**Test Report**

**for**

**Password & Encryption Detector**

Version 2.0

by Sertaç Ataç & Ramazan Bağış

02.07.2025

**Table of Contents:**

[**1. Introduction 3**](#_1d5c5loa5585)

[1.1 Purpose 3](#_wek6h4u5xefo)

[1.2 Scope 3](#_xt3hwjdvlkq7)

[**2. Test Environment 3**](#_nlx8xvqciecs)

[2.1 Hardware and Software Configuration 3](#_qp8j3xku3z46)

[2.2 Test Tools 3](#_mfem795dne66)

[**3. Test Strategy 4**](#_g52wnr1tmf8v)

[3.1 Test Types 4](#_uy2guknjdf4p)

[3.1 Entry and Exit Criteria 4](#_lxr5mvw1o5av)

[**4. Test Results 4**](#_1pdb47doz0l7)

[4.1 Functional Test 4](#_xr6jwla9mf9)

[4.2 Interface Test 4](#_ta1eia9o6siy)

[4.3 Performance Test 5](#_y0eg1xm7rowl)

[4.4 Robustness Test 5](#_hzf0ktpgt37d)

[4.5 Summary Table 5](#_3t8svpn94r2r)

[**5. Defect Log 5**](#_cicanr9yn39k)

[**6. Conclusion 5**](#_7kfk4xguqfpm)

### **1. Introduction**

#### **1.1 Purpose**

This Test Report documents the verification and validation results for the "Password & Encryption Detector" software. It ensures that the system meets the functional, performance, and quality requirements defined in the Software Requirements Specification (SRS).

#### **1.2 Scope**

This document covers the results of testing the following functionalities:

* File processing in both single and batch modes
* File type identification via Magika
* Format-specific password and encryption analysis
* Entropy-based fallback analysis
* Result reporting including status, encryption, confidence score, and execution time
* Performance in both asynchronous and synchronous modes

### 

### **2. Test Environment**

#### **2.1 Hardware and Software Configuration**

* **Operating System:** Windows 11 Pro 64 bit
* **Python Version:** 3.8+
* **Libraries Used:** msoffcrypto, PyPDF2, pikepdf, rarfile, py7zr, pypff, and olefile (These libraries are used as-is and not modified)
* **System Specs:**
  + CPU: AMD Ryzen 7 7840HS w/ Radeon 780M Graphics (16 CPUs), ~3.8GHz
  + RAM: 32768 MB
  + Free Storage: 411 GB on 1 TB Kingston SNV2S1000G SSD

#### **2.2 Test Tools**

* Built-in Python time library
* Custom CLI test scripts
* Performance monitoring on Windows Task Manager

### **3. Test Strategy**

#### **3.1 Test Types**

* **Functional Testing**: Testing individual handlers and modules (file\_handlers.py, entropy.py, etc.)
* **Interface Testing**: Testing command line interactions
* **Performance Testing**: Timed analysis for large datasets in both sync and async modes
* **Robustness Testing**: Testing behaviour of the program with results on edge cases

### 

### **4. Test Results**

#### **4.1 Functional Test**

* **File Type Detection:** All supported formats (pdf, csv, xlsx, pptx, docx, ods, tsv, rar, zip, odm, odp, odt) are correctly identified.
* **Success Rates:**
  + **First test files (**Total 225 Test files: **“pdf, xlsx, pptx, docx,rar, zip”** 30 per file formats 15 password protected and 15 not protected, 15 per file formats **“ods, csv, tsv”** 15 not password protected**):** 
    - **Password-protected files:** 100% detection (all 15 files correctly identified)
    - **Not password-protected files:** 100% correct identification (all 15 files)
  + **Second test files (**1 by each:ODM, ODP, ODS, ODT**):** 100% detection
* **Exceptions:** 
  + 3 PPTX files incorrectly reported as "NOT PASSWORD PROTECTED" with confidence=0.30
  + 1 XLSX file incorrectly reported as "NOT PASSWORD PROTECTED" with confidence=0.30

#### **4.2 Interface Test**

* **CLI Behaviour:**
  + Single file mode: Success (logasync4.txt, logsync4.txt)
  + Batch mode: Success (all directory scans completed)
  + Sync/Async modes: Both operational with expected performance differences

#### **4.3 Performance Test**

| **Test Case** | **Total File Size** | **Total File Amount** | **Mode** | **File Types** | **Average Time** | **Total Time** | **Accuracy Rate** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| #1 | 2,13 GB | 225 | Async | pdf, xlsx, pptx, docx, rar, zip, ods, csv, tsv | 3.215 ms | 0.7234 s | 97% |
| #1 | 2,13 GB | 225 | Sync | pdf, xlsx, pptx, docx, rar, zip, ods, csv, tsv | 7.241 ms | 1.6293 s | 97% |
| #2 | 1,33 MB | 7 | Async | odm, docx, odp, ods, odt, rar | 23.257 ms | 0.1628 s | 100% |
| #2 | 1,33 MB | 7 | Sync | odm, docx, odp, ods, odt, rar | 14.557 ms | 0.1019 s | 100% |
| #3 | 168 GB | 1650 | Async | pdf, xlsx, pptx, docx, rar, zip, ods, csv, tsv, py, exe, c, cpp, txt, class, java, md, html, css, mp3, mp4, gif, jpg, mp4, vdi, vbox, log, 7z, pfx, vmdk, vmxf, nvram, odp, odm, odt | 2.528 ms | 4.1721 s | 98% |
| #3 | 168 GB | 1650 | Sync | pdf, xlsx, pptx, docx, rar, zip, ods, csv, tsv, py, exe, c, cpp, txt, class, java, md, html, css, mp3, mp4, gif, jpg, mp4, vdi, vbox, log, 7z, pfx, vmdk, vmxf, nvram, odp, odm, odt | 8.432 ms | 13.9133 s | 98% |
| #4 | 1,03 GB | 1 | Async | zip | 83 ms | 0.0837s | 100% |
| #4 | 1,03 GB | 1 | Sync | zip | 80 ms | 0.0800s | 100% |

#### **4.4 Robustness Test**

* **Special Cases:**
  + LibreOffice formats (ODM, ODP, ODS, ODT): All correctly handled
  + Mixed archive types (ZIP, RAR): 100% accuracy
  + Very small files (<1KB): No failures observed
  + Very big files (>1GB): No failures observed

#### **4.5 Summary Table**

| **Test Area** | **Total Cases** | **Passed** | **Failed** | **Comments** |
| --- | --- | --- | --- | --- |
| Functional Testing | 8 | 6 | 2 | 4 False Negatives (3 pptx 1 xlsx), otherwise all test cases are clear |
| Interface Testing | 8 | 8 | 0 | All CLI modes functional |
| Performance Testing | 8 | 8 | 0 | Async is faster when file size/number increases, buy Sync is faster with small size/number files |
| Robustness Testing | 8 | 8 | 0 | Edge cases handled properly |

### 

### **5. Defect Log**

* PPT/XLSX False Negatives:
  + Severity: Medium
  + Files: sifreli (12/13/15).pptx, sifreli (15).xlsx
  + Status: Unresolved
  + To-Do’s: OfficeOpenXMLHandler's PPT/XLSX detection logic can be modified

### 

### **6. Conclusion**

The detector shows higher than 95% accuracy for most file types (PDF, DOCX, XLSX, archives).

Async mode is much more faster when big amount of files (>10) are processed.

Sync mode is significantly faster when small amount of files (<10) are processed.