REPORT OF CRYPTOGRAPHY

BY SACHCHITANAND YADAV

CRYPTOGRAPHY

MODULE - 20

Learning Objectives -

- Explain Cryptography Concepts
- Use different cryptography tools
- Use different cryptanalysis tools
- Explain Cryptography Countermeasures

TABLE OF CONTENTS

---

3. SSL Certificate Management

3.1 ZeroSSL

3.1.1 Overview
3.1.2 Features
3.1.3 Use Cases
3.1.4 ZeroSSL vs Let's Encrypt

---

4. Hashing Tools

4.1 HashCalc

4.1.1 Overview
4.1.2 Key Features
4.1.3 Primary Use Cases
4.1.4 Installation and Usage

---

5. Steganography

5.1 Introduction to Steganography
5.2 Working of Steganography
5.3 Types of Steganography
5.4 Steganography vs Cryptography

---

6. Steganography Tools

6.1 Image Steganography Tool

6.1.1 Features
6.1.2 Encoding Process
6.1.3 Decoding Process

6.2 Steghide

6.2.1 Overview
6.2.2 Supported File Types

---

7. Cryptography Countermeasures

---

8. Module Summary: Cryptography

Explain Cryptography Concepts: -

Cryptography

Cryptography is the ancient art with a modern glow-up—born from secret inks and war-time ciphers, now guarding our data in a hyper-connected world. At its core, cryptography protects information from prying eyes and dirty hands, making sure data stays **private, accurate, and trustworthy** whether it's stored or flying across networks.

Definition

Cryptography is the discipline of transforming readable data (plaintext) into an unreadable form (ciphertext) using mathematical algorithms, ensuring protection against unauthorized access.

Objectives of Cryptography (C.I.A.A.N)

Cryptography isn't just about secrecy—it's about trust. The five core objectives are:

Objective	Explanation
Confidentiality	Ensures data is accessible only to authorized users
Integrity	Guarantees data has not been altered or tampered with
Authentication	Confirms the identity of users or systems
Authorization	Restricts access based on permissions
Non-Repudiation	Prevents denial of actions like sending messages or signing documents

---

Simple Illustration

Suppose you send a message:

Plaintext:
```
I love cryptography
```

After encryption using AES:

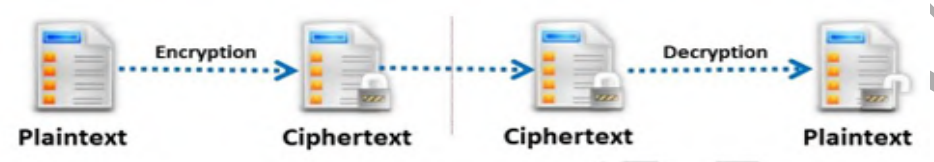
Ciphertext:
```
Y34Dd92@kLmXZ3qWs81!
```

Without the correct key, this message is just noise—beautiful, meaningless noise.

Cryptography Process

Cryptography follows a structured workflow:

1. Plaintext is encrypted using an algorithm and key
2. Ciphertext is transmitted or stored securely
3. Authorized recipients decrypt it using the correct key
4. Original plaintext is restored safely



Types of Cryptography

Cryptography is broadly categorized based on how keys are used and how data protection is achieved.

---

1. Symmetric Key Cryptography

(Secret Key Cryptography)

Overview:
The same key is used for both encryption and decryption. It's fast and efficient—but sharing the key securely is the Achilles' heel.

Algorithms:

- AES
- DES
- RC4, RC5, RC6
- Blowfish, Twofish

Applications:

- File and disk encryption
- VPN tunnels (IPSec)

- Secure data storage systems

---

2. Asymmetric Key Cryptography

(Public Key Cryptography)

Overview:
Uses a key pair—public key for encryption, private key for decryption. Slower, but far more secure for open networks.

Algorithms:

- RSA
- ECC
- Diffie-Hellman
- ElGamal

Applications:

- Digital signatures
- Secure email (PGP)
- SSL/TLS for websites

Asymmetric Key Encryption

---

3. Hash Functions

(One-Way Cryptography)

Overview:
Transforms data into a fixed-size hash. Irreversible by design. Perfect for integrity, terrible for secrecy—which is exactly the point.

Algorithms:

- MD5 (obsolete)
- SHA-1 (deprecated)
- SHA-256, SHA-512
- SHA-3

Applications:

- Password storage
- File integrity checks
- Blockchain systems

You can verify truth—without revealing it.

Public Key Infrastructure (PKI)

PKI is the trust engine of the internet. It provides the structure that allows secure communication over untrusted networks using public key cryptography.

Purpose of PKI

- Identity verification
- Secure communication
- Data integrity
- Non-repudiation

Core Components of PKI

Component	Role
Certificate Authority (CA)	Issues and signs certificates
Registration Authority (RA)	Verifies user identity
Digital Certificates (X.509)	Bind identity to public keys
Public/Private Keys	Used for encryption and signing
CRL	Lists revoked certificates
OCSP	Checks certificate validity in real time

PKI Workflow (Simplified)

1. User generates key pair and submits CSR
2. CA/RA verifies identity
3. CA issues signed certificate
4. Secure communication begins

No trust? No connection. Simple as that.

PKI in Action: HTTPS

When you visit an HTTPS website:

- Browser validates the site's certificate
- CA trust is verified
- Secure TLS session is established
- Data flows encrypted and safe

That little padlock? It's doing heavy lifting.

Signed vs Self-Signed Certificates

CA-Signed Certificate

Issued by trusted authorities.

Key Traits:

- Trusted by browsers
- Verified identity
- Used for public-facing services

Self-Signed Certificate

Signed by the owner itself.

Key Traits:

- Not trusted by default
- No third-party verification
- Used in testing or internal labs

Free isn't always safe—context matters.

Digital Signature

A digital signature ensures **who sent the data**, **what was sent**, and **that it wasn't altered**. Think of it as a cryptographic fingerprint with legal weight.

Objectives

- Authentication
- Integrity
- Non-repudiation

Digital Signature Process

Signing:

1. Message is hashed
2. Hash is encrypted using private key
3. Encrypted hash becomes the signature

Verification:

1. Signature decrypted using public key
2. Message re-hashed
3. Hashes compared

Match? Trust granted.

Common Algorithms

- RSA
- DSA
- ECDSA
- EdDSA

Digital vs Electronic Signature

Feature	Digital	Electronic
Cryptography	Yes	Not always
Security	Very High	Variable
Legal Validity	Strong	Depends

One is math. The other is mostly vibes.

SSL & TLS

SSL was the pioneer. TLS is the upgrade that fixed its mistakes.

Purpose

- Encrypt data
- Authenticate parties
- Maintain integrity

TLS Handshake (Simplified)

1. Client Hello
2. Server Hello + certificate
3. Certificate verification
4. Key exchange
5. Encrypted session starts

Asymmetric for trust. Symmetric for speed.

TLS Versions

Version	Status
TLS 1.0 / 1.1	Deprecated
TLS 1.2	Secure
TLS 1.3	Latest & fastest

Evolution matters. Legacy breaks things.

Cryptanalysis

Cryptanalysis is the dark mirror of cryptography—the study of breaking encryption systems without authorized keys.

Common Cryptanalysis Techniques

- Brute Force
- Dictionary attacks
- Frequency analysis
- Known / Chosen plaintext attacks
- Side-channel attacks
- MITM
- Birthday attacks

Common Targets

- Weak algorithms
- Poor key management
- Predictable passwords

Crypto fails less often than humans do. Brutal truth.

DataLock Using Private Encryption Application

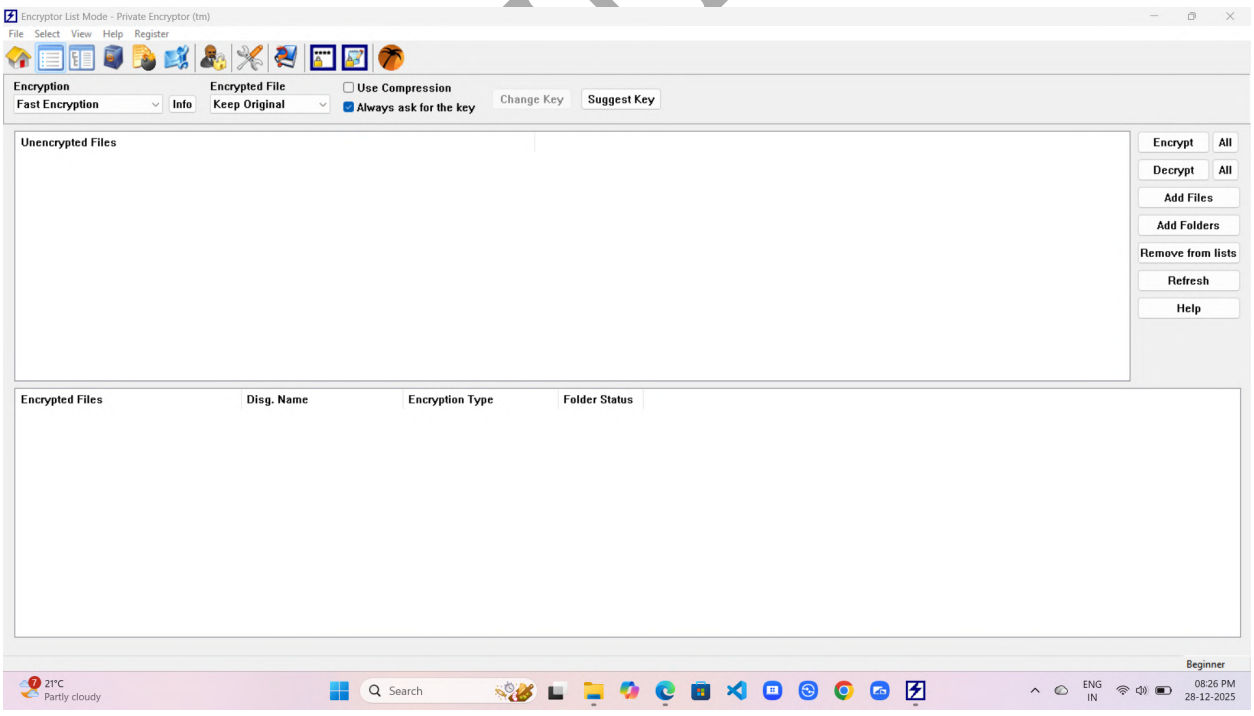
Private Encryptor (DataLock) is a data protection application built to safeguard sensitive and personal information by transforming readable files into an encrypted, unreadable form. It uses robust cryptographic algorithms such as **AES** and **RSA** to ensure that data remains secure even if unauthorized users gain access to the system.

Access to the original data is strictly restricted—only users who possess the correct **password or private key** can decrypt and view the contents. This mechanism guarantees **confidentiality** and effectively blocks unauthorized access.

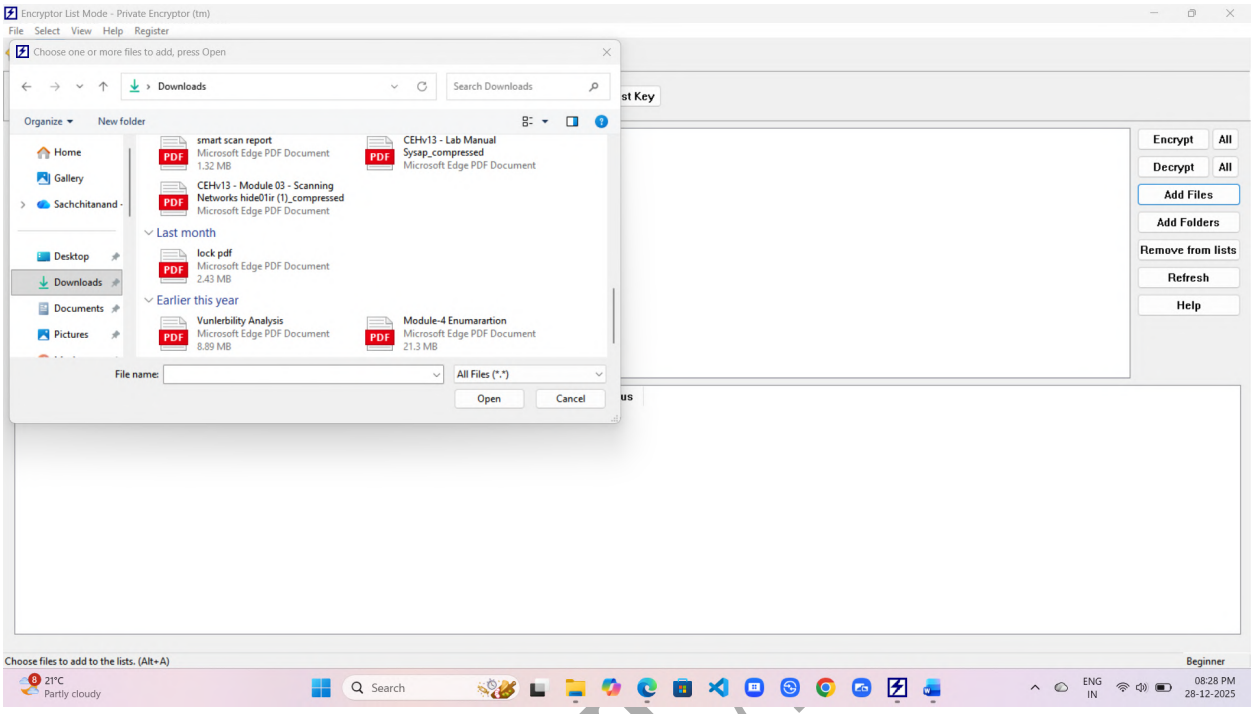
The application is commonly used to secure **documents, folders, and personal files** stored on a local machine, making it a reliable solution for individuals and organizations that need strong data protection without relying on external networks.

