Practical privacy-preserving k-means based on Homomorphic Encryption

Lorenzo Rovida
l.rovida1@campus.unimib.it

Department of Informatics, Systems and Communication - University of Milano-Bicocca - Milan, Italy

3rd Workshop on Artificial Intelligence and Cryptography - AICRYPT@Lyon, France - April 22, 2023

Table of Contents

Table of Contents

The K-means algorithm

- Unsupervised machine learning technique for clustering data points

The K-means algorithm

- Unsupervised machine learning technique for clustering data points
- Divides data into k clusters based on similarity

The K-means algorithm

- Unsupervised machine learning technique for clustering data points
- Divides data into k clusters based on similarity
- Algorithm:
  - Initialize k centroids randomly
  - Assign each data point to the closest centroid
  - Calculate the mean of each cluster and move the centroid to the mean

The K-means algorithm

- Unsupervised machine learning technique for clustering data points
- Divides data into k clusters based on similarity
- Algorithm:
  - Initialize k centroids randomly
  - Assign each data point to the closest centroid
  - Calculate the mean of each cluster and move the centroid to the mean
- Applications: image segmentation, customer segmentation, anomaly detection, etc.

Setting overview

Most of the works that implement k-means using Homomorphic Encryption (HE) in literature are either impractical[1] (e.g. hours for a single iteration) or allocate workload on the client[2].

Given a honest-but-curious server, we want:

[1] Jäschke and Armknecht, "Unsupervised Machine Learning on Encrypted Data".

[2] Theodouli, Draziotis, and Gounaris, "Implementing private k-means clustering using a LWE-based cryptosystem".

Setting overview

Most of the works that implement k-means using Homomorphic Encryption (HE) in literature are either impractical[1] (e.g. hours for a single iteration) or allocate workload on the client[2].

Given a honest-but-curious server, we want:

- the execution of the k-means algorithm;

[1] Jäschke and Armknecht, "Unsupervised Machine Learning on Encrypted Data".

[2] Theodouli, Draziotis, and Gounaris, "Implementing private k-means clustering using a LWE-based cryptosystem".

Setting overview

Most of the works that implement k-means using Homomorphic Encryption (HE) in literature are either impractical[1] (e.g. hours for a single iteration) or allocate workload on the client[2].

Given a honest-but-curious server, we want:

- the execution of the k-means algorithm;
- the client data to never be visible to anyone;

[1] Jäschke and Armknecht, "Unsupervised Machine Learning on Encrypted Data".

[2] Theodouli, Draziotis, and Gounaris, "Implementing private k-means clustering using a LWE-based cryptosystem".

Setting overview

Most of the works that implement k-means using Homomorphic Encryption (HE) in literature are either impractical[1] (e.g. hours for a single iteration) or allocate workload on the client[2].

Given a honest-but-curious server, we want:

- the execution of the k-means algorithm;
- the client data to never be visible to anyone;
- practicability;

[1] Jäschke and Armknecht, "Unsupervised Machine Learning on Encrypted Data".

[2] Theodouli, Draziotis, and Gounaris, "Implementing private k-means clustering using a LWE-based cryptosystem".

Setting overview

Most of the works that implement k-means using Homomorphic Encryption (HE) in literature are either impractical[1] (e.g. hours for a single iteration) or allocate workload on the client[2].

Given a honest-but-curious server, we want:

- the execution of the k-means algorithm;
- the client data to never be visible to anyone;
- practicability;
- a server-centric workload.

[1] Jäschke and Armknecht, "Unsupervised Machine Learning on Encrypted Data".

[2] Theodouli, Draziotis, and Gounaris, "Implementing private k-means clustering using a LWE-based cryptosystem".

Table of Contents

The CKKS scheme

The Cheon-Kim-Kim-Song (CKKS) scheme[1] is a leveled Homomorphic Encryption scheme that encodes and encrypts vectors of complex numbers. It is based on Ring-Learning with Errors[2] and uses its noise as a part of the message.

$$\mathbf{v} \subset \mathbb{C}^{N/2} \xrightarrow{encode} \mathbb{Z}[X]/(X^N+1) \xrightarrow{encrypt} (\mathbb{Z}_q[X]/(X^N+1))^2$$

[1] Cheon, A. Kim, *et al.*, "Homomorphic Encryption for Arithmetic of Approximate Numbers".

