Privacy-Preserving Ridge Regression with only Linearly-Homomorphic Encryption

Irene Giacomelli
ISI Foundation

Joint work with Somesh Jha (UW-Madison), Marc Joye (NXP Semiconductors), C. David Page (UW-Madison) and Kyonghwan Yoon (UW-Madison)

3rd June 2018

Center for Predictive Computational Phenotyping

Machine Learning

Applications:

- recommendation systems
- antispam software
- ....
- bioinformatics & medicine

  e.g. -genomics

  -personalized medicine (pharmacogenetic)

  -adverse drug event detection,

  -disease/disorder prevention

Applications:

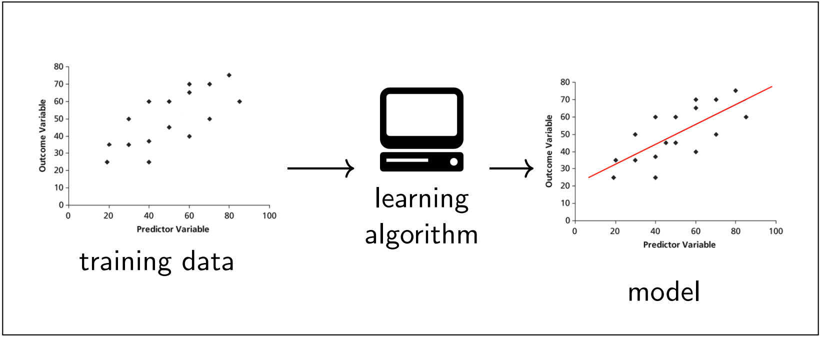
- recommendation systems
- antispam software
- ....
- bioinformatics & medicine

  e.g. -genomics
  -personalized medicine (pharmacogenetic)
  -adverse drug event detection,
  -disease/disorder prevention

Privacy		**Information Sharing**
confidential data	VS	improved results
proprietary models		larger usability

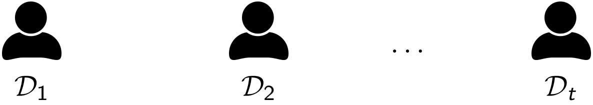
→ **Privacy-Preserving Machine Learning**

Detection of a pattern (model) in data via a learning algorithm



The efficacy of the learned model is improved by training on larger number of more diverse data

Privacy-Preserving Training



Training data = merge of private data silos

Goal: Train a model on $\mathcal{D} = \mathcal{D}_1 \cup \mathcal{D}_2 \cup \cdots \cup \mathcal{D}_t$,
while keeping each $\mathcal{D}_i$ secret!

Same as in MPC: run a function (training algorithm) on private inputs $(\mathcal{D}_1, \dots, \mathcal{D}_t)$,
revealing no extra info beside what is leaked from the function output (model)

Privacy-Preserving (PP) Training

In 2000,

- "PP Data Mining" Lindell and Pinkas, *CRYPTO 2000*
  (ID3 algorithm for learning a tree on the merge of 2 silos)

[1] Cappe, I love u

Privacy-Preserving (PP) Training

In 2000,

- "PP Data Mining" Lindell and Pinkas, *CRYPTO 2000*
  (ID3 algorithm for learning a tree on the merge of 2 silos)

after that, a large number of works propose[1] privacy-preserving training systems for different ML models in diverse settings.
E.g., **ridge regression**:

- "PP Ridge Regression on Hundreds of Millions of Records"
  Nikolaenko et al, *S&P 2013*
- "PP distributed Linear Regression on High-Dimensional Data"
  Gascón et al, *PoPETS 2017*
- "SecureML" Mohassel and Zhang, *S&P 2017*

[1] Cappe, I love u

Privacy-Preserving Ridge Regression with only Linearly-Homomorphic Encryption

Irene Giacomelli[1], Somesh Jha[1], Marc Joye[2], C. David Page[1], and Kyonghwan Yoon[1]

[1] University of Wisconsin-Madison, Madison, WI, USA
[2] NXP Semiconductors, San Jose, CA, USA

Abstract. Linear regression with 2-norm regularization (*i.e.*, ridge regression) is an important statistical technique that models the relationship between some explanatory values and an outcome value using a linear function. In many applications (*e.g.*, predictive modeling in personalized health-care), these values represent sensitive data owned by several different parties who are unwilling to share them. In this setting, training a linear regression model becomes challenging and needs specific cryptographic solutions. This problem was elegantly addressed by Nikolaenko *et al.* in S&P (Oakland) 2013. They suggested a two-server system that uses linearly-homomorphic encryption (LHE) and Yao's two-party protocol (garbled circuits). In this work, we propose a novel system that can train a ridge linear regression model using only LHE (*i.e.*, without using Yao's protocol). This greatly improves the overall performance (both in computation and communication) as Yao's protocol was the main bottleneck in the previous solution. The efficiency of the proposed system is validated both on synthetically-generated and real-world datasets.

