Hiding in the Crowd: A Massively Distributed Algorithm for Private Averaging with Malicious Adversaries

Pierre Dellenbach, Aurélien Bellet, Jan Ramon

Hiding in the Crowd: A Massively Distributed Algorithm for Private Averaging with Malicious Adversaries

Pierre Dellenbach[1], Aurélien Bellet*[1], and Jan Ramon[1]

[1]INRIA

Abstract

The amount of personal data collected in our everyday interactions with connected devices offers great opportunities for innovative services fueled by machine learning, as well as raises serious concerns for the privacy of individuals. In this paper, we propose a massively distributed protocol for a large set of users to privately compute averages over their joint data, which can then be used to learn predictive models. Our protocol can find a solution of arbitrary accuracy, does not rely on a third party and preserves the privacy of users throughout the execution in both the honest-but-curious and malicious adversary models. Specifically, we prove that the information observed by the adversary (the set of maliciours users) does not significantly reduce the uncertainty in its prediction of private values compared to its prior belief. The level of privacy protection depends on a quantity related to the Laplacian matrix of the network graph and generally improves with the size of the graph. Furthermore, we design a verification procedure which offers protection against malicious users joining the service with the goal of manipulating the outcome of the algorithm.

1 Introduction

Through browsing the web, engaging in online social networks and interacting with connected devices, we are producing ever growing amounts of sensitive personal data. This has fueled the massive development of innovative personalized services which extract value from users' data using machine learning techniques. In today's dominant approach, users hand over their personal data to the service provider, who stores everything on centralized or tightly coupled systems hosted in data centers. Unfortunately, this poses important risks regarding the privacy of users. To mitigate these risks, some approaches have been proposed to learn from datasets owned by several parties who do not want to disclose their data. However, they typically suffer from some drawbacks: (partially) homomorphic encryption schemes (Paillier, 1999; Graepel et al., 2012; Aslett et al., 2015) require the existence of a trusted third party, secure multi-party computation techniques (Yao, 1982; Lindell and Pinkas, 2009) are generally intractable when the number of parties is large, and exchanging noisy sketches of the data through (local) differential privacy (Dwork, 2006; Duchi et al., 2012) only provides approximate solutions which are quite inaccurate in the highly distributed setting considered here. Furthermore, many of these techniques are not robust to the presence of malicious parties who may try to manipulate the outcome of the algorithm.

In this paper, our goal is to design a massively distributed protocol to collaboratively compute averages over the data of thousands to millions of users (some of them honest-but-curious and some corrupted by a malicious party), with arbitrary accuracy and in a way that preserves their privacy. For machine learning algorithms whose sufficient statistics are averages (e.g., kernel-based algorithms in primal space and decision trees), this could be used as a primitive to privately learn more complex models. The approach we propose is fully decentralized: users keep their own data locally and exchange information asynchronously over a

*Corresponding author: first.last@inria.fr

peer-to-peer network (represented as a connected graph), without relying on any third party. Our algorithm (called GOPA: GOssip for Private Averaging) draws inspiration from a randomized gossip protocol for averaging (Boyd et al., 2006), augmented with a first phase involving pairwise exchanges of noise terms so as to mask the private values of users without affecting the global average. We first analyze the correctness of the algorithm, showing that the addition of noise has a mild effect on the convergence rate. We then study the privacy guarantees of GOPA in a Bayesian framework, where the adversary has some prior belief about the private values. Specifically, we give an exact expression for the posterior variance of the adversary after he has observed all the information output by the protocol, and show that the variance loss is negligible compared to the prior. This is equivalent to showing that the uncertainty in the adversary's predictions of the private values has not been significantly reduced. Interestingly, the proportion of preserved variance depends on the variance of the noise used to mask the values but also on an interpretable quantity related to the Laplacian matrix of the network graph. To the best of our knowledge, we are the first to draw a link between privacy and a graph smoothing operator popular in semi-supervised learning (Zhu and Ghahramani, 2002; Zhou et al., 2003), multi-task learning Evgeniou and Pontil (2004) and signal processing (Shuman et al., 2013). We show how this result motivates the use of a random graph model to construct the network graph so as to guarantee strong privacy even under rather large proportions of malicious users, as long as the number of users is big enough. The practical behavior of GOPA is illustrated on some numerical simulations. Finally, we further enhance our protocol with a verification procedure where users are asked to publish some values in encrypted form, so that cheaters trying to manipulate the output of the algorithm can be detected with high probability while preserving the aforementioned privacy guarantees.

The rest of this paper is organized as follows. Section 2 describes the problem setting, including our adversary and privacy models. Section 3 presents some background on (private) decentralized averaging and partially homomorphic encryption, along with related work and baseline approaches. Section 4 introduces the GOPA algorithm and studies its convergence qs well as privacy guarantees. Section 5 describes our verification procedure to detect cheaters. Finally, Section 6 displays some numerical simulations. Proofs can be found in the supplementary material.

2 Preliminaries

We consider a set $U = \{1, \dots, n\}$ of $n \geq 3$ users. Each user $u \in U$ holds a personal value $X_u \in \mathbb{R}$, which can be thought of as the output of some function applied to the personal data of u (e.g., a feature vector describing u). The users want to collaboratively compute the average value $X^{avg} = \frac{1}{n}\sum_{u=1}^{n} X_u$ while keeping their personal value private. Such a protocol could serve as a building block for privately running machine learning algorithms which interact with the data only through averages, such as linear regression models (ordinary least-squares, ridge regression), decision trees and gradient descent for empirical risk minimization problems. We denote by X the vector $X = [X_1, \dots, X_n]^\top \in \mathbb{R}^n$.

Instead of relying on a central server or any third party to exchange information, users communicate over a peer-to-peer network represented by a connected undirected graph $G = (U, E)$, where $(u, v) \in E$ indicates that users u and v are neighbors in G and can exchange messages directly. For a given user u, we denote by $N(u) = \{v : (u, v) \in E\}$ the set of its neighbors. We denote by A the adjacency matrix of G, by $d = (d_1, \dots, d_n)$ the degree vector ($d_u = \sum_v A_{uv}$) and by $L = \mathrm{diag}(d) - A$ its Laplacian matrix.

2.1 Adversary Models

We consider two commonly adopted adversary models for users, which were formalized by Goldreich (1998) and are used in the design of many secure protocols. An *honest-but-curious* (*honest* for short) user will follow the protocol specification, but can use all the information received during the execution to infer information about other users. In contrast, a *malicious user* may deviate from the protocol execution by sending incorrect values at any point (but we assume that they follow the required communication policy; if not, this can be easily detected). Malicious users can collude, and thus will be seen as a single malicious party who has access to all information collected by malicious users. We only restrict the power of attackers by requiring

that honest users communicate through secure channels, which means that malicious users only observe information during communications they take part in.

Each user in the network is either honest or malicious, and honest users do not know whether other nodes are honest or malicious. We denote by $U^H \subseteq U$ the set of honest users and by $f = 1 - |U^H|/n$ the proportion of malicious users in the network. We also denote by $G^H = (U^H, E^H)$ the subgraph of G induced by U^H, so that $E^H = \{(u, v) \in E : u, v \in U^H\}$. Throughout the paper, we will rely on the following natural assumption on G^H (we will discuss how to generate G such that it holds in practice in Section 4.4).

Assumption 1. *The graph of honest users G^H is connected.*

This implies that there exists a path between any two honest users in the full graph G which does not go through a malicious node. In the rest of the paper, we will use the term *adversary* to refer to the set of malicious users (privacy with respect to a honest user can be obtained as a special case).

2.2 Privacy Model

Recall that our goal is to design a protocol which deterministically outputs the *exact* average (which we argue does not reveal much information about individual values in the large-scale setting we consider). This requirement automatically rules out Differential Privacy (Dwork, 2006) as the latter implies that the output of the protocol has to be randomized.