[2] Lyubashevsky, Peikert, and Regev, "On Ideal Lattices and Learning with Errors over Rings".

The CKKS scheme

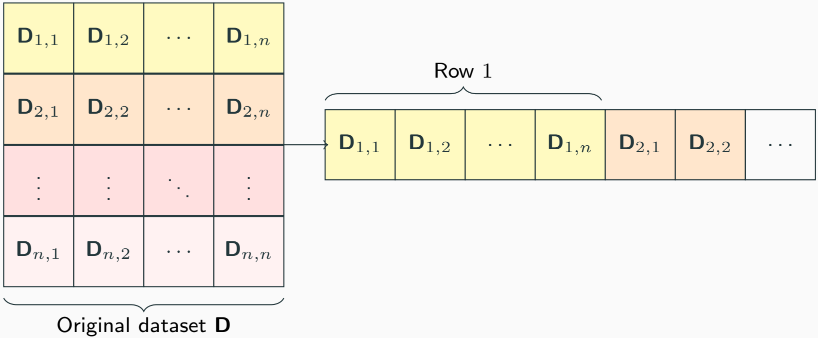
- From a purely algorithmic point of view, we can consider a CKKS ciphertext as a vector $\mathbf{v} \subset \mathbb{C}^{N/2}$.
- Homomorphic operations performed on such encrypted vectors are applied slot-wise, enabling powerful Single Instruction, Multiple Decoding (SIMD) functionality.

$$
\begin{array}{r|c|c|c|c|l}
\hline
 & \mathbf{v}_0 & \mathbf{v}_1 & \cdots & \mathbf{v}_{N/2} & * \\
\hline
 & \mathbf{w}_0 & \mathbf{w}_1 & \cdots & \mathbf{w}_{N/2} & \\
\hline
\approx & \mathbf{v}_0 * \mathbf{w}_0 & \mathbf{v}_1 * \mathbf{w}_1 & \cdots & \mathbf{v}_{N/2} * \mathbf{w}_{N/2} & \\
\hline
\end{array}
$$

Table of Contents

Packing data into Ciphertexts

- Data must be encoded cleverly in order to obtain fast computations;
- Manipulating data in CKKS is not an easy task!
- We encode the dataset **D** in Row-major order.



Packing data into Ciphertexts

- Data must be encoded cleverly in order to obtain fast computations;
- We encode each centroid $\mathbf{c}_i \ : \ 0 < i < k$ in Repeated shape:

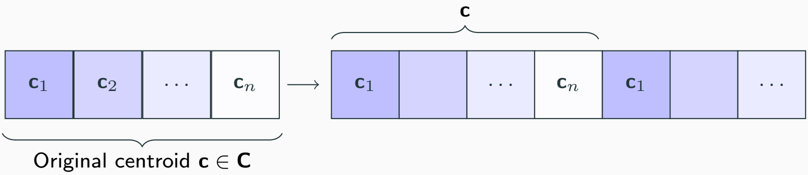


Table of Contents

Comparing values

- A key operation in k-means is the evaluation of $a < b$, since we want to find the least distance.
- Problem: we can not evaluate that expression trivially in CKKS!

