Accessibility of covariance information creates vulnerability in Federated Learning frameworks

Manuel Huth [1,2], Jonas Arruda [2], Roy Gusinow [1,2],
Lorenzo Contento [2], Evelina Tacconelli [3], and Jan Hasenauer [1,2,*]

[1] Helmholtz Zentrum München - German Research Center for Environmental Health, Institute of Computational Biology, 85764 Neuherberg, Germany
[2] University of Bonn, Life and Medical Sciences Institute, 53115 Bonn, Germany.
[3] University of Verona, Department of Diagnostics and Public Health, Division of Infectious Diseases, 37124 Verona, Italy.

Abstract

Federated Learning (FL) is gaining traction in various fields as it enables integrative data analysis without sharing sensitive data, such as in healthcare. However, the risk of data leakage caused by malicious attacks must be considered. In this study, we introduce a novel attack algorithm that relies on being able to compute sample means, sample covariances, and construct known linearly independent vectors on the data owner side. We show that these basic functionalities, which are available in several established FL frameworks, are sufficient to reconstruct privacy-protected data. Additionally, the attack algorithm is robust to defense strategies that involve adding random noise. We demonstrate the limitations of existing frameworks and propose potential defense strategies analyzing the implications of using differential privacy. The novel insights presented in this study will aid in the improvement of FL frameworks.

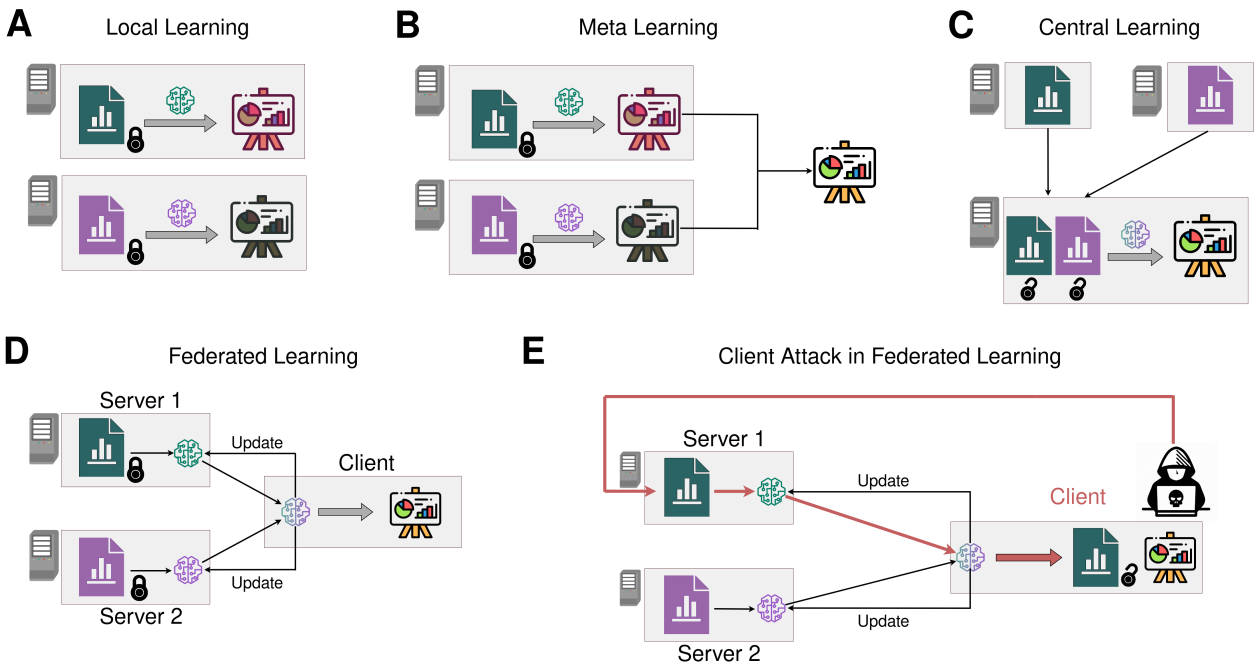

Figure 1: **Concept of (attacks in) Federated Learning. A** In Local Learning, all models are trained separately on different servers. **B** In Meta Learning, all models are trained separately on different servers but individual results are subsequently averaged to obtain meta results. **C** In Central Learning, the data is pooled and one model is trained. Hence the data must be shared. **D** In Federated Learning, the data is kept private on the servers. One model is trained with continuous updates between the client and the servers. **E** Illustration of a client side attack in Federated Learning. A malicious client uses the information received from the server to retrieve private data. This figure has been designed using resources from Flaticon.com.

Main Text

Large-scale data sets have been shown to be highly valuable for data-driven discovery in various fields such as clinical research [1–4], self-driving cars [5, 6], and smartphone keyboard word predictions [7, 8]. The COVID-19 pandemic has highlighted the importance of the rapid acquisition of new evidence for interventions in public health. Yet, data are often collected by different sides, e.g. hospitals, and established legal frameworks limit direct sharing [9], reducing the speed and statistical power of the analyses with possibly harmful consequences for patients [10]. To facilitate the integrative analysis of distributed data sets, federated learning has been introduced by Google Researchers in 2016 [11]. This supposedly allows for privacy-preserving estimation of statistical models from distributed data, making it an

*jan.hasenauer@uni-bonn.de

essential tool for the rapid assessment of new treatments to improve the fast acquisition of evidence-based interventions in public health. Security is a key topic in the field, as data leakage can result in deontological and consequentialist privacy harms [12].

Federated learning is based on sharing informative summary statistics by individual data owners (each running a data server) with a central hub (Figure 1D). This central hub is responsible for model building. Servers do not share individuals' data but only non-disclosive summary statistics. This approach is considered privacy-preserving. The focus on privacy in these areas has naturally precipitated extensive research on potential attack vectors. In particular, that sharing parameter gradients – a particular type of summary statistic – in deep neural structures can reveal the training data [13–17]. Algorithms were able to recreate images and texts [16, 17]. Further, data leakage threats have been summarized [18].

In this study, we complement the previous work by focusing on attacks based on basic functionalities that are available in established Federated Learning frameworks. We consider the possibilities of a malicious client who tries to obtain the data stored across different data owners and introduce a new attack concept. To perform the attack, we generate known linearly independent vectors on each server. After concatenating them on the client side, we use sample means and sample covariances to reconstruct the server side data nearly up to numerical precision (Fig 2). In contrast to the well-studied gradient approach, the presented method requires comparatively little time, and no model knowledge. Moreover, our algorithm carries desirable statistical properties: Repeated execution allows for exact data reconstruction even if random additive noise is applied to the covariance and means. In our opinion, this combination of features makes this attack strategy more problematic than any approach outlined previously. We discuss our algorithm theoretically and demonstrate its use in the open-source frameworks R DataSHIELD (version 6.2.0) [19], TensorFlow Federated (version 0.36.0) [20] and implemented a proof-of-concept for further platforms using PyTorch [21].

Covariance-Based Attack Algorithm

The distributed infrastructure consists of n_h servers. The j-th server hosts observations $s = 1, \ldots, n_j$ (e.g. patient data sets), each with information for variables $k = 1, \ldots, n_p$. Accordingly, each server stores a data matrix $X_j = \begin{pmatrix} x_{j,1} & \ldots & x_{j,k} & \ldots & x_{j,n_p} \end{pmatrix}$ with $x_{j,k} \in \mathbb{R}^{n_j}$, where each vector $x_{j,k}$ contains information about the variable k for all n_j samples on the server j. Without loss of generality, the malicious client focuses on a specific variable k on a specific server j, denoted by $x_{j,k}$. Retrieving the remaining variables and servers can be subsequently obtained analogously. We assume that the attacker has at least three basic tools: (T1) a sample mean function $\mathrm{Mean}(x)$, (T2) a sample covariance function $\mathrm{Cov}(x, y)$,