How to use it -

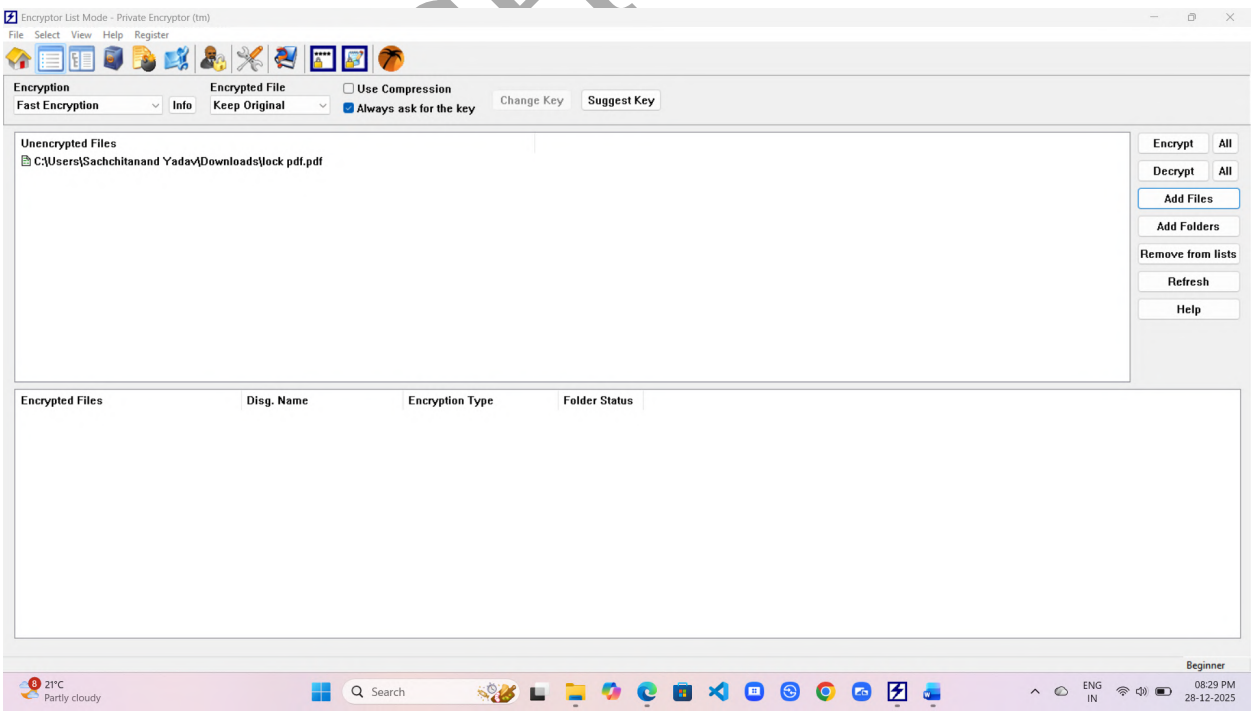
- Private encryption interface



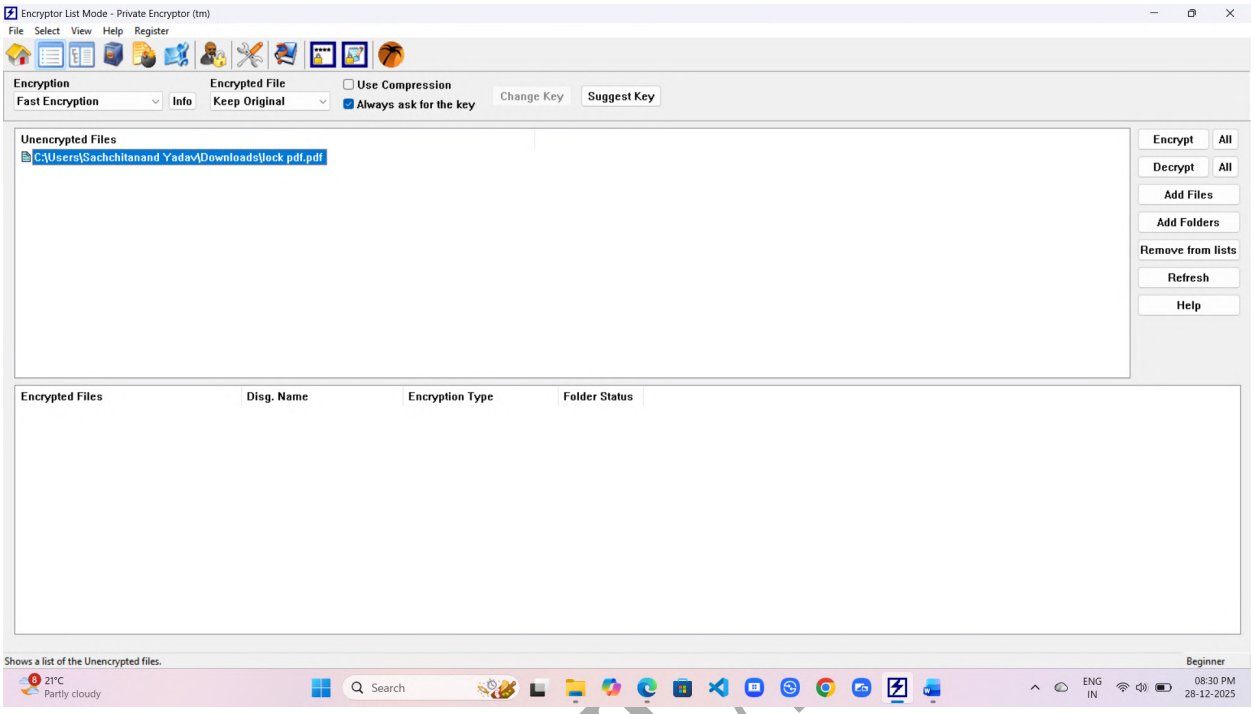
- select file and click on open



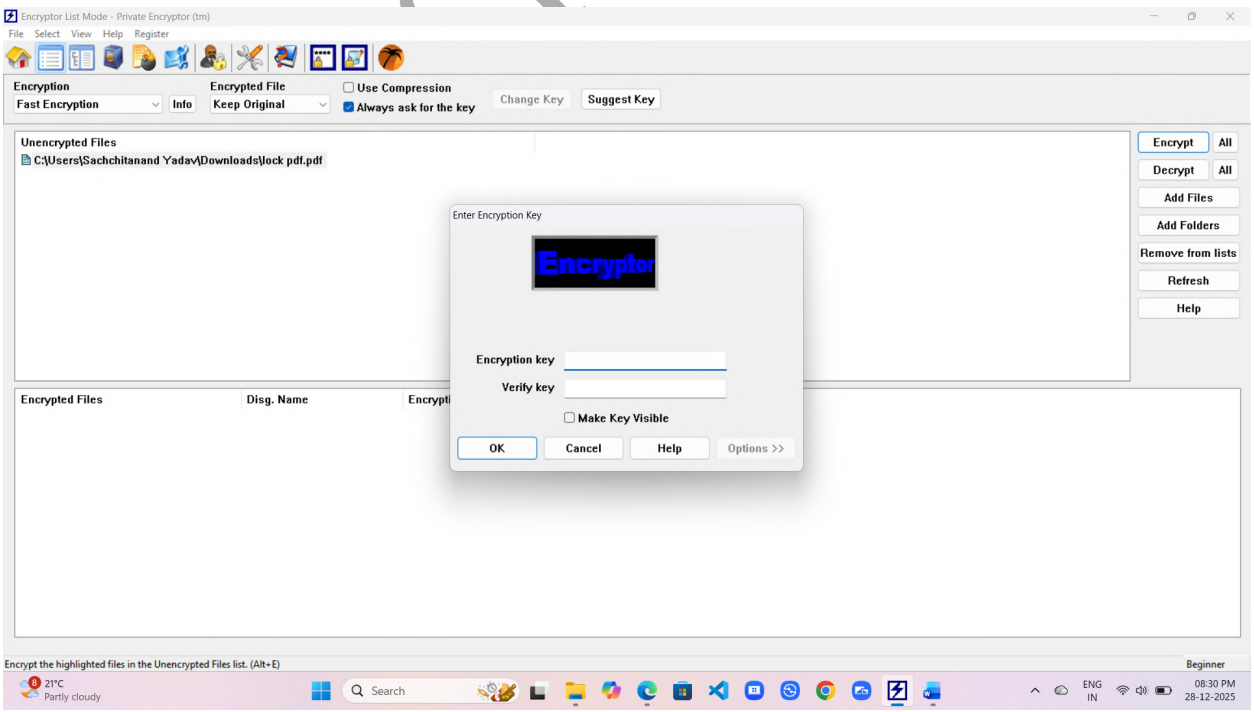
- file added successfully



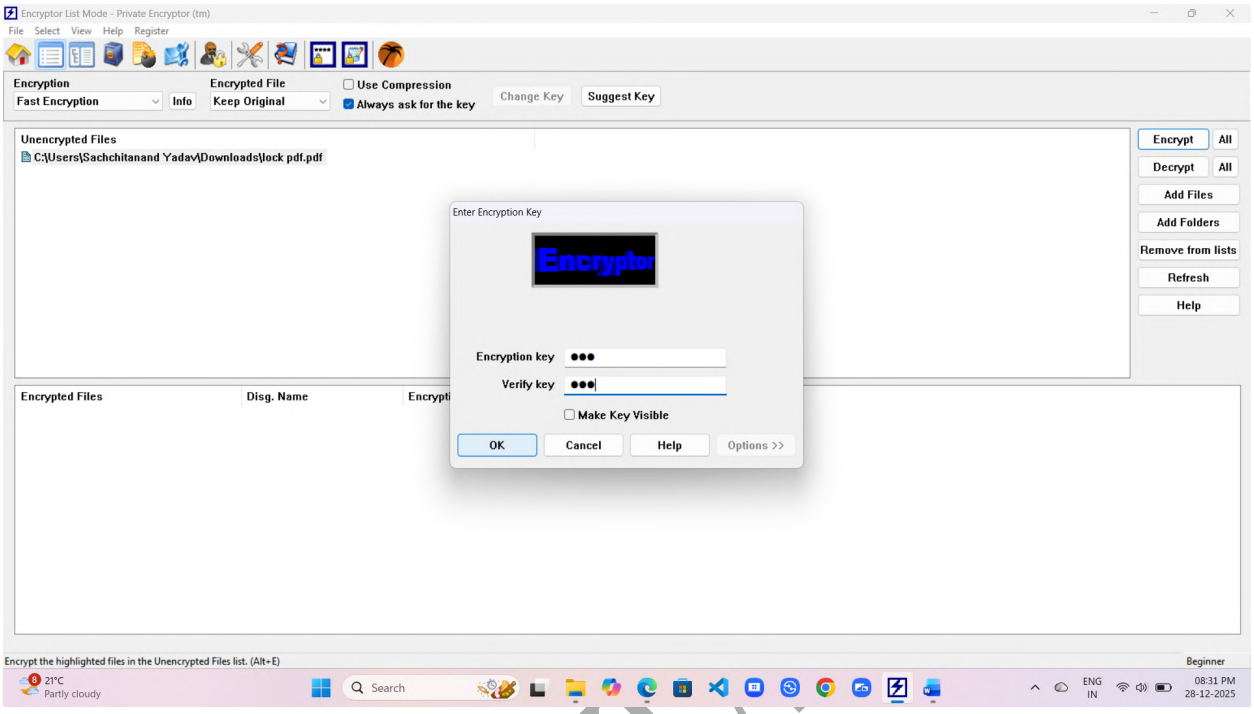
- click on file



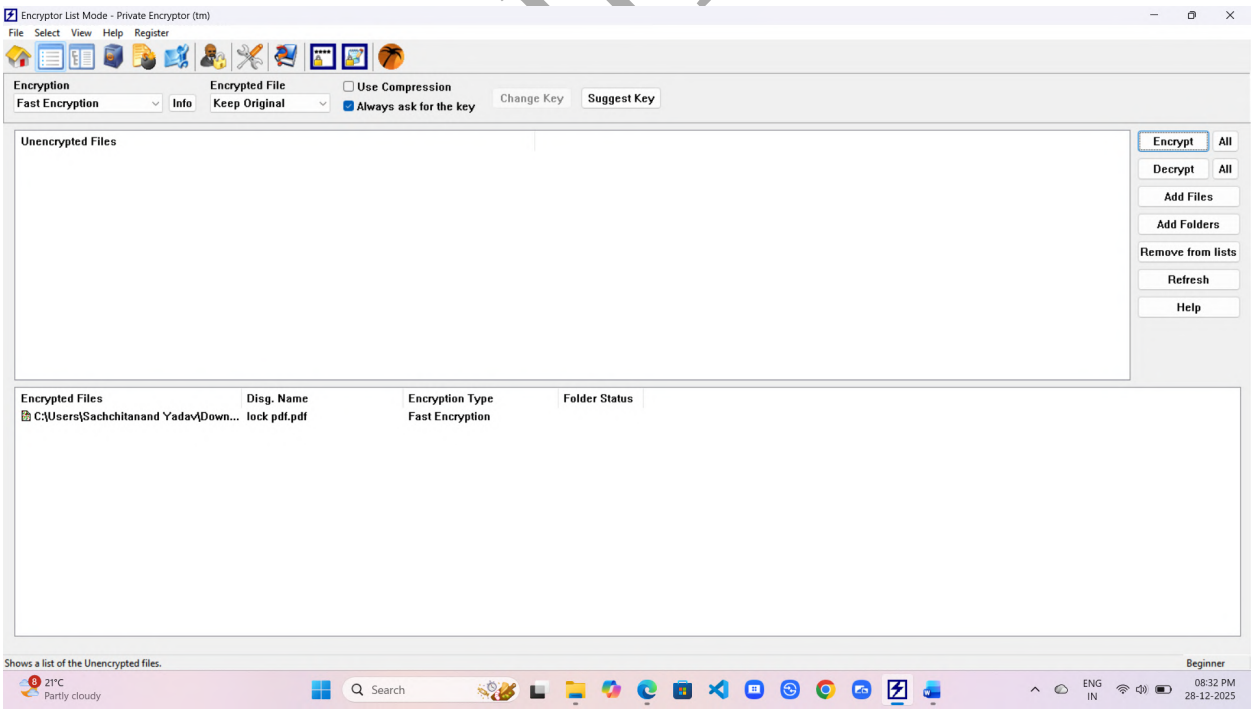
- and then click on encrypt
- set strong password for file



