Contents lists available at ScienceDirect

Information and Computation

www.elsevier.com/locate/yinco

Computing in social networks[☆,☆☆]

Andrei Giurgiu[a], Rachid Guerraoui[a], Kévin Huguenin[a,*,1], Anne-Marie Kermarrec[b]

[a] EPFL, School of Computer and Communication Systems, EPFL, 1015 Lausanne, Switzerland
[b] INRIA Rennes – Bretagne Atlantique, Campus de Beaulieu, 35042 Rennes Cedex, France

ARTICLE INFO

Article history:
Available online xxxx

Keywords:
Distributed computing
Security
Privacy
Social networks

ABSTRACT

This paper defines the problem of Scalable Secure computing in a Social network: we call it the S^3 problem. In short, nodes, directly reflecting on associated users, need to compute a symmetric function $f : V^n \to U$ of their inputs in a set of constant size, in a *scalable* and *secure* way. Scalability means that the spatial, computational and message complexity of the distributed computation does not grow too fast with the number of nodes n. Security encompasses (1) accuracy and (2) privacy: accuracy holds when the distance from the output to the ideal result is negligible with respect to the maximum distance between any two possible results; privacy is characterized by how the information disclosed by the computation helps faulty nodes infer inputs of non-faulty nodes, which we capture in our context by the very notion of probabilistic anonymity.

We first prove that under mild regularity conditions the problem of computing an arbitrary function can be reduced to that of component-wise addition of vectors of integers. More specifically, if the function f is Lipschitz-continuous and the maximum distance between two possible results is $\Omega(n)$, any protocol that S^3-computes component-wise addition of vectors of integers S^3-computes f.

We then present AG-S3, a protocol that S^3-computes a class of aggregation functions, that is that can be expressed as a commutative monoid operation on U: $f(x_1, \ldots, x_n) = x_1 \oplus \cdots \oplus x_n$, assuming the number of faulty participants is at most $\sqrt{n}/\log^2 n$. We further prove that AG-S3 S^3-computes component-wise addition of vectors of integers thus extending its application spectrum to regular functions. Key to our protocol is a dedicated overlay structure that enables secret sharing and distributed verifications which leverage the social aspect of the network: nodes care about their reputation and do not want to be tagged as misbehaving.

1. Introduction

The past few years have witnessed an explosion of online social networks and the number of users of such networks is still growing by the day, *e.g.*, Facebook boasts by now more than 400 millions users. These networks constitute huge live

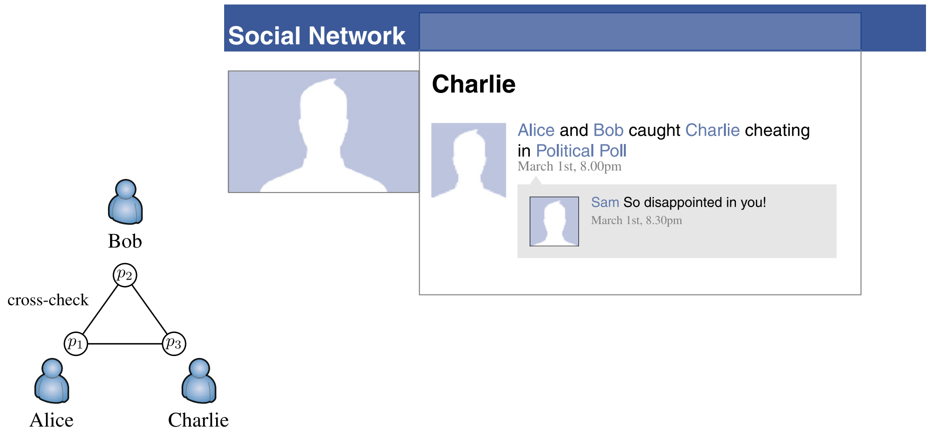
[☆] This paper is a revised and extended version of a paper that appeared in the Proceedings of the 12th International Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS 2010) [1].

[☆☆] This work has been partially supported by the ERC Starting Grant GOSSPLE number 204742.

[*] Corresponding author.
E-mail addresses: andrei.giurgiu@epfl.ch (A. Giurgiu), rachid.guerraoui@epfl.ch (R. Guerraoui), kevin.huguenin@epfl.ch (K. Huguenin), anne-marie.kermarrec@inria.fr (A.-M. Kermarrec).

[1] This research was partially carried out while Kévin Huguenin was working for his PhD at Université de Rennes 1/IRISA, France.

0890-5401/$ – see front matter © 2013 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.ic.2013.11.001



Charlie

Alice and Bob caught Charlie cheating in Political Poll
March 1st, 8.00pm

Sam So disappointed in you!
March 1st, 8.30pm

Fig. 1. Distributed verifications and reputation concern.

platforms that are exploited in many ways, from sharing personal information to conducting polls about political tendencies. An illustrative example of computation in a social network is crowdsourcing which exploits human-based computation capabilities and subjective and personal information of a group of users as a source of knowledge or ideas, *e.g.*, Amazon Mechanical Turk. It is clearly appealing to perform large-scale general purpose computations on such platforms and one might be tempted to use a central authority for that, namely one provided by the company orchestrating the social network. Yet, this poses several privacy problems, besides scalability. For instance, there is no guarantee that Facebook will not make any commercial usage of the personal information of its users. In 2009, Facebook tried to change its privacy policy to impose new terms of use, granting the company a perpetual ownership of personal contents—even if the users decide to delete their account. The new policy was not adopted eventually, but highlighted the eagerness of such companies to use personal and sensitive information.

We argue for a decentralized approach where the participants in the social network keep their own data and perform computations in a distributed fashion without any central authority. A natural question that arises then is what distributed computations can be performed in such a decentralized setting. Our primary contribution is to lay the ground for expressing the question precisely. We refer to the underlying problem as the S^3 problem: *Scalable Secure computing in a Social network*. Whereas *scalability* characterizes the spatial, computational and message complexity of the computation, the *secure* aspect of S^3 encompasses accuracy and privacy. *Accuracy* refers to the robustness of the computation and aims at ensuring accurate results in the presence of dishonest participants. This is crucial in a distributed scheme where dishonest participants might, besides disrupting their own input, also disrupt any intermediary result for which they are responsible. The main challenge is to limit the amount of bias caused by dishonest participants. *Privacy* is characterized by the amount of information on the inputs disclosed to other nodes by the computation. Intuitively, achieving all three requirements seems impossible. Clearly, tolerating dishonest players and ensuring privacy calls for cryptographic primitives. Yet, cryptographic schemes, typically used for multi-party computations, involve too high a computation overhead and rely on higher mathematics and the intractability of certain computations [2–4]. Instead, we leverage users' concern for reputation using an information theoretical approach and alleviate the need for cryptographic primitives. A characteristic of the social network context is indeed that the nodes are in fact users who might not want to reveal their input, nor expose their misbehavior if any. This reputation concern, as illustrated in Fig. 1, determines the extent to which dishonest nodes act: up to the point where their misbehavior remains discrete enough not to be discovered. In a system where users report on the misbehaviors they detect, dishonest node might be tempted to issue spurious reports on other users. However, in the context of social networks, two key factors help thwarting such a threat: First, reports are intended to be read by users (not programs) who can assess the credibility of the reports and decide whether to take them into account; Second, the knowledge of the social ties between users can be leveraged. For instance, reports from an enemy or a joint report issued by users that are connected in the social network could be disregarded. Such techniques proved efficient in areas as diverse as on-line games [5], recommendation systems [6], and spam filtering [7].

Solving the S^3 problem is challenging, despite leveraging this reputation concern: to ensure privacy, an algorithm must ensure that the information obtained by the coalition of faulty nodes during the protocol is not enough to determine with certainty a node's input. This property should hold even when all the non-faulty nodes except one have the same inputs: faulty nodes taking part in the computation must not know which non-faulty node had a different input. This requires the existence of two configurations of inputs that differ for two non-faulty nodes having different inputs, which with high probability lead to the same sequence of messages received by the faulty nodes. In turn, this comes down to *swapping* two nodes' inputs transparently (from the standpoint of the faulty nodes), which is challenging when the protocol needs to be also scalable and accurate. The scalability requirement (*i.e.*, each node communicates with a limited number of nodes) makes it difficult to find a chain of messages that can be swapped transparently between any two nodes in the system. The trade-off between privacy and accuracy can be illustrated by the following paradox: on the one hand verifying that nodes do not corrupt the messages they receive (without digital signature) requires the verifier to gather some information about what the verified node received; on the other hand the more the nodes know about the messages exchanged the more the privacy of the nodes is compromised.

