Solving the Double Transposition Challenge with a Divide and Conquer Approach

George Lasry
Mishmar Hayarden 5
Givataim, 5358203 Israel
george.lasry@gmail.com

Nils Kopal
University of Kassel
Pfannkuchstr. 1
34121 Kassel
nils.kopal@uni-kassel.de

Arno Wacker
University of Kassel
Pfannkuchstr. 1
34121 Kassel
arno.wacker@uni-kassel.de

Abstract

The double transposition cipher was considered to be one of the most secure types of manual ciphers. It was extensively used in both World Wars and during the Cold War. In 1999, Otto Leiberich, the former head of the German federal office for information security, suggested that a double transposition challenge be published with specific parameters designed to ensure its security. Such a challenge was published by Klaus Schmeh in 2007. In November 2013 we solved the challenge using a ciphertext only hill climbing attack. We also solved the challenge using a dictionary attack. In this paper we describe both methods, which are based on a "divide and conquer" approach. We additionally discuss the impact of our solutions with respect to the general security of the double transposition cipher.

Keywords: Double Transposition, Columnar Transposition, Hill Climbing, Dictionary Attack, Cryptanalysis, Challenge, Manual Ciphers, Unsolved Ciphers, Doppelwürfel

1 The Double Transposition Cipher

The double transposition cipher has been one of the most popular manual ciphers. It did not require the use of a device for encryption and decryption. Because of its simplicity and its high level of security, it was often the cipher of choice for intelligence and secret operations organizations [12].

The process of encryption and decryption is relatively simple. First, two transposition keys, K_1 and K_2, must be chosen and agreed in advance. Keys are usually derived from keywords or key phrases which in turn are converted to numerical keys. Such keywords or key phrases are often taken from books or newspapers. As an example consider the encryption presented in Figure 1. Here, we encrypted the plaintext P = "`THISISASECRETTEXTENCRYPTEDBYTHEDOUBLETRANSPOSITIONCIPHER`" using the keys K_1 = "`KEYWORD`" and K_2 = "`SECRET`". First, we prepare the first transposition rectangle by writing down the first key K_1 and its numerical equivalent (Figure 1(a)). Underneath the key we fill the rectangle by writing the plaintext row-by-row. We then apply the first columnar transposition by changing the order of the columns according to the numerical equivalent of key K_1. This results in an intermediate rectangle (Figure 1(b)). After that, we prepare the second transposition rectangle by writing down the second key K_2 and its numerical equivalent (Figure 1(c)). Then we extract column-by-column the text from the intermediate rectangle in Figure 1(b) and write it row-by-row into the rectangle underneath the key K_2. We then apply the second columnar transposition by changing the order of the columns according to the numerical equivalent of key K_2. This results in the final

The original submitted version is available at `http://dx.doi.org/10.1080/01611194.2014.915269`. This version can differ from the camera ready paper (due to copyright restrictions).

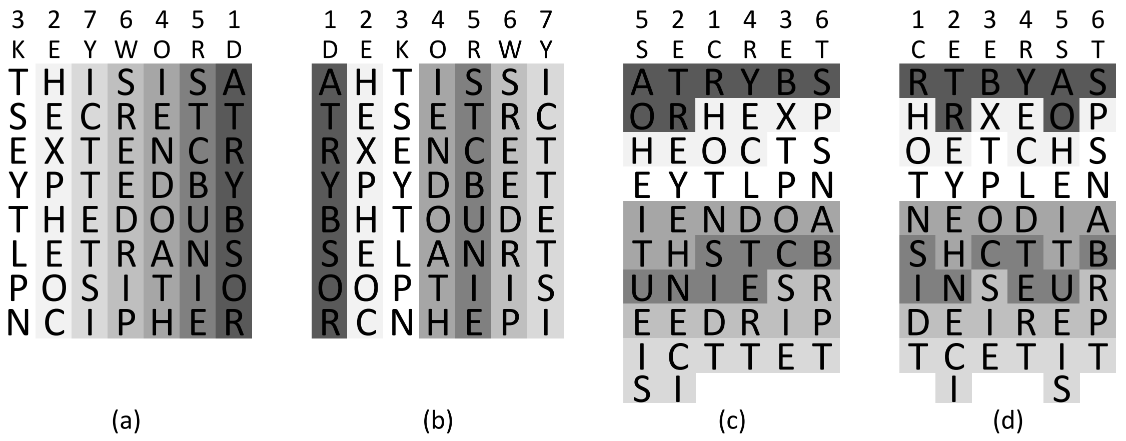

Figure 1: Example for the Double Columnar Transposition

rectangle (Figure 1(d)). The final ciphertext is extracted column-by-column from this rectangle yielding the ciphertext C = "`RHOTNSIDTTREYEHNECIBXTPOCSIEYECLDTERTAOHEITUEISSPSNABRPT`". To decrypt the ciphertext we apply the inverse steps. We first need to undo the second columnar transposition with K_2. After that, we undo the first transposition with K_1.

The German Army used the double transposition cipher (in German: "Doppelwürfel"[1]) in World War I in a less secure form by using the same key for K_1 and K_2. The French "Bureau de Chiffre", who called this cipher Übchi, regularly solved the cipher until the German Army replaced it with another cipher following leaks in the French press [12]. During World War II it was extensively used by different countries. In the US it was used by the Army either with the same or with different K_1 and K_2 keys and by the Office of Strategic Services (OSS) as an emergency cipher. In Great Britain it was used by the British Special Operations Executive (SOE) to communicate with its agents in continental Europe. The Czechoslovakian government in exile in London used it as well as the French Resistance and the German Abwehr operatives in Latin America [12, 2]. During the Cold War, the East Germany's Stasi used double transposition ciphers to communicate with agents in West Germany. West Germany's cryptographic agency, the "Zentralstelle für das Chiffrierwesen" (in English: center for ciphers), was able to find solutions using a computerized keyword dictionary attack [12]. In his 2012 book about unsolved ciphers, Klaus Schmeh estimated that the double transposition cipher might still be in use [12].

2 Prior Cryptanalysis of the Double Transposition Cipher

Several NSA declassified publications present the classical manual methods for the cryptanalysis of the double transposition cipher. In [5], Friedman presents a set of special cases of double transposition messages which can be solved relatively easily. Such as a perfect rectangle where the length of the ciphertext is equal to the product of the lengths of the two keys or when the operator forgets to apply the second transposition. In [6], Kullback presents the classical method of multiple anagramming which requires several messages of the same length and encrypted with the same key. He also presents a method to recover the key once the multiple anagram sequences have been identified. Furthermore, his book includes a dictionary attack, where keys are derived from words. However, this attack cannot be used for the challenge at hand as its keys were derived from longer key phrases.

In the most comprehensive document about classical cryptanalysis of the double transposition cipher, Barker [1] presents several manual methods including solutions for special cases, multiple anagramming, and a known-plaintext attack. The most generic method is the rotating matrix. Barker establishes an equivalence for a given ciphertext length between a double transposition cipher with keys of lengths $|K_1|$ and $|K_2|$ and a single transposition cipher with a key K of length $|K| = |K_1| \cdot |K_2|$. The rotating matrix method requires a ciphertext of at least $2 \cdot |K|$ letters making this method not applicable to the challenge at hand with only 599 letters.

