Privacy Preserving Estimation of Social Influence

Tamir Tassa
The Open University
Ra'anana, Israel
tamirta@openu.ac.il

Francesco Bonchi
Yahoo Labs
Barcelona, Spain
bonchi@yahoo-inc.com

ABSTRACT

Exploiting word-of-mouth effect to create viral cascades in social networks is a very appealing possibility from the marketing standpoint. However, in order to set up an effective viral marketing campaign, one has first to accurately estimate social influence. This is usually done by analyzing user activity data. As we point out in this paper, the data analysis and sharing that is needed to estimate social influence raises important privacy issues that may jeopardize the legal, ethical and societal acceptability of such practice, and in turn, the concrete applicability of viral marketing in the real world.

In this paper we devise secure multiparty protocols that allow a group of service providers and a social networking platform to jointly compute social influence, in a privacy preserving manner.

1. INTRODUCTION

The idea of *viral marketing* is that of exploiting a pre-existing social network in order to increase brand awareness or to achieve other marketing objectives (such as product sales) through self-replicating viral processes, analogous to the spread of viruses. More concretely, the idea is to "target" the most influential users in the network so that, hopefully, they will create a word-of-mouth driven cascade, potentially delivering the marketing message to a large portion of the network, with a small initial marketing cost.

The key computational problem behind viral marketing is that known as *influence maximization*, formalized by Kempe *et al.* [1] as a constrained optimization problem: given a directed social graph, where each arc is labeled with a probability representing the strength of social influence along the arc, and given a budget k, find k "seed" nodes such that activating them maximizes the expected number of nodes that eventually get activated. This problem has received a great deal of attention by the data mining research community in the last decade. Regardless of this effort, the field is still far from maturity: influence maximization is still an ideal problem, mainly studied from the algorithmic standpoint, under strong assumptions and with a large gap from real-world applicability.

A first observation is that the majority of research in the field assumes that the input social graph comes already with the influence strength associated to each link. However, this is not the case in real-world social networks. Following this observation, various researchers [2, 3, 4] have turned their attention to the problem of learning social influence strength. Given (1) the social graph and (2) a log of past propagation traces (for instance, the history of sales of some products to the users in the social network) and assuming a *propagation model* which governs the manner in which influence-driven propagations occur, the problem is to learn the parameters of the model, i.e., the influence probability associated to each arc.

In real life, however, social networking platforms are owned by a third party such as Facebook or Twitter (we refer to the network owner as the *host*). The proprietary social graph is an important asset with inestimable value, thus the host keeps it secret for obvious reasons of commercial benefits, as well as due to privacy legislation[1][2]. For the very same reasons, the service provider that is interested in setting up the viral marketing campaign (e.g., an e-book store), must keep his historic sales data secret. Therefore, we face a scenario in which two distinct parties (the host H and the service provider P) hold, one each, the two pieces that constitute the input to the problem of learning the influence strength: H has the social graph, while P holds the log of past propagation traces.

Viral marketing is offered by H as a service to P [5, 6]: this might be in the form of advertising space, paid by P to H on a pay-per-impression basis. H attempts to optimize the placement of advertisements in order to maximize influence diffusion. This optimization is provided as a service to P, with the primary goal of making the social network more attractive as a marketing platform.

However, in order to solve the influence maximization problem and to set up the viral marketing campaign, it is first needed to learn the influence strength. It is in the interest of both H and P to learn peer influence accurately so that the viral marketing campaign is more likely to be successful.

So how can H and P learn the influence strength while keeping their data secret? This is the problem that we study in this paper.

[1] http://techcrunch.com/2013/01/24/my-precious-social-graph/
[2] http://mashable.com/2012/07/27/twitter-instagram-find-friends/

10.5441/002/edbt.2014.50

Beyond the mere privacy issues, the problem that we study is also motivated by the accuracy of the influence learning task. It has been already observed [7, 8] that the propagation models have a very large number of parameters (the influence strength of each of the social links), which makes the learning task *prone to overfitting.* In order to reduce the risk of overfitting, larger logs of past propagations (i.e., more data) are preferable. Therefore, it would make perfect sense if various service providers, $P_1, \ldots, P_m$, which are interested in doing viral marketing with H, would put their data together (in a secure way) in order to arrive to more accurate estimates of the social influences.

When we consider the case of multiple service providers conjoining their propagations, a distinction between two cases is in order. In the first case, each service provider sells different items: this means that the propagation of an item is completely contained in one provider's data. We refer to this setting as the *exclusive* case. In the second, the so-called *non-exclusive* case, the same item may be offered by different providers. This means that the trace of a propagation of an item might be partitioned between datasets of different parties. Consider, for instance, a case where user v was influenced by her friend u in H's social network to buy a specific book. However, while u bought the book from the book store P_1, v got hers from a different store P_2. If the two stores would not share their sales data, none of them would have any evidence of u influencing v on buying that book. In order to obtain a complete propagation trace, it is necessary for such service providers to conjoin their data. This means that *having more parties participating in the secure computation of social influence results not only in a larger corpus of data, but also in "better", more complete data.* Since those parties cannot disclose the activity logs of their customers to each other, such conjoining of data must be carried out in a private manner.

The two main contributions of this paper are:

- Secure multiparty protocols that allow a group of service providers and a social network host to jointly compute the link influence strengths, in either the exclusive or the non-exclusive case (Section 5).
- Secure multiparty protocols to compute directly a *user influence score*, for each user in the social network, in either the exclusive or the non-exclusive case (Section 6).

These, in turn, build upon the following more fundamental contributions:

- Two general-purpose secure arithmetic protocols: one for computing integer additive shares of a sum of private inputs, and another for computing the quotient of two private integers (Section 4).

We analyze the privacy of each of the proposed protocols under the standard assumption that the participating players are semi-honest, i.e., they respect the protocol, but try to learn as much as they can from their own view of the protocol on the private information held by other players.[3]

Analyzing and sharing user activity on modern social media and e-commerce platforms raises important privacy issues. Understanding such issues and devising adequate technological solutions is crucial, both from the legal point of view and that of social acceptance of techniques such as viral marketing. Without this, the real-world applicability of such practice is in jeopardy. To the best of our knowledge, we are the first to tackle these important privacy issues and to propose concrete solutions.

[3]See [9, 10] for a discussion and justification of that assumption.

2. BACKGROUND AND RELATED WORK

Learning link influence strength. As stated in the introduction, a basic computational problem in the area of viral marketing is that of selecting the set of users to be targeted by the campaign [11, 12], which was formalized by Kempe *et al.* [1] as a discrete optimization problem, named influence maximization : given a social network where each link is associated with an estimate of influence strength, and a budget k, find k nodes that provide the maximum *expected spread*, i.e., the expected number of active nodes at the end of the process. Part of the contribution of Kempe *et al.* [1] was to provide a simple (but computationally prohibitive) greedy algorithm with approximation guarantees. Following that seminal work, considerable effort was devoted to developing methods for improving the efficiency of influence maximization. Most of those studies assume a weighted social graph as input and do not consider the problem of computing the link influence strength, e.g. [13, 14].

Saito *et al.* [2] were the first to study how to learn the link influence probabilities from a set of past propagations. They neatly formalized the likelihood maximization problem and then applied Expectation Maximization (EM) to solve it. Although elegant, their formulation has some limitations when it comes to practice. First, real data needs to be heavily discretized to meet the assumed input format [15]. Second, the EM-based method is particularly prone to overfitting [7]. Finally, it is not very scalable as it needs to update the influence probability associated to each arc in each iteration.

For these reasons and for the sake of simplicity, we avoid the complexity of the EM-based approach when learning the influence probabilities for the influence maximization problem, and instead we follow a simpler definition by Goyal *et al.* [4], which also considered temporal decay of influence. We provide the details of this definition in Section 3.1.

Learning user influence score. The whole learning framework underlying the influence maximization problem (i.e., first learn an influence probability for each link, then apply an algorithm to discover influential users) might be considered cumbersome and inefficient. An alternative might be to just mine the social graph together with the log of available propagations in order to assign a score of influence to each user. Goyal *et al.* [16, 17] defined a notion of leadership based on how frequently a user exhibited an influential behavior. In particular, the size of the propagation sub-graph "below" a certain user u was used as a measure of u's influence. The same measure was used by Bakshy *et al.* [18] in the context of Twitter. We consider also this measure of influence (formally defined in Section 3.2).

Privacy preserving data mining has mainly been studied under two distinct settings. In the first setting, the data owner and the data miner are two different entities, and the goal is to protect the data records from the data miner. The main approach in this context is to apply data perturbation [19, 20]. The idea is that the perturbed data can be used to infer general trends in the data, while revealing (almost) no information on the original records.

In the second setting (which is the one considered in this paper), the data is distributed among several parties who aim to jointly perform data mining on the unified corpus of data that they hold, while protecting the data records of each of the data owners from the other data owners. This is a problem of *secure multi-party computation* (SMC). In the general setting of SMC, there are several parties (or players), $P_1, \ldots, P_n$, where each party P_i holds a private value x_i. The goal is to compute the value $f(x_1, \ldots, x_n)$, where f is some publicly known function of n variables, so that each party does not learn anything about the private inputs of the other parties, except the information that is implied by his own input and the output result $f(x_1, \ldots, x_n)$. The problem of secure two-party computation was solved by Yao [21] for any function f that can be represented by a binary or an algebraic circuit. While generic protocols, such as Yao's and its extensions to any number of players, apply in theory to a wide class of functions, their applicability in practice is limited to functions that have a compact representation as a circuit, due to their high computational and communication complexities. The aim of further studies in this field is to find more efficient solutions for specific problems of SMC: e.g., decision trees [22], clustering [23], and association rule mining [24, 25, 26]. It should be noted that some relaxations of the notion of perfect privacy are usually inevitable when looking for practical solutions, provided that the excess information is deemed benign (see examples of such protocols in e.g. [9, 24, 25, 27]). For each of the protocols that we present in this paper, we bound the excess information that it may leak to the interacting players and explain why such leakage of information is benign, or how it may be reduced.

3. PROBLEM DEFINITION

In this section we define the data model, the computation of influence strength for each social link (following [4]) and the node influence score (inspired by [16, 18]).