Keywords: Ridge regression; linear regression; privacy; homomorphic encryption.

to appear at ACNS 2018 (ePrint Report 2017/979)

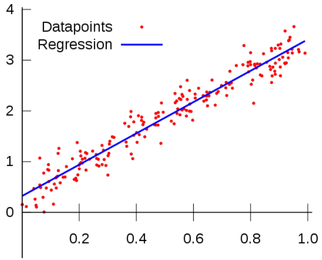
Ridge Regression

Data point: $(\mathbf{x}, y)$, $\mathbf{x} \in \mathbb{R}^d$ and $y \in \mathbb{R}$

Model: $\mathbf{w} \in \mathbb{R}^d$ vector of weights

Scoring: $y \approx f_{\mathbf{w}}(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle$

$$= \sum_{j=1}^{d} \mathbf{w}(j)\mathbf{x}(j)$$



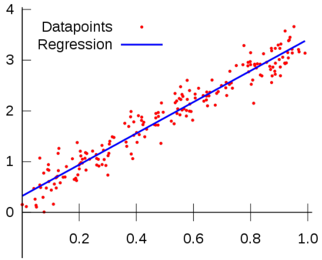
Ridge Regression

Data point: $(\mathbf{x}, y)$, $\mathbf{x} \in \mathbb{R}^d$ and $y \in \mathbb{R}$

Model: $\mathbf{w} \in \mathbb{R}^d$ vector of weights

Scoring: $y \approx f_{\mathbf{w}}(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle$

$$= \sum_{j=1}^{d} \mathbf{w}(j)\mathbf{x}(j)$$



Example: warfarin maintenance dose

$\mathbf{x}$=(VKORC1 and CYP2C9 genotypes, age, bodyweight, ...)
y = dose

Ridge Regression

Training: given $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1,\ldots,n}$

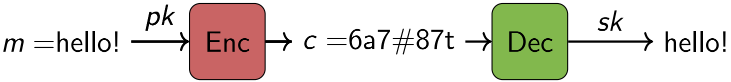
$$\text{find argmin of } F(\mathbf{w}) = \underbrace{\sum_{i=1}^{n}(y_i - \langle \mathbf{w}, \mathbf{x}_i \rangle)^2}_{\text{mean squared error}} + \lambda \underbrace{||\mathbf{w}||_2^2}_{\text{regularization}}$$

This can be done in two steps:

- Step 1: Compute the matrix $A = \sum_{i=1}^{n} \mathbf{x}_i^\top \mathbf{x}_i + \lambda I$ and the vector $\mathbf{b} = \sum_{i=1}^{n} y_i \, \mathbf{x}_i$
- Step 2: Solve the linear system $A \cdot \mathbf{w} = \mathbf{b}$

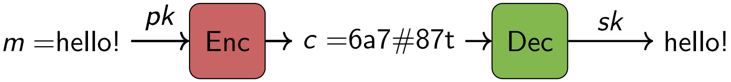
Linearly-Homomorphic Encryption (LHE)

- Key Generation: $(sk, pk) \leftarrow \text{Gen}(\kappa)$
- Encryption: $\mathbf{c} \leftarrow \text{Enc}_{pk}(\mathbf{m})$
- Decryption: $\mathbf{m} = \text{Dec}_{sk}(\mathbf{c})$



Linearly-Homomorphic Encryption (LHE)

- Key Generation: $(sk, pk) \leftarrow \mathsf{Gen}(\kappa)$
- Encryption: $\mathbf{c} \leftarrow \mathsf{Enc}_{pk}(\mathbf{m})$
- Decryption: $\mathbf{m} = \mathsf{Dec}_{sk}(\mathbf{c})$



