Ciphertext-only attack on d×d Hill in O(d13)

Article *in* Information Processing Letters · September 2016

DOI: 10.1016/j.ipl.2016.09.006

CITATIONS
9

READS
342

2 authors:

Shahram Khazaei
Sharif University of Technology
40 PUBLICATIONS **471** CITATIONS
SEE PROFILE

Siavash Ahmadi
Sharif University of Technology
12 PUBLICATIONS **56** CITATIONS
SEE PROFILE

Ciphertext-only attack on $d \times d$ Hill in $O(d13^d)$

Shahram Khazaei * and Siavash Ahmadi **

Sharif University of Technology
shahram.khazaei@sharif.edu,s_ahmadi@ee.sharif.edu

Abstract. Hill is a classical cipher which is generally believed to be resistant against ciphertext-only attack. In this paper, by using a divide-and-conquer technique, it is first shown that Hill with $d \times d$ key matrix over $\mathbb{Z}_{26}$ can be broken with computational complexity of $O(d26^d)$, for the English language. This is much less than the only publicly known attack, i.e., the brute-force with complexity of $O(d^3 26^{d^2})$. Then by using the Chinese Remainder Theorem, it is shown that the computational complexity of the proposed attack can be reduced to $O(d13^d)$. Using an information-theoretic approach, supported by extensive simulation results, it is shown that the minimum ciphertext length required for a successful attack increases by a factor of about 7 and 9.8, respectively for these two attacks in comparison with the brute-force attack. This is the only serious attack on Hill since its invention in 1929.

Keywords: Hill cipher, ciphertext-only attack, classical ciphers, Chinese Remainder Theorem, entropy, redundancy

1 Introduction

Classical ciphers refer to a type of historically used ciphers which now have fallen into disuse. They are usually divided into substitution and transposition ciphers. In substitution ciphers, groups of letters are systematically replaced throughout the plaintext with other groups of letters, while in transposition ciphers, the letters themselves are kept unchanged, but their order within the plaintext is scrambled according to some well-defined scheme. There are several classical ciphers such as Caesar, monoalphabetic and polyalphabetic substitution ciphers, Vigenère square, Great, Morse Code, Pigpen, Scytale, Columnar, Chinese cipher, and so on [7, 5, 8].

Cryptanalysis of classical ciphers is usually a simple work, and most of them are broken not only with Known Plaintext Attacks (KPA), but also with Ciphertext-Only Attacks (COA). Of course, the requisite of breaking a cipher with COA is

* Shahram Khazaei is with the Department of Mathematical Sciences, Sharif University of Technology.

** Siavash Ahmadi is with the Information Systems and Security Lab (ISSL), Department of Electrical Engineering, Sharif University of Technology.

existence of some redundancy in the messages (e.g., an English text), else the cipher is information-theoretically secure.

Interestingly, some of these ciphers are still quite resistant against the COA. Our focus on this paper is on the Hill cipher [2], a polygraphic substitution cipher based on linear algebra, invented by Lester S. Hill in 1929. This cipher uses matrix multiplication of a $d \times d$ secret key matrix with $1 \times d$ plaintext blocks over $\mathbb{Z}_{26}$ to compute the ciphertext blocks. Because of linear property of the cipher, one can easily show that with KPA, the Hill cipher can be simply broken and the matrix of production (i.e., the key matrix) can be extracted. Nevertheless, it is generally accepted that mounting COA on Hill is much harder [10, 8].

The trivial exhaustive key space search on $d \times d$ Hill requires 26^{d^2} matrix multiplications. To the best of our knowledge, no better attack has ever been reported in the literature. We remark that taking into account the invertibility of the key matrix does not lead to a substantially improved attack. It is well-known (e.g., see [4]) that the number of invertible $d \times d$ matrices over $\mathbb{Z}_{26}$, i.e., the size of key space $\mathcal{K}$, can be calculated as follows:

$$|\mathcal{K}| = 26^{d^2} \prod_{i=1}^{d} (1 - 2^{-i})(1 - 13^{-i}) > 0.229 \times 26^{d^2}$$

As it can be seen, this leads to marginal improvement and the asymptotic complexity of the attack does not change, let alone that determining the invertibility of a matrix demands some extra effort. Although the complexity of 26^{d^2} for brute-force COA on Hill is common knowledge (see [11]), it seems that there is no report on the possibility of any further improvements in the literature. We have also reviewed several cryptography books. Some are content to say that "[Hill] can be difficult to break with a ciphertext-only attack" [8], some just say that "A ciphertext-only attack [on Hill] is harder" [10]. In addition the latter states that cryptanalysis based on letter frequency does not work because the Hill cipher encrypts blocks of letters together. Nonetheless, in this paper, we show that cryptanalysis based on letter frequency works just fine. In addition, we have surveyed several introductory crypto courses offered by renowned crypto scholars and noticed none of them pays enough attention to COA attack on Hill. One exception is a professor of mathematics at Northern Kentucky University [1] who discusses COA on a 2×2 Hill using bigram frequencies in his lecture notes. In another course at the Department of Computer Science of University of Rochester [3], lots of effort is made to explain the COA on 3×3 Hill, in a course project devoted to COA on Hill using a specific computer program. None of these methods can be extended to break even 4×4 Hill.

