Internet of Vehicles Based on TrustZone and Optimized RSA

Zhaoyang Wang\textsuperscript{a}, Min Zuo\textsuperscript{b}, Shuangshun Yao and Nijiati Aihemaiti

School of computer and information engineering, Beijing Technology and Business University, China

\textsuperscript{a}908251078@qq.com, \textsuperscript{b}zuomin1234@163.com

Abstract. In recent years, the problem of data leakage in Internet of Things applications has become a major social issue, causing strong concern in all aspects. Especially for today’s popular car networking technology, if the driving data and other information is stolen, it will not only expose personal privacy and personal information, but also can even harm life safety. At the same time, some manufacturers tamper with the collected vehicle data, in order to forge false vehicle performance and deceive consumers. In the light of the data security risks faced by current Internet of Vehicles applications, based on the TrustZone architecture, this paper builds a trusted execution environment, uses timestamp authentication technology, and introduces Rabin, Huffman coding and random components, respectively, from the perspective of improving encryption speed and reducing the size of ciphertext files, to optimize the traditional asymmetric RSA algorithm, thus to construct a secure data acquisition method of Internet of Vehicles based on TrustZone and optimized RSA, which realizes the information security of the data collection of the vehicle network data. The experimental environment was built based on raspberry PI 3B experimental board, and the test and comparative experiment were conducted. The experimental results show that the method can encrypt the vehicle driving data efficiently and safely, prevent the attacker from stealing and tampering with the data, and ensure the data security of the collecting terminal. The optimized RSA algorithm improves the encryption speed while ensuring security, and the ciphertext file is smaller, which is more suitable for the embedded environment with large data volume and limited hardware computing capacity.

1. Introduction
At present, the Internet of Things technology has become an important development direction of communication technology. The concepts of Internet of Vehicles, Smart City, Internet of Things, and Environmental Monitoring have been deeply rooted in the hearts of the people. Everything is beginning to have spirituality and can be interconnected. In addition to the application of network technology, another key technology of interconnection between things is the collection of massive data. Data collection is one of the important functions in the operation of the data acquisition system based on the Internet of Things. In the Internet of Vehicles application environment of the Internet of Things, the collected data often involves the privacy of some users, and is highly vulnerable to illegal theft and tampering. In order to ensure the timeliness, accuracy and security of data collection results, the hardware of data acquisition system under the action of Internet of things need to be designed in a
standard way, to reduce the security risk while improving the security and stability of the system operation, and encrypt the collection results. For another, under the influence of informationization and digitization, the embedded system has ushered in a period of rapid development. The embedded software technology is becoming increasingly mature, the number of microprocessors continuously rising, the kernel function constantly enhanced, and the cost lower. Traditional MCUs are more suitable for IoT applications, and data acquisition systems based on embedded technology are now widely used.

At the data storage layer, the blockchain technology with decentralized distributed structure application, non-tamperable timestamp, and security trust mechanism has become a research hotspot of information technology. Blockchain is decentralized, unique, autonomous, tamper-resistant, anonymous, and etc. [1], which is an emerging solution for data security storage at present. By writing the intelligent contract of access rights, it can effectively prevent unauthorized users from adding and accessing the data on the chain, and further guarantee the security of the data on the server side. (Server-side encryption)

However, technologies such as blockchain cannot guarantee the security of user data before it is uploaded to the "chain". If user information is stolen or tampered before being uploaded to the server, subsequent distributed storage will be meaningless. Aiming at this problem, this paper designs and implements a secure data acquisition method based on TrustZone and optimized RSA, which uses trusted execution environment, time stamp, asymmetric encryption algorithm and other technologies, and designs encryption mechanism, which can be deployed in embedded data acquisition terminal, and provides security for user information before being uploaded to the server side. In addition, due to the huge amount of data in the IoT application environment, the limited hardware computing capacity and slow operation speed in the embedded environment, this paper optimizes the RSA algorithm to improve the operation speed while ensuring security, in order to prevent the problem of data accumulation.

The main work of this paper includes:
(1) This paper designs a secure data acquisition method based on TrustZone and asymmetric encryption algorithm.
(2) This paper optimizes the RSA algorithm for the limitation of the large volume data in the Internet of Vehicles application environment and the limited computing capacity in the embedded environment.
(3) This paper presents the results of the data security acquisition method, and evaluates the encryption efficiency and ciphertext file size before and after RSA algorithm optimization.