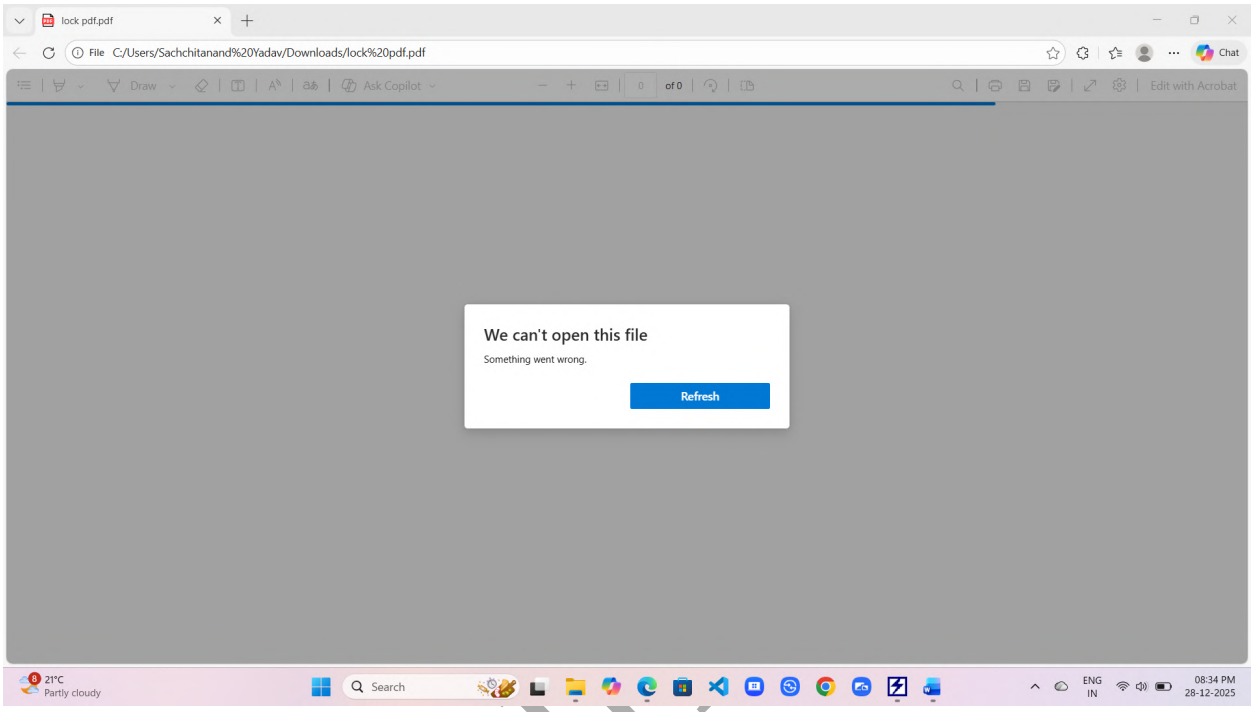
- click on ok



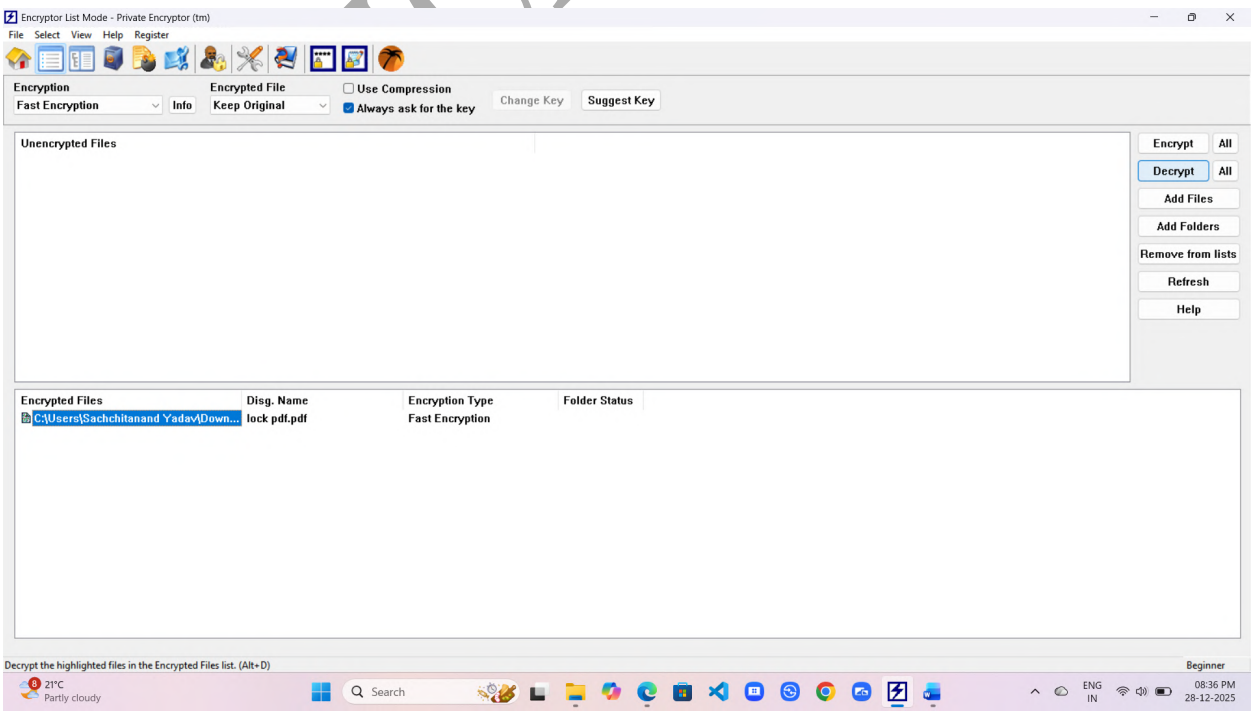
- File encrypted



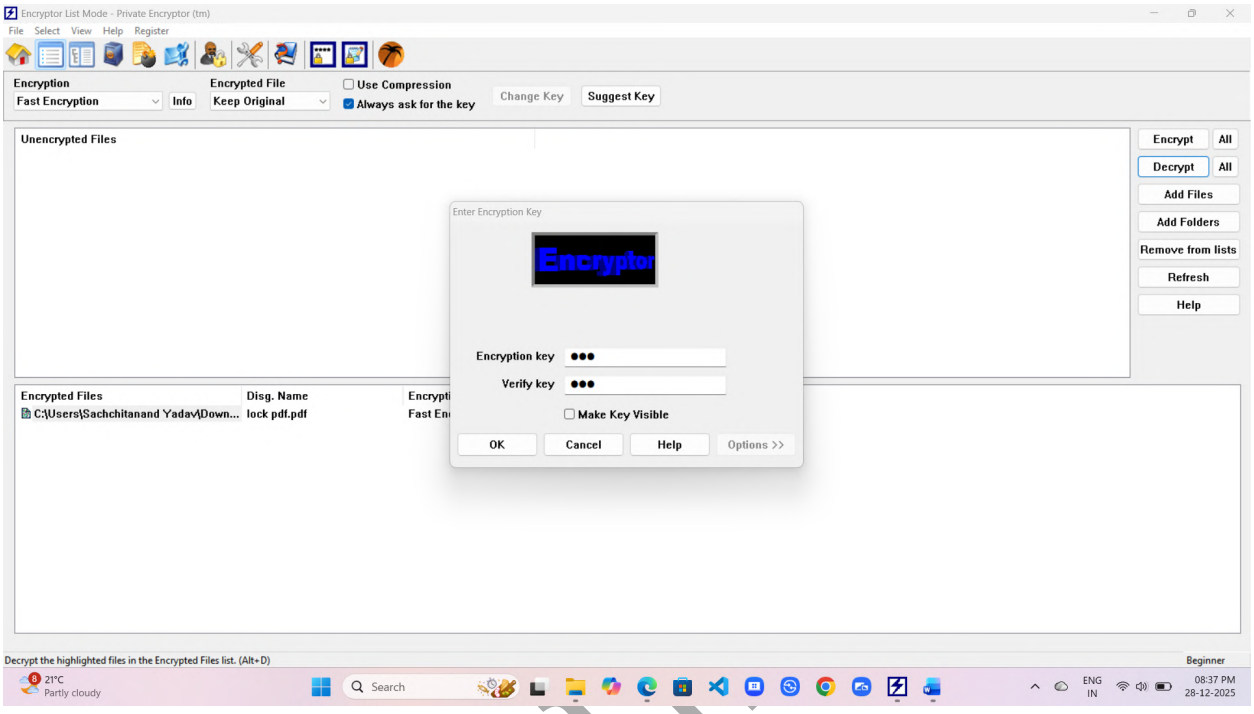
- Now open file to check its encrypted or not
- File encrypted —the main thing of this application,it not show file is encrypted it show something else



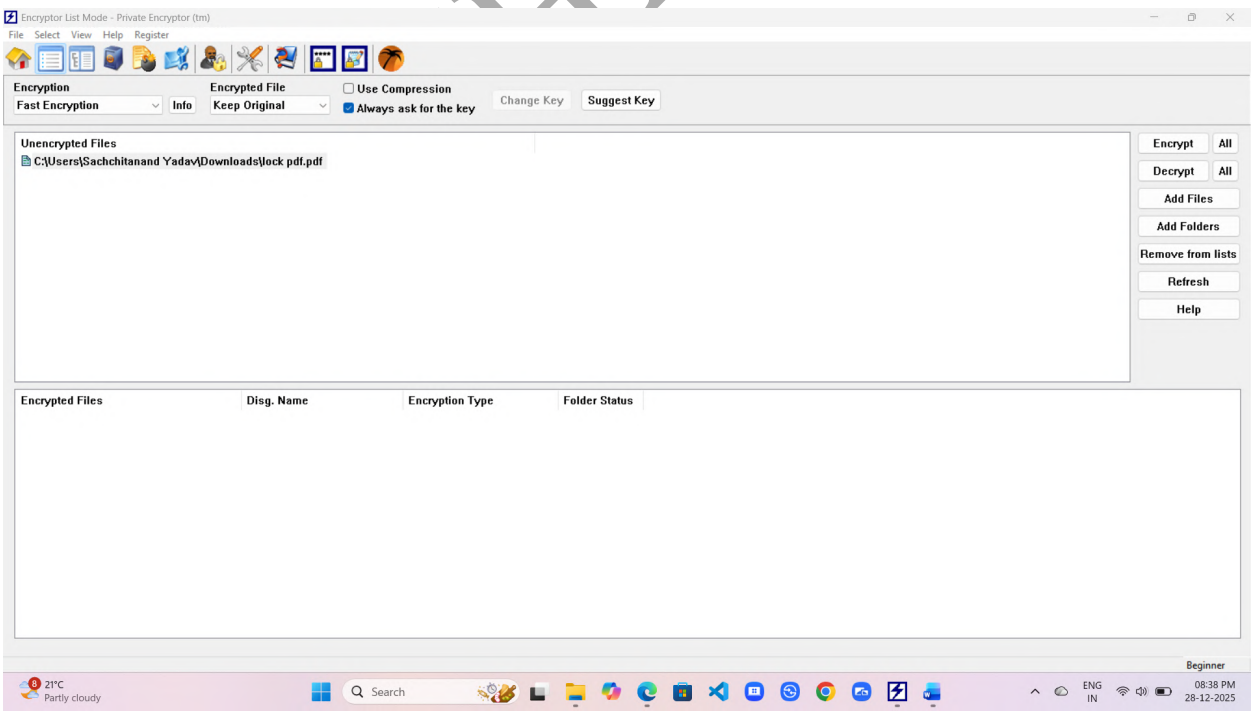
- Click on fille
- Click on Decrypt



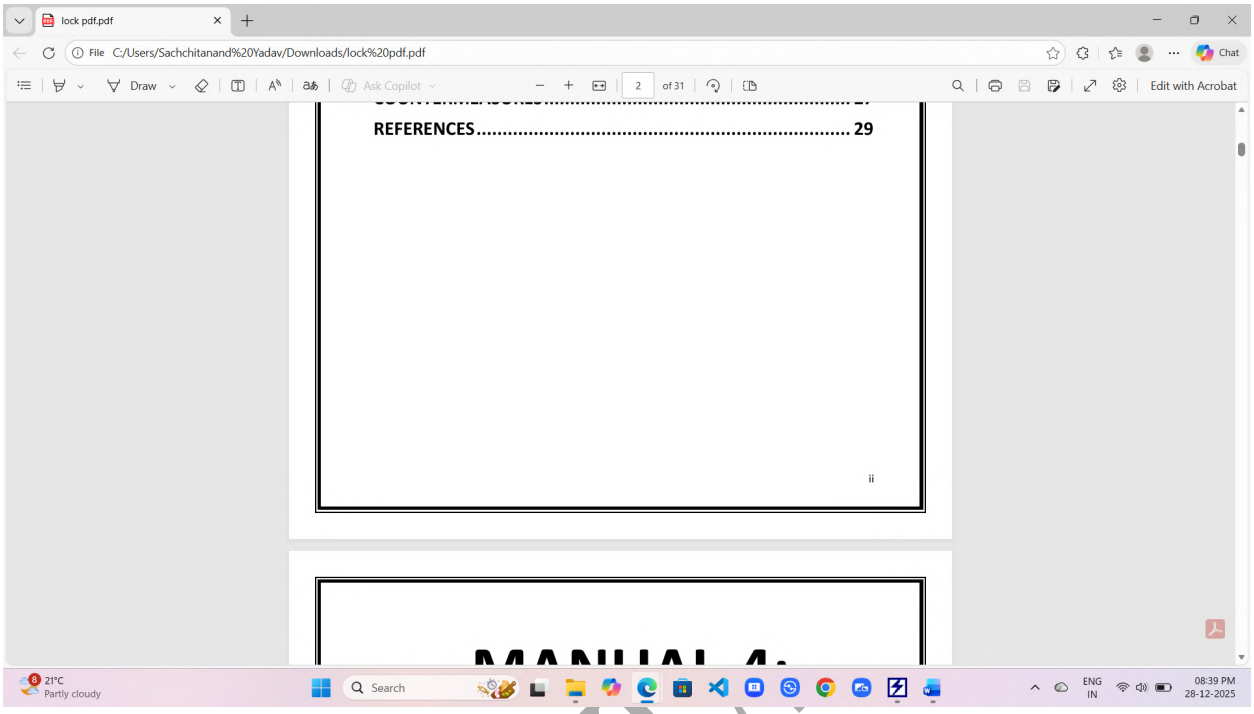
- Enter a password that you set during encryption
- Click on ok



- File decrypted



- Result



Confidential Data Encryption with VeraCrypt on Windows

VeraCrypt is a free and open-source disk encryption tool used to protect confidential data on Windows systems. It secures information by encrypting entire storage devices, individual partitions, or container files, ensuring that data remains inaccessible without proper authentication. Developed as a successor to the discontinued **TrueCrypt** project, VeraCrypt improves upon its predecessor by implementing stronger encryption algorithms and enhanced security mechanisms.

Key Features

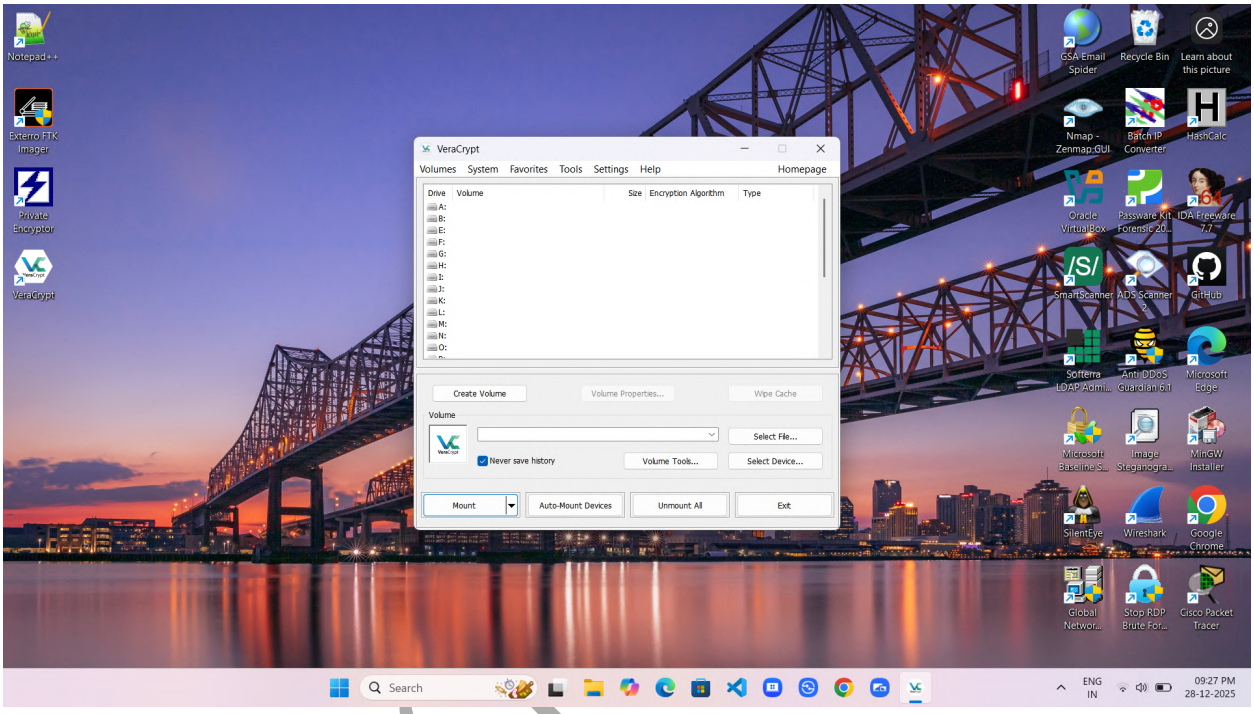
- **Robust Encryption Algorithms**
  Supports advanced cryptographic standards such as **AES, Serpent, Twofish**, and multi-layered combinations (e.g., AES + Twofish + Serpent) for maximum security.
- **On-the-Fly Encryption**
  Automatically encrypts and decrypts data during read/write operations, requiring no manual intervention from the user.
- **Encrypted Volume Creation**
  Allows users to create virtual encrypted disks that function like regular drives once mounted.
- **Hidden Volumes**
  Provides plausible deniability by allowing the creation of concealed volumes within encrypted containers.
- **Full Disk and Partition Encryption**
  Enables encryption of entire internal drives, external storage devices, and USB flash drives.
- **Portable and Cross-Platform Support**
  Compatible with **Windows**, **Linux**, and **macOS (experimental)** environments.

Use Cases

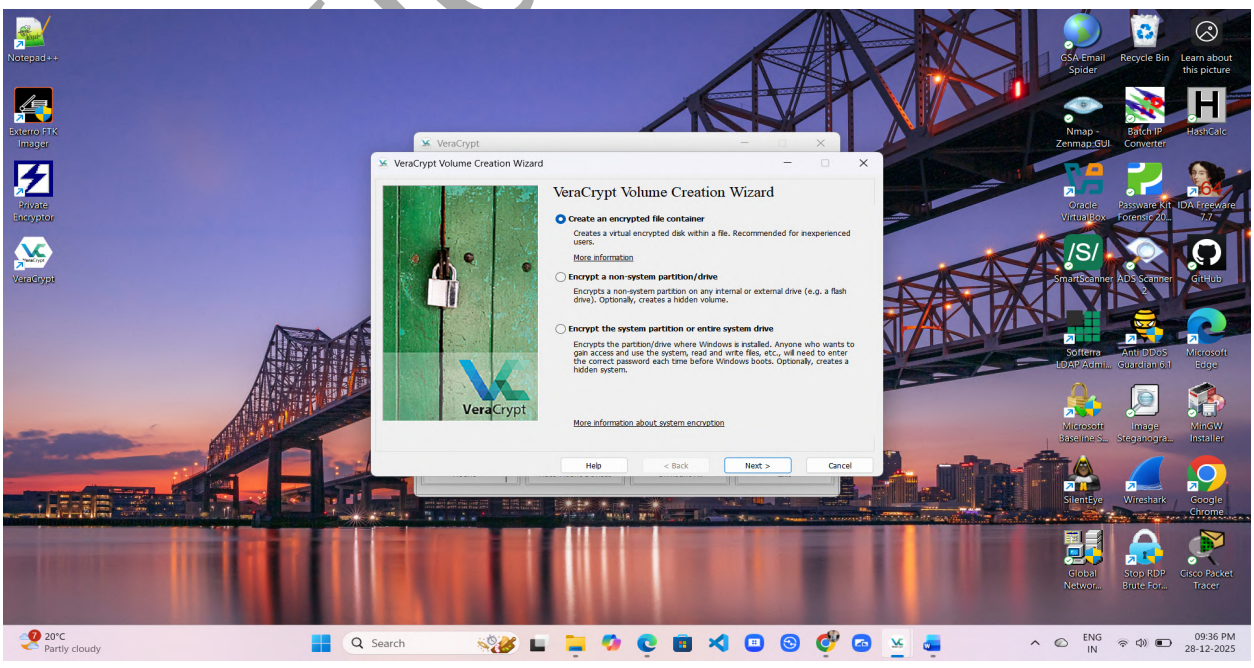
- Securing personal documents and system backups
- Encrypting portable storage devices such as USB drives
- Protecting sensitive data against device theft or loss
- Implementing hidden and deniable storage for high-risk scenarios

