Henry Post

ITMS 428

Data in Transit:

Ensuring integrity and confidentiality

Contents

[Live demonstration 1](#_Toc526083124)

[Private keys 2](#_Toc526083125)

[Public keys 2](#_Toc526083126)

[Public key exchange 2](#_Toc526083127)

[Sending your public key 2](#_Toc526083128)

[What your friend sees 4](#_Toc526083129)

[Your friend’s response 5](#_Toc526083130)

[Receiving a message encrypted with your public key 7](#_Toc526083131)

[Integrity 8](#_Toc526083132)

[Original program’s signature 8](#_Toc526083133)

[Modified program’s signature 8](#_Toc526083134)

[Confidentiality 9](#_Toc526083135)

# Live demonstration

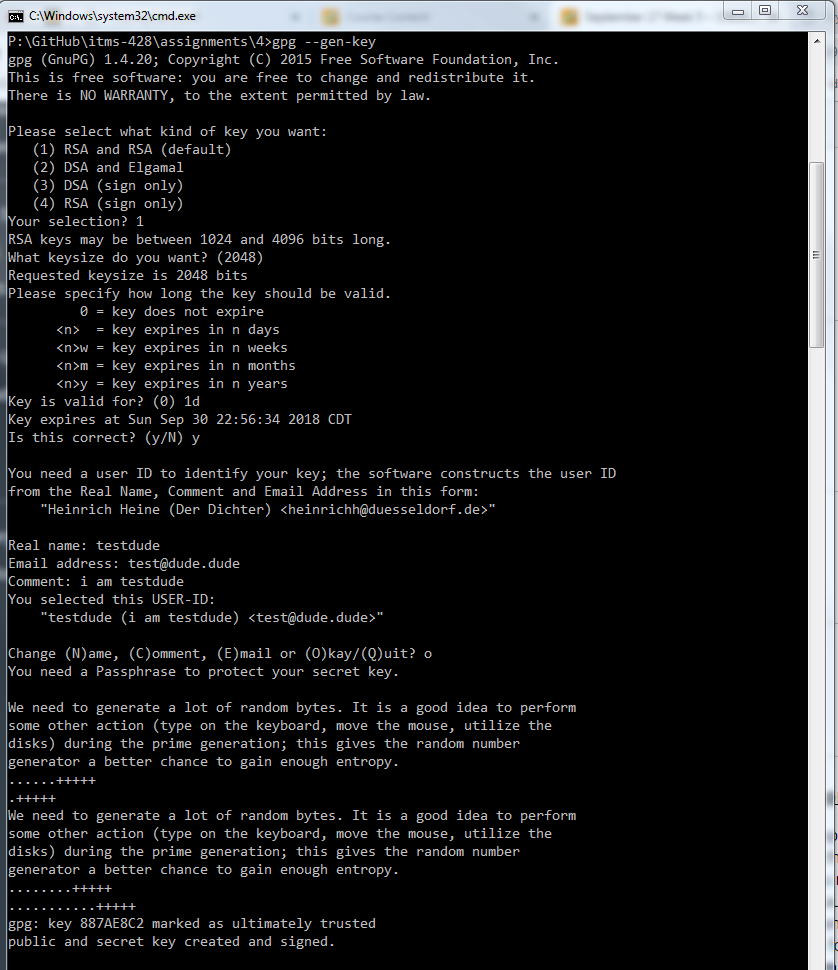
The first thing to do when you want to do when either encrypt or sign a piece of data is to create a public-private key pair to encrypt messages for others, and to decrypt messages from other people.

This can be done using `GPG`, an implementation of an encryption standard known as `PGP`, or ‘Pretty Good Privacy’.

After having installed `GPG`, the following command can be run to generate a key:

`gpg --gen-key`

Below is what the output may look like:



## Private keys

Now that you’ve got a private and public key, here’s how you use them:

Your private key is used to:

* Prove that you have the public and private keys, i.e. that *you are you*.
* Decrypt messages meant only for you
* Digitally sign data

Your private key is meant to remain private, and shouldn’t be shared with anyone.

## Public keys

Your public key, however, is meant to be…public.

