Interpretable Encrypted Searchable Neural Networks

Kai Chen[1], Zhongrui Lin[2], Jian Wan[2], and Chungen Xu[1](✉)

[1] School of Science, Nanjing University of Science and Technology, Nanjing, CHN
[2] School of Computer Science and Engineering, NJUST, Nanjing, CHN
{kaichen,zhongruilin,wanjian,xuchung}@njust.edu.cn

Abstract. In cloud security, traditional searchable encryption (SE) requires high computation and communication overhead for dynamic search and update. The clever combination of machine learning (ML) and SE may be a new way to solve this problem. This paper proposes interpretable encrypted searchable neural networks (IESNN) to explore probabilistic query, balanced index tree construction and automatic weight update in an encrypted cloud environment. In IESNN, probabilistic learning is used to obtain search ranking for searchable index, and probabilistic query is performed based on ciphertext index, which reduces the computational complexity of query significantly. Compared to traditional SE, it is proposed that adversarial learning and automatic weight update in response to user's timely query of the latest data set without expensive communication overhead. The proposed IESNN performs better than the previous works, bringing the query complexity closer to $O(\log N)$ and introducing low overhead on computation and communication.

Keywords: Searchable Encryption· Searchable Neural Networks· Probabilistic Learning · Adversarial Learning· Automatic Weight Update

1 Introduction

The frequent and massive disclosure of private data has drawn the growing attention of the public to the *cyberspace security*. Meanwhile, *cloud storage* services are increasingly attracting individuals and enterprises to outsource data into cloud server with the rapid development of *cloud computing*. Unfortunately, outsourcing data into cloud server may reveal the privacy of data [10,14]. In *cloud security*, *searchable encryption* (SE) has received widespread attention as it protects the privacy of outsourced data and prevents sensitive information from leaking [5]. However, traditional SE [1,3,6,8,9,10,12,13,14] requires high computation and communication overhead to enable *dynamic search* and *dynamic update*, which makes SE still unable to satisfy user's experience and requirements of the actual application adequately. Actually, *machine learning* (ML) can provide intelligent and efficient means yet the current popular ML only supports plaintext data training and can not satisfy the special requirements of encrypted cloud data. Therefore, it is necessary to discuss the cross-fusion problem of ML and SE, and introduce intelligence and high-efficiency into SE.

SE has been continuously developed since it was proposed [8], and *multi-keyword ranked search* scheme is recognized as excellent [5]. Cao et al. [1] first discussed privacy-preserving multi-keyword ranked search over encrypted cloud data (MRSE) for single data owner model, and established strict privacy requirements. They first used *asymmetric scalar-product preserving encryption* (ASPE) [12] to obtain the *similarity score* of the query vector and the index vector. In this way, cloud server can retrieve *top-k* documents that are most relevant to the data user's query request. However, since matrix operations require high computation overhead, MRSE is not suitable for practical application scenario. For the purpose of managing the keyword dictionary dynamically and improving system performance, Li et al. [6] proposed efficient multi-keyword ranked query over encrypted data in cloud computing (MKQE) based on MRSE, which owns a low overhead index construction algorithm and a novel trapdoor generation algorithm. However, it still has no major breakthrough in improving search efficiency when the data set is large. To achieve *dynamic search*, Xia et al. [13] provided a secure and dynamic multi-keyword ranked search scheme over encrypted cloud data (EDMRS) to support dynamic operation in SE. For tree-based index structures, search efficiency is improved by the greedy depth-first search (GDFS) algorithm and parallel computing. Regrettably, the search efficiency of ordinary balanced binary tree they used gradually decreases and tends to linear search efficiency when migrating to multiple data owners model with large amount of differential data. Moreover, maintaining such an index tree is not flexible and efficient. Guo et al. [3] discussed secure multi-keyword ranked search over encrypted cloud data for multiple data owners model (MKRS_MO) and designed a heuristic weight generation algorithm based on the relationships among keywords, documents and owners (KDO). They considered the correlation among documents and the impact of documents' quality on search results. Experiments on the real-world data set showed that MKRS_MO is better than the schemes using traditional $TF \times IDF$ keyword weight model [9]. However, the fly in the ointment is that the operations of calculating index similarity in MKRS_MO may lead to "curse of dimensionality", which limits the availability of the system. Last but not least, they ignored the secure solution in *known background model* [1] (*threat model* for measuring the ability of cloud server to evaluate private data and the risk of revealing private information in SE system).

