Data Encryption

September 29, 2024

1 Encryption and Hashing

Owen Kroeger

ITT-305

Prof. Sluiter

- Comparing hashing with encryption
- Differences between symmetric and asymmetric encryption
- Configuring secure communication on mobile devices
- Quantum Computing
- Encryption screenshots

1.0.1 Hashing vs Encryption

Aspect	Hashing	Encryption
Purpose	Hashing is used for data integrity, making sure that data has not been altered.	Encryption is used for confidentiality, ensuring that data cannot be understood by unauthorized users.
Reversibility	Hashing is a one-way process (irreversible). You cannot obtain the original data from the hash.	Encryption is a two-way process (reversible). The original data can be retrieved using a decryption key.
Use Cases	Used for password storage, data integrity checks, and digital signatures.	Used for securing sensitive data like messages or files.
Output Length	The output of a hash is always a fixed length (depending on the encryption key length), regardless of the size of the input.	The length of the encrypted output depends on the input and encryption algorithm.
Security	Hashes are designed to be unique for each input; however, collisions (same hash for different inputs) can happen, though it's rare.	The security depends on the strength of the encryption algorithm and key size (e.g., AES-256 is highly secure).

1.0.2 Symmetric vs Asymmetric Encryption

Aspect	Symmetric Encryption	Asymmetric Encryption
Key Usage	Uses a single key for both encryption and decryption.	Uses a pair of keys: a public key (for encryption) and a private key (for decryption).
Speed	Faster and more efficient for encrypting large amounts of data.	Slower compared to symmetric encryption, but good for secure key exchanges and small data.
Use Cases	Often used for encrypting data at rest (files) and in transit (VPNs).	Used for secure key exchanges (e.g., TLS/SSL), digital signatures, and encrypting small amounts of data (emails, messages).
Key Distribution	Key distribution is hard because the same key must be securely shared between the communicating parties.	The public key can be openly shared, while the private key remains secret, simplifying secure communication.
Examples	AES (Advanced Encryption Standard), DES (Data Encryption Standard)	RSA (Rivest-Shamir-Adleman), ECC (Elliptic Curve Cryptography)

1.0.3 Configuration of Phone Applications for Secure Communication

Traditionally, I have not cared much about the security of my messaging. However, as of recently, I have been more conscious of comparing and using different private messaging services.

- **WhatsApp**: Since it uses end-to-end encryption by default, I do like WhatsApp as an option for simple texting. However, I did get banned on WhatsApp as soon as I sent a Russian message, so I am not entirely sure that I trust them.
- **Signal**: This app offers high levels of privacy and security, with open-source encryption algorithms and no data retention. I have actively been using this already between my family, as my sister at one point needed to securely communicate with the rest of the family without the risk of data interception. We have continued to use it for our group chat since.
- **iMessage**: Between iOS users, I will continue using iMessage for secure conversations. I will ensure my messages are set to use end-to-end encryption and avoid backing up messages to iCloud unless the backups are encrypted. Although I really have nothing in iMessage that I would care getting out, I do feel better about knowing that the platform I use most often is secure.

1.0.4 Research on Quantum Computing vs. Traditional Computing

Quantum Computing vs. Traditional Mainframe/Cloud Computing: Quantum computers operate differently from traditional computers. Instead of using classical bits, quantum computers use qubits that can represent 0, 1, or both simultaneously due to superposition. This allows quantum computers to process a massive amount of data simultaneously, making them far more powerful than classical computers in specific applications.

What can current quantum computers do?

- Current quantum computers are still in the experimental phase and are mainly used for research, optimization, and testing small-scale quantum algorithms.
- **Google** and **IBM** have achieved "quantum supremacy," meaning their quantum computers can solve specific problems faster than classical supercomputers. However, these are specialized cases, and quantum computers are not yet ready for general-purpose release or tasks.

Future of Quantum Computing:

