



HAL
open science

Computing in Social Networks

Andrei Giurgiu, Rachid Guerraoui, Kévin Huguenin, Anne-Marie Kermarrec

► **To cite this version:**

Andrei Giurgiu, Rachid Guerraoui, Kévin Huguenin, Anne-Marie Kermarrec. Computing in Social Networks. [Research Report] RR-7295, 2010, pp.12. inria-00492201v1

HAL Id: inria-00492201

<https://inria.hal.science/inria-00492201v1>

Submitted on 6 Jul 2010 (v1), last revised 15 Jul 2010 (v2)

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Computing in Social Networks

Andrei Giurgiu — Rachid Guerraoui — Kévin Huguenin — Anne-Marie Kermarrec

N° 7295

May 2010

— Distributed Systems and Services —



*R*apport
de recherche

Computing in Social Networks

Andrei Giurgiu* , Rachid Guerraoui* , Kévin Huguenin , Anne-Marie Kermarrec

Theme : Distributed Systems and Services
Networks, Systems and Services, Distributed Computing
Équipe-Projet ASAP

Rapport de recherche n° 7295 — May 2010 — 12 pages

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 function $f : V \rightarrow U$ of their inputs in a *scalable* and *secure* way.

Scalability means that the message and computational complexity of the distributed computation should be at most $\mathcal{O}(\sqrt{n} \cdot \text{polylog } n)$, which prevents from using asymmetric cryptography. Security refers to (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 the amount of information on the inputs disclosed by the computation to other nodes.

We present AG-S3, a protocol that S^3 -computes a class of aggregation functions which derive from a commutative monoid operation on U : $f(x_1, \dots, x_n) = x_1 \oplus \dots \oplus x_n$. Key to our protocol is a dedicated overlay structure that enables secret sharing and distributed verifications. The efficiency of verifications relies on our leveraging of the social aspect of the network: nodes can be tagged as suspected to misbehave. Since they care about their reputation, this is used as an incentive for them not to misbehave.

Key-words: Social networks, Privacy, Scalability, Security, Accountability

* Laboratoire de programmation distribuée (LPD), École Polytechnique Fédérale de Lausanne (EPFL)

Calcul distribué dans les réseaux sociaux

Résumé :

Mots-clés :

1 Introduction

The past few years have witnessed an explosion of online social networks and the number of users of such networks is growing exponentially by the day. Facebook for instance boasts more than 400 millions users at the moment. This creates a huge live platform that is leveraged in many ways, from conducting polls about political tendencies to gathering thousands of students in the streets of several French University towns to share an evening drink.

Clearly, social networks constitute appealing platforms to perform large-scale computations. The use of a central authority, while tempting from the point of view of the host provider, poses, besides scalability, important privacy issues. As much as Facebook is successful, privacy flaws keep arising. At the time of writing this paper, a major privacy hole enabled users to view live chats of all their friends, revealing potentially sensitive information. In fact, there is no guarantee that the social network host will not make usage of the personal information of its users, for commercial purposes for example. In 2009, Facebook tried to change its privacy policy to impose new terms of use granting Facebook an eternal ownership of any personal content – even if the users decide to delete their account. This did not get through eventually but highlights 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 their own computation without any central authority. The motivation of this work is to study what distributed computations can be performed in a social network. Our primary contribution is to define the very meaning of *computation* in that setting. We refer to the underlying problem as the S^3 problem: *Scalable Secure Computing in a Social network*. In short, for a set of nodes (users), the problem consists in computing a function of their inputs in a *scalable* and *secure* way. A characteristic of the social network context is that the nodes are in fact users who might not want to reveal their input, nor expose any misbehavior. This concern for reputation determines the extent to which dishonest nodes act: up to the point that their misbehavior remains discrete enough not to be discovered. In short, users do care about their reputation.

Whereas *scalability* characterizes the message and computational complexity of the distributed computation, the *secure* aspect of S^3 accounts for 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 disrupt their input, as well 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 might seem impossible. Clearly, tolerating dishonest players and ensuring privacy calls for cryptographic primitives. Yet, cryptographic schemes, typically used for multi-party computation involve too high a computation overhead, including key distribution mechanisms and encryption, and thus do not scale. Instead, we leverage users' concern for reputation to alleviate the need of cryptographic primitives: nodes verify each other's actions and may report detected faults on their profile in the social network.

