

RICE UNIVERSITY

**Large-Scale Privacy-Preserving Matching and  
Search**

by

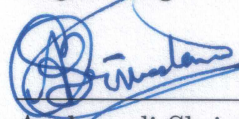
**M. Sadegh Riazi**

A THESIS SUBMITTED  
IN PARTIAL FULFILLMENT OF THE  
REQUIREMENTS FOR THE DEGREE  
**Master of Science**

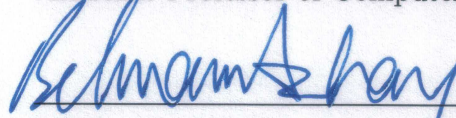
APPROVED, THESIS COMMITTEE:



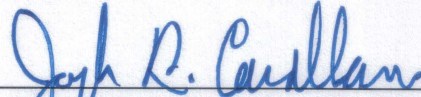
Farinaz Koushanfar, Chair  
Professor of Electrical and Computer  
Engineering



Anshumali Shrivastava  
Assistant Professor of Computer Science



Behnaam Aazhang  
J.S. Abercrombie Professor of Electrical  
and Computer Engineering



Joseph R. Cavallaro  
Professor of Electrical and Computer  
Engineering and Computer Science

Houston, Texas

April, 2016

## ABSTRACT

### Large-Scale Privacy-Preserving Matching and Search

by

M. Sadegh Riazi

The past few decades have witnessed considerable efforts for achieving a Privacy-Preserving Computing (PPC) scheme where the input data of each engaging party is not revealed to any other party. PPC, in fact, prevents any data leakage or invasion of privacy. While there exist provably secure solutions, they suffer from high communication overheads and therefore they cannot scale for large-scale datasets. This thesis proposes several novel techniques to overcome this limitation and opens a new door for real-world applications. In this thesis, we focus on two of the most important branches in this area which are matching and search. In the process of matching, two groups of individuals are interested to be matched with each other given certain rules and based on their preference list while keeping all preference lists private. In the process of search, a user holding a query wants to find the most similar profile (with a specific similarity metric) in a bank of profiles while keeping both query and bank private. Our approach is based on a well-known protocol called Garbled Circuit (GC) protocol which can securely evaluate any function of choice. Previous works suffer from immense overheads and lack of scalability. In this thesis, we introduce a new set of techniques that makes the GC-based protocols more efficient and make them scalable. We also study a new paradigm which is based on Randomized Embeddings to develop a new framework for a faster privacy-preserving

search that can deliver unprecedented speed up. We have developed a framework that introduces a privacy/accuracy trade-off and can process search query for Millions of users in real-time, an infeasible task prior to this work. Proof-of-concept evaluations on large-scale datasets prove the practicality and scalability of our approach. For the matching case, we evaluate our proposed solution on a bank of one million different genomes. Privacy-preserving stable matching is performed on a set size as big as National Residency Matching Program (NRMP). The last experiment is performed on the largest MovieLens dataset which contains hundreds of thousands different user profiles that is used to recommend a movie to a user based on the most similar profile without compromising the privacy of user and profiles in the dataset.

## Acknowledgments

Firstly, I would like to express my sincere gratitude to my advisor Professor Farinaz Koushanfar for the continuous support of my study and related research, for her patience, motivation, and immense knowledge. Her guidance helped me in all the time of research and writing of this thesis. I could not have imagined having a better advisor and mentor.

My sincere thanks also goes to Professor Anshumali Shrivastava. The door to his office was always open whenever I ran into a trouble spot or had a question about my research or writing.

I would like to thank the rest of my thesis committee: Professor Behnaam Aazhang and Professor Joseph R. Cavallaro for their insightful comments and encouragement. I would also like to thank the experts who were involved in the validation survey for this research project: Professor Ahmad-Reza Sadeghi, Professor Thomas Schneider, and Ebrahim M. Songhori. Without their passionate participation and input, the validation survey could not have been successfully conducted.

Last but not the least, I would like to thank my family: my wife, my parents, and my sister for supporting me spiritually throughout writing this thesis and my life in general.

# Contents

Abstract	ii
Acknowledgments	iv
List of Illustrations	ix
List of Tables	xii
<b>1 Introduction</b>	<b>1</b>
1.1 Privacy-Preserving Stable Matching . . . . .	2
1.2 Secure DNA Compatibility Testing . . . . .	4
1.3 Privacy-Preserving Near-Neighbor Search . . . . .	6
1.4 Contribution . . . . .	9
1.4.1 P3SM . . . . .	9
1.4.2 GenMatch . . . . .	9
1.4.3 S-LSH . . . . .	10
<b>2 Preliminaries and Background</b>	<b>12</b>
2.1 Secure Function Evaluation . . . . .	12
2.2 Garbled Circuit Protocol . . . . .	14
2.2.1 Optimizations . . . . .	15
2.2.2 Oblivious RAM . . . . .	16
2.2.3 Adversary Model . . . . .	16
2.3 Stable Matching . . . . .	18
2.3.1 General Stable Matching . . . . .	20
2.3.2 Limited Stable Matching . . . . .	22
2.4 Human Genome and DNA . . . . .	23

2.5	Approximate Near-Neighbor Search and Locality Sensitive Hashing (LSH) . . . . .	24
2.5.1	Popular LSH 1: Minwise Hashing and Resemblance Similarity	25
2.5.2	Popular LSH 2: Signed Random Projections (SimHash) and Cosine Similarity . . . . .	25
2.5.3	Mapping LSH to 1-bit . . . . .	26
<b>3</b>	<b>P3SM: Practical Privacy-Preserving Stable Matching</b>	<b>27</b>
3.1	High Level Architecture . . . . .	27
3.2	Circuits for Stable Matching . . . . .	29
3.2.1	Combinational Circuit . . . . .	30
3.2.2	Sequential Circuit . . . . .	31
3.2.3	Number of Proposals and Circuit Runtime . . . . .	33
3.2.4	Sub-linear Sequential Circuit . . . . .	35
3.2.5	Memory Analysis and Choosing ORAM Scheme . . . . .	37
3.3	Methodology for Scaling GC . . . . .	39
3.3.1	Scalable Circuit Generation . . . . .	39
3.3.2	Early Termination Technique . . . . .	40
<b>4</b>	<b>GenMatch: Secure DNA Compatibility Testing</b>	<b>42</b>
4.1	Scenario . . . . .	42
4.2	Pre-processing . . . . .	42
4.3	Boolean Circuit . . . . .	43
4.4	Circuit Optimizations and Size Model . . . . .	46
4.4.1	Circuit Size Model . . . . .	48
<b>5</b>	<b>S-LSH: Secure Locality Sensitive Hashing</b>	<b>50</b>
5.1	LSH and Triangulation Attack . . . . .	50
5.2	Making LSH Secure . . . . .	51

5.2.1	Formalization . . . . .	54
5.2.2	Secure Minwise Hashing (Secure MinHash) . . . . .	55
5.2.3	Secure Signed Random Projections (Secure SimHash) . . . . .	57
5.2.4	Scenario and the Formal Protocol . . . . .	59
5.2.5	Assumptions and Practical Issues . . . . .	62
5.2.6	Some Remarks: Impossibility . . . . .	63
5.3	Direct Applications . . . . .	65
5.4	Integrating with SFE Protocols . . . . .	67
<b>6</b>	<b>Experimental Evaluations</b>	<b>69</b>
6.1	P3SM Evaluations . . . . .	69
6.1.1	Evaluation Setup . . . . .	69
6.1.2	ORAM Analysis . . . . .	69
6.1.3	End-to-End Secure Stable Matching . . . . .	72
6.1.4	Discussion . . . . .	74
6.2	GenMatch Evaluations . . . . .	75
6.3	S-LSH Evaluations . . . . .	76
6.3.1	Sanity Check . . . . .	76
6.3.2	Performance Analysis . . . . .	77
<b>7</b>	<b>Related Work</b>	<b>82</b>
7.1	Privacy-Preserving Stable Matching Protocols . . . . .	82
7.2	Secure DNA Testing Algorithms . . . . .	85
7.3	Privacy-Preserving Near-Neighbor Search Algorithms . . . . .	87
7.3.1	Computational Security . . . . .	87
7.3.2	Information-Theoretic Security . . . . .	88
7.3.3	Privacy-Preserving Randomized Embeddings . . . . .	88
<b>8</b>	<b>Summary</b>	<b>90</b>

	viii
<b>A Triangulation Attack</b>	<b>92</b>
<b>B Circuit for Stable Match Algorithm</b>	<b>95</b>
<b>Bibliography</b>	<b>101</b>



# Illustrations

2.1	Different SFE protocols. . . . .	13
2.2	Example of preference matrices. . . . .	19
2.3	Example of stable and unstable match. . . . .	20
2.4	Sample HLA template for an individual. . . . .	24
3.1	Global flow of our secure SM system. . . . .	28
3.2	Block diagram architecture of the sequential circuit for implementing secure SM. . . . .	32
3.3	Preference Compare Circuit (PCC). . . . .	33
3.4	Number of proposals needed for different scenarios in the general SM. Statistical worst case is computed for $\alpha = 8$ ). The error bar shows the minimum and maximum value in the simulation. Experiment is performed 10 000 times for each set size with randomly generated preference list. . . . .	35
4.1	Block diagram of total end-to-end system including pre-processing (offline stage) and secure function evaluation (online stage). . . . .	43
4.2	Different stages for converting the Boolean Circuit to required input for Yao's garbled circuit protocol . . . . .	45
4.3	Architecture of the sequential circuit used in the GC protocol. . . . .	46
4.4	Number of non-XOR gates of the circuit vs. database size $N$ . . . . .	49

5.1	Triangulation Attack. The user $q$ and the random points $A$ , $B$ , and $C$ . The $d_A$ , $d_B$ , and $d_C$ are estimated distances from comparing hashes of $q$ and the other three points. . . . .	51
5.2	LSH and Secure LSH. . . . .	52
5.3	<b>Left:</b> The probability of collision plotted as function of $R$ (resemblance) for different values of $k$ (degree). <b>Right:</b> Mutual Information between two 1-bit hashes as a function of $R$ (resemblance) for different values of $k$ . . . . .	56
5.4	<b>Left:</b> Collision probability plotted as function of $\cos(\theta)$ for different values of $k$ (degree of transformation). <b>Right:</b> Mutual Information between two SimHashes as a function of $\cos(\theta)$ for different values of $k$ . . . . .	58
5.5	The collision probability of a 1-bit hash for traditional LSH and ideal secure LSH with threshold 0.8. . . . .	64
6.1	Comparison of memory access cost of different ORAM schemes for general SM without using ETT. . . . .	70
6.2	Comparison of memory access cost of different ORAM schemes for general SM using early termination technique. . . . .	71
6.3	Comparison of memory access cost of different ORAM schemes for limited SM with $k = 20$ . . . . .	72
6.4	Number of bit matches between a given query point and all other points in the MovieLens dataset as the function of Jaccard similarity where $l = 300$ and $k = 10$ . . . . .	77
6.5	Number of bit matches between a given query point and all other points in the MovieLens dataset as the function of cosine similarity where $l = 2^{10}$ and $k = 10$ . . . . .	78

6.6	Number of bit matches between randomly selected point and all other points in the uniform randomly generated dataset as a function of cosine similarity where $l = 2^{10}$ and $k = 10$ $n = 50000$ , $\mathbb{D} = 4$ . . . . .	80
6.7	Number of bit matches between randomly selected point and all other points in the uniform randomly generated dataset as a function of cosine similarity where $l = 2^{10}$ and $k = 10$ $n = 10000$ , $\mathbb{D} = 3$ . . . . .	81
A.1	Triangle localization. Three selected points are $A$ , $B$ , and $C$ . The calculated location of $q$ is the median of the triangle $DEF$ . . . . .	93

# Tables

3.1	Total complexity for implementing each variant of SM using different ORAM schemes, where $n$ is the set size and $k$ is the limit of number of proposals in limited SM. . . . .	38
6.1	Timing and communication results without using early termination technique. . . . .	73
6.2	Timing and communication results using early termination technique. . . . .	73
6.3	Timing and communication results for limited SM where $k = 20$ . . . . .	74
6.4	Number of XOR and non-XOR gates of circuit and total timing and communication for different size of database. . . . .	75
6.5	Experimental results for our near-neighbor search. $k$ is the degree of transformation. The last column shows the speed up over linear scan on hashes. . . . .	79
7.1	Different protocols for performing secure SM and corresponding complexities, where $n$ is the size of each group and $\nu$ is the number of matching authorities. Our protocol is faster by orders of magnitudes as it uses efficient Symmetric-Key operations instead of costly Public-Key operations. . . . .	83

# Chapter 1

## Introduction

This thesis aims at developing new solutions for privacy-preserving computing and looks at different approaches to reach its goal. The contribution of this thesis is two folds: (i) introducing a novel set of techniques that makes Garbled Circuit (GC) protocol more efficient and scalable. GC protocol is one of the well-known Secure Function Evaluation (SFE) protocols that can securely evaluate any function. Due to the high computational and communication cost of this protocol, it has limited practical usage. The first two part of this thesis explain how we can make the GC protocol more efficient. To demonstrate the effectiveness of our techniques, we have studied one of the hardest problems in this domain, called Stable Matching (SM). We will define SM and how to achieve a privacy-preserving version in Section 1.1. The results are published in the Proceedings of the Privacy Enhancing Technologies Symposium (PoPETs) 2017 [1]. We also introduce GenMatch, a secure DNA compatibility testing system. GenMatch allows a person holding her binary DNA information to find an organ donor that has compatible DNA in a case of organ transplant. GenMatch preserves both the privacy of person's sensitive DNA information and all other organ donors that have their DNA information in the genome bank. We explain this system in Section 1.2. The results are published in Hardware Oriented Security and Trust (HOST) 2016 [2]. (ii) Introducing a new framework that can translate state-of-the-art sub-linear near-neighbor search algorithms into a secure version [3]. Near-neighbor search is defined as: for a given query input, finding the most similar entity in the

database. Locality Sensitive Hashing (LSH) is an algorithm that enables sub-linear near-neighbor search and is based on Randomized Embeddings. This framework called Secure LSH (S-LSH) has taken its name from the fact that it can translate any LSH algorithm into a secure version one. In Section 1.3, we will define near-neighbor search and we will explain our framework. This is a powerful framework since it enables real-time search for millions of users for the first time. The computational complexity of this scheme is sub-linear in contrast to other SFE methods that are linear.

## 1.1 Privacy-Preserving Stable Matching

Stable Matching (SM) has substantial real-world applications: The National Residency Matching Program (NRMP) matches around  $32k$  graduating medical students to residency programs in the US every year [4, 5]. The New York City Department of Education (NYCDOE) matches over  $90k$  entering students to public high schools [6]. Also there are many financial applications that require SM, such as vertical networks and their application in supply chains [7]. In SM, there are two groups of individuals, e.g., men and women. Each individual ranks the members of the other group in a list sorted based on preference. The goal is to assign the members of these two groups to each other while satisfying the following post-match condition: there shall be no pairs from the two groups such that they prefer each other more than their already assigned partners.

SM use cases typically involve sensitive preference lists which have to be kept private. The current practice to ensure data privacy in SM is by exposing the personal preferences to a third party server and relying on its trustworthiness to perform a secure matching. However, relying on the trusted third party might be unacceptable

because there is still a high risk of information leakage and data abuse by the third party. Even if a third party server is indeed trustworthy, it can accidentally expose the user’s private data in the event of a compromise. In addition to information leakage, multiple studies [8, 9, 10] show that if certain individuals in a SM problem know the input of others, they could leverage this information to manipulate the results [11, 12]. Golle proposed a privacy-preserving SM system based on the Homomorphic Encryption [13]. Franklin et al. then improved this system and made it more efficient using an efficient multi-party indirect indexing [14, 15]. However, most of the previously proposed protocols for secure SM use a large number of expensive public-key operations and have not been implemented yet. The only solution based on symmetric key encryption is proposed by Keller et al. [16]. They reported that their approach can solve secure SM for 8 192 pairs (1/4 of the size of the NRMP) in  $1.5 \cdot 10^{12}$  seconds, i.e., almost 47 000 years! (see Table 7.1 in Section 7.1 for a detailed comparison).

This thesis introduces the first scalable secure SM system. Our approach leverages a well-known SFE protocol, called Yao’s Garbled Circuit (GC) [17], which is mainly based on efficient symmetric cryptographic operations. The input to this protocol is a Boolean circuit description of the function that needs to be evaluated securely. The total cost of the GC protocol is the total number of gates in the Boolean circuit. The conventional approaches in the GC protocol mainly relied on a combinational (directed acyclic) circuit description. In contrast, we utilize a recently proposed approach in [18] to describe the SM functionality as a compact sequential circuit. The size of the sequential circuit is less than the combinational circuit but it has to be evaluated for multiple iterations. Therefore, the total cost of the GC protocol is the product of the number of iterations and the size of the sequential circuit. We reach to

an unprecedented level of efficiency for solving secure SM by two sets of innovations. (i) We propose the first compact and efficient circuit with sub-linear size with respect to the number of pairs in SM ( $n$ ). The size of our circuit is  $\mathcal{O}(\log^3 n)$  which significantly improves over  $\mathcal{O}(n^2 \log n)$  for the naive sequential circuit. (ii) We present a novel technique, called early termination, to significantly decrease the computation time by reducing the total number of iterations that the SM sequential circuit needs to be evaluated. For example, for the set size  $8k$ , early termination results in three orders of magnitude improvement in computation time and communication (reducing the execution time from 5.62 years to 1.25 days). Although our focus is on SM algorithm, our methodology can be applied to other memory intensive algorithms by adapting our novel early termination technique and creating sub-linear sequential circuit.

As an application, we look at the NRMP which assigns around  $32k$  medical students to residency programs each year in the US. Our approach can perform this assignment securely, for the first time, with a reasonable timing and communication budget: 23.47 days computation and 272.2 TB total communication.

## 1.2 Secure DNA Compatibility Testing

Whole genome sequencing is a scientific process that is used to determine the complete DNA sequence of an organism [19]. A lot of progress has been made in this area in the past couple of decades. In fact, efforts are being made to commercialize this process because of the potential applications. A number of companies are competing for the market share of cost-effective platform for full genome sequencing [20].

The availability of embedded platforms capable of the full sequencing of a genome with on-board storage of the data in digital hardware, introduces new applications,



challenges, and opportunities. On one hand, several new and exciting applications can be realized at a fraction of today's cost. One such emerging application is personalized medicine where the effect of a drug on a person can be tested genetically to determine the drug's compatibility with the person. This can also lead to reducing the risks of side effects and adverse reactions. Another application is genetic compatibility which determines agreeableness of genomes between an organ donor and a recipient. The first generation sequencing companies have mostly failed in this type of testing, mainly due to the high cost associated with these procedures. Furthermore, in the earlier commercial solutions, both partners had to send genome samples to the testing company, which was both inconvenient and privacy invasive.

On the other hand, the privacy requirements for handling sensitive genome data arise serious challenges. Not only the sensitive DNA data reveals important personal information about the individual, but also, the process is irreversible and could cause lifelong irreparable damages to the DNA owner and her relatives sharing similar genes. As a result, it is necessary to devise methods that protect the privacy of individuals interacting with third party companies or research organizations handling genome data. Earlier work in this area explored ways of preserving privacy by anonymizing the data or formulating secure protocols. The current literature falls within one of the two categories: (i) heuristic solutions that lack strong security proofs or guarantees within the standard model [21, 22] or (ii) provably secure solutions that have limited scalability on real DNA datasets [23].

Our work explores practical privacy-preserving DNA testing based on Yao's provably secure Garbled Circuit (GC) protocol. Specifically, we consider Human Leukocyte Antigen (HLA) analysis which is a crucial test in organ transplantation [24, 25, 26, 27]. First, the raw genome data of the organ receiver (or donor) is

processed to obtain required HLA information for performing the privacy-preserving genome test. (In our case, we used the raw genome data from the Human Genome Project [28] database.) Then, the GC protocol is used to compare this HLA data with an existing HLA database of donors (or receivers) to compute the best match for the HLA data while preserving the privacy of all parties involved. We adopt the TinyGarble platform [18] to implement the GC protocol. TinyGarble takes a sequential circuit description of the function that needs to be evaluated securely. We formulate a sequential circuit for performing the HLA data compatibility test and show that it can be efficiently implemented using Verilog.

### 1.3 Privacy-Preserving Near-Neighbor Search

Near-neighbor search is one the most fundamental and important tasks in many different applications. In this problem, given a  $\mathbb{D}$  dimensional query  $q$  and a database  $F$  containing  $n$  different  $\mathbb{D}$  dimensional points, we are interested to find any “near” points in  $F$  to the  $q$ . The term near can be defined as several metrics such as euclidean distance, Jaccard similarity (resemblance), cosine similarity and etc. However, in many crucial applications, it is critical that the  $q$  is not revealed to the server and also the points in the database do not get revealed to the query holder. This problem is called privacy-preserving near-neighbor search (PP-NNS). PP-NNS is necessary for privacy-preserving face recognition [29], secure biometric authentication [30, 31], privacy-preserving speech recognition [32] and private recommender systems [33].

The focus of this section will be on “efficient” privacy-preserving near-neighbor search “scalable” to modern big-data. With the increased emphasis of algorithms on strong privacy of the user information, this simple task of near-neighbor search becomes less straightforward. Previous arts have broken down the PP-NNS problem

into two sub-problems: first, private calculation of pairwise distances between query and all points in the dataset and second, finding the minimum of these distances securely.

Privacy-preserving near-neighbor search is a very well studied topic due to its significance and impact. There are different methodologies addressing this critical problem in a variety of diverse settings. Primarily, methodologies for PP-NNS can be classified into three broad categories: 1) Computational privacy, 2) Information-theoretic privacy, and 3) Randomized embeddings.

First group uses various cryptographic primitives to securely compute the pairwise distances by performing the computation on encrypted version of data. The security of this approach just like cryptographic tools, is based on the hardness of some problems in number theory (e.g. factorization of large numbers) which makes it intractable for an adversary with limited computational resource. Since every single bit in the computation is encrypted, these solutions are impractical and unscalable to modern big-data. The second group, unlike the first group, is information-theoretic secure meaning that no adversary even with unlimited computational power can comprise the data. This approach is based on sharing some secret information to perform the secure computation. Securely computing pairwise distances is a relatively easy task for these schemes. However, comparing distances for finding the minimum requires comparison which cannot be performed using secret-sharing alone and needs a number of cryptographic blocks [34, 35] which then limits the overall scalability.

The third group relies on randomized embeddings. Embedding is a transformation that maps the data into another space (usually lower-dimensional) while preserves the distance between any pair of points, with high probability. The new representation does not reveal direct information about the original attributes [36]. These method-

ologies typically are based on Jonson-Lindenstrauss [37] or Locality Sensitive Hashing (LSH). We argue that the securely finding all pairwise distances is sufficient but not necessary for the task of near-neighbor search. In fact, if we are only interested in the task of near-neighbor search, being able to estimate distance between *all* pair of points (even points which are not close) compromises privacy and are susceptible to “triangulation” attack (see Section 5.1 for details).

In this thesis we address all shortcomings of previous works by introducing new embedding. Our proposal is a novel transformed embedding on the top of LSH embedding, we call it as Secure-Locality Sensitive Hashing (S-LSH), where only distances between close enough points are preserved. Our embedding further ensures that non-neighbor distances are indistinguishable from a random point. Therefore, it is impossible to estimate their distances if they are not among the neighbors. Thus, we avoid the unnecessary leak in information, making our proposal secure against triangulation attacks. This added security comes at no price and our proposal retains all the nice guarantees for sub-linear near-neighbor search.

Note that there has been successful progress of differential privacy [38]. The idea behind differential privacy is to perturb the distance estimates (or the vectors themselves) using smartly tailored noise to trade-off privacy with utility. But the security model of differential privacy is different from us. They assume a trusted server that holds all sensitive data and the goal is to provide statistical information while preserving the privacy of users’ data in the database. This is in contrast with our security model where we do not trust the server or any party engaging in the computation.

## 1.4 Contribution

The explicit contributions of this thesis are as follows:

### 1.4.1 P3SM

- We introduce the first feasible, scalable, and efficient secure SM for real-world set sizes.
- We design the first sequential circuit for the SM algorithm for Yao’s GC protocol. The compactness achieved by the sequential description (as opposed to prior combinational one) reduces the circuit size from  $\mathcal{O}(n^4 \log n)$  to  $\mathcal{O}(n^2 \log n)$ .
- We design the first sub-linear size circuit (w.r.t. the SM set size). This circuit achieves unprecedented computation and communication efficiency compared to the prior art by integrating sub-linear ORAM and various memory access strategies.
- We introduce mathematical and statistical methodologies for early protocol termination which allows us to trade-off between security and efficiency.
- We demonstrate a proof-of-concept implementation of our approach for different secure SM settings with various set sizes. We benchmark the state-of-the-art sub-linear ORAM schemes and report the best solution for different range of set size.