All the above reasons affirm the COA resistance of Hill, and in fact, we can say that the best publicly known COA on Hill cipher requires full search over all $O(26^{d^2})$ possible secret keys. To be more precise, this attack requires $O(26^{d^2})$ operations consisting of $O(1)$ multiplications of $d \times d$ square matrices. Therefore, if one does not bother to use fast algorithms for matrix multiplication such

as Strassen's method [9], the complexity of trivial brute-force attack will be $O(d^3 26^{d^2})$. In fact, by using Strassen's method, the complexity will reduce to $O(d^{\log_2 7} 26^{d^2}) \approx O(d^{2.807} 26^{d^2})$. However, a better approach is to, by eliminating the repeated calculations, tweak the attack to one with computational complexity of $O(d26^{d^2})$. We will not explore this possibility further but the idea will be clear when we present Algorithm 1 in Section 5.

We show that for the English language, a COA can be applied on Hill much faster than full search of key matrix elements (i.e., the brute-force attack). Our first key contribution leads us to a new COA that uses a divide-and-conquer technique by searching over each column of the decryption key matrix, separately. Here, by leveraging the non-uniformity of monograms, we perform the attack. The computational complexity of our COA is $O(d26^d)$ which is already dramatically lower than the previous ones. In our second contribution, using the Chinese Remainder Theorem (CRT), we devise an attack with computational complexity of $O(d13^d)$, leveraging non-uniformity of monograms when considered both modulo 2 and 13. This leads to practical COA on Hill with d as large as say 10 (notice that $10 \times 13^{10} \approx 2^{40.4}$ which is considered affordable on a typical PC in a tolerable amount of time).

The rest of paper is organized as follows. Section 2 presents the preliminaries of the paper. In section 3, Hill cipher will be described. COA on Hill using monograms and CRT-based divide-and-conquer attack are introduced in Section 4 and Section 5, respectively. Experimental results are presented in Section 6. Finally, we conclude the paper in Section 7.

2 Preliminaries

We need the following preliminaries to describe and analyze our attacks, most of which (expect Definitions 4 and 5) can be found in [8].

English language properties. Various people have estimated frequencies, i.e., probabilities of occurrence of the 26 letters of the English language. Let f_i denote the correct frequency of the i^{th} letter of English alphabet, also known as monogram frequencies. We use the frequencies reported in [8] as our reference. If X is a random variable with probability distribution of that of English monograms, then the entropy of X, define as $H_1 = \sum_i f_i \log_2 f_i$, is estimated to be $H_1 \approx 4.1718$. The frequencies of n-gram's and n-gram entropy, denoted by H_n, are similarly defined.

Definition 1 (natural entropy and redundancy). *The natural entropy H and the redundancy R of a language with alphabet $\mathcal{P}$ is defined as:*

$$H = \lim_{n\to\infty} \frac{H_n}{n},$$

$$R = 1 - \frac{H}{\log_2 |\mathcal{P}|}. \tag{1}$$

Experimental results for the natural entropy of English language, with $|\mathcal{P}| = 26$, shows that $1.0 \leq H \leq 1.5$. For later references, we use the more conservative lower bound $H \approx 1.0$.

Definition 2 (unicity distance). *The minimum ciphertext length required to break a cipher, i.e., determine the secret key almost uniquely, is called the unicity distance of the cipher.*

Using information theory concepts, it is straightforward to show that the unicity distance of a cipher can be calculated according to the following theorem.

Theorem 1. *The unicity distance of a cipher with key space $\mathcal{K}$, over a plaintext space with alphabet $\mathcal{P}$ and redundancy R, is:*

$$n_0 \approx \frac{\log_2 |\mathcal{K}|}{R \log_2 |\mathcal{P}|}.$$

We remark that many cryptanalysis techniques mount a COA only using the non-uniformity of monograms, instead of taking advantage of the full redundancy in the plaintext. This, however, causes to require a longer ciphertext to find the key almost uniquely. In this case, one should notice that, for calculating the the minimum ciphertext length, the redundancy R must be computed according to Eq. (1) with $H = H_1$ when using Theorem 1. The cryptanalysis techniques based on monogram frequencies usually use index of coincidence, defined below, to measure how well a string of English alphabet matches with English language in terms of monograms.

Definition 3 (index of coincidence). *The index of coincidence (IC) of a string P over English alphabet with the observed (normalized) frequency $\hat{f}_i$ for the i^{th} letter (i.e., $\sum_i \hat{f}_i = 1$) is defined as:*

$$\mathrm{IC}(P) = \sum_i \hat{f}_i f_i$$

For many ciphers, when a given ciphertext is decrypted using a random wrong key, it is reasonable to assume that the decrypted string is uniformly random. We call this assumption Simple Uniform Wrong Key Decryption (SUWKD). Under

SUWKD assumption, using statistical hypothesis testing similar to those used in correlation attacks [6], one can argue that the optimum criteria to measure how well a string matches with English language in terms of monograms, is the index of maximum likelihood, defined as follows.