Our contributions are twofold:

- Firstly, we define the Scalable Secure computing problem in a Social network, namely the S^3 problem and prove that it can be reduced, for regular functions, to that of computing component-wise addition of vectors of integers with exactly one nonzero component, which is equal to 1. More specifically, if the function f is Lipschitz-continuous and the maximum distance between two possible results is $\Omega(n)$, any protocol that S^3-computes component-wise addition of vectors of integers S^3-computes f.
- Secondly, we present a distributed protocol, we call AG-S3 (*i.e.*, S^3 for AGgregation), that solves the problem for a class of aggregation functions that derive from a monoid operation on U: $f(x_1, \dots, x_n) = x_1 \oplus \cdots \oplus x_n$, under the assumption that the number of faulty nodes is upper-bounded by $\sqrt{n}/\log^2 n$. We further prove that AG-S3 S^3-computes component-wise addition of vectors of integers thus extending its application spectrum to regular functions. At the core of our protocol lie (1) a structured overlay where nodes are clustered into groups, (2) a homomorphic secret sharing scheme that allows the nodes to obfuscate their inputs, and (3) a verification procedure which potentially tags the profiles of suspected nodes.

Beyond these contributions, our paper can be viewed as a first step toward characterizing what can be computed in a large scale social network while accounting for the human and social nature of its users.

2. The S^3 problem

This section defines the problem of *Scalable Secure* computing in a *Social network*: the S^3 problem. The problem involves an S^3 candidate, namely the function to be computed, and a set of nodes $\Pi = \{p_1, \dots, p_n\}$. We assume the number n of nodes to be known by the nodes. The main notations used throughout the paper are summarized in Table A.1.

2.1. Candidates

Definition 1 *(S^3 candidate).* An S^3 candidate is a quadruple (f, V, U, d), where V is an arbitrary set, f is a function $f: V^* \to U$ such that $f(v_1, \dots, v_n) = f(v_{\sigma(1)}, \dots, v_{\sigma(n)})$ for any permutation σ of the inputs, and (U, d) is a metric space.

Each node in Π has an input value in the set V, and an S^3 candidate maps the inputs of the nodes to a value in a metric space. The function f is assumed to be symmetric in the sense that the output depends on the multiset of inputs but not on their assignment to nodes. For example, a binary poll over Π can be modeled by the S^3 candidate $((v_1, v_2, \dots, v_n) \mapsto v_1 + \cdots + v_n, \{-1, +1\}, \mathbb{Z}, (z_1, z_2) \mapsto |z_1 - z_2|)$. Indeed, in a binary poll, each node starts with a value coding either the “yes” or the “no” options and the result of the poll is the proportion of participants who chose each of the two options. The result of the poll can be computed by encoding the options with the integers $+1$ and -1 respectively and by summing them (yielding an integer output r in $\mathbb{Z}$). More specifically, the proportion of participants who chose the “yes” option is $(n + r)/2n$ and the majority is determined by the sign of r (*e.g.*, with 8 participants, an output of 2 means that the “yes” option has the majority and that 62.5% of the participants chose it). A natural distance on the output space (*i.e.*, $U = \mathbb{Z}$) — used to quantify the accuracy of the computation as we shall see — is the absolute difference $(z_1, z_2) \mapsto |z_1 - z_2|$. In the case of m-ary polling, one could consider the component-wise addition on $U = \mathbb{Z}^m$, where V is the set of all vectors of length m with exactly one nonzero component, which is either $+1$ or -1. The distance function is then just ℓ^1 (or Manhattan distance).

The nodes considered in the S^3 problem are users of a social network, able to (1) communicate with private message passing and (2) tag the public profile of each other. As such, every node directly reflects on the associated user. Nodes care about their privacy and their reputation: a user wants neither the private information contained in her input, nor her misbehavior, if any, to be disclosed. This reputation concern is crucial to make the problem tractable.

To ensure security, part of the computation consists in checking the correctness of other nodes’ behavior. The output of a node p is a value in U, *i.e.*, the result of the distributed computation of the S^3 candidate, plus a set $\mathcal{F}_p$ of nodes that p detected as faulty. This information is eventually reported on the public profile of the involved nodes by means of tags of the form “p detected nodes in $\mathcal{F}_p$ as faulty”. Nodes are allowed to report only on the nodes they communicate with and the number of reports a node can issue is limited to limit the impact of spurious reports against non-faulty nodes.

Faulty nodes are considered rational: their goal is only to bias the output of the computation and infer the inputs of the users taking part in the computation. Also, they never behave in such a way that their misbehavior is exposed with certainty. As such, their behavior is more restricted than that of Byzantine users [8].[2] To achieve their goal, faulty nodes may collude.

In our context, a protocol $\mathcal{D}$ to perform the distributed computation of an S^3 candidate on the set of nodes Π is a sequence of message exchanges and local computations such that any non-faulty node p eventually outputs a value o_p. The content of the message and the nodes' outputs are random variables whose values are determined by the random choices made by the nodes during the computation. In the following, we define the desirable properties of a protocol in a social network, namely scalability and security, itself encompassing privacy and accuracy.

2.2. Scalability

Scalability means intuitively that the computation is able to handle a large number of nodes (*i.e.*, large values of n), that is that the amount of resources needed by the protocol grows reasonably with the number of input values. Consequently, the properties are expressed in the form of asymptotic bounds.

Definition 2 *(g-Scalability).* A protocol to perform a distributed computation is said to be g-scalable (for a function $g : \mathbb{N} \to \mathbb{N}$) if the message, spatial and computational complexities at each node are $\mathcal{O}(g(n) \cdot \operatorname{polylog} n)$ in the worst case.

The logarithmic factor comes from the fact that representing a node identifier ranging from 1 to n requires $\log_2 n$ bits. Therefore, storing an identifier or comparing two identifiers requires $\log_2 n$ basic operations. A g-scalable protocol can therefore involve $\mathcal{O}(g(n))$ operations on node identifiers.

Note that g-scalability may not be a desirable property for any function g. In particular, exp-scalability is of little interest, 1-scalability is best, log-scalability is very desirable and $\sqrt{\ }$-scalability (achieved in this paper as we shall see in Section 3) is acceptable.

2.3. Accuracy

The definition of the accuracy of a computation relies on the metric space structure of the output space U as the distance measure enables to quantify the gap between the result of the computation and the actual result, that is $f(v_1, \dots, v_n)$. To render it meaningful, we normalize this distance by the maximum distance between any two possible results, namely the diameter of $f(V^n)$ (where $f(V^n)$ denotes the image, w.r.t. f, of the set of sequences of n values in V), for a distributed computation over n nodes.

Definition 3 *(g-Accuracy).* A distributed computation is said to g-accurately compute an S^3 candidate (f, U, V, d) if:

$$\frac{1}{\Delta(n)} \cdot \max_{p \text{ non-faulty}} d\big(o_p, f(v_1, \dots, v_n)\big) = \mathcal{O}\left(\frac{1}{g(n)}\right),$$

where v_i is the input of the i-th node and

$$\Delta(n) = \max_{\substack{(x_1,\dots,x_n) \\ (y_1,\dots,y_n)}} d\big(f(x_1, \dots, x_n), f(y_1, \dots, y_n)\big).$$

This definition highlights the importance of carefully specifying the distance measure of an S^3 candidate: endowing the output space with the coarse grain distance $d(x, y) = 0$ if $x = y$, and 1 otherwise, will restrict the class of S^3 computations to those that output the exact result of f. Meanwhile, for binary polling for instance, considering the natural distance (*i.e.*, $d(x, y) = |x - y|$) and a function g that tends to infinity when n tends to infinity includes computations for which the error on the tally is negligible when compared to the sample size n, as $\Delta(n) = 2n$.

2.4. Privacy

The privacy of users can be compromised by the information exchanged by the nodes during the course of the computation. We characterize the privacy leaks of a distributed computation by the manner in which the information gained by curious nodes taking part in the distributed computation enables them to recover with certainty the input of a particular non-faulty node. The information acquired by a coalition of curious nodes is composed of (1) their input values, (2) the output of the computation and (3) the pieces of information contained in the messages exchanged during the computation (specified by the protocol). Only the information from (3) is inherent to the computation. Clearly, the cases where an input can be inferred from only the output and the inputs of the faulty nodes must be ignored when looking at the privacy leaks

[2] The fault model considered in the paper is indeed more restricted than the Byzantine fault model. However, the problem addressed, *i.e.* S^3, is not directly comparable to that of Byzantine consensus as S^3 includes scalability and privacy and relaxes the accuracy requirement. Consequently, the maximum number of faulty nodes tolerated by solutions to S^3 may be less than for the Byzantine consensus problem (*i.e.*, $(n-1)/3$).

of a computation. In a perfectly private distributed computation, a coalition of faulty nodes should be able to recover the input of a non-faulty node if and only if its input can be inferred from the output of the computation and the inputs of the faulty nodes alone. Such configurations of inputs are captured by the notion of *non-ambiguous* input configuration that we formalize below.

Consider for instance that all the non-faulty nodes have the same input value, say v_0, and that this configuration of inputs for non-faulty nodes is the only one which, when combined with the inputs of faulty nodes, yields the observed output. Then, the faulty nodes know that all non-faulty nodes had input v_0, which breaks privacy. More concretely, consider the case of binary polling in a system of five nodes, two of them being curious and colluding. If these two curious nodes voted respectively for -1 and $+1$ and the outcome of the poll is $+3$ then they know with certainty that all the three other nodes voted $+1$. In general, when all the non-faulty nodes have the same input, be it -1 or $+1$, the outcome of the poll minus the sum of the votes of faulty nodes is equal to the number of non-faulty nodes and thus the vote of non-faulty nodes can be inferred with certainty by the coalition.

