ALMV

Security Testing Documentation

Combined Cryptographic and Steganographic Techniques for Image Metadata Security Algorithm

A Thesis Project

Version 1.0 – Offline/ Path-Based

Overview

Our thesis project, "**Combined Cryptographic and Steganographic Techniques for Image Metadata Security Algorithm**" presents an algorithm designed to secure sixteen privacy-sensitive EXIF metadata fields through a freeware application, named ALMV. This is a hybrid cryptographic workflow that hashes a user-provided password, extracts the specified metadata, and then encrypts them alongside with the steganographic workflow that identifies the busiest block within the image based on Shannon Entropy and embeds the encrypted metadata into that block through LSB manipulation.

Prerequisites and Setup

Hardware/software requirements	OS: Windows 10/11 **Note:** *You can still use other OS for testing purposes only*
Test environment isolation	All tests MUST be run in an isolated VM or Docker container
Assets	Assets include the freeware application and test-images found in the sent repository.

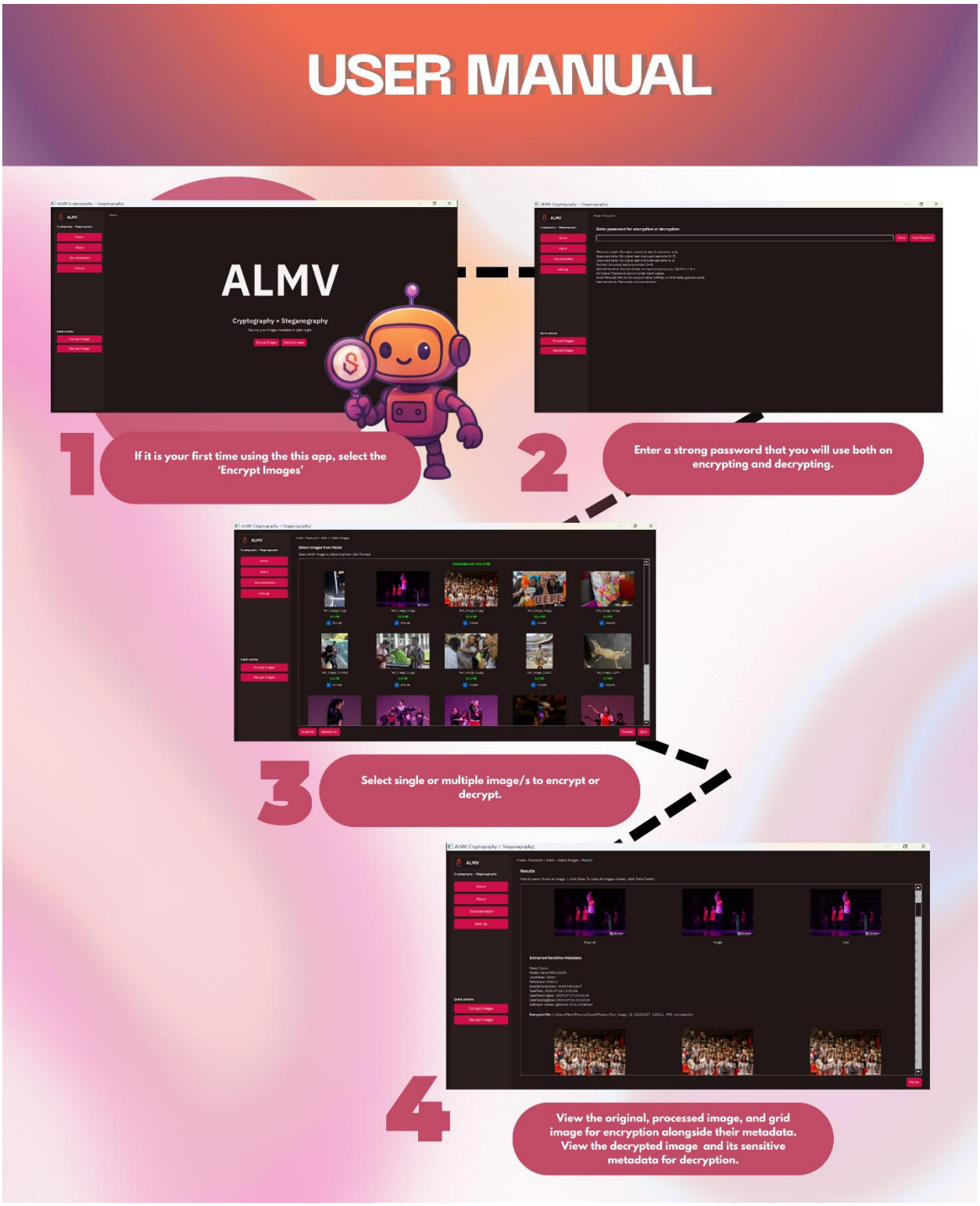
Running ALMV.exe (Path-Based)

