Projeto Crypto Files: Relatório Técnico e Apresentação

Projeto: Crypto Files – Envelope Encryption System

Disciplina: Segurança no Ciberespaço

Parte 1: Relatório Acadêmico e Fundamentação

Esta seção detalha as respostas para as questões acadêmicas propostas, expandindo os conceitos técnicos e teóricos utilizados no desenvolvimento da ferramenta.

1. Arcabouço de Conhecimentos e Domínio de Segurança

Para fundamentar academicamente o desenvolvimento do *Crypto Files*, o projeto foi enquadrado nos principais corpos de conhecimento de Segurança da Informação: **CISSP** (*Certified Information Systems Security Professional*) e **CyBok** (*Cyber Security Body of Knowledge*).

Domínio Selecionado (CISSP)

O projeto se situa primordialmente no **Domínio 3 do CISSP: Engenharia e Arquitetura de Segurança (*Security Architecture and Engineering*)**.

Justificativa Detalhada: Este domínio foca nos conceitos, princípios, estruturas e padrões usados para projetar, implementar, monitorar e proteger sistemas. O *Crypto Files* implementa diretamente estes princípios ao:

- **Proteção de Dados em Repouso:** Aplicação de criptografia para garantir que arquivos armazenados no disco sejam ilegíveis sem a chave correta[cite: 4, 5].
- **Gerenciamento de Chaves:** Implementação de geração, carga e proteção de chaves RSA, assegurando que a chave privada esteja protegida por senha e não apenas salva em texto plano[cite: 11, 15].
- **Princípios de Design Seguro:** Uso de algoritmos robustos e validados pela indústria, evitando a criação de criptografia proprietária insegura.

Área de Conhecimento (CyBok)

No contexto do **CyBok**, a área de conhecimento central é **Cryptography**. O projeto demonstra a aplicação prática de primitivas matemáticas e protocolos (como AES, RSA e SHA) necessários para garantir confidencialidade e integridade[cite: 4, 8].

---

2. Tecnologias de Segurança Utilizadas

O projeto utiliza uma arquitetura de **Envelope Encryption** (Criptografia de Envelope), combinando a eficiência de algoritmos simétricos com a segurança de algoritmos assimétricos[cite: 5].

Stack Tecnológico Implementado

1. **Criptografia Simétrica (AES-256-GCM):**
    - Utilizada para criptografar o conteúdo do arquivo com uma chave aleatória.
    - **Por que GCM?** O modo *Galois/Counter Mode* provê **autenticidade integrada** (tag GCM)[cite: 7]. Diferente de modos como CBC, o GCM gera uma tag que verifica se os dados foram alterados. Isso garante a integridade do arquivo contra corrupção ou adulteração maliciosa[cite: 80, 85].
2. **Criptografia Assimétrica (RSA-4096 OAEP):**
    - Utilizada para proteger a chave AES de cada arquivo[cite: 8].
    - **Por que OAEP?** O preenchimento *Optimal Asymmetric Encryption Padding* com SHA-256 é essencial para encapsular a chave simétrica com segurança[cite: 8, 82].
3. **Hashing e Integridade:**
    - **SHA-256:** Utilizado para gerar o *fingerprint* da chave pública nos metadados e como função de hash no esquema OAEP[cite: 8, 9].

O Papel da IA e Tecnologias Disruptivas (Visão de Futuro)

Embora o projeto utilize criptografia determinística clássica, a integração com IA e Cybersecurity poderia elevar o nível de proteção:

- **Detecção de Comportamento Anômalo (UEBA):** Implementação de *Machine Learning* para monitorar logs. Se um usuário tentar descriptografar arquivos em massa, a IA poderia bloquear o acesso, prevenindo exfiltração.
- **Auditoria de Código Generativa:** Uso de LLMs no pipeline de desenvolvimento para auditar o código `crypto_core.py` e `key_management.py` em busca de vulnerabilidades lógicas.

---

3. Arquitetura da Solução

A arquitetura do sistema segue o modelo de envelope digital, implementada através dos seguintes módulos:

- **Core (`crypto_files/crypto_core.py`):** Responsável pela criptografia simétrica e montagem do formato do arquivo[cite: 14].
- **Gerenciamento de Chaves (`crypto_files/key_management.py`):** Responsável pela geração e carga segura das chaves RSA[cite: 15].
- **Interface (`crypto_files/cli.py` e `app.py`):** Pontos de entrada via linha de comando ou interface gráfica[cite: 15, 30].

O fluxo consiste em gerar uma chave AES aleatória para o arquivo, criptografar o arquivo com essa chave, e então criptografar a chave AES com a chave pública RSA do usuário, armazenando tudo em um pacote único[cite: 5, 82].

