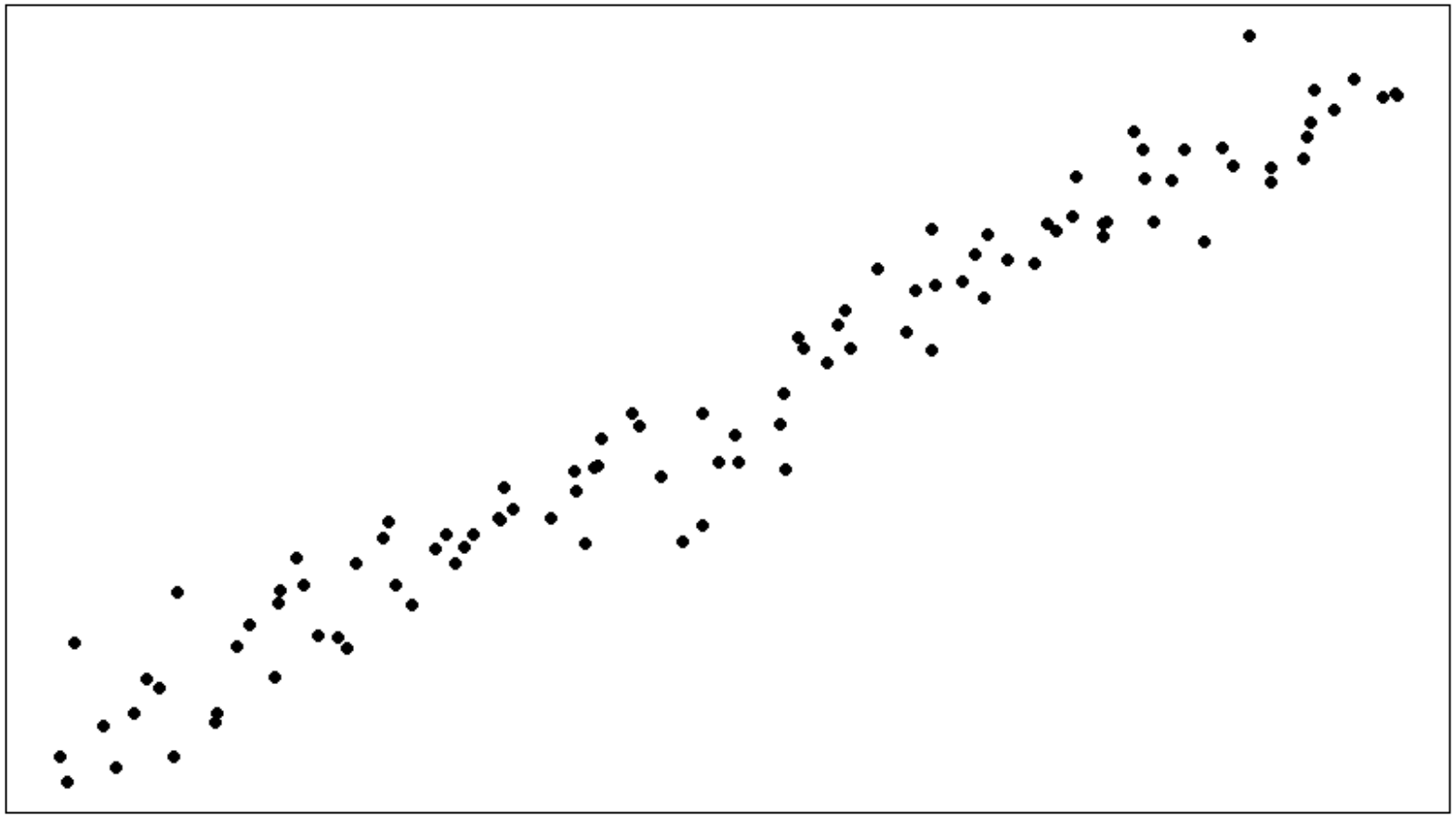
Efficient Secure Ridge Regression from Randomized Gaussian Elimination

Frank Blom, Niek J. Bouman,
Berry Schoenmakers, Niels de Vreede

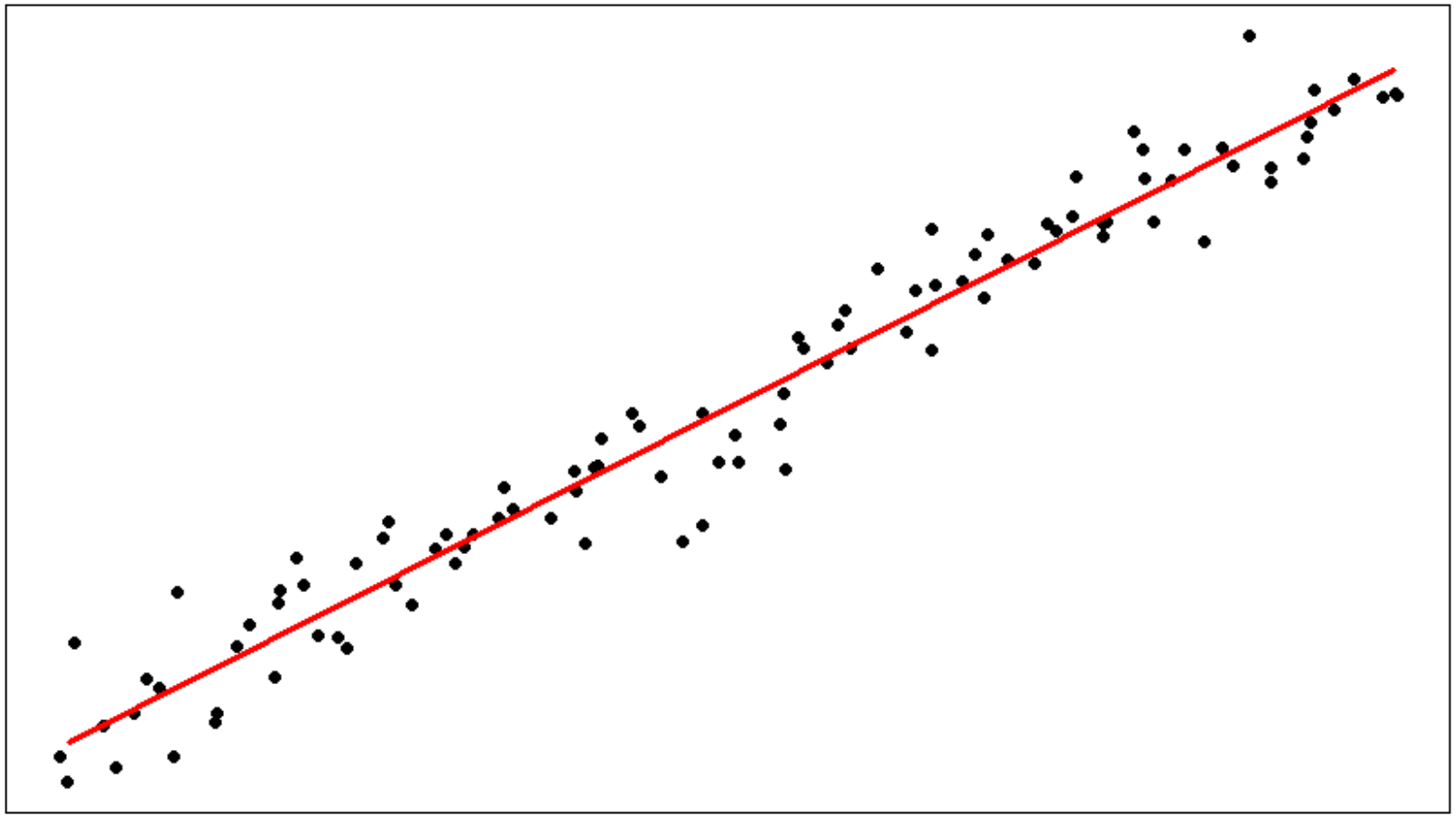
TU/e EINDHOVEN UNIVERSITY OF TECHNOLOGY

Funded by EU H2020 SODA

Linear regression



Linear regression



Multi-linear regression

UCI Machine Learning Repository
Center for Machine Learning and Intelligent Systems

residual sugar	chlorides	pH	alcohol	wine quality
1.50	0.091	3.25	9.6	7
3.50	0.029	3.53	11.7	3
12.30	0.044	3.11	9.9	5
10.60	0.035	3.20	10.4	9

