Cryptanalysis of the Toorani-Falahati Hill Ciphers [*]

Liam Keliher [†] Anthony Z. Delaney [†]

Abstract

In 2009 and 2011, Toorani and Falahati introduced two variants of the classical Hill Cipher, together with protocols for the exchange of encrypted messages. The designers claim that the new systems overcome the weaknesses of the original Hill Cipher, and are resistant to any ciphertext-only, known-plaintext, chosen-plaintext, or chosen-ciphertext attack. However, we describe a chosen-plaintext attack that easily breaks both Toorani-Falahati Hill Ciphers, and we present computational results that confirm the effectiveness of our attack.

1 Introduction

The Hill Cipher is a classical symmetric-key cipher that was published by Lester Hill in 1929 [1]. The Hill Cipher represents each plaintext as a vector of integer values, and encrypts this vector using a single multiplication by a square key matrix. This has the advantage of simplicity, but renders the cipher vulnerable to a straightforward known-plaintext attack based on linear algebra [2]. Despite this weakness (or perhaps because of it), the Hill Cipher is often described in cryptography textbooks, where it serves to introduce students to a number of important concepts, including modular arithmetic, linear algebra, and basic cryptanalysis.

Several researchers have presented variants of the Hill Cipher that attempt to correct its security flaws; these are based on a variety of techniques. One approach is to add features borrowed from modern symmetric-key cryptography, such as iteration of a simpler encryption step (or "round") that involves matrix multiplication internally. An example is a modified Hill Cipher due to Sastry et al. [3], which was broken by Keliher [4] using a known/chosen-plaintext attack similar to the linear algebraic attack on the original Hill Cipher. Another approach is to modify the Hill Cipher so that the encryption process changes for each plaintext, since the standard known-plaintext attack exploits the fact that each plaintext is encrypted in an identical fashion. Saeednia [5] derives a new key matrix for each plaintext by permuting the rows and columns of the key matrix used for the previous plaintext, but Lin et al. [6] demonstrate weaknesses in this approach. Ismail et al. [7] employ vector-matrix multiplication to derive a new key matrix for each plaintext, however Li et al. [8] showed that this cipher can also be broken using a known-plaintext attack.

In 2009 and 2011, Toorani and Falahati introduced two variants of the Hill Cipher, together with protocols for the exchange of encrypted messages [9, 10]. We refer to these ciphers as Toorani-Falahati Hill Cipher #1 (2009) and #2 (2011), and we shorten the names to TFHC1 and TFHC2. Both ciphers use a one-way hash function to modify the encryption process for each plaintext,

[*]Copyright IEEE. Presented at IEEE Symposium on Computers and Communications (ISCC 2013).

[†]AceCrypt Research Group, Department of Mathematics and Computer Science, Mount Allison University, Sackville, New Brunswick, Canada.

and Toorani and Falahati claim that both are secure against ciphertext-only, known-plaintext, chosen-plaintext, and chosen-ciphertext attacks. However, we describe a chosen-plaintext attack that easily breaks TFHC1 and TFHC2 by bypassing their main security features, and we present computational results that confirm the effectiveness of our attack.

Although TFHC1 and TFHC2 have significant differences, our attack works identically for both, so we focus primarily on TFHC1. The remainder of this paper is organized as follows. In Section 2 we introduce some basic concepts from cryptanalysis. In Section 3 we discuss the classical Hill Cipher and the Affine Hill Cipher. In Section 4 we give the details of TFHC1. In Section 5 we describe our attack on TFHC1, and in Section 6 we present computational results from the implementation of our attack. In Section 7 we briefly explain the differences between TFHC1 and TFHC2, and in Section 8 we conclude the paper.

2 Basic Cryptanalysis Concepts

The primary goal of cryptanalysis applied to a symmetric-key cipher is to gain the ability to decrypt any ciphertext. Typically this involves learning the secret key, although on occasion an attacker is able to construct an algorithm that decrypts any ciphertext without determining the key (see [4], for example).

