PARTIAL HOMOMORPHIC ENCRYPTION FOR STATISTICAL QUERIES ON MUSIC STORED IN CLOUD SERVER

TIMUR BLAIR GORDON

TBG252

DATASET

- 3 Separate datasets:
  - My Top Songs 2019 (100 tracks)
  - Large Magic FM Playlist (608 tracks)
  - Incredibly Large Playlist (1042 tracks)
- Columns
  - Index, Track Name, Album Name, Artist Name, Track Duration (ms), Play Count

QUERIES

- Alice can query Carol with the following functions:
  - Get track by name or index (NA in homomorphic implementation)
  - Sum of track duration in playlist
  - Sum of play count in playlist
  - Average of track duration in playlist
  - Average of play count in playlist
- The queries are implemented in both a unencrypted and encrypted scheme to compare correctness and performance.

QUERIES (CODE)

```
print("Client is querying the song stored at index 6...")
print(testClient.querySongByIndex(6))

print("Client is querying the song stored at index 6...")
print(testClient.querySongByIndex(6))

print("Client is querying the song stored at index 12...")
print(testClient.querySongByIndex(12))

print("Client is querying the song named \"Sisters of Arequipa\"....")
print(testClient.querySongByName("Sisters of Arequipa"))

print("Client is querying for average length of songs in the playlist...")
print(testClient.queryStats("Track Duration (ms)", "average"))

print("Client is querying for average play count for songs in the playlist...")
print(testClient.queryStats("Play Count", "average"))

print("Client is querying for the total length of the tracks in playlist...")
print(testClient.queryStats("Track Duration (ms)", "sum"))

print("Client is querying for the total play count of the playlist...")
print(testClient.queryStats("Play Count", "sum"))
```

HOMOMORPHIC QUERIES

- In this example, the stats are calculated without decrypting values to their plain numerical form via Paillier encryption.

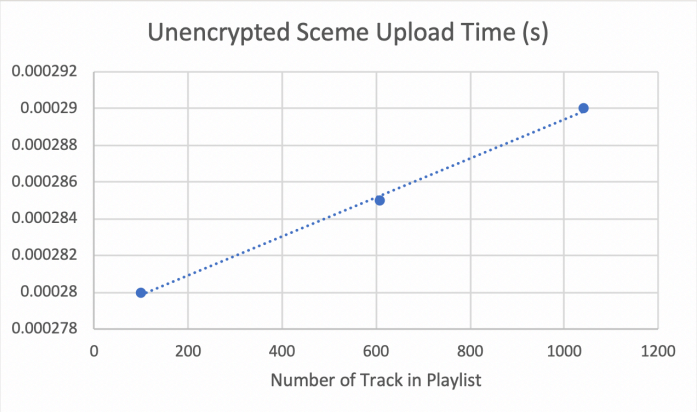
```
def statQuery(self, column, stat_type, zero_cipher):
    if stat_type == "average":
        total = zero_cipher
        for index, row in self.database.iterrows():
            total += row[column]
        return (total * self.inverse_size)
    if stat_type == "sum":
        total = zero_cipher
        for index, row in self.database.iterrows():
            total += row[column]
        return total
```

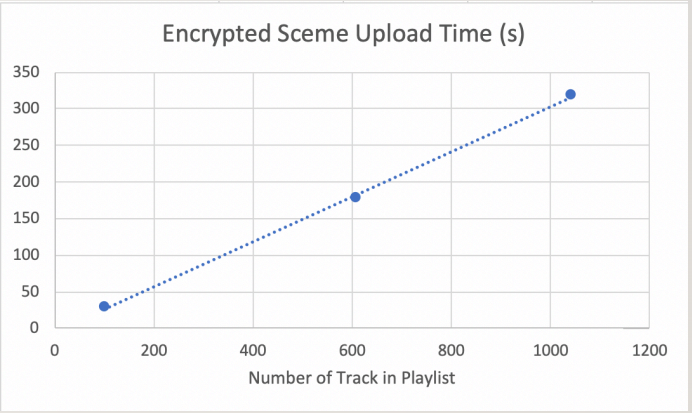
ENCRYPTION SCHEMES

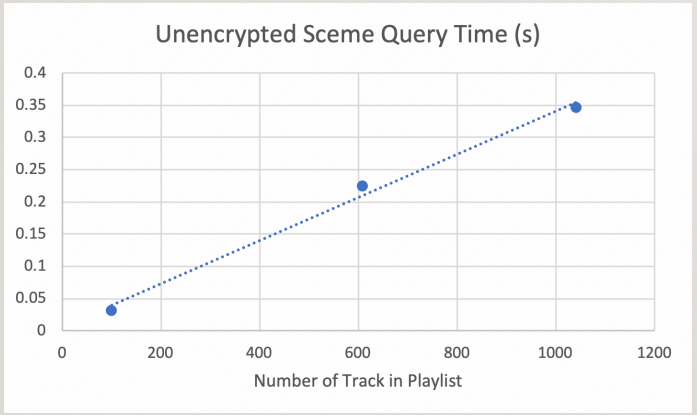
- Symmetric encryption schemes are used to encrypt data:
  - AES for alphabetical data
  - Partial Homomorphic Paillier Scheme for numerical data

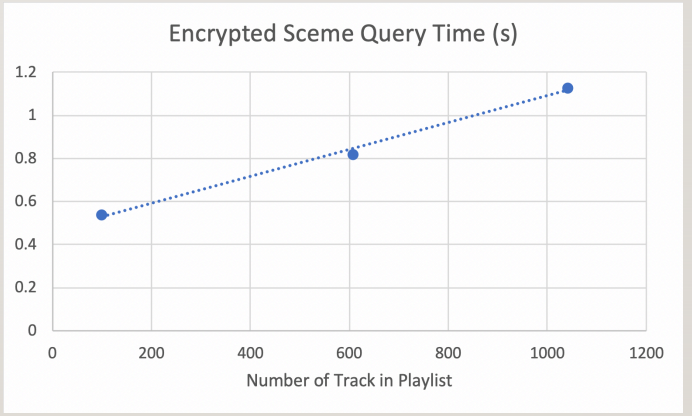
- Since the statistical queries Alice can make only use multiplication and summation:
  - Paillier is an appropriate encryption scheme for this application

PERFORMANCE (GRAPHS)









PERFORMANCE

- Encryption drastically increased upload times.
- Query times didn't show a significant difference between unencrypted and encrypted schemes
  - Once uploaded, homomorphic encryption doesn't cause additional delays
  - In fact, since queries are computed in cloud server, less computation is done by client
- Larger datasets needed to infer time complexity
  - However looks like between linear and squared polynomial time for uploading in encrypted scheme.