We consider a social network, which is a graph $G = (V, E)$ where the nodes are users and the links denote some relation between the users. When the relation is symmetric (such as the "friendship" relation in Facebook) the graph is undirected; when the relation is asymmetric (such as the "following" relation in Twitter) the graph is directed. We shall assume hereinafter that the graph is directed[4] and a link $(u, v) \in E$ indicates the fact that v is a follower of u, i.e., v is notified about u's activities, or in other terms, u can influence v.

We are also given a relation $\mathbb{L}(User; Action; Time)$, that we call *action log*. Each record in that log is a tuple of the form (v, α, t) which indicates the fact that user v performed action α at time t. We assume that: (a) the projection of $\mathbb{L}$ on the first column is contained in the set V of nodes of the social graph $G = (V, E)$;[5] (b) the set of all possible actions is denoted $\mathcal{A}$; and (c) time is represented by positive integers. Moreover, we assume that any given user performs any given action at most once. (Hence, if a user bought, for example, the same book twice, we consider only the first purchase.)

There are m service providers, $P_1, \ldots, P_m$, and each P_k, $1 \leq k \leq m$, owns an action log $\mathbb{L}_k$ that consists of the actions performed at the site of that service provider. The unified log is $\mathbb{L} = \bigcup_{k=1}^{m} \mathbb{L}_k$. These m service providers, together with the host, wish to compute the link influence strength (as defined in Section 3.1) and the user influence score (as defined in Section 3.2) in a secure manner.

[4]Undirected graphs will be thought of as directed graph where each undirected edge $\{u, v\}$ is replaced by the two directed arcs (u, v) and (v, u).

[5]In practice, the service providers may have users that are not members of the social network. Such users may be ignored since questions of influence are relevant only in a social framework.

3.1 Link influence strength

Assume that $V = \{v_1, \ldots, v_n\}$. Here we define the influence probabilities $p_{i,j}$ for all $\{(i, j) : (v_i, v_j) \in E\}$. The value $p_{i,j}$ is an estimate to the probability that user v_j will perform some action if v_i performs that action. The estimate is based on the past activity of those users as reported in the action logs. To define $p_{i,j}$, we introduce the following notations:

- a_i is the number of tuples in $\mathbb{L}$ in which the first component is v_i. It equals the number of actions that v_i did.
- $b^h_{i,j}$ is the number of actions α such that $\mathbb{L}$ includes a record (v_i, α, t) as well as a record (v_j, α, t'), where $t < t' \leq t + h$, for some integer $h \geq 1$. It represents the number of times in which v_j followed v_i in doing some action, assuming a memory window of width h.
- $c^h_{i,j}$ is the number of actions α such that $\mathbb{L}$ includes a record (v_i, α, t) as well as a record $(v_j, \alpha, t + h)$. It is the number of times in which v_j followed v_i in doing some action exactly h time steps after v_i performed that action.

One way of defining $p_{i,j}$ is by picking some value of h and then set

$$p_{i,j} = \frac{b^h_{i,j}}{a_i} \,. \tag{1}$$

Namely, it is the fraction of times in which v_i succeeded in influencing v_j (to follow him within h time steps). A more generalized definition would be

$$p_{i,j} = \frac{\sum_{\ell=1}^{h} w_\ell c^\ell_{i,j}}{a_i} \,, \tag{2}$$

where $0 < w_\ell$ and $\sum_{\ell=1}^{h} w_\ell = h$. The definition in equation (1) is a special case of the definition in equation (2) when $w_\ell = 1$ for all $1 \leq \ell \leq h$. By taking $w_1 > \cdots > w_\ell$, one may achieve a temporal decay effect; namely, the faster v_j follows v_i in doing an action, the more we consider v_j's action to be a result of v_i's influence on him. In either of these definitions, $p_{i,j}$ is set arbitrarily to zero if the denominator a_i is zero (since if v_i performed no action, we have no way of determining his influence on others).

3.2 User influence score

Above we defined a way to compute the influence probability for each link, which is a needed input for the influence maximization problem [1]. When focusing on links, one only needs to count the episodes of influence, i.e., the number of actions that "travelled" along the link. Then the influence maximization framework will take care of combining this information to find out influential users.

We next define an alternative to the influence maximization framework: a score of influence for each node, obtained directly by mining the social graph together with the log of past propagation traces. In this case, we cannot just consider how a node u influences its immediate neighbors; we must take into account how u's influence transitively affects the neighbors of u's neighbors, thus propagating in the network. This is captured in the next definitions.

Definition 3.1. *Given an action $\alpha \in \mathcal{A}$, its propagation graph $PG(\alpha) = (V, E(\alpha))$ is a labeled graph where there is an arc $(v_i, v_j) \in E(\alpha)$, with label Δt, if $(v_i, v_j) \in E$ and $(v_i, \alpha, t_i), (v_j, \alpha, t_j) \in \mathbb{L}$ with $\Delta t := t_j - t_i > 0$.*

Definition 3.2 (User influence sphere). *Given an action $\alpha \in \mathcal{A}$ and a maximum propagation time threshold τ, the τ-influence sphere of a user $v_i \in V$, denoted $Inf_\tau(v_i, \alpha)$, is the set of nodes reachable from v_i in $PG(\alpha)$ by a path whose sum of labels is at most τ.*

Finally, we are ready to define the user influence scores.

Definition 3.3 (User influence score). *Given a social graph $G = (V, E)$, an action log $\mathbb{L}$, and a time threshold τ, the τ-influence score of a node $v_i \in V$ is the average size of all its τ-influence spheres,*

$$score(v_i) = \frac{\sum_{\alpha \in \mathcal{A}} |Inf_\tau(v_i, \alpha)|}{a_i}, \qquad (3)$$

where a_i is the number of actions performed by v_i. (If $a_i = 0$, then $score(v_i)$ is set to zero.)

In Sec. 5 we describe secure distributed protocols for computing the link influence strength as defined in Sec. 3.1, while in Sec. 6, we describe secure distributed protocols for computing user influence scores as defined above.

4. SECURE ARITHMETIC PROTOCOLS

In this section we present two general-purpose secure arithmetic protocols that will be used in the subsequent sections: a secure protocol for computing *integer* additive shares of a sum of private inputs (Section 4.1) and a secure protocol for computing the quotient of two private integers (Section 4.2).

4.1 Secure computation of integer additive shares of a sum of private integers

Let $P_1, \ldots, P_m$ be m players, each holding a private integer $x_k \in [0, A]$, such that $x := \sum_{k=1}^m x_k \le A$. Protocol 2 enables them to compute *integer* additive shares of x; specifically, at the end of the protocol, player P_1 holds a random integer $s_1 \in [0, S-1]$, where S is some large integer that is set upfront, and P_2 holds the integer $s_2 = x - s_1$.

First, we consider a simpler problem where, for the same setting as described above, P_1 and P_2 wish to compute two *modular* additive shares of x. Protocol 1, due to Benaloh [28], is a well-known perfectly secure protocol that solves that problem.

Protocol 1 - Secure computation of *modular* additive shares of a sum of private inputs.

```
Input: $A, S \in \mathbb{Z}^+$ such that $S \gg A$.
    An integer $x_k \in [0, A]$ for each player $P_k$, $1 \le k \le m$, such that $x := \sum_{k=1}^m x_k \le A$.
Output: $P_1$ gets a random $s_1 \in [0, S-1]$;
    $P_2$ gets $s_2 \in [0, S-1]$ such that $s_1 + s_2 = x \bmod S$.
1: Each player $P_k$, $1 \le k \le m$, selects $m$ random values $x_{k,j} \in \mathbb{Z}_S$, $1 \le j \le m$, such that $\sum_{j=1}^m x_{k,j} = x_k \bmod S$.
2: $P_k$ sends $x_{k,j}$ to $P_j$, for all $1 \le k \ne j \le m$.
3: $P_j$ computes $s_j = \sum_{k=1}^m x_{k,j} \bmod S$, for all $1 \le j \le m$.
4: Players $P_3, \ldots, P_m$ send $s_3, \ldots, s_m$ to $P_2$.
5: $P_2$ updates $s_2 \leftarrow s_2 + s_3 + \cdots + s_m \bmod S$.
```

It is easy to see that the two shares s_1 and s_2 satisfy $s_1 + s_2 = x \bmod S$. Moreover, since each player P_k breaks his input x_k into a sum of m random shares, the value of s_1 (as well as that of s_2) can be any of the elements in $[0, S-1]$ with equal probabilities. Since player P_k is exposed only to one additive share in the input x_ℓ of P_ℓ, he learns nothing on that input, $1 \le k \ne \ell \le m$. Finally, since every player P_k, $k \ne 2$, receives just one additive share in x, while P_2 receives only $m-1$ of the additive shares of x $(s_2, s_3, \ldots, s_m)$, no player learns any information on x (beyond what is implied by his own input).

Now, we wish to turn Protocol 1 into a protocol that computes *integer* additive shares of x. Protocol 1 computes modular shares s_1 and s_2 such that $s_1 + s_2 = x \bmod S$. Hence, if we view those shares as integers from $[0, S-1]$, then either $s_1 + s_2 = x$ or $s_1 + s_2 = S + x$. Therefore, all we need to do is to decide whether

$$s_1 + s_2 < S. \qquad (4)$$

If inequality (4) holds, then s_1 and s_2 are already integer additive shares of x; otherwise, s_1 and $s_2 - S$ are integer additive shares of x. Hence, Protocol 1 could be extended by verifying inequality (4) and, consequently, replacing s_2 with $s_2 - S$ if needed. To that end, P_1 and P_2 may engage in a secure multiparty protocol for verifying (4). This is an instance of the millionaires' problem, that was introduced by Yao [21]. Alas, all known protocols that solve this basic problem invoke costly sub-protocols for oblivious transfer [29]. In Protocol 2 we suggest a much simpler solution which depends on the existence of a curious-but-honest third party. That third party could be P_3 (if $m > 2$) or H (the host).