There are four standard categories of attacks on symmetric-key ciphers, based on the type of data the attacker is able to obtain/capture:

1. **ciphertext-only:** attacker can obtain one or more ciphertexts
2. **known-plaintext:** attacker can obtain one or more plaintexts and the corresponding ciphertexts
3. **chosen-plaintext:** attacker can choose one or more plaintexts and obtain the corresponding ciphertexts
4. **chosen-ciphertext:** attacker can choose one or more ciphertexts and obtain the corresponding plaintexts

The *data complexity* of an attack is the number of plaintexts/ciphertexts that must be obtained in order for the attack to succeed, and the *time complexity* is the number of operations that must be performed. (There is no fixed definition of "operation," but often a natural unit of computational work, such as a single encryption, is used.)

The simplest attack is *exhaustive search of the key space*, also known as *brute force*. This is a known-plaintext attack in which the attacker first obtains a small number of plaintext-ciphertext pairs (often 2 or 3 are sufficient), and then systematically tries all possible cipher keys until one is found that encrypts each plaintext to the corresponding ciphertext. With high probability, a candidate key that passes this test is the true key. The data complexity of exhaustive search is small, but the time complexity is approximately equal to the size of the *keyspace* (the set of all keys), which is prohibitively large for most modern ciphers.

Brute force is viewed as a "baseline"—that is, an attack is considered to be significant if it has lower time complexity than brute force (but possibly higher data complexity).

3 The Classical Hill Cipher

For the classical Hill Cipher, each plaintext is a row vector $\mathbf{X} = (x_1, x_2, \ldots, x_n)$ of length $n \geq 2$, where each x_i is an integer (mod m), with $m \geq 2$ (traditionally $m = 26$, since there are 26 letters in the English alphabet). A plaintext $\mathbf{X}$ is encrypted to the corresponding ciphertext $\mathbf{Y}$ through multiplication by an $n \times n$ invertible key matrix $\mathbf{K}$, that is, $\mathbf{Y} = \mathbf{XK}$ (mod m). The ciphertext is decrypted by computing $\mathbf{X} = \mathbf{YK}^{-1}$ (mod m). In most situations, the key matrix $\mathbf{K}$ is known only to the sender (Alice) and the receiver (Bob).

Here is the standard known-plaintext attack against the Hill Cipher: If an attacker is able to learn n or more plaintext-ciphertext pairs such that n of the the plaintexts are linearly independent, the attacker places the linearly independent plaintexts in the rows of an $n \times n$ matrix $\mathbf{R}$, and places the corresponding ciphertexts in the rows of an $n \times n$ matrix $\mathbf{S}$. It follows that $\mathbf{RK} = \mathbf{S}$, and therefore $\mathbf{K} = \mathbf{R}^{-1}\mathbf{S}$. Since the attacker knows $\mathbf{R}$ and $\mathbf{S}$, and can compute $\mathbf{R}^{-1}$ ($\mathbf{R}$ is guaranteed to be invertible because its rows are linearly independent), the attacker can determine the secret key matrix $\mathbf{K}$, and therefore the cipher is broken.

In the *Affine Hill Cipher*, Alice and Bob share both a secret matrix $\mathbf{K}$ and a secret vector $\mathbf{V}$, so technically the key is the pair $(\mathbf{K}, \mathbf{V})$. A plaintext $\mathbf{X}$ is encrypted using $\mathbf{Y} = \mathbf{XK} + \mathbf{V}$. The Affine Hill Cipher is also easily broken using a known-plaintext attack [11].

4 Toorani-Falahati Hill Cipher #1 (TFHC1)

One reason the classical Hill Cipher and the Affine Hill Cipher are easily attacked is that exactly the same encryption process is applied to each plaintext. TFHC1 uses pseudo-random numbers produced by a one-way hash function, $H(\,)$, to modify the encryption step for each plaintext. $H(\,)$ is assumed to be a strong standard hash function such as SHA-1 [12]. In addition, operations in TFHC1 are performed (mod p), where p is a prime number. Note that several of the ideas used in TFHC1 are based on the cipher introduced by Lin et al. [6], but Lin et al. make more extensive use of $H(\,)$ by applying it multiple times in the encryption of each plaintext; this potentially increases security, but significantly reduces efficiency.