Definition 4 *(Non-ambiguous input configuration).* An element $\mathbf{v} = \{v_p\}_{p\in\Pi}$ of V^n is said to be a non-ambiguous input configuration for a coalition B if there exists a node $p \notin B$ such that for all input configuration $\mathbf{v}'$ that matches $\mathbf{v}$ for all nodes in B, $f(\mathbf{v}) = f(\mathbf{v}')$ implies $v_p = v'_p$.

Definition 5 *(Trivial input configuration).* An element $\mathbf{v}$ of V^n is said to be a trivial input configuration for a coalition B if all the nodes that are not in B have the same input.

Since S^3 candidates are symmetric by definition, all the non-faulty nodes have the same input in a non-ambiguous input configuration, otherwise it would not possible to map two different inputs to the corresponding non-faulty nodes: a non-ambiguous is necessarily trivial. However, a trivial input may be ambiguous. Consider for illustration the case of addition with the input set $V = \{1, 2, 3\}$. The configuration where all the non-faulty nodes have 2 as input is trivial but ambiguous for the faulty nodes as it yields the same output as the configuration where half of the non-faulty nodes have 1 as input and the other half has 3.

We say in our context that a distributed computation is *private* if the probability of recovering the input of at least one non-faulty node decreases as $1/n^\alpha$ for some positive α (referred to as *with high probability*). We capture this notion more formally through the notion of *probabilistic anonymity*, itself based on the very notion of *message trace*. We distinguish between weak and strong probabilistic anonymity depending on whether all trivial input configurations are ignored or only non-ambiguous ones. In the context of the S^3 problem where the functions to be computed are symmetric, strong probabilistic anonymity implies weak probabilistic anonymity. Note that for binary polling, *i.e.*, addition of input values in $\{-1, +1\}$, the trivial input configurations are all non-ambiguous: the trivial input configurations (all -1 or all $+1$) are the only input configurations that yield an output of $-n$ and $+n$ respectively (note that to be able to infer the input values of non-faulty nodes, the coalition needs to know the total number n of nodes). Weak and strong probabilistic anonymity are therefore equivalent in this case.

Definition 6 *(Message trace).* A *message trace* (or *trace* for short) of a distributed computation is a set of messages sent and received in a possible execution of the computation. A partial trace is the set of the messages sent or received by a subset of the nodes. A partial trace is said to be *compatible* with an input configuration $\mathbf{v}$ if it can be obtained from $\mathbf{v}$ with nonzero probability. We say that two traces are equivalent with respect to a coalition B of faulty nodes if each node in B receives the exact same messages in both traces, *i.e.*, the two partial traces restricted to the nodes of B are equal.

We are now ready to introduce the concepts of weak and strong probabilistic anonymity, which encapsulate the degree of privacy we require in the S^3 problem.

Definition 7 *(Weak probabilistic anonymity).* A distributed computation is said to be *weakly probabilistically anonymous* if for any coalition B of faulty nodes, for any non-faulty node p, and for any trace T compatible with a *non-trivial* (w.r.t. B) input configuration $\mathbf{v}$, there exists with high probability a trace T' compatible with an input configuration v' (w.r.t. B) such that (1) T and T' are equivalent w.r.t. B and (2) $\mathbf{v}$ and $\mathbf{v}'$ differ on the input value of node p.

Definition 8 *(Strong probabilistic anonymity).* A distributed computation is said to be *strongly probabilistically anonymous* if for any coalition B of faulty nodes, for any non-faulty node p, and for any trace T compatible with an *ambiguous* (w.r.t. B) input configuration $\mathbf{v}$, there exists with high probability a trace T' compatible with an input configuration $\mathbf{v}'$ (w.r.t. B) such that (1) T and T' are equivalent w.r.t. B and (2) $\mathbf{v}$ and $\mathbf{v}'$ differ on the input value of node p.

The intuition behind these definitions is that one can change the values of the non-faulty nodes in such a way that, with high probability, the messages received by the coalition remain unchanged. Consequently, the coalition of faulty nodes cannot distinguish between different executions of a computation in which non-faulty nodes had different inputs. Thus the coalition hesitates between at least two input values for each node and cannot infer their inputs with certainty.

Note that possible disruptions committed by faulty nodes are not taken into account in the definitions of anonymity. However, such disruptions can only increase the privacy of the nodes as they would confuse the faulty nodes that try to infer other nodes' inputs and lead them to incorrect conclusions.

Definition 9 *(S^3 computation).* A distributed computation is said to (g, h, weak) S^3-compute (resp. (g, h, strong) S^3-compute) an S^3 candidate $\mathcal{C}$ if it is g-scalable, it h-accurately computes $\mathcal{C}$ and it is weakly probabilistically anonymous (resp. strongly probabilistically anonymous) for a set Π of nodes tight to users of a social network.

2.5. Reduction to the addition problem

This sub-section shows that a protocol that $(g, h, \mathit{weak}/\mathit{strong})$ S^3-computes component-wise addition of vectors of integers with exactly one nonzero component which is equal to 1, (g, h, weak) S^3-computes any Lipschitz-continuous candidate such that V is finite and the diameter $\Delta(n)$ is $\Omega(n)$. The goal of this theorem is to show that, by solving the problem for one specific S^3 candidate we solve it for a much wider class of candidates as well. In particular, it motivates our choice to focus on aggregation functions (which include component-wise addition) in this paper, as we shall see in Section 3.

The reduction of the S^3 problem relies on a compact representation of the multiset of input values in a finite set V using integer vectors such that the union of the multisets corresponds to the component-wise addition of their vector representation. Therefore, if a protocol computes accurately the multiset of input values and if the function to be computed is regular enough, *i.e.*, a small deviation on the multiset of input values translates into a small deviation on its output, then each node can locally and accurately compute the output. We now formalize the notions of Lipschitz continuity and compact representation and prove the reduction theorem.

Definition 10 *(Lipschitz continuity).* A function $f : A \to B$ is said to be *Lipschitz-continuous* with respect to the distance measures d_A and d_B, if for any two elements x and y in A the distance between $f(x)$ and $f(y)$ is within a constant factor of the distance between x and y. That is, there exists a constant number k such that for all x, y in A:

$$d_B\big(f(x), f(y)\big) \leqslant k \cdot d_A(x, y).$$

Consider now an S^3 candidate (f, V, U, d) as in Definition 1. Since f is symmetric, it can be thought of as a function that maps a multiset of input values in V to a value in U. A natural distance between multisets, that we denote d_{ms}, is the number of elements that appears in only one of the two multisets. For instance, $d_{\mathrm{ms}}(\{\{1, 1, 2, 2, 3\}\}, \{\{2, 3, 3, 4\}\})$ is 5: 2 because of the 1s, 1 because of the 2, 1 because of the 3, and 1 because of the 4.

Provided that V is of finite size, *i.e.*, $V = \{v_1, \ldots, v_{|V|}\}$, a way to represent a multiset is to use a vector of $|V|$ integers where the i-th component represents the multiplicity of v_i in the multiset, *i.e.*, the number of occurrences of v_i. Basic operations on multisets such as union and the natural distance d_{ms} can be directly computed from the compact representation: union corresponds to component-wise addition and distance corresponds to the ℓ^1 norm of the component-wise difference. Consider for the sake of illustration the set $V = \{v_1, v_2\}$ and the multisets $S_1 = \{\{v_1, v_1\}\}$ and $S_2 = \{\{v_1, v_2\}\}$. The compact representations of S_1 and S_2 are $(2, 0)$ and $(1, 1)$ respectively. The distance $d_{\mathrm{ms}}(S_1, S_2)$ is 2 which is equal to $\|(2, 0) - (1, 1)\|_1 = \|(1, -1)\|_1 = |1| + |-1| = 2$. The union of S_1 and S_2 is $\{\{v_1, v_1, v_1, v_2\}\}$ which corresponds to the compact representation $(2, 0) + (1, 1) = (3, 1)$. The following diagram illustrates the concept of compact representation on this sample example.

$$\begin{tikzcd}
\{\{v_1, v_1\}\},\ \{\{v_1, v_2\}\} \arrow[r, "\cup_{\mathrm{ms}}"] \arrow[d] & \{\{v_1, v_1, v_1, v_2\}\} \\
(2, 0),\ (1, 1) \arrow[r, ".+."] & (3, 1) \arrow[u]
\end{tikzcd}
\qquad
\begin{tikzcd}
\{\{v_1, v_1\}\},\ \{\{v_1, v_2\}\} \arrow[dr, "d_{\mathrm{ms}}"] \arrow[dd] & \\
& 2 \\
(2, 0),\ (1, 1) \arrow[ur, "\|.-.\|_1"] &
\end{tikzcd}$$

That being said, we can now define a Lipschitz-continuous S^3 candidate:

Definition 11 *(Lipschitz-continuous S^3 candidate).* An S^3 candidate (f, V, U, d) is said to be Lipschitz-continuous if f is Lipschitz-continuous with respect to the distances d_{ms} and d.