Apart from those declassified publications, very few publications are available on this subject and

[1]The term "Doppelwürfel" literally translates to "double cube" which refers to the two transposition rectangles.

even fewer presenting modern methods for the cryptanalysis of the cipher. In [17], Tim Wambach presents a known-plaintext attack, which requires the knowledge of the full plaintext of the encrypted message. In the public Google Group sci-crypt and in a private mail Jim Gillogly briefly describes a hill climbing approach, which can achieve solutions for ciphers encrypted with short keys of up to 12 elements.

3 The Challenge

Since the 1950s, Otto Leiberich worked for Germany's main cryptographic agency, the "Zentralstelle für das Chiffrierwesen" in Bonn. He eventually became its director in 1972. During the Cold War, he and his team worked intensively on the cryptanalysis of double transposition ciphers. One of their results led in 1974 to the discovery of the spying activities of Günter Guillaume who was Willy Brandt's senior aide. Later, Otto Leiberich became the head of the "Bundesamt für Sicherheit in der Informationstechnik (BSI)" (in English: federal office for information security) in Germany. [12]

In order to encourage research on the double transposition cipher he suggested in 1999 that a double transposition challenge be published [7]. Leiberich's recommendations for the challenge included:

- Both transposition keys should be long enough: 20 to 25 elements.
- The lengths of the two keys should be co-primes (no common divisor except 1).
- The length of the ciphertext should not be a multiple of the length of either key.
- A ciphertext of approximately 500 characters (which is also approximately the product of the lengths of the two keys) should be used.

Those requirements were intended to avoid vulnerabilities, known to exist in special cases [5, 6, 1]. They were also based on Otto Leiberich's own experience with cryptanalysis of double transposition ciphers. In 2007 Klaus Schmeh published the double transposition challenge based on those guidelines [10]. He chose an English plaintext and encrypted it using two transposition keys, both derived from English key phrases longer than 20. The length of the plaintext was 599. The challenge was also published in the cryptographic challenges site "MisteryTwister C3" [14], in another book by Klaus Schmeh [12], and in several web sites and blogs [11, 14, 13]. The double transposition challenge was ranked as #5 of the top 25 unsolved ciphers [13], and was included in a list of the "World's Greatest Unsolved Ciphers" [3]. Otto Leiberich considered the cipher to be unbreakable. He literally stated, "Für mich wäre es eine Sensation, wenn jemand dieses Rätsel lösen könnte." with respect to Klaus Schmeh's challenge [12]. This sentence loosely translates to, "it would be a real sensation to me if someone could solve this riddle". Below we provide the ciphertext:

```
VESINTNVONMWSFEWNOEALWRNRNCFITEEICRHCODEEAHEACAEOHMYTONTDFIFMDANGTDRVAONRRTORMT
DHEOUALTHNFHHWHLESLIIAOETOUTOSCDNRITYEELSOANGPVSHLRMUGTNUITASETNENASNNANRTTRHGU
ODAAARAOEGHEESAODWIDEHUNNTFMUSISCDLEDTRNARTMOOIREEYEIMINFELORWETDANEUTHEEEENENT
HEOOEAUEAEAHUHICNCGDTUROUTNAEYLOEINRDHEENMEIAHREEDOLNNIRARPNVEAHEOAATGEFITWMYSO
THTHAANIUPTADLRSRSDNOTGEOSRLAAAURPEETARMFEHIREAQEEOILSEHERAHAOTNTRDEDRSDOOEGAEF
PUOBENADRNLEIAFRHSASHSNAMRLTUNNTPHIOERNESRHAMHIGTAETOHSENGFTRUANIPARTAORSIHOOAE
UTRMERETIDALSDIRUAIEFHRHADRESEDNDOIONITDRSTIEIRHARARRSETOIHOKETHRSRUAODTSCTTAFS
THCAHTSYAOLONDNDWORIWHLENTHHMHTLCVROSTXVDRESDR
```

4 Solving the Challenge

The purpose of the work presented in this paper was to develop new cryptanalytic methods with the immediate goal of solving the challenge published by Klaus Schmeh in 2007. Therefore, additional research is still needed to accurately assess the full potential of our developed methods for a general solution of the double transposition cipher. Nevertheless, we were able to identify several general vulnerabilities, which we present in the last section (see Section 5).

The remainder of this section is organized as follows: We first give a brief walk-through of the entire process and an overview of the methods. After that, we present in Section 4.2 our prior work on the single columnar transposition and in Sections 4.3 to 4.8 we discuss each step and its consequences in detail.

4.1 Overview

Some of the building blocks and insights for solving the double transposition challenge are a direct result from our prior research on solving difficult cases of single columnar transposition ciphers. We present the relevant facts from this research in Section 4.2 and a detailed technical publication of this work is currently under preparation.

We discuss our first attempt to solve the double transposition cipher challenge in Section 4.3 and refer to it as *Step 1.* In this step we explored a hill climbing algorithm, which searches in parallel over both the K_1 and K_2 key spaces. However, we found that this approach was only successful for keys lengths up to 15, which is less than required for the challenge at hand.

Our second attempt (*Step 2*) was the application of a known-plaintext attack as described in Section 4.4. In order to apply this attack, we tried to guess parts of the original plaintext. However, this was also a dead end for the challenge, since the attack itself required a relatively long known plaintext sequence, which we did not have. Additionally, we were not able to successfully guess any parts of the plaintext.

It became clear, that we needed to reduce the huge search space spanned by K_1 and K_2 together. Hence, in *Step 3* we evaluated several alternatives for an effective "divide and conquer" approach. The general idea is, that if we can first find the second transposition key K_2 then finding K_1 is just a matter of solving a single transposition cipher. For that purpose, we developed the Index of Digraphic Potential (IDP) as an effective and cost-efficient fitness measure for K_2 as discussed in Section 4.5.

Armed with the knowledge from Step 1 and Step 3 we modified the hill climbing algorithm to incorporate the IDP and to search over the K_2 keyspace while ignoring K_1. On November 25th we achieved our first breakthrough and solved the challenge based on a partial solution from this hill climbing method. We obtained the numerical keys for both K_1 and K_2 as well as the original plaintext. We sent the solution to Klaus Schmeh, who acknowledged its correctness. We refer to this first breakthrough as *Step 4* and describe its details in Section 4.6.

Even though the challenge was solved, we did not stop there. We also implemented an efficient K_2 dictionary attack based on the IDP and using a database of known expressions as described in *Step 5.* With this dictionary attack we achieved our second breakthrough: it also produced – independently from our first solution – the correct solution for the challenge and additionally the original English key phrase used to create the K_2 transposition key. We discuss the details of this step in Section 4.7.

The only part missing at this point was the original English key phrase for K_1 equivalent to the already known numerical value of K_1. In our final step we, therefore, relied on earlier work by Jim Gillogly to deduce the English key phrase for K_1. Hence, in *Step 6*, as described in Section 4.8, we finally had all the elements of the solution, i.e. the plaintext, the transposition keys, and the key phrases.

