Privacy of Federated QR Decomposition Using Additive Secure Multiparty Computation

Anne Hartebrodt, *Member, IEEE*, and Richard Röttger

***Abstract*— Federated learning (FL) is a privacy-aware data mining strategy keeping the private data on the owners' machine and thereby confidential. The clients compute local models and send them to an aggregator which computes a global model. In hybrid FL, the local parameters are additionally masked using secure aggregation, such that only the global aggregated statistics become available in clear text, not the client specific updates. In this context, we investigate the data leakage of three popular algorithms for QR decomposition, Gram-Schmidt orthonormalization, the Householder algorithm and Givens rotation. We show that, even when using additive SMPC, Givens rotation and the Householder matrix leak raw data and are therefore not suited for this computation paradigm. Gram-Schmidt orthonormalization relies on inner vector products and does not leak raw data points.**

***Index Terms*— Federated learning, federated matrix orthonormalization, privacy analysis, linear regression, QR factorization.**

I. Introduction

FEDERATED learning has risen in popularity following the seminal article by McMahan et al [1], and possibly accelerated by a search for new privacy preserving data analysis techniques following the introduction of the GDPR in Europe. Federated learning is a data analysis paradigm, where the data stays on the data owners' machine and only aggregated parameters are exchanged with the other participants or a central aggregator. There are two main versions of federated learning, cross-silo federated learning and cross-device federated learning. Cross-device FL connects many devices with relatively low computational power, such as mobile phones or sensors in a learning process. The devices have access to limited data, for example for one user. Cross-silo federated learning, the learning paradigm adopted in this article, joins multiple data silos containing records for a larger group of participants together [2]. The federated setting adopted in this article is a type of hybrid federated learning which relies on secure parameter aggregation (SMPC). This means the computations at the client sides are done on clear text, but the aggregation is performed using secure multiparty computation. Therefore, only the aggregated parameters become known to the participants, not the individual clients' updates. The participants are honest-but-curious, following the protocol, but trying to infer as much information as possible from the updates they receive [3]. Since we "only" use secure aggregation and allow the disclosure of intermediate and final results, the advantage is, that we can directly chain together different algorithms into pipelines. Modern data analysis workflows rarely only use a single tool, therefore the use of secure aggregation allows reasonable privacy guarantees, without the need to develop new protocols for every workflow.

We recently identified federated QR orthonormalization as a contributor to a more privacy preserving principal component analysis (PCA) algorithm [4]. Using federated QR orthonormalization for singular value decomposition allows the right, patient-associated singular vectors to remain private when using federated power iteration. QR decomposition is a versatile tool used for many more applications in linear algebra, including the solution of systems of linear equations [5].

In centralized learning, the traditional machine learning setup, where all data is on a global server, three algorithms for QR factorization are available. They are based on Householder reflection, Givens rotation and the Gram-Schmidt procedure. The Householder algorithm is the most efficient for general applications, while Givens rotation is advantageous for sparse matrices and parallel computing architectures [6]. Gram-Schmidt orthonormalization is not used as much in practice due to numerical instabilities on special matrices [5]. However, a stabilized version of the algorithm exists and privacy considerations may take precedence over numerical issues. Consequently, it is interesting to evaluate the algorithms with regards to their suitability for federated learning. The primary goal of this article is to evaluate the data disclosure of the three algorithms when deployed in a federated setting. In an earlier article, we introduced federated Gram-Schmidt orthonormalization [4], but it does not return the full decomposition. Therefore, it needs to be extended to return a full QR factorization. The other algorithms have not been explicitly introduced for cross-silo FL, therefore in this article we develop prototypes of their federated versions. The main objective of this article is to show that Householder reflection and Givens rotation have properties that render them unsuitable for federated computation, when secure additive aggregation is used.

There are many federated algorithms that can be used for PCA, including a QR based scheme which has been introduced

Manuscript received 12 October 2022; revised 2 May 2023; accepted 16 June 2023. Date of publication 3 August 2023; date of current version 23 August 2023. This work was supported by the European Union's Horizon 2020 Research and Innovation Program funded by the Feature Cloud Project under Grant 826078. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. George Theodorakopoulos. *(Corresponding author: Anne Hartebrodt.)*

Anne Hartebrodt was with the Department of Mathematics and Computer Science, University of Southern Denmark, 5230 Odense, Denmark. She is now with the Department Artificial Intelligence in Biomedical Engineering (AIBE), Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU), 91052 Erlangen, Germany (e-mail: anne.hartebrodt@fau.de).

Richard Rottger is with the Department of Mathematics and Computer Science, University of Southern Denmark, 5230 Odense, Denmark.

Digital Object Identifier 10.1109/TIFS.2023.3301710

by [7] and which we extend using the orthonormalization scheme developed in this article. We consider the privacy of this scheme, when using federated QR orthonormalization. Notably, the question we want to answer is whether the introduction of the federated QR scheme increases the privacy of the algorithm. We find, that upper triangular matrices are vulnerable to data leakage even when applying Gram-Schmidt orthogonalization.

Furthermore, to highlight the use for the federated QR procedure in other applications, we apply federated QR decomposition to compute linear regression. This could be used as an alternative solver for the federated linear regression computation suggested for instance in [8]. Our experiments demonstrate the same accuracy of the federated linear regression as standard standalone tools. Additionally, we provide a federated implementation of the Gram-Schmidt algorithm using secure multiparty computation with competitive runtime.

To summarize, our contributions are the following:

- We analyze the Householder, Givens and Gram-Schmidt algorithms for QR decomposition with respect to their privacy when using hybrid federated learning with secure additive aggregation.
- In order to do so, we develop prototypes of federated Householder reflection and Givens rotation and extend a previous algorithm to return the full QR decomposition.
- We conclude that both federated Householder reflection and Givens rotation introduce critical data leaks even when using secure additive aggregation.
- We investigate a special application case of federated Gram-Schmidt orthogonalization on upper triangular matrices which may expose the input data.
- We provide a realistic use case of federated QR decomposition, linear regression.

The remainder of this manuscript is organized as follows: in Section II, the preliminaries, including the three centralized QR algorithms are introduced. Section III introduces related work. Based on the centralised descriptions, in Section IV, we develop federated QR schemes for all algorithms, and a more detailed description for the most suitable QR algorithm, the federated Gram-Schmidt procedure (Section IV-C). In Section V, we evaluate the privacy of a QR based PCA scheme. Section VI describes how to compute federated linear regression based on QR decomposition. We provide empirical results for our analyses in Section VII. Lastly, the results are briefly discussed in Section VIII. Section IX concludes the work.

II. Preliminaries

A. Data Model and Architecture

In this manuscript we assume matrix $\mathbf{A} \in \mathbb{R}^{n \times m}$ to be partitioned into a set of $s \in [S]$ partial data sets such that $\mathbf{A}^s \in \mathbb{R}^{n^s \times m}$. $[S]$ denotes the set of clients joining the learning system. This partitioning is referred to as horizontal. We assume all participants have a share of the data, and the ordering of the rows is known and fixed. We describe our algorithm using a star-like architecture. We expect the parameters to be masked using additive secure aggregation (cf. Section II-C), therefore we assume that peer-to-peer communication is possible via secure channels, regardless of the underlying architecture. This implies that our algorithms could be run on a fully decentralized architecture. The main reason for the choice of an aggregator-based architecture is the reduction in overall communication, because without SMPC the clients do not have to transmit the intermediate parameters to all their peers, only to the aggregator.

TABLE I
Notation Table

Syntax	Semantics
$[N] \subset \mathbb{N}$	index set $[N] = \{i \in \mathbb{N} \mid 1 \leq i \leq N\}$
$S \in \mathbb{N}$	number of sites
$m \in \mathbb{N}$	number of features
$n \in \mathbb{N}$	total number of samples
$n^s \in \mathbb{N}$	number of samples at site $s \in [S]$
$\mathbf{A} \in \mathbb{R}^{n \times m}$	complete data matrix
$\mathbf{A}^s \in \mathbb{R}^{n^s \times m}$	subset of data available at site $s \in [S]$
$\mathbf{U} \in \mathbb{R}^{n \times k}$	orthogonal matrix with $\text{span}(\mathbf{U}) = \text{span}(\mathbf{A})$
$\mathbf{U}^s \in \mathbb{R}^{n^s \times k}$	sub matrix of $\mathbf{U}$ available at sites $s \in [s]$
$\mathbf{Q} \in \mathbb{R}^{n \times k}$	orthonormal matrix with $\text{span}(\mathbf{Q}) = \text{span}(\mathbf{A})$
$\mathbf{Q}^s \in \mathbb{R}^{n^s \times k}$	sub matrix of $\mathbf{Q}$ available at sites $s \in [s]$
$\mathbf{R} \in \mathbb{R}^{k \times k}$	upper triangular matrix

B. Notation

Vectors and matrices are denoted in boldface, scalars in normal font. Matrices are noted in upper case letters and consist of column vectors which are noted in lower case letters. For instance, the matrix $\mathbf{A}^{n \times m}$ consists of m column vectors $\mathbf{a}_i$ where i is the index of the column. Sometimes we refer to columns and rows of a matrix as $\mathbf{A}_{\bullet,i}$ and $\mathbf{A}_{i,\bullet}$ respectively. Table I contains an overview over the most frequently used variables in this work.