Protocol 2 - Secure computation of *integer* additive shares of a sum of private inputs.

```
Input: $A, S \in \mathbb{Z}^+$ such that $S \gg A$.
    An integer $x_k \in [0, A]$ for each player $P_k$, $1 \le k \le m$, such that $x := \sum_{k=1}^m x_k \le A$.
Output: $P_1$ gets a random $s_1 \in [0, S-1]$;
    $P_2$ gets $s_2 = x - s_1$.
1: The players perform Protocol 1, after which $P_i$ holds $s_i$, $i = 1, 2$, such that $s_1 + s_2 = x \bmod S$.
2: $P_2$ generates uniformly at random an integer $r \in [0, S - A - 1]$.
3: $P_1$ sends $s_1$ to $P_3$
4: $P_2$ sends $s_2 + r$ to $P_3$.
5: $P_3$ computes $y = s_1 + s_2 + r$.
6: $P_3$ informs $P_2$ whether $y \ge S$ or not.
7: if $y \ge S$ then
8:     $P_2$ updates $s_2 \leftarrow s_2 - S$.
9: end if
```

Protocol 2 is based on the idea that in order to determine which equality holds, $s_1 + s_2 = x$ or $s_1 + s_2 = S + x$, it suffices to add to $s_1 + s_2$ an integer $r \in [0, S - A - 1]$ and then check whether the resulting sum (y in Step 5) is greater than or equal to S. If $s_1 + s_2 = x$ (in which case s_1 and s_2 are already integer additive shares as needed) then $s_1 + s_2 \le A$; in that case $y = s_1 + s_2 + r \le A + (S - A - 1) = S - 1$, and, consequently, P_2 leaves s_2 as is. If, however, $s_1 + s_2 = S + x$, then $y \ge S$; in that case, P_2 replaces s_2 with $s_2 - S$, so that s_1 and s_2 become integer additive shares of x (Steps 7-8).

Protocol 2 is "almost" perfectly secure in the following sense.

Theorem 4.1. *At the end of Protocol 2 none of the players learns anything about the inputs of the other players. As for the sum x:*

(a) Player P_2 may learn either a lower bound on x, in probability x/S, or an upper bound, in probability $(A-x)/S$, or nothing at all, in probability $(S-A)/S$.

(b) Player P_3 may learn either a lower bound on x, in probability at most $A/(S-A)$, or an upper bound, in probability at most $A/(S-A)$, or nothing at all, in probability at least $(S-3A)/(S-A)$.

(c) All other players learn nothing about x.

PROOF. The proof follows the real vs. ideal paradigm. In this paradigm, we consider the real-world model, in which the protocol is executed, versus an ideal model, involving a trusted third party who executes the computational task. If the view of any real-world adversary can be simulated by an ideal-world adversary, the protocol is perfectly secure.

Consider first any player P_k where $k \neq 2, 3$. The view of such a player in Protocol 2 is exactly like his view in Protocol 1. As the view of each of the players in Protocol 1 consists of shares that distribute uniformly at random over $\mathbb{Z}_S$, it can be simulated by an ideal-world simulator. Hence, it suffices to focus on P_2 and P_3 who are the only players whose view in Protocol 2 differs from that in Protocol 1. P_3's view in Protocol 2 contains (in addition to his view in Protocol 1) the value of $y = s_1 + s_2 + r$ (Step 5). P_2's view in Protocol 2 contains the answer to whether $y \geq S$ or not (Step 6). Since those two pieces of information cannot be simulated, we need to analyze what can be inferred from them.

We start with P_3, who learns the value of y. If $y < S$ he infers that $y = x + r$; otherwise he infers that $y - S = x + r$. In any case, he learns the value $x + r := z$. Since $x = z - r$ and $0 \leq r \leq S - A - 1$, it follows that

$$z - (S - A - 1) \leq x \leq z \,. \qquad (5)$$

Since it is known upfront that $0 \leq x \leq A$, the upper bound in (5) reveals new information on x only when $z < A$. For every $i \in \{0, 1, \ldots, A-1\}$, $\Pr(z = i) = \sum_{j=0}^{i} \Pr(x = j) \cdot \Pr(r = i - j)$. Since r distributes uniformly on $[0, S-A-1]$, it follows that $\Pr(z = i) = \frac{1}{S-A} \sum_{j=0}^{i} \Pr(x = j) \leq \frac{1}{S-A}$. Hence, $\Pr(z < A) = \sum_{i=0}^{A-1} \Pr(z = i) \leq \frac{A}{S-A}$. Therefore, P_3 may learn a non-trivial upper bound on x in probability no greater than $A/(S-A)$. Similarly, we can show that he may learn a non-trivial lower bound in probability no greater than $A/(S-A)$. The last two probability inequalities imply that in probability at least $(S-3A)/(S-A)$, the value of z does not allow P_3 to exclude any possible value of x in the range $[0, A]$. In addition, the value of z does not induce an a-posteriori belief probability distribution on x that differs from the belief probability distribution that P_3 had prior to seeing z, owing to the uniform random manner in which the masking value r is chosen.

We now turn to P_2. If he learns that $s_1 + s_2 < S$, then P_2 also learns that $x = s_1 + s_2$ in $\mathbb{Z}$, whence $0 \leq s_1, s_2 \leq x$. Since s_2 is a sum of random $\mathbb{Z}_S$-shares of x_k, $1 \leq k \leq m$, it is uniformly distributed over $\mathbb{Z}_S$. Since the case $s_1 + s_2 < S$ occurs only if $s_2 \in \{0, 1, \ldots, x\}$, its probability is $(x+1)/S$. In that case, P_2 may infer that $x \geq s_2$. That lower bound is non-trivial only when $s_2 > 0$. Hence, P_2 learns a (non-trivial) lower bound on x in probability x/S.

If P_2 learns that $s_1 + s_2 \geq S$, it can happen only if both s_1 and s_2 are strictly greater than x (since if, say, $s_1 \leq x$, then $s_2 = x - s_1 \leq x$). Hence, in that case $x \leq s_2 - 1$. Only when $s_2 \leq A$, the upper bound is non-trivial. Therefore, P_2 learns a (non-trivial) upper bound on x in probability $(A-x)/S$. When $s_2 > A$, P_2 can learn nothing on x.

Hence, P_2 learns nothing on x when $s_2 = 0$ or when $s_2 > A$. Since s_2 distributes uniformly in $\mathbb{Z}_S$, P_2 learns no information at all in probability $(S-A)/S$. □

To summarize, the only potential leakages of information are to only two players. Those potential leakages of information are only with respect to x (but not with respect to the private inputs of other players), and only in the form of a lower or an upper bound. Moreover, the probability of those potential leakages to occur can be made negligible since A is a given integer and S can be chosen arbitrarily large.

Later on, in Section 5, we need to compute additive shares in a large number of counter values (a_i and $b_{i,j}^h$ that were introduced earlier in Section 3.1). Instead of invoking Protocol 2 separately for each of those computations, we perform them in parallel. That enables us to reduce the communication costs, as well as to improve the privacy of the protocol. Specifically, the parallel execution of computing the additive shares for all of those counters essentially removes the potential leakage of information to P_3 that was described in Theorem 4.1. (See more details in Section 5.)

4.2 Secure division of private integers

The secure computation problem that we address here involves three players: P_1, P_2, and H. Each of the first two players holds an integer a_i that is taken from a range $[0, A]$. They wish to let H compute the *real* quotient $q := a_1/a_2$ (where q is interpreted as zero if $a_2 = 0$), without disclosing the values of a_1 and a_2 to H, beyond what is implied by q. (Note that we are interested here in computing a_1/a_2 as the real quotient of two integers. This problem differs from the problem of computing $a_1 \cdot a_2^{-1}$ where a_1 and a_2 are viewed as elements in a finite field; the latter problem may be solved, e.g., by means of homomorphic encryption.)

Protocol 3 - Secure computation of the quotient of private integers.

Input: P_i has an integer $a_i \in [0, A]$, $i = 1, 2$.
Output: H gets $q := a_1/a_2$ if $a_2 > 0$ and $q = 0$ if $a_2 = 0$.

```
1: P_1 and P_2 jointly generate a random real number $M \sim Z$, where $Z$ is the distribution on $[1, \infty)$ with pdf (probability density function) $f_Z(\mu) = \mu^{-2}$.
2: P_1 and P_2 jointly generate a random real $r \sim U(0, M)$.
3: P_1 sends $ra_1$ to H.
4: P_2 sends $ra_2$ to H.
5: if $ra_2 \neq 0$ then
6:     H computes $q = ra_1/ra_2$.
7: else
8:     H computes $q = 0$.
9: end if
```

Protocol 3 solves that problem. In order to discuss its privacy, we first prove the next theorem.

THEOREM 4.2. *Let X be a random variable that takes values in $[0, A]$, and let $f_X(x)$ denote its pdf. Let R be a uniform random variable on $(0, \mu)$, for some real number $\mu > 0$, and let $Y = XR$. Then for all $x > 0$ and $y > 0$, $f_X(x|Y = y) = G_\mu(x, y)$ where*

$$G_\mu(x, y) := \begin{cases} 0 & y > \mu x \\ \frac{f_X(x)}{x \cdot \alpha(y,\mu)} & y \leq \mu x \end{cases} , \qquad (6)$$

and $\alpha(y, \mu) := \int_{y/\mu}^{A} \frac{f_X(t)}{t} dt$.

PROOF. The definition of Y as the product of X and R, where R is a uniform random variable on $(0, \mu)$, implies that $f_Y(y|X = x) = 0$ if $y > \mu x$, while $f_Y(y|X = x) = \frac{1}{\mu x}$ otherwise (f_Y being the pdf of Y). Therefore, $f_Y(y) = \int_0^A f_Y(y|X = t) f_X(t) dt = \int_{y/\mu}^A \frac{f_X(t)}{\mu t} dt$. Our claim now follows from Bayes Theorem, by which $f_X(x|Y = y) = f_Y(y|X = x) \cdot \frac{f_X(x)}{f_Y(y)}$, $\square$

Let us fix $1 \leq i \leq 2$ and let X denote the random variable that governs the value of a_i. Assume that H has an a-priori belief probability $f_X(x)$ regarding the value of X. In Protocol 3, H learns the value $y := rx$, where $x = a_i$. That value of y enables H to extract an a-posteriori belief probability regarding the value of $x = a_i$, as follows. If H had known the value of M that was chosen as an upper bound for r, then given $y = rx$, he could be able to infer, by Theorem 4.2, that $f_X(x|Y = y) = G_M(x, y)$. In particular, since $G_M(x, y)$ is nonzero only when $x \geq y/M$, he would be able to infer that $x \geq y/M$. That is a non-trivial lower bound since the expected value of y is $Mx/2$, and therefore the expected value of the lower bound that H can derive is $x/2$. However, P_1 and P_2 choose M randomly from the distribution Z on $[1, \infty)$. That selection of M leaves every $x > 0$ as a possible pre-image, given the value of $y = rx > 0$, as we show next.