### 1.4.2 GenMatch

- Introduction of the first efficient, practical, and scalable methodology for secure organ and tissue transplantation compatibility test. Our method leverages

hardware synthesis techniques to formulate the genome matching algorithm for the GC protocol. The low memory footprint of our method allows the first implementation of provably secure matching on embedded devices.

- Design of the first sequential circuit for the purpose of organ compatibility testing. The circuit performs a comparison with one entity in the database at each sequential clock cycle and incurs a circuit size of logarithmic complexity which can scale well for big database sizes.
- Creation of special computational blocks customized for the GC-based DNA matching applications. We also suggest a new method for a cumulative addition where the output bit-length increases with each stage.
- Proof-of-concept implementation of privacy-preserving organ and tissue transplantation compatibility test and demonstrating the scalability of our work by performing a compatibility test on a database in the order of million user profiles.

### 1.4.3 S-LSH

- We propose Secure-Locality Sensitive Hashing (S-LSH), the first of its kind, a generic transformation which makes any given locality sensitive hashing scheme secure by preventing the leakage of information completely unnecessary for the task of near-neighbor search. This advantage comes at no additional cost and we retain all the advantages of locality sensitive hashing required for sub-linear search.
- Our proposed protocol is simple to implement, massively parallelizable and leads to provably sub-linear search algorithm for privacy-preserving near-neighbors

search, making it ideal for massive datasets.

- Experimental evaluations on largest MovieLens dataset supports our theoretical claims and clearly demonstrates the practicality of our approach.
- Our framework is flexible and can incorporate different distance thresholds and accuracy tolerance.
- We provide the formal analysis of our approach and show information theoretic bounds.

## Chapter 2

### Preliminaries and Background

In this chapter, we bring some background information that are necessary for better understanding the terminologies used throughout this thesis. In Section 2.1, we explain what is Secure Function Evaluation (SFE). In the follow up section, we describe one of the popular protocols in SFE, called Garbled Circuit (GC) protocol. In Section 2.3, we describe Stable Matching (SM) algorithm and its variants. The basic information about human genome and DNA are covered in Section 2.4. Last section, defines approximate near neighbor search and Locality Sensitive Hashing (LSH).

#### 2.1 Secure Function Evaluation

SFE allows to evaluate a function on private inputs from multiple parties where each party wants to keep her own inputs private. More formally, suppose given number of participants,  $p_1, p_2, \dots, p_N$ , each having a private data,  $x_1, x_2, \dots, x_N$ , respectively. They are interested to find the value of publicly known function  $f(\cdot)$  on their inputs,  $f(x_1, x_2, \dots, x_N)$ , while no information is revealed to other parties other than what can be inferred from output. In fact,  $p_i$  learns nothing about  $\{x_j \mid j = 1 \dots N, j \neq i\}$ .

Perhaps the most famous example is Yao's Millionaire problem introduced in 1985 by Andrew Yao. In this problem, two millionaires want to know which one has more wealth while not telling anyone their actual wealth. The general version of this problem is solving inequality without revealing the actual values. The Millionaire's



problem is one of the important problems in cryptography where its solution is used in e-commerce and data mining.

We can categorize different SFE protocols into two inherently different approaches: (i) computational security and (ii) information-theoretic security. Figure 2.1 shows different protocols for SFE. The first group uses various cryptographic primitives and preserves the privacy of inputs by performing the computation on encrypted version of data. The security of this approach just like cryptographic tools is based on the hardness of some problems in number theory (e.g. factorization of large numbers) which makes it intractable for an adversary with a limited computational resource. Since every single bit in the computation is encrypted, these solutions are impractical and unscalable to modern big-data. The second group, unlike the first group, is information-theoretic secure meaning that no adversary even with unlimited computational power can compromise the data. This approach is based on sharing some secret information to perform the secure computation. However, secret-sharing by itself is not capable of solving Millionaire’s problem and therefore some cryptographic protocols are needed if the function in SFE protocol needs a secure comparison of two numbers.

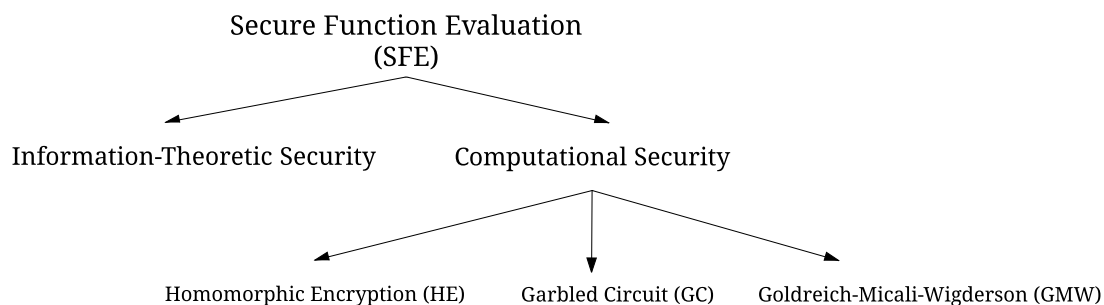


Figure 2.1 : Different SFE protocols.

The three most practical protocols in computational security are Homomorphic Encryption (HE), Garbled Circuit (GC), and Goldreich-Micali-Wigderson (GMW). HE is mostly based on computationally expensive public key encryption. On the contrary, GC mainly uses symmetric key encryption which is far less expensive and this makes GC protocol to be more promising and has been used more in real-world applications. GC protocols takes as input a Boolean circuit description of a function that needs to be evaluate securely. GMW, similar to GC needs Boolean description of the function. GMW is more efficient than GC protocol when the depth of the circuit is relatively small. Today, the common drawback of SFE protocols is their low efficiency, mainly due to the communication. Therefore, for SFE protocols to be widely adopted, it is crucial to devise methodologies that increase the efficiency.

## 2.2 Garbled Circuit Protocol

Yao's GC protocol [17] is one of the most promising solutions for two-party SFE. In this protocol, two parties (Garbler and Evaluator) jointly evaluate a function on their inputs while keeping their own data private. The function is represented as a Boolean circuit. The GC protocol consists of three algorithms: circuit garbling which is done only by the Garbler, data exchange which involves both parties, and evaluation which is done only by the Evaluator. First, Garbler assigns two random keys to each Boolean value in the circuit and then encrypts the truth table of each gate using the keys. Garbler sends the encrypted tables to the Evaluator together with the keys corresponding to her inputs. Evaluator revives the keys corresponding to his inputs from the Garbler through 1-out-of-2 Oblivious Transfer (OT) protocol, without letting her to know his inputs. Then, Evaluator decrypts the tables, one by one, using the received keys until he reaches the output keys. Finally, Garbler reveals

the mapping of the keys to the semantic values to Evaluator in order to achieve the final result in plain-text.

### 2.2.1 Optimizations

We benefit from the state-of-the-art optimizations for the GC protocol. We utilize Free-XOR technique [39] which makes the cost of garbling an XOR gate virtually zero. Therefore, the cost of GC protocol can be measured only in terms of the number of AND gates in the circuit. We also use garbling with a fixed-key block-cipher [40] together with the half gates technique [41] for efficient evaluation of AND gates. For OT required in the initial data exchange of the GC protocol, we use the OT Extension method [42, 43]. We use TinyGarble [18], an automated framework for generating optimized Boolean circuits for the GC which is based on logic synthesis tools. It optimizes the generation of a Boolean circuit for the GC protocol by customizing the flow of the logic synthesis tool. It uses a customized technology library consisting of logical descriptions of basic gates. The library also includes the corresponding parameters like timing and area. In the case for the GC, the timing parameter is not needed as the GC depends only on the size of the circuit. The area parameter of an XOR gate is set to 0 and all other two input gates to 1. TinyGarble also proposes to use sequential circuits that are more compact than the conventional combinational circuits. However, sequential circuit has to be evaluated for multiple *iterations*. Sequential circuit stores the state of the computation in memory elements (registers). Unlike combinational circuit, the output of the sequential circuit depends both on the input to the circuit and the value of the registers. Sequential description reduces the memory footprint of the GC protocol.

### 2.2.2 Oblivious RAM

Goldreich and Ostrovsky [44] proposed a two-party mechanism that lets a client store her data on a remote server while hiding her data and access pattern from the server. They also showed that the lower bound for the cost of accessing a single entry in the memory is sub-linear with respect to the size of the memory. A naïve implementation of ORAM linearly scans the entire memory for each access such that the client can choose the desired entry using multiplexer (MUX) which is called Linear ORAM. There are several improvements on the original idea of ORAM including [45, 46, 47] which reduced the amortized per-access complexity to  $\mathcal{O}(\log^3 n)$ . Gordon et al. [48] proposed to use ORAM mechanism inside two-party SFE (e.g., GC) in order to reduce the amortized cost of accessing a memory entry from linear to sub-linear. So far, the best asymptotic complexity for ORAM inside SFE is Circuit ORAM proposed by Wang et al. [49]. A recent paper [50] presents Square-Root ORAM, an alternative ORAM structure that has lower initialization cost than Circuit ORAM. Despite the fact that it has higher asymptotic complexity, it outperforms Circuit ORAM for medium-sized memory.

### 2.2.3 Adversary Model

The GC protocol, on which we base our implementation, is secure against honest-but-curious (also known as semi-honest or passive) adversaries which means that all parties should follow the protocol but they may be curious to extract additional data from the data which they are receiving. Although this security model is not the strongest attack model, it is a first step towards being secure and it is commonly used in the literature. Also there are several reasons for having this security model:

- According to [51] there are some scenarios in which this security model is acceptable: when parties are reasonably trusted like hospitals and companies but they need to obscure their private information for legal reasons or to prevent future break-ins. For example when the parties are government agencies, they can assure each other that they will follow the protocol.
- When only one party, evaluator, will receive the results and the other party or parties will not get any result including the successful completion of protocol. In this case if we use the oblivious transfer (OT) that is secure against malicious adversaries we would achieve full security. Notably, no one learns anything about other parties's inputs except what the evaluator can conclude by knowing the output.
- Since we design Boolean circuits that are evaluated with Yao's GC protocol, the very same circuits can be evaluated with (less efficient) protocols that provide security against stronger active/malicious adversaries, e.g., [52, 53, 54, 55].
- Many useful privacy-preserving applications inherently have the properties that let them fit well into our protocol. For instance, when all parties have an incentive to generate flawless outcomes like financial fraud detection when banks pull together all of the data to detect corrupt accounts or personalized medicine when a patient and drug company collaborate to find the best medicine, they are basically interested to reach the correct result and hence they will adhere to the protocol steps.

However, revealing the output itself could inevitably leak some information. A simple example is a circuit that evaluates one AND gate to which each party provides an input. When one party knows that its own input is  $x = 1$  and after finishing the

protocol sees that the output is  $x \wedge y = 1$  it can infer that the other party's input has been  $y = 1$  as well. This information leakage is not a flaw of the protocol, but an inevitable property of the function that is being computed securely. Knowing how much private information is leaked by the output itself is a crucial and challenging problem, but out of the scope of this work.

### 2.3 Stable Matching

In this section, we illustrate the detailed description of the SM problem. A number of researchers have focused on addressing the SM problem, but Gale and Shapley [56] were the first to formalize the SM algorithm. They centered their work on the special case of the marriage problem. In this case, there is a set of men and a set of women. They introduced an algorithm which resulted in stable marriage. Gale and Shapley also proved that the stable match always exists. However, they showed that there can be more than one stable assignment and so the stable matching is not a unique assignment. Roth [11] demonstrated that there is always a stable match preferred by men (male-optimal) and there is always a stable match preferred by women (female-optimal). The algorithm which Gale and Shapley proposed consists of a number of rounds in which the men propose and the women review these proposals. This algorithm always produces a match which is preferred by men and thus is male-optimal. (For more detail, see [9].)

In a general SM problem we have two groups of individuals that can represent various types of entities. Thus, for simplicity, we call these two groups women and men. We denote the size of the group of women by  $|W|$  and that of men by  $|M|$ . In general the cardinality of these two sets can be different. We assign an ID from 0 to  $|W| - 1$  to each woman and 0 to  $|M| - 1$  to each man; ID is unique among each group.

Each woman and man ranks the members of the other group as she/he prefers which we call vector of preferences. Each woman can rank up to  $K_w$  men and each man can rank up to  $K_m$  women. Generally,  $K_w$  and  $K_m$  can be different from  $|M|$  and  $|W|$  respectively. For simplicity we will show the complexity of our protocol in terms of  $n$  throughout this thesis, where  $n = |M| = |W|$ . The preference vectors altogether, will form a preference matrix. Thus we have one preference matrix for the women's group and one for men's in which each row represents the preference vector of each member of that group, see Figure 2.2 for an example: If woman #1 ranks the men as  $[1, 2, 0]$ , this means that she prefers man #1 over man #2 and so on.

Woman ID	Priority Higher $\rightarrow$ Lower			Man ID	Priority Higher $\rightarrow$ Lower		
0	1	0	2	0	2	0	1
1	1	2	0	1	2	1	0
2	0	1	2	2	0	1	2

(a) Women
(b) Men

Figure 2.2 : Example of preference matrices.

Therefore the preference matrix is of size  $|W| \times K_w$  for women and  $|M| \times K_m$  for men. These two matrices are the only input to the SM algorithm. After the algorithm is finished, the result is a stable match, meaning that there are no two individuals that they both prefer to be matched to each other but are not already assigned.

Figure 2.3 shows one stable match and one unstable match. The match in Figure 2.3a is stable because it satisfies the above definition and the match in Figure 2.3b is unstable because man #0 prefers woman #2 over woman #1 and also woman #2 prefers man #0 over man #1 and this violates the definition of SM.

Woman ID	Man ID
0	2
1	1
2	0

(a) Stable

Woman ID	Man ID
0	2
1	0
2	1

(b) Unstable

Figure 2.3 : Example of stable and unstable match.

A stable match is called optimal for men or women if every member of that group is matched to the best person he/she prefers that could have been matched in any other stable matches. Roth [11] showed that this specific version of SM can always be found for one of the sets but not for both.

### 2.3.1 General Stable Matching

The first algorithm relates to the case when each individual ranks all of the members from the other group. More specifically,  $K_m = |W|$  and  $K_w = |M|$ . There is a list of size  $|W|$  which holds the up-to-now assigned partner to each woman. This temporary assignment will become finalized when the algorithm terminates. We have the notion of free and engaged persons during the execution of the algorithm. Engaged persons are those who have a partner up to that round and free persons are those who do not. Each man could become engaged and after that free and again engaged and so on. But as soon as a woman gets engaged, she remains engaged until the end of the matching process. The algorithm consists of  $R$  proposals. In each proposal, a free man proposes to the woman he prefers the most and he has not proposed to yet. Then if that woman who is proposed to is free, they become engaged. If she is engaged, she looks at her preference list and sees whether she prefers the proposing man over her



already assigned partner or not; if yes, she becomes engaged to the new man and the previous man becomes free and if she does not prefer, she rejects the proposal and the man remains free. The algorithm runs until all men are engaged. Algorithm 1 shows the pseudo-code of the limited SM algorithm where  $w \longleftrightarrow m$  denotes assigning man  $m$  to woman  $w$  and  $pc$  is the proposal counter for men.

---

**Algorithm 1** General Stable Matching

---

```

while |free men|  $\neq$  0 do
     $m \leftarrow$  choose from free men
     $w =$  most preferred woman that  $m$  has not yet proposed to
    if  $w$  is free then
         $w \longleftrightarrow m$ 
    else
         $m' \leftarrow$  man who is currently engaged to  $w$ 
        if  $w$  prefers  $m$  over  $m'$  then
             $w \longleftrightarrow m$ 
             $m'$  gets free
        end if
    end if
end while

```

---

Gale and Shapley showed that this algorithm takes at most  $R_{max} = n^2 - n + 1$  proposals ( $R \leq R_{max}$ ), where  $n$  is the size of each participating group. As proven by Gale and Shapley, a stable match always exists and the algorithm terminates with the male-optimal stable match.

### 2.3.2 Limited Stable Matching

In the following, we describe a variant of the SM algorithm where each person ranks only a subset of the other group. More specifically,  $K_m < |W|$  and  $K_w < |M|$ . This version is called limited SM. In this case, each person ranks up to a certain number, meaning that he/she prefers to be unmatched rather than being assigned to a person who is not in the list. Algorithm 2 is the pseudo-code for limited SM where the only difference to Algorithm 1 is that there is a limit on the total number of listed preferences.

---

#### Algorithm 2 Limited Stable Matching

---

```

initialize  $pc[i] = K$  for  $i \in \{0, |M| - 1\}$ 
while  $\exists \text{man} : pc[\text{man}] > 0 \wedge \text{man is free}$  do
     $m \leftarrow$  choose that free man
     $w =$  most preferred woman that  $m$  has not yet proposed to
     $pc[m] \leftarrow pc[m] - 1$ 
    if  $w$  is free then
         $w \longleftrightarrow m$ 
    else
         $m' \leftarrow$  man who is currently engaged to  $w$ 
        if  $w$  prefers  $m$  over  $m'$  then
             $w \longleftrightarrow m$ 
             $m'$  gets free
        end if
    end if
end while

```

---

The number of proposals in this case is not flexible and in fact is fixed to  $R = K_m \cdot |M|$ . Although this algorithm limits the number of choices, it takes a fixed number of proposals which is far smaller than that of the general version:  $R_{\text{general}} \in O(n^2)$  whereas  $R_{\text{limited}} = K_m \cdot |M| \in O(n)$  for constant  $K_m$ .

## 2.4 Human Genome and DNA

A human genome is the complete set of genetic and biological information of a person. It consists of about 3 billion nucleotides of types guanine (G), adenine (A), thymine (T), or cytosine (C). Several active research projects aim at analyzing the genome data to extract specific information or finding a correlation between genome patterns and physical/mental traits.

We focus on a specific part of DNA called the Human Leukocyte Antigen (HLA). HLA genes are the human equivalents of Major Histocompatibility Complex (MHC) genes found in most vertebrates. HLA genes encode proteins that are responsible for regulation of the immune system in humans. In other words, HLA is responsible for determining whether a tissue is native or foreign. HLA is present in chromosome 6 of the genome. There are several classes of this part of the DNA and they have different functions.

The HLA genes that are inherited by an individual on a single chromosome constitute a “haplotype” [57]. Each person receives a pair of HLA genes, one from each of the parents. An example of HLA data of a person is shown in Figure 2.4.

HLA mismatch in case of organ transplant is a major cause of transplant rejections and graft-versus-host-disease. Therefore, it is important to match HLA data before transplantation. It should also be noted that HLA data can reveal information about genetic ancestry and predisposition of the person towards certain autoimmune

Paternal inheritance	Maternal inheritance
HLA-A*01:01	HLA-A*02:01
HLA-B*07:02	HLA-B*08:01
HLA-C*07:01	HLA-C*16:01
HLA-DQA*02:01	HLA-DQA*05:01
HLA-DQB*02:01	HLA-DQB*02:01
HLA-DRB*03:01	HLA-DRB*07:01

Figure 2.4 : Sample HLA template for an individual.

diseases. Apart from this, HLA can also be used for genetic compatibility testing. It was observed that married couples were less likely to share HLA alleles [58].

## 2.5 Approximate Near-Neighbor Search and Locality Sensitive Hashing (LSH)

A popular technique for approximate near-neighbor search, uses the underlying theory of *Locality Sensitive Hashing* (LSH) [59]. LSH is a family of functions, with the property that similar input objects in the domain of these functions have a higher probability of colliding in the range space than non-similar ones. In formal terms, consider  $\mathcal{H}$  a family of hash functions mapping  $\mathbb{R}^D$  to some set  $\mathcal{S}$ .

*Definition 1 Locality Sensitive Hashing (LSH) Family* A family  $\mathcal{H}$  is called  $(S_0, cS_0, p_1, p_2)$ -sensitive if for any two point  $x, y \in \mathbb{R}^D$  and  $h$  chosen uniformly from  $\mathcal{H}$  satisfies the following:

- if  $Sim(x, y) \geq S_0$  then  $Pr_{\mathcal{H}}(h(x) = h(y)) \geq p_1$
- if  $Sim(x, y) \leq cS_0$  then  $Pr_{\mathcal{H}}(h(x) = h(y)) \leq p_2$

For approximate nearest neighbor search typically,  $p_1 > p_2$  and  $c < 1$  is needed. An LSH allows us to construct data structures that give provably efficient query time algorithms for approximate near-neighbor problem.

### 2.5.1 Popular LSH 1: Minwise Hashing and Resemblance Similarity

One of the most popular measures of similarity between web documents is **resemblance (or Jaccard similarity)**  $\mathcal{R}$  [60]. This similarity measure is only defined over sets which can be equivalently thought of as binary vectors over the universe, with non-zeros indicating the elements belonging to the given set. The resemblance similarity between two given sets  $x, y \subseteq \Omega = \{1, 2, \dots, D\}$  is defined as

$$\mathcal{R} = \frac{|x \cap y|}{|x \cup y|} = \frac{a}{f_1 + f_2 - a}, \quad (2.1)$$

where  $f_1 = |x|$ ,  $f_2 = |y|$ , and  $a = |x \cap y|$ .

Minwise hashing [61] is the LSH for resemblance similarity. The minwise hashing family applies a random permutation  $\pi : \Omega \rightarrow \Omega$ , on the given set  $x$ , and stores only the minimum value after the permutation mapping. Formally MinHash is defined as:

$$h_{\pi}^{min}(x) = \min(\pi(x)). \quad (2.2)$$

Given sets  $x$  and  $y$ , it can be shown by elementary probability argument that

$$Pr_{\pi}(h_{\pi}^{min}(x) = h_{\pi}^{min}(y)) = \frac{|x \cap y|}{|x \cup y|} = \mathcal{R}. \quad (2.3)$$

### 2.5.2 Popular LSH 2: Signed Random Projections (SimHash) and Cosine Similarity

SimHash is another popular LSH for the cosine similarity measure, which originates from the concept of **Signed Random Projections (SRP)** [62, 63, 64]. Given a

vector  $x$ , SRP utilizes a random  $w$  vector with each component generated from i.i.d. normal, i.e.,  $w_i \sim N(0, 1)$ , and only stores the sign of the projection. Formally SimHash is given by

$$h_w^{sign}(x) = \text{sign}(w^T x). \quad (2.4)$$

It was shown in the seminal work [65] that collision under SRP satisfies the following equation:

$$Pr_w(h_w^{sign}(x) = h_w^{sign}(y)) = 1 - \frac{\theta}{\pi}, \quad (2.5)$$

where  $\theta = \cos^{-1}\left(\frac{x^T y}{\|x\|_2 \cdot \|y\|_2}\right)$ . The term  $\frac{x^T y}{\|x\|_2 \cdot \|y\|_2}$ , is the cosine similarity. Since  $1 - \frac{\theta}{\pi}$  is monotonic with respect to cosine similarity  $\mathcal{S}$ , it is a valid LSH.

### 2.5.3 Mapping LSH to 1-bit

LSH, such that Minhash, in general, generates an integer value, which is expensive from the storage perspective. Having a single bit hashing schemes, or binary locality sensitive bits, has many advantages. It is also not difficult to obtain 1-bit LSH. The idea is to apply random universal hash function to the LSH and map it to 1 bit.

As an example, the 1-bit MinHash is computed by applying universal hash function to MinHash, to rehash it to 1-bit, given by the equation

$$h_\pi^{min,1bit}(x) = h_{univ}(h_\pi^{min}(x)), \quad (2.6)$$

where  $h_{univ}$  is any universal hashing scheme [66] randomly mapping the input to 0 or 1. It can be easily shown that the collision probability under this 1-bit scheme is given by

$$Pr_\pi(h_\pi^{min,1bit}(x) = h_\pi^{min,1bit}(y)) = \frac{\mathcal{R} + 1}{2}, \quad (2.7)$$

Another convenient (and efficient) 1-bit rehashing is to simply use the parity, or the most significant bit, of  $h_\pi^{min}(x)$  as 1-bit hash [67].

## Chapter 3

# P3SM: Practical Privacy-Preserving Stable Matching

### 3.1 High Level Architecture

We implement a secure stable matching system based on Yao's GC protocol [17] which is an efficient method for secure function evaluation between two parties. Secure SM is inherently a multiparty SFE problem where multiple parties provide their inputs. However, we use a known technique based on XOR-secret-sharing that translates this problem into two-party SFE.

In the GC protocol, the underlying function has to be described as a Boolean circuit. The input of each party is an input to this circuit. Then two parties, called Garbler and Evaluator, run the GC protocol and find the results. Implementing the entire SM as an acyclic Boolean circuit (combinational) is neither practicable nor scalable and hence no one has tried to implement it so far. In our work, we design a compact sequential circuit which enables an efficient implementation of the SM algorithm.

Here, we explain how two proxies (the Garbler and Evaluator) are able to securely perform the SM for multiple parties using the GC protocol (see [68] for details). For each input bit  $I$ , each party does the following:

1. Generate a random mask  $I_A$ .
2. Compute  $I_B = I_A \oplus I$ .

3. Send  $I_A$  to Garbler and  $I_B$  to Evaluator.

The SM circuit is extended by one layer of XOR gates for each input. Each XOR gate receives  $I_A$  and  $I_B$  as inputs, provided by Garbler and Evaluator respectively, and computes  $I = I_A \oplus I_B$  as output. The outputs of these gates are the inputs to the main SM circuit.

