scientific reports

Check for updates

OPEN

Low power IoT device communication through hybrid AES-RSA encryption in MRA mode

Qiang Chang[1], Tianqi Ma[2]✉ & Wenzhong Yang[2]✉

In modern edge scenarios, the security of sensor networks plays a pivotal role in ensuring the safe communication of IoT data. Therefore, deploying encryption algorithms among edge devices is of paramount importance. However, most existing research predominantly emphasizes the efficiency of data transmission while often overlooking critical aspects of data integrity and confidentiality. The communication between sensor nodes and from sensors to end nodes necessitates additional robust security measures.This paper presents an enhanced approach to the RSA and AES algorithms, tailored for deployment on low-energy IoT devices commonly found in edge environments. The improved AES algorithm transitions from a single-threaded encryption and decryption process to a three-threaded design, reducing the ten-round operation to seven rounds to enhance both speed and energy efficiency. The enhanced RSA algorithm replaces the traditional dual-prime system with a triple-prime system, increasing the number of primes while reducing the bit length of each prime. This modification optimizes computational efficiency and strengthens security.Furthermore, we investigate the feasibility of integrating the RSA and AES algorithms and propose a novel RSA-AES hybrid algorithm MRA. This fusion approach leverages the strengths of both algorithms, encrypting the AES key using the improved RSA algorithm to bolster data security while maintaining efficient transmission.Performance evaluations of the RSA-AES hybrid algorithm demonstrate its exceptional capability in securing data transmission within edge scenarios,and the security of this hybrid algorithm is analyzed. The experimental results highlight that this algorithm not only ensures robust data security but also achieves high transmission efficiency, offering a reliable and practical solution for secure communication in edge environments.

Keywords Edge network, Data transfer, RSA-AES, Hybrid encryption, Security analysis

With the rapid advancements in sensors, data analytics, and high-performance devices connected to the Internet, the Internet of Things (IoT) is steadily maturing, with applications spanning smart grids, microgrids, intelligent industries,and smart agriculture. Moreover, devices such as smartphones, routers, and sensors play a pivotal role in this ecosystem[1,2]. In this context, ensuring the integrity, confidentiality, and accessibility of data is essential, limiting access strictly to authenticated users and servers. However, despite employing the most effective authentication methods[3,4], certain security vulnerabilities persist, particularly with regard to safeguarding data transmitted from IoT edge devices[5]. A significant challenge in the IoT domain is achieving reliable, low-latency, real-time data transmission in heterogeneous and dynamically changing environments. As IoT continues to expand into diverse sectors, there is a growing demand for optimized communication protocols and transmission algorithms that can meet the requirements of various communication scenarios.

Thus, the need for research on secure data transmission and processing is becoming increasingly urgent. IoT data communication can be broadly categorized into three types: (1) Device-to-Device (D2D). This mode involves direct communication between devices without the need for a central server or gateway, typically used for short-range communications such as Bluetooth and Zigbee[6]. (2) Device-to-Gateway. In this case, devices communicate with a local gateway, which is responsible for collecting, processing, and forwarding data to the cloud[7,8]. This method provides additional processing power and a layer of security via the gateway. (3) Device-to-Server (D2S). Here, devices communicate with a centralized server, which is mainly used in scenarios requiring centralized management and control[9]. The server coordinates and processes data exchanged between devices. In this paper, we focus on the second type of data transmission scenario. The radio frequency signals used by

[1]Xinjiang Transportation Investment Technology Co. Ltd, Urumqi 830000, XinJiang, China. [2]School of Computer Science and Technology (School of cyberspace Security), XinJiang University, Huarui street, Urumqi 10587, China. ✉email: 107552304112@stu.xju.edu.cn; yangwenzhong@xju.edu.cn

wireless communication devices are susceptible to eavesdropping, allowing hackers to intercept data transmitted by sensors and access sensitive information. Encryption and authentication techniques can mitigate this risk. Attackers may also attempt to tamper with data sent by sensors, potentially altering monitoring results or causing the system to take inappropriate actions. Techniques such as data integrity verification and encryption can be employed to detect and defend against such attacks. Therefore, during data transmission, it is necessary to process the data to ensure its secure transmission. Numerous studies have explored these challenges extensively. B.Sahu[10]proposed batch on demand data transfer which reduces the power consumption of data transfer and is effective in terms of energy efficiency, but data is vulnerable to attacks by ignoring security during media transfer. Lu Huang[11] utilised steganography for data transmission without using encryption algorithms, he took into account the limitations of the sensor's capabilities, but missing the security during data transmission. The ticket concept of J.G. Steiner's Kerberos[12]is also often used for authentication in the Internet of Things, where key distribution centres (KDCs) can help to establish a connection between two communicating parties, but remain vulnerable to denial of service attacks[13]. The proposal of TinyECC[14] evaluates low-bit elliptic curves that can be deployed in low-power IoT devices to reduce transmission power consumption, however, despite the shortened key lengths, the computational requirements of ECC elliptic curves are still a challenge for low-power devices and some security is lost. The common problem with the above research methods is the excessive focus on transmission energy consumption at the expense of data encryption. This paper proposes to optimise the RSA algorithm as well as the AES algorithm respectively, and proposes a model that combines the two, taking advantage of the security of the RSA algorithm as well as the fact that the AES can encrypt and decrypt a large amount of data with high speed[15]. That optimises the power consumption of cryptographic algorithms in IoT device protocols.

Our motivation and contributions

In this article, we focus on IoT scenarios with resource-constrained IoT devices, where existing lightweight encryption solutions are inadequate. This is due to three main reasons.

- Traditional encryption schemes have high computationaland storage requirements, which are impractical for resource-constrained IoT devices.
- Current compact encryption approaches have enhanced computational efficiency, yet they continue to be inadequate in resolving the latency and energy usage challenges encountered by IoT devices.
- There is currently limited research on encryption schemes for the LoRaWAN protocol, tailored to meet the stringent confidentiality requirements of modern IoT scenarios.

The main contributions of this work are as follows:

- Improve the AES algorithm by adopting an asynchronous operation for the AES algorithm, processing multiple subgroups at the same time, and splitting the number of subgroup threads of plaintexts to be processed into three groups. The original ten-round process is changed to seven rounds, which improves both efficiency and security, and no effective attack has been found against the seven-round version. The expansion of the algorithm is not much different from the traditional AES algorithm.
- Improvement of the RSA algorithm. For shorter key lengths, such as 1024 bits, RSA is no longer secure. The growth in key length means that the process of encryption and decryption will be more cumbersome, affecting efficiency. Increasing the number of prime numbers effectively reduces the number of digits per prime, reduces the time required for factorisation, and reduces the amount of key generation operations. In this paper, we propose to change the multiplication from traditional two prime numbers to three prime numbers. The factorisation time consumption is better than traditional RSA.
- A model combining the modified RSA algorithm (M-RSA) with the modified AES algorithm (S-AES) is proposed and simulated to be deployed in the LoRaWAN protocol for experiments. Its encryption and decryption time consumption has a small increase compared to AES, but the security is improved and more difficult to decrypt.

  The rest of the paper is organised as follows. Section II describes the RSA algorithm as well as the AES algorithm and the LoRaWAN protocol (where the algorithm is deployed). Part III describes the improvement of the AES and RSA algorithms and the encryption and decryption process and presents the improved combination model. The fourth part discusses the security and performs experimental simulations. The fifth section includes the limitations of the study and directions for future work.The sixth part concludes the article.

Preliminaries

In this section, we introduce two very important concepts, as well as the threat model of the scheme. The notation used in this paper is listed in Table 1.

AES and RSA algorithms

AES is a grouped iterative symmetric encryption algorithm with roughly the same process of encryption and decryption, both of which process the data by rounds. First of all, we need to expand the key, the AES algorithm supports different lengths of the key, usually 128-bit, 192-bit and 256-bit lengths, after the expansion of the key to generate the round key (Round Keys). During the encryption process, the initial round plaintext is subjected to a bitwise XOR operation with the first round key. In the decryption process, the initial round performs a bitwise dissimilarity operation between the ciphertext and the last round key. Then comes the heart of the matter in AES, the multi-round operation[16]. Each round includes four basic operations: SubBytes, ShiftRows, MixColumns

Symbol	Definition
$\oplus$	XOR
φ	Euler's function
S_i	The i-th packet
JoinNonce	Counter value in OTAA
NwkSKey	Network session key
AppSKey	Application session key
$Pr[n]$	The probability of attack success after adopting strategy n.
$\mathcal{A}$	Adversary
Adv_n^m	In n-mode, the attacker's advantage under m
q_s	Send queries
q_h	Hash queries
q_e	Execute queries
ϵ	Linear bias
$\mathbb{F}$	AES S-box

Table 1. Notations.

and AddRoundKey. That is, the process of the following figure, the operation of the standard AES encryption algorithm consists of ten cycles, the first nine rounds of the execution of the operation of the process is the same, and the last encryption process is no column mixing transformation.

In the above multi-round operation shown in Fig. 1.Before starting to encrypt or decrypt any data, the AES algorithm needs to expand the original key into a key schedule table (Key Schedule), which contains the round keys required for each round of encryption or decryption. The purpose of key expansion is to generate the subkeys needed for each round of encryption or decryption operations. First, assign the initial key used for expansion to a 4x4 state matrix, then each column of this matrix forms a word. The four words composed of the 4 columns of the matrix are$W[0], W[1], W[2], and\ W[3]$, respectively. After that, expand 40 new columns based on these 4 columns, and on this basis, construct a new key expansion array. The key expansion is performed as shown in formulas 1 and 2.

$$W[4i+j] = W[4(i-1)+j] \oplus W[4i-1+j] \tag{1}$$

$$W[4i] = W[4(i-1)+j] \oplus g(W[4i-1]) \tag{2}$$

The function g in the formula first requires performing a row shift on the input four bytes, then executing a byte substitution, and finally performing an XOR operation with the first byte and the result of the previous two steps as the round constant.

Byte Substitution (SubBytes): performs S-box substitution for each byte in the state matrix, adding non-linear properties. ShiftRows: cyclically shifts the rows of the state matrix according to a specified rule, adding diffusion properties. MixColumns: performs linear transformations on the columns of the state matrix to add confusion properties. AddRoundKey: Performs a bitwise permutation of the round key with the state matrix

RSA (Rivest-Shamir-Adleman) is an asymmetric encryption algorithm proposed by Ron Rivest, Adi Shamir and Leonard Adleman in 1977.The RSA algorithm is widely used in areas such as secure transmission of data and authentication. It is based on the mathematical difficulty that the product of two large prime numbers is difficult to decompose[17,18] into the product of its original prime numbers, this enables the use of different keys for the encryption and decryption processes. the RSA algorithm performs the following process:

1. Randomly generate two different large prime numbers a and b;
2. Using two prime numbers to calculate the modulus n=a * b;
3. Calculate the Euler function value again using two prime numbers:

$$\varphi(n) = (a-1)(b-1) \tag{3}$$

4. select an e value as the public key, which should be as large as possible and meet the requirement of $1 < e < \varphi(n)$ And e is related to $\varphi(n)$ mutual quality;
5. Using the values of e and Euler functions to calculate the private key:

$$e \times d \equiv 1(\mathrm{mod}\varphi(n)) \tag{4}$$

6. The public key of the password is (e, n), and the private key is (d, n);
7. Combine n and e into the public key, and n and d into the private key, so the public key is (n, e), and the private key is (n, d);

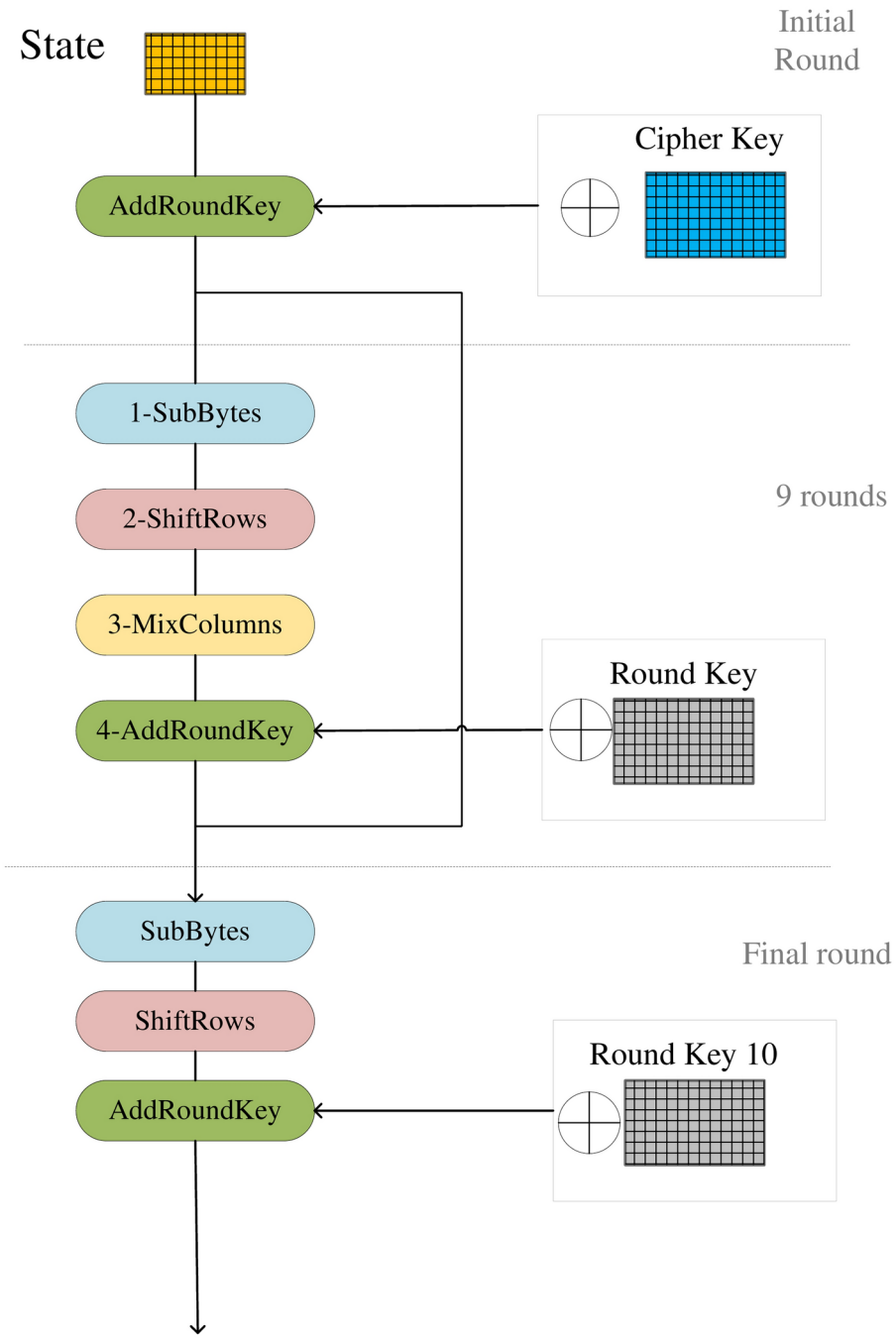

Fig. 1. AES algorithm ten rounds iteration process.

