J Inf Process Syst, Vol.20, No.4, pp.432~441, August 2024
https://doi.org/10.3745/JIPS.03.0199
ISSN 2092-805X (Electronic)

JOURNAL OF INFORMATION PROCESSING SYSTEMS JIPS

Cloud-based Full Homomorphic Encryption Algorithm by Gene Matching

Pingping Li[1,*] and Feng Zhang[2]

Abstract
To improve the security of gene information and the accuracy of matching, this paper designs a homomorphic encryption algorithm for gene matching based on cloud computing environment. Firstly, the gene sequences of cloud files entered by users are collected, which are converted into binary code by binary function, so that the encrypted text is obviously different from the original text. After that, the binary code of genes in the database is compared with the generated code to complete gene matching. Experimental analysis indicates that when the number of fragments in a 1 GB gene file is 65, the minimum encryption time of the algorithm is 80.13 ms. Aside from that, the gene matching time and energy consumption of this algorithm are the least, which are 85.69 ms and 237.89 J, respectively.

Keywords
Binary System, Cloud Computing, Full Homomorphic Encryption Algorithm, Gene Matching

1. Introduction

With the reduction of “cloud” network computing costs, cloud computing technology is becoming more mature. In this context, users do not carry their own biological information genetic data, but instead store more biological information genetic data and services on cloud servers. However, using this method carries certain risks. Bioinformatics genetic data can be directly exposed to external attackers and staff of "cloud" network service providers, so the user's private biological information may be leaked or abused [1]. Currently, to improve the reliability of biological data stored in the manager, most servers use encryption schemes to store and manage biological data. However, with the development of science and technology and the social progress, the amount of data generated every day is increasing, and the information related to people's lives is exploding [2, 3].

The structure of this paper is as follows: The first part is the introduction and the contents of references; the second part is the research methods and materials, mainly including the homomorphic encryption algorithm and the principle of gene information coding and matching. The third part is the experiment part, which mainly compares the proposed method with the traditional encryption algorithm. The last part is the conclusion, which mainly summarizes the results of the experiment.

Manuscript received December 21, 2022; first revision February 16, 2023; accepted March 11, 2023.
* **Corresponding Author:** Pingping Li (lipingping10489@163.com)
[1] Institute of Digital Information Technology, Zhejiang Technical Institute of Economics, Hangzhou, China (lipingping10489@163.com)
[2] Applied Engineering College, Zhejiang Business College, Hangzhou, China (13867193665@163.com)

2. Materials and Methods

2.1 Full Homomorphic Encryption Algorithm

To improve the security of gene matching based on cloud computing, this paper uses homomorphic encryption algorithm to encrypt cloud-based genes. Firstly, the encryption principle and steps of the homomorphic encryption algorithm are analyzed in detail. After that, the overall situation of homomorphic encryption is analyzed. Finally, homomorphic encryption is applied to gene matching based on cloud computing. The detailed analysis results are as follows.

2.1.1 The principle of full homomorphic encryption algorithm

Assuming that the encryption function is EK; The decryption function is DK; The operation is α, and the plaintext is $M = (m_0, m_1, m_2, \dots m_i)$, then the calculation formula for the full homomorphic encryption algorithm is:

$$\alpha\left(DK\left(M(m_0, m_1, m_2, \dots m_i)\right)\right) = D\left(\alpha\left(M(m_0, m_1, m_2, \dots m_i)\right)\right). \quad (1)$$

Meanwhile, a homomorphic encryption scheme is mainly composed of four algorithms. The four algorithms are the key part, the encryption part, the decryption part and the additional evaluation part, respectively[4,5]. The detailed analysis results of the four parts are as follows:

- Key part (Key): The public key pk and the private key $sk$are generated according to the given parameter γ.
- Encryption part (Enc): Encrypt the plaintext M with the public key pk to obtain the encrypted ciphertext C.
- Decryption part (Dec): Decrypt the ciphertext $C$with the private key to obtain plaintext M.
- Evaluate: Outputs t input circuits C (composed of mod2 addition gates and multiplication gates) and public key pk, as well as ciphertext $c$corresponding to plaintext [6,7]. The extra evaluation section can be used to formulate the output of the additional evaluation section, and the output formula is $Evaluate(pk, C, c)$.