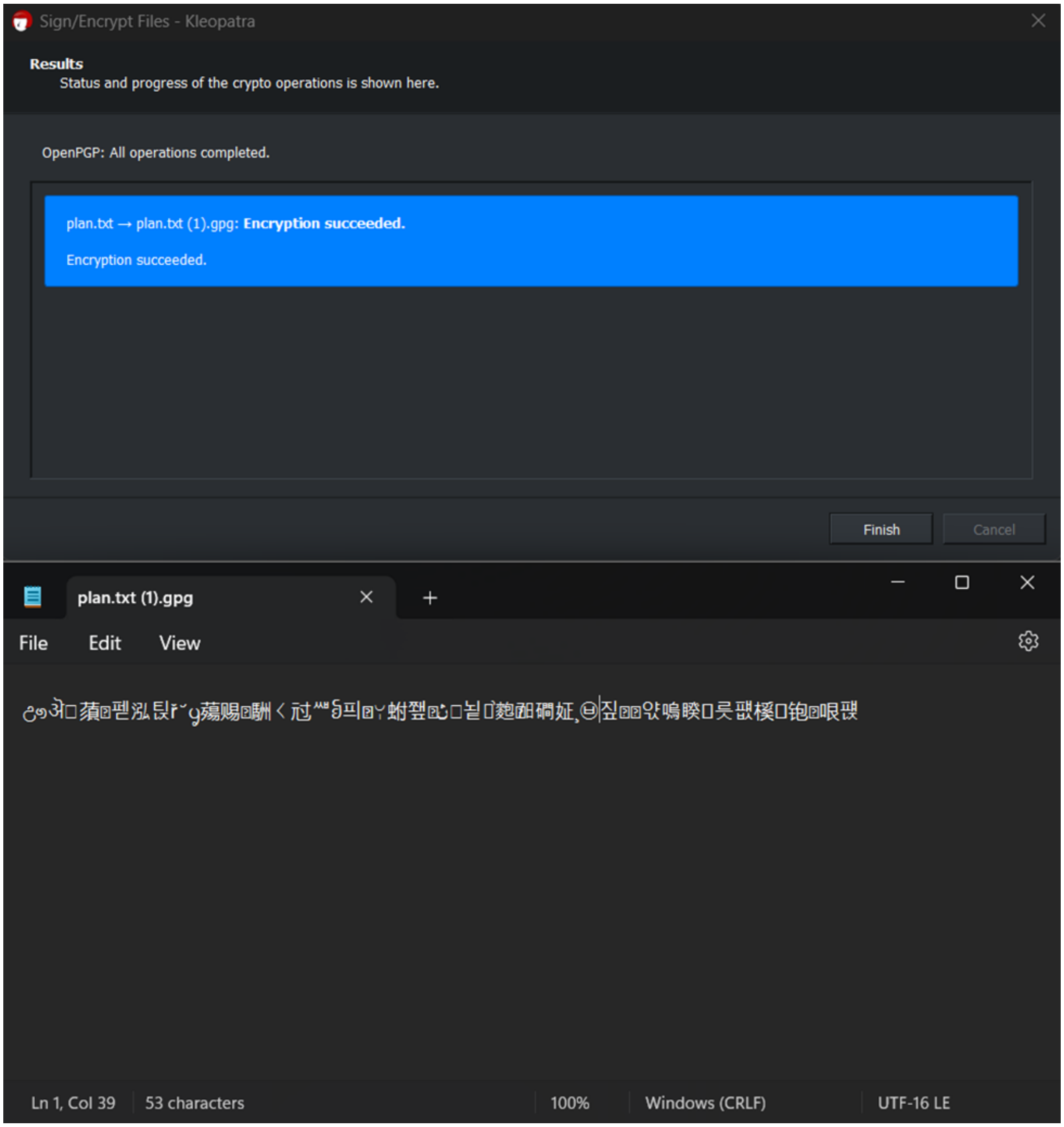
- **Shor's Algorithm**: This could break widely-used encryption algorithms like RSA by efficiently factoring large numbers.
- **Grover's Algorithm**: Offers quadratic speedup for unstructured search problems.
- **Molecular Simulations**: Quantum computers could accurately simulate complex molecular interactions, revolutionizing fields like drug discovery and materials science.

While there has been great progress, most experts believe that practical quantum computers that outperform classical computers in many tasks are still 10-20 years away. Challenges include maintaining quantum coherence, reducing error rates, and building scalable quantum systems.

Sources:

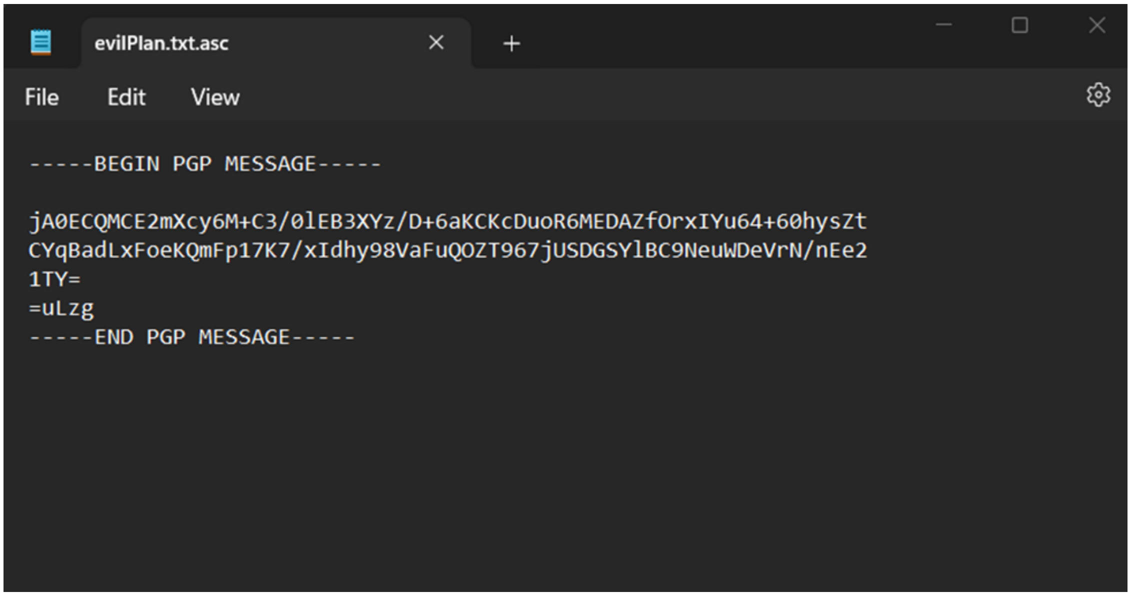
1. The Quantum Insider
2. AWS - Quantum Computing
3. IBM Quantum Computing Overview
4. Google Quantum Computing Overview