We take a Bayesian, semantic view of privacy promoted by several recent papers (Kasiviswanathan and Smith, 2014; Li et al., 2013; Kifer and Machanavajjhala, 2014; He et al., 2014). We consider a family of prior distributions which represent the potential background knowledge that the adversary may have about the private values of honest users (since the adversary controls the malicious users, we consider he has exact knowledge of their values). We will denote by $P(X_u)$ the prior belief of the adversary about the private value of some honest user $u \in U^H$. Given all the information $\mathcal{I}$ gathered by the adversary during the execution of the protocol, the privacy notion we consider is that the ratio of prior and posterior variance of the private value X_u is lower bounded by $1 - \epsilon$ for some $\epsilon \in [0, 1]$:

$$\frac{var(X_u \mid \mathcal{I})}{var(X_u)} \geq 1 - \epsilon \tag{1}$$

The case $\epsilon = 1$ describes the extreme case where observing $\mathcal{I}$ removed all uncertainty about X_u. On the other hand, when $\epsilon = 0$, all variance was preserved.

It is important to note that (1) with ϵ close to 0 does not guarantee that the adversary learns almost nothing from the output $\mathcal{I}$: in particular, the posterior expectation $\mathbb{E}[X_u \mid \mathcal{I}]$ can largely differ from the prior expectation $\mathbb{E}[X_u]$, especially if the observed global average was very unlikely under the prior belief. This is related to the "no free lunch" theorem in data privacy (Kifer and Machanavajjhala, 2011), see also the discussions in Kasiviswanathan and Smith (2014); Li et al. (2013). What (1) does guarantee, however, is that the squared error that the adversary expects to make by predicting the value of some private X_u after observing $\mathcal{I}$ is almost the same as it was before observing $\mathcal{I}$. In other words, the *uncertainty* in its prediction has not been significantly reduced by the participation to the protocol. This is formalized by the following remark.

Remark 1 (Expected squared error)**.** *Assume that* (1) *is satisfied. Let $Y_u = \mathbb{E}[X_u]$ be the best prediction (in terms of expected square error) that the adversary can make given its prior belief. After observing the output $\mathcal{I}$, the adversary can make a new best guess $Y_u^{\mathcal{I}} = \mathbb{E}[X_u \mid \mathcal{I}]$. We have:*

$$\frac{\mathbb{E}[(Y_u^{\mathcal{I}} - X_u)^2]}{\mathbb{E}[(Y_u - X_u)^2]} = \frac{var(X_u \mid \mathcal{I})}{var(X_u)} = 1 - \epsilon.$$

Our results will be valid for Gaussian prior distributions of the form $X_u \sim \mathcal{N}(0, \sigma_X^2)$, for all $\sigma_X^2 > 0$.[1] We assume for simplicity that the prior variance is the same for all u, but our analysis straightforwardly extends

[1] We use Gaussian distributions for technical reasons. We expect similar results to hold for other families of distributions which behave nicely under conditioning and linear transformations, such as the exponential family.

Algorithm 1 Randomized gossip (Boyd et al., 2006)

Input: graph $G = (U, E)$, initial values $(X_u(0))_{u \in U}$
for $t \in \{1, 2, ...\}$ **do**
  Draw (u, v) uniformly at random from E
  Set $X_u(t), X_v(t) \leftarrow \frac{1}{2}(X_u(t-1) + X_v(t-1))$
end for

to the more general case where $X_u \sim \mathcal{N}(0, \sigma^2_{X_u})$. We use centered Gaussians without loss of generality, since (1) depends only on the variance.

Remark 2 (Privacy axioms)**.** *One can show that our notion of privacy* (1) *satisfies the axioms that any reliable privacy definition should satisfy according to Kifer and Lin (2012), namely "transformation invariance" and "convexity".*

3 Background and Related Work

3.1 Decentralized Averaging

The problem of computing the average value X^{avg} of a set of users in a fully decentralized network without a central coordinator has been extensively studied (see for instance Tsitsiklis, 1984; Kempe et al., 2003; Boyd et al., 2006). In most existing approaches, users iteratively compute the weighted average between their value and those of (a subset of) their neighbors in the network. We focus here on the randomized gossip algorithm proposed by Boyd et al. (2006) as it is simple, asynchronous and exhibits fast convergence. Each user has a clock ticking at the times of a rate 1 Poisson process. When the clock of an user u ticks, it chooses a random neighbor v and they average their current value. As discussed in Boyd et al. (2006), one can equivalently consider a single clock ticking at the times of a rate n Poisson process and a random edge $(u, v) \in E$ drawn at each iteration. The procedure is shown in Algorithm 1, and one can show that all users converge to X^{avg} at a geometric rate.

Although there is no central server collecting all users' data, the above gossip algorithm is not private since users must share their inputs directly with others. A way to ensure privacy is that each user locally perturbs its own input before starting the algorithm so as to satisfy local differential privacy (Duchi et al., 2012; Kairouz et al., 2016).

Baseline 1. *Assume $|X_u| \leq B_X$ for some finite $B_X \in \mathbb{R}$. Each user u computes a perturbed value $\widetilde{X}_u = X_u + \eta_u$, where η_u is a noise value with Laplacian distribution $P(\eta_u) = \exp(-|\eta_u|/b)/2b$ with $b = 2B/\epsilon$. This guarantees that $\widetilde{X} = [\widetilde{X}_1, \ldots, \widetilde{X}_n]^T$ is an ϵ-differentially private approximation of X (see e.g., Dwork, 2008), hence running Algorithm 1 on $\widetilde{X}$ is also ϵ-differentially private.*

Unfortunately, this protocol converges to an approximate average $\hat{X}^{avg} = \sum_{u=1}^{n}(X_u + \eta_u)/n$, and the associated RMSE $\sqrt{\mathbb{E}[(X^{avg} - \hat{X}^{avg})^2]} = b\sqrt{2/n}$ can be high even for a large number of users. For instance, for $n = 10^4$ users, $\epsilon = 0.1$ and $B = 0.5$, the RMSE is approximately 0.14.

Other attempts have been made at designing privacy-preserving decentralized averaging protocols under various privacy and attack models, and additional assumptions on the network topology. For an adversary able to observe all communications in the network, Huang et al. (2012) proposed an ϵ-differentially private protocol where users add exponentially decaying Laplacian noise to their value. The protocol converges with probability p to a solution with an error radius of order $O(1/\epsilon\sqrt{(1-p)n})$. Manitara and Hadjicostis (2013) instead proposed that each user iteratively adds a finite sequence of noise terms which sum to zero, so that convergence to the exact average can be achieved. The adversary consists of a set of curious nodes which follow the protocol but can share information between themselves. Under some assumption on the position of the curious nodes in the network, their results prevent *perfect* recovery of private values but do not bound the

estimation accuracy that the adversary can achieve. Mo and Murray (2014) proposed to add and subtract exponentially decaying Gaussian noise, also ensuring convergence to the exact average. They consider only honest-but-curious nodes and rely on a strong topological assumption, namely that there is no node whose neighborhood includes another node and its entire neighborhood. Their privacy guarantees are in the form of lower bounds on the covariance matrix of the maximum likelihood estimate of the private inputs that any node can make, which are difficult to interpret and very loose in practice. Finally, Hanzely et al. (2017) introduced variants of Algorithm 1 which intuitively leak less information about the private inputs, but do not necessarily converge to the exact average. Importantly, they do prove any formal privacy guarantee.

In the above approaches, privacy is typically achieved by asking each user to independently add decaying local noise to its private value, which results in accuracy loss and/or weak privacy guarantees. In contrast, the protocol we propose in this paper relies on *sharing* zero-sum noise terms across users so as to "dilute" the knowledge of private values into the crowd. This will allow a flexible trade-off between privacy and convergence rate even when a node has a large proportion of malicious neighbors, while ensuring the convergence to the exact average. Furthermore and unlike all the above approaches, we provide a verification procedure to prevent malicious nodes from manipulating the output of the algorithm. This verification procedure relies on partially homomorphic encryption, which we briefly present below.

3.2 Partially Homomorphic Encryption

A standard technique for secure computation is to rely on a partially homomorphic encryption scheme to cipher the values while allowing certain operations to be carried out directly on the cypher text (without decrypting).

