Privacy preservation in peer-to-peer gossiping networks in presence of a passive adversary

Mohammad Al-Aggan

Privacy preservation in peer-to-peer gossiping networks in presence of a passive adversary *

Mohammad ALAGGAN †
Supervisors: Anne-Marie Kermarrec, Sébastien Gambs
INRIA - ASAP Research Team
Masters by Research in Computer Science (IFSIC)

June 24, 2010

A thesis submitted in partial fulfillment of the requirements for the degree of
Masters by research in Computer Science
from
IRISA / Université Rennes 1

Abstract

In the Web 2.0, more and more personal data are released by users (queries, social networks, geo-located data,...), which create a huge pool of useful information to leverage in the context of search or recommendation for instance. In fully decentralized systems, tapping on the power of this information usually involves a clustering process that relies on an exchange of personal data (such as user profiles) to compute the similarity between users. In this internship, we address the problem of *computing similarity between users while preserving their privacy* and *without relying on a central entity*, with regards to a passive adversary.

*This research is supported by the Gossple ERC Starting Grant number 204742.
†Funded by a grant from Fondation Michel Métivier.

Contents

Introduction

Context. Our work addresses privacy concerns in similarity computation in a fully decentralized distributed system in the context of the Gossple project [1]. In this section we introduce the Gossple project and gossiping networks as well as the privacy concerns associated with similarity computations in such networks.

Gossple. Gossple [2] is a social network, that depends on automatically grouping acquaintances who share similar interests. The main interest of this step is to make use of the relevant search experience of similar acquaintances to help other people in finding more personalized and relevant search results. Gossple is implemented as a fully decentralized distributed system. Decentralization brings two advantages, the first one is scalability, basically by using the collective resources of the entire network. The second advantage is to prevent the "big-brother" syndrome; as each user controls her own information and there is no central entity in control of the information about everyone. The main idea behind Gossple is automatically harvest the network for similar users. Then, by exchanging information with those users, a personalized query response will be more viable.

As Gossple is a social network (or is designed to be an extension of a pre-existing one), it assumes that users have profiles. Those profiles are usually a list of items the user have showed an interest in. For instance, in the case of Delicious[1], an item corresponds to a specific Web page and/or the set of tags used to tag the Web page collection.

Gossple is designed as a *gossiping* network, which is a special form of distributed systems where there is no centralized coordinator, and where the network has no specific structure [3]. Since a gossiping network is a form of peer-to-peer (P2P) systems, we are going to adopt the name *"peer"* instead of "user" for the rest of this document. In a gossiping network, peers keep meeting new peers drawn randomly from the network [4]. In Gossple, at each exchange peers measure their similarity to decide whether (or not) to become acquaintances.

In this internship we have mainly concentrated on computing similarity between peers. Moreover, we are focusing on what information about peers' private data can leak through this process and how to address this problem. More precisely, we address two different and orthogonal definitions of privacy.

Privacy. Gossple has already developed some protocols to achieve gossiping-based similarity matching between peers. Experiments have shown that they are converging and provide reasonable correctness [2]. One withstanding concern however was about the privacy of the peers' data exchanged in order to perform the matching. The protocol requires the users to exchange their profiles, which might be considered by some users as private information.

What exactly does "Privacy" mean ? There is two common definitions in the literature. We first define the context in which the privacy is defined: there is the *private input data*, there is a *function* to be computed on these data, and there is the *public output* of the function which is intended to be released at the end of the computation.

[1] http://www.delicious.com.

In the traditional cryptographic definition [5], privacy means that anything that cannot be computed from the public output regarding the private input of one of the participants, should stay so after the output is released. This is equivalent to say that anything that can be deduced from the public output is not considered a privacy leak. The cryptographic model focuses on insuring that the *steps* in computing the function itself are secure, i.e. that the private input is not leaked *during* the computation. Whatever happens after the computation ends and the result is released is not a concern there.

On the other hand, the *differential privacy* setting [6] assumes that the function itself is *already* securely computed by a trusted curator. Differential privacy then focuses on what happens *after* that: what does the function output itself reveal about the private input, and how to protect this input. Protecting the input (or preserving its privacy) in this context means that whatever the attacker (who receives the output) knows about a single item in the profile after receiving the output, should not differ too much (up to a privacy parameter) whether or not the input did actually contain that item or not (hence the name "differential"). Of course, the differential privacy makes sense only when the input constitutes of several items and we want to protect the privacy of individual items while allowing capturing global properties (such as sum, average, or similarity). In particular, this applies in the case of protecting the privacy of profiles, because a profile contains several interests, and we want to protect the privacy of individual interests.

As we can see, the two approaches are complementary, and we shall address them in Section 2 and Section 3, respectively. Before that, we provide the necessary background on these notions in Section 1.2.

Internship contribution. In this internship, we propose a protocol based on cryptographic tools that computes the similarity between two user profiles in such a way that each user only learns the output of the similarity computation but not the profiles themselves. The novelty of our approach is twofold. Firstly, we introduce a secure protocol for *threshold similarity*. This protocol outputs only one bit of information stating whether or not two users are similar beyond a predetermined threshold. Compared to the *exact similarity* computation from which more information can be extracted, this protocol is more "privacy-preserving" as it reveals less information. Second, we go beyond the traditional cryptographic framework by analyzing the similarity computation within the context of *differential privacy*.

We have submitted a paper with the contribution of this internship to the Distributed Computing conference (DISC 2010) entitled "Private similarity computation in distributed systems: from cryptography to differential privacy".

Technical challenges. We address the following technical challenges. Considering a well-known similarity metric, namely cosine similarity [7], we propose a two-party threshold similarity protocol for this metric and prove its correctness and security (for all the variants we propose) against a passive adversary. While we focus on the cosine similarity for illustration purpose, our method is generic enough to be applied to other metrics such as Jaccard index [8].

Afterwards, we design a differentially-private protocol for the exact similarity and threshold similarity and analyze its impact with respect to utility. To the

best of our knowledge, this is the first attempt to address differential privacy in the context of distributed similarity computation. More specifically, we first analyze the sensitivity of the similarity metric in the context of a protocol computing exactly the similarity between two profiles. We then show that the (Laplacian) noise required to ensure differential privacy can be applied before the application of the threshold function without affecting the differential privacy property of the protocol.

We also study the impact of the differential privacy (which requires the addition of random noise) on the resulting *utility* of the similarity measure. The achievable trade-off between utility (the percentage of threshold decisions that will be correct [2]) and privacy depends on: the value of the threshold, the size of the item domain, and ϵ (the privacy parameter). We use the utility also with help of a statistical model to help in choosing the threshold value for the threshold similarity. We also propose a variant of (global) sensitivity computation which we call the *parametrized sensitivity* to enhance the utility and we prove that it makes the differential privacy guarantees hold.

We show that in a gossiping-based model to retain the differential privacy guarantees, it is necessary to assume the existence of a bidirectionally anonymous channel. We also provide a candidate implementation for such channel, which is called "gossip-on-behalf".

We finally address the distributed nature of the systems by providing three different mechanisms for distributed noise generation. More specifically, in two of them, each peer involved in the two-party computation can generate noise *independently* and we then show that by applying these noises we reach differential privacy guarantees. In the third mechanism we use a *semi*-trusted third-party to add the noise instead.

The report is organized as follows. Firstly, Section 1 provides the required background and some preliminaries. In Section 2, we introduce the threshold similarity protocol and prove its security with respect to a passive adversary. Afterwards in Section 3, we describe differentially-private protocols for the exact and threshold similarity and provide the utility analysis. Finally, we conclude in Section 4.

1 Preliminaries

1.1 System model

Notation. Let "$a \longleftarrow A$" denotes that a is uniform-randomly chosen from the set A. Let $a \twoheadleftarrow_{\sim b} A$" means that a is chosen from A because a maximizes a given similarity measure with respect to b. Let the domain of items be $\mathbb{I}$.

The indicator function $\mathbf{1}_S$, (sometimes called the characteristic function of the set S) is normally a function defined on the the set S such that $\mathbf{1}_S : S \to \{0, 1\}$. $\mathbf{1}_S(x) = 1$ if $x \in S$, and zero otherwise. Assuming the set's domain $\mathbb{I}$ is ordered, we abuse the notation for the rest of this report and we refer to the bit vector $\mathbf{1}_S(\mathbb{I})$ simply as the "indicator function" $\mathbf{1}_S$ of the set S. For example

[2] There is two types of errors, false positive and false negative. Our utility function measures the probability of not having false negatives. This corresponds to the probability that a decision which was a negative after applying differential privacy was also a negative before applying it.

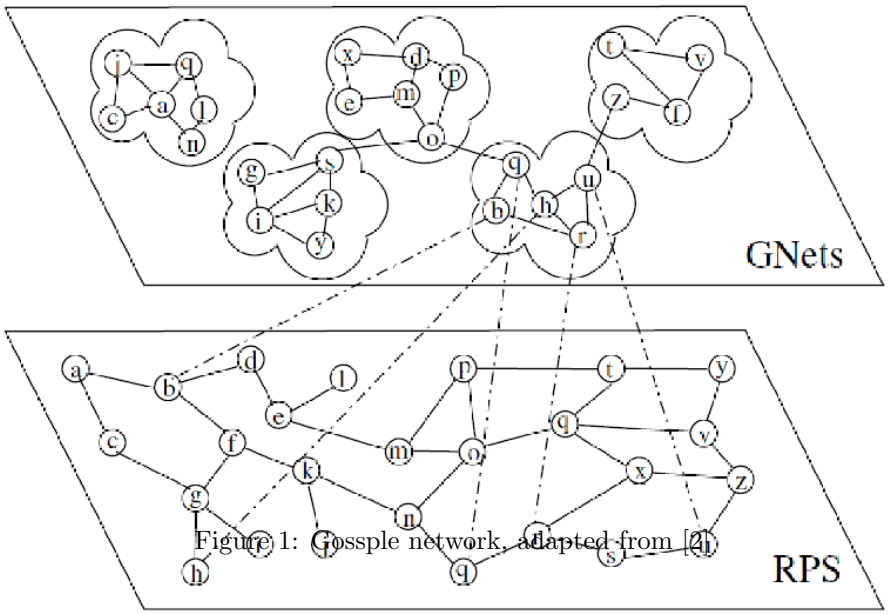


Figure 1: Gossple network, adapted from [2]

if the ordered domain $\mathbb{I}$ is $\{cat, dog, mouse\}$ and the set S is $\{dog\}$, then the indicator function $\mathbf{1}_S = (0, 1, 0)$.

A similarity-based gossiping network consists a set of peers N. That set changes over time due to arrivals and departures. Each peer $i \in N$ is a tuple $\langle \mathcal{P}, \mathcal{RS}, \mathcal{PN} \rangle$. Where:

- The peer's profile of interests is the set $\mathcal{P}$ such that $\mathcal{P} \subseteq \mathbb{I}$ is the peer's profile of interests.
- $\mathcal{RS}$ is a random set of peers, constituting the random sampling layer [4].

$$\mathcal{RS} = \{ac_j : j \longleftarrow N\} \ ,$$

  where ac_j is a bidirectionally anonymous channel to peer j (Section 1.2.2).

- $\mathcal{PN}$ is the *personal network* of the peer, chosen based on similarity that the peer uses to pick his social acquaintances.

$$\mathcal{PN} = \{ac_j : j \twoheadleftarrow_{\sim i} \mathcal{RS}\} \ .$$

Such a model can be seen in Gossple as represented in Figure 1. The figure is composed of two layers, the first layer, called RPS (Random Peer Sampling), corresponds to each peer's $\mathcal{RS}$ in our model whereas in the other layer, GNets, corresponds to the personal network $\mathcal{PN}$. We note that in the second layer (GNets), peers are gathered in clusters according to their similarity, while in the lower (random) layer, there is a random graph of peers. In our analysis we will be focusing mainly on the interaction of two peers at a time.

1.2 Privacy

1.2.1 Adversary model

In this study, we focus only on a computationally-bounded passive adversary [5], which is assumed to have the ability to access to the memory of some peers he has corrupted, but does not actively cheat during the protocol. Note that although we assume the channels between peers are private, the adversary has access (only) to the channels connected to the peers it has corrupted. This model is also called the "semi-honest" model because participants are assumed

to follow the protocol (thus being honest) but record all the information they have seen during its execution (thus being curious). Other adversary models exist, like the active adversary (where some participants can be malicious), the adaptive adversary and the covert adversary. However, we will not consider these models for our work, and will focus only on the passive adversary model which is the first step before providing privacy in stronger models.

1.2.2 Privacy, anonymity, and unlinkability

Anonymity. Anonymity that can be provided is one of three types [9] : anonymity of the recipient, anonymity of the sender, and unlinkability of sender and receiver. Note that the latter (which is called *relationship anonymity*) is weaker than the first two [10].

Relationship anonymity can be achieved via a MIX-net [11], assuming that an adversary does not have full control on all the MIXes. In this scenario, because even if an adversary wiretapped the similarity value, he cannot use this information directly unless he knows the sender and the receiver. However he may learn for a certain receiver its similarity distribution with respect to the network. This is not an issue because an adversary who is not controlling one of the peers cannot eavesdrop because the channel between two peers is a private channel (implemented via cryptographic means).

