EncryptEdge Labs

Cybersecurity Analyst Internship

Task Report

atalmamun@gmail.com

Task No: 08

Credit: Offensive Security

EncryptEdge Labs

Table of Contents

1.0 EncryptEdge Labs Internship Task Report

1.1 Introduction

Encryption is a critical aspect of modern cybersecurity, ensuring that sensitive data remains secure and accessible only to authorized individuals. It is a fundamental practice in protecting information from unauthorized access, data breaches, and cyber threats. This report explores the core principles of encryption, the differences between symmetric and asymmetric encryption, and their applications in real-world cybersecurity scenarios. Through theoretical understanding and hands-on practice with encryption tools such as OpenSSL and GPG, this task aims to enhance knowledge and practical skills in cryptographic techniques. Additionally, the completion of TryHackMe labs will reinforce the learning experience by providing real-world encryption scenarios and problem-solving exercises.

1.2 Objective

The primary objectives of this task are:

- To develop a foundational understanding of encryption concepts and their significance in cybersecurity.
- To differentiate between symmetric and asymmetric encryption, understanding their respective roles and applications.
- To explore real-world applications of encryption, including its role in securing email communications, file storage, and online transactions.
- To gain hands-on experience with encryption tools such as OpenSSL and GPG by encrypting and decrypting files and messages.
- To complete practical exercises through TryHackMe labs, reinforcing theoretical knowledge with applied cryptographic tasks.
- To document encryption processes, demonstrating proficiency in using cryptographic tools and understanding the security mechanisms they provide.

1.3 Requirements

To successfully complete this task, the following resources and tools are required:

- **Theoretical Knowledge**: Understanding of basic encryption concepts, including symmetric (AES, DES) and asymmetric (RSA) encryption algorithms, public/private key cryptography, and key exchange mechanisms.
- **Cryptographic Tools**:
  - OpenSSL: A widely used open-source tool for implementing encryption and security protocols.
  - GPG (GNU Privacy Guard): A tool for encrypting and signing files and messages to ensure confidentiality and authenticity.
- **Online Platforms**:
  - TryHackMe Labs: "Introduction to Cryptography" and "Encryption: Crypto 101" to provide hands-on learning experiences.
- **System Requirements**:
  - A computer with a compatible operating system (Windows, Linux, or macOS) capable of running OpenSSL and GPG.
  - Internet access for research, downloading required tools, and completing online labs.
- **Documentation Tools**:
  - A text editor or word processor for compiling the report.
  - Screenshot capture tools to document encryption and decryption processes.

By fulfilling these requirements, participants will be equipped to complete the task efficiently, gaining both theoretical and practical insights into cryptographic techniques and their significance in cybersecurity.

2.0 Theoretical Summary

Build a foundational understanding of encryption by covering key concepts and algorithms.

2.1 Encryption Basics

Encryption is the process of converting plaintext data into ciphertext to prevent unauthorized access. It plays a critical role in ensuring data confidentiality, integrity, and authenticity. Two main types of encryption are used in cybersecurity: symmetric encryption and asymmetric encryption.

2.2 Symmetric Encryption

Symmetric encryption uses a single key for both encryption and decryption. This method is efficient and fast, making it suitable for encrypting large amounts of data. However, the challenge lies in securely sharing the key between communicating parties. Common symmetric encryption algorithms include:

- **AES (Advanced Encryption Standard)**: A widely used encryption standard known for its security and efficiency.
- **DES (Data Encryption Standard)**: An older encryption standard, now largely replaced by AES due to its vulnerability to brute-force attacks.

2.3 Asymmetric Encryption

Asymmetric encryption uses a pair of keys: a public key for encryption and a private key for decryption. This method enhances security by eliminating the need to share a single key. It is commonly used in secure communication protocols and digital signatures. Key asymmetric encryption algorithms include:

- **RSA (Rivest-Shamir-Adleman)**: A widely used algorithm that relies on the difficulty of factoring large prime numbers.
- **ECC (Elliptic Curve Cryptography)**: A more efficient alternative to RSA, providing strong security with smaller key sizes.

2.4 Public/Private Key Pairs and Key Exchange Mechanisms

- **Public Key**: Shared openly and used for encryption.
- **Private Key**: Kept secret and used for decryption.
- **Key Exchange Mechanisms**: Secure methods like the **Diffie-Hellman key exchange** allow two parties to establish a shared secret key over an insecure channel.

2.5 Comparison of Symmetric and Asymmetric Encryption

Feature	Symmetric Encryption	Asymmetric Encryption
Key Usage	Same key for encryption and decryption	Public key for encryption, private key for decryption
Speed	Faster	Slower
Security	Requires secure key exchange	More secure, no need to share private key
Common Algorithms	AES, DES	RSA, ECC

2.6 Role in Cybersecurity

- **Symmetric Encryption** is commonly used for encrypting large data volumes, such as securing stored files and database records.
- **Asymmetric Encryption** is crucial for securing communications, including email encryption (GPG), digital certificates (SSL/TLS), and authentication mechanisms.

By understanding these encryption methods and their differences, cybersecurity professionals can apply them effectively to protect sensitive information and enhance digital security.

3.0 Explore Encryption Applications

Gain insight into how encryption is applied in real-world cybersecurity scenarios.

3.1 Common Encryption Applications

Encryption is widely used in various fields to protect data from unauthorized access and tampering. Some key applications include:

- **Email Security:** Encryption ensures that email communications remain private and tamper-proof. GPG (GNU Privacy Guard) is commonly used for encrypting and signing emails, providing confidentiality and authentication.
- **File Encryption:** Sensitive files are encrypted to prevent unauthorized access. Tools like OpenSSL and GPG are used to encrypt data before storing or sharing it.
- **Secure Communication Channels:** Encryption is essential for securing online communications. Protocols such as HTTPS use SSL/TLS encryption to protect data exchanged between users and websites, ensuring data integrity and authenticity.

3.2 Ensuring Confidentiality, Integrity, and Authenticity

Encryption plays a key role in cybersecurity by ensuring:

- **Confidentiality:** Data is accessible only to authorized users.
- **Integrity:** Encrypted data cannot be altered without detection.
- **Authenticity:** Digital signatures and certificates verify the sender's identity.

For example, secure messaging apps like Signal use end-to-end encryption (E2EE) to protect messages from interception, while online banking transactions rely on encryption to safeguard financial data.

3.3 Real-World Examples of Encryption Use

- **Financial Transactions:** Encryption protects sensitive data in online banking and payment systems, ensuring that credit card details and transactions remain secure.
- **Healthcare Data Protection:** Medical records are encrypted to comply with regulations like HIPAA, preventing unauthorized access to patient data.
- **Cloud Storage Security:** Encrypted cloud storage services, such as Google Drive and Dropbox, use encryption to protect user data from breaches and cyberattacks.

Encryption is a fundamental component of modern cybersecurity, enabling secure communication, data protection, and digital trust in various applications

4.0 Hands-On Encryption Practice

Use encryption tools to encrypt and decrypt files and messages.

4.1 Installing OpenSSL and GPG on Kali Linux

Kali Linux includes OpenSSL and GPG by default. Verify installation with:

openssl version

gpg --version

If missing, install them:

sudo apt update

sudo apt install openssl gnupg -y

4.2 Encrypting and Decrypting Files Using OpenSSL

Encrypting a File with AES-256

openssl enc -aes-256-cbc -salt -in message.txt -out encrypted_message.txt -pass pass:kali

Decrypting the File

openssl enc -aes-256-cbc -d -in encrypted_message.txt -out decrypted_message.txt -pass pass:kali

4.3 Encrypting and Decrypting Files Using RSA

Generating an RSA Key Pair

openssl genpkey -algorithm RSA -out private_key.pem

openssl rsa -pubout -in private_key.pem -out public_key.pem

Encrypting a File with the Public Key

openssl rsautl -encrypt -pubin -inkey public_key.pem -in message.txt -out encrypted_rsa.txt

Decrypting the File with the Private Key

openssl rsautl -decrypt -inkey private_key.pem -in encrypted_rsa.txt -out decrypted_rsa.txt

4.4 Encrypting, Signing, and Verifying Messages Using GPG

Generating a GPG Key Pair

gpg --full-generate-key

Encrypting a Message

echo "Confidential Data" | gpg --encrypt --armor --recipient "your_email@example.com" > message.gpg

Decrypting the Message

gpg --decrypt message.gpg

Signing a Message

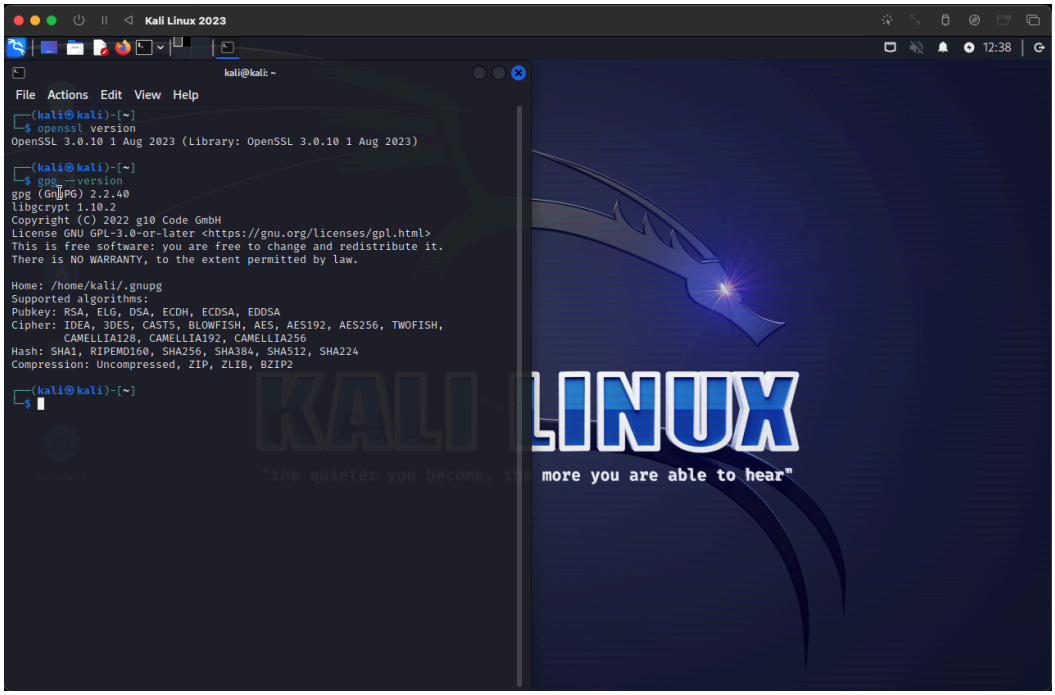
echo "This is my signed message." | gpg --clearsign > signed_message.gpg