C. Secure Aggregation

The secure aggregation scheme used in this work relies on the additive aggregation protocol used by [3]. It assumes honest-but-curious participants, i. e. all clients perform the computations following the protocol but try to infer as much information as possible from the exchanged parameters [9]. All s clients create $i = S$ random shares $x_{s,i}$ of their secret value x_s such that $\sum_i^S x_{s,i} \mod p = x_s$, where p is a large prime known to all participants. One can think of the "s" in $x_{s,i}$ as the source of the share and "i" the destination. All clients send the respective shares $x_{s,i}$ to the respective recipients i where the sum of the shares is computed as $\sum_s^S x_{s,i} = x_i$. None of the shares disclose any information on the original values. Lastly, the clients announce their aggregated secret share x_i, such that the global sum $x = \sum_i^S x_i \mod p$ of all private shares can be formed. This scheme is suitable for a cross-silo federated learning system with reliable clients (i. e. they do not randomly drop out) and relatively few participants. Other secure aggregation schemes, such as Shamir's protocol, which are more fault tolerant to client dropout [3] could be used instead without conceptual change of the algorithm.

D. Centralized QR Decomposition

The QR decomposition is the factorization of a square matrix into a square orthonormal matrix $\mathbf{Q}$ and an upper triangular matrix $\mathbf{R}$.

$$\mathbf{A} = \mathbf{QR} \tag{1}$$

It exists also for non-square matrices (reduced QR decomposition) which is significantly more memory efficient if $n > m$. Three popular schemes exist for the computation of the decomposition, the Householder, Givens and Gram-Schmidt algorithms. In centralized systems, the Householder algorithm and Givens rotation are more popular, because they do not suffer numerical instability as the canonical version of Gram-Schmidt orthonormalization. Generally, Householder reflection is more efficient, and preferred unless the matrices are sparse or parallel compute architecture can be used [6]. See [5] for more details on the algorithms.

1) Householder Transformation: The Householder reflection proceeds column-wise, setting all the elements below the diagonal to 0 using a Householder reflector. Therefore, it requires $m-1$ Householder reflections to form an upper triangular matrix $\mathbf{R}$ starting from a matrix $\mathbf{A} \in \mathbb{R}^{n\times m}$. A Householder reflector is defined as

$$\mathbf{Q}^u = I - \frac{2\mathbf{u}\mathbf{u}^\top}{\mathbf{u}^\top\mathbf{u}}, \quad \mathbf{u} \neq 0 \tag{2}$$

For each column vector $\mathbf{a}_i$ in matrix $\mathbf{A}$ the Householder reflection is computed using the following steps. First, $\mathbf{a}$ is normalized as $\mathbf{a}_i = \frac{\mathbf{a}_i}{||\mathbf{a}_i||_\infty}$ to avoid numerical overflow. Then the vector $\mathbf{u}$, i. e. the vector required for the construction of the Householder reflector, is computed as $\mathbf{u}_i = \mathbf{a}_i \pm ||\mathbf{a}_i||_2 \cdot \mathbf{e}$, where $\mathbf{e} = [1\ 0\ \cdots\ 0]^\top \in \mathbb{R}^{m\times 1}$ denotes a vector of length m containing a 1 in the first position and 0 otherwise. For ease of notation, the scaling factor $\frac{2}{\mathbf{u}^\top\mathbf{u}}$ is denoted β. The Householder reflection is computed implicitly to increase the computational performance. The resulting matrix $\mathbf{Q}^u\mathbf{A} = \mathbf{A} - \beta\mathbf{u}\mathbf{u}^T\mathbf{A}$ contains 0 in the column corresponding to vector $\mathbf{a}_i$. Algorithm 1 summarizes a single Householder reflection. In the Householder QR algorithm this operation is performed for all vectors $\mathbf{a}_i$ of $\mathbf{A}$, transforming in each step only the sub matrix, which is not yet upper triangular by choosing the remaining reflection matrix as the identity matrix I. The full description of the Householder algorithm is shown in Algorithm 2.

Algorithm 1 Householder Reflection

Input: Data matrix $\mathbf{A}^s \in \mathbb{R}^{m\times m}$
Output: Matrix $\mathbf{A}$ upper triangular up to $\mathbf{a}_i$, reflection vector $\mathbf{u}_i$

$\bar{\mathbf{a}} = \frac{\mathbf{A}_{\bullet,i}}{||\mathbf{A}_{\bullet,i}||_\infty}$;
$\mathbf{u} = \bar{\mathbf{a}} + sgn(\mathbf{A}_{1,i}) \cdot ||\bar{\mathbf{a}}||_2 \cdot \mathbf{e}$;
$\mathbf{Q}_i\mathbf{A} = \mathbf{A} - \beta\mathbf{u}\mathbf{u}^T\mathbf{A}$;

2) Givens Rotation: Givens QR algorithm sequentially sets subdiagonal elements of the matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$ to 0 by multiplying the matrix with the corresponding "Givens matrix" [5]. After $\frac{n\cdot(m-1)}{2}$ operations all elements below the diagonal are 0 resulting in an upper triangular matrix $\mathbf{R}$. Through careful choice of the parameters in the Givens matrices, their product results in an orthogonal matrix $\mathbf{Q}$ which is the desired result.

Algorithm 2 Householder Algorithm

Input: Data matrix $\mathbf{A}^s \in \mathbb{R}^{m\times m}$
Output: Orthonormal matrix $\mathbf{Q}$ and upper triangular matrix $\mathbf{R}$

$\mathbf{R} \leftarrow \mathbf{A}$ $\mathbf{Q} \leftarrow \mathbf{I}$;
for $i \in m-1$ **do**
  // Update relevant entries of $\mathbf{R}$
  $\mathbf{R}_{i:m,i:m}, \mathbf{u} \leftarrow$ `householder-reflection`$(\mathbf{R}_{i:m,i:m})$;
  // Implicit multiplication of Q
  $\mathbf{Q}_{1:m,i:m} \leftarrow \mathbf{Q}_{1:m,i:m} - (\frac{2}{||\mathbf{u}||_2^2})\mathbf{Q}_{1:m,i:m}(\mathbf{u}\mathbf{u}^\top)$;

A Givens matrix has the following form, where i and j are the indices of c and s.

$$J(i,j,c,s) = \begin{pmatrix} 1 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ 0 & \ddots & & & & & \vdots \\ 0 & & c & \cdots & s & & 0 \\ \vdots & & \vdots & \ddots & \vdots & & \vdots \\ 0 & & -s & \cdots & c & & 0 \\ \vdots & & & & & \ddots & 0 \\ 0 & \cdots & 0 & \cdots & 0 & 0 & 1 \end{pmatrix} \tag{3}$$

$J(i,j,c,s)$ is orthogonal if $c^2 + s^2 = 1$.

Let $\mathbf{A}$ be the matrix of interest and i and j with $i < j$ indices of the element to be set to 0. Then one can set

$$c = \frac{x_{i,i}}{\sqrt{x_{i,i}^2 + x_{j,i}^2}} \tag{4}$$

and

$$s = \frac{x_{i,j}}{\sqrt{x_{i,i}^2 + x_{i,j}^2}} \tag{5}$$

and compute the respective Givens matrix according to Equation (3).

Then $\mathbf{A}' = J(i,j,c,s)\mathbf{A}$ contains a 0 at position (i,j). The product of a Givens matrix with a general matrix can be computed efficiently, by updating only rows i and j of the matrix as

$$\mathbf{A}_{i,\bullet} = [ca_{i,1} + sa_{j,1}, ca_{i,2} + sa_{j,2}, \cdots, ca_{i,m} + sa_{j,m}] \tag{6}$$

and

$$\mathbf{A}_{j,\bullet} = [ca_{j,1} + sa_{i,1}, ca_{j,2} + sa_{i,2}, \cdots, ca_{j,m} + sa_{i,m}] \tag{7}$$

The full QR decomposition in a centralized setting is summarized in Algorithm 3.

