Privacy preserving Biometric Search using Fuzzy Labeled Private Set Intersection and Intel SGX

Project-2 (CS47006) report submitted to

Indian Institute of Technology Kharagpur

in partial fulfilment for the award of the degree of

Bachelor of Technology

in

Computer Science and Engineering

by

Vineet Amol Pippal

(20CS30058)

Under the supervision of

Professor Satrajit Ghosh

Department of Computer Science and Engineering

Indian Institute of Technology Kharagpur

Spring Semester, 2024-25

April 7, 2024

DECLARATION

I certify that

(a) The work contained in this report has been done by me under the guidance of my supervisor.

(b) The work has not been submitted to any other Institute for any degree or diploma.

(c) I have conformed to the norms and guidelines given in the Ethical Code of Conduct of the Institute.

(d) Whenever I have used materials (data, theoretical analysis, figures, and text) from other sources, I have given due credit to them by citing them in the text of the thesis and giving their details in the references. Further, I have taken permission from the copyright owners of the sources, whenever necessary.

Date: April 7, 2024 (Vineet Amol Pippal)

Place: Kharagpur (20CS30058)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR

KHARAGPUR - 721302, INDIA

CERTIFICATE

This is to certify that the project report entitled "**Privacy preserving Biometric Search using Fuzzy Labeled Private Set Intersection and Intel SGX**" submitted by **Vineet Amol Pippal** (Roll No. 20CS30058) to Indian Institute of Technology Kharagpur towards partial fulfilment of requirements for the award of degree of Bachelor of Technology in Computer Science and Engineering is a record of bona fide work carried out by him under my supervision and guidance during Spring Semester, 2024-25.

Date: April 7, 2024

Place: Kharagpur

Professor Satrajit Ghosh
Department of Computer Science and Engineering
Indian Institute of Technology Kharagpur
Kharagpur - 721302, India

Abstract

Name of the student: **Vineet Amol Pippal** Roll No: **20CS30058**

Degree for which submitted: **Bachelor of Technology**

Department: **Department of Computer Science and Engineering**

Thesis title: **Privacy preserving Biometric Search using Fuzzy Labeled Private Set Intersection and Intel SGX**

Thesis supervisor: **Professor Satrajit Ghosh**

Month and year of thesis submission: **April 7, 2024**

The document explores the challenges and advancements in the field of privacy-preserving biometric search discussing various studies made in the field of PSI and its extensions to Labeled and Fuzzy settings, essentially exploring the development of protocols that allow two parties to determine the intersection of their respective sets of biometric data while preserving the privacy of individual elements. We present and implement a system using Fuzzy Labeled PSI and Intel SGX which aims to protect sensitive data and code from unauthorized access while performing biometric searches. We demonstrate feasibility, although challenges such as query performance and optimization remain for future work. Furthermore, we discuss potential enhancements including the incorporation of other cryptographic technologies and mitigating security concerns associated with SGX. We present a theoretical basis for two protocols based on Private Intersection Cardinality Testing and Set Reconciliation. Overall, this report provides insights into the development of privacy-preserving protocols for biometric data search applications.

Acknowledgements

I would like to extend my heartfelt thanks to my mentor, Prof. Satrajit Ghosh, for their exceptional guidance and unwavering support throughout this project. Without their expertise and encouragement, this project would not have reached its full potential. Prof. Ghosh's continuous motivation to explore diverse avenues, delve into numerous research papers, and experiment with various ideas has been invaluable.

I would also like to thank my parents, who motivated me to do the project and provided me with moral support. Furthermore, I am very grateful to Umang Singla, my fellow batchmate, for whose ideas and contributions this work has been a great success. Umang's insights and collaborative efforts have played a significant role in the project's achievements.

Vineet Amol Pippal

Contents

List of Figures

Chapter 1

Introduction

1.1 Introduction

The advent of deep learning (DL)-based biometric identification systems has revolutionized the realm of surveillance and security, enabling real-time identification of individuals in various public and private settings. These systems, powered by advanced algorithms, offer immense benefits in terms of public safety and customer convenience. However, their widespread adoption has raised significant concerns regarding privacy infringement and potential human rights abuses. Existing biometric surveillance systems often store and process collected data in plaintext on centralized servers, posing a serious risk to individuals' privacy. Furthermore, individuals lack the ability to opt out of these systems, as their faces captured in video footage are automatically uploaded to remote servers. While the identification of "persons of interest" may be justifiable for security purposes, indiscriminate tracking of all individuals raises legitimate concerns about illegitimate surveillance and privacy violations. By developing and evaluating newer protocols based on the principles outlined in the proposed $FLPSI$ protocol, which tackles the challenge of fuzzy

matching in biometric data by reducing it to an easier exact-matching subproblem, this thesis project seeks to contribute to the advancement of secure biometric search technologies, addressing critical privacy concerns while maintaining the efficacy of surveillance systems in safeguarding public safety and security.

1.2 Multi-Party Computation

Multi-party computation (MPC) is primarily concerned with a secure computation carried out collectively by many parties, without fully revealing information about each other; its subroutines like Private Set Intersection (PSI) enable two or more parties to determine the common elements in their respective sets without revealing any additional information, making it a pivotal tool for privacy-preserving data analysis. MPC and its associated challenges have been at the forefront of modern research, drawing significant attention for their intricate nature and potential to reshape secure collaborative computing. In this study, we build upon research done in the domain of Private Set Intersection over the recent years.

1.3 PSI and Related Protocols

Private Set Intersection is a cryptographic protocol that facilitates the secure and confidential determination of common elements between two or more sets without disclosing the individual elements. Protocols implemented have had many applications like online advertising (Pinkas et al., 2015), private contact discovery and botnet detection (Nagaraja et al., 2010) and have been studied in the two-party, the multi-party, and the server-aided setting with both passive and active security. Modifications to the protocol have been made in the past for privately computing the size of the set intersection rather than the intersection itself (Pinkas et al., 2018;

Kissner and Song, 2005). In contrast, other studies have been made to find out whether the set intersection size is above a certain threshold (Ghosh and Nilges, 2019; Ghosh and Simkin, 2023; Badrinarayanan et al., 2021; Branco et al., 2021).

Another primitive related to Private Set Intersection is Labeled PSI ($LPSI$) where two parties can identify common elements (matches) between their datasets based on the associated labels while keeping the actual data confidential. This approach finds applications in various domains, including privacy-preserving biometric identification, collaborative filtering for personalized recommendations, and secure multi-party computation. There have been extensive studies for exact private matches in a variety of scenarios (Chen et al., 2018, 2017; Kolesnikov et al., 2016). Other works have incorporated LPSI with fuzzy, real-world data for domains such as biometric identification (Uzun et al., 2021; Chen et al., 2020).

