Decrypting Scytale Ciphertexts Using Dynamic Programming

Wilson Tandya – 13519209 [1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
[1] *13519209@std.stei.itb.ac.id*

***Abstract*—Cryptography plays a major part in today's society; every technological aspect of our lives is connected with cryptography. Many of us are negligent about cryptography, even though our private data and information are exchanged daily through many platforms. Although cryptography has changed significantly in the past decades, the fundamentals of it are still similar, therefore this paper will discuss the decryption method of scytale ciphertexts, one of the oldest forms of cryptography using dynamic programming.**

Keywords—Cryptography, scytale, dynamic programming, ciphertexts

I. INTRODUCTION

Cryptography is a technique to achieve confidentiality of messages. The term itself has a meaning in Greek which means secret writing [1]. Cryptography in the early days focused mainly on message secrecy, conversion of messages from a comprehensible form into an incomprehensible one and back again, making it unreadable by interceptors or eavesdroppers without some understanding of the encryption technique.

In this modern era, however, the privacy of individuals and organizations is served by cryptography at a higher level and has expanded beyond secrecy concerns, making sure that information sent is secure in a way that the authorized receiver can access this information, integrity checking, identity authentication, digital signatures, interactive proofs, secure computation, and many more.

Many people around the globe use cryptography daily to protect their private data and information, although most of them do not know that cryptography is involved in it. For instance, social media passwords are not stored in databases as plaintext. Hackers can gain passwords easily if one day the database is breached. One of the many algorithms used for storing passwords is the hash function. RSA is one of the commons among many hash functions, which generates public and private keys to encrypt plaintexts to ciphertexts. With the advancement of technology nowadays, cryptography becomes more and more complex making it harder and harder to decrypt without certain knowledge of the encryption key.

But, with all the advancements that we have seen nowadays, cryptography is still the same fundamentally from the war era. Therefore, this paper will discuss one of the earliest forms of cryptography, which is a scytale. More specifically, decrypting scytale ciphertexts using dynamic programming.

II. BASIC THEORY

A. Cryptography

Cryptography or cryptology originated from Ancient Greek *kryptós* which means "hidden, secret" and *graphein* which translates to "to write" or "study". Combined, *kryptós graphein* means the practice and study of techniques for secure communication in the presence of third parties called adversaries [2]. Taken from the Merriam-Webster dictionary, cryptography means the enciphering and deciphering of messages in secret code or cipher.

Based on the historical roots, cryptography can be considered an old technique that is still being established. The main traditional cipher types are called transposition ciphers, which is essentially rearranging the order of letters in a message, and substitution ciphers, which methodically replace letters or groups of letters with other letters or groups of letters. Uncomplicated forms of either have never offered much confidentiality from resourceful opponents.

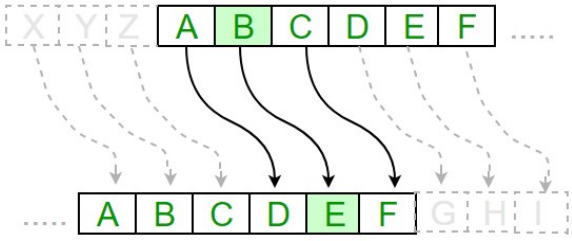

Figure 1. Caesar Cipher (Source: geeksforgeeks.org)

One of the earliest uses of the transposition cipher is a scytale, claimed to have been used by the Spartan military and early utilization of substitution cipher was the Caesar cipher, in which each letter in the plaintext was replaced by a letter by some fixed number of positions further down the alphabet set by the sender.

Before the early 20th century, cryptography was mainly involved with linguistic and lexicographic patterns, however, the emphasis has shifted and cryptography now makes extensive use of mathematics, such as number theory, computational complexity, statistics, combinatorics, abstract algebra, and many others. From the vast development of digital computers and electronics in cryptanalysis, more complex ciphers are made possible. Computers allowed for the encryption of any sort of information representable in any binary format, unlike classical ciphers which only encrypted written language messages or texts.

