# BACHELOR THESIS PROJECT - 2 Presentation


Vineet Amol Pippal
20CS30058

April 10th, 2024

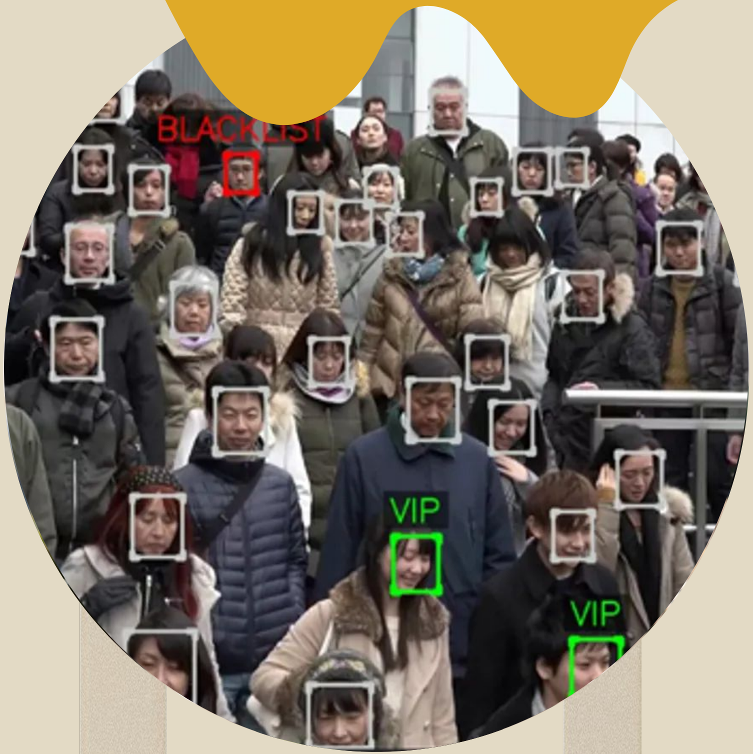
Dept. of CSE, IIT KGP

# 01

# OVERVIEW

Recent advances in DL-based Biometric Identification have made real-time identification possible by surveillance equipment and this trend is beneficial for public safety and customer convenience. However, vendors store and process plaintext data on server and people cannot opt-out of these systems which may open doors to illegitimacy and human right abuse.

**How can "persons of interest" be identified without compromising everyone else?**



# Secure Multi-party Computation

**Working together while keeping our data confidential**

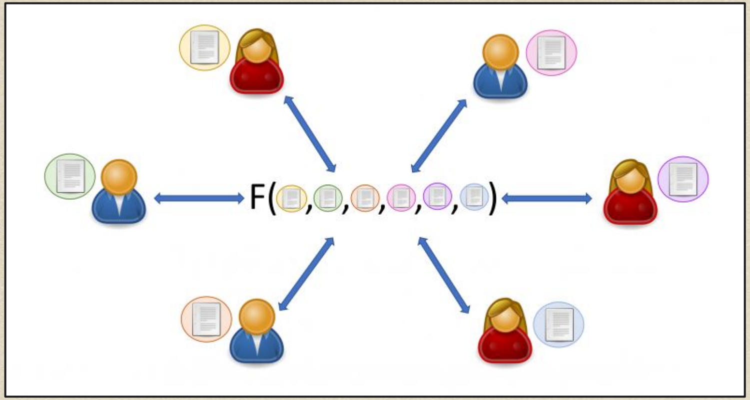
AIIMS
All India Institute Of Medical Sciences

Apollo HOSPITALS
TOUCHING LIVES

ManipalHospitals
LIFE'S ON

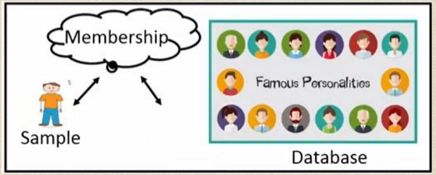
### Privacy Preserving Data Mining

Hospitals want to compute statistics without revealing their data



### Secure Online Dating

Are Alice and Bob mutually interested in each other?