- Addition of ciphertexts:
  $\mathsf{Enc}_{pk}(\mathbf{m}_1) \boxplus \mathsf{Enc}_{pk}(\mathbf{m}_2) = \mathsf{Enc}_{pk}(\mathbf{m}_1 + \mathbf{m}_2)$
- Multiplication of a ciphertext by a plaintext: ($\mathbf{m}_1$ is public!)
  $\mathbf{m}_1 \boxtimes \mathsf{Enc}_{pk}(\mathbf{m}_2) = \mathsf{Enc}_{pk}(\mathbf{m}_1 \times \mathbf{m}_2)$

2-Server Model

Two non-colluding servers:

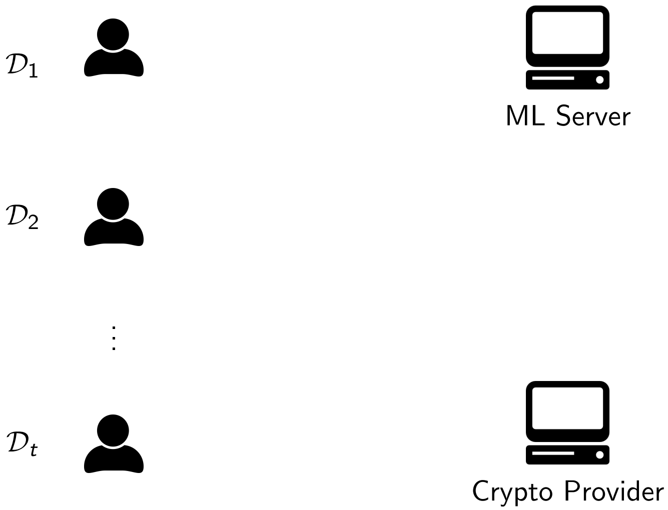
Crypto Provider

- NOT trusted to handle data
- trusted to follow the protocol
- trusted to generate keys and store *sk*