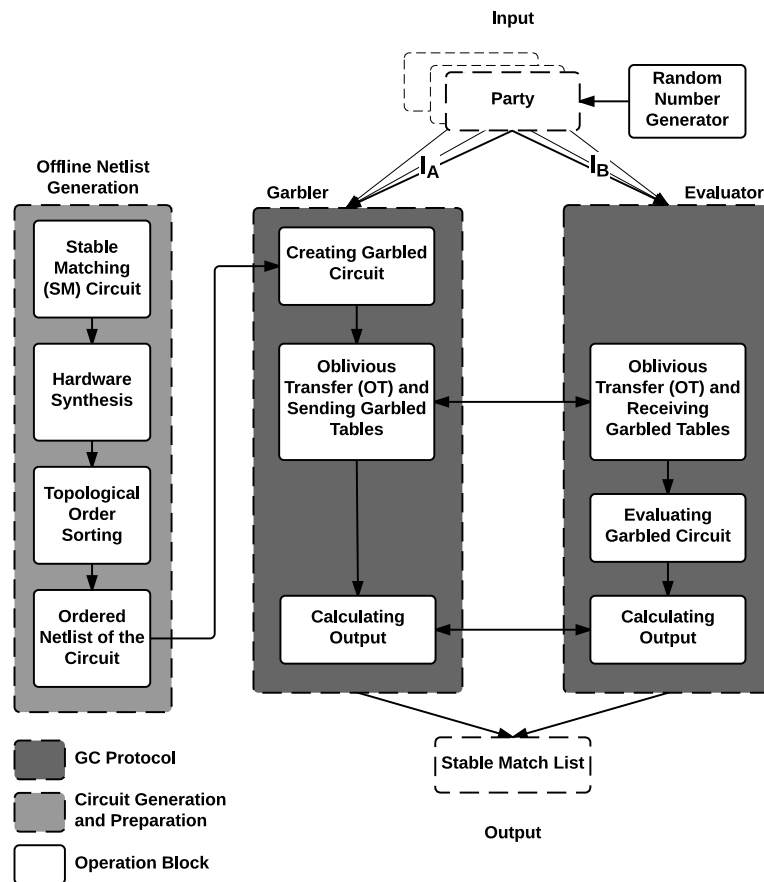


Figure 3.1 : Global flow of our secure SM system.

Figure 3.1 shows the global flow of our system which consists of two parts. First, the offline pre-processing and then the online execution which performs the GC protocol. In the first part we synthesize the circuit, generate the netlist and sort the gates



topologically. This ordered netlist is given to the Garbler to generate the garbled circuit for the online phase. In the online phase, the constant-round GC protocol is executed: the Evaluator gets all information that she needs to evaluate the garbled circuit. At the end, both Garbler and Evaluator share the results to find the outputs and deliver them to all parties.

The SM algorithm is considered to be secure if it outputs a stable match without revealing any additional information about preference list except what can be inferred from the output. Our scheme is secure against honest-but-curious adversaries. Our system realizes secure SM as it takes the encrypted inputs from each party and securely computes a stable match. Our methodology is secure as it relies on the security of the GC protocol which has been proven to be secure against honest-but-curious adversary in [69]; getting inputs from each party with the XOR-sharing technique is secure because Garbler and Evaluator see only random numbers from which they cannot infer any information [68]. The output is a correct stable match since our circuit for SM implements the Gale and Shapley algorithm that was proven to correctly compute a stable match [56].

## 3.2 Circuits for Stable Matching

As explained in Section 2.2, the GC protocol for the two-party SFE takes a Boolean circuit as a function description. In this section, we describe how we generate such circuits for SM. We describe a combinational circuit for SM in Section 3.2.1. Afterwards we introduce the first sequential circuit ever implemented for SM in Section 3.2.2. In Section 3.2.3, we analyze the circuit runtime. Section 3.2.4 describes a novel sub-linear size circuit for SM that we have designed. Finally, in Section 3.2.5 we analyze the memory usage and ORAMs' access cost complexities.

### 3.2.1 Combinational Circuit

To the best of our knowledge, there exists no implementation of the combinational circuit that gets the men's and women's preference matrices as an input and outputs the stable match. The reason might be that it is relatively complicated to design such a large circuit without proper tool support. In this section, we calculate a lower bound for the size of this circuit. Considering  $n$  to be the number of the pairs in SM, we have preference matrices of size  $\mathcal{O}(n^2 \log n)$  bits, because each of the  $n$  individuals ranks the  $n$  members of the other group and each entry needs to be represented by  $\log n$  bits. Accessing an entry, without using a sub-linear ORAM, requires a multiplexer which needs  $\mathcal{O}(n^2 \log n)$  AND gates. Comparing two entities of length  $\log n$  requires  $\log n$  AND gates. We have to compare  $n$  entities for each man, yielding total  $\mathcal{O}(n^2 \log n)$  AND gates. This is just for accessing and comparing two preferences. To implement the algorithm, it is extraordinarily hard to design a generic circuit. One solution might be to design a circuit that has  $R$  (the number of proposals) layers and each layer processes one proposal made by a man until we reach the stable match. Knowing that in the worst case, we need  $R_{max} = n^2 - n + 1$  proposals, this accounts for the number of layers of the circuit. Another solution might be for each man to find his match instantly. This requires that for each of the  $n$  women on his list, we find whether she accepts the proposal by that man or not and this also depends on the preference list of other men. So in this way, we need  $n$  circuits of size  $\mathcal{O}(n)$  (for each woman) times  $n$  (for each man) to compare the preferences. This results in  $\mathcal{O}(n^3)$  comparisons on  $\mathcal{O}(n \log n)$  values. In both cases, the lower bound of the circuit size is  $\mathcal{O}(n^4 \log n)$  which is a huge circuit size even for relatively small values of  $n$ . Therefore, the real implementation is not feasible for real-world applications, since there is no automated tool to generate and synthesis circuit of such size.

### 3.2.2 Sequential Circuit

Describing the SM circuit in a sequential way enables us to implement the circuit in a highly compact format. This compact format allows us to synthesize the circuit for larger group size and meet the size of real-world applications. This compactness also provides us with the ability to store the circuit on the platforms with lower computational and memory capabilities.

In the rest of this section, we describe our sequential circuit for SM. Figure 3.2 shows the block diagram representation of the circuit. The inputs to the circuit are the preference matrices of men and women. The circuit consists of the following modules: Algorithm Computation Circuit (ACC), vectorized Preference Compare Circuit (PCC), and Man Selection Circuit (MSC), and Memory (registers for storing the state values).

Here, we explain how the entire circuit works. We design the sequential circuit such that it processes one proposal in one iteration of the sequential circuit. MSC finds a free man. In Section 3.2.3, we will explain how MSC and Finish Signal work. Once a free man is selected, he should propose to the most preferred woman that he has not already proposed to (see Algorithm 1). Therefore, a counter storing the number of the proposals is required for each man; we call it Proposal Counter (PC). In order to access the desired women ID, a MUX on the man's preference list is required. This needs  $\mathcal{O}(n^2 \log n)$  AND gates.

ACC implements the main part of Algorithm 1 as a Boolean circuit. ACC processes the proposal: if a woman is free, then the man and she will be engaged and the status of the assignment will be updated in the memory. If she is not free, ACC needs to check whether or not she prefers the man over her current match. Therefore, we need to access the preference vector of the woman using MUX and deliver it to

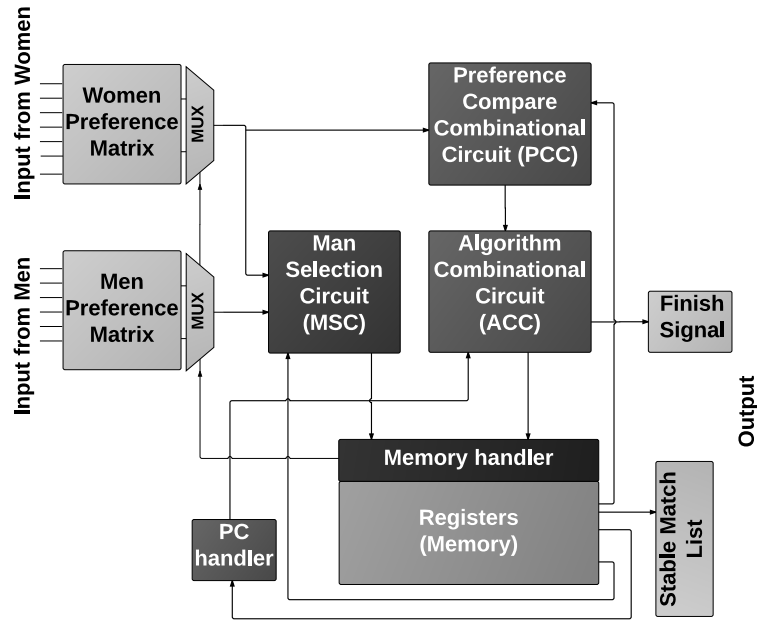


Figure 3.2 : Block diagram architecture of the sequential circuit for implementing secure SM.

PCC together with the values for the man ID and the women's current match ID. PCC linearly scans the preference vector and checks which one of the two men is more preferable (which one comes first in the preference list). It is important that PCC does this task in one iteration, otherwise the entire circuit will be unnecessarily evaluated and the efficiency will be reduced. As a result, one proposal can be done in *one* iteration of the sequential circuit. In Section 3.2.3, we will explain how many iterations the circuit should be evaluated.

Figure 3.3 illustrates the inside of the PCC. This circuit is designed to minimize the number of AND gates. It uses two linear layers of XOR gates and three linear layers of AND gates which at the end are OR-ed and produce the output. The final logical OR gate is implemented by  $n - 1$  two-input AND gates and  $3n - 3$  XOR gates.

Thus, PCC uses  $4n - 1$  AND gates. In summary, the total number of AND gates in the circuit is  $\mathcal{O}(n^2 \log n)$  dominated by MUXs for accessing the preference list.

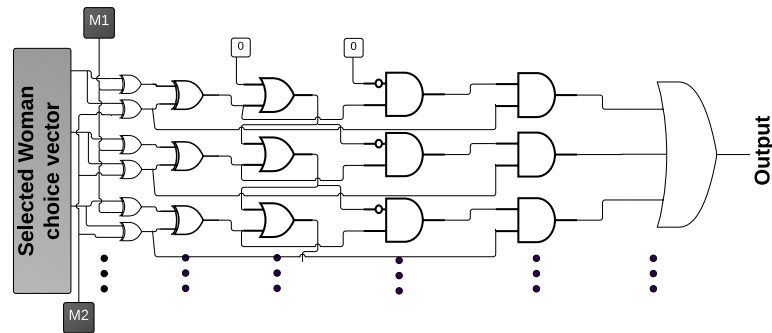


Figure 3.3 : Preference Compare Circuit (PCC).

### 3.2.3 Number of Proposals and Circuit Runtime

Gale and Shapley have proved that after at most  $R_{max} = n^2 - n + 1$  proposals in Algorithm 1, the algorithm will output a stable match. However in the GC, the intermediate status of each man's engagement is encrypted and cannot be accessed in plain-text. Therefore, the process of finding the next free man becomes challenging. One naïve implementation can be to iterate over all men and let them to propose regardless of the engagement status. But if the man was not free, the proposal should be invalidated. This approach is dramatically inefficient. The reason is as follows: since all the information in the circuit is encrypted, there is no way to distinguish a valid proposal from an invalid one. Thus, we have to evaluate the circuit regardless of the validity of the proposal. As a result, we need to run the circuit even for more iterations than the mathematical worst case of the number of proposals ( $R_{max}$ ). This problem arises both for general SM and limited SM. To overcome this problem, we

have designed Man Selection Circuit (MSC). This circuit wisely chooses the next free man ahead of time to avoid invalid proposals (unnecessary execution of the circuit).

MSC makes the number of iterations that the circuit should be evaluated *exactly equal* to the number of required proposals. In the worst case, this number is equal to  $R_{max}$ . However, based on our extensive statistical analysis, we have observed that the average number of required proposals is linear in terms of set size  $n$ . It can be seen from Figure 3.4 that the average and statistical maximum of number of the proposals are linear with respect to the set size. More precisely, the average number is  $\mathcal{O}(n)$  while the worst case number is  $\mathcal{O}(n^2)$ . The error bars show the minimum and maximum number of proposals for each set size. Hence, we can take advantage of this linear average by two different approaches. The first approach is to run the circuit for a constant number of times. This number can be statistically guaranteed instead of mathematically guaranteed and it means that we can run the circuit for some fixed statistically driven number like  $m + \alpha \times \sigma$ , where  $m$  is the average number of proposals,  $\sigma$  is the standard deviation, and  $\alpha$  is a constant to be determined by the probability of proper termination.  $\alpha$  is fixed prior to running the protocol and is known by both Garbler and Evaluator. The second approach is to produce the Finish Signal by ACC and stop the algorithm as soon as the stable match is reached. We call this approach Early Termination Technique (ETT). ACC produces the Finish Signal by checking if all the men (or women) are engaged. The cost of producing this signal is  $n - 1$  AND gates because it is the result of the logical AND over all the men's status; if all of them are matched, then the Finish Signal is true.

In order to verify the termination condition in the middle of the GC protocol, the parties have to reveal the shared secret associated only to the Finish Signal. Revealing this secret in an iteration will not leak any information about the rest of the secure

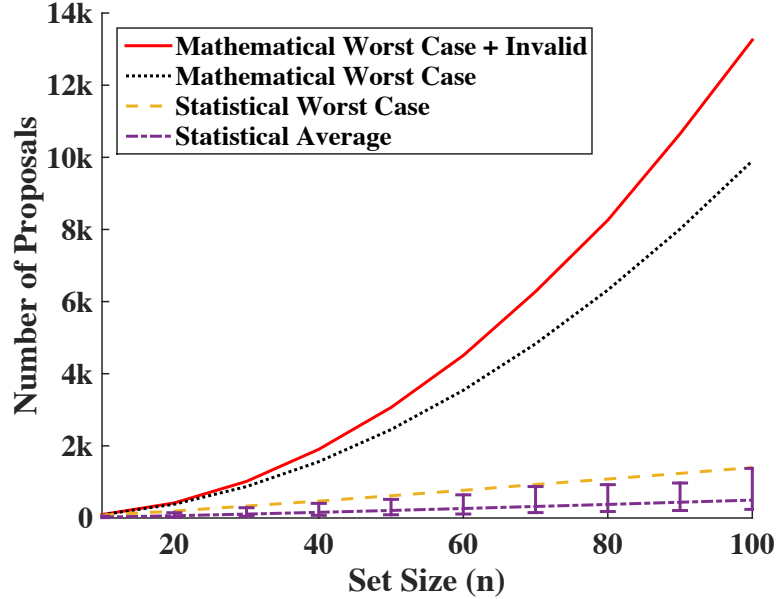


Figure 3.4 : Number of proposals needed for different scenarios in the general SM. Statistical worst case is computed for  $\alpha = 8$ ). The error bar shows the minimum and maximum value in the simulation. Experiment is performed 10 000 times for each set size with randomly generated preference list.

computation except the fact whether or not the computation is finished up to this iteration. Thus in our implementation, the computation and communication can be scaled down by a factor of  $\mathcal{O}(n)$  (see Section 6.1.4 for more details on information leakage and Section 6.3 for results).

### 3.2.4 Sub-linear Sequential Circuit

In Section 3.2.2, we proposed a sequential circuit for general and limited SM. The size of the circuit is  $\mathcal{O}(n^2 \log n)$ . We also discussed in Section 3.2.3 that the circuit should run for  $\mathcal{O}(n^2)$  iterations for the worst case scenario and can be reduced using early termination (ETT) to  $\mathcal{O}(n)$ . Therefore, the overall computation and communi-

cation complexity of the secure SM is  $\mathcal{O}(n^3 \log n)$  using ETT. This overall cost is the product of the number of iterations and the size of the circuit. For the set size of the NRMP ( $n = 32k$ ), the cost is in the order of  $2^{50}$  AND gates which incurs  $32PB$  communication! The number of iterations (proposals) is already minimized by MSC and ETT and cannot be reduced further. Thus, the only way is to reduce the circuit size. To do so, we propose four tweaks for the sequential circuit.

First, we replace the MUXs accessing the memory (e.g., preference list of men and women) with sub-linear ORAM. It allows us to access the memory of size  $m$  with the cost of less than  $\mathcal{O}(m)$ . We will discuss the choice of ORAMs and their trade-offs in Section 3.2.5.

Second, we need to change the size of PCC from linear to sub-linear. PCC outputs a binary indicating if a woman prefers the new proposing man over her already assigned man. PCC linearly scans the woman's preference list (a vector of man's ID) in order to find which man came first in the list. (Note that the list is sorted from most desirable to the least.) This incurs  $\mathcal{O}(n)$  AND gates. To make PCC sub-linear, we store the inverse of the preference list of each woman. For example if man #5 is the 3<sup>rd</sup> one in the preference list, in the inverse list, the value of the 5<sup>th</sup> position is 3. Now instead of a linear search, we need two accesses to the inverse list and compare their values to determine the more desirable man. However, access cost using MUX is still  $\mathcal{O}(n)$ . But the MUX can be replaced with sub-linear ORAM to make the total cost sub-linear with respect to  $n$ .

Third, ETT needs to access the status of all men in order to determine the Finish Signal ( $\mathcal{O}(n)$  AND gates). Accessing the men's status list using sub-linear ORAM does not solve the problem since we still need to access men's status list  $n$  times at each iteration. This makes the overall cost even worse than Linear ORAM. Therefore,



we have to amortize the cost by running ETT once every  $\beta \cdot n$  iterations, where  $\beta$  is a constant, making the amortized cost of ETT  $\mathcal{O}(1)$ .

Forth, MSC requires scanning over the status of all men at each iteration to choose the next free man. Similar to ETT, this incurs  $\mathcal{O}(n)$  AND gates. Unlike the solution for ETT, we cannot skip this computation for any iteration. We propose to store the ID of all free men in a *queue*. We dequeue a free man and process his proposal. If his proposal gets rejected, we enqueue him to the back of the queue. Or if in the process of the proposal, a previously assigned man becomes free, we enqueue this man to the queue. The queue should be initialized by all men at the beginning of the SM. Therefore, the capacity of the queue should be exactly  $n$ . We implement this queue using an array of size  $n$  and circular indexing. This is the most efficient and simplest solution and incurs a constant number of accesses per each enqueue or dequeue operation. By storing this array in a sub-linear ORAM, the size of MSC will become sub-linear.

### 3.2.5 Memory Analysis and Choosing ORAM Scheme

There are several ORAM schemes in the literature [49, 50]. Each of them has different initialization cost and access complexity. Depending on the intensity of memory accesses and size of the memory, each ORAM may outperform the others. Thus, it is important to find at which memory size, one ORAM scheme starts to outperform the others. This is called the *breakeven* point. To the best of our knowledge, Circuit ORAM [49] has the best asymptotic complexity  $\mathcal{O}(\log^3 n)$ . However, for small memory sizes, Linear ORAM (MUX) outperforms all sub-linear ORAMs including Circuit ORAM. The reason is twofold: (i) The cost of initialization for Linear ORAM is negligible compared to sub-linear ORAMs. (ii) The constant coefficient for its access

cost complexity is lower than the one for sub-linear ORAMs.

For medium set size, another sub-linear ORAM, Square-Root ORAM [50], outperforms both Linear ORAM and Circuit ORAM. The reason is that it has lower initialization cost compared to Circuit ORAM but has better access cost complexity of  $\mathcal{O}(\sqrt{n \log n})$  compared to Linear ORAM. As the set size increases, Circuit ORAM eventually outperforms both Square-Root ORAM and Linear ORAM.

Based on the characteristics and performance of different ORAM schemes, we integrated our sequential circuit with the best ORAM scheme. The choice of ORAM also depends on which variant of secure SM we want to use (General SM or Limited SM) and whether or not we are interested to use the early termination technique.

Table 3.1 shows the overall asymptotic complexity which includes both initialization and per-access cost. Note that in order to find the most efficient one, we need to perform experiments. In Section 6.1.2, we will show what is the best choice of ORAM depending on the set size and SM variant based on our experiments.

Table 3.1 : Total complexity for implementing each variant of SM using different ORAM schemes, where  $n$  is the set size and  $k$  is the limit of number of proposals in limited SM.

ORAM Scheme	General SM	General SM + ETT	Limited SM
Linear ORAM	$\mathcal{O}(n^4 \log n)$	$\mathcal{O}(n^3 \log n)$	$\mathcal{O}(n^2 k^2 \log n)$
Square-Root ORAM	$\mathcal{O}(n^3 \log^2 n)$	$\mathcal{O}(n^2 \log^2 n)$	$\mathcal{O}((nk)^{1.5} \log^{1.5}(nk) \log^{0.5} n)$
Circuit ORAM	$\mathcal{O}(n^2 \log^3 n)$	$\mathcal{O}(n^2 \log^3 n)$	$\mathcal{O}(nk \log^2(nk) \log n)$

### 3.3 Methodology for Scaling GC

In this section, we use two techniques that allow to scale the GC protocol to a real world problem size. (1) Scalable circuit generation (Section 3.3.1): which aims to mitigate the GC netlist generation scalability problem by exploiting the inherent structures of various parts in the circuit. (2) Early termination technique (Section 3.3.2): which significantly reduces the computation by revealing the termination status at several points in the execution of the protocol.

#### 3.3.1 Scalable Circuit Generation

A number of methods [18, 70] have been proposed in recent years to generate an efficient circuit for GC. Unfortunately, none of them provide a scalable solution for the real-world-size secure SM problem. In order to obtain a practical solution, we developed optimizations and efficient circuit designing techniques described in Sections 2.2.1 and 3.2 which give us a tremendous improvement toward scalable secure SM. However, the scalability of those solutions is still limited to the capability of generating a netlist using the underlying synthesis tool. For example, the Synopsys Design Compiler which is used in [18] is able to synthesize a circuit with at most  $\sim 5 \times 10^5$  gates\* which limits us to scale up to a real-world-size with  $\sim 10^8$  gates. To bypass this limitation, we divide the circuit into two parts first:

**(1) Structural** which follows a specific pattern regardless of the problem size, e.g., MUX, encoder, and decoder. The optimized circuit for a structural module can be described based on only its size parameters e.g., number of entities for encoder.

---

\*According to our experiments

The structural module’s description can be achieved independently of the rest of the circuit.

**(2) Non-structural** which simply is not structural and does not follow a specific pattern, e.g., controller and FSM. The netlist for a non-structural module can not be easily described and it requires a sophisticated tool to optimize and generate. On the other hand, non-structural parts require a considerably smaller number of gates compared to the structural parts. For example in our SM problem, the non-structural modules allocate  $\sim 500$  gates compared to  $\sim 10^8$  for structural parts.

In our methodology, we use two different flows for generating the netlists for the structural and non-structural parts of the circuit. For structural part, we develop a tool which synthesizes and generates the netlist for any arbitrary size parameters and hence can be scaled to a real-world set size. Unlike previous GC circuit generation methods, our tool is aware of the inherent pattern of structural modules and can generate the optimized netlist. For the non-structural part, we can use any previous method for generating a netlist because the size of the non-structural part does not scale with the size of the problem and hence can be easily synthesized.

At the end, we can merge the resulting netlists from the previous steps together to make the final netlist. The final netlist is optimized since each submodule is separately optimized. We can use this netlist as the input to the GC protocol.

### 3.3.2 Early Termination Technique

It is well known that the execution time of a number of algorithms (e.g., SM and iterative algorithms with a convergence condition) depends on the input and can not be determined easily, however, the termination condition can be efficiently verified

during the execution. If the computation of these algorithms is done in plain-text, the termination condition can be checked in every iteration, thus, the execution is aborted accordingly. Whereas, when the computation is done securely, e.g. in the GC protocol, all intermediate values are encrypted and can not be determined without concession of the parties involving in the secure computation. Hence the computation should continue for the predetermined worst case scenario. This results in an unnecessary computation after reaching the final result. This useless computation can be avoided, only if the termination condition can be verified at some points in the middle of the process. For example in SM, the average number of proposals required for the stable match is much smaller than the worst case number of proposals (see Section 6.3). More precisely, the average number is  $\mathcal{O}(n)$  while the worst case number is  $\mathcal{O}(n^2)$ .

In two-party GC to verify the termination condition in the middle of the process, the parties reveal the shared secret associated only to the termination signal (wire). Revealing this secret will not leak any information about the rest of the secure computation except the fact that whether or not the computation is finished.

Thus in SM, the computation and communication can be scaled down by a factor of  $\mathcal{O}(n)$  by revealing the termination signal at the end of each proposal (see Section 6.1.4 for more detail on information leakage and Section 6.3. In Algorithm 1, the termination condition can be easily verified by observing the matching status of all the men. Once all the men are matched, the match will be stable. Thus, the termination condition is the logical AND over all the men's status; if all of them are matched, then the termination signal is true.

