Robust Privacy-Preserving Gossip Averaging

Amaury Bouchra Pilet, Davide Frey, François Taïani

Robust Privacy-Preserving Gossip Averaging

Amaury Bouchra Pilet[1,2], Davide Frey[1], and Francois Taiani[1]

[1] Univ Rennes, Inria, CNRS, IRISA
[2] École Normale Supérieure « Ulm »

Abstract Decentralized solutions are emerging as promising candidates to overcome the privacy risks associated with centralized data services. Such solutions suffer however from their own range of privacy vulnerabilities, arising from untrusted and malicious peers. In this paper, we consider the emblematic problem of privacy-preserving decentralized averaging, and propose a novel gossip protocol that exchanges noise for several rounds before starting to exchange actual data. This makes it hard for an honest but curious attacker to know whether a user is transmitting noise or actual data. Our protocol and analysis do not assume a lock-step execution, and demonstrate improved resilience to colluding attackers. We prove the correctness of this protocol as well as several privacy results. Finally, we provide simulation results about the efficiency of our averaging protocol.

1 Introduction

The recent evolution of applications like the Internet of Things has fostered interest in protocols that enable large networks of devices to perform collaborative computations on their own data. For these protocols, privacy protection acquires paramount importance, since the data being processed may be personal or otherwise confidential, e.g. location or medical data. While it is possible to simply centralize data processing, centralization raises privacy and durability issues. The provider of a centralized service has access to the personal data of all users and may cut off the service at any time. To prevent these issues, several authors have proposed decentralized solutions to the problem of data aggregation [8,20], including peer-to-peer protocols that compute the average of values initially held by individual peers, an important topic in a range of statistical and machine-learning applications [7].

These peer-to-peer solutions remove the need for a central server, and the risk of being spied by the server's operator or by third parties that may obtain access to server-side data. While this goes in the direction of privacy protection, these algorithms also have serious disadvantages for privacy. They require users to send information to unknown peers, which may allow not only big companies or governmental agencies, but also criminal organizations or "curious" people to access personal data rather easily, without having to compromise heavily secured servers and communications.

In order to overcome this central weakness of peer-to-peer averaging protocols, several algorithms have been proposed that allow gossip averaging while

protecting their users' privacy [1,2,8,26]. These algorithms unfortunately suffer from a number of limitations that expose them to some eavesdropping attacks [1], constrain how peers must coordinate their exchanges [8,23,26], or require the use of costly cryptographic primitives such as homomorphic encryption [19].

In this paper, we propose to overcome these limitations with a novel peer-to-peer protocol for decentralized averaging. Following earlier solutions [1,26,28], our approach injects randomized values into the averaging process while ensuring deterministic convergence to the exact wanted value. In contrast to earlier attempts, the successive values exposed by an individual node in our protocol's early stage are independent of this node's initial value. Contrary to [26], we also do not assume a fully lock-step execution model to implement our protocol and perform our analysis.

Our design relies on a random peer sampling (RPS) service [21]. For our algorithm to attain its goals in a context where the RPS service itself may be attacked, this RPS service need to be resilient to attacks, especially attacks trying to isolate a specific peer, surrounding it by malicious peers. One example of such an attack-resilient RPS protocol is Brahms [4].

Overall, the random injection we present combined with random peer sampling removes the need for peers to tightly coordinate their actions, allows peers to protect their privacy without having to make explicit privacy protection requests to others, and improves resilience to attacks involving the use of numerous malicious peers controlled by an attacker. We evaluate our protocol by first proving its correctness as well as several privacy properties. We present simulation results in combination with two different peer sampling protocols.

2 System Model and Problem

System model. We consider an asynchronous decentralized system consisting of a large number of peers $\{p_1, ..., p_i, ..., p_N\}$ that can communicate through message passing. We use the terms "node" and "peer" interchangeably throughout this paper. We assume the network is reliable (messages do not get lost), but we do not make any assumption regarding the synchrony of the network or of the execution at different peers. Messages may take an arbitrary (albeit finite) time to arrive, while the protocol's execution evolves independently at each peer. Peers synchronize only in pairs for the duration of a message exchange. A node involved in a message exchange with another node simply waits for a response or for a failure-detection timeout before engaging in other exchanges.

We assume peers have access to a random peer sampling service (RPS for short) [21] that provides each individual peer with a sample stream of other peers present in the network. For our analysis, we assume this RPS protocol is resilient to attempts to bias its results by individual peers [4].

The private averaging problem Each peer p_i possesses a local initial value val_i and wants to compute the average value of all the peers in the system $\frac{1}{N}\sum_i val_i$, while giving other peers as little information as possible regarding its own local

```
1  Function gAvg(val):
2      while true do
3          peer ← randomPeer()
4          sendTo(peer, val)
5          rcv ← recvFrom(peer)
6          val ← (val+rcv)/2
7  Function answer():
8      rcv ← recvFrom(peer)
9      sendTo(peer, val)
10     val ← (val+rcv)/2
```

Algorithm 1: Non-private Gossip Averaging

value. We consider honest but curious attackers in the sense that they observe exchanged values, but they do not inject fake values or try to prevent the algorithm from computing a correct result.