We use the popular Paillier cryptosystem (Paillier, 1999), which is additive. Formally, one generates a public (encryption) key $K^{pub} = (N, g)$, where $N = pq$ for $p \neq q$ two independent large primes and $g \in \mathbb{Z}^*_{N^2}$, and a secret (decryption) key $K^{priv} = \text{lcm}(p-1, q-1)$. A message $m \in \mathbb{Z}^*_N$ can then be encrypted into a cypher text with

$$\varepsilon(m) = g^m \cdot r^N \bmod N^2, \tag{2}$$

where r is drawn randomly from $\mathbb{Z}^*_N$. With knowledge of the secret key, one can recover m from $\varepsilon(m)$ based on the fact that $\frac{\varepsilon(m)^\lambda - 1}{g^\lambda - 1} \equiv m \bmod N$. Denote the decryption operation by ε^{-1}. Paillier satisfy the following homomorphic property for any $m_1, m_2 \in \mathbb{Z}^*_N$:

$$\varepsilon^{-1}(\varepsilon(m_1) \cdot \varepsilon(m_2) \bmod N^2) = m_1 + m_2 \bmod N, \tag{3}$$

hence $\varepsilon(m_1) \cdot \varepsilon(m_2)$ is a valid encryption of $m_1 + m_2$. For the purpose of this paper, we will consider the Paillier encryption scheme as perfectly secure (i.e., the computational complexity needed to break the encryption is beyond reach of any party). We can use the Paillier scheme to design a second simple baseline for private averaging.

Baseline 2. *Consider $X_1, \dots, X_n \in \mathbb{Z}^*_N$ and assume that users trust two central honest-but-curious entities: the server and the third party. The server generates a Paillier encryption scheme (K^{pub}, K^{priv}) and broadcasts K^{pub} to the set of users. Each user u computes $\varepsilon(X_u)$ and sends it to the third party. Following* (3)*, the third party then computes $\prod_{u=1}^n \varepsilon(X_u)$ and sends it to the server, which can obtain X^{avg} by decrypting the message (and dividing by n).*

If the server and the third party are indeed honest, nobody observes any useful information except the outcome X^{avg}. But if they are not honest, various breaches can occur. For instance, if the third party is malicious, it can send an incorrect output to the server. The third party could also send the encrypted values of some users to the server, which can decrypt them using the private key K^{priv}. Similarly, if the server is malicious, it can send K^{priv} to the third party which can then decrypt all of the users' private values.

In contrast, we will design a protocol which eliminates the need for such central trusted entities and instead distributes the trust across many users in the network (Section 4). We will however rely on homomorphic encryption to detect potential malicious users (Section 5).

Algorithm 2 Randomization phase of GOPA

Input: graph $G = (U, E)$, private values $(X_u)_{u \in U}$, distribution $\mu : \mathbb{R} \to [0, 1]$
for all neighbor pairs $(u, v) \in E$ s.t. $u < v$ **do**
  u and v jointly draw a random number $\delta \in \mathbb{R}$ from μ
  $\delta_{u,v} \leftarrow \delta$, $\delta_{v,u} \leftarrow -\delta$
end for
for all users $u \in U$ **do**
  $\Delta_u \leftarrow \sum_{v \in N(u)} \delta_{u,v}$, $\widetilde{X}_u \leftarrow X_u + \Delta_u$
end for
Output: noisy values $(\widetilde{X}_u)_{u \in U}$

4 GOPA: Private Gossip Averaging Protocol

4.1 Protocol Description

We describe our GOPA protocol (GOssip for Private Averaging), which works in two phases. In the first phase (*randomization phase*), users mask their private value by adding noise terms that are correlated with their neighbors, so that the global average remains unchanged. In the second phase (*averaging phase*), users average their noisy values. For simplicity of explanation, we abstract away the communication mechanisms, i.e., contacting a neighbor and exchanging noise are considered as atomic operations.

Randomization phase. Algorithm 2 describes the first phase of GOPA, during which all neighboring nodes $(u, v) \in E$ contact each other to exchange noise values. Specifically, they jointly draw a random real number δ from a probability distribution μ (a parameter of the algorithm that the community agrees on), that u will add to its private value and v will subtract. Following the common saying "don't put all your eggs in one basket", each user thereby distributes the noise masking his private value across several other users (his direct neighbors but also beyond by transitivity), which will provide some robustness to malicious parties. The idea is reminiscent of one-time pads (see for instance Bonawitz et al., 2017, Section 3 therein), but our subsequent analysis will highlight the key role of the network topology and show that we can tune the magnitude of the noise so as to trade-off between privacy on the one hand, and convergence speed as well as the impact of user drop out on the other hand. The result of this randomization phase is a set of noisy values $\widetilde{X} = [\widetilde{X}_1, \ldots, \widetilde{X}_n]^\top$ for each user, with the same average value X^{avg} as the private values. Note that each user u exchanges noise exactly once with each of his neighbors, hence the noisy value $\widetilde{X}_u$ consists of d_u noise terms.

Averaging phase. In the second phase, the users start from their noisy values $\widetilde{X}$ obtained in the randomization phase and simply run the standard randomized gossip averaging algorithm (Algorithm 1).

4.2 Correctness and Convergence of GOPA

In this section, we study the correctness and convergence rate of GOPA. Let $\widetilde{X}(t) = [\widetilde{X}_1(t), \ldots, \widetilde{X}_n(t)]^\top$ be the values after $t \geq 0$ iterations of the averaging phase (Algorithm 1) initialized with the noisy values $\widetilde{X}$ generated by the randomization phrase (Algorithm 2). Following the seminal work of Boyd et al. (2006), we will measure the convergence rate in terms of the τ-*averaging time*. Given $\tau \in (0, 1)$, the τ-averaging time is the number of iterations $t(\tau)$ needed to guarantee that for any $t \geq t(\tau)$:

$$\Pr(\|\widetilde{X}(t) - X^{avg}\mathbf{1}\|/\|X\| \geq \tau) \leq \tau. \tag{4}$$

Note that the error in (4) is taken relatively to the original set of values X to account for the impact of the addition of noise on the convergence rate. We have the following result for the case where all users are honest (we will lift this requirement in Section 5).

Proposition 1 (Correctness and convergence rate)**.** *When all users are honest, the sequence of iterates* $(\widetilde{X}(t))_{t\geq 0}$ *generated by* GOPA *satisfies* $\lim_{t\to+\infty}\mathbb{E}[\widetilde{X}(t)] = X^{avg}\mathbf{1}$*. Furthermore, the* τ*-averaging time of Algorithm 1 is:*

$$t(\tau) = 3\log\Big(\frac{2B_\delta(d_{max}+3)}{\tau B_X}\Big)\Big/\log(1/C_G), \tag{5}$$

where $C_G = 1-\lambda_2(L)/|E| \in (0,1)$ *with* $\lambda_2(L)$ *the second smallest eigenvalue of the Laplacian matrix* L *(Boyd et al., 2006; Colin et al., 2015),* $d_{max} = \max_u d_u$ *is the maximum degree and* B_X, B_δ *are upper bounds for the absolute value of private values and noise terms respectively.*[2]

Proposition 1 allows to quantify the *worst-case* impact of the randomization phase on the convergence of the averaging phase. The τ-averaging time of GOPA is only increased by a constant additive factor compared to the non-private averaging phase. Importantly, this additive factor has a mild (logarithmic) dependence on B and d_{max}. This behavior will be confirmed by numerical experiments in Section 6.

4.3 Privacy Guarantees

We now study the privacy guarantees of GOPA. We consider that the knowledge $\mathcal{I}$ acquired by the adversary (colluding malicious users) during the execution of the protocol contains the following: (i) the noisy values $\widetilde{X}$ of all users at the end of the randomization phase, (ii) the full network graph (and hence which pairs of honest users exchanged noise), and (iii) for any communication involving a malicious party, the noise value used during this communication. The only unknowns are the private values of honest users, and noise values exchanged between them. Note that (i) implies that the adversary learns the network average.