4.2 Own Preliminary Work

Preceding our work on the double columnar transposition cipher and the challenge at hand, we investigated and evaluated new general solutions for the single columnar transposition cipher. The most difficult cases are very long key lengths and incomplete transposition rectangles. Furthermore, for incomplete transposition rectangles the worst case is when about half of the columns are longer than the other columns. We found that our new approach is able to successfully decrypt cipher messages in the worst case with key lengths up to 120. In the best case, our algorithm can even cope with key lengths up to 1000.

Our new approach for the single columnar cipher is based on a hill climbing algorithm in two phases. For the first phase we developed two new fitness functions, i.e. the *pairing score* and the *alignment score.* The pairing score relates to the likelihood of any column j being right next to another column i. The alignment score reflects the precise starting position and alignment of the text of each column. For the second phase of hill climbing, we use plaintext quadgram log-frequencies as a fitness score. Additionally, we show that segment-wise transformations, e.g. swapping n consecutive elements of the key with another segment of n consecutive elements, are more effective than transformations on singular key elements, such as swapping key element i with key element j.

With our research on the single columnar transposition we gained important insights for the work on the double transposition, and in particular for the development of the Index of Digraphic Potential. We also reused some of the building blocks such as segment-wise transformations. A publication with all technical details and an evaluation of this research on the single columnar transposition is currently under work and will be available soon.

4.3 Step 1: Hill Climbing over K_1 and K_2 in Parallel

Generally, a hill climbing algorithm for cryptanalytic purposes consists of the following steps:

1. Create an initial key, which is usually a random key. Therefore, this method is sometimes called *shotgun hill climbing.*
2. Apply transformations on the key. In most cases these are small modifications of the current key to produce "neighboring" keys.
3. Evaluate the *fitness* of each such new key: The fitness score of a key is determined by decrypting the ciphertext with this key and calculating some scoring function on the resulting text.
4. If the fitness score of any new key (i.e. the transformed keys) is higher than the score of the best key so far (the *best score*), keep this new key as the new *best key.*
5. Repeat (2), (3), (4), for all possible transformations as long as the best score can be improved.
6. If the best score cannot be improved, repeat the whole algorithm, starting from (1).

We implemented such a hill climbing algorithm for the double transposition. In contrast to the general one, our algorithm searches for the solution for K_1 and K_2 in parallel. In each iteration it looks for a change in either K_1 or K_2 which may improve the fitness score of the resulting decrypted text. We used log-frequencies of plaintext trigrams as a measure for the fitness. The transformations were segment slides (rotating or shifting a segment of the key), segment swaps as well as 3-party swap transformations.

This algorithm requires the knowledge of the correct key length. In case this is not known, hill climbing must be applied on the ciphertext for each possible combination of the key lengths. For the case of the challenge at hand the lengths could only be 21, 22, 23, 24, or 25 and co-primes. Hence, there are only 16 possible key length combinations. For the remainder of this paper and for each method which we present, we implicitly test all those combinations and can therefore assume to know the key lengths.

We tested our new algorithm on simulated texts and keys with particular attention on worst case incomplete rectangles. As already mentioned before, such a worst case occurs, when the number of long columns almost equals the number of short columns. We tested worst cases for the K_1 transposition rectangle as well as for the K_2 transposition rectangle. In these cases we found this method to have about 90% probability of success for keys lengths up to 13, and about 50% for key lengths of 14-15. Additionally, we observed that for the following special cases our algorithm succeeded also with longer keys:

- The length of the ciphertext is exactly equal to the product of the lengths of the two keys, i.e. $|C| = |K_1| \cdot |K_2|$. In this case we have a *perfect transposition rectangle* i.e. a complete transposition rectangle for both the K_1 and K_2 transpositions.
- The length of the ciphertext $|C|$ is a multiple of only one of the two key lengths, i.e. $|K_1|$ or $|K_2|$. In this case one of the two columnar transpositions results in a complete transposition rectangle.
- The length of the ciphertext only slightly deviates from the product of the lengths of the two keys, i.e. $|C| = |K_1| \cdot |K_2| \pm 1$, resulting in an *almost perfect transposition rectangle.*

The first two cases are well known in the classical literature [5]. Solutions are easier to find if one or both of the transposition rectangles are complete, as there is less ambiguity in the position of the columns after transposition. The case of an almost perfect transposition rectangle was less known, and could be relevant for the challenge at hand if the used key lengths would have been 24 and 25, since $|C| = 599 = 24 \cdot 25 - 1$ or $|C| = 599 = 25 \cdot 24 - 1$. Therefore, we tested this assumption by running our algorithm with the challenge ciphertext with the key lengths $|K_1| = 24$, $|K_2| = 25$ and $|K_1| = 25$, $|K_2| = 24$. Unfortunately, those tests failed to produce a solution, but they allowed us to rule out those two combinations of key lengths. The number of possible key lengths combinations was thus reduced from 16 to 14. Still, we needed another approach to solve the challenge.

4.4 Step 2: Known-Plaintext Attack

Next we tried to solve the cipher with a known-plaintext attack. No such text was available for the challenge, but we assumed that the plaintext might be related to Klaus Schmeh domains of expertise. Therefore, some words or expressions might be guessed. For that purpose, we created a database of expressions and sentences based on several cryptography and computer security books and articles. We implemented a known-plaintext attack by adapting the hill climbing algorithm described in Step 1 and only changing the scoring function. The scoring function was simply the number of correctly recovered letters in the decrypted text compared to the expected plaintext.

We showed with simulations that obtaining a solution for keys longer than 20 would require hundreds of characters. Obviously, this is more than one can possibly guess. To improve the algorithm, we implemented a "hybrid" approach. We allocated 50% of the score to plaintext recovery, as described above, and the other 50% to a trigram score, as described in Step 1. With this improved hybrid method, we could solve simulated ciphers with parameters similar to those of the challenge with only 100 letters (out of 599) of known or guessed plaintext and in rare cases with only 60. Again, that was not enough to solve the challenge. While this improved algorithm was useful at a later stage (Step 4), we needed a new and more powerful approach.

4.5 Step 3: Reducing the Search Space

One of the main challenges in searching for the keys of a double transposition cipher is the size of the combined key space. Basically, the key space is $|K_1|! \cdot |K_2|!$. For keys with lengths up to 25, this is about 2^{162} or roughly equivalent to the key space of a 3DES encryption with three different keys. This creates significant challenges with our algorithm in Step 1, which searches K_1 and K_2 in parallel. The size of the key space creates similar challenges for dictionary attacks.

The complexity of a dictionary attack, when searching for both K_1 and K_2 in the dictionary is $O(n^2)$ where n is the dictionary size. Short keys are usually derived from single keywords (e.g. "transposition") while longer keys are usually derived from full sentences or expressions such as "adifficultcipher" or "thiscipherisnoteasy". Thus, dictionary attacks for double transpositions are easier for shorter keys compared to longer keys, as there are less combinations to check. Word dictionaries are available with a number of entries n, varying from tens of thousands and containing the most common words, up to several hundreds of thousands with inflections, rare words, and names of places. Thus, there may be between $n^2 = (10\,000)^2 \approx 2^{27}$ to $n^2 = (400\,000)^2 \approx 2^{38}$ combinations to check. In practice, we don't need to check all those combinations since only words matching the required lengths need to be tested. We roughly estimated that testing about $100\,000\,000 \approx 2^{27}$ combinations of dictionary key words would be a feasible attack for our available hardware. However, this attack would cover only all combinations of single words, which is not good enough for the challenge at hand. A key of length 20 to 25 may only be derived from an expression or a sentence. Databases of the most common expressions and sentences are also available, starting with $n = 1\,000\,000$ entries [4]. However, for a good coverage, a larger database may often be required. Such databases are available, with up to 98 billions of entries [15]. Assuming that 10% of the entries are of lengths relevant to the challenge, we would need to test about $(98 \cdot 10^9/10)^2 \approx 2^{67}$ cases. Clearly, this was not practical for this work.

