could not produced any key length (m) predictions. These cases happened to be the shortest ciphertext files where there were no repeated patterns in the ciphertext. Interestingly, there were also two cases were repeated patterns are accidental causing Kasiski test to output a wrong prediction. Nevertheless, the Kasiski test demonstrated its strength by generating the correct key length for over 76% of the cases where among more than half the time repeated patterns of length five discovered.

As it is shown in Table 4, the IC-predict-m test was on target less than one third of the time. For the remaining cases, overestimates of the key length (21 cases) were higher than the underestimates (16 cases). As expected, the IC-predict-m test is less reliable than the Kasiski test. However, where the Kasiski test has no output the IC-predict-m test is useful to guide the key search.

The overall results of our tests given in Table 5 show that half of the time our cryptanalysis program had perfect runs while less than one tenth of the time it failed. For the remaining two fifth of the cases, the key attained by interactively trying on the average 3.01 different m values. It usually takes few minutes of wall clock time to get a result on a test case. For the multiple-m trial cases when the correct key length is provided, the MIC test acquired the exact key at 38% of the time. If we add to it the perfect runs, for about 70% of the time MIC test produced perfect results. Again for the multiple-m trial cases with the right key length, about 62% of the time the average difference between the key generated by MIC test and the correct key was 0,6745 i.e., less than a single character.

In another test case we have fed plaintext files as ciphertext to the program to see if we can fool it. However, IC-predict-m test has immediately detected that the IC of the input is close

Table 4: The IC-predict-m Test Results

No Prediction	Correct Prediction	Wrong Prediction	
		under	over
0	16	15	21
0%	30.76%	28.84%	40.38%

Table 5: Overall Test results for the Program

No Result	Perfect Runs	Success		
		cases	m-trials	avg. difference
5	26	21	3.01	0.6745
9.62%	50%	40.38%	–	–

to normal plaintext distribution and concluded that m=1; that is, either we have a plaintext or a shift of a plaintext. The IC-substring test has determined that shift is 0 and returned our artificial ciphertext as plaintext.

Our Vigenere cryptanalysis program assumes that the underlying plaintext is in English. We wanted to see if the program will work if we feed it with ciphertexts where the source plaintexts are in another language e.g., Turkish. The success rates achieved were close to that of English based ciphertexts. Clearly, the Kasiski test finds repeated patterns and it is not in any way affected by a source language change. The index of coincidence tests (IC-predict-m and MIC test) are closely coupled with the letter distribution of the source language, and therefore they will be effected. Interestingly, the index of coincidence value of a Turkish text written in English alphabet, 0.0715 (see Table 2), is quite close to the IC value of standard English texts, 0,0667. Therefore, the IC based tests in our program still were able to generate results almost as good as those where the underlying text is English. Note that, from table 2, the IC value of Turkish (written in Turkish alphabet) is 0,0597 highly distanced to the IC of English at 0,0667. Therefore, it is plausible to assume that most natural languages written in English alphabet will have their index of coincidence values to move closer to that of English. As a consequence, our cryptanalysis program designed to work with English as its source language will work, with small degradation, on ciphertexts based on other languages.

Classical textbooks [e.g., Pfleeger[7]] state that the Kasiski test and IC-predict-m test works if there is a large body of ciphertext. Nevertheless, our experience shows that even for cases where under 100 characters available, satisfactory results can be attained with our approach. The shortest ciphertext instance that the program cryptanalyzed successfully is consisted of only 50 characters, and with a key length of 7 each substring had less than 8 characters to work with.

5 Conclusions and Future Work

We have reported an interactive cryptanalysis program for the Vigenere cipher. In addition to the classic cryptanalysis techniques such as Kasiski test and the IC-predict-m test, we have explored a recently proposed test which is based on the tabulation of the IC values of the shifted substrings for a possible key length. We have evaluated the performance of the individual tests and the overall performance of our cryptanalysis program. The new use of the IC concept proved to be exceptionally good leading to over 90% success rate of the program. In addition, our results show that cryptanalysis is possible for short text lengths where classical cryptanalsis approaches fail. We have also reported the index of coincidence values for Turkish and few

other languages under different number of enciphering alphabets.

Using the facts that (i) multi-round encryption greatly increases the strength of a cipher, and (ii) a ciphertext running key is extremely difficult to cryptanalyze, we work on the design and implementation of a a multi-round auto-cipher-key Vigenere cryptosystem . A formal strength analysis of our multi-round cryptosystems is due.

References

[1] D. Kahn, *The Codebreakers: The story of Secret Writing.* NewYork: Macmillan, 1967. (abriged edition, NewYork: New American Library, 1974).

[2] H. C. van Tilborg, *An Introduction to Cryptology.* Kluwer Academic Publishers, 1988.

[3] C. Shannon, "Communication Theory and Secrecy Systems," *Bell System Technical Journal*, vol. 28, pp. 656–715, Oct 1949.

[4] D. R. Stinson, *Cryptography: Theory and Practice.* CRC Press, 1995.

[5] A. Menezes, P. van Oorschot, and S. Vanstone, *Handbook of Applied Cryptography.* CRC Press, 1997.

[6] D. R. Stinson, "A More Efficient Method of Breaking a Vigenere Cipher," 1997. unpublished manuscript.

[7] C. P. Pfleeger, *Security in Computing.* Prentice-Hall, 1989.

[8] A. H. Koltuksuz, *Simetrik Kriptosistemler icin Turkiye Turkcesinin Kriptanalitik Olcutleri.* PhD thesis, Ege University, 1996. (in Turkish).

[9] T. Goksu and L. Ertaul, "Yer Degistirmeli, Aktarmali ve Dizi Sifreleyiciler Icin Turkce'nin Yapisal Ozelliklerini Kullanan Bir Kriptoanaliz," in *Proc. of the 3rd Symposium on Computer Networks*, pp. 184–194, June 1998. (in Turkish).

[10] M. E. Dalkilic and G. Dalkilic, "Language Statistics of Present-Day Turkish with Cryptology Applications," February 2000. (working paper).

[11] J. Seberry and J. Pieprzyk, *Cryptography: An Introduction to Computer Security.* Prentice Hall, 1989.

[12] A. S. Tanenbaum, *Computer Networks, 3rd Ed.* Prentice Hall, 1996.