LoRaWAN protocol

The RSA-AES encryption proposed in this topic needs to be applied in LoRaWAN protocols to secure the communication. LoRaWAN (Low Power Wide Area Network) is a wireless communication protocol designed for long-range transmission, making it particularly suitable for low-power devices. LoRaWAN[19–21], built on LoRa[22] technology, enables devices to achieve an optimal balance between low power consumption and long-distance communication . The protocol operates using a star topology, where end devices communicate with a gateway via the LoRaWAN protocol. The gateway then forwards the collected data to the network server, which ultimately processes it at the application server level. LoRaWAN incorporates both Over-The-Air Activation (OTAA) and Activation by Personalization (ABP)[23]. This dual activation approach allows for the dynamic adjustment of configurations based on network conditions, enabling the generation of new session keys,also providing the flexibility for key activation without requiring real-time communication with the network server. Each IoT device is equipped with an embedded LoRaWAN communication module, facilitating communication with the gateway. LoRaWAN gateways, typically deployed near the application site,receive data from these IoT

devices and are responsible for forwarding this data to the LoRaWAN network. Traditionally, the data within the LoRaWAN protocol is encrypted using AES-128, ensuring data integrity and security. However, in this paper, we propose a new cryptographic algorithm to replace AES-128, aiming to enhance the protocol's security against emerging threat models. The network server processes the received data, including decryption and integrity verification . LoRaWAN employs an end-to-end encryption mechanism to safeguard data throughout its transmission. The network server serves as a conduit between LoRaWAN devices and application servers, routing data to the appropriate application server and relaying commands and configurations back to the IoT devices. The following section details the flow of the encryption algorithm in the traditional LoRaWAN protocol.

Figure 2 illustrates the workflow of device authentication and key generation in the LoRaWAN protocol. The end device initializes by generating or preloading relevant parameters, including the LoRa RSSI value, AppEUI (application identifier), DevEUI (unique device identifier), and a random number DevNonce. The device sends these parameters to the LoRaWAN server via a Join Request message. Upon validating the request, the server responds with a Join Accept message containing AppNonce (a server-generated random number), NetID (network identifier), and DevAddr (device address). Using the shared AppKey and parameters (AppNonce, NetID, and DevNonce), both the device and the server independently generate session keys-NwkSKey for network layer security and AppSKey for application layer encryption-via the AES-128 encryption algorithm as shown in Fig. 3. Finally, the server assigns DevAddr to the device, completing the secure authentication process and establishing an encrypted communication channel.

Threat model

We propose a threat model in which the authentication and key exchange processes occur in an environment that lacks inherent security. This model addresses both the unauthenticated link adversarial model (UM) and the authenticated link adversarial model (AM) within the context of LoRaWAN communications. In this threat model, the adversary has the capability to eavesdrop, manipulate, or replay traffic across all communication links during authentication events. These links include the channels between the server and the devices, as well as between the server and the database storing the device's data. By exploiting these communication channels, the adversary can intercept exchanged messages and attempt to identify repetitive patterns. Furthermore, the adversary can execute man-in-the-middle and spoofing attacks to gain unauthorized access. The device's open interface also allows the adversary to perform brute force queries using any past or adaptively chosen current messages or challenges.

Proposed method

System architecture

Here, different sensors are divided into different clusters. In each cluster there is a cluster head which is used for integration and forwarding of messages within the cluster and inter-cluster communication also depends on multi-hop routing between the cluster heads[24]. The intra-cluster communication uses the lorawan protocol.

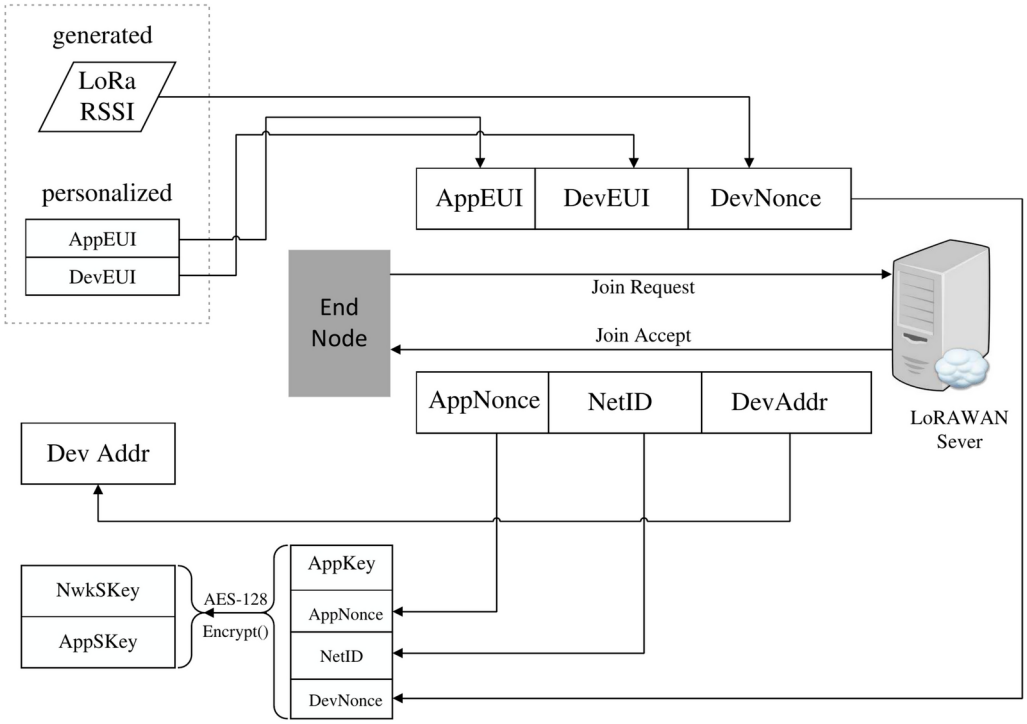

Fig. 2. Lorawan protocol response flow.

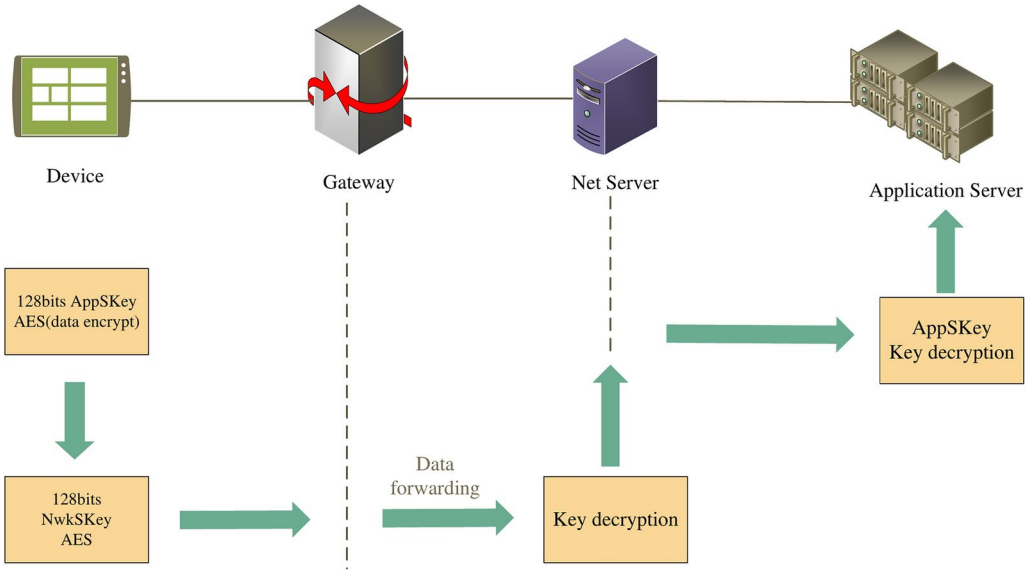

Fig. 3. Encryption algorithms in lorawan protocols.

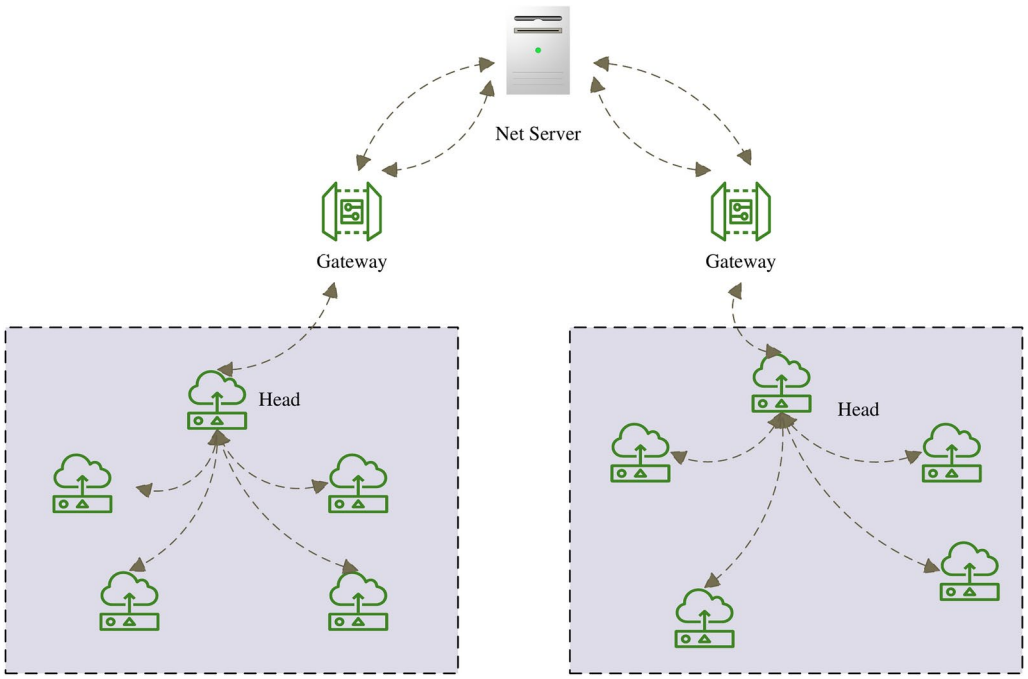

Fig. 4. System architecture diagram.

Here, our subsequent algorithmic model is used for data transfer between cluster head to gateway and gateway to server. The system architecture diagram is shown in Fig. 4.

Improved AES algorithm for lightweight devices(S-AES)

In order to improve the processing speed of data encryption and decryption, this paper changes the single-thread operation of the standard AES algorithm to a multi-thread operation[25]. In practical applications, the AES algorithm is usually used in a single thread for encryption and decryption operations. When larger plaintext groupings occur, the data needs to be split.In S-AES, asynchronous operation will be taken to process multiple groupings simultaneously. The number of threads of plaintext groupings to be processed will be split into 3 groups to improve the efficiency of data processing. The original ten rounds operation AES encryption process was designed to allow for security redundancy and we can make improvements to this as well. Optimizing

the original ten-round operation to seven rounds improves both efficiency and security. The structure of the algorithm is shown in Fig. 5.Moreover, no effective attack has been found against the seven-round version[26,27].

