Faculty of Computers, Informatics and Microelectronics

Technical University of Moldova

SI

Laboratory work # 2

Encryption: Rot13

Author:
Petru Negrei

Supervisor:
A. Railean

September 2014

1 Introduction

1.1 Objective

Understand the rot13 cipher and build a tool that break it.

1.2 Definition

ROT13 ("rotate by 13 places", sometimes hyphenated ROT-13) is a simple letter substitution cipher that replaces a letter with the letter 13 letters after it in the alphabet. ROT13 is an example of the Caesar cipher, developed in ancient Rome.

In the basic Latin alphabet, ROT13 is its own inverse; that is, to undo ROT13, the same algorithm is applied, so the same action can be used for encoding and decoding. The algorithm provides virtually no cryptographic security, and is often cited as a canonical example of weak encryption.

2 Implementation

The following program implements two way of breaking the rot13 substitution cipher:

- Frequency analysis
- Each substitution number

2.1 Cypher module

The text is received from a existing document.

```
def cipher
  str = ""
  File.open("data.txt", "r") do |f|
    f.each_line { |line| str << line }
  end
  rot(str, 13)
end
```

Then each letter is replaced by the given number.

```
def rot string, num
  origin = "a-zA-Z"
  cipher = [('a'..'z'), ('A'..'Z')].map {|range| range.to_a.rotate(num).join }.join
  string.tr origin, cipher
end
```

And the last step is to initiate a dictionary of words.

```
def initiate_dictionary
  # /usr/share/dict/words
  File.open("./dictionary.txt") do |file|
    file.each { |line| @words[line.strip] = true }
  end
end
```

2.2 Frequency analysis

First we begin by iterating through known most used letters in english dictionary and check each for matching number of words.

```
def rot_num
  chars = ["e", "t", "a", "o", "i", "n"]

  char = chars.detect {|char| valid?(ord_num(char))}
  ord_num char
end
```

```
def valid? num
  rot(cipher, num).split.select{ |word| @words[word.downcase] }.count > 1
end

def ord_num char
  most_used_char.ord - char.ord
end

def most_used_char
  @msch ||= cipher.scan(/\w/)
                          .inject(Hash.new(0)) { |h, c| h[c] += 1; h }
                          .max_by { |_,v| v }.first
end
```

and at the end we output the original string.

```
def original
  rot cipher, rot_num
end
```

2.3 Step analysis

This implementation is easier to apply, we iterate through each possible number of permutation of alphabet (26) and check which has the most valid words, and then apply the algorithm.

```
def check_rot
   (0..26). map do |num|
    cipher.split.select{ |word| @words[rot(word, num).downcase] }.count
  end
end

def rot_num
  26 - check_rot.each_with_index.max[1]
 end

def show_original
  rot cipher, rot_num
end
```

3 Conclusion

After making this laboratory work I learn about different methods to break the ROT13 (and caesar ciphers) substition cypher and it is easy to observe that algorithm provide almost no cryptographic security. Some of the possible aproaches to improve the security are:

- *The Vignere cipher* - to use different shifts at different positions in the message. (additional key)
- *One-time pads* - where the previous additional key used is infinitely long or a completely random key.
- *Multi-character alphabets* - treate larger chunks of text as the characters of our message.