### Database Membership

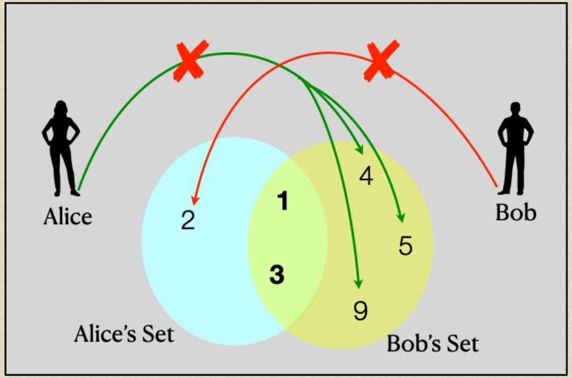
Does the sample belong to the DB?

A cryptographic protocol with the goal of creating methods for parties to jointly compute a function over their inputs while keeping those inputs private.

# Private Set Intersection

**PSI** is a cryptographic protocol that facilitates the secure and confidential determination of common elements between two or more sets without disclosing the individual elements.

Abstraction that tackles many applications like **Online advertising** , **private contact discovery** and **botnet detection** and has been studied in the **two-party**, the **multi-party**, and the **server-aided** setting with both **passive** and **active** security.




Some applications require modifications to PSI. Scenarios like **Biometric Search** require privately computing the size of the set intersection rather than the intersection itself (Private Intersection Cardinality Testing or **PICT**).

Others like **Online Dating** require finding out whether the set intersection size is above a certain threshold (Threshold PSI or **TPSI**).