Data Set Characteristics:	Multivariate	**Number of Instances:**	4898	**Area:**	Business
Attribute Characteristics:	Real	**Number of Attributes:**	12	**Date Donated**	2009-10-07
Associated Tasks:	Classification, Regression	**Missing Values?**	N/A	**Number of Web Hits:**	943674

Multi-linear regression

residual sugar	chlorides	pH	alcohol	wine quality
1.50	0.091	3.25	9.6	7
3.50	0.029	3.53	11.7	3
12.30	0.044	3.11	9.9	5
10.60	0.035	3.20	10.4	9

$X \in \mathbb{R}^{n \times d}$

$y \in \mathbb{R}^{n}$

Multi-linear regression

$X \in \mathbb{R}^{n \times d}, y \in \mathbb{R}^n$ represent an n x d over-determined system:

$$Xw = y \ (n \gg d)$$

Least-squares solution:

$$\arg\min_w \{\|Xw - y\|_2\}$$

Least-squares solution is an unbiased estimator, but might have high variance

Ridge regression: reduce variance (at the cost of some bias)

Ridge regression

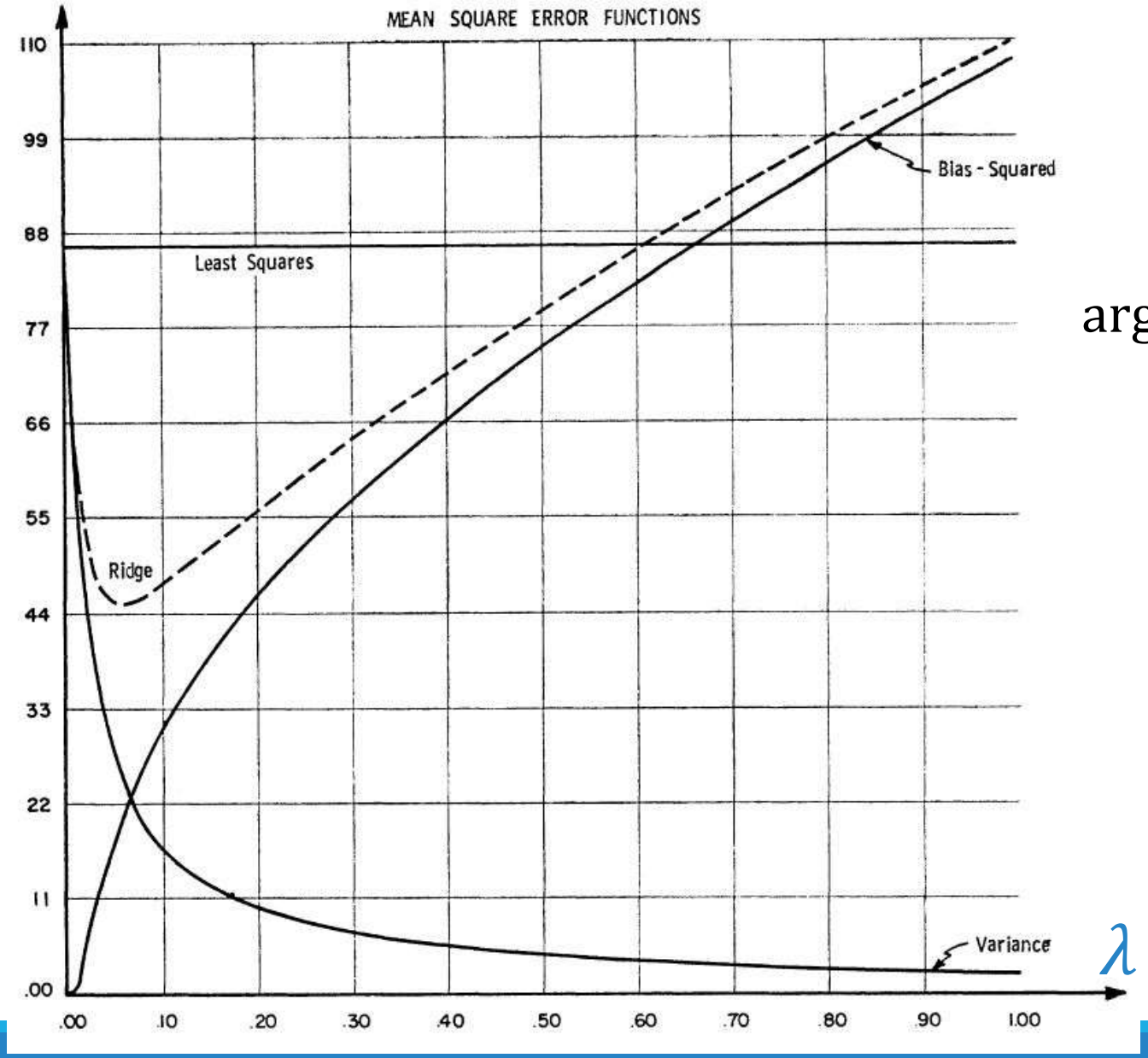
$X \in \mathbb{R}^{n \times d}, y \in \mathbb{R}^n$ represent an n x d over-determined system:

$$Xw = y \ (n \gg d)$$

Regularized least-squares solution:

$$\arg\min_w \{\|Xw - y\|_2 + \lambda \|w\|_2\}_{\lambda > 0}$$

Bias-variance trade-off



$$\arg\min_w \{\|Xw - y\|_2 + \lambda\|w\|_2\}_{\lambda > 0}$$

MSE = Bias^2 + variance + noise

How to solve Ridge Regression?

Iterative method:

- possibly: numerical instability
- stopping condition

Direct method:

$$\arg\min_w \{\|Xw - y\|_2 + \lambda\|w\|_2\}_{\lambda > 0}$$

$$w = (X^T X + \lambda I)^{-1} X^T y$$

$$w = A^{-1} b$$

$$A = X^T X + \lambda I,$$
$$b = X^T y$$

MPC roadmap

- Design choices
- Converting input: real numbers to field elements
- Sketch of main ideas
- Choosing the modulus
- Experimental results

Ridge Regression as a Secure Multiparty Computation

- Arithmetic Black Box over a finite field with “cheap” inner product
- “Cheap” inner product: same round complexity as single secure multiplication
- Example: Shamir’s scheme

Notation: $[\![x]\!]$ represents a sharing of x

$[\![x]\!]_p$ represents a sharing of x with specific prime modulus p

Design choices

- Direct solution (as opposed to iterative solution)
- Avoid rational reconstruction
- No assumption on data partitioning

Horizontal Partitioning

Party i

residual sugar	chlorides	pH	alcohol	wine quality
3.50	0.029	3.53	11.7	3
12.30	0.044	3.11	9.9	5