Definition 4 (index of maximum likelihood). *The index of maximum likelihood (IML) of a string P over English alphabet with the observed (normalized) frequency $\hat{f}_i$ for the i^{th} letter is defined as:*

$$\mathrm{IML}(P) = -\sum_i \hat{f}_i \log_2 f_i$$

To the best of our knowledge, this is the first time that IML is used for cryptanalysis of classical ciphers instead of IC. Our simulation results in Section 6 verifies the optimality of IML. We formalize the notion of how well a string over English alphabet matches with English language using the following definition. We ignore to provide a similar definition based on IC due to its non-optimality.

Definition 5 (monogram-wise meaningful string). *We say that a string P over English alphabet is monogram-wise meaningful if the corresponding* IML(P) *is greater than a certain threshold.*

Let us discuss how a ciphertext C of a cipher with key space $\mathcal{K}$ can be broken using brute-force attack and based on monogram frequencies. To find a few candidates as the potential correct keys, one approach is to decrypt the ciphertext with all possible candidate keys $K \in \mathcal{K}$ and report those whose corresponding decrypted string is monogram-wise meaningful for an appropriately chosen threshold. Another approach is to report the key (or a few keys) with the highest IML for the corresponding decrypted string. Essentially, if the threshold in Definition 5 is chosen properly based on the ciphertext length and the key space size, these two approaches are statistically the same. The interested reader is referred to [6] for justification.

3 Description of Hill

Without loss of generality, we assume that for Hill cipher, the plaintext space is the set of all meaningful English strings of length a multiple of an integer d. Each character is naturally interpreted as an element of $\mathbb{Z}_{26}$. To encrypt a plaintext $P = (p_1, p_2, ..., p_{md})$ using a $d \times d$ key matrix K over $\mathbb{Z}_{26}$, it is first divided into m blocks of d characters. Let $P_i = (p_{(i-1)d+1}, p_{(i-1)d+2}, ..., p_{(i-1)d+d})$ denote the i^{th} block of the plaintext for $i = 1, 2, \cdots, m$. The corresponding ciphertext block C_i is then calculated as $C_i = P_i K$ to construct the final ciphertext $C = (C_1, C_2, ..., C_m)$.

KPA on Hill. Hill can be easily broken in the standard KPA model. Suppose that the attacker knows d linearly independent blocks of plaintext, $(P_{i_1}, P_{i_2}, ..., P_{i_d})$, and the corresponding ciphertext blocks, $(C_{i_1}, C_{i_2}, ..., C_{i_d})$. All block pairs (P_{i_j}, C_{i_j}) can be collected from one single plaintext/ciphertext pair or from multiple plaintext/ciphertext pairs. The attacker can then construct the matrices $U = (P_{i_1}^T, P_{i_2}^T, ..., P_{i_d}^T)^T$ and $W = (C_{i_1}^T, C_{i_2}^T, ..., C_{i_d}^T)^T$, and easily calculate the corresponding key matrix as $K = U^{-1}W$. Because of linear independence of the matrix U rows, the invertibility of this matrix is ensured.

COA on Hill. Although Hill is easily broken with KPA, there is no reported attack faster than brute-force attack in COA model, with the assumption that the plaintext is an English text. The existing redundancy of English text can be used to mount COA on Hill, e.g, using a brute-force attack. In this case, we have $|\mathcal{K}| \approx 26^{d^2}$, $|\mathcal{P}| = 26$, and $H \approx 1.0$. Therefore, $R \approx 0.787$ and by Theorem 1, the attack can determine the secret key almost uniquely if the ciphertext is of length at least:

$$n_0 \approx \frac{\log_2 26^{d^2}}{0.787 \log_2 26} \approx 1.27 d^2 \tag{2}$$

In the rest of the paper, we show how our new attacks can be applied on Hill cipher in the COA model much faster than brute-force attack, with slightly increased required ciphertext length.

4 COA on Hill using monograms

In this section, we describe a brute-force attack on Hill using monograms only (instead of the whole language redundancy). We then further improve the attack using the divide-and-conquer technique. However, this will not result in finding the correct secret decryption matrix and, instead, we will find the secret matrix uniquely up to an unknown permutation of its columns. The correct order of the columns can then be determined using bigram frequencies very efficiently which is equivalent to breaking a permutation cipher. First we define the notion of a representative key. Before definition, note that a ciphertext block C is decrypted under a decryption key matrix K^{-1} according to $P = CK^{-1}$. We abuse the notation and similarly denote the decryption of a ciphertext C of an arbitrary length, but a multiple of d, by $P = CK^{-1}$.

Definition 6 (representative key). *For a given ciphertext C, we say that a (candidate) decryption key matrix K^{-1} is a representative key if the decrypted string $P = CK^{-1}$ is monogram-wise meaningful (for a certain threshold).*

4.1 brute-force attack on Hill

In order to characterize the representative keys for the Hill cipher, we need to take into account the following theorem, whose proof is straightforward based on the given definitions so far.

