中国科学技术大学 University of Science and Technology of China | 网络空间安全学院 School of Cyber Science and Technology

作品类别：☑软件设计 ☐硬件制作 ☐工程实践

《密码学导论》课程大作业作品设计报告

基于 SM3 和 SM4 国密算法的保密文件库

2025 年 6 月 3 日

基本信息表
作品题目：基于 SM3 和 SM4 国密算法的保密文件库
作品类别：☑软件设计 ☐硬件制作 ☐工程实践
作品内容摘要： 摘要本项目尝试通过对 SM3 和 SM4 国密算法原理的研究，在不直接引用密码算法库的前提下，利用 C 语言和 Python 实现了基于 SM3 和 SM4 算法，具有基本功能的保密文件库。
关键词： SM3，SM4，文件保险箱，密钥管理方案，文件加密系统

目录

1 作品功能与性能说明 **4**

1.1 引言 4

1.2 研究背景与意义 4

2 设计与实现方案 **5**

2.1 实现原理 5

2.1.1 SM3 密码杂凑算法 5

2.1.2 SM4 分组密码算法 6

2.1.3 加解密流程 6

2.2 运行结果 7

2.3 技术指标 8

3 系统测试与结果 **9**

3.1 测试方案 9

3.2 功能测试 9

3.3 性能测试 11

4 应用前景 **12**

5 结论 **13**

1 作品功能与性能说明

1.1 引言

本项目设计并实现了一套基于国密算法 SM3 和 SM4 的文件保险箱系统。该系统将 SM3 杂凑算法提供的完整性验证能力与 SM4 分组密码算法的高强度加密功能相结合，通过密钥派生、数据分块加密、认证机制等多层防护，构建了一个具有一定安全性的简单的本地文件保护方案。用户可通过口令管理专属保险箱，实现文件的加密存储与授权访问，有效防御非授权窃取、篡改等安全威胁。

1.2 研究背景与意义

在数字化时代，数据安全已成为个人隐私与企业机密的核心保障。随着网络攻击手段的日益复杂和敏感数据泄露事件的频发，如何高效、可靠地保护本地存储的文件成为亟待解决的关键问题。尤其在我国大力推进国产密码算法应用的战略背景下，采用国密标准（如 SM 系列算法）构建安全系统，不仅符合国家信息安全自主可控的要求，更具备技术先进性与实践必要性。

2 设计与实现方案

2.1 实现原理

2.1.1 SM3 密码杂凑算法

SM3 是我国商用密码体系中的密码哈希算法，输出 256 位固定长度杂凑值，主要用于数字签名、数据完整性校验等场景。其核心流程包含三阶段：

1. **消息填充**:
   - 对输入消息末尾添加比特"1"
   - 补足 k 个"0" 使得总长度满足: $(l+1+k) \equiv 448 \pmod{512}$
   - 追加 64 位消息长度 (大端序表示)
2. **消息扩展**:

$$W_j = \begin{cases} M_j^{(i)} & 0 \le j \le 15 \\ P_1(W_{j-16} \oplus W_{j-9} \oplus (W_{j-3} \lll 15)) \oplus (W_{j-13} \lll 7) \oplus W_{j-6} & 16 \le j \le 67 \end{cases}$$

   其中 P_1 为置换函数: $P_1(X) = X \oplus (X \lll 15) \oplus (X \lll 23)$
3. **迭代压缩**: 初始化 8 个 32 位寄存器 $\{V^0\}$ 为固定常量:

$$\begin{aligned} A &= \texttt{7380166f} & B &= \texttt{4914b2b9} \\ C &= \texttt{172442d7} & D &= \texttt{da8a0600} \\ E &= \texttt{a96f30bc} & F &= \texttt{163138aa} \\ G &= \texttt{e38dee4d} & H &= \texttt{b0fb0e4e} \end{aligned}$$