2.1.2 The step full homomorphic encryption algorithm

Before using the full homomorphic encryption algorithm for encryption, some homomorphic somewhat scheme needs to be constructed to change the modulo 2 operation to the modulo 4 operation, so that the 2-bit ciphertext can be encrypted at one time[8]. Let λ be the security parameter, and the specific construction process of the full homomorphic encryption algorithm is as follows.

- KeyGen (λ): Aη bit key p is generated by the security parameter λ.
- $Encrypt(sk, m)$: Encryption $m = \{00,01,10,11\}$ yields $c = m + 4r + pq$. Wherein, r is the randomly generated integer of ρ;ρrepresents the noise length; $m$is a randomly generated γ-bit integer in the encryption process.
- $Desrypt(sk, c)$: $m = (c \bmod p) \bmod 4$.

The value of $c \bmod p$is noise, that is, only when $m + 4r < \frac{p}{2}$, $c \bmod p = m + 4r$, then the decryption result obtained is correct. According to the security parameters, as long as it is a "fresh" ciphertext, the

plaintext obtained after decryption by the full homomorphic encryption algorithm is always established [9]. In the above process, KeyGen (λ) represents the key generation algorithm; $Encrypt(sk, m)$ 22 represents the encryption algorithm; $Desrypt(sk, c)$ represents the decryption algorithm. To analyze the homomorphism of the full homomorphic encryption algorithm, the homomorphism of the full homomorphic encryption algorithm is verified. The specific verification process is as follows:

$$c_1 = m_1 + 4r_1 + pq_1 \tag{2}$$

$$c_2 = m_2 + 4r_2 + pq_2 \tag{3}$$

$$[(c_1 + c_2)\mathrm{mod}p]\mathrm{mod}4 = [m_1 + m_2 + 4(r_1 + r_2)]\mathrm{mod}4 = m_1 + m_2 \tag{4}$$

$$[(c_1 * c_2)\mathrm{mod}p]\mathrm{mod}4 = [m_1 + m_2 + 4(m_2 r_1 + 4r_1 r_2 + m_1 r_2)]\mathrm{mod}4 = m_1 m_2 \tag{5}$$

The above formula indicates that the ciphertext in the process of full homomorphic encryption algorithm is "fresh," so ciphertext addition and multiplication satisfy the homomorphism. However, the noise generated in the continuous operation will become larger [10].

The steps of the full homomorphic encryption algorithm are as follows:

- KeyGen (λ): A η-bit private key is randomly generated during key generation p. Let $x_0 = pq_0$, and x_0 be an odd number;$r_p(x_0)$ can be divisible by 4. According to the some scheme, $2\sqrt{\tau}$ ciphertexts generated by 0 encryption is generated $b\{0,1\}, 1 \le i \le \sqrt{\tau}$, , and $x_{i,b} = pq_{i,b} + 4r_{i,b}$. The final public key size is $2\sqrt{\tau}$, and $pk = \langle x_0, x_{1,0}, x_{1,1}, x_{2,0}, x_{2,1}, \dots, x_{\sqrt{\tau},0}, x_{\sqrt{\tau},1} \rangle$.
- $Encrypt(pk, m)$: τ dimensional vector $b = \langle b_{i,j} \rangle (1 \le i, j \le \sqrt{\tau}, b_{i,j} \in \{0,1\})$, and the fixed large prime number q is randomly generated (the number of bits in p is greater than the number of digits in q). Plaintext $m \in \{00,01,10,11\}$, the expression of ciphertext c is:

$$c = \left(m + 4r + p + 4Q \sum_{1 \le i,j \le \sqrt{\tau}} b_{i,j} x_{i,Q} x_{j,1} \right) mod x_0 \tag{6}$$

- $Dexrypt(sk, c)$: Decrypting the ciphertext, the resulting plaintext is $m = (x\mathrm{mod}p)\mathrm{mod}4$;the purpose of modularizing x_0 in the encryption process is to reduce the ciphertext size.

2.2 Cloud Computing based Full Homomorphic Encryption Algorithm by Gene Matching

The overall flow of cloud computing based full homomorphic encryption algorithm by gene matching is as shown in Fig. 1.

In Fig. 1, the input of the full homomorphic encryption algorithm by gene matching is divided into two parts. The first part is to collect the gene sequence input by the user, and the other part is to read the gene sequence stored in the cloud file. The two parts correspond to the gene sequence to be searched and the database gene sequence, respectively. The default search gene is a specific gene with a certain length. If the specific length range is 50, the length of the gene to be searched is also within this length range. The gene to be searched is encoded and encrypted, which is decrypted after being transmitted to the terminal. The degree of agreement between the decrypted results and the genes in the gene pool is analyzed to find

the most compatible genes for gene matching. The output shows the final matching result. The various parts of the algorithm flow are analyzed in following detail.

