Cairo University
Faculty of Engineering

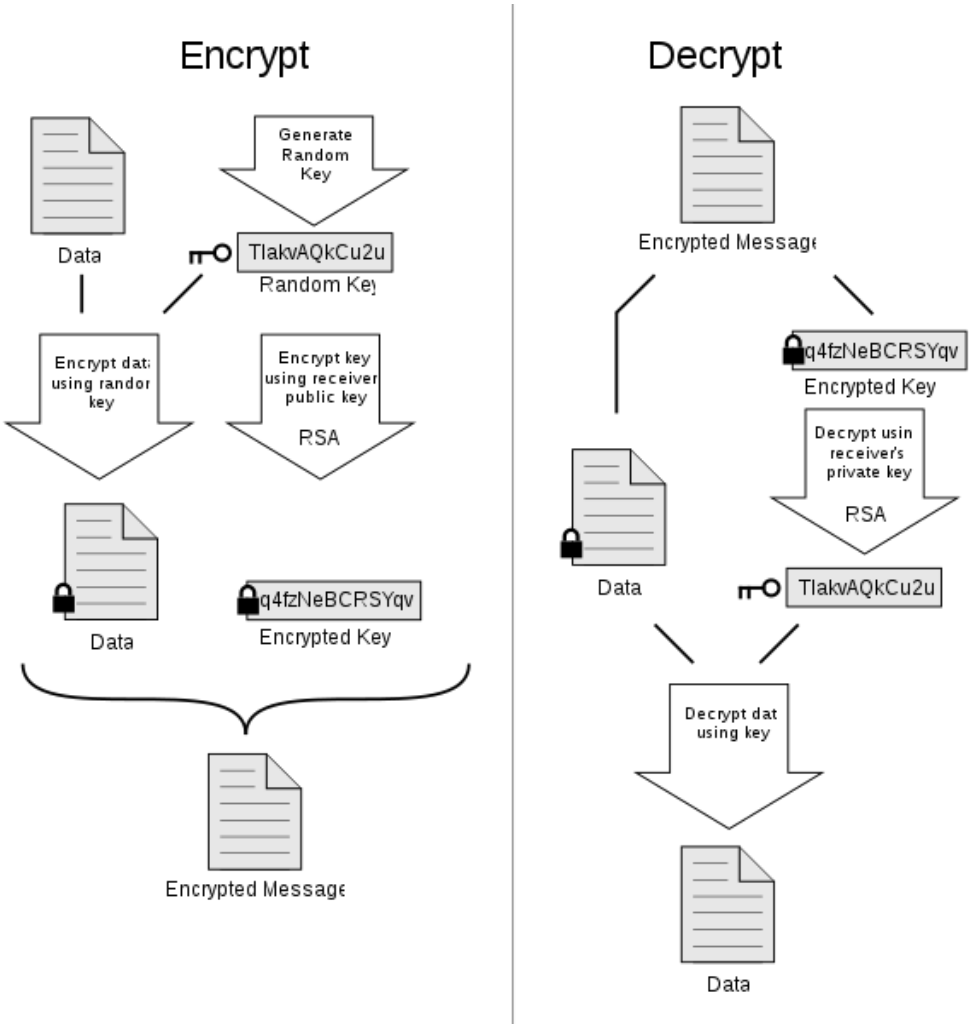
Cryptography and Network Security - CMPN 426
Spring 2020

Security Project Report

Name	Ahmed Osama Abdelatty Mohamed
ID	1152151
Project	Security Project 1: Sending Secure Emails
Submitted to	Professor Samir Shaheen Eng. Lydia Waheed

Overview

Both sender and receiver has to be configured to generate a public - private key pair and start save them in a special folder called “keys” this is used to communicate the public keys between the two parties of the program. Figure.1 illustrates the PGP algorithm applied and main flow of the program.