## Chapter 4

### GenMatch: Secure DNA Compatibility Testing

#### 4.1 Scenario

We explore the scenario where a patient needs an organ transplantation and is looking for compatible donors. We assume that the patient has undergone full genome sequencing. The HLA data is obtained from pre-processing the genome data. Fully sequenced genomes are very large files and secure computation on the data as a whole is both infeasible and unnecessary. Instead, we extract the information required for compatibility testing through off-line processing. Assuming we have a database of HLA data of donors, we find the best possible match for the patient's HLA data.

For two parties, Figure 4.1 shows the block diagram of our system. Figure 4.1 shows the block diagram of our system.

#### 4.2 Pre-processing

The HLA data needed for our test can be extracted from the fully sequenced genome. Samples of fully sequenced genome data are available through the Human Genome Project to researchers for analysis and application specific usage [28]. This data can be processed using HLA-genotyper [71], a python based library used for HLA typing. This is an offline process as it involves no interaction between the two parties and it is also convenient as it only has to be done once and the obtained data can be stored and used in the future.

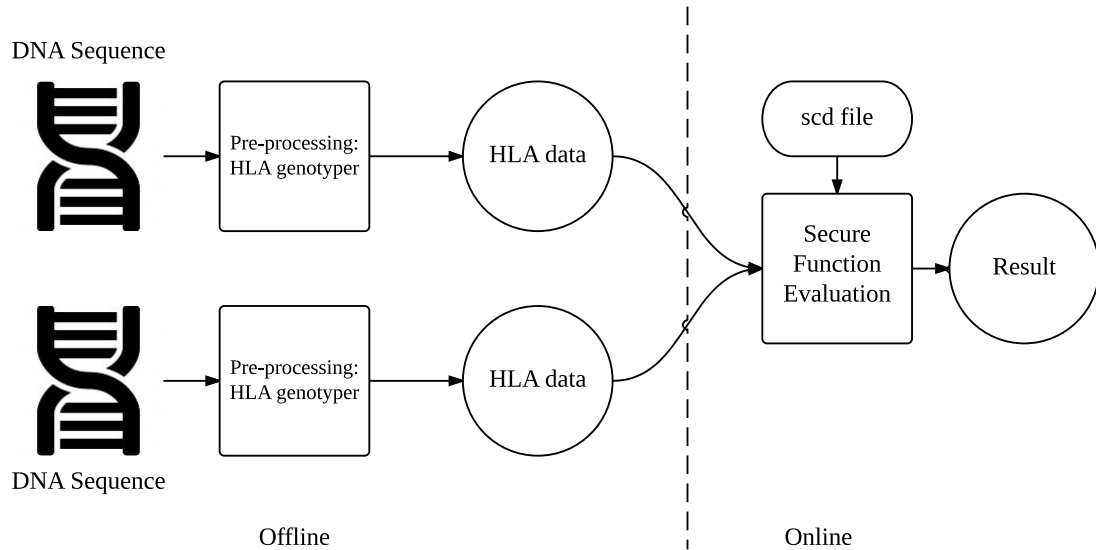


Figure 4.1 : Block diagram of total end-to-end system including pre-processing (offline stage) and secure function evaluation (online stage).

### 4.3 Boolean Circuit

The extracted data from pre-processing contains a pair of haplotypes. One from the mother and the other from the father. There are 6 pairs of haplotype data, HLA-A, HLA-B, HLA-C, HLA-DQA, HLA-DQB, and HLA-DRB. The Algorithm 3 describes the process of comparison of two HLA data [57, 72, 73].

However, this hardware descriptor file itself is not the direct input to GC protocol. As Figure 4.2 shows several process stages are needed to make this circuit an acceptable input. First, the Verilog code of the circuit should be synthesized to give the netlist file of the circuit. Netlist file describes the circuit only as gates and how they are connected to each other rather. Second, we need to process this netlist to sort the gates topologically to prevent GC protocol getting stopped in the middle of garbling and evaluating process. This is necessary because if the garbler garbles the

---

**Algorithm 3** Algorithm for computing percentage of HLA compatibility between 2 persons.

---

**Inputs:** 6 pairs of HLA data from person 1 (HLA1 [index] [pair]) and 6 pairs of HLA data from person 2 (HLA2 [index] [pair]).

**Outputs:** Percentage of compatibility between two samples.

```

1: total compatibility = 0
2: for  $n = 1$  to 6 do
3:   if  $HLA1[n][1] == HLA2[n][1]$  then
4:     if  $HLA1[n][2] == HLA2[n][2]$  then
5:       compatibility = 1
6:     else
7:       compatibility = 0.5
8:     end if
9:   else if  $HLA1[n][2] == HLA2[n][1]$  then
10:    if  $HLA1[n][1] == HLA2[n][2]$  then
11:      compatibility = 1
12:    else
13:      compatibility = 0.5
14:    end if
15:   else if  $HLA1[n][1] == HLA2[n][2]$  then
16:     compatibility = 0.5
17:   else if  $HLA1[n][2] == HLA2[n][2]$  then
18:     compatibility = 0.5
19:   else
20:     compatibility = 0
21:   end if
22:   total compatibility = total compatibility +  $\frac{1}{6} \times$  compatibility

```

---



circuit in an order other than topological order, it is possible that he can not continue garbling because the input to the gate he is garbling is not ready yet. We also need to describe that to which party (garbler or evaluator) each input wire belongs. Finally, these information are stored in SCD file.

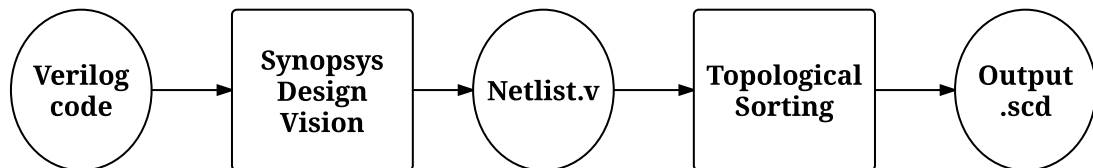


Figure 4.2 : Different stages for converting the Boolean Circuit to required input for Yao’s garbled circuit protocol

We use Verilog to characterize the Boolean circuit required for the GC protocol. This circuit calculates the percentage of compatibility between two samples of HLA data and is shown in Figure 4.3. This is a sequential circuit which takes the patient’s HLA data and also one HLA profile in a database as inputs at each clock cycle. The circuit compares the compatibility among pairwise HLA profile as described in Algorithm 3. The final compatibility is the average of all pairwise HLA type compatibilities. Instead of directly finding the average, we propose an alternative approach, described in Section 4.4, which results in a more efficient computation. The final compatibility is then compared with the previously most compatible profile (pre-)stored in the memory (D-FFs). If this new HLA profile is more compatible than the previous one, this new value along with the index of this new profile will be stored in the memory. To keep track of which index we are comparing at a given time, we have embedded a counter in this circuit which is incremented at each clock cycle by one. Since we compare one profile from the database at each clock cycle, we need to evaluate the circuit for  $N$  clock cycles, where  $N$  is the number of profiles in

the database.

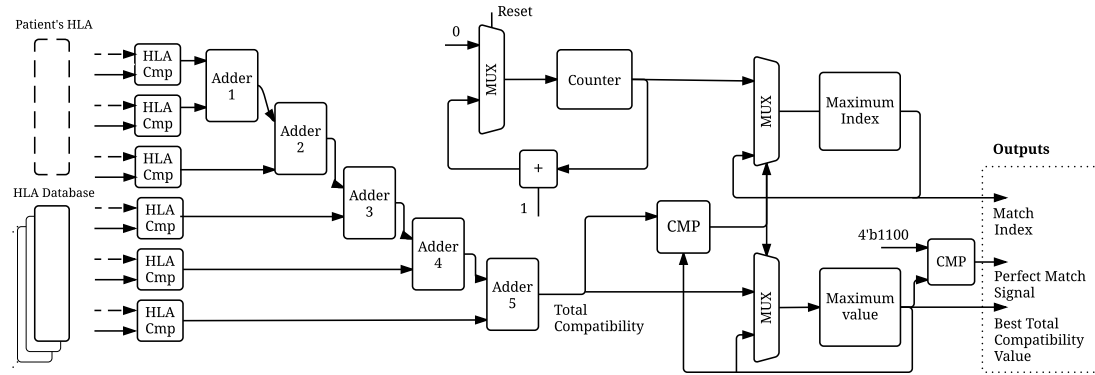


Figure 4.3 : Architecture of the sequential circuit used in the GC protocol.

#### 4.4 Circuit Optimizations and Size Model

The optimizations we describe here are different from the state-of-the-art optimizations that are used for executing the GC protocol in Section 2.2.1. In Section 2.2.1, we explain that an XOR gate does not need a garbled table and hence the cost of computation of an XOR gate is negligible. Therefore, the goal is to minimize the number of non-XOR gates in the circuit. Here are some techniques we propose:

**Translating floating-point operations into integer operations** As discussed in Section 4.2, we need to calculate the compatibility of pairwise HLA data between the patient and a profile in the database. Possible outcomes for each HLA type comparison can be 0, 0.5 or 1, we can encode these outcomes into 3 binary values  $00_2$ ,  $01_2$  and  $10_2$ . Then only 2 bits are sufficient to represent them. For calculating the final average, we need to do an integer addition instead of the floating-point addition and the division can be omitted since it is a division by fixed number for

all comparisons among different profiles (division by 6). This will result in a huge reduction in the circuit size.

**Designing special building blocks** Since the goal is to reduce the number of non-XOR gates in the circuit, we need to design each block to get the minimum number of non-XOR gates possible. To achieve this goal we have designed special comparison block. This block has only  $b - 1$  non-XOR gates for comparing the binary input of length  $b$ -bit. At each clock cycle, we need to increment the counter by one and this is fixed for all clock cycles and hence, we have optimized this counter to have the least number of non-XOR gates.

**Adder optimization** In order to find the summation of 6 pairwise comparisons between different HLA types, we propose a hierarchical structure as shows in Figure 4.3 instead of instantiating the default addition block. This structure is more efficient because it gives us a dynamic bit-length. The bit-length of output for each adder increases as we add the results of more comparisons. As an example, in the worst-case-scenario, for the first addition, we only need to add  $10_2$  and  $10_2$  which will result in 2-bit adder and use a carryout bit. As we progress in this structure we need higher bit-length to represent the result and this will give us the most efficient way to do this summation. Also, each adder is specially designed to have the least number of non-XOR gates.

**Perfect match signal** We have a 1-bit signal in the circuit which sets to high if we find a perfect match between patient's HLA data and one of the profiles in the bank. Therefore, there is no need to continue the protocol and we can terminate it sooner. The perfect match happens when all HLA types are same. Since every value during

the execution of the GC protocol is encrypted, both Garbler and Evaluator need to reveal their shared secret to achieve the actual value of this signal. While revealing this bit will inform us as soon as we have found the best match possible, it will not reveal anything about any HLA data from patient side and from database side.

#### 4.4.1 Circuit Size Model

Here we analyze the size of the circuit and present a quantitative formula for its size. There are some parts of the circuit that are not dependent on the database size ( $N$ ) such as HLA comparison blocks and hierarchical structure of adders. However, some parts have a number of non-XOR gates proportional to  $\log N$ , including counter and maximum value comparison block. Therefore, the final number of non-XOR gates is a linear function of  $\log N$  plus a constant factor. Figure 4.4 proves our analysis and shows the number of non-XOR gates for different number of database size ( $N$ ). The experimental results deviate slightly from theoretical function due to the heuristic approaches used for synthesizing the circuit. The formula for calculating the number of non-XOR gates is  $\alpha \times \log N + \beta$ , where  $\alpha = 18.6$  and  $\beta = 395$ . As we discussed, we need to evaluate this sequential circuit for total number of  $N$  clock cycles to compare the patient's HLA data with all of the HLA data in the database. Therefore, total number of garbled tables we need to evaluate, one for each non-XOR gate, is  $N$  times the total number of non-XOR gates in the circuit:

$$\text{Total \# of garbled tables} = N \times (\alpha \times \log N + \beta)$$

Above equation shows that our approach can be easily scaled up to huge database sizes such as million due to the *linearithmic* complexity. Note that in the GC protocol, the computational time and communication are proportional to the total number of garbled tables.

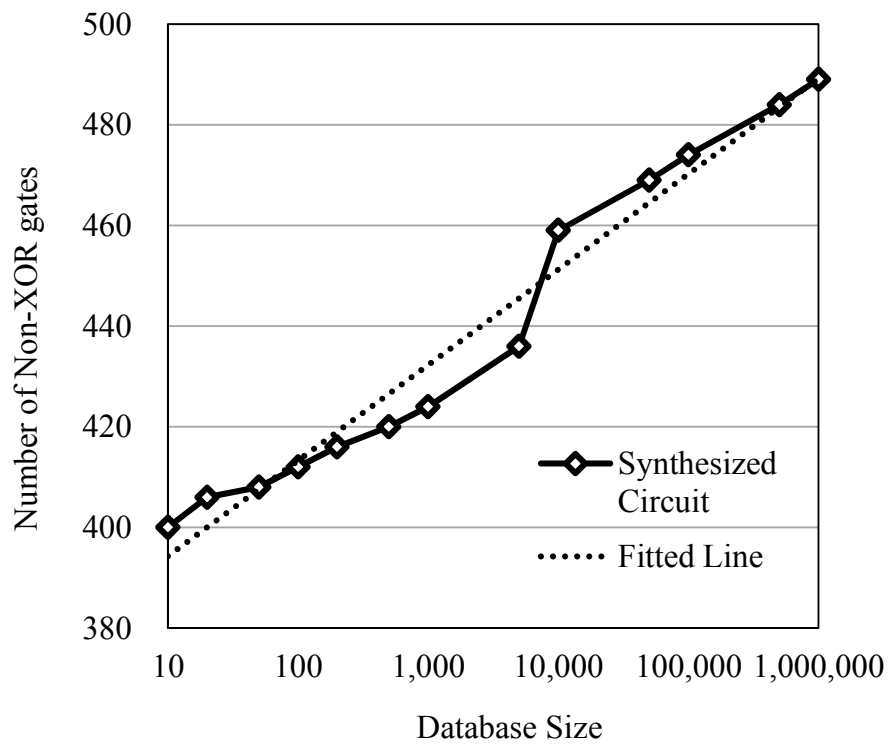


Figure 4.4 : Number of non-XOR gates of the circuit vs. database size  $N$ .

## Chapter 5

### S-LSH: Secure Locality Sensitive Hashing

#### 5.1 LSH and Triangulation Attack

Here we explain why any scheme that allows for estimation of *any* pairwise distances is susceptible to triangulation attack. For illustration, we focus in the context of hashing (or embedding) based approach. Suppose an attacker has the knowledge of the location of any three random points in original space and their corresponding hashes (public). Such an information is not hard to obtain, in fact, it can be fake online profiles. Given these three points and their corresponding hashes, any attacker can compute the original location of *any target point* (user's input) by estimating its distance with the three pre-chosen points. Figure 5.1 shows a two-dimensional illustration of our setup. Let the three different points be  $A, B$ , and  $C$ , as shown. The target point is illustrated as  $q$ . The estimated distances derived from hash comparisons are described as  $d_A, d_B$ , and  $d_C$ .

From point  $A$ 's perspective,  $q$  can be anywhere on a circle with radius  $d_A$  with  $A$  as the center. Figure 5.1 shows three circles corresponding to the points  $A, B$  and  $C$ . If our estimate of distances are accurate, which is true with general LSH because they can be easily used for estimation [74], then intersection of the three circles is a very close estimate of the exact location of the  $q$ . Even if the distance estimation is not very accurate, the three circles would intersect at six points and it is still possible to find the location of the  $q$  as explained in supplementary material. In general, if

the data is  $\mathbb{D}$  dimensional, an attacker needs  $\mathbb{D} + 1$  random points to find the exact location of the  $q$ .

Thus, protocols revealing the distances for any pair of points are not “truly” secure. Our approach, on the other hand, is secure against this attack because our hash comparison reveals the distances only if the points, under consideration, are very close to each other, meaning that the server should know the location of the  $q$  beforehand. It should be noted that we cannot hope to do better, because any near-neighbor oracle, having reasonable utility, should identify close by points.

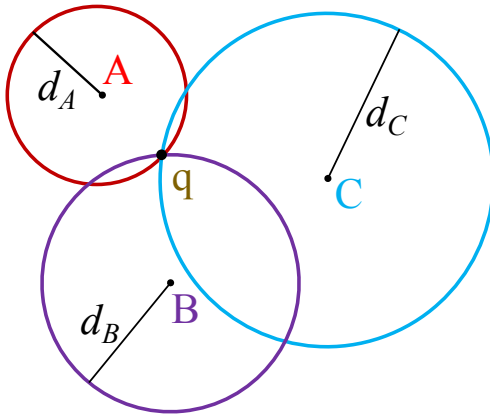


Figure 5.1 : Triangulation Attack. The user  $q$  and the random points  $A$ ,  $B$ , and  $C$ . The  $d_A$ ,  $d_B$ , and  $d_C$  are estimated distances from comparing hashes of  $q$  and the other three points.

## 5.2 Making LSH Secure

Our proposal is a generic framework for making LSH signatures privacy-preserving and allowing for secure public release. See Section 5.2.4 for the description of our protocol. We illustrate the main idea using 1-bit MinHash and later we formally introduce the methodology. The collision probability, for any two given data points

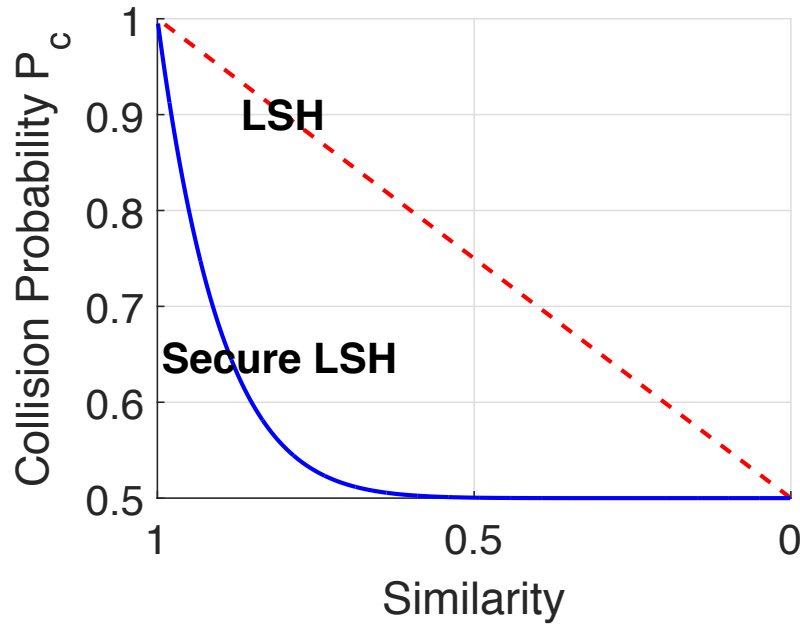


Figure 5.2 : LSH and Secure LSH.

$x$  and  $y$ , under 1-bit MinHash is given by  $\frac{\mathcal{R}(x,y)+1}{2}$ . This quantity linearly varies, between 1 to 0.5 as  $\mathcal{R}(x,y)$  varies from 1 to 0, with a constant gradient of  $\frac{1}{2}$ . See Figure 5.2 (dotted red line). Thus, even when  $\mathcal{R}(x,y)$  is small, the variation of the collision probability with distance keeps changing and it gets reflected in the hamming distance between the public  $l$ -bit hash string required for near-neighbor search. This also allows us to estimate the distances accurately by simply counting the number of bit-matches out of the publicly released  $l$ -bits.

For making the LSH privacy-preserving and still useful for the task of near-neighbor search, we want to achieve the collision probability curves, given by bold black line, in Figure 5.2. For this new curve, once the similarity between the pairs of interest  $x$  and  $y$  is below the satisfactory level, the collision probability becomes flat with no gradient. With such collision probability, the hamming distance between the publicly available  $l$ -bit hash codes, for any pair of random points  $x$  and  $y$ , will



be always around  $l/2$  (due to 0.5 probability of agreement) not allowing any efficient estimation of distances between  $x$  and  $y$  based on observed  $l$ -bits.

At this point, we have realized that we have to transform the collision probability curve. The main challenge at hand is to find an expression for collision probability curve which is close to the desired bold blue curve, in Figure 5.2, and at the same time represents the collision probability of some 1-bit hashing scheme. We found that the expression given by  $\frac{\mathcal{R}(x,y)^{k+1}}{2}$  has the required “sweet” property. In particular, we will construct a new 1-bit private MinHash with collision probability  $\frac{\mathcal{R}(x,y)^{k+1}}{2}$  for any positive integer  $k$ , instead of  $\frac{\mathcal{R}(x,y)+1}{2}$ . The key observation is that since  $\mathcal{R} \leq 1$ ,  $\mathcal{R}^k$  for reasonably large  $k$  quickly falls to zero as  $\mathcal{R}(x,y)$  goes away from 1, and so the quantity  $\frac{\mathcal{R}(x,y)^{k+1}}{2}$  will be very close to  $\frac{1}{2}$  for even moderately low similarity. Furthermore, we can control the decay of the probability curve by choosing the appropriate  $k$ . We will later show that  $\frac{\mathcal{R}(x,y)^{k+1}}{2}$  follows the desired trend of collision probability, and it is also secure from information theoretic perspective.

We can generate 1-bit hash functions with collision probability  $\frac{\mathcal{R}(x,y)^{k+1}}{2}$  by combining  $k$  independent MinHashes. Note that, given  $x$  and  $y$ , the probability of agreement of an independent MinHash value is  $\mathcal{R}(x,y)$ . Therefore, the probability of agreement of all  $k$  independent MinHashes will be  $\mathcal{R}(x,y)^k$ . To generate a 1-bit hash value from  $k$  integers, we will, in addition, need a universal hash function, that will take a vector of  $k$  MinHashes and map it uniformly to 1-bit. The final collision probability will be precisely  $\frac{\mathcal{R}(x,y)^{k+1}}{2}$  as required. The overall idea is quite general and applicable to any LSH. We formalize it in the next section.

### 5.2.1 Formalization

We will need a universal hashing scheme,  $h_{univ} : \mathbb{N}^k \mapsto \{0, 1\}$ , which maps a vector of  $k$  integers uniformly to 0 or 1. There are many ways to achieve this, a common candidate is

$$h_{univ}(x_1, x_2, \dots, x_k) = (r_{k+1} + \sum_{i=1}^k r_i x_i) \bmod p, \bmod 2,$$

where  $r_i$  are fixed integer numbers generated randomly.

Given a hash function  $h$  uniformly sampled from the locality sensitive family  $\mathcal{H}$ , let us denote the probability of agreement (collision) of hash values of  $x$  and  $y$  under the sampling of  $h$  by  $P_c$

$$P_{collision} \equiv P_c \equiv Pr_{\mathcal{H}}(h(x) = h(y)). \quad (5.1)$$

Our proposed secure 1-bit LSH,  $h_{sec}^{1bit}$ , for any point  $x$  is given by

$$h_{sec}^{1bit}(x) = h_{univ}(h_1(x), h_2(x), \dots, h_k(x)), \quad (5.2)$$

where  $h_i$ ,  $i \in \{1, 2, \dots, k\}$  are  $k$  independent hash functions sampled uniformly from the LSH family of interest  $\mathcal{H}$ . It is not difficult to show the following

*Theorem 5.1*

*For any vectors  $x$  and  $y$ , under the randomization of  $h$  and  $r_i$  we have*

$$P_c^{sec} = Pr_{\mathcal{H}, r}(h_{sec}^{1bit}(x) = h_{sec}^{1bit}(y)) = \frac{P_c^k + 1}{2} \quad (5.3)$$

*Proof 5.1* It should be noted that  $h_{sec}^{1bit}(x) = h_{sec}^{1bit}(y)$  can happen due to the random bit collision with probability  $\frac{1}{2}$ , otherwise the two are equal if and only if  $(h_1(x), h_2(x), \dots, h_k(x)) = (h_1(y), h_2(y), \dots, h_k(y))$  which happens with probability  $P_c^k$ , because each of the  $h_i$ 's is independent and  $Pr(h_i(x) = h_i(y)) = P_c$ . Therefore, the total probability is  $\frac{1}{2} + \frac{1}{2}P_c^k$  leading to the desired expression.

We illustrate the usefulness of the above-proposed framework in deriving secure 1-bit hash for two most popular similarity measures: (i) Secure MinHash for Jaccard similarity and (ii) Secure SimHash for Cosine similarity. Secure hashing schemes for these similarity measures were not known before.

