Designing a secure and scalable service model using blockchain and MQTT for IoT devices

Tse-Chuan Hsu
Department of Computer Science & Information Management, Soochow University
Taipei, Taiwan

Han-Sheng Lu
Department of Computer Science & Information Management, Soochow University
Taipei, Taiwan

*Corresponding author: Email: tchsu@gm.scu.edu.tw

Abstract

In the realm of Internet of Things (IoT) communication, where many devices operate within resource-constrained environments, the MQTT communication protocol is often employed to establish a swift and efficient network for sharing and exchanging data.It primarily supports message broadcasting rather than point-to-point data exchange. Each endpoint can merely broadcast messages to other endpoints subscribed to the same topic. Furthermore, MQTT lacks built-in encryption mechanisms, leaving data transmission vulnerable to potential eavesdropping.

In response to these shortcomings, this research leverages blockchain technology and enhances it with features such as public and private key management, broadcasting, and message verification. The objective is to enhance communication quality and ensure the reliability of message encryption. To achieve this, every device within the network is equipped with the public keys of other devices through a broker broadcast. Before encrypting a message using these public keys, a verification step is performed to ensure the consistency of public keys across all devices. This approach facilitates Message on Transmission Protocol (MTP),Subject-Specific Communication Protocol (SSCP) and mitigates the risk of compromised public keys.

In the experimentation, it is demonstrated that the experimental performance of this architecture, whether with 3 devices, 10 devices, or 100 devices, exhibits a latency almost difference of less than 1 second. Therefore, this validates that our designed architecture not only enhances security but also boasts excellent performance.

1. Introduction

The absence of intrinsic data encryption capabilities in MQTT poses an acute problem, significantly compromising the integrity of sensitive information within IoT applications. In light of these challenges, blockchain technology emerges as a compelling solution. Blockchain, a decentralized and distributed architectural paradigm, fortifies the sanctity of data records through an immutable ledger maintained via consensus mechanisms among network nodes. Additionally, it endows data exchanges with an impervious shield of security through the utilization of public-private key pairs.
In this study, multiple enhancements to improve the MQTT security posture become apparent by integrating blockchain technology into the MQTT node communication model. This enhancement includes enhancements against data tampering and corrects potential risks of MQTT, including for Communication Protocol; within the conventional MQTT framework, nodes must resort to the topic subscription mechanism when they seek to establish Single Transmission Protocol. Although this approach permits multiple recipients, it fails to guarantee data security in scenarios where only one of the recipients is authentic. In the architecture we propose for our research, we introduce a novel approach that empowers each node to attain original Single Transmission Protocol. This communication is accomplished by utilizing RSA and AES encryption mechanisms, ensuring robust data security and confidentiality. In our proposed architectural framework, each topic is endowed with a unique RSA public-private key pair. This design feature serves the crucial function of restricting access to the content of messages to only those nodes that have subscribed to the respective topic.

2. BACKGROUND KNOWLEDGE AND RELATED WORK

MQTT with Internet of Things (IoT)

The concept of the Internet of Things (IoT) was first introduced by Kevin Ashton in 1999 [1]. Its goal is to enable intelligent monitoring, data collection, and automated control of various objects in the real world, connecting physical devices in a network, including sensors, software, and data exchange mechanisms [2]. The critical feature of IoT is its applicability across various domains, significantly improving efficiency and reducing costs, whether in smart homes, industrial production, urban infrastructure, agricultural monitoring, etc. It facilitates intelligent communication between different devices, devices with people, and devices with the environment [3]. MQTT is a communication protocol used for transmitting information between IoT devices, first proposed by IBM in 1999 and widely adopted. The MQTT protocol is based on a publish/subscribe model, with a central server called a broker responsible for coordinating message transmission and reception. Devices can receive the information they need by subscribing to specific topics and transmitting data by publishing messages to corresponding topics [4]. In IoT, MQTT provides a reliable, scalable, and resource-efficient means of data exchange. However, it lacks built-in encryption mechanisms, which can pose a risk of data leakage in unsecured MQTT communication [5]. Therefore, ensuring the security and encryption of MQTT communication is essential in IoT applications.