Hereinafter, we concentrate on the case that interests us, where X is a discrete random variable that takes values in the set of integers $\{0, 1, \ldots, A\}$. In that case, $f_X(x)$ is a discrete distribution of the form $f_X(x) = \sum_{k=0}^A f_k \delta(x - k)$, where $\delta(\cdot)$ is the Dirac delta function. For such discrete distributions, any value of $y > 0$ leaves all values of x as possible pre-images, unless x is a-priori known to be an impossible value:

THEOREM 4.3. *Given $y > 0$, $f_X(x|Y = y) > 0$ if $f_X(x) > 0$, and $f_X(x|Y = y) = 0$ if $f_X(x) = 0$.*

PROOF. We distinguish between two cases: $y \leq A$ and $y > A$. In the first case, every value of $\mu \geq 1$ is possible; in the second case, the value of y reveals that μ must have been at least y/A. Therefore, the a-posteriori distribution of μ, given the value of y, remains Z in the first case, with the pdf $f_Z(\mu) = \mu^{-2}$, but in the second case it is the distribution on $[y/A, \infty)$ with the rescaled pdf $\frac{y}{A} \cdot f_Z(\mu)$. Hereinafter, $\Phi(\mu)$ is the a-posteriori pdf of μ.

By the complete probability theorem,

$$f_X(x|Y = y) = \int_1^\infty f_X(x|Y = y, M_i = \mu)\Phi(\mu)d\mu$$
$$= \int_1^\infty G_\mu(x, y) \cdot \Phi(\mu)d\mu\,.$$

Using Equation (6), we infer that

$$f_X(x|Y = y) = \frac{f_X(x)}{x} \cdot \int_{\max\{1, y/x\}}^\infty \frac{\Phi(\mu)d\mu}{\alpha(y, \mu)}\,. \qquad (7)$$

We need to show that this probability is well defined and positive for all $x \in \{1, \ldots, A\}$ with $f_X(x) > 0$. Since $f_X(x) = \sum_{k=0}^A f_k \delta(x - k)$, we have

$$\alpha(y, \mu) = \sum_{k=\lceil y/\mu \rceil}^A \frac{f_X(k)}{k}\,. \qquad (8)$$

As $y/\mu \leq y/(y/x) = x \leq A$, the above sum contains at least the term $f_X(A)/A$. Without loss of generality, we may assume that $f_X(A) > 0$ (since A may be taken as the maximal integer for which $f_X(A) > 0$). Hence, $\alpha(y, \mu) > 0$ for all $\mu > y/x$. Therefore, the integrand in Equation (7) is well defined and positive over the domain of integration. (The integral is improper, but it is finite since the denominator is bounded from below by $f_X(A)/A$ and $\int_{\max\{1, y/x\}}^\infty \Phi(\mu)d\mu < \infty$.) That completes the proof. $\square$

Perfect privacy would be achieved if $f_X(x|Y = y) = f_X(x)$, namely, if the a-posteriori belief probability regarding the value of x would be the same as the a-priori one. Our mechanism does not provide that. The value $y = 0$ reveals that $x = 0$, while any value $y > 0$ induces an a-posteriori belief probability that differs from $f_X(x)$. *Albeit not perfect, this is a significant level of privacy.*

In our context, $x = a_2$ is the number of times that a particular user did some action, while $x = a_1$ is the number of times in which one user followed his friend in doing some action. In that case, the value $x = 0$ would be considered insensitive, since, typically, the sensitive information is performing an action, not the opposite. As for $y > 0$, while it induces an a-posteriori belief probability that differs from $f_X(x)$, Theorem 4.3 implies that all values of x are "suspicious", and that offers a significant shield of privacy. The next theorem spells out explicitly the a-posteriori belief probability.

THEOREM 4.4. *Let $\psi(j) := 1/\sum_{k=j}^A f_X(k)/k$ and $\Psi(x) := \sum_{j=1}^x \psi(j)$ for every integer $1 \leq j \leq A$. Then, for every $x \in \{1, \ldots, A\}$, the following holds: if $y \leq A$,*

$$\frac{f_X(x|Y = y)}{f_X(x)} = \begin{cases} \frac{\Psi(x)}{xy} & x \leq y \\ \frac{\Psi(\lfloor y \rfloor)}{xy} + \frac{\psi(\lceil y \rceil)(1 - \lfloor y \rfloor / y)}{x} & x > y \end{cases}\,, \qquad (9)$$

while if $y > A$ then

$$\frac{f_X(x|Y = y)}{f_X(x)} = \frac{\Psi(x)}{Ax}\,. \qquad (10)$$

PROOF. By Equation (8), $\alpha(y, \mu) = \frac{1}{\psi(\lceil y/\mu \rceil)}$. We claim that this equality implies that

$$I := \int_{y/x}^\infty \frac{\mu^{-2}d\mu}{\alpha(y, \mu)} = \frac{\Psi(x)}{y}\,. \qquad (11)$$

Indeed, the interval of integration can be split into x intervals: $I = \sum_{j=1}^x \int_{y/j}^{y/(j-1)} \frac{\mu^{-2}d\mu}{\alpha(y,\mu)}$. Along the j'th integral of integration, $\lceil y/\mu \rceil = j$, whence $1/\alpha(y, \mu) = \psi(j)$ there. Therefore,

$$I = \sum_{j=1}^x \psi(j) \int_{y/j}^{y/(j-1)} \mu^{-2}d\mu = \sum_{j=1}^x \frac{\psi(j)}{y} = \frac{\Psi(x)}{y}\,.$$

The first case in Equation (9) follows from Equations (7) and (11), since in that case $\Phi(\mu) = \mu^{-2}$ and the lower limit of the integral in Equation (7) is $\max\{1, y/x\} = y/x$. The proof of Equation (10) is similar, where the only difference is that in that case (in which $y > A$), $\Phi(\mu) = \frac{y}{A}\mu^{-2}$, whence we get eventually $\frac{y}{A} \cdot \frac{\Psi(x)}{xy} = \frac{\Psi(x)}{Ax}$.

It remains to prove the second case in Equation (9) where $x > y$ and, therefore, the lower limit of the integral in Equation (7) is $\max\{1, y/x\} = 1$. In that case, we split the interval of integration as follows: $[1, \infty) = [1, y/\lfloor y \rfloor) \cup [y/\lfloor y \rfloor, \infty)$.

By Equation (11), the integral over $[y/\lfloor y\rfloor, \infty)$ equals $\frac{\Psi(\lfloor y\rfloor)}{y}$. As for the interval $[1, y/\lfloor y\rfloor)$, along it we have $\lceil y/\mu\rceil = \lceil y\rceil$. Hence, the remaining integral is

$$\int_1^{y/\lfloor y\rfloor} \frac{\mu^{-2}d\mu}{\alpha(y,\mu)} = \psi(\lceil y\rceil)\int_1^{y/\lfloor y\rfloor} \mu^{-2}d\mu = \psi(\lceil y\rceil)(1-\lfloor y\rfloor/y)\,.$$

Hence, the integral in Equation (7) equals in this case $\frac{\Psi(\lfloor y\rfloor)}{y} + \psi(\lceil y\rceil)(1-\lfloor y\rfloor/y)$. That completes the proof of the second case in Equation (9). $\Box$

Note that any value of $y > A$ generates the very same a-posteriori belief probability (given in Equation (10)), regardless of the value of x; that effect is a result of our selection of the probability distribution Z in Protocol 3.

In Section 7.2 we perform extensive experimentation that demonstrates the privacy preservation offered by Protocol 3.

5. SECURE PROTOCOLS FOR LINK INFLUENCE STRENGTH

The players H (the host) and $P_1,\ldots,P_m$ (the service providers) wish to compute jointly influence probabilities $p_{i,j}$ for all $\{(i,j) : (v_i,v_j) \in E\}$, as defined in Section 3.1. Their goal is to perform those computations while preserving their private information. The private information of P_k, $1 \le k \le m$, is his private log $\mathbb{L}_k$. The private information of H is the arc structure E. As discussed in the Introduction, we consider two cases: (1) the exclusive case, where each action is supported exclusively by one of the players, and (2) the non-exclusive case, where each action can be supported by more than one player.

5.1 The exclusive case

We first consider the simpler definition (Equation (1)) and discuss later the definition in Equation (2). As defined in Section 3.1, a_i is the number of actions that user v_i performed, while $b^h_{i,j}$ is the number of actions that v_j did following v_i. For each $1 \le k \le m$, let $a_{k,i}$ denote the number of records in $\mathbb{L}_k$ that involve v_i. Since we assumed that each user may perform any given action at most once, then $a_i = \sum_{k=1}^m a_{k,i}$. (This equality holds in both the exclusive and non-exclusive cases.) Let $b^h_{k,i,j}$ denote the number of actions α such that $\mathbb{L}_k$ includes a record (v_i,α,t) as well as a record (v_j,α,t'), where $t < t' \le t+h$. Then in the exclusive case, where each action α can appear in only one of the logs, we have $b^h_{i,j} = \sum_{k=1}^m b^h_{k,i,j}$.

Protocol 4, which involves the service providers and the host, enables the latter to compute the influence probabilities $p_{i,j}$, for all arcs $(v_i,v_j) \in E$, according to the definition in Equation (1). First, H hides his edge set within a larger set of edges, E', which he sends to $P_1,\ldots,P_m$ (Steps 1-2); the edges in $E' \setminus E$ are selected uniformly at random from the set of all pairs $(v_i,v_j) \notin E$, where $v_i \ne v_j$.