For the first time, this paper proposes *interpretable encrypted searchable neural networks* (IESNN) to explore *intelligent SE*. Based on the *neural network*, we propose *sorting network* and employ *probabilistic learning* to obtain the query ranking for encrypted searchable index. To be specific, firstly it performs a sufficient amount of random queries (obey *uniform distribution*) and then calculates the sum of the inner product of each index vector and all random query vectors. Finally it sorts the index vectors according to the match scores from high to low. Therefore, the probabilistic ranking of the index is close to the ranking in the actual query, which reduces the computational complexity of the query significantly. Moreover, *probabilistic query* with computational complexity close to $O(\log N)$, is used to retrieve *top-k* documents. In order to achieve secure

weight update without revealing private information to "semi-trusted" cloud server [10,14], we propose *searching adversarial network* and *weight update network* in an encrypted cloud environment. Specifically, in order to respond to user's timely query of the latest data set, we employ *adversarial learning* [2] and *optimal game equilibrium* to make the probabilistic ranking of the index close to its popular ranking. Furthermore, we combine *backpropagation neural network* [4] with *discrete Hopfield neural network* [7] to enable *automatic weight update*. It is worth mentioning that the update operations are done in the cloud, which means there is no expensive communication overhead. So we can use IESNN for model training and intelligent system implementation. On the one hand, it introduces intelligence into the SE system, which improves user's experience and reduces system overhead. On the other hand, training data sources for ML can be derived from ciphertext. It means that data mining based on ciphertext analysis can not only obtain results consistent with plaintext analysis but also strengthen the intensity of data privacy protection.

Table 1. Comparison of related works.

Item	MRSE [1]	MKRS_MO [3]	MKQE [6]	EDMRS [13]	IESNN
high-precision query	√	√	√	√	√
privacy-preserving query	√	×	√	√	√
automatic weight update	×	×	×	×	√
high-quality ranked search	×	√	√	×	√
efficient multi-keyword search	√	√	√	√	√
flexible dynamic maintenance	×	×	×	×	√

Our main contributions are summarized as follows:

(1) Towards *intelligent SE* by combining popular ML with traditional SE effectively;
(2) We employ *probabilistic learning* method to achieve *maximum likelihood searching* and improve search efficiency significantly;
(3) We use IESNN to implement *flexible dynamic operation and maintenance* in an encrypted cloud environment.

The remainder of this paper is organized as follows: Section 2 describes the SE model. Section 3 describes the details of IESNN and its performance tests. Section 4 discusses our solution and its implications.

2 Searchable Encryption Model

2.1 System Model

The system model proposed in this paper consists of three parties, is depicted in Fig. 1, and the specific description is as follows:

Data owners(*DO*): *DO* are responsible for building searchable index and original IESNN, encrypting the data and sending them to cloud server.

Data users(DU): DU are consumers of cloud services. Once the license is granted, they can retrieve the encrypted cloud data.

Cloud server(CS): CS is considered"semi-trusted"in SE [10,14]. It provides cloud service, including running authorized access controls, performing searches for encrypted cloud data based on query requests, returning *top-k* documents to DU and enabling dynamic operation and maintenance with IESNN.

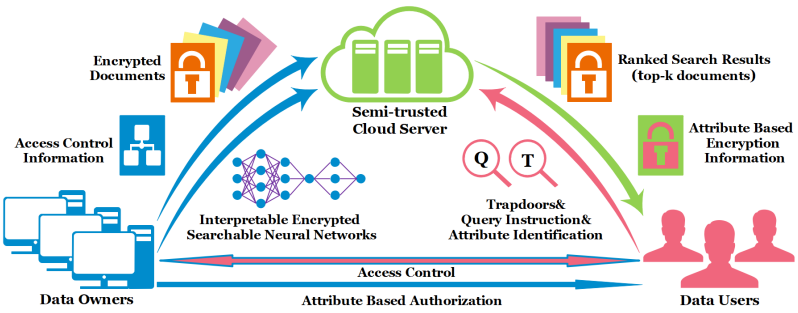

Fig. 1. The basic architecture of searchable encryption system

2.2 System Framework

***Setup*:** Based on privacy requirements in *known background model* [1], DO_i determines the size N_i of dictionary D_i, the number U_i of pseudo-keyword, sets the parameter $V_i = U_i + N_i$. For all data owners $DO = \{DO_1,\ldots,DO_m\}$, we have $V = \{V_1,\ldots,V_m\}$, $U = \{U_1,\ldots,U_m\}$, $N = \{N_1,\ldots,N_m\}$.