Decentralized Architecture in Blockchain

Blockchain is a technology emphasizing decentralized architecture and immutable data storage and exchange. It divides data into blocks containing a certain amount of data and a digital signature to ensure block integrity. These blocks are arranged chronologically, with their content cryptographically linked, forming a chain. Consensus mechanisms provide data that cannot be tampered with or falsified [6]. Blockchain technology is not limited to cryptocurrency, such as Bitcoin, but is widely used in various areas, including supply chain management, healthcare, finance, real estate, etc. [7]. Its primary advantage is providing a secure and reliable way to record and share data while eliminating the need to trust intermediaries and the risk of data tampering, making blockchain an essential tool for increasing transparency and trust in many industries [8].

Symmetric Encryption Algorithms

Symmetric encryption, also known as shared-key encryption, is a cryptographic technique that uses the same key for encryption and decryption [9]. This key can be a number, a string, or a particular data file if it is sufficiently long and complex to ensure data security. The key must be kept secret during the encryption and decryption processes, as anyone with access to the key can easily decrypt the data [10]. Symmetric encryption transforms data using the key, making it difficult to understand.

The encryption process takes plaintext (unencrypted data) and the key as inputs, using a series of mathematical operations to generate ciphertext (encrypted data). The decryption process reverses these operations to recover the original plaintext data. Symmetric encryption offers several advantages, including speed and efficiency, suitability for various applications like data transmission, file encryption, network security, data storage, and simplicity of implementation. [11] However, it also presents challenges, such as securing key management and exchange when sharing encrypted data.

Asymmetric Encryption Schemes

Asymmetric uses a pair of keys: one is the public key, and the other is the private key. We used the public key to encrypt data and be openly shared, while the private key must be kept confidential, allowing only the holder to decrypt the data. This method is more secure because third parties cannot decrypt the data unless they illegally obtain the private key [12]. The core principle of asymmetric encryption involves using mathematically complex algorithms to generate this essential pair. RSA is the most well-known asymmetric encryption algorithm, named after its inventor [13]. Senders use the recipient's public key to encrypt messages; only the recipient's private key can decrypt and read the statements. To ensures the confidentiality of data during transmission, as third parties cannot decipher the data unless they obtain the private key [14].

3. RESEARCH FRAMEWORK

Given the resource constraints of IoT devices, in this study's scope, we posit that integrating blockchain's public-private key authentication and broadcasting methods with the MQTT protocol is currently the most practical solution. The research also demonstrates that performance remains unaffected when many devices are concurrently operational. Simultaneously, this integration enhances the security of MQTT transmissions. Figure 1 presents a scenario involving a substantial quantity of real-world machines. In the ensuing discourse and visual representations, our primary emphasis will center on four specific devices, thus enhancing the clarity of our explanation.

Our proposed architecture primarily comprises the Single Transmission Protocol and the Topic Transmission Protocol. A common feature between both is that when a message needs to be transmitted, it initiates the recipient's public key broadcast verification first, as depicted in Figure 2. This verification guarantees that the recipient's public keys

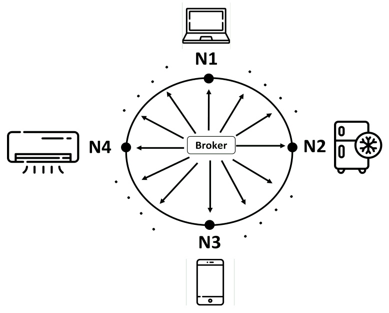


Figure 1. The anticipated real-world operational scenario involves a large number of devices and centralizes around the integration of a Broker with blockchain technology.