1.0.5 Symmetric Encryption

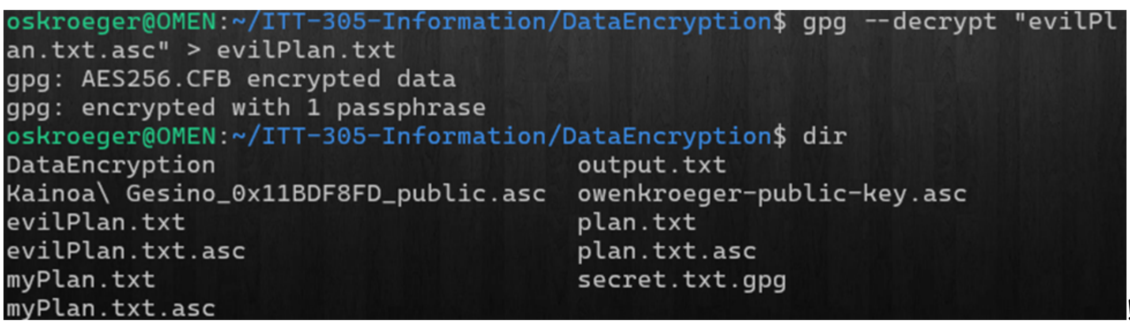


Caption: ↑ This is a screenshot of GNU PG / GPG being installed on my Windows system, and it being used to encrypt `plan.txt (1)` -> `plan.txt (1).gpg`. The second picture is the result of opening the encrypted file in Notepad.

Caption: ↑ This picture is from the command line, where I used `gpg` to encrypt `evilPlan.txt` -> `evilPlan.txt.asc`. We can see the command that was run to encrypt the file, along with the encrypted file located in the directory.

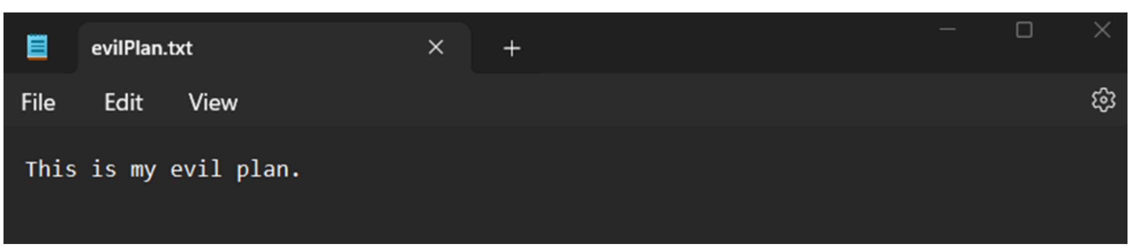


Caption: ↑ This picture shows that the encrypted file, `evilPlan.txt.asc`, was then opened in Notepad.



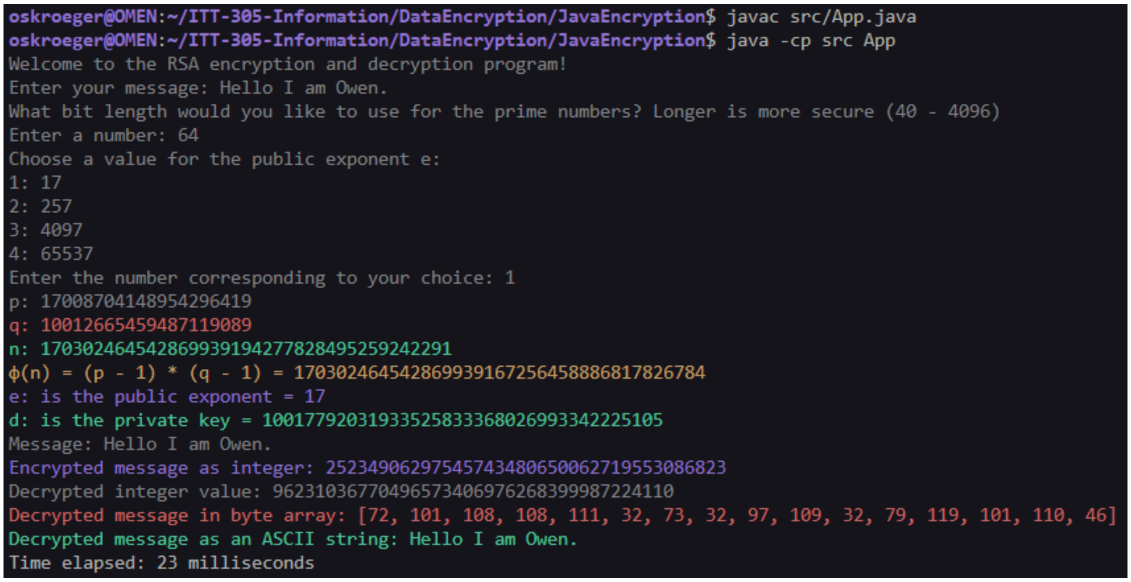
!

