Secure Federated Learning in Linear Regression: A Zero-Knowledge Protocol

Firdavsbek Ismoilov$^{1[0009-0002-7745-6092]}$ and
Alisher Ikramov$^{1,2,3[0000-0002-4302-6791]}$

[1] New Uzbekistan University, Tashkent, Uzbekistan
`{f.ismoilov1,a.ikramov}@newuu.uz`
https://newuu.uz

[2] National University of Uzbekistan, Tashkent, Uzbekistan
https://nuu.uz/

[3] V.I.Romanovskiy Institute of Mathematics, Uzbekistan Academy of Sciences, Tashkent, Uzbekistan
https://mathinst.uz/

Abstract. Federated Learning has emerged as a powerful paradigm for collaborative model training across distributed data holders while preserving data confidentiality. Despite these advantages, naive FL protocols may still leak sensitive information through shared model updates. In this paper, we propose a Zero-Knowledge protocol for secure federated linear regression that conceals the closed-form expressions and intermediate gradients exchanged among participants. We provide a proof of this protocol to be a Zero-Knowledge as well as the estimates for the probability of successful recovery of the initial data. The server randomly generate an invertible matrix, each client measures gradients on this matrix and sends only this information back. Server sums values from all clients and finds the optimal solution. We also proposed use of a random rescaling factor for each client to hide size of their dataset.
Unlike prior works focusing solely on gradient encryption or differential privacy, we address the security challenge through a lightweight ZK proof strategy that restricts knowledge leakage to the global model alone. Furthermore, by formulating the update step as a fixed-point iteration, our approach can be integrated with iterative optimizers such as FedAvg or FedSGD.
We conducted experiments to verify correctness of the Protocol and its advansed version. Experimental results on a synthetic dataset, under various noise and scaling conditions, show that our protocol conserves the same evaluation scores while maintaining strong confidentiality guarantees.

Keywords: Federated Learning · Linear Regression · Secure Multi-Party Computation · Secure Aggregation · Gradient Obfuscation

1 Introduction

Federated Learning (FL) enables a collection of distributed data holders (clients) to collaboratively train a global model without ever exchanging their raw data

[3,4]. Instead of pooling information at a central location, each client locally computes model updates on its private dataset and sends those updates to a central server. This approach significantly reduces direct exposure of sensitive information compared to classical centralized training. Nonetheless, reconstruction attacks on shared gradients [6] and other more subtle leakage pathways [8] demonstrate that FL still faces pressing privacy challenges. In linear models, for instance, certain features of local data can occasionally be recovered by inverting or analyzing the gradients [7], compromising client confidentiality.

To mitigate these vulnerabilities, a variety of privacy-enhancing techniques, including homomorphic encryption [9], differential privacy [2], and secure multiparty computation [1] have been proposed. While these solutions offer rigorous privacy bounds, they frequently incur high computational or communication overhead, making them challenging to scale in practical systems. Zero-Knowledge (ZK) proofs present another dimension in secure distributed learning, allowing a client to prove correct computation of local updates without exposing the actual data points [10]. However, existing ZK-based protocols often require intricate multi-round interactions or specialized cryptographic primitives, hindering their adaptability to large-scale linear regression tasks.

In this paper, we introduce a secure FL protocol for linear regression that protects clients' data from a potentially curious central server by leveraging ZK techniques. Our contributions include:

1. **Closed-Form + Iterative Interpretations.** Although our protocol leverages a closed-form solution for linear regression, it is also amenable to iterative algorithms, effectively interpreting the secure update step as a fixed-point iteration. Consequently, the same principle can be incorporated into standard FL optimizers like FedAvg or FedSGD [11,12].
2. **Client-specific column permutations.** The server sends a uniquely permuted weight matrix $\mathbf{W}^{(i)}$ to every client and later inverts that permutation during aggregation, an idea inspired by permutation defences such as FedPerm [13]. This extra indirection misleads adversaries with negligible overhead.
3. **Rescaling for Enhanced Privacy.** Each client randomly samples a scalar β in a small interval (e.g. $(-0.1, 0.1)$) and multiplies its local gradient matrix by $(1-\beta)$. This technique adds an additional obfuscation that even subtle patterns in the gradients remain hidden from adversaries.
4. **Experimental Evaluation.** We conduct experiments on synthetic data for multiple clients, with varying dataset sizes, feature distributions, and noise levels. We also evaluate different ranges of β to demonstrate the protocol's efficacy at masking private updates. Our results indicate that the solution achieves strong privacy guarantees while closely matching the performance of conventional, non-secure FL systems on metrics such as mean squared error and R^2

The remainder of this paper is organized as follows. The **Related Work** section reviews the current literature in secure federated learning and privacy-preserving linear regression, highlighting how this research advances the field. The **Methods** section describes the proposed secure FL protocol in detail, including an overview of the ZK justification and a discussion of how gradient rescaling is integrated. The **Experiments** section presents the evaluation setup, reporting empirical results under various parameter settings. Finally, the **Conclusion** section discusses our key findings—particularly noting that all clients must agree on the same number of features n—and suggests future directions, including applying this approach to more complex ML tasks.

2 Related Work

Early research in Federated Learning focused on methods for aggregating locally computed gradients while reducing communication overhead [3, 4]. However, as reconstruction attacks on gradient updates became more evident, the field began to adopt privacy-preserving techniques such as differential privacy [2], homomorphic encryption [9], and secure multiparty computation [1]. Although these methods provide strong theoretical guarantees, their computational and communication costs can be prohibitive in practice.

Zero-Knowledge (ZK) proofs represent another significant line of research for privacy-preserving machine learning [10]. In the ZK paradigm, a prover can demonstrate knowledge or correctness of a computation to a verifier without revealing any other information. Most existing ZK protocols for FL require elaborate interaction patterns or specialized cryptographic primitives (e.g., circuit-based proofs), limiting their scalability. Several works have tackled secure linear regression specifically, often leveraging additive masking techniques or encrypted gradients [6, 8]. However, few have considered random transformations (rescaling) at the model-parameter level to obfuscate gradient contributions, an approach that can be combined with iterative optimizers.

An orthogonal line of defence randomises the *ordering* of model parameters before clients perform local computations. Mozaffari *et al.* introduce **FedPerm**, where intra-model parameter shuffling strengthens both privacy and robustness by decoupling the semantic meaning of individual coordinates during aggregation [13]. Our protocol adopts a lighter variant: each client receives a unique permutation of the columns of the weight matrix $\mathbf{W}$, and the server later inverts that permutation before summing the masked gradient matrices. As discussed in the next section, this extra indirection further misleads an adversary who might otherwise exploit consistent column positions to correlate gradients across rounds.

Another direction in privacy-preserving FL proposes modifications to standard optimizers such as FedAvg and FedProx to handle statistical heterogeneity [11, 12]. While these methods address the data distribution disparities across clients, they typically assume that sharing gradients does not pose significant privacy risks. By contrast, our approach treats gradient sharing itself as a confidentiality bottleneck, by retrieving the gradient function coefficients locally on the server based on the received gradient update matrix and previously sent weight matrix $\mathbf{W}$ in one communication round, and introducing a random scaling $(1-\beta)$ to further mask local data properties. The random perturbations have been explored in differential privacy [2] and certain homomorphic schemes [9].

In summary, this work positions itself at the intersection of ZK-based techniques, secure aggregation protocols, and standard FL optimizers. It aims to achieve privacy guarantees akin to more complex cryptographic approaches yet does so with minimal overhead and broad compatibility with iterative training pipelines.

3 Methods

The proposed protocol aims to ensure secure federated linear regression through transformations on the gradient update matrix received from a client and client-side gradient rescaling. The foundational idea is that each client locally computes perturbed gradients in such a way that they can still be aggregated by the server to produce an accurate global model. However, these transformations obfuscate the raw updates sufficiently to prevent an adversary (including a curious server as usually) from inferring private details about client data.

