5-2021

Fast and Secure Friend Recommendation in Online Social Networks

Raiyan Hossain
Montclair State University

ABSTRACT

Online Social Networks have completely transformed communication in the world of social networks. Participation in online social networks have been growing significantly and is expected to continue to grow in the upcoming years. As user participation in online social media is on the rise, so is the concern pertaining to user privacy and information security; users want to interact on social media without jeopardizing their privacy and personal information. Extensive research has been conducted in the area of developing privacy-preserving protocols to allow users to interact in a secure and privacy-preserving environment. One of the elements that social media have is the feature or ability to befriend other users. While a user may manually search for friends to "add", social media networks like Twitter, Facebook, Instagram, Snapchat and others facilitate friend recommendations to their users based on different criteria. We examine and compare the advantages and disadvantages of existing privacy-preserving techniques and schemes. We also analyze different models used to implement friend recommendation protocols and study proximity measurement metrics used in existing works. This thesis scrutinizes the security weaknesses and vulnerabilities of three Friend Recommendation Protocols from existing work and develop a corresponding solution. We propose a ($FSFR$) protocol that is based on Shamir's Secret Sharing to facilitate friend recommendations in Online Social Networks in a fast, secure and private manner. After comparing our protocol with existing protocols in terms of security, computation efficiency, costs, flexibility and more, we conclude that our $FSFR$ protocol guarantees a superior and more efficient friend recommendation protocol.

MONTCLAIR STATE UNIVERSITY

Fast and Secure Private Friend Recommendations In Online Social Networks

By

Raiyan Hossain

A Master's Thesis Submitted to the Faculty of

Montclair State University

In Partial Fulfillment of the Requirements

For the Degree of

Masters of Science

May 2021

School College of Science and Mathematics

Department Computer Science

Thesis Committee:

[redacted]

Dr. Bharath Kumar Samanthula
Thesis Sponsor

[redacted]

Dr. Kazi Zakia Sultana
Committee Member

[redacted]

Dr. Jiacheng Shang
Committee Member

FAST AND SECURE FRIEND RECOMMENDATION IN ONLINE SOCIAL NETWORKS

A THESIS

Submitted in partial fulfillment of the requirements

For the degree of Master of Science

by

RAIYAN HOSSAIN

Montclair State University

Montclair, NJ

May 2021

ACKNOWLEDGMENTS

First and above all, I praise God for providing me this opportunity and granting me the ability to successfully proceed with pursuing my Masters and completing it at Montclair State University.

I would also like to express my deepest gratitude towards Dr. Bharath Samanthula for making this work possible and for allowing me to pursue this research under his mentorship. Aside from pursuing research under his guidance, Dr. Samanthula advised me throughout my graduate school journey regarding current and future plans and has been an inspirational pillar of support at Montclair State University. He was always just an email away and maintained an positive environment. Upon being admitted to Montclair State University, Dr. Samanthula was the first professor I worked with; fortunately, he is the professor I am closing my Graduate School chapter with as well. I would also like to thank my committee members for agreeing to be a part of my thesis committee in an undoubtedly busy time.

Finally, I would also like to thank my family for their constant encouragement. I thank my mother and father, to whom I will forever be indebted to, for their endless sacrifice and support. My parents have been constant support systems throughout my life in all aspects and my educational endeavors were not an exception. In particular, I would like to thank my younger sister, Radeyah, for her unwavering support and understanding. Thank you for always being there for me as a friend and sister and for supporting me and my pursuits. I also thank my friends for always supporting me through various ways and encouraging me all the time.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

1. INTRODUCTION

1.1. BACKGROUND

According to [1],it is estimated that the number of users who are on any form of social media was recorded to have reached 4.2 billion as of January 2021; the most popular social network was revealed to be Facebook with a total of about 2.74 billion user accounts. Social Media allows for users who can be at different locations to interact with one another. With such a large number of users utilizing Online Social Networks, there is an abundance of personal information and data that needs to be protected and kept private. Users can keep personal information such as passwords, private messages, phone numbers, email addresses and shared content private by not publicizing it on the network. For instance, on Twitter, users have the option to limit who may view the content they post by selecting "Protect my Tweets". Likewise, if one does not wish to be discovered by other users, they have the ability to control their discoverability settings and manage contacts imported. Users can decide whether they would like for people who have their email address or phone number to find them or not; they also have the ability to limit location information mined by Twitter through unchecking the feature in the settings. Twitter also offers the feature of allowing users to choose if they would allow Twitter to keep track of the sites they visit that incorporate Twitter content such as embedded timelines. Thus, users have some control over the content they would like to keep private. However, while users are offered the option to protect certain information, the majority of the data users provide, even unintentionally, could jeopardize their privacy.

Social media supplies an overwhelming amount of data feeds that are extracted and utilized to be used for different purposes such as marketing [2]. This results in a large amount of information exposure of casual social media users. Records of users such as their identities, friend lists, likes and comments are centrally stored in Social Data Generation [3].

Several questions are brought up in [4,5] that challenge the credibility of Online Social Networks (OSN). Questions are raised pertaining to what pieces of information exactly are supplied by users to the OSNs. Concerns are also raised regarding the default privacy settings for the framework of the social network. It is important to note that OSNPs like Twitter and Facebook are needed to allow social networks to function as these network providers provide the functionalities to the users. Consequently, a framework that utilizes a central network provider but implements privacy-preserving techniques to ensure user privacy is required.

In general, many online social networks collect sensitive user profile data and store the data after having them encrypted on their server. Essentially, users have no control over what the OSNP chooses to do with the data. OSNs are free to share this stored data with third parties for business purposes [6]. Moreover, sensitive information may be obtained through unauthorized methods like hacking. In April 2021, Facebook was met with a surge of individuals threatening legal action after they decided to not notify more than 530 million users from across 106 countries of a breach that resulted in their personal information being exposed. The leaked personal data was obtained through hacking and included phone numbers, locations, full names, and email addresses [5].

1.2. CONTRIBUTION

This thesis focuses on friend recommendations in online social networks and proposes a protocol, $FSFR$ (Fast and Secure Friend Recommendation), that allows to carry out the friend recommendation process in a more efficient, secure and privacy-preserving manner than the existing protocols. We propose two $FSFR$ solutions, $FSFR_1$ and $FSFR_2$ with the latter being a more secure implementation of the former. While we utilize Shamir's Secret Sharing in both of our solutions, $FSFR_2$ utilizes a combination of Shamir's Secret Sharing with Paillier's encryption.

Our protocol utilizes a Federated Cloud Model by requiring for the utilization of more than one cloud server to communicate with the OSNP in order to facilitate the friend recommendation. When implementing the PAFR protocol [6], the computation time of the cloud server was negligible as it was used primarily as a storage medium as opposed to being utilized in the computation process. However, this placed additional responsibility and burden on the OSNP as it left all the expensive computations such as decryption to the OSNP. The presence of a single cloud contributed to the OSNP's comparatively high computation time.

Our $FSFR$ protocol will reduce this excess computation time by utilizing an additional cloud server. By adopting a federated cloud environment, we introduce a framework based on the collaboration of multiple public cloud environments. Thus, the OSNP can be spared of some of it's responsibilities by allowing the additional cloud server to perform some of the computations. By adopting a federated cloud environment, the responsibilities will now be distributed across three parties as opposed to only two parties.

The $FSFR$ protocol is expected to maintain the following:

- Security: the proposed protocol utilizes Shamir's Secret Sharing to prevent any single party from having access to user's information or the entire data of any single user. It also utilizes permutations and randomization to ensure higher security. More information on the security of our protocol will be discussed in Section 5.

- Efficiency: the proposed protocol does not encrypt the entire matrix of each user; rather, we split the data. This is more efficient and less expensive. For our second solution, we utilize Paillier's encryption; however, we limit the encryption to only encrypting the random values instead of encrypting the whole matrix which would have been expensive. A comparative analysis on the efficiency of our protocol in comparison to other existing protocols can be seen in Section 6.

- User convenience: the protocol does not require users to be online throughout the friend recommendation process.
- Distributed Computation: As discussed earlier, utilizing a collaboration between the OSNP and federated cloud environment will allow for computation time to be significantly less while allowing for the computations to be distributed fairly equally.

The remaining sections are organized as follows; Section 2 discusses related works, Section 3 explains the preliminaries, Section 4 explains the framework of our protocol, Section 5 goes into depth explaining both of the solutions of our $FSFR$ protocol and their corresponding steps, Section 6 evaluates the performance of the protocol while comparing it to existing protocols, and finally, we conclude with future recommendations.