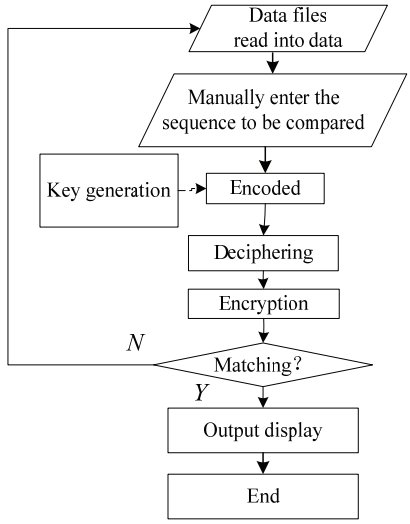

Fig. 1. Flow chart of full homomorphic encryption algorithm by gene matching.

3. Results and Discuss

3.1 Encryption Decryption Effect

The MATLAB program is applied to collect the gene sequence input by the user and read the code book generated by the gene sequence stored in the cloud file.Set the safety parameters of the system $\gamma = 4$, $\eta = 64$, $\rho = 4$,$\tau = 1028$. The experiment is conducted on the VMware Workstation 9.0.0 build-812388 virtual machine platform with Ubuntu 12.10 as the operating system. This platform has a 20.3GB hard drive and 772MB of memory.The dataset used in this experiment is from the open data provided by the biological information database of DNADataBank of Japan, European Bioinformatics Institute, and National Center for Biotechnology Information. It mainly consists ofserialized data of DNA molecules, ranging in length from 1,950 billion to 595 million. The personal biological information gene data segment is selected as the basic dataset of the experiment. Select fragments as experimental data.

Through analysis, the XOR operation is performed on the possible combinations of the four bases to obtain a truth. By observing Table 1, it can be seen that the XOR operation results of A and T, C and G are all 1, while any other combination cannot guarantee that the result is all 1. Demonic simulation results of encryption and decryption are shown in Fig. 2.

Table 1. Redesigned coding format

Base abbreviation	Binary code
A	00
C	10
G	01
T	11

Fig. 2 demonstrates that after the results of the encryption process are summarized, the obtained ciphertext is 0110001100111101001001000010101101111101011000100101100000001. The decrypted plaintext can be obtained by the same reason. Compared with the original plaintext binary sequence and the decrypted binary sequence, the two are almost identical, which indicates that the encryption and decryption process of the algorithm is more accurate. Meanwhile, from the security analysis in the process of encryption and decryption, it is found that only the recipient and the sender have the DNA codebook under the algorithm. Therefore, the attacker cannot select the key from the password book, which guarantees the reliability of the key, and the attacker's chance of brute force cracking is small.

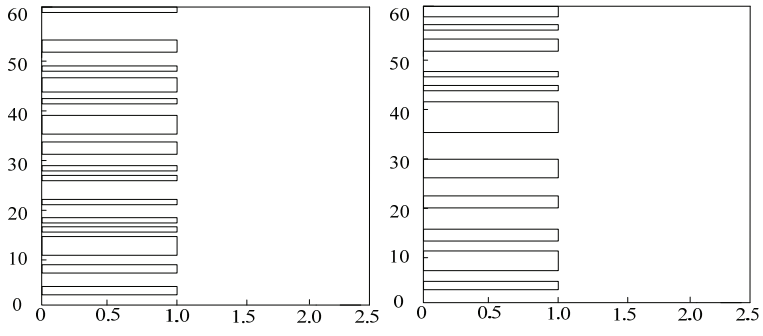

Fig. 2. Analysis of encryption and decryption effect: (a) key simulation results and (b) ciphertext decryption results.

3.2 Encryption Efficiency

The algorithm of this paper is applied to encrypt the cloud-based genes that need to be matched, as well as to count the time of encrypting gene files in different gene file fragments. In the meantime, the overall acceleration ratio in the encryption process is calculated. The encryption efficiency of the algorithm, the genetic matching based on symmetric encryption algorithm and the genetic matching based on asymmetric algorithm are compared. The comparison results are shown in Table 2.