How to download it -

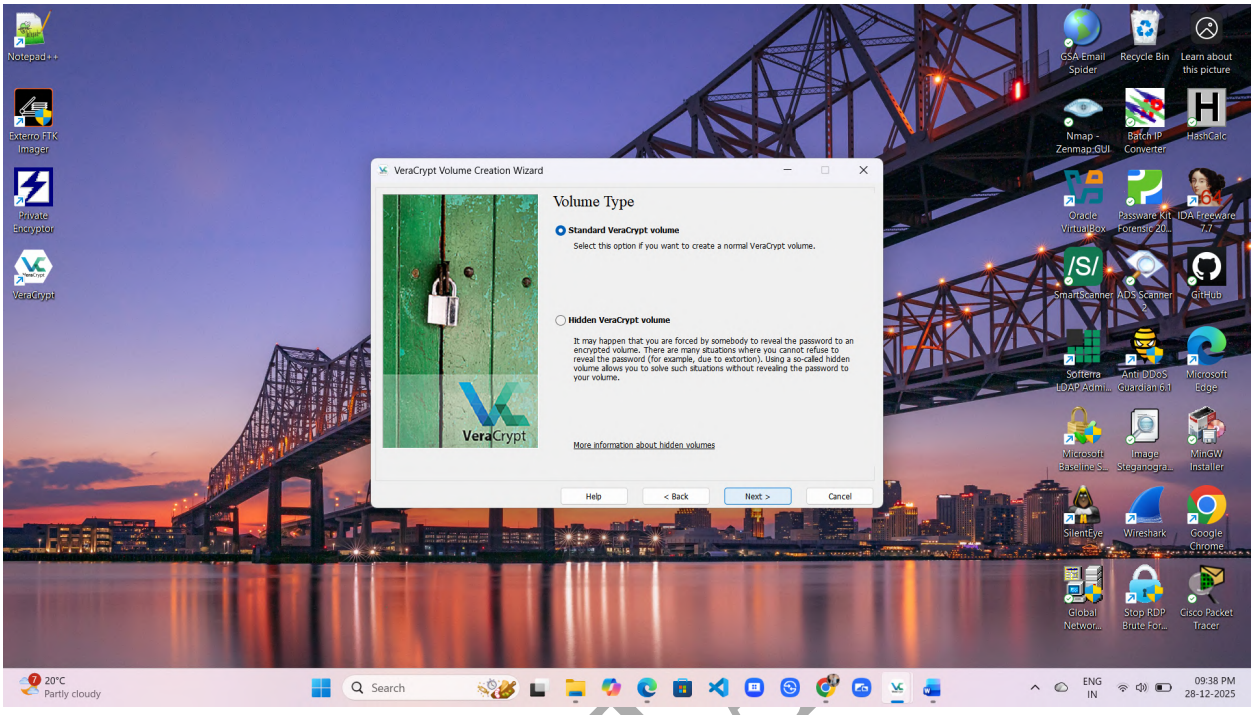
- Open Browser and search Veracrypt
- click on Download
- After setup veracrypt open it
- Click on create volume



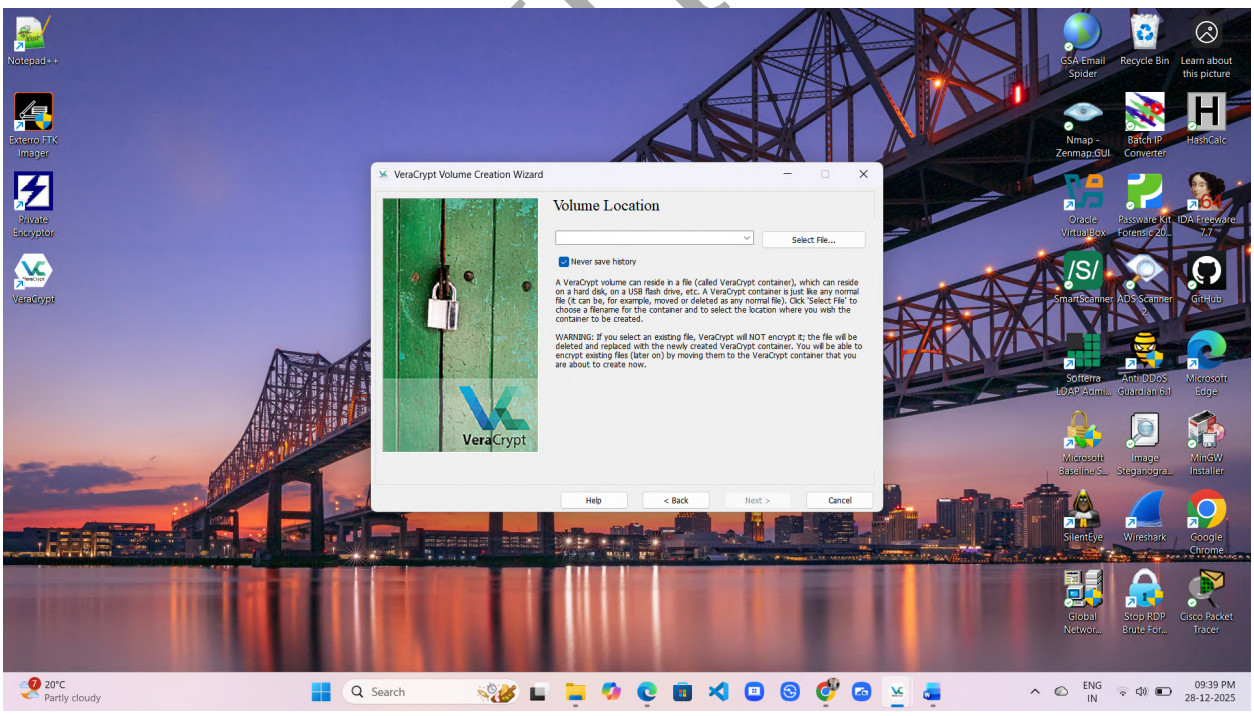
- Select first option – Create an encrypted file container



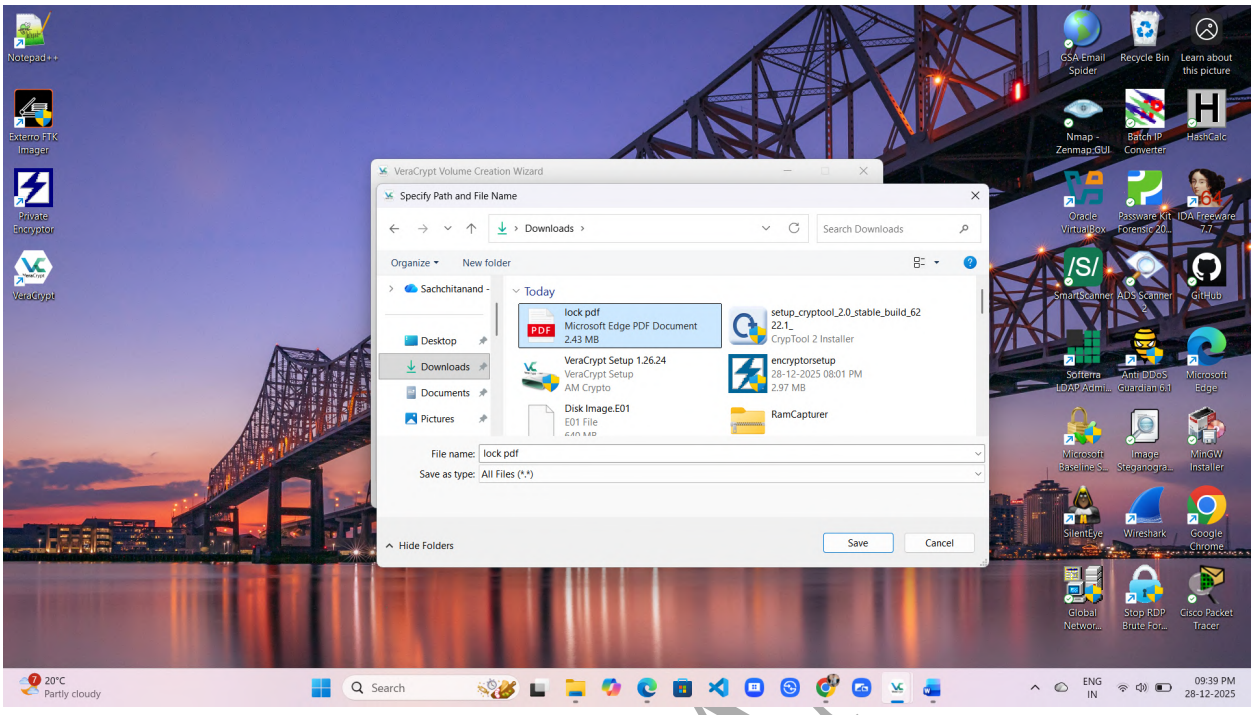
- Now select first option – Standard VeraCrypt volume



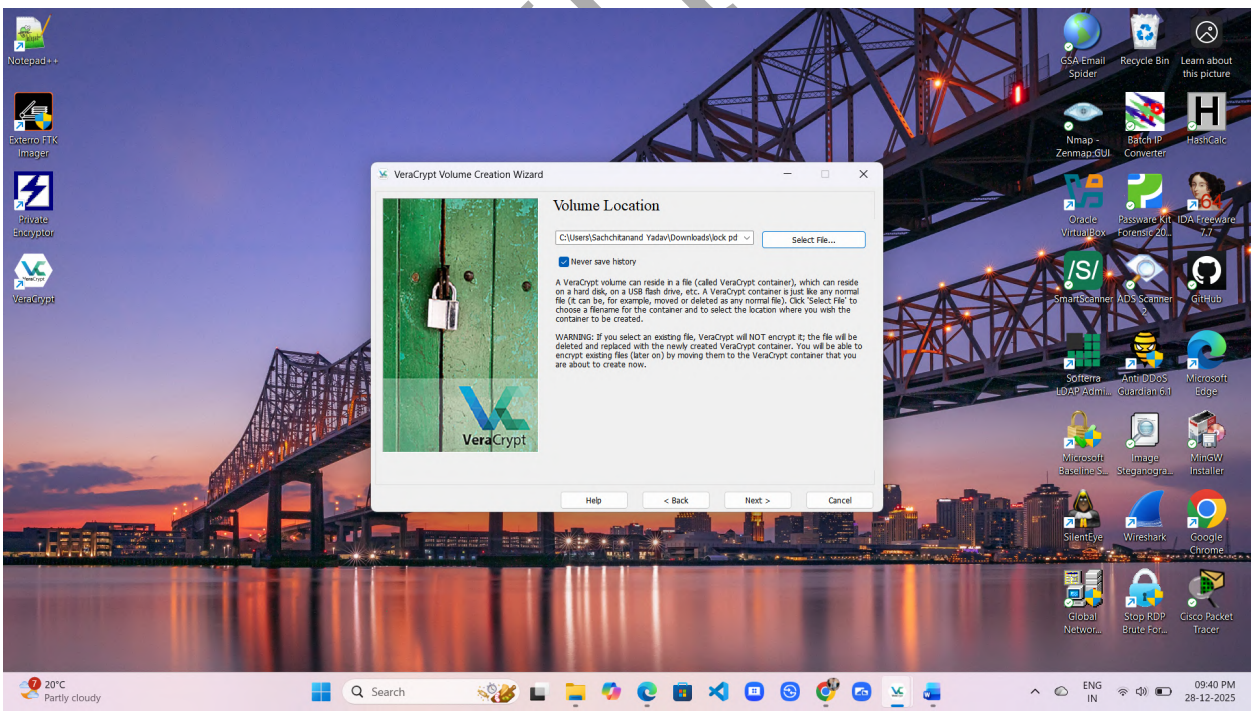
- Select file that you want to encrypted



