Poster: Secure and Differentially Private k^{th} Ranked Element

Gowri R Chandran
Technical University
Darmstadt, Germany
chandran@encrypto.cs.tu-darmstadt.de

Philipp-Florens Lehwalder
Technical University
Darmstadt, Germany
philipp-florens.lehwalder@stud.tu-darmstadt.de

Leandro Rometsch
Technical University
Darmstadt, Germany
leandro@rometsch.org

Thomas Schneider
Technical University
Darmstadt, Germany
schneider@encrypto.cs.tu-darmstadt.de

ABSTRACT

The problem of finding the k^{th} Ranked Element (KRE) is of particular interest in collaborative studies for financial and medical agencies alike. Many of the applications of KRE deal with sensitive information that needs to be protected. The protocol by Chandran et al. (SECRYPT'22) considers a model where multiple parties hold datasets with many elements and wish to compute the k^{th} element of their joint dataset. In their model, all participating parties interact with a central party in a star network topology. However, they leak some intermediate information to the central party.

In this work we use differential privacy techniques to hide this leakage. We use the Laplace mechanism for introducing differentially private noise and use sigmoid scaling to improve the accuracy of the protocol. We show that our modifications have only a small impact on the accuracy. We also give experimental performance results and compare our work to the previous works on KRE.

CCS CONCEPTS

• **Security and privacy → Cryptography**.

KEYWORDS

Secure multi-party computation; Differential privacy; k^{th} ranked element

ACM Reference Format:
Gowri R Chandran, Philipp-Florens Lehwalder, Leandro Rometsch, and Thomas Schneider. 2023. Poster: Secure and Differentially Private k^{th} Ranked Element. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security (CCS '23), November 26–30, 2023, Copenhagen, Denmark*. ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3576915.3624392

1 INTRODUCTION

As more and more services are shifting to online platforms, the data generated and stored by individual clients are increasing exponentially. This data, is often sensitive which requires secure processing by mutually distrusting entities, an area called privacy-preserving data analysis.

Among many different functions used in data analysis, finding the k^{th} ranked element (KRE) is a very important function. KRE allows the parties to learn the k^{th} element of a sorted set. The KRE is a widely used analytical function in financial and medical fields. It helps to measure the tendencies of data distribution, for instance the measure of central tendency, mean, or the maxima or minima of a database. For example, the median function is used by insurance companies to evaluate the median of medical claims by customers to determine future offers and expenses. In many applications the inputs of the computation are sensitive and need to be protected, while still providing accurate results. This leads to the requirement of techniques such as Secure Multi-Party Computation (MPC) which can be used to compute the KRE function securely.

Although a generic MPC protocol [2, 7] can be used to compute the KRE, these protocols have quadratic complexity in the number of parties. A specific protocol for finding the KRE among n parties was first given in [1]. This protocol implements a binary search algorithm which requires a non-constant number of sequential rounds. Later, in [10], a constant round protocol for finding the KRE in a star network topology was proposed. The star topology is a natural setting in applications like the one mentioned above where the insurance agency can communicate with all hospitals, but the hospitals don't need to communicate with each other. This protocol however traded-off communication for runtime. Even though the constant rounds resulted in efficient runtime, the communication complexity is increased drastically as the n parties need to conduct n^2 comparisons. Another protocol for computing the KRE in a star network topology was proposed by Chandran et al. in [4]. Their protocol was based on the protocol of [1], where a binary search was conducted between the parties, but in a star network topology, reducing the communication cost of the protocol. In this protocol each party holds a database with multiple inputs and then securely compute the KRE of the joint database. However, this protocol leaks some intermediate information to the central party, which, even though doesn't leak any individual party's information, reveals the overall distribution of the elements in the union of the databases.

We extend the protocol of [4] to reduce the intermediate leakage by using differential privacy techniques. Differential privacy (DP) [5] has been used in several works [8, 9] to reduce the leakage by revealing a differentially private value. We add noise to the intermediate leakage in the protocol of [4] and show that the resulting leakage

CCS '23, November 26–30, 2023, Copenhagen, Denmark

ACM ISBN 979-8-4007-0050-7/23/11.
https://doi.org/10.1145/3576915.3624392

is differentially private and thus does not let any adversary learn anything about the database distribution. For a practical application such as the one mentioned above, the result of the computation need not be 100 percent precise and an approximate solution suffices.

