Accessibility of covariance information creates vulnerability in Federated Learning frameworks

Manuel Huth [1,2], 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 approaches are becoming increasingly relevant in various fields. These approaches promise to facilitate an integrative data analysis without sharing the data, which is highly beneficial for applications with sensitive data such as healthcare. Yet, the risk of data leakage caused by malicious attacks needs to be assessed carefully. In this study, we consider a new attack route and present an algorithm that depends 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, available in several established Federated Learning frameworks, suffice to reconstruct privacy-protected data. Moreover, the attack algorithm is robust to defence strategies that build on random noise. We demonstrate this limitation of existing frameworks and discuss possible defence strategies. The novel insights will facilitate the improvement of Federated Learning frameworks.

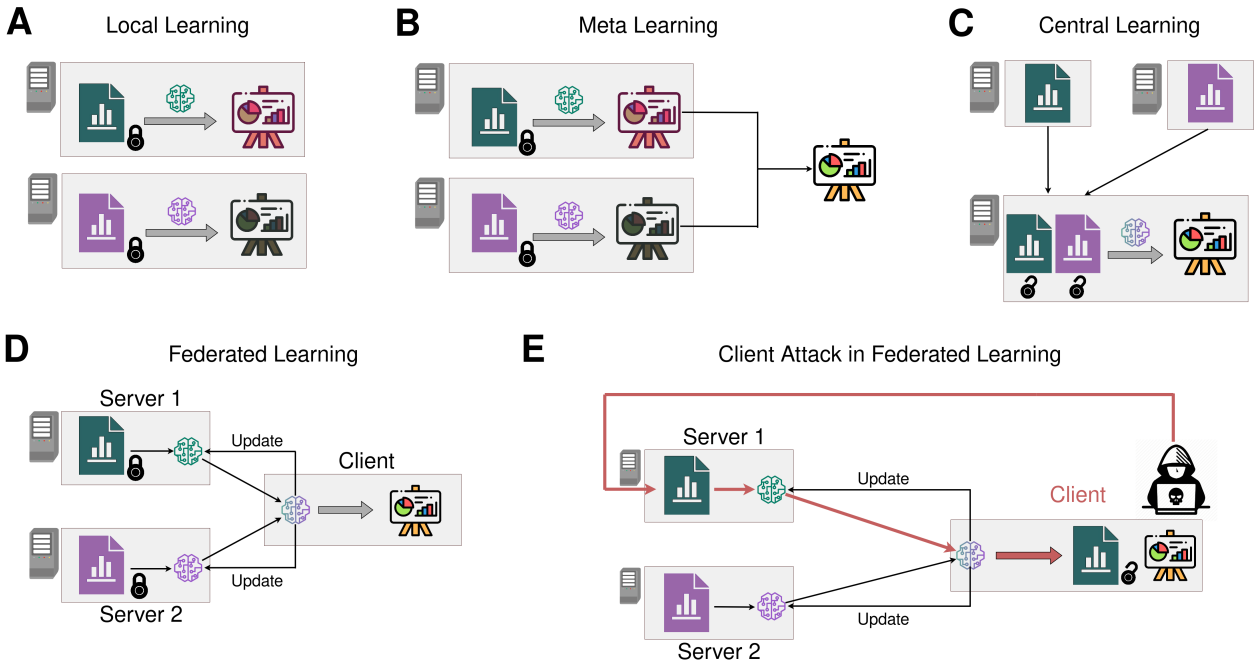

Figure 1: **Concept of (attacks in) Federated Learning.** **A** In Local Learning all models are trained separately on different servers. **B** In Meta Learning all models are trained separately on different servers but individual results are subsequently averaged to obtain meta results. **C** In Central 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

*jan.hasenauer@uni-bonn.de

for privacy-preserving estimation of statistical models from distributed data, making it an 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 (on the level of individual pixels) and texts [16, 17]. Further data leakage threats have been summarized [18].

In this study, we complement the previous work by focusing 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. 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: Repeatedly executing it 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 previously outlined approach. We discuss our algorithm theoretically and demonstrate its usage on the open-source frameworks R DataSHIELD (version 6.2.0) [19] and TensorFlow Federated (version 0.36.0) [20].