   每轮计算 (共 64 轮):

$$\begin{aligned} SS1 &= ((A \lll 12) + E + (T_j \lll j)) \lll 7 \\ TT1 &= FF_j(A,B,C) + D + SS2 + W_j' \\ TT2 &= GG_j(E,F,G) + H + SS1 + W_j \\ \text{寄存器更新} &: (A,B,C,D,E,F,G,H) \leftarrow (TT1, A, B, C, TT2, E, F, G) \end{aligned}$$

   最终输出 $A\|B\|C\|D\|E\|F\|G\|H$。

2.1.2 SM4 分组密码算法

SM4 是我国商用对称加密算法标准，采用 128 位分组长度和 128 位密钥长度。其主要组成包括：

- **密钥扩展**: 主密钥 MK 分 4 字处理:

$$\begin{bmatrix} K_0 \\ K_1 \\ K_2 \\ K_3 \end{bmatrix} = \begin{bmatrix} MK_0 \\ MK_1 \\ MK_2 \\ MK_3 \end{bmatrix} \oplus \begin{bmatrix} FK_0 \\ FK_1 \\ FK_2 \\ FK_3 \end{bmatrix}$$

  轮密钥计算 ($i = 0$ 到 31):

$$\begin{aligned} rk_i = K_{i+4} &= K_i \oplus T'(K_{i+1} \oplus K_{i+2} \oplus K_{i+3} \oplus CK_i) \\ T'(Z) &= L'(\tau(Z)) \\ L'(B) &= B \oplus (B \lll 13) \oplus (B \lll 23) \end{aligned}$$

- **加解密流程**: 加密结构 (X_i 为 32 位状态字):

$$X_{i+4} = F(X_i, X_{i+1}, X_{i+2}, X_{i+3}, rk_i) = X_i \oplus T(X_{i+1} \oplus X_{i+2} \oplus X_{i+3} \oplus rk_i)$$

  非线性变换 T:

$$\begin{aligned} T(Z) &= L(\tau(Z)) \\ L(B) &= B \oplus (B \lll 2) \oplus (B \lll 10) \oplus (B \lll 18) \oplus (B \lll 24) \\ \tau &= (Sbox(a_0), Sbox(a_1), Sbox(a_2), Sbox(a_3)) \quad (4 \times 8\ S\quad) \end{aligned}$$

  解密仅需将轮密钥逆序使用：$rk'_i = rk_{31-i}$。

2.1.3 加解密流程

运行时，本程序首先接受用户输入的密钥，对其进行 SM3 加密后取前 16 字节作为用户密钥，并初始化保险箱，生成`vault_index.json`文件并创建`vault`目录存储加密文件。随后对于用户输入的每个文件，程序会生成一个 16 字节的随机密钥，并用主密钥根据 SM4 算法加密后保存，作为该文件的独立密钥。

需要解密时，程序会先用主密钥解密文件密钥，再根据文件密钥解密并导出。

必须注意的是，本程序出于演示更基本直观，实现难度略小等原因采用了 ECB 加密模式，实际应用中 ECB 模式很可能导致重复模式泄露、无法抵挡重放攻击等严重的安全性问题。CBC 模式的加解密暂时因时间精力不足等原因未完成，预计将在未来的版本中实现。