1.4 Motivation

The widespread adoption of biometric surveillance systems raises significant privacy concerns, with personal data often stored in plaintext on centralized servers. Lacking opt-out mechanisms, individuals face increased privacy risks. This thesis aims to address these concerns by developing novel protocols based on Labeled Private Set Intersection (LPSI) techniques. Leveraging the $FLPSI$ protocol as per (Uzun et al., 2021) and Private Intersection-Computation Threshold $PICT$ protocol (Ghosh and Simkin, 2019), the research will explore hardware optimizations like Intel SGX to devise more efficient protocols. By striking a balance between security and privacy, this work seeks to enhance the effectiveness and scalability of biometric identification systems while safeguarding individuals' privacy rights.

1.5 Our Contributions

In this thesis, we present two significant contributions that advance the field of privacy-preserving biometric identification. Building upon recent studies by (Uzun et al., 2021) and (Ghosh and Simkin, 2019), we introduce two novel protocols: FLPSI using Intel SGX and FLPSI using PICT. Our first contribution involves the implementation and comprehensive analysis using the FLPSI protocol utilizing Intel SGX, a hardware-based security technology. Through meticulous examination, we assess the protocol's performance while considering relevant security specifications. Additionally, we lay the theoretical groundwork for future exploration and development of the FLPSI protocol using PICT. Lastly, we briefly explore biometric membership using a modified set reconciliation protocol using FHE. These contributions underscore our commitment to integrating privacy-preserving techniques into real-world applications like biometric search, ensuring both security and efficiency in sensitive data handling.

Chapter 2

Related Work

This section provides a brief overview of pivotal research studies on our topic. These papers are significant references that influence and inform our work. We present concise summaries and critical insights, highlighting their contributions to the field and their relevance to our work.

2.1 Communication Complexity of Threshold Private Set Intersection

A study by Ghosh and Simkin (Ghosh and Simkin, 2019) investigates the communication complexity of Threshold private set intersection and establishes the first upper and lower bounds. The study shows that any such protocol has a communication complexity lower bound of $\Omega(t)$, where t is a threshold parameter. (Freedman et al., 2004) have previously proven a lower bound of $\Omega(n)$ on the communication complexity of any private set intersection protocol, where n is the size of the smaller input set which directly extends to the case of protocols that only compute the intersection size. It constitutes a fundamental barrier to the efficiency of these protocols.

This study shows an upper bound of $\tilde{\mathrm{O}}(t)$, assuming fully homomorphic encryption improving upon the earlier bound. It presents an efficient TPSI protocol with a communication complexity of $\tilde{\mathrm{O}}(t^2)$ based on additively homomorphic encryption. These results are specifically relevant to our work when we consider applications like Biometric Search, where a given fingerprint has to have a large intersection with a fingerprint from a database. This study also presents a communication-efficient protocol for private intersection cardinality testing, which privately computes whether two sets differ by more than a given threshold t or not. Our work directly builds upon these results as our protocol implementation employs a Private Intersection Cardinality Testing ($PICT$) functionality and other key ideas, such as encoding set elements as polynomial roots.

2.2 TPSI in a multi-party setting

Several studies have explored the TPSI in the multi-party setting. (Badrinarayanan et al., 2021) show that any multi-party TPSI protocol must have a communication complexity of $\Omega(nT)$. Further, they build two functionalities build around the problem running in $\mathrm{O}(nT)$ assuming threshold FHE and construct a computationally more efficient protocol running in $\tilde{\mathrm{O}}(nT)$ under a weaker assumption of threshold additively homomorhpic encryption. A study by (Branco et al., 2021) presents a new Cardinality Testing protocol that allows N parties to check if the intersection of their input sets is larger than $n - t$. The protocol incurs in $\tilde{\mathrm{O}}(Nt^2)$ communication complexity; henceforth obtaining a Threshold PSI scheme for N parties with communication complexity $\tilde{\mathrm{O}}(Nt^2)$. Implementation-wise, their protocol is similar to (Ghosh and Simkin, 2019) using PKE (Public Key Encryption) calls instead of OLE calls in the π_{TPSI} functionality, and their main contribution is a reduced communication complexity of $\tilde{\mathrm{O}}(Nt^2)$ for the multi-party setting. These studies are very

relevant for the multi-party setting and can be deemed useful for future extensions to this work.

2.3 Labeled PSI

In $LPSI$, the sender associates a label with each item in its set, allowing the receiver to obtain only the labels corresponding to the intersecting items without revealing additional information (Chen et al., 2017; Kolesnikov et al., 2016). This is particularly relevant in unbalanced PSI scenarios, where the receiver's set is significantly smaller and may involve low-power devices. Building upon existing protocols, recent advancements in LPSI have led to efficient solutions with small communication complexity and enhanced security features, such as Oblivious Pseudo-Random Function (OPRF) usage. A study by (Chen et al., 2018) builds upon prior works by (Chen et al., 2017) to tackle LPSI in an Unbalanced setting where (1) the receiver's set is significantly smaller than the sender's, and (2) the receiver (with the smaller set) has a low-power device. They add efficient support for arbitrary length items, construct and implement an unbalanced Labeled PSI protocol with small communication complexity, and also strengthen the security model using Oblivious Pseudo-Random Function (OPRF) in a pre-processing phase, improving upon (Chen et al., 2017). They develop a protocol for Labeled PSI to allow the receiver to learn a label l_i for each item $x_i \, \epsilon \, X \cap Y$, while still keeping the communication sublinear in $|X|$ (larger set). Further, they discuss how LPSI can be can be leveraged to achieve a reasonable notion of security against a malicious sender.

2.4 LPSI in fuzzy setting

Various studies have been made to apply Labeled PSI for fuzzy, real-life data in the domain of privacy preserving biometric systems. The SAANS 2020 paper (Chen et al., 2020) introduces novel techniques for secure approximate k-Nearest Neighbors Search (k-NNS), addressing the privacy concerns inherent in cloud-based services where the server takes queries from clients and returns responses without revealing sensitive information. Leveraging techniques such as Additive Homomorphic Encryption (AHE), Garbled Circuits (GC), and Distributed Oblivious RAM (DORAM), SAANS achieves scalability to databases with millions of entries while ensuring confidentiality of client queries and search results. Building upon previous works in secure k-NNS, SAANS presents improved protocols for top-k selection and a new algorithm tailored to secure computation, optimizing communication complexity and enhancing security using Oblivious Pseudo-Random Function (OPRF) in a pre-processing phase. Additionally, SAANS proposes algorithms for k-NNS, including an optimized linear scan approach and a clustering-based method that avoids computing all distances, demonstrating the efficacy of these protocols in achieving secure and efficient biometric identification in real-world applications.

