# Private Similarity Computation in Distributed Systems: from Cryptography to Differential Privacy

Mohammad Alaggan, Sébastien Gambs, Anne-Marie Kermarrec

# Private Similarity Computation in Distributed Systems: from Cryptography to Differential Privacy*

Mohammad Alaggan[1], Sébastien Gambs[2], and Anne-Marie Kermarrec[3]

[1] Université Rennes 1 – IRISA, Rennes, France
[2] Université de Rennes 1 – INRIA/IRISA, Rennes, France
[3] INRIA Rennes Bretagne-Atlantique, Rennes, France

*Keywords*: Privacy, similarity measure, homomorphic encryption, differential privacy.

**Abstract.** In this paper, we address the problem of computing the similarity between two users (according to their profiles) while preserving their privacy in a fully decentralized system and for the passive adversary model. First, we introduce a two-party protocol for privately computing a threshold version of the similarity and apply it to well-known similarity measures such as the scalar product and the cosine similarity. The output of this protocol is only one bit of information telling whether or not two users are similar beyond a predetermined threshold. Afterwards, we explore the computation of the exact and threshold similarity within the context of differential privacy. Differential privacy is a recent notion developed within the field of private data analysis guaranteeing that an adversary that observes the output of the differentially private mechanism, will only gain a negligible advantage (up to a privacy parameter) from the presence (or absence) of a particular item in the profile of a user. This provides a strong privacy guarantee that holds independently of the auxiliary knowledge that the adversary might have. More specifically, we design several differentially private variants of the exact and threshold protocols that rely on the addition of random noise tailored to the sensitivity of the considered similarity measure. We also analyze their complexity as well as their impact on the utility of the resulting similarity measure. Finally, we provide experimental results validating the effectiveness of the proposed approach on real datasets.

## 1 Introduction

In the Web 2.0, more and more personal data are released by users (queries, social network, geolocated data...), which creates 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 some kind of clustering process that relies on an exchange of personal data (such as profiles) to compute similarity between users [2]. In this paper, we address the problem of *computing similarity between users while preserving their privacy* and *without relying on a central entity*. Dissociating the identifiers of users from their data, through the use of pseudonyms for instance, is clearly not sufficient to protect their privacy. In fact, just looking at these *Personal Identifiable Information* (PII) may sometimes be enough to infer the identity of the associated users thus causing a privacy breach [3, 22, 21]. Moreover, to preserve the fully distributed nature of such systems, no trusted third party (e.g. central server) should be required.

In this paper, we propose a protocol based on cryptographic primitives that computes the similarity between two user profiles (represented as vectors) 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. First, considering well-known similarity metrics, namely scalar product and cosine similarity, we propose a two-party *threshold similarity* protocol for these metrics and prove its security against a passive adversary. Instead of revealing the exact value of the 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 in the sense that it reveals less information. While, we focus on

* Supported by the ERC GOSSPLE project.

the scalar product and the cosine similarity for illustration purpose, our method is generic enough to be applied to other similarity metrics.

Second, we go beyond the traditional cryptographic framework by analyzing the similarity computation within the context of *differential privacy* [9]. In a nutshell, *differential privacy* is an orthogonal and complementary notion to cryptography that, by adding random noise to the output of a function, provides strong privacy guarantees with respect to how well an adversary observing the output of the function can deduce the presence (or absence) of a specific item in a profile. We design a differentially private protocol for the exact and threshold similarity and analyze their 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 these similarity metrics in the context of a protocol computing exactly the similarity between two user profiles. Finally, we also study the impact of the differential privacy (which requires the addition of random noise) on the resulting *utility* of the similarity measure, both through a theoretical analysis and experimental validation.

The paper is organized as follows. Section 2 describes the system model and provides the required background. In Section 3, we introduce the threshold similarity protocol and prove its security with respect to a passive adversary. In Section 4, we describe differentially-private protocols for the exact and threshold similarity, while in Section 5, we provide a theoretical analysis of the impact on utility of the differentially-private protocol as well as experimental results. Finally, we briefly review related work in Section 6 before concluding.

## 2 Preliminaries

In this section, we describe the system model, the definitions of the similarity metrics considered, and the background in cryptography required in the context of our contributions.

**System model.** We consider a distributed system of $n$ nodes, connected via an unstructured network [17]. (Each node is typically connected to $O(\log n)$ other nodes picked uniformly at random [4].) The nodes need to periodically run a clustering protocol that requires computing similarity between pairs of nodes. This semantic clustering can later be used to improve the search, provide content recommendation or personalized query expansion. Nodes are characterized by their profile representing their interests. For example, a node's profile can be a vector of items the associated user has tagged using a collaborative system [1] such as delicious[4]. We assume that for two different nodes $A$ and $B$, their profiles $S_A$ and $S_B$ can be represented as binary vectors of size $l$, where $l$ is the size of the domain. More precisely, $S_A = \{a_1, \ldots, a_l\}$ and $S_B = \{b_1, \ldots, b_l\}$, such that $a_i = 1$ if item $i$ is in A's profile and 0 otherwise ($b_i$ is defined similarly for the second node). For illustration purpose, we shall call the first node Alice and the second node Bob in the rest of the paper.

The profile is a personal and private information that should be protected, and therefore our main concern is *how to compute the similarity measure while preserving its privacy*. In this context, this means not revealing the content of the profile and restricting the possibility for an adversary to infer the presence or absence of a particular item in this profile. Moreover, besides the private computation of the similarity, we also assume the existence of a bidirectional anonymous lossless channel to break the link between a node's identity and its profile. Although, it is not the focus of this paper to detail how such a channel could be implemented in practice, we describe in Appendix A a simple implementation of this channel called *gossip-on-behalf*[5] that relies on the use of a third node acting as an anonymizer to break the link between the two nodes computing their similarity. Obviously, other implementations of the bidirectional anonymous channel are possible but they require non-trivial modifications of current

[4] `http://delicious.com/`

[5] The protocol described here is a modification of a protocol published earlier [4].

anonymous communication networks [5, 26, 8] and are beyond the scope of this paper[6]. In order to guarantee a high level of anonymity, as measured for instance by the size of the anonymity set, it is also necessary to assume that the size of the network is sufficiently large ($n \gg 3$). Moreover, in order to avoid the possibility for an adversary to query several times the similarity computation with different forged profiles, it is also necessary to restrict to limit the use of a particular bidirectional anonymous channel (for instance to use it only once).

**Similarity measures.** Nodes aim at detecting the most similar other nodes (i.e. those which share similar interests). Thus, we assume the existence of similarity measures that can be used by the two nodes to quantify how similar they are. A *similarity measure* $\mathsf{sim}$ is a function that takes as input two sets $S_A$ and $S_B$ representing the profiles of users Alice and Bob and outputs a value in the range between 0 and 1 (i.e. $\mathsf{sim}(S_A, S_B) \in [0, 1]$), where 0 indicates that the sets are entirely different (the profiles have no items in common) while 1 means that the sets are identical (and therefore the users can be considered as sharing exactly the same interests).