Design choices

- Direct solution (as opposed to iterative solution)
- Avoid rational reconstruction
- No assumption on data partitioning

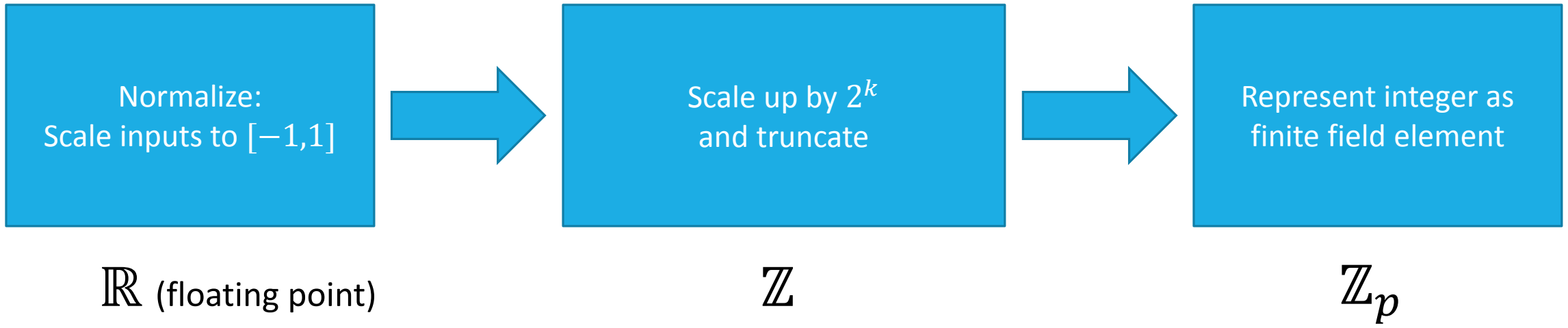
Vertical Partitioning

Party i

chlorides	pH
0.091	3.25
0.029	3.53
0.044	3.11
0.035	3.20

Input scaling

Plaintext precomputation:



Representing the output

$$w = A^{-1}b = \frac{1}{\det(A)} \operatorname{adj}(A)b$$

$$A \in \mathbb{Z}^{d\times d} \implies \operatorname{adj}(A) \in \mathbb{Z}^{d\times d}$$

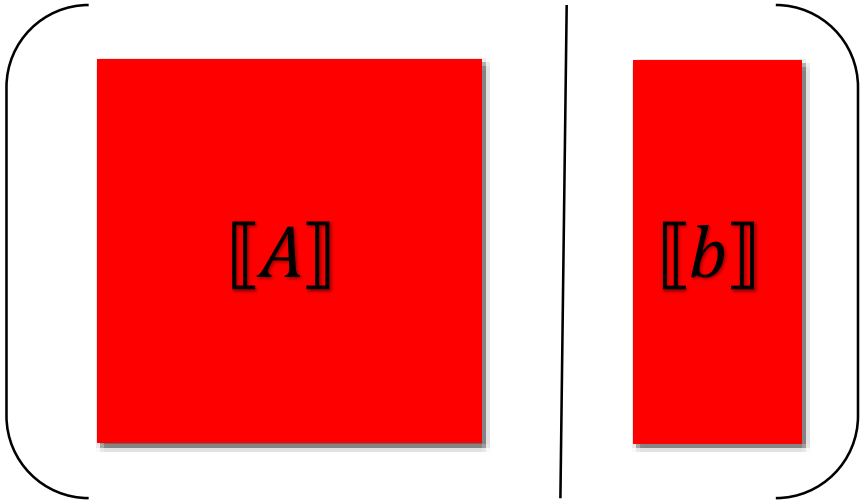
Represent w in two parts

Numerator: $\det(A)\, w\ (= \operatorname{adj}(A)b)$

Denominator: $\det(A)$

Randomized Gaussian Elimination

$A \in GL_d(\mathbb{Z}_p)$

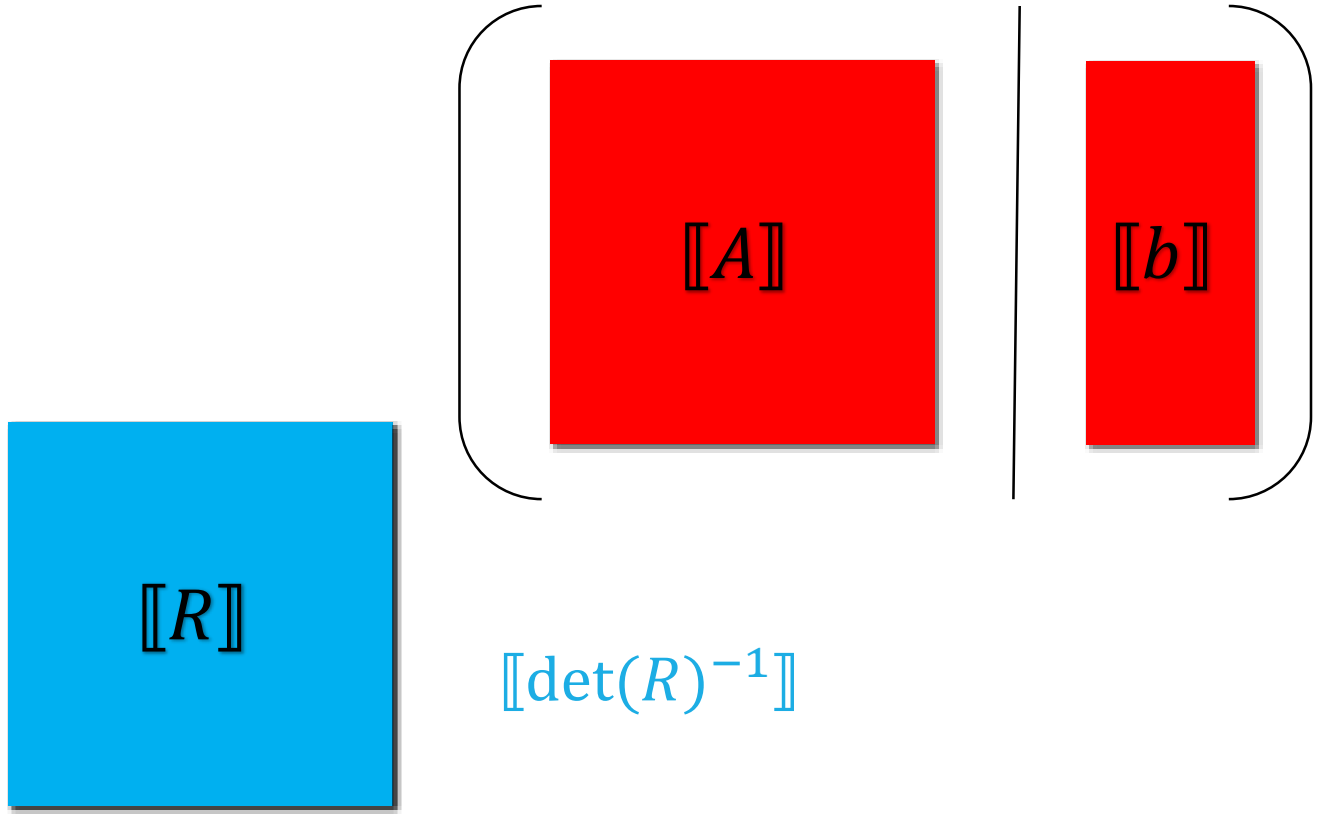


Return:([[adj$(A)b$]], [[det(A)]])