Comparing values

- A key operation in k-means is the evaluation of $a < b$, since we want to find the least distance.
- Problem: we can not evaluate that expression trivially in CKKS!
- Proposed solution: re-write the $a < b$ expression:

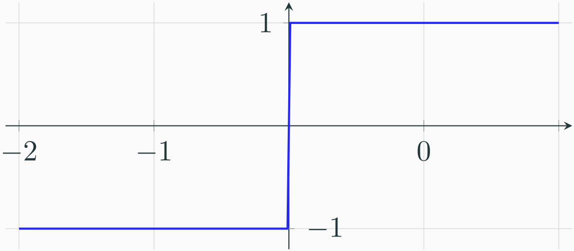
if $a < b$ **then**
  return a
else
  return b
end if

$\xRightarrow[\text{becomes}]{}$

$d \leftarrow (1 + \mathsf{sgn}(a - b))/2$
return $a \cdot (1 - s) + b \cdot s$

Comparing values

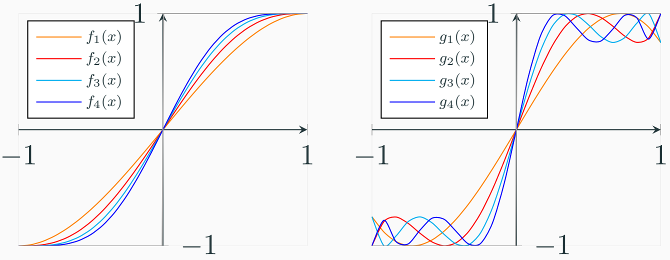
The sgn(x) function is not polynomial, how do we evaluate it using only additions and multiplications?



Comparing values

We will exploit some existing work[1] that provides, approximations of $\text{sgn}(x)$ in $[-1, 1]$ that can composed together.

There are two families of functions: $f_n(x)$ and $g_n(x)$:

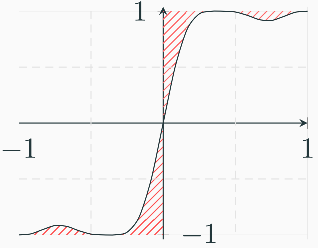


(a) $f_n(x)$ **(b)** $g_n(x)$

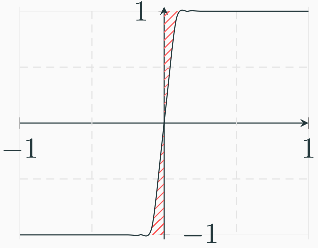
[1] Cheon, D. Kim, and D. Kim, "Efficient Homomorphic Comparison Methods with Optimal Complexity".

Comparing values

We composed different functions and defined two approximations:



(c) High speed
$g_1(x) \circ g_1(x) \circ f_1(x)$
Poly degree: 12



(d) High precision
$g_1(x) \circ g_1(x) \circ g_1(x) \circ g_1(x) \circ f_1(x)$
Poly degree: 20

A note on approximations

- Approximations work on $[-1, 1]$,
- We will compare L2-distances, their differences is not always $\in [-1, 1]$.
- We want all distances to be less or equal than 1:

$$\sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \cdots + (p_n - q_n)^2} \leq 1$$

A note on approximations

We will ask the client to perform a preprocessing before sending data:

- Transform each column $(\mathbf{D})_j^T$ from $[\min_j, \max_j]$[1] to $[0, \sqrt{n}/n]$.
- Now the maximum length of a vector is given by:

$$
\begin{aligned}
&(0 - \frac{\sqrt{n}}{n})^2 + (0 - \frac{\sqrt{n}}{n})^2 + \cdots + (0 - \frac{\sqrt{n}}{n})^2 \\
&= n \cdot (0 - \frac{\sqrt{n}}{n})^2 \\
&= n \cdot \frac{1}{n} = 1
\end{aligned}
$$

[1] These values represent the min and the max value of the j-th column

Table of Contents

Computing L2-norm

- The first step of k-means is to compute the distances[1] between all the points and all the centroids.
- We can perform this operation very quickly by taking advantage of SIMD operations.

[1] We can skip the square root computation, since $a < b \leftrightarrow \sqrt{a} < \sqrt{b}$

Computing L2-norm

- The first step of k-means is to compute the distances[1] between all the points and all the centroids.
- We can perform this operation very quickly by taking advantage of SIMD operations.
- Given $\mathbf{d}$, the ciphertext containing the dataset, for each encrypted centroid $\mathbf{c}_i$ (in Repeated mode), we evaluate $(\mathbf{d} - \mathbf{c}_i)^2$, component-wise.