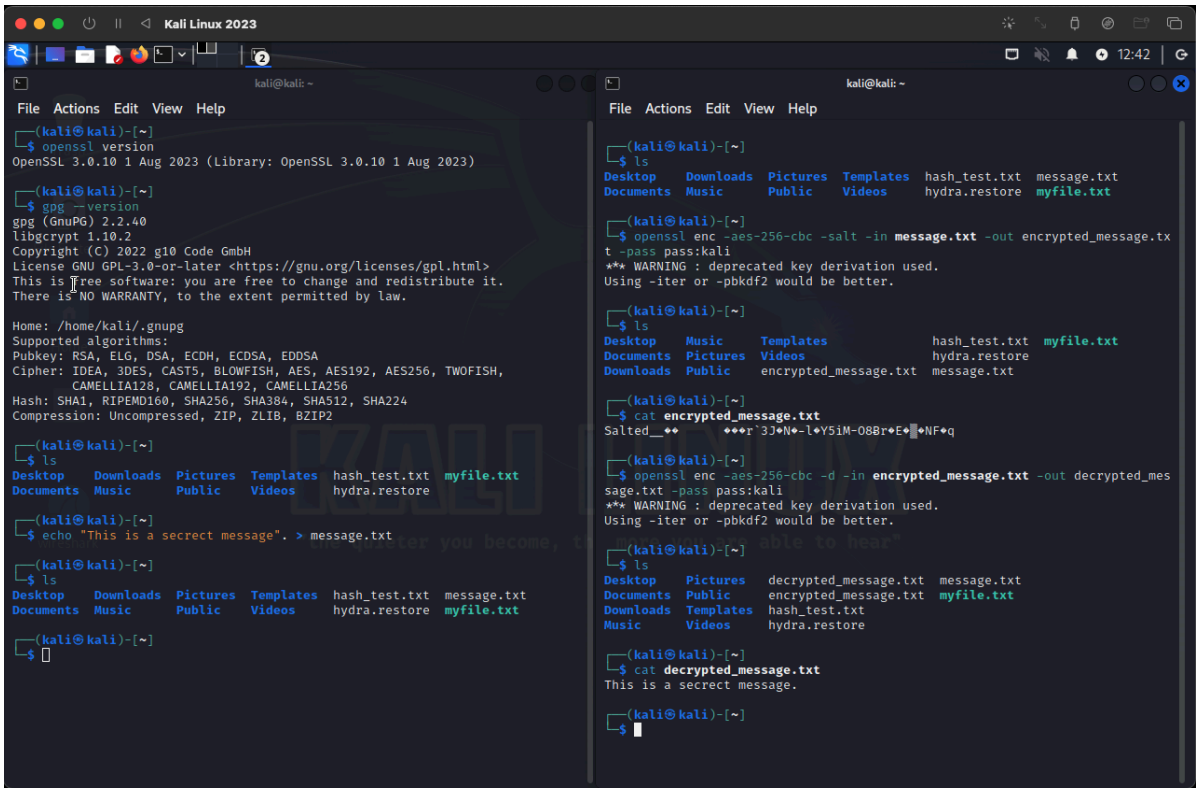
Verifying the Signature

gpg --verify signed_message.gpg

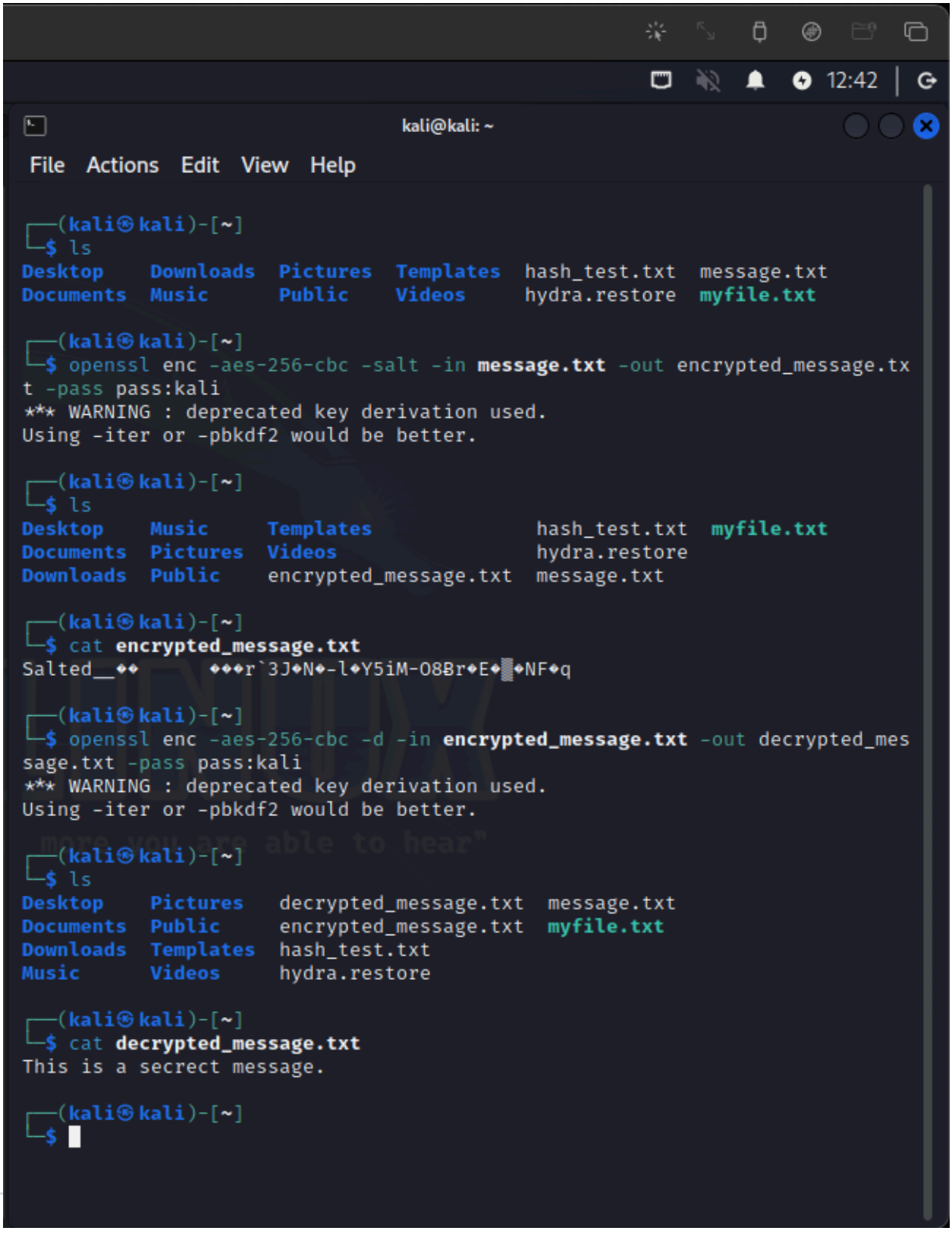
4.5 Documentation & Screenshots

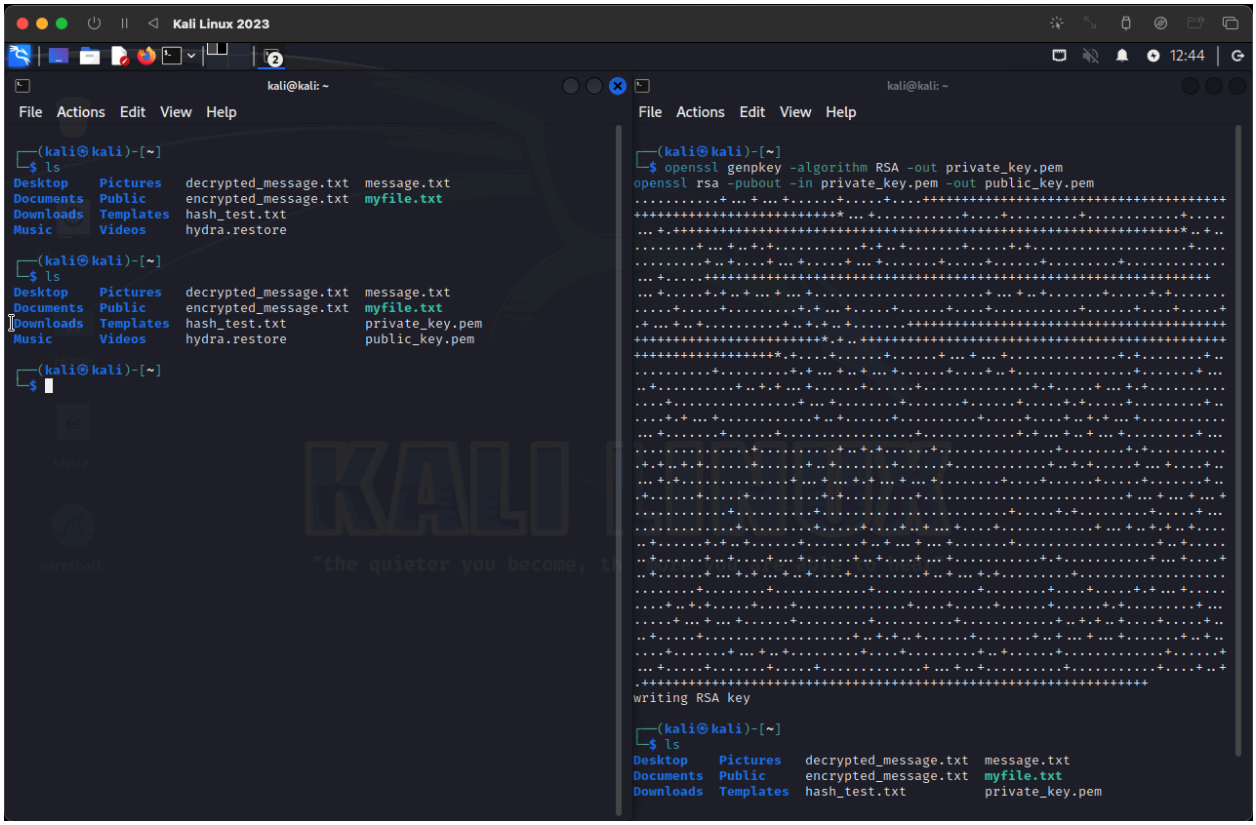
This hands-on practice reinforces encryption concepts by applying them in real-world scenarios.


Kali Linux 2023
12:38
kali@kali: ~
File Actions Edit View Help
(kali㉿kali)-[~]
$ openssl version
OpenSSL 3.0.10 1 Aug 2023 (Library: OpenSSL 3.0.10 1 Aug 2023)
(kali㉿kali)-[~]
$ gpg --version
gpg (GnuPG) 2.2.40
libgcrypt 1.10.2
Copyright (C) 2022 g10 Code GmbH
License GNU GPL-3.0-or-later <https://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Home: /home/kali/.gnupg
Supported algorithms:
Pubkey: RSA, ELG, DSA, ECDH, ECDSA, EDDSA
Cipher: IDEA, 3DES, CAST5, BLOWFISH, AES, AES192, AES256, TWOFISH,
CAMELLIA128, CAMELLIA192, CAMELLIA256
Hash: SHA1, RIPEMD160, SHA256, SHA384, SHA512, SHA224
Compression: Uncompressed, ZIP, ZLIB, BZIP2
(kali㉿kali)-[~]
$
LINUX
more you are able to hear"