We consider the two kinds of attackers defined in [1], possibly coexisting in a single entity. *Edge attackers* eavesdrop on data exchanged by other peers; they may, for example, try to obtain a user's value by retrieving all the values s/he exchanges with other users. *Node attackers* use a set of peers under their control to get information from other peers.

Algorithm 1 [20] describes the classical approach to decentralized averaging (which does not protect privacy). Each peer starts with an initial value. Pairs of peers regularly exchange their local values (lines 4-5) and replace them with their average (line 6)—the function ANSWER is invoked when receiving a message sent using SENDTO. While this method ensures eventual convergence of all values to their average, provided that the network's graph is connected, peers have to expose their local values to potential strangers, thus raising critical privacy issues if this value is sensitive and other peers cannot be fully trusted. Our private-averaging problem consists in computing the same average while hiding initial values from edge and node attackers.

3 Privacy-Preserving Averaging

We address private averaging with a protocol that uses random values to protect the initial values of peers. Unlike some existing work [26], our protocol does not require synchronous lock-step rounds. Moreover, a peer can protect its value without any of its neighbors doing so and without explicitly informing anyone. In particular, we can have a mix of peers using privacy protection and peers not using it.

3.1 The Algorithm

The solution we proposed is described in Algorithm 2. (As in Algorithm 1 the function ANSWER is called upon receiving a message.)

```
1  Function privGAvg(val, privLvl):
2      err ← 0
3      for i = 0 to i = privLvl − 1 do
4          peer ← randomPeer()
5          fakeVal ← rand()
6          err+ = val − fakeVal
7          sendTo(peer, fakeVal)
8          rcv ← recvFrom(peer)
9          val ← (fakeVal+rcv)/2
10     val+ = err
11     while true do
12         peer ← randomPeer()
13         sendTo(peer, val)
14         rcv ← recvFrom(peer)
15         val ← (val+rcv)/2
16 Function answer():
17     rcv ← recvFrom(peer)
18     if i < privLvl then
19         fakeVal ← rand()
20         err+ = val − fakeVal
21         sendTo(peer, fakeVal)
22         val ← (fakeVal+rcv)/2
23     else
24         sendTo(peer, val)
25         val ← (val+rcv)/2
```

Algorithm 2: Private Gossip Averaging

Our proposal consists in adding a simple *privacy-generation phase* to the averaging algorithm. The privacy-generation phase (lines 2-10) behaves exactly in the same way as the averaging protocol. However, a peer performing this phase sends a random value *fakeVal* (line 7) instead of its current value to the selected peer. The difference between its current value and the sent random value being accumulated in a local error variable *err* (line 6).

After having initiated and completed the desired number of random exchanges (determined by the protocol parameter *privLvl* at line 3), the peer sums its accumulated error variable and its original value (line 10), and then it continues with the original averaging protocol by using this sum as its value (lines 11-15). Furthermore, all communication between peers is encrypted using standard techniques. So we assume that an eavesdropper may only know the time, the sender and the receiver of any communication, not its content.

Different termination conditions can be used with this algorithm: time based, communication-round based, value-change based, etc. The best termination condition being dependent on the considered use-case, and this paper being about private gossip averaging in general, we do not suggest any specific condition for our algorithm.

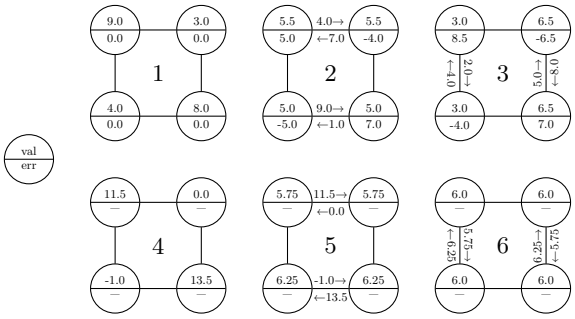


Figure 1: Four peers executing our algorithm

An example of the execution of the algorithm is shown in Figure 1 on a toy example of four peers with $privLvl = 1$, assuming a static synchronous network for simplicity's sake. Each peer is represented by a circle with two numbers. The top number is the current value *val* held by the peer: its initial value at the start of the protocol (Step 1, top-left corner), which should converge arbitrarily close to the network's average (here 6) as the protocol progresses. (In this particular example, peers do converge to the exact average value at Step 6, bottom-right corner.) The bottom number if the error *err* progressively accumulated by each peer.

During the first two rounds (shown at Steps 2 and 3), the peers execute the privacy-generation phase of Algorithm 2. During this phase, peers exchange fully random values, while accumulating the resulting error in their local *err* variable. At Step 4, each peer executes line 10 of Algorithm 2 and corrects its initial value *val* with the error accumulated so far. Note how at each step the overall average value of the network remains 6. Finally, at Steps 5 and 6, our protocol executes a standard decentralized averaging protocol with the corrected values.