Theorem 1 *(Reduction of the S^3 problem). Let $\mathcal{C} = (f, V, U, d)$ be a Lipschitz-continuous S^3 candidate such that* (1) *V is of finite size,* (2) *the diameter $\Delta(n)$ is $\Omega(n)$ and* (3) *there exists an algorithm $\mathcal{A}$ that locally computes f from the compact representation of the multiset of inputs in $\mathcal{O}(g(N) \cdot \mathrm{polylog}(n))$ basic operations. If a protocol $\mathcal{D}$, (g, h, weak) S^3-computes (w.r.t. to the ℓ^1 norm) component-wise addition of vectors of integers with exactly one nonzero component which is equal to* 1, *then there exists a protocol that (g, h, weak) S^3-computes f.*

Please cite this article in press as: A. Giurgiu et al., Computing in social networks, Inform. and Comput. (2013), http://dx.doi.org/10.1016/j.ic.2013.11.001

Proof. To prove the theorem, we build a protocol and prove that it meets the scalability, accuracy and privacy requirements of the S^3 problem. Consider the following protocol: each node p transforms its input v_p into the compact representation of the multiset $\{\!\{v_p\}\!\}$ and runs protocol $\mathcal{D}$ that outputs the compact representation ms_p of the multiset of the nodes' inputs. Using a compact representation of the multiset is possible as V is finite. Each node p then locally computes $o_p = f(\mathrm{ms}_p)$ using algorithm $\mathcal{A}$ which takes the compact representation as input.

- **Scalability.** Protocol $\mathcal{D}$ is g-scalable by assumption. Therefore, its complexity is in $\mathcal{O}(g(n) \cdot \mathrm{polylog}\,(n))$. So is the complexity of $\mathcal{A}$ by assumption. Building the compact representation of an input value is $\mathcal{O}(\log n)$ as it consists in building a constant-size vector of integers coded on $\log_2(n)$ bits. The complexity of the proposed protocol is therefore $\mathcal{O}(g(n) \cdot \mathrm{polylog}\,(n))$, which proves the g-scalability property.
- **Accuracy.** By assumption, the multiset of inputs is h-accurately computed at each node by protocol $\mathcal{D}$. The candidate $\mathcal{C}$ is Lipschitz-continuous, which ensures that the error on the final result is within a constant factor of the error on the multiset of inputs. The diameter $\Delta_{\mathrm{me}}(n)$, with respect to the distance measure d_{me}, for the computation of the multiset is $2n$ as the multisets are of size n. Finally, we have $\Delta(n) = \Omega(n)$, with respect to distance d, for candidate $\mathcal{C}$. Putting everything together, we get:

$$\begin{aligned}\frac{1}{\Delta(n)} \cdot \max_{p \text{ non-faulty}} d\big(o_p, f(v_1, \ldots, v_n)\big) &= \frac{1}{\Delta(n)} \cdot \max_{p \text{ non-faulty}} d\big(f(\mathrm{ms}_p), f(v_1, \ldots, v_n)\big) \\ &\leqslant k \cdot \frac{1}{\Delta(n)} \cdot \max_{p \text{ non-faulty}} d_{\mathrm{ms}}\big(\mathrm{ms}_p, \{\!\{v_1, \ldots, v_n\}\!\}\big) \\ &\leqslant 2k \cdot \frac{n}{\Delta(n)} \cdot \frac{1}{\Delta_{\mathrm{ms}}(n)} \cdot \max_{p \text{ non-faulty}} d_{\mathrm{ms}}\big(\mathrm{ms}_p, \{\!\{v_1, \ldots, v_n\}\!\}\big), \\ \frac{1}{\Delta(n)} \cdot \max_{p \text{ non-faulty}} d\big(o_p, f(v_1, \ldots, v_n)\big) &= \mathcal{O}(1) \cdot \mathcal{O}\left(\frac{1}{h(n)}\right).\end{aligned}$$

  Therefore, the proposed protocol h-accurately computes candidate $\mathcal{C}$.
- **Privacy.** First note that for the computation of the multiset of inputs, as for binary polling, trivial inputs are non-ambiguous. The proposed protocol can therefore, at best, achieve weak probabilistic anonymity since the inputs of the non-faulty nodes can be inferred from the multiset when the input configuration is trivial. Since the local computation does not bring any further information to faulty nodes, the weak probabilistic anonymity of protocol $\mathcal{D}$ implies the weak probabilistic anonymity of the proposed protocol for the computation of candidate $\mathcal{C}$. □

Theorem 1 is an incentive to focus on the special case of component-wise vector addition and to extend the application spectrum of existing protocols which can compute component-wise vector addition.

3. Protocol

In this section, we focus on a class of aggregation functions and propose a protocol, namely AG-S3 (S^3 for AGgregation), which $(\surd, \surd, \mathit{weak})$ $S1^3$-computes such functions for $|B| \leqslant \sqrt{n}/\log^2 n$ faulty nodes. We further prove that the proposed protocol satisfies the conditions of Theorem 1, thus extending the application spectrum of AG-S3 to regular functions. The point $(\surd, \surd, \mathit{weak})$ in the design space is of practical interest as it constitutes a quite scalable and private solution providing relatively fast asymptotic accuracy, *i.e.*, the error on the output tends quite rapidly to zero when n tends to infinity. In this paper, we focus on the formal definition of the problem and on the theoretical analysis of the solution. Regarding the practical aspects of the protocol, an experimental analysis of the Dpol protocol (which makes use of many building blocks and basic techniques common with AG-S3, as described in the related work in Section 4), reporting on its deployment on the PlanetLab testbed in the presence of faulty nodes, message loss, synchronization issue, *etc.*, can be found in [9].

3.1. Assumptions

We consider S^3 candidates for which the function f is an aggregation function, *i.e.*, deriving from an associative binary operation on U: $f(v_1 \ldots, v_n) = v_1 \oplus \cdots \oplus v_n$. Because an S^3 candidate must be symmetric, the '$\oplus$' operation is commutative. This induces a commutative monoid structure on $(U, \oplus)$ and it implies that V is a subset of U. We further assume that the '$\oplus$' operation is *compatible* with the distance measure d in the sense that

$$d\big(v_1 \oplus v_2, v'_1 \oplus v'_2\big) \leqslant d\big(v_1, v'_1\big) + d\big(v_2, v'_2\big). \tag{1}$$

As an example, note that the S^3 candidate $((v_1, v_2, \ldots, v_n) \mapsto v_1 + \cdots + v_n, \{-1, +1\}, \mathbb{Z}, (z_1, z_2) \mapsto |z_1 - z_2|)$, introduced in the previous section, satisfies the compatibility condition described above. A simple example of S^3 candidate which cannot be expressed as an aggregation is the one given by the sum of products of pairs of inputs, *i.e.*, $f(x_1, \ldots, x_n) = \sum_i \sum_{j \neq i} x_i \cdot x_j$. This function is symmetric, and choosing $U = \mathbb{Z}$ turns this function into a valid S^3 candidate, but it is clearly not an aggregation function.

Please cite this article in press as: A. Giurgiu et al., Computing in social networks, Inform. and Comput. (2013), http://dx.doi.org/10.1016/j.ic.2013.11.001

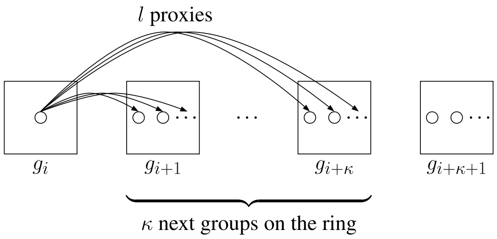


Fig. 2. Overview of the overlay.

We assume the size of the set of possible inputs to be constant and the size of the output space to be polynomial in n, implying that any input or output can be represented by $\mathcal{O}(\log n)$ bits. The computational complexity of $\oplus$ and d are assumed to be linear in the size of their operand, *i.e.*, $\mathcal{O}(\log n)$. In addition, we assume that the diameter $\Delta(n)$ of the output space is $\Omega(n)$. Due to this assumption, bit operators do not fall into our definition. Finally, we assume that V is closed with respect to inverses: if v is in the input set V then $\ominus v$ is in V as well, where $\ominus v$ denotes the inverse of v with respect to the '$\oplus$' operation. We denote by δ_V the diameter of V: $\delta_V = \max_{v,v' \in V} d(v, v')$.

3.2. Design rationale

The main challenge of S^3 computing is the trade-off between scalability, accuracy and privacy. We describe below this trade-off and how we address it before describing the protocol in details.

To ensure scalability, we cluster the nodes into groups of size $\sqrt{n}$, and require that a node sends messages only to other nodes in a small set of neighboring groups. We introduce two parameters of the protocol, κ and l. A node p is allowed to send messages to any other node in its own group, and to exactly l nodes in each of κ other groups. For scalability, l and κ need to be low, since they are directly proportional to message complexity. The same for accuracy: intuitively, the larger l and κ, the more opportunities a node has to cheat (*i.e.*, corrupt the unique pieces of information it receives before forwarding them), which entails a higher impact on the output. To preserve privacy (*i.e.*, probabilistic anonymity), we need a mechanism which, for any node p, transforms any trace into another trace, in such a manner that all messages received by the coalition of faulty nodes are preserved, and p has a different input in the two traces. This prevents the coalition from determining the input value of p. It will become apparent in our proof of privacy that both κ and l need to be large in order to obtain reasonable privacy requirements. To summarize, accuracy and scalability require the parameters κ and l to be small, whereas privacy requires them to be large. As a trade-off, we pick them both to be $\Theta(\log n)$, which ensure good S^3 properties, that is $\sqrt{\ }$-scalability, $\sqrt{\ }$-accuracy and weak anonymity.

