Single-Stack Deep-Query Quad-Tree RFID Tag Anti-collision Algorithm

Yanling Zhou, Linshan Ma, and Thomas Clemen

***Abstract*—The problem of tag collision is the main problem affecting the performance of RFID systems. Probabilistic tag anti-collision algorithms have tag starvation and cannot recognize some tags. This paper proposes a deterministic query-tree algorithm, that is, a single-stack deep query-tree RFID tag anti-collision algorithm, which successfully implements tag anti-collision by generating new query code and tag recognition. The algorithm uses the highest two collision bits of the collision code to form four new query codes. The query tree formed by the query code is a depth-first traversal quad-tree. The algorithm introduces a single-stack storage mechanism, and the query code uses a depth-first traversal algorithm. In the entire tag recognition process, the structure of the query code spanning tree was improved, and the degree of the branch node was 4. This combination of depth-first traversal and single-stack mechanism effectively shortened the recognition time, saved memory space, and reduced the number of tag collisions. And the algorithm is simple and easy to implement. When the number of tags increases, the efficiency of the algorithm will be more clearly reflected.**

***Index Terms*—RFID, anti-collision algorithm, query tree, single stack.**

I. Introduction

RFID is an easy-to-operate, fast and practical automatic identification technology [1]. A typical RFID system consists of a reader and multiple low-cost and small-sized tags and each tag corresponding to a unique identifier is attached to the object. The reader realizes the automatic recognition of the target object by reading the tag information [2]. If there are multiple tags in the recognition range of the reader, and these tags transmit information to the reader at the same time, the reader will detect the conflict, which is called "collision", which causes the tag recognition to fail. Due to the large number of tags used in radio frequency identification, the problem of tag collision has become a bottleneck that limits the efficiency of RFID systems.

Commonly used RFID tag anti-collision algorithms are divided into two categories; one is based on Aloha algorithm, also known as probabilistic algorithm [3]. In the probabilistic algorithm, because the number of free time slots and collision time slots cannot be determined, some tags may not be recognized due to tag starvation. Another algorithm is a tree-based algorithm, also known as a deterministic algorithm [4], which can avoid starvation and ensure that each label is accurately identified. However, because the tree-based algorithm polls and traverses according to the binary combination rule, the delay of reading tags is longer. When the number of tags is large, the efficiency of the algorithm will be significantly reduced. Therefore, the tree-based anti-collision algorithm needs to group the IDs of tags according to a certain length. When the packet length is 1, it is a binary tree algorithm. When the packet length is 2, it is a quad-tree algorithm. When the packet length is 3, it is the octree algorithm.

Based on the query tree-based anti-collision algorithm, this paper proposes a single-stack deep-query quad-tree RFID tag anti-collision algorithm. By introducing a combination of depth-first traversal and single-stack mechanism, the identification time is effectively shortened and the memory space is saved, the number of label collisions is reduced, and the recognition efficiency is improved.

Manuscript received April 15, 2020; revised July 12, 2020.

Yanling Zhou is with College of Artificial Intelligence and Big Data, HeFei University, HeFei, China (e-mail: zhouyanling1006@163.com).

Linshan Ma is with HeFei University Library, Hefei University, Hefei, China (e-mail: lsmao@hfuu.edu.cn).

Thomas Clemen is with Deputy Dean of School of Computer Engineering, Hamburg University of Applied Sciences, Germany (e-mail: thomas.clemen@haw-hamburg.de).