Input: $Plaintext, key$
Output: $Ciphertext$
  $state1\ state2\ state3\ ...\ staten$ ¸ $Plaintext$
  If $n < 3$ **then**
    while$i < n$**do**
      $state \leftarrow AES\_Encrypt(Plaintext, key, rounds = 7)$
    End
  Else
  $Initialize(thread.pool) : task\_factory :$
  $Initialize(task.list) : task\_list;$
  For $i < n$ **do**
    $task \leftarrow task\_factory.StartNew(AES\_Encrypt(Plaintext, key, rounds = 7);$
  $task\_list+ = task$
    If $takslist.count > 3$ **then**
      $task.wait.Any(task\_list.complete);$
    End
  End
  $Ciphertext \leftarrow State1, 2, 3. . . .Staten;$
  Return $Ciphertext$

Algorithm 1. Improved S-AES encryption algorithm

The improved AES encryption algorithm is designed to optimize performance for large-scale data encryption by incorporating parallel processing. The algorithm first determines the number of plaintext blocks (n) and chooses the encryption strategy accordingly: if n<3, a sequential encryption approach is applied, where each block undergoes 7 rounds of optimized AES encryption. For n>=3, a thread pool is initialized to handle parallel encryption tasks, distributing each block's encryption process across multiple threads. A task list is maintained to monitor active tasks, and a waiting mechanism ensures task synchronization. After all encryption tasks are completed, the encrypted blocks are combined to produce the final ciphertext. This approach enhances efficiency by leveraging parallelism while maintaining the security features of AES.

Input: $Ciphertext, key$
Output: $Plaintext$
  $state1\ state2\ state3\ ...\ staten$ ¸ $Plaintext$
  If $n < 3$ **then**
    while $i < n$ **do**
      $state \leftarrow AES\_Decrypt(Plaintext, key, rounds = 7)$
    End
  Else
  $Initialize(thread.pool) : task\_factory :$
  $Initialize(task.list) : task\_list;$
  For $i < n$ **do**
    $task \leftarrow task\_factory.StartNew(AES\_Decrypt(Plaintext, key, rounds = 7);$
    If $takslist.count > 3$ **then**
      $task.wait.Any(task\_list.complete);$
    End
  End
  $Plaintext \leftarrow State1, 2, 3. . . .StateN;$
  Return $Plaintext$

Algorithm 2. Improved S-AES decryption algorithm

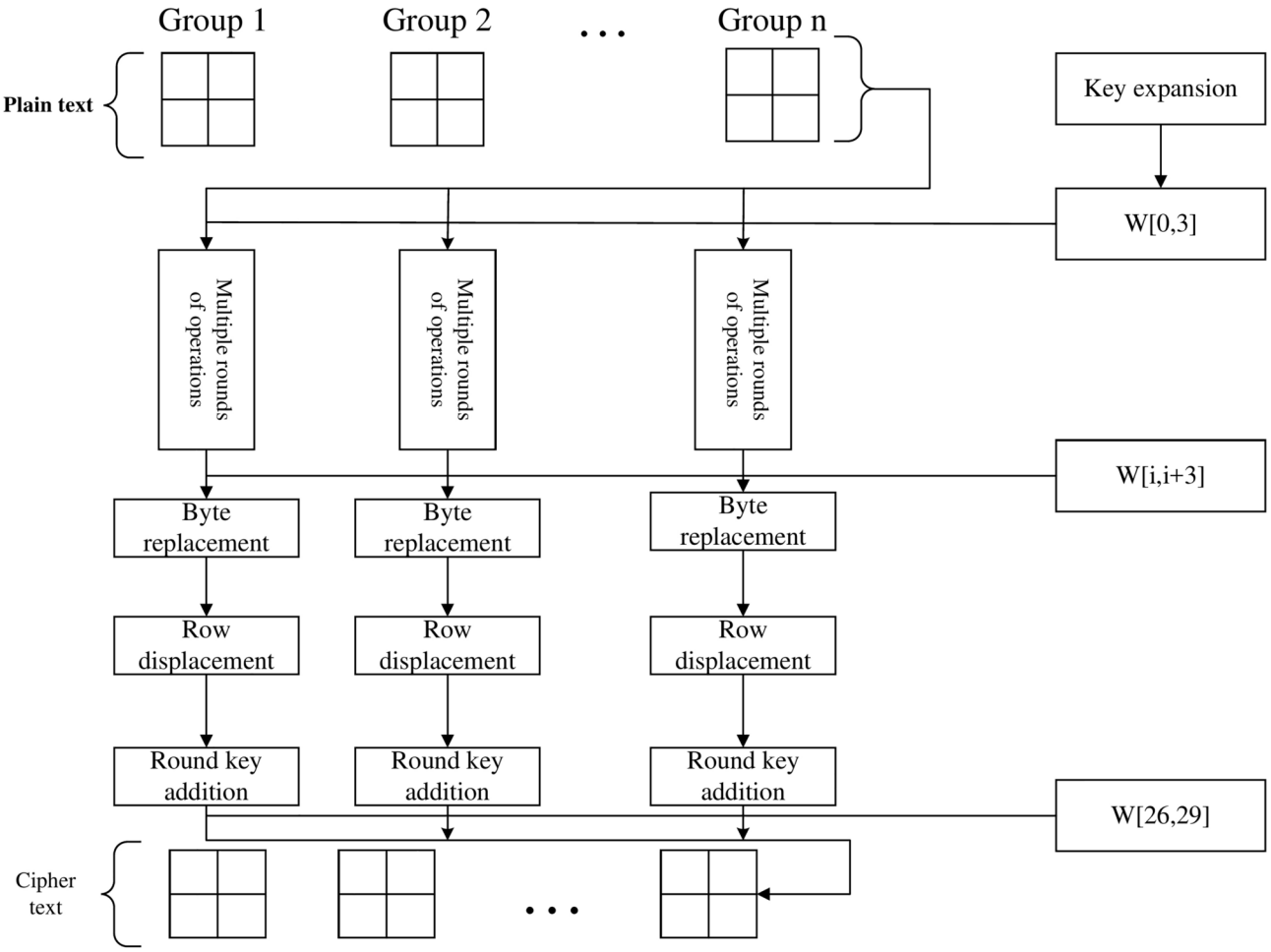

Fig. 5. Schematic diagram of S-AES algorithm.

Algorithm 2 is improved S-AES decryption algorithm efficiently handles ciphertext blocks by adapting to the number of blocks (n). If the number of blocks is small (n<3), the algorithm performs a sequential decryption for each block, using 7 optimized decryption rounds per block. For larger numbers of blocks (n>=3), the algorithm leverages parallel processing by using a thread factory to handle decryption tasks concurrently. Each decryption task is added to a task list, and a synchronization mechanism ensures that the number of active tasks does not exceed a set limit. Once all tasks are completed, the decrypted blocks are combined to reconstruct the plaintext. This approach improves decryption efficiency while maintaining the security properties of AES.

Modified RSA algorithm (M-RSA)

The RSA algorithm is still one of the most commonly used public key encryption algorithms. It is used in secure communication protocols such as SSL/TLS protocols[28,29], SSH[30], digital certificates, VPNs, and others. Although RSA has been widely used for many years, it is still considered secure as long as the key is long enough, e.g., by using a key of 2048 bits or longer. However, as computational power increases, RSA is no longer secure for shorter key lengths, such as 1024 bits[17], so there is a need to gradually change to longer key lengths. But the growth in key length means that the process of encryption and decryption will be more cumbersome, affecting efficiency. In order to improve efficiency, this paper proposes to change from the traditional multiplication of two prime numbers to three prime numbers. Increasing the number of prime numbers effectively reduces the number of bits in each prime number, which can reduce the time needed for factorization, reduce the amount of operations in the key generation process, and thus improve the overall computational efficiency, especially when modulo the same length is taken into account. In the RSA algorithm with three prime numbers, one more prime is added to generate the modulus, we choose three prime numbers a, b and c. The encryption process is as follows.

1. Randomly generate 3 different large prime numbers a,b and c;
2. Use 3 prime numbers to calculate the modulus n=a*b*c;

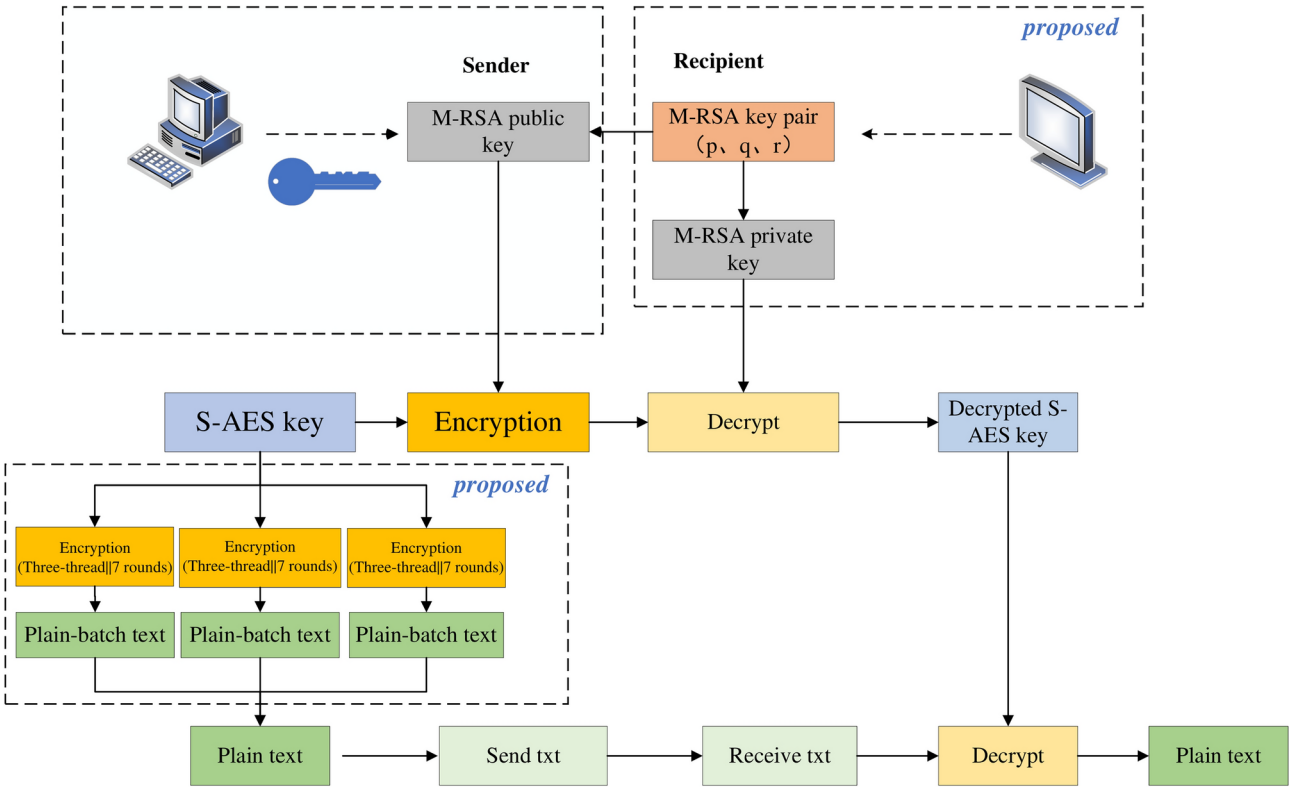

Fig. 6. Hybrid encryption basic process.

3. Use 3 prime numbers again to calculate the Euler function value:

$$\varphi(n) = (a-1)(b-1)(c-1); \tag{5}$$

4. Choose an e value as the public key. The public key should be as large as possible. This value satisfies $1 < e < \varphi(n)$and e and $\varphi(n)$ are relatively prime;
5. Then use e and the Euler function value to calculate the private key:

$$e \times d \equiv 1(\mathrm{mod}\varphi(n)); \tag{6}$$

6. The public key of the password is (e, n) and the private key is (d, n);
7. Combine n and e into the public key, and n and d into the private key, so the public key is (n, e) and the private key is (n, d);
8. The plaintext is M and the ciphertext is:

$$C \equiv M^e(\mathrm{mod}n); \tag{7}$$

9. The decryption process is as follows:

$$M \equiv C^d(\mathrm{mod}n); \tag{8}$$

Fusion model of RSA and AES algorithms (MRA)

The RSA algorithm is based on the mathematical challenges of the large prime number decomposition problem and modulo power operations, for which no effective solution has yet been found, and therefore has a high level of security. However, the RSA algorithm involves multiplication and modulo operations of large integers, which requires high computational resources, especially when the key length is large, and the computational complexity of encryption and decryption is high. And the RSA algorithm is suitable for encrypting small chunks of data, but not for direct encryption of large data because the computational overhead of encrypting and decrypting large amounts of data is too great. Due to the high computational complexity of the RSA algorithm, especially when the key length is large, the performance of encryption and decryption is low, which affects the speed of real-time data encryption and decryption.