Remark 3. *Since we assume that the noisy values* $\widetilde{X} = [\widetilde{X}_1,\dots,\widetilde{X}_n]^\top$ *are known to the adversary, our privacy guarantees will hold even if these values are publicly released. Note however that computing the average with Algorithm 1 provides additional protection as the adversary will observe only a subset of the noisy values of honest nodes (those who communicate with a malicious user at their first iteration). In fact, inferring whether a received communication is the first made by that user is already challenging due to the asynchronous nature of the algorithm.*

Our main result exactly quantifies how much variance is left in any private value X_u after the adversary has observed all information in $\mathcal{I}$.

Theorem 1 (Privacy guarantees)**.** *Assume that the prior belief on the private values* $X_1,\dots,X_n$ *is Gaussian, namely* $X_u \sim \mathcal{N}(0,\sigma_X^2)$ *for all honest users* $u \in U^H$*, and that the noise variables are also drawn from a Gaussian distribution* $\mu = \mathcal{N}(0,\sigma_\delta^2)$*. Denote by* $e_u \in \mathbb{R}^n$ *the indicator vector of the* u*-th coordinate and let* $M = (I + \frac{\sigma_\delta^2}{\sigma_X^2}L^H)^{-1} \in \mathbb{R}^{n\times n}$*, where* L^H *is the Laplacian matrix of the graph* G^H*. Then we have for all honest user* $u^H \in U$*:*

$$\frac{var(X_u \mid \mathcal{I})}{var(X_u)} = 1 - e_u^\top M e_u. \tag{6}$$

We now provide an detailed interpretation of the above result. First note that the proportion of preserved variance only depends on the interactions between honest users, making the guarantees robust to any adversarial values that malicious users may send. Furthermore, notice that matrix M can be seen as a smoothing operator over the graph G^H. Indeed, given some $y \in \mathbb{R}^n$, $My \in \mathbb{R}^n$ is the solution to the following optimization problem:

$$\min_{s\in\mathbb{R}^n} \|s-y\|_2^2 + \alpha s^\top L^H s, \tag{7}$$

where $\alpha = \sigma_\delta^2/\sigma_X^2 \geq 0$. This problem is known as *graph smoothing* (also graph regularization, or graph filtering) and has been used in the context of semi-supervised learning (see Zhu and Ghahramani, 2002; Zhou et al., 2003), multi-task learning (Evgeniou and Pontil, 2004) and signal processing on graphs (Shuman et al.,

[2]We use a bounded noise assumption for simplicity. The argument easily extends to boundedness with high probability.

2013), among others. The first term in (7) encourages solutions that are close to y while the second term is the Laplacian quadratic form $s^\top L^H s = \sum_{(u,v)\in E^H}(s_u - s_v)^2$ which enforces solutions that are smooth over the graph (the larger α, the more smoothing). One can show that M is a doubly stochastic matrix (Segarra et al., 2015), hence the vector My sums to the same quantity as the original vector y. In our context, we smooth the indicator vector e_u so $e_u^\top M e_u \in [0,1]$: the larger the noise variance σ_δ^2 and the more densely connected the graph of honest neighbors, the more "mass" is propagated from user u to other nodes, hence the closer to 0 the value $e_u^\top M e_u$ and in turn the more variance of the private value X_u is preserved. Note that we recover the expected behavior in the two extreme cases: when $\sigma_\delta^2 = 0$ we have $e_u^\top M e_u = 1$ and hence the variance ratio is 0, while when $\sigma_\delta^2 \to \infty$ we have $e_u^\top M e_u \to 1/|U_H|$ and hence $var(X_u \mid \mathcal{I})/var(X_u) = 1 - 1/|U_H|$. This irreducible variance loss accounts for the fact that the adversary learns from $\mathcal{I}$ the average over the set of honest users (by adding their noisy values and subtracting the noise terms he knows). Since we assume the number of users to be very large, this variance loss can be considered negligible. We emphasize that in contrast to the lower bounds on the variance of the maximum likelihood estimate obtained by Mo and Murray (2014), our variance computation (6) is exact, interpretable and holds under non-uniform prior beliefs of the adversary.

It is important to note that smoothing occurs over the entire graph G^H: by transitivity, all honest users in the network contribute to keeping the private values safe. In other words, GOPA turns the massively distributed setting into an advantage for privacy, as we illustrate numerically in Section 6. Note, still, that one can derive a simple (but often loose) lower bound on the preserved variance which depends only on the local neighborhood.

Proposition 2. *Let $u \in U^H$ and denote by $|N^H(u)|$ the number of honest neighbors of u. We have:*

$$var(X_u \mid \mathcal{I}) \geq \sigma_X^2 \cdot \frac{\sigma_\delta^2(\left|N^H(u)\right| + 1)}{\sigma_X^2 + \sigma_\delta^2(|N^H(u)| + 1)} \cdot \frac{\left|N^H(u)\right|}{|N^H(u) + 1|}.$$

Proposition 2 shows that the more honest neighbors, the larger the preserved variance. In particular, if $\sigma_\delta^2 > 0$, we have $var(X_u \mid \mathcal{I}) \to \sigma_X^2$ as $|N^H(u)| \to \infty$.

We conclude this subsection with some remarks.

Remark 4 (Composition)**.** *As can be seen from inspecting the proof of Theorem 1, $P(X_u \mid \mathcal{I})$ is a Gaussian distribution. This makes our analysis applicable to the setting where* GOPA *is run several times with the same private value X_u as input, for instance within an iterative machine learning algorithm. We can easily keep track of the preserved variance by recursively applying Theorem 1.*

Remark 5 (G^H not connected)**.** *The results above still hold when Assumption 1 is not satisfied. In this case, the smoothing occurs separately within each connected component of G^H, and the irreducible variance loss is ruled by the size of the connected component of G^H that the user belongs to (instead of the total number of honest users $|U_H|$).*

Remark 6 (Drop out)**.** *The use of centered Gaussian noise with bounded variance effectively limits the impact of some users dropping out during the randomization phase. In particular, any residual noise term has expected value 0, and can be bounded with high probability. Alternatively, one may ask each impacted user to remove the noise exchanged with users who have dropped (thereby ensuring exact convergence at the expense of reducing privacy guarantees).*

4.4 Robust Strategies for Network Construction

We have seen above that the convergence rate and most importantly the privacy guarantees of GOPA crucially depend on the network topology. In particular, it is crucial that the network graph $G = (U, E)$ is constructed in a robust manner to ensure good connectivity and to guarantee that all honest users have many honest neighbors with high probability. In the following, we assume users have an address list for the set of users and can select some of them as their neighbors. Note that the randomization phase can be conveniently executed when constructing the network.

Algorithm 3 Verification procedure for the randomization phase of GOPA

Input: each user u has generated its own Paillier encryption scheme and has published:
- Public key K_u^{pub}, $\varepsilon_u^P(X_u)$ (before the execution of Algorithm 2)
- Noise values $\varepsilon_u^P(\delta_{u,v})$ (when exchanging with $v \in N(u)$ during Algorithm 2)
- $\varepsilon_u^P(\widetilde{X}_u)$, $\varepsilon_u^P(\Delta_u)$ (at the end of Algorithm 2)

for all user $u \in U$ **do**
  $\varepsilon^{\Delta} \leftarrow \prod_{v \in N(u)} \varepsilon_u^P(\delta_{u,v})$, $\varepsilon^{\widetilde{X}} \leftarrow \varepsilon_u^P(X_u) \cdot \varepsilon_u^P(\Delta_u)$
  Verify that $\varepsilon_u^P(\Delta_u) = \varepsilon^{\Delta}$ and $\varepsilon_u^P(\widetilde{X}_u) = \varepsilon^{\widetilde{X}}$ (if not, add u to cheater list)
end for
for all user $u \in U$ **do**
  Draw random subset V_u of $N(u)$ with $|V_u| = \lceil (1-\beta) d_u \rceil$, ask u to publish $\delta_{u,v}$ and $r_{\delta_{u,v}}$ for $v \in V_u$
  for all $v \in V_u$ **do**
    Ask v to publish $r_{\delta_{v,u}}$, verify that $\varepsilon_v^P(\delta_{v,u}) = \varepsilon_v(-\delta_{u,v})$ and $\varepsilon_u^P(\delta_{u,v}) = \varepsilon_u(\delta_{u,v})$ (if not, add u, v to cheater list)
  end for
end for

A simple choice of network topology is the complete graph, which is best in terms of privacy (since each honest user has all $|U_H| - 1$ other honest users as neighbors) and convergence rate (best connectivity). Yet this is not practical when n is very large: beyond the huge number of pairwise communications needed for the randomization phase, each user also needs to create and maintain $n - 1$ secure connections, which is costly (Chan et al., 2003).