Table 2 indicates that under the same file size of the same algorithm, as the number of gene files increases, the encryption time is gradually shortens; the overall acceleration ratio gradually increases. When the number of fragments of the gene file is the same, the encryption time of the algorithm is the shortest and the acceleration of the algorithm is relatively large. When the number of 1 GB gene files is 65, the algorithm has the shortest encryption time and the maximum overall acceleration ratio. The encryption time is as short as 80.13 ms, and the overall acceleration ratio is up to 3.6. That is to say, the encryption efficiency of the algorithm is accelerated, which can significantly improve the speed of cloud-based gene matching. This is because the homomorphic encryption algorithm designed in this paper converts gene information into binary values, directly encrypting 2-bit ciphertext at once, greatly accelerating the encryption process and improving the encryption efficiency.

3.3 Actual Matching Effect

To verify the gene matching effect in the cloud computing environment under the algorithm, a large number of genes need to be used for experimental analysis. The results of the analysis are as shown in Fig. 3.

Table 2. Statistical results under the number of slices of different documents

Type of algorithm	Number of slices	Encryption time (ms)			Overall acceleration ratio SP		
		1 GB	2 GB	3 GB	1	2	3
This paper algorithm	1	316.12	609.31	903.79	1.1	1.1	1.1
	5	106.18	206.69	310.05	3.2	3.2	3.2
	9	97.25	192.01	308.52	3.3	3.2	2.9
	17	84.36	161.28	245.31	3.8	3.7	3.6
	34	83.27	157.74	241.87	3.7	3.7	3.6
	49	81.56	152.52	236.82	3.7	3.7	3.6
	65	80.13	145.68	231.83	3.7	3.7	3.6
Gene matching based on symmetric encryption algorithm	1	472.55	740.56	1005.24	0.9	0.9	0.9
	5	262.43	337.94	411.5	3.0	3.0	3.0
	9	253.61	323.26	409.97	3.1	3.0	2.7
	17	237.96	292.53	346.76	3.6	3.5	3.4
	34	235.74	289.99	341.32	3.5	3.4	3.3
	49	229.66	284.77	337.27	3.5	3.4	3.3
	65	224.55	278.93	334.28	3.5	3.4	3.3
Gene matching based on asymmetric encryption algorithm	1	603.82	869.35	1104.28	0.6	0.6	0.6
	5	393.91	466.73	510.26	2.7	2.7	2.7
	9	384.86	452.05	508.73	2.8	2.7	2.4
	17	369.21	421.32	445.52	3.3	3.2	3.1
	34	351.98	417.78	441.08	3.2	3.1	3.0
	49	347.07	412.56	437.03	3.2	3.1	3.0
	65	343.83	409.72	431.04	3.2	3.1	3.0

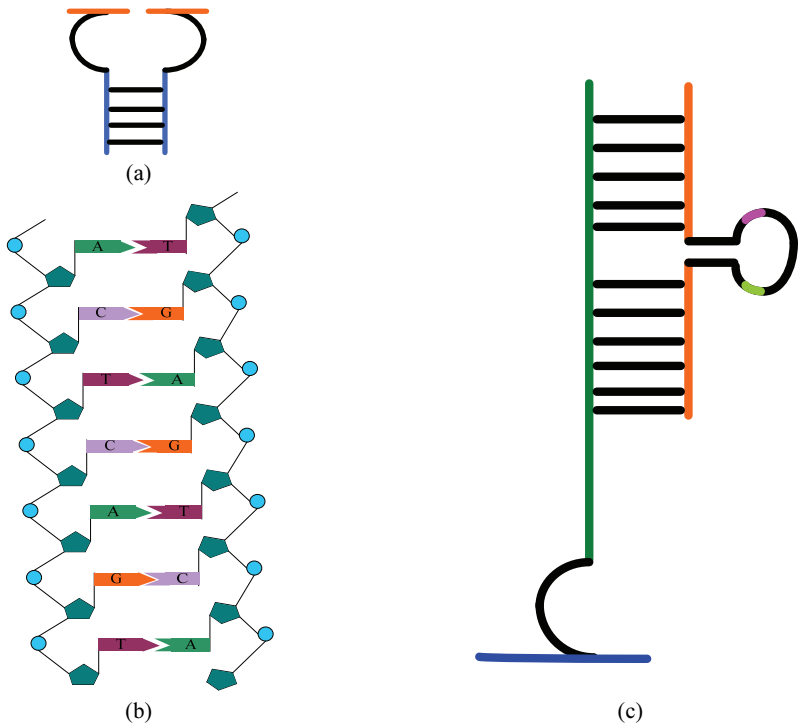

Fig. 3. Analysis of matching results: (a) original gene chain, (b) original gene chain DNA sequence, and (c) match result.