[1] We can skip the square root computation, since $a < b \leftrightarrow \sqrt{a} < \sqrt{b}$

Computing L2-norm

For each centroid $\mathbf{c}_i$:

$\mathbf{d}$:	$\mathbf{d}_{0,0}$	$\mathbf{d}_{0,1}$	$\cdots$	$\mathbf{d}_{0,n}$	$\mathbf{d}_{1,0}$	$\mathbf{d}_{1,1}$	$\cdots$
$\mathbf{c}_i$:	$\mathbf{c}_{i,0}$	$\mathbf{c}_{i,1}$	$\cdots$	$\mathbf{c}_{i,n}$	$\mathbf{c}_{i,0}$	$\mathbf{c}_{i,1}$	$\cdots$

$\downarrow$

$(\mathbf{d}_{0,0} - \mathbf{c}_{i,0})^2$	$(\mathbf{d}_{0,1} - \mathbf{c}_{i,1})^2$	$\cdots$	$(\mathbf{d}_{0,n} - \mathbf{c}_{i,n})^2$	$(\mathbf{d}_{1,0} - \mathbf{c}_{i,0})^2$	$(\mathbf{d}_{1,1} - \mathbf{c}_{i,1})^2$	$\cdots$

By summing these n values, we obtain the squared
L2 distance between the first point and the i-th centroid

Generating the sub-masks

Given a point $\mathbf{p} \in \mathbf{D}$, for each pair of centroids $\mathbf{c}_i$ and $\mathbf{c}_j$:

$$\mathbf{c}_i \text{ vs } \mathbf{c}_j = \begin{cases} 1 \text{ if } ||\mathbf{p} - \mathbf{c}_i|| < ||\mathbf{p} - \mathbf{c}_j|| \\ 0 \text{ otherwise} \end{cases}$$

- Basically, this value is 1 if a given point is closer to c_i than to c_j.

Generating the sub-masks

Given a point $\mathbf{p} \in \mathbf{D}$, for each pair of centroids $\mathbf{c}_i$ and $\mathbf{c}_j$:

$$\mathbf{c}_i \text{ vs } \mathbf{c}_j = \begin{cases} 1 \text{ if } ||\mathbf{p} - \mathbf{c}_i|| < ||\mathbf{p} - \mathbf{c}_j|| \\ 0 \text{ otherwise} \end{cases}$$

- Basically, this value is 1 if a given point is closer to c_i than to c_j.
- This value is computed by applying a comparison function to each pair of distances.

$$\text{compare}(a, b) = \frac{\text{sgn}'(a - b) + 1}{2}$$

We will call these values **sub-masks**.

Generating the masks

- We put all these sub-masks in a matrix, which we will call **comparison matrix**.
- Each i-th row contains the comparisons made using $\mathbf{c}_i$ and all the other centroids.

$\mathbf{c}_1$ vs $\mathbf{c}_1$	$\mathbf{c}_1$ vs $\mathbf{c}_2$	$\dots$	$\mathbf{c}_1$ vs $\mathbf{c}_k$
$\mathbf{c}_2$ vs $\mathbf{c}_1$	$\mathbf{c}_2$ vs $\mathbf{c}_2$	$\dots$	$\mathbf{c}_2$ vs $\mathbf{c}_k$
$\vdots$	$\dots$	$\ddots$	$\vdots$
$\mathbf{c}_k$ vs $\mathbf{c}_1$	$\mathbf{c}_k$ vs $\mathbf{c}_2$	$\dots$	$\mathbf{c}_k$ vs $\mathbf{c}_k$

Generating the masks

Key idea

If, for a point p, we find all ones in the i-th row, it means that **that point is closer to that centroid than all the other**! On the other hand, if there is at least one zero, there is another centroid whose distance is smaller.