We propose instead a simple randomized procedure to construct a sparse network graph with the desired properties based on *random k-out random graphs* (Bollobás, 2001), also known as *random k-orientable graphs* (Fenner and Frieze, 1982). The idea is to make each (honest) user select k other users uniformly at random from the set of all users. Then, the edge $(u, v) \in E$ is created if u selected v or v selected u (or both). This procedure is the basis of a popular key predistribution scheme used to create secure peer-to-peer communication channels in distributed sensor networks (Chan et al., 2003). Fenner and Frieze (1982) show that for any $k \geq 2$, the probability that the graph is k-connected goes to 1 almost surely. This provides robustness against malicious users. Note also that the number of honest neighbors of a honest node follows a hypergeometric distribution and is tightly concentrated around its expected value $k(1 - f)$, where f is the proportion of malicious users. It is worth noting that the probability that the graph is connected is actually very close to 1 even for small n and k. This is highlighted by the results of Yağan and Makowski (2013), who established a rather tight lower bound on the probability of connectivity of random k-out network graphs. For instance, the probability is guaranteed to be larger than 0.999 for $k = 2$ and $n = 50$. The algebraic connectivity $\lambda_2(L)$ is also large in practice for random k-out graphs, hence ensuring good convergence as per Proposition 1.

5 Verification Procedure

We have shown in Section 4.2 that GOPA converges to the appropriate average when users are assumed not to tamper with the protocol by sending incorrect values. In this section, we complement our algorithm with a verification procedure to detect malicious users who try to influence the output of the algorithm (we refer to them as *cheaters*). While it is impossible to force a user to give the "right" input to the algorithm (no one can force a person to answer honestly to a survey),[3] our goal is to make sure that given the input vector X, the protocol will either output X^{avg} or detect cheaters with high probability. Our approach is

[3]We can use range proofs (Camenisch et al., 2008) to check that each input value lies in an appropriate interval without revealing anything else about the value.

based on asking users to publish some (potentially encrypted) values during the execution of the protocol.[4] The published information should be publicly accessible so that anyone may verify the validity of the protocol (avoiding the need for a trusted verification entity), but should not threatens the privacy.

In order to allow public verification without compromising privacy, we will rely on the Paillier encryption scheme described in Section 3.2. For the purpose of this section, we assume that all quantities (private values and noise terms) are in $\mathbb{Z}_N$. This is not a strong restriction since with appropriate N and scaling one can represent real numbers from a given interval with any desired finite precision. Each user u generates its own Paillier scheme and publishes encrypted values using its encryption operation $\varepsilon_u(\cdot)$. These published cypher texts may not be truthful (when posted by a malicious user), hence we denote them by the superscript P to distinguish them from a valid encryption of a value (e.g., $\varepsilon_u^P(v)$ is the cypher text published by u for the quantity v).

Algorithm 3 describes our verification procedure for the randomization phase of Gopa (the averaging phase can be verified in a similar fashion). In a first step, we verify the coherence of the publications of each user u, namely that $\Delta_u = \sum_{v \in N(u)} \delta_{u,v}$ and $\widetilde{X}_u = X_u + \Delta_u$ are satisfied for the published cypher texts of these quantities. To allow reliable equality checks between cipher texts, some extra care is needed (see supplementary material for details). The second step is to verify that during a noise exchange, the user and his neighbor have indeed used opposite values as noise terms. However, each user has his own encryption scheme so one cannot meaningfully compare cypher texts published by two different users. To address this issue, we ask each user u to publish *in plain text* a random selected fraction $(1-\beta) \in [0,1]$ of his noise values $(\delta_{u,v})_{v \in N(u)}$ (the noise values to reveal are drawn publicly). The following result lower bounds the probability of catching a cheater.

Proposition 3 (Verification)**.** *Let C be the number of times a user cheated during the randomization phase. If we apply the verification procedure (Algorithm 3), then the probability that a cheater is detected is at least $1 - \beta^{2C}$.*

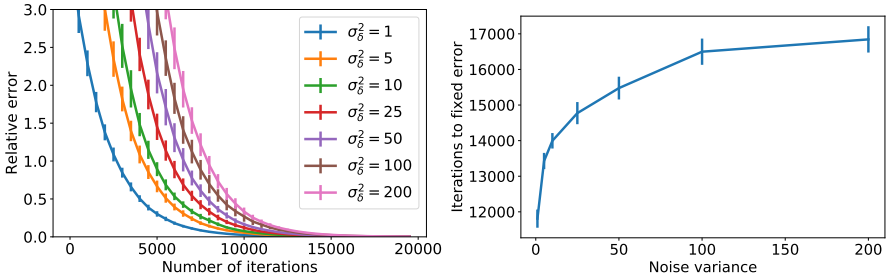
Proposition 3 shows that Algorithm 3 guards against large-scale cheating: the more cheating, the more likely at least one cheater gets caught. Small scale cheating is less likely to be detected but cannot affect much the final output of the algorithm (as values are bounded with high probability and the number of users is large). Of course, publishing a fraction of the noise values in plain text decreases the privacy guarantees: publishing $\delta_{u,v}$ in plain text for $u, v \in U^H$ corresponds to ignoring the associated edge of G^H in our privacy analysis. If the community agrees on β in advance, this effect can be easily compensated by constructing a network graph of larger degree, see Section 4.4.

6 Numerical Experiments

In this section, we run some simulations to illustrate the practical behavior of Gopa. In particular, we study two aspects: the impact of the noise variance on the convergence and on the proportion of preserved data variance, and the influence of network topology. In all experiments, the private values are drawn from the normal distribution $\mathcal{N}(0,1)$, and the network is a random k-out graph (see Section 4.4).

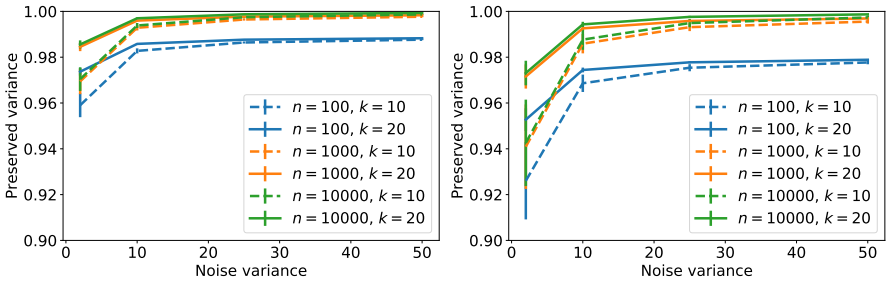
Figure 1 illustrates the impact of the noise variance on the convergence of the averaging phase of Gopa, for $n = 1000$ users and $k = 10$. We see on Figure 1(a) that larger σ_δ^2 have a mild effect on the convergence rate. Figure 1(b) confirms that the number of iterations t needed to reach a fixed error $\|\widetilde{X}(t) - X^{avg}\mathbf{1}\|/\|X\|$ is logarithmic in σ_δ^2, as shown by our analysis (Proposition 1). Figure 2 shows the proportion of preserved data variance for several topologies and proportions of malicious nodes in the network. Recall that in random k-out graphs with fixed k, the average degree of each user is roughly equal to $2k$ and hence remains constant with n. The results clearly illustrate one of our key result: beyond the noise variance and the number of honest neighbors, the *total* number of honest users has a strong influence on how much variance is preserved. The more users, the more smoothing on the graph and hence the more privacy, without any additional cost

[4] We assume that users cannot change their published values after publication. This could be enforced by relying on blockchain structures as in Bitcoin transactions (Nakamoto, 2008).