3.2 Peer-Sampling Adjustments

As mentioned above, we assume a Byzantine-resilient peer sampling protocol [4]. This ensures that peers can select their communication partner from a uniformly random sample of the network. However, a malicious peer could still try to contact and exchange information with a target peer by bypassing the peer sampling mechanism. For this reason, we introduce an adjustment to connection establishment. When a peer receives a connection request from another one, A, it never answers directly. Rather, it waits for another peer B's contact and forwards to it the information received from A. Then it replies to A with the response received from B. This simple mechanism makes it difficult for an attacker to target a peer in order to monitor its exchanges.

4 Evaluation

We evaluate the performance of our averaging algorithm by means of a theoretical analysis and an experimental evaluation of its performances in conjunction with different peer sampling protocols.

4.1 Averaging Correctness

We start by proving that, in a classical gossip averaging algorithm as proposed in [20], if some of the values exchanged by peers are replaced by random values and, if later, an appropriate correction is done on the value of peers, then, the algorithm will converge the same way as without these operations. Let us consider a classical theoretical continuous-time model of gossip averaging where each peer is a vertex of a complete graph $K(V)$ (which we use to capture the RPS protocol we rely on). For conciseness, we note in the following $x_i(t)$ the value of the variable val_i of peer p_i at time t, and $y_{i,k}$ is the k^{th} value *fakeVal* used by peer p_i.

Theorem 1 (Correctness Theorem) *Let $p_i \in \{p_0, ..., p_k\} \subseteq V$ be a peer, and let us replace its value, $x_i(t)$, by the values $y_{i,0}; ...; y_{i,r_i}$ at times $t_{i,0}; ...; t_{i,r_i}$ (i.e. $\forall_{0 \leqslant i \leqslant k} x_i(t_{i,j}+\varepsilon) = y_{i,j}$). If we later add (at time t_*) the value $\sum_{j=0}^{r_i}(x_i(t_{i,j})-y_{i,j})$ to $x_i(t_*)$ (i.e. $\forall_{0 \leqslant i \leqslant k} x_i(t_* + \varepsilon) = x_i(t_*) + \sum_{j=0}^{r_i}(x_i(t_{i,j}) - y_{i,j})$), then, executing a classical gossip averaging algorithm on this graph will ultimately make every vertex's value converge to the average of all initial values.*

Proof. Let us consider a single peer p_i and a single replacement occurring at time $t_{i,j}, j \in [0, r_i]$. In this transformation, we remove $x_i(t) - y_{i,j}$ from the value of x_i. This modifies the average of all values into $avg(t+\varepsilon) = avg(t) - \frac{x_i(t)-y_{i,j}}{|V|}$. Since averaging operations have no effect on the average value, if we later, at time t', add $x_i(t) - y_{i,j}$ to the value of any peer, we restore the original value of avg. Iterating this reasoning over all values of i and j proves the theorem. □

Now we know that we can replace the value of a peer by a random value an arbitrary number of times. The averaging algorithm will still work, provided that we later add to the value of our peer the sum of all the differences between the value our peer had at the time of replacement and the random value. The correctness of our algorithm follows trivially.

Property 1 (Correctness property) *If the Gossip Privacy Protector algorithm is applied by any number of peers at the start of a classical gossip averaging algorithm, this does not change the value to which all peers will converge.*

The proof is a simple application of our Correctness Theorem.

4.2 Attack Resilience

We now show that our protocol has good privacy properties. We start by observing that by definition, all the random values sent by peers are independent of each other and of the original value of the peer. We then distinguish two types of attack: direct and indirect. Throughout the analysis, we let l (instead of *privLvl*) denote the privacy level chosen by the considered peer. In addition, we let $\tau = \frac{\#\text{corrupted peers}}{\#\text{peers}}$ represent the fraction of corrupted peers. Other than that, we use the same notation as in the algorithm (with time indexes added).

Direct Attacks In direct attacks, the attacker tries to learn the value of a peer by communicating directly with it. We prove two properties in this context.

Property 2 (Deterministic Privacy Property)
An attacker needs to get all the random values sent to compute the exact original value of the peer.

Proof. We will use the same notation as in the algorithm (with time indexes added, and l instead of *privLvl*). The first non-random value is exactly the value exchanged at the l-th round, val_l.

$$val_l = val_{l-1} + err_{l-1} = \frac{fakeVal_{l-1} + rcv_{l-1}}{2} + \sum_{i=0}^{l-1}(val_i - fakeVal_i)$$

$$val_l = \frac{fakeVal_{l-1} + rcv_{l-1}}{2} + \sum_{i=1}^{l-1}\left(\frac{fakeVal_{i-1} + rcv_{i-1}}{2} - fakeVal_i\right) + val_0 - fakeVal_0$$

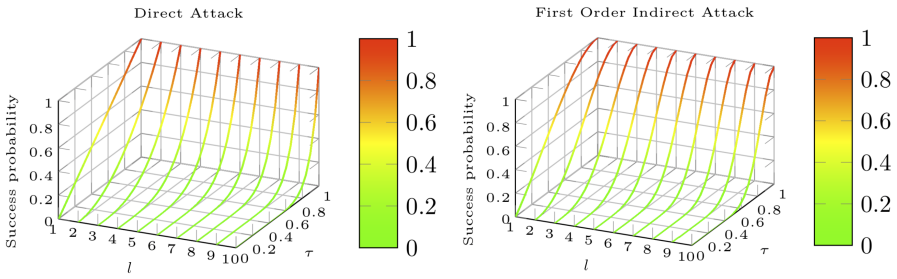
$$val_l = \sum_{i=0}^{l-1}\left(\frac{rcv_i - fakeVal_i}{2}\right) + val_0 \implies val_0 = val_l - \sum_{i=0}^{l-1}\left(\frac{rcv_i - fakeVal_i}{2}\right)$$