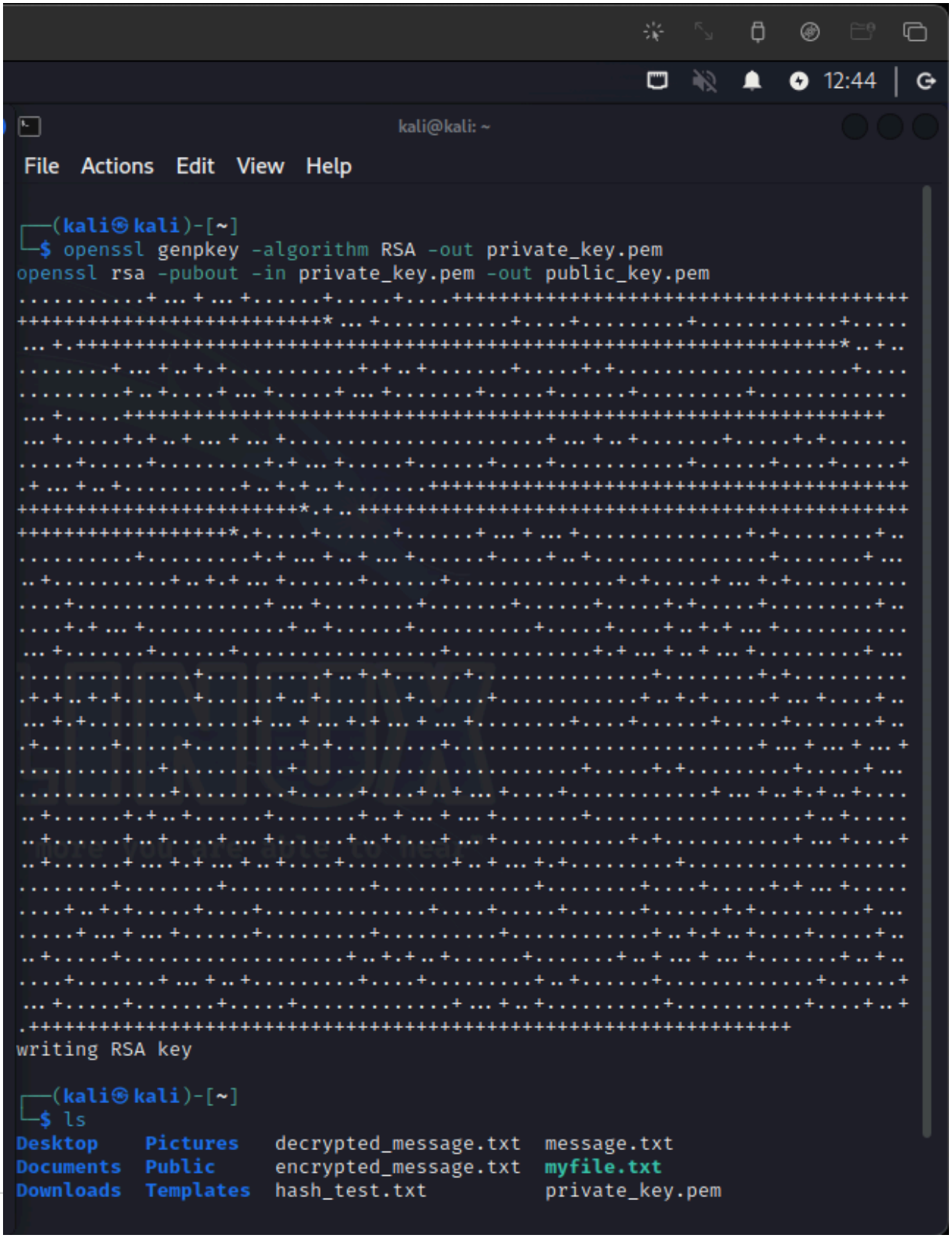

Kali Linux 2023
12:42
kali@kali: ~
File Actions Edit View Help
(kali㉿kali)-[~]
$ openssl version
OpenSSL 3.0.10 1 Aug 2023 (Library: OpenSSL 3.0.10 1 Aug 2023)
(kali㉿kali)-[~]
$ gpg --version
gpg (GnuPG) 2.2.40
libgcrypt 1.10.2
Copyright (C) 2022 g10 Code GmbH
License GNU GPL-3.0-or-later <https://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Home: /home/kali/.gnupg
Supported algorithms:
Pubkey: RSA, ELG, DSA, ECDH, ECDSA, EDDSA
Cipher: IDEA, 3DES, CAST5, BLOWFISH, AES, AES192, AES256, TWOFISH,
CAMELLIA128, CAMELLIA192, CAMELLIA256
Hash: SHA1, RIPEMD160, SHA256, SHA384, SHA512, SHA224
Compression: Uncompressed, ZIP, ZLIB, BZIP2
(kali㉿kali)-[~]
$ ls
Desktop Downloads Pictures Templates hash_test.txt myfile.txt
Documents Music Public Videos hydra.restore
(kali㉿kali)-[~]
$ echo "This is a secrect message". > message.txt
(kali㉿kali)-[~]
$ ls
Desktop Downloads Pictures Templates hash_test.txt message.txt
Documents Music Public Videos hydra.restore myfile.txt
(kali㉿kali)-[~]
$
kali@kali: ~
File Actions Edit View Help
(kali㉿kali)-[~]
$ ls
Desktop Downloads Pictures Templates hash_test.txt message.txt
Documents Music Public Videos hydra.restore myfile.txt
(kali㉿kali)-[~]
$ openssl enc -aes-256-cbc -salt -in message.txt -out encrypted_message.tx
t -pass pass:kali
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
(kali㉿kali)-[~]
$ ls
Desktop Music Templates hash_test.txt myfile.txt
Documents Pictures Videos hydra.restore
Downloads Public encrypted_message.txt message.txt
(kali㉿kali)-[~]
$ cat encrypted_message.txt
Salted__◆◆ ◆◆◆r`3J◆N◆-l◆Y5iM-O8Br◆E◆◆NF◆q
(kali㉿kali)-[~]
$ openssl enc -aes-256-cbc -d -in encrypted_message.txt -out decrypted_mes
sage.txt -pass pass:kali
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
(kali㉿kali)-[~]
$ ls
Desktop Pictures decrypted_message.txt message.txt
Documents Public encrypted_message.txt myfile.txt
Downloads Templates hash_test.txt
Music Videos hydra.restore
(kali㉿kali)-[~]
$ cat decrypted_message.txt
This is a secrect message.
(kali㉿kali)-[~]
$