A recent study by (Uzun et al., 2021) considers querying of real-life biometric scan against a private biometric database, where the querier learns only the label(s) of a matching scan and the database server learns nothing. Fuzzy Labeled Private Set Intersection (FLPSI) protocol is proposed, improving upon the communication complexity and overheads of the state-of-the-art (Chen et al., 2020), leveraging techniques such as binary encoding and noise incorporation to generate 'bio-bit vectors' from biometric input, ensuring hamming closeness between vectors of the same individual. To address the challenge of fuzzy matching, subsampling masks are applied

in a secure multi-party setting, followed by a t-out-of-T secret sharing scheme to mitigate false matches. Additionally, Set Threshold LPSI (STLPSI) protocol is introduced to further enhance privacy, allowing the client to distinguish matching subsamples while preserving secrecy. The proposed protocols are demonstrated to be secure against semi-honest adversaries, although limitations such as offline precomputation and vulnerability to malicious attacks are acknowledged. Through extensive experimentation, the effectiveness and efficiency of FLPSI and STLPSI protocols are validated, paving the way for secure and privacy-preserving biometric search applications.

2.5 Applications with Intel SGX

Intel Software Guard Extensions (SGX) is a hardware-based security feature introduced by Intel to enhance the security of application code and data. It provides a secure enclave in the CPU where sensitive computations can be performed, shielding them from other software and even the operating system. SGX employs secure memory regions called enclaves, which are isolated from the rest of the system's memory and are encrypted to prevent unauthorized access. However, SGX is susceptible to various attacks, including side-channel attacks such as cache-timing attacks and page-fault attacks, as well as software-based attacks facilitated by a compromised operating system. Despite these vulnerabilities, SGX is a powerful tool for privacy-preserving protocols, as it allows sensitive data to be processed in a secure environment, protecting it from unauthorized access or tampering. There have been a number of studies employing the use of SGX (Fisch et al., 2017; Ankele et al., 2016; Arnautov et al., 2018) noting its utility in privacy-based protocols albeit its security assumptions and difficulty in modeling and proving system-level security.

Chapter 3

Protocol Framework

3.1 Preliminaries

3.1.1 Public Key Encryption Scheme

A public key encryption scheme $\varepsilon = (KeyGen, Enc, Dec)$ consists of three algorithms:

KeyGen(1^λ): This algorithm generates a key pair consisting of a secret key (**sk**) and a public key (**pk**).

Enc(pk, m): This algorithm encrypts a message $m \in \mathcal{M}$ using the public key **pk** and produces a ciphertext c.

Dec(sk, c): This algorithm decrypts the ciphertext $c \in \mathcal{C}$ using the secret key **sk** and recovers the plaintext message m.

3.1.2 AHE and FHE

We refer to a public key encryption scheme ε as being additively homomorphic when it allows us to perform addition on encrypted values and multiply them by constants in plaintext. In simpler terms, if there are operations $\oplus$ and $\odot$ such that for any a and b in a set $\mathcal{M}$, and for any two ciphertexts $c_1 = Enc(pk, m_1)$ and ${}_c2 = Enc(pk, m_2)$, the following equation holds true: $(a \odot c_1) \oplus (b \odot c_2)$ results in the ciphertext $Enc(pk, a.m_1 + b.m_2)$. We assume that the message space for the protocol is a field $\mathbb{F}_p$. Similarly, we refer to a public key encryption scheme ε as being fully homomorphic when it allows us to perform both addition and multiplication operations on encrypted values, without needing access to the decryption key. $FLPSI$ as per (Uzun et al., 2021) use the Microsoft SEAL library as an implementation of FHE.

3.1.3 Oblivious Linear Function Evaluation (OLE)

Algorithm 1 Oblivious Linear Function Evaluation Functionality

Require: Sender input $(a, b) \in \mathbb{F}^2$ and receiver input $x \in \mathbb{F}$

1: Upon receiving a message $(\mathbf{inputS}, (a, b))$ from the sender with a, b; store a and b.
2: Upon receiving a message $(\mathbf{inputR}, x)$ from the receiver with x, store x.
3: Compute $y = a \cdot x + b$ and send $(\mathbf{output}, y)$ to the receiver.

In Oblivious Linear Function Evaluation, the receiver wants to obliviously evaluate a linear or affine function $f(x) = ax + b$ where a, b are input values known only to the sender and x is an evaluation point known only to the receiver. Through an ideal functionality $\mathcal{F}_{OLE}$ (as shown above), the receiver only learns the value $f(x)$, and the sender does not learn anything about the evaluation point x.

3.1.4 Secret Sharing

Shamir's secret sharing scheme is a (k, n)-threshold scheme based on polynomial interpolation over finite fields. It aims to divide a secret S (e.g., a safe combination) into n shares $S_1, \ldots, S_n$ such that:

1. Knowledge of any k or more shares S_i makes S computable. That is, the entire secret S can be reconstructed from any combination of k shares.

2. Knowledge of any $k-1$ or fewer shares S_i leaves S completely undetermined, such that the possible values for S remain as likely with knowledge of up to $k-1$ shares as with knowledge of 0 shares. The secret S cannot be reconstructed with fewer than k shares.

If $n = k$, then all shares are needed to reconstruct the secret S. Assuming the secret S can be represented as an element a_p of a finite field GF(q) (where q is greater than the number of shares being generated), Shamir's secret sharing involves randomly choosing $k-1$ elements $a_1, \ldots, a_{k-1}$ from GF(q) and constructing the polynomial $f(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_{k-1} x^{k-1}$. Any n points on the curve can be computed by setting $i = 1, \ldots, n$ to find points $(i, f(i))$. Each participant is given a point (a non-zero input to the polynomial) and the corresponding output. Given any subset of k of these pairs, a_0 can be obtained using interpolation, with one possible formula:

$$a_0 = f(0) = \sum_{j=0}^{k-1} y_j \left(\prod_{\substack{m=0 \\ m \neq j}}^{k-1} \frac{x_m}{x_m - x_j} \right),$$

where the list of points on the polynomial is given as k pairs of the form (x_i, y_i). Note that $f(0)$ is equal to the first coefficient of the polynomial $f(x)$.

3.1.5 Super-Bit Locality Sensitive Hashing

SBLSH (Ji et al., 2012) is an enhancement to the sign-random-projection LSH (SRP-LSH) for approximating angular similarity. SBLSH aims to provide an unbiased estimate of angular similarity with reduced variance compared to SRP-LSH. The key idea behind SBLSH is to orthogonalize random projection vectors in batches, ensuring that the resulting binary codes offer a more informative representation of data samples. By grouping these orthogonalized vectors into super-bits, SBLSH generates binary codes of a specified length, improving the accuracy of estimating pairwise angular similarity.

Given N independent and identically distributed (i.i.d.) random vectors $v_1, v_2, \ldots, v_N$ in $\mathbb{R}^d$ sampled from the normal distribution $N(0, I_d)$, where $1 \leq N \leq d$, performing the Gram-Schmidt process on them and producing N orthogonalized vectors $w_1, w_2, \ldots, w_N$, then for any two data vectors $a, b \in \mathbb{R}^d$, by defining N indicator random variables $X_1, X_2, \ldots, X_N$ as $X_i = I(h_{w_i}(a) \neq h_{w_i}(b))$, where $h_{w_i}(x) = sign(w_i^T x)$, we have $E[X_i] = \frac{\arccos(a^T b/||a||||b||)}{\pi}$, for any $1 \leq i \leq N$.