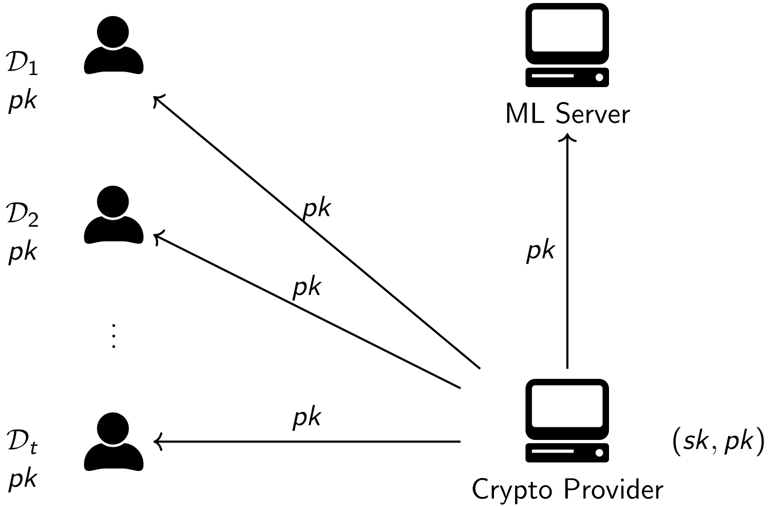
ML Server

- NOT trusted to handle data
- trusted to follow the protocol

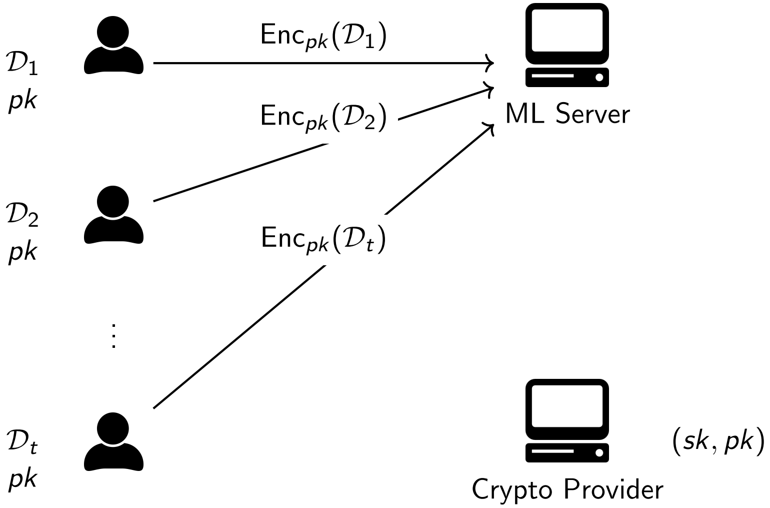
System Overview



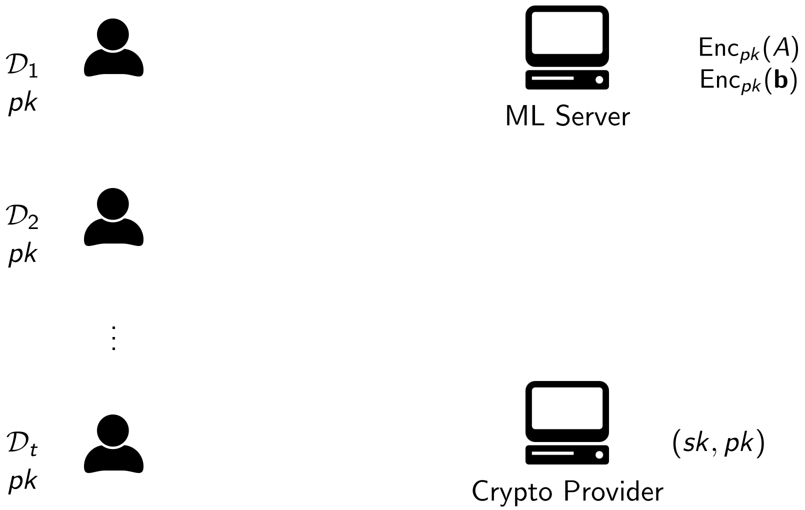
System Overview



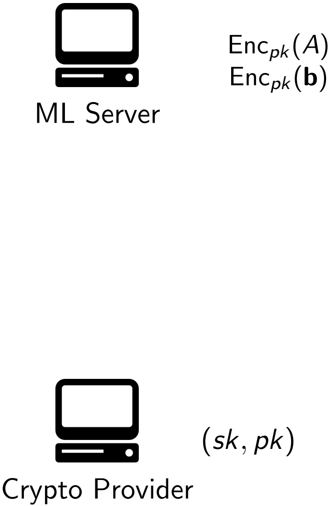
System Overview



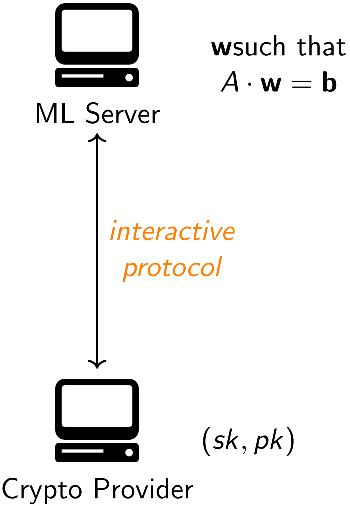
System Overview



System Overview



System Overview



Phase 1: merging the local data silos

Input: User i with data $\mathcal{D}_i$ $(i = 1, 2, \dots)$

Output: $\mathrm{Enc}_{pk}(A)$, $\mathrm{Enc}_{pk}(\mathbf{b})$ for the ML Server

$$A = \sum_{i=1}^{n} \mathbf{x}_i^\top \mathbf{x}_i + \lambda I$$

$$\mathbf{b} = \sum_{i=1}^{n} y_i \, \mathbf{x}_i$$

$$\begin{pmatrix} - & \mathbf{x}_1 & - & y_1 \\ - & \mathbf{x}_2 & - & y_2 \\ - & \mathbf{x}_3 & - & y_3 \\ \vdots & \vdots & & \vdots \\ - & \mathbf{x}_n & - & y_n \end{pmatrix}$$

Phase 1: merging the local data silos

Goal: compute the encryption of $A = \sum_{i=1}^{n} \mathbf{x}_i^\top \mathbf{x}_i + \lambda I$ and $\mathbf{b} = \sum_{i=1}^{n} y_i \, \mathbf{x}_i$

It depends on the distributed setting:

$$\begin{pmatrix} - & \mathbf{x}_1 & - & y_1 \\ - & \mathbf{x}_2 & - & y_2 \\ - & \mathbf{x}_3 & - & y_3 \\ \vdots & \vdots & & \vdots \\ - & \mathbf{x}_n & - & y_n \end{pmatrix}$$