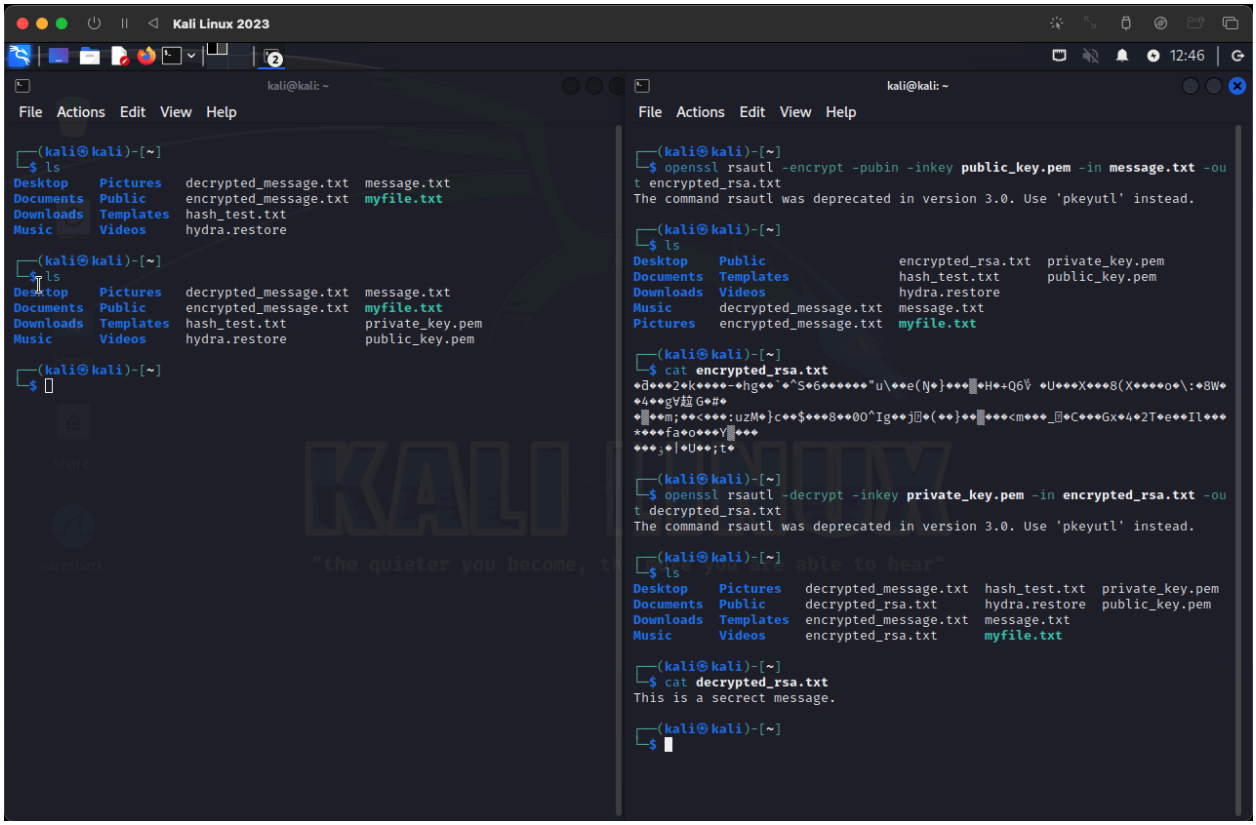

EncryptEdge Labs

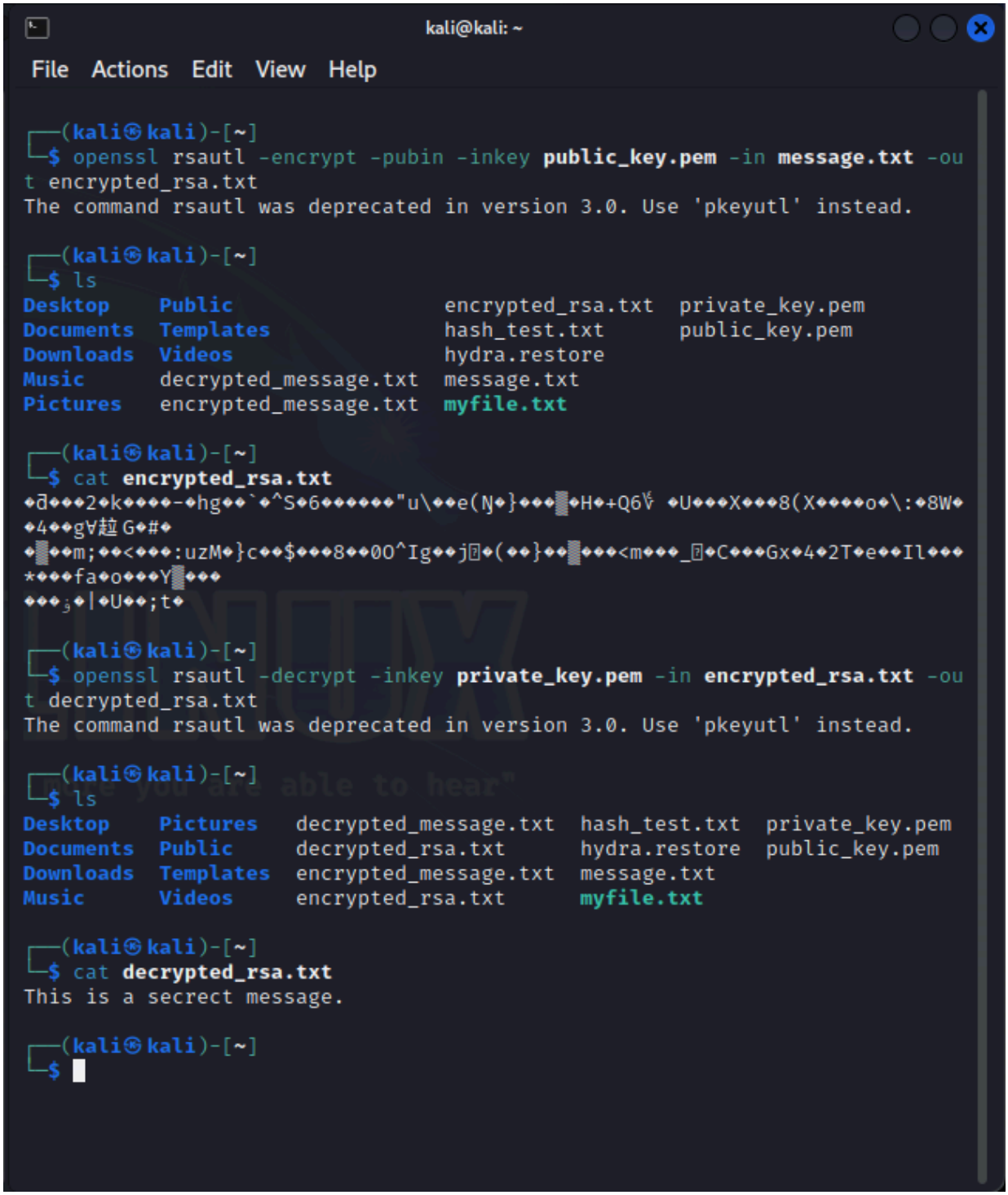

12:42
kali@kali: ~
File Actions Edit View Help
┌──(kali㉿kali)-[~]
└─$ ls
Desktop Downloads Pictures Templates hash_test.txt message.txt
Documents Music Public Videos hydra.restore myfile.txt
┌──(kali㉿kali)-[~]
└─$ openssl enc -aes-256-cbc -salt -in message.txt -out encrypted_message.tx
t -pass pass:kali
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
┌──(kali㉿kali)-[~]
└─$ ls
Desktop Music Templates hash_test.txt myfile.txt
Documents Pictures Videos hydra.restore
Downloads Public encrypted_message.txt message.txt
┌──(kali㉿kali)-[~]
└─$ cat encrypted_message.txt
Salted__◆◆ ◆◆◆r`3J◆N◆-l◆Y5iM-O8Br◆E◆ ◆NF◆q
┌──(kali㉿kali)-[~]
└─$ openssl enc -aes-256-cbc -d -in encrypted_message.txt -out decrypted_mes
sage.txt -pass pass:kali
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
┌──(kali㉿kali)-[~]
└─$ ls
Desktop Pictures decrypted_message.txt message.txt
Documents Public encrypted_message.txt myfile.txt
Downloads Templates hash_test.txt
Music Videos hydra.restore
┌──(kali㉿kali)-[~]
└─$ cat decrypted_message.txt
This is a secrect message.
┌──(kali㉿kali)-[~]
└─$


Kali Linux 2023
12:44
kali@kali: ~
File Actions Edit View Help
(kali㉿kali)-[~]
$ ls
Desktop Pictures decrypted_message.txt message.txt
Documents Public encrypted_message.txt myfile.txt
Downloads Templates hash_test.txt
Music Videos hydra.restore
(kali㉿kali)-[~]
$ ls
Desktop Pictures decrypted_message.txt message.txt
Documents Public encrypted_message.txt myfile.txt
Downloads Templates hash_test.txt private_key.pem
Music Videos hydra.restore public_key.pem
(kali㉿kali)-[~]
$
kali@kali: ~
File Actions Edit View Help
(kali㉿kali)-[~]
$ openssl genpkey -algorithm RSA -out private_key.pem
openssl rsa -pubout -in private_key.pem -out public_key.pem
writing RSA key
(kali㉿kali)-[~]
$ ls
Desktop Pictures decrypted_message.txt message.txt
Documents Public encrypted_message.txt myfile.txt
Downloads Templates hash_test.txt private_key.pem