In several works [3, 6], DP techniques are combined with MPC protocols to offer additional privacy guarantees. While these protocols focus on output privacy, our work addresses the leakage in the specific KRE protocol of [4], while also providing high accuracy.

2 DIFFERENTIALLY PRIVATE KRE

We extend the KRE protocol of [4], into a differentially private variant where the intermediate leakage of their protocol is reduced by using DP techniques. Our protocol also runs a binary search on the joint database to find the KRE.

Leakage analysis. In [4], Chandran et al. propose a KRE protocol in a star network topology, where a central party communicates with all the other parties. Although this reduces the communication complexity of the protocol, some intermediate values are leaked to the central party. More specifically, in each round, the central party learns the number of elements in the union of the databases that are less than and greater than a particular value. This lets an adversary learn the distribution of the elements in the joint database. To eliminate this leakage, we introduce noise sampled from the Laplace distribution, which is proven to be ϵ−differentially private [5], where ϵ is the privacy parameter. Since, in our protocol, the noise is generated locally by each party, the central party cannot differentiate between the noise levels added. The noise generation technique is discussed in detail below, where we can see that even though the noise is data dependent, it does not leak anything to the central party.

Protocol description. We give a brief description of our protocol Π_{KRE} (Figure 1). Let $n \geq 2$ parties be participating in the computation, with each party having database D_i, for $i \in [1, n]$ as its input. Our protocol is secure against a semi-honest adversary which can corrupt at most $n-1$ parties. We assume, without loss of generality, that party P_1 is the central party. Let the elements in $\bigcup D_i$ be in range $[a, b]$. The rank k, the range $[a, b]$, and the size of the joint database N is publicly known to all parties (the size of the individual database, $|D_i|$, is private). The parties run a binary search algorithm to search for the k^{th} ranked element in the union of their database. In our protocol, each party adds noise to the number of elements less (l_i) or greater than (g_i) m and sends the noisy counts to the central party. This way, after the decryption, P_1 learns a noisy value for both the number of elements less and greater than m, therefore preventing an adversary from recognising the actual data distribution. Figure 1 gives a detailed description of our protocol. The step for adding the differentially private noise is highlighted in gray.

Noise analysis. Note that, as fresh noise is added in each round of the protocol, the noise doesn't accumulate to form a large noise towards the last rounds of the protocol. Moreover, we use Sigmoid scaling for the Laplace distribution so that the noise added depends on the size of the database and the individual parties' data distribution. We now show how using Sigmoid scaling for the Laplace function still preserves the DP guarantees. As the noise is added to the count function, the DP guarantees are proved with respect to this function. The Global Sensitivity (GS) of the count function is 1, as changing one element in the database changes the value of the function by at most 1. Then, the scale of the Laplace distribution should be

$$\lambda = \frac{GS}{\epsilon} = \frac{1}{\epsilon}. \tag{1}$$

We set the Sigmoid function to be dependent on the individual database size and the counter value, such that the amount of noise is reduced as the last rounds approach, giving us more accuracy. Our Sigmoid function is defined as follows:

$$\sigma(x) = \frac{1}{1 + e^{px-5}} \cdot \log N, \tag{2}$$

where p is the compression of the Sigmoid function which has a minimum value of 0 and scales w.r.t the privacy parameter ϵ, and x is computed as $l_i/|D_i|$ or $g_i/|D_i|$ and varies between 0 and 1. Using the Sigmoid function (2) for scaling the Laplace distribution we get

$$\lambda = \sigma(x). \tag{3}$$

Then, from (1) and (3) we get

$$\epsilon = \frac{1 + e^{px-5}}{\log N}. \tag{4}$$

We see that for a database size of 2, the maximum value of ϵ is achieved at approximately 1.48. The recommended value for ϵ

Protocol Π_{KRE}

Parameters: k is the rank, N is the total number of elements in the union of the databases and $[a, b]$ is the range of elements in the union.