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 the information about variable k for all n_j samples on 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 $\mathrm{Cov}(x, y)$ for

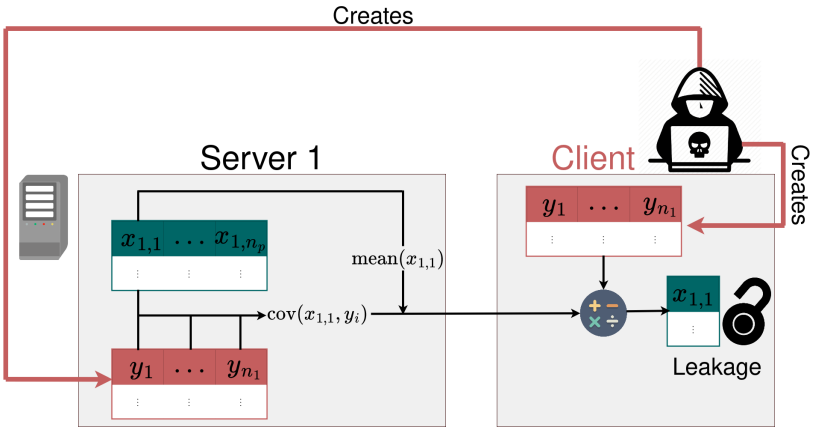

Figure 2: **Covariance-Based Attack Algorithm setup** for reconstructing data $x_{j,1}$ on the first server. The malicious client generates linear independent vectors $y_1, \ldots, y_{n_1}$ on the server and client side, computes the covariances of them together with the attacked vector $x_{1,1}$, and 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

a vector y provided by the client, 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 appear least obvious, but it is indeed 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. For any system where this operation is possible, assumption (T3) must therefore be satisfied. Well-known and widely used distributed analysis frameworks for which assumptions (T1)—(T3) are fulfilled are TensorFlow Federated [20] and R DataSHIELD [19].

Assuming that (T1)—(T3) are met, the centrepiece 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 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*.

R 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 R DataSHIELD (version 6.2.0) framework and its base package dsBaseClient [19]. This tool is well-established and used in various biomedical applications [3, 21–24] in which data sharing is limited, e.g. to ensure compliance with privacy regulations, such as the General Data Protection Regulation (GDPR). The utilised data set is the CNSIM