Solving the S^3 problem without cryptography is challenging: to ensure privacy, a protocol must ensure that in the case where all the nodes except one have the same inputs, the information obtained by the coalition of faulty nodes cannot know which non-faulty node had a different input. This means that there exist two configurations of inputs that differ for two nodes, which (w.h.p.) lead to the same sequence of messages received by the faulty nodes. This boils down to *swapping* two nodes' inputs transparently (from the standpoint of the faulty nodes). This is even more difficult to achieve when the protocol must be scalable and accurate. The scalability requirement (i.e., each node communicates with a limited number of nodes) makes difficult to find a chain of messages that can be changed 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 have some information about what the inspected node received; on the other hand the more the nodes know about the message 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, the S^3 problem. Secondly, we provide a protocol AG-S3 that solves the problem for a class of aggregation functions, that is, derives from a monoid operation on U : $f(x_1, \dots, x_n) = x_1 \oplus \dots \oplus x_n$. At the core of our protocol lie (1) a structured overlay where nodes are clustered into groups, (2) a secret

sharing scheme that allows the nodes to obfuscate their inputs, and (3) verification procedures which tag the profile of suspected nodes. As such, AG-S3 can be seen as a generalization of Dpol [7]. However, it uses a more sophisticated sharing scheme, which combined with a specific overlay, ensures scalability, privacy and accuracy as defined in S^3 . The privacy guarantees are much stronger than Dpol as it does not rely on any assumption on the inputs of nodes, which makes the problem much more challenging.

It is important to notice that this paper can be viewed as a first general step toward characterizing what can be computed in a distributed way while accounting for the human nature of social networks: nodes can be tagged as suspected and this is used as an incentive for nodes to behave correctly.

2 Problem statement

This section defines the problem of Scalable Secure Computing in a Social network: the S^3 problem. The problem involves a S^3 candidate, namely the function to be computed, and a set of nodes $\Pi = \{p_1, \dots, p_n\}$.

Definition 1 (S^3 candidate). *A S^3 candidate is a quadruple (f, V, U, d) , where V is an arbitrary set, f is a function $f : V^* \rightarrow U$ 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 a S^3 candidate maps the inputs of the nodes to a value in a metric space. The symmetry assumption on f follows from the observation that the output depends on the multiset of inputs but not on their assignation to nodes. For example, a binary poll over Π can be modeled by the S^3 candidate $((v_1, v_2, \dots, v_n) \mapsto v_1 + \dots + v_n, \{-1, +1\}, \mathbb{Z}, (z_1, z_2) \mapsto |z_1 - z_2|)$.

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. They care about their privacy and their reputation: a user wants neither the private information contained in her input to be disclosed nor her faults during the computations, if any, to be exposed. The rationale of our problem specification lies on this reputation concern, which can be leveraged to achieve secure distributed computations. To achieve 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 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”.

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. As such, they are weaker than Byzantine users. To achieve their goal, faulty nodes may collude. As explained above, users care about their reputation and thus do not even want to be suspected of being faulty. More formally, we assume that faulty nodes never behave in such a way that the probability of being suspected (i.e., to appear in \mathcal{F}_p) by a non-faulty node is 1.

In our context, a distributed computation \mathcal{D} 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 , which is a random variable whose value is determined by the random choices made by the nodes during the computation. We now define the desirable properties of a distributed computation in a social network, namely scalability and security. Security here refers to privacy and accuracy.

Scalability means that the computation is able to handle a large number of inputs (i.e., large values of n): consequently, the properties are expressed in the form of asymptotic bounds.

Definition 2 ($\sqrt{\cdot}$ -Scalability). *A distributed computation is said to be $\sqrt{\cdot}$ -scalable if the spatial and computational complexities at each node are $\mathcal{O}(\sqrt{n} \cdot \text{polylog } n)$.*

The intuition behind the logarithmic factor in the asymptotic bound is that operations with the nodes' identifiers and the memory needed to store such identifiers remain within $\mathcal{O}(\log n)$.

The definition of the accuracy of a computation relies on the metric space structure of the output space U : accuracy is given by the distance between the output of the computation and the actual value of the output of f . To render it meaningful, we normalize this distance by the diameter of $f(V^n)$ for a distributed computation over n nodes.