Given a dataset of n vectors $X = x_1, x_2, \ldots, x_n$ in $\mathbb{R}^d$ and a code length K, the SBLSH method generates binary codes of length K for each vector in X using the following steps:

- Randomly generate K i.i.d. random matrices $P_1, P_2, \ldots, P_K$ uniformly distributed on the Grassmann manifold $G_{d/2,d/2}$.
- For each vector $x_i \in X$, compute the K orthogonalized vectors $w_{i,1}, w_{i,2}, \ldots, w_{i,K}$ by performing the Gram-Schmidt process on the K random matrices $P_1, P_2, \ldots, P_K$ and the vector x_i.
- For each vector $x_i \in X$, generate the binary code $h(x_i) \in -1, +1^K$ by setting $h(x_i)k = sign(wi, k^T x_i)$ for $k = 1, 2, \ldots, K$.

Experimental results demonstrate that SBLSH achieves a significant mean squared error reduction in estimating angular similarity compared to SRP-LSH, making it superior in approximate nearest neighbor (ANN) retrieval experiments. Additionally, SBLSH maintains the same computational efficiency as SRP-LSH, making it a practical and effective solution for large-scale applications requiring efficient distance or similarity computations.

3.1.6 Gramine LibOS

Gramine-SGX (Tsai et al., 2017) facilitates the seamless execution of unmodified applications within SGX enclaves by integrating the Gramine library OS. The framework's design, as depicted in Figure 4.1, incorporates the libOS within the enclave to provide the runtime environment necessary for applications. This libOS includes the glibc C library and supports dynamic loading and linking, enabling the execution of unmodified binaries within the enclave. Additionally, the libOS interfaces with the Gramine library OS to implement the standard system call API. To optimize enclave efficiency, the libOS implements OS features such as memory segmentation to minimize enclave transitions. For communication beyond the enclave, a custom Gramine Application Binary Interface (ABI) is utilized. These ABI calls are intercepted by a Platform Adaptation Layer (PAL), which translates the enclave's Gramine ABI calls into a restricted set of system calls on the host OS. This section provides an overview of the OS Gramine library's features, enclave customizations, and mechanisms for securing the external interface.

3.1.7 Intel SGX: Security Guarantees

- **Integrity:** Intel SGX guarantees the integrity of code and data within enclaves by safeguarding them against tampering from other software on the same platform. This is achieved through mechanisms such as the Memory Encryption Engine (MEE) to protect DRAM and an integrity tree comprising cryptographic primitives for encryption, Message Authentication Code (MAC), and anti-replay measures.

- **Confidentiality:** SGX ensures that enclave contents remain confidential, shielded from any other software on the platform, including the operating system and hypervisor. This confidentiality is maintained through the use of a modified AES counter mode encryption scheme with a defined confidentiality boundary, leveraging the randomness approximation of AES as a permutation.

- **Authentication and Authorization:** SGX validates the authenticity and authorization of enclaves, verifying that they are executing on genuine Intel SGX-capable platforms and have the necessary authorization to operate. This validation process involves both local and remote attestation mechanisms.

3.1.8 Intel SGX: Security Concerns

Intel SGX offers enhanced security features by providing a secure execution environment called enclaves, which protect sensitive data and code from unauthorized access, even from privileged software such as the operating system. However, the security of SGX is still evolving, and the current version is susceptible to various side-channel attacks (Fisch et al., 2017). These attacks can be classified into physical attacks, where the attacker requires physical access to the CPU, and software attacks, which can be executed by compromised software running on the same host

as the CPU. While SGX does not claim to defend against physical attacks like power analysis, successful demonstrations of software attacks such as cache-timing attacks, page-fault attacks, branch shadowing, and synchronization bugs have raised concerns. To mitigate these vulnerabilities, it is crucial to ensure that enclave programs are data-oblivious, meaning they do not exhibit memory access patterns or control flow branches dependent on the values of sensitive data. By implementing data-oblivious programming techniques, developers can minimize the risk of information leakage through side-channel attacks and enhance the overall security guarantees provided by Intel SGX.

3.2 Protocols

We present formal definitions of previously defined protocols and use them to build or devise new protocols over varied settings.

3.2.1 FLPSI

This section presents Fuzzy Labeled PSI as per (Uzun et al., 2021). Our work later builds upon this protocol based on its parameters and specifications. The full protocol is defined in stages as given below:

- **Binary Encoding** : the client and server locally turn each of their raw biometric inputs (e.g., facial photos) $q, d_i \in \mathrm{D}$, for $i \in [N]$, into bio-bit vectors $y = \mathrm{Encode}(q)$ and $x_i = \mathrm{Encode}(d_i)$, respectively, so that if there is a q,d_i pair of the same person, then y, x_i are likely Hamming close. Bio-bit vector conversion utilizes a binary encoding step, where C and S locally

  - Encode their raw biometrics (q and d_i , resp.) into embedding vectors, by using a DL model (e.g., FaceNet (Schroff et al., 2015)) and
  - Translate the vectors in Euclidean space into Hamming, by using Super-Bit Locality Sensitive Hash (SBLSH) (Ji et al., 2012), that converts DL embeddings into bio-bit vectors (binary representation).

- **Subsampling and 2PC** : The client and server apply random projections to each bit vector y and x_i, respectively. This results in sets of subsampled bit vectors, $Y = \{y_1, \ldots, y_T\}$ and $X_i = \{x_{i1}, \ldots, x_{iT}\}$, where if y and x_i are close, then some subsamples of them will likely be the same. The server hides the subsampling projections from the client to avoid reconstructing the inputs in the database. To achieve this, the server chooses a 128-bit AES block cipher key k_S and generates the projection masks $\{mask_1, \ldots, mask_T\}$. The server can locally compute its subsample set X_i for each of its bio-bit vectors x_i such that $x_{ij} = \text{AES}_{k_S}(x_i \wedge mask_j)$, where $j \in [T]$. Next, both parties execute a 2-party computation (2PC) protocol, denoted as CAES and SAES. This protocol privately computes each $y_j \in Y$ such that $y_j = \text{AES}_{k_S}(y \wedge mask_j)$ for the client, allowing it to learn matched encrypted subsamples without acquiring knowledge of the projection masks and k_S. This 2PC protocol is implemented using Yao's Garbled Circuits (GC) through the EMP toolkit.