2. Related work

In view of the related problems of data trusted collection, this paper summarizes the research results at home and abroad. In addition, for the selection of asymmetric encryption algorithms, the article lists several mainstream asymmetric encryption algorithms, and conducts comparison and analysis based on their advantages and disadvantages, and selects the basic encryption algorithm applied to this method.

2.1. Research status
Research status: In view of malicious behaviors such as theft, tampering and destruction of the data collection end by attackers, some researchers in the industry have carried out studies and proposed feasible protection schemes based on ARMA wireless sensor network, digital watermarking, CMTU algorithm, binding between data and user-defined tag. Wang Haiyuan and Wang Weichuan of Nanjing University of Posts and Telecommunications [2] proposed a reliable data acquisition method based on ARMA for wireless sensor networks, which ensures the high reliability of the collected data and significantly improves the overall performance of the network. Li Hongtao and Li Aiguo [3] of Xi'an University of Science and Technology proposed a new digital watermark embedding method to ensure the security, reliability and integrity of the classified images in the transmission process. Each node in
the CMTU [4] algorithm uses a different key for encryption, which can well resist internal attacks. YAASE [5] combines protected data with user-defined tags and controls data transfer based on fine-grained access policies to prevent malware from stealing sensitive information through network access or collusion attacks, enabling tag-based access control mechanisms. Zhao et al [6] use static random access memory to generate a trusted root without adding security hardware to obtain the security key of the encrypted data, and ensure the data security by storing the encrypted data on the system hardware. Hein et al [7] designed a security device based on security key and Merkle-Tree authentication encryption, which can effectively prevent attackers from tampering, stealing and destroying data.

2.2. Asymmetric encryption algorithm selection
Asymmetric encryption algorithm contains Public Key (PK) and Private Key (SK). Public key and private key is a pair. If the public key is used to encrypt the data, only the corresponding private key can be used to decrypt the data. The strength of asymmetric encryption algorithm is complex, and the security is mainly determined by the algorithm and key management. Symmetric cryptosystem is a single key and is non-public. Therefore, to ensure its security is to ensure the security of the key, but the key must be disclosed to the other party when decrypting. The asymmetric key system has two keys. One of them is public, so that it is not necessary to transmit the key to the other party like a symmetric password, ensuring the security of the key pair, so the asymmetric encryption algorithm is more secure. This paper lists RSA, DSA, ECC, RABIN, the four most widely used asymmetric encryption algorithms [8-12], and compares the characteristics of maturity, security, computing speed, resource consumption, etc. The comparison results are shown in table 1.

| Name  | Maturity | Security | Computing speed | Resource consumption |
|-------|----------|----------|-----------------|----------------------|
| RSA   | High     | High     | Low             | High                 |
| DSA   | Medium   | High     | Low             | Medium               |
| ECC   | Low      | High     | Medium          | Low                  |
| RABIN | High     | High     | Low             | High                 |

In the embedded environment, development is more difficult. Therefore, it is necessary to select a highly secure and mature asymmetric encryption algorithm to facilitate further optimization according to specific application requirements. On the other hand, the hardware level in the embedded environment is low, and because the asymmetric encryption algorithm is more complicated, the operation speed is slower, the system load is too large, and the resource consumption problem is more serious. Therefore, the algorithm needs to be optimized in terms of pin computation efficiency, ciphertext size, and resource requirements. By comparing the four algorithms of RSA, DSA, ECC and RABIN in Table 1 above, the RSA algorithm is finally selected according to the characteristics of maturity, security, computing speed and resource consumption.

3. Design of data security collection method for Internet of vehicles
Data security collection method of Internet of vehicles in this paper mainly includes three parts: data acquisition layer, encryption layer and storage.

3.1. Data acquisition type
Vehicle data acquisition refers to automatic collection of driving data (non-electrical or electrical signals) from analog and digital units under test, such as sensors and other devices to be tested, and sent to the host computer for analysis and processing. Due to the disadvantages of the traditional acquisition system, such as slow response, low accuracy, poor reliability, low efficiency and cumbersome operation, it has been unable to fully adapt to the current demand in the field of Internet.
of vehicles. At present, the embedded technology has been relatively mature, so the vehicle movement data acquisition device based on the embedded is the best choice.