B. Cryptanalysis

Cryptanalysis is the study of analyzing information systems in order to examine the hidden aspects of the systems. Cryptanalysis is utilized to breach cryptographic security systems and obtain access to the contents of the encrypted messages, even when the cryptographic key is unidentified. The goal of cryptanalysts is to get as much data as possible about the original, unencrypted information or known as plaintext [3].

Attacks or breaches can be categorized based on what type of information the attacker has available. There are *ciphertext-only*, which the cryptanalyst has access only to a compilation of ciphertexts to work with, *known-plaintext*, which the person has a set of ciphertexts to which they know the corresponding original message/plaintext, *chosen-plaintext*, which the attacker can get the ciphertexts corresponding to an arbitrary set of plaintexts of their own choosing, *adaptive chosen-plaintext*, which is similar to *chosen-plaintext*, but the attacker can choose subsequent plaintexts based on information learned from the preceding encryptions, *related-key attack,* which is also similar to *chosen-plaintext,* except the attacker can get ciphertexts encrypted under two different keys, the keys are unidentified, but the connection between them is identified [4].

Attacks can also be characterized based on the resources they need, such as, time, the amount of computation steps it required, memory, the storage required to execute the breach, data, the quantity, and type of plaintexts and ciphertexts needed for a certain tactic. Even though the word cryptanalysis itself is relatively new, the methods for breaking codes and ciphers are much ancient [5].

C. Ciphertext

Ciphertext is the result of encryption done on plaintexts using a specific algorithm, called cipher [6]. Ciphertext is also recognized as encrypted information because it has a form of the original message that is unreadable without the proper algorithm used to decrypt it.

Figure 2. Enigma I (Source: sothebys.com)

There are two types of ciphers, the first one is historical ciphers, which include polyalphabetic substitution cipher, a substitution cipher using multiple substitution alphabets, for instance, Vigenère cipher and Enigma machine, polygraphic substitution cipher, a substitution cipher using the sequence of two or more letters rather than just one, for example, Playfair cipher, transposition cipher, the ciphertext is a permutation of the plaintext, such as, rail fence cipher. The second type is modern cipher, which are more advanced than the classical one and intended to endure a broad variety of attacks, which include private-key cryptography and public-key cryptography.

D. Scytale

The word scytale comes from ancient Greek *skutálē* which means a baton or cylinder. A scytale is an instrument used to operate a transposition cipher, consisting of a cylinder with a strip of parchment wound around it on which is inscribed a message. The receiver uses a cylinder of the same size in diameter on which the parchment is wrapped to read the encrypted message.

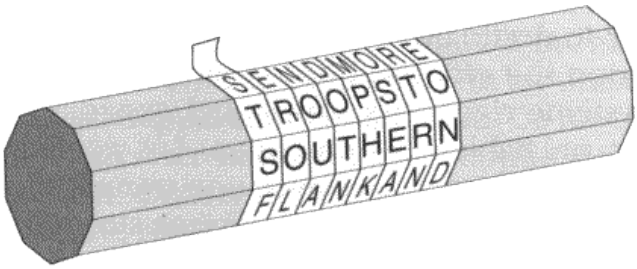

Figure 3. Scytale (Source: asecuritysite.com)

The scytale is operated by making two cylinders the same size in length and thickness so that each corresponds to the others in its dimension, and the sender keeps one for themselves and gives the other one to their messenger. These cylinders made from wood are what they call a scytale. After both the messenger and the receiver have the scytale, whenever they wish to send some secret message, they would make a long and narrow scroll made from parchment and wrap it around their scytale, leaving no space, covering all the surface

area. Then, they write the secret message as it wraps on the scytale, and when they have finished writing their message, they take the parchment off and send it to the receiver.

There is also an alternative possibility from the use of scytale, that it was used for message authentication rather than encryption [7]. The sender would give some message that would be read when wrapped around the same scytale dimension, confirming that the message is original, therefore making it difficult for enemy spies to give false messages to the receiver.

E. Dynamic Programming

Dynamic programming is mostly an optimization method on a recursion. We can optimize a solution that has a recursive aspect to it with dynamic programming. Dynamic programming was developed by Richard Bellman in the 1950s.