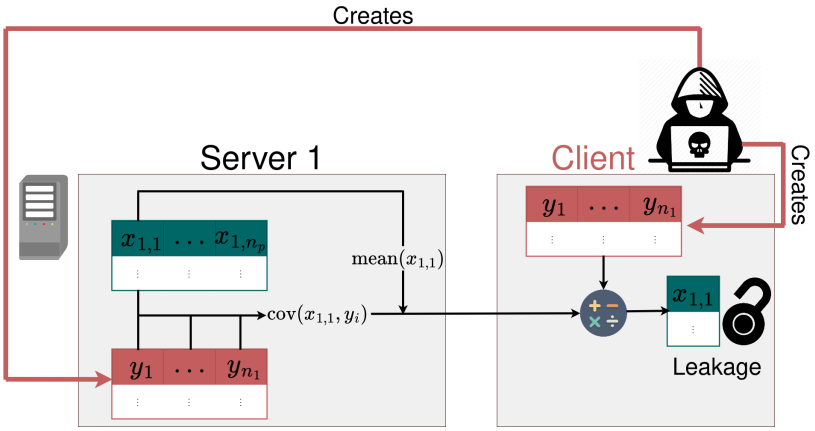

Figure 2: **Covariance-Based Attack Algorithm setup** for reconstructing data $x_{1,1}$ on the first server. The malicious client generates linear independent vectors $y_1, \ldots, y_{n_1}$ on the server and client side, computes the covariances of them together with the attacked vector $x_{1,1}$, and returns them with the mean of $x_{1,1}$ to the client side. Subsequently, the returned information is used with the means of $y_1, \ldots, y_{n_1}$ to compute $x_{1,1}$ on the client side. The algorithm can be repeated for all $x_{j,k}$ to obtain the full data set. This figure has been designed using resources from Flaticon.com.

and (T3) an algorithm $\mathcal{A}$ generating n_j linearly independent vectors $y_i \in \mathbb{R}^{n_j}$ on the server side and their column-wise collection as a matrix $Y \in \mathbb{R}^{n_j \times n_j}$ on the client side, $\mathcal{A}(n_j) = \begin{pmatrix} y_1 & \ldots & y_i & \ldots & y_{n_j} \end{pmatrix} = Y$ with $y_i \in \mathbb{R}^{n_j}$ (Figure 2).

These requirements are met by many distributed analysis frameworks, virtually all of which include functions for computing sample means (T1) and covariances (T2). The input of the covariance function is usually not restricted to subsets of the data matrix X but allows for other inputs y (T2). The availability of a function for the construction and sharing of linearly independent vectors (T3) might seem the least obvious, but it is available in most tools. For instance, it is necessary in the context of optimisation via federated averaging: The client receives the server side gradients, updates the parameters, and sends them back to the servers. Therefore, for any system where this operation is possible, assumption (T3) must be satisfied, since we need to be able to sent vectors from the client side to the server side. Well-known and widely used distributed analysis frameworks for which assumptions (T1)—(T3) are fulfilled are TensorFlow Federated [20] and DataSHIELD [19].

Assuming that (T1)—(T3) are met, the centerpiece of our algorithm is the fact that evaluating the sample covariance makes it possible to reconstruct the inner vector products between the attacked vector $x_{j,k}$ and the linearly independent vectors $y_1, \ldots, y_{n_j}$. This yields a linear

Algorithm 1 Covariance-Based Attack Algorithm

Input: Position of attacked server side variable $x_{j,k}$

Output: Retrieved data $x_{j,k}$

Require: Data matrix X_j on the server side, function $\text{Mean}(x)$ returning the sample mean, function $\text{Cov}(x, x')$ returning the sample covariance, algorithm $\mathcal{A}(n_j)$ returning n_j known linearly independent vectors $y_i \in \mathbb{R}^{n_j}$ on the server side and their column-wise collection as a matrix Y on the the client side

procedure
- $Y, y_1, \ldots, y_{n_j} \leftarrow \mathcal{A}(n_j)$ ▷ Client and server side
- initialise $\tilde{V}, \tilde{m} \in \mathbb{R}^{n_j}$ ▷ Client side
- **for** i in $1 : n_j$ **do**
  - $\tilde{m}[i] \leftarrow \text{Mean}(y_i)$ ▷ Client side
  - $\tilde{V}[i] \leftarrow \text{Cov}(x_{j,k}, y_i)$ ▷ Client side
- $x \leftarrow (n_j - 1) \left(Y^T\right)^{-1} \tilde{V} + n_j \text{ Mean}(x_{j,k}) \left(Y^T\right)^{-1} \tilde{m}$ ▷ Client side

return x

system of n_j equations which can be solved for $x_{j,k}$ and written in matrix form as

$$x_{j,k} = (n_j - 1) \cdot \left(Y^T\right)^{-1} \underbrace{\begin{pmatrix} \text{Cov}\left(x_{j,k}, y_1\right) \\ \vdots \\ \text{Cov}\left(x_{j,k}, y_{n_j}\right) \end{pmatrix}}_{:=\tilde{V}} + n_j \cdot \text{Mean}(x_{j,k}) \cdot \left(Y^T\right)^{-1} \underbrace{\begin{pmatrix} \text{Mean}(y_1) \\ \vdots \\ \text{Mean}(y_{n_j}) \end{pmatrix}}_{:=\tilde{m}}, \quad (1)$$

where the right-hand side of (1) is known by the malicious client. Derivations of the computations are reported in the section Mathematical Computations of the Methods. The presented procedure can be repeated for each variable $k = 1, 2, \ldots, n_p$ and server $j = 1, 2, \ldots, n_S$, using the same or newly generated linearly independent vectors y_i, until all data $X_1, \ldots, X_{n_s}$ is obtained. As the covariance calculation is essential, we refer to the strategy as *Covariance-Based Attack Algorithm*.

DataSHIELD is vulnerable to the algorithm

To demonstrate the Covariance-Based Attack Algorithm and the vulnerability of existing distributed analysis frameworks, we considered different software packages. First, we provide an example implementation in DataSHIELD (version 6.2.0) and its base package dsBaseClient [19]. This tool is well established and used in various biomedical applications [3, 22–25] in which data sharing is limited, e.g. to ensure compliance with privacy regulations, such as the General Data Protection Regulation (GDPR). The data set utilized is the CNSIM data set

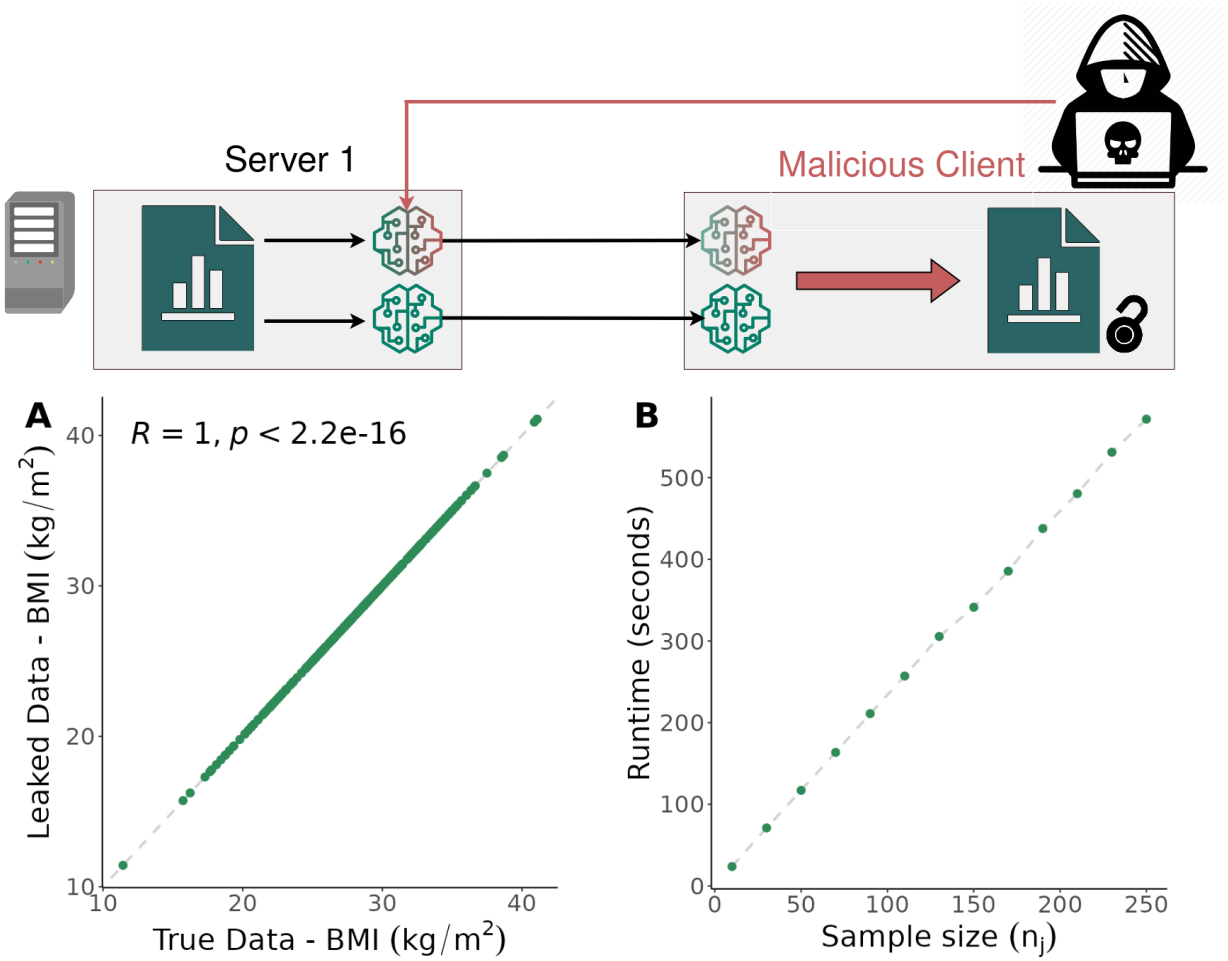

Figure 3: **Leakage results and computation times** for DataSHIELD. **A** True data values from the first server of the CNSIM data set vs. the corresponding retrieved data provided by the Covariance-Based Attack Algorithm. **B** Computation time of the algorithm for different sample size. This figure has been designed using resources from Flaticon.com.