Theorem 2 (permutation rule). *For a given ciphertext C, if a matrix K^{-1} is a representative key, then so is a matrix produced by any arbitrary permutation of the columns of K^{-1}.*

Consequently, for a long enough ciphertext and for a well-chosen threshold, all the $d!$ matrices derived from the correct key matrix are the only representative keys. We conclude that in order to find all the representative keys, one can exhaust all the $O(26^{d^2}/d!)$ matrices — which are equivalent in terms of column permutation — instead of a brute-force attack over all possible matrices. Let us discuss the unicity distance of the attack, i.e., the minimum ciphertext length to determine the $d!$ correct representative keys almost uniquely. We need to plug $|\mathcal{K}| \approx 26^{d^2}/d!$, $|\mathcal{P}| = 26$ and $R \approx 0.1107$ (corresponding to $H \approx 4.1758$) in Theorem 1:

$$\frac{\log_2(26^{d^2}/d!)}{0.1107 \log_2 26} \approx 8.96d^2 - O(\log d)$$

In other words, if the ciphertext length is about the above amount, then the secret matrix is almost uniquely determined up to an unknown permutation over its columns. The bigram frequencies can then be used to distinguish the correct permutation of a correct representative key with computational complexity of $O(d^2)$, ignorable in comparison with the overall exponential complexity of the attack. Therefore, we can ignore the complexity of the permutation attack and consider finding a correct representative key as the end goal. To summarize, the computational complexity of the attack, without bothering to use fast matrix multiplication algorithms, is $O(d^3 26^{d^2}/d!) = O(d^2 26^{d^2}/\log d)$.

4.2 A divide-and-conquer attack on Hill

We describe how a divide-and-conquer technique can be used to transform the brute-force attack into a new attack with high improvement in computational complexity. This attack is based on three key observations:

i) Let K_j^{-1} denote the j^{th} column of a candidate matrix K^{-1}. Given a ciphertext block C_i, the j^{th} element of the decrypted string block $P_i = C_i K^{-1}$, which we denote by $p_i^{(j)}$, can be calculated according to $p_i^{(j)} = C_i K_j^{-1}$.

ii) We expect that the monogram frequencies are still observed if we decimate an English language string (e.g., by keeping the letters which are d positions apart). In other words, for enough number of ciphertext blocks C_i's, and for a column K_j^{-1} of the correct decryption matrix K^{-1}, the sequence of $p_i^{(j)}$'s is monogram-wise meaningful.

iii) We conclude that each column of decryption key matrix can be found separately and independently by trying all $(2^d-1)(13^d-1) \approx 26^d$ possibilities for a column. Since there is no superiority in considering each column, guessing a single column of K^{-1} actually reveals all the correct columns, provided that the ciphertext is long enough.

Using Theorem 1, the enough number of decrypted letters for almost uniquely determining a column, in fact all the d columns, of the decryption matrix can be calculated as:

$$\frac{\log_2 26^d}{0.1107 \log_2 26} \approx 8.96d.$$

Notice that the enough ciphertext length for obtaining the above amount of decrypted letters is equal to

$$n_0^{(26)} \approx 8.96d^2 \ . \tag{3}$$

Compared with the trivial brute-force attack, which takes advantage of the full redundancy of the English plaintext, the required minimum length to mount a successful attack increases by a factor of $n_0^{(26)}/n_0 \approx 7$; see Eq. (2).

Now, let us discuss the complexity of the attack. Our divide-and-conquer attack performs $O(26^d)$ vector by matrix multiplications. Therefore, a naïve implementation of the attack has a computational complexity of $O(d^2 26^d)$. By eliminating repeated calculations, we present an improved attack in Algorithm 1 with complexity of $O(d26^d)$. The idea behind the improved algorithm is as follows. In the naïve implementation, a ciphertext block C_i is multiplied by each guessed column vector x for K_j^{-1} in time $O(d)$. Assume that one has already computed the product $p_i = C_i x'$, where x' is the lexicographically precedent vector of x. The product $C_i x = p_i + d_{i,t}$ can then be computed in $O(1)$, where $d_{i,t}$'s are some precomputed values and $0 \leq t \leq d-1$ is the number of zeros on the top of the column vector x. The precomputation takes time $O(d^2)$ and requires the same amount of memory. Let x' and x be of the following form where $x_t \neq 25$,

$$x'^T = [\underbrace{25, \cdots, 25}_{t}, x_t, x_{t+1}, \cdots, x_d] \ ,$$
$$x^T = [\underbrace{0, \cdots, 0}_{t}, x_t + 1, x_{t+1}, \cdots, x_d] \ .$$

Notice that $x - x' \mod 26$ is a 0-1 vector whose only top $t+1$ elements are 1. Since $d_{i,t} = C_i(x - x')$, the $d_{i,t}$'s can be precomputed as the modulo 26 sum of the first $t+1$ elements of C_i. The reader can easily convinces himself that the **for** loop at the Step 12 in Algorithm 1 updates the IML of the sequence of p_i's accordingly.