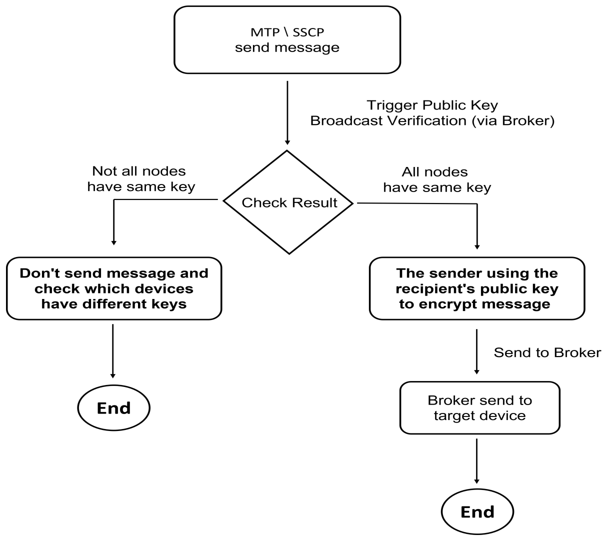
for all nodes in the architecture are uniform, ensuring the accuracy of the recipient's public keys. Only when this condition is satisfied the message is encrypted and transmitted. Otherwise, no transmission occurs, and non-matching nodes' public keys are logged.

The framework of trigger public key broadcast verification is shown in Figure 2 since asymmetric encryption is unsuitable for encrypting large amounts of data. In the data transmission, after checking that all devices have the same key, the encryption algorithm used is symmetric encryption with AES. Therefore, after encryption, the sender must transmit the private key to the recipient for successful decryption. In our architecture, a private key is obtained after AES encryption, which is then encrypted with the recipient's RSA public key before transmission. Based on the structure, it ensures the security of the private key transmission process, and it ensures that only the legitimate recipient can receive the symmetrically encrypted private key and further decrypt it into plaintext. Figure 3 shows describes explicitly the encryption process.

Consequently, we have established the following experimental framework that describes the testing environment and architecture, which consists of four devices from N1 to N4 with a broker. As the devices are presented using different devices, the numbers 1111 to 1114 represent devices N1 to N4, respectively.

3.1. Message on Transmission Protocol (MTP)

When each device communicates using the MQTT protocol, it generates its public-private key pairs. These public keys are sent to the broker, while the private keys are kept on the respective devices. In this structure, the broker has the public keys of all devices. Once the broker receives these public keys, it broadcasts them to all other devices. This means that each device possesses the public keys of different devices, enabling mutual communication; the devices will use MQTT transmission with a broker like Figure 4. In our study, we used the number of public-private key pairs



MTP \ SSCP send message
Trigger Public Key Broadcast Verification (via Broker)
Check Result
Not all nodes have same key
All nodes have same key
Don't send message and check which devices have different keys
The sender using the recipient's public key to encrypt message
Send to Broker
Broker send to target device
End
End

Figure 2. The commonality between the two protocols lies in the activation of a public key broadcast verification mechanism during transmission, verifying if all recipients share the same public key, thereby determining whether to proceed with the transmission of encrypted messages.

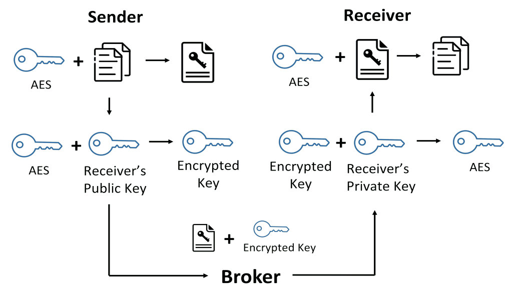


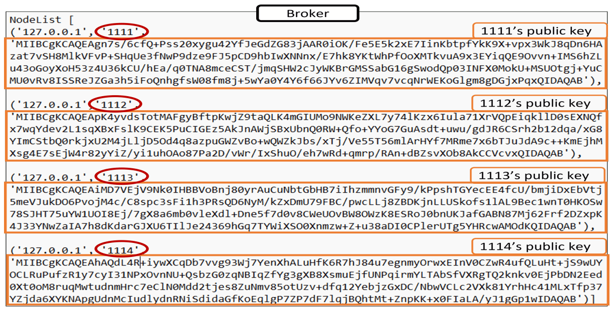
Figure 3. The data encryption and transmission process

owned by each device, which can be summarized in Table 1. Figure 4 shows that the broker gets regeneration. Each node transmits a public key to the broker and retrieves all distinct device public keys. Figure 5 illustrates Node1 as an example; the content for other nodes remains the same.

