# Collusion-free Cross-feature Logistic Regression

**Anonymous Author(s)**
Affiliation
Address
email

## Abstract

Federated learning is an emerging machine learning paradigm to learn a shared model from multiple independent participants while the data privacy is well preserved. Much recent research is based on the *honest-but-curious* security model, which assumes that there's no collusion among the participants. In this paper, we take collusion into consideration and propose a framework for the cross-feature federated logistic regression on the *semi-honest-but-curious* security model. The proposed framework is secure enough even $N-1$ participants collude with each other. We further provide an implementation of the proposed framework based on the Ec-Elgamal encryption. Experimental results on 5 widely used datasets validate the effectiveness of our proposed framework.

## 1 Introduction

Federated learning (FL), recently proposed in [1], aims to learn a shared model from independent participants, where each participant has different data. The key is to ensure that the data only stays locally when training the federated model, thus data privacy is preserved. Specifically, if each participant holds disjoint feature subsets of the same samples, it is called cross-feature FL [2][1]. The ability allows for privacy preserving in several widely used applications including but not limited to credit assessment, personalized recommendation and medical diagnosis.

Generally, there are three kinds of participants in cross-feature federation: an active party holding a feature subset and the labels, multiple passive parties holding only feature subsets and an authority party may be required to assist in the data exchange and the encryption during the training process. Note the authority party usually has privileges in the federation, e.g. generate the encryption key pairs [6], aggregating gradients [1]. It is difficult to assume that there will be no colluding among all the participants. For instance, the passive parties may collude to acquire the label information from the active party. It worth noting that since the authority party usually has a superior position in the federation, the risk of information leaking is especially high if the colluding participants include the authority party.

In this paper, we present a collusion-free version of cross-feature logistic regression. The intuition behind our method is as follows: on the one hand, we removed the superior authority party in the federation, on the other hand, we further make sure that if a ciphertext is to be decrypted, all participants need to participate in the decryption in order to complete the decryption process. Thus no ciphertext will be leaked even when collusion happens. The main contributions of this paper are listed as follows:

- We have analyzed the robustness to collusion of several popular cross-feature logistic regression methods.

[1] Cross-feature FL is also known as vertical FL in [6].

Submitted to 34th Conference on Neural Information Processing Systems (NeurIPS 2020). Do not distribute.

- We proposed a framework for collusion-free cross-feature logistic regression. The authority party is no longer necessary in the proposed framework. Furthermore, we proved that the method is robust to both the collusion among the active party and partial passive parties and the collusion among the passive parties.
- We showed that the Ec-Elgamal encryption is able to perform exchangeable distributed decryption and described an implementation of the proposed framework based on the Ec-Elgamal encryption.

# 2 Related Works

Much recent research of cross-feature logistic regression has been focused on optimizing the encryption methods or the logistic function approximations. Anti-collusion is merely mentioned. We try to supply the analysis of the collusion robustness of the modern popular cross-feature logistic regression methods in this section.

Recently, Hardy et al. [6] proposed a cross-feature logistic regression method. This method is based on the Taylor expansion of the logistic regression loss function and the additive homomorphic encryption. There are three participants: A, B and C, where A and B hold data and C generates the encryption key pairs. Suppose one of the intermediate variable $y = w \times (x_A + x_B)$ is transmitted to C, in which $x_A$ and $x_B$ are encrypted and hold by A and B separately. When A colludes with C, then A could send the encrypted $x_A$ to C, and C can decrypt and build equations to recover features $x_B$ belonging to B. Our method also utilized the Taylor expansion of the logistic regression loss function, but will not leak confidential data even if the participants collude.

Kim et al. [3] used an homomorphic encryption for approximate arithmetic and developed a least square approximation of the logistic function in privacy preserving logistic regression. The private date at multiple institutions are encrypted and sent to a public server, where the approximated logistic regression is performed totally in the encrypted format. This method enjoys high security, high accuracy and very low transmission expense when no collusion occurs. However, if we take collusion into consideration, all the confidential data may be leaked when one of the participants who holds the private key of encryption collude with the public server.