Algorithm 1 Divide-and-conquer attack on Hill in $O(d26^d)$

Require: A ciphertext C of length $n = md$
Ensure: A representative decryption matrix K^{-1}
1: Divide C into m blocks $C_1, \cdots, C_m$
2: **for** $t = 0$ **to** $d - 1$ **do**
3: **for** $i = 1$ **to** m **do**
4: Set $d_{i,t}$ to the modulo 26 sum of the first $t+1$ elements of C_i
5: Set K^{-1} to an arbitrary $d \times d$ matrix
6: Set I to an all $-\infty$ vector of length d
 {the j'th element of I corresponds to the IML of the j'th column of K^{-1}}
7: **for** $i = 1$ **to** m **do**
8: Set $p_i = 0$
 {p_i is a decrypted letter of C_i using an all-zero guess for a column of K^{-1}}
9: Set $iml = \text{IML}(p_1, \cdots, p_m) = -\log_2 f_0$
10: **for** all $d \times 1$ vectors x (except the all-zero vector) in lexicographical order **do**
11: Let t denote the maximum number of zeros at the top of x.
12: **for** $i = 1$ **to** m **do**
13: $iml = iml - \frac{1}{m} \log_2 f_{p_i}$
14: $p_i = p_i + d_{i,t} \mod 26$
15: $iml = iml + \frac{1}{m} \log_2 f_{p_i}$
16: **if** x is not all-zero modulo 2 or 13 **then**
17: **if** K^{-1} has a column y whose corresponding IML is smaller than iml **then**
18: Replace the column y in K^{-1} with x
19: Replace the corresponding IML value in I with iml

5 A CRT based divide-and-conquer attack on Hill

Based on the following Observation, the computational complexity of the divide-and-conquer attack can be improved using the Chinese Reminder Theorem (CRT).

Observation 3 ($\mathbb{Z}_{13}$ and $\mathbb{Z}_2$ entropy) *Let X denote a random variable over $\mathbb{Z}_{26}$ with the probability distribution of that of English letters. The $\mathbb{Z}_{13}$ entropy of English monograms is defined to be the entropy of the random variable (X mod 13). Similarly, we define the $\mathbb{Z}_2$ entropy. A simple calculation shows that $\mathbb{Z}_{13}$ and $\mathbb{Z}_2$ entropy of English monograms are 3.4052 and 0.9865, respectively. The corresponding redundancies are then 0.0798 and 0.0135.*

With this observation, the same procedure for divide-and-conquer attack can be done in order to find the columns of the decryption key matrix modulo 2 and 13. To be more precise, for example, in order to find the decryption key matrix columns modulo 13, the ciphertext is first reduced modulo 13. Then all possible $13^d - 1$ values for a column modulo 13 are tried using the brute-force attack. Finally, the best d candidates for the probable columns modulo 13 are identified. The minimum ciphertext length required to almost uniquely find a representative decryption key matrix modulo 13 can be approximated as follows:

$$n_0^{(13)} \approx \frac{\log_2 13^d}{0.0798 \log_2 13} \times d \approx 12.5d^2. \tag{4}$$

A similar procedure can be performed to find a representative decryption key matrix modulo 2, requiring the following amount of ciphertext length:

$$n_0^{(2)} \approx \frac{\log_2 2^d}{0.0135 \log_2 2} \times d \approx 74d^2. \tag{5}$$

Interestingly, to find a representative key modulo 26, the attack can be devised in two different ways using the CRT. Although the computational complexities are the same, the required ciphertext lengths for a successful attack are different.

Combinational attack. Perform the divide-and-conquer attack modulo 13 and 2, and attain a representative decryption key matrix over $\mathbb{Z}_{13}$ and one over $\mathbb{Z}_2$. Here, the enough ciphertext lengths for performing these attacks using Theorem 1 are $12.5d^2$ for $\mathbb{Z}_{13}$ and $74d^2$ for $\mathbb{Z}_2$. Now, combine each column j of the representative key over $\mathbb{Z}_{13}$ with all the d columns of the representative key over $\mathbb{Z}_2$ and extract d vectors for the j'th column over $\mathbb{Z}_{26}$ using the CRT. Then by calculating IML for all of these d possibilities, the vector with largest index is considered as the j'th column of the representative key over $\mathbb{Z}_{26}$. The computational complexity of this method is $O(d13^d + d2^d + d^4) = O(d13^d)$, and the enough ciphertext length is $74d^2$.

Lifting attack. Perform the divide-and-conquer attack over modulo 13 only, and attain a representative key over $\mathbb{Z}_{13}$. Then for each column j of the matrix do the following steps to lift the $\mathbb{Z}_{13}$ representative key into one over $\mathbb{Z}_{26}$:

a) combine all the $2^d - 1$ non-zero vectors over $\mathbb{Z}_2$ with the j'th column and compute the corresponding vectors over $\mathbb{Z}_{26}$ using the CRT.

b) calculate the IML for each one and choose the vector with the largest index as the j'th column of the representative key matrix over $\mathbb{Z}_{26}$.

Now a representative key over $\mathbb{Z}_{26}$ is attained. The computational complexity of this method is $O(d13^d + d^3 2^d) = O(d13^d)$, but the enough ciphertext length is $12.5d^2$, i.e., only a factor of $n_0^{(13)}/n_0 \approx 9.8$ larger than the trivial brute-force attack.