Dynamic programming works by simply saving the results of the subproblems so that redundant computations are not needed in future computation. Dynamic programming reduces time complexities from exponential to polynomial.

Characteristics of problems that can be solved with dynamic programming, first, problems can be divided into multiple stages, and each stage requires one decision to be made, second, each stage consists of states, which are input possibility on the stage itself.

Dynamic programming offers two methods to solve a problem, firstly, top-down with memoization, in this approach, solving the problem starts by recursively finding the solution to smaller sub-problems. Whenever we solve a sub-problem, we save its result so that making the same computation is not needed, and that is the reason why it is called memoization. Secondly, bottom-up with tabulation, this method is the opposite of the top-down approach, in this method, the problem is typically solved by filling up an N-dimensional table. Based on the results in the table, the solution to the original problem is then computed. The tabulation method, unlike memoization, avoids using recursive patterns.

III. USING DYNAMIC PROGRAMMING IN DECRYPTING THE SCYTALE

Different kinds of algorithms can be used to decrypt a scytale, one of the fastest ways is using dynamic programming. Each scytale ciphertext has several turns of the band, and that number is unknown to the attacker.

To design a program to decrypt the scytale, we must get the number of rolls of the band, and separating each word to create a logical sentence that a computer or human can read. Those can be done by using a brute force algorithm, but it would be very slow. Therefore, using dynamic programming can reduce the computation time significantly. The part that can be solved recursively is the "word-break" problem, which makes spaceless text, into readable texts. It is done by using a lookup table that stores the solutions to the sub-problems, marking 1 for true, 0 for false, and -1 for the sub-problems that have not been computed. If the input strings can be segmented, all the words contained in the spaceless string must be in the local dictionary, then the program will print the original messages that are readable.

First, the program will ask for an input of the scytale ciphertext, then the program will do a loop to test the number of turns of the band, and each loop will go through the "word-break" function that will first create a lookup list filled with -1 and has a length of the scytale ciphertext. Then, it would check if the prefix of the ciphertext exists in the local dictionary, continue by marking the lookup list with either 0 or 1. Lastly, the program will return the original message if the ciphertext entered is solvable with the local dictionary given.

IV. IMPLEMENTATION

The method that will be used in the implementation is top-down with memoization. The lookup table being the memoization part. The word-breaks function that utilizes dynamic programming used in decrypting scytale ciphertext is as follow:

```
def wordBreak(dictionary, str, lookup,
result):
 n = len(str)
 if n == 0:
    return True


 if lookup[n] == -1:
    lookup[n] = 0
    for i in range(1, n + 1):
       prefix = str[:i]
       if (prefix in dictionary and
       wordBreak(dictionary, str[i:], lookup,
       result)):
           lookup[n] = 1
           result.insert(0, prefix)
           return True
 return lookup[n] == 1
```

Figure 4. Word-breaks Function Using Dynamic Programming (Source: Author)

To fully operate the decryption program, the number of turns of the band must be brute-forced, from 1 being the lowest and the length of the input string is the highest. Those are done on the following code:

```
import time


if __name__ == '__main__':
  start = time.time()
  # local dictionary (35 words for this test)
  dictionary = [ "one", "two", "three",
```

```
"four", "five", "six", "seven", "eight",
"nine", "ten", "me", "you", "we", "to",
"with", "please", "help", "write", "read",
"think", "apples", "bananas", "melons",
"mangoes", "mushrooms", "strawberries",
"can", "give", "and", "not", "do", "want",
"build", "together", "houses"]

# sample scytale ciphertext
input = "dbtheouoevyigleolepnudtmmwhheaaoeg
nnurigtsavoteneeosdss"

now = 0
for i in range (1, len(input)-1):
   str = ""
   str += (input[0])
   for j in range (len(input)-1):
      now += i
      if (now > len(input)-1):
         now = now % (len(input)-1)
      str+=(input[now])
   str = str.replace('_','')
   lookup = [-1] * (len(str) + 1)
   result = []
   if wordBreak(dictionary, str, lookup,
   result):
      print("The decrypted message is:")
      for words in result:
         print(words, end=" ")
      print()
      break
   else:
      if (i == len(input)-2):
         print("Scytale doesn't have any
         secret message")
end = time.time()
print(round(end-start, 8), "seconds are
needed to find the solution above")
```

