Author: Student 217490462

Experiment 02 – Discovery Task – 10 Targets, Multiple Parties

Objective

Assess matching and encryption methods on a target discovery task where ten targets may be present in multiple parties' data. Report back positive matches in a useful data format which enables automated follow up queries to the parties holding the data.

Method

Using python, create one or more methods to match ten targets against ten lists each containing ten suspects.

- **Target** will be a list of ten integers of a set length known as the 'ID' field. This replicates the type of ID number which would be used in an ideal financial intelligence scenario to uniquely identify a given target.
- **Multiple parties** will be various holders of 'lists' containing records including names, addresses, and an 'ID field'. The 'ID' field will be matched against the target ID, some target IDs should be present in some lists, but not all target IDs should be present in all lists.

First iteration

The target will be matched against the multiple parties lists with no encryption and the results will be returned along with the time required to process the matches. Results will be stored in a useful format which could include a numpy array, dataframe or other format.

Performance is expected to be high, and this first iteration will be the baseline against which encrypted iterations are compared.

Second iteration

The target will be matched against the multiple parties lists using the SHA-256 hash algorithm and the results will be returned along with the time required to process the matches. Results will be stored in a useful format which could include a numpy array, dataframe or other format. Hashing provides a low level of protection which is easily defeated using a rainbow table for short integer strings like those proposed in the ID fields.

Performance is expected to be better than the first iteration, based on the performance measured during SIT378 T1 Experiment 01.

Third iteration

The target will be matched against the multiple parties lists using homomorphic encryption (HE). HE should always provide high protection to the data, even during the match operation. The results will be returned along with the time required to process the matches.

Performance is expected to be significantly lower than both the first and second iterations due to the increased length of time required to complete match operations using homomorphic encryption. The total number of operations is expected to be the main predictor of performance.

Next steps

Additional iterations of the experiment can be considered depending on the results of the experiment. Permutating the number of targets, lists and suspects may be useful.

Author: Student 217490462

Experiment 02

Parameters

10 target IDs matched against 10 lists of 10 suspect IDs.

Outcome

The experiment was successful.

The team now has additional quantitative data on the performance impact of different encryption methods on a target matching task in a financial intelligence use case.

Results

First iteration – cleartext matching

Expected outcome: Performance expected to be high.

Actual outcome: Performance was high. Average seconds per operation was 0.0000116.

Second iteration – Hashed matching

Expected outcome: Performance was expected to be similar to cleartext matching.

Actual outcome: Performance was significantly better than cleartext matching. Average seconds per operation was 0.0000058.

Third iteration – HE matching

Expected outcome: Performance expected to be significantly lower than cleartext or hashed matching. A linear decrease in performance as number of operations increases.

Actual outcome: Performance was orders of magnitude slower than both the first and second

iterations. Average seconds per operation was 0.0502898. More than 8,682 times slower. However, still faster than the homomorphic encryption in SIT378 – T1 2024, experiment 01 which was about 20,000 times slower than cleartext and hashing methods.