The *cosine similarity* is commonly used to assess the similarity between two sets [4] and can be seen as a normalized overlap between the sets. Formally, it is defined as

$$\frac{|S_A \cap S_B|}{\sqrt{|S_A| \times |S_B|}} \ , \tag{1}$$

where $S_A$ and $S_B$ are the private sets of the first and second node respectively and $|S_A|$ and $|S_B|$ their corresponding sizes (i.e. the number of 1s in their profiles for binary vectors). The *size of the set intersection* between $S_A$ and $S_B$ (i.e. $|S_A \cap S_B|$) is equivalent to the *scalar product* in the case where the sets are represented as binary vectors. For instance, the scalar product of two vectors of length $l$, $a = (a_1, \cdots, a_\ell)$ and $b = (b_1, \cdots, b_\ell)$, is defined as $\sum_{i=1}^{\ell} a_i b_i$. Other similarity metrics can be considered such as the Jaccard index [16], but for the sake of clarity, we focus on the cosine similarity metric and the scalar product in the sequel.

**Cryptographic background.** In this paper, we only consider privacy against a computationally-bounded *passive adversary* (also sometimes called *semi-honest* or *honest-but-curious*) that can control a fraction of the nodes (see [14] for a formal cryptographic definition). Note that in this model (contrary to the active one), nodes do not misbehave and follow the recipe of the protocol. However, they may try to infer as much information as possible regarding the private inputs of other participants from the interactions and messages they have seen and recorded.

**Definition 1 (Privacy – passive adversary [14]).** *A protocol is said to be* private with respect to passive adversary *controlling a node (or a collusion of nodes), if this adversary cannot learn (except with negligible probability) more information from the execution of the protocol that it could from its own input (i.e. the inputs of the nodes he controls) and the output of the protocol.*

In our work, we rely on a cryptographic primitive known as *homomorphic encryption*, which allows to perform arithmetic operations (such as addition and/or multiplication) on encrypted values.

**Definition 2 (Homomorphic cryptosystem).** *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*[7] $sk$*. A cryptosystem is* 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 ciphertext and a plaintext, such that* $\mathsf{Dec}(\mathsf{Enc}(c) \odot a) = c \times a$*.*

[6] However, see `http://www.torproject.org/docs/hidden-services.html` for a description of how to build an anonymous server that can be accessed by anonymous users within the network of the Tor project.

[7] In order to simplify the notation, we 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 paper.

Besides, the elementary operations of addition and multiplication, more complex arithmetic operations can also be performed on the ciphertexts, such as for instance protocols for the comparison of integers [12, 23]. These protocols take as input two encrypted integers and output whether or not they correspond to the same integer or which one is greater than the other, but without revealing the corresponding plaintexts (i.e. values of the integers).

Paillier's cryptosystem [25] is an instance of a homomorphic encryption scheme that is both additive and affine. Moreover, Paillier's cryptosystem is also *semantically secure* [14], which means that a computationally-bounded adversary cannot derive non-trivial information about the plain text $m$ encrypted from the cipher text $\mathsf{Enc}(m)$ and the public key $pk$. For instance, a computationally-bounded adversary who is given two different cipher texts encrypted with the same key of a semantic cryptosystem, cannot even decide with non-negligible probability if the two cipher texts correspond to the encryption of the same plain text or not. This is because a semantically secure cryptosystem is by essence *probabilistic*, meaning that even if the same message is encrypted twice, the two resulting ciphertexts will be different except with negligible probability. In this paper, we also use a *threshold version* of the Paillier's cryptosystem [7].

**Definition 3 (Threshold cryptosystem).** *A* $(t, n)$ threshold cryptosystem *is a public cryptosystem where at least* $t > 1$ *nodes out of* $n$ *need to actively cooperate in order to decrypt an encrypted message. In particular, no collusion of even* $(t-1)$ *nodes can decrypt a cipher text. However, any node may encrypt a value on its own using the public-key* $pk$*. After the threshold cryptosystem has been set up, each node* $i$ *gets as a result his own secret key* $sk_i$ *(for* $1 \leq i \leq n$*).*

The cooperation between nodes for the decryption usually involves an interactive cryptographic protocol during which several nodes need to combine their own secret keys with an encrypted value to be able to perform the corresponding decryption.

## 3 Threshold Similarity Protocol

The threshold similarity protocol preserves privacy by outputting only one bit of information stating whether (or not) the similarity between two profiles is above some well-chosen threshold $\tau$. To this end, we define thereafter the notion of threshold similarity.

**Definition 4 (Threshold similarity).** *Two nodes are* $\tau$*-similar if the output of applying a similarity measure* $\mathsf{sim}$ *on their respective profiles is above a certain threshold* $0 \leq \tau \leq 1$ *(i.e.* $\mathsf{sim}(S_A, S_B) > \tau$*).*

A threshold similarity protocol takes as input two profiles $S_A$ and $S_B$ (one profile per node) and outputs one bit of information, which is 1 if $S_A$ and $S_B$ are $\tau$-similar (i.e. $\mathsf{sim}(S_A, S_B) > \tau$ for $\mathsf{sim}$ a predefined similarity measure and $\tau$ the value of the threshold) and 0 otherwise. In practice, the value of the threshold $\tau$ is application dependent and is set empirically so as to be significantly above the average similarity between nodes in the population. The threshold similarity is very appealing with respect to privacy as it guarantees that the output of the similarity computation only reveals one bit of information, which is potentially much less than disclosing the exact value of the similarity measure. As a practical illustration, we show how to compute privately the cosine similarity between two profiles (Equation 1) represented as binary vectors using an algorithm that we called $\mathsf{ThresholdCosine}$. Note that our approach is generic enough to accommodate other similarity metrics such as for example Jaccard index or Hamming distance. The value of the threshold is set once and for all in advance and therefore the adversary cannot perform a kind of binary with different values for $\tau$.

As a preprocessing step to this protocol, the two nodes engage in the setup phase of a distributed key generation protocol of a threshold affine homomorphic cryptosystem [7] (see for instance [24] for a detailed description of a distributed key generation protocol without a trusted third party for the

Paillier cryptosystem). At the end of this key generation phase, both nodes receive the same public key $pk$ and each one of them gets as private input a different secret key, respectively $sk_A$ for the first node and $sk_B$ for the second node. The threshold cryptosystem[8] is such that any node can encrypt a value using the public key $pk$ but that the decryption of a homomorphically encrypted value requires the active cooperation of the two nodes.