from the DataSHIELD tutorial [26] to ensure an easy-to-reproduce test case. This data set consists of 3 servers with a total of 9,379 synthetic observations of 11 personalised obesity-related variables. We have reconstructed information on individual body mass index (BMI) measurements from the first server using a complete case analysis with a sample size of $n_j = 250$.

DataSHIELD meets the requirements (T1)—(T3) and is therefore vulnerable to the Covariance-Based Attack Algorithm. The functions to compute sample means (T1) and sample covariances (T2) are `ds.mean` and `ds.cov`, respectively. These functions return the means and covariances directly, but require mild conditions on the attacked data $x_{j,k}$: (C1) the sample sizes n_j must exceed the thresholds $n_j > 3$ (`ds.mean`) and $n_j > 6$ (`ds.cov`); and (C2) both levels of a dichotomous variable must occur at least 3 times in the given data vectors. The conditions

(C1) and (C2) ensure a privacy-preserving analysis if the functions are applied once. If any of the assumptions (C1) and (C2) were violated, descriptive statistics or further analysis with $x_{j,k}$ would be impossible. Therefore, it is reasonable to assume their validity. In our example, the data have $n_j = 250$ observations of a continuous variable so that requirements (C1) and (C2) are clearly satisfied. The construction of n_j linearly independent vectors (T3) can be implemented in several ways. We used the function `ds.dmtC2S` to send client side matrices to the server side. Hence, it is possible for the client to create suitable linearly independent vectors y_i on the client side and to send them to the server side. Note that since the covariance operation is performed on $x_{j,k}$ and all y_i, (C1) and (C2) must hold for all y_i as well. Since $x_{j,k}$ and y_i have the same length n_j, (C1) holds. To meet (C2) and the linear independence condition, we draw each element y_i from a standard normal distribution, so that y_i almost surely consists of n_j distinct entries and that $y_1, \ldots, y_{n_j}$ are almost surely linearly independent. In principle, however, the malicious client can use any linearly independent vectors $y_1, \ldots, y_{n_j}$ that meet the requirements (C1) and (C2).

After creating all linearly independent y_i on the server and client side and computing the relevant means and covariances, the data can be obtained as described in (1). Our evaluation of the above example shows that the true data can be reconstructed almost perfectly (see Figure 3A). The Pearson correlation coefficient between true and retrieved BMI values is 1.0. The highest absolute error observed is $2 \cdot 10^{-12}$, which is close to numerical accuracy. This demonstrates that the Covariance-Based Attack Algorithm is not limited to theoretical settings. Instead, data leakage can also be achieved in real-life setups.

Computation complexity of data reconstruction grows linearly with sample size

To study the applicability of the Covariance-Based Attack Algorithm, we considered the scaling of the computation time with growing sample size n_j. As computation time, we consider the wall time required to obtain the result.

In theory, the sample size determines the time requirements in different ways. First, it determines the size of the system of equations (1). This size is identical to n_j, meaning that n_j requests must be sent to the j-th server. The communication overhead for a request is constant, but the computation time will generally grow linearly with n_j $[\mathcal{O}(n_j)]$, as the dimensionality of the scalar product increases. Secondly, the computation time for solving the linear system from (1) grows cubically $\left[\mathcal{O}(n_j^3)\right]$ using the `solve` command in R and the `linalg.inv` command in Python available through the NumPy library. Hence, there are linear, and cubic contributions, with different pre-factors, to the computation time.

To evaluate the scaling behavior in practice, we considered subsets of the CNSIM data set of different sizes and determined the wall time required to complete the attack (Figure 3B). We observed linear scaling (Figure 3B), meaning that the communication overhead determines overall wall time. Indeed, even for the largest data set considered, matrix inversion required only 0.004 seconds, meaning that it contributed only $7 \cdot 10^{-6}$ percent to the overall time.

The essentially linear scaling behavior in the relevant regime, compared to the theoretically cubic scaling behavior, leads to this attack being feasible in many real-world scenarios.

The Covariance-Based Attack Algorithm is robust against noise perturbations in DataSHIELD

The Covariance-Based Attack Algorithm allows for the reconstruction of the data on the servers. We further investigated whether our approach is robust to adding zero-mean noise to the means and covariances before returning them to the client. In this case, the client observes noise-corrupted data estimates

$$\begin{aligned} x_{j,k}^{noisy} &= (n_j - 1)\left(Y^T\right)^{-1}(\tilde{V} + \varepsilon) + n_j\left(Y^T\right)^{-1}(\mathrm{Mean}(x_{j,k}) + \gamma)\,\tilde{m} \\ &= x_{j,k} + (n_j - 1)\left(Y^T\right)^{-1}\varepsilon + n_j\left(Y^T\right)^{-1}\tilde{m}\gamma, \end{aligned}$$

with zero-mean and finite-variance noise terms ε and γ.

The noise-corrupted data estimate $x_{j,k}^{noisy}$ can be decomposed into the true data $x_{j,k}$ and a noise component so that the malicious client cannot retrieve the original data (Figure 4A). However, the malicious client is, given suitable communication and computational budgets, which is given for example in DataSHIELD, able to run the algorithm R times. If R is sufficiently large, the zero-mean noise components average out such that the mean $\frac{1}{R}\sum_{r=1}^{R} x_{j,k,r}^{noisy}$ converges in probability to the data $x_{j,k}$ (Figure 4B). We provide a proof in the Method section. Hence, even if noise is added to the means and covariances, a malicious client can retrieve the data if no further restrictions on the analyst are imposed.

TensorFlow Federated is vulnerable to the algorithm

To assess whether other tools allow for the implementation of similar attack strategies, we considered TensorFlow Federated (version 0.36.0)[20]. In contrast to DataSHIELD, this open-source framework for computations on decentralized data is meant for experimentation with Federated Learning. Yet, if experimentation environments allow for (non-trivial) disclosive computations, these are likely to find their way into application. Accordingly, we evaluated