***KeyGen*(V):** DO generate secret key $SK = \{SK_1,\ldots,SK_s\}$, where $SK_i = \{S_i, M_{i,1}, M_{i,2}\}$, $M_{i,1}$ and $M_{i,2}$ are two invertible matrices with the dimension $V_i \times V_i$ and S_i is a random V_i-length vector.

***Extended-KeyGen*(SK_i, Z_i):** For *dynamic search* [6], if Z_i new keywords are added into the i-th dictionary D_i, DO_i generates a new $SK_i' = \{S_i', M_{i,1}', M_{i,2}'\}$, two invertible matrices $M_{i,1}'$ and $M_{i,2}'$ with the dimension $(V_i + Z_i) \times (V_i + Z_i)$, and a new $(V_i + Z_i)$-length vector S_i'.

***BuildIndex*(I, SK):** In order to reduce the possibility that "semi-trusted" cloud server [10,14] evaluates the private data successfully, DO first build searchable indexes for documents and obtain the weighted index vectors, and then fill index vectors with random pseudo-keywords (obey *Gaussian distribution*) and obtain secure index vectors with high privacy protection strength [1]. Finally they use secure index vectors to build IESNN ($\mathcal{I}$) and send $\mathcal{I}$ to CS (specific example: DO_i "splits" index vector I_i into two random vectors $\{I_{i,1}, I_{i,2}\}$. Specifically, if $S_i[t] = 0$, $I_{i,1}[t] = I_{i,2}[t] = I_i[t]$; else if $S_i[t] = 1$, $I_{i,1}[t]$ is a random value, $I_{i,2}[t] = I_i[t] - I_{i,1}[t]$. DO_i encrypts I_i as $\widetilde{I}_i = \{M_{i,1}^T I_{i,1}, M_{i,2}^T I_{i,2}\}$ with SK_i).

***Trapdoor*(Q, SK):** *DU* send query request (query keywords and k) to *DO*. *DO* generate query $Q = \{Q_1,\ldots,Q_m\}$ (where Q_i is a weighted vector with dimension V_i) and calculate the trapdoor $T = \{T_1,\ldots,T_m\}$ using SK and send T to *DU* (specific example: DO_i "splits" query vector Q_i into two random vectors $\{Q_{i,1}, Q_{i,2}\}$. Specifically, if $S_i[t] = 0$, $Q_{i,1}[t]$ is a random value, and $Q_{i,2}[t] = Q_i[t] - Q_{i,1}[t]$; else if $S_i[t] = 1$, $Q_{i,1}[t] = Q_{i,2}[t] = Q_i[t]$. Finally, DO_i encrypts Q_i as $T_i = \{M_{i,1}^{-1}Q_{i,1}, M_{i,2}^{-1}Q_{i,2}\}$ with SK_i).

***Query*$(\mathcal{I}, T, k)$:** *DU* send trapdoors, query instruction and attribute identification to *CS*. *CS* performs searches based on the query, and returns *top-k* documents to *DU*.

3 Interpretable Encrypted Searchable Neural Networks

3.1 Maximum Likelihood Searching

We employ *inner product similarity* [11] to quantitatively evaluate the effective similarity between the query vector and the index vector. As illustrated in Fig. 2 (for an intuitive understanding, it shows the unencrypted network), in *sorting network*, it performs a sufficient amount of random queries (obey *uniform distribution*: $X \sim U(-\sigma\sqrt{3}, \sigma\sqrt{3})$, that is $f(x) = \frac{1}{2\sigma\sqrt{3}}, x \in [-\sigma\sqrt{3}, \sigma\sqrt{3}]$), and then calculates the sum of the inner product $\sum_{j=1}^{m} I_i^T \cdot Q_j$ of each index vector and all random query vectors with formula 1. Finally it sorts the index vectors according to the match scores from high to low. Therefore, the index ranking obtained by *probabilistic learning* is close to the ranking in the actual query.

$$Score = \widetilde{I}_i \cdot T_i = \{M_{i,1}^T I_{i,1}, M_{i,2}^T I_{i,2}\} \cdot \{M_{i,1}^{-1}Q_{i,1}, M_{i,2}^{-1}Q_{i,2}\} = I_i^T \cdot Q_i \qquad (1)$$

