Assignment 2

**Domain** – Telecom

**focus** – Optimization

**Business challenge/requirement**

LifeTel Telecom is the latest entrant in the highly competitive Telecom market of Singapore. It issues SIM to the verified users. Till now verification was manual through the photocopy of approved id card document. However, government has recently introduced Social ID called Reference ID which is mapped to fingerprint of user. LifeTel should now verify user against the fingerprint and Reference ID

**Key issues**

Build a system where when user enters Reference ID it is encrypted, so that hackers cannot view the mapping of Reference ID and finger print Considerations System should be secure

**Considerations** - System should be secure

**Data volume** - NA

**Additional information** - NA

**Business benefits**

Company will be able to quickly issue SIM to user and expected gain in volume is approximately 10 times as the manual process of verification is replaced with secure automated system

Approach to Solve

1. Read the input from command line – Reference ID
2. Check for validity – it should be 12 digits and allows on number and alphabet
3. Encrypt the Reference ID and print it for reference

from cryptography.fernet import Fernet

import re

def validate\_reference\_id(reference\_id):

if re.match("^[a-zA-Z0-9]{12}$", reference\_id):

return True

else:

return False

def encrypt\_reference\_id(reference\_id, key):

fernet = Fernet(key)

encrypted\_id = fernet.encrypt(reference\_id.encode())

return encrypted\_id

def generate\_key():

return Fernet.generate\_key()

def main():

#reference\_id = input("Enter the 12-digit Reference ID (alphanumeric): ")

reference\_id = "ABk23456CD7a"

if not validate\_reference\_id(reference\_id):

print("Invalid Reference ID. It should be 12 characters long and alphanumeric.")

return

key = generate\_key()

encrypted\_reference\_id = encrypt\_reference\_id(reference\_id, key)

print(f"Encrypted Reference ID: {encrypted\_reference\_id.decode()}")

print(f"Encryption Key (for decryption): {key.decode()}")