Figure 5. Main Program Code for Dynamic Programming (Source: Author)

The program above with the sample scytale ciphertext *"dbtheouoevyigleolepnudtmmwhheaaoegnnurigtsavoteneeosdss"* gives an output of :

```
The decrypted message is:
do you want to build houses together and help me give seven mangoes
0.00500607 seconds are needed to find the solution above
```

Figure 6. Main Program Output (Source: Author)

The word-break function can also be made with brute force algorithm, backtracking algorithm, and normal recursive algorithm (unoptimized). For comparison purpose, the code for unoptimized normal recursive approach is as follow:

```
def wordBreak(dict, str, out=""):
  if not str:
    print(out[1:])
    return

  for i in range(1, len(str) + 1):
    prefix = str[:i]
    if prefix in dict:
        wordBreak(dict, str[i:], out + " " +
        prefix)
```

Figure 7. Word-breaks Function Using Normal Recursive Approach (Source: Author)

Running it with the main program that tries the number of turns of the band as the following:

```
import time
if __name__ == '__main__':
  start = time.time()
  # local dictionary (35 words for this test)
  dictionary = [ "one", "two", "three",
  "four", "five", "six", "seven", "eight",
  "nine", "ten", "me", "you", "we", "to",
  "with", "please", "help", "write", "read",
  "think", "apples", "bananas", "melons",
  "mangoes", "mushrooms", "strawberries",
  "can", "give", "and", "not", "do", "want",
  "build", "together", "houses"]

  # sample scytale ciphertext
  input = "dbtheouoevyigleolepnudtmmwhheaaoeg
  nnurigtsavoteneeosdss"

  now = 0
  print("The decrypted message is:")
  for i in range (1, len(input)-1):
```

```
    str = ""
    str += (input[0])
    for j in range (len(input)-1):
       now += i
       if (now > len(input)-1):
          now = now % (len(input)-1)
       str+=(input[now])
    str = str.replace('_','')
    wordbreak(dictionary, str)
end = time.time()
print(round(end-start, 8), "seconds are
needed to find the solution above")
```

Figure 8. Main Program Code for Normal Recursive Approach (Source: Author)

With the same sample ciphertext of *"dbtheouoevyigleolepnudtmmwhheaaoegnnurigtsavoteneeosdss"* will give an output of:

```
The decrypted message is:
do you want to build houses together and help me give seven mangoes
0.01300716 seconds are needed to find the solution above
```

Figure 9. Output Using Normal Recursive Approach (Source: Author)

V. ADDITIONAL TESTS

Using the same 35 words dictionary, additional tests using various scytale ciphertexts are presented below. Each length and number of turns are tested 3 times.

Length	Number of Turns	Time [DP] (seconds)	Time [NR] (seconds)
40	10	0.00399876	0.00689483
40	10	0.00317885	0.00723572
40	10	0.00397455	0.00691886
50	10	0.00400205	0.01200146
50	10	0.00389536	0.01286424
50	10	0.00300685	0.01423521
60	15	0.00400233	0.02500892
60	15	0.00393673	0.02611323
60	15	0.00410055	0.02561353
75	15	0.00499368	0.02700615
75	15	0.00480665	0.02543054
75	15	0.00497325	0.02816433
100	20	0.00700259	0.04401731
100	20	0.00687933	0.04401088
100	20	0.00600576	0.04200792
280	20	0.25606821	0.42823291
280	20	0.26663038	0.43708312
280	20	0.26708362	0.44123233

Figure 10. Additional Data Tests (Source: Author)

Length of ciphertexts are stated in characters, number of turns represents the number of turns of the scytale band with the found solution, DP stands for dynamic programming and NR stands for normal recursive.