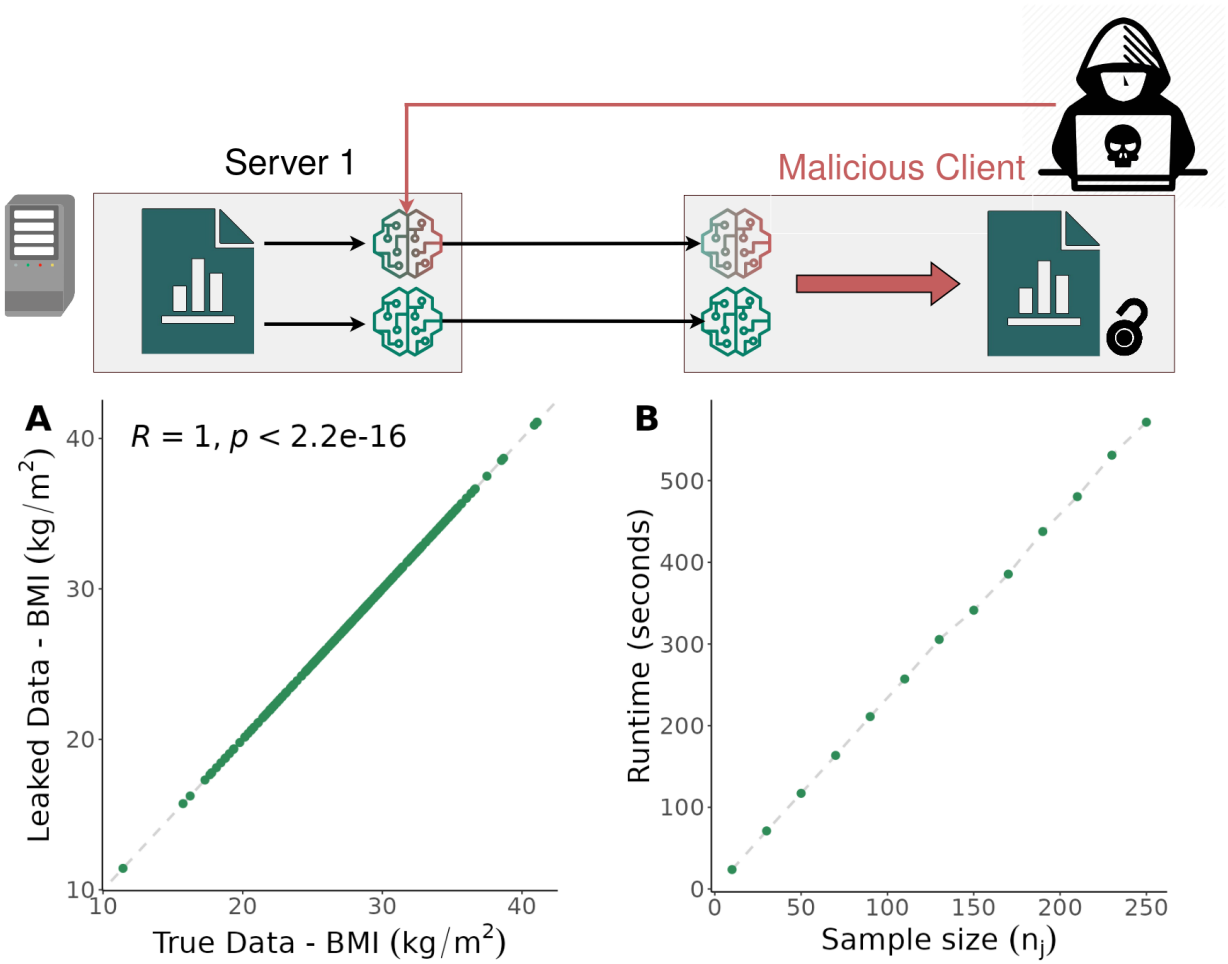

Figure 3: **Leakage results and computation times** for R DataSHIELD. **A** True data values from the first server of the CNSIM data set vs. the corresponding retrieved data provided by the Covariance-Based Attack Algorithm. **B** Computation time of the algorithm for different sample size. This figure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

data set from the R DataSHIELD tutorial [25] 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 the individual Body Mass Index (BMI) measurements from the first server using a complete case analysis with a sample size of $n_j = 250$.

R 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 has $n_j = 250$ 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 which allows for sending 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 the client side and computing the relevant means and covariances, the data can be obtained as described in (1). Our evaluation of the example above shows that the true data can be reconstructed almost perfectly (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 set-ups.

TensorFlow Federated provides functionality for the Algorithm

To assess whether other tools allow for the implementation of similar attack strategies, we considered TensorFlow Federated (version 0.36.0)[20]. This is an open-source framework for computations on decentralized data. In contrast to R DataSHIELD, the website mentions explicitly that this tool 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 the possibility of implementing the Covariance-Based Attack Algorithm using a set of basic functions.[1]

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

Our assessment revealed that TensorFlow Federated meets the tool requirements (T1)—(T3) and allows for the implementation of a Covariance-Based Attack Algorithm. Functions can 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 R 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 afore-mentioned CNSIM data set. We found that this allows for a reconstruction of the data up to numerical accuracy (Supplementary Figure 5). 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.

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. Firstly, it determines the size of the system of equations (1). This size is identical to n_j, meaning that n_j requests need to be sent to the j-th server. The communication overhead for a request is constant, but the computation time will in general grow linearly with n_j $[\mathcal{O}(n_j)]$, as the dimensionality of the scalar product grows. 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.

In order to evaluate the scaling behaviour 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 considered data set, 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

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, 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 means and covariances, a malicious client is able to retrieve the data.

Discussion

Federated Learning is a powerful tool and has been proven to be essential in a large number of fields. During the SARS-CoV-2 pandemic, a large number of consortia heavily relied on Federated Learning and outlined its potential [1, 26, 27]. Yet, it must be ensured that the data of the participating servers remain private. To achieve this, attack strategies need to be studied in detail. Here, we have proposed the novel Covariance-Based Attack Algorithm to which established Federated Learning systems are vulnerable.