3.1 Overview of the Secure Protocol

3.1.1 Problem Setup. Consider N clients in an FL system, each holding a local dataset $\mathcal{D}_i = \{(\mathbf{X}^{(i)}, \mathbf{y}^{(i)})\}$, where $\mathbf{X}^{(i)} \in \mathbb{R}^{m_i \times n}$ is the feature matrix and $\mathbf{y}^{(i)} \in \mathbb{R}^{m_i}$ is the target vector. The global objective is to solve the linear regression problem for all N clients:

$$\min_{\mathbf{w}} \sum_{i=1}^{N} L_i(\mathbf{w}) = \sum_{i=1}^{N} \frac{1}{m_i} \|\mathbf{X}^{(i)}\mathbf{w} - \mathbf{y}^{(i)}\|^2, \tag{1}$$

without exposing $\mathbf{X}^{(i)}$, $\mathbf{y}^{(i)}$, or raw gradients.

3.1.2 Generation of Invertible Weight Matrix. To initialize the process, the central server draws a randomly generated matrix $\mathbf{W} \in \mathbb{R}^{(n+2)\times(n+2)}$ with last row consisting of 1's. This matrix is formed by horizontally stacking $n+2$ randomly generated weight vectors $\mathbf{w} \in \mathbb{R}^{(n+2)}$ as columns. A weight vector $\mathbf{w}$ is in the form of $[w_0, w_1, ..., w_n, 1]^\top$ the first $n+1$ elements of which represent the weights (parameters) of a linear regression model and the last '1' element is added to count for the free term in a loss gradient function $L_i'(\mathbf{w}) = \frac{\partial L_i}{\partial \mathbf{w}}$.

3.1.3 Local Gradient Calculation. First, each client extends its feature matrix to $\tilde{\mathbf{X}}^{(i)} \in \mathbb{R}^{m_k \times (n+1)}$ by adding a column of ones (for the intercept) as the first column. Removing the last row of $\mathbf{W}$ (which consists of ones) yields a submatrix $\tilde{\mathbf{W}}$ of dimension $\mathbb{R}^{(n+1)\times(n+2)}$. The client then forms the matrix

$\mathbf{Y}^{(i)} \in \mathbb{R}^{m_i \times (n+2)}$ by stacking $(n+2)$ copies of $\mathbf{y}^{(i)}$ horizontally to align with the columns of $\tilde{\mathbf{W}}$. For linear regression with a mean squared error MSE loss, the local gradient matrix can be computed (without normalization for batch size) as:

$$\mathbf{L}^{(i)} = \tilde{\mathbf{X}}^{(i)^\top} \left(\tilde{\mathbf{X}}^{(i)} \tilde{\mathbf{W}} - \mathbf{Y}^{(i)}\right) \quad \in \mathbb{R}^{(n+1)\times(n+2)}. \tag{2}$$

where $\mathbf{Y}^{(i)}$ has the same number of columns as $\tilde{\mathbf{W}}$.

3.1.4 Rescaling $(1-\beta)$.

To provide an additional layer of security, each client may sample a small β uniformly from $(-0.1, 0.1)$ (or another chosen interval) and multiply its locally computed gradient matrix $\mathbf{L}^{(i)}$ by $(1-\beta)$ before sending:

$$\mathbf{L}^{(i)}_{\text{scaled}} = (1-\beta_i)\,\mathbf{L}^{(i)} \tag{3}$$

This simple yet effective mechanism makes it more challenging for an adversary to isolate exact gradient patterns across clients.

3.1.5 Secure Aggregation.

All correspondence between server and client must be encrypted. The server collects all client submissions and sums them to obtain:

$$\mathbf{L} = \sum_{i=1}^{N} \mathbf{L}^{(i)}_{\text{scaled}}, \tag{4}$$

where N is the total number of clients. Since each $\mathbf{L}^{(i)}_{\text{scaled}}$ shares the same structure (i.e., are in dimension of $\mathbb{R}^{(n+1)\times(n+2)}$), their summation is element-wise consistent.