VI. ANALYSIS

As seen in Figure 4, the use of dynamic programming in the word-breaks function reduces the amount of computation needed to obtain the plaintext result. The time complexity of the word-breaks algorithm is O(n*s), where n is the length of the input string, and s is the length of the largest string in the local dictionary, or can be simplified as $O(n^2)$, assuming the length of the input string is always longer than the length of the longest string in the local dictionary. Combined with the brute force algorithm to determine the number of loops of the band, the overall time complexity to decrypt a scytale ciphertext is $O(n^3)$.

Compared with the word-breaks algorithm seen in Figure 7, the function works by recursively calling the wordBreak function as much as the number of loops of bands. The time complexity of the word-breaks algorithm is $O(2^n *s)$, where n is the length of the input string, and s is the length of the largest string in the local dictionary, or can be simplified as $O(2^n *n)$, assuming the length of the input string is always longer than the length of the longest string in the local dictionary. Combined with the brute force from the main program, the time complexity of the overall program is $O(2^n *n^2)$.

This shows that the use of dynamic programming in decrypting scytale ciphertexts is very effective compared to the normal recursive approach, let alone full brute force algorithm or backtracking which has a time complexity of $O(n^n)$.

The time needed to find the solution for the scytale ciphertexts is 0.00500607 seconds using dynamic programming and 0.01300716 seconds using a normal recursive approach. The test was done by using a 35 words local dictionary, with the longest word having 12 characters, and the ciphertext length is 55 characters long. Seen from the time complexity, the longer the ciphertext is, the larger the time difference it will have.

As seen in Figure 10, the time in seconds required to obtain a solution is much efficient using dynamic programming compared to the normal recursive approach.

VII. CONCLUSION

The scytale ciphertexts can be decrypted with many algorithms. The program that is made with dynamic programming top-down with memoization method turns out to be faster (tested with a 35 words local dictionary, with the longest word having 12 characters, and the ciphertext varies in length from 40 to 280 characters long), hence being more effective compared to the normal recursive algorithm which is unoptimized. The time complexity of the overall decryption scytale ciphertexts using dynamic programming is $O(n^3)$.

VIDEO LINK AT YOUTUBE

Video about this paper can be accessed in the following link: https://youtu.be/1YdrS_Z0anA

PROJECT LINK AT GITHUB

The code that is included in this paper can be accessed in https://github.com/WilsonTandya/DecryptScytale

ACKNOWLEDGMENT

The author would like to thank God Almighty for the completion of this paper and Dr. Ir. Rinaldi Munir, MT. as the author's lecturer in IF2211 Algorithm Strategies course K-04. The author would also thank fellow friends and colleagues who have supported me in the journey of writing this paper.

REFERENCES

[1] Rivest, Ronald L. (1990). "Cryptography". In J. Van Leeuwen (ed.). Handbook of Theoretical Computer Science. 1. Elsevier.

[2] Liddell, Henry George; Scott, Robert; Jones, Henry Stuart; McKenzie, Roderick (1984). A Greek-English Lexicon. Oxford University Press.

[3] Dooley, John F. (2018). History of Cryptography and Cryptanalysis: Codes, Ciphers, and Their Algorithms. History of Computing. Cham: Springer International Publishing. doi:10.1007/978-3-319-90443-6. ISBN 978-3-319-90442-9. S2CID 18050046.

[4] Shannon, Claude (4 October 1949). "Communication Theory of Secrecy Systems". Bell System Technical Journal. 28 (4): 662. doi:10.1002/j.1538-7305.1949.tb00928.x. Retrieved 20 June 2014.

[5] Kahn, David (1996). The Codebreakers: The Comprehensive History of Secret Communication from Ancient Times to the Internet. Simon and Schuster. ISBN 9781439103555.

[6] Berti, Hansche, Hare (2003). Official (ISC)² Guide to the CISSP Exam. Auerbach Publications. pp. 379. ISBN 0-8493-1707-X.

[7] Russell, Frank (1999). Information Gathering in Classical Greece. U. Michigan Press. p. 117. ISBN 0-472-11064-0.

STATEMENT

I hereby declare that the paper I wrote is my writing, not an adaptation, or a translation of someone else's paper, and not plagiarism.

Tangerang, May 11th 2021

Wilson Tandya - 13519209