Basic Cryptanalysis Techniques
Craig Smith
November 17th, 2001

Introduction
Cryptography is a complex and mathematically challenging field of study. It involves taking some data or message and obfuscating it so that it is unreadable by parties that the message was not intended. Before the message becomes encrypted it is referred to as the plain text. Once a message becomes encrypted it is then referred to as the cipher text.

The study of cipher text in an attempt to restore the message to plaintext is known as cryptanalysis. Cryptanalysis is equally mathematically challenging and complex as cryptography. Because of the complexity involved with cryptanalysis work this document is only focused on the basic techniques needed to decipher monoalphabetic encryption ciphers and cryptograms.

The only application referenced in this document is the CRyptoANalysys ToolKit (CRANK). This program can be found at http://crank.sourceforge.net/. A basic understanding of cryptanalysis is essential to appreciating the complexities of a good cryptographic algorithm. For example a manager of a software company or someone who is involved with code auditing would find it is essential that good well tested algorithms are used instead of a weak in house cipher. This paper will give you the basic tools necessary to begin a rudimentary examination of a cipher.

Definition of terminology
This section will define several terms as well as give a brief introduction into cryptography. A term used specifically for cryptanalysis is called known text. Known text is when there is an encrypted message and a known corresponding plaintext. This may not be the whole message but perhaps a section of the message, e.g. every message sent ends with the plaintext letters "EOT". By using cipher text with known text you can attempt to deduce the complete key used to encrypt all messages, which will greatly facilitate future deciphering.

There are several basic methods that can be used to encrypt a message. One method is called a transporational cipher. This cipher only changes the order of the plaintext within the message, e.g. "LEAVE AT NOON" might become "EVAELTANOON". Another method is known as a substititional cipher. This method exchanges the characters in the plaintext with other characters defined by a key. The key is the mapping of characters from the plain text to the cipher text as in the following:

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ
zyxwvutsrqponmlkjihgfedcba
```

Using the same message from the above example this key would produce the following message: "OVZEVZGMLLM". This method of substitution is known as a Monoalphabetic Unilateral Substitution cipher. This term implies that for each letter in a plaintext message there is only one equivalent cipher character. (Note: The majority of
this document will focus on these types of cipher systems. Monoalphabetic Unilateral Substitution systems will simply be referred to as a substitution cipher for the sake of clarity and brevity.)

**Basic cryptanalysis techniques**

One good method for solving basic substitution ciphers is with frequency counts. A frequency count can be conducted on a cipher to learn what the most and least common characters are in the cipher. The most common letters in the English language are E,T,N,R,O,A,I and S. These eight characters make up around 67% of the words in the English language. Vowels, A,E,I,O, and U make up around 40% of English text. The frequency may vary depending on what the plaintext is. For example, if the message is source code it will use many more symbols than a message that is just written in English. If you conduct a frequency count of this paragraph your results would be: E, T, A, O, and S.

As you can see the results are not exactly the same. This is because the there are approximately 500 characters in the above paragraph. If you use a sample of 1000 characters or more your results will become more accurate. The frequency count of a single character is referred to as a *Unigraph*. If the frequency of cipher text is actually the same as plaintext then the encoded method is actually a transpositional cipher instead of a substitution cipher. Consider the following example:

*Plaintext:* IF WE DO NOT PROPERLY PROTECT THE USERS DATA WE CAN SIMPLY HIDE BEHIND THE DMCA IF SOMEONE NOTICES!!

*Transpositional:* I OOYFDTP O EPW PRRENRLOTTSDWEHEAECERT T SAC U ANPIE LDHTSYEIH NEMHBD DIM CMFENC OOSASNT! OEI!

*Substitution:* RU DV WL MLG KILKVIOB KILGVXG GSV FHVIH WZGZ DV XZM HRNKOBSRWW YVSRMW GSV WNXZ RU HLNVLMV MLGRXVH!!

Top 5 Unigraph Frequency counts:

- Plaintext: E, O, T, I and D
- Transpositional: E, O, T, I and D
- Substitution: V, G, L, R and H

Even though the transpositional cipher is a small sample, it has the top 4 letters used in plaintext with E being the highest.

When dealing with a substitution cipher you should check the frequency of letters and their adjacent letters as well. A pair of letters together is referred to as a *Digraph*. The common digraphs in the English language are, TH, HE, EN, RE and ER. There are also *Trigraphs* that consist of frequency of three letters next to each other THE, ING, CON, ENT and ERE.

**Roughness of the cipher**

If some characters are significantly more frequent than others then the cipher is considered "rough". Rough ciphers are a sign that a monoalphabetic unilateral cipher is
being used. A more complex cipher will distribute the frequency, making the cipher appear flat.

If the cipher was more complex than it would maintain a flat look even with a greater sample. Here's is an example of what a flat layout might look like.

