PoC2: E-Coupons Protocol

1 The Implementation

The implementation consists of seven scripts:

- `Issuer.py`: Performing the E-coupon issuer's operations.
- `Client.py`: Illustrating the process at the client.
- `PS.py`: The Proximity Service (PS) at the shop.
- `Ecoupon.py`: A module for customized functions dedicated to the relevant functions in our context.
- `DataGen.py`: A script for generating e-coupons data.
- `Stats.py`: For the statistics relevant to the advertisement conversion rate.

The communication is assumed to take place over a secure channel, the scripts assume that the communication over sockets is secured, the encryption methods for this are out of the scope of the present implementation, and the choice of the encryption primitives is left to the implementer. The shop is assumed to share the E-coupons database with the issuer based on their own agreement, i.e., the number and value of the e-coupons. Based on that, the shop can verify the e-coupon provided by the client by comparing it against the shop database, i.e., the membership test.

1.1 The Parties' Implementation

For the implementation, we use the following standard Python libraries.

- 'socket': The 'socket' library in Python provides access to the socket interface, allowing communication over networks using sockets.
- 'pickle': Python's 'pickle' module is used for serializing and deserializing Python objects, enabling the conversion of complex data structures into a byte stream.
- 'base64': The 'base64' module provides functions to encode and decode data in Base64 format, commonly used for encoding binary data into ASCII characters.
- 'random': The 'random' module provides functions for generating random numbers, sequences, and making choices randomly in Python.
- 'pandas': The 'pandas' library is essential for data manipulation and analysis in Python, providing data structures like DataFrames for handling structured data efficiently. Used mainly to make and sort the list of the candidate locations.
- 'time': The 'time' module in Python provides functions for working with time, including time measurement, conversions, and manipulation. Used mainly to measure the implementation running time.
- 'qrcode': The 'qrcode' module is a standard Python library for generating QR codes.

  These libraries are open-source standard Python libraries.

1.2 The Database generation

For our PoC, the (full) e-coupon consists of two parts. For demo purposes and readability, the coupon parts are generated as a four-digit integer. Clearly, longer integers are required for better security. The two parts are generated randomly at the issuer. For this end, we use.

- 'random': The random module in Python provides functions for generating random.

Please note that the security of such a step is based on the security of the random generator. Thus, this module should be chosen carefully when adapting our implementation to other environments.

The two (e-coupon) parts are combined using a hash function as follows:

```
def double_hash(int1, int2):

int1 = int(int1)
int2 = int(int2)

# Convert to bytes
int1_bytes = int1.to_bytes((int1.bit_length() + 7) // 8, byteorder='big')
int2_bytes = int2.to_bytes((int2.bit_length() + 7) // 8, byteorder='big')
l1_bytes=len(int1_bytes).to_bytes((int1.bit_length() + 7) // 8, byteorder='big')
l2_bytes=len(int2_bytes).to_bytes((int1.bit_length() + 7) // 8, byteorder='big')

# Concatenate the byte representations
combined_bytes = l1_bytes+int1_bytes + l2_bytes+int2_bytes

# Hash the combined byte string using SHA-256
hash_object = hashlib.sha256(combined_bytes)

# Return the hexadecimal digest of the hash
return hash_object.hexdigest()
```

The full coupon is a result of a 256 bits hash, for demonstration purpose, namely the readability of the produced coupon, we only take the first four digits of the e-coupon. In a real-life implementation, please consider the full 256 bits.

The next step consists of generating three separate databases:

- `IssuerData.cvs`: The issuer database containing the first part, the second part, and the full coupon. The second database is for the shop.
- `ShopData.cvs`: This only contains the hash of coupons of the full coupons. Hence, whenever a client presents an e-coupon, the shop checks if the hash of such coupon is present in this data.
- `PSData.cvs`: This consists of the first and second parts of the coupon. Such a database is at the issuer used to keep track of the clients who visited the shop, i.e., asked for verifying the first half via the PS.

The following code illustrates the case where the shop and the issuer agreed on delivering 40 e-coupons:

```
num_rows = 40 # Number of E-Coupons (agreed between shop and issuer)
data = {
    'First Half': np.random.randint(0, 2**32, size=num_rows),
    'Second Half': np.random.randint(0, 2**32, size=num_rows)
}
df = pd.DataFrame(data)

# Apply the double hash function to each row
df["E-coupon"] = df.apply(lambda row: double_hash(row['First Half'], row['Second Half']), axis=1)
```

```
def hash_hex_string(hex_string):
    # Convert the hex string to bytes
    hex_bytes = bytes.fromhex(hex_string)

    # Hash the byte representation using SHA-256
    hash_object = hashlib.sha256(hex_bytes)

    # Return the hexadecimal digest of the hash
    return hash_object.hexdigest()

df["E-coupon Hash"] = df.apply(lambda row: hash_hex_string(row["E-coupon"]), axis=1)

df1 = df["E-coupon Hash"]
C1=['First Half','Second Half',"E-coupon"]
df=df[C1]
C2=['First Half','Second Half']
df2 = df[C2]


# Create the Issuer Data Base
current_dir = os.getcwd()
file_path = os.path.join(current_dir,'IssuerData.cvs')
df.to_csv(file_path, index=False)

# Create the Shop Data Base
file_path1 = os.path.join(current_dir,'ShopData.cvs')
df1.to_csv(file_path1, index=False)

# Create the Proximity Service Data Base
file_path2 = os.path.join(current_dir,'PSData.cvs')
df2.to_csv(file_path2, index=False)
```

1.3 The Process

1. Once the client is connected to the issuer, this last will deliver the first part of the e-coupon to the client, then remove it from `IssuerData`.

2. Once the client is detected by the PS at the shop, the PS gets the client's coupon's first part.

3. The PS sends the e-coupon first part to the issuer for verification, if the first part is in the delivered first parts' database, then the issuer sends the full coupon (QR code) to the client, then the full coupon is deleted from the `PSData`.

4. The issuer can count the number of clients who got the first part and visited the shop, i.e., from the number of verified coupons, and also those who got the first part and (still) didn't visit the shop, i.e., the number of initial coupons minus the number of remaining coupons in the `IssuerData`.

Please note that to use the e-coupon at the shop, the client has to present the (full) coupon; the shop checks if the presented coupon is in its database, i.e., `ShopData`. This way, the shop will either accept or reject the coupon based on a membership test. Moreover, the issuer is not aware of the shop's sales, i.e., the used coupons.

2 Performance Analysis

We run the implementation in a workstation (MacBook Pro) with a 3.5 GHz Core (i7- 7567U). We put a timer to compute the data generation time. Please note that this algorithm is supposed to be used once before the beginning of the protocol. The average total time taken to generate the data is 0.0103 s. Moreover, we put a timer starting when the issuer process is triggered by the client to request the first half. The timer stops when the issuer sends the first half. The average total time to send the first half to Alice is 0.006 s.

Finally, we put a timer starting when the issuer process is triggered by the PS to request verifying and sending the full coupon to the client. The timer stops when the issuer sends the full coupon. The average total time taken to send the final E-coupon to the client is 0.0065 s.

Thus, the (average) total time required to receive the e-coupon is **0.012 s**. Please note that here we discarded

- The time required to generate the initial data.
- The time required to walk to the PS/shop and present the first half to the PS by the client.
- The time required by the shop to verify the e-coupon, which simply involves performing one single hash and an equality test.