# Private querying of a real-life biometric scan against a private biometric database.

**Client**

**Server**

2. Sends query to cloud

Cloud Service

3. Similar Face Search

1. Detects a face

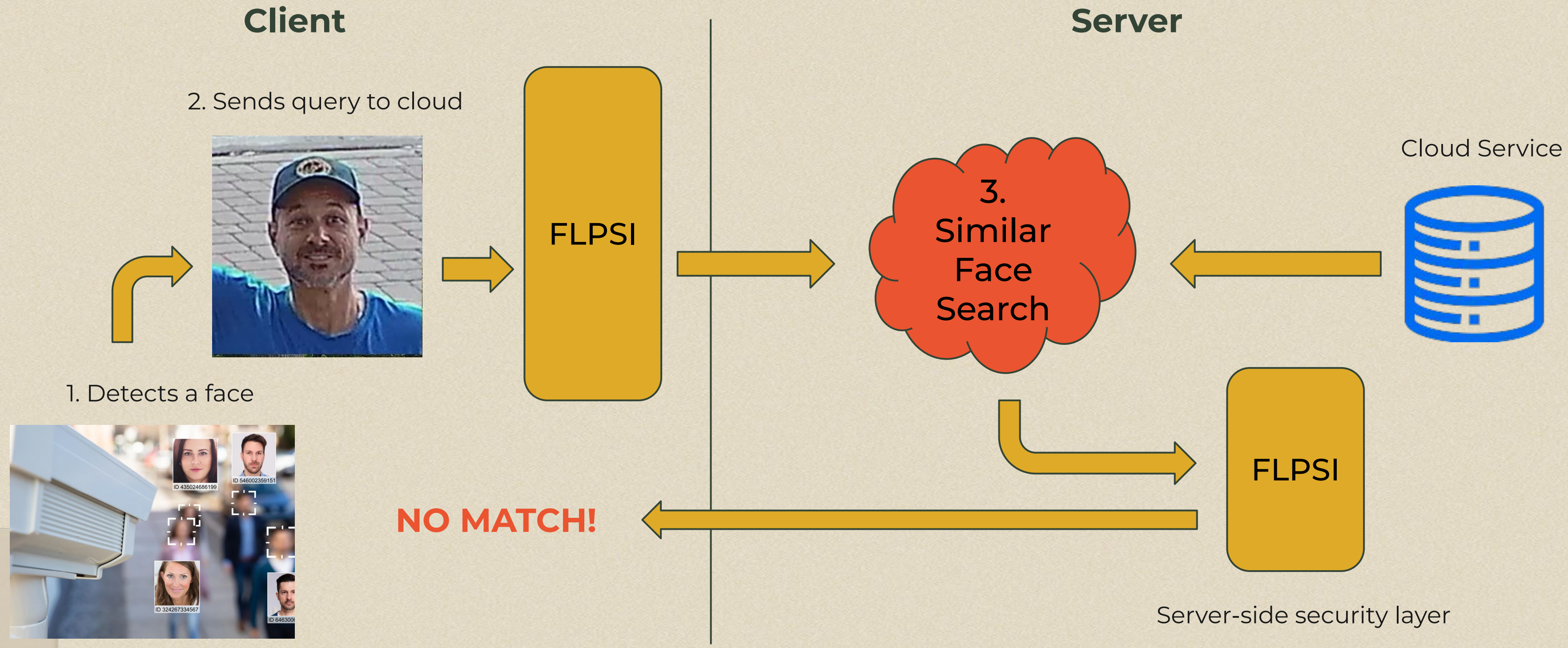

NO MATCH!

4. Displays result to client

- Client (eg: police surveillance) is interested in associated labels
- Server (eg: an AI company) learns the query and the result

## Issue: Privacy risk! Server learns the query and the result

## Solution: Fuzzy Labeled PSI (Erkam Uzun, Simon P. Chung et.al)



## Fuzzy

Biometric data is noisy. Matches are approximate. Two embeddings for the same person are not same.  Comparison by similarity.

## Labeled

There are distinct identifiers or labels associated with each data point in the database. Client is interested in labels.

# FLPSI acts as a privacy layer between client and server w/o extra hardware requirement

## GOAL

Client learns the result and learns nothing about the database. Server learns nothing about the query.

## BENEFITS

- Noise associated with biometric data is incorporated
- Communication sublinear in database size
- Protocol satisfies the given goal

# STATE OF THE ART

**CHLR 2018** : LPSI from FHE with malicious security

Hao Chen, Peter Rindal et. al.

+ Exact private matching
+ Sublinear communication
+ Efficient computation
+ Not directly applicable to fuzzy matching

**SAANS 2020**: Secure Approximate k-NN Search

Hao Chen, Illaria Chillotti et. al.

+ Accommodates fuzzy matching
+ High-bandwidth requirement; 1.7-5.4GB communication for 1M row database (500MB/s with 0.5ms latency)

# PERFORMANCE

## Response Time

For a 10K-row database, over WAN (resp. fast LAN) is 146 ms (resp. 47ms), transferring 12.1MB

## Offline precomputation

Offline precomputation (with no communication) time is 0.94s

## Scaling

For a 1M-row database, online time is 1.66s (WAN) and 1.46s (fast LAN) with 40.8MB of data transfer in online phase and 37.5s in offline precomputation. improving SAANS by 9-25x (on WAN) and 1:2-4x (on fast LAN)

## False Rate

0.75% for at most 10 false matches over 1M-row DB

# Binary Encoding

Biometric Data

$\xrightarrow[\text{as FaceNet}]{\text{DL model such}}$

$$\begin{bmatrix} \begin{bmatrix} -0.0722 \\ 0.06020 \\ \dots \\ -0.1332 \end{bmatrix} \\ \cdots \\ \begin{bmatrix} 0.07048, \\ \dots, \\ 0.008841 \end{bmatrix} \end{bmatrix}$$

Embeddings
(in Euclidean space)

$\xrightarrow[\text{Sensitive Hash}]{\text{Locality}}$

$$\begin{bmatrix} 1 \\ 0 \\ 0 \\ .. \\ .. \\ 1 \end{bmatrix}$$

$\xrightarrow[\text{Sensitive Hash}]{\text{Super-bit locality}}$

**BIO-BIT VECTORS**
256-bit binary vectors where two vectors from same person are **hamming close**

+ **LSH: Probabilistic dimensionality reduction** aiming to hash similar data samples to the same hash code.

$$h_v(x) = sign(v^T . x)$$

$$v \leftarrow \mathcal{N}\left(0, I_d\right)$$

+ **SBLSH (NIPS 2012, Jianqiu Ji, Jianmin Li et. al)**
Orthogonalization via Gram-Schmidt process, which projects the current vector orthogonally onto the orthogonal complement of the subspace spanned by the previous vectors.