- Click on save

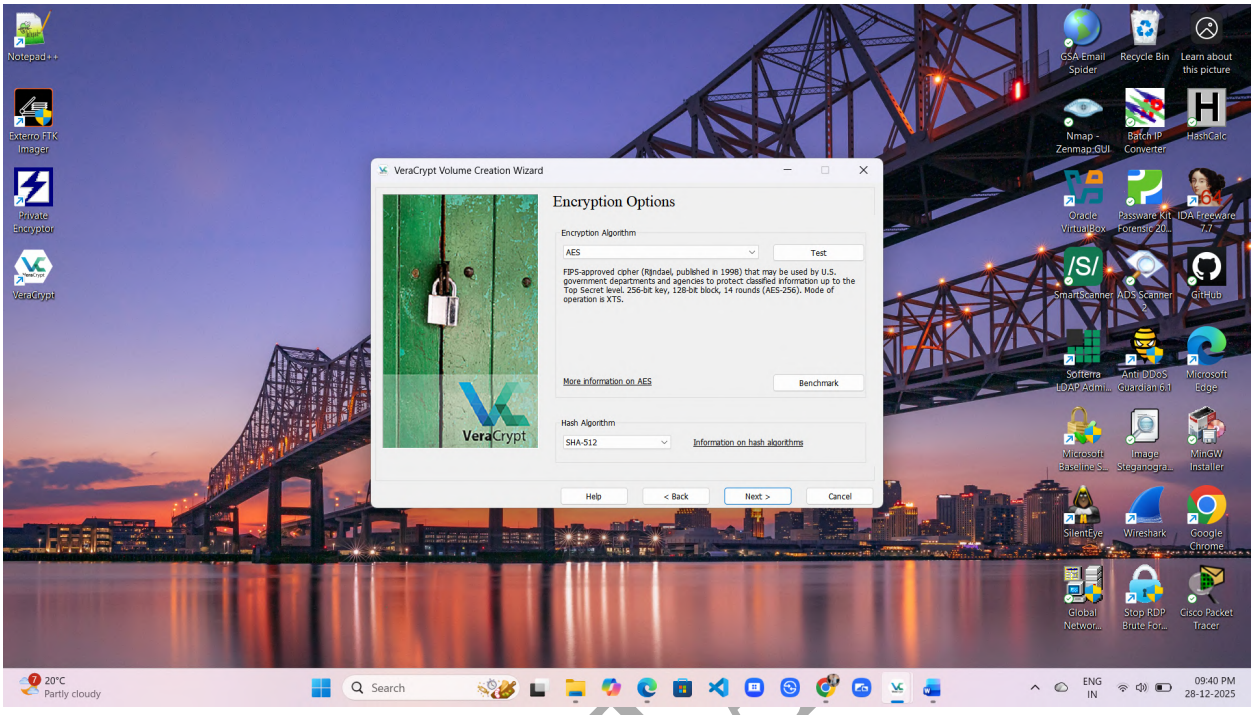


- Click on next

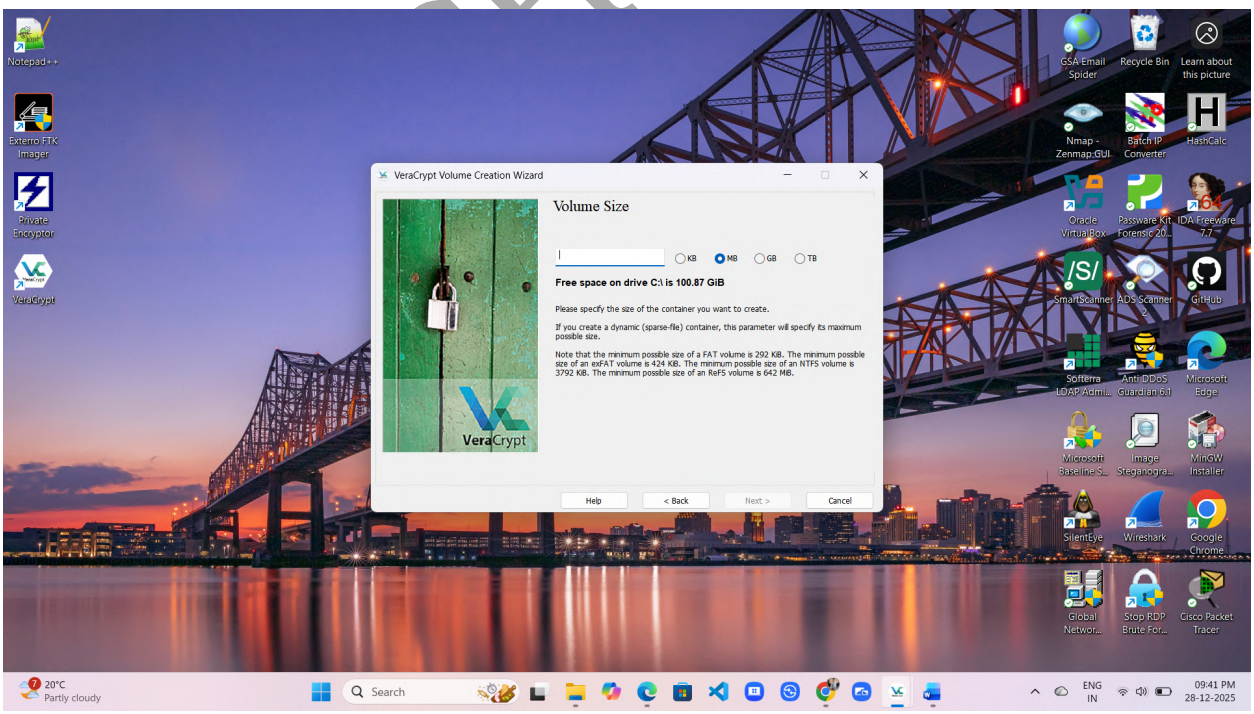


- Click on drop-down arrow – to select encryption type

- Click on next

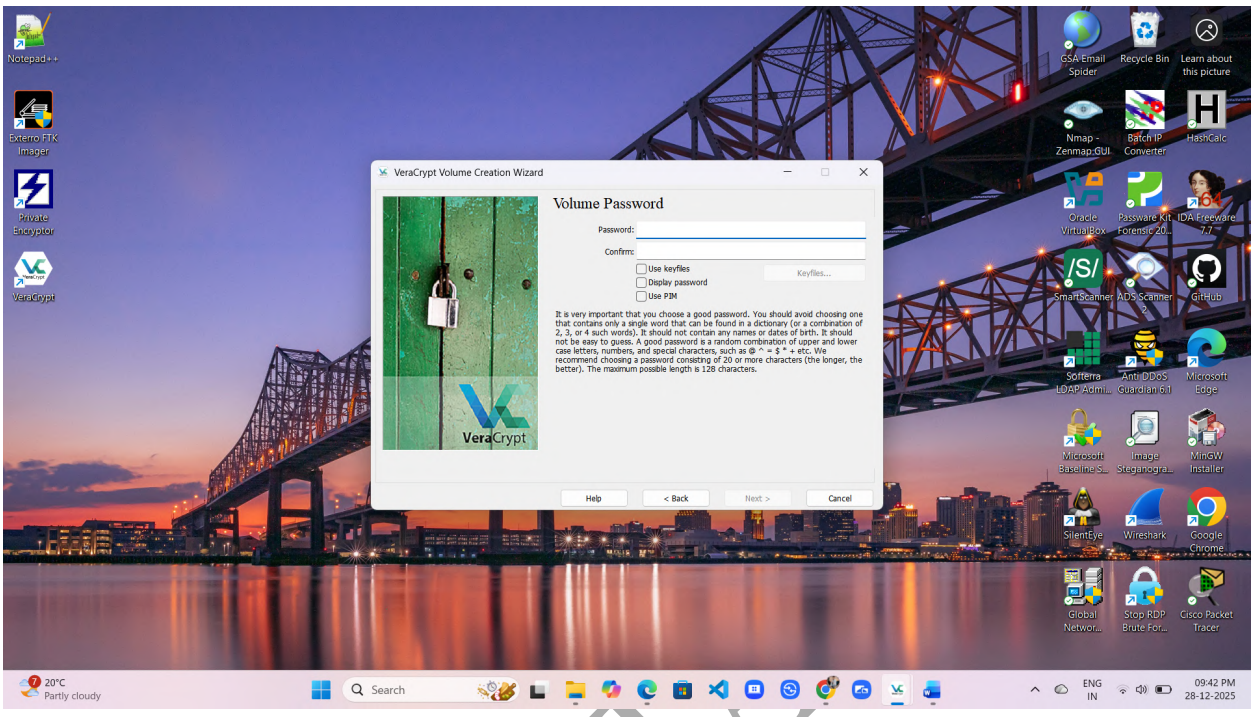


- Select free space on drive
- Click on next

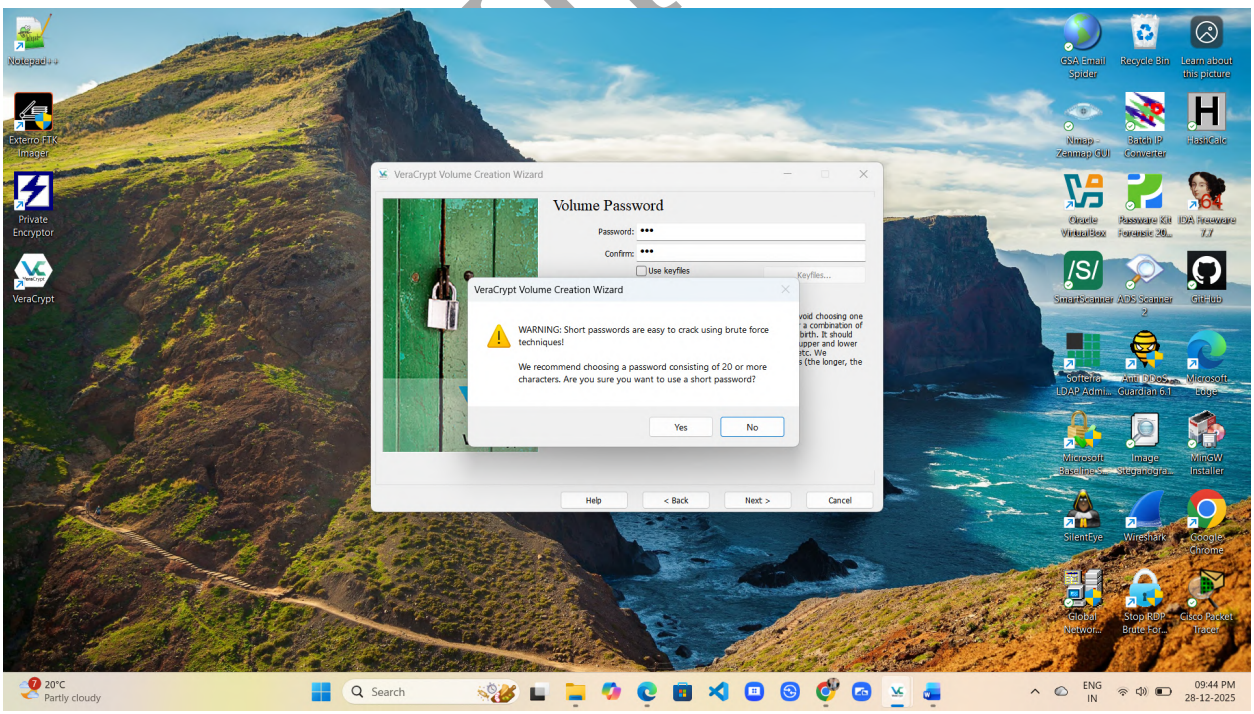


- now set a password

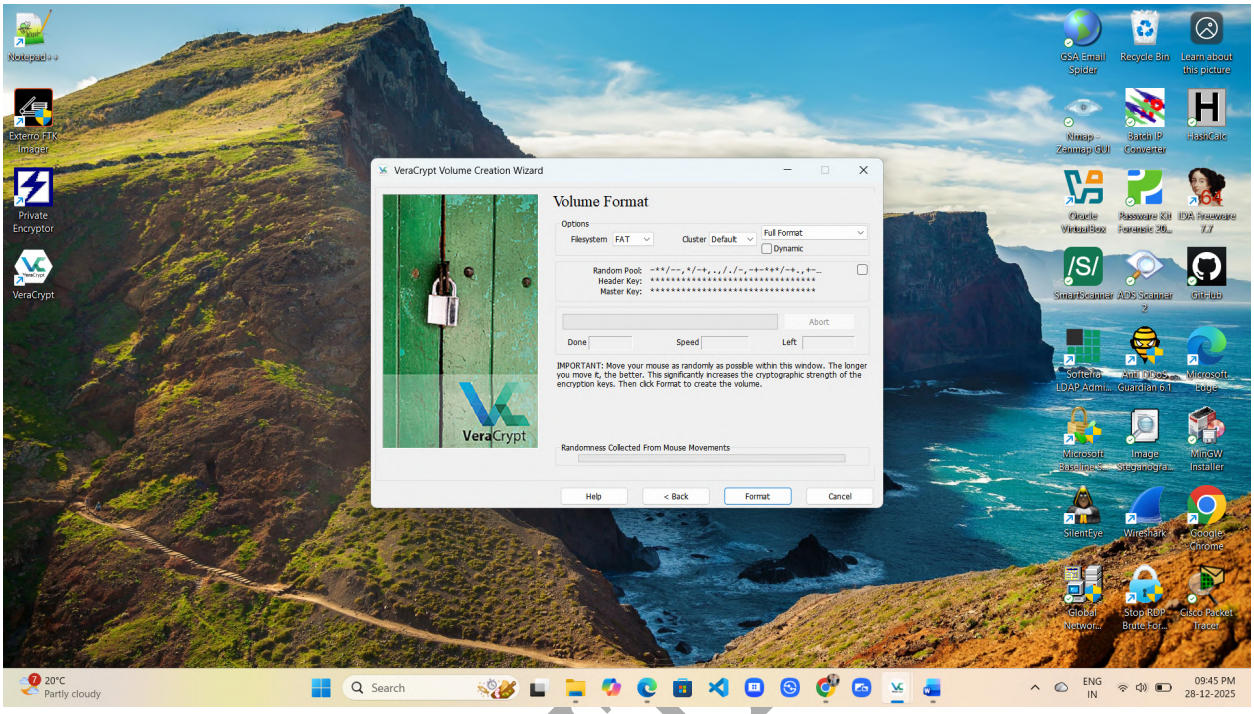
- click on next



- click on yes



- Encryption started
- Click on format



What Actually Happens When You Click "Format" in VeraCrypt

The moment you hit **Format**, there's no going back—this is the point of commitment. Old-school rule applies: *measure twice, cut once.*

Here's the play-by-play:

1. **File System Creation**
   VeraCrypt formats the volume using your chosen file system (in your case, **FAT**). This sets the basic structure—like laying down the roads before traffic starts moving.
2. **Full Encryption Begins**
   The entire volume is encrypted using the selected encryption algorithm. No shortcuts, no gaps. Every byte gets wrapped in cryptographic armor.
3. **Encrypted Container Is Born**
   A virtual encrypted file container is created. Think of it as a digital vault—looks harmless on the outside, Fort Knox on the inside.
4. **Confirmation Message**
   VeraCrypt gives you a success message once the process finishes. This is your green light: the volume is ready for action.

5. **Mount & Use Like a Normal Drive**
   Now you can mount the volume from the main VeraCrypt window, enter the correct password, and boom—it behaves like a regular drive.
   Copy files, edit them, delete them—everything is encrypted **on the fly** and decrypted only when accessed. Silent, seamless, secure.

⚠ Important Note (Read This Like a Warning Sign)

- Clicking **Format destroys all existing data** in the target volume. No mercy, no recycle bin, no second chances.
- After formatting, always mount the volume through the VeraCrypt main window using the **correct password**—wrong password means the vault stays sealed.

Create Free SSL Certificate Using ZeroSSL

What is ZeroSSL?

ZeroSSL is a **free SSL/TLS certificate provider**. Old-school goal, modern hustle:
👉 **Turn HTTP into HTTPS** without charging rent.

In plain speak: it encrypts data between browser and server so attackers just see gibberish. Same tradition as SSL itself—privacy over chaos—but with a Gen-Z twist: *free and fast*.

What actually goes down?

- Issues **DV (Domain Validation) SSL certificates**
- Uses **ACME protocol** (same lane as Let's Encrypt)
- Certificates are usually **valid for 90 days**
- Supports **auto-renewal** (because manual renewal is pain)

Why people use it

- Free tier (no wallet damage)
- Easy setup with cPanel, Plesk, Certbot
- Trusted by major browsers
- Good for labs, portfolios, startups, test servers

But let's not glaze it

- ❌ No OV/EV in free tier
- ❌ Short validity = you *must* automate renewal
- ❌ Not "set it and forget it" unless configured properly

Security tools don't babysit you. Tradition says: **you stay alert or you get burned.**

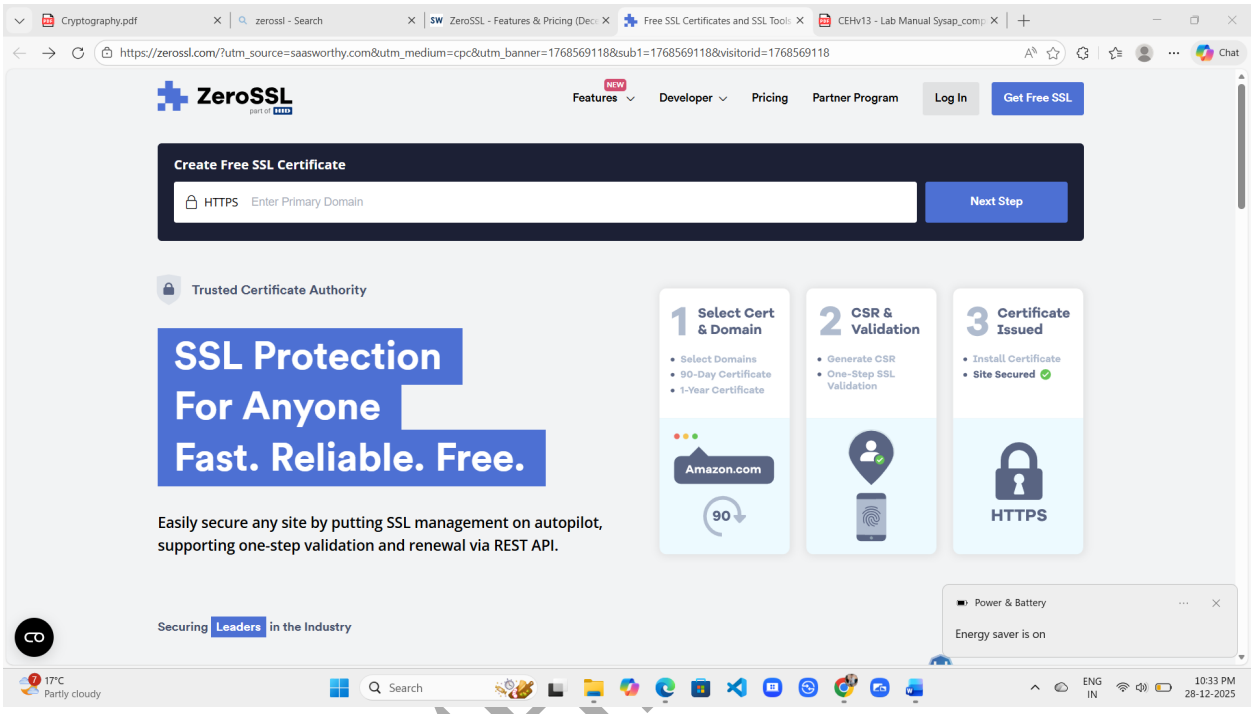
ZeroSSL vs Let's Encrypt (real talk)

- Let's Encrypt = OG, more widely automated
- ZeroSSL = cleaner UI, beginner-friendly
  Both are legit. Neither makes you "unhackable." Anyone selling that dream is lying.

Cyber rule to remember

SSL ≠ security.
SSL = **privacy in transit**, not immunity from bad code, weak passwords, or sloppy configs.

So yeah—ZeroSSL is solid.
Not magic. Not hype. Just a sharp tool.
Use it right, automate renewals, and don't act surprised when security still needs effort.



HashCalc

HashCalc is a free, Windows-based utility used to compute **cryptographic hash values, checksums, and HMACs** (Hashed Message Authentication Codes) for files, text, and strings. It is developed by **SlavaSoft** and is commonly used to verify **file integrity** and perform hash-related security tasks.

Key Features of HashCalc

Feature	Description
Hash Functions	Supports MD5, SHA-1, SHA-256, SHA-384, SHA-512, RIPEMD-160, etc.
Input Support	Can hash files, plain text, hex strings, or binary data
HMAC Calculation	Supports keyed hashing (HMAC)
Multiple Output Formats	Displays hash values in hexadecimal, Base64, etc.
Fast & Lightweight	Small, portable, and quick to execute

Primary Use Cases

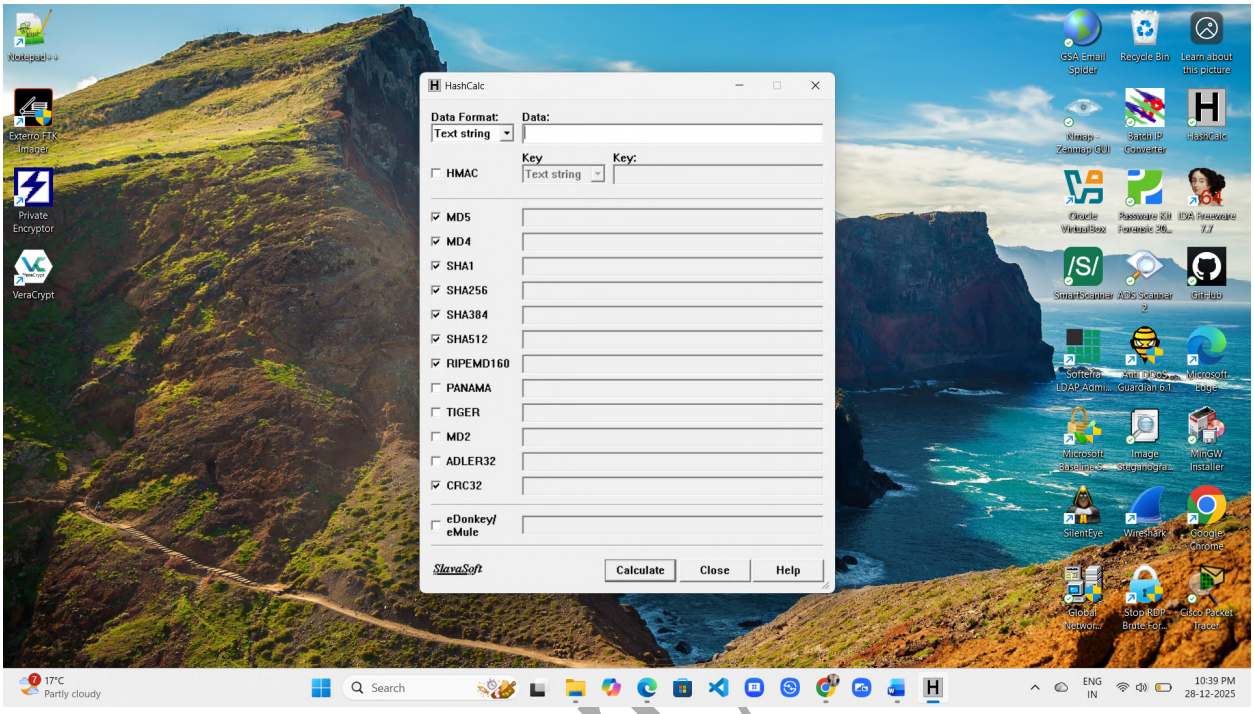
Use Case	Example
File Integrity Checking	Compare downloaded ISO file hash with the original hash
Password Hashing Practice	Test and understand password hashing using text input
HMAC Generation for APIs	Secure message authentication using shared secret keys
Malware Analysis	Generate MD5/SHA-1 hashes for malware identification
Digital Forensics	Hash evidence files to ensure data integrity during investigations