Sender Implementation

The sender starts by reading the email message from a text file, then it generates a random DES key with length 56 bits and then encrypt it with the receiver's public key. Then encrypt the email message using DES CBC mode (padding the data if necessary) and finally concatenates both the base64 representation of the encrypted DES key after applying RSA on it and the encrypted message and then converting the result to a base64 string to send off to Gmail service.

Receiver Implementation

The receiver starts with getting all the possibly encrypted messages from the mail and starts decrypting one after the other like the following. It decodes the base64 email message, then separate the encrypted DES key from the encrypted mail message. It then decrypts the DES key by using the receiver's private key. After recovering the DES key it attempts to decrypt the message (unpadding after, if necessary) and after all this it prints the email to the terminal as well as saving it in an external file.

Analysis Part:

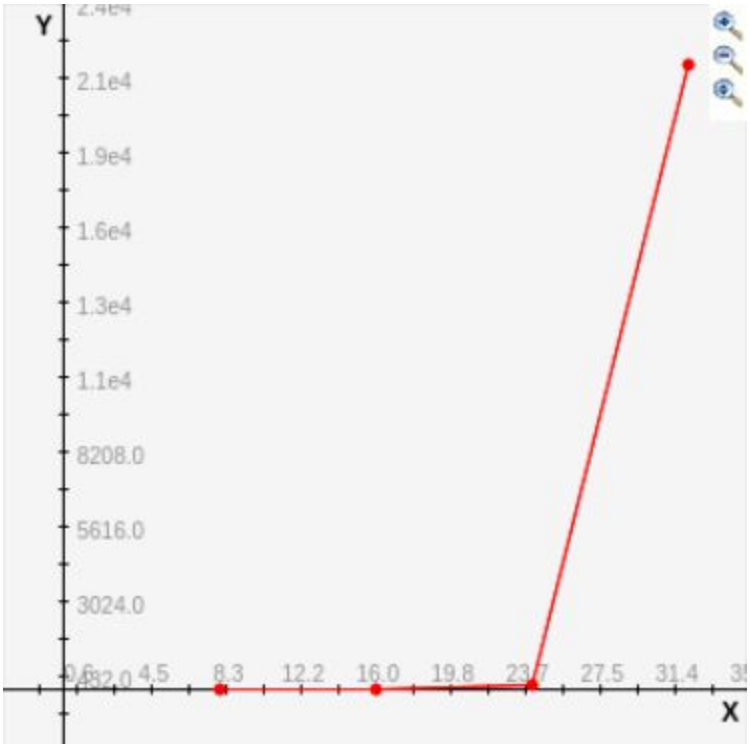
A brute force attack was implemented and applied on the DES, given the original text and the corresponding cipher, the analysis module shall be able to crack the DES key. The test was performed on DES keys of multiple sizes.

The module ran on a **Intel Core i7-4500U (4th Gen) processor with 4GB of RAM.**
Here's the running times of the brute force algorithm tried on different key sizes.

- 8-bit key: 2.486476ms
- 16-bit key: 641.955105ms
- 24-bit key: 2m58.276062259s
- 32-bit key: 6h11m34.593129975s

The **32-bit key** was the maximum size the module was able to retrieve, going on with this brute force algorithm on higher bit keys will need a lot of time, or higher computing power.

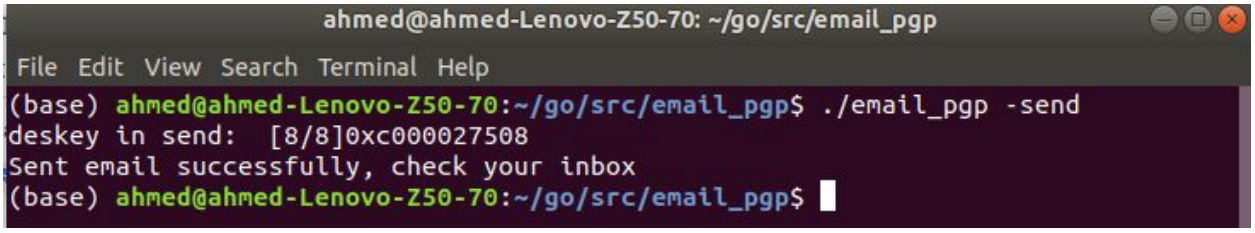
Here's a graph that plots the time to break the DES key in seconds on the y-axis versus key length in bits on the x-axis.