Definition 3 ($\sqrt{\cdot}$ -Accuracy). A distributed computation \mathcal{D} is said to $\sqrt{\cdot}$ -accurately compute a S^3 candidate (f, U, V, d) if:

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

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

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

This definition highlights the importance of specifying the distance measure of a S^3 candidate: providing 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 value of f . Meanwhile, for binary polling for instance, considering the natural distance on relative numbers includes computations for which the error on the tally is negligible when compared to the sample size n (i.e., $\Delta(n) = 2n$).

Privacy characterizes how the information gained by the nodes taking part in the distributed computation can help a coalition of faulty nodes to recover the input of a particular non-faulty node. Intuitively, 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 of a computation. In a perfectly private distributed computation, a coalition of faulty nodes is able to recover the input of a non-faulty node if and only if its input can be inferred from just the output of the computation and the inputs of the faulty nodes. Such configurations of inputs are captured by the notion of *trivial* inputs. An example of such configuration of inputs is the case where all non-faulty nodes taking part in a binary poll have the same input, be it -1 or 1 . Since S^3 candidates are symmetric by definition, a trivial input is a configuration where all nodes start with the same input.

Definition 4 (Trivial input). An element v of V^* is said to be a trivial input for a coalition B if there is a node $p \notin B$ such that for all input configuration v' that coincides with v for all nodes in B , $f(v) = f(v')$ implies $v_p = v'_p$.

In our context, a distributed computation is said to be *private* if the probability of recovering the input of a particular non-faulty node (knowing that it cannot be inferred from the output alone, i.e., the configuration of inputs is non-trivial) decreases as $1/n^\alpha$ for some positive α .

Definition 5 (Realizations). A realization of a distributed computation is a snapshot of a run of the computation. A realization is said to be compatible with an input configuration v if the realization can be obtained from v with a nonzero probability. We say that two realizations are equivalent with respect to a coalition of faulty nodes B if each node in B receives the exact same messages in both realizations.

Definition 6 (Privacy). A distributed computation \mathcal{D} is said to respect privacy if for any coalition of faulty nodes B , for any non-faulty node p , and for any realization D compatible with a non-trivial (w.r.t. B) input configuration v , there exists with high probability a realization D' compatible with an input configuration v' such that (1) D and D' are equivalent w.r.t. B and (2) v and v' differ on the input value of node p .

The intuition behind this definition is that a coalition of faulty nodes cannot distinguish, with high probability, between different runs of the protocols in which non-faulty nodes had different inputs.

Definition 7 (S^3 computation). A distributed computation is said to S^3 compute a S^3 candidate if it is $\sqrt{\cdot}$ -scalable, $\sqrt{\cdot}$ -accurate and private with respect to the candidate.

3 Solving S^3 for aggregation functions

In this section, we focus on a class of aggregation functions and propose a protocol, namely AG-S3, which S^3 -computes such functions for $|B| \leq \sqrt{n}/\log^2 n$ faulty nodes.

3.1 Model

We consider S^3 candidates for which the function f is an aggregation function deriving from an associative binary operation on U : $f(v_1 \dots, v_n) = v_1 \oplus \dots \oplus v_n$. For symmetry reasons, 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

$$d(v_1 \oplus v_2, v'_1 \oplus v'_2) \leq d(v_1, v'_1) + d(v_2, v'_2) . \quad (1)$$

We assume the size of the set of possible inputs to be constant and the size of the set of possible outputs to be polynomial in n implying that any input or output can be represented by $\mathcal{O}(\log n)$ bits. Finally, we assume that V is closed with respect to inverse: 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

We describe the main challenge of S^3 computing, the trade-off between scalability and accuracy on one hand and privacy on the other.

To ensure the scalability requirement, we cluster the nodes into groups of size approximately \sqrt{n} , and require that one node can only send messages to other nodes in a small set of neighboring groups. We introduce two parameters of the protocol, κ and l . A non-faulty 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 can be said for accuracy: intuitively, the larger l and κ are, 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 show that AG-S3 ensures privacy, we need a mechanism which for any node p , transforms any realization (i.e. input values and messages) into another realization, 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 realizations. This proves that the coalition is not able to determine the input value of p . It will become apparent in the proof of privacy that both κ and l need to be large in order to obtain reasonable privacy requirements.