---

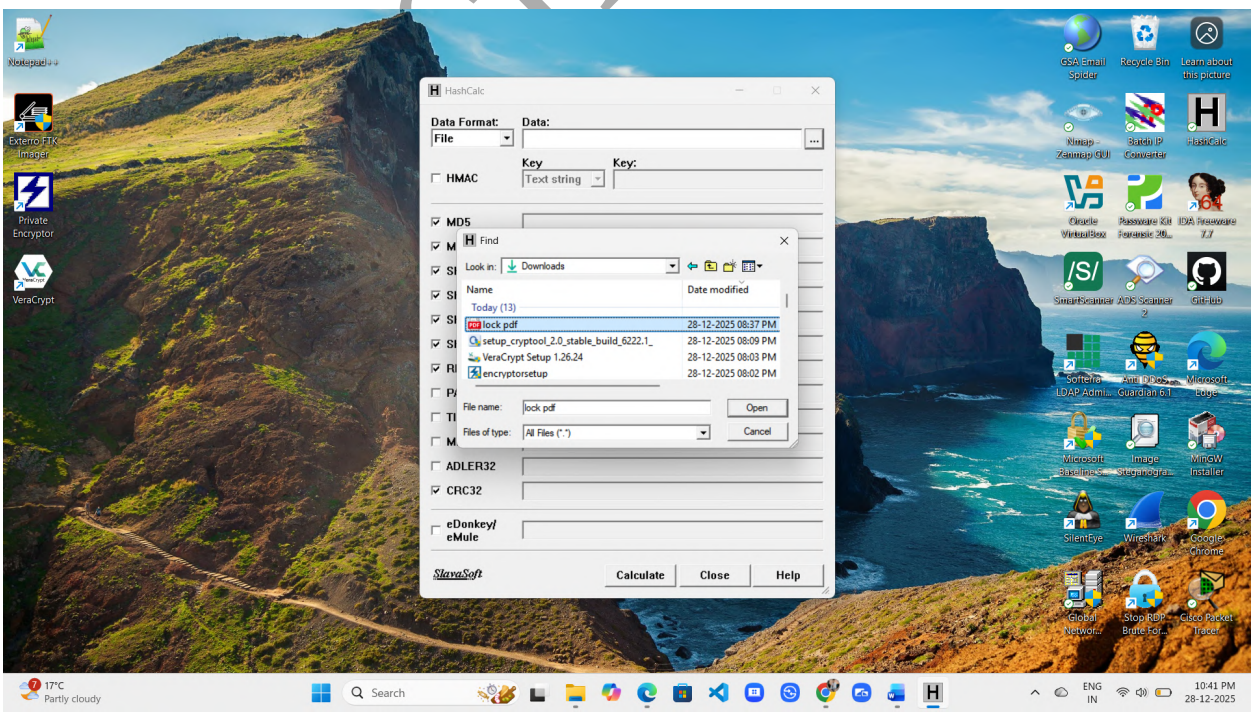
How to Download HashCalc

1. Open a web browser
2. Search for **“HashCalc SlavaSoft”**
3. Open the **official Softonic website**
4. Download and install the tool

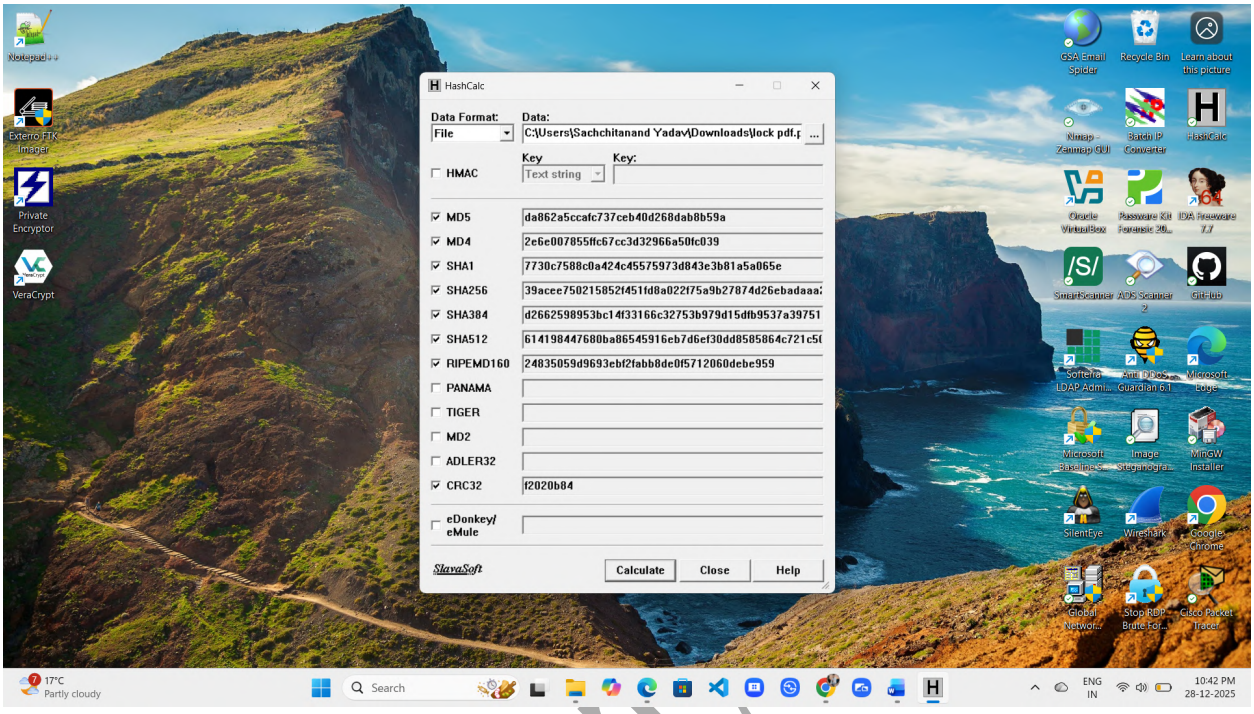
- After installed application , open it



- Select file and click on open



- After select the file, click on calculate button



Steganography

Steganography is the art and science of **hiding secret information inside another file** (image, audio, video, or text) so that the existence of the message itself is concealed.

Not encryption. Different game.
Encryption screams *"there's a secret here."*
Steganography whispers *"nothing to see."*

Tradition says hide in plain sight. Future says automate it.

How Steganography Works

1. A **cover file** is chosen (image, audio, video, text)
2. **Secret data** is embedded inside it
3. The result is a **stego file**
4. To normal users: looks totally normal
5. To the right tool + key: message revealed

No drama. Just quiet deception.

Types of Steganography

Type	Example
Image Steganography	Hiding data in image pixels (LSB method)
Audio Steganography	Hiding data in sound files
Video Steganography	Data hidden across video frames
Text Steganography	Using spaces, fonts, or formatting

Common Tools

- Steghide
- OpenStego
- SilentEye

- StegSolve

Old tools, still sharp. Like a well-used lockpick.

Use Cases

- Secure communication
- Digital watermarking
- Covert data transfer
- Cyber warfare & intelligence
- Malware command-and-control (yeah, it's used there too)

Let's not pretend it's always for good. Tech is neutral; intent isn't.

Steganography vs Cryptography

Steganography	Cryptography
Hides data existence	Scrambles data
Looks normal	Looks encrypted
Harder to detect	Easy to detect, hard to break

Image Steganography Tool

This **Image Steganography** tool is used to **hide secret data inside an image file** without visibly changing the image. To the naked eye, it's just a normal picture. To the right person? It's a locked whisper.

Classic technique. Ancient mindset. Modern execution.

What This Tool Does

This tool allows you to:

- **Embed (encode)** secret text or files into an image
- **Extract (decode)** hidden data from a stego-image
- Optionally **encrypt** the hidden data for extra security

It's not loud like encryption. It's subtle. And that's the point.

Working of the Tool (Step-by-Step)

1. Selecting the Cover Image

- You choose an image file (PNG recommended)
- This image becomes the **carrier** of the secret message
- Bigger image = more hiding capacity (physics never lies)

---

2. Input Data Selection

You can hide:

- **Text** (as shown: `hello`)
- **File** (documents, scripts, etc.)

This data is what gets secretly embedded into the image.

---

3. Steganography Mode

The tool offers multiple modes, but the most common is:

- **Embed**
  → Inserts hidden data inside the image's pixel data

Other modes (advanced):

- **Enlarge** – increases image size to fit more data
- **Difference** – shows pixel-level changes (for analysis)

---

4. Encode Operation

- You select **Encode**
- Optionally enable **Encrypt** (smart move—never trust obscurity alone)
- The tool modifies the **least significant bits (LSB)** of image pixels
  → Image looks the same, but binary data is altered internally

Old trick. Still undefeated.

---

5. Output Image Generation

- A new **stego-image** is created at the chosen output path
- This image:
  - Opens normally
  - Looks unchanged
  - Contains hidden data

Mission accomplished. No alarms triggered.

---

6. Decoding (Reverse Process)

To extract data:

1. Select the stego-image
2. Choose **Decode**
3. Enter password (if encrypted)
4. Hidden message/file is revealed

If the password's wrong? Silence. As it should be.

Why This Tool Matters

Practical Use Cases

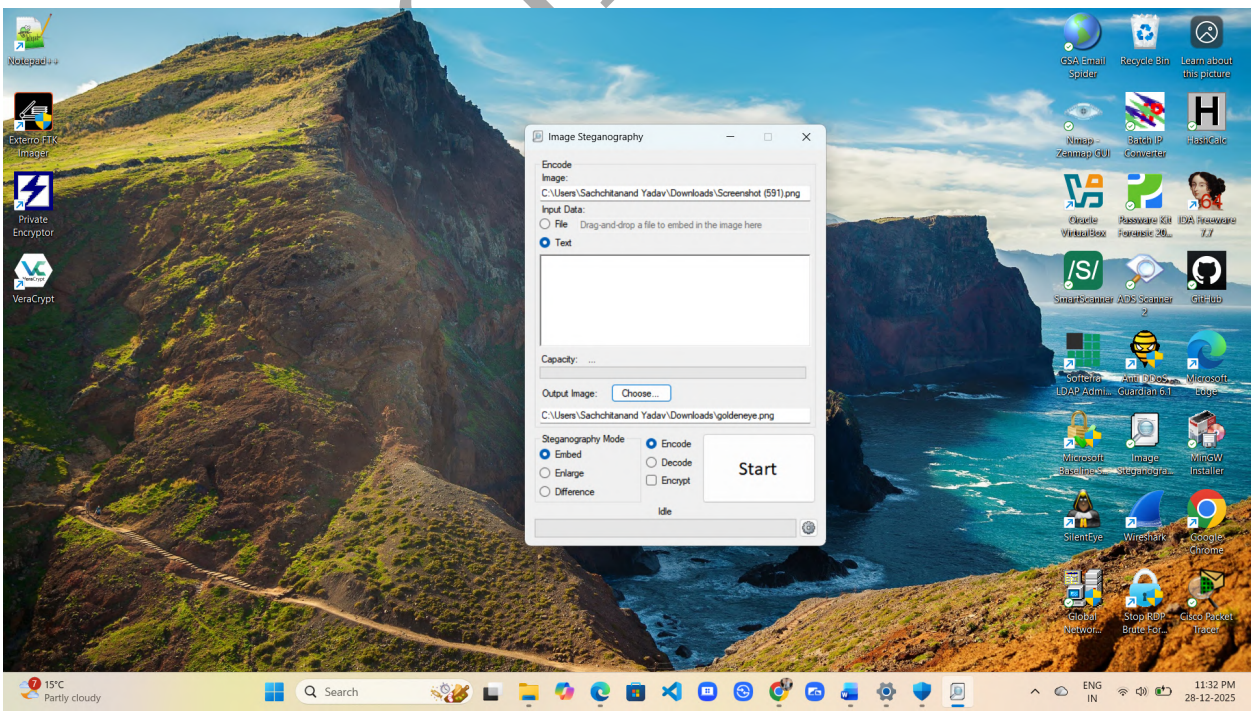
- Covert communication
- Digital watermarking
- Cyber forensics experiments
- Malware analysis research (yes, attackers use this too)
- Academic labs & cybersecurity training

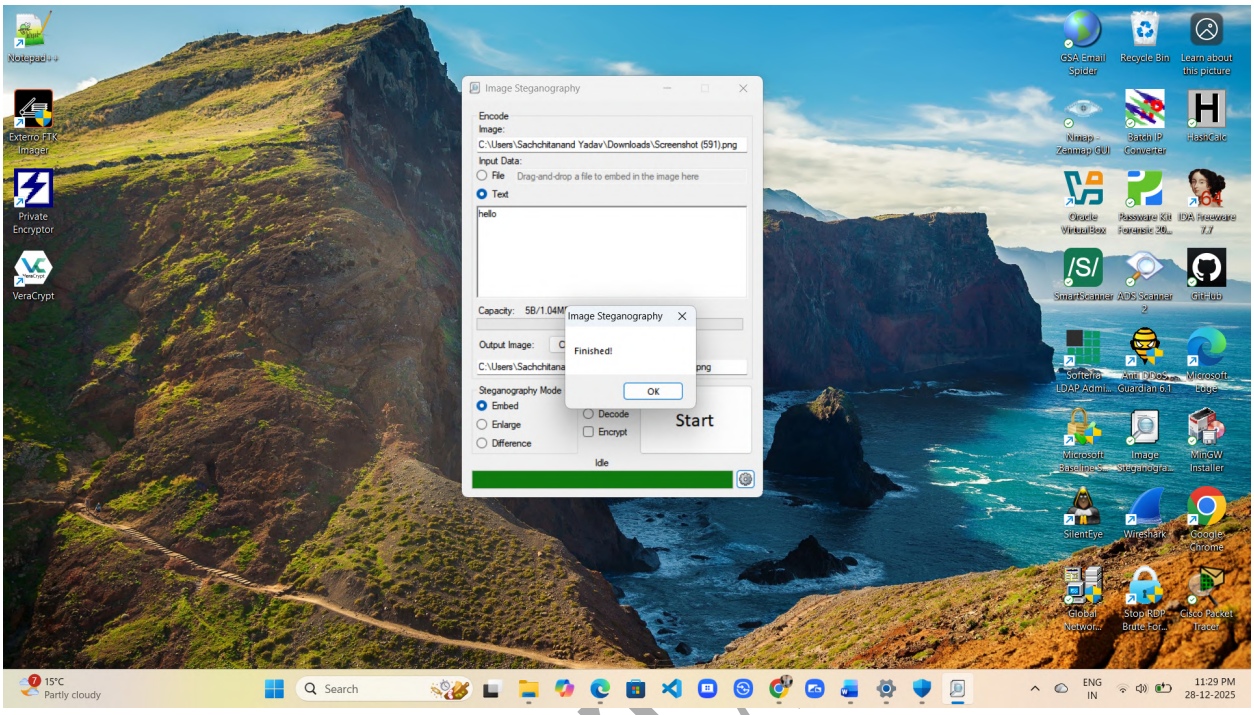
Let's be real—**steganography isn't about safety, it's about invisibility**.