2.2 运行结果

经测试，程序可以成功初始化保险箱，对文件进行加解密，并列出加密后文件列表。效果如图。

图 1: 列出文件后的程序主界面

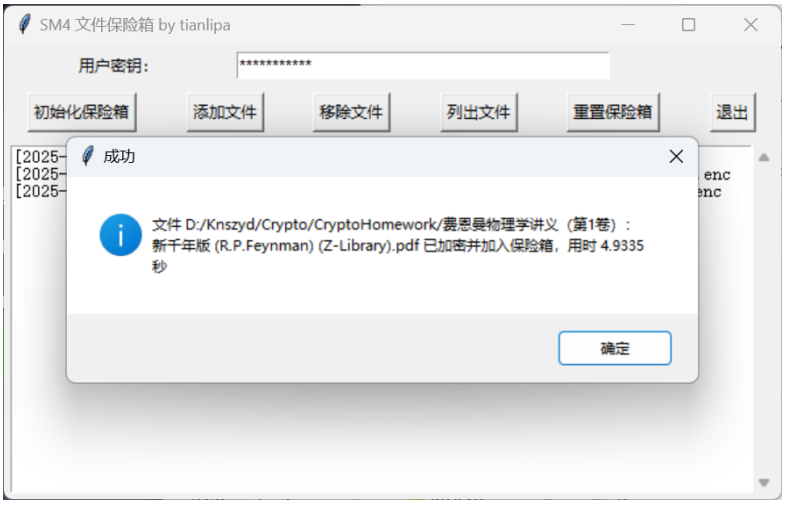

图 2: 加密示例

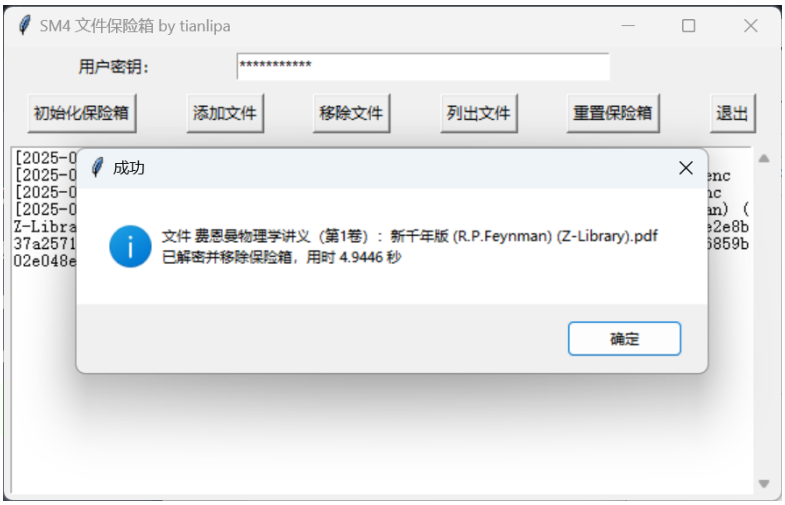

图 3: 解密示例

2.3 技术指标

经测试，本程序可以实现任意二进制文件的 ECB 模式 SM4 加密和解密，使用中可以无视文件格式统一加解密，并保留原扩展名。更详细的性能测试参见3.3节。

3 系统测试与结果

3.1 测试方案

测试采用尝试加解密占存储空间较大的文件，使用错误密钥，加密文件名冲突的文件，解密不在保险箱中的文件等操作，尝试据此衡量程序的加密效率、稳定性和鲁棒性。

3.2 功能测试

经测试，程序可以正常实现文件的加解密，且加解密结果与标准 SM4 算法实现得到的结果一致。

密钥错误时，会禁止除重置保险箱外的所有操作，如图。

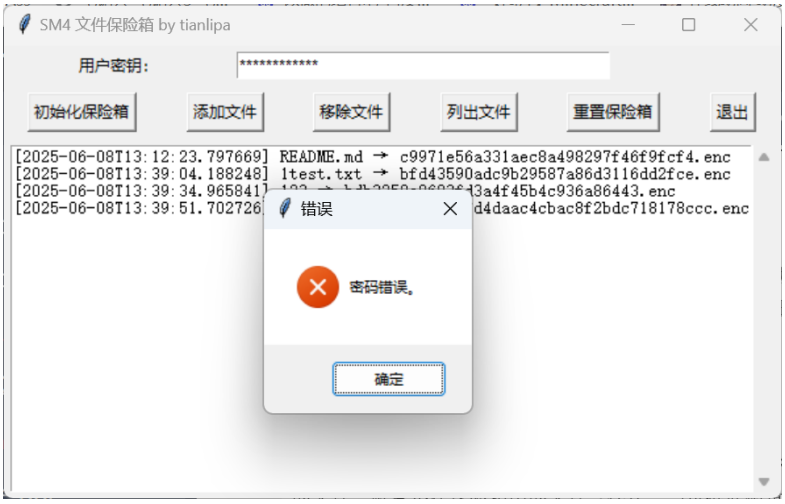

图 4: 密钥错误时报错

加密文件名冲突即保险箱中已有同名文件时，会给出提示，如图。

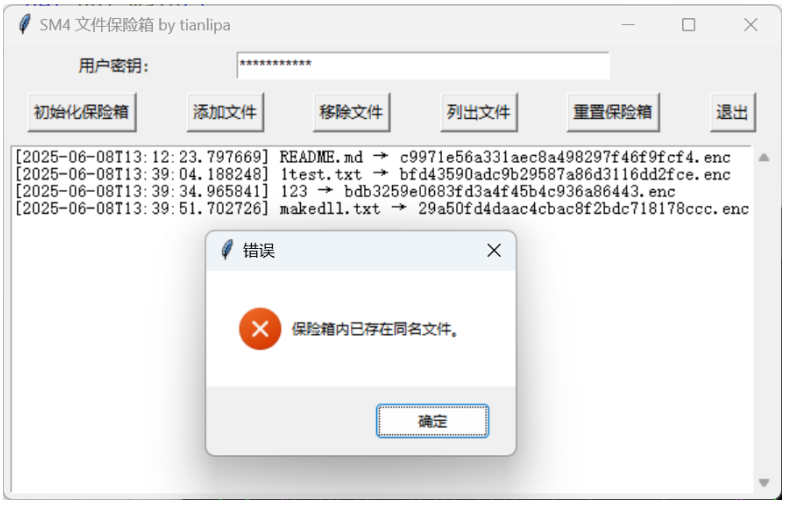

图 5: 已存在同名文件时报错

尝试解密不存在的文件时同样会报错，如图。

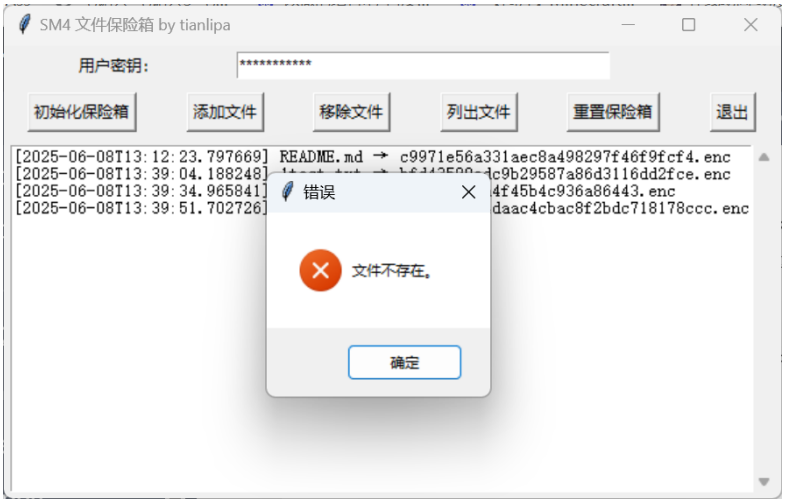

图 6: 解密文件不存在时报错

综上，程序拥有正常运行的功能性和基本的稳定性。

3.3 性能测试