3.1.6 Global Model Recovery.

Finally, to recover the global model parameters, the server computes:

$$\mathbf{C} = \mathbf{L}\,\mathbf{W}^{-1}$$

By partitioning $\mathbf{C}$ into $\left[\mathbf{A} \mid \mathbf{b}\right]$ (splitting off the last column), we solve the system

$$\mathbf{A}\,\mathbf{w}_{\text{opt}} = -\mathbf{b},$$

where $\mathbf{w}_{\text{opt}} \in \mathbb{R}^{(n+1)}$ is the optimal weight vector for all the clients. Once solved for $\mathbf{w}_{\text{opt}}$, it can be used as the global linear regression model, i.e., $\hat{y} = \mathbf{w}_{\text{opt}}^\top \tilde{x}$.

3.2 Permutation-Based Obfuscation

To amplify privacy, the server maintains a client-specific column permutation $\pi_i : \{1, \ldots, n+2\} \to \{1, \ldots, n+2\}$ for every client i.

1. **Dispatch phase.** The server permutes the columns of $\mathbf{W}$ via the permutation matrix $\mathbf{P}_{\pi_i}$, obtaining $\mathbf{W}^{(i)} = \mathbf{W}\mathbf{P}_{\pi_i}$, and transmits $\mathbf{W}^{(i)}$ to client i.

2. **Local computation.** Client i computes its rescaled gradient matrix $\mathbf{L}_{\text{perm}}^{(i)} = (1-\beta_i)\mathbf{L}^{(i)}$ in the permuted coordinate system.
3. **Restoration phase.** Upon receipt, the server applies the inverse permutation, $\mathbf{L}_{\text{scaled}}^{(i)} = \mathbf{L}_{\text{perm}}^{(i)}\mathbf{P}_{\pi_i}^{\top}$, before secure aggregation.

Because each client's column ordering is unique and secret, an eavesdropper – or even a curious server colluding across rounds – cannot reliably align gradient coordinates to mount reconstruction attacks. Compared with more heavyweight shuffling frameworks such as FedPerm [13], this method incurs negligible cost while integrating seamlessly with our Zero-Knowledge workflow.

3.3 Security Analysis

Lemma 1. *If $\mathbf{X}^{(i)} \in \mathbb{R}^{m_i \times n}, \mathbf{y}^{(i)} \in \mathbb{R}^{m_i}$, $m_i > 2n$ for each $i \in [1, N]$, and all features in $\mathbf{X}^{(i)}, \mathbf{y}^{(i)}$ have joint symmetrical distribution with origin being the mean of that distribution, then, given $\mathbf{L}^{(i)}$ and $\mathbf{W}$, an adversary cannot uniquely determine $\mathbf{X}^{(i)}$ or $\mathbf{y}^{(i)}$, that is, the probability of guessing $\mathbf{X}^{(i)}$ and $\mathbf{y}^{(i)}$ correctly is less or equal 2^{-m_i}.*

Proof. Let us consider a client that receives matrix $\mathbf{W}$ as a request. We form a matrix $\mathbf{B}^{(i)}$ by concatenating to the matrix $\mathbf{X}^{(i)}$ column of $-\mathbf{y}^{(i)}$. Then the client's response is $\mathbf{L}^{(i)} = \mathbf{X}^{(i)^{\top}}\mathbf{B}^{(i)}\mathbf{W}$. As the matrix $\mathbf{W}$ is invertible by choice, we can find

$$\mathbf{L}^{(i)}\mathbf{W}^{-1} = \mathbf{X}^{(i)^{\top}}\mathbf{B}^{(i)} \tag{5}$$

As the left side contains all the information the server knows from the interaction, we can consider the uniqueness of the right side.