```
X X X X X X
X X X X X X
XX XX X XX
XXX XXX XX XXX X
XXX XXXX XX XXX X
X X XXX XXXXX XX XXX X
X X XXX XXXXX XX XXX XXXX
ABCDEFHIJKLMNOPQRSTUVWXYZ
```

Determining a ciphers roughness helps in determining it's complexity and how much work might be involved in deciphering it.

**Format and patterns of a cipher**

Every outside piece of information can help in deciphering a message. What language, underlying content, word sizes, can all greatly assist in determining the plain text. Many ciphers will not show spaces and punctuation with their cipher. For instance they may group all the words in chunks of five with a set character if padding is needed. If the message is "DO NOT TRANSMIT", then the cipher may be broken up to look like this: WLMLG GIZMH NRGXX, "X" is padding; with simple Sunday paper cryptograms you get spaces and sometimes punctuation. Special tricks can be used to decipher these ciphers very quickly. One very fast trick is to find a word that is greater than six characters and determine it's pattern. For each unique letter in the word assign it a sequential number. We will decipher the following cryptogram as an example.

**UEY WKHH YOA OAXABSH PKNNABAFZ KFOZBYGAFZO ZE PAQKDLAB S GAOOSMA**

Here the cipher is small and our frequency count may be off by a significant amount but the benefit of knowing the size of the words (assuming that the spaces are correct) exists. First we'll select a large word, "KFOZBYGAFZO" and determine it's pattern.

```
KFOZBYGAFZO
12345678243
```
The first 8 letters are unique and the last three are repeats from the beginning of a word. Using this pattern and comparing it to 11 letter words in the dictionary, the possibilities are reduced provided "KFOZBYGAFZO" translates into an English word. Below is a simple perl script used to simplify this task.

```perl
#!/usr/bin/perl
# Searches a dictionary for a word pattern
# 2001 Craig Smith
#
# die "Usage: $0 <word>

my $DICT_FILE="/usr/share/dict/american-english"; # English text
my $my_pattern=get_pattern(uc($ARGV[0])); # Retrieve our pattern

open DICT, $DICT_FILE || die "Couldn't open $DICT_FILE!
while(<DICT>) { chomp($dict_word=$_); # Remove Carriage Return from word next if length($dict_word) ne length($ARGV[0]); # Only compare same length print "Try $dict_word.
" if $my_pattern eq get_pattern(uc($dict_word));
}
close DICT;
exit(0);

# Usage: get_pattern(TEXT) returns numeric pattern
sub get_pattern {
  my $word=shift;
  my $letter;
  my $pat_count=0;
  my $letter_pat;
  my $pattern;

  for(my $cnt=0; $cnt<length($word); $cnt++) {
    $letter=substr($word,$cnt,1);
    $letter_pat{$letter}=++$pat_count if !$letter_pat{$letter};
    $pattern.=+$letter_pat{$letter};
  }
  return $pattern;
}

Using this program the encrypted word "KFOZBYGAFZO" yields:
Try instruments.

If this word works then there will be 8 letters to the cipher. After replacing KFOZBYGAFZO with the word "instruments" the results look rather promising. The capital letters are the original cipher text and the lower case letters are what is believed to be the plain text.
Message: UEu WiHH use seXerSH PiNNerent instruments tE PeQiDLer S messSMe

Taking the word before "instruments" and using the program to see if any of the results end with 'erent'. With the assistance of the UNIX "grep" command the program yeilds:

./find_pattern.pl PKNNABAFZ | grep erent
Try different.

As evident these type of ciphers are extremely trivial and can be decoded quickly. This technique is only effective when there are spaces between the words in a cipher. There are many other different tricks that can used depending on the circumstances and any knowns about the cipher or the underlying plain text.

Using CRANK to assist in deciphering messages
CRANK is the CRyptANalysis too1Kit created by Matthew Russell. The program and source code can be obtained at http://crank.sourceforge.net/. This program focuses on Monoalphabetic substitution ciphers as well as transpositional ciphers. At the time of this writing there are two versions of CRANK, v1.4 and v2.1, though v2.1 is still in beta. Version 1.4 can help with much of the math and statistics when dealing with a substitution cipher.