- **Set Threshold LPSI (STLPSI)** : A preliminary design involves C and S agreeing on an FHE scheme, where C generates public and secret keys (pk, sk) and sends pk to S. C then homomorphically encrypts each set item $y_j \in Y$ into a ciphertext $[[y_j]]$ and transmits it to S. Subsequently, S computes $[[Z_{ij}]] = r \times ([[y_j]] - x_{ij}) + SS_{ij}$ under encryption, where $r \in \mathbb{R}^P$ is refreshed for each computation, and SS_{ij} represents the secret share associated with x_{ij}. S forwards the ciphertext $[[Z_{ij}]]$ to C. Secret shares are sampled uniformly from

items in SS (equivalent to MS). Here, $z_{ij} = s_{ij}$ if and only if $y_j = x_{ij}$. Otherwise, z_{ij} is random on SS. This design ensures that C can reconstruct the label l_i if it obtains at least t shares of l_i. Otherwise, C gains no information and cannot discern potential matches below the threshold t.

Scaling the above evaluation technique to large databases, such as those containing millions of sets, is impractical. To address this issue, (Uzun et al., 2021) implement optimizations drawn from the LPSI literature, aimed at compressing database items and reducing the circuit depth in FHE evaluations. These optimizations, with the exception of bucketing, closely mirror those outlined in (Chen et al., 2018).

Parameter	Description	Value
t	Matching threshold	2
T	Number of subsamples	64
$\mathcal{L}$	Length of bio-bit vectors	256
$\mathcal{C}_{rb}$	Consistency threshold ratio	0.9
$\mathcal{N}_{sb}$	Number of subsampled bits	14
k	S's key for a AES blockcipher	$\{0, 1\}^{128}$
$\boldsymbol{P}$	Prime mod. of domain $\mathcal{F}_{\boldsymbol{P}}$	8519681
λ	security param. for token 0^{λ}	$\lfloor \log \boldsymbol{P} \rfloor = 23$
N	Number of database entries	[10K - 10M]

FIGURE 3.1: List of parameters and their fixed values for FLPSI

3.2.2 FLPSI-SGX

With the fixed parameter values noted earlier, we use the basic setting for FLPSI to devise a new protocol using Intel SGX. We refer to the SGX enclave running inside the server machine as simply SGX and the public key (sampled for RSA encryption) for communication with SGX, enlisted with a key-server as p_k. We refer to the AES

key sampled by client by k_{AES}. The protocol thus relies on the hamming-closeness of the bio-bit vectors produced in the offline phase of FLPSI.

FLPSI-SGX allows a client with a query data q and a server with a database D to securely perform a private biometric search. The client and server first convert their biometric inputs into bio-bit vectors using binary encoding. The server enlists a key with the SGX and provides threshold parameters. The client encrypts the query bio-bit vector and sends it to SGX. SGX decrypts the query and compares it with the database. If there's a match, SGX sends the corresponding label encrypted over a key to the client; otherwise, it sends a "nomatch" signal. The client receives the result from SGX, and both client and server output accordingly. Roughly, the bio-bit vectors are directly compared inside the SGX enclave preventing the need of separate t-out-of-T secret shared labels and a Set Threshold PSI phase; thus reducing circuit complexity and communication overhead.

Algorithm 2 FLPSI-SGX

Require: Client has a query data q and Server has database $D = \{(D_1, l_1), (D_2, l_2), ..., (D_N, l_N)\}$

1: Client and Server run an offline computation to convert their biometric inputs into bio-bit vectors Y and X_i for $i \in [N]$ respectively, via binary encoding step in FLPSI
2: SGX enlists p_k with the key-server and receives l_i for $i \in [N]$ and threshold parameter t from Server
3: Client receives p_k from the key-server and samples AES key k_{AES}. $[OS.urandom(16)]$
4: Client encrypts query bio-bit vector Y with k_{AES} and k_{AES} with p_k
5: Client sends $< Y >$ and $< k_{AES} >$ to SGX
6: SGX decrypts Y and k_{AES}. If $\sum Y \oplus X_i \leq t$, SGX sets RES $= < l_i >$ encrypted over k_{AES}; otherwise, sets RES $= <$ "$nomatch$" $>$
7: Client queries SGX and receives RES.
8: Client outputs RES and Server outputs $\perp$

As can be inferred, since query bio-bit vector is sent to the server by the client only once and all bio-bit vectors have a constant size of $\pounds = 256$ bits, the overall communication taking place over the protocol is independent of the size of the database

albeit excluding the communication of encrypted AES keys i.e. a constant number of bits are communicated over the secure channel. Since most of the computation happens within the SGX enclave and the key server acts as a certified trusted authority, the security specifications of the protocol rely on the security specifications of the SGX enclave. The protocol ensures, in a semi-honest setting, the confidentiality of both the query data and the database. The client encrypts the query bio-bit vector with an AES key, ensuring that only the SGX enclave can decrypt and process it. Similarly, the server's database remains encrypted within the enclave, protecting it from unauthorized access. We leave formally proving the security of the protocol in a malicious setting for future work.

3.2.3 PICT

The protocol by (Ghosh and Simkin, 2019) is split into two main subprotocols where Private Intersection Cardinality Testing is first done on sets S_A and S_B to determine if the sets are similar or not. Formally, we test if $|(S_A \setminus S_B) \cup (S_B \setminus S_A)| \leq 2t$ without revealing set information through the ideal functionality $\mathcal{F}_{PICT}^{2t}$ (as shown below) implemented by the private intersection cardinality test protocol Π_{ICT}^{2t} (as shown below).

Algorithm 3 Ideal Functionality, Private Intersection Cardinality Testing

Require: Alice and Bob have as input set S_A and S_B respectively.

1: Upon receiving message (**inputA**, S_A) from Alice, store S_A.
2: Upon receiving message (**inputB**, S_B) from Bob, store S_B.
3: If $|(S_A \setminus S_B) \cup (S_B \setminus S_A)| \leq 2t$, then the functionality outputs **similar**, otherwise **different**, to Alice and Bob.

- Alice and Bob (the two participating parties) encode their sets as monic polynomials $p_A(x) = \sum_{i=1}^{n} x^{a_i}$ and $p_B(x) = \sum_{i=1}^{n} x^{b_i}$ in $\mathbb{F}_q[X]$. Alice samples a

random u from the field and computes her Hankel matrix evaluated over u before sending it Bob.

- After receiving the encrypted matrix from Alice, Bob leverages the encryption scheme's homomorphic properties to calculate the Hankel matrix associated with the polynomial representation of the symmetric set difference.

Algorithm 4 Private Intersection Cardinality Testing

Require: Alice and Bob have input set $S_A = \{a_1, ..., a_n\} \in \mathbb{F}_p^n$ and $S_B = \{b_1, ..., b_n\} \in \mathbb{F}_p^n$ respectively.