The key for TFHC1 is an invertible $n \times n$ matrix $\mathbf{K}$ with integer values in the interval $[0, p-1]$, i.e., integer values reduced (mod p). We use k_{ij} to denote the entry in row i and column j of $\mathbf{K}$, for $1 \leq i, j \leq n$. In keeping with good practice in cryptography, we adopt *Kerkhoffs' Principle* [12], which states that the security of a cipher should rest entirely in the key, with all other details assumed to be known. In particular, for TFHC1 we assume the attacker knows n and p.

Suppose Alice has a message $\mathbf{M}$ consisting of T plaintexts $\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_T$ that she wants to send to Bob. In essence, TFHC1 is an Affine Hill Cipher for which Alice derives a new secret pair $(\mathbf{K}_t, \mathbf{V}_t)$ for each plaintext $\mathbf{X}_t$ ($1 \leq t \leq T$). (In this sense, TFHC1 is behaving like a *stream cipher* [12].) Alice computes the corresponding ciphertext $\mathbf{Y}_t$ using $\mathbf{Y}_t = \mathbf{X}_t\mathbf{K}_t + \mathbf{V}_t$, and sends $\mathbf{Y}_1, \mathbf{Y}_2, \ldots, \mathbf{Y}_T$ to Bob, along with a small amount of information that allows Bob to derive the identical pairs $(\mathbf{K}_t, \mathbf{V}_t)$, and thereby recover each plaintext $\mathbf{X}_t$ using $\mathbf{X}_t = (\mathbf{Y}_t - \mathbf{V}_t)\mathbf{K}_t^{-1}$. We now give the exact details.

Assume that prior to communicating, Alice and Bob have established the shared key $\mathbf{K}$. Here are the steps for TFHC1:[1]

[1]We break the process into a larger number of steps than in [9].

1. Alice chooses a random integer $a_0 \in [1, p-1]$, and computes $a_1, a_2, \ldots, a_T$, where $a_t = H(a_{t-1})$ for $t \geq 1$ (recall that $H(\,)$ is the one-way hash function).

2. For $1 \leq t \leq T$, Alice assigns $v_0 = a_t \pmod p$, unless the resulting value of v_0 is 0, in which case $v_0 = 1$. She then sets $\mathbf{K}_t = v_0\mathbf{K}$.

3. Alice constructs $\mathbf{V}_t = (v_1, v_2, \ldots, v_n)$ as follows: For $1 \leq i \leq n$, she sets $j_i = (v_{i-1} \mod n)+1$ and $v_i = (k_{ij_i} + \tilde{v}_{i-1}\, v_0) \pmod p$, where $\tilde{v}_{i-1} = 2^{\lceil \gamma/2 \rceil} + (v_{i-1} \mod 2^{\lceil \gamma/2 \rceil})$ and $\gamma = \lfloor \log_2 v_{i-1} \rfloor + 1$ ($\lfloor \cdot \rfloor$ is the floor function).[2]

4. Alice encrypts each $\mathbf{X}_t$ using $\mathbf{Y}_t = \mathbf{X}_t\mathbf{K}_t + \mathbf{V}_t$. Let $E(\mathbf{M})$ denote the entire encrypted message, i.e., $E(\mathbf{M}) = (\mathbf{Y}_1, \mathbf{Y}_2, \ldots, \mathbf{Y}_T)$.

5. Alice chooses a random integer $b \in [1, n^2]$, and computes $x = \lceil b/n \rceil$ ($\lceil \cdot \rceil$ is the ceiling function) and $y = b - n(x-1)$. She then computes $r = a_0\, k_{xy} \pmod p$.[3] (Note that if the elements in $\mathbf{K}$ are enumerated in row-major order from 1 to n^2, then b is the index of k_{xy}.)

6. Alice sends $(E(\mathbf{M}), b, r)$ to Bob. *This is the one-pass protocol in* [9].

7. Bob derives x and y from b as in Step 5, and obtains k_{xy} from $\mathbf{K}$. He recovers a_0 by computing $r\, k_{xy}^{-1} \pmod p = a_0\, k_{xy}\, k_{xy}^{-1} \pmod p = a_0$.[4] Bob can then compute each $\mathbf{K}_t$ and $\mathbf{V}_t$ in the same manner as Alice, and recover each plaintext of the original message $\mathbf{M}$ using using $\mathbf{X}_t = (\mathbf{Y}_t - \mathbf{V}_t)\mathbf{K}_t^{-1}$.