2. RELATED WORKS

There are several existing works in the field of privacy-preservation. This encompasses studies related to privacy-preserving techniques, decentralized vs centralized architectures, models, frameworks, and actual protocols to implement the friend recommendation process. We analyzed these topics while studying their advantages and disadvantages. We refer to three friend recommendation protocols for analysis and examine the strengths and weaknesses of each of them while identifying security weaknesses that we sought to eliminate. The first protocol, Privacy-Preserving Friend Recommendation with Homomorphic Encryption ($PPFR_h$) [7] is founded on the homomorphic encryption scheme which utilizes a universal hash function. The second protocol, Privacy-Preserving Friend Recommendation with Source Privacy ($PPFR_{sp}$) [7] utilizes the concept of protecting source privacy through anonymous message routing using secret sharing while the third protocol, Privacy-Aware Friend Recommendation ($PAFR$) [6], proposes a framework that utilizes the concept of outsourcing encrypted profiles to a cloud environment in a privacy-aware manner approaching it by utilizing Paillier's encryption as well as AES encryption. All three protocols utilize the common neighbors approach for the proximity measure in the friend recommendation process.

2.1. PRIVACY PRESERVING FRIEND RECOMMENDATION

This section discusses the implementation of the two Privacy-Preserving Friend Recommendation Protocols and provide a comparison between the two.

2.1.1. $PPFR_h$. - The first protocol which utilizes homomorphic encryption is $PPFR_h$. The fundamental procedures associated with $PPFR_h$ is to first encrypt the ID of each friend independently and perform a homomorphic addition on the encrypted data. Each user generates an encrypted matrix that contains his/her friend list information and forwards it to target user A. Then, A aggregates the encrypted friend lists component-wise to get the encrypted frequency for each potential friend. The first column entry contains an actual user ID while the row stores the hashed value of the user ID. After this, with the help of the third party who holds the private key pk, A securely retrieves the IDs of users whose frequency is greater than or equal to the threshold. One of the biggest limitations of the $PPFR_h$ protocol is that there is no off-line support. The recommendation process can be carried out only when all parties involved are online. Also, the protocol leaks the common neighbor score to a third part violating the privacy of all the users involved.

2.1.2. $PPFR_{sp}$. The next protocol is the ($PPFR_{sp}$) protocol. Alkanhal et al. [7] discusses the protocol as the following; they denote the user waiting for friend recommendations as A, the friend of A as B, and a single friend of B (potential friend candidate to A) as C_{ij} in the subset of C (all of B's friends). In this protocol, it is assumed that each user in the network generates a public-private key pair using the RSA public key system. Each C_{ij} uses an AES encryption algorithm to encrypt his/her data. The secret key is divided into a number of individual shares; the maximum number of shares is the number of friends in the friend list of C_{ij}. This results in requiring a number of minimum shares to be reconstructed in order to obtain the key. The main idea of the $PPFR_{sp}$ protocol is to allow C to introduce themselves to A in manner that preserves the privacy of the source, or user seeking the friend recommendation; this is done through anonymous communication. This method is similar to the Onion Routing method. Here, A waits for the self introductions to come. B, which is the intermediate user between A and C_{ij}, arranges a random path along which C_{ij} will pass the self introduction to A. The random path can hide the identities of all C_{ij} in the subset of C by preventing A from tracking back to them.

Once again, the goal of this protocol is founded on allowing for C to make friends and pass introductions to A to without disclosing who they are sending introductions to. This preserves the privacy of both A and B. Moreover, as User A cannot trace back to C_{ij}, the privacy of all C_{ij} in C is preserved. However, although none of the parties involved can be identified, users who have a common neighbor score greater than the threshold are revealed. These are users who are likely to be recommended as friends to the target user.

2.2. PAFR

In [6], the authors discuss the Privacy-Aware Friend Recommendation protocol (PAFR) framework which allows users to outsource their encrypted profile data to a cloud environment. The framework utilizes a hybrid encryption approach which is a combination of homomorphic encryption scheme with Paillier's encryption and AES. This protocol utilizes a decentralized architecture and involves three components: the user, a cloud, and an OSNP. The user's role is to encrypt his/her profile data and outsource it in it's encrypted form to the cloud which stores and manages all of the encrypted profile data. The cloud is also the entity that will be communicating with the OSNP when facilitating the friend recommendation process. The OSNP will create as well as share the cryptographic keys with users. The $PAFR$ protocol is divided into two stages; Secure Outsourcing Profile Data and Secure Collaborative Friend Recommendation (SCFR). The Outsourcing stage defines what is needed to be done before sending the profile data to the cloud. The SCFR stage defines the communication phase and explains the collaboration between the cloud and OSNP.

Zhou et al. [8] discuss the homomorphic encryption scheme on Integer Vectors. They discuss some implementations of computation tasks such as feature extraction and data aggregation. Upon discussing the features, they agree that while it is difficult to develop universal homomorphic encryption-schemes due to it's computation and communication costs, homomorphic-encryption schemes may be able to be developed for specific applications. Homomorphic encryption differs from general encryption methods by having the ability to perform computations directly onto encrypted data without accessing the data itself. The scheme supports both "addition" and "multiplication" operations on encrypted data. Thus, when using the homomorphic encryption scheme, addition and multiplication operations can be performed on a cipher-text rather than accessing the actual plain-text. This is significant and advantageous as it prevents any encounter with the actual data eliminating the possibility of tampering with the data that is meant to be protected during the computation process.

The paper also raises the question aimed to reduce costs of computations and overall communication time involved with homomorphic encryption operations. In order to reduce computation time, we propose a protocol that initiates a collaboration between a federated cloud model and OSNP utilizing Shamir's Secret Sharing Scheme. The scheme, like homomorphic encryption, allows for addition and multiplication operation to be performed on data; however, in contrast to the computations being performed on encrypted data, it is performed on secret shares.

2.3. PRIVACY-PRESERVING MODEL

Samanthula et al. [7] discusses models used to implement Privacy-Preserving Friend Recommendation protocols. The authors refer to existing work and reference a study by Machanavajjhala et al. [9] who mentions of a trade-off between accuracy and privacy when implementing private friend recommendation using differential privacy. The model adopted in their work, however, had some limitations; namely, choosing to maintain accuracy at the expense of privacy and vice versa. The authors [7] propose a superior model called Secure Multiparty Computation (SMC). The SMC Model is able to maintain better accuracy while preserving the privacy of all users involved. (SMC)

2.4. CONTENT-BASED AND TOPOLOGY-BASED ALGORITHMS

There are two ways algorithms can be implemented to recommend new friends to a user. One of the algorithms is content-based while the other is topology-based. Friend recommendation algorithms can also utilize a combination of both content-based and topology-based algorithms.

Yu et al. [10] introduces a framework to allow friend recommendation based on similarities between user contents. This algorithm utilizes users' content as the basis to decide whether to recommend a user by considering users' profile information such as interests, education and age. The framework enables friend recommendations by analyzing the extracted information from user's profile pages. The protocol will be able to generate the frequency of keywords and measures the frequency of each word. This is used to compare with other users who also undergo the same analysis with their profile pages. Users with the highest level of proximity, or matching keywords, will be recommended to each other [10]. Thus, the likelihood of A being recommended to B depends on the similarity between their profiles. Alkanhal et al. [6] discusses the second option which is based on a topology-based friend recommendation framework. This framework focuses on the topological structures of social networks and how users are interconnected on the network. Through topology-based friend recommendation protocols, the friend recommendation process is implemented by recommending new friends to a user from a set of their friends' friends. More details on this framework can be studied in Section 4.

2.5. NETWORK ARCHITECTURES

Nilizadeh et al. [11] discusses a decentralized architecture known as Cachet which uses a combination of hash tables and attribute based encryption to protect confidentiality, integrity and availability of user content. It also preserves the privacy of user relationships through a decentralized architecture that involves no network provider; all communications happen directly between users. Cutillo [14] explains that while privacy exposures can be eliminated through creating more secure protocols and frameworks to work with the OSN, the risk of user's privacy is inevitable due to the central storage aspect of centralized architectures. Thus, they opt for a Peer-to-peer architecture as an alternative to the centralized framework.

In contrast, Samanthula et al. [7] opts for a centralized architecture which allows for users' data to be encrypted and stored on the server of an OSN. The paper focuses on an OSN framework where the data resides on the OSN as opposed to Cachet which has no central network provider. The paper [7] argues that decentralization would force users to choose between availability, convenience or reliability through forcing them to store data on their own systems. If not that, users will be forced to store data on network providers that they do not know or would have no more trust in than they would in a centralized network provider. In the case where users choose to store their data on a separate network provider, they might as well have a central network provider. The solution [7] opts for a centralized network and assumes that users can encrypt their profile data before sending it to be stored on the OSN's server. However, despite the encryption of users' profile data, concerns are raised regarding how trustworthy the provider actually is. As discussed earlier, we need to rely on network providers as the losses when moving to a decentralized structure would outweigh the benefits. This creates the incentive to develop protocols that guarantee efficiency and privacy.