II. Tag Anti-collision Algorithm Based on Multi-fork Tree

The first application in the RFID system is the binary tree anti-collision algorithm. Its basic principle is: the reader sends a one-bit query code Q (0 or 1 two kinds of tags, which are equivalent to forming two subtrees, querying 0 subtree first, and then querying 1 subtree). Each tag within the response range of the reader judges whether its ID starts with Q, and if so, transmits its ID to the reader. At this time, there may be three situations: recognition (only one label starts with Q), collision (two or more labels start with Q), and idle (five labels start with Q) [5]. If a collision occurs, add 0 and 1 respectively to the previous query code to form two new query codes (equivalent to splitting into left and right subtrees). First the reader sends a new query code with 0 at the end to the label and queries the left subtree; then sends a new query code with 1 at the end to the label and query the right subtree. If a collision occurs again during the query, the above operation is repeated until all tags are successfully identified [6]. In the tree-based RFID anti-collision algorithm, in addition to the binary tree anti-collision algorithm, there is also the quad-tree anti-collision algorithm. In the quad-tree anti-collision algorithm, the label packet length is 2, and there are four combinations of node codes. A tree with a branch of 4 is formed during the query process.

Currently, the query tree algorithm is the most commonly used algorithm in deterministic algorithms. And based on the query tree algorithm, there are many improved algorithms. Literature [7] proposed a method to use Manchester coding to find the specific collision position of the returned information of the tag, and then change the prefix code of the query

DOI: 10.7763/IJCTE.2020.V12.1279

through the obtained collision position. The query speed of this algorithm is relatively slow when the label is identified. When the number is large or the length of the tag ID is relatively long, the query efficiency of the algorithm decreases rapidly. Literature [8] proposed a quad-tree query tree algorithm, which can effectively reduce the number of label collisions at the beginning of the query, but as the number of labels in the branch decreases, a lot of idle time will be generated during the query process. Search efficiency has not been really improved. Literature [9] proposes a pre-detection query tree anti-collision algorithm, in which the tags to be identified are pre-processed, and the dynamic binary algorithm is used to query the tags. The memory usage also increases with the number of tags. As a result, queries have increased, and the complexity of the algorithm is large, so it is not used in actual scenarios. Reference [10] proposes a hybrid query tree algorithm. Removing the prefix from each matching prefix tag in the original query queue in the algorithm, the delay time is determined by the number of 1 in upper three bits and decides to respond to the reader. In some cases, the algorithm can significantly increase the number of tag collisions, thereby making the algorithm less feasible. Literature [11] proposes an effective time-based anti-collision algorithm, which uses a double query prefix matching method; the purpose is to eliminate the idle time slot in the traditional query tree algorithm. The query speed increases due to the extension of the matching time of the matching code. As the number of tags increases, the complexity increases. Reference [12] proposes a hybrid query tree anti-collision algorithm HQT, which combines the advantages of dynamic binary query tree and quad query tree, and dynamically selects the binary query tree and quad query tree according to the label information returned by the tag. As the number of tags increases, the memory usage also increases significantly, and the complexity also increases significantly.

The comparison between the quadtree anti-collision algorithm and the binary tree anti-collision algorithm can be found that in these two types of algorithms, the initial node and the identification node are the same, but the difference is the number of the collision node and the idle node. In the quadtree anti-collision algorithm, there are fewer collision nodes than the binary tree collision algorithm, but there are more idle nodes. If the coding algorithm of the query code of the quadtree can be improved to reduce the number of free nodes, the recognition efficiency will be greatly improved. It has proved that the quad-tree anti-collision algorithm works best. Therefore, in order to optimize the algorithm and improve the recognition efficiency, in addition to reducing the number of collision nodes and idle nodes, it is also necessary to consider the amount of memory space occupied during the entire tag query process and the time used in the query process. On the basis of comprehensively considering the above problems, this paper proposes a new anti-collision algorithm for RFID tags, namely: single-stack deep-query quad-tree RFID tag anti-collision algorithm.