此版本程序使用 C 语言实现加解密功能，再在 Python 中调用 C 语言实现的加解密函数，以此大幅提高了加解密效率。

以加密一个占存储空间 173MB 的视频为例，本程序加解密用时均在 20 秒左右，且占据内存在可接受范围内。

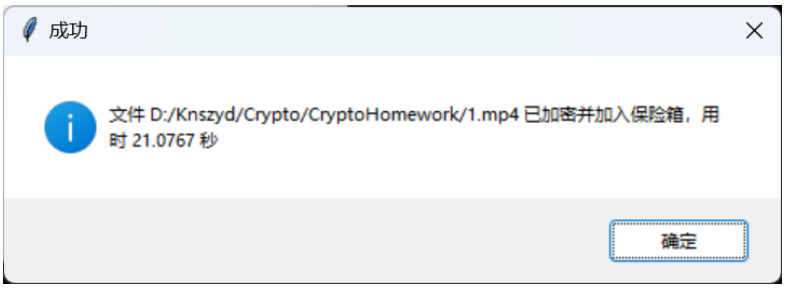

图 7: 加密 173MB 的视频文件

相对地，用 Python 实现的 SM4 加密算法效率明显低下，仅仅是加密约 2.5MB 的文件就耗时约 50 秒，这也是本程序改用 C 语言实现加解密算法的原因。

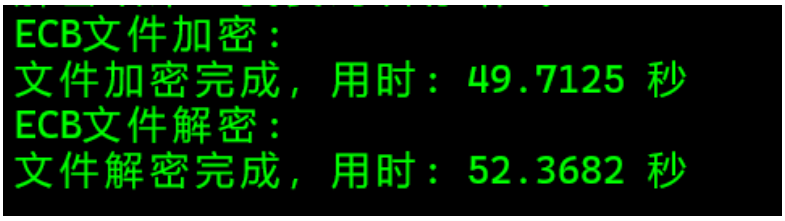

图 8: 使用 Python 实现的 SM4 加解密

需要注意的是，本程序在运行时占据的内存很可能仍有优化空间，可以通过分块读取文件等方法进一步优化，但暂时因时间精力不足等原因未完成全部优化工作。

4 应用前景

本程序基于 SM3/SM4 国密算法，既可为个人敏感信息提供本地化加密防护，也可保障企业核心文档安全，甚至可为政府机密等敏感数据筑起安全壁垒，实现从日常生活到关键领域的多层次数据保护。

5 结论

国密算法（SM 系列）的实现对国家的数字安全有着重要意义。作为我国自主可控的密码标准体系，国密算法的研发与应用突破了对国际密码体系的依赖，从根本上保障了国家网络空间的主权与安全防线。在技术层面，SM2/SM3/SM4 等算法通过设计自主的数学难题与结构模型（如基于椭圆曲线的公钥机制、非线性迭代的哈希构造），有效抵御已知攻击路径并满足商用密码的高强度要求。