Caption: ↑ This picture shows the encrypted file, `evilPlan.txt.asc`, being decrypted using the command line.



Caption: ↑ This picture shows the decrypted file being opened with Notepad.

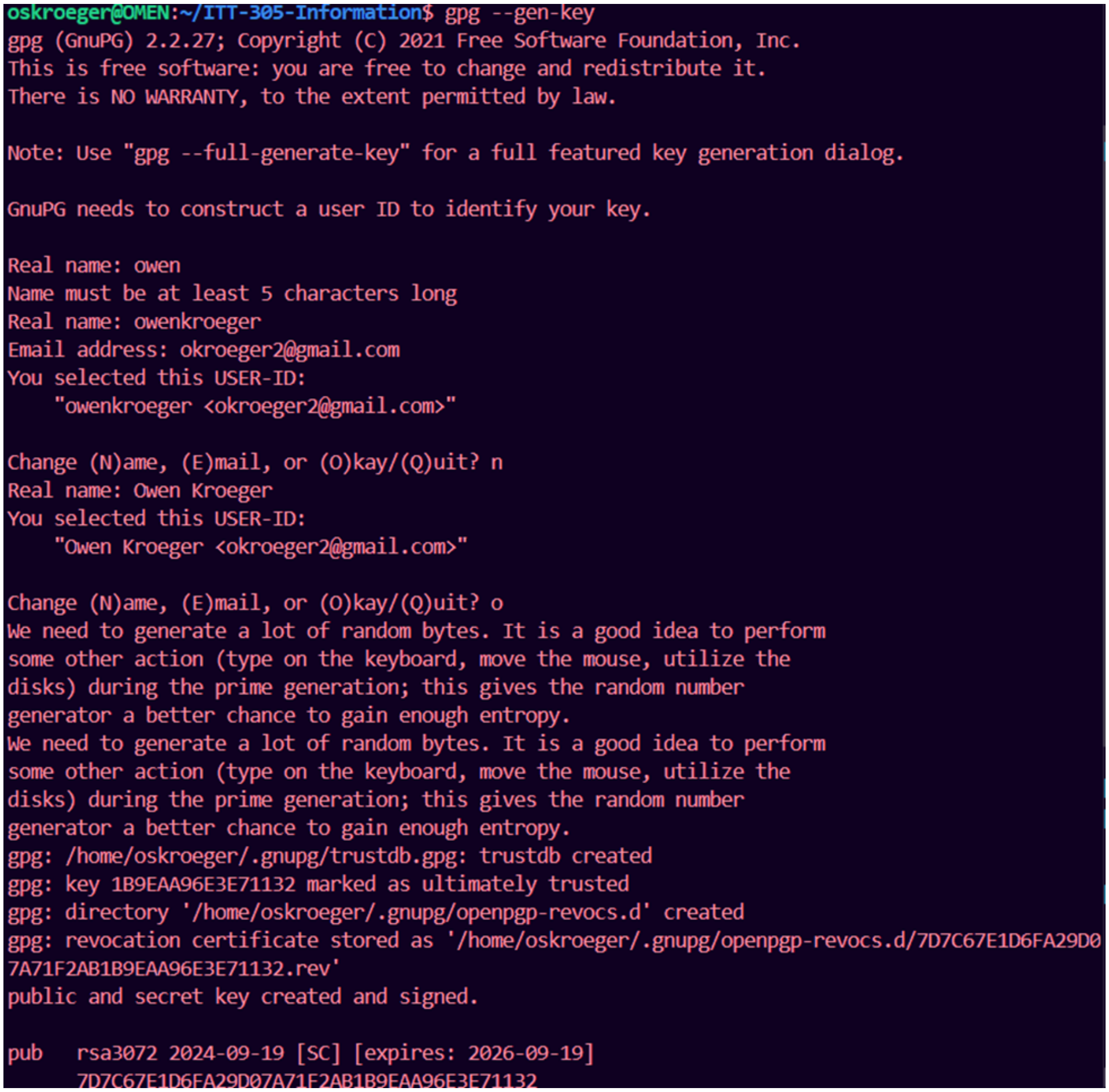
1.0.6 Java Encryption Program



Caption: ↑ This is the Java program running on the small message, "Hello I am Owen," with a small key value.

