On the *2ROT13* Encryption Algorithm

ak, #mum cryptolabs

September 25, 2004

Abstract

This paper describes the research results of the #mum cryptolabs crew regarding a previously underestimated encryption algorithm, the *Double-Rotate by 13* – or short "2ROT13" – algorithm, including a detailed analysis of the algorithm, a close look at some practical issues and an outlook at possible future applications of this encryption algorithm.

Contents

1 Introduction **1**

2 Description of the 2ROT13 Encryption Algorithm **2**

3 An Implementation of 2ROT13 **2**

4 Future Goals of 2ROT13 Development **3**

5 Source Code of PG2ROT13P **3**

1 Introduction

People have used different ways of encoding information since the dawn of mankind. Some ways were meant to be understood by everybody, like natural, coloquial language, other ways are meant to hide information or to make it available to a very limited number of persons, e.g. a "language for the gods" that is spoken only by a caste of elite shamans or secret strategic plans that the enemy must not know about.

One of the first well-known and well-documented encryption algorithms is the "Caesar Code", a simple code where each letter of the alphabet is associated with another letter of the alphabet, rotated by *n* positions. Here, *n* is the secret key that is necessary to both encrypt the human-readable message to the cipher text and to decrypt the cipher text to the human-readable message.

At the time of writing, the author of this paper is not aware of any *scientific research paper* that showed any cryptographical weaknesses in this simple yet efficient system. A lot of people called it "obvious" that the algorithm is insecure, but this is totally

unscientific, and cannot be accepted as a serious analysis. Yet, the Caesar code has been under constant peer review for over 2000 years, and is thus considered secure by the #mum cryptolab crew.

Over the time, other, more complicated crypto systems have been invented, but most of them have already been broken, e.g. the Enigma machine, the DES encryption algorithm, or RSA-1024[WLB03]. Thus, the author currently considers the Caesar code more secure than the other mentioned encryption algorithms.

2 Description of the 2ROT13 Encryption Algorithm

The 2ROT13 encryption algorithm is based on the ROT13 encryption algorithm. The ROT13 encryption algorithm is a special case of the Caesar code, with a fixed key of 13. ROT13 has been under wide use for over 20 years, where it was mostly useful for encrypting parts of usenet and email messages. Today, most Unix-like operating systems feature an implementation of ROT13 and/or the Caesar code. ROT13 can thus be considered ubiquitous in the world of IT.

As briefly mentioned before, ROT13 is based on the principle that every letter in a reference alphabet corresponds to another letter in a rotated alphabet. In the case of ROT13, the rotated alphabet is rotated by 13 places relative to the reference alphabet. The rotation width of 13 was chosen because the latin alphabet that is common in the western world consists of 26 letters. This leads to the nice effect that e.g. the letter *A* of the reference alphabet corresponds to the letter *N* of the rotated alphabet and the letter *N* of the reference alphabet corresponds to the letter *A* in the rotated alphabet. This clearly shows that ROT13 is a secret key encryption algorithm, with the secret key being *13* (but don't tell anyone).

A number of other encryption algorithms are designed in a way that the process of encrypting the data is done several times, so-called "rounds". Round-based encryption algorithms are e.g. AES, DES or 3DES, which consists of 3 rounds of DES. It can thus be assumed that more rounds of encryption bring more security.

Based on this idea, the 2ROT13 encryption algorithm has been developed. It consists of nothing but two rounds of encrypting the cipher text with ROT13. Due to the special properties of ROT13, the number of rounds must be even, otherwise the algorithm provides only as much security as simple ROT13. The following mnemonic rhyme describes this situation pretty well:

> "What you can't divide by two, is not very good for you."

Good implementations would be e.g. 2ROT13, 4ROT13, 6ROT13 or 2048ROT13. But so far, the authors see no need in using more than two rounds, as it is currently considered hard enough to break. Bad implementations would be ROT13, 3ROT13, 7ROT13 or 1697ROT13.

3 An Implementation of 2ROT13

Currently, an implementation of 2ROT13 exists, which is called *Pretty Good Double ROT13 Privacy* – or short PG2ROT13P – and is meant to be a successor to the infa-

mous *Pretty Good Privacy* cryptography toolkit. It implements both encrypting and decrypting cipher texts, and is written in the Perl programming language. You can find the source code in the last section of this paper.

4 Future Goals of 2ROT13 Development

The development of 2ROT13 has just begun. 2ROT13 needs to become more widespread. This means that we will blackmail all Linux vendors and distributors to secretly replace GnuPG by PG2ROT13P to help PG2ROT13P become more widespread. Other ideas are to implement 2ROT13 in the SSL and SSH cipher suites, and to make them the *default*. The `#mum` cryptolab crew is confident that this would make the world a safer and more peaceful palce on earth. Long-term goals are also to make the EU parliament pass a law that requires all personal letters, postcards and even face-to-face conversations to be encrypted with 2ROT13.

5 Source Code of PG2ROT13P

```
#!/usr/bin/perl
# pretty good double-rot13 privacy - PG2ROT13P
# (c) 2004 ak, #mum cryptolabs

$header = "-----BEGIN 2ROT13 MESSAGE-----";
$footer = "-----END 2ROT13 MESSAGE-----";

sub rot13($_) {
  my $x = shift;
  $x =~ tr/a-zA-Z/n-za-mN-ZA-M/;
  return $x;
}

sub do_2rot13_encrypt() {
  @lines = <STDIN>;
  print "$header\n";
  foreach $line (@lines) {
    foreach $i (1..2) { # two rounds of ROT13
      $line = &rot13($line);
    }
    print $line;
  }
  print "$footer\n";
}

sub do_2rot13_decrypt() {
  print STDERR "pg2rot13p: go ahead and type your message...\n";
```

```
  @lines = <STDIN>;
  if ($lines[0] ne "$header\n" or $lines[$#lines] ne "$footer\n") {
    print STDERR "Sorry, this is not a valid 2ROT13 message\n";
    exit(1);
  }
  shift(@lines); pop(@lines);
  foreach $line (@lines) {
    foreach $i (1..2) {
      $line = &rot13($line);
    }
    print $line;
  }
}

sub usage() {
  print STDERR "usage: pg2rot13p [-e|-d]\n";
  exit(1);
}

if ($#ARGV == -1 or $ARGV[0] eq '-d') {
  &do_2rot13_decrypt();
} elsif ($ARGV[0] eq '-e') {
  &do_2rot13_encrypt();
} else {
  &usage();
}
```

References

[WLB03] Weis, Lucks, Bogk: Sicherheit von 1024 bit RSA Schlüsseln gefährdet http://cryptolabs.org/rsa/WLBrsaDuD.pdf