Team 9: Ressa Reneth Sarreal & Dominic Schellin
CMSC426 - Dr. Marron
Lab 2: Modern Cryptography Lab
Due: 04/10/2018

---

1 Introduction

This report details the work done for the Modern Cryptography Lab. The objective of the lab was to obtain experience with breaking a loosely-made encryption algorithm, and understanding how encryption/decryption schemes may be implemented. To run the code, make sure Python3 is installed, and pycrypto libraries are also included on the system. To execute the decryption attempt, type the following with the appropriate parameters into the command line:

```
python lab2.py <encrypted_message>.enc <desired_output_file>.txt
```

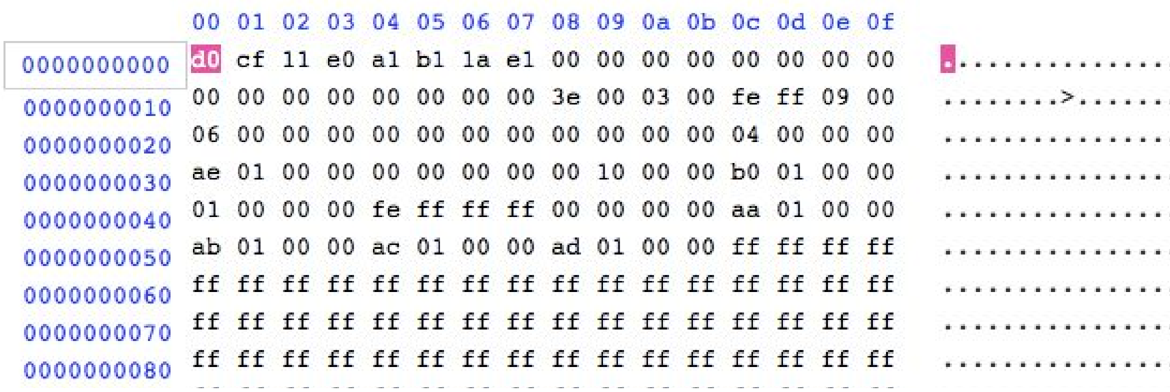
2 Problem 1: What Type of Encryption?

After looking at the HexEditor view of msg1.enc, which is partially shown in Figure 1, several characteristics of the encryption can be determined.

```
            00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f
0000000000  bc cd 40 8b b2 36 aa c3 19 1f 18 8e 2f b7 9d 47   ..@..6......./..G
0000000010  19 1f 18 8e 2f b7 9d 47 ff e2 54 69 87 1e 09 d4   ..../..G..Ti....
0000000020  9c b4 5d 66 0b 2b c9 b5 92 89 8d 24 53 01 0a 81   ..]f.+.....$S...
0000000030  44 cf 83 2e fd f8 89 b5 2f 31 81 37 0e c2 bb d3   D......./1.7....
0000000040  46 43 b4 e8 16 12 a3 d3 19 1f 18 8e 2f b7 9d 47   FC........../..G
0000000050  d1 be cc 39 df 6d d1 e6 d1 be cc 39 df 6d d1 e6   ...9.m.....9.m..
0000000060  d1 be cc 39 df 6d d1 e6 d1 be cc 39 df 6d d1 e6   ...9.m.....9.m..
0000000070  d1 be cc 39 df 6d d1 e6 d1 be cc 39 df 6d d1 e6   ...9.m.....9.m..
0000000080  d1 be cc 39 df 6d d1 e6 d1 be cc 39 df 6d d1 e6   ...9.m.....9.m..
```

Figure 1: HexEditor View of msg1.enc

Team 9: Ressa Reneth Sarreal & Dominic Schellin
CMSC426 - Dr. Marron
Lab 2: Modern Cryptography Lab
Due: 04/10/2018