3) Gram-Schmidt Orthonormalization: The Gram-Schmidt algorithm produces an orthonormal matrix $\mathbf{Q} = [\mathbf{q}_1 \dots \mathbf{q}_k]$ and an upper triangular matrix $\mathbf{R} = [\mathbf{r}_1 \dots \mathbf{r}_k]$ [10]. With a matrix $\mathbf{A} = [\mathbf{a}_1 \dots \mathbf{a}_k] \in \mathbb{R}^{n\times m}$ of m linearly independent column vectors, the matrix $\mathbf{U} = [\mathbf{u}_1 \dots \mathbf{u}_k] \in \mathbb{R}^{n\times m}$ of orthogonal column vectors is computed, such that it has the same span as

Algorithm 3 QR Factorization Using Givens Rotation

Input: Data matrix $\mathbf{A}^s \in \mathbb{R}^{n \times m}$
Output: Orthonormal matrix $\mathbf{Q}$ and upper triangular matrix $\mathbf{R}$

```
foreach i ∈ [1, ..., m − 1] do
    foreach j ∈ [i + 1, ..., m] do
        [s, c] ← compute-givens-parameter();
        A = J(i, j, c, s)A;
        Q = J(i, j, c, s)Q;
R = A;
Q = Q^⊤;
```

A. Let $r_{i,j} = \mathbf{u}_j^\top \mathbf{a}_i / n_j$ and $n_j = \mathbf{u}_j^\top \mathbf{u}_j$ then

$$\mathbf{u}_i = \begin{cases} \mathbf{a}_i & \text{if } i = 1 \\ \mathbf{a}_i - \sum_{j=1}^{i-1} r_{i,j} \cdot \mathbf{u}_j & \text{if } i \in [k] \setminus \{1\}, \end{cases} \tag{8}$$

$$\mathbf{q}_i = \frac{\mathbf{u}_j}{||\mathbf{u}_j||} \tag{9}$$

$$\mathbf{r}_{j,i} = \begin{cases} \mathbf{q}_j \cdot \mathbf{a}_i & \text{if } j \leq i \\ 0 & \text{if } j > i \end{cases} \tag{10}$$

E. Centralized Singular Value Decomposition

Singular value decomposition (SVD) is a matrix decomposition frequently used in data mining applications. A matrix $\mathbf{A} \in \mathbb{R}^{n \times m}$ is decomposed into two orthonormal matrices of singular vectors $\mathbf{U} \in \mathbb{R}^{n \times k}$and $\mathbf{V} \in \mathbb{R}^{k \times m}$ and a diagonal matrix $\Sigma \in \mathbb{R}^{k \times k}$ containing the singular values in non-increasing order $\mathbf{A} \in= \mathbf{U}\Sigma\mathbf{V}^\top$ [5]. In the federated domain, SVD has been studied extensively, and multiple algorithms exist (e. g. [4], [7], [11]). Given the vertically distributed matrix $\mathbf{A}^s \in \mathbb{R}^{m \times n^s}$ with dimension $m \times n^s$ at sites s the federated singular value decomposition is defined as

$$\mathbf{A}^s = \mathbf{U}\Sigma\mathbf{V}^{s\top} \tag{11}$$

where $\mathbf{U}$ is the full left singular vector and $\mathbf{V}^s$ are the partial right singular vectors. The right singular vectors should not be shared due to potential privacy breaches [4].

F. Solution of Systems of Linear Equations

In centralized computation, QR factorization can be used to compute the solution of systems of linear equations. Given a system $\mathbf{Ax} = \mathbf{b}$, one can compute $\mathbf{A} = \mathbf{QR}$. By setting $\mathbf{QRx} = \mathbf{b} \Leftrightarrow \mathbf{Rx} = \mathbf{Q}^{-1}\mathbf{b}$ the system can be solved efficiently because due to the orthonormality of $\mathbf{Q}$, $\mathbf{Q}^{-1} = \mathbf{Q}^\top$ and $\mathbf{y} = \mathbf{Q}^{-1}\mathbf{b}$ can be computed. This leaves to solve a system of the form $\mathbf{Rx} = \mathbf{y}$, which can be solved efficiently as $\mathbf{R}$ is an upper triangular matrix [10]. This can be used for instance for linear regression [8].

III. Related Work

Recently, the authors of [12] have developed a federated PCA algorithm which includes a QR subroutine. The whole computation is performed under encryption, including the intermediate parameters, and the results can be obtained in a realistic run time with high accuracy. Previously, federated QR algorithms have been suggested mainly in the field of peer-to-peer networks relying on the PushSum algorithm and gossiping [13], [14], [15]. While these schemes can be implemented in a modern federated learning system, the assumptions governing FL make these algorithms unsuited. Notably, in cross-device FL, the client-to-client communication is assumed to be a bottleneck [2] and client-aggregator communication is preferred. Secondly, cross-silo FL assumes more data and higher compute power at the nodes, so local computational constraints do not impact the computations as severely. In medical systems, practitioners might want to avoid approximation errors at the cost of higher compute time [16]. In distributed memory contexts, diverse schemes have been proposed to efficiently and quickly compute the QR decomposition (e.g. [17]). In these systems, there is usually one owner of the data, so we are unaware of privacy analyses in this context. Outsourcing the data is a fundamentally different compute set up, it is mentioned here for completeness. In the outsourced, encrypted domain, the work of [18] and [19] still suggest very high expected execution times due to the encryption or masking overhead.

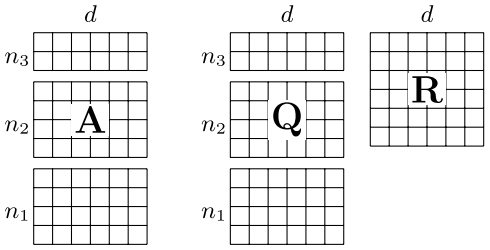

Fig. 1. Schematic QR decomposition with 3 participants. **A** and **Q** remain private. **R** is known to all participants.

IV. Federated QR Decomposition

In this section, we describe and analyse approaches to federate QR factorization. To our knowledge, there exist no descriptions of federated versions of the Householder reflection or Givens rotation-based algorithms. Therefore, we first provide descriptions of the federated algorithms and demonstrate that they are not suitable for the chosen federated setting. Lastly, we describe the extended Gram-Schmidt algorithm which also returns the upper triangular matrix $\mathbf{R}$. Recall that we assume the data $\mathbf{A} \in \mathbf{R}^{n \times m}$ to be partitioned row-wise into chunks $\mathbf{A}^s \in \mathbf{R}^{n^s \times m}$. The goal of federated QR decomposition is to compute $\mathbf{Q}^s$ and $\mathbf{R}$ such that $\mathbf{A}^s$ and $\mathbf{Q}^s$ stay confidential, meaning the raw data does not leave site s and $\mathbf{Q}$ can only be computed at s. $\mathbf{R}$ is common to all sites.

A. Federated Householder Algorithm

We describe a straightforward algorithm for a federated Householder reflector. This subroutine could be used to compute the full QR decomposition in a federated manner. Let t_s be the row index set of $\mathbf{A}^s$ at site s.

In Algorithm 4 we describe a federated Householder reflector. The algorithm proceeds column wise. Initially, the global

Algorithm 4 Federated Householder Reflection
Client-Side Computations Are Marked in Blue

Input: Data matrices $\mathbf{A}^s \in \mathbb{R}^{n^s \times m}$ at sites $s \in [S]$.
Output: Partial matrices $\mathbf{A}^s$ upper triangular up to $\mathbf{a}_i$, 'oracle' reflection vector $\mathbf{u}_i$

// t_s is the index set for the rows of $\mathbf{A}^s$
// Compute the global max element $||\mathbf{A}_{\bullet,i}||_\infty$
1 $m^s \leftarrow$ `send-to-aggregator`$(||\mathbf{A}_{t_s,i}||_\infty)$;
2 $||\mathbf{A}_{\bullet,i}||_\infty = \max_{s\in[S]} m^s$;
// Run at all clients
3 **for** $s \in [S]$ **do**
  // Compute the local portion of $\mathbf{u}$
4 $\bar{\mathbf{a}}^s \leftarrow \frac{\mathbf{A}_{t_s,i}}{||\mathbf{A}_{\bullet,i}||_\infty}$;
  // Compute the norm of $\bar{\mathbf{a}}$
5 $n^s \leftarrow \bar{\mathbf{a}}^s \top \bar{\mathbf{a}}^s$;
6 $||\bar{\mathbf{a}}|| \leftarrow \sqrt{\sum_s n^s}$;
7 $\mathbf{u}^s \leftarrow \bar{\mathbf{a}}^s \pm ||\bar{\mathbf{a}}||_2 \cdot \mathbf{e}$;
// Oracle step: stack $\mathbf{u}^s$ to form $\mathbf{u}\mathbf{u}^\top$
8 $\mathbf{u} \leftarrow$ `stack-vertically`$([\mathbf{u}^1, \cdots \mathbf{u}^s])$;
// Update A
9 **for** $s \in [S]$ **do**
10 $\mathbf{H}_A^s = \beta \mathbf{u}\mathbf{u}^T \mathbf{A}^s$;
11 $\mathbf{H}_A = \sum_S \mathbf{H}_A^s$;
12 **for** $s \in [S]$ **do**
13 $\mathbf{A}^s = \mathbf{Q}_i^s \mathbf{A}^s = \mathbf{A}^s - \mathbf{H}_A$;

infinity norm $||\mathbf{A}_{\bullet,i}||_\infty$ is computed as the max over all local infinity norms (lines 1 to 2)) and the norm of the scaled vector is computed (lines 3 to 6). Then, the clients locally compute $\mathbf{u}$ (Line 7). In order to compute the Householder reflector, the clients send their partial vectors $\mathbf{u}^s$ to the aggregator (Line 8). We call this step an "oracle step" to indicate that under the chosen secure computation paradigm, the aggregation itself cannot be performed privately. The reflection matrix $\mathbf{H}_A$ needs to be computed collaboratively by adding up the local shares (lines 9 to 11). Finally, at the clients, the reflection is performed (lines 13).

Ad-hoc, this naive federated implementation of the procedure would take four communication rounds per column vector, one for the computation of the maximal element, one for the computation of the norm, and two for the computation of the reflector $\mathbf{H}_A$ and the reflection.