We see that computing the original value from the first non-random value requires knowledge of all the random values (and of the answers from contacted peers). All later non-random values having even more noise from other peers and no non-random values being transmitted before, this proves the property. □

Property 3 (Probabilistic Privacy Property)
If the attacker lacks k random values (but not necessarily the associated answers), the best s/he can do is take their expected values (if known), with a level of uncertainty at least as large as that of guessing the value of $\frac{\sum_{i=0}^{k} fakeVal_i}{2}$.

Proof. Since peers generate independent random values, it is impossible to guess anything from them, except by using their relation to val_l.
Since $val_0 = val_l - \sum_{i=0}^{l-1}\left(\frac{rcv_i - fakeVal_i}{2}\right)$, if the attacker lacks k values, s/he has to guess them without any hint. Therefore, the best s/he can do consists in taking their expected values and nothing will give her a lower level of uncertainty than the natural level of uncertainty of these random variables. □

From the Deterministic Privacy Property, we can derive that, if the peer sampling is effectively random, the chance that an attacker will get the exact original value of a peer is $\leqslant \tau^l$. See Figure 2a for a plot.



(a) Direct attack (b) First Order Indirect attack

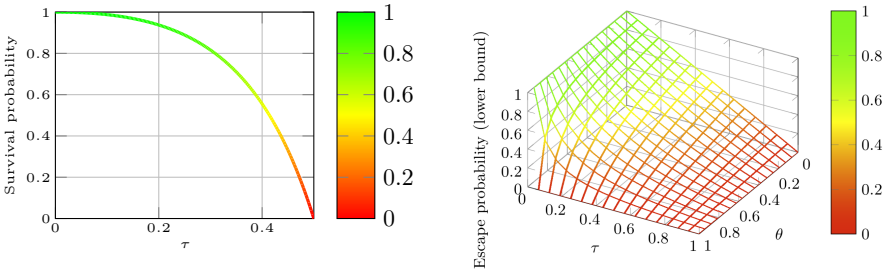
Figure 2: Success probabilities for direct and indirect attacks (upper bounds). l: privacy level, τ: fraction of corrupted peers

Indirect Attacks In indirect attacks, the attacker exploits information exchanged between the target and other peers. For example, the attacker may guess the value of a peer p by observing that p communicates with another peer q whose value changes from val'_q to val''_q as a result of the exchange. This is only possible if the attacker is both a node and an edge attacker. The attacker requires fake peers to obtain the values of nodes, and eavesdroppers to know which values s/he needs. Also, note that the attacker needs to get values older than the ones s/he wants to obtain. Moreover, the attack will only work if all involved peers, but the target, have ended their privacy-generation phases since values sent by a peer generating privacy are all independent of each other. This implies that, if all peers start the process at the same time, it is unlikely that these attacks will be possible. Except for the very simple case of spying the target peer's neighbors, this kind of attack will probably be very impractical for most, if not all, cases on a real network.

Indirect attacks rely on the following principle: if peer i exchanges values with peer j at time t, it is possible to compute the exchanged values by knowing $v_j(t)$ (the value sent by j) and $v_j(t+1) = \frac{v_i(t)+v_j(t)}{2}$. This attack can be iterated: if the attacker lacks one, or both, of $v_j(t)$ and $v_j(t+1)$, s/he may get it using the same attack. We call this a Higher Order Indirect Attack, the base case being a First Order Indirect Attack.

First order indirect attacks give the attacker a second chance to get a value with respect to a direct attack, but require two contacts instead of one. This increases the upper bound of the probability of an exact evaluation from $\leqslant \tau^l$ to $\leqslant (\tau + (1-\tau)\tau^2))^l = (\tau + \tau^2 - \tau^3))^l$. See Figure 2b for a plot.

Higher-order indirect attacks are very unlikely in practice. But from a theoretical point of view, they can be very powerful. We will prove two results showing the limitations of these attacks. We consider a node-edge attacker which is a universal eavesdropper (s/he can see all messages). The attacker may take an arbitrarily long (finite) time to get the desired information and may have to use arbitrarily old information. We state our first theorem under Assump-



(a) Survival probability (lower bound) given by our HO Survival Theorem

(b) Escape probability (lower bound) given by our Higher Order Escape Theorem

Figure 3: Lower bounds on survival and escape probabilities. τ: fraction of corrupted peers, θ fraction or unsafe edges

tion 1, which holds with high probability in the presence of a uniform random peer sampling over an infinite set of peers. The theorem states that as long as the assumption holds, the target can survive a higher-order indirect attack from a universal eavesdropper. We therefore refer to it as *Higher Order Survival Theorem*. We plot the theorem result in Figure 3a.