	00	01	02	03	04	05	06	07	08	09	0a	0b	0c	0d	0e	0f	
0000000000	d0	cf	11	e0	a1	b1	1a	e1	00	00	00	00	00	00	00	00	....
0000000010	00	00	00	00	00	00	00	00	3e	00	03	00	fe	ff	09	00	........>.......
0000000020	06	00	00	00	00	00	00	00	00	00	00	00	04	00	00	00	....
0000000030	ae	01	00	00	00	00	00	00	10	00	00	b0	01	00	00		....
0000000040	01	00	00	00	fe	ff	ff	ff	00	00	00	00	aa	01	00	00	....
0000000050	ab	01	00	00	ac	01	00	00	ad	01	00	00	ff	ff	ff	ff	....
0000000060	ff	ff	ff	ff	ff	ff	ff	ff	ff	ff	ff	ff	ff	ff	ff	ff	....
0000000070	ff	ff	ff	ff	ff	ff	ff	ff	ff	ff	ff	ff	ff	ff	ff	ff	....
0000000080	ff	ff	ff	ff	ff	ff	ff	ff	ff	ff	ff	ff	ff	ff	ff	ff	....

Figure 2: HexEditor View of a .doc File

A. **Examine the file msg1.enc using a hex viewing tool. Based on material presented in the lectures, you should be able to make an educated guess as to what algorithm is being used to encrypt the data.**
The encryption algorithm looks like DES in Electronic Codebook Mode because there is highly structured output in msg1.enc that repeats every 64 bits in many parts of the file. That corresponds to the block size of DES, and ECB preserves the general structure of the input.

B. **Continuing with your analysis of msg1.enc, try to determine the format of the underlying plaintext. Find a .doc file, a .docx file, a pdf file, and some image files; use a hex viewing tool to determine if any of these types of file have patterns matching what you observe in the ciphertext.**
When msg1.enc is compared to a sample .doc file, partially shown in Figure 2, common patterns are apparent. For example, a repeating pattern is evident in both files from 0x50 to 0x1F0 in both msg1.enc and the generic .doc file. The msg1.enc file was also compared to a test .pdf file and a test .docx file and there was no structural similarity.

Team 9: Ressa Reneth Sarreal & Dominic Schellin
CMSC426 - Dr. Marron
Lab 2: Modern Cryptography Lab
Due: 04/10/2018

---

3 Problem 2: Something Has Changed…

A. Examine the file genkey.py. The comment suggests that there has been a recent change to the encryption program. Analyze the key generation scheme.

It is evident that an LCG is being used as the key generator because the code follows the exact formula that creates one. The generator is seeded by the last three bytes of the system time provided by the python time() function, and the large constants in this formula are hardcoded, making it much easier to generate valid keys for attacking the encrypted files that use this program. The only unknown is the time at which the key for each encrypted file was generated. LCGs are not suitable for encryption because their output is both linear and dependent on previous outputs. This makes it simple to set up a system of equations to determine the exact formula used.

B. Focus on the seeding of the key generation algorithm. You will need to read about the Python time module and time() function to fully understand how the seeding works (see http://docs.python.org/2/library/time.html) Describe in words how the seed is being computed. Is this a good method? Why or why not?

The initial seed value for the LCD is dependent on the last three bytes of the system time returned by the python time() function. This is a poor method of picking seeds because it follows a very predictable pattern. It thus becomes much more likely that an attacker could brute force valid keys by making educated guesses about the times those keys were generated. Even if no timing information is available, the seed is a pitiful 24 bits long; every combination can be attempted in a reasonable amount of time.

C. **Consider all you know about the files msg2.enc and msg3.enc. How can you use this information, along with what you have learned from genkey.py, to attack the encrypted messages?**

   The only unknown needed to recreate the keys used to encrypt msg2.enc and msg3.enc is the exact seed value given when each key was generated. The posted metadata provides the times at which those files were created. It is therefore likely that the system time used to seed the key generator for each file is slightly less than those posted times, given that the python time() function returns seconds since January 1, 1970. To attack these files, keys are generated with the algorithm used in genkey.py. The seed values start at the value time() would return at the posted file creation times (discounting the most significant byte) and decrement each iteration. The decryption algorithm used in combination with these keys is single DES in ECB mode. The first eight bytes of the resulting output file is compared to the file signature of a regular .doc file; if they match, the document is considered successfully decrypted.

