Parker Newton
March 17, 2016
CSCI 181
Final Project

Cryptanalysis of a Stream Cipher consisting of a Repeated English Keystream

Parker Newton
March 17, 2016
CSCI 181 Applied Cryptography
Final Project

Introduction:

The goal of this project is to cryptanalyze and decrypt to sensible English plaintext ten ciphertext files. We are told only that the ciphertext was encrypted with stream cipher consisting of a repeated sensible English phrase as the keystream. A stream cipher is a symmetric key cryptosystem that implements bitwise XOR-ing (addition modulo 2) of the plaintext and a keystream to encrypt the plaintext to ciphertext. The keystream should be the length of the plaintext, and can be generated in a variety of ways. Modern stream ciphers typically use a pseudo-random bit generator algorithm, like RC4, to produce a pseudo-random bit string of the same length as the plaintext. However, in this case, we are told that the keystream is the repetition of a sensible English phrase. By applying several cryptanalysis methods such as the Kasiski Method and a variation of the classical Vigenere cipher analysis, I was able to decrypt to sensible plaintext all ten of the ciphertext files. All software programs used to decrypt the ciphertext files in this project were implemented in C++.

Cryptanalysis Methods:

Consider the case of a stream cipher for which the keystream is the repetition of a sensible English phrase. Essentially, this can be thought of a variation of the classical Vigenere cipher, but with addition modulo 2 instead of addition modulo 26[1]. The Vigenere cipher is a type of monalphabetic shift cipher. The symmetric key is the repetition of an English keyword. To encrypt the plaintext with a Vigenere cipher, cycle through each character in the keyword, encode it to a value between 0 ('A') and 25 ('Z'), and add modulo 26 that encoded value to the encoded value of the corresponding plaintext character. Most attacks on Vigenere ciphers consist of two steps. First, find the length of the keyword. Second, determine the keyword.

For finding the length of the keyword, we can implement a cryptanalysis method known as the Kasiski Method. The Kasiski Method scans the ciphertext for repeated n-graphs[2]. Once a pair of two repeated n-graphs are found at positions m and n in the ciphertext, respectively, we calculated the distance m - n between them. The Kasiski Method states that if there exists a divisor d | (m - n), then d could be the length of the keyword, and hence the n-graphs at positions m and n were encrypted the same way. For eight of the ten ciphertexts, I found it sufficient to set n = 3 and look for repeated trigraphs. However, the remaining two ciphertexts' lengths were so small that no repeated trigraphs were found. So, I proceeded to look for repeated digraphs and letters for those two ciphertexts. After a computer program I wrote generated a list of distances between the repeated n-graphs, I examined the results, found the greatest common divisor (gcd) of several distances, and conjectured a keyword length. Below is an excerpt of the source code for my program which implements the Kasiski Method. The procedure accepts as parameters an array of encoded ciphertext values (`ct[]`), the length of the ciphertext (`ct_len`), the value of n for the repeated n-graphs we are looking for (`N`), and a pointer to the file to which we will write the results. For every repeated n-graph that is found, we write to the output file the distance between them.

---

[1] Addition modulo 26 is common for a classical monalphabetic shift cipher based on the English alphabet.

[2] An n-graph is a sequence of n characters that appear together in plaintext or ciphertext.

```
void kasiski_method(unsigned int ct[], unsigned int ct_len, unsigned int N,
FILE *out)
{
	unsigned int buf[N], s[N], i;
	bool found;
	for(i = 0; i < ct_len - N; i++) {
		for(unsigned int j = 0; j < N; j++) {
			buf[j] = ct[i + j];
		}
		for(unsigned int j = i + N; j < ct_len - N; j++) {
			found = true;
			for(unsigned int k = 0; k < N; k++) {
				if(ct[j + k] != buf[k]) {
					found = false;
					break;
				}
			}
			if(found) {
				fprintf(out, "\n trigraph: %u %u %u \t", buf[0],
	buf[1], buf[2]);
				for(unsigned int k = 0; k < N; k++) {
					s[k] = buf[k];
					unsigned int buf5[5];
					for(unsigned int l = 0; l < 5; l++) {
						buf5[5 - (l + 1)] = s[k] % 2;
						s[k] = s[k] / 2;
					}
					for(unsigned int l = 0; l < 5; l++)
						fprintf(out, "%d", buf5[l]);
				}
				fprintf(out, "\t m - n: %u\n", j - i);
			}
		}
	}
}
```