As shown in Fig. 3, comparing Fig. 3(a) with Fig. 3(c), it can be seen that the original gene chains all appear in the output gene chains, indicating that the algorithm can match a part of the original gene chains to the complete gene chains in the cloud computing environment. Comparing the matching results with the actual results, as shown in Fig. 3(b), it is found that the genes matched by the algorithm are consistent with the actual genes, indicating that the algorithm has high matching accuracy. This is also because the homomorphic encryption algorithm in this paper can simplify the expression of ciphertext by using binary values, and achieve high-precision matching.

3.4 Matching Performance

To detect the gene matching performance in the cloud computing environment when using the algorithm, it is required to analyze the time, energy consumption and accuracy of gene matching. The three algorithms used to compare the time and energy consumption of gene matching under different gene sequence numbers, and the results are shown in Table 3.

Table 3. Results of time and energy consumption analysis of gene matching

Type of algorithm	Text size (GB)	Time (ms)	Energy consumption (J)
This paper algorithm	1	85.69	237.89
	2	146.89	241.58
	3	231.92	247.63
	4	235.68	249.36
	5	237.25	251.48
	6	241.29	253.57
	7	246.87	259.86
	8	249.63	261.43
	9	252.36	263.56
	10	259.48	267.89
Gene matching based on symmetric encryption algorithm	1	102.53	263.87
	2	163.72	267.56
	3	248.73	273.61
	4	252.49	275.34
	5	254.06	277.46
	6	258.23	279.55
	7	263.68	285.84
	8	266.44	287.41
	9	269.17	289.54
	10	276.29	293.87
Gene matching based on asymmetric encryption algorithm	1	114.09	295.44
	2	175.28	299.13
	3	260.29	305.18
	4	264.05	306.91
	5	265.62	309.03
	6	269.79	311.12
	7	275.24	317.41
	8	278.63	318.98
	9	280.73	321.11
	10	287.85	325.44

From Table 3, as the size of the gene sequence increases, the matching time of the three algorithms is also increasing. At the same gene sequence size, the gene matching time and energy consumption of the algorithm are the least.

As shown in Fig. 4, the method in this paper is finally compared with the traditional homomorphic encryption algorithm, A* algorithm and regularized homomorphic encryption algorithm in the dataset. As the number of characters retrieved by matching increases, the matching speed of all methods is increasing. However, when the number of characters is between 600 and 5,400, the method proposed in this paper is the least time-consuming.

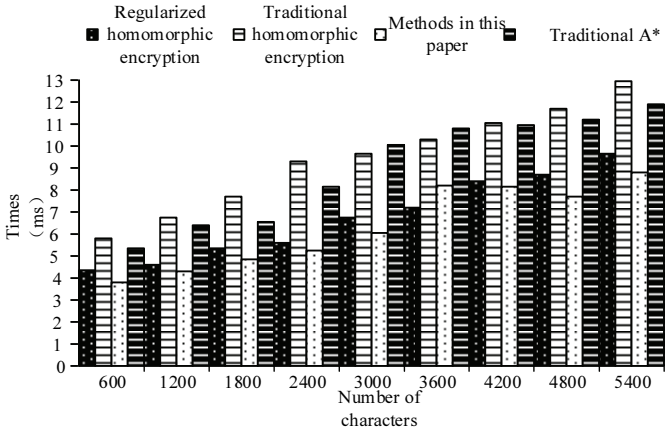

Fig. 4. Status analysis of matching time increasing with the number of characters under different algorithms.

5. Conclusion

To improve the security and accuracy of gene matching, this paper studies the cloud-based homomorphic gene matching encryption algorithm. When the encrypted gene is transmitted to the matching terminal, it is decrypted. The decrypted gene is compared with the gene in the database and the binary sequence is compared to achieve the purpose of gene matching. Experimental analysis shows that the algorithm can significantly improve the safety of gene matching. Meanwhile, the algorithm has shorter encryption and decryption time in the gene matching process, which can greatly reduce the gene matching time. Through the actual test, it is found that the gene matching effect is better under the algorithm and the matching accuracy is higher. There are still some limitations in this paper, and in the future research work, two aspects should be noticed: first, the distributed computing model will be studied to further improve the efficiency of fully homomorphic encryption computing; secondly, the integration of different privacy protection principles, anonymous technology and homomorphic encryption technology will be further studied.