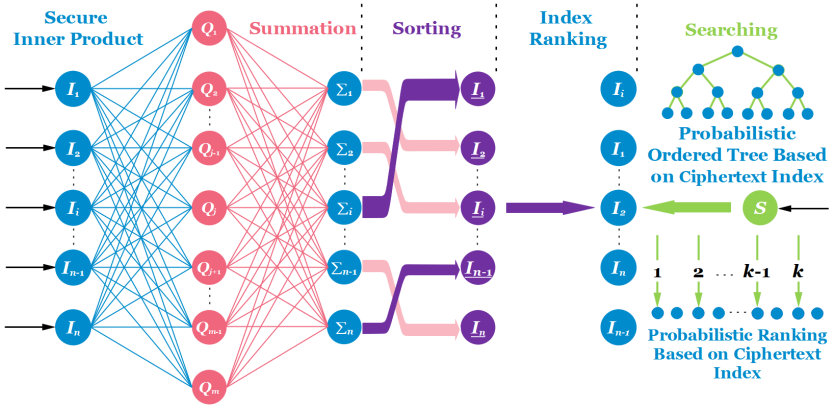

Fig. 2. Sorting network

We implement the proposed scheme using Python in Windows 10 operation system with Intel Core i5 Processor 2.40GHz and test its efficiency on a real-world document set (IEEE INFOCOM publications, including 400 papers and 2,000 keywords). The *probabilistic query* algorithm based on the probabilistic ranking of encrypted searchable index brings the query complexity closer to $O(\log N)$. As shown in Fig. 3, when retrieving the same number of *top-k* documents, *probabilistic query* performs better than the related works that based on tree search [3,13] and matrix operation [1,6]. As the ordered feature of the balanced binary tree is not guaranteed in the index tree and the query based on matrix operation needs to traverse all indexes to retrieve *top-k* documents, the number of retrieved indexes is far more than the number of retrieved documents.

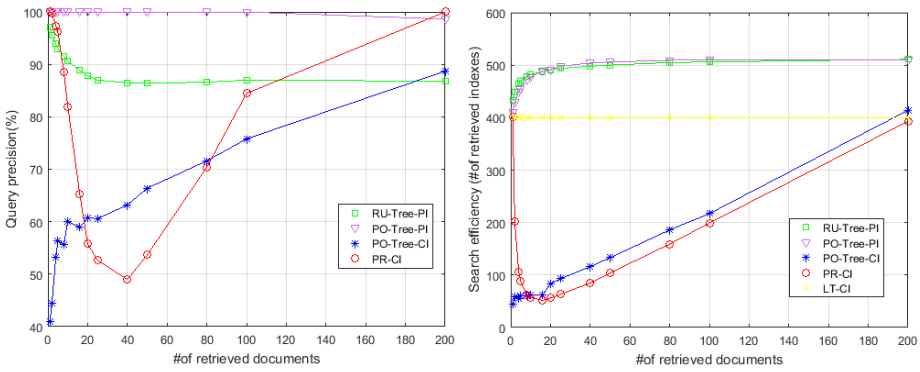

Fig. 3. Performance testing of multiple query algorithms. {The *query precision* and *search efficiency* for different numbers of retrieved documents with the same document collection (400) and dictionary (2,000). It requires an average of 100 experimental results to measure performance of the following subjects: random unordered tree based on plaintext index(RU-Tree-PI) [3,13], probabilistic ordered tree based on plaintext index(PO-Tree-PI), probabilistic ordered tree based on ciphertext index(PO-Tree-CI), probabilistic ranking based on ciphertext index(PR-CI), linear traversal based on ciphertext index(LT-CI) [1,6]. The number of retrieved *top-k* documents is the factor of 400: $k = 1, 2, 4, 5, 8, 10, 16, 20, 25, 40, 50, 80, 100, 200$. Note: the nodes of the tree are also included in the number of retrieval indexes. According to the experimental results, *probabilistic query* can significantly improve the search efficiency. When k takes a specific interval value, the query precision is high or low. It is because that the probabilistic ranking of the index vector is not strictly ordered, and the query is random. Therefore, when the query vector is very "unpopular", the query precision will become lower, and when the query vector is "popular", the query precision and search efficiency will perform well.}

3.2 Adversarial Learning

Adversarial network works when the probabilistic ranking of the index deviates from the index ranking in the actual query result. As shown in Fig. 4, it employs *optimal game equilibrium* to make the probabilistic ranking of the index close to its popular ranking (described by formula 2, $p_i(x)$ and $p_q(y)$ are the *probability distributions* of the index ranking and query result, respectively).