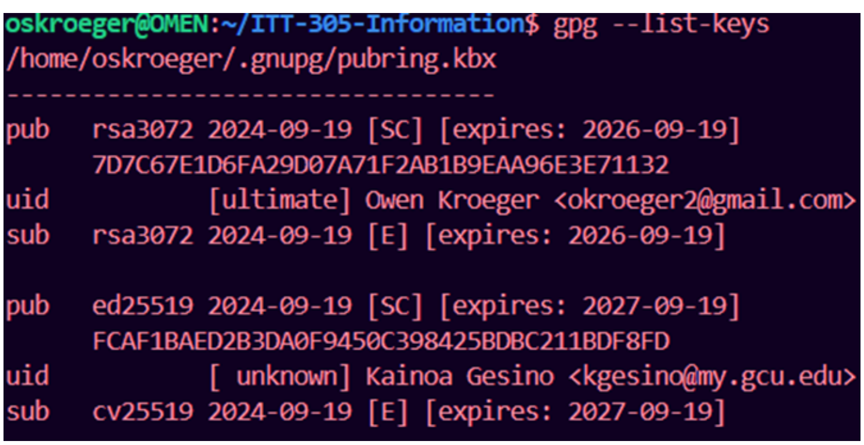
Caption: ↑ This is the same message being encrypted with a much larger key value, noting a large increase in the length of the encrypted message along with the time it took to encrypt the message (23ms $<$ 197ms).

1.0.7 Asymmetric Encryption



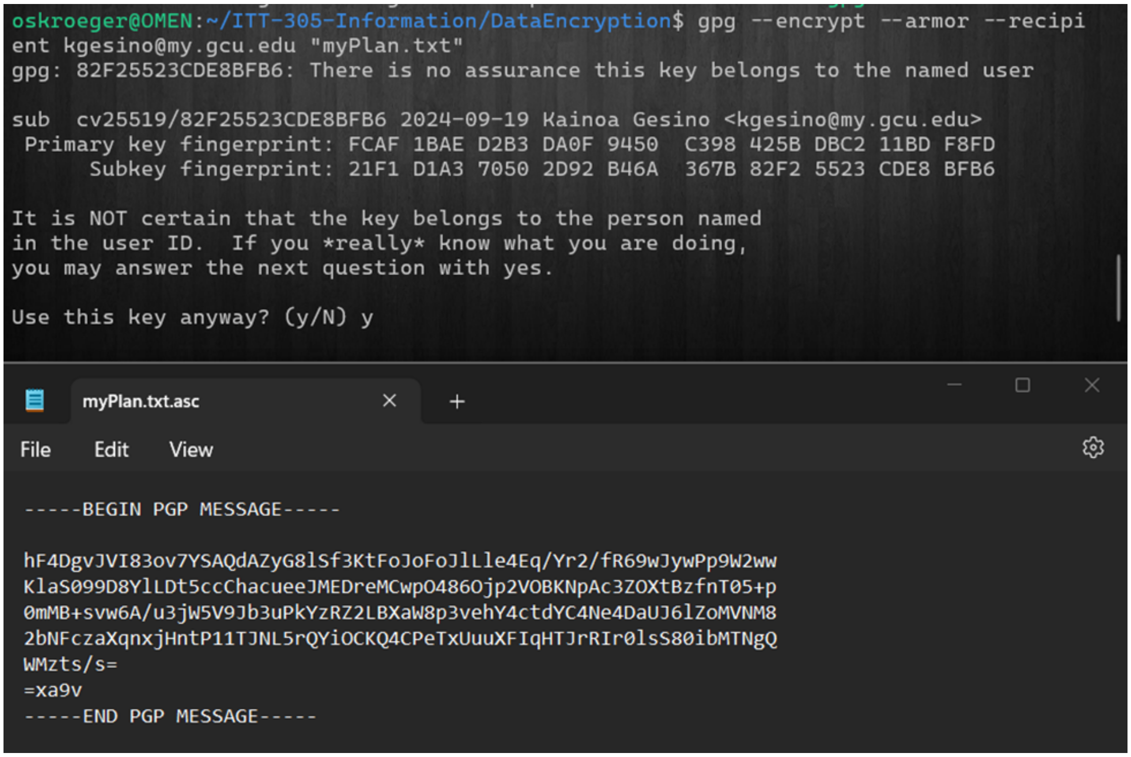
```
oskroeger@OMEN:~/ITT-305-Information$ gpg --gen-key
gpg (GnuPG) 2.2.27; Copyright (C) 2021 Free Software Foundation, Inc.
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

Note: Use "gpg --full-generate-key" for a full featured key generation dialog.

GnuPG needs to construct a user ID to identify your key.

Real name: owen
Name must be at least 5 characters long
Real name: owenkroeger
Email address: okroeger2@gmail.com
You selected this USER-ID:
    "owenkroeger <okroeger2@gmail.com>"

Change (N)ame, (E)mail, or (O)kay/(Q)uit? n
Real name: Owen Kroeger
You selected this USER-ID:
    "Owen Kroeger <okroeger2@gmail.com>"

Change (N)ame, (E)mail, or (O)kay/(Q)uit? o
We need to generate a lot of random bytes. It is a good idea to perform
some other action (type on the keyboard, move the mouse, utilize the
disks) during the prime generation; this gives the random number
generator a better chance to gain enough entropy.
We need to generate a lot of random bytes. It is a good idea to perform
some other action (type on the keyboard, move the mouse, utilize the
disks) during the prime generation; this gives the random number
generator a better chance to gain enough entropy.
gpg: /home/oskroeger/.gnupg/trustdb.gpg: trustdb created
gpg: key 1B9EAA96E3E71132 marked as ultimately trusted
gpg: directory '/home/oskroeger/.gnupg/openpgp-revocs.d' created
gpg: revocation certificate stored as '/home/oskroeger/.gnupg/openpgp-revocs.d/7D7C67E1D6FA29D0
7A71F2AB1B9EAA96E3E71132.rev'
public and secret key created and signed.

pub   rsa3072 2024-09-19 [SC] [expires: 2026-09-19]
      7D7C67E1D6FA29D07A71F2AB1B9EAA96E3E71132
```