4.5.1 A Divide and Conquer Approach

To achieve any kind of breakthrough, we needed a method to reduce the key space. Specifically, we needed an approach to find one key independently of the other. Furthermore, if we could find a method for *scoring* one key without knowing the other, this would open the door for effective "divide and conquer" hill climbing or dictionary attacks. We focused on the case of first finding or scoring the second transposition key K_2, independently of K_1. Once the second transposition is undone using a known or putative K_2 key, we are just left with a single transposition, i.e. the first transposition with the K_1 key. We considered a first, albeit naive, method for scoring a given putative K_2 without knowing K_1 a priori as follows:

1. First undo the second transposition with the assumed K_2. This putative K_2 could be the result either from the current iteration of hill climbing or from a dictionary search.

2. Find that K_1 which produces the plaintext with the highest bigram score after decryption. This can be done by solving the K_1 transposition, which can now be treated as a single transposition. For doing so we could rely on our own prior work for solving single columnar transpositions with hill climbing as described in Section 4.2. As a fitness function we use bigram log-frequencies.

3. Use the best bigram score found in (2) as the fitness score for the assumed K_2.

For the challenge at hand the transposition rectangle is incomplete for all possible K_1 key lengths. Based on our prior findings, our algorithm for solving the incomplete single columnar transposition on K_1 would take around one minute to complete on a standard home PC. This time consuming process must be repeated for each putative K_2 key. Clearly, this approach is not practical for key phrase dictionary attacks on K_2 with approximately $n = 98 \cdot 10^9/10 \approx 2^{33}$ sentences and expressions to check. It is neither practical for hill climbing over K_2. For a key of $|K_2| = 25$ elements there are about $3 \cdot 25^3 = 46\,875 \approx 2^{16}$ possible transformations and resulting putative K_2 keys to be checked for each iteration of hill climbing. This translates into hundreds of hours per iteration, while tens of such iterations may be required.

4.5.2 The Index of Digraphic Potential

Even though the naive approach from last section is not practical, it provides some valuable insight paving the way for our new fitness function as described in this section. The naive method – though highly inefficient – provides a value, which reflects the best possible plaintext bigram (or digraph) score achievable with a putative K_2 key. This value may also be viewed as a quantitative index, reflecting the "digraphic potential" of a given K_2. Therefore, we named this score the *Index of Digraphic Potential* (IDP). We hypothesized that the closer a putative K_2 key is to the correct K_2 key, the higher the potential to reconstruct the original plaintext and its bigrams. From this we deduced that there should be a negative correlation between the number of errors in K_2 and its IDP. Hence, we looked for an efficient method to compute this IDP for a putative K_2 key without having to fully solve the K_1 transposition as described in (2) of the naive approach.

The following insight came from our prior research and solutions for single transpositions with incomplete rectangles. For simplicity, we only consider here a single transposition with an incomplete rectangle and a K_1 key of length $|K_1|$. In the encryption process the plaintext is written in rows, where the length of each row is $|K_1|$. In an incomplete rectangle and before transposition, the leftmost columns are longer than the rightmost columns by one row. Next in the encryption process, the order of columns is reshuffled using the transposition key K_1 and the resulting text is extracted column by column to form the ciphertext. A cryptanalyst does not know where the text of each original plaintext column appears in the ciphertext since he does not know K_1. Nevertheless, he does know that each one of the transposed columns appears in the ciphertext as a continuous segment. Furthermore, for a key K_1 of length $|K_1|$ and a ciphertext C of length $|C|$ the number of *full* rows in the transposition rectangle equals to $r = \lfloor \frac{|C|}{|K_1|} \rfloor$. Therefore, he may also assume that the length of each column may either be r or $r + 1$. As a result, there can be several possible starting and ending positions for each column since preceding columns might each be either a long or a short column. It is clear that the first column in the ciphertext always starts at position 1 and that the last column always ends at position $|C|$. If the first column is short then the second column will start at position $r+1$. If instead the first column is long, then the second column starts at $r + 2$. The same reasoning can be further applied for the next columns and also backwards from the end of the ciphertext to determine the range of possible starting positions for each column. If the number of long columns is approximately equal to the number of short columns, then columns in the "middle" of the ciphertext have approximately up to $\frac{|K_1|}{2}$ possible starting positions. In order to solve such a single transposition and to find the $|K_1|$, we need to determine the original order of the columns. For that purpose, we want to evaluate the likelihood that a given column j in the ciphertext was the right neighbor, in the original plaintext, of another ciphertext column i. One way to evaluate this is to compute the sum of log frequencies of all the bigrams created by juxtaposing column j next to the right of i. As discussed above, there can be several possible starting positions for both columns. Therefore, this bigram score must be computed for all those possible starting positions of both columns i and j in the ciphertext. Such a process may not only indicate which column j is likely to be the neighbor next right to column i, but it may also help to determine the exact positions of those columns in the ciphertext and how they are aligned next to each other. This analysis and the techniques described above lie at the core of our prior solutions for solving complex single transpositions.

We applied the same analysis and a similar approach to develop an efficient algorithm for computing the IDP of putative K_2 keys. This algorithm is deterministic and it computes the IDP of a putative K_2 key without solving the K_1 columnar transposition:

1. First undo the second transposition with the given (putative) K_2 key.
2. Prepare an empty matrix $B[i, j]$ and perform for each possible combination of ciphertext columns i and j the following calculations:
   (a) Compute the sum of log frequencies of all the bigrams created by juxtaposing column j to the right side of column i. To normalize the result, divide this sum by the number of rows. Store this value in the matrix cell $B[i, j]$.
   (b) Repeat this calculation for all possible starting positions of both i and j continuously updating the matrix cell $B[i, j]$ with the best (highest) value found.
3. As a result from the last operation we have a matrix $B[i, j]$ with the best possible digraphic values for all pairs of columns i and j. However, in a real transposition, each column (or key element) can have only one right neighbor, as well as only one left neighbor. Therefore, in order to further reduce the matrix perform the following operations iteratively:
   (a) Select the column pair (i, j) from the matrix $B[i, j]$ with the highest value.
   (b) Mark j as the likely right neighbor of i, and i as the likely left neighbor of j. Furthermore, mark all $B[i, *]$ and $B[*, j]$ cells except $B[i, j]$ as invalid.
   (c) Repeat (a) and (b) with columns i which do not yet have a likely right neighbor and columns j, which do not have yet a likely left neighbor, until all columns have been assigned both a right and a left likely neighbor.
4. Sum all $B[i, j]$ values of all pairs, considered as likely neighbors. We define this sum as the value for IDP(K_2) for the given key K_2.