The problem is when one of the peers *is* the adversary. We want to avoid the possibility for a peer (controlled by the adversary) to query the same peer again and again by impersonating other identities. That can be achieved by preventing a peer from being able to identify the other peer over a second channel, which means that the other peer should be anonymous. By symmetry, both of them should be anonymous to each others, and therefore we need a "bidirectionally anonymous channel".

The bidirectionally anonymous channel The definition of *unlinkability* in [10] states that two items are unlinkable if the probability that they are related stays the same before (a priori knowledge) and after a execution of the system (a posteriori knowledge). In our definition, the two items that we require to be unlinkable are two individuals over two different channels. An "execution of the system", before and after which we measure the a priori and a posteriori knowledge, refers to the establishment of the channel, but does not take into account what is exchanged over that channel after it is established. We assume that attacker might be one of the correspondents (from the "bidirectionality" requirement) and it follows that it always have access to the channel contents (even if it is private), therefore if a peer sends his identity over the channel, this would break the unlinkability but is unavoidable.

Definition 1 (Bidirectionally anonymous channel)**.** *Let there be three peers A, B, B'. Before establishing communication channels to B and B', A has a priori knowledge that peers B and B' might be the same peer with probability p. The channels are called* bidirectionally anonymous *if and only if (for all A, B, B') after A establishes those channels and before any information is exchanged, the a posteriori knowledge of A that B and B' might be the same peer is still exactly p, and that the a posteriori knowledge B and B' (even if they are*

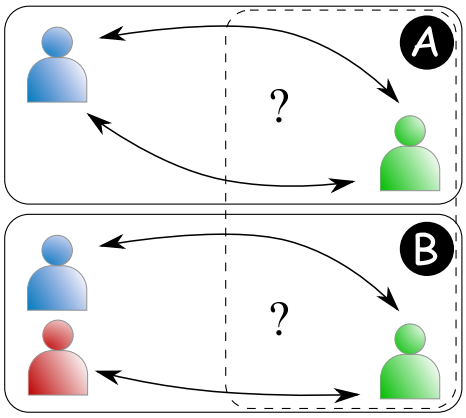


Figure 2: Bidirectionally anonymous channel: The green peer cannot distinguish situation A from situation B. The same also hold for the blue peer.

the same peer) that the two channels are to the same peer A is exactly the same as their a priori knowledge.

As in Figure 2, the green peer is unable to distinguish case A from case B. The same holds for all peers (including the blue peer).

Gossip on behalf One way to implement such a channel is gossip-on-behalf like Gossple [1]. In *gossip-on-behalf* (Figure 3), the peer A starts by choosing at random another peer P in his neighborhood. It then generates a pair of public key/secret key for this session and asks P to select one of his acquaintances (that we call B) as the second peer that will be involved in the similarity computation. The peer P does not disclose the identity of B to A and vice-versa, therefore acting as an anonymizer. Afterwards, P transmits the public key of A to B which uses it to encrypt the communication exchanged with A. Finally, B either also generates a pair public key/secret key for this session or a secret to be used as the key of the symmetric cryptosystem (such as AES) and transmit it to A encrypted with his public key via P as a relay. The peers A and B have now a private bidirectionally anonymous channel between themselves, given the P is honest and does not collude with either one. The communication between A and B goes through P but as it is encrypted, this forbids P to learn any information exchanged during A and B interactions. More complex techniques based on a chain of proxies or MIX-nets are possible [12, 11, 13, 14].

1.2.3 Privacy in secure multiparty computation

Consider a group of peers, each having their private input, who participated in computing the value of a predefined function of their inputs. They all get the output of the function at the end. Each peer knows its input and the output of the function and a log of all the messages he has seen (been a sender of a receiver of) during the execution of the protocol. An adversary has access to the information available for a subset of peers he has corrupted.

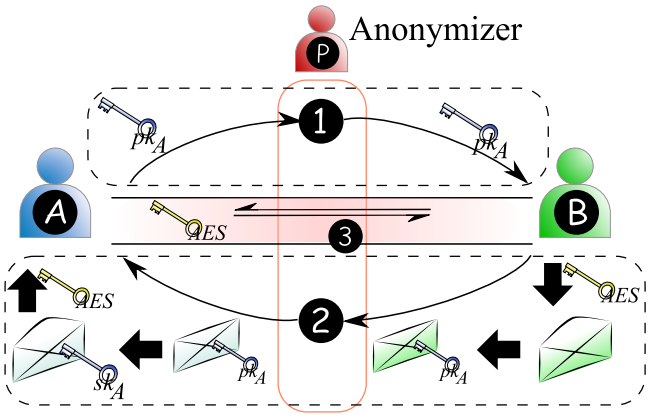


Figure 3: Gossip-on-behalf: Peer P makes sure peer A and B does not know each other identities. Step 1 and 2 is for securing the channel via exchanging an AES symmetric encryption key. Step 3, a bidirectionally anonymous secure channel is established via peer P who cannot read the encrypted contents of the channel.

A protocol is declared private in this model if the adversary cannot deduce any information about the *private inputs* of the honest peers (not controlled by him) other than what can be deduced from the function output and the inputs accessible to him. In particular, the output itself might carry some private information, but secure multiparty computation only secures the process of computing the function and does not take into account what the output itself might reveal.

For example, the result of a similarity computation between two peers might give some information to the peer whether a particular item was contained in the other peer's profile. The most obvious case is where one peer has a set composed of only one item, yet, this is not considered a privacy leak in the framework of secure multiparty computation.

1.2.4 Privacy in the setting of differential privacy

Differential privacy was essentially designed in the context of statistical database privacy protection. Its goal is to allow releasing statistics of a database while preserving privacy of individual rows. In this context, preserving the privacy of an individual row is equivalent to hiding the fact whether it was in the database or not from an attacker who has access to the output of the function and arbitrary background knowledge (including possibly knowledge about every other row in the database). In our case, the database is two input profiles, the statistic is the similarity value, and the single row is an item in the profile.

1.3 Secure two-party and multi-party computation

The definition of privacy in the model of secure multi-party computation is:

Definition 2 (Privacy with respect to a passive adversary [15]). *A multi-party protocol is said to be* private with respect to passive adversary *controlling a peer*

(or a collusion of peers), if this adversary cannot learn (except with negligible probability) more information from the execution of the protocol that he could from its own input (i.e. the inputs of the peers he controls) and the output of the protocol.

The privacy is usually modeled as a comparison to an ideal world, where a trusted third party is available to perform the computation. In that model all peers send their inputs to the trusted third party, which computes the function and then return the output to the peers. An adversary in that model gain information only from its input and the output it receives. Secure multiparty computation is about producing a secure distributed protocol, that the peers can execute it without relying on a third party, and without enabling anyone to learn more than what he would have learned in the ideal model.

General constructions exist [16, 17, 18]. The first and most general cryptographic one, due to Yao, works by constructing an encrypted bit-wise circuit where each binary gate has four possible encrypted outputs and the gate's inputs are the keys to decrypt the correct output [16]. There are other constructions which requires less communication cost, typically divided into two broad categories: *homomorphic encryption* and *secret-sharing* schemes [19].

1.3.1 Homomorphic encryption

Definition 3 (Additive and affine homomorphic encryption)**.** *Consider a public-key (asymmetric) cryptosystem where (1)* $\mathsf{Enc}_{pk}(a)$ *denotes the encryption of the message* a *under the public key* pk *and (2)* $\mathsf{Dec}_{sk}(a) = a$ *is the decryption of this message with the secret key* sk[3]*. This cryptosystem is said to* additively *homomorphic if there is an efficient operation* $\oplus$ *on two encrypted messages such that* $\mathsf{Dec}(\mathsf{Enc}(a) \oplus \mathsf{Enc}(b)) = a + b$*. Moreover, such an encryption scheme is called* affine *if there is also an efficient scalaring operation* $\odot$ *taking as input a cipher-text and a plain-text and such that* $\mathsf{Dec}(\mathsf{Enc}(c) \odot a) = c \times a$*.*

Paillier's cryptosystem [20] is an instance of a homomorphic encryption scheme that is both additive and affine. Moreover, Paillier's cryptosystem is also *semantically secure* (cf. Definition 4) which means that a computationally-bounded adversary cannot learn any information about a plain-text m given its encryption $\mathsf{Enc}(m)$ and the public key pk. In this paper, we will also use a *threshold version* of the Paillier's cryptosystem [21] (cf. Definition 5).

Definition 4 (Semantic security [15])**.** *An encryption scheme is said semantically secure if a computationally-bounded adversary cannot derive non-trivial information about the plain text encrypted from the cipher text and the public key only. For instance, a computationally-bounded adversary who is given two different cipher texts cannot even decide with non-negligible probability if the two cipher texts correspond to the encryption of the same plain text.*

Definition 5 (Threshold cryptosystem)**.** *A* (t, n) threshold cryptosystem *is a public-cryptosystem where at least* $t > 1$ *peers out of* n *need to actively cooperate in order to decrypt an encrypted message. In particular, no collusion of even* $t - 1$ *peers can decrypt a cipher text. However, any peer may encrypt a value*

[3] In order to simplify the notion, we will drop the indices and write $\mathsf{Enc}(a)$ instead of $\mathsf{Enc}_{pk}(a)$ and $\mathsf{Dec}(a)$ instead of $\mathsf{Dec}_{sk}(a)$ for the rest of the report.

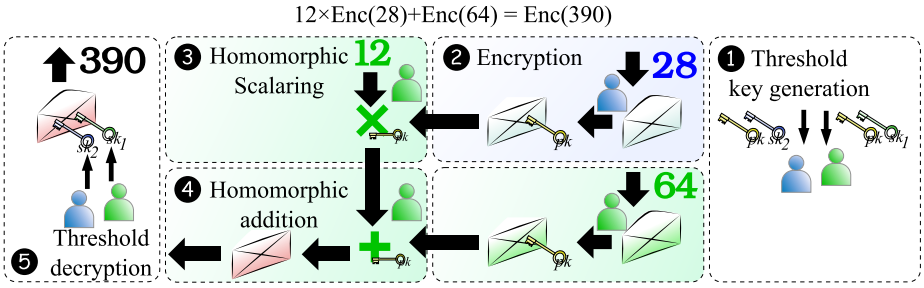


Figure 4: Threshold homomorphic cryptosystem: Both peers generate the keys (step 1), encrypt their numbers (step 2), and perform calculations on the encrypted values (step 3 and 4). To decrypt in step 5 they need both their secret keys.

on its own using the public-key pk. When the threshold cryptosystem is set up, each peer i gets his own secret key sk_i *(for* $1 \leq i \leq n$*).*

In Figure 4, you can see an example of a threshold homomorphic cryptosystem where peers perform computations on the cipher-text. The To decrypt the result both peers must cooperate using their secret keys.

1.3.2 Secret sharing

Secret-sharing [19] is a technique that provides *perfect secrecy* (i.e. provides security even against unbounded adversaries. cf. Definition 6). Hence also allows for much less communication cost than other cryptographic techniques. Perfect secrecy provides information-theoretic security, which means that is when the security of the system holds even if the attacker possesses unbounded computational power. More specifically, when the system does not depend on the assumptions of computational hardness of certain mathematical problems, like factoring or discrete logarithm. In other terms, witnessing the cipher-text and the public key does not add any information about the encrypted message to the adversary, regardless of its computational power.

Definition 6 (Perfect secrecy)**.** *A cipher-text m is said to be* perfectly secure *if any information about m that is learned can also be learned without witnessing the encrypted message or the key used to encrypt it, and that is given without any assumptions on the distribution of the potential messages [22, Definition 3].*

However, a main problem with secret-sharing technique is that it is not able to provide multiplication of secrets when there is only two parties. Since our computation focus on two-party similarity metric, we use homomorphic encryption instead which can provide multiplication of two cipher-texts in secure two-party computation (using the multiplication gate [23]).

1.4 Differential privacy

Originally, differential privacy [6] was developed in the context of private data analysis and its main guarantee is that if a differentially private mechanism is

applied on a dataset composed of the personal data of individuals, no outputs would become significantly more (or less) probable whether or not a participant removes his data from the dataset. This means that for an adversary which observes the output of the mechanism, he will only gain a negligible advantage from the presence (or absence) of a particular individual in the database. This statement is a statistical property about the behavior of the mechanism (function) and holds independently of the auxiliary knowledge that the adversary might have. In particular, it means that even if the adversary knows all the database except one individual row, a mechanism satisfying differential privacy will still protect the privacy of this individual up to a privacy parameter ϵ. The parameter ϵ is public and it may takes different value depending of the application (for instance it could be 0.01, 0.1 or even 0.25).

In alternative terms, differential privacy aims at protecting the privacy of individual rows of a database while allowing capturing global properties of the entire dataset. It has been shown that the process of the computing global properties must depend on interactive queries [24], and there is no "sanitization" techniques (such as k-anonymity, l-diversity and t-closeness [25, 26, 27]) releasing a sanitized version of the database for offline querying, that can provide absolute privacy and good accuracy for a large number of queries. In the an interactive model, a trusted *third party* (curator) interactively reply to the queries. However, in our context, to preserve the fully decentralized nature of our systems, no trusted third party should be required. The trusted third party can be replace by a distributed protocol by using secure multi-party computation techniques (Section 1.3).