$$\min_{S}\max_{A} V(A,S) = \mathbb{E}_{x\sim p_i(x)}[\log A(x)] + \mathbb{E}_{y\sim p_q(y)}[\log(1-A(S(y)))] \qquad (2)$$

Inspired by *generative adversarial networks* (GAN) [2] and *self-attention generative adversarial networks* (SAGAN) [15] but different from GAN and SAGAN, *searching adversarial networks* (SAN) do not require complex gradient calculations and extensive iterative training. As a matter of fact, SAN only require simple residual calculations and index sorting floating steps. Specifically, after completing the query, the ranked search result list is feedback to *adversarial network* in SAN. *Adversarial network* calculates the residual before and after the weight change of the index corresponding to *top-k* documents, and calculates the relative floating of the index ranking of the feedback result(i.e. new index ranking) and the original index ranking. Finally, SAN send the results of the calculation (the residual of the weights) and the index ranking changes to the *weight update network* as a target for index update (see Fig. 5 for details).

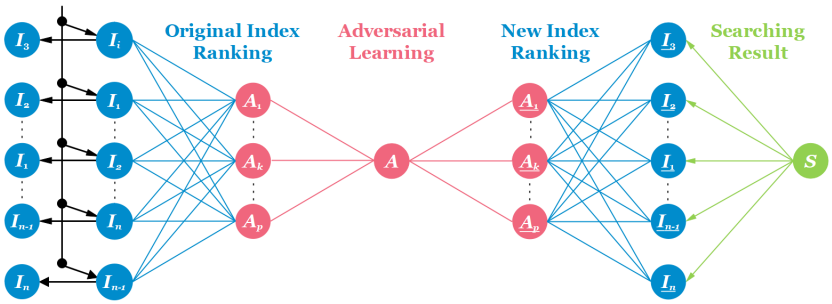

Fig. 4. Searching adversarial networks

3.3 Automatic Weight Update

As illustrated in Fig. 5, in order to achieve *automatic weight update* and respond to users' queries for the latest data sets in a timely manner, *weight update network* (WUN) combines *backpropagation neural network* (BPNN) [4] with *discrete Hopfield neural network* (DHNN) [7]. In WUN, the update of index weights uses vector and matrix operations to approximate the actual increments, which has the characteristics of local homomorphism for ciphertext operations and plaintext operations. For instance, considering index vector $I_\alpha = [\alpha_1, \ldots, \alpha_n]^T$, query vector $Q_\gamma = [\gamma_1, \ldots, \gamma_n]^T$, and two invertible matrices $M = (a_{ij})_{n\times n}$ and $M^{-1} = (b_{ij})_{n\times n}$. The *update principle* of ciphertext index is as follows:

Matrix and vector multiplication: $I_\alpha^T M = [\sum_{i=1}^{n} \alpha_i a_{i1}, \ldots, \sum_{i=1}^{n} \alpha_i a_{in}], M^{-1}Q_\gamma = [\sum_{j=1}^{n} b_{1j}\gamma_j, \ldots, \sum_{j=1}^{n} b_{nj}\gamma_j]$;*Secure inner product calculation*: $I_\alpha^T M \cdot M^{-1}Q_\gamma = I_\alpha^T \cdot Q_\gamma = \sum_{i=1}^{n} \alpha_i\gamma_i$; *Index vector update*: $(I_\alpha^T + \Delta I_\alpha^T)M = [\sum_{i=1}^{n}(\alpha_i + \Delta\alpha_i)a_{i1}, \ldots, \sum_{i=1}^{n}(\alpha_i +$

$\Delta\alpha_i)a_{in}] \approx [\sum_{i=1}^{n} \alpha_i a_{i1} + \Delta\beta_1, \ldots, \sum_{i=1}^{n} \alpha_i a_{in} + \Delta\beta_n] = I_\alpha^T M + \Delta I_\beta^T M$;*Inner product approximation*: $(I_\alpha^T + \Delta I_\alpha^T)M \cdot M^{-1}Q_\gamma \approx (I_\alpha^T M + \Delta I_\beta^T M) \cdot M^{-1}Q_\gamma = I_\alpha^T \cdot Q_\gamma + \Delta I_\beta^T \cdot Q_\gamma = \sum_{i=1}^{n} \alpha_i\gamma_i + \sum_{i=1}^{n} \Delta\beta_i\gamma_i$.