III. Single-Stack Deep-Query Quad-Tree RFID Tag Anti-collision Algorithm

The single-stack deep-query tree anti-collision algorithm mainly uses a single stack to store the query code, and uses the highest two collision bits of the collision code to form a new query code, so that each time a collision code is generated, four new query codes can be formed. Push the query codes generated each time into the stack, and then pop up a query code from the top of the stack as a new query code to identify the tags in the RFID range. The query tree formed by the query codes is a depth-first traversal quadtree. The algorithm introduces a single-stack storage mechanism. The query code uses a depth-first traversal mechanism. During the entire tag recognition process, the structure of the query code spanning tree is improved. The degree of the branch node is 4. This combination of depth-first traversal and single-stack mechanism can effectively shorten the recognition time, save memory space and reduce the number of label collisions. At the same time, the algorithm is simple, easy to implement. In the case of an increase in the number of labels, the efficiency of the algorithm will be more clearly reflected by the query code. The algorithm is mainly divided into two parts of the generation and label recognition.

A. Query Code Generation

There are many electronic tags to be identified in the RFID identification area. The RFID reader needs to send an inquiry code at a certain moment to inquire whether the electronic tags within the identification range are responding. When two or more electronic tags meet the condition of the query code at the same time, a response signal will be sent to the RFID reader at the same time, which will cause a collision. According to the principle of Manchester encoding, when multiple electronic tags send a response signal, the response signal received by the RFID reader is a fuzzy response signal. At this time, this fuzzy response signal is a collision code. Fig. 1 is an example of the collision code and the generated new query code set. The code position represented by X in the collision code is a fuzzy code position, which means that when the RFID card reader receives the response signal, the signal at this position is uncertain, that is, two cases of 0 and 1 occur at the same time. Six collision bits appear in the CC1 collision code in Fig. 1. At this time, RFID needs to send a new query code. In this algorithm, the principle of generating a new query code is to encode the highest two bits of the collision bit, and all other collision bits are set to 1. According to this principle, the new query codes generated by the CC1 collision code at this time are four query codes 10001111, 10101111, 11001111, and 11101111. Similarly, for the collision codes CC2 and CC3, the same principle is used to generate a new query code. In the 10001111 query code, 1000 is the matching code, and 1111 is the range code; also in the query code 10100011 formed by the CC2 collision code, 101000 is the matching code, and 11 is the range code.

Every time the RFID reader queries the tags within the recognition range. In the query process, a certain query code generates a collision code. At this time, the newly generated collision code generates four new query codes again and the new query codes are pressed onto the stack at the same time. And then the top element of the stack is popped as the next query code, and RFID reader continues to query the tags within the RFID range. The following Fig. 2 is the corresponding stack diagram after the collision code is generated. In the query process of the RFID reader, three

collisions have been generated; each collision code generates four new query codes that are pushed onto the stack, where the contents of stack S vary with the collision code.

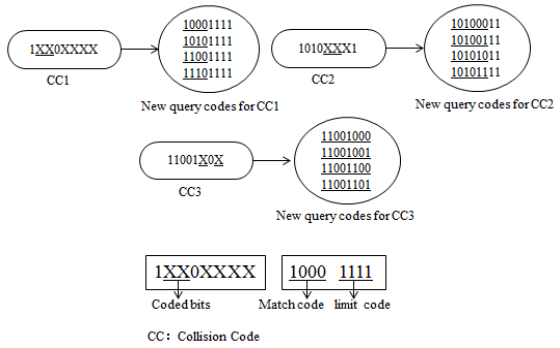

Fig. 1. Collision codes and the generated new query code set.

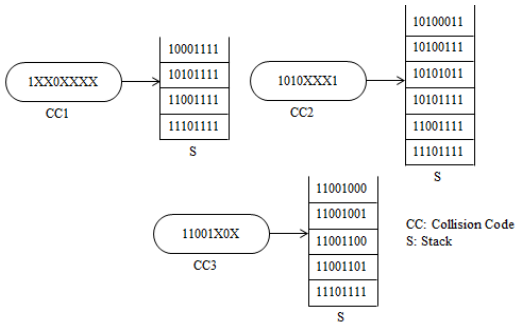

Fig. 2. The corresponding stack diagram after the collision code is generated.