After investigating the existing field of Internet of Vehicles, this paper summarizes the data acquisition types of Internet of Vehicles and determines the data acquisition types of the method in this paper. The vehicle network data safety acquisition method designed in this paper based on TrustZone and RSA is mainly aimed at vehicle speed, driving time, driving distance, vehicle acceleration, vehicle position, in-car temperature, engine coolant temperature, in-car humidity and other data acquisition.

3.2. Method and encryption mechanism design
The hardware devices required by this method mainly include: power supply, ARM core control module, sensor, storage device and other modules. The method frame diagram is shown in Figure 1.

Data acquisition of vehicle data in the Internet of Vehicles application environment is performed by calling a temperature sensor and a position sensor. The collected data is transmitted from the CA end to the TA end in the ARM core control module. In the trusted execution environment built by the TrustZone technology, the data is encrypted by the encryption mechanism based on the time stamp and the optimized asymmetric encryption algorithm, and the ciphertext is stored.

Before uploading to the server, the data can be decrypted and verified by the public key to further ensure the credibility of the data. Since all encryption processes are performed in a trusted execution environment, not only can the privacy of the private key be ensured, but also the stealing and tampering from an external attacker can be prevented because the private key is stored in a secure permanent memory.

![Method frame diagram](image)

Figure 1. Method frame diagram.

The data acquisition process, vehicle data usually consists of two parts: data and acquisition time. In the past Internet of Vehicles information collection and data applications, criminals can tamper with vehicle data and acquisition time locally before the data is uploaded to servers or blockchains to obtain false vehicle performance or damage data in the system. This paper will use the encryption mechanism to ensure the credibility and timeliness of the data (time recursion). The encryption mechanism is shown in Figure 2.

(1) A public key private key pair (PK, SK) is generated within the trusted execution environment, and the private key is permanently stored in the trusted execution environment.

(2) Transfer the data (m) collected by the sensor to a trusted execution environment.
(3) Time stamping the data within the trusted execution environment to ensure the singularity of the time parameters in the driving data.
(4) A number of driving data with time stamps will form a data block in memory and wait for processing.
(5) Using the asymmetric encryption algorithm, the driving data is encrypted by the private key.
(6) The time stamp and the ciphertext are combined to form an encrypted message and stored.
(7) Before being uploaded to the block chain or server, it is necessary to verify ciphertext (C) with public key (PK) to ensure the credibility of the driving data that hasn’t been uploaded to the chain.

Figure 2. Diagram of encryption mechanism.

4. Based on the technology of TrustZone and the method of optimizing RSA
This paper designs an encryption mechanism based on asymmetric encryption algorithm and time stamp to encrypt vehicle information in trusted execution environment based on TrustZone to ensure the security of user information and prevent illegal stealing and cracking. In order to solve the problems of poor computing power in embedded environment, large amount of driving data information in the application of Internet of vehicles, and heavy load on the server platform, Rabin algorithm, Huffman coding and random components s will be introduced to optimize RSA algorithm.

4.1. Method implementation based on TrustZone Technology
TrustZone is a hardware-based isolation mechanism proposed by ARM to build a secure and reliable environment for codes requiring high security [13-14]. It divides the hardware and software resources of the system into two parts, one is the trusted execution environment, the other is the general environment. All sensitive operations should be protected in a trusted execution environment, such as fingerprints, facial features and key storage, security authentication, digital rights management, etc. The rest of the operations that require less security are performed in a normal execution environment.

On the hardware level, first, each physical processor core of the TrustZone ARM processor provides two virtual cores, one called Non-secure (NS) and the other called Secure Core (S). Data exchange between the areas is not free. Non-secure cores can only access their own system resources. The security core can access all resources to ensure that resources stored in the secure area are not stolen. The mechanism for switching between a secure core and a non-secure core is called a monitor mode. A process that is not a secure core can enter the monitor mode through the SMC instruction or the hardware exception mechanism, thereby obtaining the service of the security core. It should be noted that the process of the non-secure core can only obtain the service of the security core and
cannot access the data of the security core. Second, TrustZone technology has made efforts in virtual memory management, Cache, peripheral bus, etc. to ensure that the security core data information is not leaked. At the software level, TrustZone implements a secure and trusted boot, that is, it starts the security zone and performs integrity verification on the boot state during the boot load process. After the verification is correct, the non-secure zone is started, which ensures the security of the system startup process.