When a device intends to use a single transmission protocol, the first step is the sender broadcasts the recipient's public key through the Broker. It waits for all other devices to verify whether their recipient's public key is matches; if any device returns a different result, it indicates a security concern, including potential tampering or incorrect content in the public key. The system will consequently cease message transmission. If all devices return matching results, the sender encrypts the message using the recipient's public key to ensure that only the legitimate recipient can decrypt it correctly. Figure 6 describes the scenario in which all devices in Figure 2 send back identical data and further en-

	Possession of Device Public Key	Possession of Device Private Key
Broker	N1、N2、N3、N4	-
N1	N1、N2、N3、N4	N1
N2	N1、N2、N3、N4	N2
N3	N1、N2、N3、N4	N3
N4	N1、N2、N3、N4	N4

Table 1. Each device having the public-private key pairs of other devices.



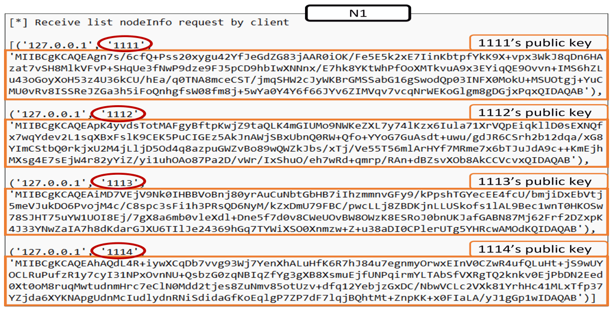
```
NodeList [
('127.0.0.1','1111',
'MIIBCgKCAQEAgn7s/6cfQ+Pss20xygu42YfJeGdZG83jAAR0iOK/Fe5E5k2xE7IinKbtpfYkK9X+vpx3WkJ8qDn6HA
zat7vSH8MlkVFvP+SHqUe3fNwP9dze9FJ5pCD9hbIwXNNnx/E7hk8YKtWhPfOoXMTkvuA9x3EYiqQE9Ovvn+IMS6hZL
u43oGoyXoH53z4U36kCU/hEa/q0TNA8mceCST/jmqSHW2cJyWKBrGMSSabG16gSwodQp03INFX0MokU+MSUOtgj+YuC
MU0vRv8ISSReJZGa3h5iFoQnhgfsW08fm8j+5wYa0Y4Y6f66JYv6ZIMVqv7vcqNrWEKoGlgm8gDGjxPqxQIDAQAB'),

('127.0.0.1','1112',
'MIIBCgKCAQEApK4yvdsTotMAFgyBftpKwjZ9taQLK4mGIUMo9NWKeZXL7y74lKzx6Iula71XrVQpEiqllD0sEXNQf
x7wqYdev2L1sqXBxFslK9CEK5PuCIGEz5AkJnAWjSBxUbnQ0RW+Qfo+YYoG7GuAsdt+uwu/gdJR6CSrh2b12dqa/xG8
YImCStbQ0rkjxU2M4jLljD5Od4q8azpuGWZvBo+wQWZkJbs/xTj/Ve55T56mlArHYf7MRme7x6bTJuJdA9c++KmEjhM
Xsg4E7sEjW4r82yYiZ/yi1uhOAo87Pa2D/vWr/IxShuO/eh7wRd+qmrp/RAn+dBZsvXOb8AkCCVcvxQIDAQAB'),

('127.0.0.1','1113',
'MIIBCgKCAQEAiMD7VEjV9Nk0IHBBVoBnj80yrAuCuNbtGbHB7iIhzmmnvGFy9/kPpshTGYecEE4fcU/bmjiDxEbVtj
5meVJukDO6PvojM4c/C8spc3sFi1h3PRsQD6NyM/kZxDmU79FBC/pwcLLj8ZBDKjnLLUSkofs1lAL9Bec1wnT0HKOSw
78SJHT75uYW1UOI8Ej/7gX8a6mb0vleXdl+Dne5f7d0v8CWeUOvBW8OWzK8ESRoJ0bnUKJafGABN87Mj62Frf2DZxpK
4J33YNwZaIA7h8dKdarGJXU6TIlJe24369hGq7TYWiXSO0Xnmzw+Z+u38aDI0CPlerUTg5YHRcwAMOdKQIDAQAB'),

('127.0.0.1','1114'
'MIIBCgKCAQEAhAQdL4R+iywXCqDb7vvg93Wj7YenXhALuHfK6R7hJ84u7egnmyOrwxEInV0CZwR4ufQLuHt+jS9wUY
OCLRuPufzR1y7cyI31NPxOvnNU+QsbzG0zqNBIqZfYg3gXB8XsmuEjfUNPqirmYLTAbSfVXRgTQ2knkv0EjPbDN2Eed
0Xt0oM8ruqMwtudnmHrc7eClN0Mdd2tjes8ZuNmv85otUzv+dfq12YebjzGxDC/NbwVCLc2VXk81YrhHc41MLxTfp37
YZjda6XYKNApgUdnMcIudlydnRNiSdidaGfKoEqlgP7ZP7dF7lqjBQhtMt+ZnpKK+x0FIaLA/yJ1gGp1wIDAQAB')]
```