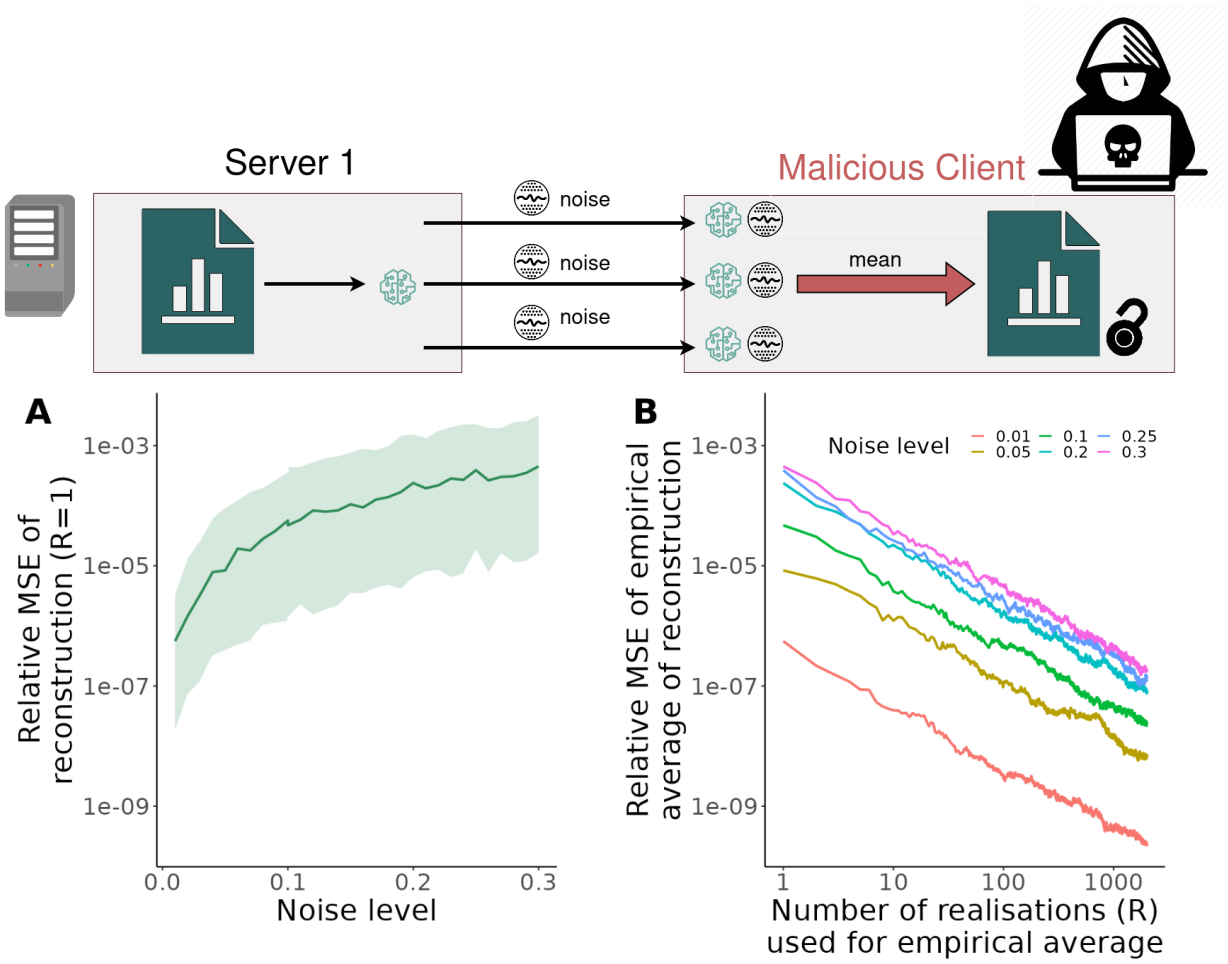

Figure 4: **Robustness of Covariance-Based Attack Algorithm** to normally distributed noise on means and covariances. **A** Relative mean squared error (MSE) of the reconstructed data values for different noise level if only a single realisation is available ($R = 1$). The median (line) and the 5th to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

We have shown that a malicious client could use the Covariance-Based Attack Algorithm to leak data from a Federated Learning system. Our approach is conceptually different from previously published studies, which focused on information leakage through gradients obtained from deep neural models. It relies on building linearly independent vectors on the server side and sample means as well as sample covariance functions that can be accessed by the client. This attack approach provides fast data leakage and superior scaling. It is easily

implemented and not thwarted by noise perturbations. This is demonstrated by applying our algorithm on R DataSHIELD (version 6.2.0.), and TensorFlow Federated (version 0.36.0) for which we were able to reconstruct the data. We have provided the respective developers with due notice.