### 5.2.2 Secure Minwise Hashing (Secure MinHash)

As an immediate consequence of Theorem 5.1 we obtain secure 1-bit MinHash for searching with Resemblance similarity,

$$h_{sec}^{min,1bit}(x) = h_{univ}(h_{\pi_1}^{min}(x), h_{\pi_2}^{min}(x), \dots, h_{\pi_k}^{min}(x)), \quad (5.4)$$

with the following Corollary:

*Corollary 1 For MinHash we have:*

$$P_c^{sec} = Pr(h_{sec}^{min,1bit}(x) = h_{sec}^{min,1bit}(y)) = \frac{\mathcal{R}^k + 1}{2} \quad (5.5)$$

Our protocol requires  $l$  independent bits. For this, we simply generate  $l$  independent  $h_{sec}^{min,1bit}$ , by using independent permutations for MinHashes and independent random numbers for the universal hashing. Therefore,  $h_{sec}^{min}$  is the concatenation of  $l$  different  $h_{sec}^{min,1bit}$ . Figure 5.3 (left) shows that the nature of new collision probability follows the desired trend.

Having secure 1-bit MinHash, we can build secure  $l$ -bit MinHash (Secure MinHash) by simply generating  $l$  different 1-bit MinHashes.

$$h_{sec}^{min}(x) = h_{sec}^{min,1bit(1)}(x) || h_{sec}^{min,1bit(2)}(x) || \dots || h_{sec}^{min,1bit(l)}(x) \quad (5.6)$$

Please note that each 1-bit MinHash in equation 5.4 is derived independently and hence uses different permutations for MinHash and different coefficient for its universal hash. It is straightforward to show that the probability of two secure MinHashes

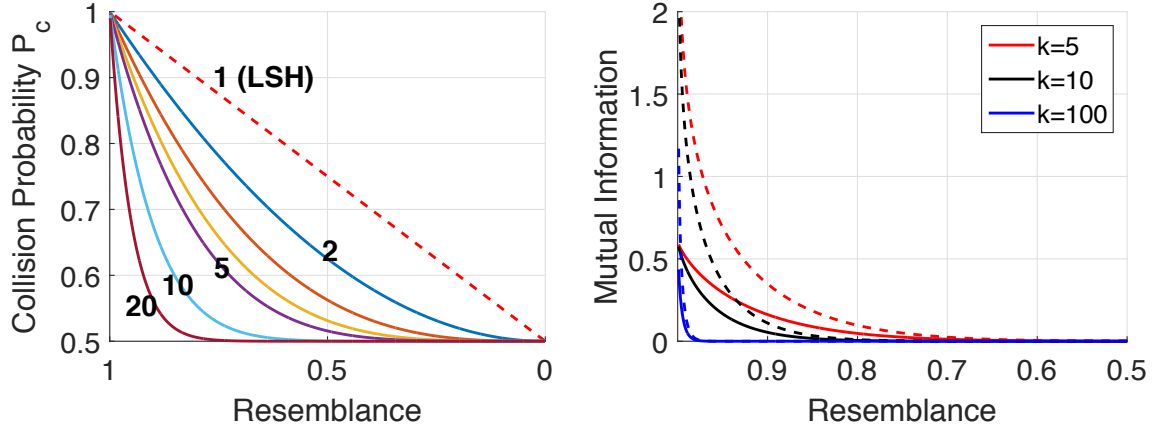


Figure 5.3 : **Left:** The probability of collision plotted as function of  $R$  (resemblance) for different values of  $k$  (degree). **Right:** Mutual Information between two 1-bit hashes as a function of  $R$  (resemblance) for different values of  $k$ .

to be equal can be calculated from the equation 5.5 since each 1-bit MinHash is independent.

$$Pr(h_{sec}^{min}(x) = h_{sec}^{min}(y)) = \left(\frac{\mathcal{R}^k + 1}{2}\right)^l \quad (5.7)$$

Next, we formally show that our bits are secure. In particular, we show that the mutual information between the two secure MinHashes,  $h_{sec}^{min}(x)$  and  $h_{sec}^{min}(y)$ , decays sharply to zero as the similarity between  $x$  and  $y$ , i.e.  $\mathcal{R}$  goes away from 1.

### Information Theoretic Bound

Here, we determine the mutual information between two MinHashes to illustrate the security property of our approach. Since each bit is driven independently of other bits, it is easy to show that the total mutual information between two hashes is just the multiplication by  $l$  (total number of bits in one hash) of the mutual information between  $i^{th}$  bit of two MinHashes. For simplicity let us call the  $i^{th}$  bit of  $h_{sec}^{min}(x)$ ,  $u_i$

and  $i^{\text{th}}$  bit of  $h_{\text{sec}}^{\text{min}}(y)$ ,  $u'_i$ . We derive the mutual information between these two bits conditioned on the similarity of  $x$  and  $y$  ( $\mathcal{R}$ ) as follows:

$$\begin{aligned}
I(u_i; u'_i | \mathcal{R}) &\equiv \sum_{u_i, u'_i \in \{0,1\}} P(u_i, u'_i | \mathcal{R}) \log \frac{P(u_i, u'_i | \mathcal{R})}{P(u_i | \mathcal{R}) P(u'_i | \mathcal{R})} & (5.8) \\
&= P_c^{\text{sec}} \log(2P_c^{\text{sec}}) + (1 - P_c^{\text{sec}}) \log(2(1 - P_c^{\text{sec}})) \\
&= \log(2(1 - P_c^{\text{sec}})) + P_c^{\text{sec}} \log\left(\frac{P_c^{\text{sec}}}{1 - P_c^{\text{sec}}}\right) \\
&= \frac{1}{2} (\log(1 - \mathcal{R}^{2k}) + \mathcal{R}^k \log\left(\frac{1 + \mathcal{R}^k}{1 - \mathcal{R}^k}\right)) \\
&< \mathcal{R}^k \log\left(\frac{1 + \mathcal{R}^k}{1 - \mathcal{R}^k}\right)
\end{aligned}$$

Therefore the mutual information between two MinHashes is bounded by the following formula:

$$I(h_{\text{sec}}^{\text{min}}(x); h_{\text{sec}}^{\text{min}}(y) | \mathcal{R}) < l \cdot \mathcal{R}^k \log\left(\frac{1 + \mathcal{R}^k}{1 - \mathcal{R}^k}\right) \quad (5.9)$$

As can be seen from Equation 5.9, the mutual information drops rapidly as the resemblance goes to zero, which shows the security of our approach. Figure 5.3 (right) summarizes the exact mutual information (bold lines), with varying  $k$ , as the function of Resemblance along with their upper bounds (dotted lines). It is evident that, for any two non-neighbor points,  $x$  and  $y$ , the generated bits behave like random bits giving no information about their similarity. The choice of  $k$  controls the similarity thresholds that we are interested in.

### 5.2.3 Secure Signed Random Projections (Secure SimHash)

Analogous to the secure MinHash, we can derive secure SimHash having similar desired properties. From Theorem 5.1 the secure 1-bit SimHash for searching with

Cosine similarity is given by,

$$h_{sec}^{sign,1bit}(x) = h_{univ}(h_{w_1}^{sign}(x), h_{w_2}^{sign}(x), \dots, h_{w_k}^{sign}(x)). \quad (5.10)$$

For this hashing scheme we have

$$h_{sec}^{sign}(x) = h_{sec}^{sign,1bit(1)}(x) \parallel h_{sec}^{sign,1bit(2)}(x) \parallel \dots \parallel h_{sec}^{sign,1bit(l)}(x) \quad (5.11)$$

with the following Corollary:

*Corollary 2 From Theorem 5.1 for SimHash we have:*

$$P_c^{sec} = \frac{(1 - \frac{\theta}{\pi})^k + 1}{2} \quad (5.12)$$

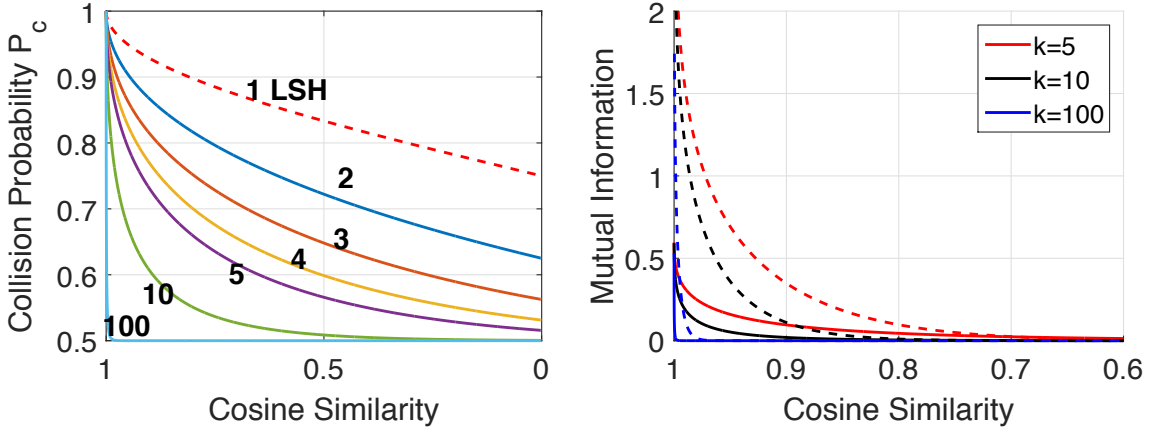


Figure 5.4 : **Left:** Collision probability plotted as function of  $\cos(\theta)$  for different values of  $k$  (degree of transformation). **Right:** Mutual Information between two SimHashes as a function of  $\cos(\theta)$  for different values of  $k$ .

and as a result:

$$Pr(h_{sec}^{sign}(x) = h_{sec}^{sign}(y)) = \left(\frac{(1 - \frac{\theta}{\pi})^k + 1}{2}\right)^l \quad (5.13)$$

We formally show the mutual information between secure bits  $h_{sec}^{sign}(x)$  and  $h_{sec}^{sign}(y)$  rapidly goes to zero as the Cosine similarity between  $x$  and  $y$  goes to zero.

### Information Theoretic Bound

Similarly, the mutual information bound between the two bits, of secure SimHash, conditioned on cosine similarity of  $x$  and  $y$  ( $\frac{x^T y}{\|x\| \cdot \|y\|} = \cos(\theta)$ ) as:

$$\begin{aligned} I(u_i; u'_i | \cos(\theta)) &= \\ & \frac{1}{2} (\log(1 - (1 - \frac{\theta}{\pi})^{2k}) + (1 - \frac{\theta}{\pi})^k \log(\frac{1 + (1 - \frac{\theta}{\pi})^k}{1 - (1 - \frac{\theta}{\pi})^k})) \\ & < (1 - \frac{\theta}{\pi})^k \log(\frac{1 + (1 - \frac{\theta}{\pi})^k}{1 - (1 - \frac{\theta}{\pi})^k}) \end{aligned} \quad (5.14)$$

Therefore the mutual information between two SimHashes is bounded by the following formula:

$$I(h_{sec}^{sign}(x); h_{sec}^{sign}(y) | \cos(\theta)) < l \cdot (1 - \frac{\theta}{\pi})^k \log(\frac{1 + (1 - \frac{\theta}{\pi})^k}{1 - (1 - \frac{\theta}{\pi})^k}) \quad (5.15)$$

Figure 5.4 summarizes both the collision probability as the function of similarity for different  $k$  and also shows the mutual information and the upper bounds. Both these figures show the desired behavior similar to secure MinHash.

#### 5.2.4 Scenario and the Formal Protocol

Our scenario is a traditional near-neighbor search, where every party (or user) is interested in finding their near-neighbors without revealing any information about their attributes. We do not assume the existence of any trusted party or server. Moreover, we want a scalable scheme to work at massive scales with potentially billions of users.

Our overall framework works as follows:

1. **Public Random Seed:** The server announces (or releases) random seeds and the hash functions (e.g. Secure MinHash), required for computing hash bits, publicly available to everyone.
2. **Secure Hash Computations:** Every user computes  $l$  secure hash bits of their own private data and releases those bits publicly to the server. These hash computations are done using the random seed and the hash functions provided by the server in the predefined order. We do not want  $l$  to be large, see Section 5.2.5 for recommendations on choosing  $l$ .

Finding neighbors of a given user boils down to finding users with significant matches (more than 50%) of secure hash bits. This can be performed using simple linear scan. However, a linear scan can be prohibitively expensive for massive number of users. For this case, we can resort to fast hashing based sub-linear search described in Section 5.2.4.

### Sub-Linear Search

Given a query  $q$  and its corresponding  $l$ -bit secure profile, our task is to find users with  $l$ -bit profiles close in hamming distance with the query's profile. This can be efficiently done in sub-linear time, because there is a known LSH scheme for hamming distance, which is based on bit sampling. Theory says that we can perform approximate near-neighbor search in sub-linear time, specifically in  $O(n^\rho)$  time, with  $\rho = \frac{\log p_1}{\log p_2}$ ,  $p_1 = 1 - \frac{d_0}{l}$  and  $p_2 = \frac{1-c}{l} \frac{d_0}{l}$ . Here,  $d_0$  is the hamming distance threshold of interest and  $c$  is the approximation ratio.

However, LSH algorithm requires the knowledge of recommended similarity (or 1-distance) thresholds  $d_0$  which is generally not available for practical problems [75].



Fortunately, due to the nature of our secure hashing scheme, we do have this recommended threshold. In particular, we are interested in points with hamming distance significantly larger than  $\frac{l}{2}$ . Thus, we can choose  $d_0 = \frac{l}{2}$ .

The LSH algorithm for hamming distance requires us to perform the following pre-processing stage only once before starting the protocol. Since this process is computed only once, its cost is amortized constant for future queries.

All parties (whom their data are going to be used in the search process) send their hash of data to the server which creates  $L$  different hash tables as follows. Each hash table uses  $b$  bits as the key. For generating this key, we randomly sample  $b$  indices, from the  $l$  bits of the user's hash. Each user is thus associated with a  $b$ -bit key value in the hash table. For example, if the hash profile of a user is 10010011. If  $b = 2$ , and the randomly sampled two bits turn out to be from positions 3 and 7, then the key for this user will be 01. Since there are  $2^b$  different possible outcomes, there are  $2^b$  possible keys. We repeat this process for all  $L$  different tables by sampling  $b$  bits *independently* every time. Once the  $L$  hash tables are created, the search process works as follows:

**Input:** User holds a query  $x$  ( $x$  can be either  $\mathbb{D}$  dimensional binary or real number data). The server has a database of all  $n$  different secure  $l$ -bit profiles.

**Output:** User obtains a list of ID's of approximate near-neighbors within a pre-defined distance of her query while server does not get the user's query ( $x$ ).

**Protocol:** User computes and sends the  $l$ -bit profile of his query  $h_{sec}(x)$  to the server in plain-text (there is no need for encryption). The server uses the presampled indices to compute the key for the query. This key is used to probe the corresponding hash tables. The server computes the union of all lists from all  $L$  tables (one list per each table) as the potential candidates. Then server compares all hashes in the

reported set with the user's hash and creates the near-neighbors list by adding only those that have significantly more bit-matches than  $\frac{1}{2}$ . Finally, the server sends this list to the user.

### 5.2.5 Assumptions and Practical Issues

**Choice of  $l$  and  $k$ :** Our protocol publicly releases  $l$  independent bits and also the corresponding encoding scheme. We are assuming the compressed sensing lower bound of requiring, at least,  $O(s \log \frac{\mathbb{D}}{s})$  measurements for any reasonable recovery of the original data vectors. Here  $s$  denotes the number of non-zeros of the data vectors. Therefore, as long as  $l \ll O(s \log \frac{\mathbb{D}}{s})$ , we can be sure due to the compressed sensing lower bound, that our protocol is secure. For real high-dimensional setting  $O(s \log \frac{\mathbb{D}}{s})$  is a reasonably big number.

The choice of  $k$  is dependent on our definition of near-neighbors. Suppose, the application at hand considers any pair of point  $x, y$  with  $Sim(x, y) < s_0$  as non-neighbors, for some problem-dependent choice of  $s_0$ . Our secure hashing scheme should not distinguish  $x$  from  $y$  with any other random point. We can conveniently do that by choosing  $k$  large enough that ensures:

$$\frac{1}{2} \leq Pr(h_{sec}(x) = h_{sec}(y)) \leq \frac{1}{2} + \epsilon, \quad (5.16)$$

for all points  $x$  and  $y$  with  $Sim(x, y) < s_0$ . Noting that  $Pr(h_{sec}(x) = h_{sec}(y))$  is a function of similarity, we can obtain the desired expression easily.

For example, with secure MinHash we have this condition satisfied if  $k \geq \frac{\log 2\epsilon}{\log s_0}$ . For secure SimHash, we need to choose  $k \geq \frac{\log 2\epsilon}{\log(1 - \frac{\cos^{-1}(s_0)}{\pi})}$ .

For more security standards, similar to public keys in RSA, the indices of minhash can be changed periodically and it does not change the cost of the protocol very much

because the authorized users can simply communicate the new indices or download them via a server with identification. Basically, these indices are just a few integer numbers and the overhead of downloading them regularly is negligible. This approach will increase the security of our suggested system since the probability of recovering hash indices from the hashes would decrease significantly.

**Fast Hashing:** One can observe that our 1-bit secure hash requires  $k$  independent LSH evaluations. This can be expensive. However, with recent success in creating near constant time fast LSH computation procedure [76, 77] this is not at all a concern.

### 5.2.6 Some Remarks: Impossibility

The ideal secure LSH scheme would have the collision probability ( $P_c$ ) of Figure 5.5 for 1-bit hash. As can be seen, given a threshold (e.g., 0.8 in Figure 5.5), the hash of two arbitrary points will have the discrete collision probability of either  $\frac{1}{2}$  or 1. This scheme is ideal because it follows the two favorable properties: (i) if two points are similar, they will have the same hash outcome while not revealing any information about how similar they are. The hash is then used to report them as neighbors. (ii) if two points are farther than a threshold, their hashes are not different than two random bits with a collision probability of  $\frac{1}{2}$ . Please note that in this case all information is masked and no attacker can use triangulation attack to find the original input from the hash value.

Here, we prove that ideal secure LSH cannot exist. For simplicity and without using any generality we prove this fact using the distance terminology (1 - similarity). Consider three different points  $A$ ,  $B$ , and  $C$  in input space with the following condition: the distance between  $A$  and  $B$  be  $\delta$ . To choose  $C$ , among all points that have distance  $\delta$  from  $B$ , we choose the one that has distance  $2 \cdot \delta$  from  $A$ . Let  $\delta$

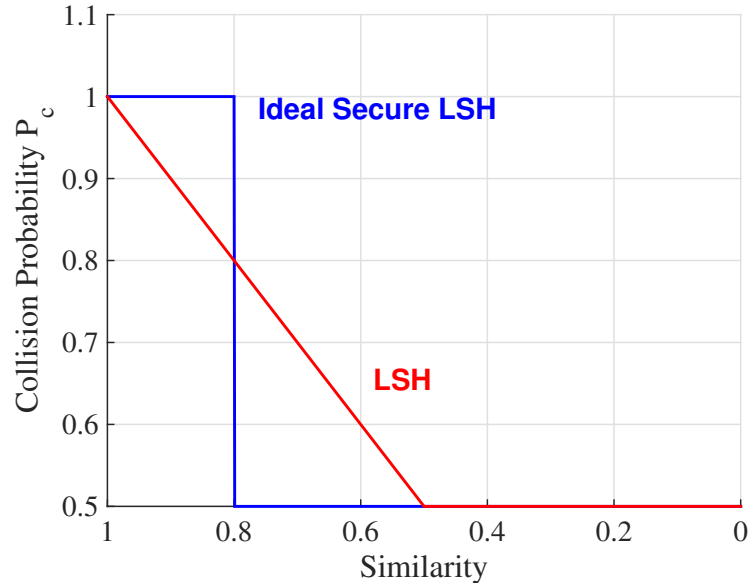


Figure 5.5 : The collision probability of a 1-bit hash for traditional LSH and ideal secure LSH with threshold 0.8.

be a value close but less than the threshold. From the Figure 5.5 we know that the collision probability for the hash values of  $A$  and  $B$  should be 1. We denote it by  $P_{c,AB}$ . Following the same notation,  $P_{c,BC} = 1$ . Since the collision probabilities of  $B$  with both  $A$  and  $C$  are 1, the probability of collision of  $A$  and  $C$  should also be 1. This is due to the fact that the 1-bit hash value of  $B$  should be equal to one's of  $A$  and  $C$ , therefore, the hash value of  $A$  and  $C$  should always be the same and this corresponds to the collision probability of one. However, this is a contradiction since from Figure 5.5 we know that the  $P_{c,AC} = 0$  because the distance between  $A$  and  $C$  is bigger than the threshold and this completes our proof. This proof is valid for  $\forall \delta > 0$  meaning that the flat region cannot exist for any range of distance (or similarity).

### 5.3 Direct Applications

**Privacy-Preserving Authentication System:** Consider an authentication system where a user is authorized by the server, if her information match, at least, one of the profiles in the database. In case this information is a pair of user-name and password, the *exact* match is required. For security reasons, the server should not store the raw user-names and passwords and it should store only the hashes of this information. User can then send the hash of her information and upon an exact match, she is authenticated. However, when the authentication information is not exact (e.g. fingerprints, face images, or other biometrics) the task is to check whether the user's information is *close enough* to at least one of the authorized profiles in the database. In this situation, it is critical that the information is not revealed to anyone else since it can cause multiple threats, implying that the process should be performed in privacy-preserving fashion. However, for practical issues, this process cannot take more than a few seconds. By using our framework, user can send the hash of her information to the server and can be authenticated with our privacy-preserving sub-linear protocol in less than a second.

**Privacy-Preserving Recommender System:** Recommender systems are widely used to suggest similar products (e.g. Amazon), movies (e.g. Netflix), and etc to the users based on their profiles. Since these systems use user's profiles and histories to make suggestions, repeatedly it is considered as privacy-invasive. However, state-of-art privacy-preserving solutions cannot scale to datasets as big as real-world applications. In contrast, we propose a new scheme. In our scheme, user's data are stored on her machine. Depending on the profile's similarity metric, user can use the corresponding LSH family that preserves that similarity and compute the hash of her profile. Then server uses this hash and finds the similar profiles by running

near-neighbor search on database in sub-linear time and report the similar items. For example, a sample user profile can be a vector of numbers rating all available movies from 1 to 5. In practice, not all

**Privacy-Preserving Nearby Friends:** Facebook has added a new feature which is called “Nearby Friends”. This option allows a user to see their nearby friends if both user and her friends share their location in real-time. In fact, the location of all engaging parties is revealed to the Facebook server. Two friends are considered as *nearby* if they are closer than a certain distance. While having privacy-preserving version of this feature is favorable, it is necessary that the computation still be practical and scalable to a high number of participants. For example, Facebook has more than a billion users, therefore, having an efficient sub-linear solution is crucial. Since our framework only needs a simple hash computation it is well-suited for resource-constrained devices (e.g. mobile phones; account for more than 78% of Facebook’s activities). In terms of communication bandwidth, the user only needs to send  $l$  bits where  $l \sim 1000$ , requiring the user to send less than a kilobyte of data. (also note that all operations are performed in plain-text as opposed to computational privacy approaches where all operations are performed in cipher-text.)

**Privacy-Preserving All Near-Neighbor Search:** In *all* near-neighbor search problem,  $n$  different parties want to find out which parties are close to them given a predefined distance threshold. One solution is that first all pairwise distances between any two parties be computed and then for each party, the distance to other parties be compared with the threshold to check whether they are close or not. Any privacy preserving protocol that relies on pairwise distance computation will incur  $O(n^2)$  complexity (because  $\frac{n^2}{2}$  distances should be computed). Existing works in this domain require the pairwise distance computation, yielding the overall computation

complexity of  $O(n^2)$ . In contrast our framework has the  $O(n^{1+\rho})$  computation complexity with  $\rho < 1$ . Please also note that we have much lower big  $O$  constant because we do not use any encryption.

## 5.4 Integrating with SFE Protocols

Secure Function Evaluation (SFE) protocols can evaluate a function which has multiple inputs from different parties without revealing any party's input to others. Examples are Garbled Circuit (GC) protocol and Fully Homomorphic Encryption (FHE). While SFE protocols are provably secure protocols that can securely evaluate any function (without error or approximation), they suffer from huge computational and communication overhead. In fact, they need multiple rounds of communication with high bandwidth usage. Here, we explain how our framework can be used together with SFE protocols to mitigate their computational overhead and make them practical for real-world applications.

We can put one or multiple SFE protocols on top of our framework. Since applying SFE protocols to a database with million entries is far beyond the capacity of state-of-the-art SFE protocols, our framework can be utilized as a preprocessing step. For instance, we can perform very low-overhead secure clustering using our framework and then given thousand or hundred entities in each cluster, we can perform SFE on the desired cluster. In this way, we can perform any functionality (not just near-neighbor search) on the inputs while preserving the privacy of users where nothing is revealed with provable security guarantees. For example, if we are interested to exactly find the center of clusters of million points, we can securely cluster the data and then perform SFE to calculate the average of all points in each group without any approximation. With this approach, we have utilized the efficiency and scalability of

our framework while securely evaluating an arbitrary function on a massive dataset.