Your public key is used to:

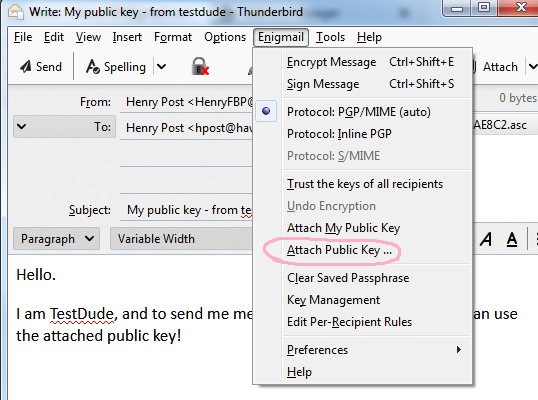
* Encrypt messages for the holder of a corresponding private key.
* That’s it!

You can see why you would want to share this with others:  
How can someone send you something meant only for you if they cannot encrypt it for you?

With this information, we can compare a private key to a key that opens something, and a public key to a lock, that locks something for the holder of the private key.

## Public key exchange

### Sending your public key

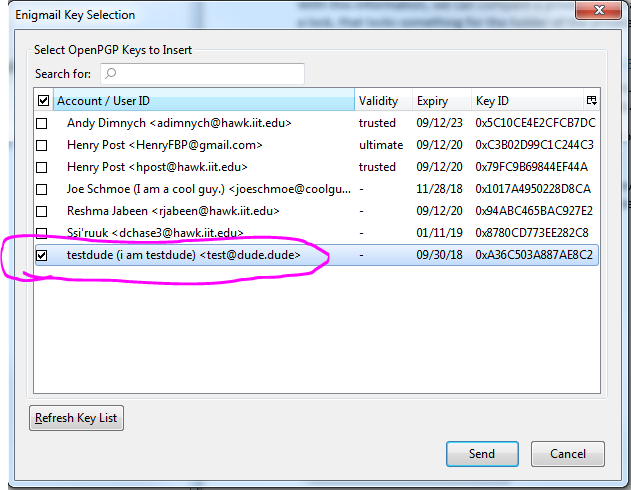
The next step of the process is to find someone who you want to send and receive encrypted messages with, and actually performing the sending and receiving.

I will be using Mozilla Thunderbird and Enigmail for this demonstration.

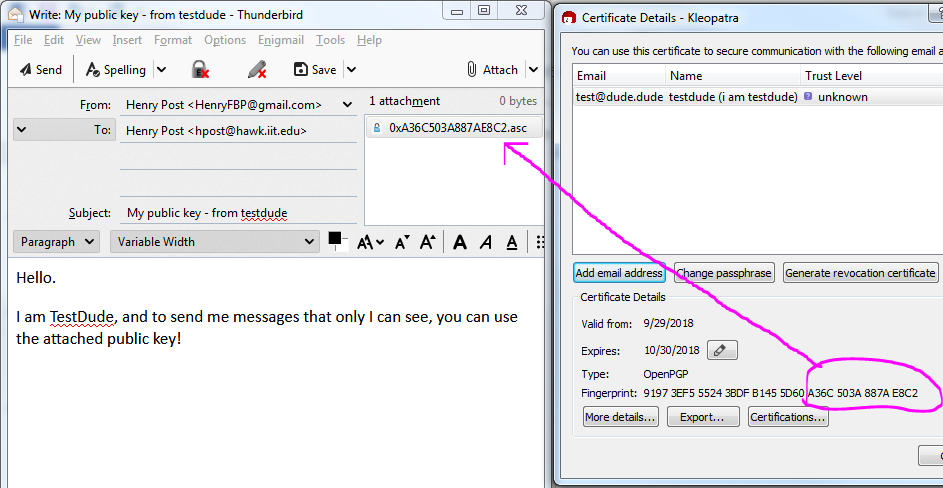
Make a new email, addressed to your friend.

I’ve sent this one to my IIT email from my personal Gmail account.

Once you have everything setup, go to `Enigmail > Attach Public Key …` and you’ll be prompted to select a public key that you have.

Then, select the public key that you want to give to your buddy.

The person who has the matching private key will be able to decrypt the message that your friend sends using the public key that you attach.