Consider a matrix $\mathbf{E} \in \mathbb{R}^{m_i \times n}$ and a vector $\mathbf{v} \in \mathbb{R}^{m_i}$. If we concatenate the matrix $\mathbf{E}$ and the vector $\mathbf{v}$, we get the matrix $\mathbf{E}_1$. Then, a variation of the right side of (5) can be written as $(\mathbf{X}^{(i)} + \mathbf{E})^{\top}(\mathbf{B}^{(i)} + \mathbf{E}_1) = \mathbf{X}^{(i)^{\top}}\mathbf{B}^{(i)} + \mathbf{X}^{(i)^{\top}}\mathbf{E}_1 + \mathbf{E}^{\top}\mathbf{B}^{(i)} + \mathbf{E}^{\top}\mathbf{E}_1$. If we can prove the existence of non-trivial $\mathbf{E}_1$ such that $\mathbf{Q}_{\mathbf{E_1}} = \mathbf{X}^{(i)^{\top}}\mathbf{E}_1 + \mathbf{E}^{\top}\mathbf{B}^{(i)} + \mathbf{E}^{\top}\mathbf{E}_1 = \mathbf{0}$, then we confirm the first part of the lemma.

First, we consider left n columns of $\mathbf{Q}_{\mathbf{E_1}}$. They form a symmetric matrix that is equal to $\mathbf{X}^{(i)^{\top}}\mathbf{E} + \mathbf{E}^{\mathbf{X^{(i)}}} + \mathbf{E}^{\mathbf{E}}$. Let us select some numbers $j_1, j_2, \ldots, j_t$ such that $1 \le j_1 < j_2 < \ldots < j_t \le m_i$, $1 \le t \le m_i$. Now, we put the rows $j_1, j_2, \ldots, j_t$ of $\mathbf{E}$ to be equal to corresponding rows of $-2\mathbf{X}^{(i)}$, while we put the remaining rows to be 0. It is obvious that $\mathbf{X}^{(i)^{\top}}\mathbf{E} + \mathbf{E}^{\top}\mathbf{X}^{(i)} + \mathbf{E}^{\top}\mathbf{E} = \mathbf{0}$ for such matrices $\mathbf{E}$. Including the trivial case, we have 2^{m_i} ways of choosing such matrices. Thus, the probability of correctly guessing which state we have in the Client cannot be greater than 2^{-m_i}.

Second, we consider the last column of $\mathbf{Q}_{\mathbf{E_1}}$. It is equal to $(\mathbf{X}^{(i)}+\mathbf{E})^{\top}(-\mathbf{y}^{(i)}+\mathbf{v})$. We need $\mathbf{E}^{\top}(-\mathbf{y}^{(i)} + \mathbf{v}) + \mathbf{X}^{(i)^{\top}}\mathbf{v} = \mathbf{0}$. As we already chose $\mathbf{E}$, we can only change $\mathbf{v}$. Thus, we need such vector $\mathbf{v}$ that

$$(\mathbf{X}^{(i)} + \mathbf{E})^\top \mathbf{v} = \mathbf{E}^\top \mathbf{y}^{(i)}. \tag{6}$$

We put $v_{j_k} = 2y_{i,j_k}$ and other components of $\mathbf{v}$ to be 0. By choice of $\mathbf{E}$ the equation (6) is satisfied.

Thus, $\mathbf{X}^{(i)}$ and $\mathbf{y}^{(i)}$ are unrecoverable.

Theorem 1. *If conditions of Theorem 1 are satisfied, then the constructed Protocol is a Zero-Knowledge Protocol for training the Linear Regression Model.*

3.4 Iterative Interpretation and Compatibility

Although the above formulation leverages a closed-form solution, it can also be interpreted as a fixed-point iteration for model parameters. Specifically, one can view the transformation $\mathbf{W}$ and subsequent inversion as a mechanism that produces "pseudo-updates" for each column in $\mathbf{W}$. In principle, these updates could be recomputed at regular intervals (i.e., per round or per epoch), similarly to how FedAvg combines local model updates from each client [11].