## Chapter 6

### Experimental Evaluations

#### 6.1 P3SM Evaluations

In this section, we summarize and compare the complexity of different algorithms for general SM and its limited variant.

##### 6.1.1 Evaluation Setup

We use Synopsys Design Compiler 2010.03-SP4 to generate our sequential circuits. The timing analysis are done on two similar machines with Intel Core i7-2600 CPU @ 3.4GHz with 12GB RAM on an Ubuntu 15 operating system connected using 1 Gbps Ethernet. In all of the experiments, the GC security parameter is 128-bit and the security failure probability for ORAM is set to at most  $2^{-80}$ . The experiments that take more than  $10^5$  seconds are estimated based on the computational and communication complexity. For sub-linear ORAMs (Circuit ORAM and Square-Root ORAM) we estimated the circuit size of ORAM based on the results in [50]. (This is due to the fact that the Square-Root ORAM has recently been publicly available and the source codes are not available yet.)

##### 6.1.2 ORAM Analysis

Figure 6.1 shows the memory cost for the general SM running for worst case scenario. There are 3 different zones. For very small set sizes ( $< 2^5$ ), the linear ORAM (MUX)

is the most efficient solution. In the next region up to set sizes ( $\sim 2^{12}$ ), the Square-Root ORAM outperforms linear ORAM. For set sizes larger than ( $\sim 2^{12}$ ), the Circuit ORAM outperforms all other solutions. For the NRMP, the set size is roughly  $2^{15}$  and therefore the Circuit ORAM is the most efficient method. However, it takes  $10^8$  seconds ( $\sim 3$  years) time and seems still not practical.

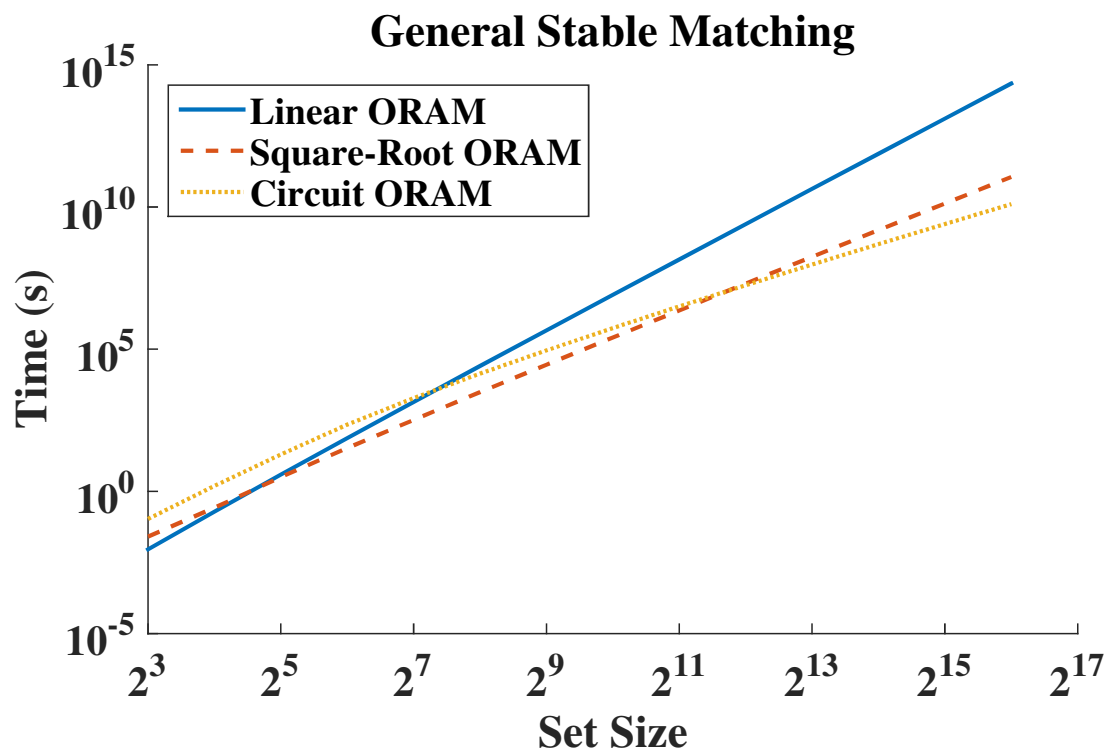


Figure 6.1 : Comparison of memory access cost of different ORAM schemes for general SM without using ETT.

Figure 6.2 shows the memory cost for the general SM using ETT. In this case the number of iterations is reduced by a factor of  $\mathcal{O}(n)$ . ETT makes the algorithm less memory intensive thus initialization cost of ORAM becomes dominant. Since the initialization cost of Square-Root ORAM is less than Circuit ORAM, it outperforms

Circuit ORAM for any set size. Here we have only one breakeven point in which Square-Root starts to outperform Linear ORAM at set size  $2^5$ . For the NRMP ( $2^{15}$  pairs), the Square-Root ORAM is the most efficient method and it takes  $10^6$  seconds ( $\sim 24$  days). This is the first time that the NRMP stable matching can be securely evaluated in less than a month. The previously best solution of [16] reports 47 000 years of computation for SM on a set size 1/4 of NRMP.

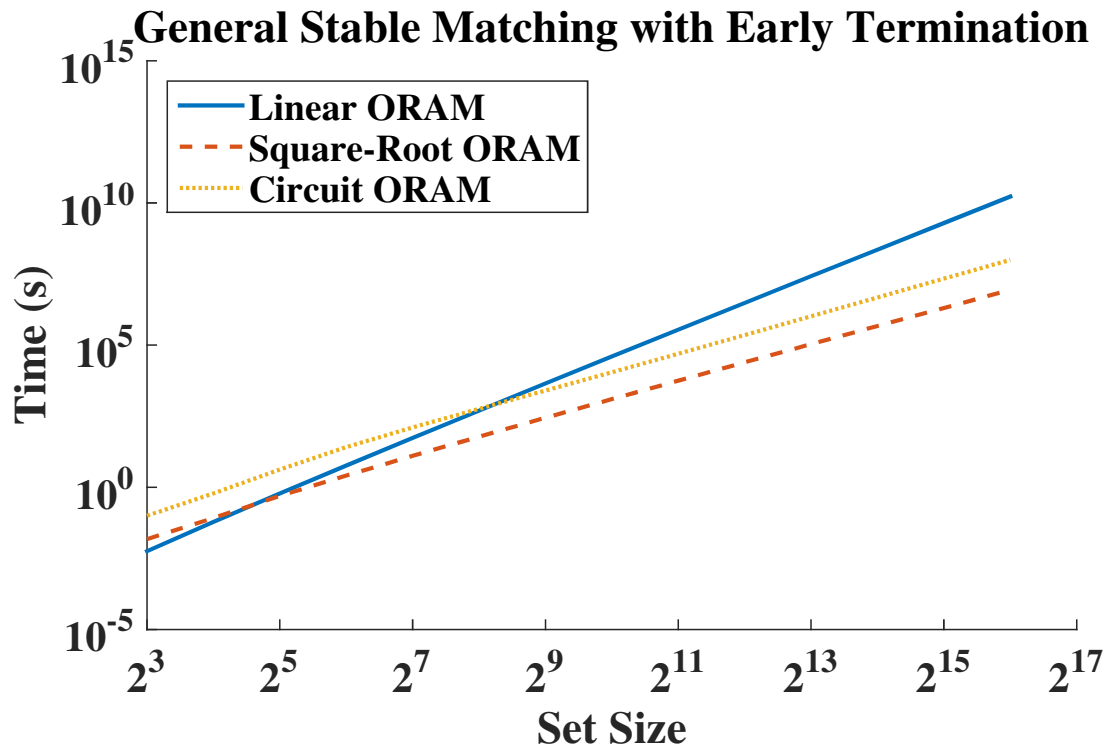


Figure 6.2 : Comparison of memory access cost of different ORAM schemes for general SM using early termination technique.

Figure 6.3 shows the memory cost for limited SM where the number of preferences for each person is limited to  $k = 20$ . The breakeven point between Linear ORAM and Square-Root ORAM is at set size  $2^5$ . In this scenario, the cost of initialization

is as important as the cost of accessing the memory. Hence, Square-Root ORAM outperforms Circuit ORAM for a wide range of set sizes.

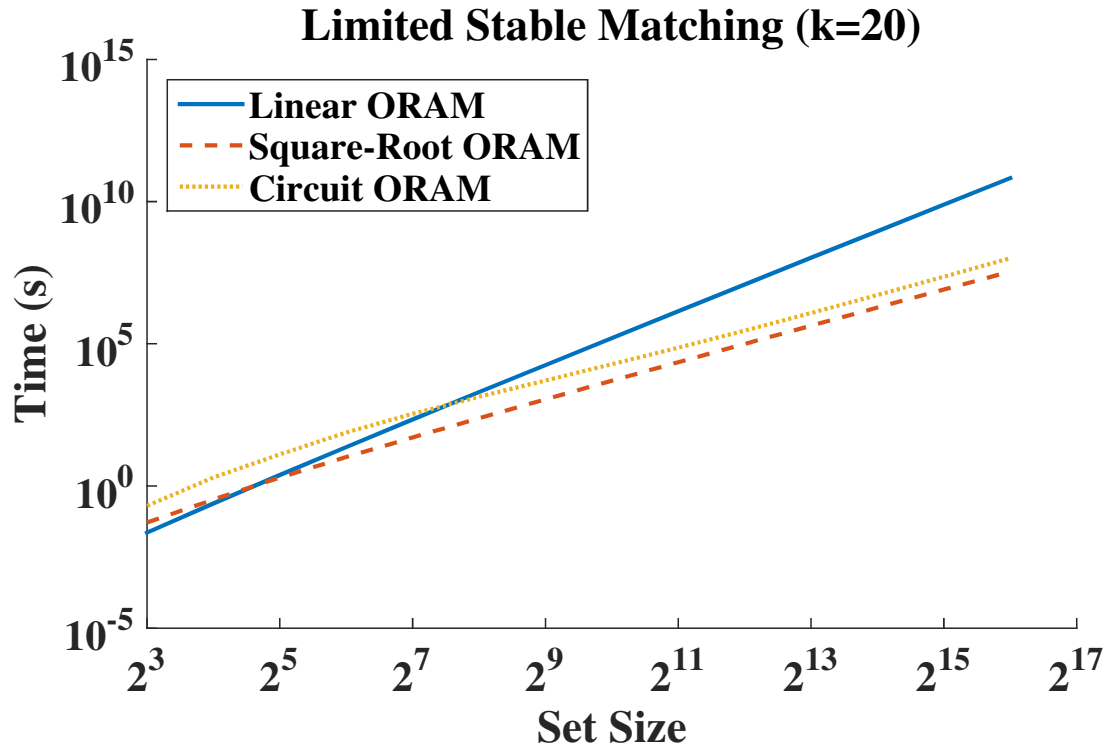


Figure 6.3 : Comparison of memory access cost of different ORAM schemes for limited SM with  $k = 20$ .

### 6.1.3 End-to-End Secure Stable Matching

Table 6.1 shows the total end-to-end execution time and communication cost of general SM for various set sizes when using three different ORAM schemes. For each set size, the best result is shown in bold format. Our scheme can securely compute a stable match for  $8k$  pairs four orders of magnitude faster than the previously best known method of [16].

Table 6.1 : Timing and communication results without using early termination technique.

Set Size	Linear ORAM		Square-Root ORAM		Circuit ORAM	
	Time	Comm.	Time	Comm.	Time	Comm.
8	<b>9.1 ms</b>	<b>1.22 MB</b>	25.7 ms	3.44 MB	107.1 ms	14.37 MB
128	23.21 min	186.92 GB	<b>5.43 min</b>	<b>43.78 GB</b>	31.68 min	255.18 GB
2k	4.54 y	19.25 PB	<b>26.59 d</b>	<b>308.4 TB</b>	36.95 d	428.54 TB
8k	-	-	5.62 y	23.79 PB	<b>3 y</b>	<b>12.71 PB</b>

Table 6.2 shows the total end-to-end execution time and communication cost of general SM with ETT for various set sizes when using three different ORAM schemes. For each set size, the best result is shown in bold format. The computation and communication cost are further decreased, for example for set size  $8k$  by three orders of magnitude compared to general SM without ETT.

Table 6.2 : Timing and communication results using early termination technique.

Set Size	Linear ORAM		Square-Root ORAM		Circuit ORAM	
	Time	Comm.	Time	Comm.	Time	Comm.
8	<b>5.69 ms</b>	<b>764 KB</b>	15 ms	2.01 MB	100.5 ms	13.48 MB
128	54.4 s	7.3 GB	<b>12.75 s</b>	<b>1.71 GB</b>	2.12 min	17.11 GB
2k	4.05 d	46.99 TB	<b>1.56 h</b>	<b>753.83 GB</b>	13.79 h	6.66 TB
8k	306.53 d	3.55 PB	<b>1.25 d</b>	<b>14.54 TB</b>	11.96 d	138.7 TB
32k	-	-	<b>23.47 d</b>	<b>272.2 TB</b>	251.65 d	2.91 PB

Table 6.3 shows the total end-to-end execution time and communication cost of limited SM with  $k = 20$  for various set sizes when using three different ORAM schemes. For each set size, the best result is shown in bold format.

Table 6.3 : Timing and communication results for limited SM where  $k = 20$ .

Set Size	Linear ORAM		Square-Root ORAM		Circuit ORAM	
	Time	Comm.	Time	Comm.	Time	Comm.
8	<b>22.76 ms</b>	<b>3.05 MB</b>	51.8 ms	6.95 MB	200 ms	26.84 MB
128	3.62 min	29.2 GB	<b>50.97 s</b>	<b>6.84 GB</b>	5.73 min	46.15 GB
2k	16.21 d	187.98 TB	<b>6.23 h</b>	<b>3.01 TB</b>	20.2 h	9.76 TB
8k	3.35 y	14.21 PB	<b>5.01 d</b>	<b>58.11 TB</b>	13.94 d	161.74 TB
32k	-	-	<b>93.77 d</b>	<b>1.08 PB</b>	264.89 d	3.07 PB

#### 6.1.4 Discussion

**What Would Be Leaked by Early Termination?** Although using the Finish Signal may leak the number of proposals  $R$ , there are plenty of different cases when this setting could be employed. Such as when each matching authority agrees not to reveal the number of real proposals because no one has any interest to do so. In this case there is no desire to reveal the number of proposals. If we add a random initiator to MSC and the next man selection algorithm, we can achieve the randomness which will make  $R$  nondeterministic. In this case, even for the same inputs, each time the circuit travels a different path and as a result,  $R$  is different and does not convey any useful information. On the other hand, leaking the number of total proposals needed to reach the SM could convey some information. For example, if the matched partners be their first choices, then the algorithm will terminate immediately after the first iteration. This is very different from the case when each individual is matched with her last choice, in which case the algorithm will terminate after  $\mathcal{O}(n^2)$  iterations.

Therefore, knowing  $R$  can give some information about the quality of the assignment but nothing about the input itself. This problem can be mitigated by running the protocol for at least the statistical worst case (see Figure 3.4). In order to avoid

unnecessary iterations and terminated the protocol,  $R$  is the least information possible to leak. In a nutshell, this variant of our protocol is a trade-off between privacy and efficiency where we have an unprecedented performance improvement but the total number of proposals is leaked. Note that the early termination technique is optional and can be avoided.

## 6.2 GenMatch Evaluations

Our experimental results are performed using two processes on Intel Core i7-2600 CPU @ 3.4GHz with 12GB RAM on a 64-bit Ubuntu 14 operating system. The security parameter in our setup (encryption key length) is 128-bit. The circuits are synthesized by the Synopsis Design Compiler. Table 6.4 shows the results. Since there is no similar GC-based work with an application performing on a database of genome data, we could not compare our results with previous works.

Database Size	# of XORs	# of Non-XORs	Total Gates	Total Garbled Tables	Communication (MBytes)	Time (s)
10	438	400	838	4,000	1.0	0.07
100	447	412	859	41,200	10.5	0.62
1,000	457	424	881	424,000	108.5	5.79
10,000	433	459	892	4,590,000	1,175.0	63.20
100,000	436	474	910	47,400,000	12,134.4	546.09
1,000,000	439	489	928	489,000,000	125,184.0	5,132.25

Table 6.4 : Number of XOR and non-XOR gates of circuit and total timing and communication for different size of database.

## 6.3 S-LSH Evaluations

In this section we provide our experimental results performed on largest MovieLens\* data. In this dataset  $\sim 247k$  users have rated the  $\sim 150k$  movies on the 1 to 5 integer scale where 5 means the most desired.

To the best of our knowledge there does not exist any encryption-free privacy-preserving near-neighbor search framework performing on an arbitrary similarity metric (e.g. Jaccard similarity and Cosine similarity). This is the first randomized embeddings framework which prevents LSH from revealing all pairwise similarities. Hence, there are no comparable baselines. We provide strong evaluations to validate our claims related to the security and efficiency of our protocol.

### 6.3.1 Sanity Check

Figure 6.4 shows the number of bit matches of secure hashes as the similarity varies. This is the number of bit matches between a hash of a randomly selected profile and all other secure hash of other user profiles. The hashing scheme here is MinHash with Jaccard similarity. As can be seen, when the similarity is less than 0.85, the number of bit matches of given two hashes does not reveal any information about the underlying profile attributes because it can correspond to any two profiles with similarity between 0 to 0.85. The hashing parameters are  $l = 300$  and  $k = 10$ . The dotted horizontal red line shows the  $\frac{l}{2}$  (150) bit matches which is also the expected number of bit matches for any two random strings.

Figure 6.5 shows similar plot for SimHash with  $l = 2^{10}$  and  $k = 10$ . The horizontal dotted red line shows the expected average number of bit matches for any two random

---

\*<http://grouplens.org/datasets/movielens/>



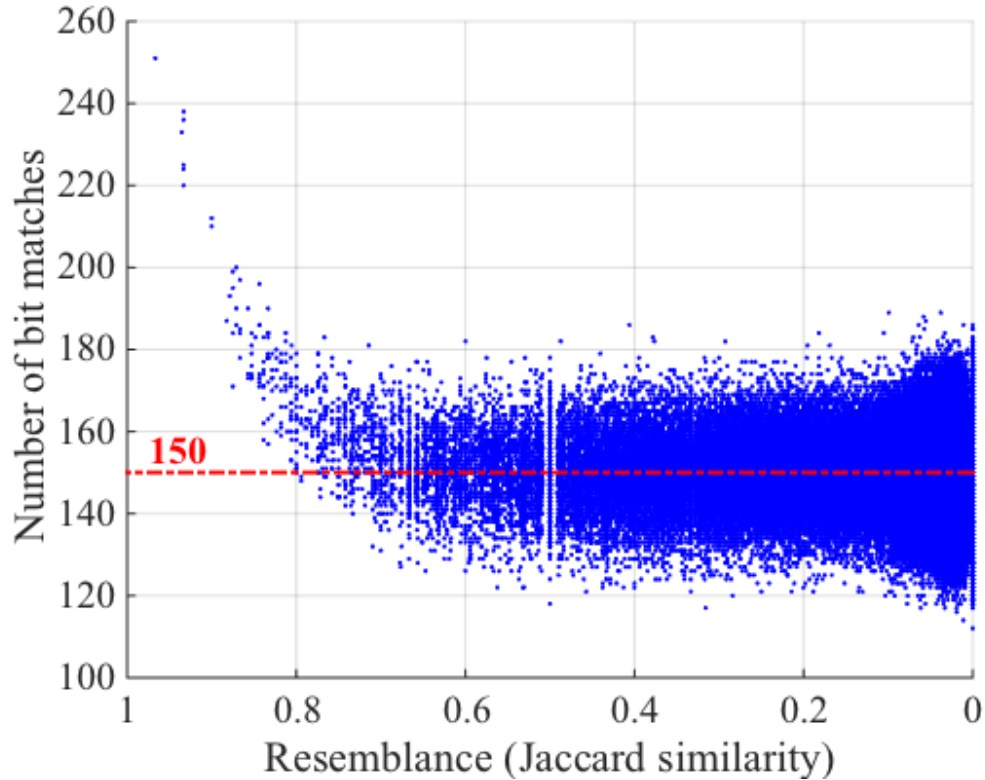


Figure 6.4 : Number of bit matches between a given query point and all other points in the MovieLens dataset as the function of Jaccard similarity where  $l = 300$  and  $k = 10$ .

strings of length  $l$ . We can clearly see that when the similarity of two points are high enough, they can be easily identified.

### 6.3.2 Performance Analysis

In this section, we illustrate the efficiency of our framework. Table 6.5 shows the precision, recall, and speed-up of our framework for two popular hash schemes, namely MinHash and SimHash for two different choices of  $k$ . Precision and recalls are cal-

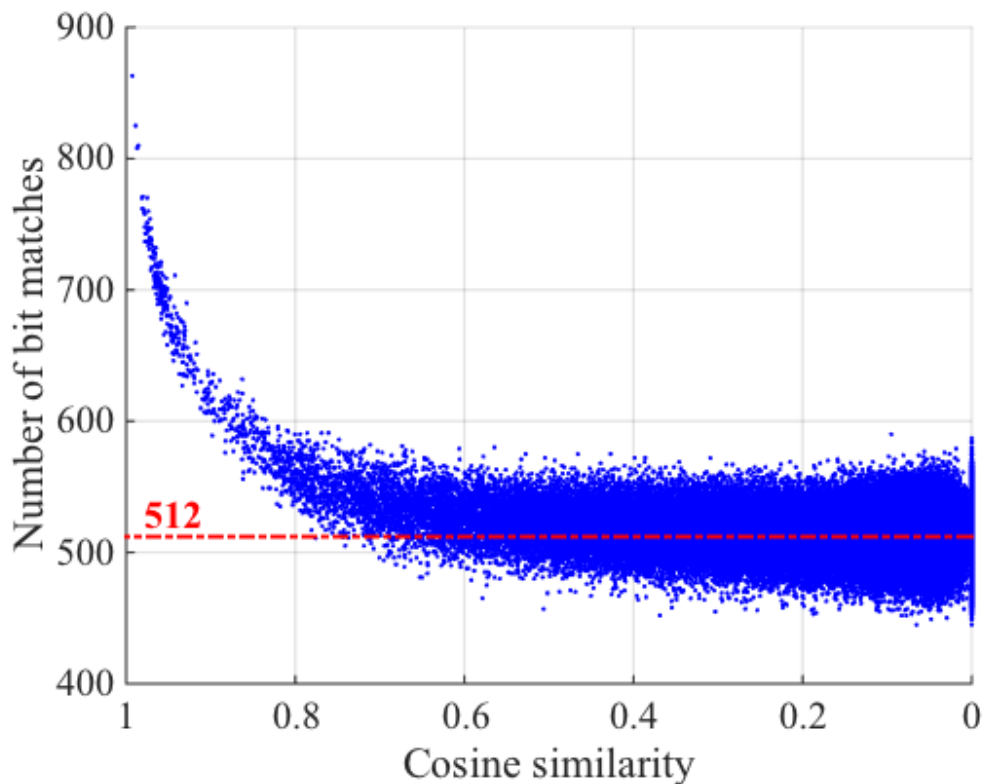


Figure 6.5 : Number of bit matches between a given query point and all other points in the MovieLens dataset as the function of cosine similarity where  $l = 2^{10}$  and  $k = 10$ .

culated based on the result of our secure sub-linear near-neighbor search.  $N$  shows the number of near-neighbors that we are interested to find. Speed up is compared with the linear scan through all hashes in order to find the top  $N$  profiles with the highest similarity. The parameters  $L$ ,  $b$ , and  $l$  are tuned to deliver the best precision and recall.

Figure 6.6 shows the result for secure SimHash. The  $n = 50000$  points are randomly generated in  $\mathbb{R}^4$  and hashes are computed with parameters  $l = 2^{10}$  and  $k = 10$ . Again the points with high cosine similarity (positive and negative) the number of

Table 6.5 : Experimental results for our near-neighbor search.  $k$  is the degree of transformation. The last column shows the speed up over linear scan on hashes.

Hashing Scheme	$k$	$N$	Precision	Recall	Speed up
MinHash	5	50	75.1	90.1	9.34
	10	20	80.2	84.3	127.93
SimHash	5	50	72.2	86.6	8.55
	10	20	89.3	93.8	10.61

bit matches of pair of hashes clearly separates them from other points.

Figure 6.7 shows the result for secure SimHash. The  $n = 10000$  points are randomly generated in  $\mathbb{R}^3$  and hashes are computed with parameters  $l = 2^{10}$  and  $k = 10$ . Again the points with high cosine similarity (positive and negative) the number of bit matches of pair of hashes clearly separates them from other points.