Mugunthan et al. [4] proposed a mechanism based on secure Multi-Party Computation and Differential Privacy. Aggregating logistic regression weights was taken as an example, where the collusion problem was resolved. Specifically, to achieve the secure Differential Privacy, each party receives its noise from the other parties, where the honest party would have contributed to the noise of the other malicious parties. The core idea of this method is very alike to our method, but we take another approach that instead of generate random noises in a distributed manner, we generate private keys of homomorphic encryption distributively. Additionally, this method aims to solve the problem of securely aggregating locally trained models instead of solving the logistic regression problem.

# 3 Preliminaries

## 3.1 The Security Model

In this paper, we assume that all the participants are *semi-honest-but-curious*:

- All the participants follow the protocol;
- They will infer as much information as they could from the information received;
- They may collude, leaking local and received information to the other participants.

Note this security model is different to the conventional *honest-but-curious* model in that a part of the participants may share the information to infer the confidential of the other participants.

## 3.2 Revisit Cross-feature Logistic Regression via Taylor Expansion

Logistic regression is a classic machine learning algorithm that can be used to distinguish two classes. It can calculate the posterior probability for each sample based on the input. Denote the input features

as $\boldsymbol{x} \in \mathbb{R}^n$, the model parameter as $\boldsymbol{\theta} \in \mathbb{R}^n$ and the data label as $\boldsymbol{y} \in \{-1, 1\}$, the average logistic loss is approximated via the Taylor series expansion as follows [6]:

$$L(\boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^{n} log(1 + e^{-y_i \boldsymbol{\theta}^T x_i}) \approx log2 - \frac{1}{2}\boldsymbol{y}\boldsymbol{\theta}^T\boldsymbol{x} + \frac{1}{8}(\boldsymbol{\theta}^T\boldsymbol{x})^2 \tag{1}$$

Note Equation 1 is a polynomial, making the use of the additive homomorphic encryption available. Based on Equation 1, the gradient is calculated as follows:

$$\nabla L(\boldsymbol{\theta}) = (\frac{1}{4}\boldsymbol{\theta}^T\boldsymbol{x} - \frac{1}{2}\boldsymbol{y})\boldsymbol{x} \tag{2}$$

Specifically, under the cross-feature setting, the calculation of the gradient can be distributed to each participant. Denote $k \in (A, B, C, \dots)$ as the indicator of a participant. $\boldsymbol{x}_k$ is the features held by each participant and $\boldsymbol{\theta}_k$ is the distributed model parameters. Thus the input features $\boldsymbol{x} = (\boldsymbol{x}_A|\boldsymbol{x}_B|\dots)$, the model parameters $\boldsymbol{\theta} = (\boldsymbol{\theta}_A|\boldsymbol{\theta}_B|\dots)$, and the gradients $\nabla L(\boldsymbol{\theta}) = (\nabla L(\boldsymbol{\theta}_A)|\nabla L(\boldsymbol{\theta}_B)|\dots)$. The gradient of loss on each local model parameters is calculated as follows:

$$\nabla L(\boldsymbol{\theta}_k) = (\sum_k \frac{1}{4}\boldsymbol{\theta}_k^T\boldsymbol{x}_k - \frac{1}{2}\boldsymbol{y})\boldsymbol{x}_k \tag{3}$$

In this paper, we also build the method based on the Taylor expansion of the logistic loss as in [6].

### 3.3 The Ec-Elgamal Encryption

Koblitz et al. introduced the analog of Elgamal cryptosystem based on the elliptic curve cryptography [8], that is the Ec-Elgamal encryption. We use the definition of elliptic curve on $F_p$ proposed in [9]. Given the elliptic curve $E(a, b)$ with a base point $P$, an integer $s$ can be randomly generated as the private key, and the product of $s$ and $P$ is the public key, namely $Q$. At encryption, the plaintext $m$ is mapped to the point $M$ on the elliptic curve and random number is chosen to generate a pair of cipher text points $(R, S)$ as depicted in Equation 4. The decryption from $(R, S)$ is shown in Equation 5. It can be seen that the encryption and decryption results are the same point on the elliptic curve.

$$Encryption : R = rP, S = M + rQ \tag{4}$$

$$Decryption : M = S - kR \tag{5}$$

Note in Equation 4, the encryption of the same plaintext point will generate different ciphertext points using random $r$. Therefore, EC-Elgamal is probabilistic.