Assumption 1 *There is neither a pair of consecutive exchanges between the same two peers, nor a pair of exchanges with one (or both) peer(s) in common separated by only one exchange.*

Theorem 2 (Higher Order Survival Theorem)
As long as Assumption 1 holds, and $\tau < 1/2$, a universal attacker will never obtain the value of the target peer with probability $\geqslant 1 - \frac{1-2\tau(1-\tau)-\sqrt{1-4\tau(1-\tau)}}{2(1-\tau)^2}$.

Proof. We model the tree of exchanges induced by the higher order attack by a Galton–Watson process. The number of descendants of an individual is the number of exchanges made just before and after the exchange corresponding to the individual which are not made with a corrupted peer. This gives the following probabilities: $p_0 = \tau^2$, $p_1 = 2\tau(1-\tau)$, $p_2 = (1-\tau)^2$. The average number of descendants is $m = p_1 + 2p_2 = 2\tau(1-\tau) + 2(1-\tau)^2$. Let us analyze the branching process' behavior as a function of τ.

$$m = 2\tau(1-\tau) + 2(1-\tau)^2 = 2\tau - 2\tau^2 + 2 - 4\tau + 2\tau^2 = 2 - 2\tau$$

So $m = 1 \Leftrightarrow 2\tau = 1 \Leftrightarrow \tau = 1/2$. Since $p_1 = 2\tau(1-\tau) = 2\,1/2^2 = 1/2 < 1$, the probability of extinction in the critical case is 1.
For the super-critical case, $m > 1$, let us analyze the generating function.

$$\varphi(s) = \sum_{n \geqslant 0} p_n s^n = p_2 s^2 + p_1 s + p_0 = (1-\tau)^2 s^2 + 2\tau(1-\tau)s + \tau^2$$

To solve $\varphi(s) = s$, we search for the roots r of the polynomial $\varphi(s) - s$.

$$\varphi(s) - s = (1-\tau)^2 s^2 + (2\tau(1-\tau) - 1)s + \tau^2$$

$$r = \frac{1 - 2\tau(1-\tau) \pm \sqrt{1 - 4\tau(1-\tau)}}{2(1-\tau)^2}, (0 \leqslant \tau < 1/2, 1 - 4\tau(1-\tau) > 0)$$

Of the two solutions, the lower one living in $[0, 1]$ is always $r = \frac{1-2\tau(1-\tau)-\sqrt{1-4\tau(1-\tau)}}{2(1-\tau)^2}$ (the other being always 1 for $0 \leqslant \tau < 1/2$). The probability of extinction (the attacker obtains the information) in the super-critical case is $\frac{1-2\tau(1-\tau)-\sqrt{1-4\tau(1-\tau)}}{2(1-\tau)^2}$, so, the probability of survival (the attacker does not obtain the information) is: $1 - \frac{1-2\tau(1-\tau)-\sqrt{1-4\tau(1-\tau)}}{2(1-\tau)^2}$. □

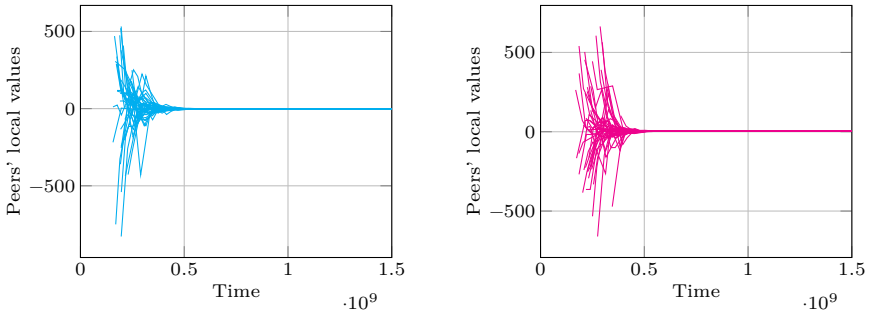
The above theorem relies on a major assumption (Assumption 1) and does not provide anything for $\tau \geqslant 1/2$. We therefore introduce a second theorem that does not have these limitations but that is not applicable to universal eavesdroppers. Let $\theta = \frac{\#\text{unsafe edges}}{\#\text{edges}}$ in the sub-graph from which all corrupted peers have been removed. We assume $0 < \tau < 1$ (if not, there is either no attacker or the attacker controls the whole network) and $0 < \theta < 1$ (if not, we have either a universal eavesdropper or no eavesdropper at all). In this case we have no assumption that needs to remain valid for an extended period. Rather, the theorem states the probability that the target will escape from the attack. We therefore refer to this theorem as *Higher Order Escape Theorem* as opposed to survival. We plot the result in Figure 3b.

Theorem 3 (Higher Order Escape Theorem)
There is a $\geqslant 1 - \frac{\tau}{1-\theta(1-\tau)}$ *probability that the attacker will never get the value of an exchange.*

Proof. This theorem relies on the fact that, if, at some time, a communication is made via a safe edge (which the eavesdropper cannot spy), then, the attacker will never know which peer to spy for the next step of the indirect attack. Since we do not assume that the same peer will not be randomly selected several times, and we only want a lower bound on the probability of success of an attack, we will consider a worst case, where only one new exchange needs to be captured at each step.