3.3. Protocol

We describe AG-S3 which computes general aggregation in an S^3 manner. The protocol is composed of two interleaved components: one computes the aggregation function while the other checks the behavior of users. The pseudo-code of all is given in Algorithm 1 and illustrated in Figs. 2 and 3.

Structure. AG-S3 uses a special structure inspired from [10], where the n nodes are distributed into groups of size $\sqrt{n}$. Such an overlay can be obtained in a distributed fashion with strong guarantees on the randomness of nodes placement in the groups even in the presence of faulty nodes [11]. The groups (or *offices*) are placed on a ring, with nodes from a particular group sending messages to either nodes from the same office (called *officemates*) or to selected nodes from the next offices on the ring (called *proxies*). More specifically, a node is connected to its $\sqrt{n} - 1$ officemates and to l proxies in each of the next κ groups on the ring (see Fig. 2). If a node p' is a proxy of p, then p is said to be a *client* of p'. The partitioning into groups and their placement on the ring are chosen uniformly at random. We further assume a perfect client-proxy matching that ensures that a proxy has exactly $\kappa \cdot l$ clients. For example, we can index the nodes inside each group and assign to the i-th node of a group the nodes $i+1, \ldots, i+l \bmod \sqrt{n}$ as proxies in each of the next κ groups on the ring. We set $\kappa = 3/2 \cdot \lfloor \log n \rfloor$ and $l = 5 \cdot |V| \cdot \lfloor \log n \rfloor + 1$. These choices will become clear in the proofs of the next section.

Aggregation. In the first phase, each participant splits its input into $\kappa \cdot l$ *shares* in V and sends them randomly to its assigned proxies. The randomized scheme ensures that the aggregate of the shares is the input value. The shares are generated as follows: $(\kappa \cdot l - 1)/2$ are chosen uniformly at random, $(\kappa \cdot l - 1)/2$ are the inverses of the randomly chosen shares, and one is the actual input of the node.

In the counting phase, each proxy aggregates the shares received in the previous phase to obtain an *individual aggregate*. Each node then broadcasts its individual aggregate to all its officemates. Each node computes the aggregate of the individual aggregates of its officemates and obtains a *local aggregate*. If all nodes are non-faulty, then all local aggregates computed in an office are identical and equal to the sum of the shares sent to proxies in the group.

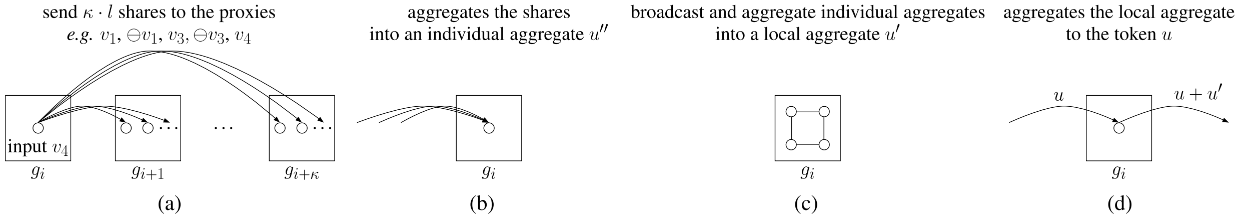

send $\kappa \cdot l$ shares to the proxies
e.g. $v_1, \ominus v_1, v_3, \ominus v_3, v_4$
input v_4
g_i
g_{i+1}
$g_{i+\kappa}$
(a)
aggregates the shares into an individual aggregate u''
g_i
(b)
broadcast and aggregate individual aggregates into a local aggregate u'
g_i
(c)
aggregates the local aggregate to the token u
u
$u + u'$
g_i
(d)

Fig. 3. Illustration of the phases of protocol.

In the *forwarding phase*, the local aggregates are disseminated to other nodes thanks to tokens forwarded along the ring, as explained below. The forwarding phase is bootstrapped by a special group (that can be determined by the social networking infrastructure at random). The nodes in this special group send a token containing the local aggregate computed in their group to their proxies in the next group only. The tokens are further forwarded along the ring. The first time a token reaches a node in a particular group, this node aggregates the local aggregate to the token and forwards it to its proxies in the next group only. When a node receives a token for the second time, the node sets its own output to the value of the token and forwards it. The third time a node receives a token, it discards it.

Algorithm 1 Pseudo-code version of the AG-S3 protocol.

```
Input: an input value $v \in V$
Variables: individual aggregate $u'' = 0_U$
local aggregate $u' = 0_U$

procedure share($v$) :
  for $i = 1$ to $(l \cdot \kappa - 1)/2$ do
    $b_i = \mathrm{rand}(V)$   # insertion of a random value in $V$ (uniformly)
    $b_{i+(\kappa \cdot l-1)/2} = \ominus s_i$   # insertion of the inverse
  end for
  $b_{l \cdot \kappa} = v$   # insertion of the input value
  $\sigma = \mathrm{rand}(\mathcal{S}_{l \cdot \kappa})$   # share dissemination following a random permutation
  for $i_{group} = 1$ to $\kappa$ do
    for $i_{proxy} = 1$ to $l$ do
      send[SHARE, $b_{\sigma(i_{group} \cdot l + i_{proxy})}$]($p_{i_{group}, i_{proxy}}$)
    end for
  end for

upon event ⟨ reception | [SHARE, $b$] ⟩ do
  check that sender is a legitimate client
  check that received share is a valid value   # $b \in V$
  $u'' = u'' \oplus b$

procedure local_counting($u''$) :
  for all officemate do
    send[LOCALAGGREGATE,$u''$](officemate)
  end for

upon event ⟨ reception | [LOCALAGGREGATE, $u$] ⟩ do
  check that sender is a legitimate officemate
  check that received aggregate is a possible aggregation of $\kappa \cdot l$ input values
    # $d(u, w_1 \oplus \cdots \oplus w_{\kappa \cdot l}) \leqslant \kappa \cdot l \cdot \delta_V$ where the $w_1 \oplus \cdots \oplus w_{\kappa \cdot l}$ are random values in $V$
  $u' = u' \oplus u$
```

Verifications. The purpose of verifications is to track nodes that deviate from the protocol. This is achieved by leveraging the value attached by the nodes to their reputation. The basic mechanism is that misbehaviors are reported by the participants who discover a faulty node and subsequently tag the latter's profile. The verifications are performed in each phase of the protocol. In the sharing phase, each proxy verifies that the shares received are valid input values. In the second phase, each node checks whether the distance between the individual aggregates sent and some random valid individual aggregate is at most $\kappa \cdot l \cdot \delta_V$. The reason for this is that due to the compatibility of the distance function with the monoid operation, for any $v_1, \ldots, v_k, v'_1, \ldots, v'_k \in V$, we have that

$$d\big(v_1 \oplus \cdots \oplus v_k, v'_1 \oplus \cdots \oplus v'_k\big) \leqslant d\big(v_1, v'_1\big) + \cdots + d\big(v_k, v'_k\big) \leqslant k \cdot \delta_V .$$

The verification in the third phase works as follows: if all the tokens received by a node in a given round (remember that tokens circulate up to three times around the ring) are not the same, then an alarm is raised and the profiles of the involved nodes are tagged. Otherwise, the node broadcasts the unique value of the tokens it received to its officemates. If it is not the case that all values broadcast are equal, again an alarm is raised.

Please cite this article in press as: A. Giurgiu et al., Computing in social networks, Inform. and Comput. (2013), http://dx.doi.org/10.1016/j.ic.2013.11.001

3.4. Correctness

We prove here that AG-S3 satisfies the S^3 conditions for $|B| \leqslant \sqrt{n}/\log^2 n$.

Theorem 2 *(Scalability). The AG-S3 protocol is $\sqrt{}$-scalable.*

Proof. The nodes need to maintain a list of officemates, a list of proxies, a list of clients and a fixed number of elements of U. This amounts to $\mathcal{O}(\sqrt{n} \cdot \log n)$ space complexity as nodes' identifiers can be represented using $\mathcal{O}(\log n)$ bits. A node performs $\mathcal{O}(\sqrt{n})$ '$\oplus$' operations each of them having a complexity in $\mathcal{O}(\log n)$. The message complexity is similarly $\mathcal{O}(\sqrt{n})$ arising from the following components: a node sends $\kappa \cdot l = \mathcal{O}(\log^2 n)$ shares during the sharing phase, $\mathcal{O}(\sqrt{n})$ copies of its individual aggregate in the counting phase, and $\mathcal{O}(\sqrt{n})$ in the forwarding phase. Due to the peer-to-peer nature of the protocol, a node receives as many messages as it sends. □

Theorem 3 *(Accuracy). The AG-S3 protocol is $\sqrt{}$-accurate in the presence of at most $\sqrt{n}/\log^2(n)$ faulty nodes tight to a social network.*

Proof. A faulty node can bias the output of the computation by either sending an invalid set of shares, changing the value of its individual aggregate, or corrupt the aggregate during the forwarding phase. However, a node never misbehaves in a way that this is exposed with certainty (by the verifications presented in the previous section).