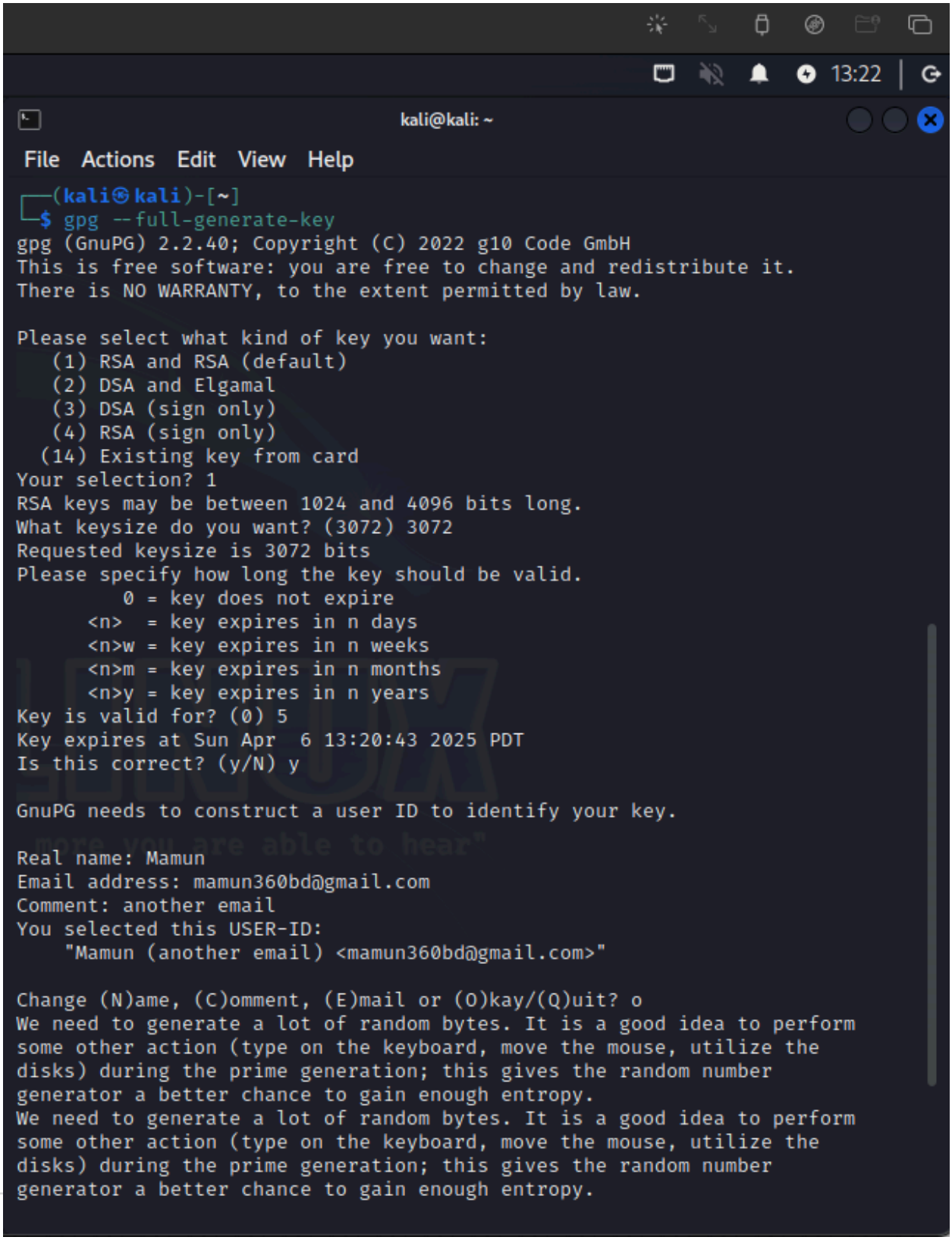
EncryptEdge Labs

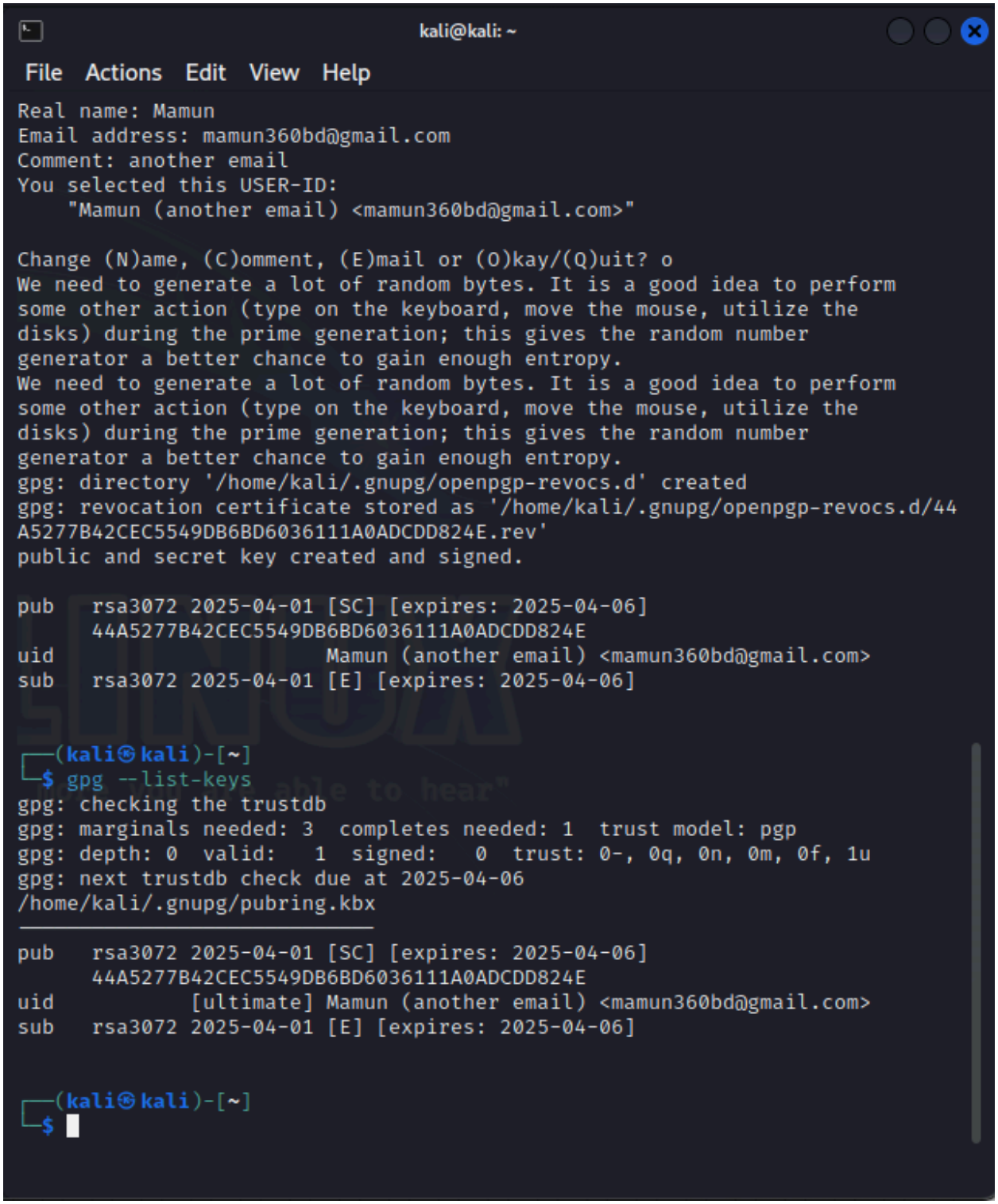


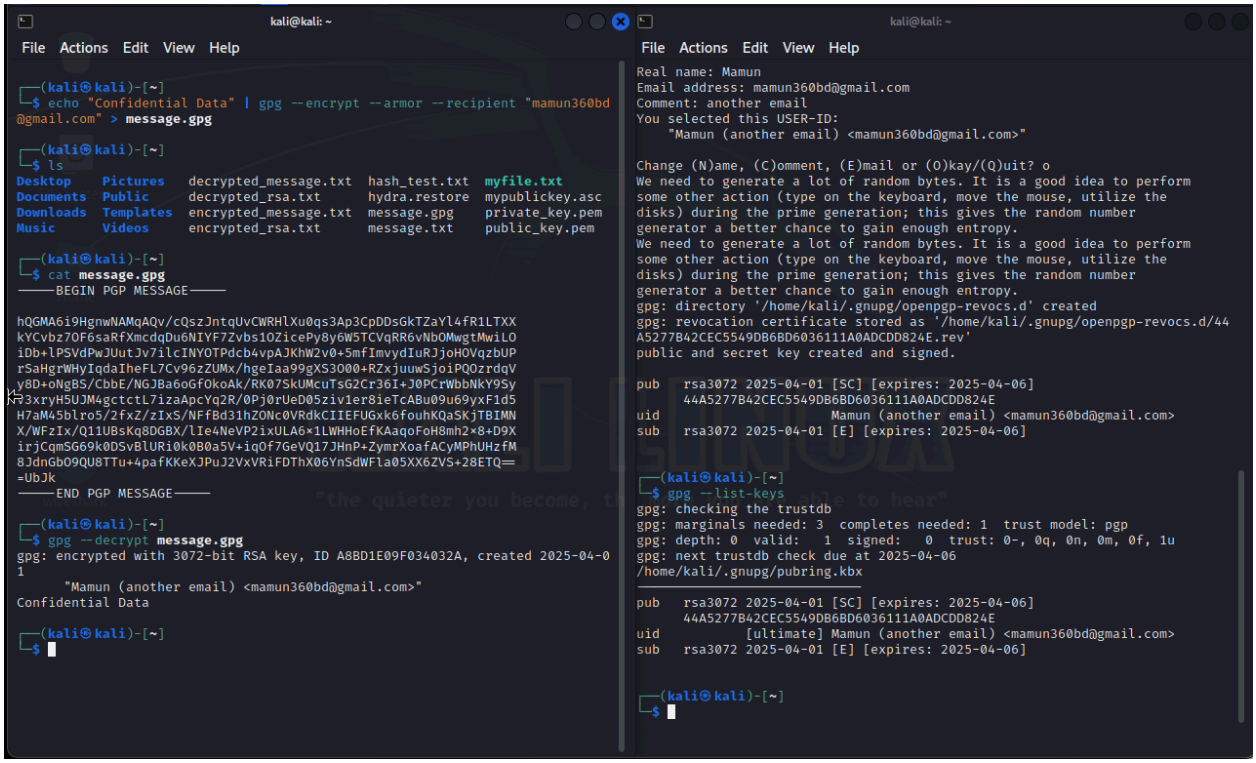

Kali Linux 2023
12:46
kali@kali: ~
File Actions Edit View Help
(kali㉿kali)-[~]
$ ls
Desktop Pictures decrypted_message.txt message.txt
Documents Public encrypted_message.txt myfile.txt
Downloads Templates hash_test.txt
Music Videos hydra.restore
(kali㉿kali)-[~]
$ ls
Desktop Pictures decrypted_message.txt message.txt
Documents Public encrypted_message.txt myfile.txt
Downloads Templates hash_test.txt private_key.pem
Music Videos hydra.restore public_key.pem
(kali㉿kali)-[~]
$
kali@kali: ~
File Actions Edit View Help
(kali㉿kali)-[~]
$ openssl rsautl -encrypt -pubin -inkey public_key.pem -in message.txt -out encrypted_rsa.txt
The command rsautl was deprecated in version 3.0. Use 'pkeyutl' instead.
(kali㉿kali)-[~]
$ ls
Desktop Public encrypted_rsa.txt private_key.pem
Documents Templates hash_test.txt public_key.pem
Downloads Videos hydra.restore
Music decrypted_message.txt message.txt
Pictures encrypted_message.txt myfile.txt
(kali㉿kali)-[~]
$ cat encrypted_rsa.txt
(kali㉿kali)-[~]
$ openssl rsautl -decrypt -inkey private_key.pem -in encrypted_rsa.txt -out decrypted_rsa.txt
The command rsautl was deprecated in version 3.0. Use 'pkeyutl' instead.
(kali㉿kali)-[~]
$ ls
Desktop Pictures decrypted_message.txt hash_test.txt private_key.pem
Documents Public decrypted_rsa.txt hydra.restore public_key.pem
Downloads Templates encrypted_message.txt message.txt
Music Videos encrypted_rsa.txt myfile.txt
(kali㉿kali)-[~]
$ cat decrypted_rsa.txt
This is a secrect message.
(kali㉿kali)-[~]
$

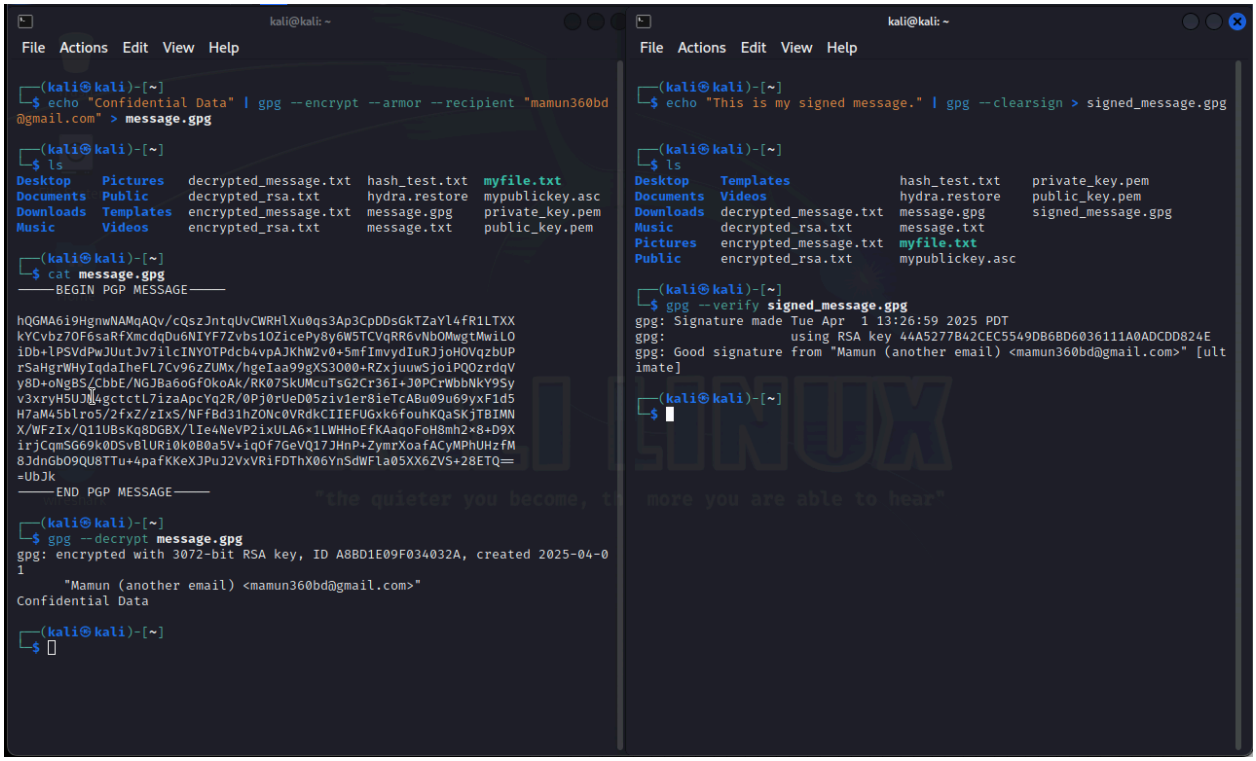

kali@kali: ~
File Actions Edit View Help
┌──(kali㉿kali)-[~]
└─$ openssl rsautl -encrypt -pubin -inkey public_key.pem -in message.txt -ou
t encrypted_rsa.txt
The command rsautl was deprecated in version 3.0. Use 'pkeyutl' instead.
┌──(kali㉿kali)-[~]
└─$ ls
Desktop Public encrypted_rsa.txt private_key.pem
Documents Templates hash_test.txt public_key.pem
Downloads Videos hydra.restore
Music decrypted_message.txt message.txt
Pictures encrypted_message.txt myfile.txt
┌──(kali㉿kali)-[~]
└─$ cat encrypted_rsa.txt
┌──(kali㉿kali)-[~]
└─$ openssl rsautl -decrypt -inkey private_key.pem -in encrypted_rsa.txt -ou
t decrypted_rsa.txt
The command rsautl was deprecated in version 3.0. Use 'pkeyutl' instead.
┌──(kali㉿kali)-[~]
└─$ ls
Desktop Pictures decrypted_message.txt hash_test.txt private_key.pem
Documents Public decrypted_rsa.txt hydra.restore public_key.pem
Downloads Templates encrypted_message.txt message.txt
Music Videos encrypted_rsa.txt myfile.txt
┌──(kali㉿kali)-[~]
└─$ cat decrypted_rsa.txt
This is a secrect message.
┌──(kali㉿kali)-[~]
└─$