In the federated setting, the computation of the Householder reflector itself is immediately problematic regarding the confidentiality of the data. Recall that algorithm relies on the computation of the outer product of $\mathbf{u}$ which is a direct transformation of the original column vectors of $\mathbf{A}$. In step 8, we call this operation an oracle step because it cannot be performed using the SMPC scheme we choose. Furthermore, even if secure multiplication is used, this "summary statistics" constitutes a privacy breach because the diagonal of $\mathbf{u}\mathbf{u}^\top$ contains the squared entries of $\mathbf{u}$. If $\mathbf{u}$, and $||\mathbf{a}||_\infty$ are known, then the original vector $\mathbf{A}_{\bullet,i}$ can be reconstructed. (We still assume, that all aggregate statistics are known, but we assume that the 'oracle step' can be computed using secure multiplication.)

Proposition 1: Assuming that only aggregate statistics, excluding $\mathbf{u}$*, but including the Householder projector* $\mathbf{H}_A = \beta \mathbf{u}\mathbf{u}^\top$ *become known in clear text to any of the participants, they can reconstruct the entire input data based on the summary statistics they know.*

Proof: Recall that during the computation of the reflector, m and n the maximal element and norm of $\mathbf{A}_{\bullet,i}$; β, the scaling factor and $\mathbf{u}\mathbf{u}^\top$ become known to the participants. The diagonal of $\mathbf{u}\mathbf{u}^\top$ contains the squared elements of $\mathbf{u}$, which can hence be computed up to the sign. Using the fact, that every participant can compute their share $\mathbf{u}^s$ of $\mathbf{u}$, it is possible to infer the sign of $\mathbf{u}$ for all participants using the off diagonal entries of $\mathbf{u}\mathbf{u}^\top$. For two sites s, and s', with index set $t_{s'}$ denoting the entries belonging to s' in $\mathbf{u}\mathbf{u}^\top$, we can compute the sign of $\mathbf{u}^{s'}$ at site s: $sgn(\mathbf{u}^{s'}) = sgn(\mathbf{u}_1^s) \cdot sgn(\mathbf{u}\mathbf{u}^\top_{t_{s'},1})$. Once the sign of $\mathbf{u}$ is known, the linear transformations can be reversed and $\mathbf{A}_{\bullet,i}$ becomes known. □

Therefore, it is not straightforward to privately compute the Householder transform using hybrid federated learning with secure aggregation. Knowledge of the procedures allows the reverse engineering of the data. This can potentially be prevented by performing the entire computation under homomorphic encryption, or an SMPC scheme which allows the evaluation of arbitrarily complex circuits. When using SMPC, it would not be sufficient to compute the outer product securely, the intermediate parameters cannot become known to any of the computing parties. Based on the incompatibility of the Householder reflection with secure aggregation, we exclude federated Householder reflection from further considerations.

B. Federated Givens Rotation

In this section, we describe a direct translation of a Givens rotation to a federated setting. Again, we only describe the relevant subroutine which would allow the implementation of the complete QR decomposition, albeit inefficiently. Realistically, one would choose a parallelized version of the operator.

Algorithm 5 summarizes the federated procedure described in the following. As precomputations, the clients perform Givens rotations to set all elements to 0 which only depend on their data. Then, the clients communicate all remaining non-zero indices below the diagonal to the aggregator. Setting an element to 0 requires only two rows i, and j to be manipulated. The clients associated with these rows are called k_1 and k_2. In the main loop, the aggregator announces the current i and j to the current clients k_1 and k_2 (Line 3). Client k_1 and k_2 compute and announce the Givens parameters s and c in collaboration with the aggregator (Lines 6 to 7). This is an "oracle step", as this implies the communication of x_i and x_j, because the Givens parameters cannot trivially be to compute using secure addition. The aggregator announces c and s to k_1 and k_2 and the clients update $\mathbf{R}$ and $\mathbf{Q}$. The broadcast can be combined with the new index broadcast (Line 3) if

Algorithm 5 QR Factorization Using Givens Rotation
Client-Side Computations Are Marked in Blue

Input: Data matrices $\mathbf{A}^s \in \mathbb{R}^{n^s \times m}$ at sites $s \in [S]$.
Output: Partial matrices $\mathbf{Q}^s$ and full matrix $\mathbf{R}$ at sites $s \in [S]$

/* Perform local precomputations, setting all possible elements to 0, send all non-zero indices to the aggregator */
1 **foreach** $i \in [1, \ldots, m-1]$ **do**
2 **foreach** $j \in [i+1, \ldots, m]$ **do**
  // Compute c and s, using values from two clients $k_1, k_2 \in [S]$
3 send-to-client(i, j);
4 $x_{i,i} \leftarrow$ send-to-aggregator$(a^{k_1}_{\mathbf{a}_{i,i}})$;
5 $x_{j,i} \leftarrow$ send-to-aggregator$(a^{k_2}_{\mathbf{a}_{j,i}})$;
6 $c = \frac{x_{i,i}}{\sqrt{x^2_{i,i}+x^s_{j,i}}}$;
7 $s = \frac{x_{i,j}}{\sqrt{x^2_{i,i}+x^s_{i,j}}}$;
  // Exchange the relevant entries required for the rotation
8 send-to-aggregator$[sa_{j,1}, sa_{j,2}, \cdots, sa_{j,m}]$;
9 send-to-aggregator$([sa_{i,1}, sa_{i,2}, \cdots, sa_{i,m}])$;
  // Perform rotation following Equation (6) and (7)
10 $\mathbf{A}^s = J(i, j, c, s)\mathbf{A}^s$;
11 $\mathbf{Q}^s = J(i, j, c, s)\mathbf{Q}^s$;
12 $\mathbf{R} = \mathbf{A}$;
13 $\mathbf{Q}^s = \mathbf{Q}^{s\top}$;

applicable. Lines 3 to 11 are repeated until all elements below the diagonal are 0.

The naive implementation of this procedure would require in the order of $N = \mathcal{O}(\frac{2 \cdot n \cdot (m-1)}{2})$ transmission rounds. Each element, would required an index broadcast and a Givens parameter broadcast. The procedure can be parallelized to zero out $\frac{n}{2}$ elements per round [6], reducing the communication complexity to $\mathcal{O}(n)$. Local precomputation would decrease the effective number of transmission costs. Furthermore, the index broadcast can most likely be done with fewer communications rounds.

However, there is a critical privacy breach when using Givens rotations. Recall that we assume the data to partitioned into s partitions $A^s \in \mathbb{R}^{n^s \times m}$. Assuming rows i and j are located in silo S_1 and S_2 respectively, the aggregator can compute the values c and s using x_i and x_j (cf. Equation (6), Equation (7)). Even if c and s are computed using SMPC and P2P communication (so that the aggregator does not gain knowledge of the parameters), x_i and x_j can be reconstructed at the current clients k_1 and k_2.

Proposition 2: A client S can reconstruct an entire row of the sub matrix $\mathbf{A}^{S'}$ of another participant S', if the Givens parameters as well as $\mathbf{a}^0_i$ and $\mathbf{a}^1_i$, the rows of $\mathbf{A}^S$ before and after the update are available in clear text at client S.

Proof: Let a^0_i denote the ith row of $\mathbf{A}$ at client k before the rotation and a^1_i denote the ith row at client k after the update. Given the Givens parameters c and s, client k can compute the jth row at client k' as $a^0_j = [(a^1_{il} - c \cdot a^0_{il})/s]$ with l the column index. □

In order to prevent this breach, the whole algorithm would have to be performed under encryption, such that S_1 does not gain access to the intermediate matrices $A^{S'}$. These considerations render this algorithm unsuitable for hybrid federated learning with secure parameter aggregation. This leaves the Gram-Schmidt algorithm as the final possible algorithm.

C. Federated Gram-Schmidt Algorithm Including the Computation of R

Based on the algorithm described in [4] and [20], where we showed that the orthogonal matrix $\mathbf{Q}$ can be computed solely based on the exchange and aggregation of vector norms and co-norms, we extend the algorithm such that the $\mathbf{R}$ matrix can be computed simultaneously. This can be done without further communication steps in comparison to the previously presented method. The main modifications are that the orthonormal vectors in $\mathbf{Q}$ need to be computed right away in order to compute the inner product of $\mathbf{q}_l \mathbf{a}_{i-1}$ contained in the matrix $\mathbf{R}$ at position $l, i-1$. Note, that the procedure also requires the orthogonal vectors $\mathbf{u}_i$. Please see [20] for details on the theoretical and empirical accuracy of federated Gram-Schmidt orthonormalization.