Furthermore, Ec-Elgamal is also additive homomorphic: Assume there are two plaintext points $M_1$ and $M_2$, with cipher text as $(R_1, S_1) = (r_1P, M_1 + r_1Q)$ and $(R_2, S_2) = (r_2P, M_2 + r_2Q)$. The sum of the ciphertexts:

$$(R_3, S_3) = (R_1 + R_2, S_1 + S_2) = ((r_1 + r_2)P, M_1 + M_2 + (r_1 + r_2)Q) \tag{6}$$

Decrypting $(R_3, S_3)$ as depicted in Equation 5, we get

$$M_3 = M_1 + M_2 + (r_1 + r_2)Q - s(r_1 + r_2)P = M_1 + M_2 \tag{7}$$

Thus, Ec-Elgamal is additive homomorphic.

Table 1: One training iteration of our approach

| | Participant A | Participant B | Participant C |
|---|---|---|---|
| Step1 | Generate the encryption key pair $(Q, (s_A, s_B, s_C))$ | | |
| Step2 | initialize model parameters $\boldsymbol{\theta}_A$ | initialize model parameters $\boldsymbol{\theta}_B$ | initialize model parameters $\boldsymbol{\theta}_C$ |
| Step3 | compute $u_A$, encrypt to $[u_A]_{ABC}$ | compute $u_B$, encrypt to $[u_B]_{ABC}$ | compute $u_C$, encrypt to $[u_C]_{ABC}$ |
| Step4 | Calculate $[w]_{ABC} = \sum_k [u_k]_{ABC}$ and $[\nabla L(\boldsymbol{\theta}_k)]_{ABC} = [w]_{ABC}\boldsymbol{x}_k$ | | |
| Step5 | send $[\nabla L(\boldsymbol{\theta}_A)]_{ABC}$ to B, get $[\nabla L(\boldsymbol{\theta}_C)]_{ABC}$ and partially decrypt it to $[\nabla L(\boldsymbol{\theta}_C)]_{BC}$ | send $[\nabla L(\boldsymbol{\theta}_B)]_{ABC}$ to C, get $[\nabla L(\boldsymbol{\theta}_A)]_{ABC}$ and partially decrypt it to $[\nabla L(\boldsymbol{\theta}_A)]_{AC}$ | send $[\nabla L(\boldsymbol{\theta}_C)]_{ABC}$ to A, get $[\nabla L(\boldsymbol{\theta}_B)]_{ABC}$ and partially decrypt it to $[\nabla L(\boldsymbol{\theta}_B)]_{AB}$ |
| Step6 | send $[\nabla L(\boldsymbol{\theta}_C)]_{BC}$ to B, get $[\nabla L(\boldsymbol{\theta}_B)]_{AB}$ and partially decrypt it to $[\nabla L(\boldsymbol{\theta}_B)]_B$ | send $[\nabla L(\boldsymbol{\theta}_A)]_{AC}$ to C, get $[\nabla L(\boldsymbol{\theta}_C)]_{BC}$ and partially decrypt it to $[\nabla L(\boldsymbol{\theta}_C)]_C$ | send $[\nabla L(\boldsymbol{\theta}_B)]_{AB}$ to A, get $[\nabla L(\boldsymbol{\theta}_A)]_{AC}$ and partially decrypt it to $[\nabla L(\boldsymbol{\theta}_A)]_A$ |
| Step7 | send $[\nabla L(\boldsymbol{\theta}_B)]_B$ to B, get $[\nabla L(\boldsymbol{\theta}_A)]_A$ and finally decrypt it to $\nabla L(\boldsymbol{\theta}_A)$ | send $[\nabla L(\boldsymbol{\theta}_C)]_C$ to C, get $[\nabla L(\boldsymbol{\theta}_B)]_B$ and finally decrypt it to $\nabla L(\boldsymbol{\theta}_B)$ | send $[\nabla L(\boldsymbol{\theta}_A)]_A$ to A, get $[\nabla L(\boldsymbol{\theta}_C)]_C$ and finally decrypt it to $\nabla L(\boldsymbol{\theta}_C)$ |
| Step8 | update $\boldsymbol{\theta}_A$ | update $\boldsymbol{\theta}_B$ | update $\boldsymbol{\theta}_C$ |
| obtain | $\boldsymbol{\theta}_A$ | $\boldsymbol{\theta}_B$ | $\boldsymbol{\theta}_C$ |

