DES 与 IDEA 加密算法的实现、比较与性能分析

刘俊宏* 杨蕴涵* 许桐恺*

【摘 要】 在普惠金融加速发展的背景下，个人信贷业务快速增长，违约风险问题日益突出，亟需高效精准的风险预测模型。本文提出一套融合深度特征工程与多模型集成的端到端预测框架。通过系统预处理与多维特征构建，重点引入群体统计量与三阶共现特征刻画个体差异；建模阶段，采用 LightGBM 进行特征筛选，并基于 Optuna 优化 XGBoost、LightGBM 和 CatBoost 三个 GBDT 模型，最终以逻辑回归为元模型实现 Stacking 融合。实验证明，融合模型在验证集上 AUC 达 0.7400，天池排行榜排名第 43（截至 2025 年 6 月 11 日），显著优于单一模型。研究验证了特征构建与模型融合在信贷风控中的关键价值，具备良好应用前景。

【关键词】 **对称加密**；DES；IDEA；**性能分析**；**信息安全**

1 引 言

1.1 研究背景与意义

研究背景：我们正处在一个以数据为核心驱动力的数字化时代。从个人隐私到金融交易，从企业运营到国家安全，数字信息的安全传输与存储已成为社会正常运转的基石。然而，开放的网络环境也使得数据面临着前所未有的窃听、篡改和伪造风险。密码学，特别是其中的对称加密技术，是应对这些威胁、保障数据机密性的核心手段。对称加密算法因其加解密效率高、资源消耗低的特点，被广泛应用于数据库加密、安全通信协议（如 TLS/SSL）以及文件系统加密等大数据量处理场景。

在对称加密算法的发展长河中，数据加密标准（DES）和国际数据加密算法（IDEA）是两个具有里程碑意义的算法。DES 作为第一个被广泛采纳的国际加密标准，统治了该领域长达二十余年，对现代密码学的商业化和学术研究产生了深远影响。而 IDEA 则是在 DES 的安全性受到挑战时应运而生的杰出代表，其创新的设计理念和卓越的安全强度使其成为后续加密算法设计的重要参考。

研究意义：本研究旨在通过对 DES 和 IDEA 这两种经典算法进行深入比较，其意义主要体现在以下三个方面：

1. **理论意义：**通过剖析两种算法在设计哲学上的根本差异——DES 的 Feistel 网络结构与 IDEA 的“混合代数运算”结构——可以深刻理解对称密码的设计原则、演化路径以及安全性与效率之间

的权衡。这对于学习和理解后续更先进的算法（如 AES）具有重要的启发价值。

2. **实践意义：**尽管 DES 已不再安全，但研究其从辉煌到被淘汰的过程，尤其是其 56 位密钥长度的致命缺陷，是信息安全教育中一个经典的警示案例。本论文通过 C++ 从零开始实现这两种算法，不仅能加深对算法内部机制的理解，还能锻炼底层编程和软件性能优化的能力。

3. **学术价值：**本文将理论分析与实证测试相结合。通过对自行实现的算法和业界优化的 Python 标准库进行性能基准测试，可以量化地揭示理论效率与实际工程实现之间的差距，为加密算法在特定应用场景下的选型提供一定的参考依据。

1.2 相关工作与文献综述

密码学界对 DES 和 IDEA 的研究已相当深入。早期工作主要集中于对算法本身的密码分析。针对 DES 算法，其官方标准由美国国家标准局在 **FIPS PUB 46**[?] 中发布。在安全性分析方面，最著名的工作莫过于 **Biham 和 Shamir** 提出的差分密码分析[?] 以及 **Matsui** 提出的线性密码分析[?]。这两项工作从理论上证明了存在比暴力破解更有效的攻击手段，揭示了 DES 在设计上的一些脆弱性，并直接推动了现代分组密码分析理论的发展。而**电子前沿基金会（EFF）**[?] 在 1999 年成功实践的暴力破解，则从工程上宣告了 DES 时代的终结。

针对 IDEA 算法，其设计由 **Lai 和 Massey**[?] 在 1991 年的论文《一种新的分组加密标准提案》中首次提出。该算法在设计之初就考虑了对差分密码分析的抵抗能力，其安全性得到了广泛的论证。后续研究，如 **Daemen 等人**的工作[?]，确认了 IDEA 对差分和线性密码分析均具有很高的抵抗力。尽管存在一些针对简化轮数（如 3 轮或 4 轮）IDEA 的理论攻击，但对于完整的 8.5 轮 IDEA，至今未发现任何比暴力破解更有效的攻击方法，其 128 位的密钥长度也保证了对穷举攻击的免疫力。