For the first step, the probability of the exchange not being captured is $1-\tau$ and then, its probability of happening on a safe edge is $1 - \theta$, which gives a $(1 - \tau)(1 - \theta)$ probability that the attacker will never learn the value of the exchange. If the communication happens on an unsafe edge, then, at each step, the chance of reaching a safe edge is multiplied by $\theta(1 - \tau)$ (the previous edge was not safe and the next peer is not corrupted). So, at step k (assuming that the first step is 0), the probability of not having reached a corrupted peer and reaching a safe edge for the first time at this step is $\theta^k(1-\tau)^{k+1}(1-\theta)$. This is a geometric series, so we can compute its sum.

$$E_n = \sum_{k=0}^{n} \theta^k (1-\tau)^{k+1}(1-\theta) = (1-\tau)(1-\theta)\frac{1 - \theta^{k+1}(1-\tau)^{k+1}}{1 - \theta(1-\tau)}$$



(a) Perfect Peer Sampling (b) Req-Pull Peer Sampling

Figure 4: Value convergence with different peer sampling protocols.

Since $\theta(1-\tau) < 1$, we have $E_{+\infty} = (1-\tau)(1-\theta)\dfrac{1}{1-\theta(1-\tau)} = \dfrac{(1-\tau)(1-\theta)}{1-\theta(1-\tau)}$

$$E_{+\infty} = \frac{1-\theta-\tau+\theta\tau}{1-\theta+\theta\tau} = 1 - \frac{\tau}{1-\theta+\theta\tau} = 1 - \frac{\tau}{1-\theta(1-\tau)}$$

□

4.3 Averaging Performance

We now report on our simulations[3] of a system consisting of 1000 peers with uniformly distributed values between −100 and 100. We first evaluate the convergence speed of the averaging protocol when running with a perfect peer sampling (complete graph, Figure 4a) and a Req-Pull peer sampling [3] (Figure 4b) in a simulator. To this end, we plot the values of 40 peers for each case omitting the privacy-generation phase. We first let the peer sampling construct local views before starting the averaging process. The time unit is a tick of c++ steady_clock (10^{-9} second here). Each peer performed ~ 40 exchanges per second.

Figure 5 shows instead the convergence time (no peers with >1% error) of the averaging process depending on the required privacy level, with different peer sampling protocols and network sizes. We observe that neither the peer sampling used nor the number of nodes has great influence on the convergence time. We see that the convergence time grows linearly with the privacy level. It is worth noting that, in our theoretical analysis, we proved that the probability of success of a direct attack decreases exponentially with the privacy level, compared to the linear growth of the convergence time.

5 Related work

Several authors have tackled the problem of privacy-preserving averaging in a decentralized setting, mostly within the automatic control community, where

[3] The code used for these experiments is available at https://github.com/ALRBP/Private_Gossip_Average

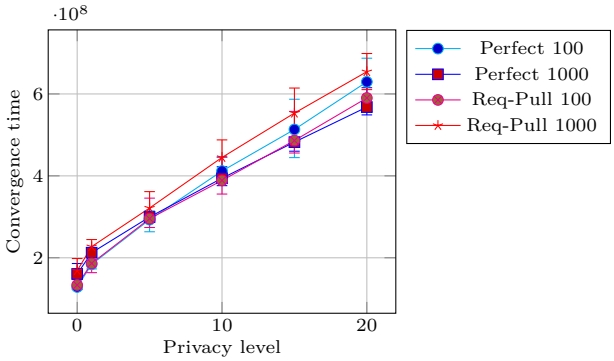


Figure 5: Convergence time versus privacy level.

this problem is known as average consensus [13,16,18,5,31,19]. But most of these algorithms either rely on central components or assume static networks that operate in synchronous rounds with a lock-step execution model.

For example, [13] requires a central authority that distributes concealing factors before the beginning of the computation. Nodes then use this information to perturb the values sent during the averaging algorithm. Similarly, [27] assumes synchronous lock-step rounds and proposes a solution in which nodes exchange values together with a decaying noise. Moreover, due to the nature of this noise, the algorithm does not converge to the average in an exact sense but only in a mean-square sense. The work in [16] considers an ad hoc network setting, and thus also relies on an update rule that reaches all neighbors at the same time. But unlike [27] it provides deterministic convergence to the average value.

The work in [26] presents close similarities to our approach. Each node adds a zero-average noise to message exchanges for an independently chosen number of steps. The paper establishes topological conditions for the effectiveness of its privacy preservation, but this relies on the assumption that nodes communicate with all their neighbors at each communication step. Moreover, like in most other protocols, nodes start by exchanging their value plus some noise, whereas our protocol only exchanges noise for several rounds and adds the actual value only later in the process.

The work in [12] adopts an approach similar to [26] in the context of a push-sum averaging protocol [22] but still assumes that the network evolves in synchronous lock-step rounds. The authors of [38] further shows that [26] is vulnerable to attacks on some topologies (for example on a ring). Our approach does not suffer from the same vulnerability thanks to its dynamic topology and the use of encrypted communication for all exchanges.