Then, the service providers perform Protocol 2 for a_i and $b^h_{i,j}$ for all pairs of nodes that appear in the augmented edge set E' (Steps 3-4). Note that Protocol 2 is designed to compute integer additive shares for a single sum. Here, we perform that protocol in parallel for $n + |\Omega_{E'}|$ sums (all a_i and $b^h_{i,j}$ counters). Several comments are in order regarding the implementation of Steps 3 and 4:

- Letting $A = |\mathcal{A}|$ be the total number of possible actions, then all counters are integers between 0 and A; the value of $S \gg A$ that is used in Protocol 2 is chosen jointly by the service providers.

Protocol 4 - Secure computation of link influence probability.

Input: H owns the social graph, $G = (V,E)$, where $V = \{v_1,\ldots,v_n\}$. P_k, $1 \le k \le m$, owns an action log $\mathbb{L}_k$.
Output: H gets $p_{i,j}$ for all $(i,j) \in E$.

```
1: $H$ generates a set $E' \subset V \times V$ such that $E' \supset E$ and $|E'| \ge c|E|$ for some given constant $c > 1$.
2: $H$ sends $\Omega_{E'} := \{(i,j) : (v_i,v_j) \in E'\}$ to $P_k$, $1 \le k \le m$.
3: $P_1,\ldots,P_m$ perform Protocol 2 *in parallel* for $a_i = \sum_{k=1}^m a_{k,i}$, for all $1 \le i \le n$. At the end of that protocol, $P_1$ has $s_1^i$ and $P_2$ has $s_2^i$ such that $s_1^i + s_2^i = a_i$.
4: $P_1,\ldots,P_m$ perform Protocol 2 *in parallel* for $b^h_{i,j} = \sum_{k=1}^m b^h_{k,i,j}$, for all $(i,j) \in \Omega_{E'}$. At the end of that protocol, $P_1$ has $s_1^{i,j}$ and $P_2$ has $s_2^{i,j}$ such that $s_1^{i,j} + s_2^{i,j} = b^h_{i,j}$.
5: For each $1 \le i \le n$, $P_1$ and $P_2$ jointly generate independent random real numbers $M_i \sim Z$, where $Z$ is as in Protocol 3.
6: For each $1 \le i \le n$, $P_1$ and $P_2$ jointly generate independent random real numbers $r_i \sim U(0, M_i)$.
7: $P_1$ sends $r_i s_1^i$ for all $1 \le i \le n$ and $r_i s_1^{i,j}$ for all $(i,j) \in \Omega_{E'}$ to $H$.
8: $P_2$ sends $r_i s_2^i$ for all $1 \le i \le n$ and $r_i s_2^{i,j}$ for all $(i,j) \in \Omega_{E'}$ to $H$.
9: For each $(i,j)$ such that $(v_i,v_j) \in E$, $H$ computes $p_{i,j} = (r_i s_1^{i,j} + r_i s_2^{i,j})/(r_i s_1^i + r_i s_2^i)$ (where the quotient is set to zero if the denominator is).
```

- Steps 3 and 4 appear in Protocol 4 as two separated steps just for the sake of clarity; in implementing the protocol we propose to handle all $n + |\Omega_{E'}|$ counters in parallel.
- In Steps 3-4 of the parallelized Protocol 2, P_1 and P_2 send to P_3 the values s_1 and $s_2 + r$ that correspond to each of the $n + |\Omega_{E'}|$ counters. In doing so, P_1 and P_2 chose a random permutation of the $n + |\Omega_{E'}|$ counters which they keep secret from P_3; then, P_1 uses that permutation to permute the sequence of $n + |\Omega_{E'}|$ values of s_1 that he sends to P_3, and P_2 does the same for his sequence of $s_2 + r$ values. This way, even though some of the $(s_1, s_2 + r)$ pairs may be used by P_3 to infer lower or upper bounds on the corresponding counters (as shown in Theorem 4.1), he will not be able to tell which of the counters those pairs correspond to. Hence, by using such a secret permutation, the potential information leakage to P_3 becomes useless.

At this stage, P_1 and P_2 hold random additive shares in a_i (the number of actions that v_i performed in total) and $b^h_{i,j}$ (the number of times where v_j followed v_i) for all users $1 \le i \le n$ and all edges in E'. They now wish to let H compute the quotient $b^h_{i,j}/a_i$, being a measure of the influence that v_i has on v_j, without leaking to him information on a_i and $b^h_{i,j}$ beyond what is implied by the quotient. To that end, P_1, P_2 and H engage in a variant of Protocol 3, where the multipliers r_i are chosen independently for each user v_i (Steps 5-9). (It is a variant of Protocol 3 since here the masking random value r_i multiplies the shares of the numerator $b^h_{i,j}$ and the denominator a_i, rather than directly $b^h_{i,j}$ and a_i).

It is clear that Protocol 4 is correct and complete in the sense that it ends with H having $p_{i,j} = b^h_{i,j}/a_i$ for all arcs in

E. (Indeed, $(r_i s_1^{i,j} + r_i s_2^{i,j})/(r_i s_1^i + r_i s_2^i) = (s_1^{i,j} + s_2^{i,j})/(s_1^i + s_2^i) = b_{i,j}^h / a_i$.)

A protocol for computing the link influences by Equation (2) goes along the same lines, since also in that definition the numerator is a sum of private values which each P_k has, $1 \leq k \leq m$. Hence, the only modification in Protocol 4 is in Step 4, where the players invoke Protocol 2 to compute additive shares in $\sum_{\ell=1}^{h} w_\ell c_{i,j}^\ell$; all other steps remain the same.

5.1.1 The privacy offered by Protocol 4

First, we discuss the privacy that the protocol provides for the host H; then, we discuss the privacy that it provides for the service providers $P_1, ..., P_m$.

H aims at protecting the arc structure. Protocol 4 does not provide full protection for that information since $P_1, \ldots, P_m$ know that E is a subset of E'. The arc structure could be fully protected by selecting E' to be the set of all pairs of nodes. However, such a selection will impose large computation and communication costs. In general, by suitably selecting the multiplying factor c, one can directly handle the privacy-efficiency trade-off. Another way to achieve a perfect hiding of E is by resorting to oblivious transfer protocols. Specifically, the players could perform Steps 3-4 in Protocol 4 for all $n^2 - n$ pairs $1 \leq i \neq j \leq n$. Then, H could execute a $|E|$-out-of-$(n^2 - n)$ oblivious transfer protocol (e.g. [30]) vis-a-vis P_1 to retrieve, obliviously, the values of $s_1^{i,j}$ for all (i,j) such that $(v_i, v_j) \in E$, and similarly with P_2 for $s_2^{i,j}$. Alas, such a solution, albeit more secure, is extremely prohibitive since it entails $O(|E|n^2)$ modular exponentiations, and, in addition, it requires $P_1, \ldots, P_m$ to perform Protocol 2 for all $n^2 - n$ pairs (i,j), $1 \leq i \neq j \leq n$.

P_k, $1 \leq k \leq m$, wish to protect their private inputs $a_{k,i}$ and $b_{k,i,j}^h$, as well as the aggregate counters a_i and $b_{i,j}^h$. During Protocol 4, $P_1, \ldots, P_m$ execute *independent* runs of Protocol 2 for each of those counters (Steps 3-4). Namely, the random shares that are generated by the players in each run of Protocol 2 are independent; consequently, views from different runs cannot be combined to infer more information than what could be extracted from the separate views. The analysis in Section 4.1 shows that none of those players learns any information on the private counters of his peers or on the aggregate counters, apart from P_2 and P_3 who may infer a lower or an upper bound on some (very few) of the aggregate counters. However, the probability of the latter event can be made as small as desired by increasing S. Specifically, if we wish to restrict the probability of either P_2 or P_3 learning any information on any of the $n + |\Omega_{E'}|$ aggregate counters to be no larger than ε, it is implied by Theorem 4.1 that all we need to do is to select $S \geq A \cdot (1 + 2(n + |\Omega_{E'}|)/\varepsilon)$.

Finally, we consider the possible inferences that H can extract on any of the counters $c \in \{a_i\}_{1 \leq i \leq n} \cup \{b_{i,j}^h\}_{(i,j) \in E'}$. For each such c, H obtains in the course of Protocol 4 a single value of the form rc for some random real number r. The analysis in Section 4.2 shows that H can only obtain a-posteriori belief probabilities regarding those counters which may differ from the corresponding priors. As those posteriors are spread quite evenly on the entire range of possible positive values of those counters, the utilized mechanism effectively protects that information also from H.

We note that if H has a prior belief about a relation between, say, a_1 and a_2 (e.g., that $a_1 = \alpha a_2$ for some α) he will get two a-posteriori belief probabilities for those counters. We omit a discussion of such models due to page limitation.

5.2 The non-exclusive case

So far we dealt with a distributed setting in which the service providers had exclusivity over the actions. Namely, each action $\alpha \in \mathcal{A}$ was supported by exactly one of the service provides P_k, $1 \leq k \leq m$. Such an exclusivity assumption does not always hold. For example, an action could be buying the book "Fifty Shades of Grey"; if $P_1, \ldots, P_d$ are all bookstores, then records with that action may appear in each of the action logs $\mathbb{L}_1, \ldots, \mathbb{L}_d$. In such a non-exclusive setting, a preprocessing stage has to be done before the solution that we described above can be executed.