4.1.1. Trusted Execution Environment. The Trusted Execution Environment (TEE) [15-16] was first proposed by the Global Platform (GP) and developed a technical specification, which is an operating environment coexisting with Rich OS (usually Android, etc.) on the device, and also provides security services to Rich OS. For the wide application of mobile devices and complex open environments, information security issues are getting more and more attention, not only end users, but also service providers, mobile operators, and chip manufacturers. At present, more than 95% of mobile terminal chips in the world use ARM or cooperate with them. Nearly half of the mainstream mobile terminal processor manufacturers implement the TEE security environment through TrustZone technology.

The trusted execution environment can be divided into two parts: hardware layer and software layer. The software layer includes system layer, interface layer and application layer, as shown in Figure 3. The hardware and software resources required by TEE are separated from Rich OS by TrustZone technology. In order to protect the confidentiality, integrity and access rights of TA resources and data, applications with high security requirements (trusted applications, TA) need to be authorized in the TEE before they can be called through the client application (CA). Each TA is independent of each other and cannot be accessed from each other without authorization. In order to ensure the trusted root of the TEE itself, the TEE is verified and isolated from the Rich OS during the secure boot process [17].

The TEE internal API mainly includes key management, cryptographic algorithms, secure storage, secure clock resources and services, and an extended trusted UI API. Trusted UI means that when related information or key information needs to be displayed or input, hardware resources such as display and keyboard will all be taken over by TEE, and applications in Rich OS cannot access it. In the development of the TA program, the TEE internal API is the programming interface provided to the TA, and the TEE external API is the underlying communication interface for the CA running in the Rich OS to access the TA service and data.

![Figure 3. Trusted Execution Environment Framework.](image)

4.1.2. Time stamp authentication. In the previous Internet of vehicles information collection and data application, some criminals will change the time information in the user data, use the tampered data to cover the real information or insert forged data to cheat the database and data authentication or obtain
false performance. In view of this problem, this paper will stamp untamable timestamp on the driving data according to the built-in clock of TrustZone to further ensure the credibility of data acquisition.

A timestamp is a complete, verifiable data that represents a piece of data that existed before a certain time, usually a sequence of characters that uniquely identifies a moment in time. The trusted execution environment based on TrustZone technology has its own time mechanism, which is unique and consistent. In a trusted execution environment, a security command will be triggered when a trusted clock source generates an interrupt or some monitored system behavior occurs. Any tampering, insertion, or deletion of time information will be rejected by the system to ensure the authenticity of the data in the system and the time series singularity. Therefore, even if the attacker invades the trusted execution environment or cracks the key pair, the driving data with time stamping cannot be tampered. Absolute timestamps will be used in the article so that users, vendors, and regulators can retrieve data from the system through time information in the server-side system with authorization.

4.1.3. Implementation of encryption mechanism. The TA program is written in OP-TEE to implement user information encryption.

(1) The Privkey Generator is used to generate a secret key pair in the TA, and the private key is permanently stored in the TA.

(2) Create a CA-TA call, transmit the user information to the TA end, and use the command ID to call the encryption program of the TA end.

(3) Time stamp the driving data, based on the feature that the trusted execution environment time can not be tampered in TrustZone technology.

(4) The time stamped travel data will be temporarily stored in the memory, and will form a data block each time the data is accumulated to 256 KB.

(5) Encrypt the user information using the optimized encryption algorithm.

(6) The time stamp and ciphertext are combined into encrypted data and transmitted back to the CA for storage.

(7) Store the public key in a smart contract in the database or blockchain and verify the data before it is passed to the server.

4.2. Huffman coding and random component X

In order to increase the encryption speed, Huffman coding [18] can be used to compress the data. It is an algorithm for lossless data compression, which can accurately recover the original data from the compressed data.