# 4 Collusion-Free Cross-Feature Logistic Regression

Anti-collusion has been widely discussed in the secure Multi-Party Computation community. Common solutions to anti-collusion is based on sharing any intermediate results used for the calculation, and all or a majority of the participants are necessary to get the final result. In fact, the whole logistic regression training process or an iteration can be regarded as a Multi-Party Computation protocol. However, exact secret sharing requires heavy inter-participants communication. In this paper, we aim to imitate the anti-collusion solutions via homomorphic encryption in each logistic regression training iteration. The core idea is to ensure that all the intermediate results and the final calculated gradient can only be decrypted if all the participants are involved, and regardless of the order of decryption. We denote this property as *exchangeable distributed decryption*.

We detail the proposed framework based on the exchangeable distributed decryption in Section 4.1 and an implementation based on the Ec-Elgamal encryption in Section 4.2.

## 4.1 The Proposed Framework

An encryption method is said to has the property of *exchangeable distributed decryption* if the private key is distributed to all the participants. Only when all the participants participate in the decryption using its own private key, a confidential is able to be decrypted.

The proposed framework is depicted in Table 1. Without losing generality, we assume that there are three participants, where participant A holds both a subset of features $\boldsymbol{x}_k$ and the labels $\boldsymbol{y}$, while participant B and C only hold subsets of features $\boldsymbol{x}_k$. This framework can be easily extended to $N$ ($N \geq 2$) participants. Note the confidential information includes but not limited to the input features, the groundtruth labels and the model parameters. The framework is decentralized, i.e. we do not need an authority party to assist the training in this framework. Detailed steps are shown as following:

- Step1: Generate the encryption key pair $(Q, (s_A, s_B, s_C))$, where $Q$ is the public key, $k_i, i \in (A, B, C)$ is the private keys of each participant.
- Step2: All participants initialize the model parameters $\boldsymbol{\theta}_k$.
- Step3: Each party calculates $u_k = \frac{1}{4}\boldsymbol{\theta}_k^T \boldsymbol{x}_k - \alpha \boldsymbol{y}$, where $\alpha$ is 0 except on Participant A it is $\frac{1}{2}$. Then $u_k$ is encrypted by $Q$ to $[u_k]_{ABC}$.
- Step4: Each participant get $[w]_{ABC} = \sum_k [u_k]_{ABC}$ and calculate gradient $[\nabla L(\boldsymbol{\theta}_k)]_{ABC} = [w]_{ABC} x_k$
- Step5-7: Steps 5-7 aim to decrypt the gradients. At each step, $[\nabla L(\boldsymbol{\theta}_k)]$ will pass to the right neighbor and performs the partial decryption by it. After looping $N$ times, each participant $k$ will get partially decrypted gradient $[\nabla L(\boldsymbol{\theta}_k)]_k$ and can decrypt it to the final plaintext by its own private key $s_k$.
- Step8: All the participants use the plaintext gradient to update their model parameters.