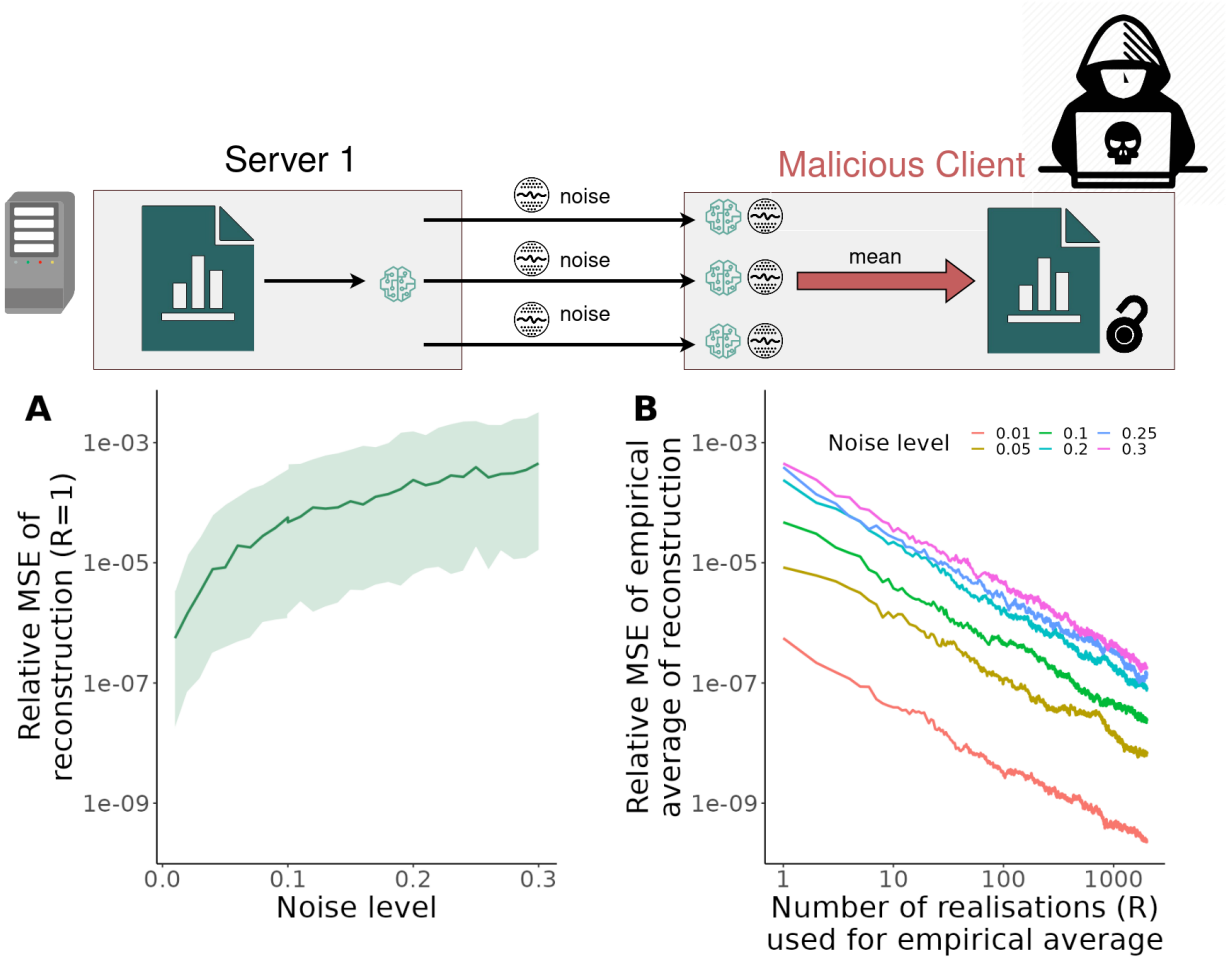

Figure 4: **Robustness of Covariance-Based Attack Algorithm** to normally distributed noise on means and covariances. **A** Relative mean squared error (MSE) of the reconstructed data values for different noise level if only a single realization is available ($R = 1$). The median (line) and the 5th to 95th-percentile (area) of 200 replicates are depicted. **B** Relative mean squared error (MSE) of the empirical mean of reconstructed data values obtained from different numbers of realization ($R = 1, \ldots, 1000$) and four different noise levels. The median (line) of 200 replicates is depicted. This figure has been designed using resources from Flaticon.com.

the possibility of implementing the Covariance-Based Attack Algorithm using a set of basic functions.[1]

Our assessment revealed that TensorFlow Federated meets the tool requirements (T1)—(T3) and allows for the implementation of the Covariance-Based Attack Algorithm. Functions can

[1]Note that the developers of TensorFlow Federated use the terms client and server in a way opposite to that of the DataSHIELD community. To avoid confusing the reader, we stick to the convention of DataSHIELD, with the client being the central hub and the servers being the data owners.

be constructed in TensorFlow Federated by wrapping functionalities from Python packages, e.g. TensorFlow or numpy, in a function and labelling it with `tf_computation`. To compute sample means (T1), a function that computes the average of $x_{j,k}$, e.g. using `numpy.mean`, can be implemented. For (T2), one can for instance wrap the function `stats.covariance` from the TensorFlow probability package. Both functions need to be applied with the functionality of `federated_map` to return values from the server side. Since TensorFlow Federated does not enforce further privacy leakage checks, these functions do not have requirements that are equivalent to (C1) and (C2) for DataSHIELD. However, we expect that if TensorFlow Federated is used in real-world applications, further disclosiveness checks, similar to (C1) and (C2), will be implemented. For (T3), TensorFlow Federated offers the `tff.federated_broadcast` function which is similar to the function `ds.dmtC2S` as it sends objects from the client to the server side. Due to the current lack of requirements such as (C1) and (C2), the vectors $y_1, \ldots, y_{n_j}$ must be linearly independent but no further restrictions have to be imposed.

The implementation of the Covariance-Based Attack Algorithm in TensorFlow Federated was applied to the aforementioned CNSIM data set. We found that this allows for a reconstruction of the data up to numerical accuracy (Supplementary Figure 6). Hence, data leakage is also possible in TensorFlow Federated, using algorithms that appear to be non-disclosive. This raises questions regarding the suitability of the framework for experimentation with Federated Learning.

PyTorch based platforms used in real-world applications are vulnerable to the algorithm

Further Federated Learning tools, such as Nvidia Flare [27], IBM Federated Learning [28], Intel's OpenFL [29], and PySyft [30], are widely used in real-world applications that handle sensitive data. For example, Nvidia Flare is used by Microsoft Azure, American College of Radiology, Rhino Health and others as indicated on their website.[2] These tools advertise their flexibility in using popular Python-based deep learning packages like PyTorch [21]. In light of the potential for malicious implementation of algorithms, we evaluated the feasibility of implementing the Covariance-Based Attack Algorithm on these platforms.

Our analysis demonstrated that the platforms under examination fulfill the necessary tool requirements (T1)–(T3), enabling the implementation of a Covariance-Based Attack algorithm. Users of these platforms have the ability to specify their models in PyTorch, Tensorflow, or Numpy, and the platforms subsequently integrate the specified models into a federated

[2] https://developer.nvidia.com/flare

learning workflow so that the final model specification is ultimately determined by the client. To compute sample means (T1) one can apply `torch.mean` on the variables of interest. Regarding tool requirement (T2), one could use the PyTorch function `torch.cov`. However, no direct algorithm for generating linear independent vectors (T3) is available. However, linear regression models `torch.nn.Linear` that incorporate the attacked variable as the dependent variable and the randomly generated vectors (T3) as independent variables, since users have full control over specifying any type of model in PyTorch. Hence, one can easily specify multiple linear models, thus incorporating the generation of (T3) in the specification of the model. The least squares estimate of β for the model $x_{j,k} = y_i\beta + \varepsilon$ can be represented as $\hat{\beta} = \frac{\mathrm{Cov}(x_{j,k}, y_i)}{\mathrm{Var}(y_i)}$. By rearranging this expression, we can express the covariance as $\mathrm{Cov}(x_{j,k}, y_i) = \hat{\beta}\,\mathrm{Var}(y_i)$, where $\hat{\beta}$ is obtained through linear regression and $\mathrm{Var}(y_i)$ is calculated easily on the client side.

The examined platforms do not enforce (C1) and (C2) as built-in security measures. However, additional security features are implemented, for example, models must undergo a review process prior to being utilized with real-world data. Our work can improve this process by raising the awareness to potential disclosive models. Moreover, the execution of algorithms might be limited by a privacy budget in accordance with differential privacy principles, as outlined in [31].

Differential privacy limits the algorithms use but precludes standard analysis

Our proposed algorithm, the Covariance-Based Attack, demonstrates effective data retrieval from platforms that do not implement computational privacy budgets. To assess the impact of such budgets on our method, we evaluated its robustness by testing its performance when platforms enforce differential privacy restrictions on the client's analysis. Differential privacy improves security by adding appropriate noise to the output of an algorithm, $\mathcal{A}$, when applied to a data set, $\mathcal{D}$, in order to ensure that no individual in $\mathcal{D}$ can be directly identified [31]. The privacy budget, $\epsilon_{\mathcal{A}}$, which controls the level of noise added, is allocated to the analyst and must be used with caution, as it limits the number of algorithm calls that can be made by the analyst.

The privacy budget consumed by the Covariance-Based Attack algorithm, denoted as ϵ_{att}, can be upper-bounded by the sum of the individual privacy budgets allocated for the computation of covariances and means

$$\epsilon_{\mathrm{att}} = \sum_{i=1}^{n_j} \epsilon_{\mathrm{cov}(x_{j,k}, y_i)} + \epsilon_{\mathrm{mean}(x_{j,k})}. \tag{2}$$

Detailed mathematical descriptions of the privacy budget consumption of the mean $\epsilon_{\mathrm{mean}(x_{j,k})}$

and covariances $\epsilon_{\text{cov}(x_{j,k},y_i)}$ are given in the Methods section.