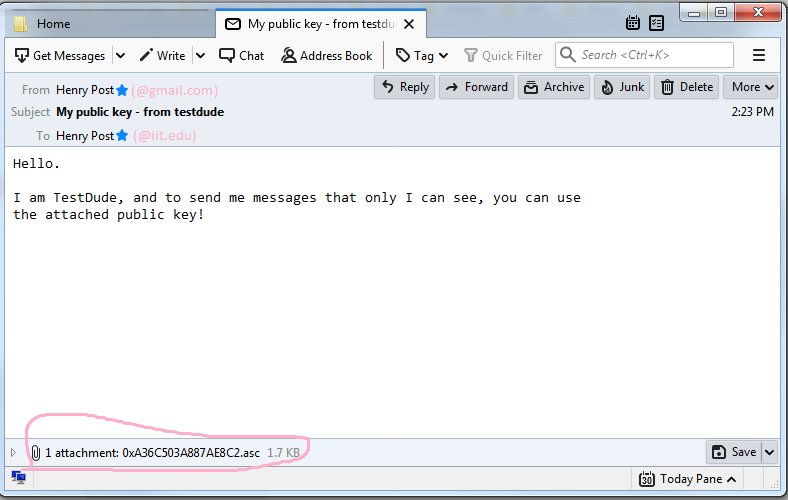
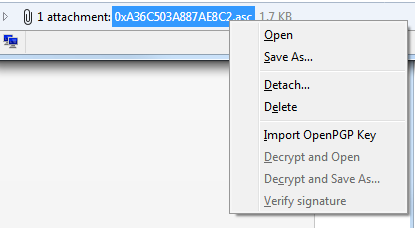
If you want to make sure that you’ve got the right public key attached, a tool like Kleopatra can be used to examine all public and private keys that you have.

Just to be sure that I’m sending the right public key here, I’ve opened up Kleopatra and made sure that the fingerprint of the attached public key matches the actual fingerprint of the public key for “testdude”.

If everything looks good, go ahead and send your email!

### What your friend sees

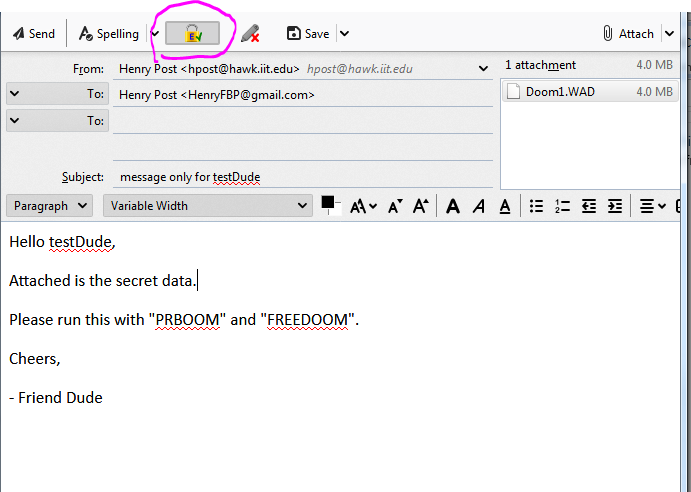
Your friend will see something like this:



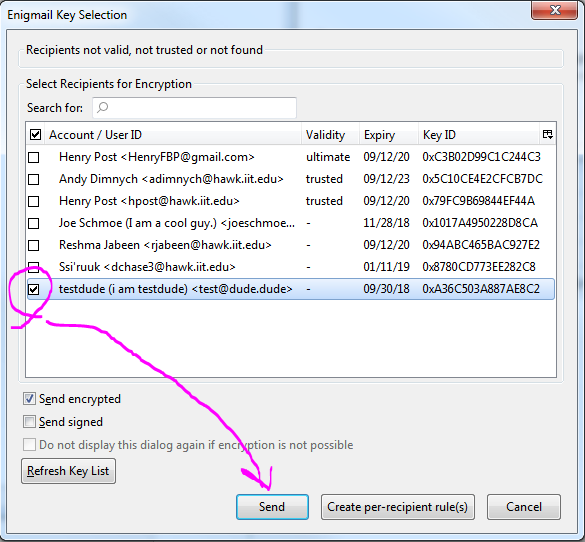
If you were the friend, you could add the key to your list of public keys by right-clicking on the attachment and then selecting `Import OpenPGP Key`.

### Your friend’s response

As the friend of TestDude, you would write your message normally, except for one ‘key detail’[[1]](#footnote-1): You’ll be clicking on this box to ensure that before your message is sent, it is encrypted only for TestDude!