This implementation of the IDP can be performed on a 3.4 GHz Intel Core i7 PC with 8 cores and parallel processing at the rate of 20 000 calculations per second. Compared to the approximate 60 seconds with the naive approach, this is an speed improvement by a factor of $1 : (60 \cdot 20\,000) = 1 : 1\,200\,000$. Additionally, with the naive approach, solving for K_1 may take more or less cycles for different keys K_2. In contrast, the above method for calculating the IDP is deterministic in time, i.e. it always takes the same time (cycles) for any K_2. This eases the implementation of parallelization.

4.5.3 Evaluation of the IDP

Before integrating the IDP into either hill climbing or a dictionary attack, we first needed to assess its suitability as a fitness function for K_2. We, therefore, analyzed how the IDP value is affected by the number of erroneous elements in the key. To do so, we defined the degradation of the IDP, $D(x)$, for x perturbations in a key as the difference between the IDP of the correct key and the IDP of the same key but after x perturbations. Hence, we formally define the degradation of the IDP as $D(x) := \text{IDP}(K^{(0)}) - \text{IDP}(K^{(x)})$, where $K^{(x)}$ denotes a key with x perturbations.

We measured this value by starting with a correct key $K_2^{(0)} = K_2$ and inserted iteratively a number of artificial perturbations. To create the erroneous keys $K_2^{(x)}$, we swap in each iteration two single elements of the key $K_2^{(x)}$ making sure that a swap does not correct an error created by a previous swap. We simulated $D(x)$ for 100 000 different K_2 keys of length 23 and applied for each key up to 23 perturbations, i.e. $x \in [0 \ldots 23]$. We present the results from this simulation in Figure 2. The X-axis represents the number of perturbations and the Y-axis the degradation in the IDP, $D(x)$.

From this simulated data we were able to verify the two most important properties of the IDP. First, it is highly *selective* for a correct K_2 key. We found only one single case with a K_2 slightly different from the correct one, which produced an IDP better than for the correct key. Therefore, the IDP can be highly effective for a K_2 only dictionary attack with $O(n)$ complexity. The K_2 key phrase, which produces the highest IDP, is almost certainly the correct key or at least an almost correct one.

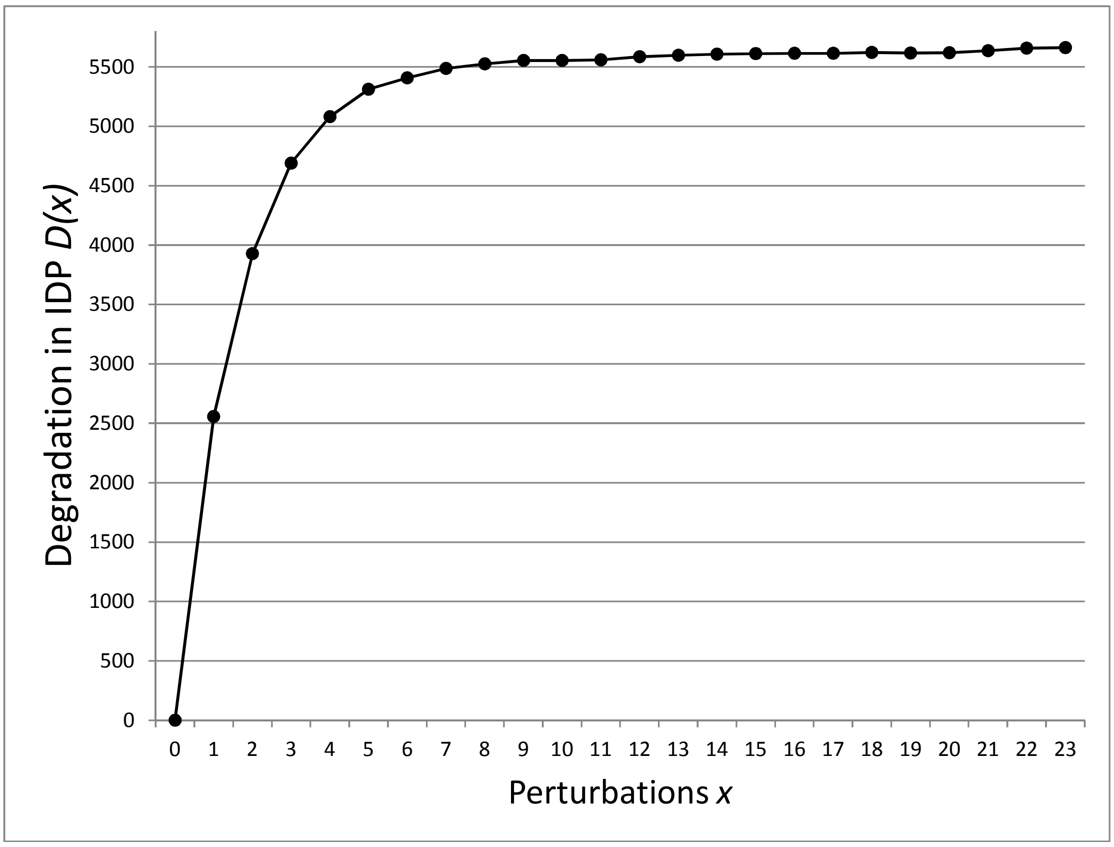

Figure 2: The Average Degradation of the IDP

Our second important finding is, that the IDP is *monotonic* for a wide range of perturbations and that the changes are steeper as we get closer to the solution (less than 10 errors). This property is especially useful and important for hill climbing on K_2 without knowing K_1, as we show in the next section.

4.6 Step 4: Improved Hill Climbing

From the evaluation of the IDP in the last section, we knew that this fitness function is highly selective for the right K_2 and also monotonic for partially correct K_2 keys. These properties make it a highly suitable choice for a K_2 hill climbing algorithm to solve double columnar transposition ciphers. Therefore, we developed an improved “divide and conquer” hill climbing algorithm based on the IDP with the main search being over the K_2 space while first ignoring K_1:

1. Generate a number of random K_2 keys and for each one compute the IDP. Keep a subset of the keys with the highest IDPs.
2. For each key in the subset obtained by (1), apply a simple “left-to-right” improvement heuristic and keep only the subset with the best IDPs after this improvement. The main purpose of this improvement is to provide the hill climbing part in the next phase (3) with such initial K_2 keys which are more likely to be closer to the solution. Perform the following operations to achieve the “left-to-right” improvement heuristic:
    (a) For each key element i starting with $i = 1$, check whether swapping the key element i with another key element j on its right ($j > i$) results in a key with a higher IDP score. If there is an improvement, perform the swap and keep the resulting key.
    (b) Repeat (a) for the next i until the penultimate key element $i = |K_2| - 1$ is reached.
3. For each key in the subset obtained by (2), run a K_2 hill climbing algorithm as follows:

   (a) Perform all possible segment slide, segment swap, and 3-partite swap transformations on the current K_2. Assess the fitness for each transformed K_2 key by calculating the IDP.
   (b) If IDP has improved, keep the new key as the current best key and repeat (a). When reaching a maximum and no more improvements are possible, compare this best key IDP to the overall best IDP, achieved by hill climbing, with any of the keys from the original (2) subset. If higher, keep that key as the best overall K_2 key.