By specifying a privacy budget, data owners face the trade-off between more function calls or increasing security. More function calls could result in more informative outcomes but increases the risk of data leakage, whereas less function calls increase security but could limit the statistical results. We examined this trade-off using the CNSIM data set. First, we computed the possible number of runs of the Covariance-Based Attack Algorithm depending on the privacy budget, and examined how the relative error of reconstruction changes with the privacy budget (Figure 5A,B). We varied the privacy budget parameter, ϵ, between ln(1.01) and ln(3), based on the range defined in [31]. Second, we computed covariances (Figure 5C) between variables of the CNSIM data to see how much privacy budget is consumed by standard descriptive statistics. Third, we analyzed the consumption of the privacy budget for a basic analysis workflow with means, and (Co-)Variances of two variables (Figure 5D) examining if a standard analysis is still feasible under the restrictions of differential privacy.

Increasing the noise variance of the outputs of the covariances and means enables the analyst to perform more iterations of the Covariance-Based Attack algorithm, as shown in Figure 5A. However, even at the highest noise variance, it is not possible to obtain a sufficient number of runs of the algorithm to recover the true data by averaging out the noise, as demonstrated in Figure 5B. Interestingly, more noise leads in general to a higher accuracy of the algorithm for the same privacy budget ϵ since more iterations of the algorithm can compensate increasing noise as one can see in Figure 5B.

However, the allocated privacy budgets are not sufficient for a non-malicious analyst to perform covariance computations on the five metric variables of the CNSIM dataset. Even when using high noise variances, the computation of one covariance consumes already the whole privacy budget, as shown in Figure 5C. The standard analysis workflow could not be computed for all considered noise variances due to the consumed accumulated privacy budget, as illustrated in Figure 5D.

Discussion

Federated Learning has demonstrated its utility and importance in various fields, and its significance has been further highlighted during the SARS-CoV-2 pandemic, with many consortia utilizing it extensively [1, 32, 33]. However, it is crucial to ensure that the data of the participating servers remains protected. To accomplish this, a thorough examination of attack strategies is necessary. In this study, we introduce the novel Covariance-Based Attack Algorithm, which highlights the vulnerability of established Federated Learning systems.

We have demonstrated that a malicious client could leverage the Covariance-Based Attack Algorithm to extract data from a Federated Learning system. Our approach differs from

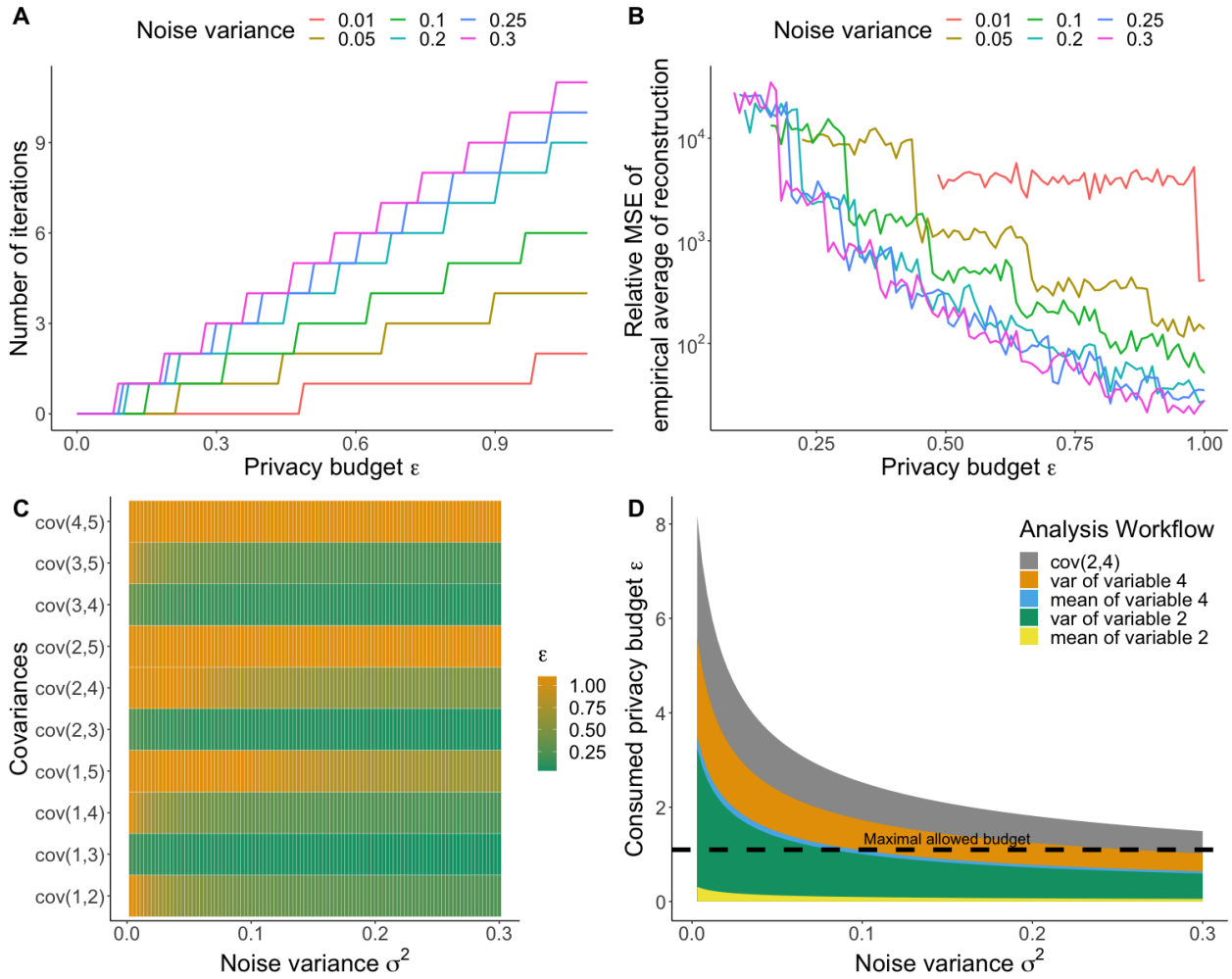

Figure 5: **A** Number of possible iterations of the Covariance-Based Attack Algorithm for different noise levels still satisfying ϵ-differential privacy using the CNSIM data set. **B** Error of the Covariance-Based Attack Algorithm after allowed numbers of iteration for different noise levels still satisfying ϵ-differential privacy using the CNSIM data set. The median (line) of 200 replicates is depicted. **C** Privacy budget consumed by computing the covariances between metric variables in the CNSIM data set for different noise levels. The consumed budget ϵ is truncated at the maximal allowed privacy budget ln(3). **D** The privacy budget consumed by performing a standard analysis on two variables (computing the mean and variance of two metric variables and their covariance).

previous studies that focused on information leakage through gradients obtained from deep neural models. It relies on constructing linearly independent vectors on the server side and utilizing sample means and sample covariance functions that are accessible to the client. The attack approach is efficient, scalable, and can be easily implemented, and it is not hindered by noise perturbations if the number of function calls is not restricted. However, implementing

differential privacy can prevent the algorithm but would also hinder the application of (linear) analysis tools making differential privacy impractical to use if rather standard tools like linear models should be used. Our algorithm has been successfully implemented on DataSHIELD (version 6.2.0) and TensorFlow Federated (version 0.36.0), where we were able to reconstruct the data. We have informed the respective developers. Furthermore, we have implemented a proof-of-concept using PyTorch (version 1.13.0) showing the usability of the algorithm on other Federated Learning platforms.

Our findings reveal that the existing functionalities of Federated Learning frameworks need to be scrutinized in regards to data leakage threats. To address this vulnerability, we propose to tackle the issue induced by (T3). An effective solution would be to immediately process these vectors within a function instead of creating a vector on the server side avoiding the possibility of computations with client-known objects.

It is important to note that the proposed strategy can only be executed if the attacker can act as the client, which typically requires obtaining log-in credentials. However, even with this access, the security of the data should not solely rely on the trustworthiness of the client, as the Covariance-Based Attack Algorithm demonstrates that data can still be retrieved. The need for Federated Learning would be eliminated if the security of the data is dependent on the trustworthiness of the client alone, as it could be replaced by Centralized Learning. Furthermore, this raises concerns about compliance with the General Data Protection Regulation (GDPR) and about responsibility and liability in the event of unknown attack strategies.