B. Label Recognition

In this algorithm, the query code is mainly generated by the highest two collision bits of the collision code, and the newly generated query codes are pushed onto the stack. When inquiring, at first it needs to judge whether the stack is empty, if it is not empty, pop an element from the top of the stack as a new query code to identify the label within the recognition range. Fig. 3 is the change chart of the stack during a certain RFID tag identification process. The seven tags identified are T1, T2, T3, T4, T5, T6, and T7, and their IDs are 10100011, 11001100, 10101001, 10101100, 10001101, 11001001, and 11101001, respectively.

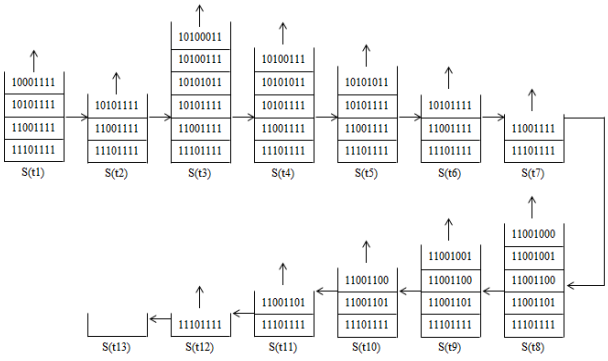

Fig. 3. Changes in the stack during the entire recognition process.

It can be found from Fig. 3 that the maximum size of the stack memory space used for identifying 7 electronic tags is 6 memory space sizes. With the increase of electronic tags in the RFID identification range, the number of collisions also increases, and the size of the used memory space also increases, but no matter how the number of electronic tags increases, the size of the stack memory space used in this algorithm will always be less than the number of electronic tags. Therefore, compared with other query tree algorithms, this algorithm obviously saves memory space and improves resource utilization in terms of saving memory space.

RFID reader identification of tags, first, the RFID reader sends the query code. If the RFID reader receives the response signal, it means that there is exactly one tag in the recognition range that meets the query code condition, which means that the tag recognition is successful. After the recognition is successful, the tag enters the sleep state, and no longer participates in the label recognition in this range. If the received response signal is a fuzzy signal, it means that a tag collision has occurred in this query, so a new collision code is generated, and then the system generates four query codes according to the upper two fuzzy bits of the collision code and pushes the query codes onto the stack. Send the query code and identification response signal again. The following Fig. 4 is the Quad-query code tree.

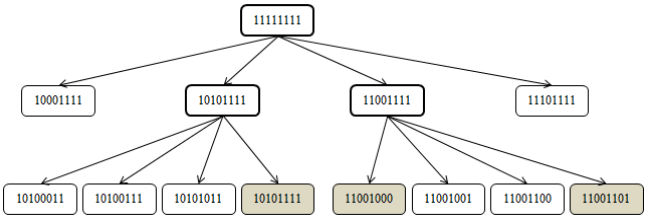

Fig. 4. Quad-query code tree.

In the process of tag identification, electronic tags may be in three states, namely active state, waiting state, and sleep state.

C. Specific Identification Process

Assuming that there are 7 tags T1, T2, T3, T4, T5, T6, T7 to be identified within the response range of the reader, their IDs are 10100011, 11001100, 10101001, 10101100, 10001101, 11001001, 11101001, defining the left side of the ID is high. The identification process is shown in Table I below.