Some authors [37,24,14] have combined noise addition with homomorphic cryptography [11]. For example, [24] improves [27] by using homomorphic encryption in order to establish a confidential interaction protocol between nodes, while [30] proposes an averaging protocol that exploits partial homomorphic encryption in pairwise interactions. The work in [8] applies homomorphic encryp-

tion in an asynchronous setting, but it requires random noise to be generated by pairs of peers, which implies that a peer cannot protect its value without collaborating with another peer that is also protecting its value. Overall, even if partially homomorphic cryptographic starts to be viable, these approaches still incur a computational cost that appears unsuitable for small, battery-operated, devices. Moreover, in the case of our protocol, simple public-key encryption is enough to protect communication against eavesdroppers.

Other authors have combined the idea of adding noise with that of state decomposition. [35] divides each node into two virtual nodes, one that talks to the original node's neighbors and the other that only talks to the first virtual node. The approach has the advantage that the attacker cannot estimate the value with any guaranteed accuracy, but only if the target has at least one neighbor which is not under the influence of or observable by the attacker. [1] creates instead virtual nodes to divide the node's original value into random shares. But this allows attackers that control a sufficiently large sample of the network to guess the actual value of the peer with a low level of uncertainty. Finally, [32] considers a random-share protocol incorporating wiretap codes [36] to reduce message overhead. This yields low message complexity but only thanks to the use of a broadcast channel that is not available in large-scale networks.

Another recent contribution [34] proposes a hybrid architecture where a set of servers collect data from a set of nodes and compute the average. The paper focuses on providing nodes with heterogeneous privacy guarantees with respect to different privacy violators. It uses noise addition and employs KL divergence to measure a privacy preserving degree (PPD). However, it cannot compute the precise average but just an approximation.

This is similar to what happens with differential privacy [10,9]. Converging to a perturbed value guarantees that attacker cannot guess information about the original data distribution from the value of the final average. But this comes at the cost of being unable to converge to the exact average value, as proven in [29] and in [15]. As a further example, [33] proposes a differentially private averaging protocol with an optimization that groups node interactions to minimize network usage, while [2] proposes a differentially private protocol that uses homomorphic encryption in the context of k-means computation. Finally, some authors have started to tackle the problem of dealing with dishonest nodes that lie on their values [17]. This could be an interesting improvement for our work.

6 Conclusion

We proposed a novel protocol for privacy-preserving gossip averaging that addresses several limitations of previous approaches. While our protocol cannot be completely immune to attacks, we characterized its guarantees by formally proving several privacy properties. Now that we have a working averaging protocol, we plan to apply it in the context of higher-level applications such as decentralized machine-learning. Another direction consists in exploring its relations with recent work in the context of multi-party computation [6,25].

References

1. T. Allard, D. Frey, G. Giakkoupis, and J. Lepiller. Lightweight privacy-preserving averaging for the internet of things. 2016.
2. T. Allard, G. Hébrail, F. Masseglia, and E. Pacitti. Chiaroscuro: Transparency and privacy for massive personal time-series clustering. In *ACM SIGMOD 2015*, pages 779–794, 2015.
3. A. Allavena, A. Demers, and J. E. Hopcroft. Correctness of a gossip based membership protocol. In *Proceedings of the Twenty-fourth Annual ACM Symposium on Principles of Distributed Computing*, pages 292–301. ACM, 2005.
4. E. Bortnikov, M. Gurevich, I. Keidar, G. Kliot, and A. Shraer. Brahms: Byzantine resilient random membership sampling. *Computer Networks*, 53(13):2340–2359, 2009.
5. C. Clifton, M. Kantarcioglu, J. Vaidya, X. Lin, and M. Y. Zhu. Tools for privacy preserving distributed data mining. *ACM SIGKDD Explorations Newsletter*, 4:28–34, 2002.
6. S. Coretti, J. A. Garay, M. Hirt, and V. Zikas. Constant-round asynchronous multi-party computation based on one-way functions. In *Proceedings, Part II, of the 22Nd International Conference on Advances in Cryptology — ASIACRYPT 2016 - Volume 10032*, pages 998–1021. Springer-Verlag New York, Inc., 2016.
7. G. Danner and M. Jelasity. Fully distributed privacy preserving mini-batch gradient descent learning. In *Distributed Applications and Interoperable Systems*, pages 30–44. Springer International Publishing, 2015.
8. P. Dellenbach, A. Bellet, and J. Ramon. Hiding in the crowd: A massively distributed algorithm for private averaging with malicious adversaries. *CoRR*, 2018.
9. C. Dwork. Differential privacy: a survey of results. volume 4978, pages 1–19, 2008.
10. C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In *Proceedings of the Third Conference on Theory of Cryptography*, pages 265–284. Springer Berlin Heidelberg, 2006.
11. K. B. Frikken. Secure multiparty computation. In M. J. Atallah and M. Blanton, editors, *Algorithms and Theory of Computation Handbook*, pages 14.1–14.16. Chapman & Hall/CRC, 2010.
12. H. Gao, C. Zhang, M. Ahmad, and Y. Wang. Privacy-preserving average consensus on directed graphs using push-sum. In *2018 IEEE Conference on Communications and Network Security (CNS)*, pages 1–9. IEEE, 2018.
13. N. Gupta and N. Chopra. Confidentiality in distributed average information consensus. In *2016 IEEE 55th Conference on Decision and Control (CDC)*, pages 6709–6714. IEEE, 2016.
14. C. N. Hadjicostis. Privary preserving distributed average consensus via homomorphic encryption. In *2018 IEEE Conference on Decision and Control (CDC)*, pages 1258–1263. IEEE, 2018.
15. J. He and L. Cai. Differential private noise adding mechanism: Basic conditions and its application. In *2017 American Control Conference (ACC)*, pages 1673–1678. IEEE, 2017.
16. J. He, L. Cai, P. Cheng, J. Pan, and L. Shi. Consensus-based privacy-preserving data aggregation. *IEEE Transactions on Automatic Control*, 2016.
17. J. He, L. Cai, P. Cheng, J. Pan, and L. Shi. Distributed privacy-preserving data aggregation against dishonest nodes in network systems. *IEEE Internet of Things Journal*, 6(2):1462–1470, 2019.