With this work, we contribute to the growing literature on data leakage problems in Federated Learning systems and aim to support research on Federated Algorithms and raise awareness about potential security risks. We did not study other distributed frameworks such as Swarm Learning, but we encourage a thorough review. Although security levels may appear to be higher as the aggregation of information is shared, an attack may still be possible if requirements (T1)—(T3) are met.

We anticipate that our results will play a role in developing design criteria for the architecture of Federated Learning platforms. We have shown that existing systems need to be enhanced to minimize the risk of data leaks.

Supplementary Information (code) is available for this paper. Correspondence and requests for materials should be addressed to Jan Hasenauer.

Methods

Proof of correctness for the Covariance-Based Attack Algorithm

In this study, we consider an attack by a malicious client and provide an algorithm for data reconstruction based on covariance information. In the following, we provide the mathematical derivation of the algorithm.

For the attack, it is necessary to compute sample means

$$\begin{aligned}\text{Mean}(x_{j,k}) &= \frac{1}{n_j}\sum_{s=1}^{n_j} x_{j,k}^{(s)},\\ \text{Mean}(y_i) &= \frac{1}{n_j}\sum_{s=1}^{n_j} y_i^{(s)},\end{aligned}$$

and sample covariances,

$$\begin{aligned}\text{Cov}\left(x_{j,k}, y_i\right) =& \frac{1}{n_j-1}\sum_{s=1}^{n_j}\left(x_{j,k}^{(s)} - \text{Mean}(x_{j,k})\right)\left(y_i^{(s)} - \text{Mean}(y_i)\right)\\ =& \frac{1}{n_j-1}\, y_i^T x_{j,k} - \frac{n_j}{n_j-1}\,\text{Mean}(x_{j,k})\,\text{Mean}(y_i),\end{aligned}$$

for all $i = 1, 2, \ldots, n_j$ on the server side and return them to the client side. The vectors y_i are chosen in a way to ensure their linear independence.

To reconstruct $x_{j,k}$, we exploit the fact that the sample covariances can be reformulated to determine the inner product $y_i^T x_{j,k}$,

$$y_i^T x_{j,k} = (n_j - 1)\,\text{Cov}\left(x_{j,k}, y_i\right) + n_j\,\text{Mean}(x_{j,k})\,\text{Mean}(y_i). \tag{3}$$

We combine the equations for $i = 1, \ldots, n_j$ from (3) to a system of equations in matrix form:

$$\begin{aligned}&\underbrace{\begin{pmatrix} y_1^T \\ \vdots \\ y_{n_j}^T \end{pmatrix}}_{Y^T} x_{j,k} =(n_j-1)\underbrace{\begin{pmatrix} \text{Cov}\left(x_{j,k}, y_1\right) \\ \vdots \\ \text{Cov}\left(x_{j,k}, y_{n_j}\right) \end{pmatrix}}_{=\tilde{V}_{j,k}} + n_j\,\text{Mean}(x_{j,k})\underbrace{\begin{pmatrix} \text{Mean}(y_1) \\ \vdots \\ \text{Mean}(y_{n_j}) \end{pmatrix}}_{=\tilde{m}}\\ &\Longrightarrow Y^T x_{j,k} =(n_j-1)\,\tilde{V}_{j,k} + n_j\,\text{Mean}(x_{j,k})\,\tilde{m}.\end{aligned}$$

Since the vectors y_i were chosen to be linearly independent, Y, and therefore also Y^T, are invertible. Hence, we can multiply both sides of equation (4) by the inverse of Y^T to obtain

$$x_{j,k} =(n_j-1)\left(Y^T\right)^{-1}\tilde{V} + n_j\left(Y^T\right)^{-1}\text{Mean}(x_{j,k})\,\tilde{m}, \tag{4}$$

where the right-hand side is known to the client. This provides a constructive proof for the recovery of $x_{j,k}$ using the proposed approach.

The same procedure can be repeated for all n_j servers and all n_p variables, yielding comprehensive information about potentially sensitive data on the servers.

Robustness of the Covariance-Based Attack Algorithm to noise perturbations in DataSHIELD

As a defense strategy against malicious clients, we consider the perturbation of mean and covariance with noise but without the additional use of differential privacy. More specifically, we consider the addition of zero-mean noise to means and covariances on the server side before sending them to the client side. Given only access to noisy data, one might assume that the client will not be able to reconstruct $x_{j,k}$ exactly. However, running the attack algorithm multiple times on the same variable and averaging these results yields a random variable that converges in probability to $x_{j,k}$ such that the malicious client is, given an appropriate communication and computational budget, able to retrieve all information about $x_{j,k}$. We prove that the empirical mean of the noisy results of the Covariance-Based Attack Algorithm $\frac{1}{R}\sum_{r=1}^{R} x_{j,k,r}^{noisy}$, with R denoting the number of calls in an attack, converges in probability to $x_{j,k}$, i.e., formally for any $c > 0$

$$\lim_{R\to\infty} \mathbb{P}\left(\left\|\underbrace{\frac{1}{R}\sum_{r=1}^{R} x_{j,k,r}^{noisy}}_{=\text{ empirical mean}} - x_{j,k}\right\|_2 \geq c\right) = 0, \tag{5}$$

where $x_{j,k,r}^{noisy}$ is the result of the r-th run of the Covariance-Based Attack Algorithm.

Let ε_r be an n_j dimensional random vector with mean $\mathbb{E}(\varepsilon_r) = 0$ and covariance matrix $\mathbb{V}(\varepsilon_r) = \sigma_\varepsilon^2 \mathbb{I}_{n_j}$ for which $\sigma_r^2 < \infty$. Let γ_r be a random variable with mean $\mathbb{E}(\gamma_r) = 0$ and variance $\mathbb{V}(\gamma_r) = \sigma_\gamma^2 < \infty$. Furthermore, let γ_r and ε_r be uncorrelated so that $\mathbb{E}(\gamma_r \cdot \varepsilon_r) = 0$. The noisy version of equation (4) is given by

$$\begin{aligned} x_{j,k,r}^{noisy} &= (n_j - 1)\left(Y^T\right)^{-1}(\tilde{V} + \varepsilon_r) + n_j\left(Y^T\right)^{-1}(\text{Mean}(x_{j,k}) + \gamma_r)\,\tilde{m} \\ &= x_{j,k} + \underbrace{(n_j - 1)\left(Y^T\right)^{-1}}_{:=A}\varepsilon_r + \underbrace{n_j\left(Y^T\right)^{-1}\tilde{m}}_{:=B}\gamma_r, \end{aligned} \tag{6}$$

such that $x_{j,k,r}^{noisy}$ can be decomposed into the true $x_{j,k}$ and a noise term. Combining (5) and (6) shows that (5) is proven if the mean of the noise term converges in probability to zero, such that it is sufficient to show that

$$\lim_{R\to\infty} \mathbb{P}\left(\left\|\frac{1}{R}\sum_{r=1}^{R}(A\varepsilon_r + B\gamma_r)\right\|_2 \geq c\right) = 0. \tag{7}$$

This can be shown by applying Markov's Inequality

$$\mathbb{P}\left(\left|\left|\frac{1}{R}\sum_{r=1}^{R}(A\varepsilon_r + B\gamma_r)\right|\right|_2 \geq c\right) < \frac{\mathbb{E}\left(\left|\left|\frac{1}{R}\sum_{r=1}^{R}(A\varepsilon_r + B\gamma_r)\right|\right|_2^2\right)}{c^2}. \tag{8}$$

Since (8) holds for all R, it is sufficient to show that the numerator of the right-hand side converges to 0 if $R \to \infty$ in order to prove (7). To facilitate notation, the entries of A^TA are denoted by $a^{(s,s')}$ and the entries of ε_r by $\varepsilon_r^{(s)}$. Note that the following holds:

- $\forall r, m : \mathbb{E}(\gamma_m \varepsilon_r^T A^T B) = \mathbb{E}(\gamma_m \varepsilon_r^T) A^T B = 0$,
- $\forall l \neq m$: by independence of ε_r and ε_m, $\mathbb{E}(\varepsilon_r^T A^T A \varepsilon_m) = \mathbb{E}(\varepsilon_r^T) A^T A \mathbb{E}(\varepsilon_m)) = 0$ and by independence of γ_r and γ_m that $\mathbb{E}(\gamma_r \gamma_m B^T B) = \mathbb{E}(\gamma_r)\mathbb{E}(\gamma_m) B^T B = 0$,
- $\forall r = m : \mathbb{E}(\varepsilon_r^T A^T A \varepsilon_m) = \sum_{s=1}^{n_j}\sum_{s=1}^{n_j} \mathbb{E}(\varepsilon_r^{(s)}\varepsilon_r^{(s')}) a^{(s,s')} = \sigma_\varepsilon^2 \sum_{s=1}^{n_j} a^{(s,s)}$ and $\mathbb{E}(\gamma_r \gamma_r B^T B) = \mathbb{E}(\gamma_r^2) B^T B = \sigma_\gamma^2 B^T B$.