1: Alice and Bob encode their sets as polynomials $p_A(x) = \sum_{i=1}^{n} x^{a_i}$ and $p_B(x) = \sum_{i=1}^{n} x^{b_i}$ in $\mathbb{F}_q[X]$
2: Alice picks uniformly random $u \leftarrow \mathbb{F}_q$
3: Alice samples an encryption key $(sk, pk) \leftarrow KeyGen(1^\lambda)$ and computes an encrypted version of her Hankel matrix H_A following $H_A[i][j] = c_{i+j}$ for $i, j \in [0, 2t]$ where $c_i \leftarrow Enc(pk, p_A(u^i))$.
4: Alice sends H_A, u and pk to Bob.
5: Bob computes his own encrypted Hankel matrix H_B and computes $H_C = H_A - H_B$
6: Alice sends sk and Bob sends H_C to the ideal functionality $\mathcal{F}_{INV}$.
7: If Bob gets back **singular** from $\mathcal{F}_{INV}$, then he sends **similar** to Alice and outputs **similar** himself; otherwise he sends and outputs **different**.

- Ideal functionality $\mathcal{F}_{INV}$ is used by Bob to learn whether the resultant Hankel matrix is singular or invertible, denoting whether the size of the set difference is above a threshold. Based on the result given by Π_{ICT}^{2t} as depicted below, the parties either output $\perp$ if their sets differ by more than $2t$ elements or engage in a secure set intersection protocol.

3.2.4 FLPSI-PICT

We use the PICT module presented above to devise a novel protocol on a theoretical basis. FLPSI-PICT outputs a bit representing if the biometric input belongs to the database with the server or not; this can be used in a setting where it is only sufficient

to inquire set membership without necessarily notifying one of the parties. The protocol essentially leverages the size of bio-bit vectors produced by the offline phase of FLPSI protocol working over the assumption that two bio-bit vectors belonging to the same person are hamming close. To incorporate Intersection Cardinality Testing to find out if this is the case, we further encode bio-bit vectors on both the client side and the server side with positional encodings. Essentially, for bio-bit vector Y, we have $pos\_encode(Y) = \{i : Y[i] = 1\,, i\,\epsilon\,[0, 255]\}$; this transforms each vector to a numerical set of positions. Next, we run the PICT protocol over these sets to check if their intersection is small enough. Since the fixed value of t for FLPSI is 2 over given assumptions and the intersection size threshold is thus $2t$, PICT requires a low communication overhead for Hankel matrices H_A and H_C ($4t^2$ elements).

Algorithm 5 FLPSI-PICT

Require: Client has a query q and Server has database $D = \{(D_1, l_1), (D_2, l_2), ..., (D_N, l_N)\}$

1: Client and Server run offline computation to convert their biometric inputs into bio-bit vectors Y and X_i for $i\ \epsilon\ [N]$ respectively, $Y = encode(q)$ and $X_i = encode(D_i)$
2: Client encodes $S_Y = pos\_encode(Y)$ and Server encodes $S_{X_i} = pos_encode(Xi)$
3: Client and Server run $\mathcal{F}_{PICT}$ with input sets S_Y, S_{X_i} for $i\,\epsilon\,[1, N]$
4: Client outputs PRESENT if $\mathcal{F}_{PICT}$ outputs **similar** for some i, otherwise outputs ABSENT. Server outputs $\perp$

The accuracy of FLPSI-PICT relies on the effectiveness of the binary encoding phase of FLPSI. Moreover, owing to the small size of used Hankel matrices, an SGX enclave can be used in place of the ideal functionality $\mathcal{F}_{INV}$. Essentially, the server can send H_C to the SGX enclave and the client can exchange a symmetric encryption key; the enclave computes the determinant of H_C and based on the answer, outputs the encrypted result to the client and/or the server. Using an SGX enclave instead of a two-party protocol like F_{INV} reduces communication complexity by allowing the server to directly send H_C to the enclave. However, this approach introduces hardware security concerns associated with SGX, such as side-channel attacks and

enclave compromise. Additionally, computations within SGX enclaves are generally slower compared to non-enclave environments, which can impact overall system performance and responsiveness. Therefore, while leveraging SGX for determinant computation may reduce communication overhead, it requires careful consideration of security risks and potential performance implications.

3.2.5 Set Reconciliation

The purpose fulfilled by the above protocol can be accomplished with Fully Homomorphic Encryption using the set reconciliation protocol presented in (Ghosh and Simkin, 2019) based on (Minsky et al., 2003) which obtains an upper bound of $\widetilde{O}(t)$ where t is a threshold parameter. Using FHE, the client encodes set S_Y as polynomial $p_C(x) = \Pi_{i=1}^{n}(x - a_i)$ and sends encrypted evaluations $\{p_C(\alpha_1), ..., p_C(\alpha_2 t)\}$ as well as an additional encrypted evaluation $p_C(z)$ with the uniformly random z. The server evaluates his set as a polynomial on the same points and then homomorphically interpolates $p_C(x)/p_S(x) = p_{C\setminus S}(x)/p_{S\setminus C}(x)$, where the gcd of numerator and denominator is 1, using the first 2t encrypted points to obtain polynomial p. Server computes

$$(p(x), p_C(x), p_S(x), z) \longmapsto \begin{cases} p_{A\setminus B}, & \text{if } p(z) = p_C(z)/p_S(z) \\ \perp, & \text{otherwise} \end{cases}$$

on the encrypted data and sends the client back the result. Although the protocol proceeds further to reveal the parties the exact set intersection, we do not have a requirement for such information in this scenario (where only membership is to be known). The client simply decrypts the result and gets to know if the biometric data with it is present with the server or not. Given the complexity of the protocol and size of bio-bit vectors noted earlier, this protocol theoretically runs with a lower

communication overhead albeit under primitives like FHE. Although, as noted in (Ghosh and Simkin, 2019), said protocol can only be instantiated from lattice-based assumptions and its practical use may be questionable owing to the fact that we require homomorphic rational polynomial interpolation on ciphertexts leading to a high communication complexity. We provide a theoretical basis for this protocol here and leave implementation and analysis to future work.

Chapter 4

Implementation and Results

This section highlights the various details about our implementation of FLPSI-SGX and the consequent analysis and performance.

4.1 Tech Stack

- **Python**: The core programming language for implementing the backend logic for client, server and key-server programs.
- **FastAPI**: A modern, fast web framework for building APIs with Python. Used for communication using fetch requests and making queries. It provides automatic interactive documentation and validation. Replaced Flask to improve performance and efficiency.
- **Cryptography Library**: Used for RSA and AES encryption and decryption operations. This library offers cryptographic primitives and recipes for Python developers. Replaced pyCrypto to improve performance and efficiency.
- **numpy**: Utilized for numerical operations and handling arrays efficiently.

- **CSV Module**: Used to read biometric data from a CSV file. Bio-bit vectors generated after offline phase are stored locally as CSV files.
- **Redis**: The server enlists the RSA public key to a key server using Redis. Redis is an open-source, in-memory data structure store, used as a database, cache, and message broker.
- **HTTP Requests**: The application makes HTTP requests to interact with the key server for enlisting the public key.
- **HTML/CSS/JavaScript**: A simple frontend is implemented using HTML for structure, CSS for styling, and JavaScript for client-side scripting. The frontend provides a user-friendly interface for uploading photographs and displaying search results.