Steganography vs Cryptography (Reality Check)

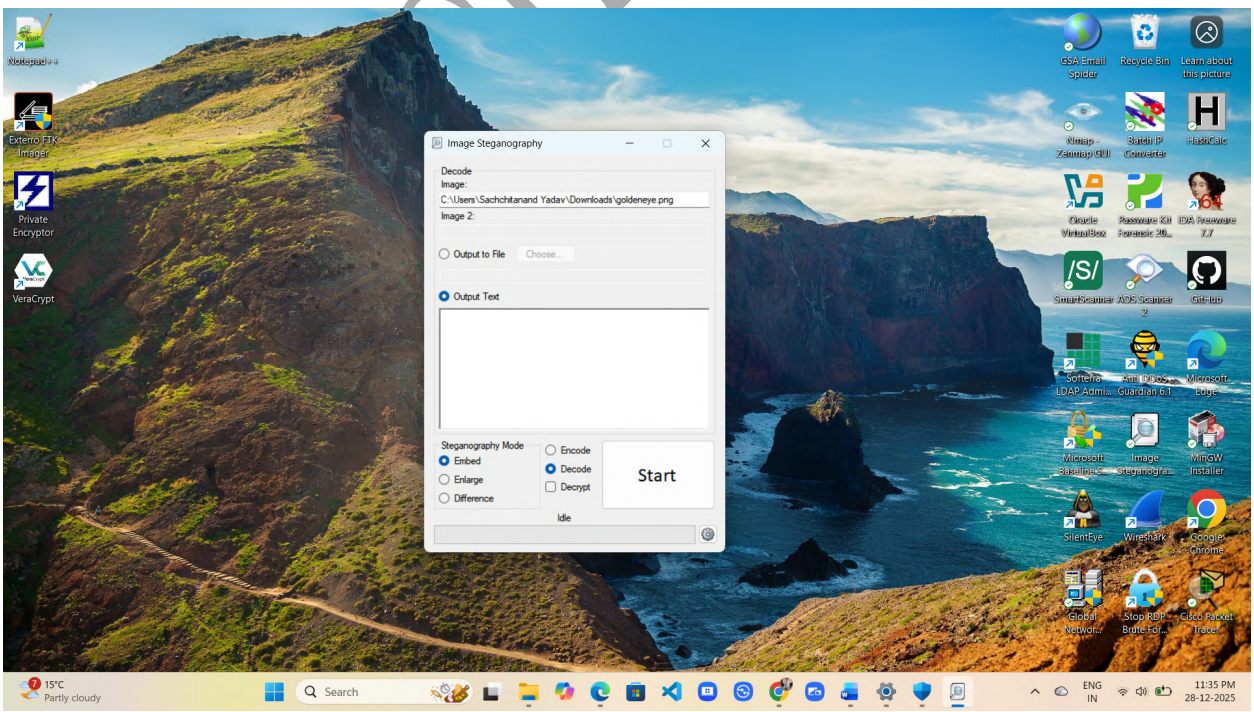
- Cryptography hides **content**
- Steganography hides **existence**

Encode Operation





Decode Operation



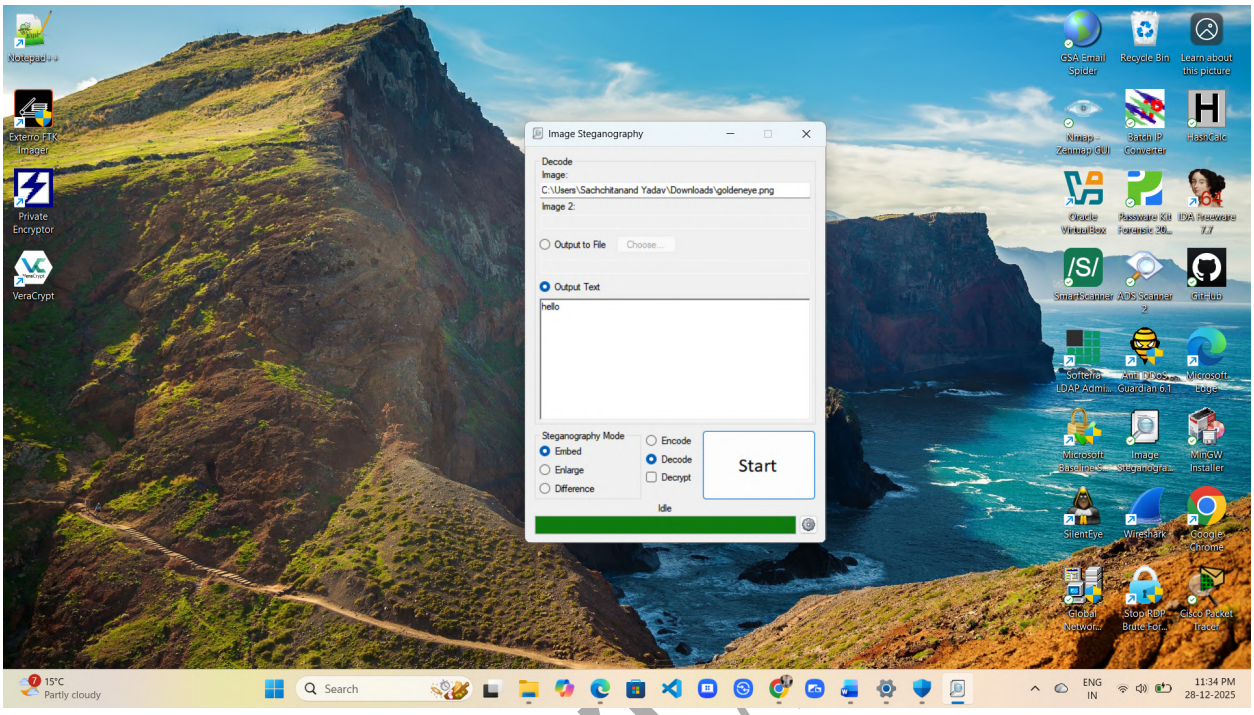


Image Steganography
Decode
Image:
C:\Users\Sachchitanand Yadav\Downloads\goldeneye.png
Image 2:
Output to File
Choose...
Output Text
hello
Steganography Mode
Embed
Enlarge
Difference
Encode
Decode
Decrypt
Start
Idle
Notepad++
Exterro FTK Imager
Private Encryptor
VeraCrypt
GSA Email Spider
Recycle Bin
Learn about this picture
Nmap - Zenmap GUI
Batch IP Converter
HashCalc
Oracle VirtualBox
Passware Kit Forensic 20...
IDA Freeware 7.7
SmartScanner
ADS Scanner 2
Git Hub
Softerra LDAP Admi...
Anti Ransomware Guardian 6.1
Microsoft Edge
Microsoft Baseline S...
Image Steganogra...
MinGW Installer
SilentEye
Wireshark
Google Chrome
Global Networ...
Stop RDP Brute For...
Cisco Packet Tracer
15°C
Partly cloudy
Search
ENG
IN
11:34 PM
28-12-2025

Steghide

Steghide is an open-source steganography tool used to **hide secret data inside image and audio files** in a way that makes detection genuinely hard. It supports **strong encryption**, compression, and integrity checking. No flashy GUI, just terminal truth.

If steganography had traditions, Steghide would be wearing them proudly.

Supported File Types

- **Images:** JPEG, BMP
- **Audio:** WAV, AU

Notice what's missing? PNG.
Yeah—because Steghide plays by structure, not vibes.

How Steghide Works (Under the Hood)

1. **Cover file selected**
   Image or audio acts as the carrier.
2. **Secret data embedded**
   Data is compressed, then **encrypted** (AES by default).
3. **LSB-based embedding**
   Data is hidden inside less noticeable parts of the file structure.
4. **Passphrase protection**
   Without the correct password, extraction is dead on arrival.

Old wisdom: *hide it, lock it, then forget it exists.*

Common Steghide Commands

Embed a file into an image

```
steghide embed -cf image.jpg -ef secret.txt
```

Extract hidden data

```
steghide extract -sf image.jpg
```

Check file capacity

```
steghide info image.jpg
```

Simple. Sharp. No babysitting.

Key Features

Feature	Description
Strong Encryption	Uses AES for data protection
Compression	Reduces size before embedding
Integrity Check	Detects tampering
CLI-based	Lightweight and script-friendly
Hard to Detect	Statistical steganalysis resistant

Use Cases (Real Ones)

- Covert communication
- Malware C2 hiding (research perspective—don't be dumb)
- Digital watermarking
- Cyber forensics labs
- CEH / security practice

Let's be honest: defenders study it because attackers love it.

Steghide vs GUI Stego Tools

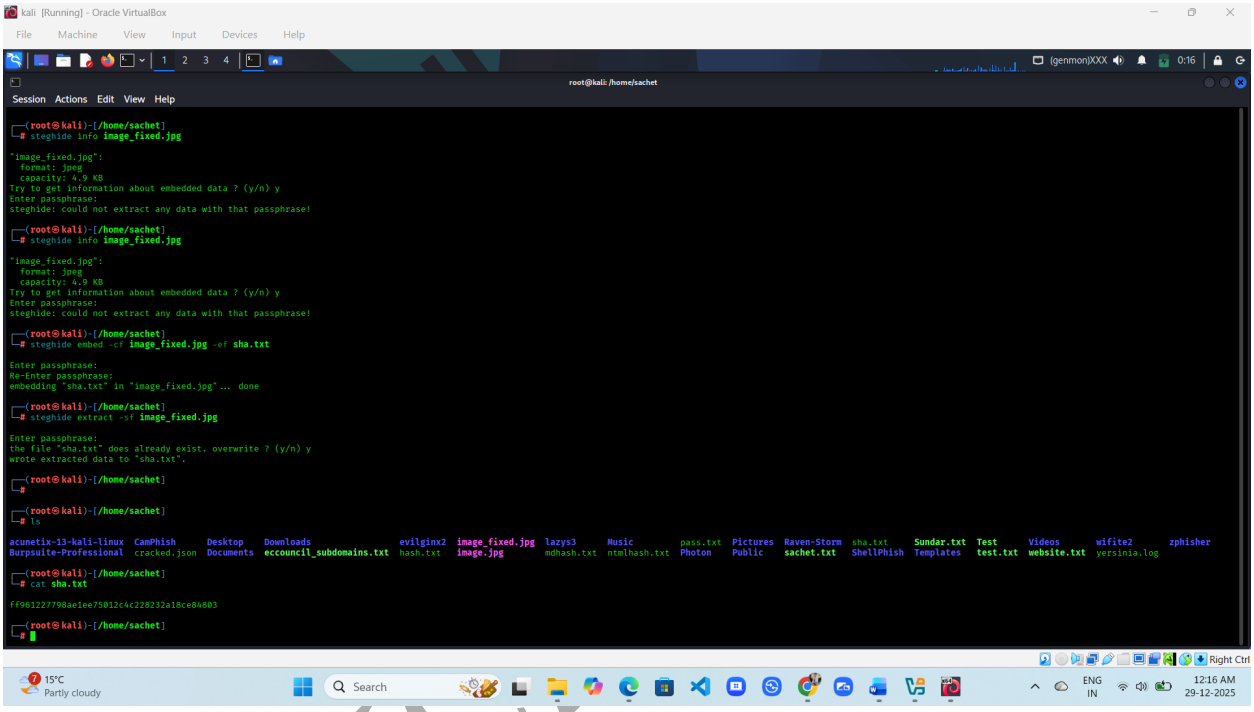
- GUI tools = beginner friendly, easier to spot
- Steghide = **silent, scriptable, serious**

Try Steghide

```
steghide info image_fixed.jpg
```

Then embed:

```
steghide embed -cf image_fixed.jpg -ef sha.txt
```



Extract the Hidden Data

```
steghide extract -sf image_fixed.jpg
```

What happens next:

- It asks for the **passphrase**
- Enter the **exact same password** you used while embedding
- If correct → file is extracted
- If wrong → silence (as designed)

Check Extracted File

List files:

```
ls
```

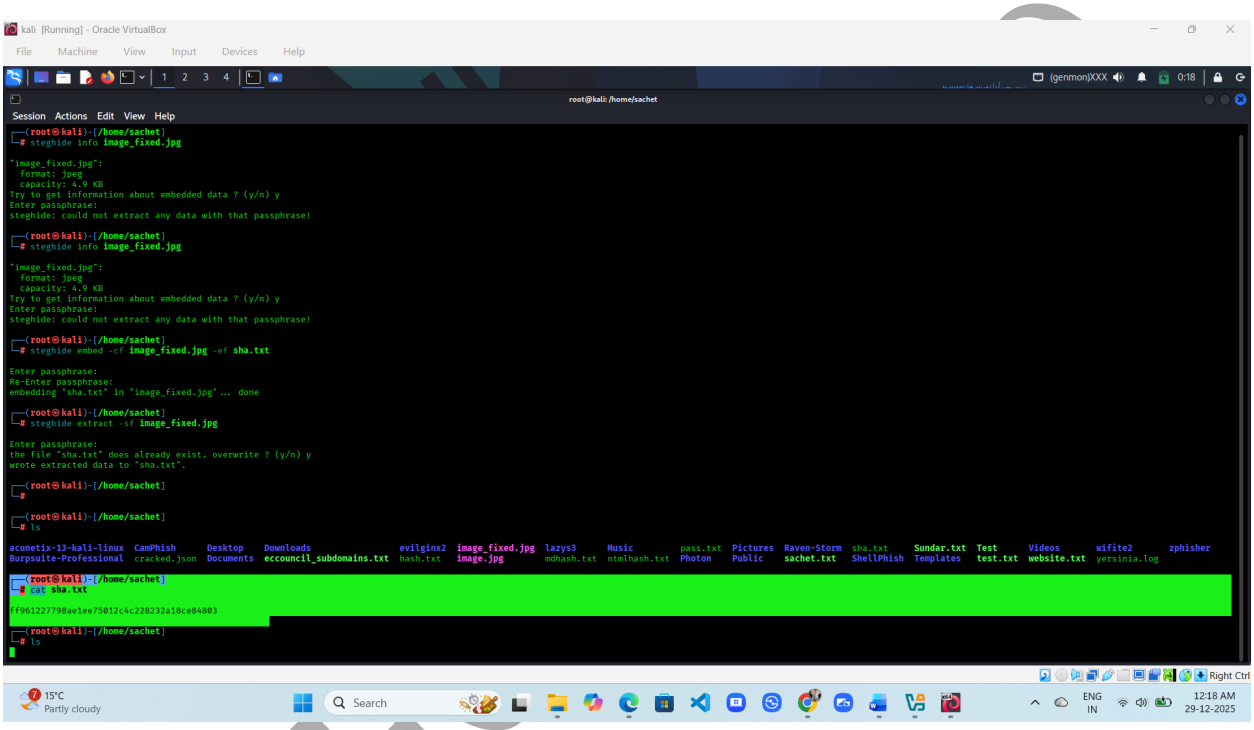
You should see:

```
sha.txt
```

View the Hidden Data

If it’s a text file:

```
cat sha.txt
```



Aperi'Solve

What is Aperi'Solve?

Aperi'Solve is an **online steganography analysis platform** designed to uncover hidden data inside images. No installs, no drama—just upload an image and let the machine start asking uncomfortable questions.

It's widely loved in:

- CTF challenges
- Digital forensics
- Ethical hacking labs
- Stego-curious minds who don't trust "just an image"

And honestly? You *shouldn't* trust images. Pixels lie.

What Does Aperi'Solve Do?

Aperi'Solve doesn't guess—it **investigates**. It runs multiple analysis techniques in parallel, because one tool alone is never enough (old rule, still true).

Key analysis features include:

- **Metadata Extraction**
  Pulls EXIF data like camera model, software used, timestamps—sometimes secrets hide in the boring stuff.
- **Strings Analysis**
  Scans for readable text buried inside the file. If there's ASCII hiding, it will snitch.
- **Bit Plane Analysis (LSB)**
  Examines Least Significant Bits—classic steganography territory. Ancient trick, still effective.
- **Color Channel Analysis**
  Separates RGB channels to spot anomalies. If one channel looks cursed, there's usually a reason.
- **Error Level Analysis (ELA)**
  Highlights areas with inconsistent compression—great for detecting tampering or embedded payloads.
- **File Signature Checks**
  Sometimes an image isn't *just* an image. ZIPs in disguise? PDFs wearing JPEG costumes? Aperi'Solve calls that out.

Why Aperi'Solve Matters

Let’s be real: manual stego analysis is slow, messy, and vibes-based. Aperi’Solve automates the boring grind so your brain can focus on **thinking**, not clicking.

It’s:

- Beginner-friendly (web-based, zero setup)
- Powerful enough for serious forensic work
- Perfect for students (yeah, especially cyber science folks 👀)

Still—don’t get lazy. Tools assist; they don’t replace intuition. The old-school mindset still wins.