# FLPSI : Offline Computation

## Client

Query
$q$

$\xrightarrow[\text{Encoding}]{\text{Binary}}$ $y = encode(q)$

## Server

$D_b = \{(d_1, l_1), \ldots, (d_N, l_N)\}$

$\xrightarrow[\text{Encoding}]{\text{Binary}}$ $x_i = encode(d_i)$

$l_i$
Label

$\xrightarrow[\text{Secret Sharing}]{\text{t-out-of-T}}$ $\{ss_{i1}, ss_{i2}, \ldots, ss_{iT}\}$

$[0^\lambda l_i]$

# FLPSI : Subsampling

## Client

Query $q$ $\xrightarrow[\text{Encoding}]{\text{Binary}}$ $y = encode(q)$

Subsampling

$\{y_1, y_2, \ldots, y_T\}$

## Server

$D_b = \{(d_1, l_1), \ldots, (d_N, l_N)\}$ $\xrightarrow[\text{Encoding}]{\text{Binary}}$ $x_i = encode(d_i)$

Subsampling

$$\begin{Bmatrix} x_{11}, x_{12}, \ldots, x_{1T} \\ \ldots \\ x_{N1}, x_{N2}, \ldots, x_{NT} \end{Bmatrix}$$

Server chooses 128-bit AES Blockcipher
Server samples $\{m_1, m_2, \ldots, m_T\}$
Server computes

$$x_{ij} = AES_{k_S}(x_i \wedge m_j)$$

Client and server run 2PC to compute $y_j = AES_{k_S}(y \wedge m_j)$

If the query matches with i'th face in database, atleast t out of T subsamples match with a high probability

# FLPSI : Strawman Design (STLPSI)

If t out of T subsamples match for a data point, the client can successfully reconstruct the label by receiving more than t secret shares

**Client**

Matched ← "John Doe"

Label

- Client and server agree on an FHE scheme
- Client samples $p_k$, $s_k$
- Client homomorphically encrypts $y_j$ and sends it with $p_k$
- Server computes

$$[[Z_{ij}]] = r \times \left( [[y_j]] - x_{ij} \right) + ss_{ij}$$

- Server sends $[[Z_{ij}]]$ to Client
- Client receives $ss_{ij}$ if $y_j = x_{ij}$ otherwise receives nothing

# Observations

- The bottleneck for FLPSI is its underlying FHE computations during Strawman phase
- The fixed value of t used for implementation originally is 2!

**i.e. most bits in matching bio-bit vectors are same!**

# Idea

**Directly compute the set intersection between two bio-bit vectors in a trusted environment.**

# Intel SGX

+ Intel Software Guard Extensions (**SGX**) is a set of instruction codes embedded in select Intel CPUs, a CPU-based mechanism for creating a Trusted Execution Environment (**TEE**), called an **enclave**, for user-level application code.
+ Enclaves, **hardware-isolated** runtime environments, defined by user-level and OS code, designate secure regions of memory protected by SGX, safeguarding data and code within them from unauthorized access.
+ CPU encryption of enclave memory **prevents access** to enclave data and code by other software, including OS and hypervisor code, enhancing security.
+ SGX finds application in secure remote computation, web browsing, DRM, and concealing proprietary algorithms and encryption keys.

## FLPSI-SGX

+ Client and Server compute bio-bit vectors over an offline phase
+ Client sends its bio-bit vector y to SGX
+ Server sends the bio-bit vectors $x_i$ for each data point to SGX along with a threshold parameter t
+ SGX computes the hamming distance between y and $x_i$ and compares it with t
+ SGX returns the label to client if a match occurs

# FLPSI-SGX acts as a privacy layer between client and server

## GOAL

Client learns the result and learns nothing about the database. Server learns nothing about the query.