- We will call the i-th mask the product of all the values in the i-th row:

$$\mathsf{mask}_i = \prod_{j=1}^{k} \mathbf{c}_i \text{ vs } \mathbf{c}_j$$

Generating the masks

- We can reduce the computational complexity of the matrix from k^2 to $\frac{k(k-1)}{2}$.
- The key is to observe that $\mathbf{c}_i$ vs $\mathbf{c}_j = 1 - \mathbf{c}_j$ vs $\mathbf{c}_i$

1	$\mathbf{c}_1$ vs $\mathbf{c}_2$	$\ldots$	$\mathbf{c}_1$ vs $\mathbf{c}_k$
$1 - (\mathbf{c}_1$ vs $\mathbf{c}_2)$	1	$\ldots$	$\mathbf{c}_2$ vs $\mathbf{c}_k$
$\vdots$	$\ldots$	$\ddots$	$\vdots$
1 - ($\mathbf{c}_1$ vs $\mathbf{c}_k$)	1 - ($\mathbf{c}_2$ vs $\mathbf{c}_k$)	$\ldots$	1

Assembling the parts

Let's put everything together!

Assembling the parts

Let's put everything together!

1. Compute the distances between all the points and all the centroids;

Assembling the parts

Let's put everything together!

1. Compute the distances between all the points and all the centroids;
2. For each pair of centroids $\mathbf{c}_i, \mathbf{c}_j$, compute the sub-mask by finding the sign of the difference between distances;

Assembling the parts

Let's put everything together!

1. Compute the distances between all the points and all the centroids;
2. For each pair of centroids $\mathbf{c}_i, \mathbf{c}_j$, compute the sub-mask by finding the sign of the difference between distances;
3. Generate the k masks, and use them to build the k clusters;

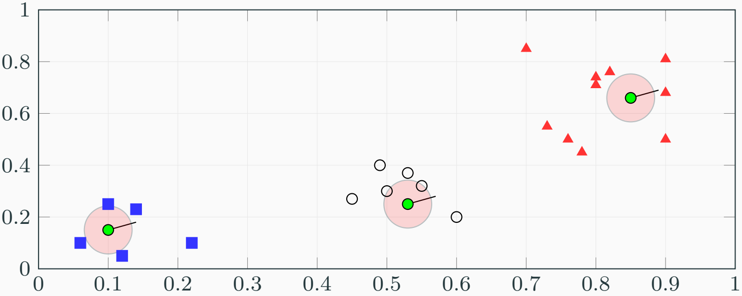
Assembling the parts

Let's put everything together!

1. Compute the distances between all the points and all the centroids;
2. For each pair of centroids $\mathbf{c}_i, \mathbf{c}_j$, compute the sub-mask by finding the sign of the difference between distances;
3. Generate the k masks, and use them to build the k clusters;
4. Return the encrypted result to the client.

Table of Contents

- Experiments are evaluated by measuring an error, which is equal to the Euclidean distance between the "real" centroids and the encrypted ones.
- In the following figure, the mean error is equal to 0.05. The green points are plain centroids, whereas the encrypted ones lie somewhere inside the red areas



Experiments

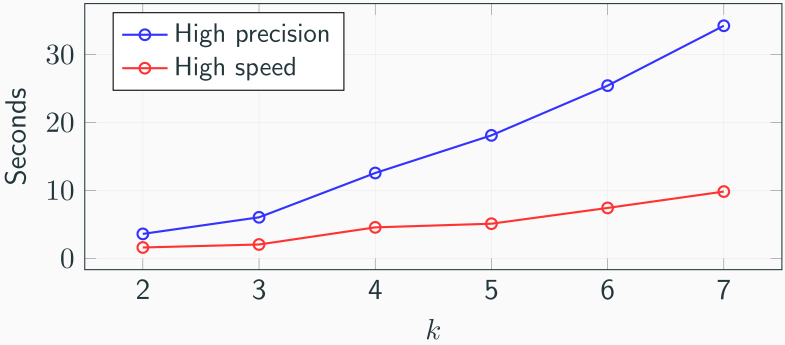
- We implemented the algorithm in Microsoft SEAL[1];
- We defined two sets of parameters: one defines fast computations, the other accurate.