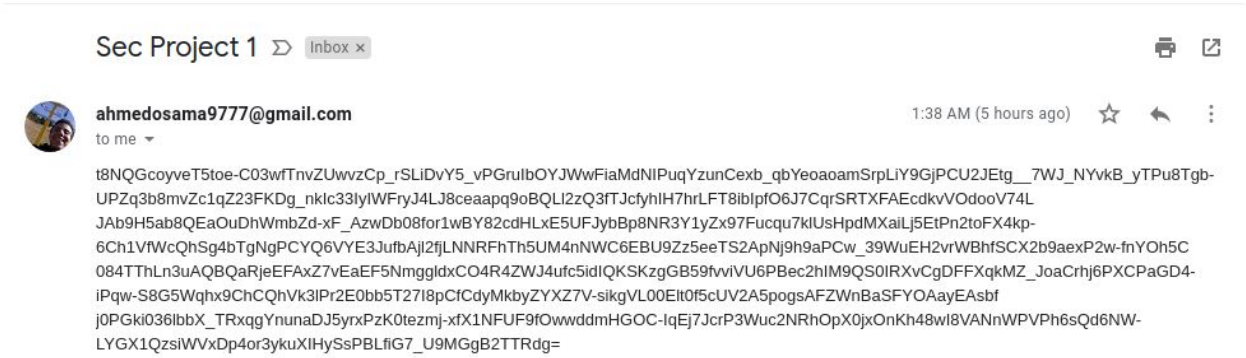
Test Sample

This is the text file that shall be read by the sender's script to be encrypted and sent over Gmail API.

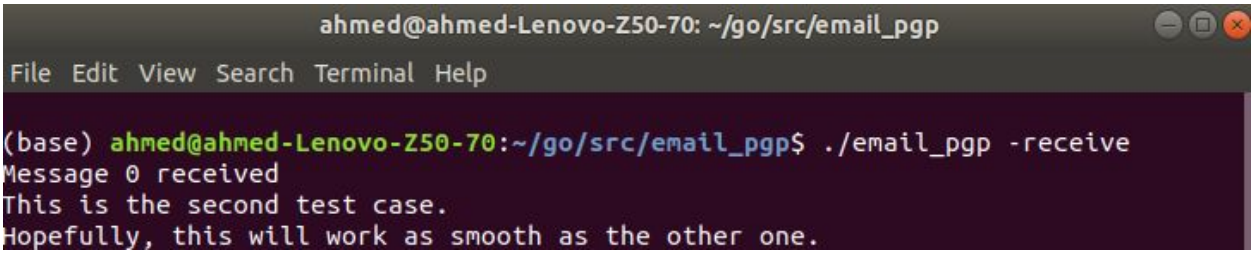
The sender module is called through this way in sender.



This is the received encrypted email sent by the sender, this shall now be read and retrieved by the receiver.