Two inputs S_1 and S_2 are said to be *neighbors* if they are both equal except for at most one entry of the inputs. For instance, if S_1 and S_2 are profiles, it would mean that they are identical except for one item.

Definition 7 (Differential privacy [6])**.** *A randomized mechanism $\mathcal{A}$ satisfies* $\epsilon-$differential privacy *if for all possible neighboring inputs S_1 and S_2 , and all $S \subseteq Range(\mathcal{A})$,*

$$\frac{\mathsf{Pr}[\mathcal{A}(S_1) \in S]}{\mathsf{Pr}[\mathcal{A}(S_2) \in S]} \leq \mathsf{exp}(\epsilon) \ , \tag{1}$$

where the probability is taken over all the coin tosses of $\mathcal{A}$.

As you can see in Figure 5, the adversary is not able to determine (up to a privacy parameter ϵ) whether the blue item is in the set or not. That holds even if the adversary knows every other item in the set and knows f and has arbitrary background about the blue item.

Dwork, McSherry, Nissim and Smith have designed a general technique, called *Laplacian mechanism* [28], that allows to achieve ϵ-differential privacy for a function f by adding random noise to the true answer before releasing it. The amount of noise that has to be added is directly proportional to the *sensitivity* of the function which measures how much the output of a function can change with respect to a small change in the input [28].

Definition 8 (Global sensitivity [28])**.** *For $f : D^n \to R$, the* sensitivity *of f is*

$$\mathsf{GS}\,(f) = \max_{S_1,S_2} \|f(S_1) - f(S_2)\|_1 \ , \tag{2}$$

for all neighboring S_1 and S_2.

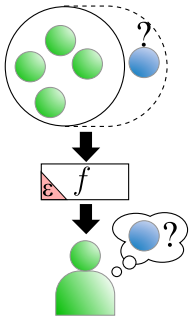


Figure 5: Differential privacy: The green peer cannot decide whether or not the blue item is in the set given the output of the ϵ-differentially private mechanism f and having an arbitrary background knowledge.

Parametrized sensitivity, that we refer thereafter simply as sensitivity, requires to fix first the size of the input to the function before computing the maximum difference to the output over all neighboring inputs of the corresponding size. In Lemma 1, we show that parametrized sensitivity insures the differential privacy condition.

Lemma 1. *Parametrized sensitivity insures the differential privacy condition as does the global sensitivity.*

Proof. The differential privacy condition [6, Page 4] is that as long as $\|f(S_1) - f(S_2)\|_1 < \mathsf{GS}(f)$, the ratio $\mathsf{exp}(-\epsilon)$ which is the requirement of the differential privacy is the upper bound in Equation (1). The parametrized sensitivity $\mathsf{PS}_{parameters}(f)$ is a tighter bound such that $\|f(S_1) - f(S_2)\|_1 \leq \mathsf{PS}_{|S_1|,|S_2|}(f) \leq \mathsf{GS}(f)$. Thus the differential privacy condition holds. □

The Laplace distribution, called $\mathsf{Lap}(0, \lambda)$ with mean 0 and scale λ is defined by the following probability density function:

$$f(y) = \frac{\mathsf{exp}(-|x|/\lambda)}{2\lambda} \quad . \tag{3}$$

The Laplacian mechanism achieves ϵ-differential privacy by adding noise directly proportional to $\mathsf{GS}(f)$.

Theorem 1 (Laplacian mechanism [28])**.** *For a function $f : D^n \to R$, a randomized mechanism $\mathcal{A}$ achieves ϵ-differential privacy if it releases on input x*

$$\mathcal{A}(x) = f(x) + \mathsf{Lap}\left(0, \frac{\mathsf{GS}(f)}{\epsilon}\right) \quad , \tag{4}$$

for $\mathsf{GS}(f)$ the sensitivity of the function f and $\mathsf{Lap}(0, \cdot)$ a randomly generated noise according to the Laplacian distribution with mean zero and scale $\frac{\mathsf{GS}(f)}{\epsilon}$.

The following lemma also shows that differential privacy is a "natural" notion that composes well.

Lemma 2 (Composition and post-processing [29]). *If a randomized mechanism $\mathcal{A}$ run k algorithms $\mathcal{A}_1,\ldots,\mathcal{A}_k$ where each $\mathcal{A}_i$ is ϵ-differentially private, and outputs a function of the results (i.e $\mathcal{A}(z) = g(\mathcal{A}_1(z),\ldots,\mathcal{A}_k(z))$ for some probabilistic algorithm g) then $\mathcal{A}$ is $\sum_{i=1}^{k}\epsilon_i$-differentially private.*

The lemma also implies that *each* execution of an ϵ-differentially private protocol releases at most an amount of privacy proportional to ϵ [28]. Hence, we want such a protocol to be run no more than once for each two peers. Unfortunately, this is impossible to prevent since an adversary having access to more than one peer can gain advantage by executing the protocol through these peers several times with the same peer. Instead, we use a bidirectionally anonymous channel (cf. Section 1.2.2) which guarantees that an adversary will not be able to link two different queries over two different channels to the same peer. The only way then to link two executions of the same protocol to the same peer is through executing both through the same anonymous channel, and since an honest peer will not agree to engage in the computation two times on the same anonymous channel, this effectively means that no peer will be able to get more than the allowed amount of information about an honest peer through an ϵ-differentially private protocol.

2 Similarity metrics and secure implementation

In this section we formally define what is similarity measures, and give two examples of them, namely cosine similarity and Jaccard index. Then we introduce the concept of *threshold similarity*. Afterwards, we discuss how to provide a secure implementation for threshold similarity computation that provides privacy in the secure multi-party computation model.

2.1 Similarity measures

In our context, the main goal of a peer is to detect similar peers (i.e. those with whom he shares similar interests). Thus, we assume the existence of a similarity measure that can be used by the two peers to quantify how similar they are. Similarity between peers depends mainly on their profile of interests, which is modeled as a set of items that peers has shown interest in (by tagging, liking, bookmarking, etc.). If the item domain is denoted as $\mathbb{I}$, then a profile $p \subseteq \mathbb{I}$. In the following definition, we denote the same concept as the power set form $p \in 2^{\mathbb{I}}$.

Definition 9 (Similarity measure). *A* similarity measure sim *is a function taking as input two sets $S_1, S_2 \in 2^{\mathbb{I}}$ representing the profiles of two peers and outputs a value in the range between 0 and 1, where* 0 *indicates that the sets are entirely different (the profiles have no items in common) whereas* 1 *means that the sets are identical (and therefore the peers can be considered as sharing exactly the same interests). Therefore, $\mathsf{sim}(S_1, S_2) \in [0, 1]$ or more formally:*

$$\mathsf{sim} : 2^{\mathbb{I}} \times 2^{\mathbb{I}} \to [0, 1] \ .$$

Cosine similarity [7] is commonly used to assess the similarity between two sets (profiles) and can be seen as a normalized overlap between the sets. It is

defined as

$$\frac{|S_1 \cap S_2|}{\sqrt{|S_1| \times |S_2|}} \ , \tag{5}$$

where $|S_1|, |S_2|$ are the sizes of the private sets S_1, S_2 of the first and second peer, respectively, and $|S_1 \cap S_2|$ is the *size of the set intersection* of S_1 and S_2.

Jaccard index [8] is another common similarity measure which is defines as

$$\frac{|S_1 \cap S_2|}{|S_1 \cup S_2|} \ , \tag{6}$$

where $|S_1 \cup S_2|$ is the *size of the set union* between S_1 and S_2. Other similarity metrics could also be used, however for the sake of clarity, we focus on the cosine similarity metric in the rest of this report.

To implement the corresponding similarity measure in our protocol we can use set intersection protocols or scalar product[4] protocols interchangeably; since the size of the set intersection of two sets is equivalent to the scalar product of the corresponding indicator functions.

In the rest of this section we discuss the secure implementation of a distributed protocol to compute the cosine similarity of two peers. In the next section we explain the first step of that protocol, which is how to securely compute the cardinality intersection of two sets (the numerator of the cosine similarity).

2.2 The intersection step

During the first step of the protocol, the two peers compute the size of the set intersection of their two profiles S_1 and S_2 by using a protocol for scalar product or a protocol for the cardinality set intersection. We assume that the profiles S_1 and S_2 can be represented as binary vectors of size l[5]: $\mathbf{1}_{S_1} = \{a_1, \ldots, a_l\}$ and $\mathbf{1}_{S_2} = \{b_1, \ldots, b_l\}$ such that $a_i = 1$ if the first peer has item i in his profile and 0 otherwise (the same goes for b_i). For illustration purpose, we call the first peer Alice and the second peer Bob.

There are advantage and disadvantages for both set-intersection based and scalar-product based approaches. To our knowledge, all existing secure set intersection protocols [30, 31, 32, 33] include steps proportional to the input set sizes, hence revealing the set sizes of the peers to each other. On the other hand, scalar product protocols [34, 35, 36, 37] inherently require communication complexity proportional to the size of the item domain, which is independent of the set sizes and thus hiding it, although at the cost of an increased communication. On the computational size, the scalar product approach has a linear computation complexity while the set intersection approach has quadratic complexity. Since our primary interest is privacy, we selected the scalar product approach since it hides the set sizes. First, we will give an overview of different scalar product protocols in the literature in the following section as well as set intersection protocols.

[4]The scalar product of two bit vectors a, b of size l is defined as $\sum_{i=1}^{l} a_i b_i$.

[5]Notice that l is the cardinality of the item domain $\mathbb{I}$. I.e $l = |\mathbb{I}|$.

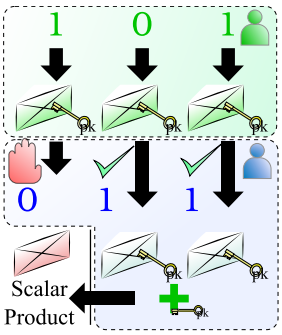


Figure 6: Scalar product protocol: The blue peer adds the encrypted bits of the green peer which correspond to his ones to get the encrypted scalar product.

2.2.1 Scalar product approach

The protocols for scalar product can be divided into two main branches, those based on secret-sharing [38, Section 4.3], and those based on homomorphic encryption [34, 35, 36]. The secret-sharing ones, although providing no inherent advantage on the communication complexity, provides an advantage in the complexity's hidden constant because since secret-sharing provides perfect secrecy and thus does not required big cipher-texts.

Unfortunately, secure multiplication in secret-sharing needs at least 3 peers, thus requiring a (semi-trusted[6]) third party [38]. It would be possible, if we are using gossip-on-behalf for anonymization (cf. Section 1.2.2), to use the anonymizer as this semi-trusted third party.

The protocol in [36] reveals a permuted sum of the two vectors. This protocol is only secure if the other vector is uniformally distributed over the integers, which is definitely not secure for bit vectors[7] (as in our case). So, by excluding this protocol we are left with [34] and [35]. According to [34], [35] is the exact same protocol that they weren't aware of at the time of submission. We use that protocol to implement our scalar-product intersection step in Algorithm 1. Further in the main protocol, we will just refer to this step as the "ScalarProduct".

Secure scalar product protocol. In a preprocessing step to this protocol, the two peers engage in the setup phase of a key generation protocol for an threshold affine homomorphic cryptosystem [21]. At the end of this key generation phase, both peers have received the same public key pk that can be used to homomorphically encrypt a value and each one of them has as private input a secret key, respectively sk_a for the first peer and sk_b for the second peer. The threshold cryptosystem[8] is such that any peer can encrypt a value using the public key pk but that the decryption of a homomorphically encrypted value require the active cooperation of the two peers. There is an illustration in Figure 6.

[6]Semi-trusted here means being a passive adversary.
[7]A peer could easily learn the number of zeros and number of ones in some situations.
[8]The threshold cryptosystem should not be confused with the threshold similarity.

Algorithm 1 ScalarProduct($\mathbf{1}_{S_1} = \{a_1, \ldots, a_l\}$,$\mathbf{1}_{S_2} = \{b_1, \ldots, b_l\}$)

for $i = 1$ to l **do**
  Alice computes $\mathsf{Enc}(a_i)$
end for
Alice sends $\mathsf{Enc}(a_1), \ldots, \mathsf{Enc}(a_l)$ to Bob
Bob sets $s = \mathsf{Enc}(a_1)$ if b_1 equals 1, and sets $s = \mathsf{Enc}(0)$ otherwise.
for $i = 2$ to l **do**
  if $b_i = 1$ **then**
    Bob sets $s = s \oplus \mathsf{Enc}(a_i)$
  end if
end for
Bob sends s to Alice

2.2.2 Set intersection approach

Like the scalar-product protocols, the protocols for computing the size of the set intersection are also divided into those who rely on secret-sharing schemes [30] and those who use homomorphic encryption [32, 33, 39]. The protocol in [30], as inherent in secret-sharing, do require the help of a third party in the two-party case, therefore we focus instead on [32, 33, 39].

The algorithms described in [33, Section 4.4] and [32, Section 5.2] rely on the same idea. More precisely, they both represent sets as polynomials, where the roots of the polynomial are the set items (represented as integer). The former is a two-party version and the latter is a multi-party one. In both however, all items must be compared to each others, which leads to quadratic complexity. Inan, Kantarcioglu, Ghinita, and Bertino [39] relaxes this requirement by providing a differentially private protocol to *block* some unneeded comparisons. In the example they presented, they could save up to 17% of the comparisons needed.