## BENEFITS

- Noise associated with biometric data is incorporated
- Constant communication (not factoring in database communication)
- Protocol satisfies the given goal

# FLPSI-SGX : Offline Computation

## Client

Query
$q$

$$q \xrightarrow[\text{Encoding}]{\text{Binary}} y = encode(q)$$

## Server

$$D_b = \{(d_1, l_1), \ldots, (d_N, l_N)\}$$

$$D_b \xrightarrow[\text{Encoding}]{\text{Binary}} x_i = encode(d_i)$$

# FLPSI-SGX

## Client

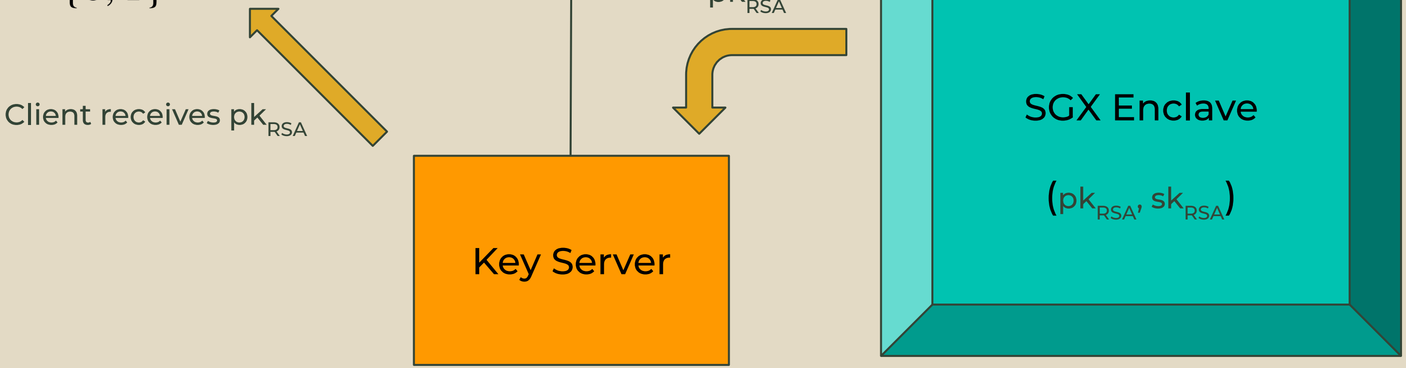
Query $q$ $\xrightarrow[\text{Encoding}]{\text{Binary}}$ $y = encode(q)$