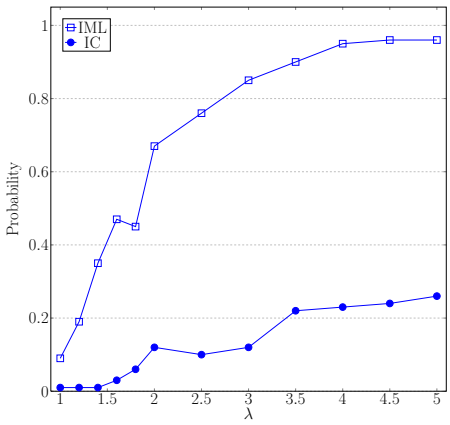
6 Experimental Results

In this section, we study and compare the experimental results with theoretical approaches. We are interested in the success probability of the attacks in terms of the ciphertext length. For this purpose, we produce graphs indicating the success probability of the attacks versus a length coefficient λ, corresponding to an attack that uses a ciphertext of length λn_0 where n_0 (see Eq. (3)–(5)) is the theoretical minimum ciphertext length for mounting a "successful" attack. If our SUWKD (simple uniform wrong key decryption) assumption holds in practice, we expect to have a significant probability of success (say 20–40%) when $\lambda = 1$. Increasing the ciphertext length by a small factor (say $\lambda \approx 4$–5) should then lead to an attack with success probability very close to one. We typically perform our simulation for λ between 1 and 5.

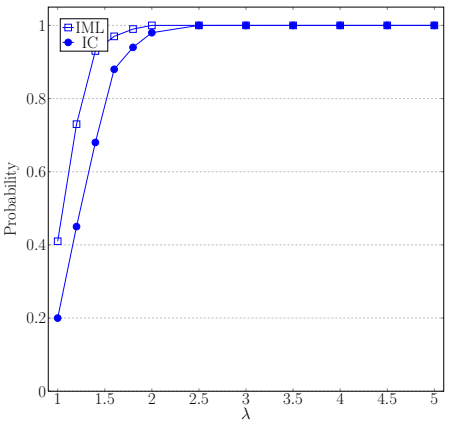
To perform our simulations, we produce plaintexts with statistically independent letters whose distributions correspond to that of English monograms. In other words, we do not model the full redundancy of English language by producing meaningful plaintexts due to not being so easy to handle. As our attacks are only based on monogram distribution, this does not really matter. To verify this statement, we also perform a final simulation using real English text.

6.1 Finding a representative key modulo 26, 13 and 2

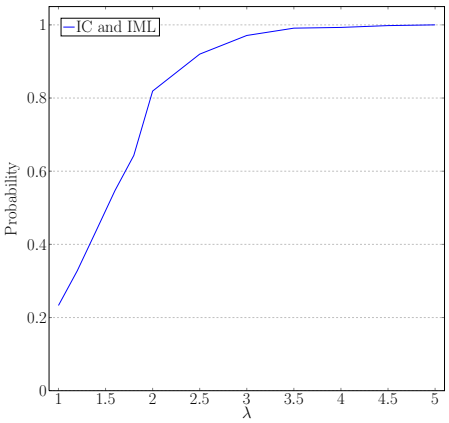
Fig. 1 presents the simulation results for finding a representative key modulo 26, 13 and 2. The modulo 26 attack exactly implements the Algorithm 1. The experiment is performed for N randomly chosen sample ciphertexts of length $5n_0$ where $n_0 \approx 8.96d^2$ (see Eq. (3)). Each sample ciphertext is produced by choosing a random key matrix and a random simulated plaintext. Then, for different values of $\lambda \in \{1, 1.2, 1.4, 1.6, 1.8, 2, 2.5, 3, 3.5, 4, 4.5, 5\}$, the attack is applied to a prefix of length λn_0 (more precisely $d \times \lfloor \lambda n_0/d \rfloor$) on each sample. The attack on the sample is considered successful if Algorithm 1 outputs a decryption key matrix whose columns are the same as those of the original decryption key matrix. We have also implemented our attack using the IC instead of IML.



(a) Modulo 26 results for $d = 5$ with $n_0 \approx 8.96d^2$ (see Eq. 3) using $N = 100$ samples.



(b) Modulo 13 results for $d = 6$ with $n_0 \approx 12.5d^2$ (see Eq. 4) using $N = 100$ samples.



(c) Modulo 2 results for $d = 6$ with $n_0 \approx 74d^2$ (see Eq. 5) using $N = 1000$ samples.

Fig. 1. Simulation results for the success probability of finding a representative key modulo 26, 13 and 2 in terms of λ using ciphertexts of length λn_0.

This can be done by modifying Algorithm 1 accordingly. Fig. 1(a) presents the simulation results for $d = 5$ using $N = 100$ samples for both IML and IC. The modulo 13 and 2 attacks have been simulated similarly. Recall that the minimum ciphertext length for these two cases are $n_0 \approx 12.5d^2$ (see Eq. (4)) and $n_0 \approx 74d^2$ (see Eq. (5)), respectively. Figures 1(b) and 1(c) present the simulation results for $d = 6$ using $N = 100$ and $N = 1000$ samples, respectively, for these two moduli.

Here, the following results can be extracted from the graphs:

1. Our simulation results verify the superiority of using IML over IC. As we can see, IML works much better than IC to find a correct representative key over $\mathbb{Z}_{26}$ and $\mathbb{Z}_{13}$. However, it does not matter to use which one over $\mathbb{Z}_2$. Using some simple calculations one can justify this by showing that IML of a string over $\mathbb{Z}_2$ can be well approximated as an affine function of its IC.
2. The SUWKD assumption holds quite well for the attack on finding a correct representative key over $\mathbb{Z}_{13}$ and $\mathbb{Z}_2$ as for $\lambda \approx 4.5$ we already reach a success probability of almost one.
3. The SUWKD assumption, however, does not hold quite well for the attack on finding a correct representative key over $\mathbb{Z}_{26}$ as for $\lambda \approx 5$ we reach a success probability of only about 96%. We will explore this issue and study the reason of this phenomena in next subsection. In any case, although interesting, this does not really matter since the success probability approaches one by using a slightly larger value for λ.