The predicting procedure is similar to the training procedure. Each participant calculates $\boldsymbol{\theta}_k^T \boldsymbol{x}_k$ and encrypt it by $Q$ to $[\boldsymbol{\theta}_k^T \boldsymbol{x}_k]_{ABC}$. Then $\sum_k [\boldsymbol{\theta}_k^T \boldsymbol{x}_k]_{ABC}$ is calculated. Assume Participant A needs the prediction result, the encrypted sum go through all the participants except A. Thus finally, A gets $\sum_k [\boldsymbol{\theta}_k^T \boldsymbol{x}_k]_A$ and decrypt it to calculate the prediction result.

### 4.2 Implementation via Ec-Elgamal

The key to implement the framework is to find a homomorphic encryption method that can perform the exchangeable distributed decryption. In this section, we provide a simple implementation based on the composition of the Ec-Elgamal encryption as follows:

- Step1: Each participant generates a pair of local public and private keys $(Q_k, s_k)$ via conventional Ec-Elgamal as described in Section 3.3.
- Step2: Calculate the public key $Q = \sum_k Q_k$

These two steps build $(Q, (s_A, s_B, \dots))$, where $Q$ is publicly available. At meanwhile, $s_k$ is distributed at each participant and is kept secretly. Suppose plaintext point $M$ is encrypted to $(R, S)$ as Equation 4, then

$$S = M + rQ = M + r \sum_k Q_k \tag{8}$$

All $s_k$ are necessary in decryption, and the order of applying $s_k$ does not affect the decryption:

$$M = M + r \sum_k Q_k - r \sum_k s_k P = S - rP \sum_k s_k \tag{9}$$

where $P$ is the base point and $Q_k = s_k P$. Thus the composition of Ec-Elgamal is able to accomplish exchangeable distributed decryption. Furthermore, obviously, encryption using $Q$ is also additive homomorphic.

### 4.3 Transmission and Security Analysis

Assume there are $N$ participants. As shown in Table 1, in one training iteration, the inter-participant transmission occurs in 3 phases: 1. the generation of the encryption keys, 2. the inter-participant sum to calculate the encrypted intermediate variables $[u_k]_{ABC}$ and 3. the distributed decryption of the encrypted gradients. Specifically, the information transmitted in phase 1 is the local public keys, which are not confidential. At phase 2, the transmitted information is confidential but encrypted by the public key $Q$. Since the decryption needs all the participants, so the confidential will not be leaked even when $N - 1$ participants collude. Phase 3 includes $N$ rounds of transmissions to decrypt the gradients. Note in the proposed framework, each participant $k$ will never send out $[\nabla L(\boldsymbol{\theta}_k)]$ that has been decrypted by its own private key $s_k$. Therefore, $\nabla L(\boldsymbol{\theta}_k)$ will never be leaked even when the other $N - 1$ participants collude.

Table 2: A brief description of datasets

| Dataset | Rows | Cols | Is Binary |
|---|---|---|---|
| breast cancer | 569 | 30 | Yes |
| digits | 1797 | 64 | No |
| credit | 30000 | 24 | Yes |
| MNIST | 70000 | 784 | No |
| covertype | 581012 | 54 | No |

Table 3: Numerical results on public datasets

| | | | ACC | | AUC | |
|---|---|---|---|---|---|---|
| Dataset | Time | Batch Size | Our LR | Original LR | Our LR | Original LR |
| breast cancer | 39m | 128 | 0.93 | 0.98 | 0.98 | 0.99 |
| digits | 3h35m | 128 | 0.83 | 0.92 | 0.91 | 0.97 |
| credit | 49m | 4096 | 0.79 | 0.77 | 0.71 | 0.73 |
| MNIST | 32h42m | 4096 | 0.85 | 0.9 | 0.93 | 0.96 |
| covertype | 29h30m | 4096 | 0.71 | 0.74 | 0.78 | 0.81 |

# 5 Experiments

## 5.1 Settings

**Benchmark datasets**: We evaluate the collusion-free cross-feature logistic regression on 5 widely used datasets: breast cancer, digits, credit, MNIST and covertype. Table 2 describes the overall characteristics of these datasets, including their size and whether the labels are binarized. For datasets with multi-category labels, we converted the labels into binary classification via odd vs. even. All data features are standardized. To simulate the cross-feature scenario, we divided the features of the data to 5 participants approximately evenly, and then ran the program on 5 independent servers. For the MNIST dataset, there are already separated train sets and test sets. For the rest datasets, we randomly split 20% from the total samples for testing to evaluate the performance.

