**COMSAT UNIVERSITY ISLAMABAD ATTOCK CAMPUS**

****

**INFORMATION SECURITY**

**NAME:** M USMAN SIDDIQUE

**REGISTRATION NO:** SP24-BSE-024

**LAB ASSIGNEMENT:** 01

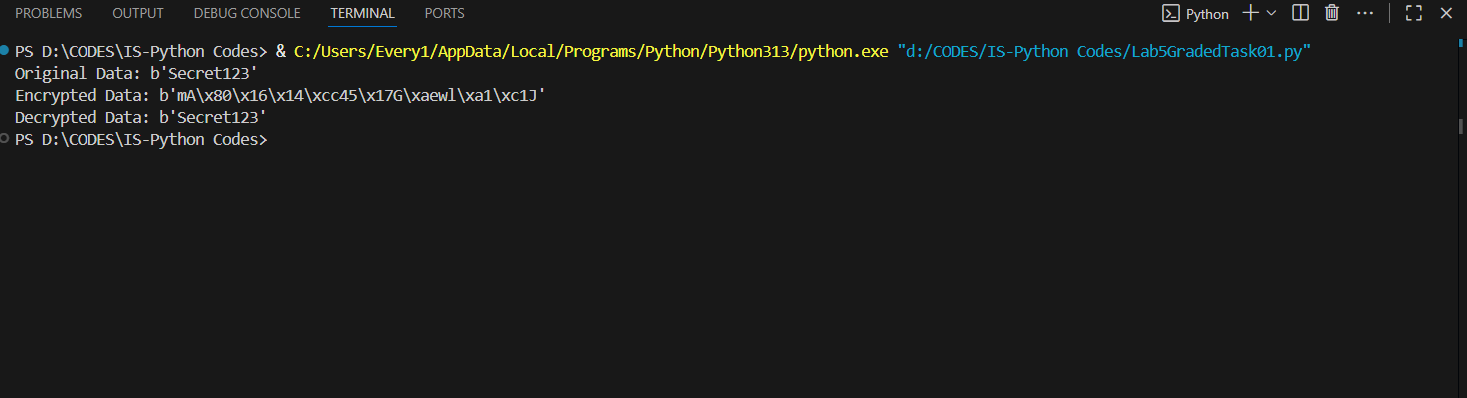
**SUBMITTED TO:** Ms. AMBAREEN GUL

**DEPARTMENT:** SOFTWARE ENGINEERING

**DATE:** 07th October 2025

**Graded lab task 1:**

Code: A screen shot of a computer program

AI-generated content may be incorrect.  
  
Output: 

**Graded Task 02:**1. Assumptions and Scope

* The attacker has access to at least one known plaintext-ciphertext pair (P, C) (known-plaintext attack).
* The demonstration uses a reduced keyspace (16-bit halves mapped to 8-byte DES keys) to make the attack run quickly in a lab environment. Real DES uses 56-bit keys — the demo is conceptual and educational.
* We operate on single 8-byte DES blocks (ECB single-block encrypt/decrypt) and use raw block operations for intermediate values (no padding during intermediate computations) to avoid decrypt-padding errors when testing wrong keys.

2. Background: Double-DES and MITM

* Double-DES applies DES twice with two independent keys:  
  [ C = E\_{K2}(E\_{K1}(P)) ]
* Meet-in-the-Middle (MITM) exploits this structure by splitting operations into a forward pass (encrypt with guessed K1) and a backward pass (decrypt with guessed K2), then looking for a match on the intermediate value. This reduces the effective brute-force complexity dramatically compared to naive exhaustive search over the full combined keyspace.

3. Attack Steps (Conceptual)

1. Intercept known plaintext P and ciphertext C.
2. For each candidate K1:
   * Compute I1 = E\_{K1}(P) and store mapping I1 -> K1 in a table.
3. For each candidate K2:
   * Compute I2 = D\_{K2}(C).
   * If I2 exists in the table, candidate (K1, K2) pairs are found.
4. Verify candidate (K1, K2) by computing E\_{K2}(E\_{K1}(P)) and checking equality with C.

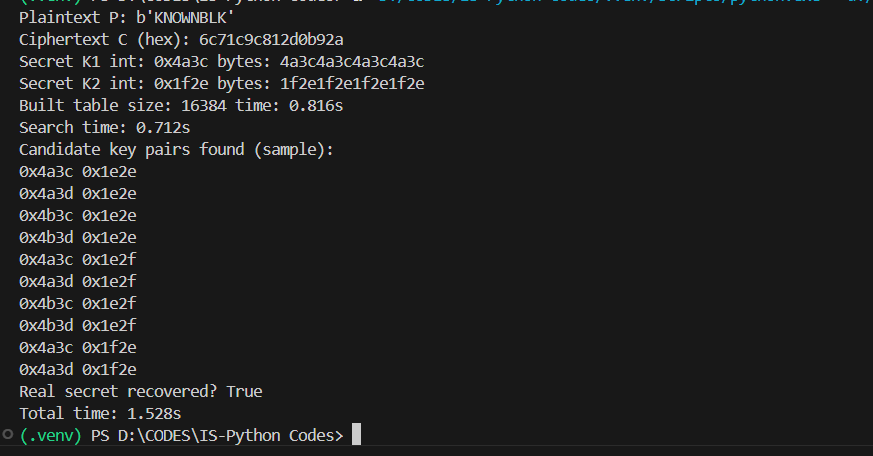
If verified, (K1, K2) is likely the correct key pair; the full key is their concatenation.