在性能比较方面，已有大量文献在不同平台（如 CPU、FPGA、ASIC）上对 DES 和 IDEA 的运行效率进行了评估。多数研究表明，由于 DES 主要依赖于位操作（置换和异或），在硬件实现上具有天然的速度优势。而在纯软件实现中，IDEA 包含的模乘运算通常会成为性能瓶颈，导致其速度慢于 DES。然而，随着现代 CPU 指令集的不断优化，这种性能差距可能发生变化。本研究将在现代计算机体系结构下，通过高级编程语言的实现来重新审视这一性能对比，并与高度优化的专业库进行比较，从而为这一经典议题提供最新的实证数据。

1.3 本文主要工作与贡献

基于上述背景和研究现状，本文旨在完成以下主要工作，并做出相应贡献：

1. **系统性的理论对比：**本文将从算法结构、密钥调度、核心运算等多个角度，对 DES 和 IDEA 的步骤复杂度和设计思想进行系统性的梳理和对比。

2. **深入的安全性剖析：**综合分析两种算法在密钥长度、抗差分/线性分析能力、以及是否存在弱密钥等方面的安全性差异，并阐述这些差异背后的设计原因。

3. **算法的编程实现：**基于 C++ 语言，从零开始完整地实现 DES 和 IDEA 两种加密算法的核心逻辑，包括数据填充、密钥生成和加解密全过程，以达到对算法内部机制的精确掌握。

4. **多维度的性能评测：**设计并实施一套性能测试方案，通过对不同大小的数据文件进行加密操作计时，完成以下两组核心对比：
   - **横向对比：**比较本文实现的 DES 与 IDEA 在相同软硬件环境下的运行效率。
   - **纵向对比：**将本文 C++ 实现的算法性能与 Python 标准加密库中的同类算法进行比较，分析手写实现与专业优化库之间的性能差距。

本文的主要贡献在于，将经典的密码学理论比较与现代编程实践和性能评测相结合，不仅提供了一份关于 DES 与 IDEA 的全面对比分析报告，更通过可复现的实证数据，为理解这两个里程碑式算法在当前计算环境下的真实性能表现提供了有价值的参考。

2 算法的数学原理与安全性分析

任何密码算法的安全性都根植于其底层的数学原理。本章旨在从数学和密码学的角度，深入剖析 DES 和 IDEA 的设计精髓，并基于这些原理对其安全性进行评估。我们将重点阐述 Feistel 网络结构、非线性 S 盒、不同代数群操作混合等核心概念，并解释它们如何为算法贡献混淆（Confusion）与扩散（Diffusion）这两个关键特性。

2.1 DES 的数学原理与安全性

2.1.1 数学原理：Feistel 网络与非线性替换

DES 的安全性主要依赖于两个核心组件的协同工作：Feistel 网络结构和高非线性的轮函数 F。

1. Feistel 网络结构 Feistel 网络是一种对称结构，用于构建分组密码。其巧妙之处在于，它将加密过程分解为多轮迭代，并且每一轮的轮函数 F 无需自身可逆，整个加密过程却天然可逆。对于第 i 轮迭代，其数学表达式为：

$$L_i = R_{i-1}$$
$$R_i = L_{i-1} \oplus F(R_{i-1}, K_i)$$

其中 L_{i-1} 和 R_{i-1} 是上一轮输出的左右两半，$\oplus$ 代表按位异或，K_i 是本轮的子密钥。

其可逆性可以通过简单的代数推导证明。要从 (L_i, R_i) 恢复 (L_{i-1}, R_{i-1})，我们可以看到：

$$R_{i-1} = L_i$$
$$L_{i-1} = R_i \oplus F(R_{i-1}, K_i) = R_i \oplus F(L_i, K_i)$$

可以看到，解密过程与加密过程使用了完全相同的结构，只需将子密钥 K_i 按相反的顺序 $(K_{16}, K_{15}, \dots, K_1)$ 应用即可。这一优雅的对称性极大地简化了 DES 的硬件和软件实现，因为无需为解密设计一套独立的逻辑。