This algorithm is used to compress data (symbols or letters) to generate variable length codes instead of fixed length codes for each symbol. The algorithm constructs a frequency table by statistical analysis of symbols or letters in the content and constructs a Huffman tree through a frequency table for each symbol to assign its appropriate code length. Applying Huffman encoding on a data file will generate two files: binary ( ) and header ( ). The binary depends on the header file used to retrieve the original data, so if the header file is missing, the real data cannot be retrieved. The header file contains all the symbols of the original data file or their corresponding ASCII code. The header file contains a unique symbol for its occurrence, where no symbol is repeated twice, and the binary file contains the code for each symbol. For example, the original text: Department of Computer and Information Engineering, e is represented by 110, and t is represented by 010, etc.

The header file is:
\u008c\u00a3\u00e8\u00ae\u00fd\u00f0\u00d0\u00e2\u008a\u00fb\u005k\u00d8\u00e5\u00a8

The binary file is:
0001011000010100011010001100110001101111110001110101000100001011110101110011111000111101011010001011110011010100011111001010000111111111011101101001101111100001011001100110100010111110100011110100011111111011010110011101111010111110000000000000000

To decompress the message, build the Huffman tree from the header file, read the binary file bit by bit from the root of the tree, and move to the left when the 0 bit is found; when the 1 bit is found,
move it to the right on the tree until find the leaf node and repeat the process for all remaining bits until all message characters are retrieved.

By introducing a random component $x$, each time a message is encrypted, a different ciphertext is obtained, so that it is difficult for an attacker to crack from the ciphertext about the original message. The letter $x$ is used herein to denote a random component, where $x$ is a random number generated by using an encrypted secure pseudo-random number generator and used once for each message (random number). This article uses $x$ to hide the ciphertext of the header file and invalidate the binary. In the case where $x$ is smaller than ciphertext, multiple $x$ is superimposed multiple times as the length of the ciphertext. If $x$ is larger than ciphertext, the single digits of $x$ are removed to be the same length of ciphertext, and when the binary file is invalidated, the same method will be applied.

In order to make the encryption process semantically secure, this paper selects the random component $x$ and calculates a new ciphertext. Again, to make the binary B semantically secure, use $x$ to blind B, for example. On the other hand, the random component $x$ should be protected. This time, we will use the Rabin encryption algorithm to encrypt $x$.

4.3. Optimized implementation
The optimized RSA algorithm relies on Huffman coding to enhance security and speed up the encryption and decryption process. To increase execution speed, the encryption algorithm only encrypts the header file and keeps the binary secret, rather than encrypting the entire message. The binding of the binary file by parameters makes the encrypted message semantically secure. The algorithm flow is shown in Figure 4. The specific optimization algorithm is as follows:

Step 1: Generate a public/private key pair at the receiving end
1) Calculate the RSA algorithm public/private key pair.
2) Calculate the Rabin algorithm public/private key pair.
Step 2: Encryption preparation at the sender end
1) Generate a random component for each piece of encrypted information.
2) Compress information using Huffman code. Output: binary file () and header file ().
Step 3: The encryption process at the sender end
1) Encryption is performed using the RSA algorithm, and the encryption result is: $C = H^e \mod N, 0 < M < N − 1$.  
2) Use $x$ to blind $c$ and generate $C' = C \oplus x$.
3) Use $x$ to blind $b$ and generate $B' = B \oplus x$.
4) Encrypt $x$ using the Rabin algorithm and generate $x' = x^2 \mod N$.
Step 4: The decryption process at the receiving end
1) Decrypt $x$ using the Rabin algorithm.
2) Calculate $C = C' \oplus x$.
3) Calculate $B = B' \oplus x$.
4) Use the RSA algorithm to decrypt $C$ and generate $H = C^d \mod N$.
Step 5: Decompress the information at the receiving end
Pass $H$ and $B$ into the Huffman code to obtain the decrypted result.

5. Experiment
5.1. Experimental environment construction and experimental data
The experimental environment is mainly divided into two parts: hardware and software.

1) Hardware part: The hardware part of the system adopts ARM-based Raspberry Pi 3B as the main control board, and uses sensors such as temperature sensor, humidity sensor module and GPS positioning device as data acquisition devices, and develops with Python. 
2) System part: The Raspberry Pi uses the Linux operating system. Based on this environment, the Python language, C language, and OP-TEE software environment are built. The data acquisition uses Python to operate GPIO. Use OP-TEE as the trust OS, C language as the development language, develop the CA-TA
program, and deploy it on the Raspberry Pi 3B. The secure operating system and development environment follow the GP TEE system development architecture specification.