6.2 Lifting a representative key over $\mathbb{Z}_{13}$ into $\mathbb{Z}_{26}$

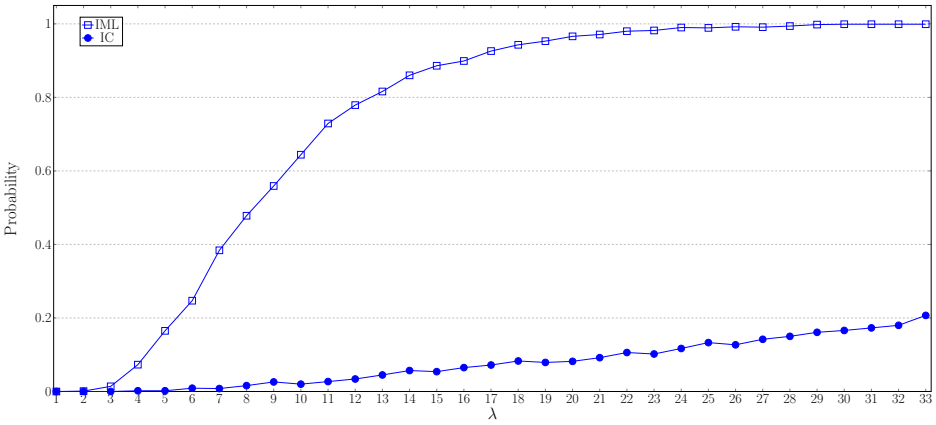
Fig. 2 presents the simulation results for finding a representative key modulo 26 assuming that we already know a representative key modulo 13, using the approach that was used in the "lifting attack" in Section 5.

Recall that each column of the representative key over $\mathbb{Z}_{13}$ is combined with all $2^d - 1$ non-zero vectors over $\mathbb{Z}_2$ to compute a vector over $\mathbb{Z}_{26}$ using the CRT. According to Theorem 1, the minimum ciphertext length to be able to find the representative key modulo 26 almost uniquely should be

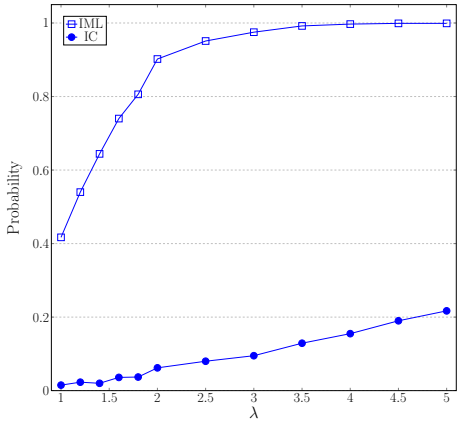
$$n_0 \approx \frac{\log_2(2^d)}{0.1107 \log_2 26} \times d \approx 1.9d^2 \ . \tag{6}$$

Fig. 2(a) depicts the simulation results for $d = 6$ using $N = 1000$ samples. As it can be seen, the SUWKD assumption does not hold quite well here since for finding a correct representative key we need λ to be greater than 30. Stated differently, if a wrong key is the same as the original key modulo 13, then the decrypted ciphertext using the wrong key can not be treated as a random string due to its correlation with the original plaintext. This justifies our third point in the previous subsection since $30 \times 1.9d^2 > 5 \times 8.96d^2$.

However, luckily our attack for finding a representative key modulo 13 already requires a ciphertext of length $n_0 \approx 12.5d^2$. Fig. 2(b) depicts the simulation results for $d = 6$ using $N = 1000$ samples in this case. In fact Fig. 2(b) corresponds to part of Fig. 2(a) which lies in the interval $6.6 < \lambda < 32.9$. Small differences in the figures are due to length rounding error.



(a) Results for $d = 6$ with $n_0 \approx 1.9d^2$ (see Eq. 6) using 1000 samples for $\lambda \leq 33$.



(b) Results for $d = 6$ with $n_0 \approx 12.5d^2$ using 1000 samples for $\lambda \leq 5$.

Fig. 2. Success probability of finding a representative key modulo 26 in terms of λ assuming that we already know a representative key modulo 13 using ciphertexts of length λn_0.

6.3 Success probability of our attack

The success probability of our attack (lifting attack) can be estimated as the product of Fig. 1(b) and Fig. 2(b), depicted as the solid-lines in Fig. 3 for $d = 6$. Direct simulation of the overall success probability of the attack also results in the same chart. So far we have performed our simulations using simulated plaintexts by generating independent letters with probability distribution of that of English monograms. To verify this approach, we have mounted our attack on 45 plaintexts of length $12.5 \times 5 \times 6^2 = 2250$, driven from the well-known book "Alice in the Wonderland". The results are shown in dashed-lines in Fig. 3.

