**Comsats University Islamabad**

**(Attock Campus)**



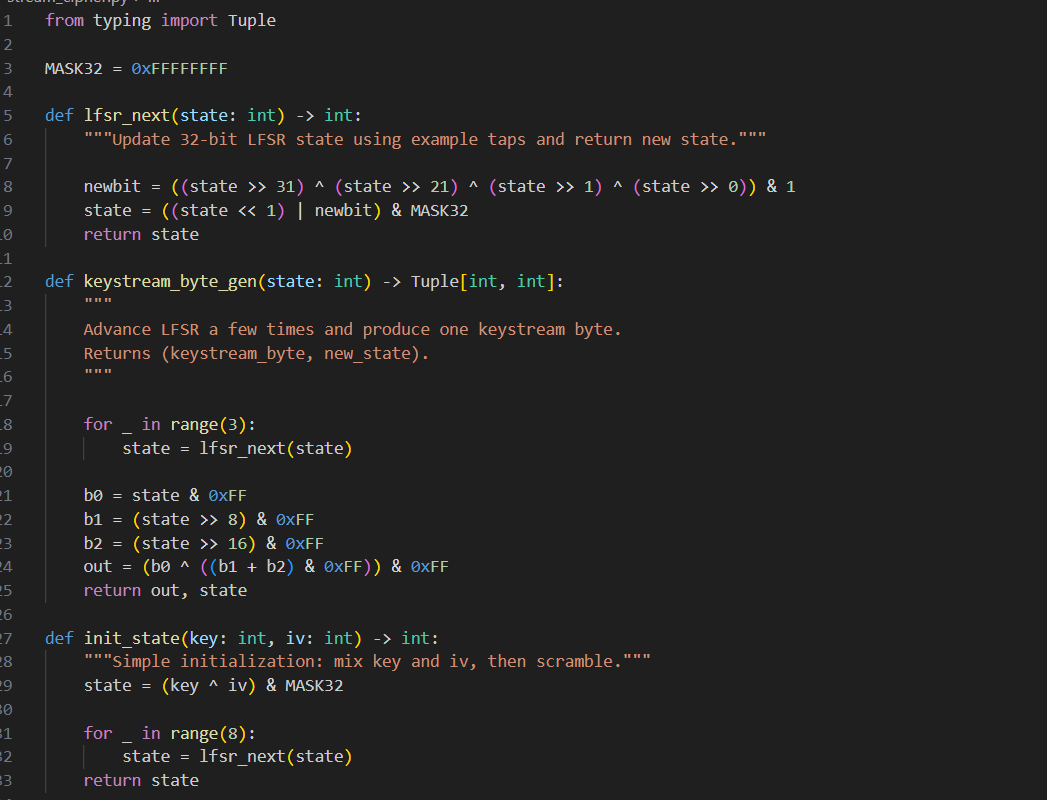
|  |  |
| --- | --- |
| NAME | WALID SHAHZEB |
| REG NO | SP24-BSE-032 |
| SUBMITTED TO | MS AMBREEN GUL |
| ASSIGNMENT | 01 |
| COURSE | INFORMATION SECURITY |
| DATE | 12-OCT-2025 |

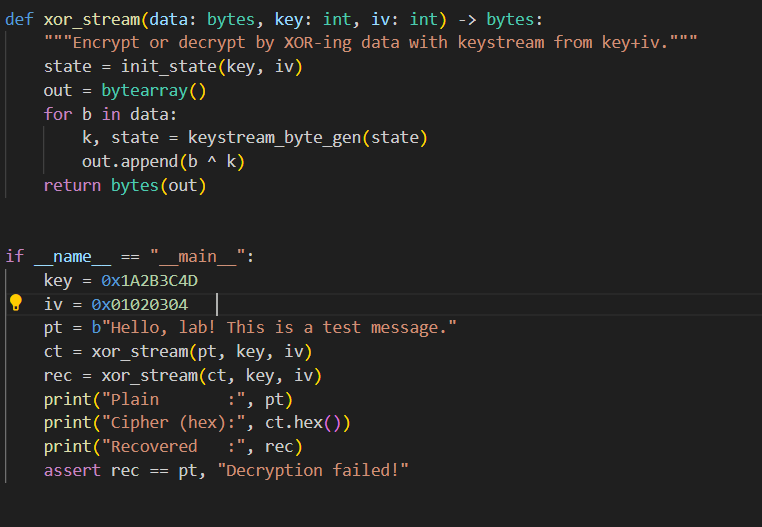
**QUESTION:**

**Write python code for your designed stream cipher approach for encryption decryption, you can use approach from more than one already developed ciphers as given in lab practice exercises.**