Caption: ↑ This picture shows my key being generated for Asymmetric Encryption.

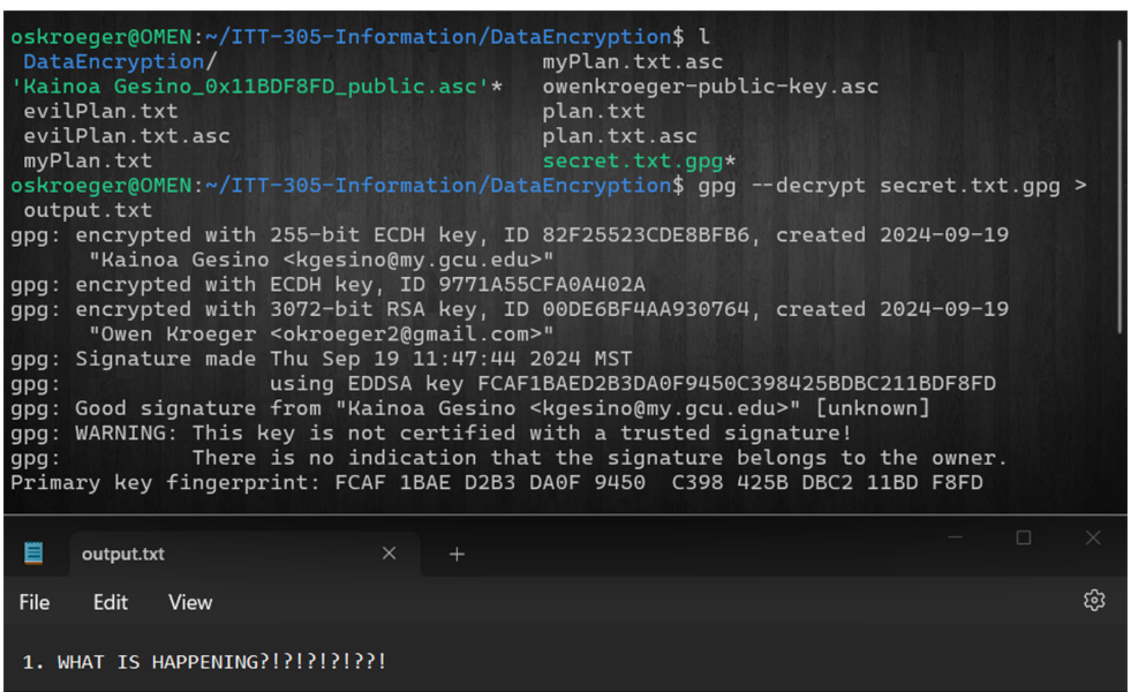


```
oskroeger@OMEN:~/ITT-305-Information$ gpg --list-keys
/home/oskroeger/.gnupg/pubring.kbx
----------------------------------
pub   rsa3072 2024-09-19 [SC] [expires: 2026-09-19]
      7D7C67E1D6FA29D07A71F2AB1B9EAA96E3E71132
uid           [ultimate] Owen Kroeger <okroeger2@gmail.com>
sub   rsa3072 2024-09-19 [E] [expires: 2026-09-19]

pub   ed25519 2024-09-19 [SC] [expires: 2027-09-19]
      FCAF1BAED2B3DA0F9450C398425BDBC211BDF8FD
uid           [ unknown] Kainoa Gesino <kgesino@my.gcu.edu>
sub   cv25519 2024-09-19 [E] [expires: 2027-09-19]
```

