上海大学
Shanghai University

《密码学》实验报告

网络空间安全专业　计算机工程与科学学院
范舒舰 23121677

DES加密解密程序与性能评测

2025 年 11 月 12 日

一、实验目的与任务

1. 实验目的

掌握DES算法的设计思路、加解密的实现过程。

2. 实验任务

1. 实现DES算法的核心组件，在此基础上实现完整的DES加解密算法;
2. 实现 3DES加解密算法;
3. 在电码本工作模式下分别利用DES和3DES对下面测试用例进行加解密(解密后需与原有明文进行对比验证，检验解密输出的正确性)，并记录两种加解密算法的运行时间，体会DES和3DES之间的效率差异。

二、实验环境

1. Windows 11
2. Python 3.11（仅用标准库 `time.perf_counter`）;
3. Spyder

三、实验内容

1. 算法原理与流程

DES 属于 Feistel 结构的 64 位分组密码：输入分为左右两半（L_0, R_0），历经 16 轮迭代。每轮以右半 R_i-1 进入轮函数 $f(\cdot, K\_i)$，与左半 L_i-1 异或得到新右半 $R\_i = L\_i-1 \oplus f(R\_i-1, K\_i)$，同时左半对调为 $L\_i = R\_i-1$。初末各有固定置换（IP/FP）。轮函数先将 R 通过扩展置换 E 扩展至 48 位，与子密钥异或后分成 8 组进入 8 个 S 盒，每盒 $6 \to 4$ 压缩，再经 P 置换输出 32 位。

3DES（EDE）通过三把独立密钥串接三次 DES 操作：加密流程为

$$C = E\_K\_1(D\_K\_2(E\_K\_3(M)))$$

解密则反序。ECB 模式按 8 字节分组独立处理，并使用 PKCS#7 补齐最后一组。

2. 关键函数节选与解释（详细版）

（1）轮函数 feistel —— DES 的核心 f 运算

```
def feistel(R_bits, subkey48):
expanded = permute(R_bits, E)           # 32 -> 48
xored    = [a ^ b for a, b in zip(expanded, subkey48)]
sboxed   = sbox_substitution(xored)      # 48 -> 32
return permute(sboxed, P)                # 32 -> 32
```

- 输入：R_bits 为 32 位列表（右半块），subkey48 为 48 位列表（当轮子密钥）。
- 输出：32 位列表（用于与上一轮的 L 异或）。

1. E 扩展： permute(R_bits, E) 将 32 位扩展到 48 位
2. 与子密钥异或： xored = expanded XOR subkey48
3. S 盒非线性压缩： sbox_substitution(xored) 将 48 位切分为 8 组，每组 6 位。对第 i 组 $[b_0..b_5]$：

$$row = (b_0 \ll 1) \,|\, b_5 \in \{0,1,2,3\}, \quad col = (b_1 \ll 3)|(b_2 \ll 2)|(b_3 \ll 1)|b_4 \in \{0..15\}$$

查 S_boxes[i][row][col] 得 0..15（4 位），8 盒拼接还原 32 位。

4. P 置换： permute(sboxed, P) 对 32 位再次重排。

（2）单块 DES：des_block_encrypt 与 des_block_decrypt

```
def des_block_encrypt(block8, subkeys):
bits = [b for byte in block8 for b in int_to_bits(byte, 8)]
bits = permute(bits, IP)
L, R = bits[:32], bits[32:]
for i in range(16):
f = feistel(R, subkeys[i])
L, R = R, [l ^ ff for l, ff in zip(L, f)]
preout = R + L                           # final swap
out = permute(preout, FP)
return bytes([bits_to_int(out[i:i+8]) for i in range(0, 64, 8)])

def des_block_decrypt(block8, subkeys):
return des_block_encrypt(block8, subkeys[::-1])
```