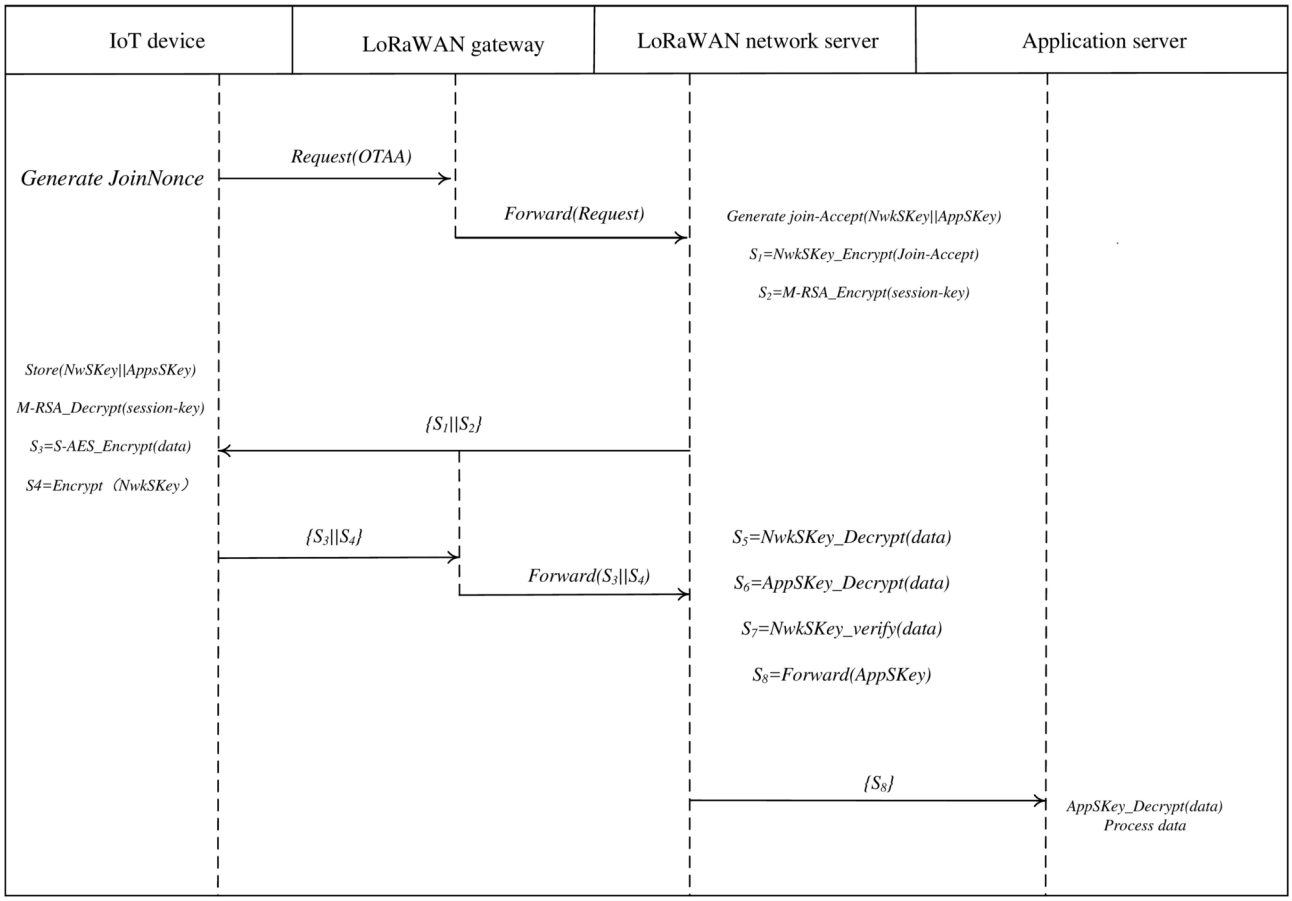

Fig. 7. The MRA proposed by us is intended to replace the AES-128 in the LoRaWAN protocol.

AES algorithm is simple in design, efficient in implementation, fast in encryption and decryption, especially suitable for encrypting and decrypting large amount of data. Moreover, AES algorithm supports multiple key lengths (128-bit, 192-bit, 256-bit), so you can choose the appropriate key length to balance security and performance according to your needs[31]. However, the AES algorithm uses symmetric key encryption, where the encrypting and decrypting parties need to share the same key, and therefore need to securely manage and distribute the key, which can be challenging in large-scale systems. Because AES is a symmetric encryption algorithm, there is no public-private key mechanism like RSA, so it can not directly realize the functions such as digital signatures, and need to work with other algorithms to achieve. In order to solve the above problems, this paper considers the combination of AES algorithm and RSA algorithm to encrypt the key of S-AES using M-RSA algorithm. The information data is encrypted using S-AES. This utilizes both the security of the M-RSA algorithm and the fact that AES can encrypt and decrypt large amounts of data at high speed. The basic process of hybrid encryption is as follows Fig. 6:

In the above figure, the sender of the data first encrypts the plaintext of the data with the S-AES key to get the ciphertext and then encrypts the S-AES key with M-RSA. The ciphertext and the encrypted S-AES key are then sent to the receiver.

Then the receiver first receives the private key of M-RSA, then decrypts the key of S-AES algorithm by using the private key, and finally decrypts the ciphertext by using the key for decryption operation[25].The “proposed” part is different from the traditional AES-RSA hybrid encryption algorithm.

Both ensure the security of data transmission and improve the efficiency of the encryption and decryption process. The realization of the process code is as follows:

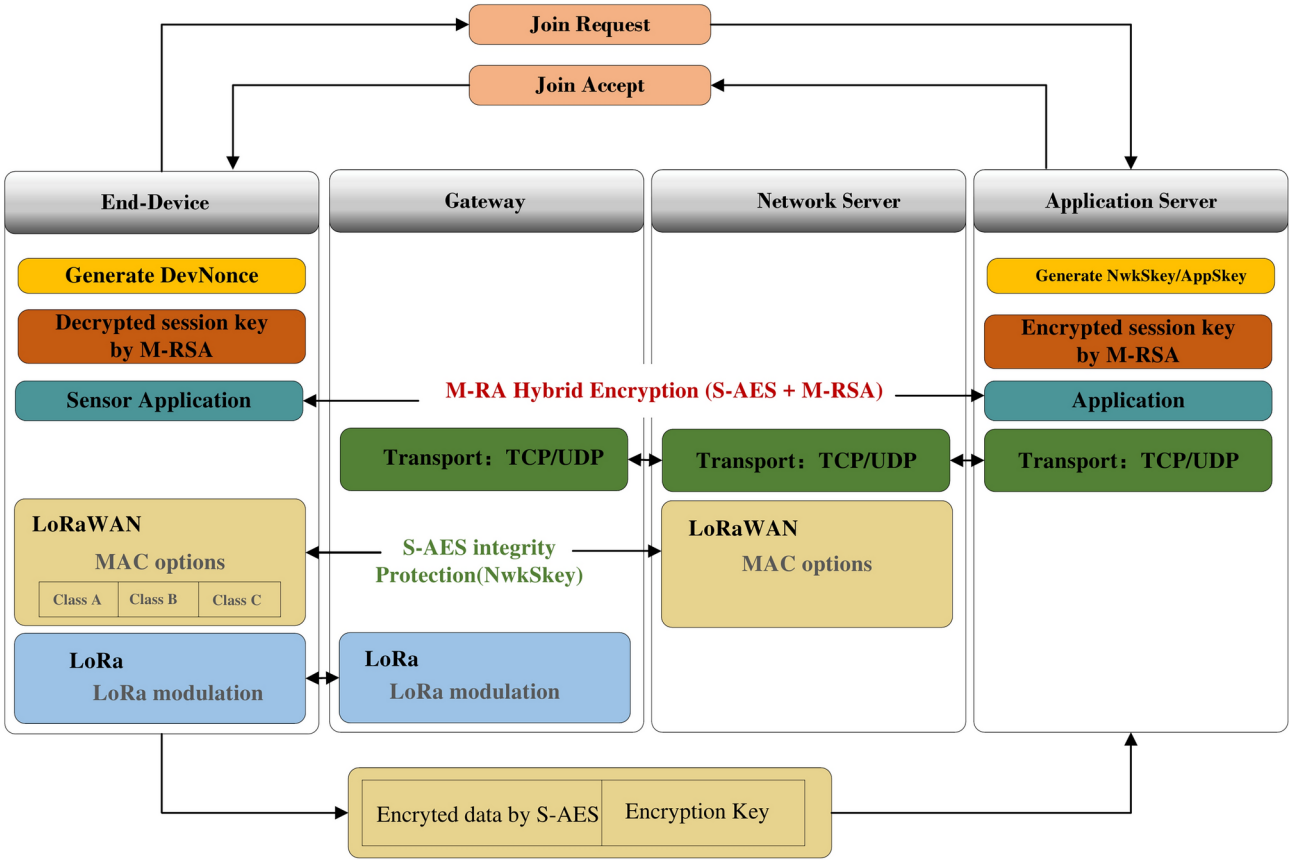

Fig. 8. MRA in the LoRaWAN protocol's encryption and decryption process.

Input: $Plaintext, AES\_key, RSA\_PublicKey(3-prime-RSA)$
Output: $Encrypted\_Data$
If AES_Key **and** $RSA\_PublicKey$ *are available* **then**
  $cipher\_RSA = PKCS1\_0AEP\_New(RSA\_publicKey)$ ▷ 3 prime RSA
  $Encrypted\_AES\_Key = RSA\_Cipher\_encrypt(AES\_KEY)$
  $Data\_to\_Encrypt \leftarrow plaintext$
  $padded\_data = data + b'n0' \times (16 - len(Data\_to\_Encrypt \leftarrow plaintext)\%16)$
▷ data padding
  $Encrypted\_data = AES\_Cipher(AES\_Key).encrypt(pedded\_data)$
  $lorawan.send.data(Encrypted\_AES\_Key + Encrypted\_data)$
Else Return $false$
Return $Encrypted\_Data$

Algorithm 3. End node algorithm

The end node encrypts sensitive data using a combination of the improved RSA (3-prime RSA) and the improved AES algorithms. First, the improved RSA algorithm encrypts the AES key to ensure secure transmission. Then, the data is padded to meet the AES block size requirement and encrypted using the improved AES algorithm. Finally, the encrypted AES key and encrypted data are concatenated and sent to the LoRaWAN server. This ensures both the key and the data are securely transmitted.

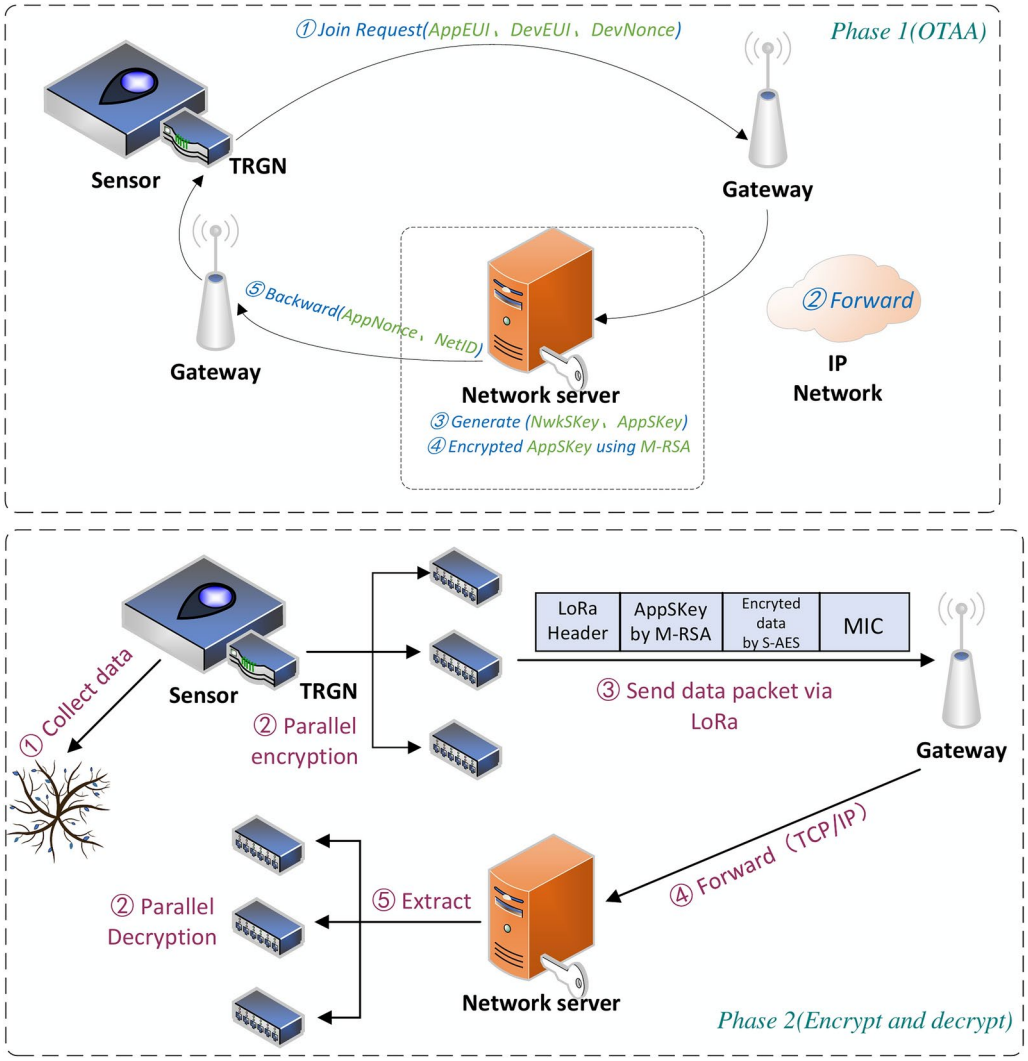

Fig. 9. Step-by-step encryption and decryption in IoT.

Input: $Encrypted\_data, RSA\_PrivateKey(3-prime-RSA)$
Output: $Decrypted\_Plaintext$
$Encrypted\_data = Lorawan.Encrypted\_data()$
Initialize RSA_Cipher **using** $RSA\_PrivateKey$ for $3-prime-RSA$
$Decrypted\_AES\_Key = RSA\_Cipher.decrypt(Encrypted\_AES\_Key)$
If $Decrypted\_AES\_Key$ **is** $valid$ **then**
  Initialize AES_Cipher **With** $Decrypted\_AES\_Key$
  $Padded\_Plaintext = AES\_Cipher.decrypt(Encrypted\_Data)$
  $Plaintext = RemovePadding(Padded\_Plaintext)$ ▷ Remove Padding
Else
  Return $False$
Return $Plaintext$

Algorithm 4. Lorawan web server node algorithm

In algorithm 4,the LoRaWAN server decrypts the incoming data by first using the improved RSA algorithm to retrieve the AES key. The encrypted AES key, which was sent along with the data, is decrypted using the server's private RSA key. With the decrypted AES key, the server initializes the AES cipher to decrypt the data. Finally, padding is removed to retrieve the original plaintext. This method ensures secure handling of both the key and data during transmission.