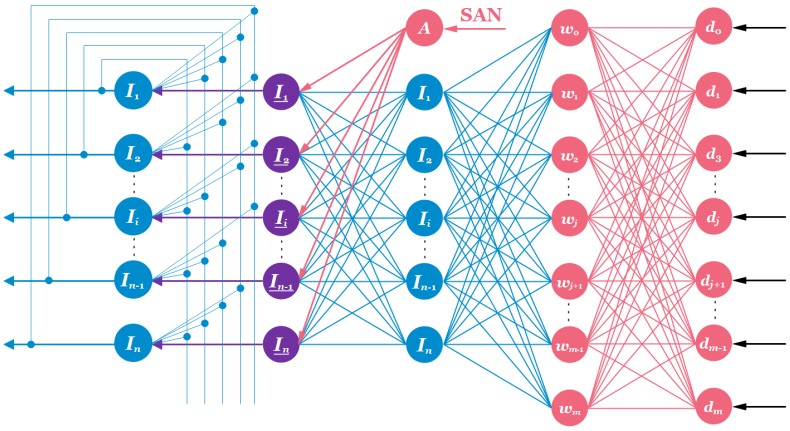

Fig. 5. Weight update network

***Asynchronous work mode of WUN*:** The update task from SAN to WUN is only updating the weight of an index, while other indexes still retain their original weight. i.e.

$$I_j(t+1) = \begin{cases} sgn[net_j(t)], & j = i \\ I_j(t), & j \neq i \end{cases}, I_j(t+1) = \begin{cases} satlins[net_j(t)], & j = i \\ I_j(t), & j \neq i \end{cases} \quad (3)$$

***Synchronous work mode of WUN*:** The synchronous work mode is parallel, i.e. the weights of all indexes are all changed in one update. The adjustment of the weight is determined according to the current input value. The weight update is complete and the weight of an index continues to be used for the next update. When the weight of each index is stabilized, the work ends.

$$\begin{cases} I_j(t+1) = sgn[net_j(t)], & j = 1, 2, \ldots, n \\ I_j(t+1) = satlins[net_j(t)], & j = 1, 2, \ldots, n \end{cases} \quad (4)$$

When updating an index, the schemes [3,13] employ tree-based index need to update the index vector itself (leaf node of index tree) and its corresponding other data (parent node of leaf node). Moreover, in order to achieve *dynamic search*, the current schemes [1,3,6,9,13] need to download the ciphertext index from the cloud, update its plaintext after local decryption, and finally upload the new ciphertext index to the cloud. In comparison, our solution only needs to update the index vector in the cloud with touching a smaller amount of data and introduce low overhead on computation and communication.

3.4 Overall Operation and Maintenance of IESNN

As shown in Fig. 6, IESNN consist of *sorting net*, *adversarial net*, *searching net* and *weight update net*. Except that the initial index weight needs to be generated by data owners, the rest of automatic update operations ("add, delete, change and investigate" operations of index) are all completed in an encrypted cloud environment. The system forms a "query-learning-update-learning-query" *self-attention* [15] loop and an "automatic operation and maintenance" mechanism. *Dynamic operation and maintenance* of SE system are almost entirely done in cloud server. On the one hand, implementing *dynamic operation and maintenance* in an encrypted cloud environment not only improves the usability and flexibility of SE system, but also enhances the strength of privacy protection. On the other hand, when it is necessary to update the index in cloud server, compared with traditional SE [1,3,6,9,13], our solution eliminates the need to rebuild the index locally and upload a new index to cover the old index stored in the cloud, which introduces low overhead on computation, communication and local storage.