Therefore, we develop a detailed description of a federated Gram-Schmidt orthonormalization procedure (see Algorithm 6). First, the global vector norm n_i of $\mathbf{u}_i$ is calculated by computing the local vector norms n^s_i at the clients and aggregating them at the central server (Lines 1 to 4). The main loop starts at index $i = 2$ and proceeds in 4 stages. Let $\mathbf{R}$ be the upper triangular matrix completed up to vector $i \in d$. First, $\mathbf{e}^s_{i-1}$, is computed by dividing $\mathbf{u}_{i-1}$ through the global norm (Lines 7). Then, the $i-1$st local column $\mathbf{r}^s_{l,i-1}$ of $\mathbf{R}$ is computed as the inner product of the partially normalized vector $\mathbf{q}^s_{i-1}$ and the partial data column $\mathbf{a}^s$ (Lines 1 to 8). Then the local residuals r^s_{ij} for vector i w. r. t. to the previous $i-1$ vectors are computed (Lines 9 to 10). In stage 2, the two parameters $\mathbf{r}^s_{l,i-1}$ and r^s_{ij} are sent to the central server and aggregated via element-wise addition (Lines 11 to 14) to form the global copy of $\mathbf{R}$ up until $i-1$. The global $\mathbf{r}_{l,i-1}$, and r_{ij} are returned to the clients, where the orthogonal vector $\mathbf{u}^s_i$ is computed (Lines 15 to 17). In the last stage, the norm of the current vector $\mathbf{u}_i$, n_i is computed by summing up the local norms of $\mathbf{u}^s_i$ (Lines 18). The procedure is repeated for all d vectors of $\mathbf{A}$. After exiting the main loop, the last column of $\mathbf{A}$ is computed, and the partial orthonormal matrices $\mathbf{Q}^s$ and $\mathbf{R}$ are returned (Lines 9 to 23). This procedure is equal to the centralized Gram-Schmidt algorithm because the vector inner products can be computed exactly in a federated fashion.

D. Privacy Considerations

Recall that according to our privacy definition, private federated QR decomposition returns $\mathbf{Q}^s$ and $\mathbf{R}$ such that $\mathbf{A}^s$

Algorithm 6 Federated Gram-Schmidt. Client-Side Computations Are Marked in Blue

Input: Data matrices $\mathbf{A}^s \in \mathbb{R}^{n^s \times m}$ at sites $s \in [S]$.

Output: Partial matrices $\mathbf{Q}^s$ and full matrix $\mathbf{R}$ at sites $s \in [S]$

// Compute norm of first orthogonal vector.

1 **for** $s \in [S]$ **do**

2 $\quad \mathbf{u}_1^s \leftarrow \mathbf{a}_1^s$;

3 $\quad n_1^s \leftarrow \mathbf{u}_1^{s\top} \mathbf{u}_1^s$;

4 $n_1 \leftarrow \sum_{s=1}^{S} n_1^s$;

// Orthogonalize all subsequent vectors.

5 **for** $i \in [d] \setminus \{1\}$ **do**

// For each client s

6 $\quad$ **for** $s \in [S]$ **do**

// Normalise to unit norm

7 $\quad\quad \mathbf{q}_{i-1}^s \leftarrow \mathbf{u}_{i-1}^s / \sqrt{n_{i-1}}$;

// Compute relevant entries for $\mathbf{R}$

8 $\quad\quad \mathbf{r}_{l,i-1}^s \leftarrow \mathbf{q}_l^{s\top} \mathbf{a}_{i-1}^s$;

// Compute client residuals for current vector.

9 $\quad\quad$ **for** $j \in [i-1]$ **do**

10 $\quad\quad\quad r_{ij}^s \leftarrow \mathbf{u}_j^{s\top} \mathbf{a}_i^s / n_j$;

// Aggregate residuals

11 $\quad$ **for** $j \in [i-1]$ **do**

12 $\quad\quad r_{ij} \leftarrow \sum_{s=1}^{S} r_{ij}^s$;

// Aggregate R

13 $\quad\quad$ **for** $l \in [i]$ **do**

14 $\quad\quad\quad \mathbf{r}_{l,i-1} \leftarrow \sum_{s=1}^{S} \mathbf{a}_{l,i-1}^s$;

// Orthogonalize vector and compute norm.

15 $\quad$ **for** $s \in [S]$ **do**

16 $\quad\quad \mathbf{u}_i^s \leftarrow \mathbf{a}_i^s - \sum_{j=1}^{i-1} r_{ij} \cdot \mathbf{u}_j^s$;

17 $\quad\quad n_i^s \leftarrow \mathbf{u}_i^{s\top} \mathbf{u}_i^s$;

18 $\quad n_i \leftarrow \sum_{s=1}^{S} n_i^s$;

19 **for** $s \in [S]$ **do**

// Compute last column of R

20 $\quad \mathbf{q}_d^s \leftarrow \mathbf{u}_d^s / \sqrt{n_{i-1}}$;

21 $\quad$ **for** $l \in [k]$ **do**

22 $\quad\quad \mathbf{r}_d \leftarrow \mathbf{q}_d^s \mathbf{a}_l^s$;

23 $\quad \mathbf{Q}^s = [\mathbf{q}_1^s \cdots \mathbf{q}_d^s]$;

and $\mathbf{Q}^s$ stay private, meaning the raw data does not leave site s and $\mathbf{Q}^s$ can only be computed at s. $\mathbf{R}$ is common to all sites.

Proposition 3: At the end of federated Gram-Schmidt decomposition, the clients do not have access to more knowledge than their data matrices $\mathbf{A}^s$, the orthonormal partial matrices $\mathbf{Q}^s$, and the global matrices $\mathbf{R}$.

Proof: We consider the case, where we have no knowledge of the type of matrix (for instance, whether it is sparse, or triangular) to be orthonormalized and analyze the knowledge at the aggregator. Let $\mathbf{A}^s = \mathbf{Q}^s\mathbf{R}$. At the end of the algorithm, the following knowledge is available at the aggregator (We only show the global aggregates, assuming that they are aggregated using secure addition):

- $[n_1, \cdots, n_d]$, the norms of $[\mathbf{u}_1, \cdots, \mathbf{u}_d]$
- $\mathbf{R}$ the upper triangular matrix

$$\begin{bmatrix} \mathbf{q}_1 \cdot \mathbf{a}_1 & \mathbf{q}_1 \cdot \mathbf{a}_2 & \cdots & \mathbf{q}_1 \cdot \mathbf{a}_d \\ 0 & \mathbf{q}_2 \cdot \mathbf{a}_2 & \cdots & \mathbf{q}_2 \cdot \mathbf{a}_d \\ \vdots & 0 & \ddots & \vdots \\ 0 & 0 & \cdots & \mathbf{q}_d \cdot \mathbf{a}_d \end{bmatrix} \tag{12}$$

- the upper triangular matrix of residuals

$$\begin{bmatrix} \mathbf{u}_1 \cdot \mathbf{a}_2 & \mathbf{u}_1 \cdot \mathbf{a}_3 & \cdots & \mathbf{u}_1 \cdot \mathbf{a}_d \\ 0 & \mathbf{u}_2 \cdot \mathbf{a}_3 & \cdots & \mathbf{u}_2 \cdot \mathbf{a}_d \\ \vdots & 0 & \ddots & \vdots \\ 0 & 0 & \cdots & \mathbf{u}_d \cdot \mathbf{a}_d \end{bmatrix} \tag{13}$$

- In particular, we do not have access to the matrices $\mathbf{U}^s$, $\mathbf{Q}^s$ or $\mathbf{A}^s$.

Since $\mathbf{q}_i = \frac{\mathbf{u}_i}{n_i}$, the total information available amounts to the information encoded in the $\mathbf{R}$ matrix. We hence have only access to one factor of the decomposition which does not allow us to find a unique solution to $\mathbf{A} = \mathbf{QR}$. We specified our privacy goal as keeping the input matrices $\mathbf{A}^s$ and the orthogonal matrices $\mathbf{Q}^s$ private, therefore the presented algorithm is private as per our definition. □

It should be noted that $\mathbf{R}$ does disclose information on the data in form of the feature covariance matrix:

$$\mathbf{A}^\top\mathbf{A} = \mathbf{R}^\top\mathbf{Q}^\top\mathbf{QR} = \mathbf{R}^\top\mathbf{R} \tag{14}$$

V. Further Privacy Investigations

In this section, we apply federated QR factorization as a subroutine in federated PCA to reveal a privacy breach that can occur, if secure aggregation is not used, or if only 2 parties participate in the computation. The original algorithm uses QR factorization as the aggregation step [7]. This centralized procedure can be replaced by federated QR orthonormalization, presumably preventing the disclosure of the local summary statistics. The algorithm is mainly of academic interest, because more efficient schemes for PCA are available for star-like architectures. However, we will show that knowledge of the procedure allows an honest-but-curious participant to exactly reconstruct the other participants' input data. Our attack exploits the fact, that the input matrices are upper triangular and that we have full knowledge of the algorithm.

A. Algorithm

The algorithm [7] relies on sending a local $\mathbf{R}$ to the aggregator, where a secondary QR decomposition is performed (Algorithm 7). We suggest centering the data globally prior to the computation of the matrix (Line 1). This implies

subtracting the mean from each column and dividing by the standard deviation to obtain variables with a mean of 0 and a variance of 1. This also avoids having to account for inter site differences in mean later on. The next step is identical to the original: all the $\mathbf{R}$ matrices are computed at the clients (Line 3). The original algorithm recursively merges the $\mathbf{R}$ matrices at a processor to form the updated $\mathbf{R}'$ matrix until only one matrix remains. By computing the QR decomposition of all clients' $\mathbf{R}$ matrices at once using federated Gram-Schmidt decomposition, sending $\mathbf{R}$ can be avoided. The federated QR algorithm returns $\mathbf{R}$ at all the clients, therefore the final SVD can be directly computed at the client. The clients can also compute the partial left eigenvectors as $U^s = A^s V$ (Line 6).