In order to obtain experimental data, the Raspberry Pi 3B experiment board with sensors for temperature, humidity, position, etc. will be deployed in 12 different vehicles to collect, encrypt and store data during driving. Data acquisition was performed every 10 seconds in the experiment, and a total of 3000 acquisitions were performed. The experimental data is shown in Table 2.

### Table 2. Sample of experimental data.

| Speed | driving time | driven distance | location               | temperature | humidity |
|-------|--------------|-----------------|------------------------|-------------|----------|
| 80    | 23           | 8.6             | 116.317611, 39.947896  | 25          | 20       |

#### 5.2. Encryption results and performance testing

The experimental data is transmitted to the TA end through the CA end, and the encryption operation is performed after the time stamp is added. Then, the encrypted data is composed of the time stamp and the ciphertext, and is transmitted back to the CA end by the TA end and stored. The final stored encrypted data is shown in Table 3.

### Table 3. Encryption result.

| Time          | Ciphertext                                                                 |
|---------------|-----------------------------------------------------------------------------|
| 2019/10/9/.13:00:00 | \x98\x92\x8b\x02\x90\x13\x0f\x9f\xdf\xb5\x80\y05\xfb\xe\x9a\xb9\x1f\xf8\xdf\x4f... |
| 2019/10/9/.13:00:10 | \x8c\xa2\x13\xdf\xc\x04\x88\x1e\x0e\x94\xd5\x0f\xd\x0e\x0e\x16\x7f... |
| 2019/10/9/.13:00:15 | \xb9\x8e\x95\x19\x8a\x95\x1f\x8c\xc\x62\x88\xa2\xf4\x03\x9f\xf1\x1k\x8e2\x0e... |

In the test, this article will use the RSA algorithm and the optimized RSA algorithm to encrypt 10 files of different sizes, each file is encrypted three times, and the average number of operations per unit time is taken. The test file size ranges from 1MB to 10MB. Figure 4 shows the encryption process for files of different sizes before and after RSA algorithm optimization. In addition, as can be seen from Figure 5, for the same plaintext, the ciphertext size generated by the optimized RSA algorithm is significantly smaller than before the optimization.

![Figure 4. Comparison of operational efficiency.](image-url)
5.3. Result analysis
From the results in 5.1 and 5.2, the secure data acquisition method based on TrustZone and optimized RSA can effectively ensure the information security of the data before the data is uploading to the server, and realize the security and trusted collection of the Internet of Vehicles data. At the same time, it can work efficiently in the embedded environment and can be deployed in more work environments.

Using the random component $x$ in the optimized algorithm can guarantee the semantic security in the encryption process. Each time the message is encrypted, different ciphertexts are obtained. At the same time, the binary file is blinded by the random component $x$, and the attacker cannot reversely crack the ciphertext corresponding to the key, which further enhances the security of the RSA algorithm.

In Figure 4, the encryption speed is proportional to the file size encrypted by the two cryptosystems, but the optimized RSA algorithm is significantly faster, which is an average of 88.0% higher than before optimization. This is because the optimized RSA algorithm does not encrypt all the information. Instead, the data is compressed by Huffman coding and then the encryption algorithm is run, which improves the speed of the operation while ensuring information security. On the other hand, the amount of data in the Internet of Vehicles application is extremely large. It can be seen from Figure 5 that the size of the ciphertext generated by the improved RSA algorithm is reduced by 55.8% compared with the average before optimization, which will help reduce the storage pressure on the local and server side and reduce service costs.

6. Conclusion
This paper proposes and implements a secure data acquisition method based on TrustZone and optimized RSA. It researches the TrustZone technology and applies it to the field of Internet of Vehicles for the first time, effectively solving the hidden dangers of data security before user data is uploaded to the server and filling the current industry gap. According to the embedded environment and application requirements, this paper designs a driving data encryption mechanism based on the optimized RSA algorithm. In the trusted execution environment, the driving data is timestamped and encrypted. The encrypted driving data can be verified by the public key on the server side. Through test and analysis, the method can encrypt the user data in the Internet of Vehicles efficiently and safely, prevent the attacker from tampering with the data, and is suitable for application in an embedded environment with large data volume and limited hardware computing capacity. In the future, according to the specific characteristics and driving modes of different vehicles, the data types will be further enriched, the data acquisition methods will be optimized, and the emerging technologies such as blockchain will be combined to realize the information security and information credibility construction of the whole process of the Internet of Vehicles technology.