Client samples AES key

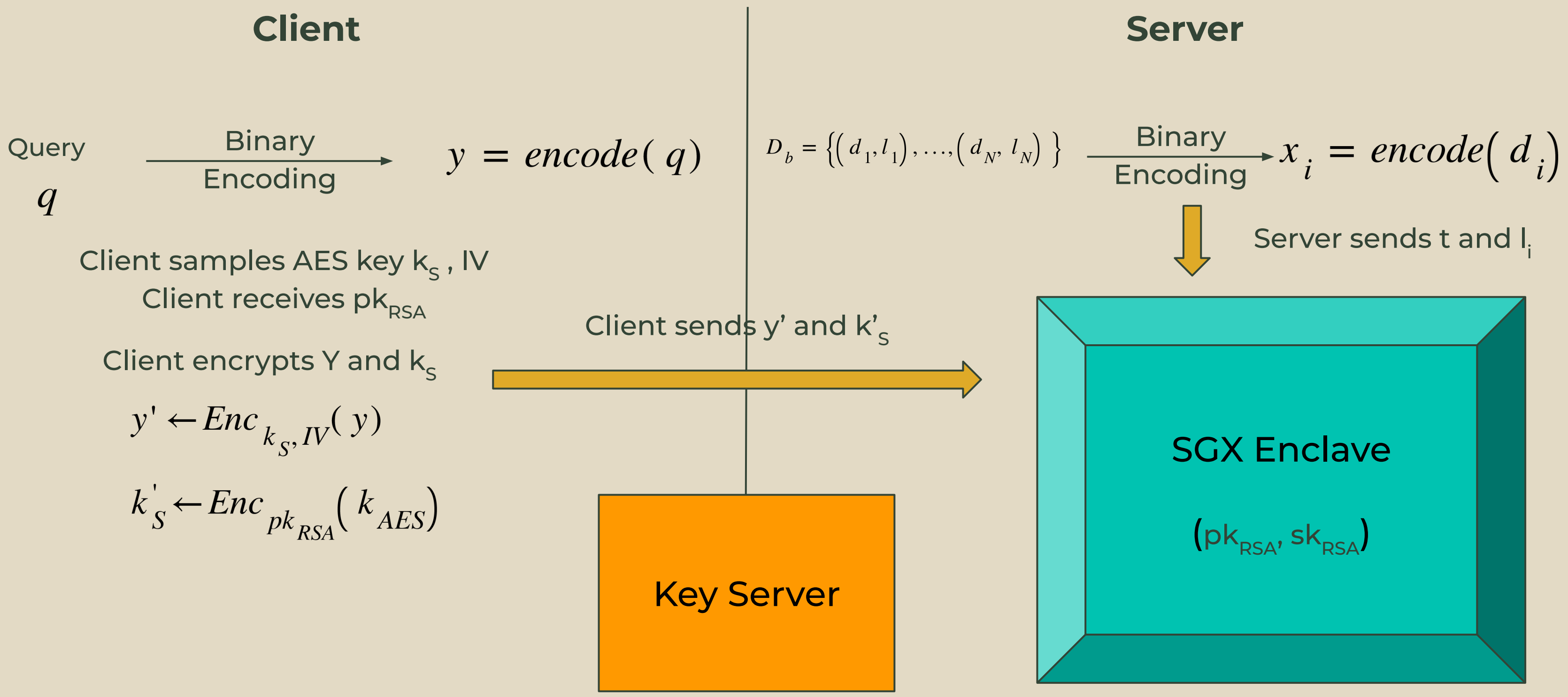
$$k_S \xleftarrow{\$} \{0, 1\}^{128}$$

## Server

$D_b = \{(d_1, l_1), \ldots, (d_N, l_N)\}$ $\xrightarrow[\text{Encoding}]{\text{Binary}}$ $x_i = encode(d_i)$



# FLPSI-SGX


Client
Server
Query
$q$
Binary Encoding
$y = encode(q)$
$D_b = \{(d_1, l_1), \ldots, (d_N, l_N)\}$
Binary Encoding
$x_i = encode(d_i)$
Server sends t and $l_i$
Client samples AES key $k_S$ , IV
Client receives $pk_{RSA}$
Client encrypts Y and $k_S$
$y' \leftarrow Enc_{k_S, IV}(y)$
$k'_S \leftarrow Enc_{pk_{RSA}}(k_{AES})$
Client sends y' and $k'_S$
SGX Enclave
$(pk_{RSA}, sk_{RSA})$
Key Server

# FLPSI-SGX

## Client

Query

$q$ $\xrightarrow[\text{Encoding}]{\text{Binary}}$ $y = encode(q)$

Client samples AES key $k_S$ , IV

Client receives $pk_{RSA}$

Client encrypts and sends Y and $k_S$

## Server

$D_b = \{(d_1, l_1), \ldots, (d_N, l_N)\}$ $\xrightarrow[\text{Encoding}]{\text{Binary}}$ $x_i = encode(d_i)$

Server sends t and $l_i$

Key Server

SGX Enclave

$k_S \leftarrow Dec_{pk_{RSA}}(k'_S)$

$y \leftarrow Dec_{k_S, IV}(y')$

SGX decrypts Y and $k_S$

# FLPSI-SGX

## Client

Query $q$ $\xrightarrow[\text{Encoding}]{\text{Binary}}$ $y = encode(q)$

Client samples AES key $k_S$ , IV

Client receives $pk_{RSA}$

Client encrypts and sends Y and $k_S$

"John Doe"