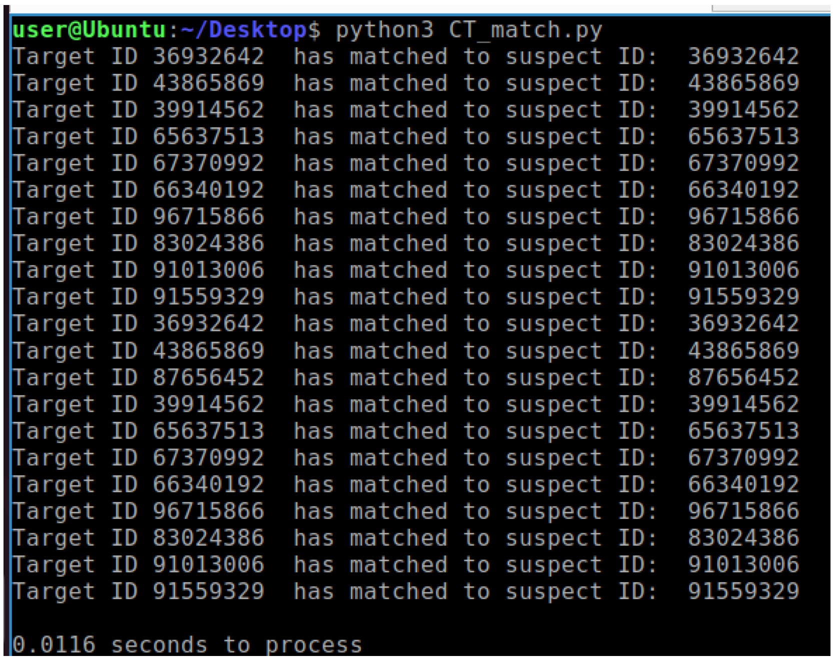
Evaluation of results

Cleartext and hashed matching are much more performant than homomorphically encrypted (HE) matching. Increasing the number of operations is trivial and both methods could be used for millions of match operations without specific investments in time or resources.

HE matching is expensive and attempting to scale up to millions of match operations would require investment in time (waiting for results) or resources (running the operations on better hardware).

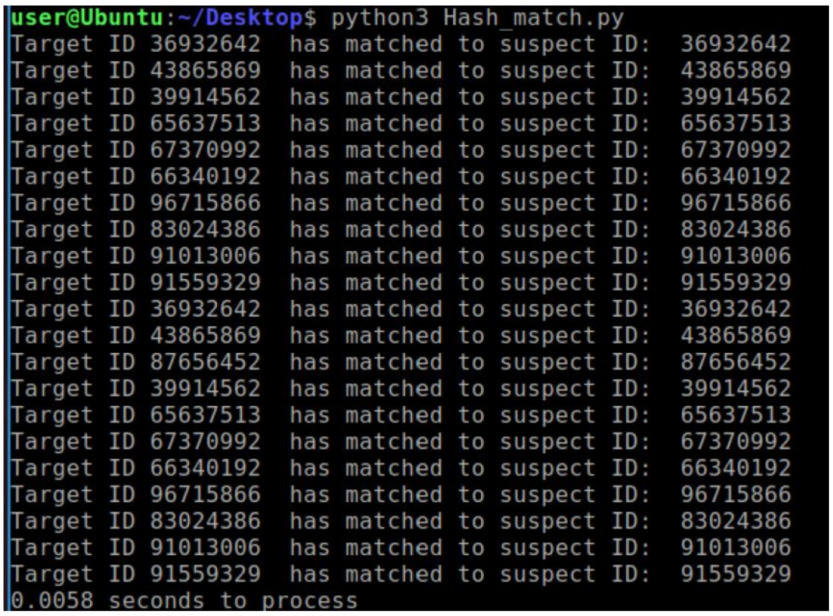
Some financial intelligence use cases would benefit from HE – where the number of match operations is low, and results can be delivered periodically rather than immediately. For example, suspect sharing among parties for deconfliction purposes could run every 24 hours or less with target and suspect lists in the low thousands, without requiring special hardware investments.

Attachment A: experiment 02 results from python in a Linux environment.