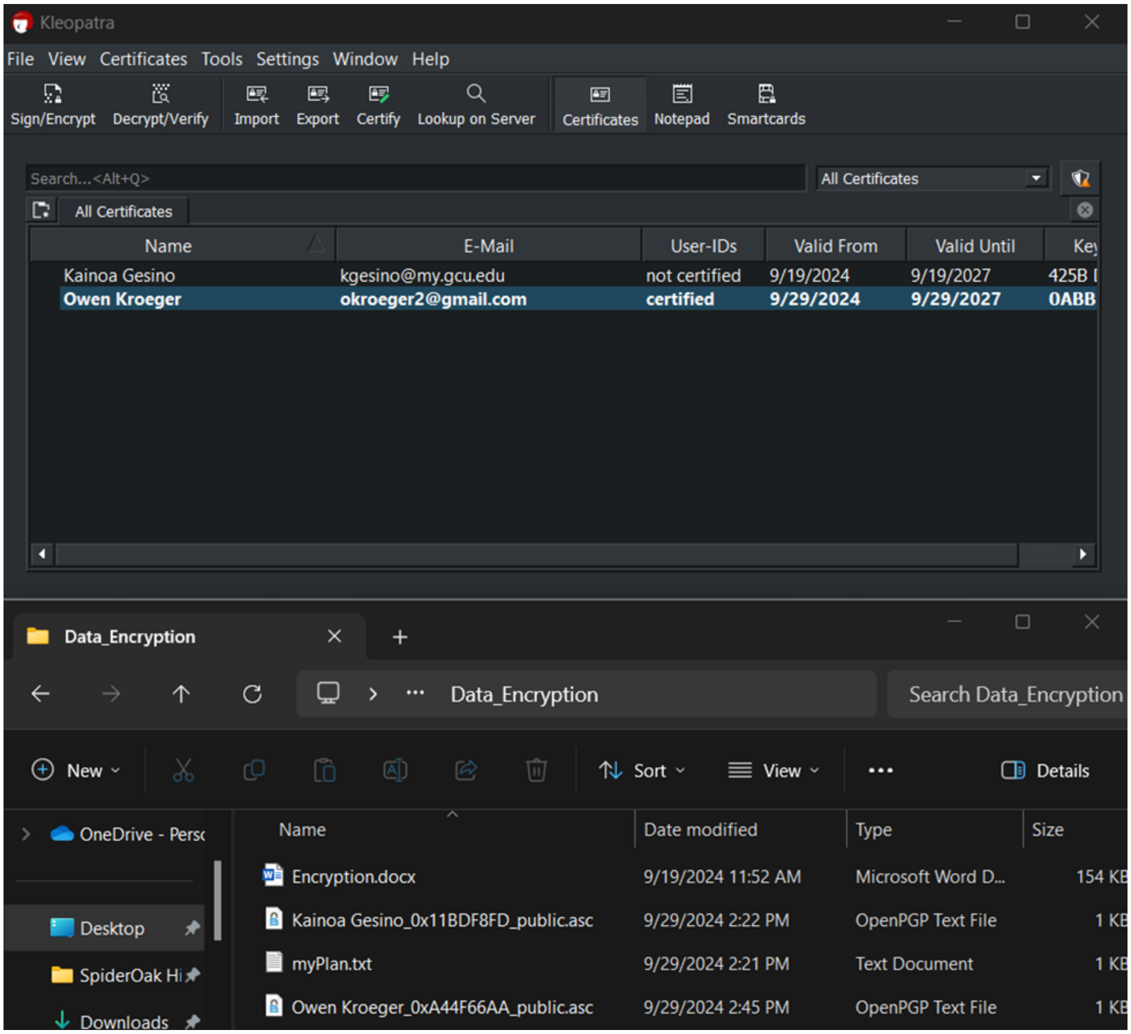
Caption: ↑ This picture shows my key in the list of public keys.