At the beginning of the protocol, the two nodes compute the (encrypted) size of the set intersection of their two profiles $S_A$ and $S_B$ by using one of the several algorithms that can be found in the literature. Once this is done, the nodes only receive as output a ciphertext that is an encrypted version of the size (and not the size itself in plaintext). Let $k$ denote the number of items in a profile and $l$ is the size of the domain (e.g. in the dataset delicious $k$ is around 200 items and $l$ is approximately 1 million items). Some of the state-of-the-art algorithms work directly with profiles represented as sets while others are specifically designed to compute the scalar product when profiles are represented as binary vectors. For instance, the two-party scalar product protocol proposed by Goethals [13] provides semantic security for one node and information-theoretic security for the other one, for a communication cost of $O(l)$ bits and a computational complexity in terms of cryptographic operations of $O(l)$ for each node. Other recent protocols for scalar product can be found in the litterature [28, 29], but they have roughly the same complexities as Goethals' protocol. Regarding the cardinality of the set intersection, a protocol presented in [19] also provides semantic security for a communication cost of $O(k \log l)$ and a computational complexity of $O(k^2)$. Apart from those specific algorithms, generic techniques from secure multiparty computation could also be used but in general they are less efficient (see for instance an analysis in [19]). In the rest of the paper, we denote by ScalarProduct the subroutine corresponding to the use of the protocol of Goethals [13].

Afterwards, instead of computing directly the cosine similarity as denoted in Equation (1), 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 at the same time the order relation. Formally, the similarity metric effectively used in ThresholdCosine is

$$\frac{|S_A \cap S_B|^2}{|S_A| \times |S_B|} \ . \tag{2}$$

On one hand for obtaining the numerator, we square the output of the scalar product by applying the multiplication gate from [6] to multiply it by itself. On the other hand, the denominator can be computed by the first node sending its homomorphically-encrypted set cardinality to the second node (i.e. $\mathsf{Enc}(|S_A|)$), who scalarizes it by its own set cardinality by doing $\mathsf{Enc}(|S_A|) \odot |S_B|$ to obtain $\mathsf{Enc}(|S_A| \times |S_B|)$. Recall, that the objective of the ThresholdCosine protocol is only to learn if the similarity between $S_A$ and $S_B$ is above a certain (publicly known) threshold $\tau$. We assume that the threshold can be represented as a fraction $\tau = a/b$ and therefore our goal is to verify whether or not the following condition holds

$$\frac{|S_A \cap S_B|^2}{|S_A| \times |S_B|} > \frac{a}{b} \Leftrightarrow b|S_A \cap S_B|^2 > a|S_A| \times |S_B|. \tag{3}$$

The left side and right side of the inequality can be compared by using secure protocols for integer comparison [12, 23]. We choose to apply specifically the comparison technique from [23] as it does not require knowledge of the input as well as a full bit decomposition of the input. 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 $\tau$.

[8] The threshold cryptosystem should not be confused with the threshold similarity.

**Algorithm 1** $\mathsf{ThresholdCosine}(S_A, S_B)$

Alice and Bob generate the keys of the threshold homomorphic encryption
Alice receives $sk_a$, Bob receives $sk_b$ and they both get the public key $pk$
Alice and Bob compute $\mathsf{Enc}(|S_A \cap S_B|) = \mathsf{ScalarProduct}(S_A, S_B)$
Alice applies the multiplication gate from [6] to obtain $\mathsf{Enc}(|S_A \cap S_B|^2)$
Alice computes $\mathsf{Enc}(|S_A|)$ and sends it to Bob
Bob computes $\mathsf{Enc}(|S_A|) \odot |S_B| = \mathsf{Enc}(|S_A| \times |S_B|)$
Alice computes $\mathsf{Enc}(|S_A \cap S_B|^2) \odot b = \mathsf{Enc}(b|S_A \cap S_B|^2)$
Bob computes $\mathsf{Enc}(|S_A| \times |S_B|) \odot a = \mathsf{Enc}(a|S_A| \times |S_B|)$
Alice and Bob use the integer comparison protocol of [23] on $\mathsf{Enc}(b|S_A \cap S_B|^2)$ and $\mathsf{Enc}(a|S_A| \times |S_B|)$
**if** $\mathsf{Enc}(b|S_A \cap S_B|^2) > \mathsf{Enc}(a|S_A| \times |S_B|)$ **then**
    output 1 to state that Alice and Bob are $\tau$-similar
**else**
    output 0
**end if**

**Theorem 1 (Threshold cosine similarity).** *The protocol* ThresholdCosine *is private with respect to a passive adversary and returns* 1 *if two nodes are* $\tau$*-similar and* 0 *otherwise. The protocol has a communication complexity of* $O(l)$ *bits and a computational cost of* $O(l)$*, for* $l$ *being the size of the binary vectors representing the profiles.*

*Proof. All the communication exchanged between Alice and Bob is done using a homomorphic encryption scheme with semantic security, therefore the encrypted messages exchanged do not leak any information about their content. Moreover as the encryption scheme is a threshold version, neither Alice nor Bob alone can decrypt the messages and learn their content. The multiplication gate [6] as well as the integer comparison protocol [23] are also semantically secure, which therefore guarantees that the protocol is secure against a passive adversary. Regarding the correctness, it is easy to see from the execution of the protocol that if Alice and Bob are* $\tau$*-similar then this will result in* $\mathsf{Enc}(b|S_A \cap S_B|^2) > \mathsf{Enc}(a|S_A| \times |S_B|)$ *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 analysis. On the other hand, the protocol* ScalarProduct *requires the exchange of* $O(l)$ *bits between Alice and Bob as well as* $O(l)$ *computations [13]. This results in a similar complexity for the global protocol* ThresholdCosine*.*

## 4 Differentially Private Similarity Computation

Cryptography gives us the tools to compute any distributed function without revealing any other information than the output of the function itself and while removing the need for a trusted third party. 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 inputs of participants. For instance, suppose that a deterministic computation of the similarity is performed and that it outputs 1 as similarity value. In this situation, both nodes know that they exactly have the same profile. Differential privacy [9] precisely aims at addressing the problem of what can be inferred about the inputs from the output of a computation by adding some randomization to it. In that respect, differential privacy can be seen as an orthogonal but complementary notion to cryptography as it addresses a different issue. Therefore, in order to get the best of both worlds, the main idea is to combine them by using cryptographic techniques to compute securely a differentially private algorithm.

### 4.1 Differential Privacy

Apart from the traditional cryptographic definition of privacy, we are also interested in a recent notion called *differential privacy* [9]. Two inputs $X_A$ and $X_B$ are said to *differ in at most one element* if they are both equal except for possibly one entry of the inputs. For instance, if $X_A$ and $X_B$ would be databases, it would mean that they are identical except for one row.

**Definition 5 (Differential privacy [9]).** *A randomized function $K$ gives* $\epsilon-$differential privacy *if for all possible inputs $X_A$ and $X_B$ differing in at most one element, and all $S \subseteq Range(K)$,*

$$\mathsf{Pr}[K(X_A) \in S)] \leq \mathsf{exp}(\epsilon) \times \mathsf{Pr}[K(X_B) \in S)]. \tag{4}$$

*This probability is taken over all the coin tosses of $K$. ($Range(K)$ is the range of the function $K$ and* exp *refers to the exponential function.)*

Originally, differential privacy was developed within the context of private data analysis and the main guarantee is that if a differentially private mechanism is applied on a dataset composed of the personal data of individuals, no output would become significantly more (or less) probable whether or not a participant removes his data from the dataset. This means that for an adversary observing the output of the mechanism, he only gains 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 gathered. More specifically, even if the adversary knows the whole database but one individual row, a mechanism satisfying differential privacy still protects the privacy of this individual. The parameter $\epsilon$ is public and may take different values depending on the application (for instance it could be 0.01, 0.1 or even 0.25). Dwork, McSherry, Nissim and Smith have designed a general technique, called *Laplacian mechanism* [11], that achieves $\epsilon$-differential privacy for a function $f$ by adding random noise to the true answer. 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 [11].