Algorithm 7 Federated PCA Using QR Factorization [7] Client-Side Computations Are Marked in Blue.

Input: Data matrices $\mathbf{A}^s \in \mathbb{R}^{n^s \times m}$, # eigenvectors k.
Output: Partial singular vectors $\mathbf{U}^{s,k} \in \mathbb{R}^{n^s \times k}$ at sites $s \in [S]$ and $V^k \in \mathbb{R}^{k \times m}$

1 $\mathbf{A}^s \leftarrow$ `federated-centering()` ;
2 **for** $s \in [S]$ **do**
  // Compute local R at all clients
3 $\mathbf{Q}^s, \mathbf{R}^s \leftarrow$ `orthonormalize`$(\mathbf{A}^s)$;
4 $[\mathbf{Q}^s], \mathbf{R} \leftarrow$ `federated-gram-schmidt`$([\mathbf{R}^1, \cdots \mathbf{R}^s])$;
5 $\mathbf{U}, \Sigma, \mathbf{V}^\top =$ SVD$(\mathbf{R})$;
6 $\mathbf{U}^s \leftarrow \mathbf{A}^s\mathbf{V}$;

B. Privacy of Federated Gram-Schmidt on Upper Triangular Matrices

In the original algorithm, the communication of $\mathbf{R}$ poses a problem: Let $\mathbf{A}^\top\mathbf{A}$ be the covariance matrix of the data. Using the fact that $\mathbf{Q}$ is an orthonormal matrix, $\mathbf{R}$ can be used to compute the local covariance matrices of the data and hence leaks information (Equation (14)). Therefore, this algorithm is no more private than sending the entire set of local eigenvectors to the next party. The advantage of Algorithm 7 over its previous version is that it allows the computation of the global $\mathbf{R}$ without communicating the local $\mathbf{R}$ in clear text. The same can be achieved by using secure addition of the covariance matrices or computing the global $\mathbf{R}$ based on the data instead of $\mathbf{R}$. Nonetheless, we investigate this algorithm, because with close analysis it reveals a privacy breach if secure aggregation is not used or only two participants join. We show, that in this case the federated QR decomposition of upper triangular matrices is no more private than sending the upper triangular matrices themselves. The reason for this is the fact that the initial vector norm of the QR step is not technically an aggregate. We visualize the aggregation step in Algorithm 7 in Equation (15), as it is the motivation for our investigation. To avoid ambiguity, we denote the resulting upper triangular matrix $\mathbf{S}$ with elements $s_{i,j}$. For the remainder of this section, we assume that secure aggregation is not used.

Proposition 4: Let $\mathbf{R}^* = \left[\mathbf{R}^1\ \mathbf{R}^2 \cdots \mathbf{R}^s\right]^\top$ *be a vertical stack of upper triangular matrices, of which we want to compute the QR decomposition as* $\mathbf{R}^* = \mathbf{QS}$. *Denote* $\mathbf{Q}^s = [\mathbf{u}_1^s, \mathbf{u}_2^s, \cdots \mathbf{u}_d^s]$ *the block wise orthogonal matrices at sites* s. *It is possible to reconstruct all* $\left[\mathbf{R}^1\ \mathbf{R}^2 \cdots \mathbf{R}^s\right]$ *as well as all* $\left[\mathbf{Q}^1\ \mathbf{Q}^2 \cdots \mathbf{Q}^s\right]$ *when applying the federated QR algorithm on* $\mathbf{R}^*$, *given one knows that* $\mathbf{R}^s$ *are upper triangular.*

$$\mathbf{R}^* = \begin{bmatrix}\mathbf{R}^1\\ \mathbf{R}^2\\ \vdots\\ \mathbf{R}^s\end{bmatrix} = \begin{bmatrix}\begin{pmatrix} r_{11}^1 & r_{12}^1 & \cdots & r_{1d}^1\\ 0 & r_{22}^1 & \cdots & r_{2d}^1\\ \vdots & 0 & \ddots & \vdots\\ 0 & 0 & \cdots & r_{dd}^1\end{pmatrix}\\ \begin{pmatrix} r_{11}^2 & r_{12}^2 & \cdots & r_{1d}^2\\ 0 & r_{22}^2 & \cdots & r_{2d}^2\\ \vdots & 0 & \ddots & \vdots\\ 0 & 0 & \cdots & r_{dd}^2\end{pmatrix}\\ \vdots\\ \begin{pmatrix} r_{11}^s & r_{12}^s & \cdots & r_{1d}^s\\ 0 & r_{22}^s & \cdots & r_{2d}^s\\ \vdots & 0 & \ddots & \vdots\\ 0 & 0 & \cdots & r_{dd}^s\end{pmatrix}\end{bmatrix} = \begin{bmatrix}\mathbf{Q}^1\\ \mathbf{Q}^2\\ \vdots\\ \mathbf{Q}^s\end{bmatrix}\mathbf{S} \quad (15)$$

Proof: Let $\mathbf{R}^* = \left[\mathbf{R}^1\ \mathbf{R}^2 \cdots \mathbf{R}^s\right]^\top$ be the matrix to be decomposed into $\mathbf{Q}$ and $\mathbf{S}$. Denote $\mathbf{R}^s$ and $\mathbf{Q}^s$ the partial matrices only available at site $s \in [S]$. Denote $[\mathbf{u}_1^s \cdots \mathbf{u}_d^s]$ the partial orthogonal vectors at sites s. We show by induction on i that $\mathbf{R}^s$ and $\mathbf{Q}^s$ can be reconstructed at the aggregator based on the intermediate summary statistics exchanged during the execution of Algorithm 6. Let $i = 1$. In the first step of the algorithm (Lines 2 to 4, Algorithm 6), when computing $n_1 = \sum_{s=1}^S \mathbf{r}_1^{s\top}\mathbf{r}_1^s$ the clients disclose $(\mathbf{r}_{1,1}^s)^2$ to the aggregator which can compute

$$\mathbf{u}_1 = [\sqrt{\mathbf{r}_{1,1}^1}, 0, \cdots, 0, \sqrt{\mathbf{r}_{1,1}^2}, \cdots, 0, \sqrt{\mathbf{r}_{1,1}^s}, 0, \cdots, 0]^\top. \quad (16)$$

Let now $i = 2$ and $j = 1$, the residuals $p_{2,1}^s \leftarrow \frac{\mathbf{u}_1^{s\top}\mathbf{r}_2^s}{n_1}$ are computed and aggregated as $p_{2,1} = \sum_{s=1}^S p_{2,1}^s$ (Line 10 and Line 12). n_1 and $\mathbf{u}_1$ are known. We can compute $\mathbf{r}_{1,1}^s = \mathbf{q}_1^{s\top} s_{1,1}^s$ because $\mathbf{q}_1^s$ is orthonormal and only contains a single non-zero entry (Line 8, (s corresponds to r in the algorithm description)). For the same reason, we can also compute $r_{1,2}^s = \frac{p_{2,1}^s \cdot n_1}{q_{1,1}}$ (Line 10).

Finally, we compute $n_2 \leftarrow \sum_{s=1}^S n_2^s$, with $n_2^s = \mathbf{u}_2^{s\top}\mathbf{u}_2^s$ where $\mathbf{u}_2^s \leftarrow \mathbf{r}_2^s - \sum_{j=1}^{i-1} p_{21} \cdot \mathbf{u}_1^s$. This can be simplified to $\mathbf{u}_2^s = \begin{pmatrix} r_{12}^s - p_{21} \cdot u_{11}^s \\ r_{22}^s \end{pmatrix}$, because only $u_{1,1}^s$ is non-zero. $r_{1,2}^s$,

$p_{2,1}$ and $u^s_{1,1}$ are known, so $r^s_{2,2} = \sqrt{n^s_2 - (r^s_{1,2} - p_{2,1} \cdot u^s_{1,i})^2}$ can be computed, which in turn means $\mathbf{u}^s_2$ is known completely. At this point, $\mathbf{u}_1$, $\mathbf{u}_2$, $\mathbf{r}^s_1$ and $\mathbf{r}^s_2$ are known to the aggregator.

For the inductive step, we assume to have computed $\mathbf{Q}$ and $[\mathbf{R}^1 \cdots \mathbf{R}^s]^\top$ up to column $i-1$, we can compute column i. We set $j = i - 1$.