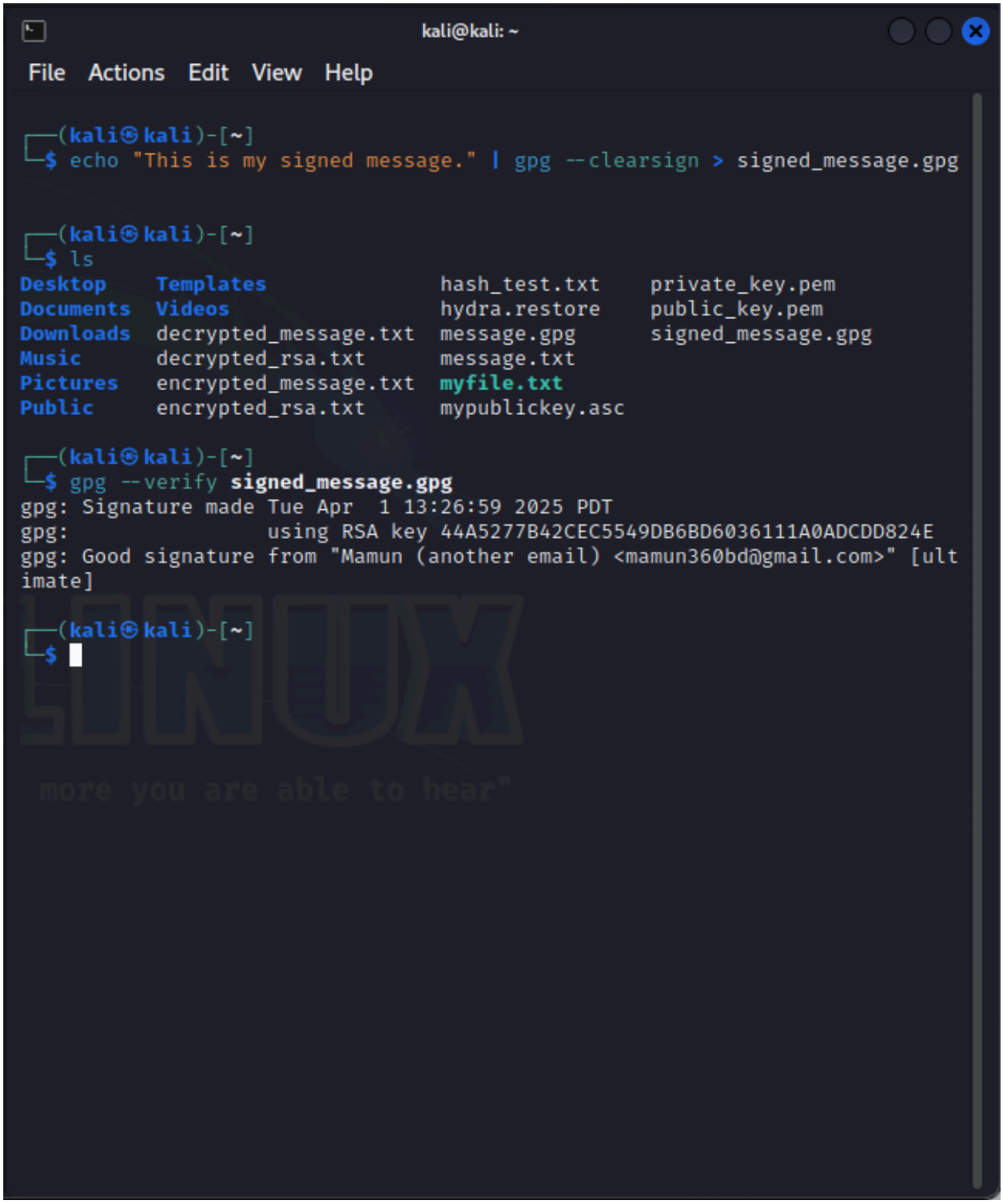

EncryptEdge Labs


13:22
kali@kali: ~
File Actions Edit View Help
┌──(kali㉿kali)-[~]
└─$ gpg --full-generate-key
gpg (GnuPG) 2.2.40; Copyright (C) 2022 g10 Code GmbH
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Please select what kind of key you want:
(1) RSA and RSA (default)
(2) DSA and Elgamal
(3) DSA (sign only)
(4) RSA (sign only)
(14) Existing key from card
Your selection? 1
RSA keys may be between 1024 and 4096 bits long.
What keysize do you want? (3072) 3072
Requested keysize is 3072 bits
Please specify how long the key should be valid.
0 = key does not expire
<n> = key expires in n days
<n>w = key expires in n weeks
<n>m = key expires in n months
<n>y = key expires in n years
Key is valid for? (0) 5
Key expires at Sun Apr 6 13:20:43 2025 PDT
Is this correct? (y/N) y
GnuPG needs to construct a user ID to identify your key.
Real name: Mamun
Email address: mamun360bd@gmail.com
Comment: another email
You selected this USER-ID:
"Mamun (another email) <mamun360bd@gmail.com>"
Change (N)ame, (C)omment, (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.


kali@kali: ~
File Actions Edit View Help
Real name: Mamun
Email address: mamun360bd@gmail.com
Comment: another email
You selected this USER-ID:
"Mamun (another email) <mamun360bd@gmail.com>"
Change (N)ame, (C)omment, (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: directory '/home/kali/.gnupg/openpgp-revocs.d' created
gpg: revocation certificate stored as '/home/kali/.gnupg/openpgp-revocs.d/44A5277B42CEC5549DB6BD6036111A0ADCDD824E.rev'
public and secret key created and signed.
pub rsa3072 2025-04-01 [SC] [expires: 2025-04-06]
44A5277B42CEC5549DB6BD6036111A0ADCDD824E
uid Mamun (another email) <mamun360bd@gmail.com>
sub rsa3072 2025-04-01 [E] [expires: 2025-04-06]
(kali㉿kali)-[~]
$ gpg --list-keys
gpg: checking the trustdb
gpg: marginals needed: 3 completes needed: 1 trust model: pgp
gpg: depth: 0 valid: 1 signed: 0 trust: 0-, 0q, 0n, 0m, 0f, 1u
gpg: next trustdb check due at 2025-04-06
/home/kali/.gnupg/pubring.kbx
pub rsa3072 2025-04-01 [SC] [expires: 2025-04-06]
44A5277B42CEC5549DB6BD6036111A0ADCDD824E
uid [ultimate] Mamun (another email) <mamun360bd@gmail.com>
sub rsa3072 2025-04-01 [E] [expires: 2025-04-06]
(kali㉿kali)-[~]
$


kali@kali: ~
File Actions Edit View Help
┌──(kali㉿kali)-[~]
└─$ echo "Confidential Data" | gpg --encrypt --armor --recipient "mamun360bd@gmail.com" > message.gpg
┌──(kali㉿kali)-[~]
└─$ ls
Desktop Pictures decrypted_message.txt hash_test.txt myfile.txt
Documents Public decrypted_rsa.txt hydra.restore mypublickey.asc
Downloads Templates encrypted_message.txt message.gpg private_key.pem
Music Videos encrypted_rsa.txt message.txt public_key.pem
┌──(kali㉿kali)-[~]
└─$ cat message.gpg
-----BEGIN PGP MESSAGE-----
hQGMA6i9HgnwNAMqAQv/cQszJntqUvCWRHlXu0qs3Ap3CpDDsGkTZaYl4fR1LTXX
kYCvbz7OF6saRfXmcdqDu6NIYF7Zvbs1OZicePy8y6W5TCVqRR6vNbOMwgtMwiLO
iDb+lPSVdPwJUutJv7ilcINYOTPdcb4vpAJKhW2v0+5mfImvydIuRJjoHOVqzbUP
rSaHgrWHyIqdaIheFL7Cv96zZUMx/hgeIaa99gXS3O00+RZxjuuwSjoiPQOzrdqV
y8D+oNgBS/CbbE/NGJBa6oGfOkoAk/RK07SkUMcuTsG2Cr36I+J0PCrWbbNkY9Sy
H73xryH5UJM4gctctL7izaApcYq2R/0Pj0rUeD05ziv1er8ieTcABu9u69yxF1d5
H7aM45blro5/2fxZ/zIxS/NFfBd31hZONc0VRdkCIIEFUGxk6fouhKQaSKjTBIMN
X/WFzIx/Q11UBsKq8DGBX/lIe4NeVP2ixULA6×1LWHHoEfKAaqoFoH8mh2×8+D9X
irjCqmSG69k0DSvBlURi0k0B0a5V+iqOf7GeVQ17JHnP+ZymrXoafACyMPhUHzfM
8JdnGbO9QU8TTu+4pafKKeXJPuJ2VxVRiFDThX06YnSdWFla05XX6ZVS+28ETQ==
=UbJk
-----END PGP MESSAGE-----
┌──(kali㉿kali)-[~]
└─$ gpg --decrypt message.gpg
gpg: encrypted with 3072-bit RSA key, ID A8BD1E09F034032A, created 2025-04-01
"Mamun (another email) <mamun360bd@gmail.com>"
Confidential Data
┌──(kali㉿kali)-[~]
└─$
kali@kali: ~
File Actions Edit View Help
Real name: Mamun
Email address: mamun360bd@gmail.com
Comment: another email
You selected this USER-ID:
"Mamun (another email) <mamun360bd@gmail.com>"
Change (N)ame, (C)omment, (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: directory '/home/kali/.gnupg/openpgp-revocs.d' created
gpg: revocation certificate stored as '/home/kali/.gnupg/openpgp-revocs.d/44A5277B42CEC5549DB6BD6036111A0ADCDD824E.rev'
public and secret key created and signed.
pub rsa3072 2025-04-01 [SC] [expires: 2025-04-06]
44A5277B42CEC5549DB6BD6036111A0ADCDD824E
uid Mamun (another email) <mamun360bd@gmail.com>
sub rsa3072 2025-04-01 [E] [expires: 2025-04-06]
┌──(kali㉿kali)-[~]
└─$ gpg --list-keys
gpg: checking the trustdb
gpg: marginals needed: 3 completes needed: 1 trust model: pgp
gpg: depth: 0 valid: 1 signed: 0 trust: 0-, 0q, 0n, 0m, 0f, 1u
gpg: next trustdb check due at 2025-04-06
/home/kali/.gnupg/pubring.kbx
pub rsa3072 2025-04-01 [SC] [expires: 2025-04-06]
44A5277B42CEC5549DB6BD6036111A0ADCDD824E
uid [ultimate] Mamun (another email) <mamun360bd@gmail.com>
sub rsa3072 2025-04-01 [E] [expires: 2025-04-06]
┌──(kali㉿kali)-[~]
└─$


kali@kali: ~
File Actions Edit View Help
(kali㉿kali)-[~]
$ echo "Confidential Data" | gpg --encrypt --armor --recipient "mamun360bd@gmail.com" > message.gpg
$ ls
Desktop Pictures decrypted_message.txt hash_test.txt myfile.txt
Documents Public decrypted_rsa.txt hydra.restore mypublickey.asc
Downloads Templates encrypted_message.txt message.gpg private_key.pem
Music Videos encrypted_rsa.txt message.txt public_key.pem
$ cat message.gpg
-----BEGIN PGP MESSAGE-----
hQGMA6i9HgnwNAMqAQv/cQszJntqUvCWRHlXu0qs3Ap3CpDDsGkTZaYl4fR1LTXX
kYCvbz7OF6saRfXmcdqDu6NIYF7Zvbs1OZicePy8y6W5TCVqRR6vNbOMwgtMwiLO
iDb+lPSVdPwJUutJv7ilcINYOTPdcb4vpAJKhW2v0+5mfImvydIuRJjoHOVqzbUP
rSaHgrWHyIqdaIheFL7Cv96zZUMx/hgeIaa99gXS3O00+RZxjuuwSjoiPQOzrdqV
y8D+oNgBS/CbbE/NGJBa6oGfOkoAk/RK07SkUMcuTsG2Cr36I+J0PCrWbbNkY9Sy
v3xryH5UJM4gctctL7izaApcYq2R/0Pj0rUeD05ziv1er8ieTcABu09u69yxF1d5
H7aM45blro5/2fxZ/zIxS/NFfBd31hZONc0VRdkCIIEFUGxk6fouhKQaSKjTBIMN
X/WFzIx/Q11UBsKq8DGBX/lIe4NeVP2ixULA6×1LWHHoEfKAaqoFoH8mh2×8+D9X
irjCqmSG69k0DSvBlURi0k0B0a5V+iqOf7GeVQ17JHnP+ZymrXoafACyMPhUHzfM
8JdnGbO9QU8TTu+4pafKKeXJPuJ2VxVRiFDThX06YnSdWFla05XX6ZVS+28ETQ==
=UbJk
-----END PGP MESSAGE-----
$ gpg --decrypt message.gpg
gpg: encrypted with 3072-bit RSA key, ID A8BD1E09F034032A, created 2025-04-01
"Mamun (another email) <mamun360bd@gmail.com>"
Confidential Data
kali@kali: ~
File Actions Edit View Help
$ echo "This is my signed message." | gpg --clearsign > signed_message.gpg
$ ls
Desktop Templates hash_test.txt private_key.pem
Documents Videos hydra.restore public_key.pem
Downloads decrypted_message.txt message.gpg signed_message.gpg
Music decrypted_rsa.txt message.txt
Pictures encrypted_message.txt myfile.txt
Public encrypted_rsa.txt mypublickey.asc
$ gpg --verify signed_message.gpg
gpg: Signature made Tue Apr 1 13:26:59 2025 PDT
gpg: using RSA key 44A5277B42CEC5549DB6BD6036111A0ADCDD824E
gpg: Good signature from "Mamun (another email) <mamun360bd@gmail.com>" [ultimate]


kali@kali: ~
File Actions Edit View Help
┌──(kali㉿kali)-[~]
└─$ echo "This is my signed message." | gpg --clearsign > signed_message.gpg
┌──(kali㉿kali)-[~]
└─$ ls
Desktop Templates hash_test.txt private_key.pem
Documents Videos hydra.restore public_key.pem
Downloads decrypted_message.txt message.gpg signed_message.gpg
Music decrypted_rsa.txt message.txt
Pictures encrypted_message.txt myfile.txt
Public encrypted_rsa.txt mypublickey.asc
┌──(kali㉿kali)-[~]
└─$ gpg --verify signed_message.gpg
gpg: Signature made Tue Apr 1 13:26:59 2025 PDT
gpg: using RSA key 44A5277B42CEC5549DB6BD6036111A0ADCDD824E
gpg: Good signature from "Mamun (another email) <mamun360bd@gmail.com>" [ultimate]
┌──(kali㉿kali)-[~]
└─$

5.0 Hands-on Labs

In this section, you will be completing two hands-on labs from TryHackMe that will help you develop a deeper understanding of cryptography concepts and encryption techniques. These labs will provide practical knowledge and hands-on experience, reinforcing the theoretical concepts you've learned.