(a) Relative error w.r.t. # iter. (b) # iter. to 10^{-2} error w.r.t. σ_δ^2

Figure 1: Impact of the noise variance on the convergence of GOPA for a random k-out graph with $n = 1000$ and $k = 10$, with mean and standard deviation over 10 random runs.



(a) 10% of malicious nodes (b) 50% of malicious nodes

Figure 2: Preserved data variance (mean and standard deviation across users) w.r.t. noise variance for several random k-out topologies and two proportions of malicious nodes.

in terms of memory or computation for each individual user since the number of neighbors to interact with remains constant. This effect is even more striking when the proportion of malicious users is large, as in Figure 2(b). Remarkably, for n large enough, GOPA can effectively withstand attacks from many malicious users.

7 Conclusion

We proposed and analyzed a massively distributed protocol to privately compute averages over the values of many users, with robustness to malicious parties. Our novel privacy guarantees highlight the benefits of the large-scale setting, allowing users to effectively "hide in the crowd" by distributing the knowledge of their private value across many other parties. We believe this idea to be very promising and hope to extend its scope beyond the problem of averaging. In particular, we would like to study how our protocol may be used as a primitive to learn complex machine learning models in a privacy-preserving manner.

Acknowledgments This research was partially supported by grant ANR-16-CE23-0016-01 and by a grant from CPER Nord-Pas de Calais/FEDER DATA Advanced data science and technologies 2015-2020. The work was also partially supported by ERC-PoC SOM 713626.

References

Aslett, L. J. M., Esperança, P. M., and Holmes, C. C. (2015). Encrypted statistical machine learning: new privacy preserving methods. Technical report, arXiv:1508.06845.

Bollobás, B. (2001). *Random Graphs*. Cambridge University Press.

Bonawitz, K., Ivanov, V., Kreuter, B., Marcedone, A., McMahan, H. B., Patel, S., Ramage, D., Segal, A., and Seth, K. (2017). Practical Secure Aggregation for Privacy-Preserving Machine Learning. In *CCS*.

Boyd, S., Ghosh, A., Prabhakar, B., and Shah, D. (2006). Randomized Gossip Algorithms. *IEEE/ACM Transactions on Networking*, 14(SI):2508–2530.

Camenisch, J., Chaabouni, R., and Shelat, A. (2008). Efficient protocols for set membership and range proofs. In *Proceedings of the 14th International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT)*, pages 234–252.

Chan, H., Perrig, A., and Song, D. X. (2003). Random Key Predistribution Schemes for Sensor Networks. In *S&P*.

Colin, I., Bellet, A., Salmon, J., and Clémençon, S. (2015). Extending Gossip Algorithms to Distributed Estimation of U-statistics. In *NIPS*.

Duchi, J. C., Jordan, M. I., and Wainwright, M. J. (2012). Privacy Aware Learning. In *NIPS*.

Dwork, C. (2006). Differential Privacy. In *ICALP*.

Dwork, C. (2008). Differential Privacy: A Survey of Results. In *TAMC*.

Evgeniou, T. and Pontil, M. (2004). Regularized multi-task learning. In *SIGKDD*.

Fenner, T. I. and Frieze, A. M. (1982). On the connectivity of random m-orientable graphs and digraphs. *Combinatorica*, 2(4):347–359.

Goldreich, O. (1998). Secure multi-party computation. *Manuscript. Preliminary version*.

Graepel, T., Lauter, K. E., and Naehrig, M. (2012). ML Confidential: Machine Learning on Encrypted Data. In *ICISC*.

Hanzely, F., Konečný, J., Loizou, N., Richtárik, P., and Grishchenko, D. (2017). Privacy Preserving Randomized Gossip Algorithms. Technical report, arXiv:1706.07636.

He, X., Machanavajjhala, A., and Ding, B. (2014). Blowfish privacy: tuning privacy-utility trade-offs using policies. In *SIGMOD*.

Huang, Z., Mitra, S., and Dullerud, G. (2012). Differentially private iterative synchronous consensus. In *ACM workshop on Privacy in the Electronic Society*.

Kairouz, P., Oh, S., and Viswanath, P. (2016). Extremal Mechanisms for Local Differential Privacy. *Journal of Machine Learning Research*, 17:1–51.

Kasiviswanathan, S. P. and Smith, A. (2014). On the 'Semantics' of Differential Privacy: A Bayesian Formulation. *Journal of Privacy and Confidentiality*, 6(1):1–16.

Kempe, D., Dobra, A., and Gehrke, J. (2003). Gossip-Based Computation of Aggregate Information. In *FOCS*.

Kifer, D. and Lin, B.-R. (2012). An Axiomatic View of Statistical Privacy and Utility. *Journal of Privacy and Confidentiality*, 4(1):5–46.

Kifer, D. and Machanavajjhala, A. (2011). No free lunch in data privacy. In *SIGKDD*.

Kifer, D. and Machanavajjhala, A. (2014). Pufferfish: A framework for mathematical privacy definitions. *ACM Transactions on Database Systems*, 39(1):3:1–3:36.

Li, N., Qardaji, W. H., Su, D., Wu, Y., and Yang, W. (2013). Membership privacy: a unifying framework for privacy definitions. In *CCS*.

Lindell, Y. and Pinkas, B. (2009). Secure Multiparty Computation for Privacy-Preserving Data Mining. *Journal of Privacy and Confidentiality*, 1(1):59–98.

Manitara, N. E. and Hadjicostis, C. N. (2013). Privacy-preserving asymptotic average consensus. In *ECC*.

Mo, Y. and Murray, R. M. (2014). Privacy preserving average consensus. In *CDC*.

Nakamoto, S. (2008). Bitcoin: A Peer-to-Peer Electronic Cash System. Available online at http://bitcoin.org/bitcoin.pdf.

Paillier, P. (1999). Public-key cryptosystems based on composite degree residuosity classes. In *EUROCRYPT*.

Segarra, S., Huang, W., and Ribeiro, A. (2015). Diffusion and Superposition Distances for Signals Supported on Networks. *IEEE Transactions on Signal and Information Processing over Networks*, 1(1):20–32.

Shuman, D. I., Narang, S. K., Frossard, P., Ortega, A., and Vandergheynst, P. (2013). The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *IEEE Signal Processing Magazine*, 30(3):83–98.

Tsitsiklis, J. N. (1984). *Problems in decentralized decision making and computation*. PhD thesis, Massachusetts Institute of Technology.

Yağan, O. and Makowski, A. M. (2013). On the Connectivity of Sensor Networks Under Random Pairwise Key Predistribution. *IEEE Transactions on Information Theory*, 59(9):5754–5762.

Yao, A. C. (1982). Protocols for secure computations. In *FOCS*.

Zhou, D., Bousquet, O., Lal, T. N., Weston, J., and Schölkopf, B. (2003). Learning with Local and Global Consistency. In *NIPS*.

Zhu, X. and Ghahramani, Z. (2002). Learning from labeled and unlabeled data with label propagation. Technical Report CMU-CALD-02-107, Carnegie Mellon University.

SUPPLEMENTARY MATERIAL

Appendix A Proof of Proposition 1

The randomization phase consists in pairs of users adding noise terms which sum to zero. Hence, in the HBC setting, we have $\sum_{u\in U}\sum_{v\in N(u)}\delta_{u,v}=0$. Therefore, the sum of the user values over the network remains unchanged after the randomization phase:

$$\sum_{u\in U}\widetilde{X}_u=\sum_{u\in U}\left[X_u+\sum_{v\in N(u)}\delta_{u,v}\right]=\sum_{u\in U}X_u.$$

The first claim then follows from the correctness of the averaging procedure of Boyd et al. (2006) used for the averaging phase.