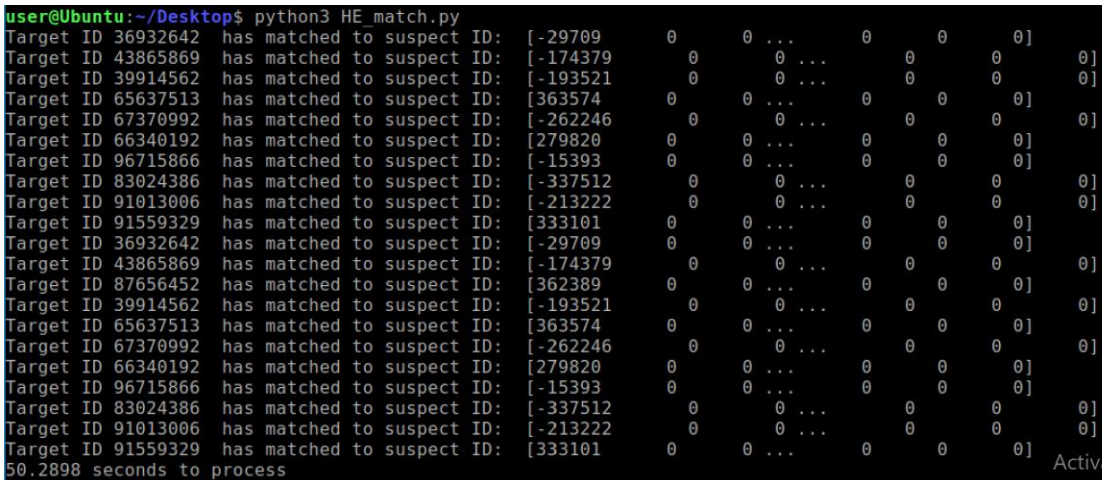
```
user@Ubuntu:~/Desktop$ python3 CT_match.py
Target ID 36932642  has matched to suspect ID:  36932642
Target ID 43865869  has matched to suspect ID:  43865869
Target ID 39914562  has matched to suspect ID:  39914562
Target ID 65637513  has matched to suspect ID:  65637513
Target ID 67370992  has matched to suspect ID:  67370992
Target ID 66340192  has matched to suspect ID:  66340192
Target ID 96715866  has matched to suspect ID:  96715866
Target ID 83024386  has matched to suspect ID:  83024386
Target ID 91013006  has matched to suspect ID:  91013006
Target ID 91559329  has matched to suspect ID:  91559329
Target ID 36932642  has matched to suspect ID:  36932642
Target ID 43865869  has matched to suspect ID:  43865869
Target ID 87656452  has matched to suspect ID:  87656452
Target ID 39914562  has matched to suspect ID:  39914562
Target ID 65637513  has matched to suspect ID:  65637513
Target ID 67370992  has matched to suspect ID:  67370992
Target ID 66340192  has matched to suspect ID:  66340192
Target ID 96715866  has matched to suspect ID:  96715866
Target ID 83024386  has matched to suspect ID:  83024386
Target ID 91013006  has matched to suspect ID:  91013006
Target ID 91559329  has matched to suspect ID:  91559329

0.0116 seconds to process
```



```
user@Ubuntu:~/Desktop$ python3 Hash_match.py
Target ID 36932642  has matched to suspect ID:  36932642
Target ID 43865869  has matched to suspect ID:  43865869
Target ID 39914562  has matched to suspect ID:  39914562
Target ID 65637513  has matched to suspect ID:  65637513
Target ID 67370992  has matched to suspect ID:  67370992
Target ID 66340192  has matched to suspect ID:  66340192
Target ID 96715866  has matched to suspect ID:  96715866
Target ID 83024386  has matched to suspect ID:  83024386
Target ID 91013006  has matched to suspect ID:  91013006
Target ID 91559329  has matched to suspect ID:  91559329
Target ID 36932642  has matched to suspect ID:  36932642
Target ID 43865869  has matched to suspect ID:  43865869
Target ID 87656452  has matched to suspect ID:  87656452
Target ID 39914562  has matched to suspect ID:  39914562
Target ID 65637513  has matched to suspect ID:  65637513
Target ID 67370992  has matched to suspect ID:  67370992
Target ID 66340192  has matched to suspect ID:  66340192
Target ID 96715866  has matched to suspect ID:  96715866
Target ID 83024386  has matched to suspect ID:  83024386
Target ID 91013006  has matched to suspect ID:  91013006
Target ID 91559329  has matched to suspect ID:  91559329
0.0058 seconds to process
```