In conclusion, the 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 reasonably ensure the S^3 requirements.

3.3 Protocol

We describe AG-S3 which performs general aggregation in a S^3 manner. AG-S3 is composed of two interleaved components: one computes the aggregation function while the other checks the behavior of other users. Algorithms 1-4 gives a pseudo-code version of the protocol including verifications.

Structure AG-S3 uses a special structure provided by the network, where the n nodes are distributed into groups of size \sqrt{n} . The groups (or *offices*) are placed in 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} officemates and to l proxies in each of the next κ groups on the ring. 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 $i + 1, \dots, i + l \pmod{\sqrt{n}}$ as proxies in each of the next κ groups on the ring. We set $\kappa = 3/2 \cdot \lceil \log n \rceil$ and $l = 5 \cdot \lceil \log n \rceil + 1$. These choices are motivated in 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 random, $(\kappa \cdot l - 1)/2$ are their inverses, and one is the actual input of the node.

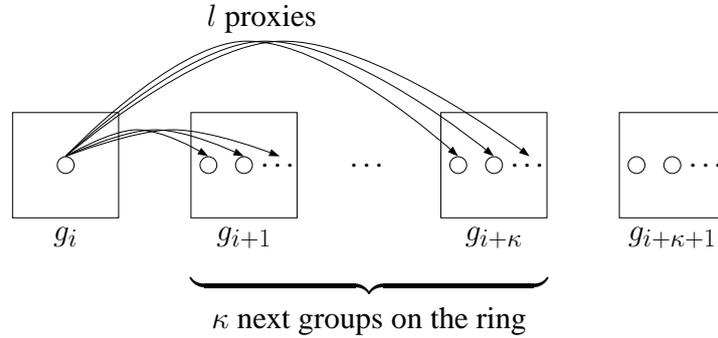


Figure 1: Overview of the overlay

```

1 procedure share_input( v );
2   for i ← 1 to (l · κ - 1)/2 do
3     |   s_i ←rand V   # random values in V;
4     |   s_{i+(l·κ-1)/2} ← ⊖ s_i;
5   s_{l·κ} ← v   # the actual input;
6   σ ←rand S_{l·κ}   # random permutation to distribute the shares;
7   for i_group ← 1 to κ do
8     |   for i_proxy ← 1 to l do
9     |   |   send (SHARE, p_{i_group, i_proxy}, s_{σ(i_group·l+i_proxy)});

```

Algorithm 1: Input sharing

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.

```

1 upon event receive (SHARE, c, s) do
2   Verify c is a client;
3   Verify s is a valid input in V   # s ∈ V;
4   u_ind = u_ind ⊕ s;

```

Algorithm 2: Individual aggregation

```

1 procedure local_count();
2   foreach officemate o do
3     |   send (INDIVIDUAL_AGG, o, u_ind);

```

Algorithm 3: Local aggregate broadcast

```

1 upon event receive (INDIVIDUAL_AGG, o, u) do
2   Verify u is a valid aggregate of κ · l shares;
3   # d(u, v_1 ⊕ ... ⊕ v_{κ·l}) ≤ κ · l · δ_V where v_1 ⊕ ... ⊕ v_{κ·l} are random values in V;
4   u_local ← u_local ⊕ u;

```

Algorithm 4: Local aggregation

In the *forwarding phase*, all local aggregates are made known to all nodes thanks to tokens forwarded along the ring, as explained below. The forwarding phase is bootstrapped by a special group (that can be designated 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 from the next group. 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. 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.

Verifications The purpose of verifications is to ensure that nodes do not 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 $v_1, \dots, v_k, v'_1, \dots, v'_k \in V$, we have that

$$d(v_1 \oplus \dots \oplus v_k, v'_1 \oplus \dots \oplus v'_k) \leq d(v_1, v'_1) + \dots + d(v_k, v'_k) \leq k \cdot \delta_V.$$

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.

3.4 Proofs

In this part, we prove that AG-S3 satisfies the S^3 conditions for $|B| \leq \sqrt{n}/\log^2 n$.