2.6. FRIENTEGRITY

To address this, Feldman et al. [12] develops a framework called Frientegrity; under this framework, the service provider sees only the encrypted data and cannot carry out any operation that is not expected of it without it being detected. However, Samanthula et al. [6] argues that while the integrity and privacy of the contents of a user is protected, this model still allows for the relationship between two users to be revealed to the service provider. The authors [6] develop a solution to counter this information leakage by creating a protocol that does not reveal the friendship between any two users to the service provider. Another weakness of the Frientegrity model is that target user A can search through his/her friends access control list to search for new friends. The paper [6] identifies this as a violation of user privacy as it allows for the friend lists of target user A's friends to be revealed to A and proposes a solution for that. In their solution, the friend list of a user is never revealed to other users. However, although the protocol allows for the content of the users' friend lists to be kept private, it allows for the possibility of the permuted friend lists to be leaked to the OSNP.

3. PRELIMINARIES

In this section, we present some concepts and notations that will be used in the proposed solution.

3.1. UNIVERSAL HASH FUNCTION

Hashing is a method used to convert a range of key values (using modulo operator) into a range of array indexes. The goal of hashing is to map the elements of a domain into a smaller domain. For instance, consider a set of integers in domain L = {0,1,..,l-1}. Through hashing, we map these integers into a smaller domain we denote as V = {0,1,..,s-1} where $s < l$ with minimum number of collisions. A hash function is defined as below:

$$h_{ab}(k) = ((ak + b) \mod p) \mod N \tag{3.1}$$

In the hash function equation above, p is a prime number ≥ 1 while a and b are random values.

3.1.1. Hash Table. A hash table, also known as a hash map, is a data structure that maps keys to values. It is used to store an index into an array of buckets or slots from which the desired value can be found. The index, or hash code, is computed by using the hash function. For our protocol, we utilize a hash table to store the user's information (ID and friend lists). An example is provided below. We denote the following:

- X as the set of all possible keys (k).
- Table T as the Hash Table with S positions. These positions or slots will contain the hash value.

	Frequently Used Notations
$PPFR_h$	Privacy-preserving Friend Recommendation with Homomorphic Encryption
$PPFR_{sp}$	Privacy-preserving Friend Recommendation with Source Privacy
$PAFR$	Privacy-Aware Friend Recommendation
$FSFR_1$	Fast and Secure Friend Recommendation Solution$_1$
$FSFR_2$	Fast and Secure Friend Recommendation Solution$_2$
C_1	$Cloud_1$
C_2	$Cloud_2$
$OSNP$	Online Social Network Provider
π_1	C_1's and C_2's random permutation function
π_2	$OSNP$'s random permutation function

Table 3.1: Frequently Used Notations

- Hash function h(k): The function maps X to $\{0,1,..,S-1\}$. In this step, function $h(k)$ will map any key (k) to one of the slots in table T. After the mapping is complete, We use $h(k)$ to refer to the location in T where the key is stored.

Suppose we want to map 4 user IDs to a hash table. Let us define the hash function as: the (sum of the digits in ID *mod* 10). In this example, the user IDs are our key values. We will map these keys to indexes in our array. The user IDs are:

- 7250903661
- 7302383966
- 7754321661
- 2740487840

For each entry, $h(k)$ = (sum of id digits *mod* 10) is computed. An example is shown below using the first entry ID: 9014638161.

$$h(k) = (7+2+5+0+9+0+3+6+6+1 \; mod \; 10) = (39 \; mod \; 10) = 9$$

Element	Key(k)	Sum of digits	Hash Value $h(k)$
1	7250903661	39	9
2	7302383966	48	8
3	7754321661	42	2
4	2740487840	44	4

Table 3.2: Mapping User IDs to Hash Table

As reflected in the table, each of the keys or User IDs are mapped to a specific index in the hash table. The index serves as a reference to the key value.

3.1.2. Hash vs Encryption. Hashing and encryption are two separate cryptographic processes that utilizes an algorithm and key, Encryption allows to convert plaintext data into something indecipherable. You can, however, decrypt the data by using a corresponding cryptographic key. On the other hand, once data is hashed, it cannot be restored to its original format as it is a one-way process.

3.2. COMMON NEIGHBOR SCORE

As discussed in the Section 2, the common neighbor proximity states that two strangers who have a friend in common are more likely to be introduced to each other than those who do not have any friends in common. The algorithm contains a threshold that allows only users with scores more than or equal to t to be recommended as friends to A. When computing the common neighbors score, it is important to note that only nodes that are two-hop neighboring nodes [6] are considered. This means that only nodes that are two nodes apart from the user is considered. Thus, if we would like to recommend A new potential friends, we will only take account of the users who are two nodes away from it.The function below represents how the Common Neighbors Score is calculated.

$$S(x, y) = |N(x) \cap N(y)|$$

In the function, $N(x)$ denotes the set of adjacent nodes to node x while $N(y)$ denotes the set of adjacent nodes to node y. The intersection element of Set theory ($\cap$) is used to indicate the intersection or common adjacent nodes identified by implementing the formula. The common nodes or intersecting nodes will be candidates that can be recommended to the target user. A common neighbor score of 0 would indicate that the users are not close and will thus not be recommended. The higher the score is, the higher the likelihood is for a friend recommendation to be made. More details on the Common Neighbors Approach can be found in the upcoming Section.

3.3. SHAMIR'S SECRET SHARING SCHEME

Shamir's Secret Sharing is a scheme that enables a private crypto key to be split into separate pieces, or shards, rendering each shard useless unless enough are assembled to reconstruct the original key. Shamir's Secret Sharing allows participants to share ownership of a secret by distributing shares; consequently, it prevents any single party from having sole ownership over the data. In order to gain access to the original secret, a minimum number of shares must be combined together; this is known as the threshold denoted as t. This scheme splits a secret S into n parts known as shares, such that with any k of n pieces (that satisfies the requirement of t), one can reconstruct the original secret. However, with any $k-1$ pieces, no information is exposed about S and the secret will not be able to be reconstructed. That is generally called a (n, k) threshold scheme [13].

3.4. RANDOM PERMUTATION

A permutation refers to an arrangement of elements. The number of permutations of n distinct objects is $n!$, which is the product of all positive integers less than or equal to n. So for 3 distinct objects, there are 6 total possible permutations as $3! = 6$

A random permutation is a permutation containing a fixed number of a random selection from a given set of elements. Permutation functions are used to strengthen the security of the user's data and to prevent data leakage. When implementing the friend recommendation process, the incorporating on random permutation functions prevents the unauthorized party from knowing which piece of data corresponds to which user in order to guarantee the privacy of the user's data. More details on the utilizing of random permutation functions can be seen in Section 5.

4. PROTOCOL FRAMEWORK

This section will discuss the framework of our protocol; it will elaborate on the model used, parties involved, metrics and algorithms used, privacy-preserving technique adopted as well as explain the justifications.

4.1. PROTOCOL MODEL

Our Protocol utilizes a centralized three party model consisting of two cloud servers (C_1 , C_2) and an Online Social Network Provider (OSNP). Through communication between C_1,C_2, and the OSNP, friend recommendations can be implemented in a privacy-preserving manner. The basic role of each of the parties is as follows:

- User: Each user's information will be stored in a matrix after being hashed. This is the only intervention the user will have in the friend recommendation process implemented by the $FSFR$ Protocol. Once the matrix is created, it is now ready to be split into shares. Each of the three parties of the model will be receiving a single share.
- C_1 and C_2: Once the shares are received by each of the parties, all the computations will take place via the federated cloud environment and the OSNP. Communications between all parties are secure and privacy-preserving.
- OSNP: The OSNP has the responsibility for providing the functionality to the user. The OSNP will perform computations on the shares it receives from the cloud servers. Moreover, the OSNP has a crucial role in the final steps of implementing the protocol when utilizing Paillier's Encryption. A detailed breakdown can be seen in Section 5.

4.2. TOPOLOGY-BASED ALGORITHM,

For our protocol, we adopt the topology-based algorithm which means that our protocol will rely on the topological structures of social networks and how users are interconnected on the network. The topological approach identifies existing links among users on a network in order to facilitate new ones. The proximity metric used in our topology based protocol is the Common Neighbors Score.

4.3. COMMON NEIGHBORS SCORE

One of the most notable metrics used to measure the proximity between any two users is obtaining their Common Neighbors score. Topology-based friend recommendation protocols are based on the Common Neighbors measure. This proximity measure provides a means to recommend friends to users in a privacy-preserving manner. In simple terms, this measure states that the closer one user is to another, the more likely they are to be recommended as a potential friend to them. The common neighbor approach also states that two strangers who have a friend in common are more likely to be introduced to each other than those who do not have any friends in common. If user A and a potential friend candidate, user B, have a common neighbors score of 10 and a threshold of 8, user A and B have a higher chance of being recommended to each other.