The residuals $p^s_{ij} \leftarrow \mathbf{u}^{s\top}_j \mathbf{r}^s_i / n_j$ are computed, and aggregated as $p_{ij} = \sum_{s=1}^{S} r^s_{ij}$. n_j, $\mathbf{r}_j$ and $\mathbf{u}_j$ are known for $j \in [i-1]$. We can compute r^s_{ij} for $j \in [i-1]$ via successive variable substitution due to the fact that the $\mathbf{R}^s_i$ are upper triangular.

$$\begin{cases} r_{1,i} = \dfrac{p_{i,1} \cdot n_1}{u_{1,1}} \\ r_{2,i} = \dfrac{p_{i,2} \cdot n_2 - u_{1,2} \cdot r_{1,i}}{u_{2,2}} \\ \vdots \\ r_{j-1,i} = \dfrac{p_{i,j} \cdot n_j - \sum_{n=0}^{i} u_{n-1} \cdot p_{i,n-1}}{u_{n-1,n-1}} \end{cases} \tag{17}$$

Finally, we compute $n_i \leftarrow \sum_{s=1}^{S} n^s_i$, with $n^s_i = \mathbf{u}^{s\top}_i \mathbf{u}^s_i$ where $\mathbf{u}^s_i \leftarrow \mathbf{r}^s_i - \sum_{j=1}^{i-1} p_{ij} \cdot \mathbf{u}^s_j$ which can be rewritten as

$$\mathbf{u}^s_i = \begin{pmatrix} r^s_{1,i} - \sum_{j=1}^{i-1} p_{i,j} \cdot u^s_{1,j} \\ r^s_{2,i} - \sum_{j=1}^{i-1} p_{i,j} \cdot u^s_{2,j} \\ \vdots \\ r^s_{j,i} - \sum_{j=1}^{i-1} p_{i,j} \cdot u^s_{j,j} \\ r^s_{i,i} \end{pmatrix} \tag{18}$$

where $\mathbf{r}_{i,i}$ is the only unknown. $\mathbf{r}_{ji} = \sqrt{n^s_i - \sum_{j=1}^{i-1} (\mathbf{r}^s_{j,i} - p_{i,j} \cdot \mathbf{u}^s_{j,i})^2}$, which in turn means $\mathbf{u}^s_j$ is complete, because n_i, $p_{i,j}$ and $\mathbf{u}^s_j$ as well as $\mathbf{r}_{j,i}$ are known. □

When using general matrices, even with only two participants, and if SMPC is used, this attack is not possible, because the first vector norm summarizes more than one element. However, the previous application highlights that tracking the parameters during federated iterations could reveal more information on the input data than the participants intend, especially when the methods are fully traceable and do not involve randomized steps. In the case of sparse matrices, a partial column $\mathbf{a}^s_i$ which contains no entries, can be detected at the aggregator as the inner product in $\mathbf{R}$ would be 0. The problem, with the methods presented here, is that the knowledge of algorithmic procedure and the absence of random elements in the algorithm allow us to backtrack more information than intended, given an 'attack angle'.

VI. Systems of Linear Equations

To showcase a more realistic use of our algorithm, we consider the application of federated orthonormalization for the solution of systems of linear equations. A popular use of QR decomposition is linear regression. For example, `R`'s `lm()` function uses the QR algorithm by default [21]. Here, we demonstrate how it is possible to solve a system of linear equations of the form $\mathbf{Ax} = \mathbf{b}$ with only one further round of communication, based on the QR decomposition.

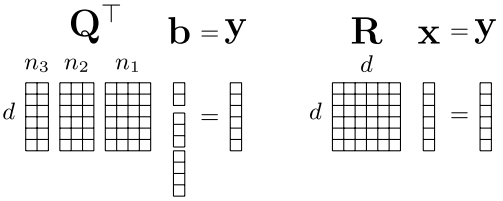

Fig. 2. Schematic solution of a system of linear equations based on federated QR factorization of matrix $\mathbf{A}$.

This technique can be used to replace the solver implemented for instance in [8]. It does not require matrix inversion and is therefore more suitable for large scale matrices. Let $\mathbf{A}$ and $\mathbf{b}$ be partitioned into $\mathbf{A}^s$ and $\mathbf{b}^s$ respectively, and $\mathbf{x}$ the solution common to all sites. After the QR decomposition of the matrix $\mathbf{A}$, $\mathbf{Q}^s$ is known at the sites, and $\mathbf{R}$ is known at all sites and the aggregator. In order to compute $\mathbf{x}$, the clients have to send their vector inner product of $\mathbf{y}^s = \mathbf{Q}^\top \mathbf{b}^s$ to the aggregator which securely computes the global vector $\mathbf{y} = \sum_s^S \mathbf{y}^s$. The aggregator can directly compute $\mathbf{x}$ by successive variable substitution and share the result with the clients (see Section II-F). With one additional step, one can also compute p-values and r^2 statistics. The clients compute the sum of the squared residuals as $rss_s = \sum(\mathbf{A}^s\mathbf{x} - \mathbf{b})^2$ and the sum of the squared fitted values $mss_s = \sum \mathbf{A}^s\mathbf{b}$ and send them to the aggregator, which computes the global sums $rss = \sum_s^S rss_s$, and $mss = \sum_s^S mss_s$. For the p-value, the variance is computes as $\sigma = \frac{rss}{n-m-1}(\mathbf{R}^\top\mathbf{R})^{-1}$, and standard error as $SE = \sqrt{\sigma}$. Here, we exploit the fact that the covariance matrix can be expressed using $\mathbf{R}$ (Equation (14)). The T-statistic used to determine the p-value can be computed as $T = \frac{\mathbf{x}}{SE}$. For more details see [8] where a more detailed description of the p-value calculation is provided. r^2 can be computed as follows: $r^2 = \frac{mss}{mss+rss}$.

VII. Experiments

A. Data Reconstruction

In order to show that Householder reflection and Givens rotation indeed lead to confidentiality losses when the intermediate parameters become known, we implemented the federated prototypes and logged the parameters that are known to all participants. We generate random Gaussian matrices of dimension 5000×10 and execute the federated prototypes. We then reconstruct the input data based on these aggregate statistics.

For the reconstruction of the input matrix based on the parameters disclosed during a Householder transformation, we log the reflection matrix $\mathbf{uu}^\top$, β, $sgn(\mathbf{A}_{1,1})$, m, and n and the first element of $\mathbf{u}$. (We assume without loss of generality that the first client is the attacker). We then apply the reconstruction described in Proposition 1. We reconstruct the data with an error of 2.21^{-15} averaged over 10 iterations. The error is calculated element-wise difference of the input and reconstructed matrices. In order to reconstruct the data based on the Given's parameters, we apply the procedure described in Proposition 2. The average reconstruction error over 10 repeated experiments is 1.3^{-14}. Therefore, we

TABLE II
RESULTS OF THE EXPERIMENTS

Dataset	Diabetes	WHO	Fish market
$\|\|\mathbf{Q}_c - \mathbf{Q}_f\|\|_F$	$3.0e^{-14}$	$1.1e^{-14}$	$3.8e^{-13}$
$\|\|\mathbf{R}_c - \mathbf{R}_f\|\|_F$	$1.7e^{-14}$	$7.5e^{-7}$	$9.3e^{-12}$
$\mathbf{x}_c - \mathbf{x}_f$	$2.23e^{-11}$	$4.5e^{-12}$	$3.04e^{-11}$
$\sum_d (p_c - p_f)$	0.003	0.019	0.029
$r_c^2 - r_f^2$	$1.4e^{-17}$	0	$2.5e^{-15}$

conclude that our theoretical attacks are indeed possible in realistic implementation.

B. Linear Regression

We implement the QR decomposition scheme and a prototype for linear regression in python to show that they provide accurate results in practice. In this experimental study we use three example data sets from `sklearn` and Kaggle: the Pima Indians diabetes [22], WHO life expectancy [23] and fish market [24] data sets. We split the data sets horizontally in 5 chunks. We compute the baseline reduced QR decomposition using `scipy.linalg.qr`. As an error measure, we use the Frobenius norm between the centralized and federated $\mathbf{Q}$ and $\mathbf{R}$ matrices ($||\mathbf{Q}_c - \mathbf{Q}_f||_F$,$||\mathbf{R}_c - \mathbf{R}_f||_F$). For the linear regression, we use the `lm` function in `R` as a reference, as it uses QR decomposition as its standard solver. As additional error measures, we compute the sum of the absolute differences between the coefficients ($\sum_{s\in[S]} \mathbf{x}_c - \mathbf{x}_f$), r^2-values ($r_c^2 - r_f^2$), and p-values ($\sum_{s\in[S]}(p_c - p_f)$). The results of these experiments are summarized in Table II. The matrices, coefficients and r^2 values are identical, and there only minor variations in the $p - value$.

C. Microbenchmark

Lastly, we implemented the algorithm using a secure multiparty computation python library MPyC [25]. This library implements the required SMPC primitives, including secure addition of floating point numbers (using fixed point arithmetic). Additionally, it allows convenient simulation of several clients. We generated small random matrices of dimensions 256×8 and used the simulation mode with 3 clients. We repeated the experiment 10 times and achieved an average run time of 0.7 seconds in simulation. In practice, the user should expect communication delay. As an error measure, we report the absolute difference in Froebenius norm for both the $\mathbf{R}$ and the $\mathbf{Q}$ matrices, between the federated and the centralized run (using scipy). The average errors are $6.024 * 10^{-10}$ for $\mathbf{Q}$ and $1.214 * 10^{-9}$ for $\mathbf{R}$. The recent publication by [12] reports run times of 227 seconds for the same operation. However, the run times should not really be compared. In [12] the input and output remain fully encrypted which is a fundamentally different scenario to the one presented here, where it is assumed that the input and output can remain unencrypted, in particular because they do not need to be shared with any party. This is a typical instance of a situation where prospective users need to trade-off privacy and run time.