(* Define types and cryptographic primitives *)
type key. (* Indicates the key type *)
type message. (*Generic message type for transferring data.*)
fun m-rsa_encrypt(key, key): key. (*Utilize M-RSA public key encryption key.*)
fun m-rsa_decrypt(key, key): key. (*Utilize M-RSA private key encryption key.*)
fun s-aes_encrypt(key, key): key. (*Encrypts data using an AES key.*)
fun s-aes_decrypt(key, key): key. (*Decrypting Data with AES Keys*)
fun hash(message): message. (*Hash function for data integrity verification*)
(* Define free variables (public keys) and private keys *)
free pubKey_device : key.
free pubKey_gateway : key.
free pubKey_netServer : key.
free pubKey_appServer : key.
free privKey_device : key[private].
free privKey_gateway : key[private].
free privKey_netServer : key[private].
free privKey_appServer : key[private].
const SensorData : key.
(* Define secrets *)
free sessionKey : key[private]. (*The session key is set to private*)
(* Define channel *)
free c : channel.
free gw_ns : channel. (*Channel between gateway and web server*)
free ns_as : channel.(*The channel between the web server and the application server*)
(* Define event *)
event data_received(message).

Table 2. Definition of code.

let device_process =
let encrypted_sessionKey = m-rsa_encrypt(pubKey_netServer, sessionKey) in
out(c, encrypted_sessionKey);
let data_message = SensorData in
let encrypted_data = s-aes_encrypt(sessionKey, data_message) in
out(c, encrypted_data).
let gateway_process =
in(c, encrypted_sessionKey: message);
out(gw_ns, encrypted_sessionKey);
in(c, encrypted_data: message);
out(gw_ns, encrypted_data).
let netServer_process =
in(gw_ns, encrypted_sessionKey: key);
let sessionKey = m-rsa_decrypt(privKey_netServer, encrypted_sessionKey) in
in(gw_ns, encrypted_data: key);
let data_message = s-aes_decrypt(sessionKey, encrypted_data) in
out(ns_as, data_message).
let appServer_process =
in(ns_as, data_message: message);
event data_received(data_message).
query attacker(m-rsa_decrypt(privKey_netServer,m-rsa_encrypt(pubKey_netServer, sessionKey))).
query attacker(s-aes_decrypt(sessionKey, s-aes_encrypt(sessionKey, SensorData))).
process
(! device_process | ! gateway_process | ! netServer_process | ! appServer_process)

Table 3. Defination of process.

```
--------------------------------------------------------------
Verification summary:

Query not attacker(mrsa_decrypt(privKey_networkServer[],mrsa_encrypt(pubKey_networkServer[],sessionKey[]))) is true.

Query not attacker(saes_decrypt(sessionKey[],saes_encrypt(sessionKey[],SensorData))) is true.

--------------------------------------------------------------
```

Fig. 10. Results of proverif.

Verification summary:
Query not attacker(m-rsa_decrypt(privKey_netServer[],m-rsa_encrypt(pubKey_netServer[],
sessionKey[]))) is true.
Query not attacker(s-aes_decrypt(sessionKey[],s-aes_encrypt(sessionKey[].SensorData))) is true.

Table 4. Results for code.

MRA in LoRaWAN protocol

Proposed Workflow Integration of Enhanced RSA-AES in LoRaWAN Protocol Fig. 7: The LoRaWAN protocol, enhanced with a hybrid RSA-AES cryptographic scheme, optimizes secure communication for IoT devices. Initially, during the Over-The-Air Activation (OTAA) process, devices dynamically generate session keys, NwkSKey and AppSKey, which are encrypted using an improved RSA algorithm and securely transmitted to the network server. Upon receipt, the network server decrypts the keys using its corresponding RSA private key. These keys facilitate subsequent data encryption via a modified AES algorithm, ensuring both the confidentiality and integrity of the transmitted data. For devices activated using Activation By Personalization (ABP), the process bypasses dynamic key generation, utilizing pre-configured NwkSKey and AppSKey, while still employing the enhanced AES for data encryption. This integration significantly strengthens LoRaWAN's security framework, particularly against sophisticated threat models, while maintaining efficient communication across the network. This protocol illustrates the integration of the hybrid encryption algorithm (improved RSA and AES) into the LoRaWAN architecture. The process begins with the IoT device generating a JoinNonce and sending an Over-The-Air Activation (OTAA) request to the LoRaWAN gateway. The gateway forwards the request to the LoRaWAN network server, which generates the join-accept message containing the session keys (NwkSKey and AppSKey). The join-accept message is encrypted using the improved AES (S1), and the session key is encrypted using the improved RSA (S2). Both encrypted messages are sent back to the IoT device, where the improved RSA decrypts the session key, and the AES decrypts the join-accept message. The IoT device then encrypts the data using AES (S3) and further encrypts it with the NwkSKey (S4) before transmitting it to the gateway. The network server decrypts the data using NwkSKey (S5) and verifies its integrity (S7). The data is then forwarded to the application server, where it is decrypted with AppSKey (S6) for processing. This process ensures robust encryption for both session key exchange and data transmission, leveraging the strengths of the hybrid algorithm to enhance security in the lightweight LoRaWAN environment. To facilitate a clearer visualization of the encryption and decryption process of MRA within the LoRaWAN framework, refer to Fig. 8. In contrast to the conventional AES-128 encryption procedure employed in the standard LoRaWAN protocol, our salient contribution lies in the substitution of the AES-128 encryption with MRA. Specifically, the AppSKey is now transmitted via M-RSA encryption. The network server generates a temporary S-AES session key, which is encrypted using the M-RSA public key. The encrypted key is then transmitted along with the S-AES encrypted data. We maintain the original MAC options (e.g., ADR, rate control), altering only the application-layer encryption. The decrypted S-AES key is utilized to generate the MIC, ensuring consistency in integrity verification.

In practice, the OTAA activation process of the LoRaWAN protocol and the step-by-step data flow diagram of encryption/decryption in IoT devices are shown in Fig. 9

Security analysis and results

Formal verification using ProVerif tool

Here we validate this model using the ProVerif[32] tool.ProVerif uses a process-based algebraic approach to model and analyze protocols. It automatically verifies that a protocol satisfies specific security properties such as confidentiality, authentication, and unforgeability. The tool runtime allows unlimited message space and session simulation, and provides common cryptographic operations that include symmetric and asymmetric encryption/decryption, signatures, and hash functions[32] .

In the ProVerif script, we establish a communication protocol encompassing four key entities: the device, gateway, network server, and application server. In Table 2, we define the public and private keys, encryption and decryption operations, and communication channels.In Table 3, the communication process among four different entities is shown, detailing whether the session key and other information are forwarded to the channel. The query() statements are then used to verify whether an attacker can compromise the protocol's confidentiality. In Fig. 10 and Table 4 are the verification results from the ProVerif tool, showing that an attacker cannot obtain the decrypted information or other critical contents such as the session key.

Here is a step-by-step explanation of the ProVerif formal verification process:

- Define cryptographic primitives and types. Clearly specify the cryptographic operations and data types used in the protocol, with specific parameter definitions provided in the notes of Table 2.
- Modelling communication channels.Define the communication links between entities in the protocol and their security assumptions.Public channel (c): the channel between the device and the gateway, assuming an attacker can listen to this channel.Internal channels (gw_ns, ns_as):Attackers cannot directly listen to or tamper with internal channels and need to influence them indirectly through man-in-the-middle attacks..
- Protocol process modelling.Translate the behaviour of the four entities in the protocol into parallel processes in ProVerif.

  (1) Device process (device_process):First, generate the session key sessionKey. Encrypt the session key using the public key of the network server: $encrypted\_sessionKey = m - rsa\_encrypt(pubKey\_netServer, sessionKey)$.Then encrypt the sensor data using S-AES: $encrypted\_data = s - aes\_encrypt(sessionKey, SensorData)$. Send the encrypted session key and data over public channel *c*.
  (2) Gateway_process: receives encrypted session key and data from channel c. Forwards the message to the web server channel gw_ns.
  (3) Network server process (netServer_process): receives encrypted session key from gw_ns, decrypts using private key: $sessionKey = rsa\_decrypt(privKey\_netServer, encrypted\_sessionKey)$. Decrypt the data:$data\_message = aes\_decrypt(sessionKey, encrypted\_data)$. Forward the decrypted data to the application server tract ns_as.
  (4) Application server process (appServer_process): receives decrypted data fromns_as and triggers the $data\_received$ event, indicating that the data arrived safely.

- Defining Security Attribute Queries.

  (1) Enquiry command:query_attacker(m-rsa_decrypt(privKey_netServer[], m-rsa_encrypt(pubKey_netServer[], sessionKey[]))). The implication is whether an attacker can decrypt the session key from the intercepted public key and ciphertext?Verification result: *true* indicates that the attacker cannot decrypt it, proving the confidentiality of the M-RSA-encrypted session key.
  (2) Enquiry command:query_attacker(s-aes_decrypt(sessionKey[],s-aes_encrypt(sessionKey[], SensorData))).Meaning: can an attacker decrypt the original data through the intercepted AES cipher and session key?Verification result: *true* indicates that the attacker cannot decrypt the data, proving the data confidentiality of AES encryption.

Through the above steps, ProVerif formal verification proves the security of the MRA algorithm in the LoRaWAN protocol, especially against passive eavesdropping and active decryption attacks.This process provides rigorous formal support for the theoretical security of the protocol.

Game-based security analysis

In this chapter, to demonstrate the security of the improved algorithm, we will analyze its defense capabilities against adversaries with multiple attack capabilities by combining the Random Oracle Model[33] with a game theory-based framework. The Random Oracle Model assumes that the hash function behaves like a random function, meaning that its output is independent and unpredictable for each new input. Under this assumption, attackers can only gather information by querying the Random Oracle. We will use this model to evaluate the security of the RSA-AES hybrid algorithm. Based on the threat model established in Chapter 2, the adversary is assumed to possess the following capabilities:

1. $Send(A_i, B_j, MSG)$:Sending queries model replay attacks on sent messages.
2. $Execute(A_i, B_j)$:Ability to passively listen to communication data between the device and the gateway.
3. $Mitm(A_i, B_j)$:Ability to perform man-in-the-middle attacks to hijack communications between devices and gateways to tamper with or falsify data.
4. $Threat(A_i, B_j)$:The ability to masquerade as a legitimate device or gateway in order to trick the other party into communicating.

We introduce the following lemma and theorem to better determine the attacker's advantage:

Difference Lemma[34]: Assume that the attacker's probability of success at a given point in time is *Pr*[*A*], and the probability of success of the attack after the defender has incorporated some defense strategy is *Pr*[*B*],if A and B perform the same operation without the occurrence of the "failure event" event F :

$$|Pr[A] - Pr[B]| \leq Pr[F] \tag{9}$$

CCM-Mode Theorem Using PRP[35] :Suppose F is a secure PRP from $\{0,1\}^n$ to$\{0,1\}^n$,Let *qE* denote the number of cryptographic queries performed by the attacker, *qT* be the number of MAC authentications performed by the attacker, and *n* denote the length of the grouped cipher block (in bits). Then the advantage of adversary $\mathcal{A}$ against CCM is:

$$Adv^{cpv}_{ccm}(\mathcal{A}) \leq qT \cdot Adv_{prp}(E) + qE \cdot \frac{qE + qT}{2^n} \tag{10}$$

For any secure PRP,$Adv_{prp}(E)$is negligible.Thus CCM provides $2^{\frac{n}{2}}$ authenticity group cipher calls. *Theorem(SK-Secure)* :Let $\mathcal{A}$ be a Probabilistic Polynomial Time(PPT) adversary adversary,with q_s *Send* queries,q_h *hash* queries,q_e *Execute* queries.Let fig7 be denoted as *m1*.Let *l* denote the CCM packet cipher bit length.Then the advantage of $\mathcal{A}$ over the session key is that

$$Adv_{m1}^{ss}(\mathcal{A}) \leq qT \cdot Adv_{prp}(E) + Adv_{es}(E) + \frac{q_s + (q_s + q_e)^2 + q_h^2}{2^l} \tag{11}$$

Since both $Adv_{prp}(E)$ and $Adv_{es}(E)$ can be ignored, Adv_{m1}^{ss}can be ignored until $2^{l/2}$ calls of all queries. Therefore, *l* can be made long enough to enhance security.

Proof Completing the G0-G5 game, we can get that the MRA algorithm is able to effectively resist a variety of complex attack methods, thus significantly improving the security of the protocol.In the following description U_i denotes the ith IoT device, GW_idenotes the ith gateway node, and so on the corner icon i denotes the ith node device and *NS* denotes the network server.

*Game G0:*The protocol is based on Random Oracle.We get :

$$Adv_{ccm}^{CPA}(\mathcal{A}) = 2Pr[S_0] - 1 \tag{12}$$

If the time imposed by attacker A exceeds the threshold, then it will be automatically terminated.