Our description of TFHC1 differs slightly from that in [9]. In particular, Toorani and Falahati do not "bundle" together v_0 and $\mathbf{K}$ to form $\mathbf{K}_t$, but instead define encryption of plaintext $\mathbf{X}_t$ as $\mathbf{Y}_t = v_0\mathbf{X}_t\mathbf{K} + \mathbf{V}_t$. However, this is equivalent to our approach, since $v_0\mathbf{X}_t\mathbf{K} = \mathbf{X}_t(v_0\mathbf{K}) = \mathbf{X}_t\mathbf{K}_t$.

Note that a_0 is critical to both Alice and Bob, since it is used to derive all the a_t values, which in turn are used to compute each $\mathbf{K}_t$ and $\mathbf{V}_t$. When Alice multiplies a_0 by k_{xy} to obtain r, she is "masking" the value of a_0 before sending it to Bob in order to hide it from an attacker. Clearly it is important that $a_0 \neq 0$ (as specified in Step 1 above), since if $a_0 = 0$, then the value r sent from Alice to Bob is also equal to 0, and the attacker immediately knows that $a_0 = 0$.

Toorani and Falahati claim that the design of TFHC1 makes it resistant to attack. However, despite the improvements in TFHC1 over previous attempts to strengthen the Hill Cipher, we are able to break TFHC1 using a chosen-plaintext attack. We present our cryptanalysis in the next section.

[2]In [9], v_i is defined by $v_i = (k_{ij_i} + \tilde{v}_{i-1}\, a_t) \pmod p$, which is equivalent to the expression used in this paper in most cases, since in general $v_0 \equiv a_t \pmod p$. The exception is when $a_t \equiv 0 \pmod p$, and thus $v_0 = 1$, in which case it is safer to use our expression, because otherwise $v_i = k_{ij_i} + 0 = k_{ij_i}$, so if the attacker can learn v_i, then the attacker immediately knows one of the entries in the key matrix $\mathbf{K}$, a situation that should be avoided.
Also note that if $v_{i-1} = 0$, then $\log_2 v_{i-1}$ is undefined. A potential fix is to assign $\gamma = 0$ in this situation, resulting in $\tilde{v}_{i-1} = 1$.

[3]We use variables x and y in Step 5 and Step 7, whereas [9] uses i and j. We made the change in order to avoid confusion with the variables in Step 3.

[4]It is important that $k_{xy} \neq 0$, since otherwise Bob cannot compute its inverse in Step 7. If Alice's choice of b in Step 5 yields $k_{xy} = 0$, she should choose another b for which $k_{xy} \neq 0$ before proceeding.

5 Cryptanalysis of TFHC1

The goal of our attack on TFHC1 is to determine all n^2 entries in $\mathbf{K}$. Once $\mathbf{K}$ is known, we can eavesdrop on any instance of the one-pass protocol to capture $(E(\mathbf{M}), b, r)$, and then recover the original message $\mathbf{M}$ using Bob's calculations in Step 7.

If reasonable values of p and n are chosen for TFHC1, the cipher is resistant to a brute force attack, since the keyspace is extremely large. The size of the keyspace is given in [13]:

$$\prod_{i=0}^{n-1} (p^n - p^i)$$

For $p = 257$ and $n = 4$ (small example values used in [9]), the number of keys is $3.6 \times 10^{38} \approx 2^{128}$, which corresponds to a 128-bit key, a standard size for modern block ciphers such as the Advanced Encryption Standard (AES) [11]. For $p = 1723$ and $n = 10$ (larger values we tested in the implementation of our attack), the number of keys is $4.25 \times 10^{323} \approx 2^{1075}$.

5.1 Attack Fundamentals

We break TFHC1 using a chosen-plaintext attack in which we repeatedly obtain the encryption of the all-zero plaintext $\mathbf{0}$. Note that the different all-zero plaintexts do not need to be consecutive within a message, nor do they even need to be part of the same message (i.e., involved in the same instance of the one-pass protocol).