D. Implementation & Hardware

The experiments were run on a standard laptop with 8 CPUs and 16 GB RAM. The algorithms were implemented in Python using `numpy` and `scipy`. The code can be found in the corresponding repository at https://github.com/AnneHartebrodt/federated-qr.

VIII. Discussion and Future Directions

In this manuscript, we evaluate the most popular algorithms for QR decomposition with respect to their confidentiality in a federated context. As explained in Section IV-A and Section IV-B, Householder reflection and Givens rotation have immediate drawbacks that make them unsuitable to hybrid federated learning where the parameters are securely aggregated, because it is possible to extract the original data from the parameters. This makes the presented federated Gram-Schmidt QR algorithm the only algorithm which does not trivially expose the original data under the assumed federated setting. We argue that the parameters revealed during the federated Gram-Schmidt orthonormalization procedure contain no more information than the upper triangular matrix $\mathbf{R}$, and therefore fulfill our privacy specification of federated QR decomposition.

In this article, we assume a hybrid federated learning setup, where the global parameters become known in clear text. This means, the results may only partially translate to systems which rely on encrypting the entire learning process under homomorphic encryption or computing the whole algorithm using more advanced secure multiparty computation schemes. These techniques are still expensive in practice [26], [27], but might be required to provide secure algorithms for Householder factorization and Givens rotation. If privacy is not a concern, detailed investigations of potential gains in transmission rounds would be required to find the most efficient QR scheme, most likely Givens algorithm according to our preliminary analysis.

The investigation of information leakage associated with the parameters exchanged during the federated QR orthonormalization spins a cautionary tale. We showed that it is possible to reconstruct the input matrices, if they are upper triangular, solely from the exchanged parameters, because the first aggregate is technically not an aggregate and triggers a revealing cascade. This means that with fewer than three parties, even the clients could reconstruct the other participants' matrices. We showed that for upper triangular matrices privacy breaches are possible. Therefore, further investigations on other special types of matrices will be required.

IX. Conclusion

The main objective of this work was to identify the most suitable algorithm for federated Gram-Schmidt orthogonalization assuming federated learning with secure parameter aggregation. To this end, we presented federated implementations of three popular QR algorithms and investigated them

with respect to their privacy. Popular applications of QR factorization, such as linear regression, can therefore be translated to the federated domain. The privacy must be considered carefully: The use of the outer vector product in Householder factorization, introduces a trivial confidentiality breach. Likewise, a trivial privacy leak in Givens rotation makes this algorithm unsuitable for secure additive multiparty computation. We come to the conclusion, that only Gram-Schmidt QR decomposition is suitable, due to its reliance on inner vector products. Special matrices, such as upper triangular matrices, may still be more vulnerable to confidentiality breaches.

ACKNOWLEDGMENT

This publication reflects only the authors' view and the European Commission is not responsible for any use that may be made of the information it contains.

REFERENCES

[1] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. Y. A. Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. 20th Int. Conf. Artif. Intell. Statist. (AISTATS)*, vol. 54, 2017, p. 10.

[2] P. Kairouz et al., "Advances and open problems in federated learning," *Found. Trends Mach. Learn.*, vol. 14, nos. 1–2, pp. 1–210, Jun. 2021.

[3] R. Cramer, I. B. Damgr ard, and J. B. Nielsen, *Secure Multiparty Computation and Secret Sharing*. Cambridge, U.K.: Cambridge Univ. Press, 2015.

[4] A. Hartebrodt, R. Nasirigerdeh, D. B. Blumenthal, and R. Röttger, "Federated principal component analysis for genome-wide association studies," in *Proc. IEEE Int. Conf. Data Mining (ICDM)*, Dec. 2021, pp. 1090–1095.

[5] W. Ford, *Numerical Linear Algebra With Applications: Using MATLAB*. Amsterdam, The Netherlands: Elsevier, 2014.

[6] A. H. Sameh, "On Jacobi and Jacobi-like algorithms for a parallel computer," *Math. Comput.*, vol. 25, no. 115, pp. 579–590, 1971.

[7] Z.-J. Bai, R. H. Chan, and F. T. Luk, "Principal component analysis for distributed data sets with updating," in *Advanced Parallel Processing Technologies*, J. Cao, W. Nejdl, and M. Xu, Eds. Berlin, Germany: Springer, 2005, pp. 471–483.

[8] R. Nasirigerdeh et al., "SPLINK: A hybrid federated tool as a robust alternative to meta-analysis in genome-wide association studies," *Genome Biol.*, vol. 23, no. 1, pp.'1–24, Jan. 2022.

[9] P. Snyder, "Yao's garbled circuits: Recent directions and implementations," Tech. Rep., 2014. [Online]. Available: https://www.peteresnyder.com/static/papers/yao-lit-review-2014.pdf

[10] R. A. Beezer. (2016). *A Second Course in Linear Algebra*. Accessed: Jan. 31, 2022. [Online]. Available: http://linear.ups.edu/scla/html/gfdl.html

[11] D. Chai, L. Wang, K. Chen, and Q. Yang, "Secure federated matrix factorization," 2019, *arXiv:1906.05108*.

[12] D. Froelicher et al., "Scalable and privacy-preserving federated principal component analysis," Mar. 2023, *arXiv:2304.00129*.

[13] O. Sluciak, H. Straková, M. Rupp, and W. N. Gansterer, "Distributed gram-Schmidt orthogonalization based on dynamic consensus," in *Proc. 46th Asilomar Conf. Signals, Syst. Comput. (ASILOMAR)*, Nov. 2012, pp. 1207–1211.

[14] O. Sluvciak, H. Straková, M. Rupp, and W. Gansterer, "Distributed gram-Schmidt orthogonalization with simultaneous elements refinement," *EURASIP J. Adv. Signal Process.*, vol. 2016, no. 1, pp. 1–13, Dec. 2016.

[15] H. Straková, W. N. Gansterer, and T. Zemen, "Distributed QR factorization based on randomized algorithms," in *Parallel Processing and Applied Mathematics*, R. Wyrzykowski, J. Dongarra, K. Karczewski, and J. Waśniewski, Eds. Berlin, Germany: Springer, 2012, pp. 235–244.

[16] J. H. Yoo, H. Jeong, J. Lee, and T.-M. Chung, "Federated learning: Issues in medical application," in *Proc. Int. Conf. Future Data Secur. Eng.*, 2021, pp. 3–22.

[17] F. Song, H. Ltaief, B. Hadri, and J. Dongarra, "Scalable tile communication-avoiding QR factorization on multicore cluster systems," in *Proc. ACM/IEEE Int. Conf. High Perform. Comput., Netw., Storage Anal.*, Nov. 2010, pp. 1–11.

[18] Y. Zhang, P. Zheng, and W. Luo, "Privacy-preserving outsourcing computation of QR decomposition in the encrypted domain," in *Proc. 18th IEEE Int. Conf. Trust, Secur. Privacy Comput. Commun./13th IEEE Int. Conf. Big Data Sci. Eng. (TrustCom/BigDataSE)*, Aug. 2019, pp. 389–396.

[19] C. Luo, K. Zhang, S. Salinas, and P. Li, "Efficient privacy-preserving outsourcing of large-scale QR factorization," in *Proc. 16th IEEE Int. Conf. Trust, Secur. Privacy Comput. Commun., 11th IEEE Int. Conf. Big Data Sci. Eng., 14th IEEE Int. Conf. Embedded Softw. Syst.*, Aug. 2017, pp. 917–924.

[20] A. Hartebrodt, R. Röttger, and D. B. Blumenthal, "Federated singular value decomposition for high dimensional data," 2022, *arXiv:2205.12109*.

[21] *R: A Language and Environment for Statistical Computing*, R Core Team, R Found. Stat. Comput., Vienna, Austria, 2021. [Online]. Available: https://www.R-project.org/

[22] (2022). *Pima Indians Diabetes Database*. Accessed: Jan. 31, 2022. [Online]. Available: https://www.kaggle.com/uciml/pima-indians-diabetes-database

[23] (2022). *W Life Expectancy Data Set*. Accessed: Jan. 31, 2022. [Online]. Available: https://www.kaggle.com/kumarajarshi/life-expectancy-who

[24] (2022). *Fish Market Data Set*. Accessed: Jan. 31, 2022. [Online]. Available: https://www.kaggle.com/aungpyaeap/fish-market

[25] B. Schoenmakers. (2018). *MPyC—Python Package for Secure Multiparty Computation*. Accessed: May 2, 2023. [Online]. Available: https://www.win.tue.nl/~berry/mpyc/TPMPC2018.pdf

[26] H. Cho, D. J. Wu, and B. Berger, "Secure genome-wide association analysis using multiparty computation," *Nature Biotechnol.*, vol. 36, no. 6, pp. 547–551, Jul. 2018.

[27] M. Al-Rubaie, P.-Y. Wu, J. M. Chang, and S.-Y. Kung, "Privacy-preserving PCA on horizontally-partitioned data," in *Proc. IEEE Conf. Dependable Secure Comput. (DSC)*, Aug. 2017, pp. 280–287.