4. Using the best overall K_2 key from (3), perform the following:
   (a) Undo the second transposition using that best overall K_2. Solve the remaining K_1 single columnar transposition using hill climbing and trigrams for scoring the fitness. This yields a key K_1 which in turn results in a plaintext with the highest trigram score. This may already be the correct K_1.
   (b) In order to improve and finalize the K_2 key, use the best K_1 key from (a) to perform the following hill climbing process:
      i. Check all possible segment slides, segment swaps, and 3-partite swaps transformations for the current K_2. However, instead of computing the IDP for each transformation, do the following:
         A. Fully decrypt the ciphertext using the current best K_2 and the best K_1 from (a) and compute the decrypted plaintext trigram score.
         B. If a higher trigram score was found than the previous best, keep this key as the new best K_2.
      ii. Repeat (i) until the trigram score cannot be further improved, i.e. no better K_2 has been found.
5. Repeat (1) to (4) until a fitness threshold is reached. In that case the keys have probably been found. The algorithm probably failed if a maximum number of iterations have been done without reaching this fitness threshold.

We tested this new algorithm, and it provided an immediate and significant improvement. The new algorithm has a success probability of over 60% for finding keys of length 19 and 20 for K_1 and K_2, respectively. For key lengths of 18 to 19 we even have a success probability of 90%. This is a significant improvement in comparison with the previous hill climbing of Step 1, which could only solve keys with lengths up to 13-15. Unfortunately, this was still not enough to solve the challenge at hand. Nevertheless, this is still a big step forward, and we intend to perform additional research in the future to fully assess the performance of this improved algorithm. For now, we turn our attention to further optimizations we introduced to the algorithm, specifically tailored to the challenge at hand.

4.6.1 Optimizations

It is publicly known that the keys for the challenge are derived from English key phrases. We hypothesized that based on this knowledge the key space for hill climbing could be reduced – or at least the search could be optimized. In any language some letters occur more often than others. Hence, in an English sentence or expression some low-frequency letters, such as Z or J, are likely to be absent, while high-frequency letters such as E and T are likely to appear several times. Additionally, when a letter appears more than once in a key phrase, the various occurrences of that repeated letter are represented in the equivalent numerical key by successive numbers. As an example consider the keyword “`REFERENCE`”, with the letter E occurring four times. The numerical key for this word is “8,**2**,6,**3**,9,**4**,7,1,**5**”, i.e. the occurrences of E are represented by the consecutive numbers **2**,**3**,**4** and **5**. A long key derived from sentences or expressions should contain at least several occurrences of such patterns. This also means that an extreme case of a key, such as “21, 20, 19, 18, 17, 16, 15, 14, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1” is unlikely to be derived from a sentence or expression. Furthermore, the longer the key phrase the higher the probability of the numerical value of each key element being closer to the position in the alphabet of the letter it represents. For example, if a numerical key has 25 elements and one of the elements is 5, it is more likely to represent an original E (or D or F), than a T (or S or U). We adapted the K_2 fitness function, so that it takes those patterns into consideration and assigned them additional scoring points in addition to the IDP. While we could see some improvement – mainly with the speed of the convergence of the algorithm – it still was not enough to solve the challenge. To fully quantify the benefits of this optimization, further research is needed.

The second optimization was a seemingly minor one, but eventually with dramatic consequences. In simulations, we could often see cases where although the algorithm would fail to produce a complete solution, it sometimes produced either a partially correct K_2 or a partially correct K_1 and even sometimes a fully correct K_1. In those cases, the trigram score of the decrypted plaintext with partially correct keys would be relatively higher than a "noise" trigram score resulting from random keys. A closer look at the resulting decrypted plaintexts showed that sometimes some fragments of words were discernible. This caught our attention and we wanted to take advantage of such partial solutions. We added an offline task which processes decrypted texts with higher trigram scores that were produced by hill climbing. It would scan those texts for possible words against a database of 10 000 words. To recognize possible words from fragments, we used a modified Levenshtein distance criteria. The standard Levenshtein measure takes into account possible wrong letters, missing letters, or displaced letters, but only in a sequential text. In transposition ciphers, displacement can often occur into cells in the rows right above or below the correct row and not just on its right or left. We modified the distance calculation accordingly, and the distance threshold was set to 3 or below. Additionally, we considered only words with 6 or more letters. During the development process, we kept one server running with the latest version of hill climbing against the challenge cipher, checking each one of the possible 14 combinations of the key lengths.

4.6.2 First Breakthrough

While the server was running with our improved algorithm and with both optimizations in place, we periodically checked the corresponding logs. Most of the time, common words, such as "letter", were produced. This was most probably due to the fact that 4 of the 6 letters are high frequency letters, appearing multiple times in the ciphertext. Thus, such particular assemblies of letters were most likely to be the result of random displacement and wrong keys. At some stage, however, an interesting and longer word was spotted. The word was "pernicious". And another word in the vicinity of the first word was "sphere", which also could be "atmosphere". We started a "detective work" to follow this lead.

In [12], Klaus Schmeh wrote that the text is at least 100 years old, so it was likely to be from a book. The word "pernicious" is not a very common word in modern English texts, so we tried a Google search for that word on the Gutenberg project. This produced a large number of books. To narrow down the search, we used both "pernicious" and "sphere" for the search, as well as "pernicious" and "atmosphere". The Google search for this last combination showed 6 books in the Gutenberg project. We downloaded the books and looked at candidate segments of length 599, on which we applied the known-plaintext attack from Step 2. We obtained almost instantly a full match with a paragraph from *Mistress Wilding* by Raphael Sabatini. We had finally solved the challenge, as we now had the ciphertext and the two numerical keys K_1 and K_2. The lengths of the keys turned out to be 21 for K_1 and 23 for K_2, not too far away from 20. A closer look showed that 23 might not have been an optimal choice as $599 = 26 \cdot 23 + 1$ resulting in an almost complete K_2 transposition rectangle. This is still harder than the very weak case with key lengths 24 and 25, i.e. $599 = 24 \cdot 25 - 1$. However, probably it is not the most secure choice. We sent the full plaintext with the numerical keys to Klaus Schmeh, who acknowledged the solution. In the following we present the first results of our long journey, i.e. the decrypted plaintext and the numerical keys with a corresponding representation as letters:

```
THEGIRLHADARRIVEDATLUPTONHOUSEAHALFHOURAHEADOFMISSWESTMACOTTANDUPONHERARRIVALSH
EHADEXPRESSEDSURPRISEEITHERFEIGNEDORREALATFINDINGRUTHSTILLABSENTDETECTINGTHEALA
RMTHATDIANAWASCAREFULTOTHROWINTOHERVOICEANDMANNERHERMOTHERQUESTIONEDHERANDELICI
TEDTHESTORYOFHERFAINTNESSANDOFRUTHSHAVINGRIDDENONALONETOMRWILDINGSSOOUTRAGEDWAS
LADYHORTONTHATFORONCEINAWAYTHISWOMANUSUALLYSOMEEKANDEASELOVINGWASROUSEDTOANENER
GYANDANGERWITHHERDAUGHTERANDHERNIECETHATTHREATENEDTOREMOVEDIANAATONCEFROMTHEPER
NICIOUSATMOSPHEREOFLUPTONHOUSEANDCARRYHERHOMETOTAUNTONRUTHFOUNDHERSTILLATHERREM
ONSTRANCESARRIVEDINDEEDINTIMEFORHERSHAREOFTHEM
```