Randomized Gaussian Elimination

$A \in GL_d(\mathbb{Z}_p)$

1. Generate random mask $[\![R]\!] \in_R GL_d(\mathbb{Z}_p)$

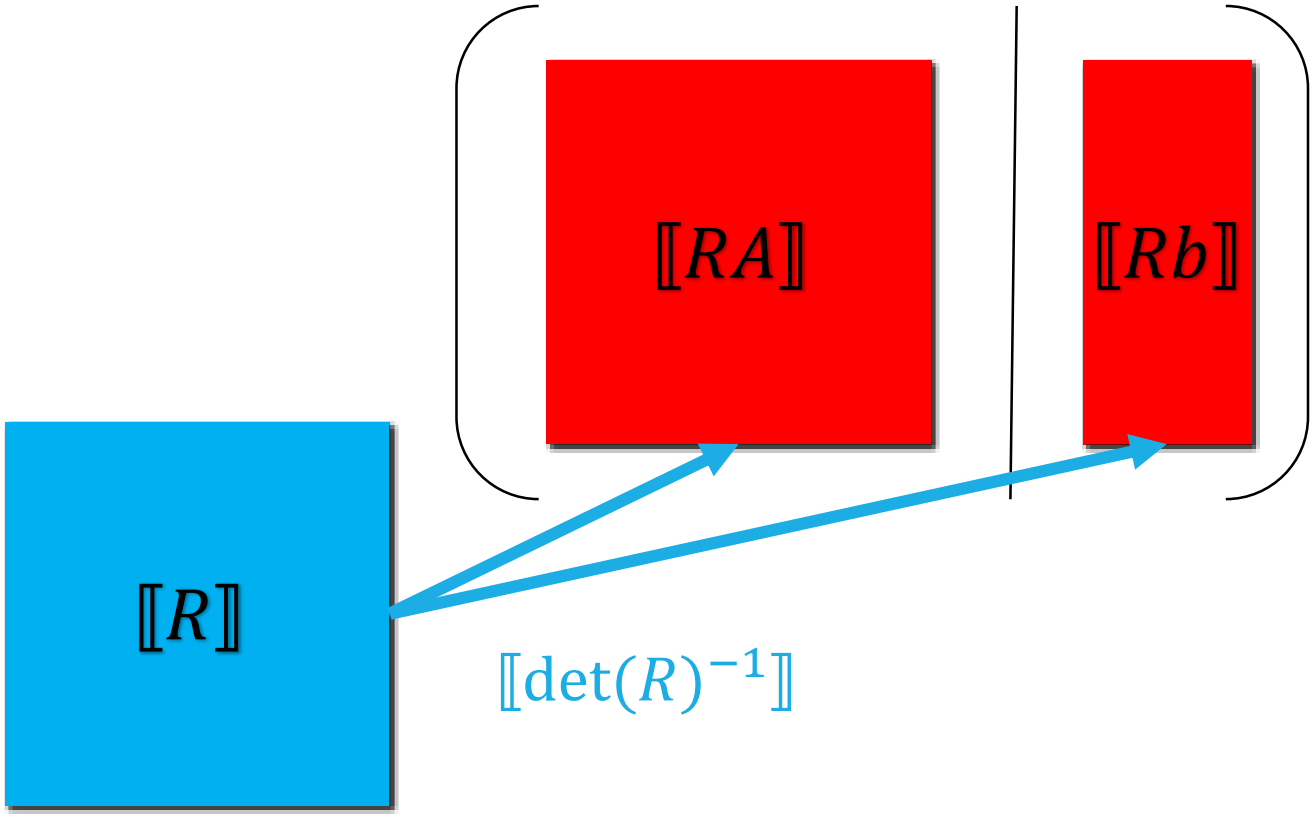


Return:$([\![\text{adj}(A)b]\!], [\![\det(A)]\!])$

Randomized Gaussian Elimination

$A \in GL_d(\mathbb{Z}_p)$

1. Generate random mask $[\![R]\!] \in_R GL_d(\mathbb{Z}_p)$
2. Randomize system $(A|b)$

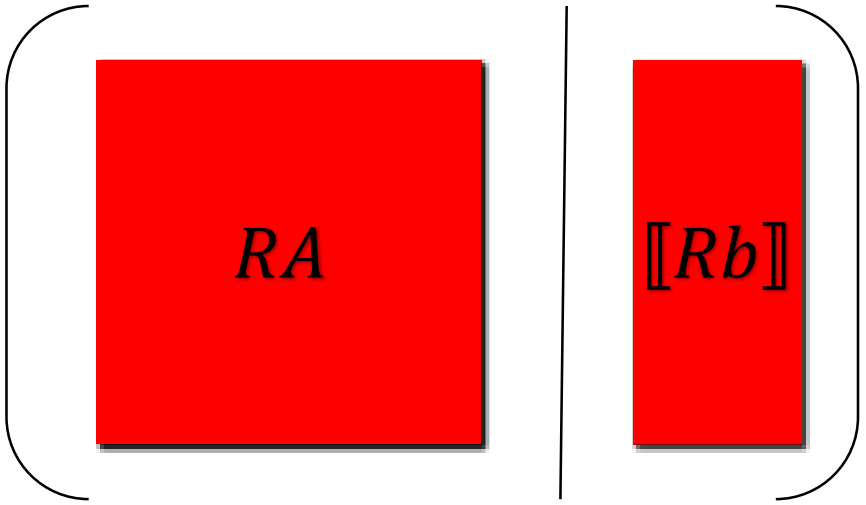


Return:$([\![\mathrm{adj}(A)b]\!], [\![\det(A)]\!])$

Randomized Gaussian Elimination