4.2 Setup

1. **Biometric Data and Database Initialization:** At the outset, the client possesses biometric data, while the server maintains a database of biometric information. The client intends to query the server's database to determine if its biometric data matches any entry in the database, without revealing the query itself. Optionally, the user may upload the biometric data to an application frontend which runs a DL model internally (eg: for making embeddings, for sampling different facial data from same image etc.) which is related to Federated Learning.
2. **Offline Computation with FaceNet:** Both the client and server engage in an offline computation phase utilizing the FaceNet model. This model is employed to convert the raw biometric data into embeddings, which are higher-dimensional representations suitable for facial recognition tasks.

3. **Conversion to Bio-bit Vectors:** Subsequently, the embeddings obtained from FaceNet undergo further processing using a Super-bit Locality Sensitive Hashing (LSH) algorithm. This algorithm transforms the embeddings into bio-bit vectors, facilitating efficient comparison and retrieval of similar biometric data.

4. **Key Server Setup:** A key server is established to manage the mapping of usernames to public keys. Redis, a data structure store, is utilized internally to maintain this mapping efficiently. The key server implementation leverages FastAPI, a web framework, for robust and scalable API development.

5. **AES Key and IV Sampling:** Before initiating the protocol, the client samples an Advanced Encryption Standard (AES) key and Initialization Vector (IV). These cryptographic primitives are crucial for ensuring secure communication between the client and server during the protocol execution.

6. **SGX Enclave Public Key Enrollment:** Within the server environment, an Intel Software Guard Extensions (SGX) enclave is employed to enhance security. The SGX enclave enlists its RSA public key with the key server, enabling secure communication and cryptographic operations within the enclave.

7. **Local Storage of Bio-bit Vectors:** Prior to commencing the protocol, the bio-bit vectors derived from the offline computation phase are stored locally. Both the client and server have access to these stored vectors, facilitating efficient query processing without the need for repeated computation.

4.3 Flow

The execution flow of the privacy-preserving protocol involves several steps, including setting up the server, running the client, and integrating an SGX enclave for

enhanced security. The following commands outline the sequence of actions to execute the protocol:

1. **Building the Server Docker Image:** The Docker image for the server component is built using the Docker Compose command. This step ensures that all dependencies and configurations are properly encapsulated within the container environment.

2. **Signing the Docker Image:** Once the enclave image is built, it is signed using the GSC command-line tool. This step involves cryptographic signing of the image using a designated enclave key, ensuring the integrity and authenticity of the enclave code.

3. **Running the SGX-Enabled Server Container:** Finally, the server container is launched with SGX support enabled. The Gramine Shielded Containers (GSC) framework is utilized for this purpose, ensuring the secure execution of sensitive operations within the enclave. This involves mounting the SGX device and AESM socket to the container, allowing secure communication with the SGX enclave during runtime. The server container is launched using the Docker run command with specified network settings and port mappings. This command initiates the FastAPI-based server application, enabling it to listen for incoming requests on port 5000.

By following these steps, the privacy-preserving protocol is instantiated, enabling secure and confidential interaction between the client and server components while leveraging the SGX enclave for enhanced security guarantees.

4.4 Results

- **Dataset:** Celebs-Faces is a collection of facial images of various celebrities and public figures, including actors, musicians, politicians, athletes, and social media influencers. The dataset includes a total of 107,818 images of 1,063 different individuals. The images are provided in JPEG format and are of varying sizes and qualities. Some images are high-quality studio portraits, while others are candid snapshots taken in various settings. The images also vary in terms of lighting, poses, facial expressions, and other factors. In addition to the images, the dataset also includes a CSV file that contains metadata for each individual, including their name, gender, and the number of images available for that individual. The dataset is intended for use in research related to facial recognition and computer vision, and could be used to train machine learning algorithms for tasks such as face recognition, facial expression analysis, and age estimation. We use Python implementation of FaceNet as used by (Uzun et al., 2021) to convert faces to embeddings (Schroff et al., 2015). We use Python implementation of SBLSH (after comparision with Java Implementation) which yields relatively bigger values of threshold t.

```
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
```

- We use the client program to make multiple queries to the server and run FLPSI-SGX over iterations to find the average amount of time it takes to make a query and the memory used by the SGX enclave and docker:

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

FIGURE 4.1: Query iterations over constant t; Size of bio-bit vectors = 256 bits; Memory allocated to SGX = 256MB; maximum number of threads = 32

- We compare our construction with a docker image of the server without the use of GSX to assess latency and memory overheads and find possible scope for improvement:

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

FIGURE 4.2: Query iterations over constant t comparison; Number of queries = 10000; Memory allocated to SGX = 256MB; maximum number of threads = 32

The above table shows that usage of SGX with Gramine OS requires longer time and more memory due to the additional overhead associated with enclave initialization and secure memory management within the SGX environment. CPU usage reaches over 1200% and memory usage peaks at 754.9MiB. Despite the higher resource utilization, the total time taken for processing queries is 483.35 seconds (48ms per query). In contrast, the system without SGX exhibits lower resource usage, with

CPU and memory usage below 103% and 73MiB, respectively. The total time taken for processing queries is significantly lower at 67.26 seconds (6ms per query). While the system without SGX demonstrates better efficiency in terms of resource utilization and query processing time, the trade-off involves sacrificing the scalability and potential security benefits provided by SGX. Additionally, the block I/O metrics show negligible differences between the two systems, indicating similar disk activity levels.

- We do not integrate the client-server backend with an application frontend UI where the user/customer might upload images (for eg: from CCTV footage) as biometric data and receives matching labels when tested against the database with the server because the computation of bio-bit vectors is an offline phase. However, we demonstrate how a simple mobile application can be built using our protocol running in backend and some binary encoding algorithm running online with the client program to generate bio-bit vectors:

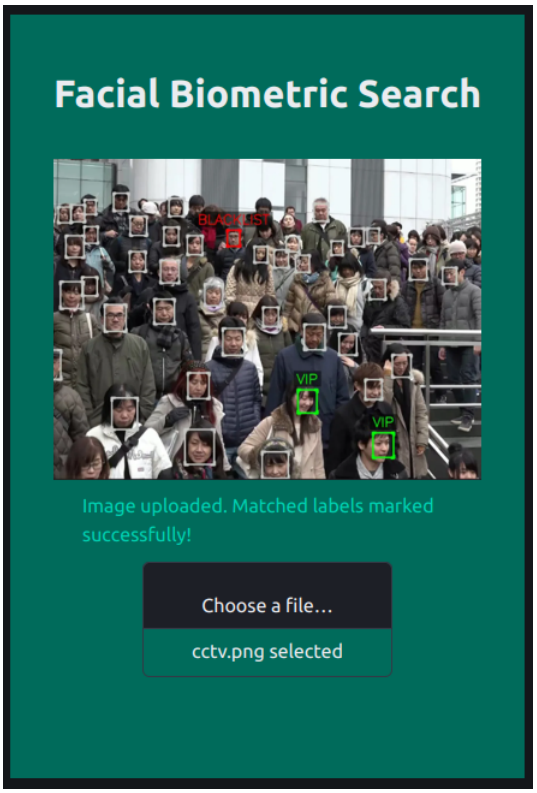

FIGURE 4.3: Biometric Search android application (demo) using FLPSI-SGX

Chapter 5

Conclusions and Future Work

In this study, we have presented privacy-preserving protocols for biometric search, leveraging state-of-the-art techniques. We achieved robust biometric representation and efficient similarity comparison, while SGX ensures secure computation and protection against unauthorized access. Our experimental results demonstrate the feasibility and effectiveness of the proposed protocol in preserving privacy while enabling biometric search functionality. However, there are several avenues for future exploration and improvement:

- Utilizing FaceNet embeddings obtained online through platforms like Kaggle, we have successfully demonstrated a prototype biometric search application. However, due to runtime constraints, the actual implementation of the application for real-time queries is avoided as the testing overheads do not factor in to query time as shown in (Uzun et al., 2021). Moreover, using online models other than FaceNet remains a Federated Learning challenge, which we defer to future investigations.

- While Super-bit Locality Sensitive Hashing (SBLSH) (Ji et al., 2012) theoretically yields threshold values close to 2, our practical implementation without

noise removal techniques results in higher threshold values (30-40 for same person faces and 60-70 for faces of different individuals). Although our protocol does not require improvement over these values, we acknowledge the need for improvements in efficiency and accuracy through heuristic approaches, which we plan to explore in future work.

- (Uzun et al., 2021) introduce several optimizations beyond the Strawman design presented in this work. We leave assessment of these to improve our own protocol for future work. Also, we plan to further work on FLPSI-PICT and Set Reconciliation protocols presented earlier.

- Future work will focus on integrating other Oblivious RAM (ORAM) technologies such as Intel TDX and addressing security concerns associated with SGX, including mitigating side-channel attacks. Additionally, we plan to explore running the server in a native SGX environment using Intel's Software Development Kit (SDK) to reduce memory overhead and enhance overall security.

Bibliography

Ankele, R., Küçük, K. A., Martin, A., Simpson, A., and Paverd, A. (2016). Applying the trustworthy remote entity to privacy-preserving multiparty computation: Requirements and criteria for large-scale applications. In *2016 Intl IEEE Conferences on Ubiquitous Intelligence & Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress (UIC/ATC/ScalCom/CBDCom/IoP/SmartWorld)*, pages 414–422. IEEE.

Arnautov, S., Brito, A., Felber, P., Fetzer, C., Gregor, F., Krahn, R., Ozga, W., Martin, A., Schiavoni, V., Silva, F., et al. (2018). Pubsub-sgx: Exploiting trusted execution environments for privacy-preserving publish/subscribe systems. In *2018 IEEE 37th symposium on reliable distributed systems (SRDS)*, pages 123–132. IEEE.

Badrinarayanan, S., Miao, P., Raghuraman, S., and Rindal, P. (2021). Multi-party threshold private set intersection with sublinear communication. In *IACR International Conference on Public-Key Cryptography*, pages 349–379. Springer.

Branco, P., Döttling, N., and Pu, S. (2021). Multiparty cardinality testing for threshold private intersection. In *IACR International Conference on Public-Key Cryptography*, pages 32–60. Springer.

Chen, H., Chillotti, I., Dong, Y., Poburinnaya, O., Razenshteyn, I., and Riazi, M. S. (2020). {SANNS}: Scaling up secure approximate {k-Nearest} neighbors search. In *29th USENIX Security Symposium (USENIX Security 20)*, pages 2111–2128.

Chen, H., Huang, Z., Laine, K., and Rindal, P. (2018). Labeled psi from fully homomorphic encryption with malicious security. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 1223–1237.

Chen, H., Laine, K., and Rindal, P. (2017). Fast private set intersection from homomorphic encryption. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 1243–1255.

Fisch, B., Vinayagamurthy, D., Boneh, D., and Gorbunov, S. (2017). Iron: functional encryption using intel sgx. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 765–782.

Freedman, M. J., Nissim, K., and Pinkas, B. (2004). Efficient private matching and set intersection. In *International conference on the theory and applications of cryptographic techniques*, pages 1–19. Springer.

Ghosh, S. and Nilges, T. (2019). An algebraic approach to maliciously secure private set intersection. In *Annual international conference on the theory and applications of cryptographic techniques*, pages 154–185. Springer.

Ghosh, S. and Simkin, M. (2019). The communication complexity of threshold private set intersection. In *Annual International Cryptology Conference*, pages 3–29. Springer.

Ghosh, S. and Simkin, M. (2023). Threshold private set intersection with better communication complexity. In *IACR International Conference on Public-Key Cryptography*, pages 251–272. Springer.

Ji, J., Li, J., Yan, S., Zhang, B., and Tian, Q. (2012). Super-bit locality-sensitive hashing. *Advances in neural information processing systems*, 25.

Kissner, L. and Song, D. (2005). Privacy-preserving set operations. In *Annual International Cryptology Conference*, pages 241–257. Springer.

Kolesnikov, V., Kumaresan, R., Rosulek, M., and Trieu, N. (2016). Efficient batched oblivious prf with applications to private set intersection. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 818–829.

Minsky, Y., Trachtenberg, A., and Zippel, R. (2003). Set reconciliation with nearly optimal communication complexity. *IEEE Transactions on Information Theory*, 49(9):2213–2218.

Nagaraja, S., Mittal, P., Hong, C.-Y., Caesar, M., and Borisov, N. (2010). {BotGrep}: Finding {P2P} bots with structured graph analysis. In *19th USENIX Security Symposium (USENIX Security 10)*.

Pinkas, B., Schneider, T., Segev, G., and Zohner, M. (2015). Phasing: Private set intersection using permutation-based hashing. In *24th USENIX Security Symposium (USENIX Security 15)*, pages 515–530.

Pinkas, B., Schneider, T., Weinert, C., and Wieder, U. (2018). Efficient circuit-based psi via cuckoo hashing. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 125–157. Springer.

Schroff, F., Kalenichenko, D., and Philbin, J. (2015). Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 815–823.

Tsai, C.-C., Porter, D. E., and Vij, M. (2017). {Graphene-SGX}: A practical library {OS} for unmodified applications on {SGX}. In *2017 USENIX Annual Technical Conference (USENIX ATC 17)*, pages 645–658.

Uzun, E., Chung, S. P., Kolesnikov, V., Boldyreva, A., and Lee, W. (2021). Fuzzy labeled private set intersection with applications to private {Real-Time} biometric search. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 911–928.