The first numerical key K_1 is "16,10,15,17,1,20,21,11,19,5,14,12,8,3,7,4,2,18,6,13,9" which can be represented with letters as K_1 = "PJOQATUKSENLHCGDBRFMI". The original cleartext from above was first transposed with this key K_1, resulting in the following interim ciphertext:

```
IHMHSGTGCVECNGDYWMOEETOHAURNEDUTEEFTTOEHYUNGOUORGHAOHOTSOIFALILSTTANTOATAMSEDAI
ITMENMEOOHETNARNDRROAFNLEUTNINTSEFDAERURLLUASTSESOINTHCOPFEUSIETAAATNEIIHRFHTDO
UIYTETSUALRHVHCSSAEHHNEOFLRTAENATACOERCERLMARABMOMOSDNUHOANRHDNPOHAIUEDEEIENTRN
EHMACLGNRTNTENAIRHTDPPFGTWHOEFVWLIYAANNEONOHEHHSSADDTEECUEEDGRTKEIIEHEYFOETRATV
PEARTDIENOOTWEAETERUHDRTLHNDHDTANEAHSOWNANAEAOASULREAHSIRRLALNTHANONSDOHEVELRNT
NMEOOORERIAETLAIIANSSNDEFSDNEDATPAUXRNCAOMDRARSELWDAECMATTVSGNFNEIUNSRHIINLDAOR
GHDRPCRRIRARTHIIDDWREOTEERSVGHRAUOTIARAEWRSOIAFEEDSDSTHADTEMEOHONDHROIESNHTAORI
TRIHAEURROMERTEDOLUSREESHRIQTNINOYESWNRTRRHMEF
```

The numerical key K_2 is "19,21,6,20,17,14,4,7,22,1,15,2,8,18,12,9,23,13,5,10,16,3,11" and can be represented in letters as K_2 = "SUFTQNDGVAOBHRLIWMEJPCK". Transposing the interim ciphertext with this K_2 yields the original ciphertext:

```
VESINTNVONMWSFEWNOEALWRNRNCFITEEICRHCODEEAHEACAEOHMYTONTDFIFMDANGTDRVAONRRTORMT
DHEOUALTHNFHHWHLESLIIAOETOUTOSCDNRITYEELSOANGPVSHLRMUGTNUITASETNENASNNANRTTRHGU
ODAAARAOEGHEESAODWIDEHUNNTFMUSISCDLEDTRNARTMOOIREEYEIMINFELORWETDANEUTHEEEENENT
HEOOEAUEAEAHUHICNCGDTUROUTNAEYLOEINRDHEENMEIAHREEDOLNNIRARPNVEAHEOAATGEFITWMYSO
THTHAANIUPTADLRSRSDNOTGEOSRLAAAURPEETARMFEHIREAQEEOILSEHERAHAOTNTRDEDRSDOOEGAEF
PUOBENADRNLEIAFRHSASHSNAMRLTUNNTPHIOERNESRHAMHIGTAETOHSENGFTRUANIPARTAORSIHOOAE
UTRMERETIDALSDIRUAIEFHRHADRESEDNDOIONITDRSTIEIRHARARRSETOIHOKETHRSRUAODTSCTTAFS
THCAHTSYAOLONDNDWORIWHLENTHHMHTLCVROSTXVDRESDR
```

4.7 Step 5: Dictionary Attack

Besides improving the hill climbing algorithm, we started in parallel to work on an improved dictionary attack. The newly developed IDP enabled a powerful K_2 dictionary attack with only $O(n)$ complexity, where n is the amount of entries in the dictionary. The IDP also provided a speed improvement of a factor of 1 : 1 200 000 versus the naive approach. First, we started with an attack where the key phrase is taken from a book. For a typical book, containing 500 000 characters and an average of 5 characters per word, there are about 100 000 possible starting positions for a key phrase. The program produced solutions for such simulated cases in less than 10 seconds by processing 20 000 K_2 keys per second. In all of our simulations with keyphrases from books, the K_2 key derived from the correct keyphrase consistently produced the highest IDP score. While those trials confirmed the validity of using the IDP for a K_2 dictionary attack, for the challenge we did not know from which book the key phrases were taken, nor did we know whether at all they were from a book. So we looked for more comprehensive solutions in the form of a database of expressions and sentences. We used a mid-size free database of N-Grams from the Corpus of Contemporary American English [4]. We integrated this database, which contains about $n = 1\,000\,000$ entries, into our dictionary attack. The dictionary attack took only a few minutes to test all the expressions and phrases from this database. The one with the highest IDP score was "PREPONDERANCEOFEVIDENCE" (Preponderance of Evidence), which is a legal term unrelated to cryptography. This key phrase was equivalent to the numeric key found in Step 4 - "19,21,6,20,17,14,4,7,22,1,15,2,8,18,12,9,23,13,5,10,16,3,11". Hence, the challenge could be solved independently with a second method. We found this second solution two days after the first one.

Besides its $O(n)$ complexity, a major advantage of this K_2 dictionary attack is that it works regardless of the length of K_2. Obviously, it can find a solution only if the key phrase appears in the used database. However, databases with a large number of entries are available [15]. Furthermore, it is possible for organizations with massive computing power to enlarge existing databases by creating more combinations and using syntax analysis to rule out low value combinations. We now had the key phrase for K_2. The last piece of the puzzle still missing was the key phrase for K_1.

4.8 Step 6: Wrapping-Up – Finding the Last Key Phrase

A first attempt to find the key phrase for K_1 using the reduced 1 000 000 entry database was not successful. Either a bigger database was needed or we needed outside help. In *Decoding the IRA* [8], Jim Gillogly describes a similar challenge he faced while solving the IRA's single transposition messages. While he could find numerical keys for the ciphertexts, he still needed to determine the original key phrases. This would provide for a better understanding of the IRA communications procedures, such as whether certain books were used, as well as helping in solving other ciphertexts. The problem of recovering a key phrase from a numerical key is similar to recovering a "hat" in the famous NSA "headline puzzles" [9]. In his book, Jim Gillogly mentions the name of Jude Patterson as an expert in those kinds of puzzles. We contacted her, and she provided useful tips on how "hats" can be solved. Based on this information, we applied a combination of manual and computerized methods, which produced the matching sentence, "`TOSTAYYOUFROMELECTION`". The phrase "`TOSTAYYOUFROMEJECTION`" also matched.