TABLE I: LABEL IDENTIFICATION PROCESS DIAGRAM

Query Code	Collision Code	Stack content	Identified tag	Waiting for tags
11111111	1XX0XXXX	10001111, 10101111, 11001111, 11101111	no	T1,T2,T3,T4, T5,T6,T7
10001111	no	10101111, 11001111, 11101111	T5	T1,T2,T3,T4 T6,T7
10101111	1010XXX1	10100011, 10100111, 10101011, 10101111, 11001111, 11101111	no	T1,T2,T3,T4 T6,T7
10100011	no	10100111, 10101011, 10101111, 11001111, 11101111	T1	T2,T3,T4 T6,T7
10100111	no	10101011, 10101111, 11001111, 11101111	no	T2,T3,T4 T6,T7
10101011	no	10101111, 11001111, 11101111	T3	T2, T4, T6, T7
10101111	no	11001111, 11101111	T4	T2, T6, T7
11001111	11001X0X	11001000, 11001001, 11001100, 11001101, 11101111	no	T2, T6, T7
11001000	no	11001001, 11001100, 11001101, 11101111	no	T2, T6, T7
11001001	no	11001100, 11001101, 11101111	T6	T2, T7
11001100	no	11001101, 11101111	T2	T7
11001101	no	11101111	no	T7
11101111	no	Ø	T7	Ø

After the identification of the seven tags within the RFID identification range, 12 query codes have been generated successively.

IV. Performance Analysis

In Fig. 4, the query code tree includes two types of nodes: branch nodes and leaf nodes, where the branch nodes are the query-code nodes when generating tag collisions during the query process. Leaf nodes include two types of nodes: one is a query code node that successfully identifies a tag in the RFID range, and the other is a query code node that is empty for query.

$$n4=n40+n44 \tag{1}$$

Because the branch node and the leaf node have the following relationship:

$$n40=4 \times n44-(n44-1)=3 \times n44+1 \tag{2}$$

So:

$$n44=(n40-1)/3 \tag{3}$$

Here $n40$ includes the sum of the number of successfully queried nodes and the number of empty query nodes. The number of collisions during the RFID querying process is $n44$.

In a binary query tree, $n20$ represents the number of leaf nodes, that is, the number of nodes with a degree of 0, $n22$ represents the number of branch nodes, that is, the number of nodes with a degree of 2, and $n2$ represents the total number of nodes in the query tree Quantity, then there is the following relationship:

$$n2=n20+n22 \tag{4}$$

$$n22=n20-1 \tag{5}$$

Therefore, the number of branch nodes is basically equal to the number of leaf nodes. The branch nodes represent the number of generating collisions during the RFID query process. Therefore, in the binary query tree, the number of generating collisions during the RFID querying process is $n22$.

In the traditional binary query tree and quadruple query tree, there is the same problem, that is, there are many empty queries. In a traditional quad query tree, the number of empty queries is almost greater than the number of query nodes. Therefore, the number of tags identified in the RFID range is $n0$. In the quad-query tree algorithm, the number of leaf nodes is greater than $2 \times n0$. Therefore, the number of branch nodes $n2 + n4$ is approximately $n0 / 2$.

In the hybrid query tree algorithm, the query tree includes two types of nodes: branch nodes and leaf nodes, where the branch nodes are query code nodes that generate tag collisions during the query process. The branch nodes include two types: the degree of one node is 2, the degree of another node is 4; leaf nodes also include two types of nodes: one is a query code node that successfully identifies a tag within the RFID range, and the other is a query code node that has an empty query type. Suppose that in the hybrid query tree, $n0$ represents the number of leaf nodes, $n02$ is the number of leaf nodes generated by the node with branch 2, and $n04$ is the number of leaf nodes generated with the node with branch 4. $n02 + n04$ is the total number of leaf nodes with degree 0, $n2$ represents the number of branch nodes with degree 2, $n4$ represents the number of branch nodes with degree 4, n represents the total number of nodes in the query tree, and thus exists the following relationship:

$$n4=(n04-1)/3 \tag{6}$$

$$n2=n02-1 \tag{7}$$

$$n2+n4=(n04+3 \times n02-4)/3 \tag{8}$$

In the hybrid tree algorithm, the number of nodes with degree 2 is basically the same as the number of nodes with degree 4, so formula (8) can be approximated as:

$$n2+n4=(2 \times n0-4)/3 \tag{9}$$

In the hybrid tree query algorithm, the number of branch nodes represents the number of collisions generated during the RFID query process, so in the hybrid tree query algorithm, the number of collisions generated during the RFID query process is $n2 + n4$.