1. Access this Github Release https://github.com/haijin2/thchoish-landing-page-p/releases/tag/v.1.1
2. Click and download the **almv.exe**
3. Double-click the application.
4. If via command prompt:

   cd [Your file path where you saved the freeware]

almv.exe

5. For reference on how to navigate the freeware, shown below is the user manual.



Scope and Rules of Engagement

System	When you download the installer, if you save it on the Programsx86, access: ALMV > ALMV > ALMV [Type: Application]
Allowed Testing	Dynamic Application Security Testing (DAST) from outside application; local Virtual Machine testing; offline file analysis; and cryptanalysis on offline data provided. Any other image formats and AI-generated images can be added as test images as well.
Forbidden	Attacks against CI/CD, developer machines, recovery of other testers' results, social engineering, unauthorized reverse engineering, denial-of-service, or public disclosure of vulnerabilities before coordinated disclosure.
Authorized Dates	**October 28 – November 8, 2025**. The testers must notify before testing.
Tools allowed	Any DAST tools.

Reporting

- Use the included report template and email to samboa.angela@ue.edu.ph with the subject "ALMV Security Test Report - [Surname of the Tester]"
- Report template is provided. Though, do not forget to include the summary, steps to reproduce, screenshots/logs, environment (OS, VM software).

Cleanup & Proof of Deletion

- All test assets, including executables, encrypted files, screenshots, reports, and VM snapshots, must be permanently deleted within seven (7) calendar days following test completion or by November 11, 2025 (whichever comes first).
- Deletion must include the *C:\ALMV_Test* folder, any derived files, and all temporary directories used for testing.
- Testers must submit a Deletion Confirmation Statement via email to *samboa.angela@ue.edu.ph*, including:
  - Date and time of deletion
  - Name of tester
  - Confirmation that no residual data or backups remain
  - Optional: Screenshot proof of deletion (e.g., Recycle Bin emptied confirmation)

Methodology

The STRIDE Methodology, focusing primarily on spoofing (unauthorized decryption attempts), and information disclosure (key leakage) are preferably to be performed during security tests. From the defined methodology, there are eight (8) attack simulations/scenarios suggested. And these are as follows:

1. **Normal key exchange and distribution.**
   - The client successfully runs a complete process of key exchange, image encryption, steganography embedding, and decryption without any intrusion.
   - This demonstrates that cryptographic and steganographic processes work when not under attack. This established a baseline to confirm the algorithm's reliability and functional integrity in uncompromised conditions.
2. **Key exchange sniffing**

- A passive attacker interrupts the initial key exchange where the keys are distributed.
- This checks if the captured public key data can be used to compromise the ongoing session.

3. **Key distribution sniffing**
    - A passive attacker intercepts the encrypted channel during the short-term session key distribution.
    - This attack simulation identifies whether there are initial key leakage flaws using the bitwise similarity to determine how much the stolen key compromises the real key.
4. **Cryptanalysis**
    - An attacker attempts to find either the key and plaintext by analyzing a large number of encrypted images without intercepting the key exchange.
    - This assesses the computation difficulty of the algorithm security layer offline against brute-force attacks.
5. **“Get encrypted image file, decrypt later” attack**
    - An attacker gets the stego-image file alongside its encrypted metadata and keys to decrypt it later. It is an attempt to test if they can compromise the long-term private RSA key.
    - This test ensures that the risked long-term key does not automatically reveal the past session information.
6. **Man-in-the-middle attack**
    - An attacker replaces their own public key in the initial exchange and mimics the authorized recipient to the sender.

- This checks if the server can detect the forged key and successfully abort the communication before any secure data is sent.

7. **Forward secrecy check**
    - A control scenario after the process ends. This verifies that even if the long-term private key is vulnerable, the previous session data remains safe.
    - This test is interrelated and confirmed to the “Get encrypted image file, decrypt later” simulation.
8. **Multivector attack**
    - An attacker uses hybrid attack techniques to compromise the system. This evaluates how secure the algorithm system is designed across various layers of defense when challenged simultaneously.

Note: The attack simulations above are just suggestions. Other security tests are welcome except denial-of-service attacks, social engineering attacks, testing outside the scope, public disclosure of vulnerabilities, and unauthorized reverse engineering. Feel free to use the template provided for your report documentation.

----------END OF DOCUMENTATION----------