Assume that the set of all possible actions, $\mathcal{A}$, is partitioned into disjoint classes, $\mathcal{A} = \bigcup_{q=1}^{Q} \mathcal{A}_q$, where each $\mathcal{A}_q$ is a class of a different type of actions: buying books, seeing movies, signing petitions, joining social groups, voting, etc. For each class of actions $\mathcal{A}_q$ there exists a subset of service providers $\mathcal{P}_q \subseteq \{P_1, \ldots, P_m\}$ that support actions from $\mathcal{A}_q$. The meaning of that is that any action $\alpha \in \mathcal{A}_q$ may appear only in action logs of players from $\mathcal{P}_q$. However, the action logs of players from $\mathcal{P}_q$ may contain also actions not in $\mathcal{A}_q$; namely, the subsets $\mathcal{P}_q$, $1 \leq q \leq Q$, are not necessarily disjoint. The classes $\mathcal{A}_q$ and the corresponding groups $\mathcal{P}_q$, $1 \leq q \leq Q$, are known to all. In Protocol 5 we describe the preprocessing procedure for one of the action classes; it has to be performed for each of the Q classes. After executing that protocol for, say, action class $\mathcal{A}_1$, one of the players in $\mathcal{P}_1$ gets the corresponding aggregated counters, $a_i[\mathcal{A}_1]$ (the total number of actions from $\mathcal{A}_1$ that were performed by user v_i through any of the service providers in $\mathcal{P}_1$) and $b_{i,j}^h[\mathcal{A}_1]$ (the total number of $\mathcal{A}_1$-actions in which v_j followed v_i, assuming a memory window of width h), $1 \leq i \neq j \leq n$. From that point on, all players in $\mathcal{P}_1$ remove from their action logs all records that correspond to $\mathcal{A}_1$-actions, since all of those actions are already recorded in the counters $a_i[\mathcal{A}_1]$ and $b_{i,j}^h[\mathcal{A}_1]$ that the representative player holds. After performing Protocol 5 for all $1 \leq q \leq Q$, the players may proceed to apply Protocol 4 that was described in Section 5.1 for the exclusive setting.

Without loss of generality, we assume in Protocol 5 that the class action is $\mathcal{A}_1$ and that $\mathcal{P}_1 = \{P_1, \ldots, P_d\}$. In performing this procedure, the players in $\mathcal{P}_1$ are assisted by a trusted third party $\hat{P}$ whom they select from among the other players, $\{H, P_{d+1}, \ldots, P_m\}$. They also select one of them, say P_1, as the player that receives at the end of the protocol the aggregated counters for all $\mathcal{A}_1$-actions and uses them, on behalf of all of them, in Protocol 4.

Protocol 5 is described as a protocol template. In Step 1, each player in $\mathcal{P}_1$ constructs $\mathbb{L}_{k,\mathcal{A}_1}$ as the subset of $\mathbb{L}_k$ that consists of all $\mathcal{A}_1$-actions, and then removes that subset from $\mathbb{L}_k$. Each player in $\mathcal{P}_1$ then transforms the proprietary $\mathcal{A}_1$-action log, $\mathbb{L}_{k,\mathcal{A}_1}$, into an obfuscated form, denoted $\hat{\mathbb{L}}_{k,\mathcal{A}_1}$, and sends it to the trusted third party $\hat{P}$ (Step 2). $\hat{P}$ unifies all obfuscated logs, and computes the relevant counters (Steps 3-4). Finally, $\hat{P}$ sends those counters to P_1 who recovers the correct counter values from them (Steps 5-6). Next, we discuss possible methods for obfuscating the log entries (Step 2).

A basic method for obfuscating the action log. The first method hides the user identities and action identifiers, but retains the time entries. Specifically, the players in $\mathcal{P}_1$

Protocol 5 - Secure computation of the aggregated counters for a class action

Input: Player P_k, $1 \leq k \leq d$, has an action log $\mathbb{L}_k$.

Output: P_1 gets the counters $a_i[\mathcal{A}_1]$ and $b^h_{i,j}[\mathcal{A}_1]$ for all $1 \leq i \neq j \leq n$.

```
1: Each $P_k$, $1 \leq k \leq d$, defines $\mathbb{L}_{k,\mathcal{A}_1} = \{(v_i, \alpha, t) \in \mathbb{L}_k : \alpha \in \mathcal{A}_1\}$ and then sets $\mathbb{L}_k \leftarrow \mathbb{L}_k \setminus \mathbb{L}_{k,\mathcal{A}_1}$.
2: Each $P_k$, $1 \leq k \leq d$, transforms $\mathbb{L}_{k,\mathcal{A}_1}$ to $\hat{\mathbb{L}}_{k,\mathcal{A}_1}$ and then sends it to $\hat{P}$.
3: $\hat{P}$ computes $\hat{\mathbb{L}}_{\mathcal{A}_1} = \bigcup_{k=1}^{d} \hat{L}_{k,\mathcal{A}_1}$.
4: $\hat{P}$ computes from $\hat{\mathbb{L}}_{\mathcal{A}_1}$ the relevant counters.
5: $\hat{P}$ sends the nonzero counters to $P_1$.
6: $P_1$ recovers from the retrieved counters the correct ones $a_i[\mathcal{A}_1]$ and $b^h_{i,j}[\mathcal{A}_1]$.
```

agree on secret identifying numbers for the actions in $\mathcal{A}_1$ and new user identifiers which are kept secret from $\hat{P}$. For example, they could jointly create secret permutations π on $\{1, \ldots, n\}$ and σ on $\mathcal{A}_1$ and then transform each record of the form (v_i, α, t) to $(v_{\pi(i)}, \sigma(\alpha), t)$. Since the time entry remains unchanged, $\hat{P}$ can compute all necessary counters for the permuted user identifiers and then P_1 can recover from them the counters for the original user identifiers.

Without any background knowledge about user activity, the only information that $\hat{P}$ can deduce is that some user did some action at a given time t, but since π and σ are selected uniformly at random, all users and all actions would be equally likely. If, on the other hand, $\hat{P}$ knows that a specific user v_j is significantly more active than other users, it may be possible to identify, with some probability, the permuted identifier that corresponds to v_j and then observe v_j's activity pattern in time. For example, if $\hat{P}$ can identify with certainty the permuted identifier of such a target user v_j, then $\hat{P}$ may deduce the number of $\mathcal{A}_1$-actions that v_j performed in each time stamp, but not which actions.

We note that P_1 may learn information on activity of users at other sites. For example, if user v_i performed $a^1_i[\mathcal{A}_1]$ actions of class $\mathcal{A}_1$ at P_1, then he must have performed $a_i[\mathcal{A}_1] - a^1_i[\mathcal{A}_1]$ actions of that class at sites $P_2, \ldots, P_d$; however, as $\hat{P}$ returns to P_1 aggregated counters (and not counters per action), P_1 is unable to reveal what actions from that class took place in those sites.

An enhanced method for obfuscating the action log. In order to obfuscate also the time information, we propose the following enhancement of the method above. Assume that the time frame is $0 \leq t \leq T - 1$. Let $S := T + h$. The $\mathcal{P}_1$-players select a random key $s \in \mathbb{Z}_S := \{0, 1, \ldots, T+h-1\}$ and then encrypt all time stamps using the shift cipher, $t \mapsto e_s(t) := t + s \mod S$. Hence, any record (v_i, α, t) in $\mathbb{L}_{k,\mathcal{A}_1}$, $1 \leq k \leq d$, is transformed to $(v_{\pi(i)}, \sigma(\alpha), e_s(t))$. The trusted party $\hat{P}$ can still compute all counters $a_i[\mathcal{A}_1]$ and $b^h_{i,j}[\mathcal{A}_1]$, for the permuted user identifiers, by considering the record $(v_{\pi(j)}, \sigma(\alpha), e_s(t'))$ as a follower of $(v_{\pi(i)}, \sigma(\alpha), e_s(t))$ if

$$e_s(t') \in \{e_s(t) + \tau \mod S : 1 \leq \tau \leq h\}. \qquad (12)$$

Indeed, since there are no original records (v_i, α, t) with $T \leq t \leq T+h-1$, condition (12) holds if and only if $t < t' \leq t+h$.

Alas, such an encryption of the time information does not stand on its own since $\hat{P}$ can easily detect the sequence of h encrypted time stamps $e_s(T), \ldots, e_s(T + h - 1)$ in which there is no activity and, consequently, invert the encryption and recover the original time stamps. Hence, it is necessary to add also fake activity to time stamps $T, \ldots, T + h - 1$ in order to prevent this inference of the original time stamps. One way of doing that is by adding fake users and use them to create fake activity that will disable $\hat{P}$ from inferring the true time stamps. Since such fake entries in the obfuscated logs may affect only counters $a_i[\mathcal{A}_1]$ in which i corresponds to a fake user and counters $b^h_{i,j}[\mathcal{A}_1]$ in which at least one of i, j corresponds to a fake user, P_1 can simply ignore those irrelevant counters.

To that end, the users decide on a number n' of fake users and then hide the true users by selecting a random injection $f : \{1, \ldots, n\} \to \{1, \ldots, n+n'\}$. Then, each player augments his $\mathcal{A}_1$-action log $\mathbb{L}_{k,\mathcal{A}_1}$ by fake records of the form (v_i, α, t) where v_i is a fake user.[6] Assume that for P_k, $1 \leq k \leq d$, the number of actions in $\mathbb{L}_{k,\mathcal{A}_1}$ with the time stamp t is $w_k(t)$, and let W_k be $\max_{0 \leq t \leq T-1} w_k(t)$. Then for all $0 \leq t \leq T - 1 + h$, P_k adds to $\mathbb{L}_{k,\mathcal{A}_1}$ records of the form (v_i, α, t), where v_i is one of the fake users, so that the resulting number of actions in $\mathbb{L}_{k,\mathcal{A}_1}$ with time stamp t is W_k. If all players do so, the trusted third party $\hat{P}$ will not be able to distinguish $e_s(T), \ldots, e_s(T + h - 1)$ from $e_s(0), \ldots, e_s(T - 1)$ since in each of those encrypted time stamps there will be the same number of actions. Consequently, given an encrypted time stamp $e_s(t)$, all values of $0 \leq t \leq T - 1 + h$ will be equally likely for $\hat{P}$.

6. SECURE PROTOCOLS FOR COMPUTING THE USER INFLUENCE SCORES

Here we discuss the secure computation of user influence scores according to Definition 3.3.

6.1 The exclusive case

Protocol 6, which we explain below, enables H to compute the propagation graphs $PG(\alpha)$ for all $\alpha \in \mathcal{A}$, with the collaboration of $P_1, \ldots, P_m$. After having obtained all these labeled graphs, H can proceed on his own to compute $Inf_\tau(v_i, \alpha)$ for every desired setting of τ, and for all $v_i \in V$ and $\alpha \in \mathcal{A}$. As a result, H will be able to compute the numerator in Equation (3). In order to complete the computation of the user influence scores, H needs also to know the denominator a_i in Equation (3), for all $v_i \in V$. That computation is already covered by Protocol 4.