## Server

$D_b = \{(d_1, l_1), \ldots, (d_N, l_N)\}$ $\xrightarrow[\text{Encoding}]{\text{Binary}}$ $x_i = encode(d_i)$

Server sends t and $l_i$

SGX Enclave

SGX sends RES

Key Server

$$RES \leftarrow \begin{cases} Enc_{k_S, IV}(l_i), & if\ \ hd(y, x_i) \le t \\ Enc_{k_S, IV}(\text{" nomatch "}), & otherwise \end{cases}$$

*here, hd(a, b) is the number of bits where a and b differ

# Observations

- The bottleneck for FLPSI is its underlying FHE computations during Strawman phase
- The fixed value of t used for implementation originally is 2!

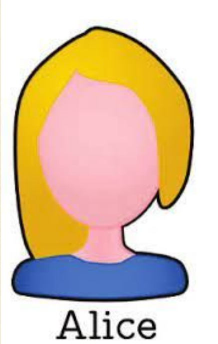
**i.e. most bits in matching bio-bit vectors are same!**

# Idea

**Directly compute the set intersection between two bio-bit vectors in a trusted environment.**

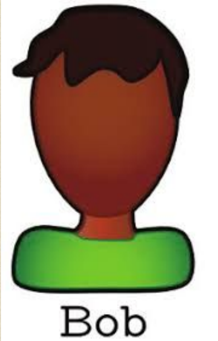
# Methodology: PICT

Ghosh, S. and Simkin, M. (2019)




Idea : Sparse polynomials are easy to interpolate

$$P(x)\text{-}Q(x) = -x^e + x^a$$




1, 3, 5, 6, 8, 9

$$P(x) = x^a + x^b + x^c$$

1, 2, 5, 6, 7, 9

$$P(x) = x^b + x^c + x^e$$

# IDEAS

01

Encode bio-bit vectors as sets of positions i.e. if the ith bit is ON, add i to the set

02

Use PICT to find out if the two input sets are similar enough to have intersection size less than threshold (t' = 2t)

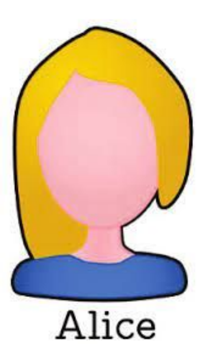
03

Since both sets are of maximum size 256, use SGX to test invertibility of Hankel matrices

# Methodology: Set Reconciliation with FHE

Ghosh, S. and Simkin, M. (2019)
extension to Minsky et. al. (2003)

Idea : If the intersection is very large, it's enough to know the set difference




1, 3, 5, 6, 8, 9     1, 2, 5, 6, 7, 9


Alice need only know about 3, 8

# IDEAS

01

Protocol requires client to send 2t+1 ciphertexts and Server to send t ciphertexts : $\mathcal{O}(t \log p)$ bits are communicated if elements drawn from $\mathbb{F}_p$

02

Relies on FHE and thus can only be instantiated from lattice based assumptions

03

Since t = 4 for sets encoded from bio-bit vectors, homomorphic rational polynomial interpolation may be practical

# FLPSI-SGX Technology Used

+ **Docker** to run the server in an isolated environment
+ **Gramine Shielded Containers (GSC)** to graminize the docker image. **Gramine Library OS** executes image in SGX
+ **Python** for implementation. **Cryptography** library for RSA and AES encryption
+ **FastAPI** as web framework, used for making fetch requests and queries
+ **HTML/CSS/JavaScript** to implement frontend

## FLPSI-SGX Dataset & Model

+ **Celebs-Faces** collection of facial images : total 107,818 images of 1063 individuals of varied sizes. Additional CSV file for metadata
+ Python implementation of **FaceNet** as used by (Uzun et al., 2021)
+ **Python** for implementation of SBLSH (after comparision with Java Implementation)

```
print(len(bioBitVectors))
>> 1063
print(len(bioBitVectors["Keanu Reeves0"]))
>> 256
hamming_dist(bioBitVectors["Keanu Reeves0"], bioBitVectors[
                              "Keanu Reeves1"])
>> 28
hamming_dist(bioBitVectors["Keanu Reeves0"], bioBitVectors[
                              "Alex Lawther1"])
>> 63
```