Author: Student 217490462



```
user@Ubuntu:~/Desktop$ python3 HE_match.py
Target ID 36932642  has matched to suspect ID:  [-29709      0      0 ...      0      0      0]
Target ID 43865869  has matched to suspect ID:  [-174379      0       0 ...       0       0       0]
Target ID 39914562  has matched to suspect ID:  [-193521      0       0 ...       0       0       0]
Target ID 65637513  has matched to suspect ID:  [363574      0      0 ...      0      0      0]
Target ID 67370992  has matched to suspect ID:  [-262246      0       0 ...       0       0       0]
Target ID 66340192  has matched to suspect ID:  [279820      0      0 ...      0      0      0]
Target ID 96715866  has matched to suspect ID:  [-15393      0      0 ...      0      0      0]
Target ID 83024386  has matched to suspect ID:  [-337512      0       0 ...       0       0       0]
Target ID 91013006  has matched to suspect ID:  [-213222      0       0 ...       0       0       0]
Target ID 91559329  has matched to suspect ID:  [333101      0      0 ...      0      0      0]
Target ID 36932642  has matched to suspect ID:  [-29709      0      0 ...      0      0      0]
Target ID 43865869  has matched to suspect ID:  [-174379      0       0 ...       0       0       0]
Target ID 87656452  has matched to suspect ID:  [362389      0      0 ...      0      0      0]
Target ID 39914562  has matched to suspect ID:  [-193521      0       0 ...       0       0       0]
Target ID 65637513  has matched to suspect ID:  [363574      0      0 ...      0      0      0]
Target ID 67370992  has matched to suspect ID:  [-262246      0       0 ...       0       0       0]
Target ID 66340192  has matched to suspect ID:  [279820      0      0 ...      0      0      0]
Target ID 96715866  has matched to suspect ID:  [-15393      0      0 ...      0      0      0]
Target ID 83024386  has matched to suspect ID:  [-337512      0       0 ...       0       0       0]
Target ID 91013006  has matched to suspect ID:  [-213222      0       0 ...       0       0       0]
Target ID 91559329  has matched to suspect ID:  [333101      0      0 ...      0      0      0]
50.2898 seconds to process
```

Attachment B: python code to run a match protocol using cleartext

import time

import numpy as np

import pandas as pd

df0 = pd.read_csv("suspects10-target.csv")

df1 = pd.read_csv("suspects10-ml01.csv")

df2 = pd.read_csv("suspects10-ml02.csv")

df3 = pd.read_csv("suspects10-ml03.csv")

df4 = pd.read_csv("suspects10-ml04.csv")

df5 = pd.read_csv("suspects10-ml05.csv")

df6 = pd.read_csv("suspects10-ml06.csv")

df7 = pd.read_csv("suspects10-ml07.csv")

df8 = pd.read_csv("suspects10-ml08.csv")

df9 = pd.read_csv("suspects10-ml09.csv")

df10 = pd.read_csv("suspects10-ml10.csv")

targets = df0['ID'].to_list()

suspects = [[]]*10

suspects[0] = df1['ID'].to_list()

suspects[1] = df2['ID'].to_list()

```
suspects[2] = df3['ID'].to_list()

suspects[3] = df4['ID'].to_list()

suspects[4] = df5['ID'].to_list()

suspects[5] = df6['ID'].to_list()

suspects[6] = df7['ID'].to_list()

suspects[7] = df8['ID'].to_list()

suspects[8] = df9['ID'].to_list()

suspects[9] = df10['ID'].to_list()



def match(tar, sus):

int1 = np.array([tar], dtype=np.int64)

int2 = np.array([sus], dtype=np.int64)


resSub = int1 - int2


if (resSub) == [0]:

print("Target ID", tar ," has matched to suspect ID: ", sus)


def listmatch(tar, sus):

for l in range(0, (len(sus))):

for t in range(0, (len(tar))):

for s in range(0, (len(sus[l]))):

#print(tar[t],(sus[l][s]))

match(tar[t],(sus[l][s]))


tic = time.perf_counter()


listmatch(targets,suspects)


toc = time.perf_counter()
```

Author: Student 217490462

```
print(f"\n{toc - tic:0.4f} seconds to process")
```

Attachment C: python code to run a match protocol using hashing

```
import time

import numpy as np

import pandas as pd

import hashlib


df0 = pd.read_csv("suspects10-target.csv")


df1 = pd.read_csv("suspects10-ml01.csv")

df2 = pd.read_csv("suspects10-ml02.csv")

df3 = pd.read_csv("suspects10-ml03.csv")

df4 = pd.read_csv("suspects10-ml04.csv")

df5 = pd.read_csv("suspects10-ml05.csv")

df6 = pd.read_csv("suspects10-ml06.csv")

df7 = pd.read_csv("suspects10-ml07.csv")

df8 = pd.read_csv("suspects10-ml08.csv")

df9 = pd.read_csv("suspects10-ml09.csv")

df10 = pd.read_csv("suspects10-ml10.csv")


targets = df0['ID'].to_list()
```