As in Protocol 4, the arc structure E is obfuscated by hiding it within a larger arc set E' (Steps 1-2). In addition, H selects keys in a public key cryptosystem and sends the public key to $P_1, \ldots, P_m$ (Step 3). Then, each player computes for each action that it supports a corresponding set of time difference values, $\Delta_{\alpha,i,j}$, for all $(i, j) \in \Omega_{E'}$. The time difference $\Delta_{\alpha,i,j}$ equals $t_j - t_i$ in case $\mathbb{L}_k$ contains records (v_i, α, t_i) and (v_j, α, t_j) and $t_i < t_j$; in all other cases $\Delta_{\alpha,i,j} = 0$. After computing Δ_α, which consists of $\Delta_{\alpha,i,j}$ for all $(i, j) \in \Omega_{E'}$ (ordered first by i and then by j), P_k encrypts that information using the public key κ of H and sends it to P_1 (Steps 4-9). P_1 accumulates all those encrypted messages from his peers and sends them to H (Step 10). Finally, H decrypts those messages (Step 11) and uses them to construct the arc

[6] Fake records should be generated carefully, so that statistical patterns such as the number of distinct users who perform actions at the same time, or the number of distinct actions performed at the same time, would not differ significantly between the real and fake time stamps.

set $E(\alpha)$ in $PG(\alpha)$ for all $\alpha \in \mathcal{A}$ (Step 12); $E(\alpha)$ consists of all arcs $(v_i, v_j) \in E$ for which $\Delta_{\alpha,i,j} > 0$ (those are the arc labels).

Protocol 6 - Secure computation of the propagation graphs $PG(\alpha)$ for all $\alpha \in \mathcal{A}$

```
Input: $H$ has the social graph, $G = (V, E)$, where $V = \{v_1, \ldots, v_n\}$. $P_k$, $1 \le k \le m$, has an action log $\mathbb{L}_k$ .
Output: $H$ gets $PG(\alpha)$ for all $\alpha \in \mathcal{A}$.
1:  $H$ generates uniformly at random a set $E' \subset V \times V$ such that $E' \supset E$ and $|E'| \ge c|E|$ for some given constant $c > 1$.
2:  $H$ sends $\Omega_{E'} := \{(i,j) : (v_i, v_j) \in E'\}$ to $P_k$, $1 \le k \le m$.
3:  $H$ selects a pair of private and public keys in a public key cryptosystem and sends the public key, $\kappa$, to $P_k$, $1 \le k \le m$.
4:  for all $1 \le k \le m$ do
5:      for all actions $\alpha \in \mathcal{A}$ that are controlled by $P_k$ do
6:          $P_k$ computes $\Delta_\alpha := \{\Delta_{\alpha,i,j} : (i,j) \in \Omega_{E'}\}$.
7:          $P_k$ sends $M_\alpha := E_\kappa(\Delta_\alpha)$ to $P_1$.
8:      end for
9:  end for
10: $P_1$ sends $\{M_\alpha : \alpha \in \mathcal{A}\}$ to $H$.
11: $H$ decrypts $M_\alpha$ and recovers $\Delta_\alpha$ for all $\alpha \in \mathcal{A}$.
12: $H$ recovers the arc set $E(\alpha)$ and the corresponding arc labels.
```

We turn our attention to the security of this protocol. As discussed in Section 5.1, the private information of H, being the arc structure E, is obfuscated, and the level of obfuscation can be increased, at the cost of increased computational and communication costs, by increasing c. We next discuss the private information of P_k, $1 \le k \le m$.

THEOREM 6.1. *Assuming that the public key cryptosystem is secure, the only potential leakage of P_k's private information, $1 \le k \le m$, is to H. H may learn that for some action $\alpha \in A$, $v_i \xrightarrow{\alpha,\Delta t} v_j$, but nothing more: neither which action is α, nor its corresponding service provider, nor when did v_i perform that action.*

PROOF. Since the public key cryptosystem is assumed to be secure, and since all messages Δ_α are of the same size (they all include a sequence of $|E'|$ integers), P_1 (who does not posses the decryption key) can learn no information on the private action logs of the other service providers, P_k, $2 \le k \le m$. It remains to discuss what private information is leaked to H. H receives all M_α, $\alpha \in \mathcal{A}$, from P_1, thus there is no way for H to associate any of those messages with any of the service providers. In addition, since each M_α contains just the full set of nonzero time differences for the action α, but it does not contain any identification of the action α, H cannot associate any such set with any specific action. Therefore, Protocol 6 enables H to learn that for some action $\alpha \in A$, $v_i \xrightarrow{\alpha,\Delta t} v_j$, but nothing more: α could be any action, it could be controlled by any of the service providers, and v_i could have performed that action at any time $0 \le t \le T - 1 - \Delta t$. □

6.2 The non-exclusive case

Also here, the non-exclusive case can be reduced to the exclusive case by performing a preprocessing stage in which all players that control the same class of actions pull together the information regarding those actions, with the aid of a trusted third party. Specifically, let $\mathcal{A}_1$ be one of the action classes and let $\mathcal{P}_1$ be the set of players that control the actions in $\mathcal{A}_1$. Then those players may execute a modified version of Protocol 5 as follows.

The modified protocol coincides with Protocol 5 in Steps 1-3. We assume that the players in $\mathcal{P}_1$ use the basic method for obfuscating the action logs, as described in Section 5.2; namely, they obfuscate the user identifiers and actions in their action logs, using permutations π and σ. (The implementation of the enhanced method of obfuscation in this context is trickier. We defer the discussion of that implementation to the full version of this paper.) Then, the trusted third party, $\hat{P}$, uses $\hat{\mathbb{L}}_{\mathcal{A}_1}$ to compute all relevant propagation information. Specifically, for each action $\sigma(\alpha)$ that $\hat{P}$ sees in $\hat{\mathbb{L}}_{\mathcal{A}_1}$, he constructs a set $\hat{\Delta}_{\sigma(\alpha)}$. That set will contain a triplet of the form $(\pi(i), \pi(j), \Delta_{\sigma(\alpha),\pi(i),\pi(j)} := t' - t)$ whenever $\hat{\mathbb{L}}_{\mathcal{A}_1}$ contains the records $(v_{\pi(i)}, \sigma(\alpha), t)$ and $(v_{\pi(j)}, \sigma(\alpha), t')$, and $t' > t$. After computing $\hat{\Delta}_{\sigma(\alpha)}$for all obfuscated actions that appear in $\hat{\mathbb{L}}_{\mathcal{A}_1}$, $\hat{P}$ sends those sets to P_1. Finally, P_1 may retrieve from those sets the original user identifiers and actions, and, consequently, reconstruct the sets Δ_α as defined in Step 6 of Protocol 6, for all $\alpha \in \mathcal{A}_1$. After implementing this preprocessing stage for all action classes, the players may proceed as described in the exclusive case in Section 6.1.

7. EVALUATION

7.1 Communication costs

In order to evaluate the performance of our two main distributed protocols, we analyze their communication costs:

- NR: Number of communication rounds. (A communication round is a stage in the protocol in which some or all of the players send messages to other players, and the protocol can proceed only after all messages are received by their recipients.)
- NM: The total number of messages sent (from any player to any other player).
- MS: The total size in bits of all NM messages.

7.1.1 Protocol 4

The communication costs of Protocol 4 are analyzed in Table 1. (Here $q := |E'|$ and f is the size in bits for representing real numbers.) Each row in the table corresponds to one communication round. The first column indicates the corresponding step in the protocol description; the second column gives the number of messages that are sent in that round; and the third column gives the size in bits of each message.

As appears from Table 1, Protocol 4 entails $NR = 8$ communication rounds. The total number of messages sent is $NM = m^2 + m + 7$. As for message sizes, the longest messages occur in rounds 2,3,4 (assuming that S is chosen to be a very large integer for enhancing privacy). Since the total number of messages in those three rounds is m^2, we get that $MS = \Theta(m^2(n + q) \log S)$. Recall that $q = c|E|$, where $c > 1$. Hence, Protocol 4 involves sending roughly $cm^2 \log S$ bits of communication messages for each link.

7.1.2 Protocol 6

The communication costs of Protocol 6 are analyzed in Table 2. Here, $|\kappa|$ and z are, respectively, the size of the public key and cyphertext in the cryptosystem that H chose, and A_k is the number of actions that are controlled by P_k, $1 \leq k \leq m$. (We focus here on the exclusive case, whence $A = \sum_{k=1}^{m} A_k$). The first two rounds correspond to Steps 2 and 3 in the protocol. In the third round, player P_k, $2 \leq k \leq m$, sends to P_1 the encrypted message $E_\kappa(\Delta_\alpha)$ for each of the A_k actions that P_k controls. Hence, we have in this round $m-1$ parallel messages, where the kth message consists of A_k sub-messages, each of which contains q encryptions of an integer. Hence, the size in bits of the kth message is qzA_k. The last round corresponds to Step 10 in which P_1 sends to H the aggregation of all encrypted messages.

As appears from the table, Protocol 6 entails $NR = 4$ communication rounds. The total number of messages sent is $NM = 3m$. The overall message size in bits is dominated by the message size in the last two rounds, which equals $qz(A + \sum_{k=2}^{m} A_k) \leq 2qzA$. Hence, Protocol 6 involves sending $2zA$ bits of communication messages for each link. The recommended setting of z, when using the RSA cryptosystem, is 1024; when using an elliptic curve cryptosystem, a recommended setting for z is 256 (FIPS PUB 186-3).