Figure 4. The diagram shows that the Broker have the public keys of all other devices.

crypt it for transmission. The design of N1 wanting to send a message to N3 is depicted. N1 initiates a broadcast for public key verification through dashed arrow flows.

In the context of a device utilizing a singular transmission protocol, the first step involves the sender distributing the recipient's public key via the intermediary known as the broker. Subsequently, all other devices await the verification of this public key. If any device yields a dissimilar outcome during the verification process, this anomaly indicates potential security issues, including the possibility of tampering or inaccuracies in the public key's content. Consequently, the message transmission system will be halted. If all devices provide matching verification results, the sender encrypts the message using the recipient's public key, ensuring that only the designated recipient can decrypt the message accurately. Figure 6 elucidates the scenario wherein, when different devices transfer their message, each node has decrypted or can't decrypt by other message, the data during the verification process and subsequently engage in further encryption for transmission. The diagram illustrates the design of N1's endeavor to transmit a message to N3. The dashed arrow flows indicate that N1 initiates the broadcast for public key validation.

Each device, including N3, in this structure responds with" Check" after successful verification. If N1 receives" Check" replies from all devices, it sends the encrypted message along the solid arrow. If any device returns different data, the content marked with the solid black lines is not executed.



```
[*] Receive list nodeInfo request by client

[('127.0.0.1','1111',
'MIIBCgKCAQEAgn7s/6cfQ+Pss20xygu42YfJeGdZG83jAAR0iOK/Fe5E5k2xE7IinKbtpfYkK9X+vpx3WkJ8qDn6HA
zat7vSH8MlkVFvP+SHqUe3fNwP9dze9FJ5pCD9hbIwXNNnx/E7hk8YKtWhPfOoXMTkvuA9x3EYiqQE9Ovvn+IMS6hZL
u43oGoyXoH53z4U36kCU/hEa/q0TNA8mceCST/jmqSHW2cJyWKBrGMSSabG16gSwodQp03INFX0MokU+MSUOtgj+YuC
MU0vRv8ISSReJZGa3h5iFoQnhgfsW08fm8j+5wYa0Y4Y6f66JYv6ZIMVqv7vcqNrWEKoGlgm8gDGjxPqxQIDAQAB'),

('127.0.0.1','1112',
'MIIBCgKCAQEApK4yvdsTotMAFgyBftpKwjZ9taQLK4mGIUMo9NWKeZXL7y74lKzx6Iula71XrVQpEiqllD0sEXNQf
x7wqYdev2L1sqXBxFslK9CEK5PuCIGEz5AkJnAWjSBxUbnQ0RW+Qfo+YYoG7GuAsdt+uwu/gdJR6CSrh2b12dqa/xG8