Conflict of Interest

The authors declare that they have no competing interests.

Funding

The research is supported by Basic Public Welfare Research Project of Zhejiang Province - Research on Cooperative Task Allocation Mechanism in Heterogeneous Multi - Mobile Robot System (No. LGG19F020009).

References

[1] Y. Li, J. H. Park, and B. S. Shin, "A shortest path planning algorithm for cloud computing environment based on multi-access point topology analysis for complex indoor spaces," *The Journal of Supercomputing*, vol. 73, pp. 2867-2880, 2017. https://doi.org/10.1007/s11227-016-1650-x

[2] M. B. Karimi, A. Isazadeh, and A. M. Rahmani, "QoS-aware service composition in cloud computing using data mining techniques and genetic algorithm," *The Journal of Supercomputing*, vol. 73, pp. 1387-1415, 2017. https://doi.org/10.1007/s11227-016-1814-8

[3] A. Acar, H. Aksu, A. S. Uluagac, and M. Conti, "A survey on homomorphic encryption schemes: theory and implementation," *ACM Computing Surveys (CSUR)*, vol. 51, no. 4, article no. 79, 2018. https://doi.org/10.1145/3214303

[4] D. Porumbel, "Ray projection for optimizing polytopes with prohibitively many constraints in set-covering column generation," *Mathematical Programming*, vol. 155, pp. 147-197, 2016. https://doi.org/10.1007/s10107-014-0840-7

[5] A. M. Yakubu and Y. P. P. Chen, "Ensuring privacy and security of genomic data and functionalities," *Briefings in Bioinformatics*, vol. 21, no. 2, pp. 511-526, 2020. https://doi.org/10.1093/bib/bbz013

[6] A. Mittos, B. Malin, and E. De Cristofaro, "Systematizing genome privacy research: a privacy-enhancing technologies perspective," Proceedings on Privacy Enhancing Technologies, vol. 2019, no. 1, pp. 87-107, 2019. https://doi.org/10.2478/popets-2019-0006

[7] M. Hosseini, D. Pratas, and A. J. Pinho, "Cryfa: a secure encryption tool for genomic data," *Bioinformatics*, vol. 35, no. 1, pp. 146-148, 2019. https://doi.org/10.1093/bioinformatics/bty645

[8] Z. Hu, S. Liu, and K. Chen, "Privacy-preserving location-based services query scheme against quantum attacks," *IEEE Transactions on Dependable and Secure Computing*, vol. 17, no. 5, pp. 972-983, 2020. https://doi.org/10.1109/TDSC.2018.2831199

[9] M. N. Sadat, M. M. Al Aziz, N. Mohammed, F. Chen, X. Jiang, and S. Wang, "SAFETY: secure gwAs in federated environment through a hybrid solution," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 16, no. 1, pp. 93-102, 2019. https://doi.org/10.1109/TCBB.2018.2829760

[10] T. Gao and F. Li, "PHDP: preserving persistent homology in differentially private graph publications," in *Proceedings of IEEE INFOCOM 2019-IEEE Conference on Computer Communications*, Paris, France, 2019, pp. 2242-2250. https://doi.org/10.1109/INFOCOM.2019.8737584

Pingping Li **https://orcid.org/0000-0002-8047-5289**

She was born in March 1981. Her title is senior engineer. She received a bachelor's degree in information engineering from Zhejiang University in 2002. In 2007, she received a master's degree in software engineering from Hangzhou Dianzi University. She is now working in Zhejiang Technical Institute of Economics. Her research fields include artificial intelligence, data processing, etc. She has published 5 academic papers and participated in 6 research projects.

Feng Zhang **https://orcid.org/0000-0002-3863-2826**

She was born on September 23, 1970. Her title is professor. She holds Master of Engineering degree from the Changchun Institute of Optics, Fine Mechanics and Physics, Chinese Academy of Sciences in 1995, majoring in electromechanical control and automation. She is now working at Zhejiang Business College. Her research interests focus on artificial intelligence. She has published 5 academic papers and participated in 5 research projects.