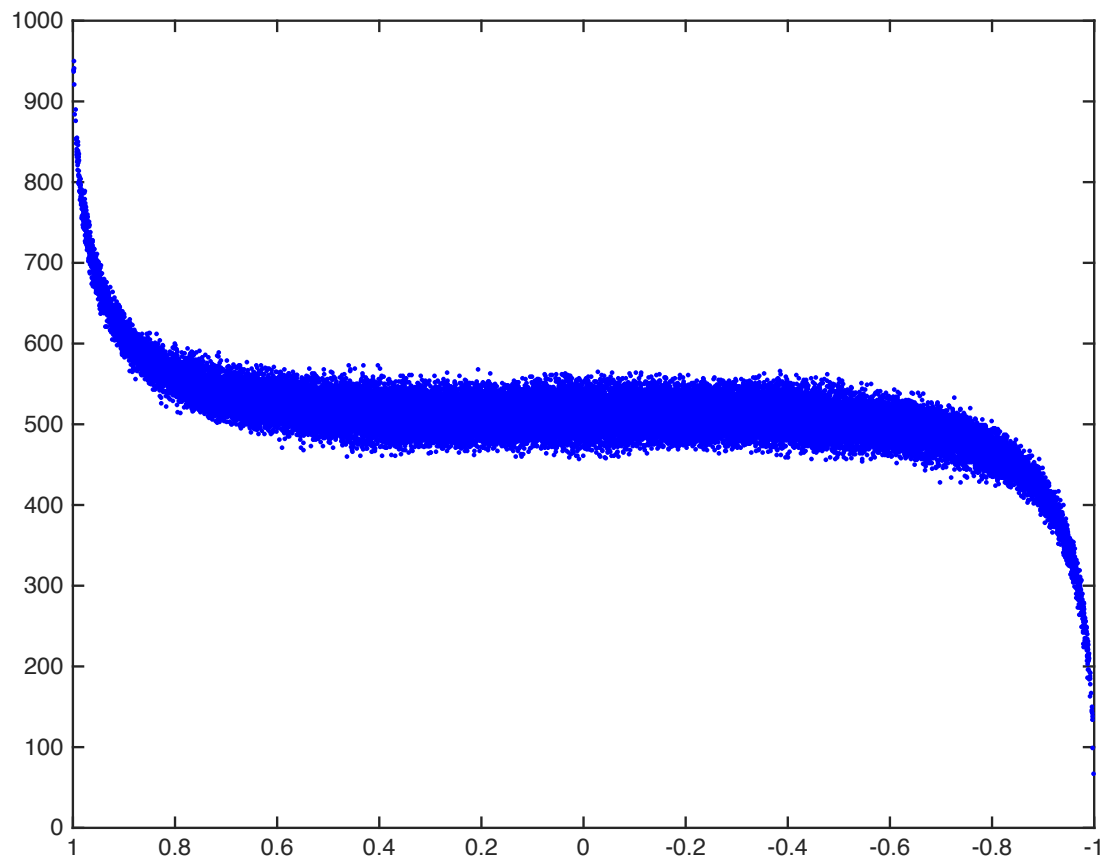


Figure 6.6 : Number of bit matches between randomly selected point and all other points in the uniform randomly generated dataset as a function of cosine similarity where  $l = 2^{10}$  and  $k = 10$   $n = 50000$ ,  $\mathbb{D} = 4$ .

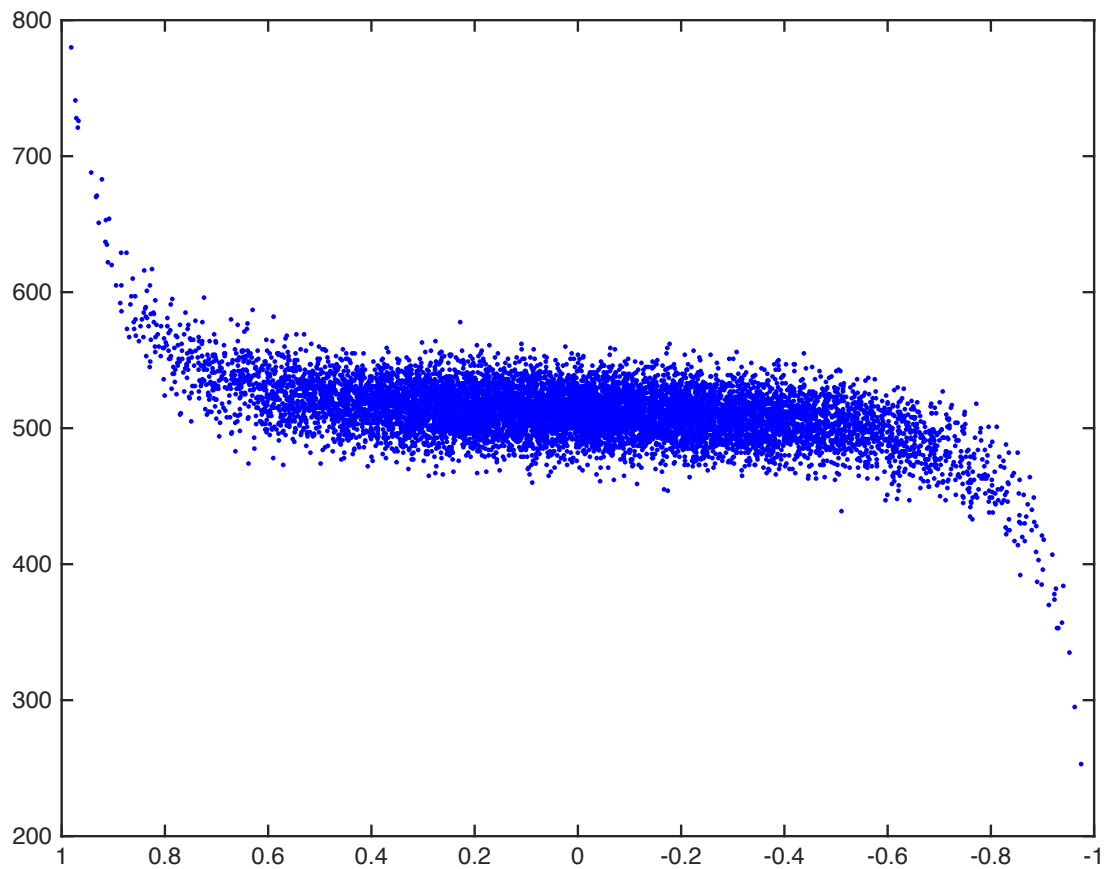


Figure 6.7 : Number of bit matches between randomly selected point and all other points in the uniform randomly generated dataset as a function of cosine similarity where  $l = 2^{10}$  and  $k = 10$   $n = 10000$ ,  $\mathbb{D} = 3$ .

## Chapter 7

### Related Work

In this chapter, we explore state-of-the-art works in each domain of our study. In Section 7.1 we bring previous privacy-preserving stable matching protocols. We survey secure DNA compatibility testings in Section 7.2. Privacy-preserving near-neighbor search algorithms are covered in Section 7.3.

#### 7.1 Privacy-Preserving Stable Matching Protocols

Golle [13] was the first to develop privacy-preserving SM. He persuasively illustrates that implementing such an algorithm has a great practical impact. In his framework, he devises a variant of the classic Gale-Shapley algorithm with some techniques for concealment. Each party should send the encrypted preference list to some honest-but-curious matching authorities. During the algorithm men and women are divided into two disjoint groups, those who are engaged and those who are free. Then,  $m$  fake men are added and are engaged to women at the beginning of the algorithm and this enables some appealing concealment properties, such as that the number of engaged men and free men are always constant and this could prevent any information leaking of intermediate changes of free and engaged men sets. Golle then defines a bid as an encrypted representation of the preference of one man for a woman in addition to some book-keeping information. There are free and engaged bids, a bid paired up with a woman. The Algorithm follows 4 steps for  $R$  iterations beginning with initial

bids to reach a stable match by resolving a conflict between bids at each time. Those four steps are (i) randomly choosing a free bid and opening it mutually by matching authorities, (ii) finding a conflict bid since there is always exactly one conflicting bid, (iii) resolving the conflict, and (iv) mixing bids internally and externally. Golle uses an additively homomorphic semantically secure threshold public key encryption scheme like a threshold version [78, 79] of Paillier [80] and re-encryption mix networks to implement this. Golle argues that the algorithm terminates after  $n$  iterations and reaches a stable match between  $n$  men and  $n$  women. The complexity then is dominated by  $\mathcal{O}(n^3)$  modular exponentiations and the corresponding communication complexity is also  $\mathcal{O}(n^2 \text{polylog}(n))$ .

Table 7.1 : Different protocols for performing secure SM and corresponding complexities, where  $n$  is the size of each group and  $\nu$  is the number of matching authorities. Our protocol is faster by orders of magnitudes as it uses efficient Symmetric-Key operations instead of costly Public-Key operations.

Protocol	Computation Complexity	Communication Complexity	Round Complexity
Golle [13]	$\mathcal{O}(n^5)$ Public-Key Operations	$\mathcal{O}(\nu n^5)$	$\mathcal{O}(n^3 \text{polylog } n)$
Franklin et al. [14]	$\mathcal{O}(n^4 \sqrt{\log n})$ Public-Key Operations	$\mathcal{O}(\nu n^3)$	$\mathcal{O}(n^2 \text{polylog } n)$
Franklin et al. [15]	$\mathcal{O}(n^4 \text{polylog } n)$ Public-Key Operations	$\mathcal{O}(\nu n^2)$	$\mathcal{O}(n^2 \text{polylog } n)$
Ours without using ORAM	$\mathcal{O}(n^4 \log n)$ Symmetric-Key Operations	$\mathcal{O}(n^4 \log n)$	$\mathcal{O}(1)$
Ours using ORAM	$\mathcal{O}(n^2 \log^3 n)$ Symmetric-Key Operations	$\mathcal{O}(n^2 \log^3 n)$	$\mathcal{O}(1)$
Keller et al. [16]	$\mathcal{O}(n^2 \log^3 n)$ Symmetric-Key Operations	$\mathcal{O}(n^2 \log^3 n)$	$\mathcal{O}(1)$
Ours using ORAM and ETT	$\mathcal{O}(n^2 \log^2 n)$ Symmetric-Key Operations	$\mathcal{O}(n^2 \log^2 n)$	$\mathcal{O}(1)$

However, Franklin et al. [14] show that Golle’s proposal is not promising and in the worst case, his algorithm will be executed  $n^2$  iterations. This results in a com-

putation complexity of  $\mathcal{O}(n^5)$  modular exponentiations and  $\mathcal{O}(\nu n^5)$  communication complexity, where  $\nu$  is the number of matching authorities. They also develop another solution for this problem based on Golle’s framework but also add  $n$  fake women. There are some modifications to the original framework. They added Private Information Retrieval (PIR) for reading and writing privately in a shared database. PIR is implemented using the protocol of Naor and Nissim [81] which is based on the Oblivious Transfer and requires  $\mathcal{O}(\text{polylog } n)$  communication and is used a constant number of times in each iteration. Franklin et al. use the GC protocol to compare the preference of women and also to increment the number of times that each man has proposed. The communication complexity is dominated by re-encryption mixnets just like Golle’s which results in  $\mathcal{O}(\nu n^3 \text{polylog } n)$  communication complexity. Yet, computation complexity is dominated by accessing the database. In each iteration, the database access takes  $\mathcal{O}(n^2 \sqrt{\log n})$  work and as a total the computation complexity is  $\mathcal{O}(n^4 \sqrt{\log n})$  public key operations. In that work they also optimize the protocol when there are exactly 2 matching authorities and they achieve a better performance as listed in Table 7.1. However, this protocol is complicated and no one has attempted to implement it.

Naor et al. [68] suggest an architecture for privacy-preserving protocols for mechanism design and they mention SM as one of the related applications. They argue that the complexity of implementing their architecture for SM problem depends on a combinational circuit which implements the SM algorithm. They did not implement such a system nor designed such a circuit. Naor et al. suggest that the classical Gale-Shapley algorithm necessitates the usage of indirect addressing of a RAM and that translation into a circuit is inefficient.

Franklin et al. [15] develop an efficient multiparty Look-Up Table (mLUT) pro-



protocol and suggest that one can design a secure SM system by implementing the algorithm of [14, Section 5] into a circuit with access to a RAM. Especially in the multiparty setting, mLUT for array/matrix access reduces the complexity of such a setting to the situation when we have 2 matching authorities as shown in Table 7.1. However, this system has only been suggested but has not been implemented yet.

[16] is the only work that has reported implementing secure SM using GC and oblivious memory. Without presenting implementation details, they reported that in the worst case, the general SM with  $8k$  pairs can be done in  $1.5 \cdot 10^{12}$  seconds. The complexity of related work is reported in Table 7.1.

## 7.2 Secure DNA Testing Algorithms

Genetic testing and its digitization raise some important privacy and ethical concerns. The authors in [82] summarize the issues and put forth the main challenges for the research community. They not only talk about technical issues such as efficiency, usability, and pitfalls of genetic testing, but also about policies that should be made into laws when handling genome data. The work in [83] answers some of the questions in [82] and provides a framework for the secure handling of genome data.

There are different technical approaches to *in silico* privacy-preserving genetic testing. In [84], the authors create android applications for genetic testing methods such as paternity testing, ancestry testing, and personalized medicine. They are concerned with the efficiency and usability of the applications and conclude that this is an area worth exploring. They use hash function based Private Set Intersection Cardinality (PSI-CA), Authorized Private Set Intersection (APSI) and additively homomorphic encryption of Secure Hamming Distance (SHD) for each of the applications and implement them on an android based platform. They show that comparing the

whole genomes for compatibility is both unnecessary and computationally infeasible for a mobile device. This makes the pre-processing an important and aiding step to extract the information that is needed for a specific test. This pre-processing stage helps to carry out the test on a mobile device. They propose  $P^3MT$  protocol to test for HLA-B mutation which is important in determining the sensitivity to a drug used in HIV treatment. Our work differs from their work in terms of the framework of the test as well as the application. They work with HLA data but their test is limited to a particular mutation as an example for personalized medicine in contrast with the database-based compatibility test described in this thesis.

The work in [23] describes a GC-based approach for measuring the similarity between two genome sequences by performing secure edit distance computation. This computation is performed on raw genome data and is not scalable. It utilizes the full genome which is unnecessary especially when only a small part of the genome data needs to be tested.

The authors of [85] propose homomorphic encryption based identification, paternity and ancestry testing exploiting the Short Tandem Repeat (STR) property exhibited by genetic sequences. Another proposal in [86] uses homomorphic encryption to perform queries on encrypted database of genome sequences while preserving the identities of each of the individuals. This is similar to our setting but the application involves testing for Single Nucleotide Polymorphisms (SNPs) in a database for research purposes involving the whole genome sequences without pre-processing. The authors in [87] describe a similar situation but consider an implementation of a privacy-preserving forensic DNA database. Each DNA in the database is encrypted using a part of its DNA sample and can only be decrypted if the person submitting the query has the appropriate key generated from the suspect's DNA. While this works

for a forensic database application, it cannot be extended to other genetic testings except for those involving identity protection.

The work in [88] concentrates on searching a finite length DNA fragment in another DNA template protecting the privacy of both parties involved. They designed a protocol that allows an oblivious evaluation of a finite state machine which has a linear relation between communication complexity on one side and the number of states and length of input data on the other side. Similar to some of the previously mentioned works, they work with raw genome data making it hard to scale and unnecessarily complex to implement.

In contrast to aforementioned works, we design a scalable database-based secure DNA compatibility testing utilizing the GC protocol. We adopt TinyGarble platform [18] to implement our method.

### **7.3 Privacy-Preserving Near-Neighbor Search Algorithms**

All the works addressing privacy-preserving near-neighbor search can be categorized as three inherently different approaches [89]: 1) Computational Security, 2) Information-Theoretic Security, and 3) Randomized Embeddings. In this section we compare these approaches and provide brief review of each of these approaches.

#### **7.3.1 Computational Security**

Computational security relies on the security of cryptographic tools which are then based on hardness of some problems in number theory such as factorization of large numbers or inverting a discrete logarithm. The security of these tools are proven for computationally bounded adversaries, meaning that there does not exist any efficient algorithm that can break these tools in a reasonable amount of time. Gar-

bled Circuit and Homomorphic Encryption are most common used protocols in this domain. While these protocols provide provably secure solution with precise results, they face critical efficiency and scalability issues since they are computationally intensive.

The computational overhead comes from the fact that every single bit of data involved in the near-neighbor search process has to be encrypted somehow to guarantee the security of these schemes. The other disadvantage of these protocols is communication overhead. They require user to interact multiple times and the process the data back and forth. Some of these protocols (e.g. GC) requires two non-colluding servers which if they collude, all of information is compromised. As a result computational security solutions cannot be deployed for real-world applications that can preserve the privacy of users and scale to modern big-data.

### **7.3.2 Information-Theoretic Security**

Information-theoretic security, unlike computational security, is secure against computationally unbound adversaries. As mentioned in Section 1.3, the near-neighbor search task can be seen as two disjoint sub-problems, first finding the pairwise distance between query data and every other data in the database and second finding the minimum of these distances. While computing the pairwise distances is a relatively easy task in this domain, finding the minimum distance is not and requires using some cryptographic tools which again makes these solutions.

### **7.3.3 Privacy-Preserving Randomized Embeddings**

Randomized embeddings are very useful tools for reducing the dimensionality of original data while preserving the geometry properties of data. These methodologies

are typically based on Jonson-Lindenstrauss (J-L) [37] or Locality Sensitive Hashing (LSH) [75]. However, these methods do not preserve the privacy of user by their own. But they can be used in conjunction with one or two secure protocols to make an overall secure solution. For example, [90] uses J-L embedding together with homomorphic encryption for securely computing  $l_1$  and  $l_2$  distances.

It turns out that the existing LSH scheme for  $l_2$  distances, based on  $p$ -stable distribution, has an interesting property. The collision probability of this scheme rapidly decays to  $\frac{1}{2}$  [91] due to the addition of random dither inherent in the hashing scheme itself. This leads to security against triangulation attack because we cannot really estimate the distance for very far points [92]. Such a scheme when remapped to 1-bit naturally produces a secure 1-bit hashing scheme for  $l_2$  distance. However this addition of random dither makes the hashing scheme inferior to other popular hashing schemes such as MinHash and SimHash [92]. The addition of dither (noise) is a popular choice for achieving privacy, however, it is known to reduce utility [91]. Moreover, it does not lead to secure hashing schemes for popular similarity measures, such as Cosine similarity or Resemblance (Jaccard similarity).

In [93], authors propose a mechanism to search over encrypted data. They utilize the useful properties of LSH to build their scheme. However, they use encryption protocols to prevent the information leakages, making the overall system computationally expensive.

## Chapter 8

### Summary

This thesis proposes novel techniques and algorithms for achieving privacy-preserving computing protocol that can be efficient and scalable for real-world applications. Previous work in this domain suffer from huge overheads and they are of limited practical usage.

The first part of this thesis introduces several novel techniques for making well-known Garbled Circuit protocol more efficient and scalable. To demonstrate the effectiveness of these techniques, we have done extensive experiments on a bank of genome with million different DNA profiles. We introduced the first practical and scalable realization of a provably secure HLA matching used in organ transplantation donor compatibility test under the honest-but-curious attack model. Our approach utilizes circuit optimization and logic synthesis for finding a scalable implementation of HLA matching in the TinyGarble framework. We demonstrated an end-to-end implementation of the system. Our results show that the methodology is highly efficient, and it takes only a few hours to perform the matching on a database of a million participants with pre-processed data. These results are scalable, and way more practical than the state-of-the-art in this field, and they enable a range of new applications for privacy-preserving genetic testing. We have also studied one of the most difficult privacy-preserving tasks called Stable Matching. This is the first work which makes secure SM feasible for real-world applications such as the National Residency Matching Program (NRMP) with  $32k$  pairs in a reasonable time. Our

approach is four orders of magnitude faster than the previously best-known method. The state-of-the-art paper reports running time of  $1.5 \cdot 10^{12}$  seconds, i.e., almost 47 000 years! for 8 192 pairs of individuals engaging in the matching process, whereas in our experiment it took only a day to finish.

In the second part of this thesis, we proposed a practical privacy-preserving algorithm for large-scale near-neighbor search. We identified the weakness of LSH based approach in preserving privacy and provided a novel solution using a “well-tailored” transformation. The resulting approach hits the sweet spot between efficient search and privacy. We believe that our scheme will be adopted in practice. We provide the formal analysis of our approach and show information theoretic bounds.

# Appendix A

## Triangulation Attack

In Section 5.1 we discussed that if any hashing approach preserves the distance for *any* pair of points, then an attacker can easily find the original location of any point  $q$  given its hash and hence the scheme is not secure. In this section, we explain that even if the distance estimation is not accurate the triangulation attack is still possible to find the location of  $q$ . Following the same notation as Section 5.1, Figure A.1 shows our setup. In practice the distances are not accurate and, therefore, the three circles will make a triangle at the intersection which is noted as triangle  $DEF$ . The median of  $DEF$  is the location of the user. Hence, we only need to find the location of the points  $D$ ,  $E$  and  $F$ . Equations A.4 and A.5 show how to compute the location of point  $F$  (the equations are provided from [94]). Using the same approach, it is easy to compute the coordinates of  $E$  and  $F$  and the location of  $q$  can be achieved.

If  $A$ ,  $B$ , and  $C$  are drawn randomly from space, the above steps will result in finding distances with more error but an attacker can repeat this process multiple times and each time selecting these three points closer to estimated user's location. Following this process results in a very accurate estimation of user's location.

$$\sqrt{(x_F - x_A)^2 + (y_F - y_A)^2} = d_A \tag{A.1}$$

$$\sqrt{(x_F - x_B)^2 + (y_F - y_B)^2} = d_B \tag{A.2}$$

$$\sqrt{(x_F - x_C)^2 + (y_F - y_C)^2} \leq d_C \tag{A.3}$$



$$x_F = 1/2p(y_F q + t) \quad (\text{A.4})$$

$$y_F = \frac{1}{p^2 + q^2} \left( pqx_A + y_B p^2 - \frac{1}{2}qt \right) \pm \frac{1}{2} \sqrt{(qt - 2y_A p^2 - 2pqx_A)^2 - s(p^2 + q^2)} \quad (\text{A.5})$$

where,

$$p = x_B - x_A, \quad q = y_B - y_A$$

$$t = d_A^2 - d_B^2 + x_B^2 - x_A^2 + y_B^2 - y_A^2$$

$$s = (4p^2 y_A^2 + t^2 - 4ptx_A + 4p^2 x_A^2 - 4p^2 d_A^2)$$

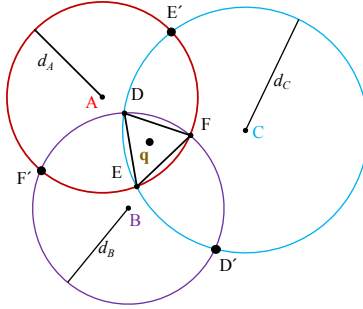


Figure A.1 : Triangle localization. Three selected points are  $A$ ,  $B$ , and  $C$ . The calculated location of  $q$  is the median of the triangle  $DEF$ .

Our approach is secure against this attack because if the attacker follows the aforementioned steps, the comparison of the three random hashes and the user's hash will give him no information since only very close points will have similar hashes not any pair of points. (Unless the randomly selected points accidentally lie in the  $\delta$  distance of the user's location which is highly unlikely and the probability is the

$(\frac{\delta}{d_{max}})^{\mathbb{D}}$  where  $d_{max}$  is the maximum possible distance between any two points in the space and  $\mathbb{D}$  is the dimensionality of the space.) The dimensionality  $\mathbb{D}$  is usually in order of hundreds or thousands which makes the probability too small, therefore, each time attacker chooses a random point, with very high probability this point is farther than the threshold and hence the result of comparing hashes will be almost just noise without any information in it and, as a result, the triangle localization cannot be computed.