YImCStbQ0rkjxU2M4jLljD5Od4q8azpuGWZvBo89wQWZkJbs/xTj/Ve55T56mlArHYf7MRme7x6bTJuJdA9c++KmEjh
MXsg4E7sEjW4r82yYiZ/yi1uhOAo87Pa2D/vWr/IxShuO/eh7wRd+qmrp/RAn+dBZsvXOb8AkCCVcvxQIDAQAB'),

('127.0.0.1','1113',
'MIIBCgKCAQEAiMD7VEjV9Nk0IHBBVoBnj80yrAuCuNbtGbHB7iIhzmmnvGFy9/kPpshTGYecEE4fcU/bmjiDxEbVtj
5meVJukDO6PvojM4c/C8spc3sFi1h3PRsQD6NyM/kZxDmU79FBC/pwcLLj8ZBDKjnLLUSkofs1lAL9Bec1wnT0HKOSw
78SJHT75uYW1UOI8Ej/7gX8a6mb0vleXdl+Dne5f7d0v8CWeUOvBW8OWzK8ESRoJ0bnUKJafGABN87Mj62Frf2DZxpK
4J33YNwZaIA7h8dKdarGJXU6TIlJe24369hGq7TYWiXSO0Xnmzw+Z+u38aDI0CPlerUTg5YHRcwAMOdKQIDAQAB'),

('127.0.0.1','1114',
'MIIBCgKCAQEAhAQdL4R+iywXCqDb7vvg93Wj7YenXhALuHfK6R7hJ84u7egnmyOrwxEInV0CZwR4ufQLuHt+jS9wUY
OCLRuPufzR1y7cyI31NPxOvnNU+QsbzG0zqNBIqZfYg3gXB8XsmuEjfUNPqirmYLTAbSfVXRgTQ2knkv0EjPbDN2Eed
0Xt0oM8ruqMwtudnmHrc7eClN0Mdd2tjes8ZuNmv85otUzv+dfq12YebjzGxDC/NbwVCLc2VXk81YrhHc41MLxTfp37
YZjda6XYKNApgUdnMcIudlydnRNiSdidaGfKoEqlgP7ZP7dF7lqjBQhtMt+ZnpKK+x0FIaLA/yJ1gGp1wIDAQAB')]
```

Figure 5. The diagram shows that the Node 1 have the public keys of all other devices.

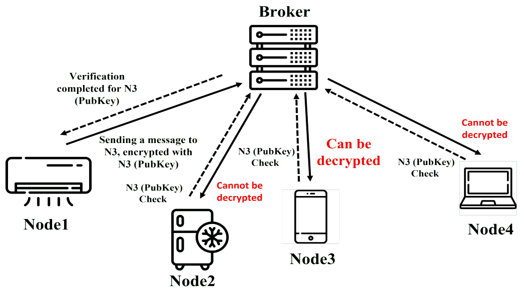


Figure 6. Regarding the Message on Transmission Protocol Flowchart, dashed lines represent the transmission of key verification results, while solid lines indicate the transmission of the judgment results. If there are no issues, the message is encrypted and sent through the Broker.

3.2. Subject-Specific Communication Protocol (SSCP)

In the MQTT transmission protocol, all devices communicate by publishing and subscribing to topics. When a device intends to publish content to a topic, it first registers the topic with the broker. The broker then generates an RSA public-private key pair for the topic.

The broker broadcasts the public key to all devices to ensure that only authorized devices can decrypt messages published to a topic. Each machine then uses its private key to encrypt messages before broadcasting them to the topic. All other devices can decrypt the messages using the public key.
Figure 7 illustrates a scenario in which N1 and N2 are subscribed to Topic A, and N3 and N4 are subscribed to Topic B. N1 publishes a message to Topic A, and the broker forwards the encrypted message to N2. N3 and N4 cannot decrypt the message because they do not have the private key for Topic A.