---

4. Boas Práticas (Seguindo CISSP e CyBok)

A implementação do projeto adere estritamente às recomendações de *Secure Software Design*:

1. **Defesa em Profundidade (*Defense in Depth*):**
    - A chave privada é protegida por senha usando **PKCS#8** com *BestAvailableEncryption*[cite: 11, 83]. Isso garante que o roubo do arquivo da chave não é suficiente para comprometer o sistema.
2. **Garantia de Integridade (*Tamper-Proofing*):**

    - O uso de **AES-GCM** assegura que qualquer modificação no arquivo criptografado invalide a tag de autenticação, gerando mensagens de erro explícitas[cite: 85].
3. **Disponibilidade e Gestão de Recursos:**
    - O software implementa **streaming de criptografia**[cite: 10]. Arquivos grandes são processados em blocos (*chunks*), evitando picos de memória que poderiam causar negação de serviço local[cite: 84].
4. **Aleatoriedade Criptográfica:**
    - Utilização de um **Nonce** aleatório único (12 bytes) para cada arquivo[cite: 7, 81], essencial para a segurança do algoritmo.

Parte 2: Roteiro da Apresentação (10 Slides)

Abaixo está a estrutura sugerida para os slides da apresentação, incluindo o conteúdo visual e notas para o orador explicarem os conceitos em profundidade.

Slide 1: Capa

- **Título:** Crypto Files: Segurança Híbrida para Arquivos
- **Subtítulo:** Implementação de Envelope Encryption com AES-256-GCM e RSA-OAEP
- **Identificação:** [Seu Nome] - Disciplina Segurança no Ciberespaço
- **Contexto:** Projeto Acadêmico [cite: 5]

Slide 2: Arcabouço Teórico e Boas Práticas

(Contextualização Acadêmica)

- **Domínio CISSP:** Engenharia e Arquitetura de Segurança (Domínio 3).
- **Práticas Implementadas:**
    - **Confidencialidade:** AES-256 (Padrão de Indústria).
    - **Integridade:** GCM (Proteção contra alteração).
    - **Defesa em Profundidade:** Chaves privadas protegidas por senha[cite: 11].
- **Nota do Orador:** "Nosso projeto segue o Domínio 3 do CISSP. Garantimos não só que o dado está escondido, mas que ele é autêntico e que as chaves estão seguras."

Slide 3: Tecnologias de Segurança Utilizadas

(O "Como" e o "Futuro")

- **Stack Atual:**
    - Simétrico: AES-256-GCM[cite: 7].
    - Assimétrico: RSA-4096 OAEP[cite: 8].
    - Hash: SHA-256.
- **Futuro (IA):** Detecção de Anomalias (UEBA) e auditoria automatizada de código.
- **Nota do Orador:** "Usamos o 'estado da arte' da criptografia. A IA entraria como uma camada de monitoramento futura."

Slide 4: Arquitetura da Solução (Infográfico)

- **Conteúdo:** [Inserir diagrama visual demonstrando o fluxo: Chave Aleatória -> Cifra Arquivo -> Chave Pública -> Cifra Chave Aleatória].
- **Nota do Orador:** "O diagrama demonstra o conceito de Envelope Digital. A chave que tranca o arquivo viaja protegida dentro do pacote."

Slide 5: O Conceito "Crypto Files"

- **Problema:** Envio de arquivos sensíveis via canais inseguros.
- **Solução:** Ferramenta CLI e Web para proteção local.
- **Funcionalidades:** `init-keys`, `encrypt`, `decrypt`, `inspect`[cite: 12].
- **Nota do Orador:** "O objetivo é a soberania dos dados. O arquivo sai da máquina já ilegível."

Slide 6: O Formato de Arquivo (.cfen)

- **Estrutura Proprietária[cite: 9, 56]:**
  - **Magic Bytes:** `CFENC1`.
  - **Metadados (JSON):** Nonce, Algoritmo, Fingerprint da chave, Timestamp [cite: 68-78].
  - **Payload:** Conteúdo criptografado (Bytes).
- **Nota do Orador:** "O formato `.cfen` carrega os metadados necessários para a descriptografia, exceto a chave privada."

Slide 7: Interface Gráfica (Streamlit)

- **Visual:** Print da aplicação `app.py`[cite: 30].
- **Usabilidade:** Geração de chaves e Criptografia *Drag-and-Drop*.
- **Nota do Orador:** "Transformamos a complexidade dos comandos em uma interface visual simples usando Streamlit."

Slide 8: Detalhes de Engenharia (Código)

- **Performance:** Streaming de dados para suportar arquivos grandes sem carregar tudo na memória[cite: 10, 84].
- **Segurança de Memória:** Variáveis de ambiente (`CRYPTOFILES_PASSPHRASE`) para evitar senhas no histórico do shell[cite: 39, 40].
- **Nota do Orador:** "O sistema processa arquivos gigantes em pequenos blocos, garantindo estabilidade."

Slide 9: Diferenciais do Projeto

- **Independência:** Execução local (Python 3.9+)[cite: 18].
- **Transparência:** Metadados auditáveis via comando `inspect`[cite: 50].
- **Auto-teste:** Comando `self-test` para validar a integridade do sistema[cite: 52].
- **Nota do Orador:** "O sistema possui ferramentas de auto-auditoria integradas."

Slide 10: Conclusão

- **Resultados:** Ferramenta segura, performática e alinhada ao NIST.
- **Próximos Passos:** Assinatura Digital e suporte a múltiplos destinatários.
- **Encerramento:** Perguntas?