## Appendix B

### Circuit for Stable Match Algorithm

In this appendix, the source code for the sequential circuit described for SM algorithm (Section 3.2) is provided. All of the proposed techniques are integrated in this circuit. The circuit is described in Verilog Hardware Description Language.

```

`timescale 1ps / 1ps
/////////////////////////////////////////////////////////////////
// IN THE NAME OF GOD
// Company: Rice University
// Engineer: M. S. Riazi
//
// Create Date:          12:19:33 02/18/2015
// Design Name:          Stable Matching
// Module Name:          mainModule
// Project Name:         stableMatching
// Target Devices:       no device
// Tool versions:        14.6
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
/////////////////////////////////////////////////////////////////
module mainModule
#(
    parameter Kr =10,
    parameter Ks =10,
    parameter S  =10,
    parameter R  =10
)
(
    clk ,
    rst ,
    g ,
    e ,

```

```

o
);

//----- Functions
function integer log2;
    input [31:0] value;
    reg [31:0] temp;
begin
    temp = value - 1;
    for (log2=0; temp>0; log2=log2+1)
        temp = temp>>1;
end
endfunction

//----- Local Parameters
localparam logS = log2(S);
localparam logR = log2(R);
localparam logKs = log2(Ks);
localparam logKr = log2(Kr);
localparam SRounded = 2**logS;

//----- I/O
input wire clk, rst;
input wire [R*Kr*logS-1 + S*Ks*logR-1 + 1:0] e;
input wire [R*Kr*logS-1 + S*Ks*logR-1 + 1:0] g;

wire [R*Kr*logS-1 + S*Ks*logR-1 + 1:0] allInputsXORed;
output wire [R*logS:0] o;

wire [R*Kr*logS-1:0] rPref;
wire [S*Ks*logR-1:0] sPref;

wire [R*logS-1:0] matchList;
wire finish;
wire [S-1:0] sIsMatchWire; // just for transferring information to the testBench
//----- Local wires
wire [logS-1:0] rPrefMatrix [R-1:0][Kr-1:0];
wire [logR-1:0] sPrefMatrix [S-1:0][Ks-1:0];
wire [Kr-1:0] s1Compare, s2Compare;
wire [Kr-1:0] wireXOR1, wireAND1, wireAND2, wireOR1;
wire better;
wire [log2(Ks+1)-1:0] pcS;
wire [logS-1:0] s1;

```

```

wire [logS-1:0] s2;
wire propose;
wire [logR-1:0] r;
wire [SRounded-1:0] canPropose;
wire [SRounded-1:0] encoderInput;
wire [SRounded-1:0] encoderOR1;
wire [logS-1:0] encoderOutput;

//----- Local Regs
reg [log2(Ks+1)-1:0] pc [S-1:0]; //proposal counts
reg [logR-1:0] sInMatch [S-1:0];
reg [S-1:0] sIsMatch;
reg [S-1:0] sIsRunning;
reg [R-1:0] rIsMatch;
reg [logS-1:0] s;
reg [logS-1:0] matchListMatrix [R-1:0];
reg finishAND;

//+++++ Generate Part
genvar i,j;

generate //[0]
  for (i=0;i<R;i=i+1)begin : rPref_asgn1
    for (j=0;j<Kr;j=j+1) begin : rPref_asgn2
      assign rPrefMatrix [i][j] = rPref [logS*Kr*i + logS*(j+1) -1 : logS*Kr*i + logS*j];
    end
  end
endgenerate

generate //[0]
  for (i=0;i<S;i=i+1)begin : sPref_asgn1
    for (j=0;j<Ks;j=j+1) begin : sPref_asgn2
      assign sPrefMatrix [i][j] = sPref [logR*Ks*i + logR*(j+1) -1 : logR*Ks*i + logR*j];
    end
  end
endgenerate

generate //[0]
  for (i=0;i<R;i=i+1)begin : matchList_asgn1
    assign matchList [logS*(i+1) -1 : logS*i ] = matchListMatrix [i] ;
  end
endgenerate

generate //[1] & [2]
  for (i=0;i<Kr;i=i+1)begin : s1_and_s2_Compare
    assign s1Compare[i]= ~(s1^rPrefMatrix [r][i]) ; // [1]
    assign s2Compare[i]= ~(s2^rPrefMatrix [r][i]) ; // [1]

    assign wireXOR1 [i] = s1Compare [i] ^ s2Compare [i]; // [2]
    if (i==0) begin
      assign wireOR1 [i] = wireXOR1[i];
    end
  end
endgenerate

```

```

        assign wireAND1 [i] = wireOR1[i];
    end else begin
        assign wireOR1 [i] = wireXOR1[i] | wireOR1[i-1];
        assign wireAND1 [i] = ~wireOR1 [i-1] & wireXOR1[i];
    end

    assign wireAND2 [i] = wireAND1 [i] & s2Compare[i];
end
endgenerate

assign better = | wireAND2 ; // [2]
assign s1=matchListMatrix[r];
assign pcS=pc[s];
assign s2=s;
assign finish=finishAND;
assign propose= |pcS & ~sIsMatch[s];
assign r=sPrefMatrix[s][Ks-pcS];
assign sIsMatchWire = sIsMatch;

assign allInputsXORed = e^g;
assign rPref[R*Kr*logS-1:0] = allInputsXORed[R*Kr*logS-1 :0] ;
assign sPref[S*Ks*logR-1:0] = allInputsXORed[R*Kr*logS-1 + S*Ks*logR-1 + 1: R*Kr*logS-1
+ 1 ] ;
assign o[R*logS]=finish;
assign o[R*logS-1:0]=matchList[R*logS-1:0];

generate // choose who to propose next
    for (i=0;i<SRounded;i=i+1) begin :propose_asgn
        if (i<S) begin
            assign canPropose[i] = |pc[i] & ~sIsMatch[i] & ~sIsRunning[i];
        end else begin
            assign canPropose[i] = 0;
        end
        if (i==0) begin
            assign encoderOR1[i] = canPropose[i];
            assign encoderInput[i] = encoderOR1[i];
        end else begin
            assign encoderOR1[i] = encoderOR1[i-1] | canPropose[i];
            assign encoderInput[i] = ~encoderOR1[i-1] & canPropose[i];
        end
    end
endgenerate

encoder
#(.logS(logS))
ENCODER
(.in(encoderInput),
.out(encoderOutput));

```

```

//-----ALGORITHM
integer k;
always@(posedge clk) begin
    if (rst) begin
        s<=0;
        sIsRunning<=1;
        for (k=0;k<R;k=k+1) begin :always_line3
            rIsMatch[k]<=1'b0;
            matchListMatrix[k]<=0;
        end
        for (k=0;k<S;k=k+1) begin :always_line
            pc[k]<=Ks;
            sIsMatch[k]<=1'b0;
            sInMatch[k]<=0;
        end
    end else begin
        if (propose) begin // if s can propose
            pc[s] <= pc[s]-1; // will lost one propose
            if (rIsMatch[r]==0) begin
                matchListMatrix[r]<=s;// assign them together
                sInMatch [s] <= r;
                sIsMatch [s] <= 1'b1;
                rIsMatch [r] <= 1'b1;
            end else begin // if r is already matched
                if (better) begin
                    matchListMatrix[r]<=s;// assign them together
                    sInMatch [s] <= r;
                    sIsMatch [s] <= 1'b1;
                    sIsMatch [s1] <= 1'b0;
                end
            end
        end
        sIsRunning[s]<=0;

        if (encoderInput[encoderOutput] ==1 ) begin
            sIsRunning[encoderOutput]<=1;
            s<=encoderOutput;
        end
    end
end

//Combinational Part

always@(*) begin
    finishAND=0;
    for (k=0;k<S;k=k+1) begin

```

```
                finishAND=finishAND&    ~|pc[k];  
            end  
end  
//..... MACROS  
endmodule
```



## Bibliography

- [1] M. S. Riazi, E. M. Songhori, A.-R. Sadeghi, T. Schneider, and F. Koushanfar, “Toward practical secure stable matching,” *Proceedings on Privacy Enhancing Technologies*, vol. 2017, no. 1, pp. 62–78, 2017.
- [2] M. S. Riazi, N. K. Dantu, L. V. Gattu, and F. Koushanfar, “GenMatch: Secure dna compatibility testing,” in *Hardware Oriented Security and Trust (HOST), 2016 IEEE International Symposium on*, pp. 248–253, IEEE, 2016.
- [3] M. S. Riazi, B. Chen, A. Shrivastava, D. Wallach, and F. Koushanfar, “Sub-linear privacy-preserving search with untrusted server and semi-honest parties,” *arXiv preprint arXiv:1612.01835*, 2016.
- [4] National Residency Matching Program. <http://www.nrmp.org>.
- [5] Stable Matching Algorithms. <http://www.dcs.gla.ac.uk/research/algorithms/stable/>.
- [6] A. Abdulkadiroğlu, P. A. Pathak, and A. E. Roth, “The New York city high school match,” *American Economic Review*, pp. 364–367, 2005.
- [7] M. Ostrovsky, “Stability in supply chain networks,” *American Economic Review*, vol. 98, no. 3, pp. 897–923, 2008.
- [8] D. Gale and M. Sotomayor, “Ms. Machiavelli and the stable matching problem,” *American Mathematical Monthly*, pp. 261–268, 1985.

- [9] D. Gusfield and R. W. Irving, *The stable marriage problem: Structure and algorithms*, vol. 54. MIT press Cambridge, 1989.
- [10] C.-P. Teo, J. Sethuraman, and W.-P. Tan, “Gale-Shapley stable marriage problem revisited: Strategic issues and applications,” in *Integer Programming and Combinatorial Optimization (IPCO’99)*, vol. 1610 of *LNCS*, pp. 429–438, Springer, 1999.
- [11] A. E. Roth, “The economics of matching: Stability and incentives,” *Mathematics of operations research*, vol. 7, no. 4, pp. 617–628, 1982.
- [12] C.-P. Teo, J. Sethuraman, and W.-P. Tan, “Gale-Shapley stable marriage problem revisited: Strategic issues and applications,” *Management Science*, vol. 47, no. 9, pp. 1252–1267, 2001.
- [13] P. Golle, “A private stable matching algorithm,” in *FC’06*, vol. 4107 of *LNCS*, pp. 65–80, Springer, 2006.
- [14] M. Franklin, M. Gondree, and P. Mohassel, “Improved efficiency for private stable matching,” in *CT-RSA’07*, vol. 4377 of *LNCS*, pp. 163–177, Springer, 2006.
- [15] M. Franklin, M. Gondree, and P. Mohassel, “Multi-party indirect indexing and applications,” in *ASIACRYPT’07*, vol. 4833 of *LNCS*, pp. 283–297, Springer, 2007.
- [16] M. Keller and P. Scholl, “Efficient, oblivious data structures for MPC,” in *ASIACRYPT’14*, vol. 8874 of *LNCS*, pp. 506–525, Springer, 2014.

- [17] A. Yao, “How to generate and exchange secrets,” in *FOCS’86*, pp. 162–167, IEEE, 1986.
- [18] E. M. Songhori, S. U. Hussain, A.-R. Sadeghi, T. Schneider, and F. Koushanfar, “TinyGarble: Highly compressed and scalable sequential garbled circuits,” in *IEEE S&P’15*, pp. 411–428, IEEE, 2015.
- [19] P. C. Ng and E. F. Kirkness, “Whole genome sequencing,” in *Genetic Variation*, pp. 215–226, Springer, 2010.
- [20] Commercialization of full genome sequencing. <http://www.genengnews.com/gen-articles/race-to-cut-whole-genome-sequencing-costs/939/>.
- [21] B. A. Malin, “An evaluation of the current state of genomic data privacy protection technology and a roadmap for the future,” *JAMIA*, pp. 28–34, 2005.
- [22] Y. Erlich and A. Narayanan, “Routes for breaching and protecting genetic privacy,” *Nature Reviews Genetics*, pp. 409–421, 2014.
- [23] S. Jha, L. Kruger, and V. Shmatikov, “Towards practical privacy for genomic computation,” in *S&P*, pp. 216–230, May 2008.
- [24] G. Opelz, T. Wujciak, B. Döhler, S. Scherer, and J. Mytilineos, “Hla compatibility and organ transplant survival. collaborative transplant study.,” *Reviews in immunogenetics*, pp. 334–342, 1998.
- [25] J. M. Cecka, “The unos scientific renal transplant registry—ten years of kidney transplants.,” *Clinical transplants*, pp. 1–14, 1996.
- [26] J. Sierra, B. Storer, J. A. Hansen, J. W. Bjerke, P. J. Martin, E. W. Petersdorf, F. R. Appelbaum, E. Bryant, T. R. Chauncey, G. Sale, J. E. Sanders, R. Storb,

- K. M. Sullivan, and C. Anasetti, “Transplantation of marrow cells from unrelated donors for treatment of high-risk acute leukemia: The effect of leukemic burden, donor hla-matching, and marrow cell dose,” *Blood*, pp. 4226–4235, 1997.
- [27] D. Speiser, J. Tiercy, N. Rufer, C. Grundschober, A. Gratwohl, B. Chapuis, C. Helg, C. Loliger, M. Siren, E. Roosnek, and M. Jeannet, “High resolution hla matching associated with decreased mortality after unrelated bone marrow transplantation,” *Blood*, pp. 4455–4462, 1996.
- [28] . G. P. Consortium *et al.*, “An integrated map of genetic variation from 1,092 human genomes,” *Nature*, pp. 56–65, 2012.
- [29] A.-R. Sadeghi, T. Schneider, and I. Wehrenberg, “Efficient privacy-preserving face recognition,” in *Information, Security and Cryptology–ICISC 2009*, pp. 229–244, Springer, 2009.
- [30] M. Barni, T. Bianchi, D. Catalano, M. Di Raimondo, R. Donida Labati, P. Failla, D. Fiore, R. Lazzeretti, V. Piuri, F. Scotti, *et al.*, “Privacy-preserving fingercode authentication,” in *Proceedings of the 12th ACM workshop on Multimedia and security*, pp. 231–240, ACM, 2010.
- [31] M. Blanton and P. Gasti, “Secure and efficient protocols for iris and fingerprint identification,” in *Computer Security–ESORICS 2011*, pp. 190–209, Springer, 2011.
- [32] M. A. Pathak and B. Raj, “Privacy-preserving speaker verification as password matching,” in *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*, pp. 1849–1852, IEEE, 2012.

- [33] J. Bobadilla, F. Ortega, A. Hernando, and A. Gutiérrez, “Recommender systems survey,” *Knowledge-Based Systems*, vol. 46, pp. 109–132, 2013.
- [34] J. Kilian, “Founding cryptography on oblivious transfer,” in *Proceedings of the twentieth annual ACM symposium on Theory of computing*, pp. 20–31, ACM, 1988.
- [35] R. Cramer, I. Damgård, and J. B. Nielsen, “Multiparty computation, an introduction,” *Contemporary cryptology*, pp. 41–87, 2009.
- [36] P. Indyk and D. Woodruff, “Polylogarithmic private approximations and efficient matching,” in *Theory of Cryptography*, pp. 245–264, Springer, 2006.
- [37] W. B. Johnson and J. Lindenstrauss, “Extensions of lipschitz mappings into a hilbert space,” *Contemporary mathematics*, vol. 26, no. 189-206, p. 1, 1984.
- [38] C. Dwork, “Differential privacy,” in *Automata, languages and programming*, pp. 1–12, Springer, 2006.
- [39] V. Kolesnikov and T. Schneider, “Improved garbled circuit: Free XOR gates and applications,” in *ICALP’08*, vol. 5126 of *LNCS*, pp. 486–498, Springer, 2008.
- [40] M. Bellare, V. T. Hoang, S. Keelveedhi, and P. Rogaway, “Efficient garbling from a fixed-key blockcipher,” in *IEEE S&P’13*, pp. 478–492, IEEE, 2013.
- [41] S. Zahur, M. Rosulek, and D. Evans, “Two halves make a whole,” in *EURO-CRYPT’15*, vol. 9057 of *LNCS*, pp. 220–250, Springer, 2015.
- [42] Y. Ishai, J. Kilian, K. Nissim, and E. Petrank, “Extending oblivious transfers efficiently,” in *CRYPTO’03*, vol. 2729 of *LNCS*, pp. 145–161, Springer, 2003.

- [43] G. Asharov, Y. Lindell, T. Schneider, and M. Zohner, “More efficient oblivious transfer and extensions for faster secure computation,” in *ACM CCS’13*, pp. 535–548, ACM, 2013.
- [44] O. Goldreich and R. Ostrovsky, “Software protection and simulation on oblivious rams,” *Journal of the ACM (JACM)*, vol. 43, no. 3, pp. 431–473, 1996.
- [45] E. Stefanov, E. Shi, and D. Song, “Towards practical oblivious RAM,” in *NDSS’12*, 2012.
- [46] B. Pinkas and T. Reinman, “Oblivious RAM revisited,” in *CRYPTO’10*, vol. 6223 of *LNCS*, pp. 502–519, Springer, 2010.
- [47] E. Shi, T.-H. H. Chan, E. Stefanov, and M. Li, “Oblivious RAM with  $\mathcal{O}(\log^3 n)$  worst-case cost,” in *ASIACRYPT’11*, vol. 7073 of *LNCS*, pp. 197–214, Springer, 2011.
- [48] S. D. Gordon, J. Katz, V. Kolesnikov, F. Krell, T. Malkin, M. Raykova, and Y. Vahlis, “Secure two-party computation in sublinear (amortized) time,” in *ACM CSS’12*, pp. 513–524, ACM, 2012.
- [49] X. Wang, H. Chan, and E. Shi, “Circuit ORAM: On tightness of the Goldreich-Ostrovsky lower bound,” in *ACM CCS’15*, pp. 850–861, ACM, 2015.
- [50] S. Zahur, X. Wang, M. Raykova, A. Gascon, J. Doerner, D. Evans, and J. Katz, “Revisiting square root ORAM: Efficient random access in multi-party computation,” 2016. To appear in IEEE S&P’16. Online: <http://jackdoerner.net/resources/pdf/sqrtoram.pdf>.

- [51] Y. Huang, D. Evans, J. Katz, and L. Malka, “Faster secure two-party computation using garbled circuits,” in *USENIX Security’11*, pp. 539–554, USENIX, 2011.
- [52] Y. Lindell and B. Pinkas, “An efficient protocol for secure two-party computation in the presence of malicious adversaries,” in *EUROCRYPT’07*, vol. 4515 of *LNCS*, pp. 52–78, Springer, 2007.
- [53] J. B. Nielsen and C. Orlandi, “LEGO for two-party secure computation,” in *TCC’09*, vol. 5444 of *LNCS*, pp. 368–386, Springer, 2009.
- [54] A. Shelat and C.-h. Shen, “Two-output secure computation with malicious adversaries,” in *EUROCRYPT’11*, vol. 6632 of *LNCS*, pp. 386–405, Springer, 2011.
- [55] Y. Lindell and B. Pinkas, “Secure two-party computation via cut-and-choose oblivious transfer,” *Journal of Cryptology*, vol. 25, no. 4, pp. 680–722, 2012.
- [56] D. Gale and L. S. Shapley, “College admissions and the stability of marriage,” *American Mathematical Monthly*, pp. 9–15, 1962.
- [57] The Science Behind HLA. [https://www.seattlecca.org/client/documents/The-Science-Behind-HLA-Typing\\_8460\\_0.pdf](https://www.seattlecca.org/client/documents/The-Science-Behind-HLA-Typing_8460_0.pdf).
- [58] C. Ober, L. R. Weitkamp, N. Cox, H. Dytch, D. Kostyu, and S. Elias, “Hla and mate choice in humans,” *The American Journal of Human Genetics*, pp. 497–504, 1997.
- [59] P. Indyk and R. Motwani, “Approximate nearest neighbors: Towards removing the curse of dimensionality,” in *STOC*, (Dallas, TX), pp. 604–613, 1998.

- [60] A. Z. Broder, “On the resemblance and containment of documents,” in *the Compression and Complexity of Sequences*, (Positano, Italy), pp. 21–29, 1997.
- [61] A. Z. Broder, M. Charikar, A. M. Frieze, and M. Mitzenmacher, “Min-wise independent permutations,” in *STOC*, (Dallas, TX), pp. 327–336, 1998.
- [62] M. S. Charikar, “Similarity estimation techniques from rounding algorithms,” in *STOC*, (Montreal, Quebec, Canada), pp. 380–388, 2002.
- [63] A. Rajaraman and J. D. Ullman, *Mining of massive datasets*. Cambridge University Press, 2011.
- [64] M. Henzinger, “Finding near-duplicate web pages: a large-scale evaluation of algorithms,” in *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 284–291, ACM, 2006.
- [65] M. X. Goemans and D. P. Williamson, “Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming,” *Journal of the ACM (JACM)*, vol. 42, no. 6, pp. 1115–1145, 1995.
- [66] J. L. Carter and M. N. Wegman, “Universal classes of hash functions,” in *STOC*, pp. 106–112, 1977.
- [67] P. Li and A. C. König, “Theory and applications b-bit minwise hashing,” *Commun. ACM*, 2011.
- [68] M. Naor, B. Pinkas, and R. Sumner, “Privacy preserving auctions and mechanism design,” in *ACM conference on Electronic Commerce (EC’99)*, pp. 129–139, ACM, 1999.



- [69] Y. Lindell and B. Pinkas, “A proof of security of Yao’s protocol for two-party computation,” *Journal of Cryptology*, vol. 22, no. 2, pp. 161–188, 2009.
- [70] B. Kreuter, A. Shelat, B. Mood, and K. R. Butler, “PCF: A portable circuit format for scalable two-party secure computation.,” in *Usenix Security’13*, pp. 321–336, USENIX, 2013.
- [71] J. J. Farrell, *The prediction of hla genotypes from next generation sequencing and genome scan data*. PhD thesis, BOSTON UNIVERSITY, 2014.
- [72] HLA Compatibility Calculator. [http://www.hiv.lanl.gov/content/immunology/hla/hla\\_compare.html](http://www.hiv.lanl.gov/content/immunology/hla/hla_compare.html).
- [73] Kidney Transplantation. <https://web.stanford.edu/dept/HPS/transplant/html/hla.html>.
- [74] R. J. Bayardo, Y. Ma, and R. Srikant, “Scaling up all pairs similarity search,” in *WWW*, pp. 131–140, 2007.
- [75] A. Andoni and P. Indyk, “Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions,” in *Foundations of Computer Science, 2006. FOCS’06. 47th Annual IEEE Symposium on*, pp. 459–468, IEEE, 2006.
- [76] A. Shrivastava and P. Li, “Densifying one permutation hashing via rotation for fast near neighbor search,” in *ICML*, (Beijing, China), 2014.
- [77] A. Dasgupta, R. Kumar, and T. Sarlós, “Fast locality-sensitive hashing,” in *KDD*, pp. 1073–1081, 2011.
- [78] I. Damgård and M. Jurik, “A generalisation, a simplification and some applications of Paillier’s probabilistic public-key system,” in *PKC’01*, vol. 1992 of

- LNCS*, pp. 119–136, Springer, 2001.
- [79] P.-A. Fouque, G. Poupard, and J. Stern, “Sharing decryption in the context of voting or lotteries,” in *FC’01*, vol. 1962 of *LNCS*, pp. 90–104, Springer, 2001.
- [80] P. Paillier, “Public-key cryptosystems based on composite degree residuosity classes,” in *EUROCRYPT’99*, vol. 1592 of *LNCS*, pp. 223–238, Springer, 1999.
- [81] M. Naor and K. Nissim, “Communication preserving protocols for secure function evaluation,” in *STOC’01*, pp. 590–599, ACM, 2001.
- [82] E. Ayday, E. D. Cristofaro, J. Hubaux, and G. Tsudik, “The chills and thrills of whole genome sequencing,” *CoRR*, 2013.
- [83] M. Naveed, E. Ayday, E. W. Clayton, J. Fellay, C. A. Gunter, J. Hubaux, B. A. Malin, and X. Wang, “Privacy and security in the genomic era,” *CoRR*, 2014.
- [84] E. De Cristofaro, S. Faber, P. Gasti, and G. Tsudik, “Genodroid: are privacy-preserving genomic tests ready for prime time?,” in *Proceedings of the 2012 ACM workshop on Privacy in the electronic society*, pp. 97–108, ACM, 2012.
- [85] F. Bruekers, S. Katzenbeisser, K. Kursawe, and P. Tuyls, “Privacy-preserving matching of dna profiles.” *Cryptology ePrint Archive*, Report 2008/203, 2008. <http://eprint.iacr.org/>.
- [86] M. Kantarcioglu, W. Jiang, Y. Liu, and B. Malin, “A cryptographic approach to securely share and query genomic sequences,” *T-ITB*, pp. 606–617, Sept 2008.
- [87] P. Bohannon, M. Jakobsson, and S. Srikwan, “Cryptographic approaches to privacy in forensic dna databases,” in *Public Key Cryptography* (H. Imai and

- Y. Zheng, eds.), Lecture Notes in Computer Science, pp. 373–390, Springer Berlin Heidelberg, 2000.
- [88] J. R. Troncoso-Pastoriza, S. Katzenbeisser, and M. Celik, “Privacy-preserving error resilient dna searching through oblivious automata,” in *CCS*, pp. 519–528, ACM, 2007.
- [89] S. Rane and P. T. Boufounos, “Privacy-preserving nearest neighbor methods: comparing signals without revealing them,” *Signal Processing Magazine, IEEE*, vol. 30, no. 2, pp. 18–28, 2013.
- [90] S. Rane, W. Sun, and A. Vetro, “Privacy-preserving approximation of l1 distance for multimedia applications,” in *Multimedia and Expo (ICME), 2010 IEEE International Conference on*, pp. 492–497, IEEE, 2010.
- [91] P. Li, M. Mitzenmacher, and A. Shrivastava, “Coding for random projections,” in *ICML*, 2014.
- [92] P. Boufounos and S. Rane, “Secure binary embeddings for privacy preserving nearest neighbors,” in *Information Forensics and Security (WIFS), 2011 IEEE International Workshop on*, pp. 1–6, IEEE, 2011.
- [93] M. Kuzu, M. S. Islam, and M. Kantarcioglu, “Efficient similarity search over encrypted data,” in *Data Engineering (ICDE), 2012 IEEE 28th International Conference on*, pp. 1156–1167, IEEE, 2012.
- [94] Y. Shang, Z. Liu, J. Wang, and X. Xiao, “Triangle and centroid localization algorithm based on distance compensation,” in *ICISCE*, IET, 2012.