2. 轮函数 F 与香农的密码学思想 轮函数 F 是 DES 安全性的核心，它负责在每一轮中引入非线性，实现混淆与扩散。

- **混淆 (Confusion):** 这是指使密文与密钥之间的关系尽可能复杂和模糊，以挫败攻击者通过统计分析等方式推断密钥的企图。在 DES 中，S 盒（Substitution-box）是实现混淆的唯一组件。S 盒本身是一个固定的、精心设计的非线性查找表。输入 6 比特，输出 4 比特。这种多输入多输出的非线性映射使得输出比特与输入比特之间不存在简单的线性关系。如果 S 盒是线性的，那么整个 DES 算法将退化为一个巨大的线性变换，可以通过解线性方程组被轻易攻破。

- **扩散 (Diffusion):** 这是指将明文中单个比特的影响尽可能快地散布到更多的密文比特中，从而隐藏明文的统计特性。如果扩散性差，明文的统计规律（如某些字母出现频率高）可能会传递到密文中。在 DES 中，P 盒（Permutation-box）和扩展置换 E 主要负责实现扩散。P 盒将 S 盒的输出比特进行重排，使得来自不同 S 盒的输出可以在下一轮中影响到更多的 S 盒输入。扩展置含 E 则将右半部分 R_{i-1} 的某些比特复制，使得单个输入比特可以影响到两个 S 盒，从而加速了扩散过程。经过多轮迭代，明文中任何一比特的改变都会以大约 50% 的概率影响到最终密文中所有比特的变化，这被称为**雪崩效应（Avalanche Effect）**。

2.1.2 安全性分析

DES 的安全性在提出之时是足够的，但随着理论研究的深入和计算能力的飞跃，其弱点也逐渐暴露。

- **密钥长度过短：**这是 DES 最根本、最致命的缺陷。其有效密钥长度仅为 56 位，意味着总共有 2^{56}（约 7.2×10^{16}）个可能的密钥。在现代计算能力面前，通过**暴力破解（Brute-force Attack）**——即尝试所有可能的密钥——来破解 DES 已经完全可行。如前文所述，EFF 的“深译”计算机在 1999 年就已证明了这一点。

- **对差分和线性密码分析的脆弱性：**

  - **差分密码分析**[?]：由 Biham 和 Shamir 提出，它通过分析特定明文对（具有特定输入差值）经过加密后输出差值的概率分布来推断密钥。研究表明，DES 的 S 盒设计在一定程度上抵抗了这种攻击，但并非完全免疫。理论上，使用 2^{47} 组选择明文即可破解 DES，这虽仍是天文数字，但已远优于暴力破解的 2^{55}（平均尝试次数）。
  - **线性密码分析**[?]：由 Matsui 提出，它试图找到一个描述明文、密文和密钥比特之间线性关系的近似表达式（一个概率不为 1/2 的异或方程）。通过分析大量已知明文-密文对，可以验证该线性关系对哪个密钥的猜测最为成立。理论上，使用 2^{43} 组已知明文即可破解 DES。

  虽然这两种分析方法在实践中仍需大量数据，但它们从理论上打破了 DES 的“黑盒”状态，证明其并非牢不可破。

- **存在弱密钥和半弱密钥：**DES 的密钥调度算法存在缺陷，导致某些特定密钥（弱密钥）的加密效果等同于解密，即 $E_K(E_K(M)) = M$。此外，还存在一些成对的半弱密钥，使得用其中一个密钥加密的结果可以用另一个密钥解开。虽然这些特殊密钥的数量极少，在实践中随机选中它们的概率微乎其微，但这也反映了其密钥调度算法设计上的不完美。

2.2 IDEA 的数学原理与安全性

2.2.1 数学原理：混合不同代数群上的运算

IDEA 的安全性建立在一个非常创新的设计哲学之上：**混合在不同代数群上的运算**。它将三种完全不同的数学运算交织在一起，以抵抗密码分析。这三种运算分别定义在 16 位二进制向量空间上：