D. **How could Dr. Nefario improve the encryption software?**

   First, he should use a stronger encryption algorithm such as AES. DES is weak because the key is too short. Additionally, instead of Electronic Codebook mode, the encryption should be employed in Cipher Block Chaining mode to hide the structure of the data that is being encrypted. A stronger key generator is recommended as well, such as the B.B.S. PRNG.

4 Problem 3: Implement the Attack

Implement the attack you outlined in (2.c) and recover the plaintext for msg2.enc and msg3.enc. Turn in the decrypted messages and the code you used to recover the plaintext.

See Appendix A, B, and C for code and plaintext.

The following procedure was implemented:

I. Use the known file creation times in the posted metadata to generate the first attempted seed value.
II. Generate a corresponding key with the provided LCG algorithm.
III. Read in the first 8 bytes of the encrypted message (one DES block).
IV. Run the DES decryption algorithm on that single block.
V. If the resulting output matches the known file signature of a .doc file, decrypt the full message and write it to an output file. Otherwise, decrement the seed value and return to step II.

5 Conclusion & Work Distribution

After successfully completing the lab, the team has a greater appreciation for the consequences of an inappropriate application of the block cipher in Electronic Codebook mode, as well as the importance of strong key generation algorithms. Both members contributed substantially towards completing the lab. We discussed topics and developed solutions during in-person meetings, and collectively contributed to the report.

6 APPENDIX A | LAB2.PY CODE

```
# lab2.py
# find a key for msg2.enc & msg3.enc and decrypt
# using DES in ECB mode w/ LCG key generation

from Crypto.Cipher import DES
import time
import sys
import os.path


def guessSeed():
   # int(time()) produces 4 bytes of time in seconds from 1970
   # approximation from today's time to the date we saw that the
   # files were generated (March 20th)
   days_ago = 12
   hours_per_day = 24
   minutes_per_hour = 60
   seconds_per_minute = 60
   seconds_ago = days_ago * hours_per_day * minutes_per_hour * seconds_per_minute
   time_today = time.time()
   time_difference = time_today - seconds_ago
   seed_guess = int(time_difference) & 0xffffff # only need 3 bytes

   return seed_guess


# GIVEN FUNCTION : New key generation function [Dr. N, 10/31/14]
def genkey(keylen, seed):
   key = ''
   modulus = pow(2,24)
   a = 1140671485
   b = 12820163
   for i in range(keylen):
       seed = (a*seed + b) % modulus
       key = key + chr(seed >> 16)

   return key


def main():
   # command line format is "python lab2.py [encrypted message file] [output_file].txt
   if len(sys.argv) == 3:
       input_file = sys.argv[1]
       output_file = sys.argv[2]
```

```
    # check if valid command line arguments
    else:
        print("ERROR: Invalid commandline input. Please try again.")
        return

    # check if valid input file
    if not os.path.exists(input_file):
        print("ERROR: Cannot open specified input file.")
        return

    # obtain a string-form of the function to pass into exec()
    input_file_string = open(input_file, 'rb')
    ciphertext = input_file_string.read(8)

    doc_format_str = '\xD0\xCF\x11\xE0\xA1\xB1\x1A\xE1'
    not_match = True
    seed_guess = guessSeed()
    key_len = 8  # for DES

    while not_match:
        # try several keys and decipher the message with each and compare
        # to the expected format of a .doc file
        key_guess = genkey(key_len, seed_guess)
        obj = DES.new(key_guess, DES.MODE_ECB)
        decipher_attempt = obj.decrypt(ciphertext)
        print("attempting to decipher ... ")
        print(" ... ")
        print("")
        # if the decipher attempt matches the correct .doc format string
        # then break the loop
        if decipher_attempt == doc_format_str:
            not_match = False

        if seed_guess >= 1:
            seed_guess = seed_guess - 1

    # decrypt the full message
    full_msg = input_file_string.read()
    plaintext = obj.decrypt(full_msg)

    # write to output file specified in commandline arguments
    with open(output_file, 'a') as the_file:
        the_file.write(plaintext)
    return plaintext

main()
```