2.3 Threshold cosine similarity

We are interested in a form of similarity which outputs only one bit of information stating whether (or not) the similarity between two profiles is above some well-chosen threshold τ.

Definition 10 (Threshold similarity)**.** *Two peers are said to be τ-similar if the output of applying some similarity measure on their respective sets is above a chosen threshold $0 \leq \tau \leq 1$ (i.e. $\mathsf{sim}(S_1, S_2) > \tau$).*

In practice, the value of the threshold τ will be application-dependent and will be set empirically so as to be significantly above the average similarity in the population. Nonetheless, in Section 3.4 we provide heuristic for selecting the threshold as a function of the desired acceptance ratio.

The threshold similarity protocol takes as input two profiles S_1 and S_2 (one profile per peer) represented as binary vectors and output one bit of information which is 1 if S_1 and S_2 are τ-similar (i.e. $\mathsf{sim}(S_1, S_2) > \tau$ for sim a predefined similarity measure and τ the value of the threshold), and 0 otherwise. The threshold similarity is very appealing with respect to privacy as it guarantees that the output of the similarity computation will only reveal one bit of information, which is potentially much less than disclosing the exact value of the

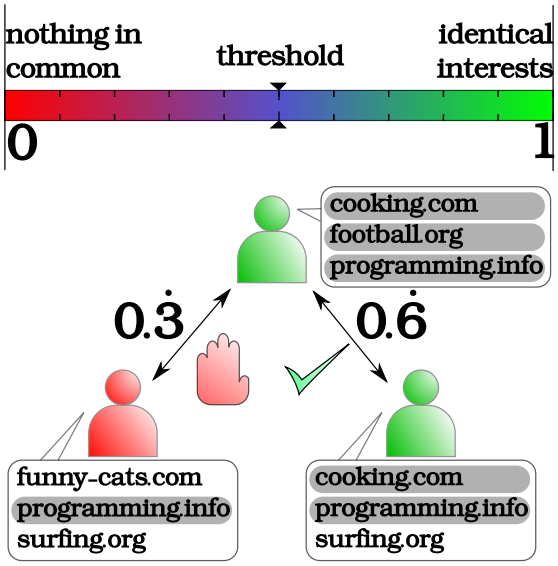


Figure 7: Threshold similarity: The peer accepts only the peers having similarity with him above a certain threshold. The similarity is based on the common items in their profiles.

similarity measure. In order to do the threshold step, we employ a secure integer comparison protocol. We revise the relevant secure integer comparison literature in the next section.

2.4 Integer comparison step

The integer comparison problem, known as the socialist millionaires' problem, where two parties have their own private inputs and want to compare them without revealing anything about their inputs. A variety of solutions to this problem have been developed since Yao [40] proposed it.

Nonetheless, all these protocols lay under two broad branches, both of which depend on bit-wise operations. One of the two branches is based on encrypted truth tables, while the other exploits homomorphic operations on encrypted bits.

The encrypted truth table technique requires the input to be known (i.e. in plain-text) to its owner as opposed to being passed as it is without requiring decryption after being evaluated from a previous circuit. The homomorphic operations on encrypted bits takes advantage of the homomorphic properties of a cryptosystem to achieve the same with less communication overhead than encrypted truth tables but at the expense of costly asymmetric encryption.

Some methods require both parties to know the value of their inputs to be able to correctly setup the protocol [41], while others can work directly on the bit-wise encryption of these inputs [42, 43]. In case where the value as a whole is encrypted but the bit-wise encryption is not available, bit-decomposition protocols are used.

Bit-decomposition. Since all these methods depend on bit-wise operations, an important ingredient of these protocols is a bit-decomposition scheme. A bit-decomposition scheme takes a homomorphically-encrypted integer and outputs the homomorphically-encrypted values of each bit of this integer [44, 45].

Least significant bit. Nishide and Ohta [43] provides better solution for integer-comparison protocol by requiring only the least significant bit to be computed instead of the entire bit-decomposition. This method worked originally for secret-sharing schemes but can be adapted to homomorphic encryption as well.

Lin's method [41] requires at least one peer to know his number in plain-text while Garay's method [42] is less efficient than Lin's as it requires both homomorphic addition and multiplication to operate on the encrypted bits. We reject the former as both peers will not have their input in plain-text, and we reject the latter because Nishide's protocol (presented in the following paragraph) is more efficient than both Lin's protocol and Garay's protocol it as it uses only the least significant bit instead of all the bits.

Nishide's protocol. To the best of our knowledge, Nishide's protocol [43] is potentially the most efficient protocol in terms of communication and communication. It operates only on the least significant bit as detailed. This method was originally presented in the context of secret-sharing, but we translate it to the setting of homomorphic encryption.

Algorithm 2 $\mathsf{CompareIntegers}(\mathsf{Enc}(a), \mathsf{Enc}(b))$ [43]

1: $w \leftarrow \mathsf{LeastSignificantBit}(2\mathsf{Enc}(a) \mod n)$
2: $x \leftarrow \mathsf{LeastSignificantBit}(2\mathsf{Enc}(b) \mod n)$
3: $y \leftarrow \mathsf{LeastSignificantBit}(2(\mathsf{Enc}(a) - \mathsf{Enc}(b)) \mod n)$
4: **if** $\mathsf{Dec}(w\bar{x} \vee \bar{w}\bar{x}\bar{y} \vee wx\bar{y}) = 1$ **then**
5: Output "$a < b$"
6: **else**
7: Output "$a \geq b$"
8: **end if**

Checking the least significant bit of $2a \bmod n$, is equivalent to checking whether $a < n/2$ or not, where n is the RSA modulus of the homomorphic encryption (or the secret-sharing scheme field cardinality.)

Nishide and Ohta have showed using a truth-table that with w, x, y you can uniquely determine each corresponding output for $a < b$. This method is efficient because it only uses affine multiplication, and a very shallow binary circuit on w, x, y, with only 3 least significant bit protocol invocations (which is the just first round of any bit-decomposition protocol). Unfortunately, because affine homomorphic cryptosystems cannot provide multiplication of encrypted values, it cannot be used to compute the last binary circuit (in the if statement). A method like the conditional gate from [46] has to be used.

The conditional gate is an interactive protocol to multiply two bits x, y, which involves each peer in turn multiplying the same random number $s_i \in \{-1, 1\}$ of his choice to both bits, ending up with $\mathsf{Enc}(s_1 s_2 x)$ and $\mathsf{Enc}(s_1 s_2 y)$. Finally they perform a threshold decryption of the the former, and scalarize the

other by that one, namely $s_1 s_2 x \odot \mathsf{Enc}(s_1 s_2 y)$. That yields $\mathsf{Enc}\left(xy\,\frac{(s_1 s_2)^2}{(s_1 s_2)^1}\right) = \mathsf{Enc}(xy)$, as desired. The conditional gate along with the negation gate[9] compose a NAND gate, which is universal for binary circuits, and is used to evaluate the Boolean logic expression in the protocol.

2.5 Private similarity computation

Instead of computing directly the cosine similarity as denoted in Equation (5), we avoid the need for performing a square root on encrypted values (an operation which is non-trivial and often costly) by squaring the whole equation. The squaring operation renders the next cryptographic operations easier while preserving as the same time the order relation. Formally, the similarity metric effectively used in ThresholdCosine is

$$\left(\frac{|S_1 \cap S_2|}{\sqrt{|S_1| \times |S_2|}}\right)^2 = \frac{|S_1 \cap S_2|^2}{|S_1| \times |S_2|} \quad . \tag{7}$$

On one hand for obtaining the numerator, we square the output of the intersection step by applying the multiplication gate from [23] to multiply it by itself. On the other hand, the denominator can be computed by the first peer sending his homomorphically-encrypted set cardinality to the second peer (i.e. $\mathsf{Enc}(|S_1|)$), who will scalarize it by his own set cardinality by doing $|S_2| \odot \mathsf{Enc}(|S_1|)$ to obtain $\mathsf{Enc}(|S_1| \times |S_2|)$.

Recall, that the objective of the ThresholdCosine protocol is only to learn if the similarity between S_1 and S_2 is above a certain (publicly known) threshold τ. We assume that the threshold can be represented as a fraction $\tau = \tau_1/\tau_2$ (for τ_1 and τ_2 some positive integers) and therefore our goal is to verify whether or not the following condition holds

$$\frac{|S_1 \cap S_2|^2}{|S_1| \times |S_2|} > \frac{\tau_1}{\tau_2} \tag{8}$$

$$\tau_2 |S_1 \cap S_2|^2 > \tau_1 |S_1| \times |S_2| \tag{9}$$

The left side and right side of the inequality can be compared by secure protocols for integer comparison (cf. Section 2.4). We choose to apply specifically the comparison technique from [43] as it neither require knowledge of the input[10] nor a full bit decomposition of the input as other protocols. Although this protocol was developed initially for secret-sharing, it can be implemented with homomorphic encryption as well. The output of this comparison step is one bit stating whether or not the (squared) cosine similarity is above the threshold τ.

Theorem 2 (Protocol for threshold cosine similarity)**.** *The protocol ThresholdCosine is private with respect to a passive adversary (cf. Section 1.2.3) and returns* 1 *if two peers are* τ*-similar and* 0 *otherwise. The protocol has a communication complexity of* $O(l)$ *bits and a computational cost of* $O(l)$*.*

Proof. All the communication exchanged between Alice and Bob is done using a homomorphic encryption scheme with semantic security (cf. Definition 4),

[9] The negation gate of an encrypted bit $\mathsf{Enc}(x)$ is simply $1 \oplus (-1 \odot \mathsf{Enc}(x)) = \mathsf{Enc}(1 - x)$.
[10] Therefore, the input can be the encrypted output of a preliminary cryptographic protocol.

Algorithm 3 $\mathsf{ThresholdCosine}(S_1, S_2, \tau = \frac{\tau_1}{\tau_2})$

1: Alice and Bob generates the keys of the threshold homomorphic encryption
2: Alice receives sk_a, Bob receives sk_b and they both get the public key pk
3: Alice and Bob compute $\mathsf{Enc}(|S_1 \cap S_2|) = \mathsf{ScalarProduct}(\mathbf{1}_{S_1}\mathbf{1}_{S_2})$
4: Alice applies the multiplication gate from [23] to obtain $\mathsf{Enc}\left(|S_1 \cap S_2|^2\right)$
5: Alice computes $\mathsf{Enc}\left(|S_1 \cap S_2|^2\right) \odot \tau_2 = \mathsf{Enc}\left(\tau_2 |S_1 \cap S_2|^2\right)$
6: Alice computes $\mathsf{Enc}(|S_1|)$ and sends it to Bob
7: Bob computes $\mathsf{Enc}(|S_1|) \odot (\tau_1 |S_2|) = \mathsf{Enc}(\tau_1 |S_1| \times |S_2|)$
8: Alice and Bob use the integer comparison protocol of [43] on $\mathsf{Enc}\left(\tau_2 |S_1 \cap S_2|^2\right)$ and $\mathsf{Enc}(\tau_1 |S_1| \times |S_2|)$
9: **if** $\mathsf{Enc}\left(\tau_2 |S_1 \cap S_2|^2\right) > \mathsf{Enc}(\tau_1 |S_1| \times |S_2|)$ **then**
10: output 1 to state that Alice and Bob are τ-similar
11: **else**
12: output 0
13: **end if**

therefore the encrypted messages exchanged do not leak any information about their content. Moreover as the encryption scheme is a threshold version, it means that neither Alice nor Bob alone can decrypt the messages and learn their content. The multiplication gate [23] as well as the integer comparison protocol [43] are also semantically secure, which therefore guarantees that the protocol is secure against a passive adversary.

Regarding the correctness, it can be seen from the execution of the protocol that if Alice and Bob are τ-similar then this will result in $\tau_2 |S_1 \cap S_2|^2 > \tau_1 |S_1| \times |S_2|$ when the integer comparison protocol is executed (and therefore an output of 1) and in 0 otherwise. The multiplication gate and the integer comparison protocols are independent of l and can be considered as having constant complexity (both in terms of communication and computation) for the purpose of analysis. On the other hand, the protocol $\mathsf{ScalarProduct}$ requires the exchange of $O(l)$ bits between Alice and Bob as well as $O(l)$ computations which result in a similar complexity for the global protocol $\mathsf{ThresholdCosine}$. □

3 Differential privacy and utility analysis

Cryptography gives us the tools to compute a distributed function in such a way that the computations themselves reveal nothing that cannot be learned directly from the output of the function. This is a strong privacy guarantee but at the same time, this does not preclude the possibility that the output itself might leak information about the private data of individuals. In order to obtain have the best of both worlds (secure multi-party computation and differential privacy), the main idea is to use cryptographic techniques to securely compute a differentially private algorithm. In this section, we give efficient and secure algorithms for computing a differentially private version of the exact similarity and the threshold similarity.

We define two profiles S_1 and S_2 as *being neighbors* if they are the same up to a particular item. Note that for simplicity and without loss of generality, we consider only neighboring profiles that retain the same size. For instance, S_1 is a neighbor of S_2 (and vice-versa) if it is identical except for one item that may have been replaced to obtain the profile S_2. If the two profiles are represented as an indicator function (binary vectors of set-membership), they are neighbors if they are within Hamming[11] distance 2 (and they both have the same L1-norm [12]).