The proposed Covariance-Based Attack Algorithm provides a prototype for a class of strategies. Improvements may, for instance, simultaneously evaluate multiple vectors y_i, enforce block structures on Y or create it with a sample mean of zero. This can reduce the time spent in communication as well as the computation time required to solve the linear system. Furthermore, additional functions on the server and client side might be (mis-)used.

Our findings suggest that existing functionalities of Federated Learning frameworks need to be reviewed with respect to data leakage threats. We propose to tackle (T3). It is necessary to send ordered vectors carrying aggregated information, e.g. parameters in optimisation, from the client to the server site. Immediately processing these vectors within a function, instead of creating a vector on the server side, is a possible defence strategy.

We note that in order to apply the proposed strategy, the attacker must have access to the client. In most cases, this is not straightforward and requires login credentials. However, if the security of the data only depended on the trustworthiness of the client, who could potentially retrieve the data with the Covariance-Based Attack Algorithm, Federated Learning were redundant as it could be replaced by Central Learning. Furthermore, it raises the question of GDPR conformity. Finally, this study raises the question of responsibility and liability in the case of unknown attack strategies.

With this work, we aim to support studies around Federated Algorithms and to raise awareness about potential security. Hence, we contribute to the emerging literature on data leakage problems in Federated Learning systems. We did no study other distributed frameworks, like swarm learning, but encourage a careful review. While security levels appear higher as the aggregation of information is shared, an attack might still be possible if requirements (T1)–(T3) are met for the data providers. If this is not the case, swarm learning is more likely to represent a preferable framework.

We expect that our results will contribute to establishment of design criteria for the structure of Federated Learning platforms. We have demonstrated that the available systems need to be improved to reduce the risk of data leaks.

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

References

1. Dayan, I. *et al.* 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. *et al.* 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. *et al.* 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. *Preprint at https://arxiv.org/abs/1903.10635* (2019).
8. Yang, T. *et al.* Applied federated learning: Improving google keyboard query suggestions. *Preprint at https://arxiv.org/abs/1812.02903* (2018).
9. Hansen, J. *et al.* Assessment of the EU Member States' rules on health data in the light of GDPR (2021).
10. Bentzen, H. B. *et al.* 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. *et al. 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. *Preprint at https://arxiv.org/abs/2001.02610* (2020).

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. *Preprint at https://arxiv.org/abs/2003.02133* (2020).

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

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

21. De Moira, A. P. *et al.* 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).

22. Pastorino, S. *et al.* 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).

23. Pearce, M. *et al.* 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).

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

25. Westerberg, A. & Wilson, R. *DataSHIELD training part 1: Introduction and logging in* `https://data2knowledge.atlassian.net/wiki/spaces/DSDEV/pages/1241120778/DataSHIELD+Training+Part+1+Introduction+and+logging+in`, Last accessed on 2021-15-09. (2022).

26. Tacconelli, E. *et al.* 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).

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

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

$$y_i^T x_{j,k} = (n_j - 1)\,\text{Cov}\left(x_{j,k}, y_i\right) + n_j\,\text{Mean}(x_{j,k})\,\text{Mean}(y_i). \tag{2}$$

We combine the equations for $i = 1, \ldots, n_j$ from (2) 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 (3) by the inverse of Y^T to obtain

$$x_{j,k} = (n_j-1)\left(Y^T\right)^{-1}\tilde{V} + n_j\left(Y^T\right)^{-1}\text{Mean}(x_{j,k})\,\tilde{m}, \tag{3}$$

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

As a defence strategy against malicious client, we consider the perturbation of mean and covariance with noise. 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 over 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 still 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 that for any $c > 0$

$$\lim_{R\to\infty} \mathbb{P}\left(\left\| \underbrace{\frac{1}{R}\sum_{r=1}^{R} x_{j,k,r}^{noisy}}_{=\text{ empirical mean}} - x_{j,k} \right\|_2 \geq c\right) \to 0, \tag{4}$$

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 $\mathbb{V}(\gamma_r) = \sigma_\gamma^2 < \infty$. Further, let γ_r and ε_r be uncorrelated so that $\mathbb{E}(\gamma_r \cdot \varepsilon_r) = 0$. The noisy version of equation (3) is given by