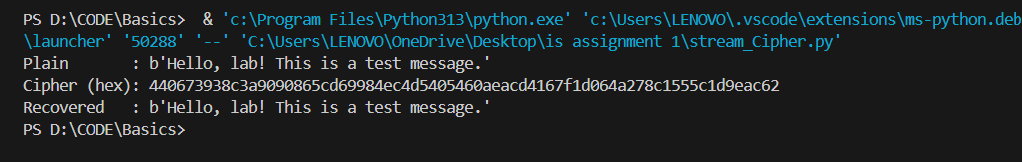
**Design and implement an adversarial attack approach for your proposed stream cipher approach.**

**STREAM CIPHER SCREENSHOTS:**

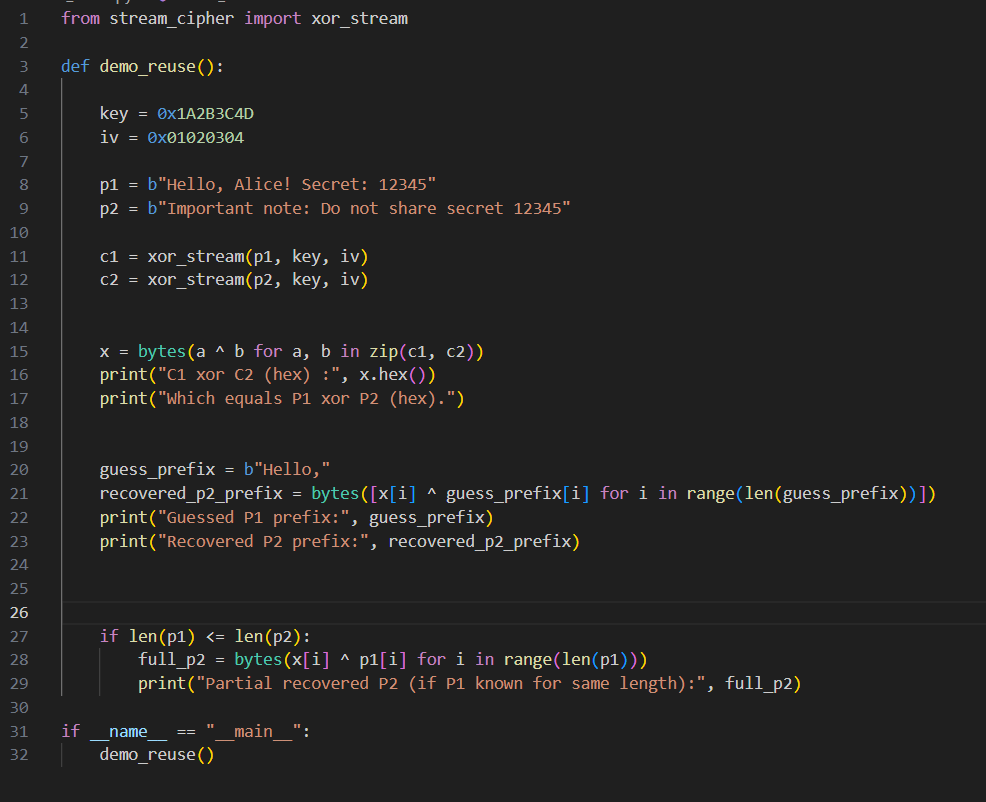
****

****

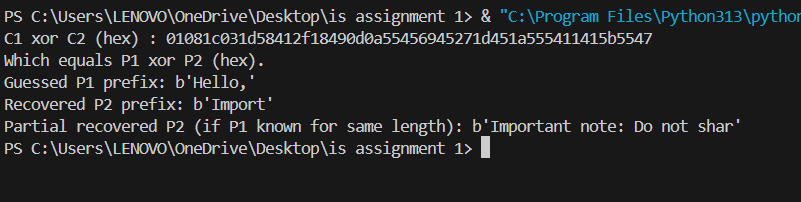
**OUTPUT:**

****

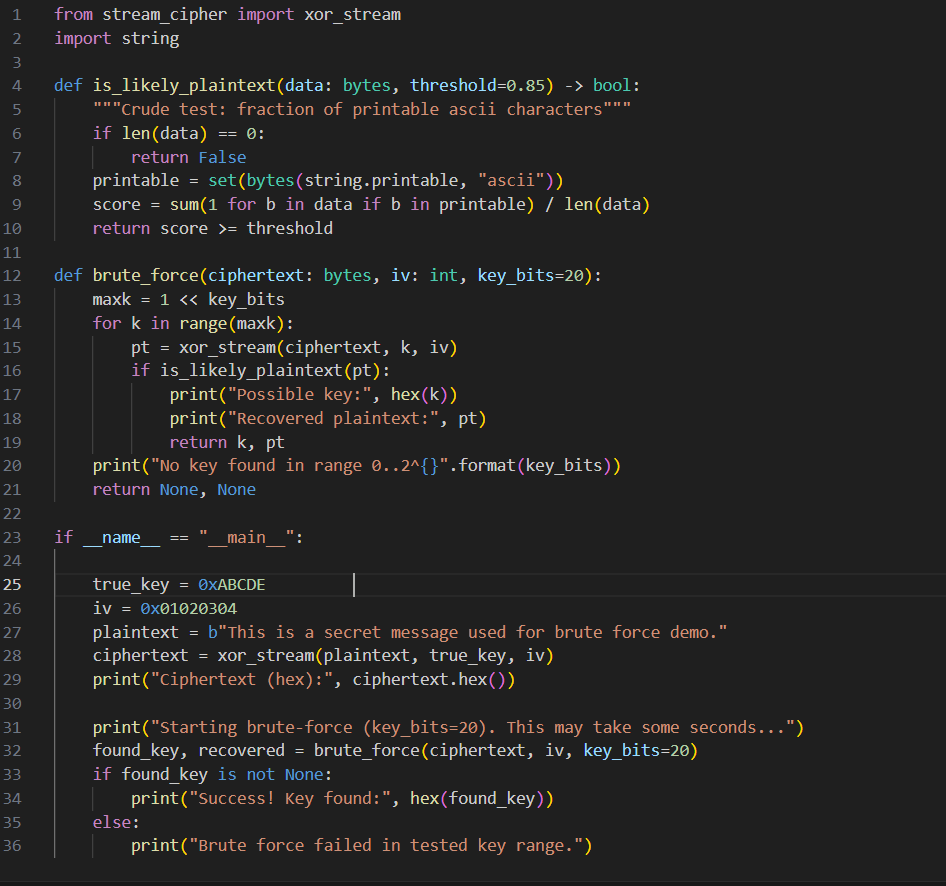
**ATTACK REUSE SCREENSHOTS:**

****

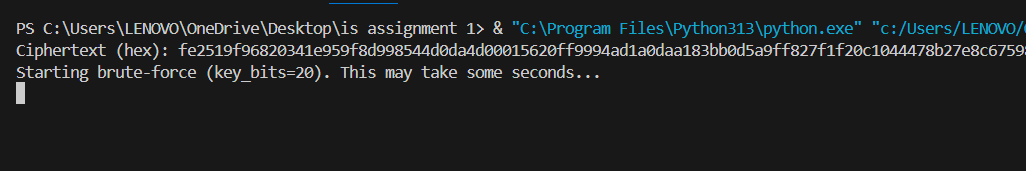
**OUTPUT:**

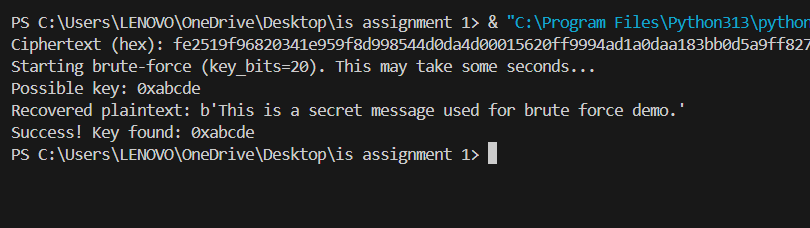
****

**ATTACK BRUTE FORCE SCREENSHOTS:**

****

**OUTPUT:**

****

****

**What this shows:**

* The script encrypted a message with a small demo key and then tried every possible key in the small keyspace.
* The brute-force script checks decrypted results with a simple printable-ASCII test and prints the candidate when the test passes.
* Success demonstrates that small keys are insecure because an attacker can try all possible keys until a readable plaintext appears.

**Why the attacks work (simple explanations)**

* **Keystream reuse:** If the same keystream is used for two messages, XOR-ing the two ciphertexts removes the keystream and results in the XOR of the two plaintexts: C1 XOR C2 = P1 XOR P2. From P1 XOR P2, if an attacker knows or can guess parts of P1, they can recover parts of P2. Reusing a nonce/IV or reusing the keystream is a critical mistake.
* **Brute-force small keyspace:** If the key is small enough, an attacker can try all possible keys (exhaustive search). For each candidate they decrypt and check if the result looks like readable plaintext — when it does, they have likely found the correct key. This is why keys must be large enough (e.g., 128 or 256 bits) in real systems.

**Security discussion**

* This toy cipher is **not secure** for real use:
  + LFSR-based state evolution is linear and subject to cryptanalysis.
  + The nonlinearity used here is weak.
  + The demonstration uses intentionally small keys for the brute-force demo.
* Real secure stream ciphers use well-studied designs with strong nonlinear mixing and large keys (examples: ChaCha20, AES-CTR with authenticated modes). For any practical system, use a vetted library rather than a home-grown cipher.

**Conclusion**

* I implemented a simple stream cipher and two attacks showing common failures: keystream reuse and small keyspace brute-force.
* The lab demonstrates why careful design, large keys, and never reusing nonces are critical for secure encryption.