Phase 1: merging the local data silos

Goal: compute the encryption of $A = \sum_{i=1}^{n} \mathbf{x}_i^\top \mathbf{x}_i + \lambda I$ and $\mathbf{b} = \sum_{i=1}^{n} y_i \, \mathbf{x}_i$

It depends on the distributed setting:

- horizontally-partitioned datasets

$$\boxplus_{i=1}^{n} \mathrm{Enc}_{pk}(\mathbf{x}_i^\top \mathbf{x}_i)$$

$$\begin{pmatrix} - & \mathbf{x}_1 & - & y_1 \\ - & \mathbf{x}_2 & - & y_2 \\ - & \mathbf{x}_3 & - & y_3 \\ \vdots & \vdots & & \vdots \\ - & \mathbf{x}_n & - & y_n \end{pmatrix}$$

Phase 1: merging the local data silos

Goal: compute the encryption of $A = \sum_{i=1}^{n} \mathbf{x}_i^\top \mathbf{x}_i + \lambda I$ and $\mathbf{b} = \sum_{i=1}^{n} y_i \, \mathbf{x}_i$

It depends on the distributed setting:

- horizontally-partitioned datasets

$$\boxplus_{i=1}^{n} \mathsf{Enc}_{pk}(\mathbf{x}_i^\top \mathbf{x}_i)$$

- arbitrarily-partitioned datasets

$$\mathsf{Enc}_{pk}(\mathbf{x}_i(j)) \cdot \mathsf{Enc}_{pk}(\mathbf{x}_k(h))$$

(1 multiplication done via Labeled Encryption, Barbosa et al. ESORICS 2017)

$$\begin{pmatrix} - & \mathbf{x}_1 & - & y_1 \\ - & \mathbf{x}_2 & - & y_2 \\ - & \mathbf{x}_3 & - & y_3 \\ \vdots & \vdots & & \vdots \\ - & \mathbf{x}_n & - & y_n \end{pmatrix}$$

Phase 2: solving $A \cdot \mathbf{w} = \mathbf{b}$

ML Server: $\text{Enc}_{pk}(A), \text{Enc}_{pk}(\mathbf{b})$
Crypto Provider: sk

Interactive protocol:

Phase 2: solving $A \cdot \mathbf{w} = \mathbf{b}$

ML Server: $\text{Enc}_{pk}(A), \text{Enc}_{pk}(\mathbf{b})$
Crypto Provider: sk

Interactive protocol:

1. ML Server "masks inside the encryption"
   $\text{Enc}_{pk}(A) \rightarrow \text{Enc}_{pk}(A \cdot R)$
   $\text{Enc}_{pk}(\mathbf{b}) \rightarrow \text{Enc}_{pk}(\mathbf{b} + A \cdot \mathbf{r})$

Phase 2: solving $A \cdot \mathbf{w} = \mathbf{b}$

ML Server: $\text{Enc}_{pk}(A), \text{Enc}_{pk}(\mathbf{b})$
Crypto Provider: sk

Interactive protocol:

1. ML Server "masks inside the encryption"
   $\text{Enc}_{pk}(A) \rightarrow \text{Enc}_{pk}(A \cdot R)$
   $\text{Enc}_{pk}(\mathbf{b}) \rightarrow \text{Enc}_{pk}(\mathbf{b} + A \cdot \mathbf{r})$
2. Crypto Provider decrypts, gets $\tilde{A} = A \cdot R$, $\tilde{\mathbf{b}} = \mathbf{b} + A \cdot \mathbf{r}$ and computes a "masked model", $\tilde{\mathbf{w}} = \tilde{A}^{-1}\tilde{\mathbf{b}}$

Phase 2: solving $A \cdot \mathbf{w} = \mathbf{b}$

ML Server: $\text{Enc}_{pk}(A), \text{Enc}_{pk}(\mathbf{b})$
Crypto Provider: sk

Interactive protocol:

1. ML Server "masks inside the encryption"
   $\text{Enc}_{pk}(A) \to \text{Enc}_{pk}(A \cdot R)$
   $\text{Enc}_{pk}(\mathbf{b}) \to \text{Enc}_{pk}(\mathbf{b} + A \cdot \mathbf{r})$
2. Crypto Provider decrypts, gets $\tilde{A} = A \cdot R$, $\tilde{\mathbf{b}} = \mathbf{b} + A \cdot \mathbf{r}$ and computes a "masked model", $\tilde{\mathbf{w}} = \tilde{A}^{-1}\tilde{\mathbf{b}}$
3. ML Server computes the real model $\mathbf{w}$ from the masked one

$$\mathbf{w} = R \cdot \tilde{\mathbf{w}} - \mathbf{r}$$

Efficiency: communication

n data points, d features

- Phase 1
  - horizontally-partitioned data: $O(d^3 \log(nd))$ bits
  - vertically-partitioned data: $O((nd^2 + d^3) \log(nd))$ bits
- Phase 2: $O(d^3 \log(nd))$ bits

Efficiency: communication

n data points, d features

- Phase 1
  - horizontally-partitioned data: $O(d^3 \log(nd))$ bits
  - vertically-partitioned data: $O((nd^2 + d^3) \log(nd))$ bits
- Phase 2: $O(d^3 \log(nd))$ bits

horizontally-partitioned, $d = 20$

- Our (phase 1+2) $\rightarrow$ 1.3 MB ($n = 10$ millions)

- Nikoleanko et al $\rightarrow$ $>$ 270 MB (garbled circuit)

vertically-partitioned, $d = 100$

- Our (phase 1+2) $\rightarrow$ 1.3 GB ($n = 5$ thousands)

- Gascón et al $\rightarrow$ $>$ 3 GB (garbled circuit)

Efficiency: communication

n data points, d features

- Phase 1
  - horizontally-partitioned data: $O(d^3 \log(nd))$ bits
  - vertically-partitioned data: $O((nd^2 + d^3)\log(nd))$ bits
- Phase 2: $O(d^3 \log(nd))$ bits

horizontally-partitioned, $d = 20$

- Our (phase 1+2) $\rightarrow$ 1.3 MB ($n = 10$ millions)

- Nikoleanko et al $\rightarrow$ $>$ 270 MB (garbled circuit)

vertically-partitioned, $d = 100$

- Our (phase 1+2) $\rightarrow$ 1.3 GB ($n = 5$ thousands)

- Gascón et al $\rightarrow$ $>$ 3 GB (garbled circuit)

SecureML (with LHE pre-processing): $O(nd + n)$
if $n = \Theta(d^{2.5})$, then "$nd + d > d^3 \log(nd)$"

Efficiency: running time

Results for seven UCI datasets (time in seconds):

Dataset	n	d	ℓ	$\log_2(N)$	R_{MSE}	Phase 1		Phase 2	
						Time	kB	Time	kB
air	6252	13	1	2048	4.15E-09	1.99	53.24	3.65	96.51
beijing	37582	14	2	2048	5.29E-07	2.37	60.93	4.26	110.10
boston	456	13	4	2048	2.34E-06	2.00	53.24	3.76	96.51
energy	17762	25	3	2724	5.63E-07	12.99	238.26	37.73	451
forest	466	12	3	2048	3.57E-09	1.66	46.08	2.81	82.94
student	356	30	1	2048	4.63E-07	9.36	253.44	30.40	483.84
wine	4409	11	4	2048	2.62E-05	1.71	39.42	2.38	70.40

LHE: Paillier's scheme with $\geq$ 100-bit security

Conclusions

We described a new system to train a ridge regression model on the merge of encrypted datasets held by mutually distrustful parties. The system is designed in the **2-server** model and is the first one based only on **LHE**.

Conclusions

We described a new system to train a ridge regression model on the merge of encrypted datasets held by mutually distrustful parties. The system is designed in the **2-server** model and is the first one based only on **LHE**.

Next

Conclusions

We described a new system to train a ridge regression model on the merge of encrypted datasets held by mutually distrustful parties. The system is designed in the **2-server** model and is the first one based only on **LHE**.

Next

- modifying the masking to improve efficiency

Conclusions

We described a new system to train a ridge regression model on the merge of encrypted datasets held by mutually distrustful parties. The system is designed in the **2-server** model and is the first one based only on **LHE**.

Next

- modifying the masking to improve efficiency
- extension to non-differentiable regularization terms

Conclusions

We described a new system to train a ridge regression model on the merge of encrypted datasets held by mutually distrustful parties. The system is designed in the **2-server** model and is the first one based only on **LHE**.

Next

- modifying the masking to improve efficiency
- extension to non-differentiable regularization terms
- active security

Thanks for your attention!