7.2 Privacy offered by the secure division protocol

Here we conduct experimentation that examines the privacy preservation offered by Protocol 3. Let us recall the corresponding privacy setting: X is a discrete random variable that takes values in a domain of integers $\{0, 1, \dots, A\}$; in our application, X stands for the number of actions that some user performed, a_i, or the number of actions in which one user influenced another, $b^h_{i,j}$. The curious-but-honest party H has an a-priori belief probability $f_X(\cdot)$ regarding the value of X. In Protocol 3, H learns the value $y := rx$, where x is the value that X took and r is a random variable that is chosen in Steps 1-2 of the protocol. That value of y enables H to extract an a-posteriori belief probability, $f_X(\cdot|Y = y)$, regarding the value of x, as characterized in Theorem 4.4. Our goal here is to check whether the a-posteriori belief probability $f_X(\cdot|Y = y)$ allows H to guess x better than the a-priori belief probability $f_X(\cdot)$ does. To that end, we perform the following experiment:

1. Select an a-priori belief probability $f_X(\cdot)$ on $\{0, 1, \dots, A\}$, and then compute $\overline{f_X} := \sum_{i=0}^{A} i f_X(i)$. (In all experiments we used $A = 10$.)
2. Fix $x \in \{1, \dots, A\}$.
3. Compute $E_{pri} := |x - \overline{f_X}|$.
4. For $j = 1, \dots, 1000$ do:
   (a) Generate r and compute $y = rx$.
   (b) Compute $f_X(\cdot|Y = y)$ and $\overline{f_X(\cdot|Y = y)} := \sum_{i=0}^{A} i f_X(i|Y = y)$.
   (c) Compute $E_{pos} := |x - \overline{f_X(\cdot|Y = y)}|$.
   (d) Set $G_j = E_{pri} - E_{pos}$.

Here, E_{pri} is the error if H tries to guess that x equals the mean of the a-priori belief probability $f_X(\cdot)$. E_{pos}, on the other hand, is the error if H tries to guess that x equals the mean of the a-posteriori belief probability $f_X(\cdot|Y = y)$. The difference G_j is the gain that H obtains, regarding the value of x, as a result of seeing $y = rx$. Positive values of G_j indicate that the value of y in the jth trial helped H to make a better guess about x; negative values of G_j indicate the opposite.

We repeated this experiment with several prior belief probabilities over $x \in \{0, 1, \dots, A\}$. Due to space limitations, we report here the results with two priors: (a) A uniform prior, $f_X(i) = 1/(A + 1)$, for all $0 \leq i \leq A$. (b) A unimodal distribution that has one maximum at $i = A/2$: $f_X(i) = (i + 1)/(1 + A/2)^2$ for all $0 \leq i \leq A/2$ and $f_X(i) = (A + 1 - i)/(1 + A/2)^2$ for all $A/2 < i \leq A$. In each experiment we computed $1000A$ gain values, $\{G_j(i) : 1 \leq i \leq A, 1 \leq j \leq 1000\}$, where $G_j(i)$ indicates the gain in the jth trial when the initial value of x was set to $x = i$. Figure 1 shows histograms that describe the distribution of the resulting gains. In each of those histograms, the height of the bar over an interval $[a, b)$ equals the number of gains $G_j(i)$ that satisfy $a \leq G_j(i) < b$. Each histogram also shows the average gain over all $1000A$ trials. As can be seen, the average gain is positive and very small: the number of trials in which y does help to come up with a better guess for x is larger than the number of trials in which y yields worse guesses, but there is no significant bias towards the first case, and, in addition, the improvement in the quality of the guess is small. To summarize: from information theoretical point of view, y does reveal some information on x; but from a practical point of view the gain of information is insignificant.

Table 1: Communication costs of Protocol 4

Communication round	Num. messages	Message size
Step 2	m	$2\lceil \log n \rceil q$
Steps 3-4; Prot. 1, Step 2	$m(m-1)$	$(n+q)\lceil \log S \rceil$
Steps 3-4; Prot. 1, Step 4	$m-2$	$(n+q)\lceil \log S \rceil$
Steps 3-4; Prot. 2, Steps 3-4	2	$(n+q)\lceil \log 2S \rceil$
Steps 3-4; Prot. 2, Step 6	1	$n+q$
Step 5	2	nf
Step 6	2	nf
Steps 7-8	2	$(n+q)f$

Table 2: Communication costs of Protocol 6

Communication round	Num. messages	Message size
Step 2	m	$2\lceil \log n \rceil q$
Step 3	m	$|\kappa|$
Steps 4-9	$m-1$	qzA_k
Step 10	1	qzA

8. CONCLUSIONS AND FUTURE WORK

Viral marketing is a popular concept in the marketing literature which is receiving a growing attention for the great potentiality, and the challenging computational problems associated. Motivated by the real-world needs and constraints, in this paper we study the problems of how different parties can learn, in a privacy-preserving way, (1) the strength of influence along each link of a social network, and (2) an influence score for each node. These are fundamental computational problems at the basis of viral marketing. We devise secure multiparty computation protocols for these two problems and analyze their privacy guarantees.

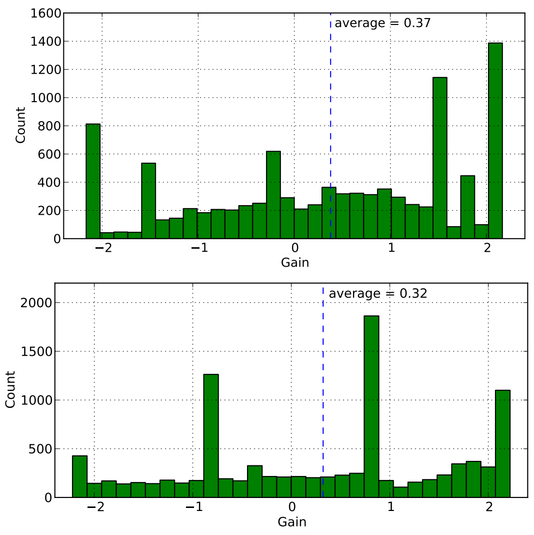

Figure 1: Histograms displaying the gain values G_j for a uniform priors (top) and a unimodal prior (bottom).

To the best of our knowledge, we are the first to discuss privacy issues of viral marketing and to propose protocols. This paper represents only a first step in this interesting and challenging domain. Future research directions include a solution of other computational problems in the setting that we discussed herein (like computing the nodes which maximize the expected spread [1]), or by considering more elaborated settings. One such setting is that in which all or some of the users are labeled by attributes (e.g., gender, location, occupation) that could be used, in conjunction with the activity logs, to better estimate the influence strengths of the links. Another setting is that of multiple hosts, where the graph data is split between several social networking platforms.

Acknowledgments. This work was supported by MULTISENSOR project, partially funded by the European Commission, under the contract number FP7-610411.

9. REFERENCES

[1] D. Kempe, J. M. Kleinberg, and É. Tardos, "Maximizing the spread of influence through a social network," in *KDD 2003*.

[2] K. Saito, R. Nakano, and M. Kimura, "Prediction of information diffusion probabilities for independent cascade model," in *KES 2008*.

[3] J. Tang, J. Sun, C. Wang, and Z. Yang, "Social influence analysis in large-scale networks," in *KDD 2009*.

[4] A. Goyal, F. Bonchi, and L. V. S. Lakshmanan, "Learning influence probabilities in social networks," in *WSDM 2010*.

[5] A. Borodin, M. Braverman, B. Luicer, and J. Oren, "Strategyproof mechanisms for competitive influence in networks," in *WWW 2013*.

[6] W. Lu, F. Bonchi, A. Goyal, and L. V. S. Lakshmanan, "The bang for the buck: Fair competitive viral marketing from the host perspective," in *KDD 2013*.

[7] A. Goyal, F. Bonchi, and L. V. S. Lakshmanan, "A data-based approach to social influence maximization," *PVLDB* 5(1):73–84, 2011.

[8] Nicola Barbieri, Francesco Bonchi, and Giuseppe Manco, "Topic-aware social influence propagation models," in *ICDM 2012*.

[9] S. Zhong, Z. Yang, and R. Wright, "Privacy-enhancing k-anonymization of customer data," in *PODS 2005*.

[10] W. Jiang and C. Clifton, "A secure distributed framework for achieving k-anonymity," *The VLDB Journal* 15:316–333, 2006.

[11] P. Domingos and M. Richardson, "Mining the network value of customers," in *KDD 2001*.

[12] M. Richardson and P. Domingos, "Mining knowledge-sharing sites for viral marketing," in *KDD 2002*.

[13] W. Chen, Y. Wang, and S. Yang, "Efficient influence maximization in social networks," in *KDD 2009*.

[14] W. Chen, C. Wang, and Y. Wang, "Scalable influence maximization for prevalent viral marketing in large-scale social networks," in *KDD 2010*.

[15] M. Mathioudakis, F. Bonchi, C. Castillo, A. Gionis, and A. Ukkonen, "Sparsification of influence networks," in *KDD 2011*.

[16] A. Goyal, F. Bonchi, and L. V. S. Lakshmanan, "Discovering leaders from community actions," in *CIKM 2008*.

[17] A. Goyal, B.-W. On, F. Bonchi, and L. V. S. Lakshmanan, "Gurumine: A pattern mining system for discovering leaders and tribes," in *ICDE 2009*.

[18] E. Bakshy, J. M. Hofman, W. A. Mason, and D. J. Watts, "Everyone's an influencer: quantifying influence on twitter," in *WSDM 2011*.

[19] R. Agrawal and R. Srikant, "Privacy-preserving data mining," in *SIGMOD 2000*.

[20] C. Dwork, "Differential privacy," in *ICALP 2006*.

[21] A. Yao, "Protocols for secure computation," in *FOCS 1982*.

[22] Y. Lindell and B. Pinkas, "Privacy preserving data mining," in *Crypto 2000*.

[23] X. Lin, C. Clifton, and M. Zhu, "Privacy-preserving clustering with distributed EM mixture modeling," *Knowl. Inf. Syst.* 8:68–81, 2005.

[24] J. Vaidya and C. Clifton, "Privacy preserving association rule mining in vertically partitioned data," in *KDD 2002*.

[25] M. Kantarcioglu and C. Clifton., "Privacy-preserving distributed mining of association rules on horizontally partitioned data," *TKDE*, 16:1026–1037, 2004.

[26] T. Tassa, "Secure mining of association rules in horizontally distributed databases," *TKDE*, to appear.

[27] T. Tassa and D. J. Cohen, "Anonymization of centralized and distributed social networks by sequential clustering," *TKDE*, 25:311–324, 2013.

[28] J. Benaloh, "Secret sharing homomorphisms: Keeping shares of a secret secret," in *Crypto1986*.

[29] S. Even, O. Goldreich, and A. Lempel, "A randomized protocol for signing contracts," *Comm. ACM* 28: 637–647, 1985.

[30] M. Naor and B. Pinkas, "Computationally secure oblivious transfer," *J. of Cryptology*, 18:1–35, 2005.