4. Complexity and Practicality

* Let n be the number of bits in each half-key. MITM requires:
  + Time ≈ (2^{n+1}) DES operations (one full pass to build the table and one full pass to search), and
  + Memory ≈ (2^{n}) intermediate values.
* For full DES (n = 56): time ≈ (2^{57}) operations and memory ≈ (2^{56}) values — feasible only with massive resources.
* MITM reduces the security gain offered by naive Double-DES (which would suggest (2^{112})), demonstrating Double-DES is not a secure way to strengthen DES.

5. Implementation (Code)

*# mitm\_double\_des\_demo.py*  
*# Educational demo of MITM attack on Double-DES using PyCryptodome.*  
*# Uses a reduced keyspace (16-bit halves mapped to 8-byte DES keys) so it runs fast for a lab.*  
  
from Crypto.Cipher import DES  
import time  
  
*# ---------- Helper functions ----------*  
  
def small\_key(i: int) -> bytes:  
 *"""*  
 *Map a 16-bit integer (0..65535) to an 8-byte DES key for demo purposes.*  
 *We repeat the 2-byte representation 4 times (2 bytes \* 4 = 8 bytes).*  
 *"""*  
 return i.to\_bytes(2, "big") \* 4  
  
  
def des\_encrypt\_block\_raw(block: bytes, key: bytes) -> bytes:  
 *"""Encrypt a single 8-byte block with DES in ECB mode (no padding)."""*  
 cipher = DES.new(key, DES.MODE\_ECB)  
 return cipher.encrypt(block)  
  
  
def des\_decrypt\_block\_raw(block: bytes, key: bytes) -> bytes:  
 *"""Decrypt a single 8-byte block with DES in ECB mode (no padding)."""*  
 cipher = DES.new(key, DES.MODE\_ECB)  
 return cipher.decrypt(block)  
  
  
def double\_des\_encrypt\_raw(plaintext: bytes, k1: bytes, k2: bytes) -> bytes:  
 *"""Compute C = E\_{K2}(E\_{K1}(P)) for a single block."""*  
 inner = des\_encrypt\_block\_raw(plaintext, k1)  
 return des\_encrypt\_block\_raw(inner, k2)  
  
  
*# ---------- Demo setup ----------*  
P = b"KNOWNBLK" *# 8 bytes exact block*  
SECRET\_K1\_int = 0x4A3C  
SECRET\_K2\_int = 0x1F2E  
K1 = small\_key(SECRET\_K1\_int)  
K2 = small\_key(SECRET\_K2\_int)  
C = double\_des\_encrypt\_raw(P, K1, K2)  
  
print("=== Demo Setup ===")  
print(f"Plaintext (P): {P}")  
print(f"Ciphertext (C) (hex): {C.hex()}")  
print(f"Secret K1 (int): {hex(SECRET\_K1\_int)} -> bytes: {K1.hex()}")  
print(f"Secret K2 (int): {hex(SECRET\_K2\_int)} -> bytes: {K2.hex()}")  
print()  
  
*# ---------- MITM attack ----------*  
MAX\_KEY = 2\*\*16 *# 65536 keys per half (demo)*  
enc\_table = {}  
  
*# Build table*  
print("Building encryption table (E\_{K1}(P) for all K1)...")  
start = time.time()  
for k1\_int in range(MAX\_KEY):  
 inter = des\_encrypt\_block\_raw(P, small\_key(k1\_int))  
 enc\_table.setdefault(inter, []).append(k1\_int)  
mid = time.time()  
print(f"Table built: {len(enc\_table)} unique intermediate values. Time: {mid - start:.3f}s")  
  
*# Search by decrypting C with all K2*  
print("Searching by decrypting C with all K2 and checking for matches...")  
found = []  
for k2\_int in range(MAX\_KEY):  
 inter2 = des\_decrypt\_block\_raw(C, small\_key(k2\_int))  
 if inter2 in enc\_table:  
 for k1\_candidate in enc\_table[inter2]:  
 *# verify*  
 if double\_des\_encrypt\_raw(P, small\_key(k1\_candidate), small\_key(k2\_int)) == C:  
 found.append((k1\_candidate, k2\_int))  
end = time.time()  
print(f"Search completed in {end - mid:.3f}s")  
  
*# Results*  
if found:  
 print("=== Candidate Key Pairs Found ===")  
 for k1\_cand, k2\_cand in found[:20]:  
 print(f" K1 = {hex(k1\_cand)} , K2 = {hex(k2\_cand)} -> K1\_bytes={small\_key(k1\_cand).hex()} K2\_bytes={small\_key(k2\_cand).hex()}")  
else:  
 print("No key pairs found (unexpected).")  
  
secret\_found = any((k1 == SECRET\_K1\_int and k2 == SECRET\_K2\_int) for (k1, k2) in found)  
print(f"\nReal secret pair found? {'YES' if secret\_found else 'NO'}")  
print(f"Total end-to-end time: {time.time() - start:.3f}s")  
  
print("\nNote: reduced keyspace used for demo. Real DES halves are 56 bits each -> infeasible here.")

6. Sample output (from running the script)  
  


7. Explanation of Results

* The MITM attack successfully found candidate (K1, K2) pairs that satisfy C = E\_{K2}(E\_{K1}(P)). The true secret pair (0x4a3c, 0x1f2e) was among the candidates and verified.
* Extra candidate pairs appear in the output because the demo uses a simplified key-mapping and small keyspace where collisions are more likely. In real scenarios, such false positives are rare; using multiple plaintext/ciphertext pairs eliminates them.

8. Reducing False Positives (Practical Tip)

* Increase the number of known (P,C) pairs and verify each candidate pair across all pairs. A false candidate will almost certainly fail for another pair.

9. Mitigations and Recommendations

* Do not use Double-DES for increased security. Use 3DES (if legacy required) or preferably AES (128/192/256-bit) for modern security.
* Use authenticated encryption (e.g., AES-GCM) to ensure confidentiality and integrity.
* Use strong key management and sufficiently long random keys.