Although our attack using IML on the real-text is highly consistent with the attack on simulated text, there is a slight difference for the attack that uses IC, the reason of which is unclear to us.

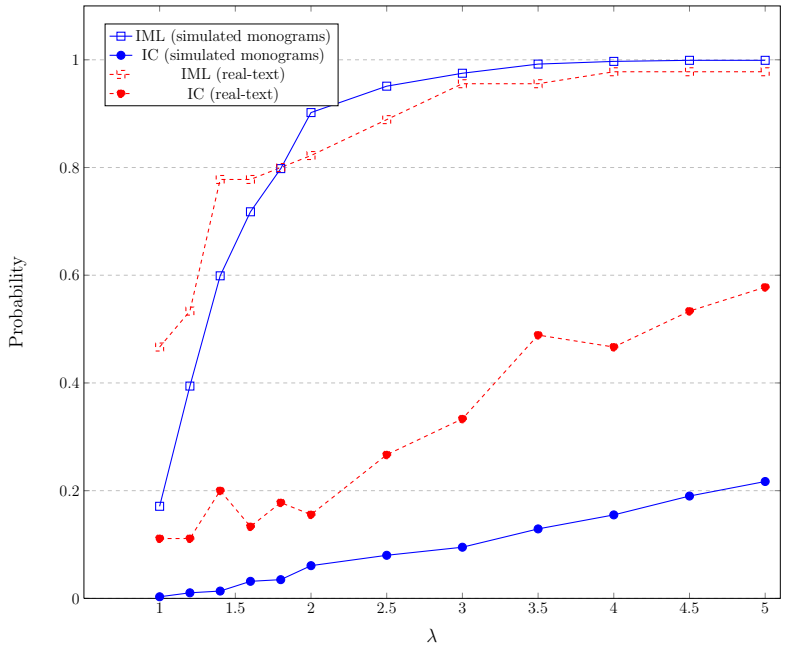


Fig. 3. Success probability of our attack (lifting attack) in terms of λ using ciphertexts of length λn_0 with $n_0 \approx 12.5d^2$ for $d = 6$: the blue solid-line is for the simulated plaintext computed as the product of Fig. 1(b) and Fig. 2(b) whereas the red dashed-line is for real text from "Alice in the Wonderland" using 45 samples.

7 Conclusion

Hill is a classical encryption system which is generally conceived as a ciphertext-only attack resistant block cipher. In this paper we presented a new COA on Hill using divide and conquer technique and the redundancy of the monograms only. Our attack then turns the ciphertext into the output of a permutation cipher, which can be easily broken using the redundancy of bigrams. The computational complexity of the attack is $O(d26^d)$ which is dramatically lower than the computational complexity of brute-force attack on Hill. Also, by using Chinese Remainder Theorem, we improved the divide-and-conquer COA on Hill to the computational complexity of $O(d13^d)$ with the cost of a slightly more data complexity. Comprehensive simulations results verifies our theoretical re-

sults. Beating our results or proving its optimality based on some reasonable computational complexity assumptions remains an open problem.

Acknowledgment. This work has been supported by Iranian National Science Foundation (INSF) under contract no. 92027548 and Sharif Industrial Relation Office (SIRO) under grant no. G931223. The initial idea of the basic divide-and-conquer attack was proposed by the following student in an introductory course in cryptography at the Department of Mathematical Sciences at Sharif University of Technology offered by the first author: Navid Alamati, Mohammad-Esmaeil Hasani, and Morteza Hasanvand. We would like to specially thank Siavash Riahi for his extensive simulation result of Section 6.

References

1. C. Christensen. Polygraphic Substitution Ciphers: The Hill Cipher, II. *http://www.nku.edu/~christensen/1402%20Hill%20cipher%20part%20II.pdf*, Accessed Summer 2015.
2. L. S. Hill. Cryptography in an algebraic alphabet. In *American Mathematical Monthly*, pages 306–312. 1929.
3. B. Hu. Introduction to Cryptology: Hill Cipher Remarks. *http://www.cs.rochester.edu/~bh/csc290/hill.html*, Accessed Summer 2015.
4. J. Overbey, W. Traves, and J. Wojdylo. On the keyspace of the hill cipher. *Cryptologia*, 29(1):59–72, 2005.
5. D. E. Robling Denning. *Cryptography and data security*. Addison-Wesley Longman Publishing Co., Inc., 1982.
6. T. Siegenthaler. Correlation-immunity of nonlinear combining functions for cryptographic applications. *IEEE Transactions on Information Theory*, 30(5):776–780, 1984.
7. A. Sinkov. *Elementary cryptanalysis: a mathematical approach*, volume 22 of *New mathematical library*. 1966. With a supplement by Paul L. Irwin. Reissued in 1975 and 1980.
8. D. R. Stinson. *Cryptography: theory and practice*. CRC press, 2005.
9. V. Strassen. Gaussian elimination is not optimal. *Numerische Mathematik*, 13(4):354–356, 1969.
10. S. S. Wagstaff Jr. *Cryptanalysis of number theoretic ciphers*. CRC Press, 2002.
11. E. Wikipedia. Hill Cipher. *https://en.wikipedia.org/wiki/Hill_cipher*, Accessed Summer 2015.