1. 字节→位与 IP 初始置换：把 8 字节按高位在前拆成 64 位，执行 IP，得到 (L_0, R_0)。

2. 16 轮 Feistel：对 $i = 1..16$，先算 $f(R_{i-1}, K_i)$，再执行

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

3. 末交换与 FP：拼接 $R_{16}\|L_{16}$，再经 `FP` 末置换，最后每 8 位打包回字节。

4. **解密：**Feistel 结构保证“同一轮函数 + 逆序子密钥”即可解密，因此 `des_block_decrypt` 仅反转 `subkeys` 顺序即可。

（3）3DES-ECB 封装 `triple_des_ecb_encrypt`（EDE）

```
def triple_des_ecb_encrypt(data_bytes, key_triple_hex):
padded = pkcs7_pad(data_bytes, 8)
key_bytes_list = [hex_to_bytes(k) for k in key_triple_hex]
subs = []
for kb in key_bytes_list:
kb_bits = [b for byte in kb for b in int_to_bits(byte, 8)]
subs.append(generate_subkeys(kb_bits))
out = bytearray()
for i in range(0, len(padded), 8):
b = des_block_encrypt(padded[i:i+8], subs[0])  # E_K1
b = des_block_decrypt(b,                subs[1])  # D_K2
b = des_block_encrypt(b,                subs[2])  # E_K3
out.extend(b)
return bytes(out)
```

- 顺序：EDE 即 **E_K1** → **D_K2** → **E_K3**；解密为 **D_K3** → **E_K2** → **D_K1**。
- 子密钥预计算：对 K1/K2/K3 各生成 16 轮子密钥并缓存，避免在每个分组上重复计算。

（4）填充/去填充 `pkcs7_pad` 与 `pkcs7_unpad`

```
def pkcs7_pad(data, block_size=8):
pad_len = block_size - (len(data) % block_size) or block_size
return data + bytes([pad_len]) * pad_len

def pkcs7_unpad(data, block_size=8):
if not data or len(data) % block_size != 0:
raise ValueError("Invalid padded data length")
pad_len = data[-1]
if pad_len < 1 or pad_len > block_size:
raise ValueError("Invalid PKCS#7 padding length")
if data[-pad_len:] != bytes([pad_len]) * pad_len:
raise ValueError("Invalid PKCS#7 padding bytes")
return data[:-pad_len]
```

- 填充规则：令 `pad_len = block_size - (len(data) % block_size)`；若余数为 0，则 `pad_len = block_size`，即补整块。例如块长 8、数据长 14 ⇒ `pad_len=2`，尾部补 `0x02 0x02`；若数据长 16，尾部补 `0x08`×8。

遇到的问题与解决方法

1. **明文能加密，但解密还原失败**
   原因：遗漏“末交换”或解密未逆序子密钥。
   解决：加密后拼接 `preout=R+L` 再 FP；`des_block_decrypt` 直接调用 `des_block_encrypt` 且 `subkeys[::-1]`。

2. **3DES 结果错误**
   原因：在 3DES 的 EDE 结构中，将中段误写为“加密”而非“解密”，使得整体流程实际变成了 $\mathrm{E_{K1}} \to \mathrm{E_{K2}} \to \mathrm{E_{K3}}$（或等价于非标准的序列），导致输出密文与标准 EDE 不一致，解密端也无法通过逆序还原。
   解决：严格按 $\mathbf{E}_{K1}\to\mathbf{D}_{K2}\to\mathbf{E}_{K3}$ 实现加密，解密按 $\mathbf{D}_{K3}\to\mathbf{E}_{K2}\to\mathbf{D}_{K1}$。自检建议：临时令 $K_1 = K_2 = K_3$，此时 3DES 应退化为单 DES；若输出仍不等同于单 DES，说明中段方向或密钥顺序仍有误。