	High Speed	High Precision
Polynomial modulus degree (N)	2^{14}	2^{15}
Scale (Δ)	2^{27}	2^{40}
Circuit depth	10	19
sgn(x) degree	12	20
$\lvert 1 - \int_0^1 \mathsf{sgn}'(x)\,dx \rvert$	≈ 0.1161	≈ 0.0374

[1] *Microsoft SEAL (release 3.7).*

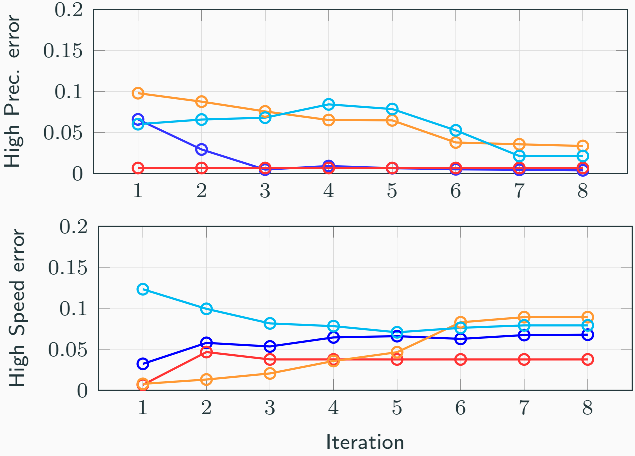
High Speed vs High Precision

- We start by evaluating the iteration runtimes of the two versions on the Iris dataset;
- The High Speed version is faster, but the order of magnitude is more or less the same (seconds).



High speed vs High precision

- Errors in High Speed are more than doubled at the last iteration! Each line represent a different centroid:



High precision training

- We start from the high precision training phase, which is run for 10 iteration;
- Performance has been evaluated on various UCI Machine Learning datasets;
- Runtimes are obtained on a standard desktop PC, equipped with a Ryzen 3600 CPU and 16GB of RAM.

Dataset name	k	n	m	Mean error	Mean runtime
Iris	4	150	4	0.0293 ± 0.0028	$6.724\text{s} \pm 0.321\text{s}$
Wine	3	178	12	0.0631 ± 0.0031	$8.024\text{s} \pm 0.502\text{s}$
Breast Cancer (Diagnostic)	2	569	30	0.0685 ± 0.0041	$7.441\text{s} \pm 0.571\text{s}$
Absenteeism at work	2	740	20	0.0910 ± 0.0011	$9.451\text{s} \pm 0.460\text{s}$
Seeds	3	210	6	0.0566 ± 0.0023	$6.944\text{s} \pm 0.290\text{s}$
Transfusion	3	748	4	0.0633 ± 0.0021	$7.345\text{s} \pm 0.367\text{s}$
Banknote Authentication	2	1372	4	0.0633 ± 0.0021	$6.784\text{s} \pm 0.288\text{s}$
Tripadvisor Review	5	979	10	0.0392 ± 0.0148	$21.621\text{s} \pm 0.572\text{s}$

Extreme high speed classification

- We propose to use an Extreme High Speed setting for the classification task only ($N = 2^{13}, \Delta = 2^{50}$);
- The server computes the distances between all the centroids and a single point, the client finds the minimum (trivial computation, minimum among k values).

Dataset name	m	k	Acc.	Execution time
Iris	4	4	1.000	$0.006\text{s} \pm 0.003\text{s}$
Wine	12	3	1.000	$0.012\text{s} \pm 0.004\text{s}$
Breast cancer (diagnostic)	30	2	1.000	$0.045\text{s} \pm 0.014\text{s}$
Absenteeism at work	20	2	1.000	$0.023\text{s} \pm 0.009\text{s}$
Seeds	6	3	1.000	$0.009\text{s} \pm 0.003\text{s}$
Transfusion	4	3	0.998	$0.005\text{s} \pm 0.010\text{s}$
Banknote authentication	4	2	0.999	$0.006\text{s} \pm 0.003\text{s}$
Tripadvisor review	10	5	1.000	$0.021\text{s} \pm 0.006\text{s}$

Summing up

After running the experiments, we ended up with an ideal setting:

- High precision can be used to train the model (on a powerful server this could be done in seconds);
- Extreme high speed can be used to ask for a classification

Summing up

Doubt

A question may arise: is it not better for the client to compute itself k-means?