There are two main windows that will do most of your work in, source and view. The source window is for the original cipher text and the view is the current guess on the plain text. The first thing is to load some text into the source window and prepare it to be encrypted. After pasting the text into the Souce window choose Text->Simple Filters.

![Image of CRANK interface]

There exists some convenient formatting tools that can strip out punctuation, spaces and even group the text into five characters a piece. This tool bar can be used for creating...
and formatting plaintext messages to encrypt.

CRANK can calculate the frequency of the unigraphs, bigraphs, and trigraphs by selecting Statistics->n-grams, then choose either unigrams, bigrams or trigrams respectively.

<table>
<thead>
<tr>
<th>Letter</th>
<th>Frequency</th>
<th>Standard Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>0.13465783</td>
<td>0.01516146</td>
</tr>
<tr>
<td>R</td>
<td>0.09492274</td>
<td>0.06552538</td>
</tr>
<tr>
<td>M</td>
<td>0.09050772</td>
<td>0.02796856</td>
</tr>
<tr>
<td>O</td>
<td>0.08388521</td>
<td>0.07084914</td>
</tr>
<tr>
<td>F</td>
<td>0.08167771</td>
<td>0.02232360</td>
</tr>
<tr>
<td>N</td>
<td>0.07505519</td>
<td>0.06641039</td>
</tr>
<tr>
<td>S</td>
<td>0.05960265</td>
<td>0.07497352</td>
</tr>
<tr>
<td>X</td>
<td>0.05298013</td>
<td>0.00507774</td>
</tr>
<tr>
<td>A</td>
<td>0.03973510</td>
<td>0.07586537</td>
</tr>
<tr>
<td>L</td>
<td>0.03532099</td>
<td>0.04363232</td>
</tr>
<tr>
<td>U</td>
<td>0.03311258</td>
<td>0.03489151</td>
</tr>
<tr>
<td>Y</td>
<td>0.03090506</td>
<td>0.01852725</td>
</tr>
<tr>
<td>C</td>
<td>0.02869757</td>
<td>0.04053648</td>
</tr>
<tr>
<td>D</td>
<td>0.02869757</td>
<td>0.0327524</td>
</tr>
<tr>
<td>Z</td>
<td>0.02649307</td>
<td>0.0090210</td>
</tr>
<tr>
<td>H</td>
<td>0.02207506</td>
<td>0.03791901</td>
</tr>
<tr>
<td>T</td>
<td>0.02207506</td>
<td>0.09295404</td>
</tr>
<tr>
<td>J</td>
<td>0.01986755</td>
<td>0.00113788</td>
</tr>
<tr>
<td>Q</td>
<td>0.01986755</td>
<td>0.00141466</td>
</tr>
<tr>
<td>E</td>
<td>0.01324503</td>
<td>0.12018794</td>
</tr>
<tr>
<td>V</td>
<td>0.00441501</td>
<td>0.01093798</td>
</tr>
<tr>
<td>K</td>
<td>0.00220751</td>
<td>0.00605160</td>
</tr>
</tbody>
</table>

Within Frequency, are the letters ordered by highest to lowest frequency. These numbers are actually the overall percentage frequency that the character show up in the text. "B" is the most frequent character at 13.4% followed by "R", "M", "O" and "F". The standard frequency column is the frequency that is normally expected. Sorting by standard frequency will show the top five characters as E, T, N, R and O. CRANK can recalculate the standard frequencies of many different types of plain text files. This can be useful for analyzing something that may not be written in complete sentences such as military communications.

CRANK also has a nice key control feature for use with basic substitution ciphers. The following toolbar is displayed when Monoalphabetic->key controls is selected.
First, the screen should have the alphabet in capital letters as well as lowercase. The capital letters represent the cipher text and the lowercase will be the key. If no key is present then press the "complete" button to fill in a default key. The key can be shifted by using "Shift<<" or "Shift>>". Press the "Reverse" button to re-arrange the key in reverse order.

Use "Clear" to wipe out the key and the display will show '*'s for unknown key entries.

By using the change key you can enter the letters you wish to substitute, e.g. since B was the most frequent it can be changed to an E. The key will be updated as well as the View window. Uppercase letters show up as entries that have not been decoded yet while the lower case letters are from the key. It should be noted that CRANK does not differentiate between Uppercase and Lowercase in the cipher itself.