Theorem 1 (Scalability). *The protocol is $\sqrt{\cdot}$ -scalable.*

Proof. The nodes need to maintain a list of officemates, a list of proxies, and a list of clients. This amounts to $\mathcal{O}(\sqrt{n} \cdot \log n)$ space complexity as nodes' identifiers can be represented using $\mathcal{O}(\log n)$ bits. 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. \square

Theorem 2 (Accuracy). *The protocol is $\sqrt{\cdot}$ -accurate.*

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

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 $v = v_1 \oplus \dots \oplus v_{\kappa \cdot l}$ to some value u . When its officemates receive its individual aggregate u they compute the distance between this aggregate and an arbitrary aggregate $w = w_1 \oplus \dots \oplus w_{\kappa \cdot l}$. 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: $d(u, v) \leq d(u, w) + d(w, v) \leq 2\kappa \cdot l \cdot \delta_V$.

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 $|B| \leq \sqrt{n}/\log^2 n$. To conclude, the local aggregates cannot be corrupted during the forwarding phase.

We have $|B| \leq \sqrt{n}/\log^2 n$, $\kappa = \mathcal{O}(\log n)$, $l = \mathcal{O}(\log n)$ and $\Delta(n) = \Omega(n)$. Putting everything together, we get that the accuracy is $\mathcal{O}(1/\sqrt{n})$ which concludes the proof. \square

Theorem 3 (Privacy). *The protocol respects privacy.*

Proof. As mentioned in the *design rationale* section, the theorem is proved if we show that with high probability there exists a mechanism that for any node p , transforms any realization 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 of 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. The basic idea of this mechanism is *swapping* the inputs of two nodes p_1 and p_2 , provided that there is a non-compromised group g (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 realization. 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 is, the larger the probability that our assumption is true. Then the aforementioned shares of p_1 and p_2 are swapped, resulting a consistent realization, 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 is, the larger the probability of existence of such a chain. 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 is concerned with making our intuitive description precise.

Let D be a realization of AG-S3 compatible with a non-trivial input v , B be a coalition of faulty nodes 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 v , and we prove that with high probability there exists a realization equivalent to D compatible with an input configuration v' which is the same as v , except that the inputs of p and p' have been swapped.

We call a group 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 $1 - \sqrt{n} \left(\frac{|B|}{\sqrt{n}}\right)^{\kappa-1}$.*

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 sequence of $\kappa - 1$ consecutive compromised groups. There are \sqrt{n} sequences of $\kappa - 1$ consecutive groups, which proves the lemma. \square

Since $\kappa = 3/2 \cdot \lfloor \log n \rfloor$ and $|B| \leq \sqrt{n}/\log^2 n$, 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 $1 - 1/n^3$.*

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 their inverses, 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 $(l - 1)/2 = 5/2 \cdot |V| \cdot \lceil \log n \rceil$, this probability is upper-bounded by $1/n^{5/2}$, which proves the lemma. \square

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 v is not trivial, let p' be a node such that its input v'_p is different from the input of p , i.e., v_p . Let i_1, \dots, i_M be a sequence of 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 $1 \leq m < M$, 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 $1 \leq m \leq M$, p_{m-1} and p_m both have proxies in g_{i_m} as depicted in Figure 2.

Using Lemma 2 and using an union bound on the $1 \leq m \leq M$, we get that the probability that for all $1 \leq m \leq M$, 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 realization compatible with a configuration of inputs where the inputs of p and p' are swapped: for all $1 \leq m \leq M$, 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 Figure 2. This realization is equivalent to D with respect to the coalition B as no share sent to a compromised group is changed and all local aggregates remain the same.

To complete the proof we show that this realization 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 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 node nodes, we obtain that privacy as defined in Definition 6 is preserved with probability $1 - 2/n$. \square