$A \in GL_d(\mathbb{Z}_p)$

1. Generate random mask $[\![R]\!] \in_R GL_d(\mathbb{Z}_p)$
2. Randomize system $(A|b)$
3. Open RA



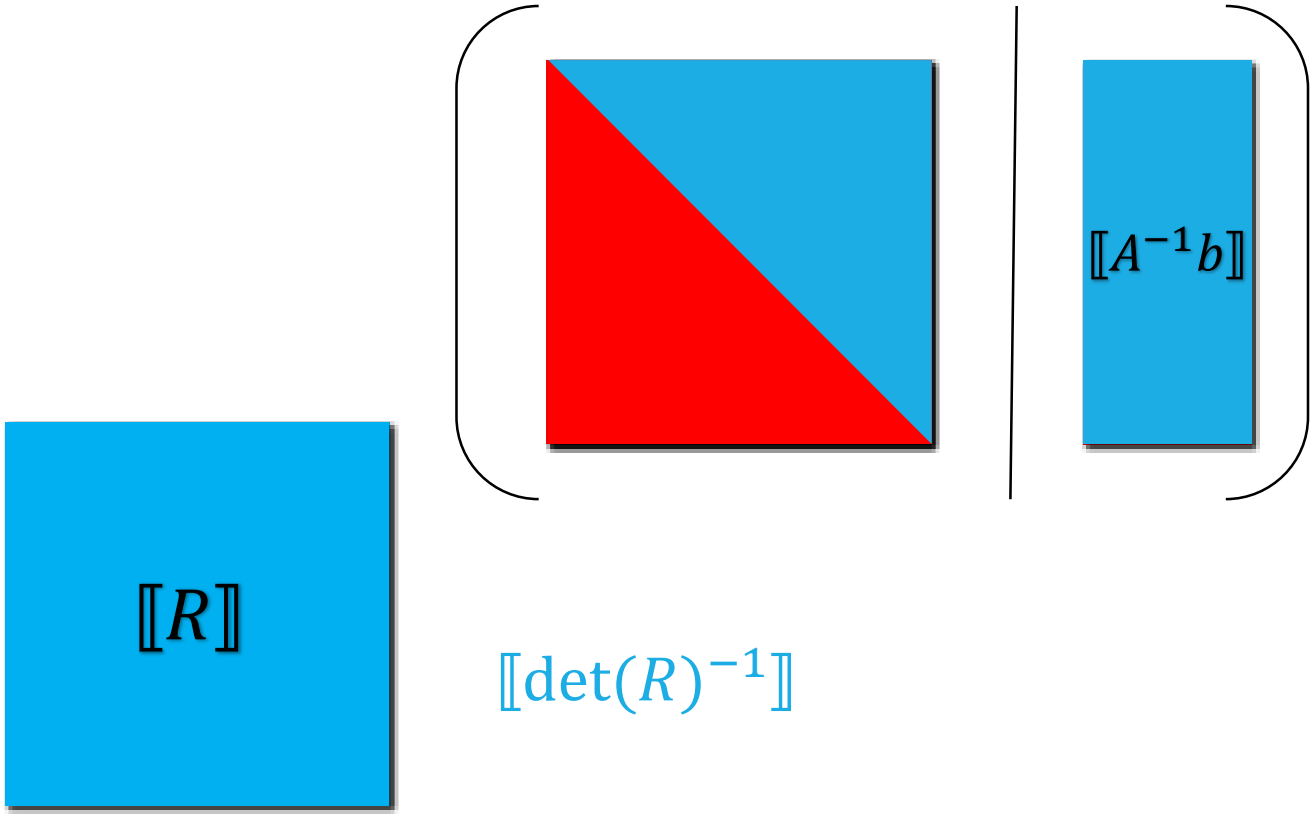
$[\![R]\!]$

$[\![\det(R)^{-1}]\!]$

Return:$([\![\mathrm{adj}(A)b]\!], [\![\det(A)]\!])$

Randomized Gaussian Elimination

$A \in GL_d(\mathbb{Z}_p)$

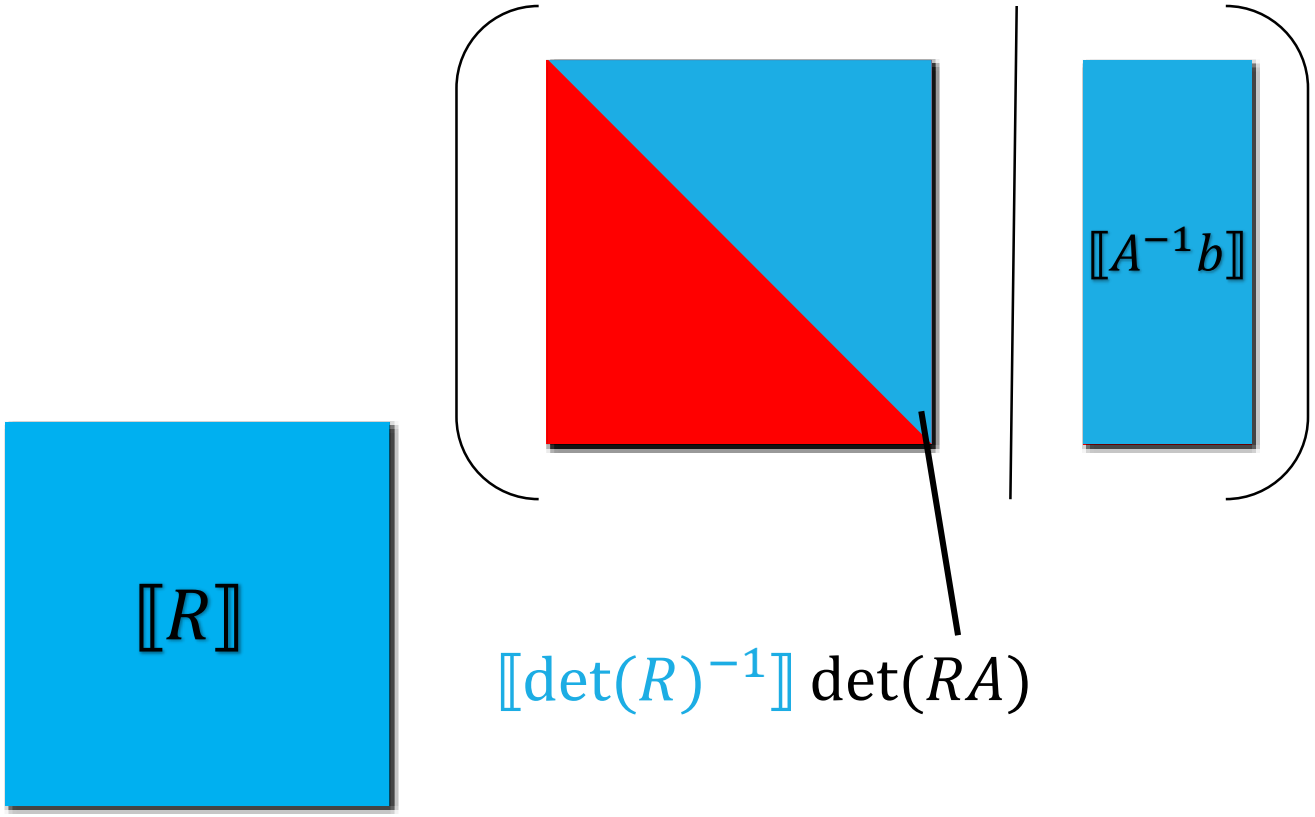


1. Generate random mask $[\![R]\!] \in_R GL_d(\mathbb{Z}_p)$
2. Randomize system $(A|b)$
3. Open RA
4. Gaussian Elimination over $(RA|[\![Rb]\!])$