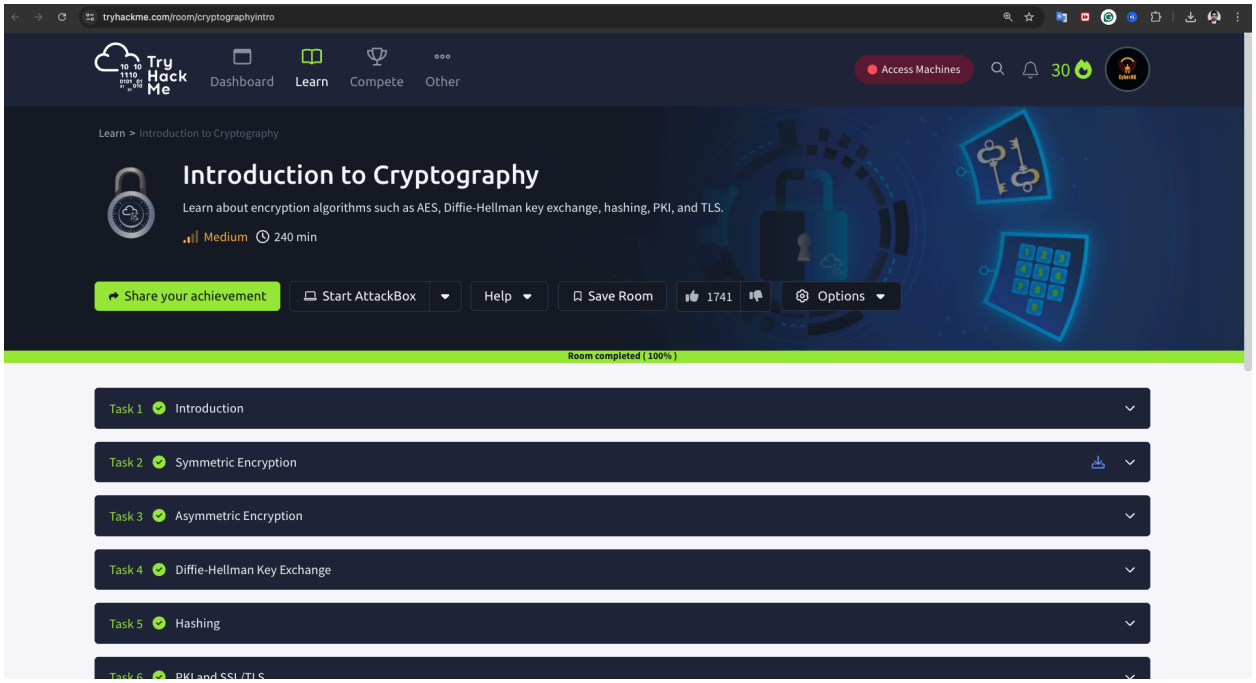
5.1 TryHackMe Lab 1: Introduction to Cryptography

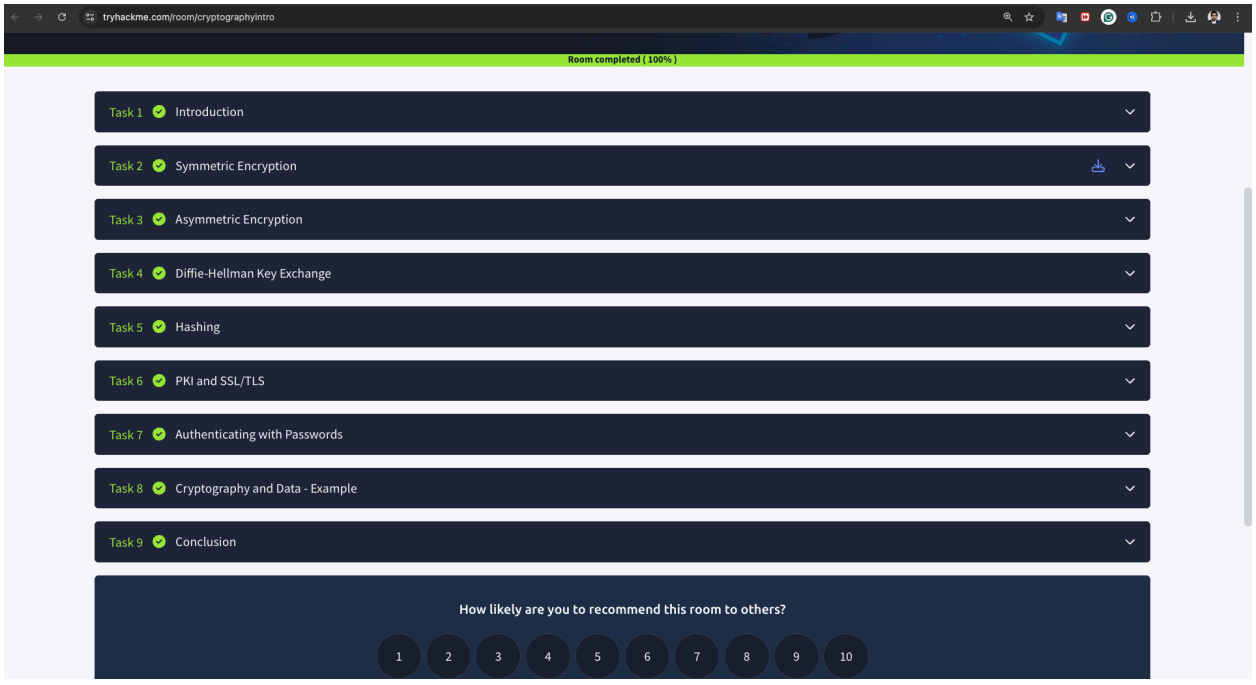
Key Topics Covered:

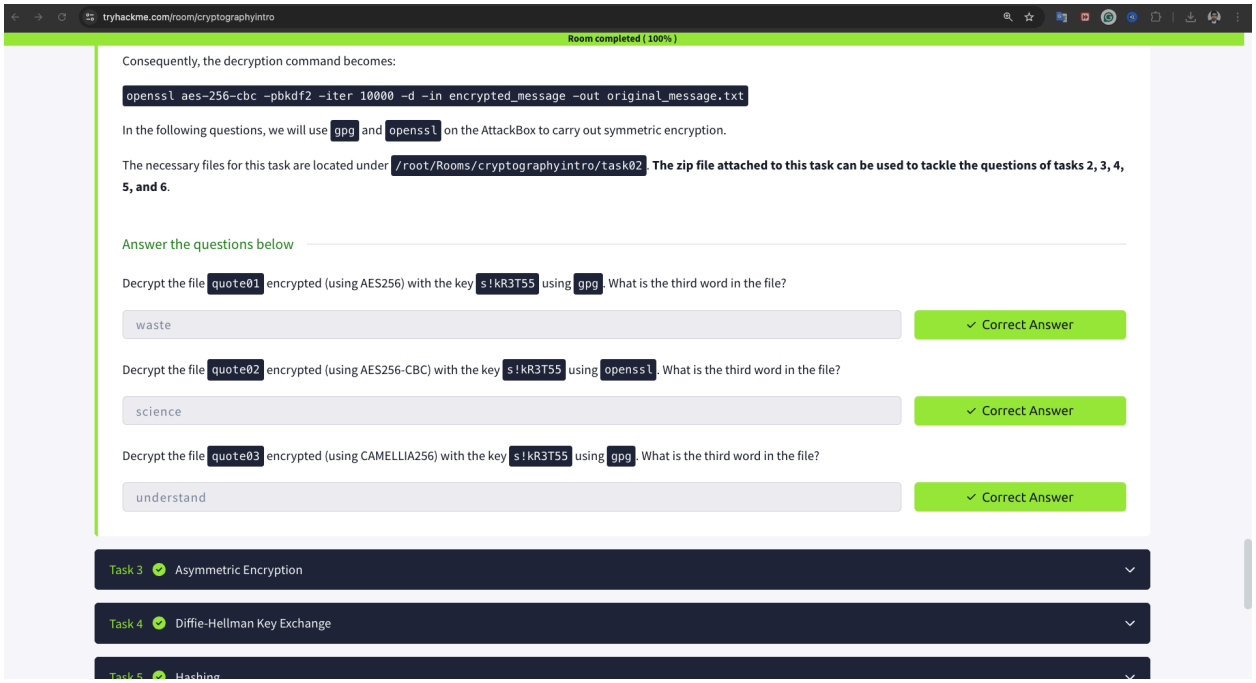
- **Basic Cryptography Concepts**: Understand the fundamental principles behind encryption, such as how encryption and decryption processes work, the role of keys, and different encryption algorithms.

- **Encryption Methods**: Learn about the different types of encryption (symmetric and asymmetric encryption), their use cases, and how they contribute to data protection.

Screenshot of the Completion Screen:


tryhackme.com/room/cryptographyintro
Try Hack Me
Dashboard
Learn
Compete
Other
Access Machines
30
Learn > Introduction to Cryptography
Introduction to Cryptography
Learn about encryption algorithms such as AES, Diffie-Hellman key exchange, hashing, PKI, and TLS.
Medium
240 min
Share your achievement
Start AttackBox
Help
Save Room
1741
Options
Room completed (100%)
Task 1 Introduction
Task 2 Symmetric Encryption
Task 3 Asymmetric Encryption
Task 4 Diffie-Hellman Key Exchange
Task 5 Hashing


tryhackme.com/room/cryptographyintro
Room completed (100%)
Task 1 Introduction
Task 2 Symmetric Encryption
Task 3 Asymmetric Encryption
Task 4 Diffie-Hellman Key Exchange
Task 5 Hashing
Task 6 PKI and SSL/TLS
Task 7 Authenticating with Passwords
Task 8 Cryptography and Data - Example
Task 9 Conclusion
How likely are you to recommend this room to others?
1 2 3 4 5 6 7 8 9 10