The numerator of the right-hand side of (8) can therefore be written as

$$\begin{aligned}
\mathbb{E}\left(\left|\left|\frac{1}{R}\sum_{r=1}^{R}(A\varepsilon_r + B\gamma_r)\right|\right|_2^2\right) =& \mathbb{E}\left(\frac{1}{R^2}\sum_{r=1}^{R}\sum_{m=1}^{R}\left(\varepsilon_r^T A^T A \varepsilon_m + 2\gamma_m \varepsilon_r^T A^T B + \gamma_r \gamma_m B^T B\right)\right) \\
=& \frac{1}{R^2}\left(R\sigma_\varepsilon^2 \sum_{s=1}^{n_j} a^{(s,s)} + R\sigma_\gamma^2 B^T B\right) \\
=& \frac{1}{R}\left(\sigma_\varepsilon^2 \sum_{s=1}^{n_j} a^{(s,s)} + \sigma_\gamma^2 B^T B\right).
\end{aligned}$$

This is a constant multiplied by R^{-1}. Accordingly, (7) holds and therefore (5) is proven.

In the manuscript, we provide an analysis of the mean squared error for different numbers of calls from an attacker and different levels of noise. The relative mean squared error (RMSE) is here defined as

$$\text{RMSE} = \frac{\left|\left|\frac{1}{R}\sum_{r=1}^{R} x_{j,k,r}^{noisy} - x_{j,k}\right|\right|_2^2}{||x_{j,k}||_2^2}. \tag{9}$$

Derivation of privacy budget consumption of mean and covariances

The privacy budget consumption of the mean (covariance) algorithm is given by

$$\epsilon_{\mathcal{A}}^2 \geq \frac{2\ S_{\mathcal{A}}^2 \log(1.25\ n_j^2)}{\sigma^2},$$

where σ^2 is the noise level of the normally distributed noise that is added to the output of the mean (covariance). $S_{\mathcal{A}}$ is the algorithm dependent local sensitivity which is the maximum amount that the output of a function can change given a change of a single input. It therefore determines the amount of noise that needs to be added to the output of a function in order to ensure that it satisfies the privacy guarantees of differential privacy [31]. Mathematically, the local sensitivity $S_{\mathcal{A}}$ is defined as

$$S_{\mathcal{A}} = \max_{\mathcal{D}} |\mathcal{A}(\mathcal{D}) - \mathcal{A}(\mathcal{D}')| \\ \text{s.t. } ||\mathcal{D} - \mathcal{D}'||_1 = 1,$$

where the $L1$ norm of a data set indicates how many individuals have been added or removed from a data set. To compute the local sensitivity $S_{\text{mean(z)}}$ of the mean of a vector $z = \begin{pmatrix} z_1 & \dots & z_n \end{pmatrix}' \in \mathbb{R}^n$, we denote the maximum absolute deviation of any component of z from the mean by z^*

$$z^* = \arg \max_{z_j \in \{z_1, \dots, z_n\}} \left| z_j - \frac{1}{n} \sum_{i=1}^{n} z_i \right|.$$

The data set $\mathcal{D}$ equals the vector z and removing one entry from z or adding an entry to z yields the neighboring data set $\mathcal{D}'$. The local sensitivity of the mean becomes therefore

$$\begin{aligned} S_{\text{mean(z)}} &= \max_{z_j \in \{z_1, \dots, z_n\}} \left| \frac{1}{n} \sum_{i=1}^{n} z_i - \frac{1}{n-1} \sum_{\substack{i=1 \\ i \neq j}}^{n} z_i \right| \\ &= \max_{z_j \in \{z_1, \dots, z_n\}} \left| \frac{n-1}{n(n-1)} \sum_{i=1}^{n} z_i - \frac{n}{n(n-1)} \sum_{\substack{i=1 \\ i \neq j}}^{n} z_i \right| \\ &= \max_{z_j \in \{z_1, \dots, z_n\}} \left| \frac{n \sum_{i=1}^{n} z_i - \sum_{i=1}^{n} z_i - n \sum_{\substack{i=1 \\ i \neq j}}^{n} z_i}{n(n-1)} \right| \\ &= \max_{z_j \in \{z_1, \dots, z_n\}} \frac{1}{(n-1)} \left| z_j - \frac{1}{n} \sum_{i=1}^{n} z_i \right| \\ &= \frac{1}{(n-1)} \max_{z_j \in \{z_1, \dots, z_n\}} \left| z_j - \frac{1}{n} \sum_{i=1}^{n} z_i \right| \\ &= \frac{1}{(n-1)} \left| z^* - \frac{1}{n} \sum_{i=1}^{n} z_i \right|. \end{aligned}$$

For the local sensitivity $S_{\text{cov(w,z)}}$ of the covariances for z and another vector $w = \begin{pmatrix} w_1 & \dots & w_n \end{pmatrix}' \in$

$\mathbb{R}^n$, the data $\mathcal{D}$ equals the matrix $\begin{pmatrix} w & z \end{pmatrix}$. The optimization problem becomes therefore

$$S_{\mathrm{cov(w,z)}} = \max_{(w_j,z_j)\in\{(w_i,z_i)\}_{i=1}^n} \left| \frac{1}{n-1}\sum_{i=1}^{n}\left(z_i - \frac{1}{n}\sum_{k=1}^{n} z_k\right) w_i - \frac{1}{n-2}\sum_{\substack{i=1\\ i\neq j}}^{n}\left(z_i - \frac{1}{n-1}\sum_{\substack{k=1\\ k\neq j}}^{n-1} z_k\right) w_i \right|,$$

which we solve in our simulation study by evaluating all possible outcomes of the absolute value when removing any (w_j, z_j).

Implementation and code availability

A code example of our attack algorithm using the open source frameworks DataSHIELD (version 6.2.0) and TensorFlow Federated (version 0.36.0) together with the proof-of-concept using PyTorch (version 1.13.0) and the differential privacy analysis is provided at GitHub (`https://github.com/manuhuth/Data-Leakage-From-Covariances.git`).

Data availability

The CNSIM1 data set which we used in the manuscript is available within the DSData R package (`https://github.com/datashield/DSData/tree/main/data`).

Acknowledgments

We thank the Interdisciplinary Research Unit Mathematics and Life Sciences at the University of Bonn, Nina Schmid, and Marc Vaisband for comments and discussions.

Funding

This study was funded by the German Research Foundation (Deutsche Forschungsgemeinschaft, DFG) under Germany's Excellence Strategy (EXC 2047 - 390873048 & EXC 2151 - 390685813), the German Ministry for Education and Research (Deutches Bundesminsterium für Bildung und Forschung, BMBF) under the CompLS program (grant agreement No 031L0293C), the University of Bonn (via the Schlegel Professorship of JH), the Helmholtz Association - Munich School for Data Science (MUDS), and the ORCHESTRA project. The ORCHESTRA project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 101016167. The views expressed in this paper are the

sole responsibility of the authors and the Commission is not responsible for any use that may be made of the information it contains. The funders had no role in the study design, data collection, data analyses, data interpretation, writing, or submission of this manuscript.

Author contributions

M.H. developed the Covariance-Based Attack Algorithm. M.H., L.C. and J.H. proved the reconstruction accuracy. M.H. implemented the algorithm in DataSHIELD. R.G. implemented the algorithm in TensorFlow Federated. J.A. examined differential privacy and the tools Nvidia Flare, IBM federated learning, Intel's OpenFL, and PySyft. J.H and E.T. conceptualized the study. M.H., J.A., and J.H. wrote the manuscript. All authors read and approved the final manuscript.

Authors and affiliations

Helmholtz Zentrum München - German Research Center for Environmental Health, Institute of Computational Biology, Neuherberg, Germany
Manuel Huth, Roy Gusinow, Jan Hasenauer

University of Bonn, Life and Medical Sciences Institute, Bonn, Germany
Manuel Huth, Jonas Arruda, Roy Gusinow, Lorenzo Contento, Jan Hasenauer

University of Verona, Department of Diagnostics and Public Health, Division of Infectious Diseases, Verona, Italy
Evelina Tacconelli

Ethics declarations

The authors have no competing interests.