Return:$([\![\mathrm{adj}(A)b]\!], [\![\det(A)]\!])$

Randomized Gaussian Elimination

$A \in GL_d(\mathbb{Z}_p)$

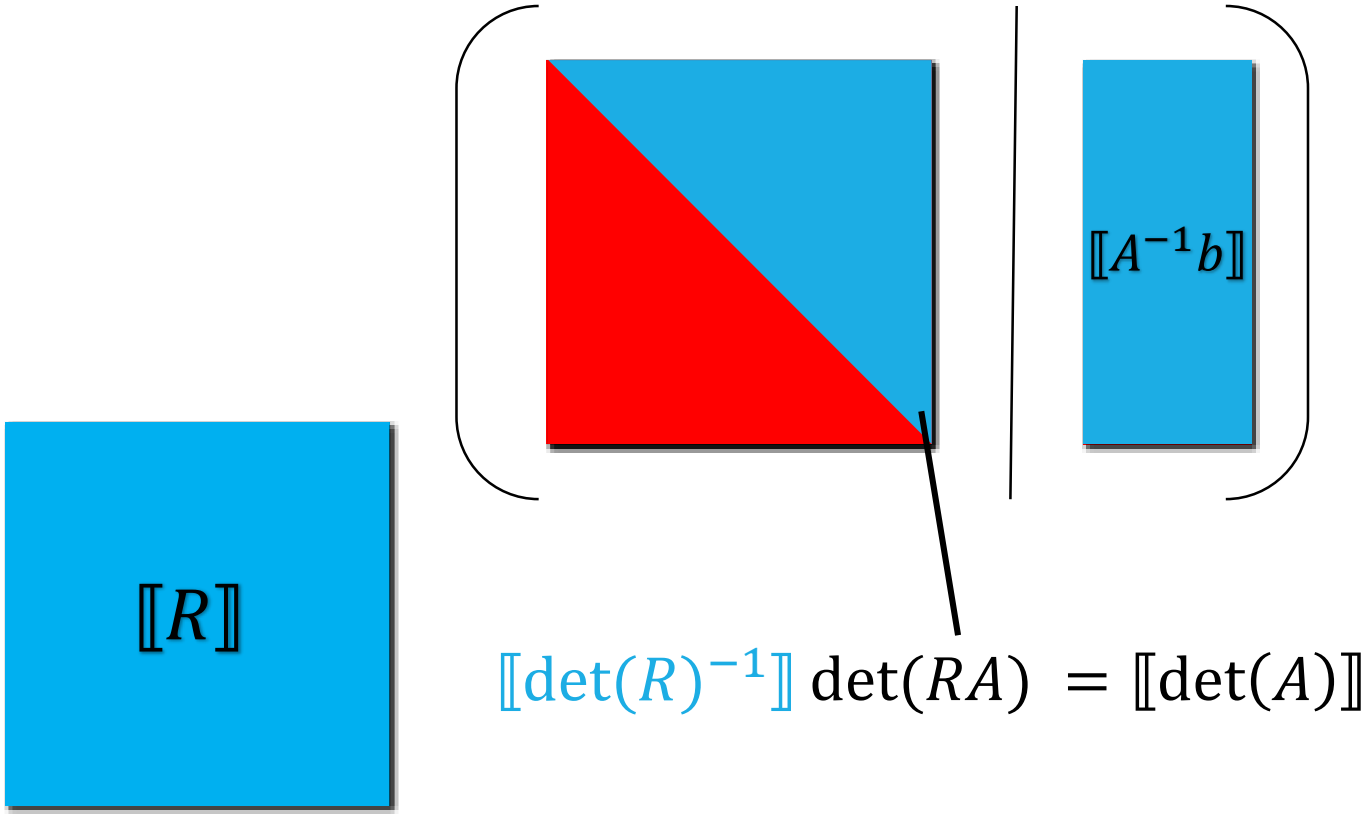


1. Generate random mask $[\![R]\!] \in_R GL_d(\mathbb{Z}_p)$
2. Randomize system $(A|b)$
3. Open RA
4. Gaussian Elimination over $(RA|[\![Rb]\!])$
5. Extract $\det(RA)$

Return:$([\![\mathrm{adj}(A)b]\!], [\![\det(A)]\!])$

Randomized Gaussian Elimination

$A \in GL_d(\mathbb{Z}_p)$



1. Generate random mask $[\![R]\!] \in_R GL_d(\mathbb{Z}_p)$
2. Randomize system $(A|b)$
3. Open RA
4. Gaussian Elimination over $(RA|[\![Rb]\!])$
5. Extract $\det(RA)$
6. Compute $[\![\det(A)]\!]$

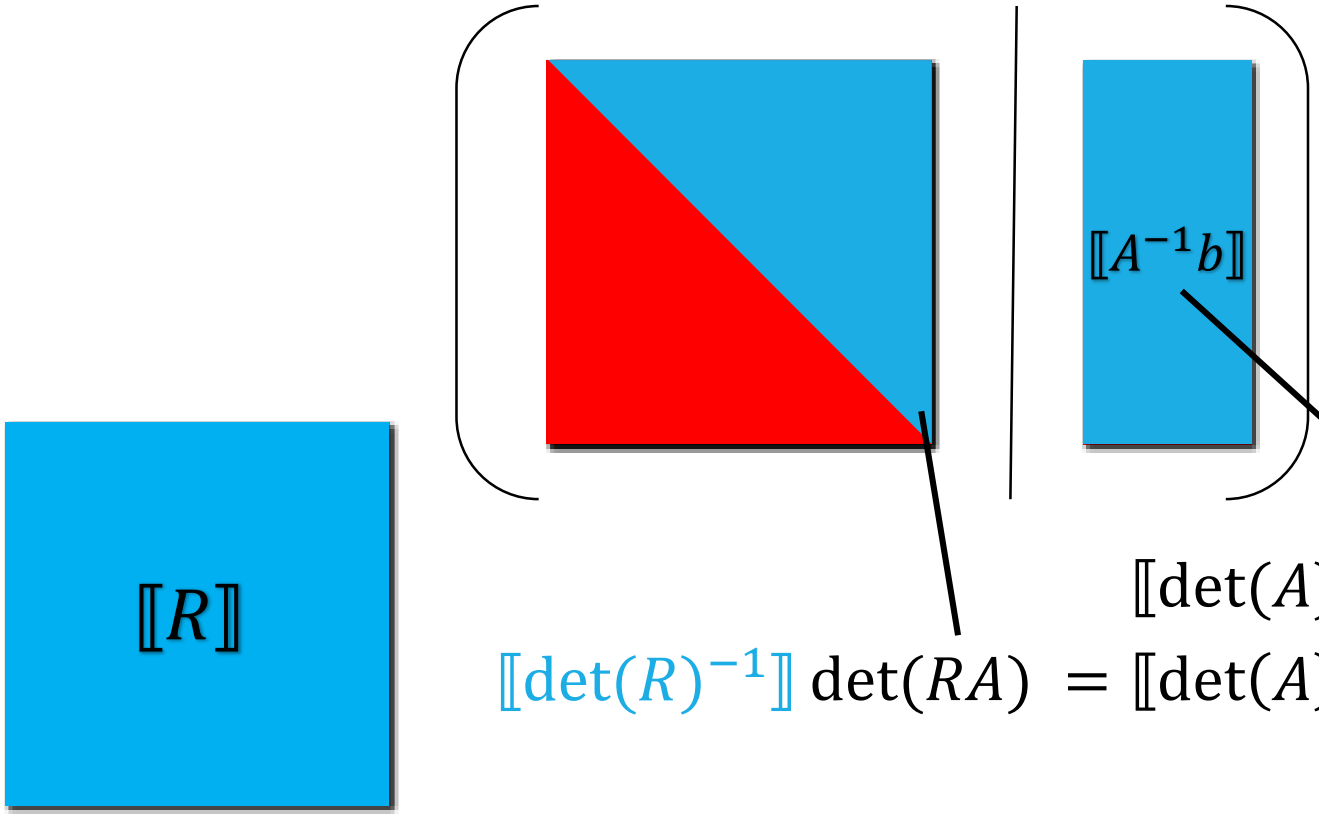
Return:$([\![\mathrm{adj}(A)b]\!], [\![\det(A)]\!])$

Randomized Gaussian Elimination

$A \in GL_d(\mathbb{Z}_p)$

1. Generate random mask $[\![R]\!] \in_R GL_d(\mathbb{Z}_p)$
2. Randomize system $(A|b)$
3. Open RA
4. Gaussian Elimination over $(RA|[\![Rb]\!])$
5. Extract $\det(RA)$
6. Compute $[\![\det(A)]\!]$
7. Compute $[\![\mathrm{adj}(A)b]\!]$

Return:$([\![\mathrm{adj}(A)b]\!], [\![\det(A)]\!])$



Generate random mask

Slight twist on Cramer, Damgård Protocol for secure computation of $\det(A)$ over a finite field [CD01]

$$R = LU = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ [\![l_{2,1}]\!] & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ [\![l_{d,1}]\!] & \cdots & [\![l_{d,d-1}]\!] & 1 \end{pmatrix} \begin{pmatrix} [\![u_{1,1}]\!] & [\![u_{1,2}]\!] & \cdots & [\![u_{1,d}]\!] \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & [\![u_{d-1,d}]\!] \\ 0 & \cdots & 0 & [\![u_{d,d}]\!] \end{pmatrix}$$

$$[\![\det(R)^{-1}]\!] = \left(\prod_{i=1}^{d} [\![u_{i,i}]\!]\right)^{-1}$$

Secure Ridge Regression

$$\text{Ridge}([\![X]\!], [\![y]\!], \lambda)$$

1. $[\![A]\!] = [\![X^T]\!][\![X]\!] + \lambda I$, and $[\![b]\!] = [\![X^T]\!][\![y]\!]$
2. $([\![\text{adj}(A)b]\!], [\![\det(A)]\!]) = \text{RandGaussElim}([\![A]\!], [\![b]\!])$
3. Set $[\![\det(A)w]\!] = [\![\text{adj}(A)b]\!]$ // $w = A^{-1}b$
4. Return $[\![\det(A)w]\!], [\![\det(A)]\!]$