The critical point is that if a plaintext $\mathbf{X}_t = \mathbf{0}$, then the corresponding ciphertext is $\mathbf{Y}_t = \mathbf{0}\mathbf{K}_t + \mathbf{V_t} = \mathbf{V}_t$, that is, the ciphertext, which are able to obtain in a chosen-ciphertext attack, is equal to $\mathbf{V}_t = (v_1, v_2, \ldots, v_n)$. Using the v_i and the information in Step 3 above, we can compute $\tilde{v}_1, \tilde{v}_2, \ldots, \tilde{v}_{n-1}$. We now know many of the values in the following system of n linear equations:[5]

$$\begin{aligned} v_1 &= k_{1j_1} + \tilde{v}_0\, v_0 \\ v_2 &= k_{2j_2} + \tilde{v}_1\, v_0 \\ v_3 &= k_{3j_3} + \tilde{v}_2\, v_0 \\ &\cdots \\ v_n &= k_{nj_n} + \tilde{v}_{n-1}\, v_0 \end{aligned}$$

For $2 \leq i \leq n$, multiply the i^{th} equation by $\tilde{v}_{i-1}^{-1}$ to give

$$(\tilde{v}_{i-1}^{-1})v_i = (\tilde{v}_{i-1}^{-1})k_{ij_i} + (\tilde{v}_{i-1}^{-1})\tilde{v}_{i-1}\, v_0$$

(A careful examination of the formula for $\tilde{v}_{i-1}$ in Step 3 reveals that $0 < \tilde{v}_{i-1} < p$, so $\tilde{v}_{i-1}^{-1}$ always exists.) If we define $c_i = \tilde{v}_{i-1}^{-1}$ and $d_i = (\tilde{v}_{i-1}^{-1})v_i$, and apply the identity $(\tilde{v}_{i-1}^{-1})\tilde{v}_{i-1} = 1$, the set of equations above simplifies to

$$\begin{aligned} [\ \ v_1 &= k_{1j_1} + \tilde{v}_0\, v_0\ \ ] \\ d_2 &= c_2 k_{2j_2} + v_0 \\ d_3 &= c_3 k_{3j_3} + v_0 \\ &\cdots \\ d_n &= c_n k_{nj_n} + v_0 \end{aligned} \tag{1}$$

[5]For simplicity, we will no longer write (mod p) to accompany an equation; this will be implied.

We emphasize here that we know the values of c_i and d_i for $2 \le i \le n$. We enclose the first equation in brackets to highlight the fact that its structure is different from the others; this is because we do not (yet) know the value of $\tilde{v}_0$, so we cannot isolate v_0.

From now on we use $\mathbf{C} = (v_1, v_2, \ldots, v_n)$ to denote *a ciphertext obtained from the encryption of an all-zero plaintext.* For any such ciphertext $\mathbf{C}$, let $W(\mathbf{C})$ be the set of n equations in (1), and let $W^*(\mathbf{C})$ denote $W(\mathbf{C})$ with the first equation removed. In addition, define $U(\mathbf{C}) = \{k_{1j_1}, k_{2j_2}, \ldots, k_{nj_n}\}$ and $U^*(\mathbf{C}) = \{k_{2j_2}, \ldots, k_{nj_n}\}$ (which is $U(\mathbf{C})$ with k_{1j_1} removed). Clearly $U(\mathbf{C})$ and $U^*(\mathbf{C})$ are sets of entries in $\mathbf{K}$.

If we take any two equations in $W^*(\mathbf{C})$, for example

$$d_2 = c_2 k_{2j_2} + v_0$$
$$d_3 = c_3 k_{3j_3} + v_0$$

and subtract one from the other, the result is that v_0 cancels out, leaving a single linear equation involving two of the entries in $\mathbf{K}$, with all other values known:

$$d_2 - d_3 = c_2 k_{2j_2} - c_3 k_{3j_3} \tag{2}$$

(c_2, c_3, d_2, and d_3 are known, and k_{2j_2} and k_{3j_3} are possibly unknown). We can repeat this process for all pairs of equations in $W^*(\mathbf{C})$. This yields $\binom{n-1}{2} = \frac{(n-1)(n-2)}{2}$ linear equations such as (2), each involving two entries in $\mathbf{K}$.

