UNIVERSITATEA TEHNICA “GHEORGHE ASACHI” din IASI
FACULTATEA DE AUTOMATICA SI CALCULATOARE
DOMENIUL: Calculatoare si Tehnologia Informatiei
SPECIALIZAREA: Tehnologia Informatiei

Sistem local de management al cheilor de criptare pentru mai multi algoritmi si management de fisiere criptare/decriptare

Coordonator,
s.l.dr.ing
A.Tudorache

Student,
Bulgaru Vlad-Andrei,
Grupa 1409A

Iasi, 2025

Baza de date:

Sistemul utilizeaza o imagine docker pentru baza de date MariaDB. Am numit baza de date locala “proiect”, ce contine urmatoarele tabele:

- tabela “security_keys”, ce gestioneaza cheile de criptare generate. Aceasta tabela include campurile:
  - -id;
  - -encryption_method;
  - -secret_key;
  - -key_status;
- tabela “secured_files”, ce tine evidenta fisierelor ce au fost criptate, avand campurile:
  - -id;
  - -file_title;
  - -encryption_method;
  - -key_ref_id;
  - -file_size;
  - -encryption_status;
  - -hash_value;
- tabela “performance_metrics”, ce stocheaza informatii despre performanta criptarii/decriptarii, avand campurile:
  - -file_ref_id;
  - -encryption_duration;
  - -decryption_duration;
  - -memory_usage;
  - -cpu_usage

Aceste tabele permit localizarea fiecarei criptari si masurarea performantei acestora.

Algoritmii de utilizare alesi:

Am ales algoritmii AES-256-CBC si RSA-2048. **Algoritmul AES-256-CBC** este un algoritm de criptare simetric(aceeasi cheie este folosita si la criptare si la decriptare).

Partea “CBC” a denumirii (cipher block chaining) se refera la faptul ca modul de cifrare in bloc care este utilizat cu algoritmul. Fiecare bloc plaintext va face operatia de XOR cu blocul de text cifrat antrerior inainte de a fi decriptat, asigurandu-se faptul ca textul cifrat rezultat nu va contine niciun model repetat.

```
subprocess.run([
    openssl_path, "enc", "-aes-256-cbc", "-salt",
    "-in", file_path, "-out", output_file,
    "-pass", "file:./key.bin"
])
```

Cheia este generata folosind Fernet.generate_key() si convertita pentru a fi folosita de OpenSSL.

Algoritmul RSA-2048 este un algoritm de criptare asimetric (se genereaza o pereche de chei publica si privata, avand lungimea de 2048 biti). Functionalitatea acestui algoritm se bazeaza pe dificultatea de a factoriza un numar mare in produsul a doua numere prime.

Sistemul a salvat doar cheia privata generata cu OpenSSL.

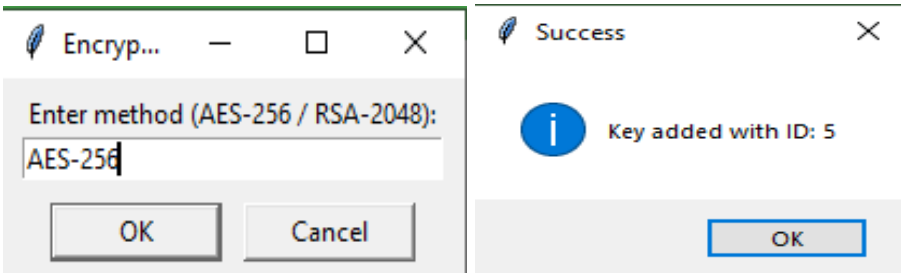
```
subprocess.run(["openssl", "genpkey", "-algorithm", "RSA", "-out",
"private_key.pem", "-pkeyopt", "rsa_keygen_bits:2048"])
```

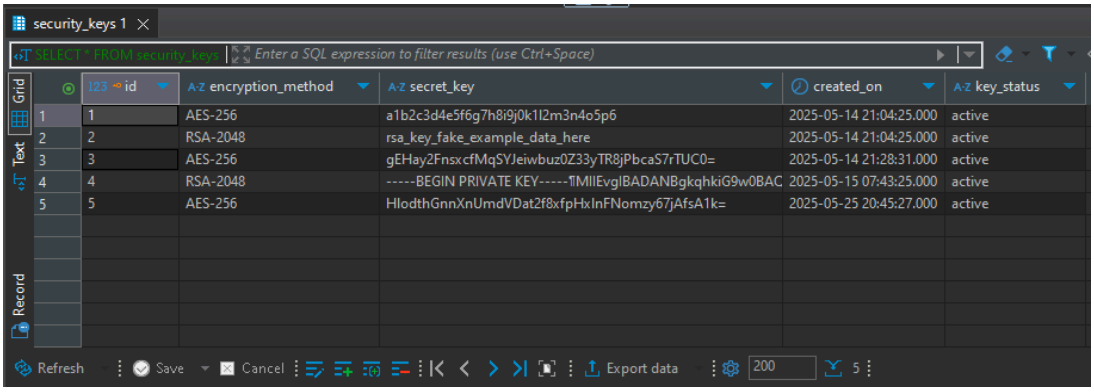
Interfata grafica:

Interfata grafica a fost construita cu ajutorul bibliotecii tkinter si ofera urmatoarele functionalitati:

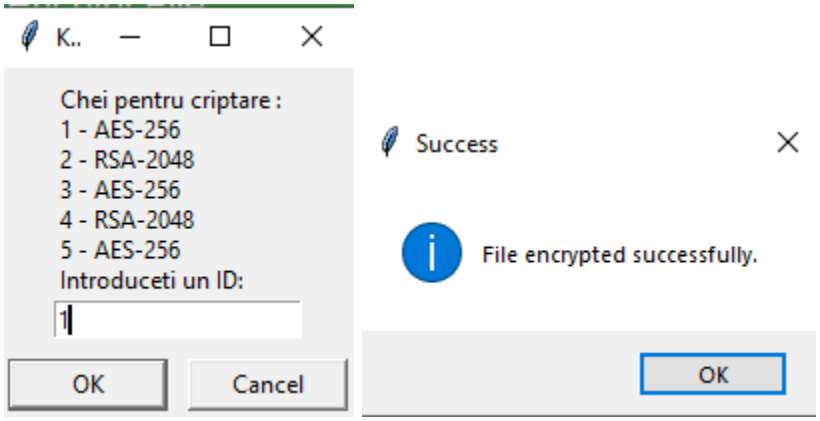
- Adaugarea cheii in baza de date prin selectarea algoritmului AES sau RSA, utilizand butonul “Add Encryption Key”;
- Selectarea unui fisier local si criptarea acestuia cu cheia selectata, folosind butonul “Encrypt File”
- Selectarea fisierului criptat, identificarea cheii asociate si decriptarea fisierului, prin apasarea butonului “Decrypt File”

Daca apasam pe butonul “Add Encryption Key”:



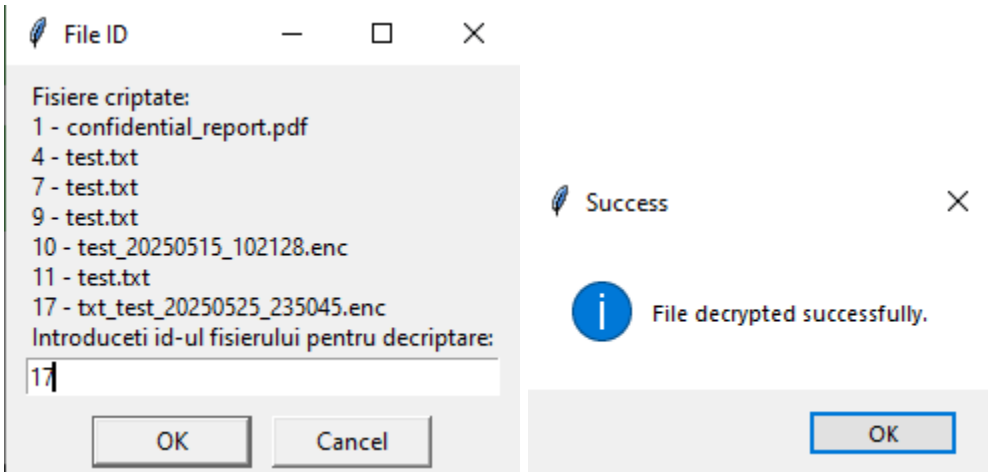


Daca apasam pe butonul “Encrypt File”:



Pentru un fisier .txt deja creat, acest buton va realiza criptarea si se va salva un fisier .enc in locatia dorita.

Daca vom dori sa decriptam:



Va fi salvat un fisier .txt decriptat cu mesajul initial.

Concluzii:

Asadar, sistemul efectueaza criptarea si decriptarea fisierelor, folosind algoritmii AES si RSA, cu o interfata grafica pentru o utilzare usoara si intuitiva si o baza de date care urmareste istoricul complet.

Bibliografie:

- https://docs.anchormydata.com/docs/what-is-aes-256-cbc
- https://www.encryptionconsulting.com/education-center/what-is-rsa/
- https://tryhackme.com/room/cryptographybasics