Choosing the modulus size

- Return:$(\llbracket \mathrm{adj}(A)b \rrbracket, \llbracket \det(A) \rrbracket)$

  Requirement: $\mathrm{p}/2 > \max\{\|\mathrm{adj}(A)b\|_{max}, |\det(A)|\}$

- Hadamard's Bound: $|\det(M)| \leq \prod_{i=1}^{d} \|m_i\|_2 \leq d^{\frac{d}{2}} \|M\|_{\max}^{d}$

  For $A = X^T X + \lambda I, b = X^T y$ with $X \in [-2^k, 2^k]^{n \times d}, y \in [-2^k, 2^k]^n$

  Bit-length p dominated by: $d \cdot \log(n)$ $n \gg d$

Choosing the modulus size

Requirement: $\mathrm{p}/2 > \max\{\|\mathrm{adj}(A)b\|_{max}, |\det(A)|\}$

n	d	bit–length p
4.208.261	16	744
11.000.000	7	278

Bit-length of p:
we save a factor of two by avoiding rational reconstruction!

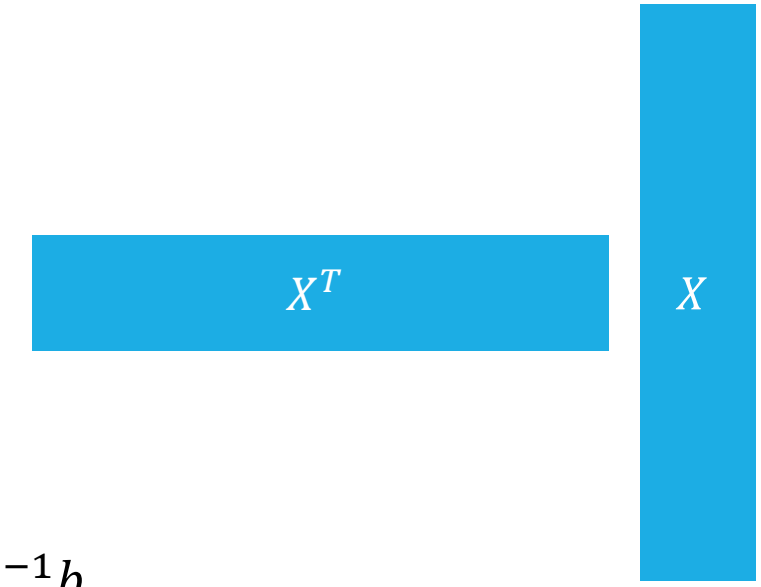
Avoiding rational reconstruction

- Represent fractions $a/b \in \mathbb{Q}$ (uniquely) as $ab^{-1} \in \mathbb{Z}_{\mathrm{p}}$
- Requirement: $p/2 > \|\mathrm{adj}(A)b\|_{max} \cdot |\det(A)|$

Large prime

$$\text{Ridge}([\![X]\!], [\![y]\!], \lambda)$$

1. $[\![A]\!] = [\![X^T]\!][\![X]\!] + \lambda I$, and $[\![b]\!] = [\![X^T]\!][\![y]\!]$
2. $([\![\text{adj}(A)b]\!], [\![\det(A)]\!]) = \text{RandGaussElim}([\![A]\!], [\![b]\!])$
3. Set $[\![\det(A)w]\!] = [\![\text{adj}(A)b]\!]$ // $w = A^{-1}b$
4. Return $[\![\det(A)w]\!], [\![\det(A)]\!]$



n	d	bit–length p
4.208.261	16	744
11.000.000	7	278

✓ Cheap inner product

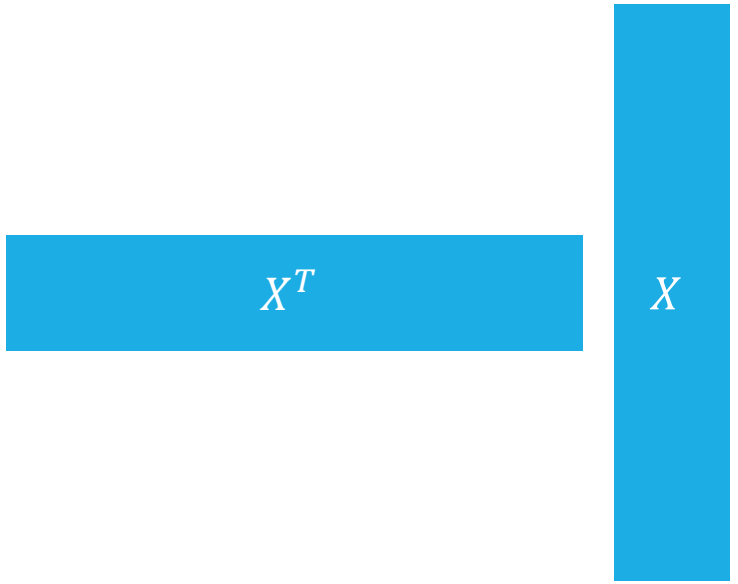
× Many field elements

High memory usage

Long computation time

Two prime moduli: $q < p$

$$\text{Ridge}(\llbracket X \rrbracket_q, \llbracket y \rrbracket_q, \lambda)$$



1. $\llbracket A \rrbracket_q = \llbracket X^T \rrbracket_q \llbracket X \rrbracket_q + \lambda I$, and $\llbracket b \rrbracket_q = \llbracket X^T \rrbracket_q \llbracket y \rrbracket_q$
2. Convert $(\llbracket A \rrbracket_q, \llbracket b \rrbracket_q)$ to $(\llbracket A \rrbracket_p, \llbracket b \rrbracket_p)$
3. $\left(\llbracket \text{adj}(A)b \rrbracket_p, \llbracket \det(A) \rrbracket_p\right) = \text{RandGaussElim}(\llbracket A \rrbracket_p, \llbracket b \rrbracket_p)$
4. Set $\llbracket \det(A)w \rrbracket_p = \llbracket \text{adj}(A)b \rrbracket_p$ // $w = A^{-1}b$
5. Return $\llbracket \det(A)w \rrbracket_p, \llbracket \det(A) \rrbracket_p$