In the context of N1's intent to transmit a message to Topic B, the initial step is to execute the actions denoted by the dashed arrows. This involves N1 broadcasting its public key for Topic B to all devices. Subsequently, N2, N3,

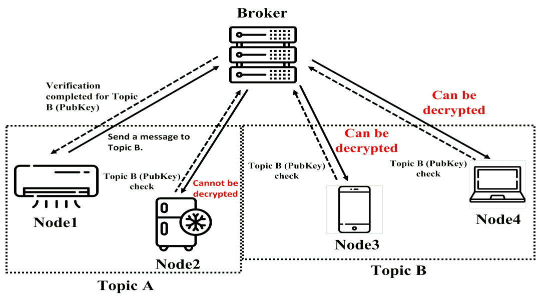
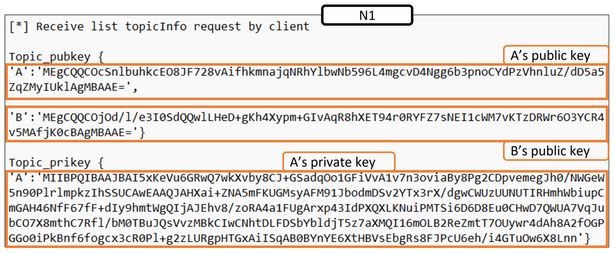


Figure 7. Subject-Specific Communication Protocol Flowchart, encompassing two main topics, A and B. Simulating a scenario where Node1 intends to transmit to Topic B, the protocol initiates public key verification represented by dashed lines, followed by the actual message transmission indicated by solid lines.

	Owned Topic Public Key	Owned Topic Private Key
Broker	A、B	A、B
N1	A、B	A
N2	A、B	A
N3	A、B	B
N4	A、B	B

Table 2. Each device having the public-private key pairs of other devices

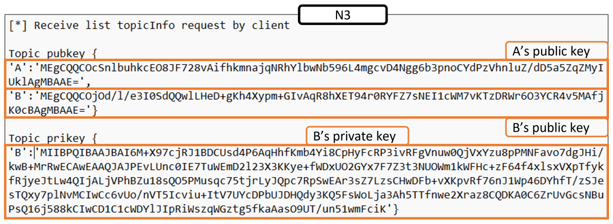


```
[*] Receive list topicInfo request by client

Topic_pubkey {
'A':'MEgCQQCOcSnlbuhkcEO8JF728vAifhkmnajqNRhYlbwNb596L4mgcvD4Ngg6b3pnoCYdPzVhnluZ/dD5a5
ZqZMyIUklAgMBAAE=',

'B':'MEgCQQCOjOd/l/e3I0SdQQwlLHeD+gKh4Xypm+GIvAqR8hXET94r0RYFZ7sNEI1cWM7vKTzDRWr6O3YCR4
v5MAfjK0cBAgMBAAE='}

Topic_prikey {
'A':'MIIBPQIBAAJBAI5xKeVu6GRwQ7wkXvby8CJ+GSadqOo1GFiVvA1v7n3oviaBy8Pg2CDpvemegJh0/NWGeW
5n90PlrlmpkzIhSSUCAwEAAQJAHXai+ZNA5mFKUGMsyAFM91JbodmDSv2YTx3rX/dgwCWUzUUNUTIRHmhWbiupC
mGAH46NfF67fF+dIy9hmtWgQIjAJEhv8/zoRA4a1FUgArxp43IdPXQXLKNuiPMTSi6D6D8Eu0CHwD7QWUA7VqJu
bCO7X8mthC7Rfl/bM0TBuJQsVvzMBkCIwCNhtDLFDSbYbldjT5z7aXMQI16mOLB2ReZmtT7OUywr4dAh8A2fOGP
GGo0iPkBnf6fogcx3cR0Pl+g2zLURgpHTGxAiISqAB0BYnYE6XtHBVsEbgRs8FJPcU6eh/i4GTuOw6X8Lnn'}
```

Figure 8. In the experiment, devices N1 have topic-specific public and private keys for communication.

and N4 respond by verifying the public key and providing their own verification results to N1. If all verification results match, N1 proceeds to encrypt the message using Topic B's public key and transmit it to the Broker. This process ensures that only devices subscribed to Topic B, which possess the corresponding private key, can successfully decrypt the transmitted message.