**Definition 6 ((Global) sensitivity [11]).** *For $f : D \to \mathbb{R}$, the* sensitivity *of $f$ is*

$$\mathsf{GS}(f) = \max_{X_A, X_B \in D} \|f(X_A) - f(X_B)\|_1 \tag{5}$$

*for all $X_A$, $X_B$ differing in at most one element, where $D$ is the domain of the function (for instance for binary vectors of $l$ bits, $D = \{0,1\}^l$).*

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

**Theorem 2 (Laplacian mechanism [11]).** *For $f : D \to R$, a randomized function $K$ achieves $\epsilon$-differential privacy if it releases on input $x$*

$$K(x) = f(x) + \mathsf{Lap}(\frac{\mathsf{GS}(f)}{\epsilon}) \tag{6}$$

*for $\mathsf{GS}(f)$ the sensitivity of the function $f$ and* Lap *is a randomly generated noise according to the Laplacian distribution parametrized by $\frac{\mathsf{GS}(f)}{\epsilon}$.*

The smaller the value of $\epsilon$, the higher the privacy but also, as a result, the higher the impact might be on the utility of the resulting output. The following lemma also shows that differential privacy is a "natural" notion that composes well.

**Lemma 1 (Composition and post-processing [18]).** *If a randomized algorithm $A$ runs $k$ algorithms $A_1, \ldots, A_k$ where each $A_i$ is $\epsilon$-differentially private, and outputs a function of the results (i.e $A(x) = g(A_1(x), \ldots, A_k(x)$ for some probabilistic algorithm $g$) then $A$ is $k\epsilon$-differentially private.*

### 4.2 Differentially Private Similarity

We define two profiles $S_A$ and $S_B$ as *neighbors* if they are the same except for one particular item. Note that for simplicity and without loss of generality, we consider only neighboring profiles of the same size. For instance, $S_A$ is a neighbor of $S_B$ (and vice versa) if it is identical except for one item that may have been replaced to obtain the profile $S_B$. If the two profiles are represented as binary vectors, they are neighbors if their Hamming distance is 0 or 2 (i.e. $\|S_A \oplus S_B\| \in \{0, 2\}$). The following lemma states the sensitivity of the squared cosine similarity. (Treatment for the differentially private computation of the scalar product can be found in Appendix B.)

**Lemma 2 (Sensitivity – squared cosine similarity).** *The sensitivity of the function* $\mathsf{ExactSquaredCosine}$ *is at most* $\frac{2\min(|S_A|,|S_B|)+1}{|S_A|\times|S_B|}$.

*Proof. Consider three different profiles $S_A$, $S_B$ and $S_C$, represented as binary vectors of same size, such that $S_B$ and $S_C$ are neighbors. The computation of the cosine similarity between $S_A$ and $S_B$ requires to compute two quantities: (1) the squared size of the set intersection $|S_A \cap S_B|^2$ and (2) the multiplication of the lengths of $S_A$ and $S_B$ (i.e. $|S_A| \times |S_B|$). Replacing an object from $S_B$ to obtain $S_C$ will only increase (or decrease) the value of the set intersection by* 1 *at most. Moreover, replacing an object from $S_B$ will not change size of the profile $|S_C|$. Therefore*

$$\mathsf{GS}(Cosine^2) = \max_{\substack{S_A,S_B,S_C \\ S_B,S_C \text{ neighbors}}} \|\mathsf{sim}(S_A,S_B) - \mathsf{sim}(S_A,S_C)\| = \max_{\substack{S_A,S_B,S_C \\ S_B,S_C \text{ neighbors}}} \left\| \frac{|S_A \cap S_B|^2 - |S_A \cap S_C|^2}{|S_A| \times |S_B|} \right\|$$

$$= \max_{S_A,S_B} \left\| \frac{|S_A \cap S_B|^2 - (|S_A \cap S_B| \pm 1)^2}{|S_A| \times |S_B|} \right\| = \max_{S_A,S_B} \left\| \frac{\pm 2|S_A \cap S_B| - 1}{|S_A| \times |S_B|} \right\|$$

*And then substituting the* max *quantifier yields:*

$$= \frac{2|S_A \cap S_B| + 1}{|S_A| \times |S_B|} \leq \frac{2\min(|S_A|,|S_B|) + 1}{|S_A| \times |S_B|} \quad .$$

There are several ways to achieve $\epsilon$-differential privacy in a distributed context. For instance, if we assume that Alice and Bob have access to a *semi-trusted party* that does not collude with any of the two nodes that computes their similarity, it can be used to help Alice and Bob during the similarity computation. This is the case for instance in the gossip-on-behalf protocol (Appendix A) in which another node acts as an anonymizer. The anonymizer is semi-trusted because although it is used to connect anonymously two nodes, it is not trusted to the point of having access to the content of the messages exchanged between them due to the semantic encryption scheme used. Another possible way to achieve differential privacy would be for the two nodes to add the noise themselves directly when executing the protocol for similarity computation.

**Differential privacy via two-party computation.** For instance, suppose that Alice and Bob want to release the result of 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 $\mathsf{Lap}(\frac{1}{\epsilon})$ using the homomorphic property of the encryption scheme. Afterwards, they could cooperate to perform the threshold decryption (which remember is not the same as the threshold similarity computation) and they would both get to learn the perturbed scalar product. Finally, Alice may subtract her own noise from the released output to recover only a version of the similarity that has been randomized with Bob's noise (which she cannot remove).

**Differential privacy via semi-trusted third party.** In the context of gossip-on-behalf (Appendix A), the node that acts as an anonymizer to set up the bidirectional anonymous channel could

also generate some random noise and add it to the similarity value that has been computed by using the homomorphic property of the cryptosystem. Afterwards, the two nodes that have been involved in the similarity computation would recover the result using the threshold decryption. The following algorithm describes this procedure.