n	d	bit–length p	bit–length q
4.208.261	16	744	75
11.000.000	7	278	66

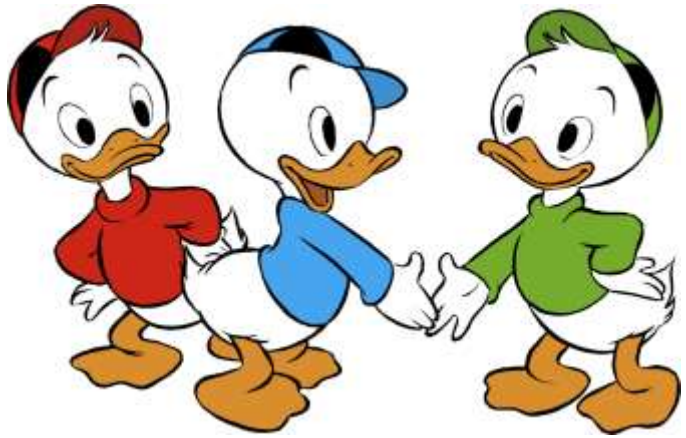
Implementation

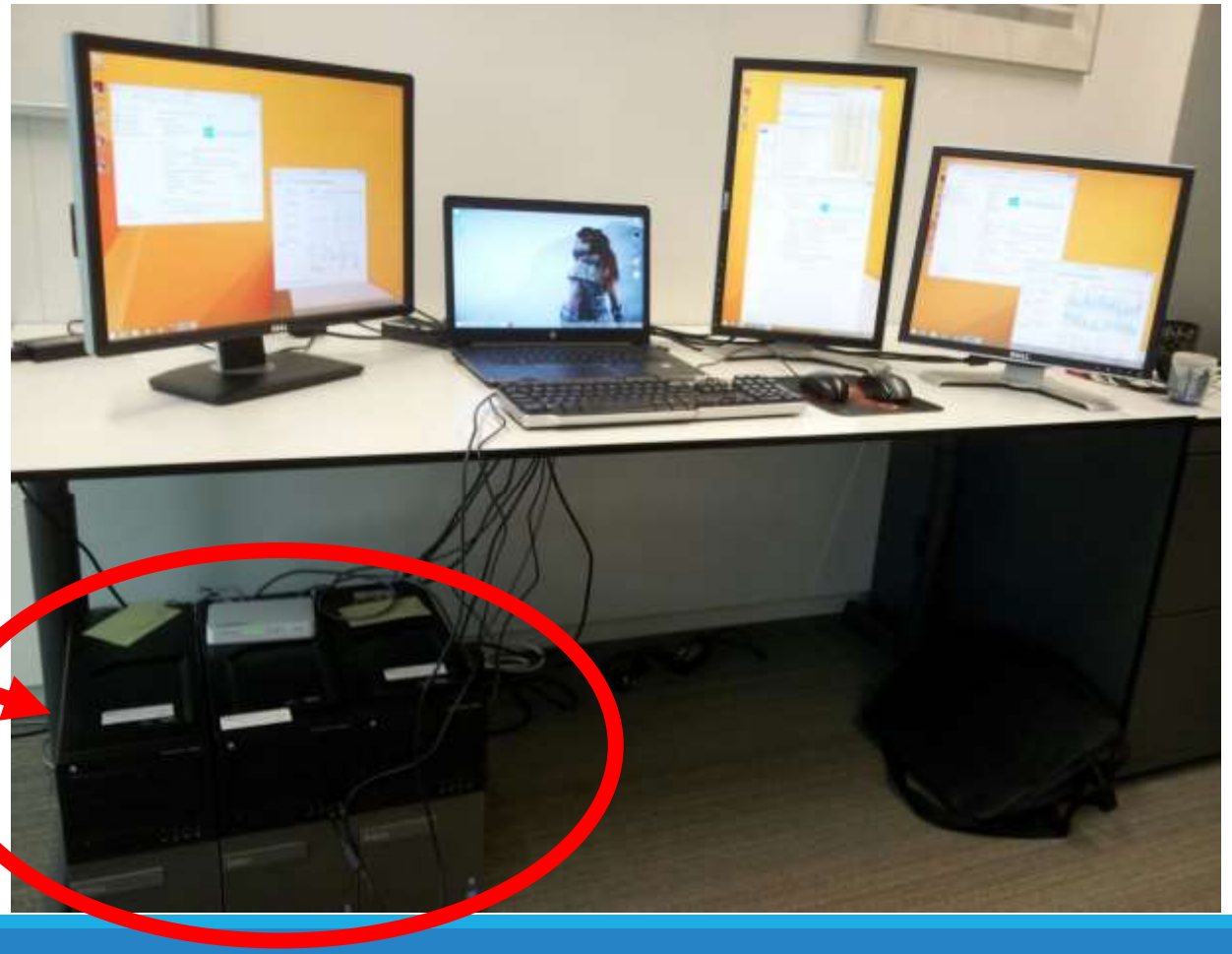
Implementation in MPyC (Shamir Secret Sharing)

MPyC **Secure Multiparty Computation in Python**

github.com/lschoe/mpyc

Multiparty computation (3 parties)





Implementation results UCI Datasets

- Nikolaenko etal. [NWI+13]
  - Horizontal data-partitioning,
  - Fixed-point arithmetic
- Giacomelli etal. [GJJ+13]
  - Horizontal & Vertical data-partitioning
  - Rational Reconstruction
- Gascón etal. [GSB+17]
  - Vertical data-partitioning
  - Fixed-point arithmetic

Dataset	n	d	This Work Time (s)	[NWI+13] HP	[GJJ+13] HP	[GJJ+13] VP	[GSB+17] VP 32-bit	[GSB+17] VP 64-bit
Student	395	30	1,61	-	39,76	328,06	5 (-0,0%)	35 (-0,0%)
Wine (white)	4898	11	0,16	45	4,09	-	0 (4,2%)	4 (-0,0%)
Year Prediction MSD	515.345	90	255	-	-	-	230 (0,0%)	808 (0,0%)
Gas Sensor Array (CO)	4.208.261	16	63,2	-	-	-	42 (5.2%)	69 (0,0%)
HIGGS	11.000.000	7	35,0	-	-	-	-	-

$(< 0.1\%)$

Conclusion

- Secure Integer valued linear system solving
- Efficient ridge regression without data-partitioning assumptions
- Competitive with existing solutions

win.tue.nl/~berry/mpyc/bareiss.pdf

Final remarks

- Slight changes to the protocol to allow underdetermined systems $(n \ll d)$
- Avoiding rational reconstruction also useful for secure computation of Moore-Penrose pseudoinverse [BdV19]

References

[BdV19] Niek J. Bouman and Niels de Vreede.
A practical approach to the secure computation of the moore-penrose pseudoinverse over the rationals.
Cryptology ePrint Archive, Report 2019/470, 2019.
https://eprint.iacr.org/2019/470.

[CD01] Ronald Cramer and Ivan Damgård.
Secure distributed linear algebra in a constant number of rounds.
In Proc. CRYPTO 2001, Santa Barbara, USA, pages 119–136. Springer, 2001.

[GJJ+13] Irene Giacomelli, Somesh Jha, Marc Joye, C. David Page, and Kyonghwan Yoon.
Privacy-preserving ridge regression with only linearly-homomorphic encryption.
In Bart Preneel and Frederik Vercauteren, editors, Applied Cryptography and Network Security - 16th International Conference, ACNS 2018, Leuven, Belgium, July 2-4, 2018, Proceedings, volume 10892 of Lecture Notes in Computer Science, pages 243–261. Springer, 2018.

References

[GSB+17] Adrià Gascón, Phillipp Schoppmann, Borja Balle, Mariana Raykova, Jack Doerner, Samee Zahur, and David Evans.
Privacy-preserving distributed linear regression on high-dimensional data.
Proceedings on Privacy Enhancing Technologies, 2017(4):345–364, 2017.

[NWI+13] Valeria Nikolaenko, Udi Weinsberg, Stratis Ioannidis, Marc Joye, Dan Boneh, and Nina Taft.
Privacy-Preserving Ridge Regression on Hundreds of Millions of Records.
In Proceedings of the 2013 IEEE Symposium on Security and Privacy, SP '13, pages 334–348, Washington, DC, USA, 2013. IEEE Computer Society.