Sharing: Not to be detected, a node must send shares in V. Therefore, the distance between the sum of a node's shares and a valid input is at most $\kappa \cdot l \cdot \delta_V$.

Counting: Suppose that a faulty node changes its individual aggregate from $v = v_1 \oplus \cdots \oplus v_{\kappa \cdot l}$ to some value u. When its officemates receive its individual aggregate u they compute the distance between this aggregate and an arbitrary aggregate $w = w_1 \oplus \cdots \oplus w_{\kappa \cdot l}$. If this distance is larger than $\kappa \cdot l \cdot \delta_V$ then the misbehavior is reported. If the distance is within the bound, the triangular inequality yields an upper-bound on the maximum impact: $d(u, v) \leqslant d(u, w) + d(w, v) \leqslant 2\kappa \cdot l \cdot \delta_V$.

Forwarding: To corrupt a token without being detected, the coalition of faulty nodes must *fool* (*i.e.*, make a node decide and forward a corrupted token without raising an alarm) all the non-faulty nodes of a group. Otherwise the corruption is detected by the verification consisting in a node broadcasting the token received to its officemates. To fool a single non-faulty node, all the l tokens it received from its clients (remember that nodes forward tokens only to their proxies in the next group) must be equal. Since nodes have l proxies in the next group, f faulty nodes can fool up to f non-faulty nodes. Assuming that a group contains f non-faulty nodes (and $\sqrt{n} - f$ faulty nodes), then corrupting a token without being detected requires another f faulty nodes in preceding groups. That is a total of $\sqrt{n}$ faulty nodes which cannot happen under the assumption $|B| \leqslant \sqrt{n}/\log^2 n$. To conclude, the local aggregates cannot be corrupted during the forwarding phase.

The impact of a faulty node on the output of the computation is bounded by $3\kappa \cdot l \cdot \delta_v$. We have $|B| \leqslant \sqrt{n}/\log^2 n$, $\kappa = \mathcal{O}(\log n)$, $l = \mathcal{O}(\log n)$ and $\Delta(n) = \Omega(n)$. Putting everything together, we get that the accuracy of Definition 3 is $\mathcal{O}(\sqrt{n}/\log^2 n \cdot \log n \cdot \log n/n) = \mathcal{O}(1/\sqrt{n})$, which concludes the proof. □

Theorem 4 *(Weak probabilistic anonymity). The AG-S3 protocol is weakly probabilistically anonymous.*

Proof. We need to show that, with high probability, there exists a mechanism that for any node p, transforms any trace in such a way that the coalition of faulty nodes receives the same messages, but p has a different input. We first give an outline of the proof.

The transformation mechanism consists in changing the values transmitted between non-faulty nodes, in such a way that any subsequent message sent by non-faulty nodes to the nodes in the coalition does not change. As a result, the coalition receives the same information. Remember that the coalition knows (1) the inputs of all the faulty nodes, (2) the shares received by all the faulty nodes, (3) the individual aggregates of all the faulty nodes' officemates, and (4) possibly all the local aggregates. We focus on weak privacy. Therefore we consider a non-trivial input configuration, with respect to the coalition. That is that at least two non-faulty nodes have different inputs. The basic idea of this mechanism is to *swap* the inputs of two nodes p_1 and p_2, provided that there is a non-compromised group g (*i.e.*, a group with no faulty nodes) that contains proxies of both p_1 and p_2. In this case, we can modify the shares sent by p_1 and p_2 to proxies in g, in such a way that the local aggregate of g is maintained. Since we assume that all nodes in g are non-faulty, the coalition does not have access to information exchanged in g during the counting phase. The coalition only sees what the nodes in g decide to broadcast in the forwarding phase, but that is identical to what is sent in the original trace. To modify the shares of p_1 and p_2, we assume that both send a share containing their own input to some proxies in g. Each of p_1 and p_2 has l proxies in g, so the larger l, the larger the probability that our assumption is true. Then the aforementioned shares of p_1 and p_2 are swapped, resulting a consistent trace, where p_1 and p_2 swapped input values.

In case there is no such common non-compromised group g for p_1 and p_2, we may still find a chain of nodes with endpoints p_1 and p_2, such that two consecutive nodes in the chain can swap input values. The larger κ, the larger the probability that such a chain exists. Afterwards, the nodes can swap shares along the chain, resulting in a consistent configuration where p_1 has as input the old input value of p_2. The rest of the proof concerns making our outline description precise.

Please cite this article in press as: A. Giurgiu et al., Computing in social networks, Inform. and Comput. (2013), http://dx.doi.org/10.1016/j.ic.2013.11.001

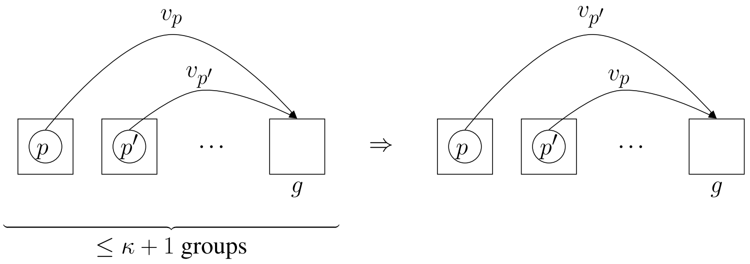


Fig. 4. Illustration of the swapping of two non-faulty nodes' inputs located in two consecutive groups.

Let T be a trace of AG-S3 generated from a non-trivial input configuration $\mathbf{v}$, B be a coalition of faulty nodes ($|B| \leqslant \sqrt{n}/\log^2 n$) and p be a non-faulty node. Since the input is non-trivial, there exists a node p' whose input is different from the input of p in $\mathbf{v}$. We prove that with high probability there exists a trace equivalent to T from the stand-point of the coalition and compatible with an input configuration $\mathbf{v}'$ which is the same as $\mathbf{v}$, except that the inputs of p and p' have been swapped.

We say that a group is compromised if it contains at least one faulty node. The coalition of faulty nodes knows the local aggregates of all the groups, the individual aggregates of the proxies in the compromised groups, the shares they received and their own inputs.

We first prove the following lemma.

Lemma 1. *The probability that in any sequence of $\kappa - 1$ consecutive groups there is at least one non-compromised group is at least $1 - \sqrt{n}(\frac{|B|}{\sqrt{n}})^{\kappa-1}$.*

Proof. This probability is minimized if no two faulty nodes lie in the same group, *i.e.*, there are $|B|$ compromised groups. Fix $\kappa - 1$ consecutive groups. The number of configurations in which these groups are compromised is $\binom{\sqrt{n}-\kappa+1}{|B|-\kappa+1}$. The total number of configurations is $\binom{\sqrt{n}}{|B|}$, so the probability that all the fixed k consecutive groups are compromised is given by the ratio of the two binomial coefficients, which is upper-bounded by $(|B|/\sqrt{n})^{\kappa-1}$. We use the union bound to upper-bound the probability that there is at least one such a sequence of $\kappa - 1$ consecutive compromised groups. There are $\sqrt{n}$ sequences of $\kappa - 1$ consecutive groups, which proves the lemma. □

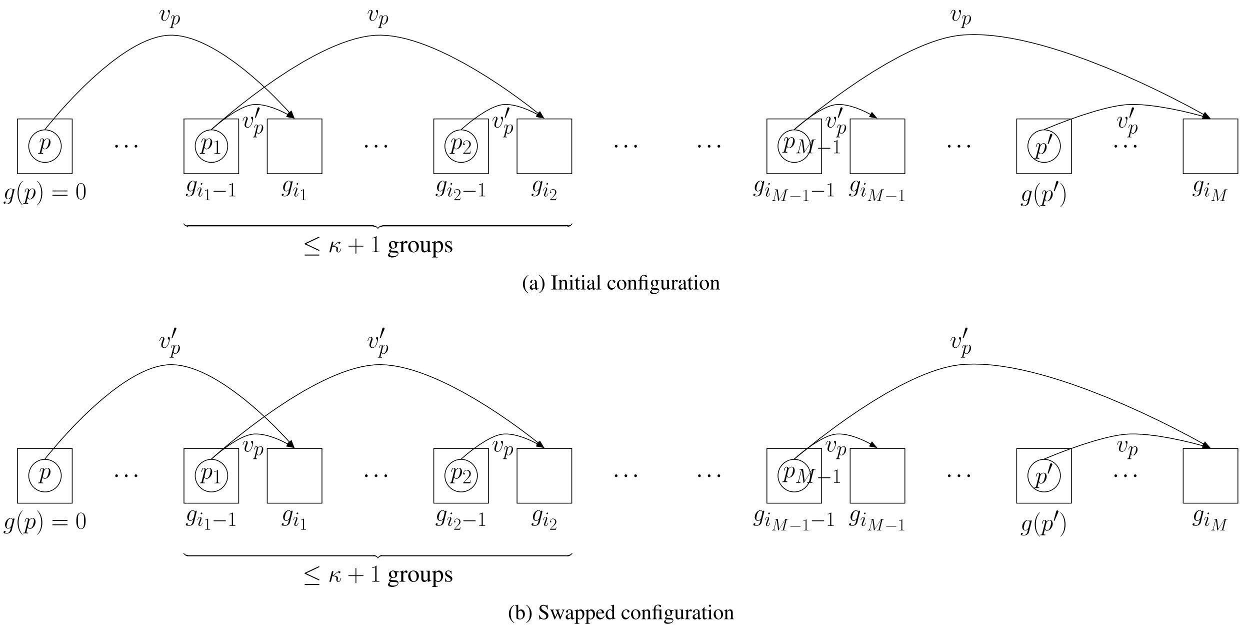
Since $\kappa = 3/2 \cdot \lfloor \log n \rfloor$ and $|B| \leqslant \sqrt{n}/\log^2 n$, we get that the probability of having κ consecutive compromised groups is at most $1/n$.