5.2 Important Observations

We summarize preliminary information about our attack in the following list of observations:

1. By subtracting equations in $W^*(\mathbf{C})$ from each other, we obtain linear equations that involve only known values and entries in $\mathbf{K}$, but no longer involve the value v_0 produced by the hash function $H(\,)$. *This means we have bypassed the security that Toorani and Falahati intended for $H(\,)$ to provide.*

2. Linear equations are the easiest equations to solve. Given a set of unknowns (in this case, the n^2 entries in $\mathbf{K}$), it is possible to solve for them by obtaining a large enough set of linear equations involving only those unknowns.

3. For any $W(\mathbf{C})$, initially we do not know v_0, but if we *do know* the value of any one of the k_{ij_i}, for $2 \le i \le n$ (i.e., the value of any entry in $U^*(\mathbf{C})$), we can solve for v_0 using the i^{th} equation,[6] and then substitute v_0 into each of the remaining equations in $W^*(\mathbf{C})$ to determine all of $U^*(\mathbf{C}) = \{k_{2j_2}, \ldots, k_{nj_n}\}$.

4. In addition, once we know v_0, we can calculate $\tilde{v}_0$, and then substitute these two values into the first equation in (1) to determine k_{1j_1}.

5. For any $W(\mathbf{C})$, if we do not know the value of *any* of the k_{ij_i}, we cannot yet apply Observations 3 and 4. Instead, we can form $\binom{n-1}{2}$ linear equations such as (2), as previously described, and store these for future use.

[6]Note that in general v_0 is not the same as a_t, since $v_0 = a_t \pmod{p}$, so learning a v_0 value does not necessarily allow us to apply $H(\,)$ to learn later a_t (and v_0) values.

6. If we repeatedly process ciphertexts, Observation 5 may lead to a situation in which we have stored two different equations involving the *same* two (unknown) entries in $\mathbf{K}$. If these equations are consistent and independent [14], we can solve for the two unknowns.

7. For any linear equation such as (2) involving two entries in $\mathbf{K}$, as soon as we know the value of one of the entries, we can immediately solve for the other, since the other entry in $\mathbf{K}$ is the only remaining unknown in the equation.

5.3 Graph Theoretic Framework

We use a graph theoretic framework to organize our attack against TFHC1 by viewing each of the n^2 unknown values in $\mathbf{K}$ as a node in a graph data structure [15], which we denote G. Two nodes are joined by an edge if and only if we have derived a linear equation involving the two corresponding unknowns. We say that a node is *solved* when we have learned the value of (solved for) the corresponding entry in $\mathbf{K}$. We incorporate the following two graph operations in our attack:
Propagate Solutions: It follows from Observation 7 that if z is a node in G, and z is solved, then we can solve every node z' connected to z. Repeating this process, we can solve every node z'' connected to z', and so on. We continue until no further nodes can be solved. The overall effect is that if it is possible to reach any unsolved node z^* by starting at a solved node z and travelling along the edges of G, then z^* can be solved. (In graph terminology, we say that if z is solved, then every node belonging to the same *connected component* as z can be solved [15].)
Transitive Closure: In this operation, we add new "implied" edges to G based on existing edges. For each triple of nodes (z_1, z_2, z_3), if there is an edge between z_1 and z_2, and also an edge between z_2 and z_3, but no edge between z_1 and z_3, we can add an edge between z_1 and z_3. The reason is that the edge between z_1 and z_2 is based on a linear equation involving the two corresponding unknowns, as is the edge between z_2 and z_3. By solving the first equation for the unknown corresponding to z_2, and then substituting the result into the second equation, we obtain an equation involving the unknowns corresponding to z_1 and z_3, which means we now have an edge between z_1 and z_3.

5.4 Complete Attack Algorithm

Here is the pseudocode for the complete attack:

```
initialize $G$ to be a graph with $n^2$ nodes and no edges
while not all entries in $K$ are known
    • get the next ciphertext $\mathbf{C}$ and form $W(\mathbf{C})$
    • Case I: at least one entry in $U^*(\mathbf{C})$ is known
        – solve for every entry in $U(\mathbf{C})$ (Observations 3 and 4)
        – call Propagate Solutions
        – for each previous ciphertext $\mathbf{C}'$, if every entry in $U^*(\mathbf{C}')$ is now known, but $k_{1j_1}$ is not
          known, solve for $k_{1j_1}$ (Observation 4)
```