3.1 Differentially private threshold similarity

Regarding the threshold similarity, it is important to notice that it is meaningless to add some random noise to a binary value (the output of the threshold similarity), because it amounts to flipping the variable with some probability. In this case, the most direct way to achieve differential privacy is instead to add the noise before the application of the threshold function instead. The following theorem shows that this does not hurt the privacy guarantee obtained.

Theorem 3 (Impact of threshold on privacy)**.** *Applying the Laplacian mechanism before the application of the threshold function does not hurt the differential privacy.*

Proof. As Lemma 2 states, a mechanism $\mathcal{A}(z)$ is ϵ-differentially private if it outputs a probabilistic post-processing function (g) of an ϵ-differentially private mechanism $\mathcal{A}_1(z)$, as shown in Equation (10).

$$\underbrace{\overbrace{\mathcal{A}(z)}^{\epsilon-\text{privacy}}}_{\text{Threshold Cosine Similarity}} = \underbrace{g}_{\text{ThresholdFunction}} \left(\underbrace{\overbrace{\mathcal{A}_1(z)}^{\epsilon-\text{privacy}}}_{\text{Cosine Similarity}} \right) \tag{10}$$

□

3.2 Parametrized sensitivity

The following lemmas state the sensitivity of scalar product (equivalent to the sensitivity of cardinality set intersection) and the squared cosine similarity.

Lemma 3 (Sensitivity of the scalar product)**.** *The sensitivity of the function ScalarProduct is 1.*

Proof. Consider three different profiles $\mathbf{1}_{S_1}$, $\mathbf{1}_{S_2}$ and $\mathbf{1}_{S_3}$, represented as binary vectors of size l bits, such that $\mathbf{1}_{S_2}$ and $\mathbf{1}_{S_3}$ are neighbors. Replacing an object from S_2 by another object to obtain S_3 will only increase (or decrease) the value of the scalar product between $\langle \mathbf{1}_{S_1}, \mathbf{1}_{S_2} \rangle$ and $\langle \mathbf{1}_{S_1}, \mathbf{1}_{S_3} \rangle$ by 1 at most, therefore the sensitivity of the scalar product is 1. □

[11] Hamming distance of two binary vectors is the number of positions in which they differ.

[12] L1-norm of a vector V is $\|V\|_1 = \sum_i |V_i|$, which for binary vectors is equivalent to the number of ones of this vector.

It follows from the previous lemma that the sensitivity for the cosine similarity, assuming the denominator stays the same, is $= \frac{1}{|S_1| \times |S_2|}$. However, for the squared cosine similarity the situation is different.

Lemma 4 (Sensitivity of the squared scalar product)**.** *The sensitivity of the function* $|S_1 \cap S_2|^2 = \langle \mathbf{1}_{S_1}, \mathbf{1}_{S_2} \rangle^2$ *(where* $\langle \cdot, \cdot \rangle$ *is the scalar product) is at most* $2\mathsf{min}(|S_1|, |S_2|) - 1$.

Proof. Consider three different profiles S_1, S_2 and S_3 such that $\mathbf{1}_{S_2}$ and $\mathbf{1}_{S_3}$ are neighbors.

$$\mathsf{PS}_{|S_1|,|S_2|}\left(\langle \cdot, \cdot \rangle^2\right) = \max_{S_1,S_2,S_3} \left\| \langle \mathbf{1}_{S_1}, \mathbf{1}_{S_2} \rangle^2 - \langle \mathbf{1}_{S_1}, \mathbf{1}_{S_3} \rangle^2 \right\|_1$$

From Lemma 3

$$\begin{aligned}
\max_{S_1,S_2,S_3} \left\| \langle \mathbf{1}_{S_1}, \mathbf{1}_{S_2} \rangle^2 - \langle \mathbf{1}_{S_1}, \mathbf{1}_{S_3} \rangle^2 \right\|_1 &\leq \max_{S_1,S_2} \left\| \langle \mathbf{1}_{S_1}, \mathbf{1}_{S_2} \rangle^2 - (\langle \mathbf{1}_{S_1}, \mathbf{1}_{S_2} \rangle \pm 1)^2 \right\|_1 \\
&= \max_{S_1,S_2} \left\| \cancelto{0}{\langle \mathbf{1}_{S_1}, \mathbf{1}_{S_2} \rangle^2 - \langle \mathbf{1}_{S_1}, \mathbf{1}_{S_2} \rangle^2} \pm 2 \langle \mathbf{1}_{S_1}, \mathbf{1}_{S_2} \rangle - 1 \right\|_1 \\
&= \max_{S_1,S_2} \pm 2 \langle \mathbf{1}_{S_1}, \mathbf{1}_{S_2} \rangle - 1 \\
&= \max_{S_1,S_2} 2 |S_1 \cap S_2| - 1 \\
&\leq 2\mathsf{min}(|S_1|, |S_2|) - 1 \ .
\end{aligned}$$

□

And by the same denominator argument, the sensitivity of the squared cosine similarity is $\mathsf{PS}_{|S_1|,|S_2|}(\mathsf{ExactSquaredCosine}) = \frac{2\mathsf{min}(|S_1|,|S_2|)-1}{|S_1| \times |S_2|}$

3.3 Differentially private similarity computation

According to Theorem 1, to provide differential privacy it is sufficient to add Laplacian noise proportional to the sensitivity of the function to the true answer before releasing it. This can be done interactively in a centralized environment where a curator is holding the data and replying to queries. For the distributed setting, we discuss three possible alternatives in the following sections. The first one depends on the availability of semi-trusted third party, while the last two are fully distributed and do not require a third party.

We denote the noise that is to be added as a random variable $N = \mathsf{Lap}\left(0, \frac{\mathsf{PS}_{|S_1|,|S_2|}(\mathsf{ExactSquaredCosine})}{\epsilon}\right) = \mathsf{Lap}\left(0, \frac{2\mathsf{min}(|S_1|,|S_2|)-1}{\epsilon \times |S_1| \times |S_2|}\right)$. Specific instances used in the protocol description are called $n_1, n_2, \ldots$.

3.3.1 Using a semi-trusted third party

In the context of gossip-on-behalf (see Section 1.2.2), the peer that acts as an anonymizer in the bidirectionally anonymous channel could also generate the required random noise. Note that the peer is *semi*-trusted in the sense that he is trusted that he will not collude with any peer to break the anonymity,

but in the same time he is not trusted enough to reveal the private profiles for him in the clear. For instance, the anonymizer can adds the random noise using the homomorphic property of the cryptosystem (as he knows the public key) to the similarity value that has been computed. Afterwards, the two peers that have been involved in the similarity computation can recover the result using the threshold decryption, either after applying the threshold (for threshold similarity) or before it (for exact similarity). Algorithm 4 describes this procedure.

Algorithm 4 DifferentialSquaredCosine(S_1,S_2,ϵ)

1: Alice and Bob generates the keys of the threshold homomorphic encryption
2: Alice receives sk_a, Bob receives sk_b and they both get the public key pk
3: Alice and Bob compute $\mathsf{Enc}\left(|S_1 \cap S_2|^2\right) = \mathsf{ScalarProduct}(\mathbf{1}_{S_1}, \mathbf{1}_{S_2})^2$
4: Alice and Bob gives to the peer acting as the anonymizer $\mathsf{Enc}\left(|S_1 \cap S_2|^2\right)$ as well as the sizes of their profiles $|S_1|$ and $|S_2|$
5: The anonymizer computes the squared cosine similarity $\mathsf{Enc}\left(\frac{|S_1 \cap S_2|^2}{|S_1| \times |S_2|}\right)$ and adds Laplacian noise n_1 where $n1 \sim N$ using the homomorphic property
6: The anonymizer sends the perturbed squared cosine similarity (which is homomorphically encrypted) to Alice and Bob
7: Alice and Bob cooperate to decrypt the homomorphically encrypted value and get as output $\mathsf{ExactSquaredCosine}(S_1, S_2) + n_1$

Theorem 4 (Protocol for differential squared cosine)**.** *Algorithm 4 is secure with respect to a passive adversary and ϵ-differentially private. The protocol has communication complexity of $O(l)$ bits and a computational cost of $O(l)$.*

Proof. It follows from Theorem 2 that the first part of the protocol before the anonymizer adds the noise, is secure with respect to a passive adversary. The anonymizer has only the knowledge of the public key and thus cannot decrypt the messages it sees. Moreover, the messages are semantically secure and therefore leak no information to the anonymizer. At the end of the protocol, assuming that the anonymizer does not collude either with Alice nor Bob, Alice and Bob only get to learn ($\mathsf{ExactSquaredCosine}(S_1, S_2) + \mathsf{Lap}\left(0, \frac{\mathsf{PS}_{|S_1|,|S_2|}(\mathsf{ExactSquaredCosine})}{\epsilon}\right)$, which ensures the ϵ-differential property of the protocol according to Theorem 1. For the complexity, because of the use of the protocol ScalarProduct as a subroutine, the protocol DifferentialSquaredCosine has a communication cost of $O(l)$ bits as well as a computational cost of $O(l)$ (we consider here that the threshold decryption has constant complexity and is negligible with respect to the cost of the scalar product). □

3.3.2 Distributed noise generation

There are several possible ways to achieve ϵ-differential privacy in the context of the exact similarity depending on whether or not we have access to another peer which is assumed not to collude with any of the two peers that computes their similarity and can act as a semi-trusted party. A possible way to achieve this

would be for the two peers to add themselves the noise directly when executing the protocol for similarity computation.

For the distributed noise generation, where each of the two peers adds his own independent noise, we give thereafter an analysis of the global amount of noise that has to be added by each peer to obtain the total amount of noise required. Notice that the noise is usually a real number and that homomorphic encryption only supports integers. For now we assume that the homomorphic encryption can support real numbers and address how to achieve this in Section 3.3.3.

Two exponentials. According to Lemma 5 the difference between two exponentially distributed random variables is a Laplacian distribution. Therefore if both peers generate randomly their own exponentially distributed random variable and one peer (say Alice for instance) adds her noise while the other (Bob) subtracts his noise from the result, then the obtained result is a Laplace-distributed variable. For instance, if the desired noise has the form of $N \sim \mathsf{Lap}\left(0, \mathsf{GS}\left(f\right)/\epsilon\right)$, then each peer generates i.i.d $Y \sim \mathsf{Exponential}\left(\epsilon/\mathsf{GS}\left(f\right)\right)$. (Note that as we assume a passive adversary, each peer generate and add his share "honestly").

Definition 11 (Characteristic function of a continuous random variable)**.** *Let $f_X(x)$ be the probability density function (PDF) of the continuous random variables X, which is the probability that $X = x$, then the characteristic function of X is $\varphi_X(t) = \int_{-\infty}^{\infty} e^{itx} f_X(x)dx$, where $t \in (-\infty, \infty)$ is the parameter of the characteristic function.*

Lemma 5. *If $E_1, E_2 \sim \mathsf{Exponential}\left(\lambda\right)$ are i.i.d random variables, and $L = E_1 - E_2$, then $L \sim \mathsf{Lap}\left(0, 1/\lambda\right)$.*

Proof. The characteristic function[13] of E_1 is $\frac{\lambda}{-it+\lambda}$ and for $-E_2$ is $\frac{\lambda}{it+\lambda}$. As the two variables are independent, adding them amounts to multiplying their characteristic function, which simplifies to $\frac{\lambda^2}{t^2+\lambda^2}$ and corresponds exactly to the characteristic function of L. □

Notice that if the exact similarity is reveals, a malicious peer can remove his noise to end up with a minimum or maximum for the real similarity. Thus this method is only recommended for the threshold similarity.

Two Laplacian distributions. Suppose that Alice and Bob want to release the result the scalar product between their two profiles. At the end of the protocol Alice and Bob could both simply add independently generated random noise with distribution $\mathsf{Lap}\left(0, \frac{1}{\epsilon}\right)$ using the homomorphic property of the encryption scheme. Afterwards, they could cooperate to perform the threshold decryption and they would both get to learn the perturbed scalar product. Then Alice can subtract her own noise from the released output to recover a version of the scalar product which have been perturbed only with Bob's noise (which she cannot remove). Unlike the two exponentials method which is not recommended for exact similarity protocol, this one is recommended (so that peers can remove

[13]The PDF of the exponential distribution parametrized by λ is $\lambda e^{-\lambda x}$. The PDF of the Laplace distribution is shown in Equation (3)

their extra noise). We provide in Algorithm 5 a protocol for scalar product that satisfies ϵ-differentially privacy using the two Laplacian technique.