Lemma 2. *Given $x \in V$, the probability that a node sends at least one share of value x to a proxy situated in a given group, assuming this node has proxies in that group, is at least $1 - 1/n^{5/2}$.*

Proof. The l shares sent to a group by a node are randomly picked from a set of $\kappa \cdot l$ shares in which $(\kappa \cdot l - 1)/2$ are random, $(\kappa \cdot l - 1)/2$ are the inverses of the random shares, and one is the actual input of the node. At least $(l - 1)/2$ of them are independent, and drawn uniformly at random from V. Thus, the probability that a is not one of them is at most $(1 - 1/|V|)^{(l-1)/2}$. Since $(l - 1)/2 = 5/2 \cdot |V| \cdot \lfloor \log n \rfloor$, this probability is upper-bounded by $1/n^{5/2}$, which proves the lemma. □

Consider two non-faulty nodes p and p' in two consecutive groups, such that $v_p \neq v'_p$. Both of them have proxies in the κ groups following their respective groups on the ring. Since they lie in consecutive groups, there are $\kappa - 1$ groups in which both of them have proxies. We assume that one of these groups is not compromised (Lemma 1 gives a lower bound on the probability that this situation occurs). We call this group g. We further assume that at least one share sent by p (resp. p') is equal to its input value v_p (resp. v'_p) (Lemma 2 gives a lower bound on the probability that this situation occurs). Swapping these two shares maintains a valid trace (in which the input of p is v'_p and the input of p' is v_p) that is equivalent to the original trace from the stand-point of the coalition. This guarantees the probabilistic anonymity of non-faulty nodes with different inputs and located in consecutive groups. Fig. 4 illustrates this transformation.

We now extend our result to non-faulty nodes in arbitrary groups. Let $g(\cdot)$ denote the index of a group in which a node lies. Without loss of generality, we assume that $g(p) = 0$. Since we assume that the input $\mathbf{v}$ is not trivial, let p' be a node such that its input v'_p is different from the input of p. Let $i_1, \ldots, i_M$ be a sequence of group indexes such that: (1) group g_{i_m} is non-compromised for all m, (2) $0 < i_1 < \kappa$, (3) $0 < i_{m+1} - i_m < \kappa$ for all m, and (4) $0 < i_M - g(p') < \kappa$. Such a sequence exists with high probability according to Lemma 1. For all $1 \leqslant m < M$, we define p_m as an arbitrary non-faulty node in group g_{i_m-1}. Additionally, we set $p_0 = p$ and $p_M = p'$. Since all nodes have proxies in the κ groups succeeding them, we have that for all $1 \leqslant m \leqslant M$, p_{m-1} and p_m both have proxies in g_{i_m} as depicted in Fig. 5.



(a) Initial configuration

(b) Swapped configuration

Fig. 5. Illustration of the proof of privacy: pairs of shares sent in the same group can be swapped ((a) → (b)) leading to an equivalent trace compatible with a different configuration of inputs.

Using Lemma 2 and using a union bound on the $1 \leqslant m \leqslant M$, we get that the probability that for all $1 \leqslant m \leqslant M$, p_{m-1} sends a share of value v_p to a proxy in g_m and p_m sends a share of value v_p to a proxy in g_m, is at least $1 - 2M/n^{5/2}$. Since M is bounded by the number of groups, namely $\sqrt{n}$, this probability is lower-bounded by $1 - 2/n^2$.

Assuming that this event occurs, we exhibit a trace compatible with a configuration of inputs where the inputs of p and p' are swapped: for all $1 \leqslant m \leqslant M$, the v_p share sent by p_{m-1} to g_{i_m} is replaced by v'_p and the v'_p share sent by p_m to g_{i_m} is replaced by v_p, as illustrated in Fig. 5. This trace is equivalent to T with respect to the coalition B as no share sent to a compromised group is changed and all local aggregates remain the same.

We complete the proof by showing that this trace is indeed compatible with the modified configuration of inputs. In the case of AG-S3, compatible means that the set of shares sent by a node is composed of $(\kappa \cdot l - 1)/2$ values of V, their inverses, and the actual input of the node. For p and p', we only change the value of one share equal to their inputs. Therefore, their set of shares remains compatible with their new inputs. For the other nodes p_m, $0 < m < M$, two of their shares are simply swapped.

We proved that the privacy of *a given* non-faulty node p is preserved with probability at least $1 - 2/n^2$, given that the event of Lemma 1 occurs. Since the probability of this event is large (according to Lemma 1), using Bayes's rule it is clear that $1 - 3/n^2$ is an upper bound on the probability that privacy of a particular node is preserved. Using a union bound over the whole set of at most n non-faulty nodes, we obtain that probabilistic anonymity as defined in Definition 7 is preserved with probability $1 - 2/n$. □

It can be seen from the previous proofs that the proposed solution trades accuracy for privacy. This is captured by parameters κ and l: privacy requires these parameters to be $\mathcal{O}(\log n)$, which entails a higher impact on the output which, in turn, reduces the maximum number of faulty-node the protocol can tolerate. Note that large values of κ and l also decrease scalability. In our case however, the overhead is proportional to $\kappa \cdot l$, which fits into the poly-logarithmic factor of Definition 2.

3.5. *Generalization*

In this sub-section we prove that AG-S3 verifies the conditions of Theorem 1. To do so, we prove that, using a compact representation of multisets, the union of all singleton multisets representing the nodes' inputs is an aggregation function that meets the requirements of AG-S3.

Proof. Let V be the input set of the S^3 candidate to compute, V' be the set of vectors of integers which components are all null except one that is in $\{-1, +1\}$, and U' be the set of vectors of integers of size $|V|$ whose components are in $\{-n \cdot \kappa \cdot l, \dots, n \cdot \kappa \cdot l\}$. We equip U' with the binary operator $\oplus$ defined as:

$$(x_1, \dots, x_{|V|}) \oplus (y_1, \dots, y_{|V|}) = \big((x_1 + y_1)_{/[-n \cdot \kappa \cdot l, n \cdot \kappa \cdot l]}, \dots, (x_{|V|} + y_{|V|})_{/[-n \cdot \kappa \cdot l, n \cdot \kappa \cdot l]}\big),$$

where $(x)_{/[-n\cdot\kappa\cdot l,n\cdot\kappa\cdot l]} = \max(\min(x, -n\cdot\kappa\cdot l), n\cdot\kappa, l)$. In short, $\oplus$ is component-wise addition of vectors of integers where values outside of the interval $\{-n\cdot\kappa\cdot l,\dots,n\cdot\kappa\cdot l\}$ are truncated. The computational complexity of $\oplus$ and $\|.-.\|$ is linear in the size of their operands. At the end of the computation, negative components are set to 0 and components greater than n are truncated. Note that this does not decrease accuracy (w.r.t. $\|.-.\|$) as a valid representation of a multiset has all its components in $\{0,\dots,n\}$.

It can now be seen that, under this formalism, the S^3 candidate $(V', U', \oplus, \|.-.\|)$ corresponding to the computation of the multiset of inputs falls into the conditions of use of AG-S3. Indeed, V is a fixed-size set, closed with respect to the inverse. The set U' is of size $2n\cdot\kappa\cdot l$, which is upper-bounded by a polynomial of n as κ and l are $\mathcal{O}(\log(n))$. Also, the diameter is $2n$ which is an $\Omega(n)$. Finally, the distance measure $\|.-.\|$ is compatible with $\oplus$. □

We have now shown that AG-S3 satisfies the condition of application of Theorem 1, which implies that AG-S3 can be used to compute a wider class of functions, that is Lipschitz continuous functions f on a finite size set such that the diameter of the image space is $\Omega(n)$ and such that f can be computed locally from the compact representation of the multiset of inputs in a $\sqrt{}$-scalable way.

4. Related work

Cryptographic primitives and secure multi-party computation [2–4] allow to compute aggregation functions in a secure way. This however comes at the price of limited scalability (at least linear in the number of nodes) as these protocols usually do not compromise on accuracy, *i.e.*, they compute the exact result. Assuming trust relationships between users of a social network, Vu et al. [12] proposed an improved secret sharing scheme to protect privacy. In that scheme, the actual relationships between nodes are used to determine the trustworthy participants, and the shares are only distributed to those. In contrast, AG-S3 exploits solely the human nature of social networks without making any assumptions on the social relationships themselves.