Initialization: Each party P_i, for $i \in [1, n]$, sorts the elements in its database in an ascending order.

Key Generation: The parties $P_1, ..., P_n$ engage in a passively secure protocol π_{Gen} to generate a public key pk and their respective shares sk_i of secret key.

Local Computation: Each party P_i, $i \in [1, n]$ does the following:

(1) Compute $m = \lfloor (a + b)/2 \rfloor$.
(2) Compute the number of elements l_i less than m and the number of elements g_i greater than m.
(3) Sample random noise $l_i^n, g_i^n \longleftarrow \mathcal{L}$, where $\mathcal{L}$ is the Laplace distribution, and compute $l_i' = l_i + l_i^n$ and $g_i' = g_i + g_i^n$.
(4) Encrypt the inputs, $c_i = \mathsf{Enc}_{\mathsf{pk}}(l_i')$ and $c_i' = \mathsf{Enc}_{\mathsf{pk}}(g_i')$.

Interactive phase:

(5) The parties $P_2, ..., P_n$ send c_i and c_i' to P_1.
(6) P_1 computes $[L] = \Sigma c_i$ and $[G] = \Sigma c_i'$.
(7) The parties P_i, $i \in [1, n]$, jointly decrypt $[L]$ and $[G]$ and party P_1 obtains L and G.
(8) P_1 does the following comparison:
  (a) If $L < k$ and $G \leq N - k$, then m is the KRE.
  (b) If $L \geq k$, then P_1 sends 0 to all parties and the parties repeat from the local computation phase with $b = m - 1$.
  (c) If $G > N - k$, then P_1 sends 1 to all parties and the parties repeat from the local computation phase with $a = m + 1$.

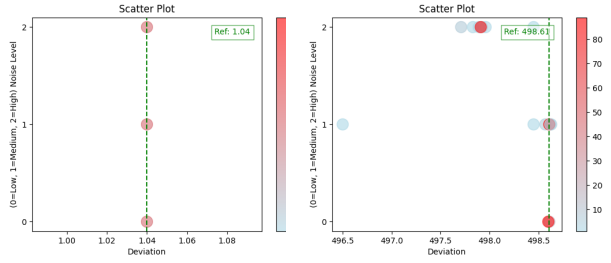
Figure 1: Protocol for secure and differentially private KRE.

is ≤ 1 [5]. We can safely assume that for practical applications, the database size is quite large, giving a very small ϵ value. Therefore, the DP requirements are safely met by using Sigmoid scaling and leakage is hence differentially private.

3 EVALUATION

We implemented our protocol and analyse its performance with regard to accuracy, communication cost, and runtime. We also compare our results with the previous works on the KRE [4, 10].

Accuracy. We analyse the accuracy of our protocol for different noise levels and for different values of k. We run the experiments for 10 parties and a total database size of 10k. In our protocol, the addition of noise increases the number of rounds required to compute the k[th] element. However, since we add new noise at each round and since the scaling of the noise generation also depends on the local computation result of each round, the protocol always terminates and does not go into an infinite loop. We notice that the accuracy of the result depends on the amount of noise added and the value of k. For instance, for $k = 1$ (or $k = n$), i.e., for the minimum (or maximum) element in the database, we obtain high accuracy with all noise levels (see Figure 2a). For $k = n/2$, i.e., the median of the database, we get lower accuracy when adding high noise levels (see Figure 2b).



(a) Minimum ($k = 1$) **(b) Median ($k = n/2$)**

Figure 2: Accuracy of computation of the k[th] element with different noise levels.

In Figure 2a, we see that the accuracy is not affected by the addition of noise, whatever level it is. This is due to the fact that when a binary search algorithm is used, in the case of minimum (or maximum), the protocol selects one branch and mostly stays in that branch till termination. As the noise is added using Sigmoid scaling, the probability of adding noise as we approach the KRE, i.e., minimum (or maximum), is lower, which makes sure that our protocol is highly accurate for these values of k. In contrast, for the median, as the correct value lies at the root of the binary tree, addition of noise causes the protocol to switch branches quite often, making it select a less accurate value for the median. More analysis on the deviation of the median element is left for future work.