*Game G1:*Simulation of stochastic Oracle based on the proposed scheme.For *Send*, *Execute*, *Mitm*, *Threat*, respectively, the following queries are performed: $Send(U_i, GW_1, AppNonce)$, $Execute(U_i, GW_1)$,$Mitm(U_i, GW_2)$,$Threat(U_i, GW_1)$,*AppNonce* represents the random number generated by the web server during the generation of NwkSKey. Three lists are used to store the query results separately:

1. L_H:Used to save hash query results.
2. L_P:Used to keep a log of the proof process

G0 and G1 are indistinguishable,getting $Pr[S_1] = Pr[S_0]$

*Game G2:*Collisions in CCM are computed here. A collision may occur if two trusted sessions choose the same random number or if a trusted party chooses a random number provided by an adversary.According to the birthday paradox, if *q* queries are made to a random function of *l* bits, then the probability of a collision is within $\frac{q^2}{2} \cdot 2^{-l}$.Since at most $(q_S + q_e)$ pairs of random numbers are selected in all the sessions initiated by U_i to GW_i and the corresponding sessions generated by Network Server to U_i.Thus the probability of conflict is maximized to be $\frac{(q_s+q_e)^2}{2^{l+1}}$. In addition, adversary A can also send $\frac{q_s}{2^l}$ pairs of requests to the responding party According to differential priming we have:

$$|Pr[S_2] - Pr[S_1]| \leq \frac{(q_s + q_e)^2}{2^{l+1}} + \frac{q_s}{2^l} \tag{13}$$

*Game G3:*In this game, we consider the possibility that adversary $\mathcal{A}$ will simulate the wrong *NwkSKey* in the absence of a random Oracle.Therefore some new operations need to be added to $Send(A_i, B_j, MSG)$ and the message needs to be checked.

1. For $Send(A_i, B_j, MSG)$:The simulator must check if$(S1||S2, *), (Request(OTAA), *) \in L_H$.The probability is $\frac{q_s+q_h}{2^l}$at most.
2. For $Send(GW_1, NS_1, JoinNonce)$:The simulator must check if$(S3||S4, *) \in L_H$, $(Request(OTAA), *) \in L_P$.The probability is $\frac{q_s+2q_h}{2^l}$at most.

Given these simulations, G2 and G3 are indistinguishable. Then we get:

$$|Pr[S_3] - Pr[S_2]| \leq \frac{(q_s + q_h)}{2^l} \tag{14}$$

*Game G4:*In this game we exclude adversary $\mathcal{A}$ from being lucky enough to obtain*AppNonce* and *JoinNonce*. if adversary $\mathcal{A}$ has access to the keys in the data encryption algorithm, then consider adversary $\mathcal{A}$ to have solved the problem. Since the key deployed in a lightweight device cannot be too long otherwise it will exceed the device arithmetic and affect the efficiency, it is considered that the adversary will perform a guessing attack. Since *AppNonce* and*JoinNonce* are short and fresh random numbers generated for each session. Therefore the probability of success will not exceed $\frac{2q_s}{2^l}$.From the above, unless a succeeds in guessing *AppNonce* and *JoinNonce*. the fourth and fifth games are the same way. Then:

$$|Pr[S_4] - Pr[S_3]| \leq \frac{2q_s}{2^l} \tag{15}$$

*Game G5:*In this game game we consider strong forward secrecy. A temporary RSA key pair is generated in each session. This game just affects the old game and the old log in L_p becomes a source of resources for

the adversary.Assuming that the probability that we get $(AppNonce||JoinNonce||NwkSKey) \in L_p$ and get (*AppNonce*||*JoinNonce*) in a session is $\frac{1}{(q_s+q_e)^2}$, then we have:

$$|Pr[S_5] - Pr[S_4]| \leq 2q_h(q_s + q_e)^2 \cdot Adv_{es}(E) \tag{16}$$

Hence, A loses the edge in surmising,and $Pr[S5] = \frac{1}{2}$,thus this theorem is proved.In order to represent the process of the game more intuitively, we list a clearer pseudocode representation here algrithm 5:□

Initialize: $Set\ security\ parameters(l, n), adversary\ queries(q_s, q_e, q_h)$
Game G0: Simulate protocol with Random Oracle, compute attacker advantage $Adv_{ccm}^{CPA} = |2Pr[S0] - 1|$
For G_0**to** G_5 **:**
$G_1 \rightarrow G_2 : Calculate\ collision\ probability\ \Delta_1 = \frac{(q_s+q_e)^2}{2^{l+1}}$
$G_2 \rightarrow G_3 : Validate\ message\ integrity, \Delta_2 = \frac{q_s+q_h}{2^l}$
$G_3 \rightarrow G_4 : Check\ key\ guessing, \Delta_3 = \frac{2q_s}{2^l}$
$G_4 \rightarrow G_5 : Enforce\ forward\ secrecy, \Delta_4 = 2q_h(q_s + q_e)^2 Adv_{es}$
Finally Advantage: $Adv_{Total} = \Delta_1 + \Delta_2 + \Delta_3 + \Delta_4 \rightarrow negligible$
Return If $Adv_{Total} < threshold, protocol\ is\ secure$

Algorithm 5. Proof of Security Based on Game Theory

Informal security analysis

Besides, in IoT environments, different entities (e.g., Network Server(NS), IoT devices, and Data Users) may be exposed to various malicious attacks. Our scheme be capable of resist the following Common Attacks Targeting LoRaWAN.

- $Prevention\ of\ Man-in-the-Middle\ Attack$: The MRA scheme effectively resists man-in-the-middle attacks through several security mechanisms.

1. All key components, such as private keys and pre-encryption parameters *Nwkskey*, *AppSKey*, are transmitted securely,preventing tampering or theft during transmission.
2. The scheme uses strong encryption and hash functions to ensure the integrity and confidentiality of messages, making it impossible for an attacker to effectively decrypt or forge data even if they intercept the ciphertext.
3. The design of the protocol incorporates intricate verification procedures for data integrity, mandating that any decrypted messages undergo stringent integrity assessments prior to any additional handling, thus efficiently safeguarding against interference or manipulation attempts by intermediaries.

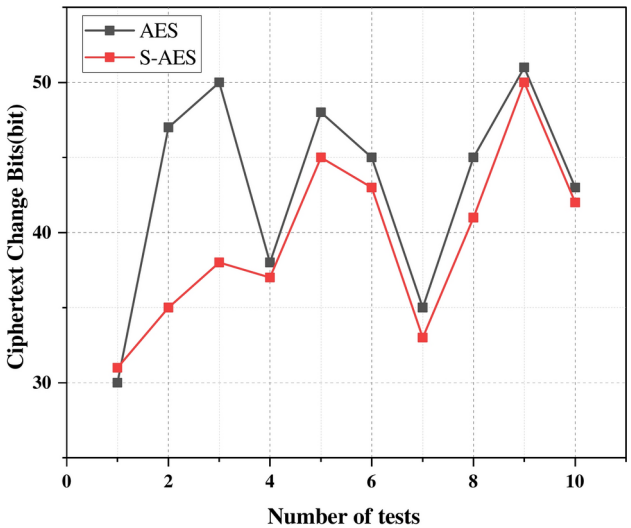

Fig. 11. S-AES vs. AES expansion test comparison chart .

Algorithm type	Time complexity
Trial division	$O(\sqrt{n})$
Prime factorization method	$O(\sqrt{n})$
Fermat method	$O(\sqrt[4]{n})$
Pollard Rho	$O(\sqrt[4]{n})$
Number field sieve method	$O(e^{(1.9+o(1))\sqrt{\ln n \ln\ln n}})$
Elliptic Curve Decomposition Algorithm	$O(e^{(2+o(1))\sqrt{\ln n \ln\ln n}})$

Table 5. Time complexity of different factorization algorithms.

Modulus length(bit)/Time consumption(ms)	Tradition-RSA	M-RSA	Fourprime numbers-RSA
60	979	1432	1121
90	2180	2896	2571
120	4201	4031	4289
150	25214	24647	31645
180	73236	98481	89243

Table 6. Factorization time of each algorithm under different modulus lengths.

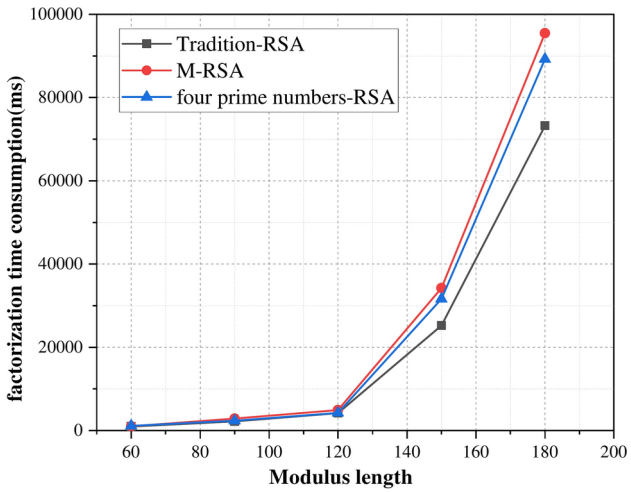

Fig. 12. Comparison of factorization time consumption .

- *Protection Against Data Leakage Attacks*:

  1. Even if an attacker succeeds in compromising a semi-honest compute server or *NS*, they will only gain access to the *AppEUI*, not the *AppSKey*. the private key is controlled by the receiver in the scheme and is critical to the decryption process.
  2. Since the encryption process involves complex key generation and random number mechanisms, an attacker cannot decrypt data or impersonate a legitimate user without a private key, even if he or she has DevNonce. Therefore, protecting the private key and preprocessing parameters is crucial to ensure that the data is protected even if the server is compromised.

Formal security analysis of 7-round AES based on mathematical analysis

Preliminary knowledge and symbol definitions

Let $AES^{(r)}: \mathbb{F}_2^{128} \to \mathbb{F}_2^{128}$ represent the AES encryption function with r rounds of operation. For the standard$AES-128$, $r = 10$. The variant proposed in this paper is denoted as $S-AES^{(7)}$, which reduces the number of rounds to $r = 7$. We analyze its security based on the following attack model:

1. *Differential cryptanalysis* Let $\Delta_{in}, \Delta_{out} \in \mathbb{F}_2^{128}$ be the input and output differentials. The differential probability (DP) is defined as:

Key length(bit)/Attack time consumption(s)	Tradition-RSA	MRA
8bit	1.57	13.265
10bit	7.621	35.231
12bit	91.265	265.651
16bit	159.424	476.326

Table 7. Comparison of time taken to resist brute-force attacks.

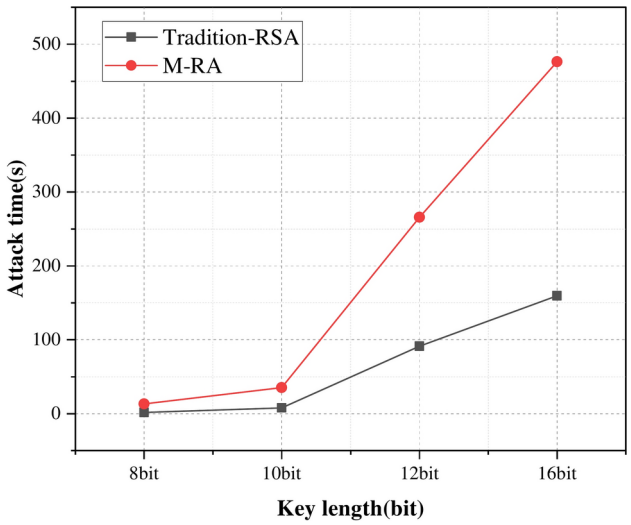

Fig. 13. Comparison of RSA and MRA deciphering time consumption.

$$DP^r(\Delta_{in}, \Delta_{out}) = Pr[AES^{(r)}(X \oplus \Delta_{in}) \oplus AES^{(r)}(X) = \Delta_{out}] \tag{17}$$

2. *Linear cryptanalysis* Let $\alpha, \beta \in \mathbb{F}_2^{128}$ be the input and output masks. The linear bias (LB) is defined as:

$$\epsilon^r(\alpha, \beta) = |Pr[\alpha \cdot X \oplus \beta \cdot AES^r(X) = 0] - \frac{1}{2}| \tag{18}$$

3. *Integral attack* Let S be a set of plaintexts with a specific structure. The attack recovers the key by analyzing the balance of S after r rounds of encryption.

Differential security analysis

1. *Lower bound on the number of active S-boxes* The MixColumns and ShiftRows operations ensure that any non-zero input difference activates at least 25 S-boxes after 4 rounds (complete diffusion theorem, Daemen & Rijmen[36], 2002). For a 7-round operation, the minimum number of active S-boxes is:

$$N_{activate}^7 = 25 \times 7 = 175 \tag{19}$$

2. *Differential probability of the S-box* For the AES S-box $S : \mathbb{F}_2^8 \to \mathbb{F}_2^8$, its maximum differential probability is limited to:

$$DP_{S-box} = \max_{\Delta_{in} \neq 0, \Delta_{out}} Pr[S(x \oplus \Delta_{in}) \oplus S(x) = \Delta_{out}] \leq 2^{-6} \tag{20}$$

3. *Total differential probability* The Maximum Expected Differential Probability (MEDP) of r-round AES is dominated by the product of the probabilities for each round:

$$MEDP^r \leq (DP_{S-box})^{N_{active}^{(r)}} = (2^{-6})^{175} = 2^{-1050} \tag{21}$$

Since $2^{-1050} \ll 2^{-128}$, differential attack is infeasible.