As the number of tags in the RFID identification range increases, the number of $n0$ also increases. With the increase of identification tags, the number of empty queries generated by the three algorithms of the binary tree query algorithm, the binary tree algorithm is the fastest increase. The quadtree and hybrid tree query tree algorithms is relatively slow, so the number of $n0$ in the quadtree query tree algorithm and the hybrid tree query algorithm are relatively consistent.

Fig. 5 is a relationship diagram between the number of identification tags and the number of collisions, where C2 represents the number of collisions generated by the traditional binary query tree algorithm, C2 + 4 represents the number of collisions generated by the traditional quad query tree algorithm, and C4 represents the number of collisions generated by the single-stack deep query tree anti-collision algorithm, C24 represents the number of collisions generated by the hybrid tree query algorithm. In the case of the same number of identified tags, the value of C4 is the smallest, the value of C2 is the largest, C24 is smaller than C2, and the number of collisions generated by C2 + 4 and C24 is equivalent, which is larger than C4. Therefore, the algorithm in this paper can reduce the number of collisions compared with traditional algorithms. At the same time, the size of the stack memory space used in this algorithm is always smaller than the number of RFID identification tags. Therefore, compared with other query tree algorithms, the number of collisions is significantly reduced. In terms of saving memory space, this algorithm significantly saves memory space and improves resource utilization.

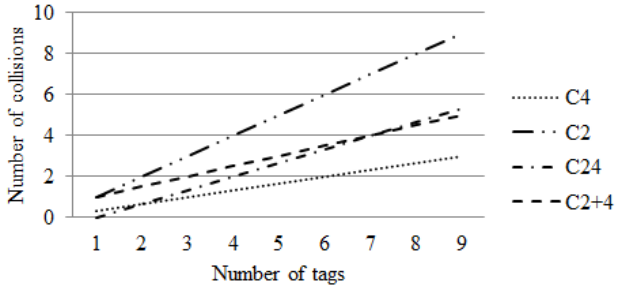

Fig. 5. Relation between tags number and collision number.

V. Conclusion

In the single-stack deep-query tree anti-collision algorithm,

the occupation of memory space is mainly affected by the number of query codes, and the number of query codes is affected by the number of collisions. The more the number of collisions, the greater the number of query codes generated. It can be seen from the analysis that the number of collisions of the traditional binary query tree is twice that of the traditional quadrature query tree. However, the traditional binary query tree generates two query codes during a collision, while the traditional quad query tree generates four query codes during a collision. Therefore, there is no obvious improvement in the size of the memory space occupied. The single-stack deep query tree anti-collision algorithm proposed in this paper, by encoding the highest two bits of the collision code to form four new query codes, all other collision bits are complemented by 1, so that the number of the query code formed by each collision is four. Through simulation analysis, the algorithm can be compared with the traditional binary query tree, quad query tree, and hybrid query tree algorithm, which greatly reduces the number of collisions. Therefore, the use of the memory space is also significantly reduced. At the same time, the algorithm proposed in this paper is simple and easy to implement. With the increase in the number of electronic tags in the RFID range, the advantages of the algorithm proposed in this paper can be more clearly reflected.

Conflict of Interest

The authors declare no conflict of interest.

Author Contributions

Yanling Zhou and Prof. Thomas Clemen conducted the research; Linshan Ma analyzed the data; Yanling Zhou wrote the paper; all authors had approved the final version.

Acknowledgment

This paper is supported by the Excellent Talents Training Funded Project of Universities of Anhui Province (No. gxgwfx2019065), Teaching Research Project of Hefei University (No. 2018hfjyxm52), Provincial Humanities and Social Science Research Project of Anhui Colleges (No.SK2018A0605) and Provincial Large-Scale Online Open Course (MOOC) Demonstration Project (No.2019mooc270).