We were not sure if this might really be the correct phrase. We turned for more help to Jim Gillogly and he applied the powerful tool which he developed for *Decoding the IRA*. One day later, he came up with the same and unique solution "`TOSTAYYOUFROMELECTION`" and he also was able to trace its source. This was the last line of the opening of scene II of the "*Merchant of Venice*" from Shakespeare. The last question was: What may Shakespeare/The Merchant of Venice, and Preponderance of Evidence have in common? They seemed totally unrelated at first glance. Further research on Google produced an interesting document named "*Who Wrote Shakespeare? The Preponderance of Evidence*" by Richard F. Whalen. This is a paper discussing the controversy about whether Shakespeare really wrote Shakespeare. With this, we could confirm that we had the final missing element, and that the solution was complete. Klaus Schmeh later confirmed that during the period of time when he created the double columnar challenge, he was studying the Shakespeare controversy.

5 Conclusion

The main goal of the work behind this paper was to develop new methods with the immediate purpose of solving the double columnar challenge from Klaus Schmeh. Therefore, more research is required to assess the suitability of those methods for a general solution of the double transposition cipher. Additionally, the exact limits of those methods need to be systematically evaluated.

Nevertheless, we were able to identify several new general vulnerabilities of this classical cipher. The primary finding is that is possible to find one of the two keys (K_2) independently from the other (K_1). We achieved this with our new fitness function, the *Index of Digraphic Potential*. With this function we can asses the fitness of K_2 without the knowledge of K_1 in a deterministic way. This has the practical effect of almost nullifying the effect of the second transposition. The IDP allows for a highly efficient K_2 *key-phrase dictionary attack*. This dictionary attack is not dependent on the key lengths, and its complexity is only $O(n)$ for a dictionary size of n instead of $O(n^2)$. Thus, any text encrypted using key phrases from books, web sites, or common expressions, may be susceptible to this attack and potentially easily decrypted. Clearly, randomly chosen numerical keys prevent this attack, but they are also harder to memorize.

The IDP also allows for an improved *hill climbing ciphertext-only attack*. This method raises the bar for the current lower limit of key lengths required to achieve security. Until now the longest solvable key length was 13-15. With our new algorithm we can now solve messages encrypted with key lengths – either random or from key phrases – up to a length of 20.

Finally, we identified additional cases of *weak transposition rectangles*. For instance, the case of an almost perfect rectangle should be avoided in addition to those already mentioned in the classical literature.

Even though the identified vulnerabilities are significant, the double transposition still can be secure and its cryptanalysis challenging when it is used with the right parameters. The team from MysteryTwister C3 already published a new double transposition challenge with additional requirements designed to overcome the vulnerabilities exposed in this work [16]. Another possible follow-up from this work could be to try and apply the techniques presented here on historical messages, which are unsolved if such messages can be found in libraries or archives.

Acknowledgements

We would like to thank Klaus Schmeh for publishing the challenge and for bringing to public attention other fascinating unsolved historical ciphers. We would also like to thank Christian Schridde for his insights about how keys driven from keyphrases differ from random keys, as well as Jude Patterson and Jim Gillogly for their assistance in solving the last part of the puzzle. Finally, we would like to thank Olga Kieselmann and our reviewers for their valuable comments on refining the paper.

About the Authors

George Lasry is a computer scientist and a business executive in the high-tech industry in Israel. He studied cryptography with the late Prof. Shimon Even at the Technion Institute of Technology in Haifa, and worked for many years in the development of military and commercial communications systems. His primary interest in cryptographic research is the application of optimization techniques for the cryptanalysis of classical codes.

Nils Kopal is a PHD student with the research group "Applied Information Security" (AIS) at the University of Kassel. He studied applied computer science at the University of Duisburg-Essen in Duisburg with focus on distributed and secure systems. He finished his studies with a masters degree. With the AIS group he is the responsible coordinator for the development of the open source e-learning tool CrypTool 2.0 which illustrates classical and modern cryptographic and cryptanalytic concepts.

Professor Arno Wacker is the head of the research group "Applied Information Security" (AIS) with the University of Kassel in Germany. He is also the technical lead of the open source project CrypTool 2.0 and member of the steering group of MysteryTwister C3. His main research interests are modern security protocols for decentralized distributed systems, e.g. for volunteer computing scenarios. Additionally, he teaches classical and modern cryptology in classes at the University of Kassel and at special workshops for pupils at local schools.

References

[1] Wayne G. Barker. *Cryptanalysis of the Double Transposition Cipher: Includes Problems and Computer Programs.* Aegean Park Press, 1995.

[2] Craig Bauer. *Secret History: The Story of Cryptology*, volume 76. CRC Press, 2013.

[3] Craig Bauer. Unsolved: The World Greatest Codes and Ciphers, (No date). `http://wdjoyner.org/video/bauer/bauer-Unsolved-ciphers.pdf`, [Accessed: December, 4th, 2013].

[4] Mark Davies. N-Grams Data - Corpus of Contemporary American English, Samples, Level 1 (Free), (No date). `http://www.ngrams.info/download_coca.asp`, [Accessed: December, 4th, 2013].

[5] William Frederick Friedman. *Military Cryptanalysis.* US Government Printing Office, 1941.

[6] Solomon Kullback. *General Solution for the Double Transposition Cipher*, volume 84. Aegean Park Pr, 1934.

[7] Otto Leiberich. Vom diplomatischen Code zur Falltürfunktion. *Spektrum der Wissenschaft. Dossier Kryptographie*, 2001:12–18, 1999.

[8] Thomas G. Mahon and James Gillogly. *Decoding the IRA.* Mercier Press Ltd, 2008.

[9] Jude Patterson. The Headline Puzzle, 2013. `https://sites.google.com/site/theheadlinepuzzle/home`, [Accessed: December, 4th, 2013].

[10] Klaus Schmeh. *Codeknacker gegen Codemacher*, volume 2. W3l, 2008.

[11] Klaus Schmeh. Wettrennen der Codeknacker, 2008. `http://www.heise.de/tp/artikel/26/26876/1.html`, [Accessed: December, 4th, 2013].

[12] Klaus Schmeh. *Nicht zu Knacken.* Carl Hanser Verlag GmbH & Co. KG, 2012.

[13] Klaus Schmeh. Top-25 der ungelösten Verschlüsselungen - Platz 5: Die Doppelwürfel-Challenge, 2013. `http://scienceblogs.de/klausis-krypto-kolumne/2013/09/13/top-25-der-ungelosten-verschlusselungen-platz-5-die-doppelwurfel-challenge`, [Accessed: December, 4th, 2013].

[14] Klaus Schmeh. MysteryTwister C3, The Crypto Challenge Contest, Double Column Transposition, (No date). `https://www.mysterytwisterc3.org/en/challenges/level-x/double-column-transposition`, [Accessed: December, 4th, 2013].

[15] Thorsten Brants, Alex Franz. Web 1T 5-gram Version 1, 2006. `http://catalog.ldc.upenn.edu/LDC2006T13`, [Accessed: December, 4th, 2013].

[16] Arno Wacker, Bernhard Esslinger, and Klaus Schmeh. MysteryTwister C3, The Crypto Challenge Contest, Double Columnar Transposition – Reloaded, (No date). `https://www.mysterytwisterc3.org/en/challenges/level-iii/double-column-transposition-reloaded-part-1`, [Accessed: December, 16th, 2013].

[17] Tim Wambach. Kryptanalyse der doppelten Spaltentranspositionschiffre, 2011.