- **Case II: none of the entries in $U^*(\mathbf{C})$ are known**
  - form $\binom{n-1}{2}$ linear equations as in Observation 5, and add the corresponding edges to G. If at any time in this process two nodes in G are joined by two edges, attempt to solve the two nodes (Observation 6)
  - call ***Transitive Closure***. As in the previous step, whenever this results in two nodes in G that are joined by two edges, attempt to solve the nodes
  - call ***Propagate Solutions***
  - for each previous ciphertext $\mathbf{C}'$, if every entry in $U^*(\mathbf{C}')$ is known, but k_{1j_1} is not known, solve for k_{1j_1} (Observations 3 and 4)

5.5 Correctness of Attack

Case I: When *at least one* of the values in $U^*(\mathbf{C}) = \{k_{2j_2}, \ldots, k_{nj_n}\}$ is already known, we use Observations 3 and 4 to determine the remaining values. Typically this means that we learn at least one new entry in $\mathbf{K}$ (unless *all* the entries in $U(\mathbf{C})$ were previously known), i.e., there are new solved nodes in the graph G, so we call ***Propagate Solutions*** to "spread" solutions as far as possible in G. At this point, because of the addition of new edges in earlier iterations of Case II, together with the call to ***Propagate Solutions*** in Case I, it is possible that there are *previous* ciphertexts $\mathbf{C}'$ for which every value in $U^*(\mathbf{C}')$ is now known, but the k_{1j_1} value is not known. This is the reason for the last step in Case I.
Case II: In this case, none of the entries in $U^*(\mathbf{C})$ are initially known, so we cannot immediately solve for any k_{ij_i} value. Instead, we add as many edges as possible to G, first by inserting the $\binom{n-1}{2}$ edges justified by Observation 5, and then by calling ***Transitive Closure***. These steps may result in new solved nodes, so we spread solutions by calling ***Propagate Solutions***. Finally, we solve for the "lone wolf" k_{1j_1} values by repeating the last step in Case I.

6 Computational Results

We implemented TFHC1 (with SHA-1 for $H(\,)$), and then ran our attack to verify its effectiveness. The computer we used was a Dell Precision M4500 laptop running Windows 7, and our implementation was in Java. The attack worked as theorized, with negligible time and space requirements, even for large keys.

An interesting statistic is the number of ciphertexts (corresponding to the all-zero plaintext) required to determine every entry in the key $\mathbf{K}$. For prime modulus $p = 257$ and $n = 4$, we ran our attack 1000 times, each time with a new value of a_0 and a new randomly generated key. Over these 1000 trials, the average number of ciphertexts required to find $\mathbf{K}$ was 12.52, and the average computation time per trial was 0.00075 seconds.

For prime modulus $p = 1723$ and $n = 10$, we also ran our attack 1000 times. The average number of ciphertexts required was 50.01, and the average computation time per trial was 0.0068 seconds.

These experimental results confirm that not only does our attack work as expected, but in addition both the data complexity and the time complexity are very low, which makes it easy to carry out the attack in practice.

7 Application of Attack to TFHC2

Toorani and Falahati introduce TFHC2 in [10], together with a two-pass protocol ("Protocol A") and a one-pass protocol ("Protocol B"). The core cipher is identical to TFHC1, except for the technique used to generate the a_t value for each plaintext. In TFHC2, a hash-based message authentication code (HMAC), which is essentially a hash function parameterized by a secret key [11], is used instead of a standard unkeyed hash function $H(\,)$. Once Alice has selected a_0, she computes each subsequent a_t using $a_t = HMAC_{k'}(a_{t-1})$, where $k' = (k_{11} \| k_{12} \| k_{13} \| \ldots \| k_{nn} \| a_{t-1})$. Here the k_{ij} values are the n^2 entries in $\mathbf{K}$ in row-major order. This construction ensures that only a party in possession of both a_{t-1} and the secret key $\mathbf{K}$ will be able to compute a_t. However, because we bypass the hash function (or the HMAC), we do not need to compute a_t, and therefore our attack against TFHC1 can be applied to TFHC2 without modification to obtain $\mathbf{K}$.

