1st_Excel Row Encryption Tool Documentation

Overview

This Python script automates the encryption of individual rows from an Excel file using **AES-256-CBC encryption** via OpenSSL. Each row is encrypted with a unique 3-character password, and a log file records the passwords for decryption purposes.

Components

main.py - Excel Row Encryption Script

Purpose: Encrypts each row of an Excel file separately for secure storage or transmission.

Features:

- Reads an Excel file (`labtest2MarkNoName.xlsx` by default).
- Generates a **random 3-character password** (A-Z, a-z, 0-9) for each row.
- Encrypts each row as a separate file using **OpenSSL (AES-256-CBC)**.
- Maintains a **log file** (`encryption_log.csv`) mapping encrypted files to their passwords.
- Automatically cleans up temporary files after encryption.

Usage

Prerequisites

- Python 3.x
- Required libraries:CopyDownload

  bash

  ```
  pip install pandas
  ```

- OpenSSL installed on the system (usually pre-installed on Linux/macOS; available for Windows via OpenSSL).

Running the Script

1. Place the Excel file (`labtest2MarkNoName.xlsx`) in the same directory as the script.
2. Execute:

   ```
   python main.py
   ```

Output

- A directory `encrypted_rows/` containing:
  - Encrypted files (`encrypted_row_0.enc`, `encrypted_row_1.enc`, etc.).
  - A log file (`encryption_log.csv`) with filenames and passwords.

Technical Details

Password Generation

- **3-character passwords** (e.g., `aB3`, `xY9`) are randomly generated from:

  ```
  string.ascii_letters + string.digits  # A-Z, a-z, 0-9 (62 possible characters)
  ```

Encryption Process

1. Each Excel row is saved as a temporary CSV file.
2. Encrypted using OpenSSL:

   ```
   openssl enc -aes-256-cbc -salt -in [input] -out [output] -k [password] -pbkdf2
   ```

   - **AES-256-CBC**: Strong encryption standard.
   - **PBKDF2**: Password-Based Key Derivation Function 2 (enhances security).

Log File Structure

Column	Description
`filename`	Name of the encrypted file (e.g., `encrypted_row_0.enc`).
`password`	3-character password used for decryption.

Security Considerations

- **Short Passwords**: The 3-character passwords (62^3 = ~238k combinations) are **weak** for real-world use but sufficient for demonstration.
- **Log File**: The `encryption_log.csv` must be **protected** as it contains decryption keys.
- **OpenSSL Security**: Uses **salt** and **PBKDF2** to resist brute-force attacks.

Example Workflow

1. **Input Excel File**:

Name	Marks
Alice	85
Bob	72

2. **Encrypted Output**:
   - `encrypted_row_0.enc` (Password: `aB3`)
   - `encrypted_row_1.enc` (Password: `xY9`)
3. **Log File**:CopyDownload

   csv

   ```
   filename,password
   encrypted_row_0.enc,aB3
   encrypted_row_1.enc,xY9
   ```

Limitations

- **Password Length**: 3-character passwords are **not secure** for sensitive data.
- **No Decryption Script**: Users must manually decrypt files using OpenSSL.
- **Single-File Dependency**: Assumes the Excel file is named `labtest2MarkNoName.xlsx`.

Future Enhancements

1. **Configurable Password Length**: Allow longer passwords via command-line arguments.
2. **Automated Decryption**: Add a companion script for decryption.
3. **GUI Interface**: Simplify usage for non-technical users.

This tool is ideal for **educational purposes** or scenarios where lightweight row-level encryption is needed. For stronger security, consider increasing the password length or integrating with a key management system.