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

such that $x_{j,k,r}^{noisy}$ can be decomposed into the true $x_{j,k}$ and a noise term. Combining (4) and (5) shows that (4) is proven if the mean of the noise term converges in probability to zero, such that it sufficient to show that

$$\lim_{R\to\infty} \mathbb{P}\left(\left\| \frac{1}{R}\sum_{r=1}^{R}(A\varepsilon_r + B\gamma_r) \right\|_2 \geq c\right) \to 0. \tag{5}$$

This can be shown by applying Markov's Inequality

$$\mathbb{P}\left(\left|\left|\frac{1}{R}\sum_{r=1}^{R}(A\varepsilon_r + B\gamma_r)\right|\right|_2 \geq c\right) < \frac{\mathbb{E}\left(\left|\left|\frac{1}{R}\sum_{r=1}^{R}(A\varepsilon_r + B\gamma_r)\right|\right|_2^2\right)}{c^2}. \tag{6}$$

Since (6) 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 (5). 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 (6) 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, (5) holds and therefore (4) is proven.

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

$$\text{RMSE} = \frac{\left|\left|\frac{1}{R}\sum_{r=1}^{R} x_{j,k,r}^{noisy} - x_{j,k}\right|\right|_2^2}{||x_{j,k}||_2^2}. \tag{7}$$

Implementation and availability

A code example of our attack algorithm using the open source frameworks R DataSHIELD (version 6.2.0) and TensorFlow Federated (version 0.36.0) with their tutorial's test data set CNSIM is provided at GitHub at

`https://github.com/manuhuth/Data-Leakage-From-Covariances.git`.

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 Bundesministerium 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 information

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 R DataSHIELD. R.G. implemented the algorithm in TensorFlow Federated. J.H and E.T. conceptualised the study. M.H. 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, 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.

Supplementary Figures

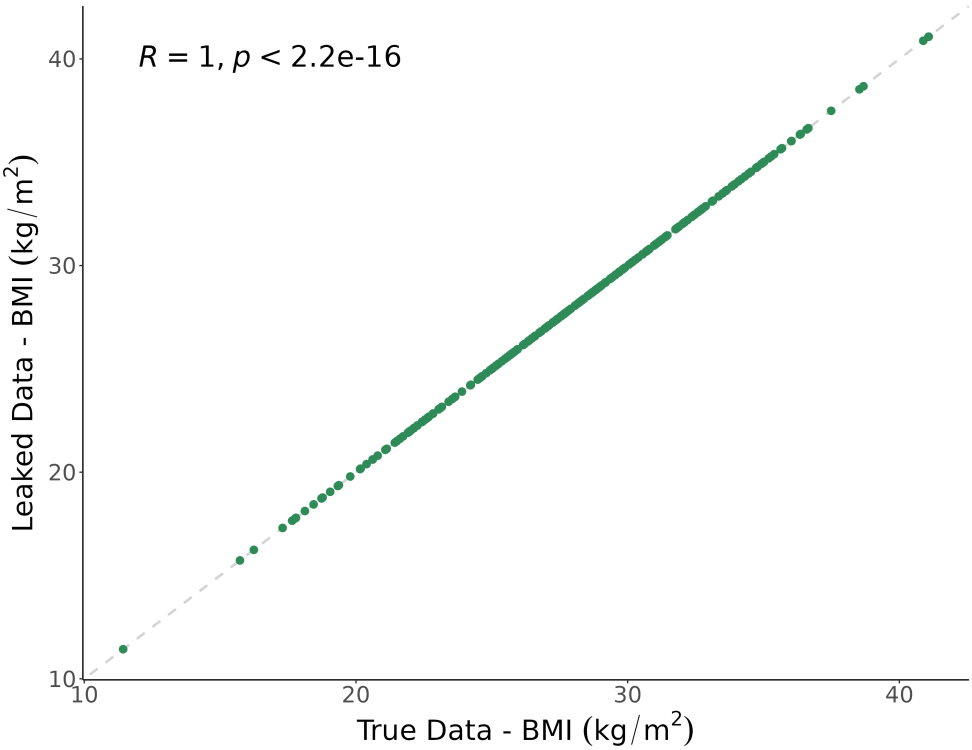

Figure 5: **Leakage results for TensorFlow Federated** are shown. The true data values from the first server of the CNSIM data set are plotted against the corresponding leaked data provided by the Covariance-Based Attack Algorithm.