The one-pass Protocol B for TFHC2 is is identical to the one-pass protocol for TFHC1, and therefore as soon as we obtain $\mathbf{K}$ using our attack, we can eavesdrop on all future communication between Alice and Bob that uses Protocol B.

The two-pass Protocol A for TFHC2, which is based on concepts from public-key cryptography, provides a higher level of security for the exchange of a_0 between Alice and Bob. Protocol A does not protect TFHC2 against the use of our attack to obtain $\mathbf{K}$ (since, again, we do not need to know a_0), but it may prevent us from simply eavesdropping on and decrypting future communication between Alice and Bob *after* we have learned $\mathbf{K}$, because at that point we *do* need to know the value of each a_0. On the other hand, it is interesting that Protocol A was also intended to thwart "intruder-in-the-middle" attacks [12] for any party that does not know $\mathbf{K}$. However, since we can use our attack to obtain $\mathbf{K}$, we can then carry out an intruder-in-the-middle attack against Protocol A.

8 Conclusion

In this paper we have presented a chosen-plaintext attack against two closely related Hill Cipher variants due to Toorani and Falahati, which we denote TFHC1 and TFHC2. Our attack bypasses the main security features of these ciphers—in particular, the incorporation of a strong one-way hash function such as SHA-1 to modify the encryption process—and exploits the relationship between key entries and any ciphertext produced from the encryption of the all-zero plaintext in order to recover the full key. We present computational results that confirm the effectiveness of our attack, and verify that it requires minimal time and space resources.

Acknowledgment

This research was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC).

References

[1] L. S. Hill, “Cryptography in an algebraic alphabet,” *American Mathematical Monthly*, vol. 36, no. 6, pp. 306–312, 1929.

[2] W. Stallings, *Cryptography and Network Security: Principles and Practice*, 5th ed., Boston: Prentice Hall, 2011.

[3] V. U. K. Sastry, D. S. R. Murthy, and S. Durga Bhavani, “A block cipher involving a key applied on both sides of the plain text,” *International Journal of Computer and Network Security*, vol. 1, no. 1, pp. 27–30, Oct. 2009.

[4] L. Keliher, “Cryptanalysis of a modified Hill Cipher,” *International Journal of Computer and Network Security*, vol. 2, no. 7, pp. 122–126, Jul. 2010.

[5] S. Saeedinia, “How to make the Hill Cipher secure,” *Cryptologia*, vol. 24, no. 4, pp.353–360, Oct. 2000.

[6] C. H. Lin, C. Y. Lee, and C. Y. Lee, “Comments on Saeednia’s improved scheme for the Hill Cipher,” *Journal of the Chinese Institute of Engineers*, vol. 27, no. 5, pp. 743–746, 2004.

[7] I. A. Ismail, M. Amin, and H. Diab, “How to repair the Hill Cipher,” *Journal of Zhejiang University SCIENCE A*, vol. 7, no. 12, pp. 2022–2030, 2006.

[8] C. Li, D. Zhang, and G. Chen, “Cryptanalysis of an image encryption scheme based on the Hill Cipher,” *Journal of Zhejiang University SCIENCE A*, vol. 9, no. 8, pp. 1118–1123, 2008.

[9] M. Toorani and A. Falahati, “A secure variant of the Hill Cipher,” *Proc. IEEE Symposium on Computers and Communications (ISCC’09)*, Sousse, Tunisia, Jul. 2009, pp 313–316.

[10] M. Toorani and A. Falahati, “A secure cryptosystem based on affine transformation,” *Journal of Security and Communication Networks*, vol. 4, no. 2, pp. 207–215, Feb. 2011.

[11] D. R. Stinson, *Cryptography: Theory and Practice*, 3rd ed., Boca Raton: Chapman & Hall/CRC, 2006.

[12] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography*, Boca Raton: CRC Press, 1997.

[13] J. Overbey, W. Traves, and J. Wojdylo, “On the keyspace of the Hill Cipher,” *Cryptologia*, vol. 29, no. 1, pp. 59–72, 2005.

[14] D. Poole, *Linear Algebra: A Modern Introduction*, 3rd ed., Boston, Massachusetts: Brooks/Cole, 2011.

[15] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Indroduction to Algorithms*, 3rd ed., Cambridge, Massachusetts: MIT Press, 2009.