Figure 1 below shows the sample output for the ciphertext file `C26.TXT`.

```
trigraph: 9 21 22     010011010110110 m - n: 245

trigraph: 16 27 2     100001101100010 m - n: 80

trigraph: 24 25 0     110001100100000 m - n: 50

trigraph: 27 12 21    110110110010101 m - n: 350

trigraph: 12 21 21    011001010110101 m - n: 350

trigraph: 21 21 11    101011010101011 m - n: 350

trigraph: 21 11 1     101010101100001 m - n: 350

trigraph: 1 22 27     000011011011011 m - n: 242

trigraph: 27 20 31    110111010011111 m - n: 35

trigraph: 12 30 20    011001111010100 m - n: 180

trigraph: 0 21 28     000001010111100 m - n: 150

trigraph: 20 7 26     101000011111010 m - n: 5
```

Figure 1: Sample output of Kasiski Method for ciphertext file `C26.TXT`.

By gcd-ing these values, we see that the most frequently-occurring gcd is 10. So, we conjecture that the length of the keyword is 10.

After finding the length of the keyword, the second step of this cryptanalysis method is to determine the keyword. We will design a method that is a variation of the cryptanalysis of the Vigenere cipher and exploits the regularity of the English language. The classical cryptanalysis of the Vigenere cipher states that, starting from the beginning of the ciphertext, we partition the ciphertext into blocks of length equal to the length of the keyword. We write each of these blocks in a row, and append each subsequent row below the previous. Now, we know that the characters in each column were encrypted with the same corresponding character of the keyword. We will hit each character in the same column with every character from the English alphabet. Consider all of the ciphertext characters in column i. For each English character in the range of our encoding scheme ([0, 25]), if the addition modulo 2 of that character with all of the ciphertext characters in column i produces another English character in the range of our encoding scheme, then we will consider that character as a possibility for character i of the keyword. But, how do we assign priority to break ties between multiple characters that could be a character in the keyword? We will assign higher priority to the characters that are more frequently occurring in the English language. For example, if both ‘e’ and ‘z’ are possibilities for keyword character i, then keyword character i is more likely to be ‘e’ than ‘z’, because ‘e’ occurs more frequently in the English language than does ‘z’. Below is a procedure which implements this method.

```
void determine_keyword(unsigned int ct[], unsigned int ct_length, unsigned
int key_length, FILE *out) {
    unsigned int size = ct_length % key_length ? (ct_length / key_length) + 1
: ct_length / key_length;
    int a[size][key_length];
    for(unsigned int i = 0; i < size; i++) {
        for(unsigned int j = 0; j < key_length; j++) {
            a[i][j] = -1;
        }
    }
    for(unsigned int i = 0; i < ct_length; i++) {
        a[i / key_length][i % key_length] = ct[i];
    }
    unsigned int freq[26] = {'e', 't', 'a', 'o', 'i', 'n', 's', 'r', 'h',
'l', 'd', 'c', 'u', 'm', 'f', 'p', 'g', 'w', 'y', 'b', 'v', 'k', 'x', 'j',
'q', 'z'};
    char results[key_length][26];
    for(unsigned int i = 0; i < key_length; i++) {
        for(unsigned int j = 0; j < 26; j++) {
            results[i][j] = 26;
        }
    }
    unsigned int x, y;
    for(unsigned int i = 0; i < key_length; i++) {
        for(unsigned int j = 0; j < 26; j++) {
            y = 1;
            for(unsigned int k = 0; k < size; k++) {
                if(a[k][i] != -1) {
                    x = a[k][i] ^ j;
                    if(!(y = x < 26)) break;
                }
            }
            if(y) {
                for(unsigned int k = 0; k < 26; k++) {
                    if(freq[k] == j + 97) {
                        results[i][k] = j + 97;
                    }
                }
            }
        }
    }
    for(unsigned int i = 0; i < key_length; i++) {
        fprintf(out, ”\n k_%u \t", i);
        for(unsigned int j = 0; j < 26; j++) {
            if(results[i][j] != 26) {
                fprintf(out, ”%c \t", results[i][j]);
            }
        }
        fprintf(out, ”\n");
    }
}
```

Figure 2 below shows the output of this routine. Each character in a row is a possible character in that position of the keyword. The characters within a single row are ordered from left to right in decreasing frequency. We assign higher priority to characters that occur with a higher frequency. After a creative analysis (and some time), we determine that the keyword is “betsywetsy”.

k_0	a	b						
k_1	e	f						
k_2	t	s	r	u	w	v	x	q
k_3	t	s	r	u	w	v	x	q
k_4	y	z						
k_5	t	s	r	u	w	v	x	q
k_6	e	f						
k_7	t	s						
k_8	t	s	r	u	w	v	x	q
k_9	y	z						

Figure 2: Possible character values for keyword of length 10 in `C26.TXT`.