5.2 TryHackMe Lab 2: Encryption: Crypto 101

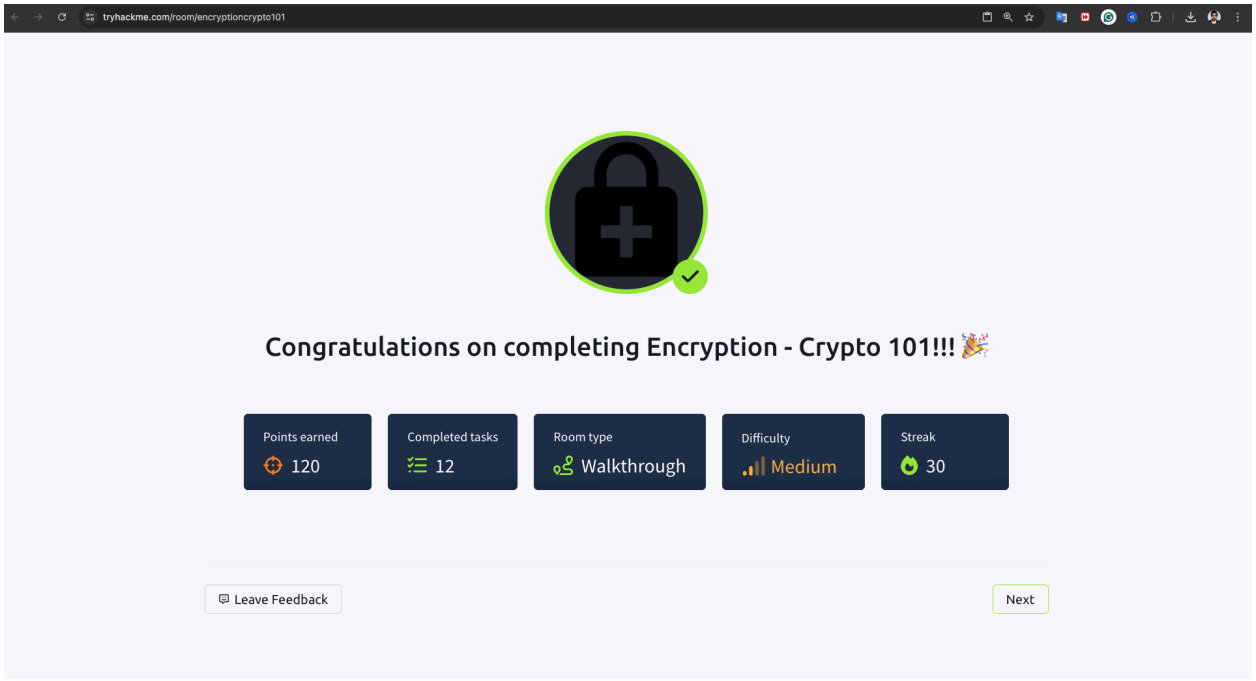
Key Topics Covered:

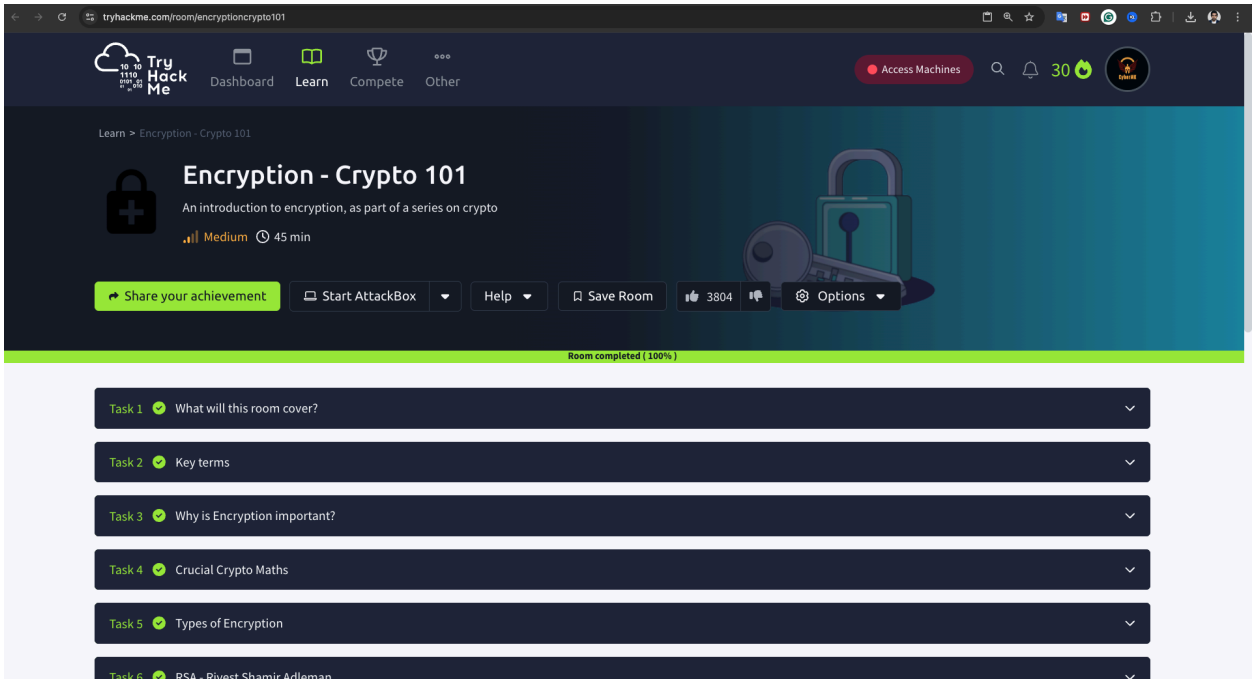
- **In-depth Encryption Techniques**: This lab dives deeper into encryption, exploring how different encryption algorithms (like AES, RSA, and others) work in practice.

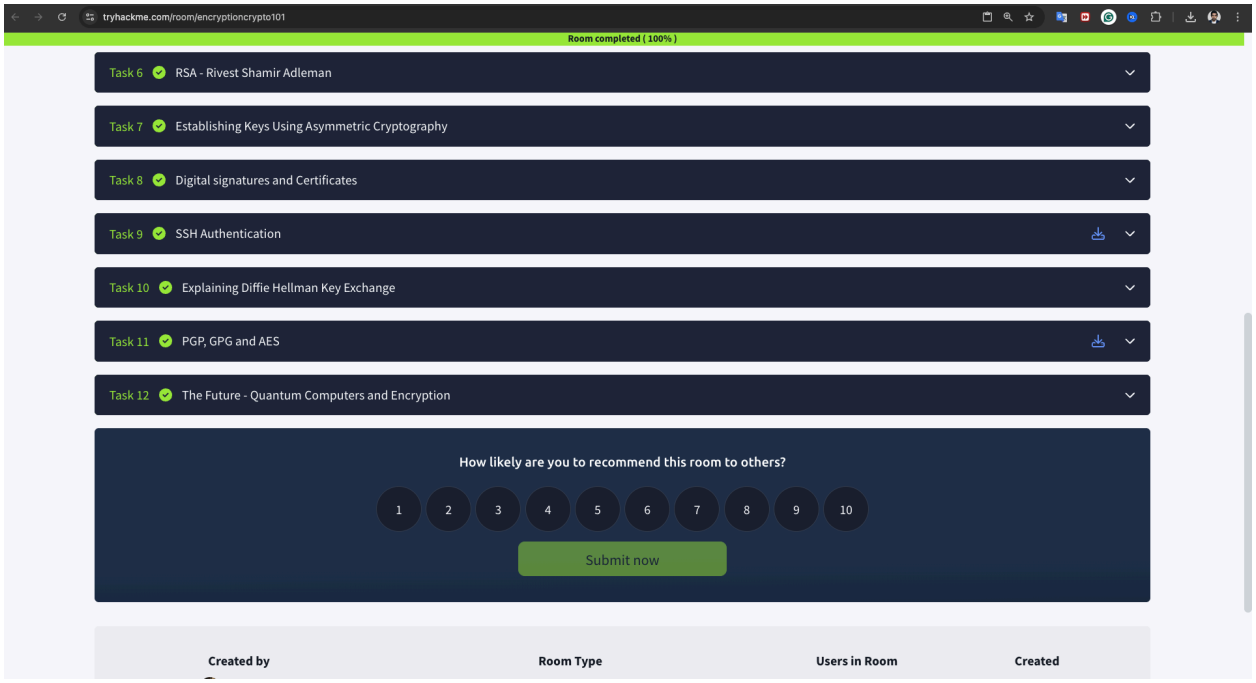
- **Encryption Tools**: Learn how to use various cryptographic tools to encrypt and decrypt messages, as well as how to generate and manage keys effectively.

Screenshot of the Completion Screen:


EncryptEdge Labs


tryhackme.com/room/encryptioncrypto101
Congratulations on completing Encryption - Crypto 101!!!
Points earned
120
Completed tasks
12
Room type
Walkthrough
Difficulty
Medium
Streak
30
Leave Feedback
Next


tryhackme.com/room/encryptioncrypto101
Try Hack Me
Dashboard
Learn
Compete
Other
Access Machines
30
Learn > Encryption - Crypto 101
Encryption - Crypto 101
An introduction to encryption, as part of a series on crypto
Medium
45 min
Share your achievement
Start AttackBox
Help
Save Room
3804
Options
Room completed (100%)
Task 1 What will this room cover?
Task 2 Key terms
Task 3 Why is Encryption important?
Task 4 Crucial Crypto Maths
Task 5 Types of Encryption



6.0 Reflection

Throughout this internship task, I have gained a deeper understanding of encryption and its crucial role in network security. Encryption is the cornerstone of securing sensitive data, ensuring that unauthorized individuals cannot access or alter information. This task helped me explore various aspects of encryption, including both theoretical concepts and practical applications. Here's a reflection on the key learnings, challenges, and insights I gained throughout this task.

6.1 Key Learnings

1. Encryption Fundamentals: I now have a solid grasp of the two primary encryption methods: symmetric and asymmetric encryption. Symmetric encryption, with algorithms like AES, is fast and efficient but requires secure key sharing, while asymmetric encryption, exemplified by RSA and ECC, offers enhanced security by using a pair of keys (public and private). This distinction helped me understand

which encryption method is suitable for different use cases.

2. Real-World Applications: I learned how encryption is applied in various real-world scenarios, such as securing email communications, file storage, and online banking transactions. Understanding how encryption helps preserve confidentiality, integrity, and authenticity in these applications emphasized its importance in maintaining privacy and trust in digital communications.

3. Hands-on Practice: Through the use of tools like OpenSSL and GPG, I gained practical experience in encrypting and decrypting data. This hands-on approach allowed me to appreciate the challenges and nuances involved in managing cryptographic keys and securely sharing data.

6.2 Challenges Faced

1. Setting Up GPG: One of the challenges I encountered was related to GPG key management. At first, I faced issues with retrieving public keys from email addresses and struggled with generating my own key pair. However, after learning how to correctly generate, list, and export keys, I was able to complete the encryption tasks successfully.

2. Key Exchange Mechanism: Another challenge was understanding how to securely exchange keys for symmetric encryption. While it's clear that asymmetric encryption can alleviate this issue, I realized how complex and risky key management is in a real-world scenario, where a compromised key exchange channel could lead to severe vulnerabilities.

3. Command Line Tools: Using command-line tools like OpenSSL and GPG was initially a bit challenging, especially when experimenting with different flags and commands. However, after repeating the steps and reading more about the options available, I became more comfortable with the syntax and gained confidence in using these tools.

6.3 Insights into Encryption's Role in Network Security

Encryption is not just a tool for data confidentiality, but an essential technique for building trust in the digital world. It ensures that sensitive information remains protected, even when transmitted over insecure channels like the internet. This task highlighted that encryption is foundational for securing online communications, banking transactions, and even day-to-day online activities.

Moreover, I gained an appreciation for the broader implications of encryption in maintaining integrity and authenticity. By protecting data from tampering and ensuring that communications are verified as coming from trusted sources, encryption supports the entire framework of secure online interactions.

As I move forward in the cybersecurity field, the practical knowledge I've gained from working with encryption tools and understanding its applications will help me build stronger, more secure networks. Encryption's role in safeguarding privacy and ensuring data integrity is invaluable, and its application will only continue to grow as digital threats evolve.

6.4 Conclusion

In conclusion, this task was an eye-opening experience that reinforced the importance of encryption in modern cybersecurity practices. The hands-on labs, theoretical insights, and practical exercises provided me with a comprehensive understanding of how encryption works and its critical role in securing sensitive information. While there were some challenges along the way, overcoming them has solidified my knowledge of encryption and its importance in network security.

EncryptEdge Labs

This Internship Task report was developed on [April, 01, 2025]

By:

atalmamun@gmail.com