- **Fixed-Point Representation**. Let $\mathbf{w}^{(k)}$ denote the global model at iteration k. A simplified interpretation of the proposed method is that each local client is computing a function $f_i(\mathbf{w}^{(k)}; X^{(i)}, y^{(i)}, \beta_i)$ that, when aggregated, yields the next global parameter estimate $\mathbf{w}^{(k+1)}$. The transform $\mathbf{W}$ ensures that the server sees only a masked version of f_i, preserving privacy under ZK constraints.
- **FedAvg or FedSGD**. If we replace the analytical solution by an iterative gradient-based approach – where each round uses the same or updated matrix $\mathbf{W}$ – the server would incorporate the aggregated, rescaled gradients into its global model at each iteration, akin to standard FL. A small learning rate could be applied to the aggregated updates, preventing divergence in case of noisy transformations or large β perturbations.

In practice, an iterative strategy may be especially useful if the data distribution is non-stationary or if the model has non-linear components (e.g., in polynomial feature expansions). Although our experiments center on linear regression with a single closed-form update step, the design readily extends to iterative optimization schemes by repeating the transform-and-aggregate procedure until convergence criteria are met [4, 11, 12].

4 Experiments

This section demonstrates the performance of our secure federated linear regression protocol on synthetic datasets. We implement two versions of the protocol – referred to as "Protocol v1" and "Protocol v2" – to highlight the effects of optional gradient rescaling, as described in Section 3. All experiments are conducted on randomly generated data, illustrating how the protocol behaves under different noise levels, data sizes, and β-ranges. For full reproducibility, the complete source code is available online at https://github.com/ifirdavs/linear-regression-fl.

4.1 Experimental Setup

We consider a setting with $N = 10$ clients, each possessing its own dataset. In all simulations, each client's data $\big(X^{(i)}, y^{(i)}\big)$ is generated by sampling m_i data points from a normal distribution, adding random Gaussian noise, and then splitting 90% of the data for training and 10% for testing. The data sizes $\{m_i\}$ are drawn uniformly in the range [1000, 10000]. The number of features is fixed at $n = 7$, ensuring all clients share the same dimensionality. Following standard practices in synthetic Federated Learning (FL) test environments [3, 5], we assign each client i a standard deviation σ_i to represent its inherent data variability. We then sample the noise level for that client based on the ratio of noise standard deviation $\sigma_{noise,i}$ to σ_i (similar to other synthetic benchmarks in FL [4, 11]), specifically:

- **Low** noise range: $\sigma_{noise,i} \sim \text{Uniform}\big(0, 0.5\sigma_i\big)$
- **Medium** noise range: $\sigma_{noise,i} \sim \text{Uniform}\big(0.5\sigma_i, \sigma_i\big)$
- **High** noise range: $\sigma_{noise,i} \sim \text{Uniform}\big(\sigma_i, 2\sigma_i\big)$

1. **Protocol v1** (base). This variant follows the algorithm in Section 3 but does **not** normalize local gradients by the batch size. Each client calculates its local gradient matrix $\mathbf{L}^{(i)}$ for the random weight matrix $\mathbf{W}$ and directly submits it to the server. The server sums these contributions, multiplies by $\mathbf{W}^{-1}$, and solves the resulting linear system for the global parameters $\mathbf{w}_{\text{opt}}$.
2. **Protocol v2** (with rescaling). This variant adds the optional rescaling factor $\big(1 - \beta^{(i)}\big)$ (where $\beta^{(i)}$ is sampled from a small interval, e.g., $(-0.1,\ 0.1)$) before sending each client's gradient matrix to the server. The motivation is to further obfuscate gradient magnitudes, making it more difficult for a curious server to reconstruct private data while maintaining the correctness of the aggregate update in expectation.

For both protocols, we measure performance by mean squared error (MSE) and coefficient of determination (R^2) on the combined test sets from all clients. We also track computational overhead (such as the cost of inverting $\mathbf{W}$) and the effect of varying β-ranges on accuracy.