References

[1] J. F. Shan, M. Chen, and J. B. Xie, "Study on RFID anti-collision technology based on ALOHA algorithm," *Journal of Nanjing University of Posts and Telecommunications: Natural Science Edition*, vol. 33, pp. 56-61, 2013.
[2] R. Want, "An introduction to RFID technology," *IEEE Pervasive Computing*, vol. 5, pp. 25-33, 2006.
[3] L. Duan, X. Zhang, Z. Wang, and F. Duan, "A feasible segment-by-segment aloha algorithm for RFID systems," *Wireless Personal Communications*, vol. 96, pp. 2633-2649, 2017.
[4] J. Su, G. Wen, and D. Hong, "A new RFID anti-collision algorithm based on the Q-ary search scheme," *Chinese Journal of Electronics*, vol. 24, pp.679-683, 2015.
[5] J. D. Shin and S. S. Yeo, "Hybrid tag anti-collision algorithms in RFID systems," in *Proc. ICCS 2007*, China: Beijing, 2007, pp. 693-700.
[6] K. H. Yeh and N. W. Lo, "An efficient tree-based tag identification protocol for RFID systems," in *Proc. 22nd International Conference on Advanced Information Networking and Applications*, Japan: Okijawa, 2008, pp. 966-970.
[7] J. X. Wu, A. Jiang, and Q. Y. Huang, "Improvement of the binary-searching-based anti-collision algorithm of RFID systems," *Journal of Hunan University: Natural Science Edition*, vol. 37, pp. 82-86, 2010.
[8] R, Jayadi, Y. C. Lai, and C. C. Lin, "Efficient time-oriented anti-collision protocol for RFID tag identification," *Computer Communication*, vol. 112, pp. 141-153, 2017.
[9] L. H. Zhu, H. Li, and L. Y. Chen, "Improved pre-detection query tree anti-collision algorithm in RFID system," *Computer Engineering and Design*, vol. 35, pp. 4040-4043, 2014.
[10] Q. Zhou and M. Cai, "Improved binary query tree algorithm for anti-collision of RFID tags," *Computer Engineering and Design*, vol. 33, pp. 209-213, 2012.
[11] C. H. Li, J. Sun, K. X. Liu, Y. Han, and H. J. Li, "Performance analysis and research of anti-collision algorithm based on query tree," *Acta Electronica Sinica*, vol. 46, pp. 2671-2678, 2018.
[12] W. Jiang, H. X. Yang, and J. Zhang, "An improved binary query tree algorithm for tags anti-collision in RFID," *Computer Technology and Development*, vol. 25, pp. 86-89, 2015.

Yanling Zhou received the B.S. degree in computer application technology from Liaocheng University, Liaocheng, China, in 2002, the M.S. degree in computer application technology from Northeast Dianli University, Jilin, China, in 2005 and the Ph.D. degree in control theory and control engineering from Donghua University, Shanghai, China, in 2010.

From 2010 to now, she is an associate professor of artificial intelligence and big data at Hefei University. She mainly engaged in research on the Internet of Things and big data processing. In recent years, she has guided students to achieve excellent results in the Internet of Things competition. And she has published nearly ten papers on the Internet of Things and big data processing.

Linshan Ma received the B.S. degree in information management from Lanzhou University, Lanzhou China, in 1998 and the M.S. degree in science and technology intelligence software engineering from the School of Software, University of Science and Technology, Hefei, China, in 2008. From 1998 to now, he has been working at Hefei College. He is mainly engaged in research on computer technology in library applications, web-based information services, reader information literacy development, and data analysis and visualization. And he has published nearly ten papers on the Internet of Things and big data processing.

Thomas Clemen holds a position as a full professor at the University of Applied Sciences in Hamburg, Germany. His teaching activities predominantly cover topics within information management and data science, whereas he is focusing on modeling and simulation of dynamic, complex and self-organizing systems in his interdisciplinary research.