Author: Student 217490462

```
suspects = [[]]*10

suspects[0] = df1['ID'].to_list()

suspects[1] = df2['ID'].to_list()

suspects[2] = df3['ID'].to_list()

suspects[3] = df4['ID'].to_list()

suspects[4] = df5['ID'].to_list()

suspects[5] = df6['ID'].to_list()

suspects[6] = df7['ID'].to_list()

suspects[7] = df8['ID'].to_list()

suspects[8] = df9['ID'].to_list()

suspects[9] = df10['ID'].to_list()


def match(tar, sus):

tar = str(tar)

sus = str(sus)

ec1 = tar.encode()

ec2 = sus.encode()


hash1 = hashlib.sha256(ec1).hexdigest()

hash2 = hashlib.sha256(ec2).hexdigest()


if hash1 == hash2:

print("Target ID", tar ," has matched to suspect ID: ", sus)


def listmatch(tar, sus):

for l in range(0, (len(sus))):

for t in range(0, (len(tar))):

for s in range(0, (len(sus[l]))):

#print(tar[t],(sus[l][s]))

match(tar[t],(sus[l][s]))
```

```
tic = time.perf_counter()


listmatch(targets,suspects)


toc = time.perf_counter()
print(f"{toc - tic:0.4f} seconds to process")
```

Attachment D: python code to run a match protocol using homomorphic encryption

```
import time
import numpy as np
import pandas as pd
from Pyfhel import Pyfhel

HE = Pyfhel()
HE.contextGen(scheme='bfv', n=2**14, t_bits=20)
HE.keyGen()

df0 = pd.read_csv("suspects10-target.csv")

df1 = pd.read_csv("suspects10-ml01.csv")
df2 = pd.read_csv("suspects10-ml02.csv")
df3 = pd.read_csv("suspects10-ml03.csv")
df4 = pd.read_csv("suspects10-ml04.csv")
df5 = pd.read_csv("suspects10-ml05.csv")
df6 = pd.read_csv("suspects10-ml06.csv")
df7 = pd.read_csv("suspects10-ml07.csv")
```

```
df8 = pd.read_csv("suspects10-ml08.csv")

df9 = pd.read_csv("suspects10-ml09.csv")

df10 = pd.read_csv("suspects10-ml10.csv")


targets = df0['ID'].to_list()


suspects = [[]]*10

suspects[0] = df1['ID'].to_list()

suspects[1] = df2['ID'].to_list()

suspects[2] = df3['ID'].to_list()

suspects[3] = df4['ID'].to_list()

suspects[4] = df5['ID'].to_list()

suspects[5] = df6['ID'].to_list()

suspects[6] = df7['ID'].to_list()

suspects[7] = df8['ID'].to_list()

suspects[8] = df9['ID'].to_list()

suspects[9] = df10['ID'].to_list()




def match(tar, sus):

int1 = np.array([tar], dtype=np.int64)

int2 = np.array([sus], dtype=np.int64)


ctxt1 = HE.encryptInt(int1)

ctxt2 = HE.encryptInt(int2)


ctxtSub = ctxt1 - ctxt2

resSub = HE.decryptInt(ctxtSub)


if (resSub[0]) == 0:
```

Author: Student 217490462

```
print("Target ID", tar ," has matched to suspect ID: ", HE.decryptInt(ctxt1))


def listmatch(tar, sus):
for l in range(0, (len(sus))):
for t in range(0, (len(tar))):
for s in range(0, (len(sus[l]))):
#print(tar[t],(sus[l][s]))
match(tar[t],(sus[l][s]))




tic = time.perf_counter()


listmatch(targets,suspects)


toc = time.perf_counter()
print(f"{toc - tic:0.4f} seconds to process")
```