4.2 Results on Synthetic Data

Overall Accuracy. In both protocols, the final global model parameters $\mathbf{w}_{\text{opt}}$ generally achieve near-optimal regression performance, closely matching a non-secure baseline that aggregates exact gradients without any transformations. Table 1 provides a representative snapshot of the results for different noise levels and β-ranges.

When β-ranges are smaller, the aggregated update approximates the true gradient more closely, which yields nearly identical MSE and R^2 compared to Protocol v1. As the absolute range for β increases (e.g., $(-0.1,\ 0.1)$), some degradation in accuracy may occur, but the effect remains modest, indicating the protocol's robustness under moderate rescaling.

	v1		v2		
Noise	MSE	R^2	β-Range	MSE	R^2
low	895.95	0.2315	$(-0.01, 0.01)$	895.97	0.2315
			$(-0.05, 0.05)$	896.04	0.2314
			$(-0.1, 0.1)$	896.16	0.2313
medium	920.77	0.2245	$(-0.01, 0.01)$	920.79	0.2245
			$(-0.05, 0.05)$	920.87	0.2244
			$(-0.1, 0.1)$	920.99	0.2243
high	1008.37	0.2061	$(-0.01, 0.01)$	1008.39	0.2061
			$(-0.05, 0.05)$	1008.49	0.2060
			$(-0.1, 0.1)$	1008.62	0.2059

Table 1. MSE and R^2 metrics for different noise levels and β-ranges. **Note:** We define *low*, *medium*, and *high* noise as described in Section 4.1.

Effect of Noise. Higher levels of noise naturally increase the MSE and reduce R^2 values, irrespective of whether rescaling is used. However, even in noisy scenarios, the final performance remains comparable to standard FL benchmarks [2, 11], demonstrating that our masking approach does not unduly amplify random data fluctuations.

4.3 Computational Overhead and Complexity

A central computational cost in both protocols is the matrix inversion of $\mathbf{W} \in \mathbb{R}^{(n+2)\times(n+2)}$. Since n (the number of features) is typically much smaller than the number of data points m_i, the $O\big((n+2)^3\big)$ complexity of inverting $\mathbf{W}$ is not onerous for moderate n. Each client's computation primarily involves multiplying $\tilde{X}^{(i)}$ by $\tilde{\mathbf{W}}$ and summing over the batch dimension, which is linear in $(n+1)\times(n+2)$. In large-scale FL deployments with many features, hierarchical or blockwise strategies for constructing $\mathbf{W}$ could mitigate these operations [4, 12].

In Protocol v2, an additional overhead arises from sampling and applying the rescaling factor $(1 - \beta^{(i)})$, but this is negligible compared to gradient computation and matrix inversion. Overall, our method preserves the low per-round communication cost typical of FL, transmitting only masked gradients rather than large encrypted payloads [1, 9].

5 Discussion

Our experiments confirm that:

1. **Security vs. Performance Trade-off**. Rescaling by $(1 - \beta)$ introduces a controlled "fuzziness" to each client's gradient updates, which can marginally affect accuracy but offers stronger defense against gradient inversion attacks

[8]. For sufficiently small β-ranges, there is minimal degradation in the final model.
2. **Benefit of client-specific permutations**. Unique column permutations make it difficult for an adversary to align gradient coordinates across rounds, adding an orthogonal layer of protection with almost no computational cost and echoing findings in FedPerm [13].
3. **Consistency with Iterative Updates**. Although demonstrated here in a single round for closed-form linear regression, the approach naturally generalizes to iterative federated optimization methods (e.g., FedAvg) by repeating the secure gradient transformation each round. Our tests confirm that repeated rounds remain stable as long as β-values are not excessively large.
4. **Robustness to Data Heterogeneity**. Clients receive no information about others' data distribution, and the server merely sees aggregated masked gradients, thus mitigating data leakage. Even when the clients' data have different means, scales, and noise levels, the final model successfully converges to a unified parameter set that balances the overall loss across participating clients [3, 12].