References

1. Dayan, I. **andothers**. Federated learning for predicting clinical outcomes in patients with COVID-19. *Nature Medicine* **27,** 1735–1743 (2021).
2. Harrison, S. L., Fazio-Eynullayeva, E., Lane, D. A., Underhill, P. & Lip, G. Y. Comorbidities associated with mortality in 31,461 adults with COVID-19 in the United States: A federated electronic medical record analysis. *PLoS Medicine* **17,** e1003321 (2020).

3. Jannasch, F. **andothers**. Associations between exploratory dietary patterns and incident type 2 diabetes: A federated meta-analysis of individual participant data from 25 cohort studies. *European Journal of Nutrition,* 1–19 (2022).

4. Li, X. **andothers**. Multi-site fMRI analysis using privacy-preserving federated learning and domain adaptation: ABIDE results. *Medical Image Analysis* **65,** 101765 (2020).

5. Pokhrel, S. R. & Choi, J. Federated learning with blockchain for autonomous vehicles: Analysis and design challenges. *IEEE Transactions on Communications* **68,** 4734–4746 (2020).

6. Posner, J., Tseng, L., Aloqaily, M. & Jararweh, Y. Federated learning in vehicular networks: Opportunities and solutions. *IEEE Network* **35,** 152–159 (2021).

7. Chen, M., Mathews, R., Ouyang, T. & Beaufays, F. Federated learning of out-of-vocabulary words. *arXiv.* 26 March 2019, preprint: not peer reviewed.

8. Yang, T. **andothers**. Applied federated learning: Improving google keyboard query suggestions. *arXiv.* 7 December 2018, preprint: not peer reviewed.

9. Hansen, J. **andothers**. Assessment of the EU Member States' rules on health data in the light of GDPR (2021).

10. Bentzen, H. B. **andothers**. Remove obstacles to sharing health data with researchers outside of the European Union. *Nature Medicine* **27,** 1329–1333 (2021).

11. McMahan, B., Moore, E., Ramage, D., Hampson, S. & y Arcas, B. A. *Communication-efficient learning of deep networks from decentralized data* **in** *Artificial Intelligence and Statistics* (2017), 1273–1282.

12. Price, W. N. & Cohen, I. G. Privacy in the age of medical big data. *Nature medicine* **25,** 37–43 (2019).

13. Geiping, J., Bauermeister, H., Dröge, H. & Moeller, M. Inverting gradients-how easy is it to break privacy in federated learning? *Advances in Neural Information Processing Systems* **33,** 16937–16947 (2020).

14. Huang, Y., Gupta, S., Song, Z., Li, K. & Arora, S. Evaluating gradient inversion attacks and defenses in federated learning. *Advances in Neural Information Processing Systems* **34,** 7232–7241 (2021).

15. Yin, H. **andothers**. *See through gradients: Image batch recovery via gradinversion* **in** *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2021), 16337–16346.

16. Zhao, B., Mopuri, K. R. & Bilen, H. idlg: Improved deep leakage from gradients. *arXiv.* 8 January 2020, preprint: not peer reviewed.

17. Zhu, L., Liu, Z. & Han, S. Deep leakage from gradients. *Advances in Neural Information Processing Systems* **32** (2019).

18. Lyu, L., Yu, H. & Yang, Q. Threats to federated learning: A survey. *arXiv.* 4 March 2020, preprint: not peer reviewed.

19. Marcon, Y. **andothers**. Orchestrating privacy-protected big data analyses of data from different resources with R and DataSHIELD. *PLoS Computational Biology* **17,** e1008880 (2021).

20. Martín Abadi **andothers**. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems* https://www.tensorflow.org/. (2015).

21. Paszke, A. **andothers**. **in** *Advances in Neural Information Processing Systems 32* 8024–8035 (Curran Associates, Inc., 2019).

22. De Moira, A. P. **andothers**. Associations of early-life pet ownership with asthma and allergic sensitization: A meta-analysis of more than 77,000 children from the EU Child Cohort Network. *Journal of Allergy and Clinical Immunology* (2022).

23. Pastorino, S. **andothers**. Heterogeneity of associations between total and types of fish intake and the incidence of type 2 diabetes: Federated meta-analysis of 28 prospective studies including 956,122 participants. *Nutrients* **13,** 1223 (2021).

24. Pearce, M. **andothers**. Associations of total legume, pulse, and soy consumption with incident type 2 diabetes: Federated meta-analysis of 27 studies from diverse world regions. *The Journal of Nutrition* **151,** 1231–1240 (2021).

25. Peñalvo, J. L. **andothers**. Unravelling data for rapid evidence-based response to COVID-19: A summary of the unCoVer protocol. *BMJ open* **11,** e055630 (2021).

26. Westerberg, A. & Wilson, R. *DataSHIELD training part 1: Introduction and logging in* `https://data2knowledge.atlassian.net/wiki/spaces/DSDEV/pages`, (11 March 2023, date last accessed). (2022).

27. Roth, H. R. **andothers**. NVIDIA FLARE: Federated Learning from Simulation to Real-World. *arXiv.* 6 December 2022, preprint: not peer reviewed.

28. Ludwig, H. **andothers**. Ibm federated learning: an enterprise framework white paper v0. 1. *arXiv.* 22 July 2020, preprint: not peer reviewed.

29. Foley, P. **andothers**. OpenFL: the open federated learning library. *Physics in Medicine Biology* **67,** 214001. `https://dx.doi.org/10.1088/1361-6560/ac97d9` (**october** 2022).

30. Ziller, A. **andothers**. **in** *Federated Learning Systems* 111–139 (Springer, 2021).

31. Dwork, C., Roth, A. **andothers**. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science* **9,** 211–407 (2014).

32. Tacconelli, E. **andothers**. Challenges of data sharing in European COVID-19 projects: A learning opportunity for advancing pandemic preparedness and response. *The Lancet Regional Health-Europe* **21,** 100467 (2022).

33. Warnat-Herresthal, S. **andothers**. Swarm learning for decentralized and confidential clinical machine learning. *Nature* **594,** 265–270 (2021).

Supplementary Figures

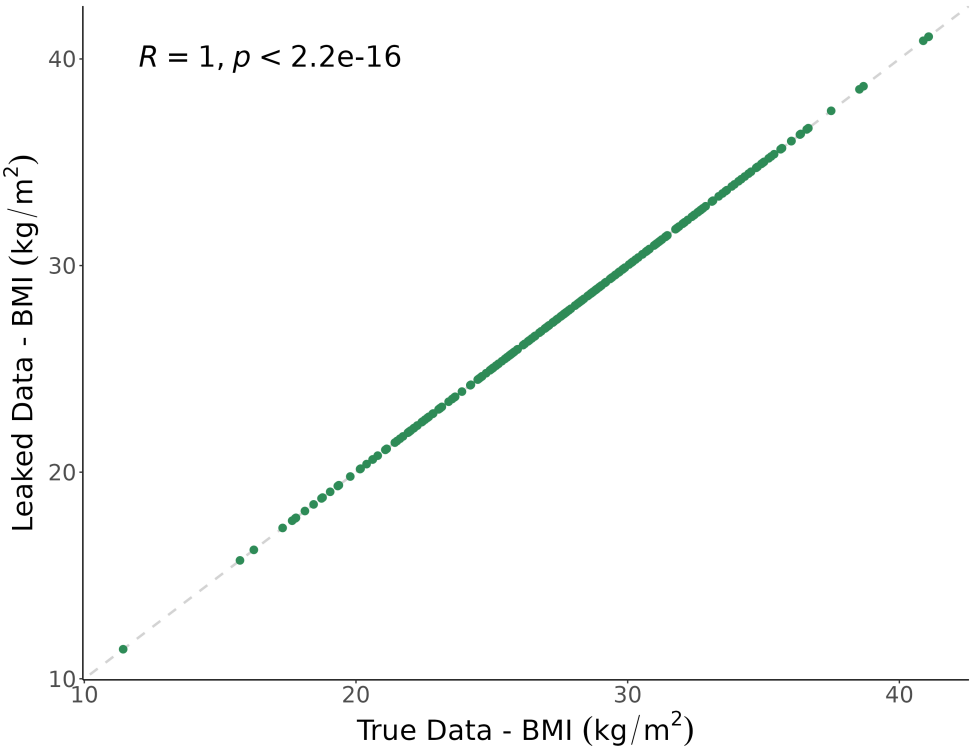

Figure 6: **Leakage results for TensorFlow Federated** are shown. The true data values from the first server of the CNSIM data set are plotted against the corresponding leaked data provided by the Covariance-Based Attack Algorithm.