The population protocol model of Angluin et al. [13] provides a theoretical framework of mobile devices with limited memory, which relates to the scalability requirement of the S^3 problem. The model however can only compute first order formulas in Presburger arithmetic [14] and can tolerate only a constant number of benign failures [15]. The community protocol model [16] relaxes the scalability requirements on the memory sizes of tiny agents which enables powerful computations and Byzantine fault-tolerance. Yet, the model breaks anonymity as agents are assigned unique ids. This illustrates the trade-off between the power and security of a model on one hand and privacy on the other hand. The problem of privacy in population protocols was also tackled in [17]. The sharing scheme of AG-S3 is inspired by the obfuscation mechanism proposed in that paper, namely adding unit noise (+1 or −1) to their inputs, upon a state exchange. Dpol [9], itself also inspired by [17], can be viewed as a restricted form of AG-S3. Dpol is restricted to binary polling: it aggregates values in $\{-1,+1\}$ and it uses a rudimentary secret sharing scheme and overly structure that assume (i) a uniform distribution of inputs, and (ii) a built-in anonymous overlay. These are the two main difficulties of the privacy challenge as defined in the S^3 problem.

Differential privacy [18] and k-anonymity [19] are two common ways to express privacy in the context of distributed computations on sensitive databases. Contrary to AG-S3, where faulty nodes take part in the computation, those techniques aim at protecting the privacy of inputs from an external attacker that queries the database. Differential privacy characterizes the amount of information disclosed by the output by bounding the impact of a single input on the output. It is typically achieved by adding noise to the output. However, as pointed out in [20], differential privacy does not capture the cases of *rare input configurations* due to the multiplicative bounds in its formulation, which is precisely the difficult case we need to address in the S^3 problem, *i.e.*, the case where everybody but one node have the same inputs. The obfuscating technique consisting in adding noise to intermediate results cannot be used directly in the context of S^3 computing as it gives more opportunities to faulty nodes to bias the output of the computation. On the other hand, k-anonymity guarantees that any input value maps to at least k input nodes. In the S^3 problem, privacy can be seen as 2-anonymity with high probability, expressed in a distributed setting. With AG-S3, faulty nodes cannot map any input to a restricted subset of nodes as any two non-faulty nodes can swap their inputs transparently. It thus ensures $n-B$-anonymity with high probability.

5. Conclusion

Social networks now constitute huge platforms on which it is very tempting to perform large scale computations. Yet, such computations are challenging for they require privacy, scalability and accuracy. We leverage the very fact that, in such platforms, behind every node lies a respectable user who cares about her reputation, in order to make the problem tractable. We define what the notion of computation means in that context and propose a protocol that computes a class of regular functions. This is a first step toward understanding what can be computed in a social network and many open questions are left open: What is the maximum number of faulty nodes an S^3 protocol can tolerate? What else besides aggregation functions can be computed in an S^3 manner? Indeed, while this paper exhibited in a constructive way an achievable trade-off, namely $(\sqrt{}, \sqrt{}, \mathit{weak})$ with less than $\sqrt{n}/\log^2 n$ faulty-nodes, the boundaries of the design space are yet to be determined.

Appendix A. Notations

Table A.1
Table of notations.

Symbol	Definition
$\Pi = \{p_1, \ldots, p_n\}$	set of nodes
V	input set
$\mathbf{v} = \{v_p\}_{p \in \Pi}$	input configuration
U	output space
$\oplus$	monoid operation on U
$f : V^* \to U$	function to be computed
$o_p \in U$	output of node p
$d : U \to \mathbb{R}^+$	distance measure on the output space
δ_V	diameter of V (when $V \subseteq U$)
$\Delta(n)$	diameter of the image space $f(V^n)$
B	coalition of faulty nodes ($\lvert B \rvert$ its size)
T	execution trace
d_{ms}	natural distance on multi-sets (size of the symmetric difference)
k	parameter of Lipschitz continuity
κ	number of next groups on the ring (param. of AG-S3)
l	number of proxies in each group (param. of AG-S3)

References

[1] A. Giurgiu, R. Guerraoui, K. Huguenin, A.-M. Kermarrec, Computing in social networks, in: SSS'10: Proceedings of the 12th International Symposium on Stabilization, Safety and Security of Distributed Systems, in: Lect. Notes Comput. Sci., Springer, 2010, pp. 332–346.

[2] J. Benaloh, Secret sharing homomorphisms: keeping shares of a secret secret, in: CRYPTO'86: Proceedings of the 6th Annual International Conference on Advances in Cryptology, in: Lect. Notes Comput. Sci., Springer, 1986, pp. 251–260.

[3] R. Rivest, A. Shamir, Y. Tauman, How to share a secret, Commun. ACM 22 (1979) 612–613.

[4] A. Yao, Protocols for secure computations, in: FOCS'82: Proceedings of the 23rd IEEE Annual Symposium on Foundations of Computer Science, IEEE, 1982, pp. 160–164.

[5] P. Kabus, W.W. Terpstra, M. Cilia, A. Buchmann, Addressing cheating in distributed MMOGs, in: NetGames'05: Proceedings of the 4th Annual Workshop on Network and Systems Support for Games, ACM, 2005, pp. 1–6.

[6] N. Tran, B. Min, J. Li, L. Subramanian, Sybil-resilient online content voting, in: NSDI'09: Proceedings of the 6th USENIX Symposium on Networked Systems Design and Implementation, USENIX Association, 2009, pp. 15–28.

[7] M. Sirivianos, K. Kim, X. Yang, SocialFilter: introducing social trust to collaborative spam mitigation, in: INFOCOM'11: Proceedings of the 30th IEEE International Conference on Computer Communications, IEEE, 2011, pp. 2300–2308.

[8] L. Lamport, R. Shostak, M. Pease, The Byzantine generals problem, ACM Trans. Program. Lang. Syst. 4 (3) (1982) 382–401.

[9] R. Guerraoui, K. Huguenin, A.-M. Kermarrec, M. Monod, Ý. Vigfsson, Decentralized polling with respectable participants, Elsevier J. Parallel Distrib. Comput. (JPDC) 72 (1) (2012) 13–26, http://dx.doi.org/10.1016/j.jpdc.2011.09.003.

[10] Z. Galil, M. Yung, Partitioned encryption and achieving simultaneity by partitioning, Inf. Process. Lett. 26 (2) (1987) 81–88.

[11] I. Gupta, K. Birman, P. Linga, A. Demers, R. van Renesse Kelips, Building an efficient and stable P2P DHT through increased memory and background overhead, in: IPTPS'03: Proceedings of the Second International Workshop on Peer-to-Peer Systems, in: Lect. Notes Comput. Sci., Springer, 2003, pp. 160–169.

[12] L.-H. Vu, K. Aberer, S. Buchegger, A. Datta, Enabling secure secret sharing in distributed online social networks, in: ACSAC'09: Proceedings of the 25th IEEE Annual Computer Security Applications Conference, IEEE, 2009, pp. 419–428.

[13] D. Angluin, J. Aspnes, Z. Diamadi, M.J. Fischer, R. Peralta, Computation in networks of passively mobile finite-state sensors, Distrib. Comput. 4 (18) (2006) 235–253.

[14] D. Angluin, J. Aspnes, D. Eisenstat, E. Ruppert, The computational power of population protocols, Distrib. Comput. 20 (4) (2007) 279–304.

[15] C. Delporte-Gallet, H. Fauconnier, R. Guerraoui, E. Ruppert, When birds die: making population protocols fault-tolerant, in: DCOSS'06: Proceedings of the 2nd International Conference on Distributed Computing in Sensor Systems, in: Lect. Notes Comput. Sci., Springer, 2006, pp. 51–66.

[16] R. Guerraoui, E. Ruppert, Names Trump Malice: tiny mobile agents can tolerate Byzantine failures, in: ICALP'09: Proceedings of the 36th International Colloquium on Automata, Languages and Programming, in: Lect. Notes Comput. Sci., Springer, 2009, pp. 484–495.

[17] C. Delporte-Gallet, H. Fauconnier, R. Guerraoui, E. Ruppert, Secretive birds: privacy in population protocols, in: OPODIS'07: Proceedings of the 11th International Conference on Principles of Distributed Systems, in: Lect. Notes Comput. Sci., Springer, 2007, pp. 329–342.

[18] C. Dwork, Differential privacy, in: ICALP'06: Proceedings of the 33rd International Colloquium on Automata, Languages and Programming, in: Lect. Notes Comput. Sci., Springer, 2006, pp. 1–12.

[19] P. Samarati, Protecting respondents' identities in microdata release, IEEE Trans. Knowl. Data Eng. 13 (6) (2001) 1010–1027.

[20] I. Roy, S. Setty, A. Kilzer, V. Shmatikov, E. Witchel, Airavat: security and privacy for MapReduce, in: NSDI'10: Proceedings of the 7th USENIX Symposium on Networked Systems Design and Implementation, USENIX, 2010, pp. 297–312.