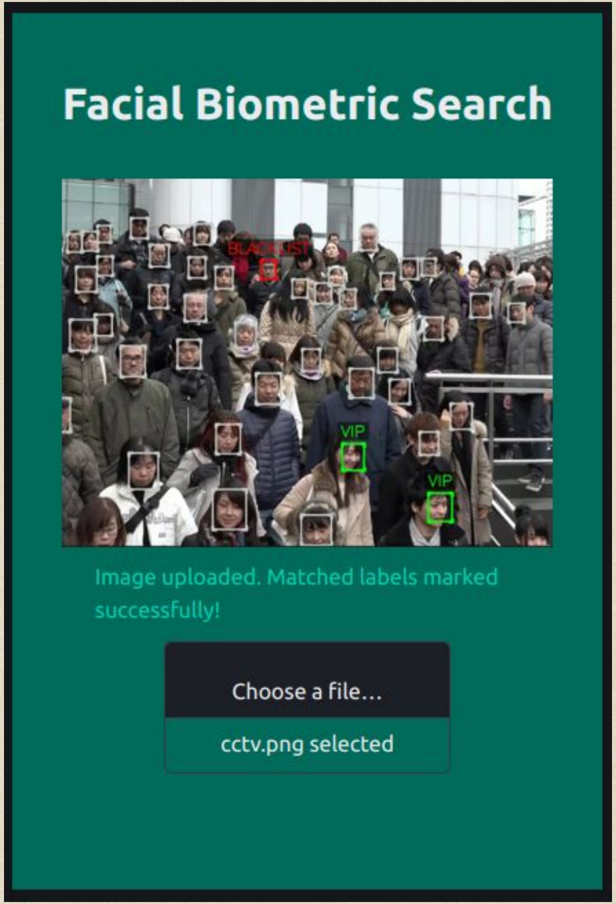
# FLPSI-SGX Results

| Number of queries (iterations) | Total Time Taken (in seconds) | Average time per query (in seconds) | Memory usage | Memory usage % | Block I/O |
|---|---|---|---|---|---|
| 1 | 0.20800304 | 0.210 | 754.6MiB | 4.82 | 4.1 KB/ 0B |
| 5 | 0.47165036 | 0.094 | 755.3MiB | 4.80 | 4.0 KB/ 0B |
| 10 | 0.57032108 | 0.057 | 755.5MiB | 4.82 | 4.1 KB/ 0B |
| 100 | 7.190250158 | 0.072 | 754.7MiB | 4.81 | 4.1 KB/ 0B |
| 1000 | 46.90907669 | 0.047 | 755MiB | 4.82 | 4.1 KB/ 0B |
| 10000 | 483.3544154 | 0.048 | 754.9MiB | 4.82 | 4.1 KB/ 0B |

Query Iterations over constant t

| Number of queries (iterations) | CPU Usage | Total Time Taken (in seconds) | Average time per query (in seconds) | Memory Usage | Memory Usage % | Block I/O |
|---|---|---|---|---|---|---|
| With SGX | 1231.25% | 483.3544154 | 0.048 | 754.9MiB | 4.82 | 4.1 KB/ 0B |
| Without SGX | 102.96% | 67.25915026 | 0.006 | 72.14MiB | 0.46% | 12.3 KB/ 4.1kB |

With and without using SGX




# FLPSI-SGX Application

## Conclusions and Future Work

+ Using models other than FaceNet for online queries. Federated learning.
+ SBLSH implementation not as efficient. Noise Removal techniques. Doesn’t pose a problem for FLPSI-SGX. Improvement left for future work.
+Optimizations beyond Strawman Design. Further work on FLPSI-PICT and Set Reconciliation
+ Implementation without hybrid encryption

# THANK YOU