**Evaluation metrics**: We utilize the classification accuracy (ACC) and area under the ROC curve (AUC) to evaluate the proposed method.

**Hardware settings**: We use 5 servers to play 5 participants in all the experiments. Each server is equipped with a Intel(R) Xeon(R) Gold 5218 CPU @ 2.30GHz and 125 GB memory.

## 5.2 Results

In the experiment, since we are not interested in the converging time, all the number of epochs are fixed to 2 and the learning rate to 0.01. For the breast cancer and the digits datasets, we set the mini-batch size to 128. For the credit, the MNIST and the covertype datasets, the batch size is set to 4096. The proposed method is compared with the original logistic regression implemented by Sklearn [10].

Table 3 shows the experimental results of the both the proposed method and the original logistic regression, including the training time and the performances on each dataset. As can be seen that the training time of proposed method is linearly and positively correlated with the size of the data set and the number of features owned by each participant. The prediction accuracy is reasonable even after only 2 epochs. The performance is worse than the original LR, which is within the expectation, because of the Taylor expansion of the logistic loss function.

# 6 Conclusions

In this paper, we have proposed a framework for collusion-free cross-feature logistic regression and provided an implementation based on the composition of the Ec-Elgamal encryption. The privacy can be protected during both training and prediction even when $N-1$ participants collude under the

*semi-honest-but-curious* security model. Experiments show that the proposed method have achieved reasonable performance on 5 classification datasets and the training time grows linearly with the size the input dataset.

# References


[1] McMahan, H. B., Moore, E., Ramage, D., & y Arcas, B. A. (2016). Federated learning of deep networks using model averaging. CoRR abs/1602.05629 (2016). arXiv preprint arXiv:1602.05629.

[2] Li, H., Meng, D., & Li, X. (2020). Knowledge Federation: Hierarchy and Unification. arXiv preprint arXiv:2002.01647.

[3] Kim, M., Song, Y., Wang, S., Xia, Y., & Jiang, X. (2018). Secure logistic regression based on homomorphic encryption: Design and evaluation. JMIR medical informatics, 6(2), e19.

[4] Mugunthan, V., Polychroniadou, A., Byrd, D., & Balch, T. H. (2019) SMPAI: Secure Multi-Party Computation for Federated Learning.

[5] Xu, R., Baracaldo, N., Zhou, Y., Anwar, A., & Ludwig, H. (2019, November). Hybridalpha: An efficient approach for privacy-preserving federated learning. In Proceedings of the 12th ACM Workshop on Artificial Intelligence and Security (pp. 13-23).

[6] Hardy, S., Henecka, W., Ivey-Law, H., Nock, R., Patrini, G., Smith, G., & Thorne, B. (2017). Private federated learning on vertically partitioned data via entity resolution and additively homomorphic encryption. arXiv preprint arXiv:1711.10677.

[7] Aono, Y., Hayashi, T., Trieu Phong, L., & Wang, L. (2016). Scalable and secure logistic regression via homomorphic encryption. In Proceedings of the Sixth ACM Conference on Data and Application Security and Privacy (pp. 142-144).

[8] Koblitz, N. (1994). A course in number theory and cryptography (Vol. 114). Springer Science & Business Media.

[9] Koblitz, N., Menezes, A., & Vanstone, S. (2000). The state of elliptic curve cryptography. Designs, codes and cryptography, 19(2-3), 173-193.

[10] Pedregosa, Fabian and Varoquaux, Gaël and Gramfort, Alexandre and Michel, Vincent and Thirion, Bertrand and Grisel, Olivier and Blondel, Mathieu and Prettenhofer, Peter and Weiss, Ron and Dubourg, & Vincent and others. (2011). Journal of machine learning research, 12(Oct), (pp. 2825-2830).