The figure below represent a sample toplogical network with a total of 9 users. Each user on this network has at least one friend while some users have multiple friends. Some of the users have common friends with other users. As mentioned in Section 2, the Common Neighbors Score can be computed by only considering two-hop neighboring nodes or nodes that are only two nodes apart from the target user.

We take User B as our target user that we would like to make friend recom-

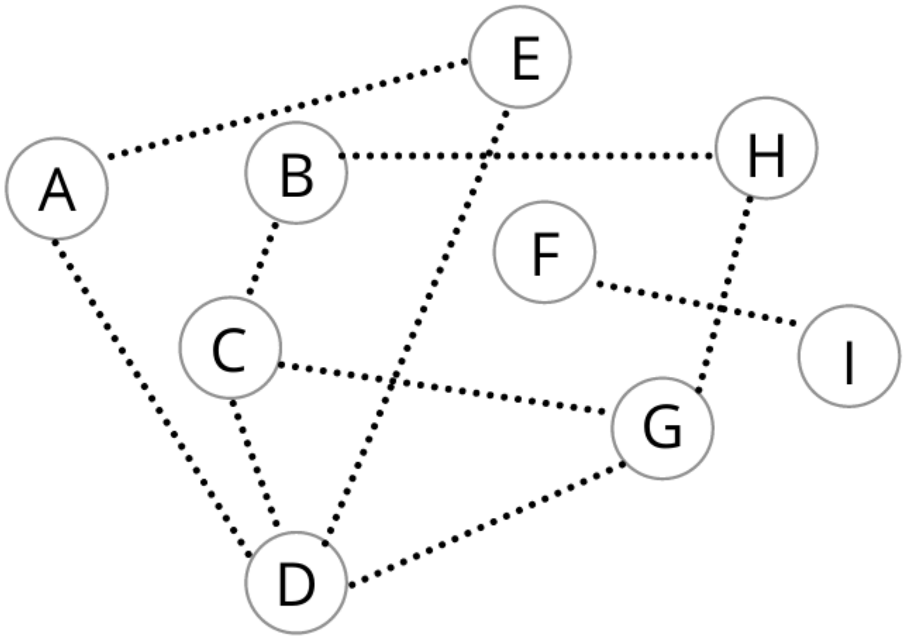


Figure 4.1: A sample topological network

mendations for.

The table shows how the Common Neighbors Score is found. We take each node x in the network and put in into the function $N(x)$. The function will note all the nodes adjacent to the specified node. We denote y as the target user; in this example, we seek to make friend recommendations for Node B. Thus, $N(y)$ lists the adjacent nodes of Node B. We will use this to compare it to other users on the network. We assume that no new friendships are being made or broken during the period of computing the Common Neighbors Score. Finally, the final column reflects the number of common nodes found between each x node and target user B. For node C and node H, note that a *null* value was stored. This is due to the fact that Node B already maintains a friendship with both nodes. Friend Recommendation are only to be applied to users who are not existing friends. If the friendship is to be

(x)	$N(x)$	$N(y)$	$S(x,y)$
A	E, D	H,C	0
B	H, C	H, C	N/A
C	B,D,G	H,C	null
D	A,C,E,G	H,C	1
E	A,D	H,C	0
F	I	H,C	0
G	H,C,D	H,C	2
H	B,G	H,C	null
I	F	H,C	0

Table 4.1: Common Neighbor Score Computations

broken, a new common neighbor score may be calculated.

4.4. SHAMIR'S SECRET SHARING

We use Shamir's Secret Sharing scheme in implementing our protocol. As briefly discussed in the Preliminaries, Shamir's Secret Sharing prevents a single party from having full authority over a secret by distributing shares among a number of users such that with any k of n pieces, the original secret can be reconstructed.

For instance, if S has been split into 7 shards and the scheme chooses $t = 4$, only 4/7 of the shards are required to reconstruct the original secret and the holders of any 4 shards can combine their shards for access. Any less number of shards will not allow access. Thus, if only 3/7 of the shards are combined, S will not be able to be reconstructed.

4.4.1. Multiplication and Addition.

Multiplication and addition are two forms of computations that could be implemented on shares under Shamir's Secret Sharing Scheme. The basis of these computations rely on them being performed on split data, known as shares.

Addition. An example of the implementation of addition under the scheme is shown below. Suppose we have two secret messages, 5 and 8. Let us denote P_0 for 5 and P_1 for 8. Under Shamir's Secret Sharing, we split both of the values into shares. We split P_0 into A_0 and A_1 and do the same to P_1 as shown below

$\boldsymbol{P_0 = 5}$	$\boldsymbol{P_1 = 8}$
a_0 = 3	b_0 = 9
a_1 = 2	b_1 = -1

As shown in the table above, Alice and Bob both have two split shares instead of their single original message. Now, Alice and Bob can send one of their own shares to each other, although this is not required. Alice and Bob own the shares as shown below:

$\boldsymbol{P_0}$	$\boldsymbol{P_1}$
a_0 = 3	a_1 = 2
b_0 = 9	b_1 = -1

Alice and Bob can do addition computations locally. The calculations expand to look like the equation below:

$$a + b = (a_0 + a_1) + (b_0 + b_1) \tag{4.1}$$

We can easily rearrange the order of the values into

$$a + b = (a_0 + b_0) + (a_1 + b_1) \tag{4.2}$$

P_0 will compute $(a_0 + b_0)$ using the shares it owns while P_1 will solve $(a_1 + b_1)$ using the shares it owns. Thus, by the end of the computation, P_0 will have only received one share from b while P_1 only gets one share of a. All computations are performed locally with no interactions between the parties after the initial exchange of shares. Most of the computations of our $FSFR$ protocol require only additive computations; however, some may require more complex operations such as multiplication to be implemented.

Multiplication. While doing additive computations are fairly simple, multiplicative computations are more complex because it requires for the parties to communicate during the computation. When applying multiplication to the shares, the formula is expanded as shown below:

$$a * b = (a_0 + a_1)(b_0 + b_1) = (a_0 * b_0) + (a_0 * b_1) + (a_1 * b_0) + (a_1 * b_1) \tag{4.3}$$

In the expanded formula above, $(a_0 * b_0)$ can be computed by P_0 locally while P_1 can compute $(a_1 * b_1)$ in the same manner. However, an obstacle is encountered upon trying to compute $(a_0 * b_1) + (a_1 * b_0)$. Neither party has both of the shares it is being asked to compute. In order to execute this computation, the party that attempts it will have to possess an additional share. For instance, if P_0 were to solve for $(a_0 * b_1)$, it would need to acquire b_1 from P_1. However, P_0 already has b_1 in it's ownership; acquiring the additional share, b_1, would allow for P_0 to have ownership over the complete secret as it now possessed all the shares required to reconstruct the message. This defeats the sole purpose of Shamir's Secret Sharing which prevents any single party of having complete ownership over the data.

In order to solve this problem, two steps are required; masking and the presence of an additional party. Through masking, we are able to introduce a new unknown number to each party. This number will be eliminated once the shares are combined. However, if either of the parties generate the unknown value, that would violate the privacy-preserving component of the protocol as the party that is generating the unknown number will have knowledge of the value of the number. Thus, a third party is required to generate these unknown numbers which will be utilized to hide the data the parties do not wish to disclose to one another. As a result, b_1 is masked from P_0 while a_0 is masked from P_1. These masks are denoted as s and t while the masked values are referred to as α and β

The multiplication of a * b performed by P_0 becomes:

$$z_0 = st_0 + (s_0 * \beta) + (\alpha * t_0) + (\alpha * \beta) \tag{4.4}$$

While the multiplication performed by P_1 becomes:

$$z_1 = st_1 + (s_1 * \beta) + (\alpha * t_1) \tag{4.5}$$

At this point, a third party is needed to create some values for masking; We denote this "helper" as P_2. This party will generate three new values. The first two values generated will be arbitrary while the third will be the product of the two numbers. As an implementation, let us assume P_2 choose random values of 7 and 11. The third value will be 77. The table below reflects the shares all parties currently own

P_0	P_1	P_2
$a_0 = 3$	$a_1 = 2$	$s = 7$
$b_0 = 9$	$b_1 =$ -1	$t = 11$
		$st = 77$

At this point, the P_2 values will be split into shares.

s	t
$s_0 = 4$	$t_0 = 5$
$s_1 = 3$	$t_1 = 6$

The way these values are used is that it is subtracted from the original data as follows

$$\alpha = (a_0 - s_0) + (a_1 - s_1) \tag{4.6}$$

$$\beta = (b_0 - t_0) + (b_1 - t_1) \tag{4.7}$$

Then, P_2 will send s_0 and t_0 to P_0 while it sends s_1 and t_1 to P_1. By the end of this step, each party has the following

At this point, the three values will be split into shares.