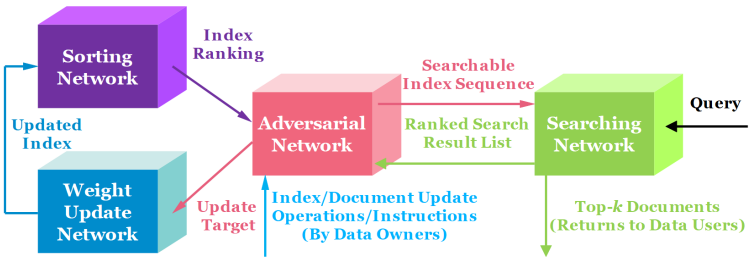

Fig. 6. The overall composition of IESNN

4 Discussion

In this paper, we discuss the cross-fusion problem of ML and SE, and propose IESNN. We creatively combine popular ML with traditional SE, which is committed to exploring *intelligent SE*. We employ *probabilistic learning* method to generate *sorting network* that is trained by a sufficient amount of random queries, which makes a contribution to achieve *maximum likelihood searching* and bring the query complexity closer to $O(\log N)$. It means that exploiting ML to optimize the query is effective in an *uncertain system*, even better than special construction methods. Obviously, traditional query algorithms based on matrix operations and tree searching are not optimistic in big data environments because high dimensional data processing can lead to "curse of dimensionality" and even system crashes. Implementing *flexible dynamic operation and maintenance* in an encrypted cloud environment with IESNN that reduces communication overhead, protects data privacy and leverages cloud computing well.

Acknowledgment

This work was supported by "the Fundamental Research Funds for the Central Universities" (No. 30918012204) and "the National Undergraduate Training Program for Innovation and Entrepreneurship" (Item number: 201810288061). NJUST graduate Scientific Research Training of 'Hundred, Thousand and Ten Thousand' Project "*Research on Intelligent Searchable Encryption Technology*".

References

1. Cao, N., Wang, C., Li, M., Ren, K., Lou, W.: Privacy-preserving multi-keyword ranked search over encrypted cloud data. IEEE Trans. Parallel Distrib. Syst. **25**(1), 222–233 (2014)
2. Goodfellow, I.J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A.C., Bengio, Y.: Generative adversarial networks. CoRR **abs/1406.2661** (2014)
3. Guo, Z., Zhang, H., Sun, C., Wen, Q., Li, W.: Secure multi-keyword ranked search over encrypted cloud data for multiple data owners. Journal of Systems and Software **137**(3), 380–395 (2018)
4. Hinton, G.E., Osindero, S., Welling, M., Teh, Y.W.: Unsupervised discovery of nonlinear structure using contrastive backpropagation. Cognitive Science **30**(4), 725–731 (2006)
5. Kumar, D.V.N.S., Thilagam, P.S.: Approaches and challenges of privacy preserving search over encrypted data. Inf. Syst. **81**, 63–81 (2019)
6. Li, R., Xu, Z., Kang, W., Yow, K., Xu, C.: Efficient multi-keyword ranked query over encrypted data in cloud computing. Future Generation Comp. Syst. **30**(1), 179–190 (2014)
7. Park, J.H., Kim, Y.S., Eom, I.K., Lee, K.Y.: Economic load dispatch for piecewise quadratic cost function using hopfield neural network. IEEE Trans. Power Syst. **8**(3), 1030–1038 (1993)
8. Song, D.X., Wagner, D.A., Perrig, A.: Practical techniques for searches on encrypted data. In: IEEE S & P 2000. pp. 44–55. IEEE Computer Society (2000)
9. Sun, W., Wang, B., Cao, N., Li, M., Lou, W., Hou, Y.T., Li, H.: Verifiable privacy-preserving multi-keyword text search in the cloud supporting similarity-based ranking. IEEE Trans. Parallel Distrib. Syst. **25**(11), 3025–3035 (2014)
10. Wang, C., Wang, Q., Ren, K., Lou, W.: Privacy-preserving public auditing for data storage security in cloud computing. In: INFOCOM 2010. pp. 525–533 (2010)
11. Witten, I.H., Moffat, A., Bell, T.C.: Managing Gigabytes: Compressing and Indexing Documents and Images, Second Edition. Morgan Kaufmann (1999)
12. Wong, W.K., Cheung, D.W., Kao, B., Mamoulis, N.: Secure knn computation on encrypted databases. In: ACM SIGMOD 2009. pp. 139–152. ACM (2009)
13. Xia, Z., Wang, X., Sun, X., Wang, Q.: A secure and dynamic multi-keyword ranked search scheme over encrypted cloud data. IEEE Trans. Parallel Distrib. Syst. **27**(2), 340–352 (2016)
14. Yu, S., Wang, C., Ren, K., Lou, W.: Achieving secure, scalable, and fine-grained data access control in cloud computing. In: INFOCOM 2010. pp. 534–542. IEEE (2010)
15. Zhang, H., Goodfellow, I.J., Metaxas, D.N., Odena, A.: Self-attention generative adversarial networks. In: ICML 2019. pp. 7354–7363. PMLR (2019)