**Algorithm 2** DifferentialSquaredCosine($S_A$,$S_B$,$\epsilon$)

1: Alice and Bob generate 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 $\mathsf{Enc}(|S_A \cap S_B|^2) = \mathsf{ScalarProduct}(S_A, S_B)^2$
4: Alice and Bob gives to the node acting as the anonymizer $\mathsf{Enc}(|S_A \cap S_B|^2)$ as well as the sizes of their profiles $|S_A|$ and $|S_B|$
5: The anonymizer computes the squared cosine similarity $\mathsf{Enc}(\frac{|S_A \cap S_B|^2}{|S_A| \times |S_B|})$ and adds Laplacian noise parametrized by $\frac{\mathsf{GS}(Cosine^2)}{\epsilon} = \frac{2\min(|S_A|,|S_B|)+1}{\epsilon \times |S_1| \times |S_2|}$ 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 $(\mathsf{ExactSquaredCosine}(S_A, S_B) + \mathsf{Lap}(\frac{2\min(|S_A|,|S_B|)+1}{\epsilon \times |S_A| \times |S_B|}))$

**Theorem 3 (Protocol for differential squared cosine).** *The protocol DifferentialSquaredCosine is private with respect to a passive adversary and $\epsilon$-differentially private. The protocol has a communication complexity of $O(l)$ bits and a computational cost of $O(l)$, for $l$ the size of the binary vectors representing the profiles.*

*Proof. All the communication exchanged between Alice and Bob are done using a homomorphic encryption scheme with semantic security, 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. At the end of the protocol providing that the anonymizer does not collude either with Alice or Bob, Alice and Bob only get to learn* $(\mathsf{ExactSquaredCosine}(S_A, S_B) + \mathsf{Lap}(\frac{2\min(|S_A|,|S_B|)+1}{\epsilon \times |S_A| \times |S_B|}))$*, which ensures the $\epsilon$-differential property of the protocol. Moreover, 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).*

Note that in this protocol, where the noise needed to reach differential privacy is added by the semi-trusted third party, it needs to know the value of $|S_A|$ and $|S_B|$ (or at least an upper bound on these values) to be able to add noise tailored to the sensitivity of the function.

**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 (for instance the output of the threshold similarity), because it amounts to flipping this value with some non-negligible probability. Instead, the most direct way to achieve differential privacy is to add the noise before the application of the threshold function. The following observation states that this does not hurt the privacy guarantee obtained.

**Observation 1 (Impact of threshold on privacy)** *Applying the Laplacian mechanism before the threshold function does not hurt the differential privacy guarantee.*

*Proof. Suppose that we have some output of a function $f$ to which we have added some Laplacian noise calibrated to the sensitivity $\mathsf{GS}(f)$ of the function as well as $\epsilon$. As stated by Lemma 1 as this output is $\epsilon$-differentially private, performing some pre-determined post-processing on it such as applying a*

*threshold function before releasing it has no impact on the privacy guarantees. Therefore, the threshold function is by itself $\epsilon$-differentially private if it is fed with some similarity measure that has been computed with a $\epsilon$-differentially private algorithm.*

In the previous observation as well as in the context of Lemma 1, note that $k = 1$ as only one differential privacy mechanism (namely $A_1 = f$) is applied. The threshold function itself corresponds to $g$ as it only counts as a post-processing step whose input is not the original profiles of nodes but rather the output of a differentially private mechanism.

## 5 Utility Analysis

In this section, we are interested in evaluating the impact of differential privacy on the utility of the application.

**Theoretical analysis.** In particular, we are interested in measuring the amount of false negatives induced by applying differential privacy on a specific similarity metric. A false negative arises when the protocol outputs that two nodes are not $\tau$-similar while in fact they are. In particular, we have derived an equation that takes the threshold value ($\tau$) and the privacy parameter ($\epsilon$) as parameters and computes the probability of having false negatives when we use the differentially private similarity metric. This equation may be use to guide and set up the different parameters of the algorithms and also to measure the achievable trade-off between utility and privacy. We focus primarily on false negatives for the analysis because we believe that a high rate of false negatives will have a big impact on the utility while a high rate of false positives will mainly hurt privacy. However, the rate of false positives can be also derived straightforwardly by following the same approach we used to compute the rate of false negatives.

In our model, the parameter $l$ is the total number of items in the domain of items (which we assume to be a finite domain) and $l_A = |S_A|$ and $l_B = |S_B|$ are the sizes of the profiles of Alice and Bob. The random variable $S$ represents the number of items in common between any two profiles picked at random with the given sizes $l_A$ and $l_B$ (i.e the size of the set intersection $|S_A \cap S_B|$).

**Lemma 3 (Hypergeometric distribution [15]).** *$S \sim Hypergeometric(\max(l_A, l_B), \min(l_A, l_B), l)$, where $l$ is the total number of items in the domain (which is the size of the binary vectors).*

*Proof. Let min be the set which has the smallest size among the two sets (we assumed it is $S_A$ without loss of generality). Fix min, and let Bob (owner of set $S_B$), pick $l_B$ items from the domain of size $l$ without replacements. A pick is successful if the item picked is also contained within the set min, hence the number of possible success is at most $l_B$. This corresponds exactly to the definition of the Hypergeometric distribution.*

Remember that the utility is measured 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 $S^2/(l_A l_B)$.

**Definition 7 (Utility function).** *The utility function is:*

$$u(l_A, l_B, l, \tau, \epsilon) = 1 - P(N \leq \tau - \frac{S^2}{l_A l_B} | \frac{S^2}{l_A l_B} > \tau) = \quad = 1 - \sum_{s=\lceil \sqrt{l_A l_B \tau} \rceil}^{\min(l_A, l_B)} \frac{f_S(s) F_N(\tau - \frac{s^2}{l_A l_B})}{1 - F_S\left(\sqrt{l_A l_B \tau}\right)} \ ,$$

for $S \sim Hypergeometric(\max(l_A, l_B), \min(l_A, l_B), l)$. The $\tau$ parameter can be chosen by substituting the desired acceptance rate of the threshold similarity (without taking into account the error caused by the addition of noise) into the inverse cumulative density function (CDF) of the Hypergeometric distribution. This is a function of $l_A$ and $l_B$ assuming that $l$ is fixed a priori and it corresponds to an integer, which when divided by the minimum size among both sets, gives the threshold $\tau$. To summarize, we have $\tau = CDF^{-1}(\max(l_A, l_B), \min(l_A, l_B), l, r)/\min(l_A, l_B)$, where $r$ is the desired

acceptance rate. The utility function can be used by nodes to set the privacy parameter $\epsilon$ dynamically depending on the size of the sets of the two nodes (see Appendix C for more details).