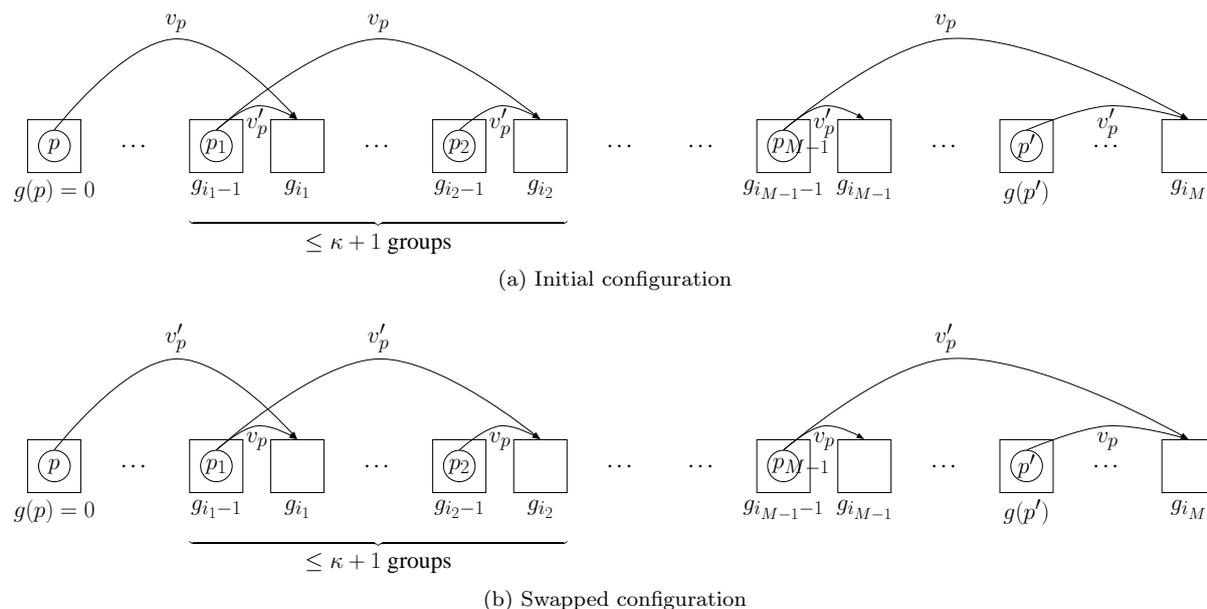


Figure 2: Illustration of the proof of privacy: pairs of shares sent in the same group can be swapped ((a) \rightarrow (b)) leading to an equivalent realization compatible with a different configuration of inputs.

4 Related work

Cryptographic protocols and secure multi-party computation [3,9,13] allow to compute, in a secure way, aggregation functions such as addition and multiplication. However, this comes at the price of non-scalability and thus cannot solve the S^3 problem as defined in this paper. Assuming trust relationships between users of a social network, Vu *et al.* [12] proposed an improved secret sharing scheme to protect privacy. AG-S3 exploits only the human nature of social network but not the social relationships themselves.

The mobile population protocols model proposed by Angluin *et al.* [1] gives a theoretical framework of a computation by tiny mobile agents with limited memory, which relates to the scalability requirement of the S^3 problem. It has been proved that in this model, a population can compute only first order formulas in Presburger arithmetic [2] and can tolerate only a constant number of benign failures [4]. The community protocols relax the scalability requirements on the memory sizes of tiny agents which enables much powerful computations and Byzantine fault-tolerance [8]. However it breaks anonymity as agents are assigned unique ids. This illustrates the trade-off between the power and security of a model on the one hand and privacy on the other hand. The problem of privacy in population protocols is tackled in [5]. The sharing scheme of AG-S3 is inspired from the obfuscation mechanism proposed in this paper, namely adding unit noise (+1 or -1) to their inputs, upon a state exchange.

Differential privacy [6] and k -anonymity [11] are two common ways to achieve 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 attacker that queries the database. Differential privacy characterizes the amount of information disclosed by the output on the inputs by bounding the impact of a single input on the output. It is usually achieved by adding noise to the output. However, as pointed out in [10], differential privacy does not capture the case of *rare input configuration* due to the multiplicative bounds in its formulation, which is precisely the difficult case we want 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 intermediary result cannot be used in the context of S^3 computing. The granularity of noise may indeed be high if elements of V are far away. In addition, it gives more opportunities to faulty nodes to bias the output of the computation. 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. With AG-S3, faulty nodes cannot map any input to a restricted subset of nodes as any two nonfaulty nodes can swap their inputs transparently. Therefore, it ensures $n - B$ -anonymity with high probability.