Limitations (No Sugar-Coating)

Aperi’Solve won’t:

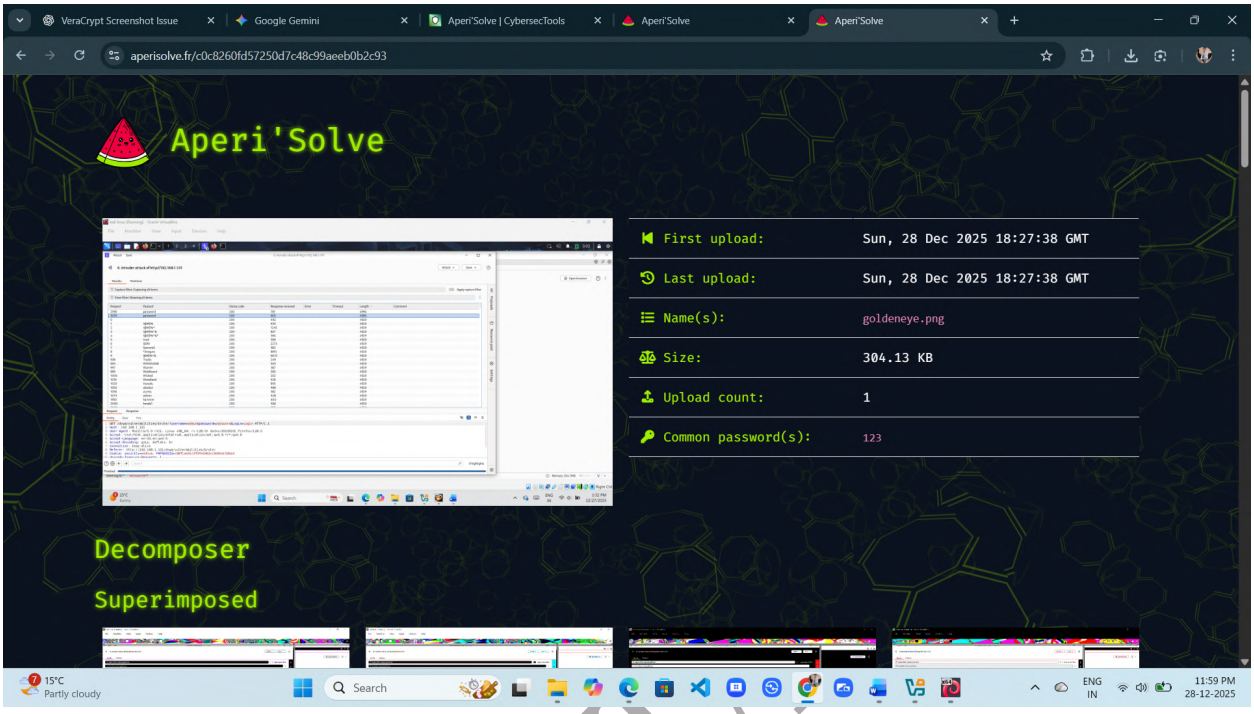
- Crack encrypted payloads for you
- Magically guess passwords
- Solve every stego puzzle

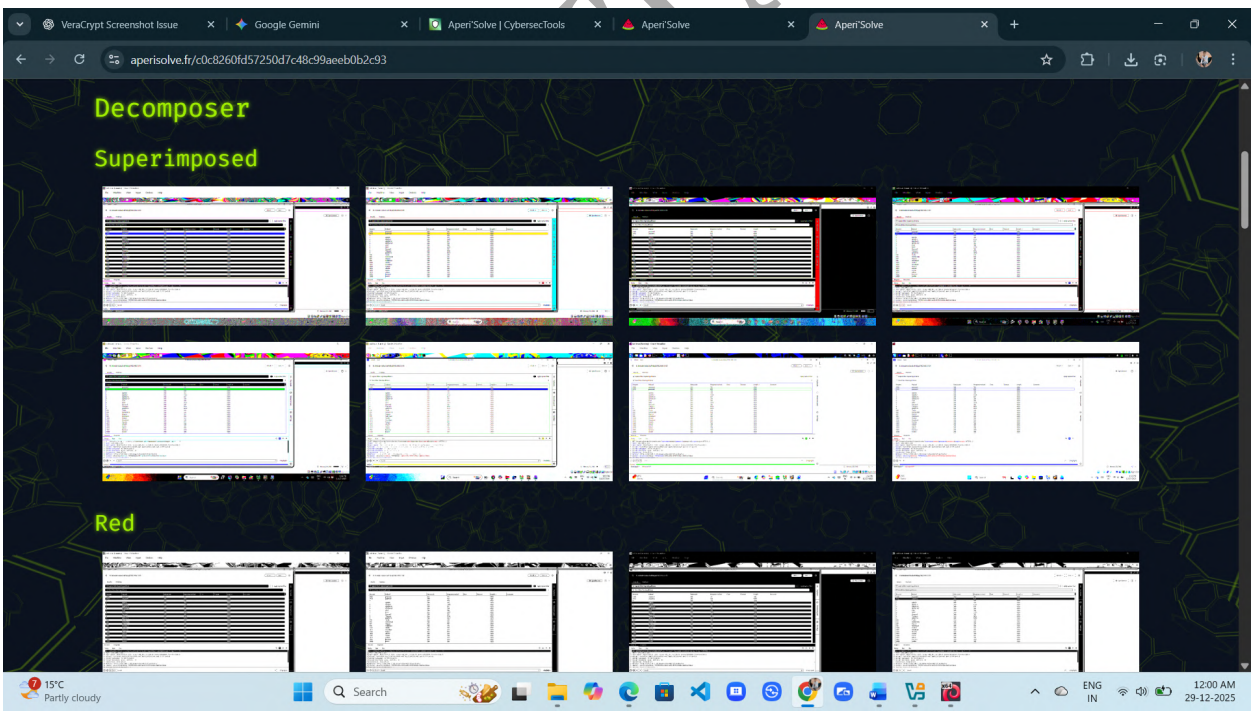
How to use –

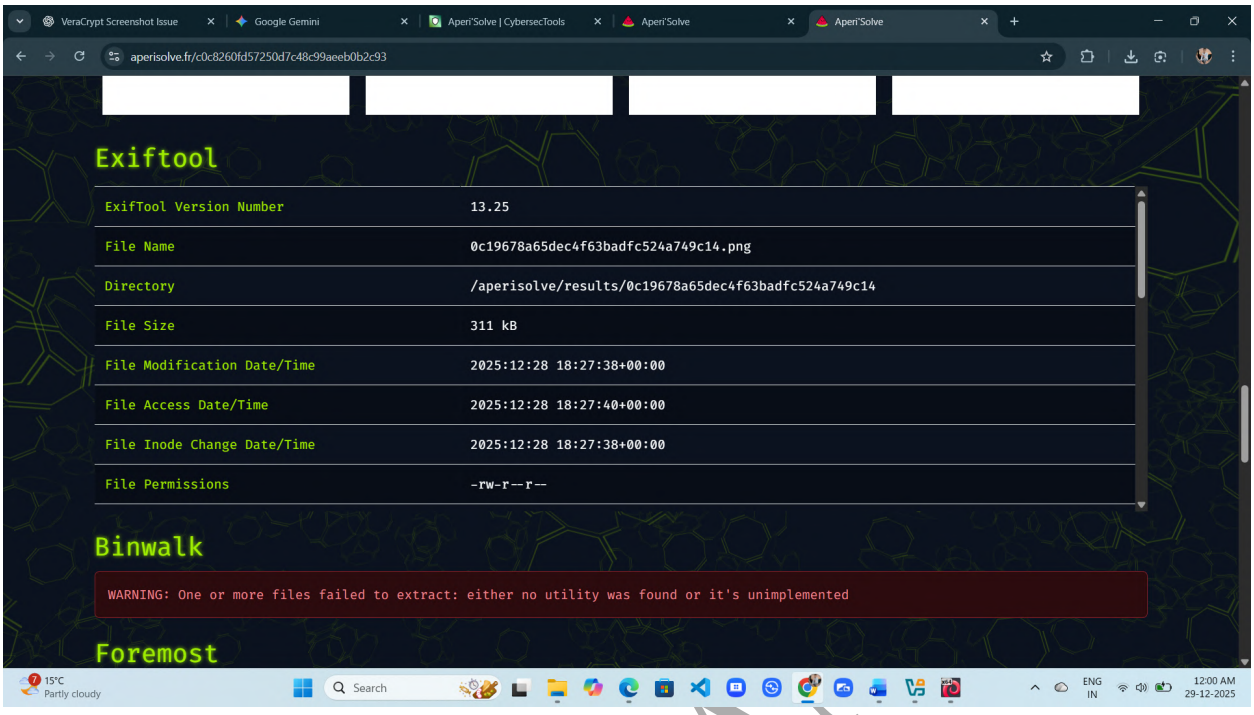
- Open Aperi’Solve website

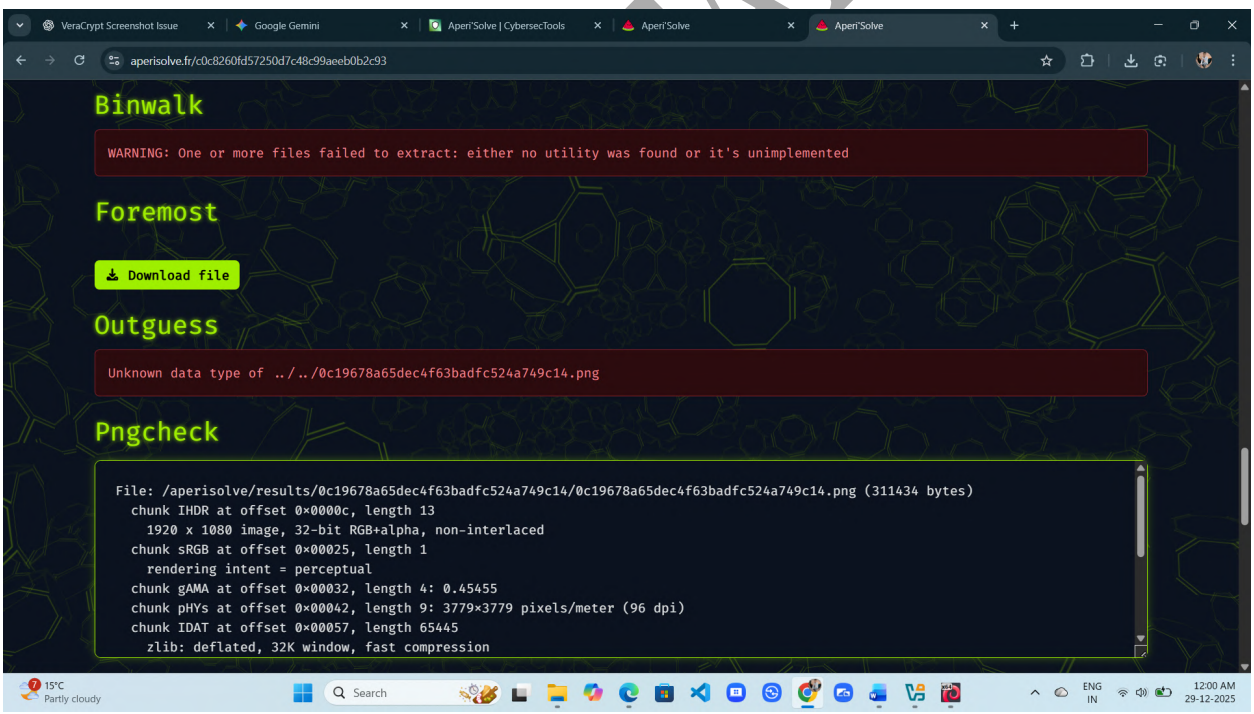
- Choose an image and click on submit button

Results -


VeraCrypt Screenshot Issue
Google Gemini
Aperi'Solve | CybersecTools
Aperi'Solve
Aperi'Solve
aperisolve.fr/c0c8260fd57250d7c48c99aeeb0b2c93
Aperi'Solve
First upload: Sun, 28 Dec 2025 18:27:38 GMT
Last upload: Sun, 28 Dec 2025 18:27:38 GMT
Name(s): goldeneye.png
Size: 304.13 KB
Upload count: 1
Common password(s): 123
Decomposer
Superimposed
15°C
Partly cloudy
Search
ENG
IN
11:59 PM
28-12-2025


VeraCrypt Screenshot Issue
Google Gemini
Aperi'Solve | CybersecTools
Aperi'Solve
Aperi'Solve
aperisolve.fr/c0c8260fd57250d7c48c99aeeb0b2c93
Decomposer
Superimposed
Red
15°C
Partly cloudy
Search
ENG
IN
12:00 AM
29-12-2025


VeraCrypt Screenshot Issue
Google Gemini
Aperi'Solve | CybersecTools
Aperi'Solve
Aperi'Solve
aperisolve.fr/c0c8260fd57250d7c48c99aeeb0b2c93
Exiftool
ExifTool Version Number 13.25
File Name 0c19678a65dec4f63badfc524a749c14.png
Directory /aperisolve/results/0c19678a65dec4f63badfc524a749c14
File Size 311 kB
File Modification Date/Time 2025:12:28 18:27:38+00:00
File Access Date/Time 2025:12:28 18:27:40+00:00
File Inode Change Date/Time 2025:12:28 18:27:38+00:00
File Permissions -rw-r--r--
Binwalk
WARNING: One or more files failed to extract: either no utility was found or it's unimplemented
Foremost
15°C
Partly cloudy
Search
ENG
IN
12:00 AM
29-12-2025


VeraCrypt Screenshot Issue
Google Gemini
Aperi'Solve | CybersecTools
Aperi'Solve
Aperi'Solve
aperisolve.fr/c0c8260fd57250d7c48c99aeeb0b2c93
Binwalk
WARNING: One or more files failed to extract: either no utility was found or it's unimplemented
Foremost
Download file
Outguess
Unknown data type of ../../0c19678a65dec4f63badfc524a749c14.png
Pngcheck
File: /aperisolve/results/0c19678a65dec4f63badfc524a749c14/0c19678a65dec4f63badfc524a749c14.png (311434 bytes)
chunk IHDR at offset 0×0000c, length 13
1920 x 1080 image, 32-bit RGB+alpha, non-interlaced
chunk sRGB at offset 0×00025, length 1
rendering intent = perceptual
chunk gAMA at offset 0×00032, length 4: 0.45455
chunk pHYs at offset 0×00042, length 9: 3779×3779 pixels/meter (96 dpi)
chunk IDAT at offset 0×00057, length 65445
zlib: deflated, 32K window, fast compression
15°C
Partly cloudy
Search
ENG
IN
12:00 AM
29-12-2025

Cryptography Countermeasures

Cryptography countermeasures are techniques and best practices used to **protect encrypted data from attacks**, misuse, and cryptographic weaknesses. The goal is simple:
👉 *even if attackers see the data, they can't read it, alter it, or fake it.*

No shortcuts. No excuses.

---

1. Use Strong Encryption Algorithms

Retire weak algorithms like they're outdated phones.

- ❌ DES, MD5, SHA-1
- ☑ AES-256, RSA-2048+, ECC, SHA-256/512

Old rule: if it's broken in theory, it's broken in practice.

---

2. Strong Key Management

Encryption is only as strong as the **key**, not the math.

Countermeasures include:

- Secure key generation
- Safe key storage (HSMs, TPMs)
- Regular key rotation
- Never hard-coding keys in source code

Lose the key = lose the kingdom.

---

3. Use Proper Key Lengths

Short keys are an open invitation.

- AES → 128/256 bits
- RSA → minimum 2048 bits
- ECC → smaller keys, same strength

Brute force doesn't care about your feelings.

---

4. Secure Password Practices

Passwords protect keys. Weak passwords ruin everything.

Countermeasures:

- Enforce strong password policies
- Use salting + hashing
- Apply key-stretching (PBKDF2, bcrypt, scrypt)

If your password is "admin123", stop reading and rethink life.

---

5. Digital Signatures & Integrity Checks

Encryption hides data. **Integrity proves it wasn't altered.**

- Use digital signatures
- Hash data before and after transmission
- Detect tampering and forgery

Silence attackers. Expose manipulation.

---

6. Proper Random Number Generation

Bad randomness = predictable crypto = instant failure.

Countermeasures:

- Use cryptographically secure random generators
- Avoid predictable seeds
- Never roll your own randomness

Randomness is sacred. Treat it that way.

---

7. Secure Communication Protocols

Encryption without secure protocols is cosplay.

- Use HTTPS, TLS, SSH, IPsec
- Disable weak cipher suites
- Avoid plain-text transmission

Tradition says: encrypt the channel, not just the data.

---

8. Protect Against Side-Channel Attacks

Attackers don't always break crypto—they **observe it**.

Countermeasures:

- Constant-time algorithms
- Hardware protections
- Power and timing attack resistance

If math won't leak, physics might.

---

9. Regular Updates & Cryptographic Audits

Crypto ages. Threats evolve.

- Update libraries regularly
- Replace deprecated algorithms
- Perform security audits

Yesterday's strong encryption is today's homework problem.

Summary: Cryptography Module

Cryptography is the science of **securing information** by transforming readable data into an unreadable form to protect it from unauthorized access. This module explains how cryptography ensures **confidentiality, integrity, authentication, and non-repudiation** in digital communication.

The module covers:

- **Basic cryptographic concepts** such as plaintext, ciphertext, encryption, and decryption
- **Types of cryptography**, including symmetric key cryptography (AES), asymmetric key cryptography (RSA, ECC), and hash functions (MD5, SHA family)
- **Cryptographic algorithms and protocols** used to secure data at rest and in transit
- **Digital signatures and certificates** for authentication and integrity verification
- **Key management practices**, including key generation, storage, and rotation
- **Cryptographic attacks** and their **countermeasures**
- Practical applications like **SSL/TLS, secure email, disk encryption, password protection, and data integrity checking**

The module stresses that strong cryptography depends not only on algorithms, but also on **proper implementation, secure key handling, and regular updates**.

THANK YOU