Algorithm 5 DifferentialScalarProduct(S_1,S_2,ϵ)

1: Alice and Bob generates the keys of the threshold homomorphic encryption
2: Alice receives sk_a, Bob receives sk_b and they both get the public key pk
3: Alice and Bob compute $\mathsf{Enc}\left(|S_1 \cap S_2|^2\right) = \mathsf{ScalarProduct}(\mathbf{1}_{S_1}, \mathbf{1}_{S_2})$
4: Alice generates Laplacian noise n_A parametrized by $\frac{1}{\epsilon}$ and computes $\mathsf{Enc}\left(|S_1 \cap S_2|\right) \oplus n_A = \mathsf{Enc}\left(|S_1 \cap S_2| + n_A\right)$ and sends the result to Bob
5: Bob generates Laplacian noise n_B parametrized by $\frac{1}{\epsilon}$ and computes $\mathsf{Enc}\left(|S_1 \cap S_2| + n_A\right) \oplus n_B = \mathsf{Enc}\left(|S_1 \cap S_2| + n_A + n_B\right)$
6: Alice and Bob cooperate to decrypt the homomorphically encrypted value and get as output $(|S_1 \cap S_2| + n_A + n_B)$

Lemma 6 (Two Laplacian)**.** *DifferentialScalarProduct (Algorithm 5) satisfies ϵ-differential privacy.*

Proof. From the composition lemma (Lemma 2) and an argument similar to the one used in Theorem 3, the algorithm DifferentialScalarProduct is at least ϵ-differentially private because from the point of view of Bob:

$$\underbrace{\overbrace{\mathcal{A}(z)}^{\epsilon-privacy}}_{\mathrm{n_A+n_B}} = \underbrace{g_{Bob}}_{\mathrm{n_B}}(\underbrace{\overbrace{\mathcal{A}_{Alice}(z)}^{\epsilon-privacy}}_{\mathrm{n_A}}) \ ,$$

and by symmetry of addition, from the point of view of Alice it is

$$\underbrace{\overbrace{\mathcal{A}(z)}^{\epsilon-privacy}}_{\mathrm{n_A+n_B}} = \underbrace{g_{Alice}}_{\mathrm{n_A}}(\underbrace{\overbrace{\mathcal{A}_{Bob}(z)}^{\epsilon-privacy}}_{\mathrm{n_B}}) \ .$$

So whether a peer removes his noise or not afterwards, the protocol remains ϵ-differentially private. □

Theorem 5 (Protocol for differential scalar product)**.** *The protocol DifferentialScalarProduct (Algorithm 5) is secure with respect to a passive adversary and ϵ-differentially private. The protocol has a communication complexity of $O(l)$ bits and a computational cost of $O(l)$.*

Proof. From Lemma 6 and argument similar to one used in Theorem 4, the proof follows for the security against a passive adversary, the ϵ-differential privacy, and the complexity. Security against a passive adversary holds because all the messages exchanges during the protocol are semantically secure. Differential privacy holds from Lemma 6. For the complexity, because of the use of the protocol ScalarProduct as a subroutine, the protocol DifferentialScalarProduct has a communication cost of $O(l)$ bits as well as a computational cost of $O(l)$ (we consider here that the threshold decryption has constant complexity and is negligible with respect to the cost of the scalar product).

□

3.3.3 Dealing with real noise as integers

One potential challenge is that noise is usually a real number while homomorphic encryption only supports integers (or rationals [47]). In this section, we provide to solve this problem for the semi-trusted third party approach, and for the two exponentials one. Note that the two Laplacian approach is similar to the two exponentials approach.

Third party noise. We assume that the third party can divide two numbers then add the noise with homomorphic encryption. Suppose that these two numbers are called x, y and that the noise to be added is called $n = n_1/n_2$ (i.e. we rationalize the noise). The operation that the third party wants to do is $\frac{x}{y} + \frac{n_1}{n_2}$, is equivalent to $\frac{xn_2+yn_1}{yn_2}$. The numerator and denominator apart can be simplified by scalarization and addition operations of integers, which can be done using the homomorphic encryption. We can then compute the group inverse of the denominator using the inversion gate from [48]. Afterwards, we multiply the numerator with the inverse of the denominator to deal with the division. In the decryption phase, the rational number can be recovered using the lattice-based technique [47]. The lattice-based technique uses a lattice with initial bases $(N, 0), (xy^{-1} \mod N, 1)$, then uses bases reduction algorithms to find the shortest vector in this lattice, whose components turn out to be the numerator and denominator.

Two exponentials noise. For the two exponentials case, we do not release the similarity but rather the threshold, therefore we do not need the lattice-based approach from [47]. Let the first exponential random variable be $\frac{e_1^A}{e_2^A}$ and the second $\frac{e_1^B}{e_2^B}$. The threshold $\tau = \frac{\tau_1}{\tau_2}$ and the computed similarity value is denoted by $\frac{\mathsf{Enc}(x)}{\mathsf{Enc}(y)}$, then the threshold equation is defined as

$$\frac{\mathsf{Enc}\,(x)}{\mathsf{Enc}\,(y)} \oplus \left(\frac{e_1^A}{e_2^A} - \frac{e_1^B}{e_2^B}\right) \leq \frac{\tau_1}{\tau_2} \ .$$

This can be simplified to the following all-integer equation

$$\left[\mathsf{Enc}\,(y) \odot \left(\tau_2 e_1^A e_2^B - \tau_2 e_2^A e_1^B\right)\right] \oplus \left[\mathsf{Enc}\,(x) \odot \left(\tau_2 e_2^A e_2^B\right)\right] \leq \mathsf{Enc}\,(y) \odot \left(\tau_1 e_2^A e_2^B\right) \ .$$

As we can see, only homomorphic addition and scalaring are needed.

3.4 Utility analysis

As differential privacy via the Laplacian mechanism implies the addition of noise, then it will also cause some error to the decision made by the threshold function, rendering some *accept* decisions to be *rejects* (false negatives) or vise versa (false positives). More precisely, a false negative arises when the protocol outputs that two peers are not (τ)-similar while they are so in fact, the opposite goes for false positives. In the case of Gossple, false negatives might reduce the number of the search results returned, while false positives might increase the number of unrelated results and thus hurt personalization and relevance of the results returned.

We chose to measure the utility in terms of the probability of *not* having false negatives because we arguably decided that a low number of results is more severe than low relevance of results. Should we change our mind and choose to measure false positives instead, a similar analysis for false positives is straightforward. In Section 3.4.5 we provide the equations for utility in terms of not having false positives, and in terms of not having false decision at all (be them positives or negatives).

3.4.1 Statistical model

We describe here the statistical model that we use for performing our analysis. This model can also be used as a guide to choose the threshold value (τ) to apply in practice as well as the privacy parameter (ϵ) (cf. Section 3.4.3), as well as measuring the achievable trade-off between utility and privacy.

In our model:

- The parameter l is the total number of items in the domain of items.
- $l_1 = |S_1|$ and $l_2 = |S_2|$ are the sizes of the profiles of the two peers (Alice and Bob).
- The random variable S represents the number of items in common between two peers (i.e the size of the set intersection $|S_1 \cap S_2|$). S has a Hypergeometric distribution [49] as we show in Lemma 7.
- N is a Laplace random variable representing the noise to be added. With parametrized sensitivity we get *adaptive* noise :

  $$N \sim \mathsf{Lap}\left(0, \mathsf{PS}_{l_1,l_2}\left(\mathsf{ExactSquaredCosine}\right)/\epsilon\right) \ ,$$

  and with the global sensitivity we get *non-adaptive* noise :

  $$N \sim \mathsf{Lap}\left(0, \mathsf{GS}\left(\mathsf{ExactSquaredCosine}\right)/\epsilon\right) \ .$$

- $F_X(x)$ is the cumulative distribution function (CDF) of the random variables X, which is the probability that $X < x$.
- $f_X(x)$ is the probability density function (PDF) of the random variables X, which is the probability that $X = x$.

Definition 12 (Hypergeometric distribution [49])**.** *A Hypergeometric distribution can be modeled as a box that contains N balls of which there are m white balls. Afterwards, n balls are drawn at random from that box without replacement. We count a success when a white ball is drawn. A Hypergeometric random variable is the numbers of successes in such a run. Straight forwardly, we also have that the maximum number of success is m.*

Lemma 7. *Assuming that all items in the domain are equally likely to be picked and that none are more frequent than others, then the the random variable S representing the number of items in common between two peers is $Hypergeometric(n = \mathsf{max}(l_1, l_2), m = \mathsf{min}(l_1, l_2), N = l)$, where l is the total number of items in the domain (which is the length of the indicator function).*

Proof. Without loss of generality assume that S_1 is the smaller set of the two sets. Let the other peer (owner of set S_2), pick $n = l_2$ items from the domain (of size $N = l$) without replacement to fill l_2 item in his set S_2. A pick is successful if the item picked is also contained within the set S_1. Therefore, the number of possible successes is at most $m = l_1$. A successful pick ends up in the set intersection of S_1 and S_2. The number of success (from the Hypergeometric distribution; Definition 12) is the size of the set intersection S as desired. □

3.4.2 The utility function

Remember that we measure utility as the percentage of similarity measures that does *not* count as a false negative after the noise has been added and that the similarity value is $S^2/(l_1 l_2)$.

Theorem 6. *The utility function is:*

$$\mathcal{U}_{-}(l_1, l_2, l, \tau, \epsilon) = 1 - \sum_{s=\lfloor\sqrt{l_1 l_2 \tau}\rfloor+1}^{\mathsf{min}(l_1,l_2)} \frac{f_S(s) F_N(\tau - \frac{s^2}{l_1 l_2})}{1 - F_S\left(\lfloor\sqrt{l_1 l_2 \tau}\rfloor\right)} \quad . \tag{11}$$

Proof. The probability that a peer gets accepted is $P(S^2/(l_1 l_2) > \tau)$, where τ is the public threshold value while the probability of getting rejected after adding the Laplacian noise is $P(S^2/(l_1 l_2) + N \leq \tau) = P(N \leq \tau - S^2/(l_1 l_2))$. Let $\theta = \sqrt{l_1 l_2 \tau}$ and $\gamma = \tau - \frac{s^2}{l_1 l_2}$. This results in the following utility function:

$$1 - P(\textit{rejected after adding the noise} | \textit{accepted before adding the noise}) =$$

$$\begin{aligned}
1 - P(N \leq \gamma | \frac{S^2}{l_1 l_2} > \tau) &= 1 - P\left(N \leq \gamma | S > \theta\right) \\
= 1 - \frac{P\left(N \leq \gamma \wedge S > \theta\right)}{1 - F_s\left(\theta\right)} &= 1 - \sum_{s>\theta} \frac{\int_{-\infty}^{\gamma} f_{N,S}(n, s)\, dn}{1 - F_s\left(\theta\right)} \\
= 1 - \sum_{s>\theta} \frac{\int_{-\infty}^{\gamma} f_N(n) f_S(s)\, dn}{1 - F_S\left(\theta\right)} &= 1 - \sum_{s>\theta} \frac{f_S(s) \int_{-\infty}^{\gamma} f_N(n)\, dn}{1 - F_S\left(\theta\right)} \\
= 1 - \sum_{s>\theta} \frac{f_S(s) F_N(\gamma)}{1 - F_S\left(\theta\right)} \quad .
\end{aligned}$$

Note that the lower limit of the sum is $\lfloor\theta\rfloor+1$ whereas the upper limit $\mathsf{min}(l_1, l_2)$. □

The utility function can be used by peers to set the security parameter ϵ dynamically depending on the size of the sets of the two peers so as to guarantees a lower bound on the utility.

The conditional probability

$$P(\textit{rejected after adding the noise} | \textit{accepted before adding the noise})$$

is equal to the probability that an item will get rejected after adding the noise, given that it would have been accepted had not we add such noise, which is

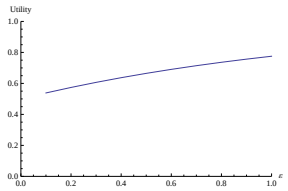


Figure 8: Setting the trade-off between privacy and utility (Non-adaptive noise, $l_1 = 10, l_2 = 10, l = 10^6, \tau = 0.8$)

the definition of a "false negative". We notice that by subtracting it from 1, we are getting the opposite probability, which is the probability of not having a false negative. Notice that this also counts the probability of having a true positive or a false positive (since the entire probability domain (1) is composed of those four cases), and hence is not equivalent to the probability of having "true negatives".

In the above proof, first we expand the conditional probability $P\left(N \leq \gamma | S > \theta\right)$ to $\frac{P(N \leq \gamma \wedge S > \theta)}{P(S > \theta)}$ by the definition of conditional probability. Then we expand the numerator to a summation (and integration) over the probabilities ($f_{N,S}$) of all the cases that count (i.e. $N \leq \gamma$ and $S > \theta$). Afterwards, since N and S are independent, then $f_{N,S} = f_N \times f_S$. We find that f_S is independent of the integration variable n, so we get it out of the integration. We notice that the integration now simply resembles the CDF of N, F_N.

In Equation 11 (which is a function of l_1, l_2, τ, ϵ, and l), assuming that l_1, l_2 are fixed, when plotted with different possible values of l_1 and l_2 shows the effect of the privacy parameter ϵ for a given threshold τ and domain cardinality l (see Figure 8).