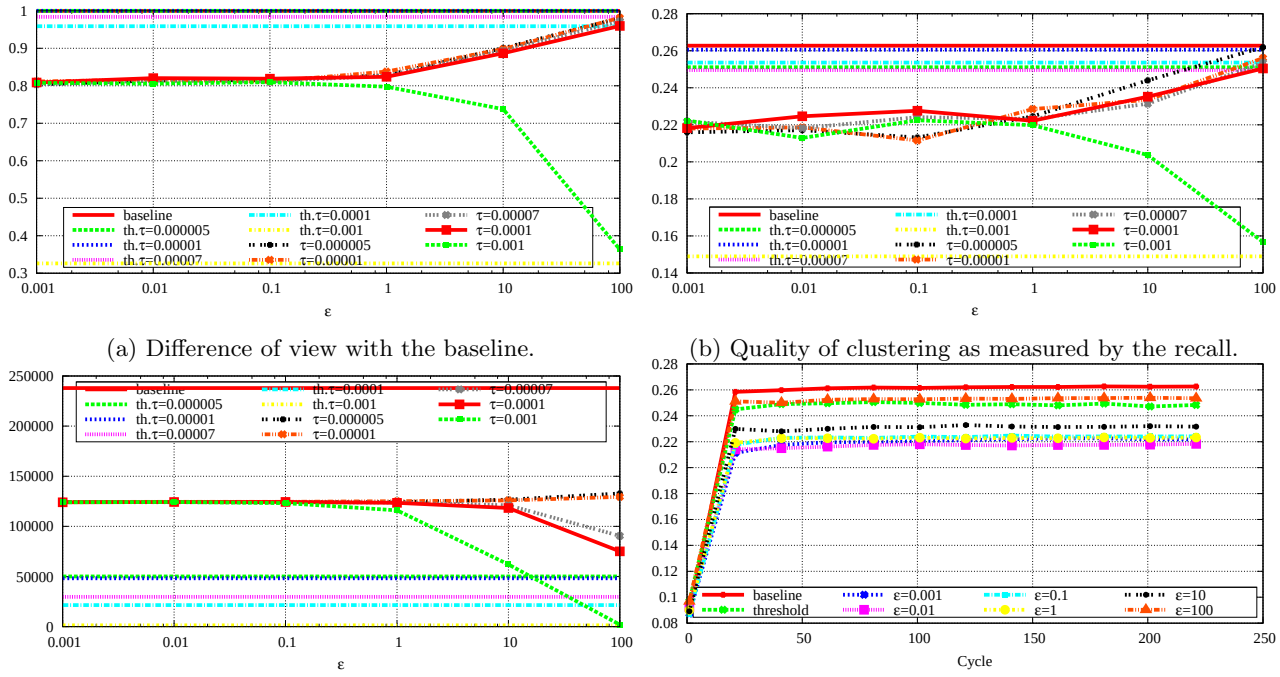
**Experimental evaluation.** We have also studied experimentally the proposed mechanisms in the context of a fully decentralized clustering algorithm [4] and evaluate the achievable trade-off between utility (as measured by the quality of the global clustering) and privacy. The clustering algorithm groups nodes according to their interests. In the baseline implementation of the clustering algorithm (which we refer simply as "baseline" in the sequel), each node samples the network and exchanges a digest of its profile that is a Bloom filter representation of its vector profile, which it uses to compute its squared cosine similarity with other nodes. Based on that value, each node iteratively sorts its clustering view and retains the $c$ closest nodes according to the computed similarity metric ($c$ is set to 10 in our experiments). After a predetermined number of cycles when the protocol converges, each node should end up with the $c$ most similar (closest) nodes in its view. (More details about this algorithm are available in [4].)

We used a dataset from delicious in which users tags items (i.e. URLs). The user profile is represented as a vector of tagged items such that there is a 1 in each vector entry corresponding to an item a user has tagged and 0 otherwise. In our experiments, we compare two models against the baseline model. The first one is a *threshold* similarity protocol, where nodes exchange their Bloom filters only if the threshold protocol presented in Algorithm 2, outputs 1. This protocol computes privately the similarity measure and outputs 1 if the similarity between the two nodes exceeds the predetermined threshold $\tau$. If a node has in its view less than $c$ nodes whose similarity is above $\tau$, the rest of the view is chosen at random and the Bloom filters are not transmitted. The second model, which is the *threshold differentially private protocol (TDP)*, is a variant of the threshold version in which we added the property of differential privacy to the cryptographic protocol. Computing similarity between two nodes requires $O(k)$ bits, where $k$ is the size of the Bloom filter in the baseline model, while using homomorphic encryption to encrypt each bitresult in an expansion factor of $\sim 2048$ due to the size of the generated ciphertexts.

*Experimental setup.* Evaluations are conducted through the simulation of a network of 500 nodes. Each node represents a user, selected randomly from a dataset of $20,000$ users from a delicious trace crawled in 2009. The resulting domain of items is a set of $1,144,000$ URLs. In this datatset, the average number of items tagged by a user is 323 and the average similarity between pair of users is 0.00004 (which explained why the chosen values for $\tau$ may seem relatively low). Specifically in the experiments, we have set $\tau \in \{10^{-3}, 10^{-4}, 7 \times 10^{-5}, 10^{-5}, 1 \times 10^{-5}, 5 \times 10^{-6}\}$ and $\epsilon \in \{0.001, 0.01, 0.1, 1, 10, 100\}$. We evaluate our two models (threshold and TDP) according to the following metrics: the *quality of the clustering* and the *level of privacy*. The quality of the clustering is measured by *(i)* the difference between the cluster view obtained by Threshold and TDP protocols compared to the baseline and, *(ii)* the recall when looking for items (previously removed from the profiles) in the profiles of the $c$ closest nodes. More specifically, clustering is done based on 90% of the tagged items of each user and the remaining 10% is used to measure the recall by comparing how many of them are in the set of tagged items of the view the node ends up with. The level of privacy is measured as the number of Bloom filters exchanged as well as the chosen value for $\epsilon$.

*Results.* In Figure 1, the x-axis represents the privacy parameter $\epsilon$. The larger its value, the less privacy (i.e. noise) is provided. We plot the experiments as a constant function with respect to $\epsilon$. The threshold line for a given $\tau$ should be interpreted as the upper bound of the performance of the "private experiments" with the same value for $\tau$. Therefore, the less the number of the Bloom filters exchanged, the better for the users' privacy. The results obtained demonstrate that the number of Bloom filters exchanged are up to half that of the baseline. Yet, for most of our choices of $\tau$, the recall and view quality in the threshold experiment are close to the one obtained with the baseline. Note that, as observed on Figure 1b, the recall of the baseline is 0.26, which is mainly due to the sparsity of the dataset. The exception being for the value of $\tau = 0.001$ which turns out to be much higher

than the average similarity (0.00004), resulting in almost no exchange of Bloom filters (which has the same effect has letting the nodes chosen their view at random). We observe that for some choices of $\tau$, adding privacy (in terms of noise) can even enhance the utility. For instance, when we add a large amount of noise and that the threshold is extremely low, this will increase the number of false positives, thus resulting in more exchanges of Bloom filters than with the use of the threshold alone without the addition of noise. Finally, Figure 1d displays the convergence time obtained with $\tau = 0.00007$. The convergence plots for the view and the Bloom exchanges are similar to the one presented before. Moreover, we observe that in all runs, the private protocols converges almost as fast as the baseline (in less than 25 cycles) to their optimal value, with respect to the quality of the view and the recall. To summarize, applying the threshold (respectively the TDP) protocol impacts only slightly the recall by 4% (respectively 12%) but reduces up to 80% the number of Bloom filters exchanged, thus providing a higher privacy. Therefore, we can conclude that it is possible to achieve reliable clustering and high recall even if instead of exchanging Bloom filters, we use a differentially private threshold mechanism for computing the similarity between nodes.




(a) Difference of view with the baseline.

(b) Quality of clustering as measured by the recall.

(c) Number of Bloom filters exchanged.

(d) Recall obtained for a fixed $\tau = 0.00007$.

Fig. 1: Experimental results obtained with 500 nodes from Delicious for different values of $\epsilon$ and $\tau$.