From Boyd et al. (2006), we know that the τ-averaging time of the averaging phase applied to the original (non-noisy) values X is $3\log(1/\tau)/\log(1/C_G)$. Hence, to achieve the same guarantee (4), we need to run the algorithm for at least

$$t(\tau)=\frac{3\log\left(\frac{1}{\tau}\frac{\|\widetilde{X}\|}{\|X\|}\right)}{\log(1/C_G)} \tag{8}$$

iterations. Due to the bound on the noise values and the amount of noise exchanges a user can make, we have for any $u\in U$:

$$B_X-B_\delta(d_{max}+3)\leq|\widetilde{X}_u|\leq B_X+B_\delta(d_{max}+3),$$

and hence

$$\frac{\|\widetilde{X}\|}{\|X\|}\leq\frac{B_\delta(d_{max}+3)}{B_X}.$$

We get the second claim by plugging this inequality into (8). □

Appendix B Proof of Theorem 1

We first introduce some auxiliary notations. Let $E^H_<=\{(u,v)\in E^H\mid u<u'\}$ be the ordered edges between honest users. Let $\mathcal{V}^{(X)}=(X_u)_{u\in U^H}$ be the vector of the private values of all honest users and $\mathcal{V}^{(\delta)}=(\delta_{u,v})_{(u,v)\in E^H_<}$ be the vector of all values exchanged by these users. Let

$$\mathcal{V}=(\mathcal{V}^{(X)},\mathcal{V}^{(\delta)})\in\mathbb{R}^{|U^H|+|E^H_<|}$$

be the concatenation of these two vectors. We will index vectors and matrices with elements of $\mathcal{V}$. To emphasize that we refer to the random variable rather than its value, we will use square brackets. For $u,v\in U^H$, we define $\xi(u,v)=1$ and $\zeta(u,v)=(u,v)$ if $u<v$, and $\xi(u,v)=-1$ and $\zeta(u,v)=(v,u)$ if $u\geq v$.

We formalize the knowledge $\mathcal{I}$ of the adversary by a set of linear equations, specifying the constraints the elements of $\mathcal{V}$ must satisfy according to its knowledge:

$$X_u+\textstyle\sum_{v\in N^H(u)}\xi(u,v)\delta_{\zeta(u,v)}=X^H_u,\qquad\text{for all } u\in U^H, \tag{9}$$

where $X^H_u=\widetilde{X}_u-\sum_{v\in N(U)\setminus N^H(u)}\delta_{u,v}$ is the noisy value of user X_u minus the sum of all noise terms exchanged with malicious users.

Let A be the matrix representing the above set of linear equations, and b the vector representing the right hand side, so that we can write (9) as $A\mathcal{V}=b$. In particular, A is a sparse $|U_H|\times(|U_H|+|E^<_H|)$ matrix whose elements are zero except for $\forall u\in U^H: A_{u,[X_u]}=1$ and $\forall(u,v)\in E^H_<: A_{u,[\delta_{u,v}]}=1$ and $A_{v,[\delta_{u,v}]}=-1$.

The matrix A has full rank and can be written as $A=\begin{bmatrix}I_{|U^H|}&B^H\end{bmatrix}$ where $I_{|U^H|}$ is the $|U^H|\times|U^H|$ identity matrix and B^H is the oriented incidence matrix of the graph G^H (where the direction of edges is given by

$E^H_<$). Let C be a full rank $|E^H_<| \times (|U^H| + |E^H_<|)$ matrix such that $AC^\top = 0$. Let $T = \begin{bmatrix} A \\ C \end{bmatrix}$. As A and C are non-singular and orthogonal to each other, T is also non-singular. Let Σ be a diagonal matrix with $\forall u \in U^H, \Sigma_{[X_u],[X_u]} = \sigma_X^2$ and $\forall (u,v) \in E^H_<, \Sigma_{[\delta_{u,v},\delta_{u,v}]} = \sigma_\delta^2$. We know that $var(\mathcal{V}) = \Sigma$. $T\mathcal{V}$ is a linear transformation of a multivariate Gaussian, and hence a multivariate Gaussian itself. In particular, it holds that $var(T\mathcal{V}) = T\Sigma T^\top$.

We now consider the random vector $C\mathcal{V}$ conditioned on $A\mathcal{V} = b$. This again gives a Gaussian distribution, with

$$\begin{aligned} var(C\mathcal{V}|A\mathcal{V}=b) &= var(C\mathcal{V}) - cov(C\mathcal{V}, A\mathcal{V})(var(A\mathcal{V}, A\mathcal{V}))^{-1} cov(A\mathcal{V}, C\mathcal{V}) \\ &= (C\Sigma C^\top) - (C\Sigma A^\top)(A\Sigma A^\top)^{-1}(A\Sigma C^\top) \\ &= C(\Sigma - \Sigma A^\top (A\Sigma A^\top)^{-1} A\Sigma) C^\top. \end{aligned}$$

Let $u \in U^H$ some honest user. We now focus on the variance of the private value X_u conditioned on the information obtained by the adversary:

$$\begin{aligned} var(X_u \mid A\mathcal{V}=b) &= var(e_u^\top \mathcal{V} | A\mathcal{V}=b) \\ &= e_u^\top C^\dagger C(\Sigma - \Sigma A^\top (A\Sigma A^\top)^{-1} A\Sigma) C^\top (C^\top)^\dagger e_u \\ &= \sigma_X^2 - e_u^\top \sigma_X^2 A^\top (A\Sigma A^\top)^{-1} A \sigma_X^2 e_u \\ &= \sigma_X^2 - \sigma_X^4 e_u^\top A^\top (A\Sigma A^\top)^{-1} A e_u \\ &= \sigma_X^2 - \sigma_X^4 e_u^\top \begin{bmatrix} I \\ B^{H\top} \end{bmatrix} (\begin{bmatrix} I & B^H \end{bmatrix} \Sigma \begin{bmatrix} I \\ B^{H\top} \end{bmatrix})^{-1} \begin{bmatrix} I & B^H \end{bmatrix} e_u \\ &= \sigma_X^2 - \sigma_X^4 e_u^\top \begin{bmatrix} I \\ B^{H\top} \end{bmatrix} (\sigma_X^2 I + \sigma_\delta^2 B^H B^{H\top})^{-1} \begin{bmatrix} I & B^H \end{bmatrix} e_u \\ &= \sigma_X^2 - \sigma_X^4 e_u^\top (\sigma_X^2 I + \sigma_\delta^2 B^H B^{H\top})^{-1} e_u \\ &= \sigma_X^2 - \sigma_X^4 e_u^\top (\sigma_X^2 I + \sigma_\delta^2 L^H)^{-1} e_u, \end{aligned} \tag{10}$$

where L^H Laplacian matrix associated with G^H. Finally, for clarity we rewrite

$$\sigma_X^2 - \sigma_X^4 e_u^\top (\sigma_X^2 I + \sigma_\delta^2 L^H)^{-1} e_u = \sigma_X^2 [1 - e_u^\top (I + \frac{\sigma_\delta^2}{\sigma_X^2} L^H)^{-1} e_u].$$

Appendix C Proof of Proposition 2

To better understand Theorem 1, we investigate a fictional scenario where the only noise exchanged in the network is between the user of interest u and his neighbors. It is clear that the amount of variance preserved in this scenario gives a lower bound for the general case (where pairs of nodes not involving u also exchange noise). Indeed, noise exchanges involving a malicious user have no effect (they are subtracted away in the linear system (9)), while those between honest users can only increase the uncertainty for the adversary.