$\boldsymbol{P_0}$	$\boldsymbol{P_1}$
$a_0 = 3$	$a_1 = 2$
$b_0 = 9$	b_1 = -1
$s_0 = 4$	$s_1 = 3$
$t_0 = 5$	$t_1 = 6$

Now, P_0 can compute $(a_0 - s_0)$ in the equation for our α value and compute $(b_0 - t_0)$ for our β value while P_1 can calculate the $(a_1 - s_1)$ for the remaining computations of the α value and similarly compute $(b_1 - t_1)$ for the β value.

So,

$$\alpha = (a_0 - s_0) + (a_1 - s_1) = (3 - 4) + (2 - 3) = -2 \tag{4.8}$$

while

$$\beta = (b_0 - t_0) + (b_1 - t_1) = (9 - 5) + (-1 - 6) = -3 \tag{4.9}$$

The values of α and β are -2 and -3 respectively. P_0 and P_1 now uses the values to compute the final output. We revisit formulas 4.4 and 4.5 to calculate the following

$$z_0 = st_0 + (s_0 * \beta) + (\alpha * t_0) + (\alpha * \beta) \tag{4.10}$$

$$\mathrm{z}_0 = 44 + (4 * -3) + (-2 * 5) + (-2 * -3)$$

$$\mathrm{z}_0 = 28$$

While the computation performed by P_1 becomes:

$$z_1 = st_1 + (s_1 * beta) + (alpha * t_1) \tag{4.11}$$

$$z_1 = 33 + (3 * -3) + (-2 * 6) \tag{4.12}$$

$$z_1 = 12 \tag{4.13}$$

Finally, we combine the values after combining the shares to get:

$$z_0 + z_1 = 28 + 12 = 40$$

which is the product of the value of the original secret.

4.5. COMPARISON BETWEEN SHAMIR'S SECRET SHARING AND HOMOMORPHIC BASED ENCRYPTION

4.5.1. Homomorphic Encryption.

1. Size: Homomorphic encryption has a larger computational overhead than plain-text operations. Also, homomorphic encryption based schemes requires to individually encrypt each input in it's own cipher-text.

2. Computation: Public key encryption schemes are based on computationally difficult problems and thus, require expensive operations such as modular exponentiation. It also makes it commercially infeasible for computationally-heavy applications.

4.5.2. Shamir's Secret Sharing Scheme.

1. Size: This scheme creates n shares of each input, where each share is of the same size as the secret.

2. Computation: the computations under this scheme consists of choosing a random polynomial and evaluating it in n points. When seeing this computation graphically, the polynomial is chosen along the same function of graph of the secret itself; this means that usually all computations are done with ordinary integers [15].

One of the greatest advantages of Shamir's Secret Sharing is that the individual shards or pieces can be dynamically added or deleted without affecting the other pieces. It is also dynamic as security can be easily enhanced without changing the secret; this is done by simply changing the polynomial and constructing new shares to the participants. It is also flexible as it can be organized based on hierarchy; we can supply each participant different number of pieces according to their importance inside the organization [15].

5. PROPOSED $FSFR$ PROTOCOL

The protocol we propose requires a number of communications between our three parties: C_1, C_2 and the OSNP. The steps of $FSFR_1$ and $FSFR_2$ are defined in this section. Before the communications among the OSNP and cloud servers begin, the user's friend-lists from the matrix must be split and outsourced as shown in Fig. 5.1.

5.1. $FSFR_1$

1. All parties know n users
   - In this model, we use 2 Cloud Servers as well as an OSNP; all three parties know the users that are selected.
   - Each user will split their matrices and send them to C_1, C_2 and the OSNP. (we do not encrypt them as done in PAFR protocol)
   - All parties will now have their respective shares.
2. The OSNP wants to make friend recommendations for n-users. It then picks shares of corresponding matrices of those n users (their unique IDs) and sends it to C2
   - C_2 and C_1 agree on a permutation that is only known to themselves exclusively.
3. C_2 will add those shares with its own corresponding shares
4. Now, C_1 and C_2 will separately permute those matrices shares and send them to the OSNP.