其更深层价值在于构建了完整的国密生态链：从芯片硬件（密码模块）、操作系统（麒麟）到应用协议（TLS 1.3+ 国密套件），实现全链路安全可控，支撑《网络安全法》等法规的落地实施。当前在金融、政务、能源等关键领域的规模化应用，不仅降低了因外来标准后门引发的系统性风险，更驱动国产密码技术成为全球数字经济治理体系中的重要一极。

本程序对国密算法的实现，诚然仍有数不胜数的不足，但同样是笔者向国家标准看齐的一小步，希望在不远的将来，笔者能向这个目标更进一步。

参考文献

Rm_mR. 国密算法 SM4 加密算法 Python 完整实现 [EB/OL]. CSDN 博客, 2023-04-03. https://blog.csdn.net/Rm_mR/article/details/129899004

养殖户. C 语言/C++ 实现国密 SM4 算法 [EB/OL]. CSDN 博客, 2022-10-27. https://blog.csdn.net/qq_61827852/article/details/127544482

Mr Wu. 深入解析 SM4 算法及其 Python 实现 [EB/OL]. 知乎专栏, 2025-01-08. https://zhuanlan.zhihu.com/p/8272374987

jiftle. 国密 SM4 算法 c 语言版源码 [EB/OL]. 博客园, 2022-12-08. https://www.cnblogs.com/jiftle/p/16965217.html

移动安全星球 _ 铭铭注定. SM4 加解密（Python）[EB/OL]. 简书, 2022-05-27. https://www.jianshu.com/p/1675082e3495