18. J. He, L. Cai, C. Zhao, P. Cheng, and X. Guan. Privacy-preserving average consensus : Privacy analysis and optimal algorithm design. *IEEE Transactions on Signal and Information Processing over Networks*, 5(1):127–138, 2019.
19. Z. Huang, S. Mitra, and G. Dullerud. Differentially private iterative synchronous consensus. In *Proceedings of the 2012 ACM Workshop on Privacy in the Electronic Society*, WPES '12, pages 81–90. ACM, 2012.
20. M. Jelasity, A. Montresor, and O. Babaoglu. Gossip-based aggregation in large dynamic networks. *ACM Transactions on Computer Systems*, 23(3), 2005.
21. M. Jelasity, S. Voulgaris, R. Guerraoui, A.-M. Kermarrec, and M. van Steen. Gossip-based peer sampling. *ACM ToCS*, 25(3), 2007.
22. D. Kempe, A. Dobra, and J. E. Gehrke. Gossip-based computation of aggregate information. In *Proceedings of the Twenty-fourth Annual ACM Symposium on Principles of Distributed Computing*, pages 482–491, 2003.
23. J. Lepiller. Private decentralized aggregation. 2016.
24. Q. Liu, X. Ren, and Y. Mo. Secure and privacy preserving average consensus. In *2017 11th Asian Control Conference (ASCC)*, pages 274–279. IEEE, 2017.
25. C.-D. Liu-Zhang, J. Loss, U. Maurer, T. Moran, and D. Tschudi. Robust mpc: Asynchronous responsiveness yet synchronous security. In *Theory and Practice of Multi-Party Computation Workshops 2019*, 2019.
26. N. E. Manitara and C. N. Hadjicostis. Privacy-preserving asymptotic average consensus. In *2013 European Control Conference (ECC)*, pages 760–765. IEEE, 2013.
27. Y. Mo and R. M. Murray. Privacy preserving average consensus. In *53rd IEEE Conference on Decision and Control*, pages 2154–2159. IEEE, 2014.
28. B. Nédelec, J. Tanke, P. Molli, A. Mostéfaoui, and D. Frey. An adaptive peer-sampling protocol for building networks of browsers. 2017.
29. E. Nozari, P. Tallapragada, and J. Cortés. Differentially private average consensus: Obstructions, trade-offs, and optimal algorithm design. *Automatica*, 81:221–231, 2015.
30. M. Ruan, H. Gao, and Y. Wang. Secure and privacy-preserving consensus. *IEEE Transactions on Automatic Control*, 2019.
31. R. Sheikh, B. Kumar, and D. K. Mishra. A distributed k-secure sum protocol for secure multi-party computations. *Journal of Computing*, 2:68–72, 2010.
32. R. Thobaben, G. Dán, and H. Sandberg. Wiretap codes for secure multi-party computation. In *2014 IEEE Globecom Workshops (GC Wkshps)*, pages 1349–1354. IEEE, 2014.
33. A. Wang, X. Liao, and H. He. Event-triggered differentially private average consensus for multi-agent network. *IEEE/CAA Journal of Automatica Sinica*, 6(1):75–83, 2019.
34. X. Wang, J. He, P. Cheng, and J. Chen. Privacy preserving collaborative computing: Heterogeneous privacy guarantee and efficient incentive mechanism. *IEEE Transactions on Signal Processing*, 67(1):221–233, 2018.
35. Y. Wang. Privacy-preserving average consensus via state decomposition. *IEEE Transactions on Automatic Control*, 2019.
36. A. D. Wyner. The wire-tap channel. *The Bell System Technical Journal*, 54(8):1355–1387, 1975.
37. T. Yin, Y. Lv, and W. Yu. Accurate privacy preserving average consensus. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 2019.
38. H. Zhou, W. Yang, and C. Yang. Privacy preserving consensus under interception attacks. In *2017 36th Chinese Control Conference (CCC)*, pages 8485–8490. IEEE, 2017.