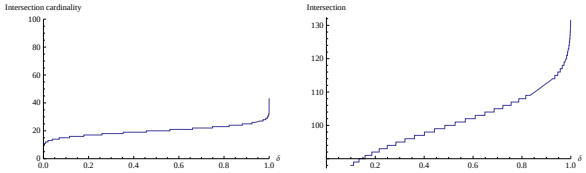
3.4.3 Selecting the threshold

Lets assume that a peer is expected on average to be similar to $r = 20\%$ of the peers he meets. Call this the *acceptance rate*. Then, the τ parameter can be chosen by substituting the desired acceptance rate (without taking into account the error caused by the addition of noise) into the *inverse* cumulative density function (inverse CDF, or CDF^{-1}) of the Hypergeometric distribution (HG). This function takes $1 - r, l_1$ and l_2 (assuming that l is fixed a priori) and gives the expected intersection size which will occure with probability near r. We say "near" because the intersection size is discrete and thus might not always match the exact desired probability. Let $S \sim Hypergeometric(n = \mathsf{max}(l_1, l_2), m = \mathsf{min}(l_1, l_2), N = l)$, then

$$1 - r = P(\frac{S^2}{l_1 l_2} \leq \tau) = CDF_S(\sqrt{l_1 l_2 \tau}) \longrightarrow CDF_S^{-1}(1 - r) = \sqrt{l_1 l_2 \tau}$$

$$\tau = \frac{CDF_S^{-1}(1 - r)^2}{l_1 l_2} \quad ,$$

where r is the desired acceptance rate. See Figure 9 for illustration of several scenarios.



(a) $CDF^{-1}HG(l_1 = 200, l_2 = 100, l = 1000)$ (b) $CDF^{-1}HG(l_1 = 10^3, l_2 = 10^5, l = 10^6)$

Figure 9: Inverse cumulative distribution function of the Hypergeometric distribution for choosing parameters

3.4.4 The impact of adaptive noise on utility

As shown by Figure 8, using the non-adaptive noise of the global sensitivity, the utility does not seem very good for small input sizes as we get a false negative rate around 50%. In particular in Figure 10, we show the utility value obtained with adaptive and non-adaptive noise for different values of τ.

τ effect on utility. The parameter τ affects the utility because the size of the set intersection cannot be bigger than the size of the smaller set. This results that the similarity (for the square cosine) is always less than or equal to $\mathsf{min}(l_1, l_2)/\mathsf{max}(l_1, l_2)$. Thus if τ is chosen above this value the false negative rate is exactly 0, because all decisions would have been also negative in the first place[14].

3.4.5 Alternative utility measures

In this section we provide alternative utility measures than the probability of not having false negatives. Choosing which measure to use depend on the application.

Utility as the probability of not having false positives. By a construction similar to that of Theorem 11, we find that the probability of not having false positives would be

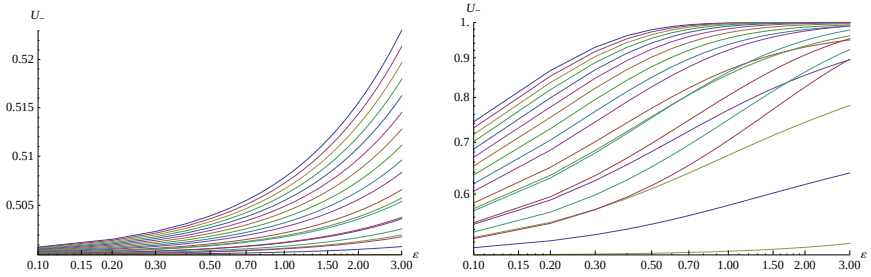
$$\begin{aligned}\mathcal{U}_+(l_1, l_2, l, \tau, \epsilon) &= 1 - P(N > \tau - \frac{S^2}{l_1 l_2} | \frac{S^2}{l_1 l_2} \leq \tau) \\ &= 1 - \sum_{s=0}^{\lfloor\sqrt{l_1 l_2 \tau}\rfloor} \frac{f_S(s)(1 - F_N(\tau - \frac{s^2}{l_1 l_2}))}{F_S\left(\lfloor\sqrt{l_1 l_2 \tau}\rfloor\right)} \quad .\end{aligned}$$

And since N is a Laplacian random variable which is symmetric about 0, then $1 - F_N(n) = F_N(-n)$, hence the utility function is

$$\mathcal{U}_+(l_1, l_2, l, \tau, \epsilon) = 1 - \sum_{s=0}^{\lfloor\sqrt{l_1 l_2 \tau}\rfloor} \frac{f_S(s) F_N(\frac{s^2}{l_1 l_2} - \tau)}{F_S\left(\lfloor\sqrt{l_1 l_2 \tau}\rfloor\right)} \quad . \tag{12}$$

We show a plot for the this utility function in Figure 11.

[14] However, this is not the case with the false positive rate.



(a) Utility with non-adaptive noise (b) Utility with adaptive noise

Figure 10: Adaptive vs. Non-adaptive noise (utility based on not having false negatives). The x-axis is the privacy parameter ϵ, and the different curves correspond to different values of the threshold τ. Both figures plot the same range of values for τ.

Utility as the probability of not having false decisions. Let r, R be the events of being rejected before and after adding noise, respectively. Let a, A be the events of being accepted before and after adding noise, respectively.

Since the events $R \wedge a, R \wedge r, A \wedge a, A \wedge r$ are independent and mutually exclusive, then

$$P(R \wedge a) + P(R \wedge r) + P(A \wedge a) + P(A \wedge r) = 1 \ .$$

So the probability of not having false decisions is

$$\begin{aligned}
\mathcal{U}_\dagger &= P\left(R \wedge r\right) + P\left(A \wedge a\right) \\
&= \sum_{0}^{\lfloor\sqrt{l_1 l_2 \tau}\rfloor} f_S\left(s\right) F_N\left(\tau - \frac{s^2}{l_1 l_2}\right) + \sum_{\lfloor\sqrt{l_1 l_2 \tau}\rfloor+1}^{\min(l_1,l_2)} f_S\left(s\right) F_N\left(\frac{s^2}{l_1 l_2} - \tau\right) \\
&= \sum_{s=0}^{\min(l_1,l_2)} f_S\left(s\right) F_N\left(d\left(s\right)\left(\tau - \frac{s^2}{l_1 l_2}\right)\right) \ ,
\end{aligned}$$

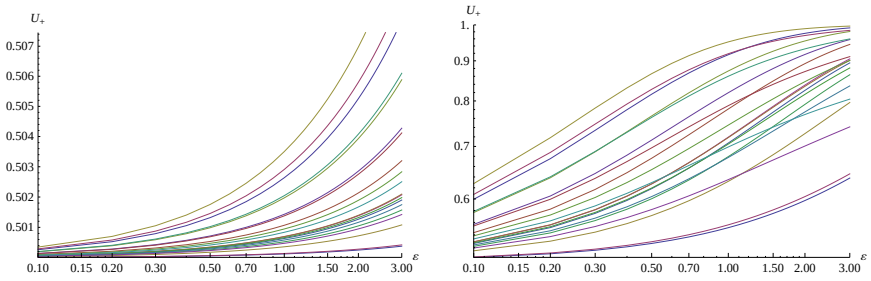
where

$$d(s) = \begin{cases} 1 & s \leq \lfloor\sqrt{l_1 l_2 \tau}\rfloor \\ -1 & s > \lfloor\sqrt{l_1 l_2 \tau}\rfloor \end{cases} \ .$$

We show a plot for the this utility function in Figure 12.

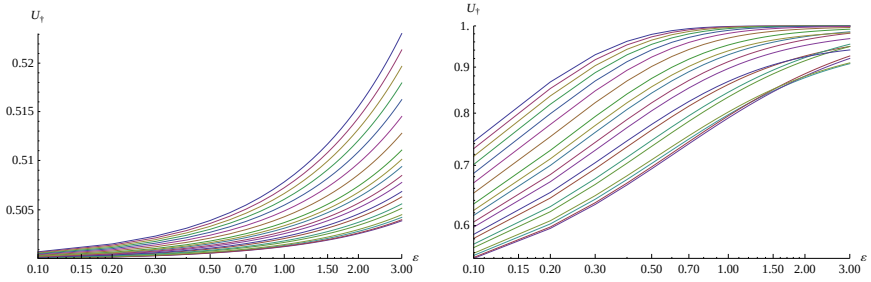
4 Conclusion and future work

The Web 2.0 has recently witnessed a proliferation of user generated content including a large proportion of personal data. Preserving privacy is a major issue to be able to leverage this information to provide personalized services. Fully decentralized systems somehow protect users privacy to be exposed to large companies, avoiding the "*Big brother is watching you*" syndrome. However, in some sense this is an illusion as they might expose personal data to other users in the network. In this internship, we have addressed this challenge in the context



(a) Utility with non-adaptive noise (b) Utility with adaptive noise

Figure 11: Utility based on the probability of not having false positives. The x-axis is the privacy parameter ϵ, and the different curves correspond to different values of the threshold τ. Both figures plot the same range of values for τ.



(a) Utility with non-adaptive noise (b) Utility with adaptive noise

Figure 12: Utility based on the probability of not having false decisions at all. The x-axis is the privacy parameter ϵ, and the different curves correspond to different values of the threshold τ. Both figures plot the same range of values for τ.

of Gossple project by providing users with a way to compute their similarity with respect to other users while preserving the privacy of their profiles.

More precisely, we have introduced a two-party threshold similarity protocol enabling a user to quantify his similarity with another user, without revealing his profile and without requiring a trusted third party. We proved that the proposed protocol is secure in the presence of a passive adversary and we have gone beyond that by providing differentially private protocols for the exact and threshold similarity. This provides a strong privacy guarantee which holds independently of the auxiliary knowledge that the adversary may have. We have also studied the impact of the noise generation on the utility of the resulting similarity.

To summarize, our work highlights the fact that *cryptography and differential privacy are two different but complementary notions*. On one hand, differential privacy gives strong privacy guarantees with respect to how much information can be learned about the inputs of the participants from the (perturbed) output of a function. Thus, differential privacy help us to reason on which type of information can be safely released with respect to privacy. On the other hand, cryptography, and more specifically secure multiparty computation, gives us the tools to compute a distributed function in a secure and robust way and remove the need for a trusted third party, which is of paramount importance in a decentralized setting. When possible, it seems therefore natural to combine differential privacy and cryptography into an integrated approach as we have done for private similarity computation in distributed systems.

The contributions of this report include a bound on the noise added to enhance the utility and including different variants of a distributed noise generation protocol. We have also identified the need for a bidirectionally anonymous channel to insure ϵ-differential privacy in a distributed setting.

Future work may include:

1. Addressing malicious participants (modeled by an active adversary) that can cheat during the execution of the protocol in order to learn the input of honest participants, perturb the output of the protocol or simply make it crash. We are especially interested in developing differentially private similarity protocols which stay robust and secure even on the presence of an active adversary. We are also interested in developing a bidirectionally anonymous channel provably secure against active adversaries by studying Crowds [13], Tor [14], AP3 [12], and MIXnets [11].

2. The application of this approach to a fully decentralized clustering algorithm and the evaluation of the achievable trade-off between utility (as measured by the quality of the global clustering) and privacy. As well as evaluating and improving the efficiency of the protocols in terms of communication and computation overhead.

3. Study the same problem for a group-similarity metric [2] that measures the overall similarity of a group of peers.

Acknowledgment

The author would like to thank his fiancée, Shaimaa Tarek El-Shoeiby, for her continuous support and for having faith in him, for giving him the motive to

keep going each day and for making him a better person, for being the happiness in his life and putting a smile on his face everyday.

He would also like to thank his brother, Ahmad Al-Alaiwat, who believed in him from the very beginning, supported him whenever he needed (and whenever he didn't), and was not just a brother, but an entire family.

The author dedicates this work to the memory of his mother, who encouraged him to read and learn ever since he was a little child, never held back an effort to push him forward in his educational path, and had dedicated her life for him. Rest In Peace.

The author acknowledges his supervisors, Sébastien Gambs and Anne-Marie Kermarrec, for their occasional meetings, inspiring discussions, non-dispensable guidance, and valuable feedback. The author is very grateful for their high level of care. The author also acknowledges Dr. Waleed Yousef for his guidance in probability.

Last but not least, this chance would not have been possible if it was not for the enormous amount of support and effort, and the huge leap forward provided by Dr. Sameh Al-Ansary, thank you so much Dr. Sameh.

References

[1] A.-M. Kermarrec, "Challenges in Personalizing and Decentralizing the Web: An Overview of GOSSPLE," in *Proceedings of the 11th International Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS '09)*, ser. Lecture Notes in Computer Science, R. Guerraoui and F. Petit, Eds., vol. 5873. Lyon, France: Springer-Verlag Berlin, Heidelberg, 2009, p. 116.

[2] M. Bertier, D. Frey, R. Guerraoui, J. Kilian, and V. Leroy, "Personalized Web Search by Gossiping with Unknown Social Acquaintances," INRIA, Research Report {RR}-6878, 2009.

[3] A.-M. Kermarrec and M. van Steen, "Gossiping in Distributed Systems," *ACM SIGOPS Operating Systems Review*, vol. 41, no. 5, pp. 2–7, 2007.

[4] M. Jelasity, S. Voulgaris, R. Guerraoui, J. Kilian, and M. van Steen, "Gossip-Based Peer Sampling," *ACM Transactions on Computer Systems (TOCS'07)*, vol. 25, no. 3, p. 8, 2007.

[5] O. Goldreich, "Cryptography and Cryptographic Protocols," *Distributed Computing*, vol. 16, no. 2, pp. 177–199, 2003.