Now that we have determined the keyword, we repeat the keyword for the length of the ciphertext, and thus we have the keystream. Because we are using a stream cipher, to decrypt the ciphertext to plaintext, we implement bitwise XOR-ing (addition modulo 2) of the ciphertext with the keystream to decrypt to plaintext. Below is the plaintext of the ciphertext file `C26.TXT`.

```
i t i s n i g h t n o w n o l o n g e r e v e n i n g b u t f u l l y n i g h
t a s i n b l a c k a s i f n o t p r e c i s e l y d e a d o f e v e n i n g
u s u a l l y h a s t h e a f t e r n o o n h a n g i n g o n i t s c o a t t
a i l s h a s a c t u a l f l e c k s o f d a y l i g h t c l i n g i n g l i
k e l i n t t o i t s l a p e l s b u t n i g h t i s s o l i t a r y a l o o
f u n c o m p r o m i s e d e x t r e m e t h e s a f e m a r g i n s o f t h
e d a y s t i l l f a i n t l y v i s i b l e d u r i n g e v e n t i d e h a
v e b e e n e r a s e d b y n i g h t s d e n s e g u m o b s c u r e d b y i
t s w a s h o f s q u i d s q u i r t i n g s p a j a m a s a u c e a n d t h
e b l u e h o n e y m a n u f a c t u r e d b y m o t h s i s t h e n i g h t
a m a s k o r i s d a y m e r e l y n i g h t s p r i m d i s g u i s e
```

Results:

Below is a list of the ciphertext files and corresponding keywords and plaintexts.

C22.TXT:

Keyword:
`froglegpie`

Plaintext:

```
b y t h e t i m e i a r r i v e d i n l h a s a t h e t i b e t a n c
a p i t a l w a s s t i l l u n c o l o n i z e d e n o u g h t o b e
r i c h i n a l l t h e i n c o n v e n i e n c e s t h a t t h e o v
e r l a n d e r n e e d s t o r e m i n d h i m s e l f t h a t t r v
e l i s c l o s e l y r e l a t e d t o t r a v a i l t h e r e w e r
e t w o s m a l l g u e s t h o u s e s i n t o w n b u t e a c h h a
d o n l y a f e w l u x u r y d o l l a r r o o m s c o m p l e t e w
i t h r o u g h m a t t r e s s e s a n d t h i c k s t r a w p i l l
o w s o t h e r w i s e t h e y o f f e r e d n o t h i n g b u t c o
m m u n a l d o r m s f i l l e d w i t h t i n y b e d s b o t h p l
a c e s b o a s t e d t a p s i n t h e i r y a r d s b u t t h a t w
a s a l l
```

Parker Newton
March 17, 2016
CSCI 181
Final Project

C23.TXT:

Keyword:
heavenhelpus

Plaintext:
t h e s t o r y b e a t m e u p h e a d m i t s a n d a t t i m e s i
t s c a r e d m e t o d e a t h h i k e s w e r e g r u e l i n g a n
d p a i n w a s b o r n e i n s i l e n c e t o c o m m u n i c a t e
w i t h p y g m y m e n n i c k r e s o r t e d t o h a n d s i g n a
l s t o d i s c u s s s u b j e c t s o f u n i v e r s a l i n t e r
e s t l i k e w o m e n a n d f o o d o n t h i s s t o r y i h a d e
n o u g h m a l e b o n d i n g

C24.TXT:

Keyword:
anoninanenuen

Plaintext:
f o r a m o m e n t h e e x a m i n e d t h e f e a t u r e l e s s c
o n v e x m e t a l w a l l t u r n i n g h i s p i s t o l t o m a x
i m u m a n d h o p i n g i t w o u l d n t m e l t i n h i s h a n d
b e f o r e i t b r o k e t h r o u g h h e o p e n e d f i r e o n t
h e d o o r w h e n t h e w e a p o n b e c a m e t o o h o t t o h o
l d h e t o s s e d i t f r o m h a n d t o h a n d a s h e d i d s o
t h e s m o k e h a d t i m e t o c l e a r a n d h e s a w w i t h s
o m e s u r p r i s e t h a t t h e d o o r h a d b e n b l o w n a w
a y p e e r i n g t h r o u g h t h e s m o k e w i t h a n u n c o m
p r e h e n d i n g l o o k o n h e r f a c e w a s t h e y o u n g w
o m a n w h o s e p r o t r a i t a r t o o d e t o o h a d p r o j e
c t e d i n a g a r a g e o n t a t o o i n e s e v e r a l c e n t u
r i e s a g o o r s o i t s e e m e d s h e w a s m o r e b e a u t i
f u l t h a n h e r i m a g e l u k e d e c i d e d s t a r i n g d a
z e d l y a t h e r h e r l o o k o f c o n f u s i o n a n d u n c e
r t a i n t y w a s r e p l a c e d b y p u z z l e m e n t

Parker Newton
March 17, 2016
CSCI 181
Final Project

C25.TXT:

Keyword:
betteroff

Plaintext:
h e s a w p a i n f l i c k e r i n h e r e y e s a n d k n e w h e h
a d t o u c h e d h e r s o m e p a r t o f h i m s e l f f e l t s h
a m e a t t h i s c h e a p v i c t o r y a l l t h e t e x t b o o k
s h e d r e a d o n t h e s u b j e c t o f d e a t h t o l d h i m t
h a t t h e b e r e a v e d s f i r s t s t r o n g i m p u l s e i s
t o g e t a w a y f r o m t h e p l a c e w h e r e i t h a p p e n e
d a n d t h a t t o s u c c u m b t o s u c h a n i m p u l s e m a y
t u r n o u t t o b e t h e m o s t h a r m f u l c o u r s e o f a c
t i o n

C26.TXT:

Keyword:
betsywetsy

Plaintext:
i t i s n i g h t n o w n o l o n g e r e v e n i n g b u t f u l l y
n i g h t a s i n b l a c k a s i f n o t p r e c i s e l y d e a d o
f e v e n i n g u s u a l l y h a s t h e a f t e r n o o n h a n g i
n g o n i t s c o a t t a i l s h a s a c t u a l f l e c k s o f d a
y l i g h t c l i n g i n g l i k e l i n t t o i t s l a p e l s b u
t n i g h t i s s o l i t a r y a l o o f u n c o m p r o m i s e d e
x t r e m e t h e s a f e m a r g i n s o f t h e d a y s t i l l f a
i n t l y v i s i b l e d u r i n g e v e n t i d e h a v e b e e n e
r a s e d b y n i g h t s d e n s e g u m o b s c u r e d b y i t s w
a s h o f s q u i d s q u i r t i n g s p a j a m a s a u c e a n d t
h e b l u e h o n e y m a n u f a c t u r e d b y m o t h s i s t h e
n i g h t a m a s k o r i s d a y m e r e l y n i g h t s p r i m d i
s g u i s e

Parker Newton
March 17, 2016
CSCI 181
Final Project

C27.TXT:

Keyword:
mustarriveontime

Plaintext:
a t l a s t t h e r e c a m e a t i m e w h e n t h e d r i v e r w e
n t f u r t h e r a f i e l d t h a n h e h a d y e t d o n e a n d d
u r i n g h i s a b s e n c e t h e h o r s e s b e g a n t o t r e m
b l e w o r s e t h a n e v e r a n d t o s n o r t a n d s c r e a m
w i t h f r i g h t i c o u l d n o t s e e a n y c a u s e f o r i t
f o r t h e h o w l i n g o f t h e w o l v e s h a d c e a s e d a l
t o g e t h e r b u t j u s t t h e n t h e m o o n s a i l i n g t h
r o u g h t h e b l a c k c l o u d s a p p e a r e d b e h i n d t h
e j a g g e d c r e s t o f a b e e t l i n g p i n e c l a d r o c k
a n d b y i t s l i g h t i s a w a r o u n d u s a r i n g o f w o l
v e s w i t h w h i t e t e e t h a n d l o l l i n g r e d t o n g u
e s w i t h l o n g s i n e w y l i m b s a n d s h a g g y h a i r t
h e y w e r e a h u n d r e d t i m e s m o r e t e r r i b l e i n t
h e g r i m s i l e n c e w h i c h h e l d t h e m t h a n e v e n w
h e n t h e y h o w l e d f o r m y s e l f i f e l t a s o r t o f p
a r a l y s i s o f f e a r i t i s o n l y w h e n a m a n f e e l s
h i m s e l f f a c e t o f a c e w i t h s u c h h o r r o r s t h a
t h e c a n u n d e r s t a n d t h e i r t r u e i m p o r t a l l a
t o n c e t h e w o l v e s b e g a n t o h o w l a s t h o u g h t h
e m o o n l i g h t h a d h a d s o m e p e c u l i a r e f f e c t o
n t h e m t h e h o r s e s j u m p e d a b o u t a n d r e a r e d a
n d l o o k e d h e l p l e s s l y r o u n d w i t h e y e s t h a t
r o l l e d i n a w a y p a i n f u l t o s e e b u t t h e l i v i n
g r i n g o f t e r r o r e n c o m p a s s e d t h e m o n e v e r y
s i d e a n d t h e y h a d p e r f o r c e t o r e m a i n w i t h i
n i t i c a l l e d t o t h e c o a c h m a n t o c o m e f o r i t s
e e m e d t o m e t h a t o u r o n l y c h a n c e w a s t o t r y t
o b r e a k o u t t h r o u g h t h e r i n g a n d t o a i d h i s a
p p r o a c h i s h o u t e d a n d b e a t t h e s i d e o f t h e c
a l e c h e h o p i n g b y t h e n o i s e t o s c a r e t h e w o l
v e s f r o m t h a t s i d e s o a s t o g i v e h i m a c h a n c e
o f r e a c h i n g t h e t r a p h o w h e c a m e t h e r e i k n o
w n o t b u t i h e a r d h i s v o i c e r a i s e d i n a t o n e o
f i m p e r i o u s c o m m a n d a n d l o o k i n g t o w a r d s t
h e s o u n d s a w h i m s t a n d i n t h e r o a d w a y a s h e s
w e p t h i s l o n g a r m s a s t h o u g h b r u s h i n g a s i d
e s o m e i m p a l p a b l e o b s t a c l e t h e w o l v e s f e l
l b a c k a n d b a c k f u r t h e r s t i l l j u s t t h e n a h e
a v y c l o u d p a s s e d a c r o s s t h e f a c e o f t h e m o o
n s o t h a t w e w e r e a g a i n i n d a r k n e s s t h e o l d m
a n m o t i o n e d m e i n w i t h h i s r i g h t h a n d w i t h a

c o u r t l y g e s t u r e s a y i n g i n e x c e l l e n t e n g l
i s h b u t w i t h a s t r a n g e i n t o n a t i o n w e l c o m e
t o m y h o u s e e n t e r f r e e l y

C28.TXT:

Keyword:
mexico

Plaintext:
o n t h e h i g h w a y i m a d e t h e s a m e t r i p i n t w o a n
d a h a l f h o u r s a l o t o f g o o d t h i n g s w e r e g i v e
n t o p e o p l e h e s a y s n o w t h e y r e a l l c o n d e m n i
n g t h e w h i t e m a n f o r c o m i n g b u t a l o t o f t h e o
l d t i m e r s a p p r e c i a t e w h a t t h e w h i t e m a n

C53.TXT:

Keyword:
import

Plaintext:
a s t h e y t a l k e d t o g e t h e r w o r d s r o s e s l o w l y
d e e p f r o m t h e i r

C60.TXT:

Keyword:
supports

Plaintext:
b u t w e a l s o n o t e d w h e n e v e r t h e y p u t d o w n

C61.TXT:

Keyword:
arguments

Plaintext:
a l l o f t h e p a r t i c i p a n t s w e r e g o n e

Conclusion:

The goal of this project is to cryptanalyze and decrypt to sensible English plaintext ten ciphertext files which we know were encrypted with a stream cipher. We also know that the keystream consists of the repetition of a sensible English phrase. By realizing this cryptosystem is similar to a classical Vigenere cipher, and by applying the cryptanalysis methods of the Kasiski Method and exploiting the regularity of the English language, I was able to decrypt to sensible plaintext all ten of the ciphertext files.