Acknowledgments
This work was financially supported by National Natural Science Foundation of China (No. 61702018), National Key Technology R&D Program of China (No.2016YFD0401205), Youth Fund...
for Humanities and Social Sciences Research of the Ministry of Education (No.17YJCZH007), Project of Beijing Natural Science Foundation (no.4202014) and Capacity building of scientific and technological innovation service9 (No. 19005902005).

References

[1] Fan Zhang. Design of revocable hybrid encryption control algorithm in encrypted data access of the Internet of things [J]. Science and technology and engineering, 2018, 18 (05): 247-251.
[2] Yingjun Zhang, Dengguo Feng, Yu Qin, et al. Protection scheme for sensitive applications in open environment based on TrustZone [J]. Computer research and development, 2017, 54 (10): 2268-2283.
[3] Yun Feng, Feng Zhai, Xiaobing Liang, et al. Trusted electricity information acquisition terminal based on domestic password [J]. Rural electrification, 2018 (10): 55-58.
[4] Lijun Yang. Research on key management and secure data aggregation technology in wireless sensor networks [D]. Nanjing University of Posts and telecommunications, 2014.
[5] Rui Chang. Research on trusted execution environment construction and security protection technology of Embedded Terminal [D]. PLA University of information engineering, 2017.
[6] Haiyuan Wang, Ruchuan Wang, Haiping Huang et al. Trusted data collection method of wireless sensor network based on ARMA model [J]. Journal of Nanjing University of Posts and Telecommunications (NATURAL SCIENCE EDITION), 2009, 29 (04): 85-89 + 96.
[7] Hongtao Li. Research on Key Technologies of reliable wireless digital image acquisition and transmission system [D]. Xi'an University of science and technology, 2013.
[8] ZHAO S, ZHANG Q, HUG, et al. Providing root of trust for ARM TrustZone using on-chip SRAM [C] Proceedings of the 4th International Workshop on Trustworthy Embedded Devices.
[9] HEIN D, WINTÉ R J, FITZEK A. Secure block device-Secure, flexible, and efficient data storage for ARM TrustZone systems [C] Proceedings of the 2015 IEEE Trustcom / BigDataSE / ISPA. Washington, DC: IEEE Computer Society, 2015, 1:222—229.
[10] Jinyu Bao. Principle and application of RSA public key cryptosystem [J]. Science and technology communication, 2019, 11 (06): 137-139.
[11] Rui Wang, Danhong Zhou. Research on Application of data encryption technology in computer network communication security [J]. Journal of Jilin Normal University of engineering technology, 2019, 35 (05): 88-90.
[12] Limin Song, Xiaorui Song. Design and implementation of a data security transmission scheme based on hybrid encryption [J]. Information network security, 2017 (12): 6-10.
[13] Wenyuan Wei, Weiping Peng, Zichen Li, et al. A digital signature scheme based on Rabin and Paillier [J]. Computer application and software, 2017, 34 (12): 301-306.
[14] Yu Li, Jun Zhang. analysis and Research on digital signature based on ECC algorithm [J]. Wireless interconnection technology, 2019, 16 (12): 110-111.
[15] Yan Liu, Xianhe Lang, Shaojing Pei. A lightweight elliptic curve encryption algorithm for small terminal equipment [J]. Science and technology and engineering, 2019, 19 (02): 131-134.
[16] Guotong Chen, Yuanyuan Fan. Middle dynamic Beidou navigation information terminal system based on arm [J]. Science and technology and engineering, 2019, 19 (23): 131-137.
[17] Weiqi D, Jun D, Qinyuan W, et al. Sblwt: a secure blockchain lightweight wallet based on TrustZone [J]. IEEE access, 2018: 1-1.
[18] Xiao Yu, Li Tian, Zhe Liu, et al. RPMB based privacy data protection method under TrustZone Architecture [J]. Computer application, 2018, 38 (S2): 164-169.