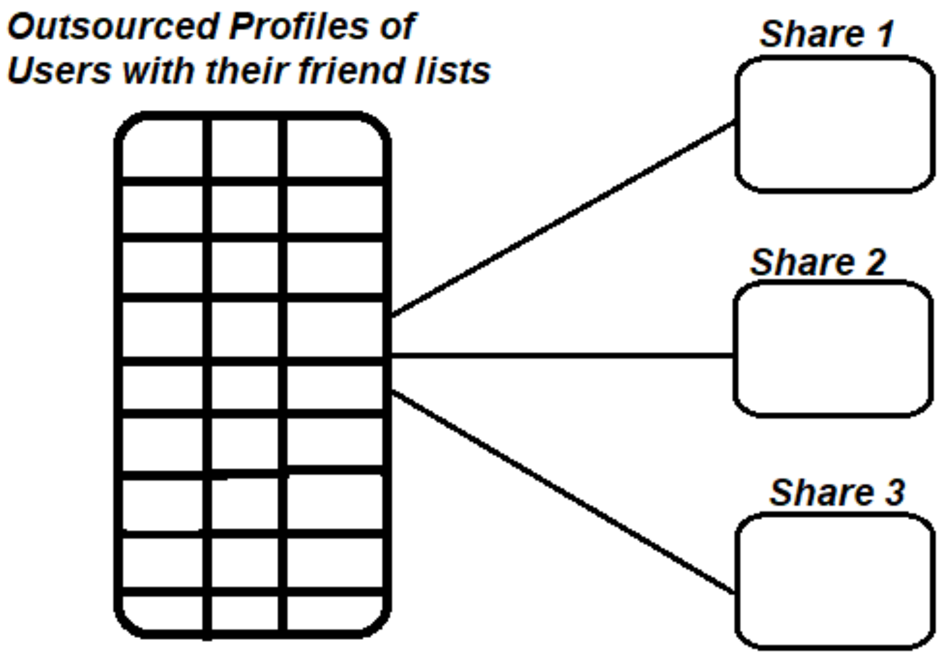


Figure 5.1: Users outsourcing data

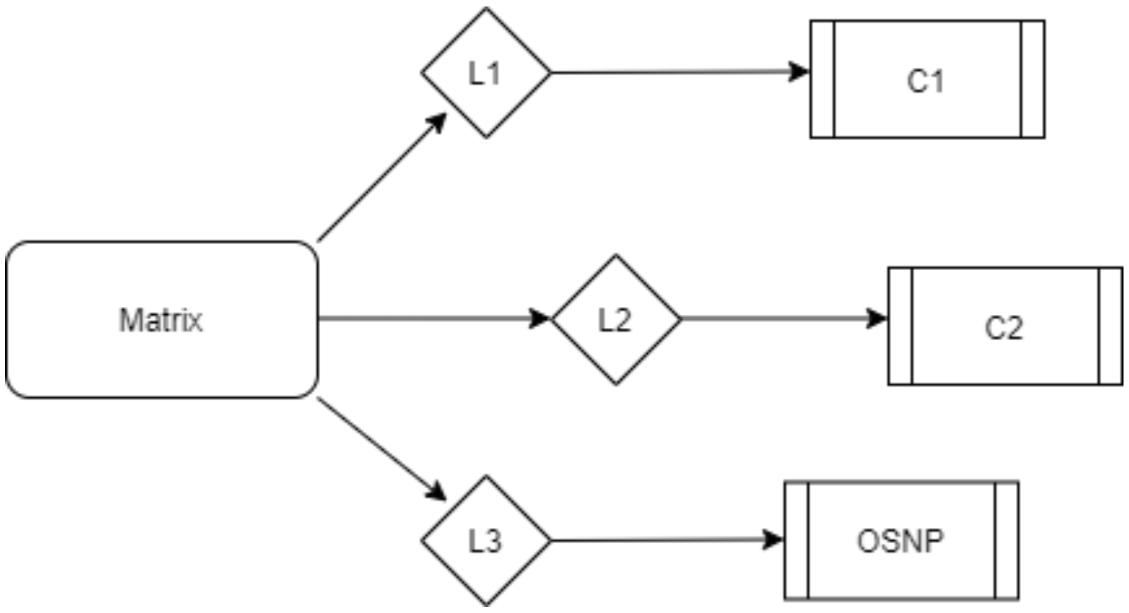


Figure 5.2: Sending matrix shares to the parties

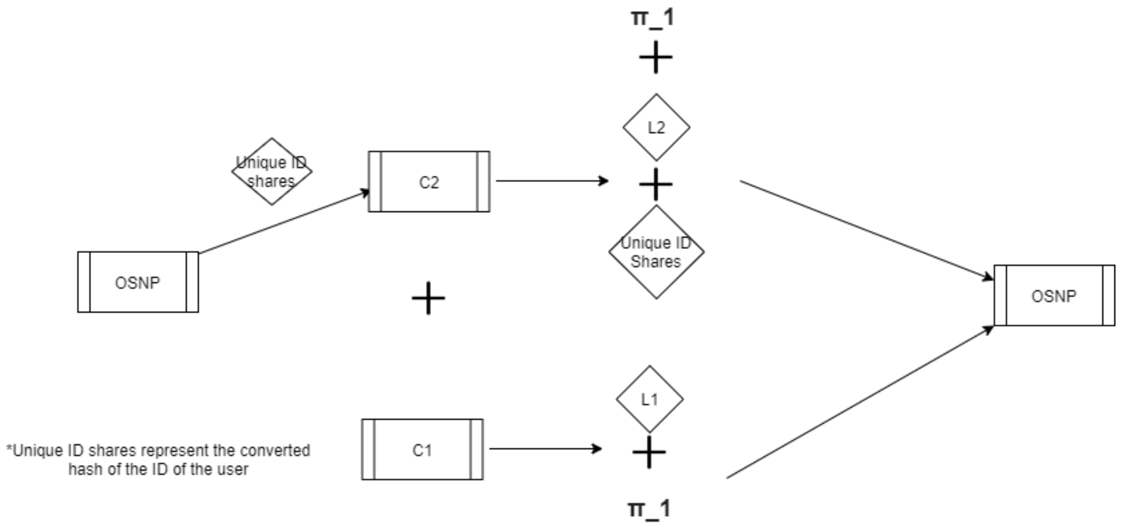

π_1
L2
Unique ID shares
C2
Unique ID Shares
OSNP
OSNP
L1
*Unique ID shares represent the converted hash of the ID of the user
C1
π_1

Figure 5.3: Permutation on shares from the OSNP

- C_2 and C_1 apply a permutation to the share they receive from the OSNP and sends them separately/individually to the OSNP. We refer to the permutation function as π_1.
- The permutation value is added to each entry in the first column of the matrices to prevent the OSNP from obtaining information on the correspondence of friend-lists.

5. The OSNP reconstructs the matrices and identifies the permuted friend-lists.
   - In this step, the OSNP will know the matrices and friend-lists but in its permuted form.
6. The OSNP permutes the lists again and sends them back to both C_1 and C_2.
   - Each list will consist of users and the aggregated matrices
7. C_1 and C_2 will separately add the shares of the corresponding matrices locally.

- Both C_1 and C_2 will know that the friend-list belongs to some users (permuted n-lists)

8. C_1 will take all the users in the friend-lists and add their matrices/combine them then send its aggregated shares to C_2.

9. C_2 combines them with its own aggregated shares and send it to the OSNP.

   - C_1 and C_2 will now have one aggregated matrix.

10. The OSNP does inverse permutation, combines them with its own shares (now possessing all of the shares) and checks the frequencies in the final matrix.

    - The OSNP decrypts the second columns and checks whether the common neighbors score frequency is greater than or equal to t. If it is, the corresponding randomized ID will be added to the new friend-list.
    - The OSNP now knows the permuted new friend-lists for n users but it does not know which friend-list corresponding to which user due to the permutation.

11. The OSNP sends the list to C_1.

    - Each ID is split among the parties (C_1 and C_2). C_1 and C_2 cannot collude.

12. C_1 inversely permutes the list to obtain the correct order of the randomized new friend-lists.

    - Now, C_1 has the correct order of the new friend-list; however it is in it's randomized form.

13. We want to get the friend-list non-randomized AND in correct order.

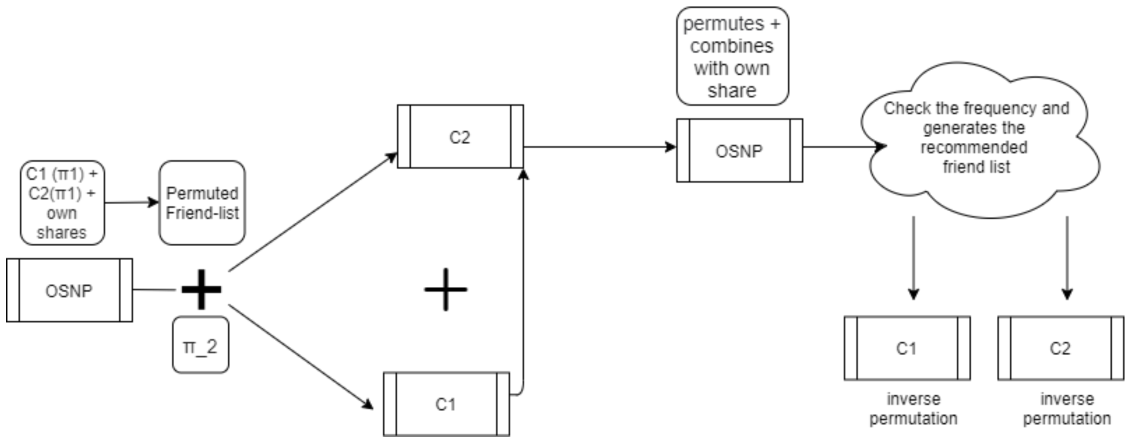

permutes + combines with own share
C2
OSNP
Check the frequency and generates the recommended friend list
C1 (π1) + C2(π1) + own shares
Permuted Friend-list
OSNP
π_2
C1
C1
inverse permutation
C2
inverse permutation

Figure 5.4: OSNP permutation and frequency computation

A second solution was developed utilizing a hybrid approach of the combination of Shamir's Secret Sharing scheme with Paillier's encryption and utilizes random numbers. The Paillier cryptosystem is advantageous as it provides fast encryption and decryption algorithms.

Random numbers are used to inject unpredictability or non-deterministic data to make the resulting data virtually unguessable. In this case, we encrypt the random values with the friend-lists. The goal of this modified solution is to allow C_2 to be able to receive the random numbers in the order of the OSNP's permutation function.

5.2. $FSFR_2$

1. All parties know n users

   - In this model, we use 2 Cloud Servers as well as an OSNP; all three parties know the users that are selected.
   - Each user will split their matrices and send them to C_1, C_2 and the OSNP. (we do not encrypt them as done in PAFR protocol)
   - All parties will now have their respective shares.