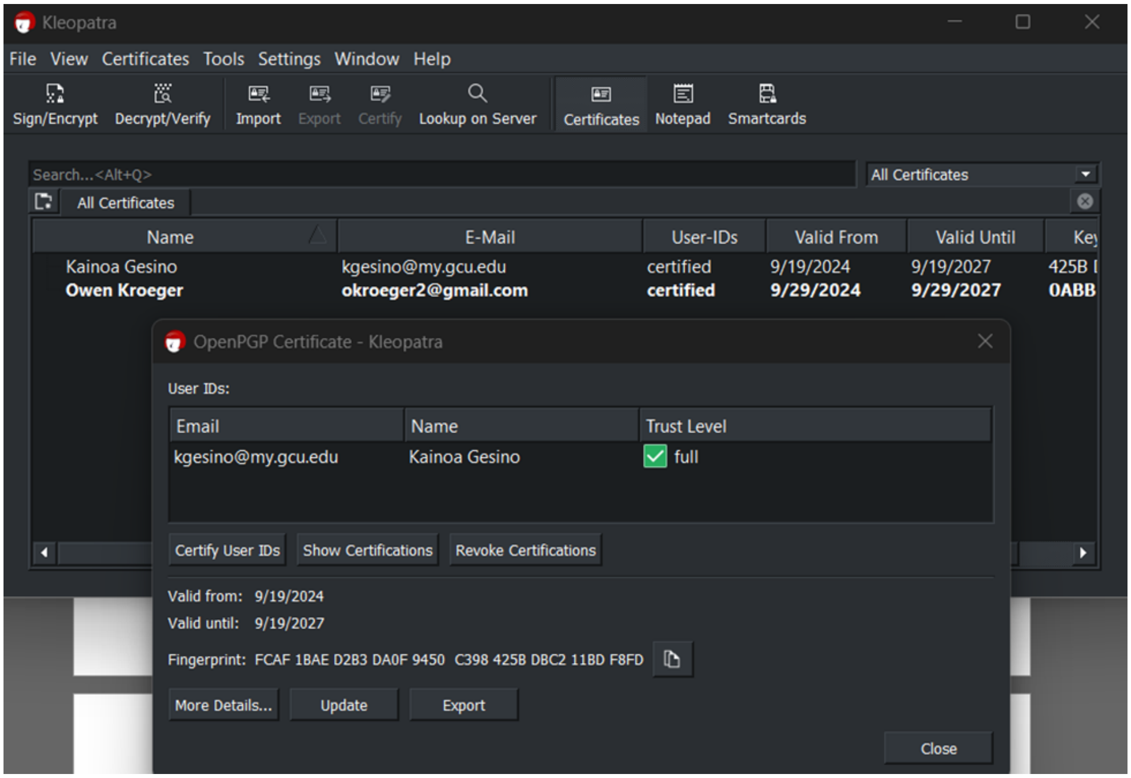
Caption: ↑ This picture shows the file `myPlan.txt` being encrypted for the recipient Kai Gesino from the command line, and then the resulting encrypted file being opened in Notepad.



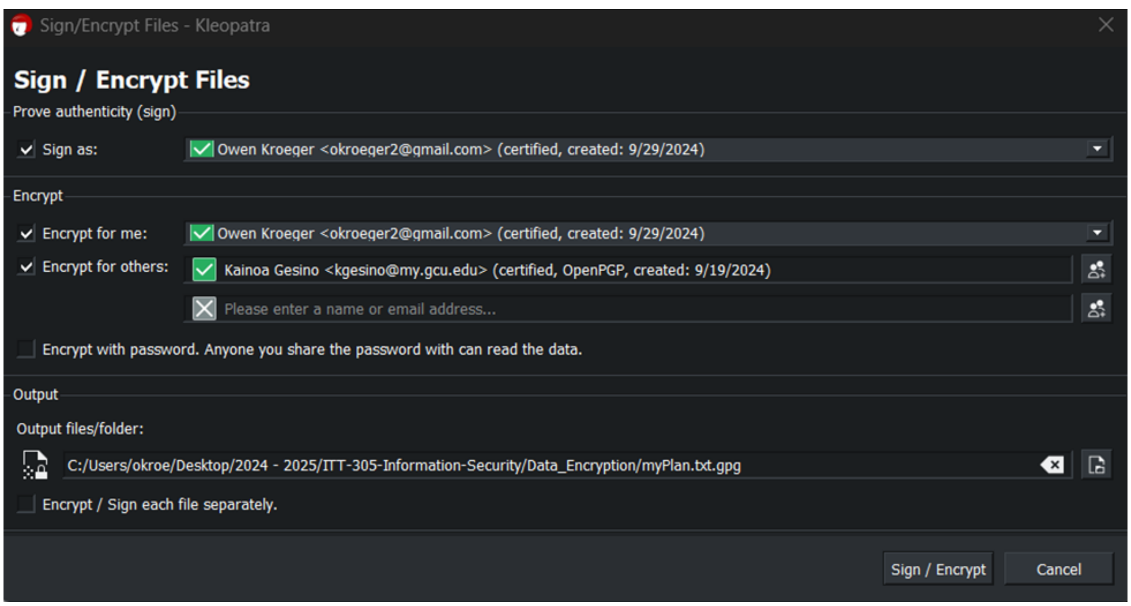
Caption: ↑ This picture shows me decrypting Kai Gesino's message to me, and writing it to `output.txt`. I then opened the decrypted file to see the private message in Notepad.



Caption: ↑ This picture shows me creating my public key using Kleopatra and saving it to a file in my folder.



Caption: ↑ This picture shows that I imported Kai's public key using Kleopatra.



Caption: ↑ This picture shows me encrypting `myPlan.txt` -> `myPlan.txt.gpg` with the intended target set for Kainoa Gesino using his public key that I imported.