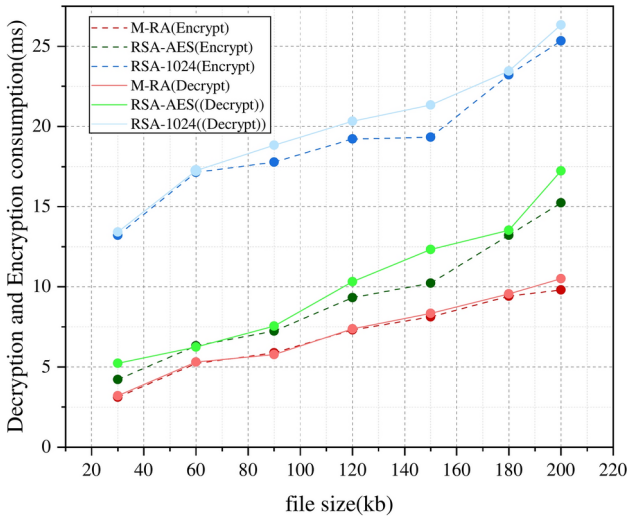

Fig. 14. Comparison of en-deciphering time consumption .

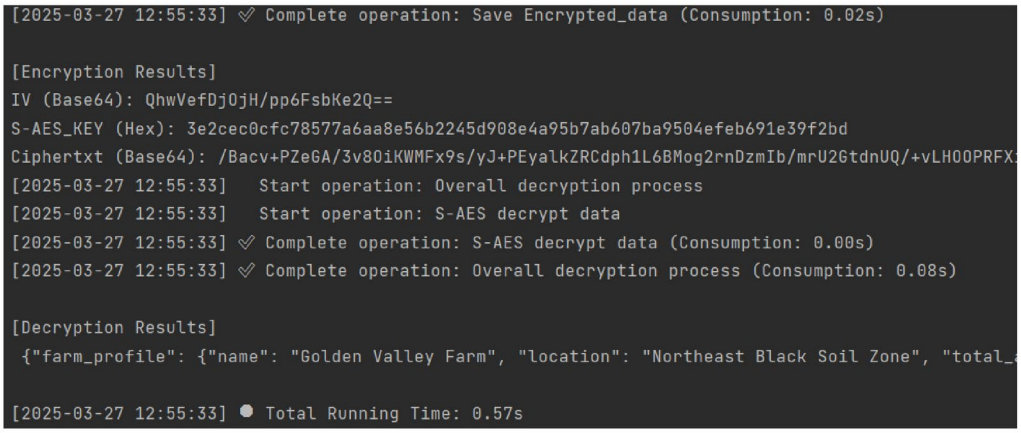

Fig. 15. Encryption and decryption execution log.

Algrithm	latency	Protective capability	Efficiency improvement
AES-RSA[15]	6ms-18ms	×	68%
RSA-TDES[10]	10ms-25ms	MITM	55%
CL-EAED[41]	10ms-20ms	MITM	65%
TLADE[42]	8ms-18ms	×	70%
Ours	**5ms-15ms**	**MITM,RPA**	76%

Table 8. Comparison table of execution time, security level, and efficiency between OURS, AES-RSA, RSA-TDES, CL-EAEDand TLADE encryption. Efficiency is the improvement margin compared to RSA-1024.

Resistance to linear cryptanalysis

1. *The linear bias of the S-box* The maximum linear bias of the S-box in AES is:

$$\epsilon_{S-box} = \max_{\alpha,\beta\neq 0} |Pr[\alpha \cdot x \oplus \beta \cdot S(X) = 0] - \frac{1}{2}| \leq 2^{-3} \tag{22}$$

2. *Bias accumulation* For $N_{active}^{7} = 175$active S-boxes, the total bias is:

```
** Event #1  t=0.01018  Net1.IoTDevice (IoTDevice, id=2)  on selfmsg timer (omnetpp::cMessage, id=0
INFO (IoTDevice)Net1.IoTDevice: send packet(Ours)
** Event #2  t=0.02036  Net1.PC (Node, id=3)  on timer (omnetpp::cMessage, id=0)
INFO (Node)Net1.PC: receive packet(Ours)
INFO (Node)Net1.PC: delay = 0.010180 s
INFO (Node)Net1.PC: energy consumption = 0.502000 mJ
```

Fig. 16. Simulation details for a IoT device in OMNeT++.

```
[INFO] 2.599s (LoRaNetwork.node[2].mac) - Packet[seq=3532] sent (size=54B) | Latency=5.44ms Energy=0.09mJ Throughput=17.02Kbps
[INFO] 2.828s (LoRaNetwork.node[7].mac) - Packet[seq=9126] sent (size=68B) | Latency=5.84ms Energy=0.35mJ Throughput=44.69Kbps
[INFO] 3.085s (LoRaNetwork.node[10].mac) - Packet[seq=5734] sent (size=89B) | Latency=9.56ms Energy=0.57mJ Throughput=97.09Kbps
```

Fig. 17. Simulation details for a IoT device in Ubuntu.

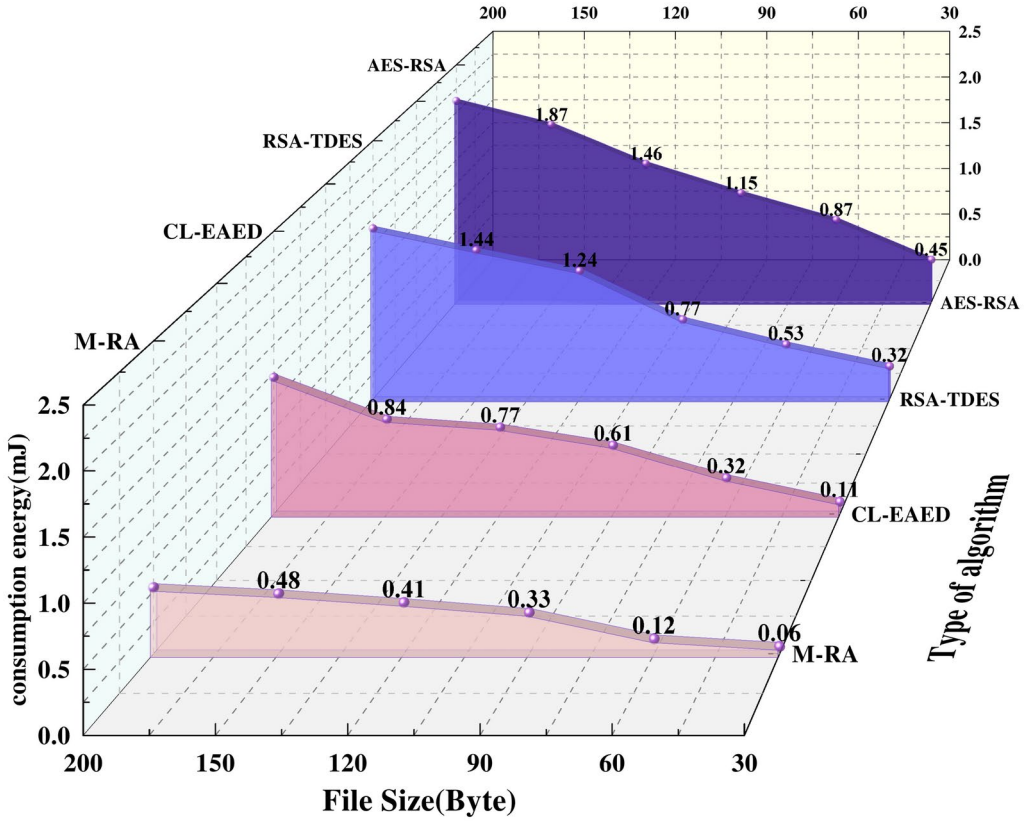

Fig. 18. Simulation details for a IoT device in Ubuntu.

$$\epsilon^7 \leq (2^{-3})^{175} = 2^{-525} \tag{23}$$

The amount of data required to successfully implement a linear attack is:

$$N_{data} \approx (\epsilon^{(7)}) = 2^{1050} \tag{24}$$

This amount of data far exceeds 2^{128} that threshold of brute force attacks , hence linear attacks are not practically meaningful.

Immunity to integral attacks
After 4 rounds of operation, MixColumns ensures that each byte of the state depends on all 16 input bytes (Daemen & Rijmen[36], 2002). For 7 rounds, the additional 3 rounds of operation enhance the non-linearity, completely destroying the balance of S.

Resistance to other attacks

1. *Biclique attack* Applies only to full-round AES (Biryukov & Khovratovich[37], 2011). For 7 rounds, the Biclique structure cannot efficiently divide the key space.
2. *Algebraic attack* The S-box of AES is defined by the inverse operation over $\mathbb{F}_{2^8}$ and affine transformations, resulting in complex systems of multivariate equations. Attacks based on Gröbner[38] bases remain computationally infeasible.

S-AES security analysis comparison

Here we take the algorithm expandability as the test criterion for security testing. Under the premise of ensuring that the key is not changed, the expandability test means that we change the bits of each bit of plaintext data and observe the bit changes of the ciphertext data generated with it, so as to analyze the impact of the encrypted ciphertext. The expandability of the improved algorithm is compared with the expandability of the traditional AES algorithm. We use the S-AES algorithm and the AES algorithm to encrypt the same size 128-bit block with the following results. In order to show the difference between the two algorithms more intuitively, we plotted it as a line graph as follows:

From the above Fig. 11, we can see that the traditional AES algorithm varies from 30 to 51 bits, while the S-AES algorithm varies from 31 to 50 bits, which is similar to the range of variation, so there is not much difference in expandability.S-AES reduces the number of rounds from 10 in the standard AES to 7, significantly improving encryption efficiency by optimizing the S-box (SubBytes) and parallel thread processing while maintaining the effectiveness of confusion and diffusion. The bit length of the key generation remains unchanged, still at 128 bits or 256 bits, and no effective differential or linear attacks have been found[39]. The logic of key generation or the entropy source has not been altered, hence the randomness of the key remains unaffected.

M-RSA security analysis comparison

The security strength of RSA algorithm mainly depends on the difficulty of factoring modulo a modulus, so we use the time consumption for factoring large prime numbers[40] as an index to evaluate the security of RSA algorithm. The current common factorization algorithms are division by trial, decomposition of prime factors, fermat's method, pollard rho method, number field sieve, elliptic curve decomposition algorithms, etc. Their time complexity is shown as Table 5.

In order to test the factorization time consumption, here we choose the elliptic curve decomposition algorithm for the traditional RSA as well as the multi-prime RSA algorithm to perform the factorization operation as a way to control the variables. Because of the limited arithmetic, large prime numbers are not taken here for decomposition.The results are shown in the Table 6 .In order to visualize the comparison of the three more, the following line graph was plotted.

From the above Fig. 12, it can be seen that the difference in factorization time consumption among the three is insignificant when the modulus is small, but when the modulus is large, the decomposition time consumption of M-RSA is significantly lower than the time consumption of the remaining two algorithms.By selecting a reasonable prime length (such as a 2048-bit modulus generated by 3 primes of 683 bits each), M-RSA can maintain the same level of difficulty for factorization as traditional RSA. The Number Field Sieve algorithm has limited efficiency gains for factoring multiple primes, hence under proper parameter configurations, the effectiveness of key entropy is not significantly reduced.[17]

MRA security analysis comparison

The MRA algorithm is a fusion of the improved S-AES algorithm as well as the M-RSA algorithm, and we have verified their security in the previous two chapters. Here we choose to test the MRA model as well as the traditional RSA with brute force attack. The experimental results are shown in the following Table 7:

From the above Fig. 13 it can be seen that the improved MRA algorithm resists attacks with much higher time consumption than the traditional RSA algorithm, and as the number of bits increases, the more difficult it is to crack the algorithm. Due to computational capabilities, the issue of large prime numbers with hundreds as well as thousands of digits is not discussed here.

Results

Comparison of encryption and decryption execution time of MRA model with conventional AES model by simulation program written in OpenSSL. Experimental environment:

1. CPU: Intel(R) Core(TM) i7-8750H CPU @ 2.20GHz 2.21 GHz
2. RAM:8GB
3. OS:Ubuntu and windows11
4. Platform environment:OpenSSL,OMNET++
5. Data:Custom .json format data

Encryption parameter settings :

1. M-RSA Key length: 1024-bit
2. M-RSA Hash algorithm: OAEP
3. M-RSA Padding method: Hash 256
4. S-AES Key length: 256-bit
5. S-AES Encryption mode: CBC
6. S-AES Padding method: PKCS7

The data is encrypted using MRA algorithm, RSA-1024, AES-RSA[15] respectively, where the data is selected from 30kb file, 60kb file,120kb file etc. of the same size. The following are the experimental results:

From the above Fig. 14, We selected the average of 10 measurements under different data sizes. In the range of data sizes from 30kb to 200kb file size, MRA showed an encryption speed improvement of 25%-35% compared to AES-RSA[15], and a decryption speed improvement ranging from 15%-39%.

In Fig. 15, we present the actual encryption and decryption logs. The data we have used is real agricultural data in .json format. It includes the key generation, execution logs of encryption and decryption, as well as time consumption and other contents.