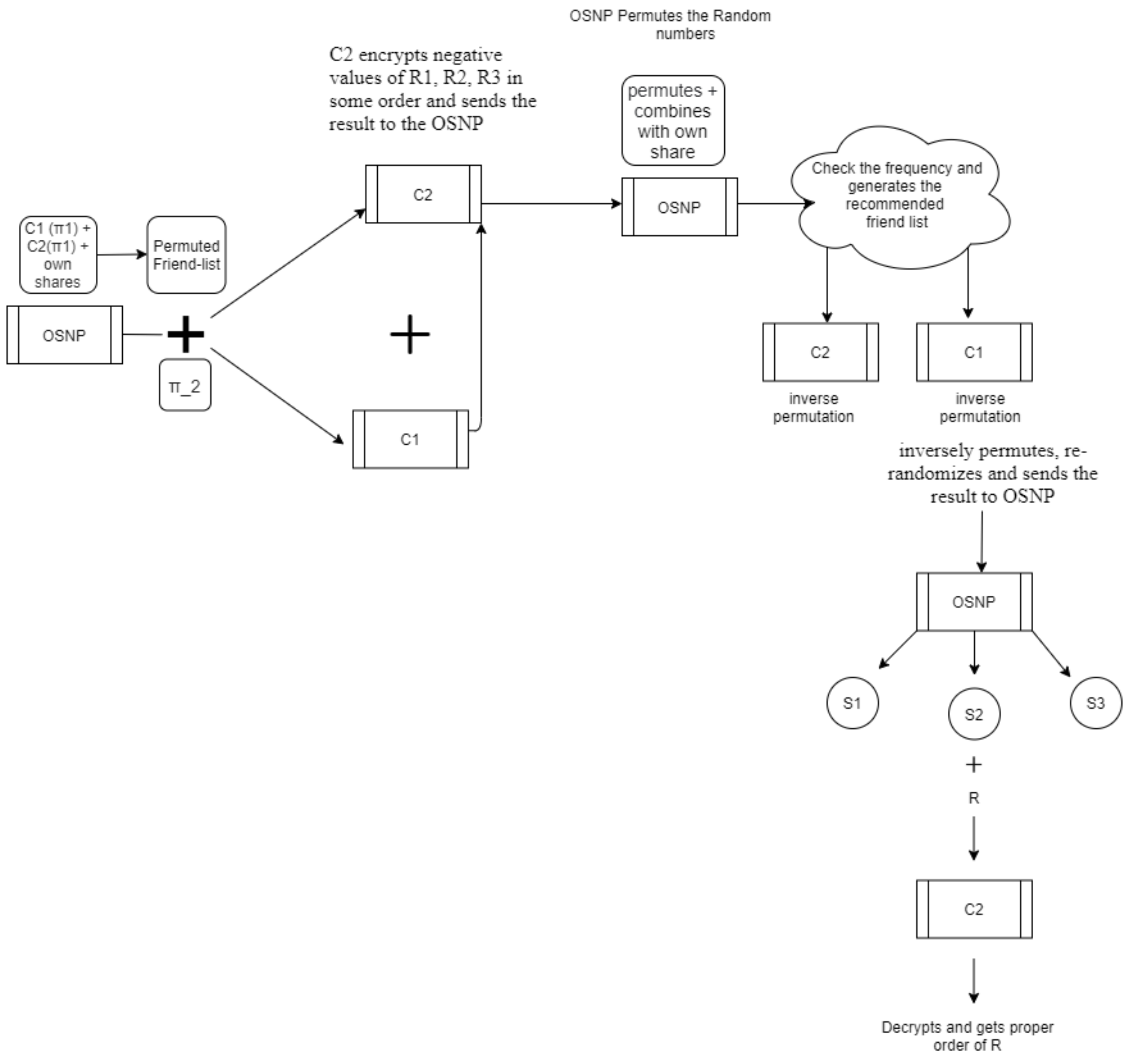

OSNP Permutes the Random numbers
C2 encrypts negative values of R1, R2, R3 in some order and sends the result to the OSNP
permutes + combines with own share
C1 (π1) + C2(π1) + own shares
Permuted Friend-list
OSNP
π_2
C2
C1
OSNP
Check the frequency and generates the recommended friend list
C2
C1
inverse permutation
inverse permutation
inversely permutes, re-randomizes and sends the result to OSNP
OSNP
S1
S2
S3
+
R
C2
Decrypts and gets proper order of R

Figure 5.5: $FSFR_2$

2. The OSNP wants to make friend recommendations for n-users. It then picks shares of corresponding matrices of those n users (their unique IDs) and sends it to C2

   - C_2 and C_1 agree on a permutation that is only known to them exclusively.

3. C_2 will add those shares with its own corresponding shares

4. Now, C_1 and C_2 will separately permute those matrices shares and send them to the OSNP.

   - C_2 and C_1 apply a permutation to the share they receive from the OSNP and sends them separately/individually to the OSNP. We refer to the permutation function as π_1.
   - The permutation value is added to each entry in the first column of the matrices to prevent the OSNP from obtaining information on the correspondence friend-lists.

5. The OSNP reconstructs the matrices and identifies the permuted friend-lists.

   - In this step, the OSNP will know the matrix and friend-lists but in its permuted form.

6. The OSNP permutes the lists again and sends them back to both C_1 and C_2.

   - Each list will consist of users and the aggregated matrices

7. C_1 and C_2 will separately add the shares of the corresponding matrices locally.

   - Both C_1 and C_2 will know that the friend-list belongs to some users (permuted n-lists)

8. C_1 will take all the users in the friend-lists and add their matrices/combine them then send its aggregated shares to C_2.

9. C_2 combines them with its own aggregated shares and encrypts negative values of random numbers in some order using its own Paillier's public key and sends them to the OSNP. Paillier's scheme are known for their efficiency; they also allow to achieve the highest security level for homomorphic encryption schemes.

    - We encrypt using C_2's public key rather than the OSNP's because our final destination is C_2; we want C_2 to receive the values in proper order by the end. If our final goal was for the OSNP to receive the values in proper order, we would encrypt using the OSNP's public key. However, C_2 cannot let the OSNP know the value of R. Thus, we encrypt it as described.

10. The OSNP does inverse permutation and combines them with its own shares and checks the frequencies in the final matrix before sending it to C_1.

    - The OSNP now has the encrypted permuted and randomized new friend-list; it also has the encrypted random numbers in correct order.

11. C_1 performs an inverse permutation on the encrypted list, re-randomizes it and sends the result to the OSNP.

    - C_1 now knows the randomized friend-list
    - After this step, the OSNP will not be able to compare the friend-list or encounter any information leakage due to the re-randomization that took place in previous step –the OSNP now has the encrypted random numbers with the randomized list.

12. The OSNP will randomly split the encrypted values into shares

    - The OSNP adds new random numbers (R) and send the results to C_2
    - C_2 decrypts the values and gets its share (L2) while OSNP retrieves it's own share (L3).

- C_2 decrypts with the private key that is known only to itself
- L2 + L3 = negative value of the random numbers (what C_2 encrypted)

13. C_2 decrypts and gets the random numbers in correct order.

14. Three parties will send their shares individually to the user.

5.3. SECURITY ANALYSIS

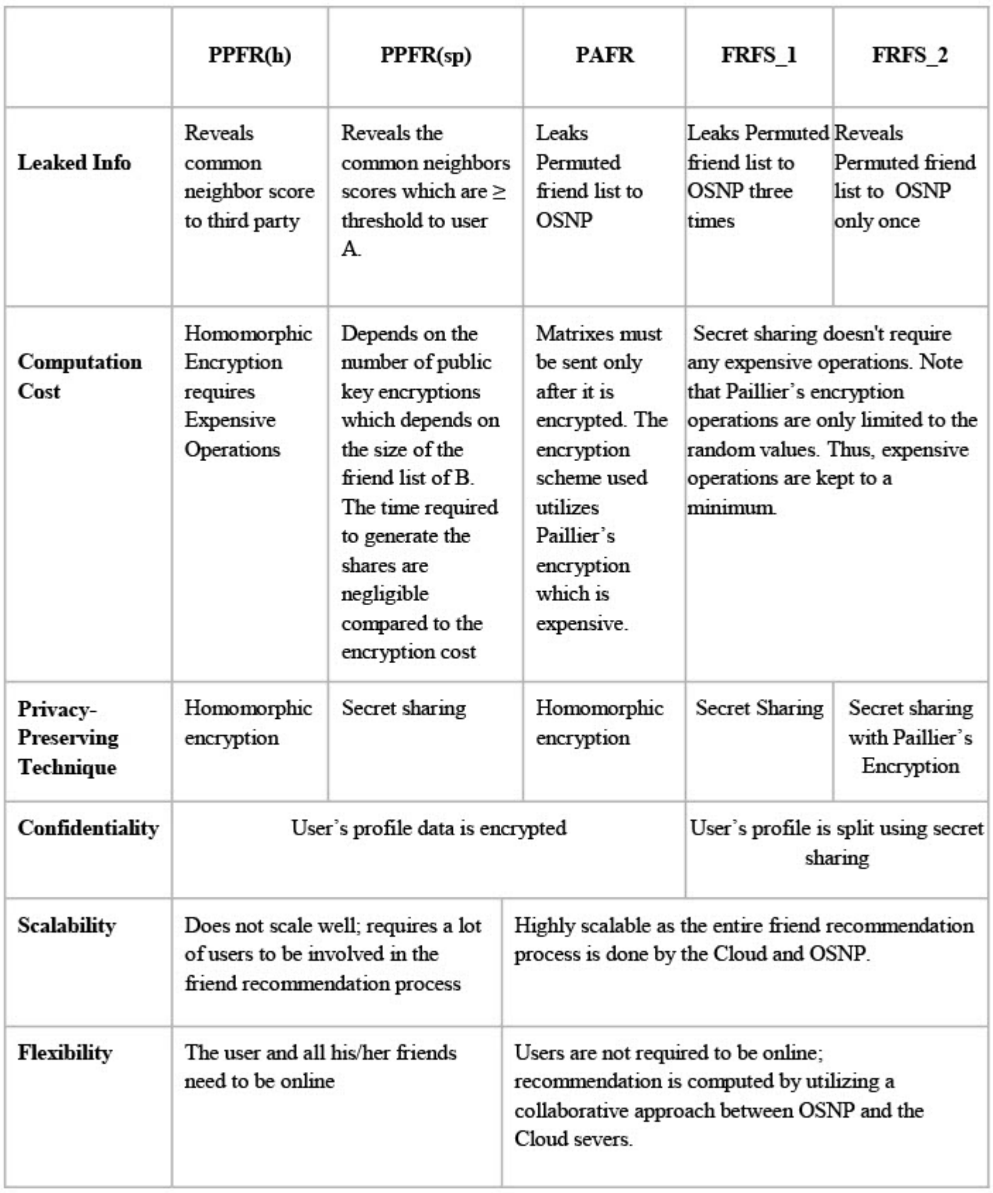
Our protocol was able to eliminate one crucial information leakage through developing $FSFR_2$. However, we were unable to successfully eliminate a a case of information leakage in Step 5 and Step 6, although is arguably less significant. In Step 5 of $FSFR_1$, the OSNP is able to identify the friend-lists in its permuted form. Although it is in its permuted form, there is still information leakage taking place as the OSNP can identify the permuted friend-list. Likewise, in Step 6, the permuted friend-list is being vulnerable to leakage once it is sent to C_1 and C_2. In Step 10, once the OSNP performs inverse permutation and combines the shares received from C_2 (which has previously been combined with shares from C_1), the permuted friend-list will be exposed and susceptible to information leakage once again. In all three instances, there is a possibility of user privacy being jeopardized. However, the potential leakage in Step 10 was able to be eliminated through the utilization of random values in our modified solution, $FSFR_2$. The reason we were only able to eliminate the weakness in Step 10 is that we could add random numbers only because no computations needed to be performed involving the contents of the friend-list. However, in Step 5 and 6, the cloud needs to know which matrices to choose to begin computing the common neighbor score. It can not randomly choose the matrices; thus, we could not add a random value to it. Future research on eliminating the leakages in Step 5 and 6 could strengthen the privacy-preserving component of the protocol.