3. **序列修正后仍不可还原**
   原因：解密阶段在逐块处理时提前调用了 `pkcs7_unpad`，或输入密文长度并非 8 字节对齐（如传输截断、十六进制串含空格/奇数长度导致解析异常），从而出现“解密后字节数非 8 的整数倍”或去填充校验失败。
   解决：仅在全部分组解密并拼接完成后执行一次 `pkcs7_unpad`；入口处增加对齐校验：`assert len(cipher_bytes)%8==0`；若密文以 hex 传入，先用 `bytes.fromhex()` 解析并检查长度，再进入分组解密与末尾去填充。

3. 程序结构所用函数及数据表

工具层： `hex_to_bytes`, `bytes_to_hex`, `int_to_bits`, `bits_to_int`, `permute`
密钥编排： `generate_subkeys`
轮函数相关： `sbox_substitution`, `feistel`
单块 DES： `des_block_encrypt`, `des_block_decrypt`
模式与填充： `pkcs7_pad`, `pkcs7_unpad`, `des_ecb_encrypt`, `des_ecb_decrypt`
3DES 封装： `triple_des_ecb_encrypt`, `triple_des_ecb_decrypt`
查找表： `IP`, `FP`, `E`, `P`, `PC1`, `PC2`, `S_boxes`, `SHIFTS`。

4. 结果输出验证

```
In [1]: runfile('F:/works/cyptowork/Second/DES.py', wdir='F:/works/cyptowork/Second')
Plaintext (hex)         : 54686520717569636B2062726F776E20666F78
Key (hex)               : 133457799BBCDFF1
Key (hex) (3DES)        : ['0123456789ABCDEF', '23456789ABCDEF01', '456789ABCDEF0123']
Ciphertext (hex)        : 7a53b8dc9c17e38ea6ff2538a5f462587139e9c3f48c745e
Ciphertext (hex) (3DES): 1ccf23869d09333ecce21c8112256fe6d04563f2734b1160
Decrypted (hex)         : 54686520717569636b2062726f776e20666f78
Decrypted (hex) (3DES) : 54686520717569636b2062726f776e20666f78
Decryption matches original?         -> True
Decryption matches original? (3DES) -> True
Encryption time         : 0.669 ms
Encryption time (3DES) : 1.902 ms
Decryption time         : 0.637 ms
Decryption time (3DES) : 1.911 ms
```

图 1: 运行结果

- **Plaintext (hex)**: `54686520717569636B2062726F776E20666F78`
- **Key (hex)**: `133457799BBCDFF1`
- **Key (hex) (3DES)**: `['0123456789ABCDEF','23456789ABCDEF01','456789ABCDEF0123']`
- **Ciphertext (hex)**: `7a53b8dc9c17e38ea6ff2538a5f462587139e9c3f48c745e`
- **Ciphertext (hex) (3DES)**: `1ccf23869d09333ecce21c8112256fe6d04563f2734b1160`
- **Decrypted (hex)**（DES/3DES）: `54686520717569636b2062726f776e20666f78`
- **一致性**: `True`（DES），`True`（3DES）

操作	DES（ms）	3DES（ms）
加密	0.669	1.902
解密	0.637	1.911

3DES 相对 DES 的倍数约为 $1.902/0.669 \approx 2.84$（加密）与 $1.911/0.637 \approx 3.00$（解密），与“EDE 三次 DES 核心计算”的理论预期一致。预计算子密钥显著降低了分组循环中的开销。

体会与收获

本实现按标准完成了 DES 的 IP/FP、16 轮 Feistel 与 S 盒流程，并在 3DES EDE 封装中正确使用了“加密—解密—加密”的次序；给定样例验证 `True` 表明位序、置换与填充均正确。

学会了按标准置换表实现 IP/FP/E/P，并全程统一“高位在前”的位序。

了解了 S 盒行列索引的正确取法（行取首末位、列取中间四位）与“末交换”不可省略的要点。

理解了 PKCS#7 的正确填充/去填充流程。

理解了 Feistel 结构“相同轮函数 + 逆序子密钥即可解密”的原理。

理解了 3DES 的 EDE 流程以及中段必须执行解密（D_{K2}）的原因。