Dpol [7] is the piece of work that relates the most to AG-S3. Dpol allows to conduct binary polls by aggregating values in $\{-1, +1\}$. Similarly to AG-S3 it uses groups placed on a ring and secret sharing. Beyond the fact that AG-S3 can achieve not only addition but a wide class of aggregation functions, it gives stronger privacy guarantees with weaker assumptions thanks to its more sophisticated secret sharing scheme and overlay structure. Effectively, in Dpol privacy is ensured with high probability assuming (i) a uniform distribution of inputs, and (ii) that faulty nodes do not know the overlay, which are the two main difficulties of the privacy challenge as defined in the S^3 problem.

5 Conclusion

Social networks form a huge platform that can be leveraged to perform distributed computations among users. The resulting platform has this specificity that behind each node stands a user who cares about her reputation. This somehow acts as an incentive to users to limit their misbehavior. In this paper, we leveraged this characteristic and addressed the problem of secure and scalable computation over social networks.

We clearly define the problem and define scalability and security in this context. We then propose a protocol addressing a class of aggregate functions and prove that it is both secure and scalable. This is a first step towards the characterization of what can be computed in a distributed way over social networks, leveraging the social nature of the considered context.

References

- [1] D. Angluin, J. Aspnes, Z. Diamadi, M. J. Fischer, and R. Peralta. Computation in Networks of Passively Mobile Finite-state Sensors. *Distributed Computing*, 4:235–253, 2006.
- [2] D. Angluin, J. Aspnes, D. Eisenstat, and E. Ruppert. The Computational Power of Population Protocols. *Distributed Computing*, 20:279–304, 2007.
- [3] J. Benaloh. Secret Sharing Homomorphisms: Keeping Shares of a Secret Secret. In *CRYPTO*, pages 251–260, 1986.
- [4] C. Delporte-Gallet, H. Fauconnier, R. Guerraoui, and E. Ruppert. When Birds Die: Making Population Protocols Fault-tolerant. In *DCOSS*, pages 51–66, 2006.
- [5] C. Delporte-Gallet, H. Fauconnier, R. Guerraoui, and E. Ruppert. Secretive Birds: Privacy in Population Protocols. In *OPODIS*, pages 329–342, 2007.
- [6] C. Dwork. Differential Privacy. In *ICALP*, pages 1–12, 2006.
- [7] R. Guerraoui, K. Huguenin, A.-M. Kermarrec, and M. Monod. Decentralized Polling with Respectable Participants. In *OPODIS*, pages 144–158, 2009.
- [8] R. Guerraoui and E. Ruppert. Names Trump Malice: Tiny Mobile Agents Can Tolerate Byzantine Failures. In *ICALP*, pages 484–495, 2009.
- [9] R. Rivest, A. Shamir, and Y. Tauman. How to Share a Secret. *CACM*, 22:612–613, 1979.
- [10] I. Roy, S. T. Setty, A. Kilzer, V. Shmatikov, and E. Witchel. Airavat: Security and Privacy for MapReduce. In *NSDI*, 2010.
- [11] P. Samarati. Protecting Respondents’ Identities in Microdata Release. *TKDE*, 13:1010–1027, 2001.
- [12] L.-H. Vu, K. Aberer, S. Buchegger, and A. Datta. Enabling secure secret sharing in distributed online social networks. In *ACSAC*, pages 419–428, 2009.
- [13] A. C. Yao. Protocols for Secure Computations. In *FOCS*, pages 160–164, 1982.



Centre de recherche INRIA Rennes – Bretagne Atlantique
IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)

Centre de recherche INRIA Bordeaux – Sud Ouest : Domaine Universitaire - 351, cours de la Libération - 33405 Talence Cedex
Centre de recherche INRIA Grenoble – Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier
Centre de recherche INRIA Lille – Nord Europe : Parc Scientifique de la Haute Borne - 40, avenue Halley - 59650 Villeneuve d'Ascq
Centre de recherche INRIA Nancy – Grand Est : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex
Centre de recherche INRIA Paris – Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex
Centre de recherche INRIA Saclay – Île-de-France : Parc Orsay Université - ZAC des Vignes : 4, rue Jacques Monod - 91893 Orsay Cedex
Centre de recherche INRIA Sophia Antipolis – Méditerranée : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
<http://www.inria.fr>
ISSN 0249-6399