1. **按位异或 (XOR, $\oplus$):** 定义在群 $(\{0,1\}^{16}, \oplus)$ 上。这是一个线性操作，其数学特性在 GF(2) 域上分析得非常透彻。
2. **模 2^{16} 加法 ($\boxplus$):** 定义在群 $(\mathbb{Z}_{2^{16}}, +)$ 上。这是一个带进位的加法，相对于 XOR 操作是非线性的。
3. **模 $2^{16}+1$ 乘法 ($\odot$):** 定义在群 $(\mathbb{Z}^*_{2^{16}+1}, \times)$ 上。这是一个更为复杂的非线性操作。需要注意的是，模数 $p = 2^{16}+1$ 是一个费马素数，这使得乘法群 $\mathbb{Z}^*_p$ 具有良好的密码学特性（例如，它是一个循环群，除了 0 之外的所有元素都有乘法逆元）。为了让所有 16 位字都能参与运算，算法规定输入值为 0 的子块在计算中被视为 2^{16}。

这种设计的核心思想是，这三种运算在代数上互不“兼容”（例如，它们之间不满足分配律）。这使得任何一种单一的密码分析方法（如仅基于线性的分析或仅基于差分的分析）都难以贯穿整个算法。攻击者如果想建立一个贯穿多轮的数学模型，就必须同时处理这三种性质迥异的运算，这极大地增加了分析的难度。

IDEA 算法的基本计算单元是 **MA 结构（Multiplication-Addition）**，它将一轮中的输入 X_1, X_2 和子密钥 $Z_1, \dots, Z_4$ 结合起来，生成输出 Y_1, Y_2。这个结构通过一系列的 $\odot, \boxplus, \oplus$ 操作，实现了强大的混淆效果。多轮迭代和轮间的数据交换（类似于一个置换）则保证了充分的扩散。

2.2.2 安全性分析

IDEA 被公认为是一种非常安全的加密算法，其安全性主要体现在以下几个方面：

- **密钥长度足够长：** IDEA 使用 128 位的密钥，其可能的密钥总数达到 2^{128}。这个数量级足以抵抗任何可预见的暴力破解攻击。即使使用全球所有计算资源，在有生之年也无法穷举所有密钥。
- **内建的抗分析能力：**
  - **抗差分密码分析：** IDEA 在设计之初就以抵抗差分密码分析为主要目标。其混合运算结构，特别是模乘和模加的交替使用，能非常有效地破坏差分传播的规律性。Lai 和 Massey 在设计时就已经证明了其对差分分析的高度免疫力。
  - **抗线性密码分析：** 混合运算结构同样对线性密码分析构成了巨大障碍。由于无法找到一个贯穿多轮的高概率线性逼近，使得线性分析对完整的 IDEA 算法无效。

  迄今为止，还没有任何已知的攻击方法能够比暴力破解更有效地攻击完整 8.5 轮的 IDEA 算法。所有成功的密码分析都局限于简化版本（例如 3 轮或 4 轮）。
- **弱密钥问题：** 与 DES 类似，IDEA 也被发现存在一个小的弱密钥类别。这些弱密钥会产生一些内部计算的特定模式，可能导致分析变得容易一些。然而，这些弱密钥的数量非常少，且在实

践中易于识别和规避（例如，在密钥生成阶段进行检测）。因此，这并不构成对 IDEA 整体安全性的实质性威胁。

综上所述，DES 的安全性基石是 Feistel 结构和 S 盒的非线性，但其过短的密钥长度使其在今天变得不再安全。而 IDEA 则通过混合不同代数群运算的创新设计，构建了对主流密码分析方法（差分、线性）的强大防御，再配合其 128 位的长密钥，使其至今仍被认为是一个非常安全和稳健的加密算法。

3 实验准备

在

4 模型构建与优化

5 实验结果与分析

表 1 IDEA / DES（C++ 与 Python）统一口径性能对比

指标	IDEA (C++)	DES (C++)	DES (Python)
输入字节数 (bytes)	2,617,258	2,617,258	2,611,018
输出字节数 (bytes)	2,617,264	2,617,264	2,611,024
数据块数量 (blocks)	327,158	327,158	326,378
加密耗时 (ms)	67.727	973.362	**24.864**
解密耗时 (ms)	67.925	975.170	**20.087**
总耗时 (ms)	139.129	1,949.650	**44.974**
起始内存占用 (KB)	**3,504**	3,508	18,220
结束内存占用 (KB)	**13,180**	11,324	28,668
峰值内存占用 (KB)	**13,180**	11,324	28,156
CPU 使用率 (%)	5.8	6.2	5.6

6 结 论