CRANK has many more features and several tools such as an auto-cracker for monoalphabetic ciphers. It also has an extensive amount of tools for dealing with transpositional ciphers as well.

**Exercise for the reader**

Now that you should be very comfortable in solving monoalphabetic unilateral ciphers, I leave you with a small exercise. Time yourself and decode the following cipher text.
Enjoy!

References


URL:http://raphael.math.uic.edu/~jeremy/crypt/intro.html (24 Nov 2001)


SANS Institute. "SANS GIAC Training and Certification".
URL:http://www.sans.org/giactc/GIAC_certs.htm (24 Nov 2001)
SANS Miami 2020 | Miami, FLUS | Jan 13, 2020 - Jan 18, 2020 | Live Event
---|---|---|---
Cyber Threat Intelligence Summit & Training 2020 | Arlington, VAUS | Jan 20, 2020 - Jan 27, 2020 | Live Event
SANS Amsterdam January 2020 | Amsterdam, NL | Jan 20, 2020 - Jan 25, 2020 | Live Event
SANS Anaheim 2020 | Anaheim, CAUS | Jan 20, 2020 - Jan 25, 2020 | Live Event
MGT521 Beta Two 2020 | San Diego, CAUS | Jan 22, 2020 - Jan 23, 2020 | Live Event
SANS San Francisco East Bay 2020 | Emeryville, CAUS | Jan 27, 2020 - Feb 01, 2020 | Live Event
SANS Las Vegas 2020 | Las Vegas, NVUS | Jan 27, 2020 - Feb 01, 2020 | Live Event
SANS Vienna January 2020 | Vienna, AT | Jan 27, 2020 - Feb 01, 2020 | Live Event
SANS Security East 2020 | New Orleans, LAUS | Feb 01, 2020 - Feb 08, 2020 | Live Event
SANS New York City Winter 2020 | New York City, NYUS | Feb 10, 2020 - Feb 15, 2020 | Live Event
SANS Northern VA - Fairfax 2020 | Fairfax, VAUS | Feb 10, 2020 - Feb 15, 2020 | Live Event
SANS Dubai February 2020 | Dubai, AE | Feb 15, 2020 - Feb 20, 2020 | Live Event
SANS San Diego 2020 | San Diego, CAUS | Feb 17, 2020 - Feb 22, 2020 | Live Event
SANS Brussels February 2020 | Brussels, BE | Feb 17, 2020 - Feb 22, 2020 | Live Event
SANS Scottsdale 2020 | Scottsdale, AZUS | Feb 17, 2020 - Feb 22, 2020 | Live Event
Open-Source Intelligence Summit & Training 2020 | Alexandria, VAUS | Feb 18, 2020 - Feb 24, 2020 | Live Event
SANS Training at RSA Conference 2020 | San Francisco, CAUS | Feb 23, 2020 - Feb 24, 2020 | Live Event
SANS Manchester February 2020 | Manchester, GB | Feb 24, 2020 - Feb 29, 2020 | Live Event
SANS Jacksonville 2020 | Jacksonville, FLUS | Feb 24, 2020 - Feb 29, 2020 | Live Event
SANS Secure India 2020 | Bangalore, IN | Feb 24, 2020 - Feb 29, 2020 | Live Event
SANS Zurich February 2020 | Zurich, CH | Feb 24, 2020 - Feb 29, 2020 | Live Event
Blue Team Summit & Training 2020 | Louisville, KYUS | Mar 02, 2020 - Mar 09, 2020 | Live Event
SANS Secure Japan 2020 | Tokyo, JP | Mar 02, 2020 - Mar 14, 2020 | Live Event
SANS Munich March 2020 | Munich, DE | Mar 02, 2020 - Mar 07, 2020 | Live Event
SANS Northern VA - Reston Spring 2020 | Reston, VAUS | Mar 02, 2020 - Mar 07, 2020 | Live Event
SANS St. Louis 2020 | St. Louis, MOUS | Mar 08, 2020 - Mar 13, 2020 | Live Event
SANS Prague March 2020 | Prague, CZ | Mar 09, 2020 - Mar 14, 2020 | Live Event
SANS Austin Winter 2020 | OnlineTXUS | Jan 06, 2020 - Jan 11, 2020 | Live Event
SANS OnDemand | Books & MP3s OnlyUS | Anytime | Self Paced