## 6 Conclusion

**Main results.** 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 paper, we have addressed this challenge 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 her similarity with another user, without revealing her profile and without requiring a trusted third party. We proved that the proposed protocol is secure in the presence of a passive adversary. We have also proposed differentially private protocols for the exact and threshold similarity and 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 helps 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 multi-party computation, gives us the tools to compute a distributed function in a secure and robust way and removes 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.

**Related work.** Distributed noise generation has been addressed in the context of the secure multi-party and differential privacy [10]. The resulting protocol has a greater complexity than our approach, but on the other hand is secure against active (Byzantine) adversary. A protocol for nearest neighbor search in distributed settings has been proposed in [27] but it was designed only within the cryptography framework and not the differential privacy context. Therefore, if it is possible that some privacy breach related to specific item in the profile may arise if the adversary has some background knowledge. Differential private protocols have also been considered in centralized systems such as [20] for analyzing the recommender system of Netflix with respect to differential privacy.

**Future work.** Currently, we have mainly focused on providing security with respect to a passive adversary, which can be seen as a privacy analysis of how much knowledge can be inferred by an adversary following the rules of the protocol but trying to extract as much information as possible from the transcript of the communications seen, the output of the protocol and its own input. While this is a first step, we plan as future work to address malicious participants (modeled by active adversaries) that can cheat during the execution of the protocol. It might also be possible to add the unlinkability property to the bidirectional anonymous channel to prevent an adversary from linking two queries to the same honest node.

## Acknowledgements

We are very grateful to the anonymous reviewers for their constructive comments that have help us to improve the quality of this paper.

## References

1. Amer-Yahia, S., Benedikt, M., Lakshmanan, L.V.S., Stoyanovich, J.: Efficient Network Aware Search in Collaborative Tagging Sites. PVLDB'08 1(1) (August, 2008)
2. Bai, X., Bertier, M., Guerraoui, R., Kermarrec, A.M., Leroy, V.: Gossiping Personalized Queries. In: EDBT'10. Lausanne, Switzerland (22-26 March, 2010)
3. Barbaro, M., Zeller, T.: A face is exposed for AOL searcher No. 4417749. New York Times (2006)
4. Bertier, M., Frey, D., Guerraoui, R., Kermarrec, A.M., Leroy, V.: The Gossple Anonymous Social Network. In: Middleware'10, ACM/IFIP/USENIX. Bangalore, India (November 29 – December 3, 2010)
5. Chaum, D.: Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms. CACM'82 24(2) (February, 1982)
6. Cramer, R., Damgård, I., Nielsen, J.B.: Multiparty Computation from Threshold Homomorphic Encryption. In: EUROCRYPT'01. Innsbruck, Austria (6-10 May, 2001)
7. Damgård, I., Jurik, M.: A Generalisation, a Simplification and Some Applications of Paillier's Probabilistic Public-Key System. In: PKC'01. Cheju Island, Korea (13-15 February, 2001)

8. Dingledine, R., Mathewson, N., Syverson, P.F.: Tor: The Second-Generation Onion Router. In: Proceedings of the 13th USENIX Security Symposium. San Diego, California, USA (9-13 August, 2004)
9. Dwork, C.: Differential Privacy: A Survey of Results. In: TAMC'08. Xi'an, China (25-29 April, 2008)
10. Dwork, C., Kenthapadi, K., McSherry, F., Mironov, I., Naor, M.: Our Data, Ourselves: Privacy Via Distributed Noise Generation. In: EUROCRYPT'06. St. Petersburg, Russia (May 28 – June 1, 2006)
11. Dwork, C., McSherry, F., Nissim, K., Smith, A.: Calibrating Noise to Sensitivity in Private Data Analysis. In: TCC'06. New York, USA (4-7 March, 2006)
12. Garay, J.A., Schoenmakers, B., Villegas, J.: Practical and Secure Solutions for Integer Comparison. In: PKC'07. Beijing, China (16-20 April, 2007)
13. Goethals, B., Laur, S., Lipmaa, H., Mielikäinen, T.: On Private Scalar Product Computation for Privacy-Preserving Data Mining. In: ICISC'04. Seoul, Korea (2-3 December, 2004)
14. Goldreich, O.: Foundations of Cryptography. Cambridge University Press (2001)
15. Harkness, W.L.: Properties of the Extended Hypergeometric Distribution. The Annals of Mathematical Statistics 36(3) (June, 1965)
16. Jaccard, P.: Étude Comparative de la Distribution Florale dans une Portion des Alpes et des Jura. Bulletin de la Société Vaudoise des Sciences Naturelles 37(142) (1901)
17. Jelasity, M., Voulgaris, S., Guerraoui, R., Kermarrec, A.M., van Steen, M.: Gossip-based Peer Sampling. TOCS'07 25(3) (August 2007)
18. Kasiviswanathan, S.P., Lee, H.K., Nissim, K., Raskhodnikova, S., Smith, A.: What Can We Learn Privately? In: FOCS'08. Philadelphia, Pennsylvania, USA (25-28 October, 2008)
19. Kissner, L., Song, D.X.: Privacy-Preserving Set Operations. In: CRYPTO'05. Santa Barbara, California, USA (14-18 August, 2005)
20. McSherry, F., Mironov, I.: Differentially Private Recommender Systems: Building Privacy into the Netflix Prize Contenders. In: SIGKDD'09, ACM. Paris, France (June 28 – July 1, 2009)
21. Narayanan, A., Shmatikov, V.: Robust De-anonymization of Large Sparse Datasets. In: Proceedings of the 29th IEEE Symposium on Security and Privacy. Oakland, California, USA (18-21 May, 2008)
22. Narayanan, A., Shmatikov, V.: De-anonymizing Social Networks. In: Proceedings of the 30th IEEE Symposium on Security and Privacy. Oakland, California, USA (17-20 May, 2009)
23. Nishide, T., Ohta, K.: Multiparty Computation for Interval, Equality, and Comparison Without Bit-Decomposition Protocol. In: PKC'07. Beijing, China (16-20 April, 2007)
24. Nishide, T., Sakurai, K.: Distributed Paillier Cryptosystem without Trusted Dealer. In: WISA'10. Jeju Island, Korea (24-26 August, 2010)
25. Paillier, P.: Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. In: EUROCRYPT'99 (2-6 May, 1999)
26. Reiter, M.K., Rubin, A.D.: Crowds: Anonymity for Web Transactions. TISSEC'98 1(1) (November, 1998)
27. Shaneck, M., Kim, Y., Kumar, V.: Privacy Preserving Nearest Neighbor Search. In: ICDM'06, IEEE. Hong Kong, China (18-22 December, 2006)
28. Wright, R.N., Yang, Z.: Privacy-Preserving Bayesian Network Structure Computation on Distributed Heterogeneous Data. In: SIGKDD'04, ACM. Seattle, Washington, USA (22-25 August, 2004)
29. Yao, D., Tamassia, R., Proctor, S.: Private Distributed Scalar Product Protocol With Application To Privacy-Preserving Computation of Trust. In: IFIPTM'07. Moncton, New Brunswick, Canada (July 30 – August 2, 2007)