[6] C. Dwork, "Differential Privacy: A Survey of Results," in *Proceedings of the 5th international conference on Theory and applications of models of computation (TAMC '08)*, ser. Lecture Notes In Computer Science. Xi'an, China: Springer-Verlag Berlin, Heidelberg, 2008, pp. 1–19.

[7] J. Imbrie and E. G. Purdy, "Classification of Modern Bahamian Carbonate Sediments," in *Classification of carbonate rocks: a symposium*, ser. American Association of Petroleum Geologists Meetings. American Association of Petroleum Geologists, 1962, pp. 253–272.

[8] P. Jaccard, "Étude Comparative de la Distribution Florale dans une Portion des Alpes et des Jura," *Bulletin de la Société Vaudoise des Sciences Naturelles*, vol. 37, pp. 547–579, 1901.

[9] A. Pfitzmann and M. Waidner, *Networks Without User Observability – Design Options*, ser. Lecture Notes in Computer Science. Springer-Verlag Berlin, Heidelberg, 1985, vol. 219, p. 245253.

[10] A. Pfitzmann and M. Köhntopp, *Anonymity, Unobservability, and Pseudonymity – A Proposal for Terminology*, ser. Lecture Notes in Computer Science. Springer-Verlag Berlin, Heidelberg, 2001, vol. 2009, p. 19.

[11] D. L. Chaum, "Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms," in *Communications of the ACM*, ser. Communications of the ACM, vol. 24, no. 2. ACM New York, NY, USA, 1981, pp. 84–90.

[12] A. Mislove, G. Oberoi, A. Post, C. Reis, P. Druschel, and D. S. Wallach, "AP3: Cooperative, Decentralized Anonymous Communication," in *Proceedings of the 11th Workshop on ACM SIGOPS European Workshop*, ser. ACM SIGOPS European Workshop. Leuven, Belgium: ACM New York, NY, USA, 2004, p. 30.

[13] M. K. Reiter and A. D. Rubin, "Crowds: Anonymity for Web Transactions," *ACM Transactions on Information and System Security (TISSEC '98)*, vol. 1, no. 1, pp. 66–92, 1998.

[14] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: The Second-Generation Onion Router," in *Proceedings of the 13th Conference on USENIX Security Symposium - Volume 13*, ser. USENIX Security Symposium. San Diego, CA: USENIX Association Berkeley, CA, USA, 2004, p. 21.

[15] O. Goldreich, *Foundations of Cryptography*. Cambridge University Press, 2001.

[16] A. C.-C. Yao, "Protocols for Secure Computations," in *Proceedings of the 23rd Annual Symposium on Foundations of Computer Science (FOCS '82)*. IEEE Computer Society Washington, DC, USA, 1982, pp. 160–164.

[17] M. Ben-Or, S. Goldwasser, and A. Wigderson, "Completeness Theorems for Non-Cryptographic Fault-Tolerant Distributed Computation," in *Proceedings of the 20th Annual ACM Symposium on Theory of Computing*, ser. Annual ACM Symposium on Theory of Computing. Chicago, Illinois, USA: ACM New York, NY, USA, 1988, pp. 1–10.

[18] O. Goldreich, S. Micali, and A. Wigderson, "How to Play Any Mental Game or A Completeness Theorem for Protocols with Honest Majority," in *Proceedings of the 19th Annual ACM Symposium on Theory of Computing (STOC '87)*, ser. Annual ACM Symposium on Theory of Computing, New York, NY, USA, 1987, pp. 218–229.

[19] A. Shamir, "How to Share a Secret," in *Communications of the ACM*, vol. 22, no. 11. ACM New York, NY, USA, 1979, pp. 612–613.

[20] P. Paillier, *Public-Key Cryptosystems Based on Composite Degree Residuosity Classes*, ser. Lecture Notes in Computer Science. Springer-Verlag Berlin, Heidelberg, 1999, vol. 1592, pp. 223–238.

[21] I. Damgå rd and M. Jurik, *A Generalisation, a Simplification and Some Applications of Paillier's Probabilistic Public-Key System*, ser. Lecture Notes in Computer Science. Springer-Verlag Berlin, Heidelberg, 2001, vol. 1992, pp. 119–136.

[22] A. Russell and H. Wang, "How to Fool an Unbounded Adversary With a Short Key," in *Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques: Advances in Cryptology (EUROCRYPT'02)*, ser. Lecture Notes In Computer Science, L. R. Knudsen, Ed., vol. 2332, no. 3. Springer-Verlag Berlin, Heidelberg, 2002, pp. 133 – 148.

[23] R. Cramer, I. Damgå rd, and J. B. Nielsen, *Multiparty Computation from Threshold Homomorphic Encryption*, ser. Lecture Notes in Computer Science. London, UK: Springer-Verlag Berlin, Heidelberg, 2001, vol. 2045, pp. 280–300.

[24] C. Dwork, F. McSherry, and K. Talwar, "The Price of Privacy and the Limits of LP Decoding," in *Proceedings of the 39th Annual ACM Symposium on Theory of Computing (STOC'07)*, ser. Annual ACM Symposium on Theory of Computing. San Diego, California, USA: ACM New York, NY, USA, 2007, pp. 85–94.

[25] L. Sweeney, "k-anonymity: A Model for Protecting Privacy," *International Journal of Uncertainty, Fuzziness and Knowledge Based Systems*, vol. 10, no. 5, pp. 557–570, 2002.

[26] A. Machanavajjhala, D. Kifer, J. Gehrke, and M. Venkitasubramaniam, "L-diversity: Privacy Beyond k-anonymity," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 1, no. 1, p. 3, 2007.

[27] N. Li, T. Li, and S. Venkatasubramanian, "t-Closeness: Privacy Beyond k-Anonymity and l-Diversity," in *Proceedings of the 23rd International Conference on Data Engineering (ICDE '07)*. The Marmara Hotel, Istanbul, Turkey: IEEE Computer Society Washington, DC, USA, 2007, pp. 106–115.

[28] C. Dwork, F. Mcsherry, K. Nissim, and A. Smith, *Calibrating Noise to Sensitivity in Private Data Analysis*, ser. Lecture Notes in Computer Science. Springer-Verlag Berlin, Heidelberg, 2006, vol. 3876, pp. 265–284.

[29] S. P. Kasiviswanathan, H. K. Lee, K. Nissim, S. Raskhodnikova, and A. Smith, "What Can We Learn Privately?" in *Proceedings of the 49th Annual IEEE Symposium on Foundations of Computer Science (FOCS '08)*. IEEE Computer Society Washington, DC, USA, 2008, pp. 531–540.

[30] G. S. Narayanan, T. Aishwarya, A. Agrawal, A. Patra, A. Choudhary, and C. P. Rangan, "Multi Party Distributed Private Matching, Set Disjointness and Cardinality of Set Intersection with Information Theoretic Security," in

Proceedings of the 8th International Conference on Cryptology and Network Security (CANS'09), ser. Lecture Notes In Computer Science, J. A. Garay, A. Miyaji, and A. Otsuka, Eds., vol. 5888. Kanazawa, Japan: Springer-Verlag Berlin, Heidelberg, 2009, pp. 21–40.

[31] R. Li and C. Wu, "An Unconditionally Secure Protocol for Multi-Party Set Intersection," in *Proceedings of the 5th international conference on Applied Cryptography and Network Security (ACNS'07)*, ser. Lecture Notes In Computer Science, vol. 4521. Zhuhai, China: Springer-Verlag Berlin, Heidelberg, 2007, pp. 226 – 236.

[32] L. Kissner and D. Song, *Privacy-Preserving Set Operations*, ser. Lecture Notes in Computer Science. Springer-Verlag Berlin, Heidelberg, 2005, vol. 3621, pp. 241–257.

[33] M. J. Freedman, K. Nissim, and B. Pinkas, *Efficient Private Matching and Set Intersection*, ser. Lecture Notes in Computer Science. Springer-Verlag Berlin, Heidelberg, 2004, vol. 3027, pp. 1–19.

[34] B. Goethals, S. Laur, H. Lipmaa, and T. Mielikäinen, *On Private Scalar Product Computation for Privacy-Preserving Data Mining*, ser. Lecture Notes in Computer Science. Springer-Verlag Berlin, Heidelberg, 2004, vol. 3506, pp. 104–120.

[35] R. Wright and Z. Yang, "Privacy-Preserving Bayesian Network Structure Computation on Distributed Heterogeneous Data," in *Proceedings of the 10th ACM SIGKDD international conference on Knowledge discovery and data mining*, ser. International Conference on Knowledge Discovery and Data Mining. Seattle, WA, USA: ACM New York, NY, USA, 2004, pp. 713–718.

[36] A. Amirbekyan and V. Estivill-Castro, "A New Efficient Privacy-Preserving Scalar Product Protocol," in *Proceedings of the 6th Australasian Conference on Data Mining and Analytics (AusDM'07) - Volume 70*, ser. ACM International Conference Proceeding Series, P. Christen, P. J. Kennedy, J. Li, I. Kolyshkina, and G. J. Williams, Eds., vol. 311. Gold Coast, Australia: Australian Computer Society, Inc. Darlinghurst, Australia, 2007, pp. 209–214.

[37] C. A. Melchor, B. Ait-Salem, and P. Gaborit, "A Collusion-Resistant Distributed Scalar Product Protocol with Application to Privacy-Preserving Computation of Trust," in *Proceedings of The Eighth IEEE International Symposium on Networking Computing and Applications (NCA '09)*. Cambridge, Massachusetts, USA: IEEE Computer Society Washington, DC, USA, 2009, pp. 140–147.

[38] I.-C. Wang, C.-h. Shen, J. Zhan, T.-s. Hsu, C.-J. Liau, and D.-W. Wang, "Toward Empirical Aspects of Secure Scalar Product," in *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews (TSMCC)*, ser. Special issue on information reuse and integration, vol. 39, no. 4. IEEE Press Piscataway, NJ, USA, 2009, pp. 440–447.

[39] A. Inan, M. Kantarcioglu, G. Ghinita, and E. Bertino, "Private Record Matching Using Differential Privacy," in *Proceedings of the 13th International Conference on Extending Database Technology (EDBT '10)*, ser. ACM International Conference Proceeding Series, vol. 426. Lausanne, Switzerland: ACM New York, NY, USA, 2010, pp. 123–134.

[40] A. C.-C. Yao, "How to Generate and Exchange Secrets (extended abstract)," in *Proceedings of the 27th Annual Symposium on Foundations of Computer Science (FOCS '86)*. Toronto, Ontario, Canada: IEEE Computer Society Washington, DC, USA, 1986, pp. 162–167.

[41] H.-Y. Lin and W.-G. Tzeng, *An Efficient Solution to the Millionaires' Problem Based on Homomorphic Encryption*, ser. Lecture Notes in Computer Science. Springer-Verlag Berlin, Heidelberg, 2005, vol. 3531, pp. 456–466.

[42] J. Garay, B. Schoenmakers, and J. Villegas, *Practical and Secure Solutions for Integer Comparison*, ser. Lecture Notes in Computer Science. Springer-Verlag Berlin, Heidelberg, 2007, vol. 4450, pp. 330–342.

[43] T. Nishide and K. Ohta, "Multiparty Computation for Interval, Equality, and Comparison Without Bit-Decomposition Protocol," in *Proceedings of the 10th International Conference on Practice and Theory in Public-Key Cryptography (PKC'07)*, ser. Lecture Notes in Computer Science, T. Okamoto and X. Wang, Eds., vol. 4450. Beijing, China: Springer-Verlag Berlin, Heidelberg, 2007, pp. 343–360.

[44] B. Schoenmakers and P. Tuyls, *Efficient Binary Conversion for Paillier Encrypted Values*, ser. Lecture Notes in Computer Science. Springer-Verlag Berlin, Heidelberg, 2006, vol. 4004, pp. 522–537.

[45] I. Damgå rd, M. Fitzi, E. Kiltz, J. B. Nielsen, and T. Toft, *Unconditionally Secure Constant-Rounds Multi-party Computation for Equality, Comparison, Bits and Exponentiation*, ser. Lecture Notes in Computer Science. Springer-Verlag Berlin, Heidelberg, 2006, vol. 3876, pp. 285–304.

[46] B. Schoenmakers and P. Tuyls, *Practical Two-Party Computation Based on the Conditional Gate*, ser. Lecture Notes in Computer Science. Springer-Verlag Berlin, Heidelberg, 2004, vol. 3329, pp. 129–145.

[47] P.-A. Fouque, J. Stern, and G.-J. Wackers, *CryptoComputing with Rationals*, ser. Lecture Notes in Computer Science. Springer-Verlag Berlin, Heidelberg, 2003, vol. 2357, pp. 136–146.

[48] J. Bar-Ilan and D. Beaver, "Non-Cryptographic Fault-Tolerant Computing in Constant Number of Rounds of Interaction," in *Proceedings of the 8th Annual ACM Symposium on Principles of Distributed Computing (PODC '89)*, ser. Annual ACM Symposium on Principles of Distributed Computing. Edmonton, Alberta, Canada: ACM New York, NY, USA, 1989, pp. 201–209.

[49] W. L. Harkness, "Properties of the Extended Hypergeometric Distribution," *The Annals of Mathematical Statistics*, vol. 36, no. 3, pp. 938–945, 1965.