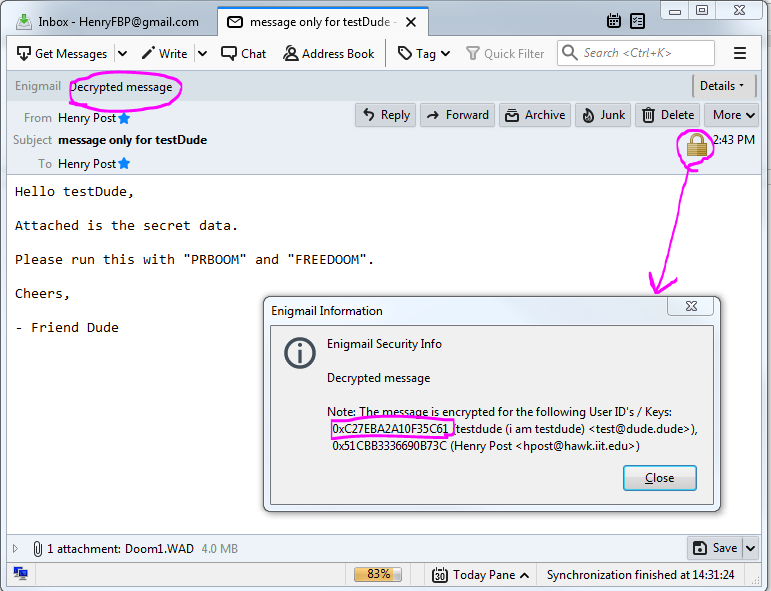
Once your message is ready and the little lock icon () is clicked, then click send!



A dialogue like the one shown above will be opened, and from there you can select who should be able to decrypt your message. As I only want to send my secret data to TestDude, I just select their key.

## Receiving a message encrypted with your public key

Once my buddy, FriendDude, is finished composing their message and has sent it to me, I’ll see something like this:



You can see that “Enigmail | Decrypted message” as well as a little lock icon shows up in the message.

Clicking the lock icon will show you who the message is encrypted for!

Well, now that the secret DOOM WAD has been downloaded, it’s time to play some DOOM.

Safely and secretly, of course.

# Integrity

Integrity on its own is a very useful thing.

Not all data needs to be confidential all the time!

Ensuring the integrity of data can prevent corruption, unwanted modification, MITM attacks, and a plethora of other issues or vulnerabilities essentially caused by one thing: **corrupt or modified data**.

Good examples of the use of just verifying integrity of a file are:

* Checksums for executable files
* Public, companywide statements released by the CEO or someone important
* Any email!

The way that OpenPGP can ensure integrity of data is by **signing**.

Signing data is a way to prove that **only the person with the private key** has approved the data that is associated with the signing.

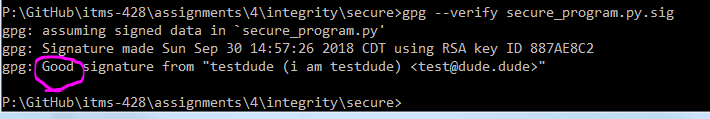
Along with this word document, you’ll find a folder, `integrity` that contains two programs.

I have signed the program in the `secure` folder, and put an intentionally modified version of the program in the `insecure` folder.

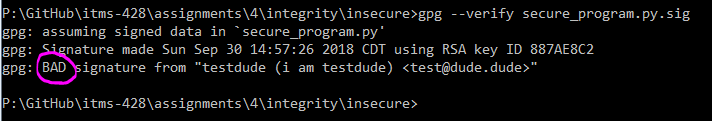
Using `GPG`, one can verify that `secure\_program.py` is indeed secure as determined by the author by using `gpg –verify <SIGNATURE.SIG> `.

Below are the results of running it on both the original and modified files.

## Original program’s signature



## Modified program’s signature



If you have Python installed, feel free to run the program to see that it does do something differently.

# Confidentiality

Confidentiality is achieved using GPG simply by encrypting data.

Once you have the public key of a person you wish to send data to, you simply use it to encrypt data that you want only that person to see.

1. This is intended to be humorous. Apologies if it isn’t.

   Hi Professor Dawson! Do you like footnotes too?

   Also, I hope you like DOOM. It’s a cool game. [↑](#footnote-ref-1)