Summing up

Doubt

A question may arise: is it not better for the client to compute itself k-means?

Outsourcing the computation can be useful, since:

- The client does not need to store data

Doubt

A question may arise: is it not better for the client to compute itself k-means?

Outsourcing the computation can be useful, since:

- The client does not need to store data
- Handling the homomorphic results (which has complexity $\mathcal{O}(n)$) is faster than executing the iteration by itself

Summing up

Doubt

A question may arise: is it not better for the client to compute itself k-means?

Outsourcing the computation can be useful, since:

- The client does not need to store data
- Handling the homomorphic results (which has complexity $\mathcal{O}(n)$) is faster than executing the iteration by itself
- In the following years precision and runtimes of HE schemes will perform much better!

Table of Contents

Future work

- Each iteration needs the help of the client to compute the new centroids, by using CKKS bootstrapping[1] (which is not implemented in SEAL) we would be able to perform all the iterations server-side;
- By using a larger polynomial modulus degree (e.g. $N = 2^{16}$) and more recent libraries (e.g. OpenFHE[2], released in recent months), we could obtain even better results;
- By finding better approximations for sgn(x) we can incraese the precision of computations.

[1] Bossuat *et al.*, "Efficient Bootstrapping for Approximate Homomorphic Encryption with Non-sparse Keys".

[2] Al Badawi *et al.*, "OpenFHE: Open-Source Fully Homomorphic Encryption Library".

References i

[21] *Microsoft SEAL (release 3.7)*. `https://github.com/Microsoft/SEAL`. Microsoft Research, Redmond, WA. Sept. 2021.

[Al +22] A. Al Badawi *et al.* "OpenFHE: Open-Source Fully Homomorphic Encryption Library". *Proceedings of the 10th Workshop on Encrypted Computing & Applied Homomorphic Cryptography*. WAHC'22. Los Angeles, CA, USA: Association for Computing Machinery, 2022, pp. 53–63.

[Bos+21] J.-P. Bossuat *et al.* "Efficient Bootstrapping for Approximate Homomorphic Encryption with Non-sparse Keys". *Advances in Cryptology – EUROCRYPT 2021*. Ed. by A. Canteaut and F.-X. Standaert. Cham: Springer International Publishing, 2021, pp. 587–617.

[Che+17] J. H. Cheon, A. Kim, *et al.* "Homomorphic Encryption for Arithmetic of Approximate Numbers". *Advances in Cryptology – ASIACRYPT 2017*. Ed. by T. Takagi and T. Peyrin. Cham: Springer International Publishing, 2017, pp. 409–437.

References ii

[CKK20] J. H. Cheon, D. Kim, and D. Kim. "Efficient Homomorphic Comparison Methods with Optimal Complexity". *Advances in Cryptology – ASIACRYPT 2020*. Ed. by S. Moriai and H. Wang. Cham: Springer International Publishing, 2020, pp. 221–256.

[JA19] A. Jäschke and F. Armknecht. "Unsupervised Machine Learning on Encrypted Data". *Selected Areas in Cryptography – SAC 2018*. Ed. by C. Cid and M. J. Jacobson Jr. Cham: Springer International Publishing, 2019, pp. 453–478.

[LPR10] V. Lyubashevsky, C. Peikert, and O. Regev. "On Ideal Lattices and Learning with Errors over Rings". *Advances in Cryptology – EUROCRYPT 2010*. Ed. by H. Gilbert. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 1–23.

[TDG17] A. Theodouli, K. A. Draziotis, and A. Gounaris. "Implementing private k-means clustering using a LWE-based cryptosystem". *2017 IEEE Symposium on Computers and Communications (ISCC)*. 2017, pp. 88–93.