## A Gossip-on-Behalf

In *gossip-on-behalf*, the first node, Alice, starts by choosing at random another node $C$ (that we refer thereafter as Charlie) from her random sample (typically provided by a random peer sampling service [17]). She then generates a pair of public key/secret key for this session and asks Charlie to select a node at random (that we called Bob) as the second node that will be involved in the similarity computation. Charlie does not disclose the identity of Bob to Alice (and vice versa), therefore acting as an anonymizer. Afterwards, Charlie transmits the public key of Alice to Bob that will use it to encrypt any data exchanged with Alice. Finally, Bob either generates also a pair of public key/secret key for this session or a secret that will be used as the key of a symmetric cryptosystem (such as AES) and transmits this to Bob encrypted with his public key via Charlie as a relay (this is similar in spirit to the SSL authentication protocol). Alice and Bob now share a secure anonymous channel. The communication between Alice and Bob goes through Charlie but as it is encrypted, this forbids Charlie from learning any information exchanged during their interactions. One important security assumption is that Charlie does not collude neither with Alice nor with Bob, or otherwise this would break the anonymity property of the channel.

## B Scalar Product

**Lemma 4 (Sensitivity – scalar product).** *The sensitivity of the function* ScalarProduct *is 1.*

*Proof. Consider three different profiles $S_A$, $S_B$ and $S_C$, represented as binary vectors of the same size, such that $S_B$ and $S_C$ are neighbors. Replacing an object from $S_B$ by another object to obtain $S_C$ increases (or decrease) the value of the scalar product by 1 at most, and therefore the sensitivity of the scalar product is 1.*

**Algorithm 3** DifferentialScalarProduct($S_A$,$S_B$,$\epsilon$)

1: Alice and Bob generate 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 $\mathsf{Enc}(|S_A \cap S_B|) = \mathsf{ScalarProduct}(S_A, S_B)$
4: Alice generates Laplacian noise parametrized by $\mathsf{Lap_A}(\frac{1}{\epsilon})$ and computes $\mathsf{Enc}(|S_A \cap S_B|) \oplus \mathsf{Enc}(\mathsf{Lap_A}(\frac{1}{\epsilon})) = \mathsf{Enc}(|S_A \cap S_B| + \mathsf{Lap_A}(\frac{1}{\epsilon}))$ and sends the result to Bob
5: Bob generates Laplacian noise parametrized by $\mathsf{Lap_B}(\frac{1}{\epsilon})$ and computes $\mathsf{Enc}(|S_A \cap S_B| + \mathsf{Lap_A}(\frac{1}{\epsilon})) \oplus \mathsf{Enc}(\mathsf{Lap_B}(\frac{1}{\epsilon})) = \mathsf{Enc}(|S_A \cap S_B| + \mathsf{Lap_A}(\frac{1}{\epsilon}) + \mathsf{Lap_B}(\frac{1}{\epsilon}))$
6: Alice and Bob cooperate to decrypt the homomorphically encrypted value and get as output $(|S_A \cap S_B| + \mathsf{Lap_A}(\frac{1}{\epsilon}) + \mathsf{Lap_B}(\frac{1}{\epsilon}))$

**Theorem 4 (Protocol for differential scalar product).** *The protocol* DifferentialScalarProduct *is private with respect to a passive adversary and $\epsilon$-differentially private. The protocol has a communication complexity of $O(l)$ bits and a computational cost of $O(l)$, for $l$ the size of the binary vectors representing the profiles.*

*Proof. All the communication exchanged between Alice and Bob are done using an homomorphic encryption scheme with semantic security, 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. At the end of the protocol, Alice and Bob only get to learn $(|S_A \cap S_B| + \mathsf{Lap_A}(\frac{1}{\epsilon}) + \mathsf{Lap_B}(\frac{1}{\epsilon}))$ which ensures the $\epsilon$-differential property of the protocol[9]. Moreover, because of the use of the protocol* ScalarProduct *as a subroutine, the protocol* DifferentialScalarProduct *has a communication cost of $O(l)$ bits and as well as a computational cost of $O(l)$ (we consider here that the threshold decryption and the generation of Laplacian noise have constant complexity and are negligible with respect to the cost of the scalar product).*

## C Utility Analysis

The utility function can be used by nodes to set the privacy parameter $\epsilon$ dynamically depending on the size of the sets of the two nodes. The probability that a node gets accepted is $P(S^2/(l_A l_B) > \tau)$, where $\tau$ is the public threshold value while the probability of getting rejected (false negative rate) after adding the Laplacian noise is $P(S^2/(l_A l_B) + N \leq \tau) = P(N \leq \tau - S^2/(l_A l_B))$. This result in the following utility function:

$$1 - P(N \leq \gamma | \frac{S^2}{l_A l_B} > \tau) = 1 - P\left(N \leq \gamma | S > \theta\right) = 1 - \frac{P\left(N \leq \gamma \wedge S > \theta\right)}{1 - F_S\left(\theta\right)} = 1 - \sum_{s>\theta} \frac{\int_{-\infty}^{\gamma} f_{N,S}(n,s)\, dn}{1 - F_s\left(\theta\right)}$$

$$= 1 - \sum_{s>\theta} \frac{\int_{-\infty}^{\gamma} f_N(n) f_S(s)\, dn}{1 - F_S\left(\theta\right)} = 1 - \sum_{s>\theta} \frac{f_S(s) \int_{-\infty}^{\gamma} f_N(n)\, dn}{1 - F_S\left(\theta\right)} = 1 - \sum_{s>\theta} \frac{f_S(s) F_N(\gamma)}{1 - F_S\left(\theta\right)} \ ,$$

[9] More precisely, Alice can learn $(|S_A \cap S_B| + \mathsf{Lap_B}(\frac{1}{\epsilon}))$ if she subtracts her own noise but this still preserves $\epsilon$-differential privacy (the same reasoning can be made for Bob).

where $\theta = \sqrt{l_A l_B \tau}$ and $\gamma = \tau - \frac{S^2}{l_A l_B}$. The upper limit of this sum is $\min(l_A, l_B)$ whereas the lower limit is $\lceil \theta \rceil$. The above equation (which is a function of $l_A$, $l_B$, $\tau$, $\epsilon$, and $l$), when plotted with different values of $l_A$ and $l_B$, shows the effect of the privacy parameter $\epsilon$ for a given $\tau$ and domain cardinality $l$. Alternatively, we can also derive the probability of not having false positives or the probability of not having false decisions as described below.

**Definition 8 (Utility as the probability of not having false positives.).**

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

**Definition 9 (Utility as the probability of not having false decisions.).**

$$\mathcal{U}_\dagger = \sum_{s=0}^{\min(l_A, l_B)} f_S(s) F_N\left(d(s)\left(\tau - \frac{s^2}{l_A l_B}\right)\right) \quad ,$$

*where*

$$d(s) = \begin{cases} 1 & if s \leq \lfloor \sqrt{l_A l_B \tau} \rfloor \\ -1 & if s > \lfloor \sqrt{l_A l_B \tau} \rfloor \end{cases} \quad .$$