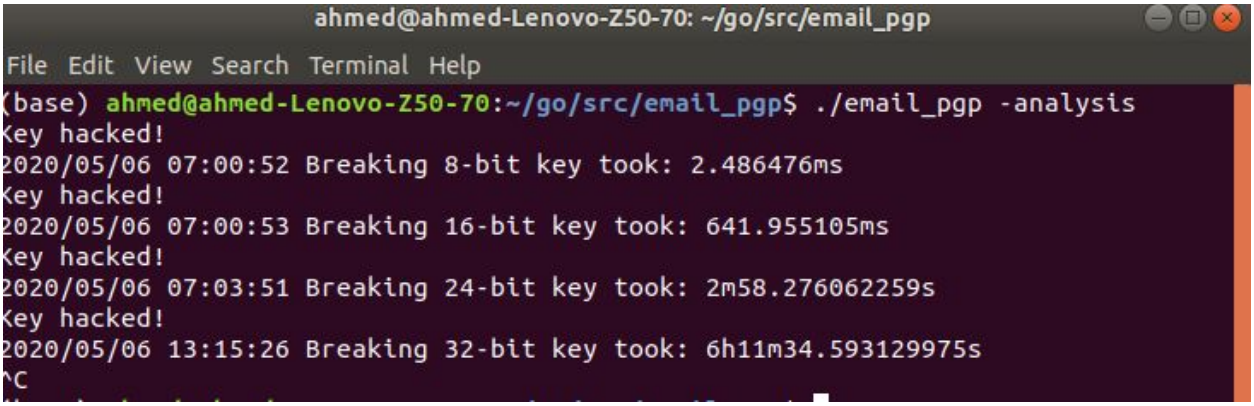


The receiver module is called through this way from the terminal. It checks the emails found in the inbox of the associated gmail account and when it finds the email with the right security project subject, it starts decrypting it. The decrypted text is shown here as an output on the termina.

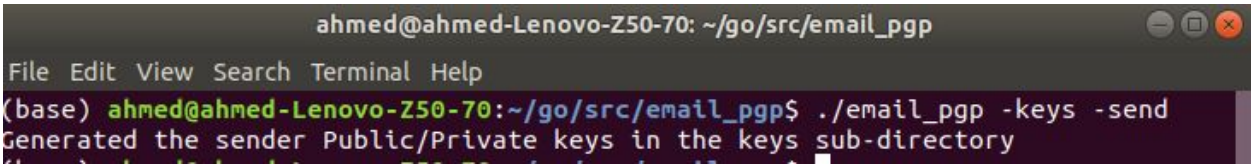


This is the .txt file that has the decrypted email after being retrieved by the receiver.

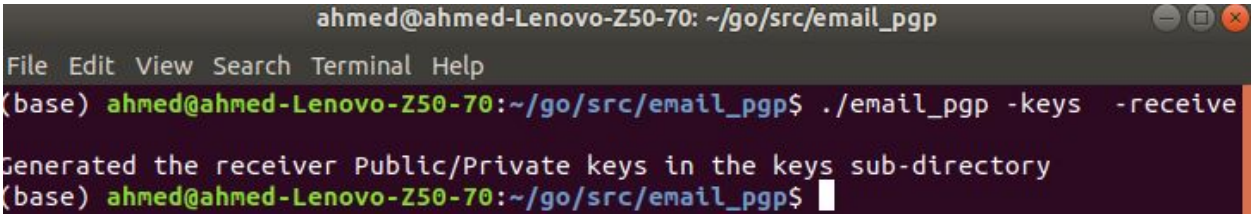
Here's how the analysis module that tries to break the DES key with different sizes is run and its output.



Finally, to generate the public and private keys for the sender and then save them in keys directory, the following command shall be run.



The same goes with the receiver.



Instructions to Run

The ./main can be used from the terminal, and this is the main entry point of the application.

1. First we have to generate the public and private keys of sender and receiver, this can be done by running the two commands: “./main -keys -send” & “./main -keys -receive”
2. To modify the receiver email, navigate to “/utils/google.go” and edit line 98, this will edit the receiver mail. Also. navigate to “/receiver/receiver.go” and edit line 61, with the receiver mail as well.
3. A credentials.json file must be acquired from Google to be able to work on the Gmail API and thus send emails over Gmail. Follow this link: Go Quickstart | Gmail API, to get the credentials.
4. Download the credentials.json, save it to “Golang_code” directory, then you can run the command “./main -send”, this will direct you to a link where you can log in to your google account (sender) and get a security code, type the security code in the terminal.
5. That’s it! Running the “/.main -receive” shall now receive the ciphered email and decrypt it.
6. The analysis module can be run through “/.main -analysis”