6. PERFORMANCE EVALUATION

$PPFR_h$ provides a strong security guarantee and utilizes a universal hash function for improving the efficiency. However, this efficiency comes at the expense of degraded accuracy due to the involved hash collisions. On the other hand, the second method utilizes the concept of protecting the source privacy through anonymous message routing and recommends friends accurately. Both of the proposed protocols preserve the privacy of each user. Nonetheless, both protocols leak different additional information. While the hashing and random permutation prevents the third party from identifying the actual source of the scores, the $PPFR_h$ protocol leaks the common neighbors scores to a third party while the $PPFR_{sp}$ protocol reveals the common neighbors scores which are more than the threshold to user A. This may allow a significant amount of information to be deduced. However, since the protocol guarantees the source privacy, A cannot determine the sources corresponding to the scores.

Our protocol protects the friend-lists and user profiles from being leaked to any party in a secure and efficient manner. In addition, we do not utilize any expensive computations other than encrypting the random values with Paillier's encryption for $FSFR_2$. Moreover, computations can be performed conveniently while preserving the privacy of all the shards that make up the secret as Shamir's Secret Sharing scheme allows for computations to be performed of split shares.

The table below reflects some key differences among the protocols discussed in this thesis.



	PPFR(h)	PPFR(sp)	PAFR	FRFS_1	FRFS_2
Leaked Info	Reveals common neighbor score to third party	Reveals the common neighbors scores which are ≥ threshold to user A.	Leaks Permuted friend list to OSNP	Leaks Permuted friend list to OSNP three times	Reveals Permuted friend list to OSNP only once
Computation Cost	Homomorphic Encryption requires Expensive Operations	Depends on the number of public key encryptions which depends on the size of the friend list of B. The time required to generate the shares are negligible compared to the encryption cost	Matrixes must be sent only after it is encrypted. The encryption scheme used utilizes Paillier's encryption which is expensive.	Secret sharing doesn't require any expensive operations. Note that Paillier's encryption operations are only limited to the random values. Thus, expensive operations are kept to a minimum.	
Privacy-Preserving Technique	Homomorphic encryption	Secret sharing	Homomorphic encryption	Secret Sharing	Secret sharing with Paillier's Encryption
Confidentiality	User's profile data is encrypted			User's profile is split using secret sharing	
Scalability	Does not scale well; requires a lot of users to be involved in the friend recommendation process		Highly scalable as the entire friend recommendation process is done by the Cloud and OSNP.		
Flexibility	The user and all his/her friends need to be online		Users are not required to be online; recommendation is computed by utilizing a collaborative approach between OSNP and the Cloud severs.		

Table 6.1: Protocol Comparison: $PPFR_h$, $PPFR_{sp}$, $PAFR$, $FSFR_1$ and $FSFR_2$

7. CONCLUSION

7.1. SUMMARY

Due to increased participation and growth of Online Social Networks, many people are concerned about their privacy and how well it is being preserved. People want to interact freely with other users without having to worry about their privacy being jeopardized; they want to be guaranteed that their information is kept private and undisclosed to anyone, including any third party. This proves to be a challenge as user information is needed to compute different types of data on OSN. A key example of this is the friend recommendation feature.

While users are capable of manually creating friendships with other users by going to the profile of the user they wish to befriend, social media platforms like Twitter, Instagram and Facebook utilize an effective ”automatic” feature known as friend recommendation. Although this seems automatic and may be perceived to be random, it is far from being an arbitrary procedure. Friend recommendations on an OSN is carried out through the implementations of protocols and algorithms in a measured and detailed manner. Several protocols exist to allow for the implementations of friend recommendations. Nonetheless, due to one reason or another, they fail to meet stringent user-privacy standards. This calls for the need to develop more secure and efficient friend recommendation protocols which this thesis develops and elaborates on.

We refer to three existing friend recommendation protocols known as $PPFR_h$, $PPFR_{sp}$ and $PAFR$ and identified weaknesses with each protocol. We were able to successfully eliminate a number of key weaknesses while increasing efficiency and strengthening the privacy-preserving components through developing the $FSFR$ protocol. Our protocol adopts a Federated cloud environment which coordinates with an OSNP to carry out computations on shares in a cooperative manner using Shamir's Secret Sharing scheme. By incorporating a collaboration of an OSNP with multiple public cloud environments, our protocol sees more balanced and comparatively equally distributed computation across the parties.We also performed a comparative analysis between the protocols and scrutinized their security implications, efficiency, performance, cost and more.

7.2. FUTURE WORK

As we have yet to actually implement the FSFR protocol, future research could focus on the execution of the friend recommendation protocol through adopting the $FSFR$ protocol. The resulting performance details would yield more accurate results. In Section 5, we discuss the information leakages that may take place in Step 5 and 6. Developing a method to prevent these leakages would help make the protocol more secure. Another direction of research could be based on developing protocols that utilizes users content based algorithms to recommend friends to users using Shamir's Secret Sharing scheme. Under such a framework, an algorithm that considers users' hobbies, education and age will be required. Another direction that can be pursued to expand this research is to develop a Secret Sharing based protocol that does not rely on any form of encryption whatsoever. Although our protocol relies fundamentally on Shamir's Secret Sharing, $FSFR_2$ adopts a hybrid framework This would require to discover a way to allow the secret shares to be received in it's original order.

REFERENCES

[1] H. Tankovska. Social media usage in u.s., Apr 2021.

[2] Ahmad Ali, Ahmad Kamran, Mansoor Ahmed, Basit Raza, and Muhammad Ilyas. Privacy concerns in online social networks: A users' perspective. *International Journal of Advanced Computer Science and Applications*, 10(7), 2019.

[3] Yalin E. Sagduyu, Alexander Grushin, and Yi Shi. Synthetic social media data generation. *IEEE Transactions on Computational Social Systems*, 5(3):605–620, 2018.

[4] Ahmed Al Faresi, Ahmed Alazzawe, and Anis Alazzawe. Privacy leakage in health social networks. *Computational Intelligence*, 30(3):514–534, 2013.

[5] Emma Bowman. After data breach exposes 530 million, facebook says it will not notify users, Apr 2021.

[6] Mona Alkanhal and Bharath K. Samanthula. A privacy-aware framework for friend recommendations in online social networks. *2019 IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC)*, 2019.

[7] Bharath K Samanthula, Lei Cen Wei Jiang, and Luo Si. Privacy-preserving and efficient friend recommendation in online social networks. *TRANSACTIONS ON DATA PRIVACY*, 2AD.

[8] Hongchao Zhou and Gregory Wornell. Efficient homomorphic encryption on integer vectors and its applications. *2014 Information Theory and Applications Workshop (ITA)*, 2014.

[9] A. Machanavajjhala, A. Korolova, and A.D. Sarma. Personalized social recommendations: accurate or private. *Proc. VLDB Endowment*, 4:440–450.

[10] Xueying Yu and Shudong Yang. Friend recommendation mechanism for social media based on content matching. *2019 International Conference on Communications, Information System and Computer Engineering (CISCE)*, 2019.

[11] Shirin Nilizadeh, Sonia Jahid, Prateek Mittal, Nikita Borisov, and Apu Kapadia. Cachet. *Proceedings of the 8th international conference on Emerging networking experiments and technologies - CoNEXT '12*, 2012.

[12] Ariel Joseph Feldman. *Privacy and Integrity in the Untrusted Cloud.*

[13] Andreas Pogiatzis. Shamir's secret sharing - a numeric example walkthrough, Apr 2020.

[14] Leucio Antonio Cutillo, Refik Molva, and Thorsten Strufe. Safebook: A privacy-preserving online social network leveraging on real-life trust. *IEEE Communications Magazine*, 47(12):94–101, 2009.

[15] Thomas B. Pedersen and Yücel Saygın. Secret sharing vs. encryption-based techniques for privacy preserving data mining.