In Table 8, we compare the computational latency, security level, and the improvement in computational efficiency relative to RSA-1024 of MRA with three other algorithms in the OMNET++ simulation platform. We used OMNET to simulate a real-world agricultural IoT scenario, which includes 2 PCs simulating the network server and application server, as well as 3 IoT devices. When sending a data packet (30B-200B), the computational latency of MRA is in the range of 5ms-15ms, which is significantly lower than the other three algorithms. Moreover, compared to RSA-1024, our computational efficiency has been greatly improved in LoRaWAN. The experimental results show that the scheme effectively handles data transmission during the encryption and decryption process and maintains good scalability under multiple devices.

Finally, we conducted simulation experiments on different IoT platforms using the OMNeT++ platform and NS3 to evaluate the performance of the MRA scheme in practical applications. Figure 16 shows the performance metrics when the IoT device sends a data packet to the PC: the IoT device sends a CT data packet with a transmission time of 0.010180 seconds, hence the reported delay on the PC is 0.010180 s, and the energy consumption is 0.502 mJ. We also conducted tests on the Ubuntu system shown as Fig. 17, which includes throughput metrics ranging from 15 kbps to 100 kbps, within the tolerance range of low-energy devices. At the same time, we compared the energy consumption of different packet sizes with references[10,15,41], as shown in Fig. 18. When sending data packets of 30B to 200B in size, our energy consumption ranges from 0.06 mJ to 0.55 mJ, which is lower than other methods. The results indicate that our scheme's performance in terms of energy consumption, latency, and throughput usage meets expectations and is suitable for practical applications.

Limitations and future work

The MRA algorithm proposed in this paper enhances security on low-energy devices, improves resistance to attacks, and reduces energy consumption and increases computational efficiency compared to some research content. However, there are still some shortcomings. MRA relies on the physical implementation of traditional cryptographic algorithms (RSA/AES), which may leak key operation characteristics (such as power consumption timing), leading to side-channel attack threats (such as differential power analysis); RSA is based on the problem of factoring large integers, and while three-prime RSA increases the difficulty of factorization, it still cannot withstand quantum computing attacks; the computational complexity of RSA key negotiation increases linearly with the number of nodes, and the hybrid encryption mode requires frequent transmission of encryption keys, which exacerbates network load and energy consumption, restricting the deployment of millions of nodes. Therefore, in our subsequent work, we consider using lattice-based cryptography (such as CRYSTALS-Kyber/Dilithium) to replace RSA and build a post-quantum secure framework based on the Shortest Vector Problem (SVP) in lattices; design hierarchical key management (such as identity-based encryption) and aggregate signature mechanisms to reduce communication overhead and support dynamic node access.

Conclusions

With the rapid growth in the number of connected devices in the Internet of Things, ensuring secure and efficient communication has become very important. This paper introduces a new hybrid AES-RSA encryption scheme (MRA) tailored for the Internet of Things environment. By reducing the number of AES iteration rounds and expanding the number of prime numbers in RSA to decrease the key bit length, our method alleviates the processing burden on resource-constrained IoT devices, thereby improving the overall system efficiency. The MRA scheme addresses the key challenge of limited research on encryption schemes for lightweight protocols, while incorporating robust security measures to guard against potential threats. Rigorous theoretical analysis and ProVerif verification confirm the robustness of the scheme in maintaining confidentiality and decryption efficiency. Experimental evaluations show that MRA excels in terms of energy consumption, latency, and scalability, making it a practical solution for real-world IoT applications.

Data availability

Data are contained within the article. Data is provided within the manuscript

Received: 25 January 2025; Accepted: 15 April 2025
Published online: 25 April 2025

References

1. Elijah, O., Rahman, T. A., Orikumhi, I., Leow, C. Y. & Hindia, M. N. An overview of internet of things (iot) and data analytics in agriculture: Benefits and challenges. *IEEE Internet things J.* **5**(5), 3758–3773 (2018).
2. Chiang, M. & Zhang, T. Fog and iot: An overview of research opportunities. *IEEE Internet Things J.* **3**(6), 854–864 (2016).
3. Cui, Z. et al. A hybrid blockchain-based identity authentication scheme for multi-wsn. *IEEE Trans. Serv. Comput.* **13**(2), 241–251 (2020).
4. Wang, C., Wang, D., Tu, Y., Xu, G. & Wang, H. Understanding node capture attacks in user authentication schemes for wireless sensor networks. *IEEE Trans. Depend. Secur. Comput.* **19**(1), 507–523 (2020).
5. Chen, S., Xu, H., Liu, D., Hu, B. & Wang, H. A vision of iot: Applications, challenges, and opportunities with china perspective. *IEEE Internet Things J.* **1**(4), 349–359 (2014).
6. Lee, J.-S., Su, Y.-W. & Shen, C.-C. A comparative study of wireless protocols: Bluetooth, uwb, zigbee, and wi-fi. In *IECON 2007-33rd Annual Conference of the IEEE Industrial Electronics Society* 46–51 (IEEE, 2007).
7. Lin, J. et al. A survey on internet of things: Architecture, enabling technologies, security and privacy, and applications. *IEEE Internet Things J.* **4**(5), 1125–1142 (2017).
8. Majid, M. et al. Applications of wireless sensor networks and internet of things frameworks in the industry revolution 4.0: A systematic literature review. *Sensors* **22**(6), 2087 (2022).
9. Abbasi, A. Z. et al. A review of wireless sensors and networks applications in agriculture. *Comput. Stand. Interfaces* **36**(2), 263–270 (2014).
10. Mishra, C. & Sahu, B. Transmission of encrypted data in wsn: An implementation of hybridized rsa-tdes algorithm (IEEE, 2020)
11. Lu, H., Li, J. & Guizani, M. Secure and efficient data transmission for cluster-based wireless sensor networks. *IEEE Trans. Parallel Distrib. Syst.* **25**(3), 750–761 (2014).
12. Steiner, J. G., Neuman, B. C. & Schiller, J. I. kerberos: An authentication service for open network systems (1988)
13. Zhang, D., Wang, Q.-G., Feng, G., Shi, Y. & Vasilakos, A. V. A survey on attack detection, estimation and control of industrial cyber-physical systems. *ISA Trans.* **116**, 1–16 (2021).
14. Liu, A. & Ning, P. Tinyecc: A configurable library for elliptic curve cryptography in wireless sensor networks. In *International Conference on Information Processing in Sensor Networks* (2008)
15. Ahmed, A., Nanne, M. F. & Gueye, B. The effectiveness of a hybrid diffie-hellman-rsa-aes model. In *2022 International Conference on Computer Communication and Informatics (ICCCI)* 1–5 (IEEE, 2022).
16. Bowen, L. Research on aes-rsa hybrid encryption algorithm and its application in army logistics management system. Master's thesis, Zhejiang University (2022)
17. Boneh, D. Twenty years of attacks on the rsa cryptosystem. Not. Ams **46** (2002)
18. Jintcharadze, E. & Abashidze, M. Performance and comparative analysis of elliptic curve cryptography and rsa. In *2023 IEEE East-West Design & Test Symposium (EWDTS)* 1–4 (IEEE, 2023).
19. Porselvi, T., Sai Ganesh, C. S., Janaki, B., Priyadarshini, K. & Shajitha, B. S. Iot based coal mine safety and health monitoring system using lorawan. In *2021 3rd International Conference on Signal Processing and Communication (ICPSC)* (2021)
20. Bor, M., Vidler, J. & Roedig, U. *Lora for the internet of things* (Junction Publishing, 2016).
21. Noura, H., Hatoum, T., Salman, O., Yaacoub, J.-P. & Chehab, A. Lorawan security survey: Issues, threats and possible mitigation techniques. *Internet Things* **12**, 100303 (2020).
22. Almuhaya, M. A., Jabbar, W. A., Sulaiman, N. & Abdulmalek, S. A survey on lorawan technology: Recent trends, opportunities, simulation tools and future directions. *Electronics* **11**(1), 164 (2022).
23. Da Rocha, A. M., De Oliveira, M. A. & Cavalheiro, G. G. H. Abp vs. otaa activation of lora devices: an experimental study in a rural context. In *2023 International Conference on Computing, Networking and Communications (ICNC)* 630–634 (IEEE, 2023).
24. Kaedi, M., Bohlooli, A. & Pakrooh, R. Simultaneous optimization of cluster head selection and inter-cluster routing in wireless sensor networks using a 2-level genetic algorithm. *Appl. Soft Comput.* **128**, 109444 (2022).
25. Soni, D., Tiwari, V., Kaur, B. & Kumar, M. Cloud computing security analysis based on rc6, aes and rsa algorithms in user-cloud environment. In *2021 First International Conference on Advances in Computing and Future Communication Technologies (ICACFCT)* 269–273 (IEEE, 2021).
26. Zhang, L., Li, B. & Zhao, X. Reconfigurable hardware implementation of aes-rsa hybrid encryption and decryption. In *2020 IEEE 5th International Conference on Signal and Image Processing (ICSIP)* 970–974 (IEEE, 2020).
27. Mahajan, P. & Sachdeva, A. A study of encryption algorithms aes, des and rsa for security. *Glob. J. Comput. Sci. Technol.* **13**(15), 15–22 (2013).
28. Kim, S.-M., Goo, Y.-H., Kim, M.-S., Choi, S.-G. & Choi, M.-J. A method for service identification of ssl/tls encrypted traffic with the relation of session id and server ip. In *2015 17th Asia-Pacific Network Operations and Management Symposium (APNOMS)* 487–490 (IEEE, 2015).
29. Kurera, C. & Navoda, D. Node-to-node secure data transmission protocol for low-power iot devices. In *2018 18th International Conference on Advances in ICT for Emerging Regions (ICTer)* 1–7 (IEEE, 2018).
30. Ylonen, T. Ssh–secure login connections over the internet. In *Proceedings of the 6th USENIX Security Symposium* vol. 37 40–52 (1996).
31. Alsyayid, F. *et al.* An experimental evaluation of the advanced encryption standard algorithm and its impact on wireless sensor energy consumption. In *2020 International Conference on Innovation and Intelligence for Informatics, Computing and Technologies (3ICT)* 1–6 (IEEE, 2020).
32. Blanchet, B., Smyth, B., Cheval, V. & Sylvestre, M. Proverif 2.00: automatic cryptographic protocol verifier, user manual and tutorial. **16**, 05–16 (2018).
33. Canetti, R., Goldreich, O. & Halevi, S. The random oracle methodology, revisited. *J. ACM (JACM)* **51**(4), 557–594 (2004).
34. Shoup, V. Sequences of games: A tool for taming complexity in security proofs. Cryptology eprint archive (2004).
35. Jonsson, J. On the security of ctr+ cbc-mac. In *Selected Areas in Cryptography: 9th Annual International Workshop, SAC 2002 St. John's, Newfoundland, Canada, August 15–16, 2002 Revised Papers 9* 76–93 (Springer, 2003).
36. Daemen, J. & Rijmen, V. The Design of Rijndael - The Advanced Encryption Standard (AES), Second Edition. *Inf. Secur. Cryptogr.* https://doi.org/10.1007/978-3-662-60769-5 (2020).
37. Bogdanov, A., Khovratovich, D. & Rechberger, C. Biclique cryptanalysis of the full AES. In *Advances in Cryptology - ASIACRYPT 2011 - 17th International Conference on the Theory and Application of Cryptology and Information Security, Seoul, South Korea, December 4-8, 2011. Proceedings. Lecture Notes in Computer Science* (eds. Lee, D. H. & Wang, X.) vol. 7073 344–371 (Springer, 2011). https://doi.org/10.1007/978-3-642-25385-0_19 .
38. Courtois, N. T. & Meier, W. Algebraic attacks on stream ciphers with linear feedback. In *Advances in Cryptology - EUROCRYPT 2003, International Conference on the Theory and Applications of Cryptographic Techniques, Warsaw, Poland, May 4-8, 2003, Proceedings. Lecture Notes in Computer Science, vol. 2656* (ed Biham, E.)345–359 (Springer, 2003). https://doi.org/10.1007/3-540-39200-9_21 .
39. Nechvatal, J. et al. Report on the development of the advanced encryption standard (aes). *J. Res. Natl. Inst. Stand. Technol.* **106**(3), 511 (2001).
40. Sun, H.-M., Wu, M.-E., Ting, W.-C. & Hinek, M. J. Dual rsa and its security analysis. *IEEE Trans. Inf. Theory* **53**(8), 2922–2933 (2007).

41. Cui, X., Tian, Y., Zhang, X., Lin, H. & Li, M. A lightweight certificateless edge-assisted encryption for iot devices: Enhancing security and performance. *IEEE Internet Things J.* **12**, 2930–2942 (2024).
42. Ahmad, S. & Mehfuz, S. Efficient time-oriented latency-based secure data encryption for cloud storage. *Cyber Secur. Appl.* **2**, 100027 (2024).

Author contributions

Conceptualization, Q.C. and W.Y.; methodology, Q.C. and T.M.; validation, Q.C.; formal analysis, W.Y.; investigation, Q.C. and W.Y.; resources, Q.C. and T.M.; writing—original draft preparation, Q.C.; writing—review and editing, Q.C.

Funding

This research was funded by the National Key Research and Development Program of China (Grant No. 2022ZD0115802), the Key Research and Development Program of the Autonomous Region (Grant No. 2022B01008), the Tianshan Science and Technology Innovation Leading talent Project of the Autonomous Region (Grant No. 2022TSYCLJ0037), the National Natural Science Foundation of China (Grant No. 62262065)

Declarations

Competing interests

The authors declare no competing interests.

Additional information

Correspondence and requests for materials should be addressed to T.M. or W.Y.