Without loss of generality, assume $u = 1$. The Laplacian matrix L^H in Theorem 1 (ignoring the nodes which did not exchange noise) is given by

$$L = \begin{bmatrix} |N^H(u)| & -1 & -1 & \dots & -1 \\ -1 & 1 & 0 & \dots & 0 \\ -1 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ -1 & 0 & 0 & \dots & 1 \end{bmatrix}.$$

Let V be a unitary $(|N^H(u)|+1)\times(|N^H(u)|+1)$ matrix for which

$$\begin{aligned} V_{:,1}^\top &= \frac{1}{\sqrt{|N^H(u)+1|^2-|N^H(u)|-1}}\left(|N^H(u)|,-1,\ldots,-1\right), \\ V_{:,2}^\top &= \frac{1}{\sqrt{|N^H(u)|+1}}\left(1,\ldots,1\right). \end{aligned}$$

We have

$$L = V\,\mathrm{diag}\left(|N^H(u)|+1,0,1,\ldots,1\right)V^\top.$$

Indeed, we can easily verify that $LV_{:,1}=(|N^H(u)|+1)V_{:,1}$ and $LV_{:,2}=0$. For every vector $V_{:,i}$ with $i\geq 3$, we can check that $V_{1,1}^2+V_{1,2}^2=1$ and hence $V_{1,i}=0$. It follows that for any $j\geq 2$, $L_{j,:}V_{:,i}=V_{j,i}$. We know that $V_{:,i}$ is orthogonal to $V_{:,2}$ and hence $\sum_{j=2}^{|N^H(u)|+1}V_{j,i}=0$. Then, $L_{1,:}V_{:,i}=-\sum_{j=2}^{|N^H(u)|+1}V_{j,i}=0=V_{1,i}$. Therefore, all vectors orthogonal to $V_{:,1}$ and $V_{:,2}$ are eigenvectors with eigenvalue 1.

We can now rewrite the matrix inverse in (10) from the proof of Theorem 1 as:

$$\begin{aligned} \left(\sigma_X^2 I+\sigma_\delta^2 L^H\right)^{-1} &= \left(\sigma_X^2 VV^\top+\sigma_\delta^2 V\,\mathrm{diag}\left(|N^H(u)|+1,0,1,\ldots,1\right)V^\top\right)^{-1} \\ &= V\left(\mathrm{diag}(\sigma_X^2)+\sigma_\delta^2\,\mathrm{diag}\left(|N^H(u)|+1,0,1,\ldots,1\right)\right)^{-1}V^\top \\ &= V\left(\mathrm{diag}\left(\sigma_X^2+\sigma_\delta^2(|N^H(u)|+1),\sigma_X^2,\sigma_X^2+\sigma_\delta^2,\ldots,\sigma_X^2+\sigma_\delta^2\right)\right)^{-1}V^\top \\ &= V\,\mathrm{diag}\left(\left(\sigma_X^2+\sigma_\delta^2(|N^H(u)|+1)\right)^{-1},\sigma_X^{-2},\left(\sigma_X^2+\sigma_\delta^2\right)^{-1},\ldots,\left(\sigma_X^2+\sigma_\delta^2\right))^{-1}\right)V^\top \end{aligned}$$

We are interested in user $u=1$ so we focus on the value

$$\begin{aligned} \left((\sigma_X^2 I+\sigma_\delta^2 L)^{-1}\right)_{1,1} &= V_{:,1}\,\mathrm{diag}\left(\left(\sigma_X^2+\sigma_\delta^2(|N^H(u)|+1)\right)^{-1},\sigma_X^{-2},\left(\sigma_X^2+\sigma_\delta^2\right)^{-1},\ldots,\left(\sigma_X^2+\sigma_\delta^2\right))^{-1}\right)V_{:,1}^\top \\ &= V_{1,1}^2\left(\sigma_X^2+\sigma_\delta^2(|N^H(u)|+1)\right)^{-1}+V_{1,2}^2\sigma_X^{-2} \\ &= \left(\sigma_X^2+\sigma_\delta^2(|N^H(u)|+1)\right)^{-1}\frac{1}{(|N^H(u)|+1)^2-|N^H(u)|-1}\left(|N^H(u)|\right)^2+\sigma_X^{-2}\frac{1}{|N^H(u)|+1} \\ &= \left(\sigma_X^2+\sigma_\delta^2(|N^H(u)|+1)\right)^{-1}\frac{|N^H(u)|}{|N^H(u)|+1}+\sigma_X^{-2}\frac{1}{|N^H(u)|+1}. \end{aligned}$$

Plugging back in (10), we finally get:

$$\begin{aligned} var(X_u|A\mathcal{V}=b) &= \sigma_X^2-\sigma_X^4\left[\left(\sigma_X^2+\sigma_\delta^2(|N^H(u)|+1)\right)^{-1}\frac{|N^H(u)|}{|N^H(u)|+1}+\sigma_X^{-2}\frac{1}{|N^H(u)|+1}\right] \\ &= \sigma_X^2-\sigma_X^4\left(\sigma_X^2+\sigma_\delta^2(|N^H(u)|+1)\right)^{-1}\frac{|N^H(u)|}{|N^H(u)|+1}-\sigma_X^2\frac{1}{|N^H(u)|+1} \\ &= \sigma_X^2\left[1-\sigma_X^2\left(\sigma_X^2+\sigma_\delta^2(|N^H(u)|+1)\right)^{-1}\right]\frac{|N^H(u)|}{|N^H(u)+1|} \\ &= \sigma_X^2\left[\frac{\sigma_\delta^2|(N^H(u)|+1)}{\sigma_X^2+\sigma_\delta^2(|N^H(u)|+1)}\right]\frac{|N^H(u)|}{|N^H(u)|+1}. \end{aligned}$$

Appendix D Details on Verification Procedure

D.1 Reliable Equality Checks

To allow reliable equality checks between cipher texts, some extra care is needed. For any $u\in U$ and $v\in N(u)$, let us denote by $r_{\delta_{u,v}}$, r_{Δ_u}, r_{X_u} and $r_{\widetilde{X}_u}$ the random integers generated (and kept private) by user u to

respectively encrypt $\delta_{u,v}$, Δ_u, X_u and $\widetilde{X}_u$ using formula (2). User u should generate these random integers so as to satisfy the following relation (to simplify notations we assume that $N(u) = \{1, \dots, d_u\}$):

$$\begin{aligned} r_{X_u}, r_{\Delta_u}(0), r_{\delta_{u,k}} &\in \mathbb{Z}_N^*, \quad \forall k \in \{1, \dots, d_u\}, \\ r_{\Delta_u}(k) &= r_{X_u}(k-1) \cdot r_{\delta_{u,k}}, \quad \forall k \in \{1, \dots, d_u\}, \\ r_{\Delta_u} &= r_{\Delta_u}(d_u) \bmod N, \\ r_{\widetilde{X}_u} &= r_{X_u} \cdot r_{\Delta_u} \bmod N. \end{aligned}$$

It is easy to see that the equality checks can then be reliably performed. Crucially, the $r_{\delta_{u,k}}$'s are random so some new randomness is added at each iteration of the above recursion, guaranteeing that the encrypted values of the noise terms and the noise sum Δ_u are perfectly secure. Note on the other hand that the encryption of the noisy value $\widetilde{X}_u$ may not be perfectly secure, which is not a concern as our privacy guarantees of Section 4.3 hold under the knowledge of $\widetilde{X}_u$.

D.2 Proof Sketch of Proposition 3

We can assume that users published all the required values (this can be easily verified). The first part of Algorithm 3 verifies the coherence of the publications, namely that for all u, $\widetilde{X}_u = X_u + \sum_{v \in N(u)} \delta_{u,v}$ is satisfied for the published encrypted version of these quantities. If the coherence test passes, then the only way for two malicious users u and v to influence the final average is by adding noise terms such that $\delta_{u,v} \neq -\delta_{v,u}$.

Assume u and v indeed cheated. The probability that neither u or v is forced to publish the corresponding noise value is β^2. Assume either one of them, say u, is forced to publish his value $\delta_{u,v}$ in plain text. We can use it to check whether the cypher texts published by both u and v are indeed encrypted versions of $\delta_{u,v}$ and $-\delta_{u,v}$ respectively. If the verification fails, then one of them (or both) cheated. Otherwise, this noise exchange did not affect the average. We have shown that each time a user cheats, this is detected with probability at least $1 - \beta^2$. Then if C is the number of times that some user cheated, the probability of not catching any cheater is β^{2C}, and the result follows.