Communication and runtime. As our protocol runs for a larger number of rounds due to the noise added, the expected trend in communication and runtime would be that they are increased substantially. We observe that the communication cost required for our protocol is, as expected, more than the leaky original [4]. However, the runtime of our protocol does not increase as much due to the high parallelizability of our protocol. For all the benchmarks we consider low noise level and sigmoid scaling of noise.

For better comparability, we benchmark our protocol in a similar setting to [10] and [4]. We consider a setting with 100 parties, each holding one element and run our benchmarks in a simulated WAN setting with a bandwidth of 100Mbits/s and a latency of 100ms. We use a 2023 Apple MacBook Pro with Apple M2 pro chip and 32GB RAM. Table 1 shows the comparison of the costs for the different KRE protocols[1]. Our protocol has comparable runtime to the KRE protocol of [4], and is especially fast in cases where the parties hold just one element. The communication cost for our protocol is $\approx 93\times$ less than that of [10], but is $\approx 16\times$ more than that of the leaky protocol [4]. Since, in our protocol, the number rounds is directly proportional to the amount of noise added, the runtime and communication are in turn directly proportional to the amount of noise.

Protocol	Leakage	Comm. (MB)	Time (s)
[10]	Non-leaky	60.88	441.00
[4]	Leaky	**0.04**	**7.44**
This work	DP Leakage	0.65	7.53

Table 1: Cost comparison of different KRE protocols.

An advantage of our implementation is that it allows flexibility with the noise generation mechanism, i.e., the Laplace mechanism can be replaced by any other mechanisms such as the Gaussian mechanism. We leave the evaluation of our protocol with different noise generation techniques as future work.

Acknowledgements. This project received funding from the ERC under the European Union's Horizon 2020 research and innovation program (grant agreement No. 850990 PSOTI). It was co-funded by the DFG within SFB 1119 CROSSING/236615297 and GRK 2050 Privacy & Trust/251805230.

REFERENCES

[1] Gagan Aggarwal, Nina Mishra, and Benny Pinkas. 2004. Secure Computation of the k th-Ranked Element. In *EUROCRYPT*.
[2] Donald Beaver, Silvio Micali, and Phillip Rogaway. 1990. The Round Complexity of Secure Protocols (Extended Abstract). In *STOC*.
[3] Jonas Böhler and Florian Kerschbaum. 2020. Secure Multi-party Computation of Differentially Private Median. In *USENIX Security*.
[4] Gowri R. Chandran, Carmit Hazay, Robin Hundt, and Thomas Schneider. 2022. Comparison-based MPC in Star Topology. In *SECRYPT*.
[5] Cynthia Dwork. 2011. A Firm Foundation for Private Data Analysis. In *Commun. ACM*.
[6] Fabienne Eigner, Matteo Maffei, Ivan Pryvalov, Francesca Pampaloni, and Aniket Kate. 2014. Differentially private data aggregation with optimal utility. In *ACSAC*.
[7] Oded Goldreich, Silvio Micali, and Avi Wigderson. 1987. How to Play any Mental Game: A Completeness Theorem for Protocols with Honest Majority. In *STOC*.
[8] Adam Groce, Peter Rindal, and Mike Rosulek. 2019. Cheaper Private Set Intersection via Differentially Private Leakage. In *PoPETs*.
[9] Xi He, Ashwin Machanavajjhala, Cheryl J. Flynn, and Divesh Srivastava. 2017. Composing Differential Privacy and Secure Computation: A Case Study on Scaling Private Record Linkage. In *CCS*.
[10] Anselme Tueno, Florian Kerschbaum, Stefan Katzenbeisser, Yordan Boev, and Mubashir Qureshi. 2020. Secure Computation of the k[th]-Ranked Element in a Star Network. In *FC*.

[1]We note that the values for previous works are taken as mentioned in their works and have not been implemented by ourselves.