---

7 APPENDIX B | MSG2_OUTPUT.TXT

; þÿ þÿÿÿ
ÿÿ
ÿÿ
ÿÿ
ÿÿ
ÿÿþÿÿÿ þÿÿÿ

þÿÿÿ
þÿÿ
ÿÿ
ÿÿ
ÿÿ
ÿÿR o o t E n t r y ÿÿÿÿÿÿÿÿÿÿÿÿ
þÿÿÿ ÿÿÿÿÿÿÿÿÿÿÿÿ þÿÿÿ
ÿÿÿÿÿÿÿÿÿÿÿÿ þÿÿÿ ÿÿÿÿÿÿÿÿÿÿÿÿ
þÿÿÿ þÿÿÿþÿÿÿ

þÿÿÿ ! " # þÿÿÿ%
þÿÿ
ÿÿ
ÿÿ
ÿÿÿ þÿ
ÿÿÿÿ À F Microsoft Word-Dokument
MSWordDoc Word.Document.8 ô9²q
[^ ñÿ ^ N o r m a l
1$ *A 3 B* OJ QJ CJ mH sH KH PJ nHtH^J aJ _H9 F þ F H e a d i n g
¤ð ¤x $ OJ QJ CJ PJ ^J aJ . B . T e x t B o d y
¤ ¤x / L i s t ^J @ " "@ C a p t i o n
¤x ¤x $ CJ 6^J aJ]& þ 2& I n d e x $ ^J C ÿÿÿÿ „
†
C C P G T i m e s N e w R o m a n 5 S y m b o l
3& A r i a l i L i b e r a t i o n S e r i f T i m e s N e w R o m a n
G€ T i m e s N e w R o m a n 5€ S i m S u n ?&€
L u c i d a S a n s 3&€ A r i a l ?$€ L u c i d a S a n s B Å h
¢ògE2+‡ 7 = ƒ 7 = '
0 0
€ þÿ à…ŸòùOh«‘ +'³Ù0
p x „ ¨ ´ À Ì Ø
ä
ð ü
éý

Team 9: Ressa Reneth Sarreal & Dominic Schellin
CMSC426 - Dr. Marron
Lab 2: Modern Cryptography Lab
Due: 04/10/2018

Chris Marron 1 @ ¶œ×Ì @ @ €ÈWAÿþÍ@ _y¨Êùÿ
þÿ ÕÍÕœ.“— +,ù®D ÕÍÕœ.“— +,ù®\ éý éý
ì¥M ð¿ 0 †
Caolan80 2 C ÿÿ ÿÿ ÿÿ ˆ ” ” ”
¨ ´ á R À 3 b Ì
À Ù

T o B I R D M A S T E R
F r o m F A L L E N E A G L E

I l o o k f o r w a r d t o r e a d i n g y o u r p l a n . I j u s t h o p e t h o s e p e s k y R e t r i e v e r s d o n ' t f o i l o u r s c h e m e s a g a i n .

P . S . P l e a s e s t o p u s i n g t h e o l d v e r s i o n o f t h e e n c r y p t i o n s o f t w a r e ! T h e n e w v e r s i o n u s e s r a n d o m k e y s a n d i s m u c h m o r e s e c u r e . T h e o l d v e r s i o n j u s t u s e d a h a s h o f t h e T o / F r o m i n f o r m a t i o n .

”

> @ †
ý û ù ÷ õ ó
0 °Ð/ °à=!°n"°n#n$n2P 1h0p 3P (2 0
R o o t E n t r y ÿÿÿÿÿÿÿÿ À F € C o m p O b j
ÿÿÿÿ j O l e
ÿÿÿÿ ÿÿÿÿ 1 T a b l e ÿÿÿÿÿÿÿÿÿÿÿÿ
• S u m m a r y I n f o r m a t i o n (ÿÿÿÿ D W o r d D o c u m e n t ÿÿÿÿÿÿÿÿÿÿÿÿ 2 D o c u m e n t S u m m a r y I n f o r m a t i o n 8 ÿÿÿÿÿÿÿÿÿÿÿÿ $ t
ÿÿÿÿÿÿÿÿÿÿÿÿ þÿÿÿ nSà ›¬Z

8 APPENDIX C | MSG3_OUTPUT.TXT

; þÿ þÿÿÿ
ÿÿ
ÿÿÿ
ÿÿÿ
ÿÿÿ
ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿþÿÿÿ

þÿÿÿ
þÿÿ
ÿÿÿ
ÿÿÿ

Team 9: Ressa Reneth Sarreal & Dominic Schellin
CMSC426 - Dr. Marron
Lab 2: Modern Cryptography Lab
Due: 04/10/2018

ÿÿ
ÿÿÿR o o t E n t r y ÿÿÿÿÿÿÿÿÿÿÿÿ
þÿÿÿ ÿÿÿÿÿÿÿÿÿÿÿÿ þÿÿÿ
ÿÿÿÿÿÿÿÿÿÿÿÿ þÿÿÿ ÿÿÿÿÿÿÿÿÿÿÿÿ
þÿÿÿ þÿÿÿþÿÿÿ

þÿÿÿ ! " # $ þÿÿÿ& ' ( ) * + , - . / 0 1 2 3 4 5 6 7 8 9 :
; < = > ? @ A B C D E F G H I J K L M N O P Q R S T U V W X Y Z [\ ]
þÿÿÿ_
þÿÿ
ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ þÿ
ÿÿÿÿ À F Microsoft Word-Dokument
MSWordDoc Word.Document.8 ô9²q
[^ ñÿ ^ N o r m a l
1$ *A 3 B* OJ QJ CJ mH sH KH PJ nHtH^J aJ _H9 B þòÿñ B A b s a t z - S t a n
d a r d s c h r i f t a r t F þ F H e a d i n g
¤ð ¤x $ OJ QJ CJ PJ ^J aJ . B . T e x t B o d y
¤ ¤x /" L i s t ^J @ " 2@ C a p t i o n
¤x ¤x $ CJ 6^J aJ]& þ B& I n d e x $ ^J à ÿÿÿÿ À À à à P G
T i m e s N e w R o m a n 5 S y m b o l 3& A r i a l i
L i b e r a t i o n S e r i f T i m e s N e w R o m a n G€ T i m e s N e w R o m a n 5€
S i m S u n ?&€ L u c i d a S a n s 3&€ A r i a l ?$€
L u c i d a S a n s B Å h Ëògl2+‡ % Û ƒ % Û ‡
0 0
€ þÿ à…ŸòùOh«‘ +'³Ù0
p x „ ¨ ´ À Ì Ø
ä
ð ü
éý
Chris Marron 1 @ €lÍ @ @ IFzÿÍ@ ÀýÔXËùÏ
ì¥M ð¿ 0 À Caolan80 2 à ÿÿ ÿÿ ÿÿ ˆ Ø
Ø Ø ì ø % R
w b
Ù

T o F A L L E N E A G L E
F r o m B I R D M A S T E R

O k a y , o k a y , I ' m u s i n g t h e n e w v e r s i o n n o w . W o u l d y o u p l e a s e e x p l a i n i n d e t a i l w h y t h e n e w v e r s i o n i s b e t t e r ? P l a n s a r e c o m i n g a l o n g , b u t h a v e b e e n s p e n d i n g t o o m u c h t i m e a t h a c k - a - t h o n s !

À
> @ ¾ À ý û ù ÷ õ
0 °Ð/ °à=!°n"°n#n$n2P 1h0p 3P (2 0 þÿ ÕÍÕœ.“— +,ù®D ÕÍÕœ.“— +,ù®\ éý
éý R o o t E n t r y ÿÿÿÿÿÿÿÿ À F C o m p O b j
ÿÿÿÿ j O l e

Team 9: Ressa Reneth Sarreal & Dominic Schellin
CMSC426 - Dr. Marron
Lab 2: Modern Cryptography Lab
Due: 04/10/2018

ÿÿÿÿ ÿÿÿÿ 1 T a b l e ÿÿÿÿÿÿÿÿÿÿÿÿ
Ù S u m m a r y I n f o r m a t i o n (ÿÿÿÿ D W o r d D o c u m e n t
ÿÿÿÿÿÿÿÿÿÿÿÿ % 2 D o c u m e n t S u m m a r y I n f o r m a t i o n 8 ÿÿÿÿÿÿÿÿÿÿÿÿ
^ t ÿÿÿÿÿÿÿÿÿÿÿÿ þÿÿÿ ˜,äáùìV