In summary, both Protocol v1 (base) and Protocol v2 (rescaling) show promise for practical deployments, offering strong security properties with minimal compromise on regression accuracy.

6 Conclusion

We have presented a Zero-Knowledge-based federated linear regression protocol that prevents sensitive data exposure through carefully designed transformations of client-side gradient updates. A randomly generated weight matrix $\mathbf{W}$ and a rescaling parameter $(1 - \beta)$ ensure that neither the central server nor third parties can easily invert aggregated updates to extract local data. Incorporating client-specific column permutations further obscures the semantic mapping between gradient coordinates and feature dimensions, offering an additional layer of protection with minimal overhead and aligning with recent permutation-based defences [13]. Experimental results indicate that this approach achieves performance comparable to non-secure methods, maintaining favorable mean squared error and R^2 values even when subjected to moderate noise or larger β-ranges.

As Linear Regression methods are widely used in Medical predictions, risk evaluation, this Zero-Knowledge Protocol allows several hospitals to unite and build the common best solution without violation of HIPAA.

A key prerequisite is that each client's data must share the same number of features n, ensuring consistent dimensions across all local computations. Nevertheless, the method generalizes naturally to iterative schemes by repeating the transform-and-aggregate step – suggesting extensions to established optimizers like FedAvg, FedSGD, and other gradient-based approaches. Future work could focus on expanding these techniques to non-linear or deep learning models, incorporating more advanced cryptographic primitives, and exploring adaptive scaling strategies for scenarios with higher dimensional feature spaces or heterogeneous data distributions.

By emphasizing simplicity, generality, and minimal overhead, this protocol offers a strong practical option for privacy-preserving linear regression in federated environments, striking a balance between theoretical security and real-world scalability.

References

1. Lindell, Y., Pinkas, B.: Secure Multiparty Computation for Privacy-Preserving Data Mining. Journal of Privacy and Data Management (2009)
2. Dwork, C.: Differential Privacy. In: Automata, Languages and Programming, pp. 1–12. Springer (2006)
3. McMahan, B., Moore, E., Ramage, D., Hampson, S.: Communication-Efficient Learning of Deep Networks from Decentralized Data. In: AISTATS (2017)
4. Konecný, J., McMahan, H.B., Ramage, D., Richtárik, P.: Federated Optimization: Distributed Machine Learning for On-Device Intelligence. arXiv preprint arXiv:1610.02527 (2016)
5. Caldas, S., Wu, P., Li, T., Konecný, J., McMahan, H.B., Smith, V.: LEAF: A Benchmark for Federated Settings. arXiv preprint arXiv:1812.01097 (2018)
6. Zhu, L., Liu, Z., Han, S.: Deep Leakage from Gradients. In: NeurIPS (2019)
7. Wang, H., Gao, W.: Tackling Intertwined Data and Device Heterogeneities in Federated Learning with Unlimited Staleness. arXiv preprint arXiv:2309.13536 (2023)
8. Wei, K., et al.: Gradient Leakage Attack on Federated Learning. In: IEEE Transaction on Parallel and Distributed Systems, 32(8), pp. 2077–2091 (2021)
9. Gentry, C.: Fully Homomorphic Encryption Using Ideal Lattices. In: STOC (2009)
10. Blum, M., Feldman, P., Micali, S.: Non-interactive Zero-Knowledge and its Applications. In: STOC (1988)
11. Li, T., Sahu, A.K., Talwalkar, A., Smith, V.: Federated Learning: Challenges, Methods, and Future Directions. IEEE Signal Processing Magazine, 37(3), pp. 50–60 (2020)
12. Reddi, S.J., Charles, Z., Zaheer, M., Garrett, Z., Rush, K., Konečnỳ, J., McMahan, H.B.: Adaptive Federated Optimization. In: ICLR (2021)
13. Mozaffari, H., Marathe, V.J., Dice, D.: FedPerm: Private and Robust Federated Learning by Parameter Permutation. arXiv preprint arXiv:2208.07922 (2022)