The objective of our research is to facilitate diverse devices in managing their public and private keys in a scalable and secure manner. As depicted in Figures 8 to 9, the content of N1 and N2 is identical in Figure 8, while the content of N3 and N4 is the same in Figure 9. To avoid occupying excessive space, only N1 and N3 are presented here. Each device holds the public keys for all topics to which it is subscribed, along with the private key for each topic to which it publishes messages. The Broker maintains a record of all public-private key pairs and device subscriptions.



```
[*] Receive list topicInfo request by client

Topic_pubkey {
'A':'MEgCQQCOcSnlbuhkcEO8JF728vAifhkmnajqNRhYlbwNb596L4mgcvD4Ngg6b3pnoCYdPzVhnluZ/dD5a5ZqZMyI
UklAgMBAAE=',
'B':'MEgCQQCOjOd/l/e3I0SdQQwlLHeD+gKh4Xypm+GIvAqR8hXET94r0RYFZ7sNEI1cWM7vKTzDRWr6O3YCR4v5MAfj
K0cBAgMBAAE='}

Topic_prikey {
'B':'MIIBPQIBAAJBAI6M+X97cjRJ1BDCUsd4P6AqHhfKmb4Yi8CpHyFcRP3ivRFgVnuw0QjVxYzu8pPMNFavo7dgJHi/
kwB+MrRwECAwEAAQJAJPEvLUnc0IE7TuWEmD2l23X3KKye+fWDxUO2GYx7F7Z3t3NUOWm1kWFHc+zF64f4xlsxVXpTfyk
fRjyeJtLw4QIjALjVPhBZu18sQO5PMusqc75tjrLyJQpc7RpSwEAr3sZ7LzsCHwDFb+vXKpvRf76nJ1Wp46DYhfT/zSJe
sTQxy7plNvMCIwCc6vUo/nVT5Icviu+ItV7UYcDPbUJDHQdy3KQ5FsWoLja3Ah5TTfnwe2Xraz8CQDKA0C6ZrUvGcsNBu
PsQ16j588kCIwCD1C1cWDYlJIpRiWszqWGztg5fkaAasO9UT/un51wmFciK'}
```

Figure 9. In the experiment, devices N3 have topic-specific public and private keys for communication.

4. Empirical Verification

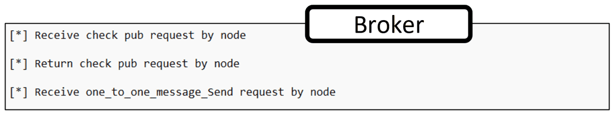
This paper presents two protocols for secure message transmission: the Message on Transmission Protocol (MTP) and the Subject-Specific Communication Protocol (SSCP). Both protocols require all nodes in the architecture to have identical public keys for each recipient. This ensures that only the intended recipient can decrypt the message.

Before transmitting a message, the sender first broadcasts the recipient's public key to all nodes. If any node detects a mismatch in the public key, the message transmission is aborted, and the non-matching public keys are logged. Otherwise, the sender encrypts the message using the recipient's public key and transmits it to the Broker. The Broker then forwards the encrypted message to the recipient.

4.1. Empirical Examination of the Message on Transmission Protocol

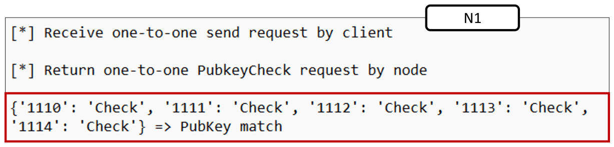
In this scenario, we ensure that the Broker receives the public keys of all nodes in Figures 10 and 11. If all responses are "Check," then the public key is valid. When N1 wants to send a message to N3, it first searches for N3's public key on its device and sends it to the Broker for broadcast verification. All other nodes immediately receive the Broker's broadcast verification request and compare it with their own N3 public keys. If the two match, the node responds with "Check"; otherwise, it responds with "Error" N1 receives and compiles these responses.

As shown in Figure 13, Node 3 can successfully decrypt the message, but Nodes 2 and 4 cannot because they are not in the same transmission potocol (Figures 12 and 14). The content inside the boxes in the message represents ciphertext that cannot be decrypted.Figure 15 shows an experiment in which N1's device changed the content of N3's (1113) public key to "None." When the Broker receives this altered public key, it conducts broadcast verification. All other nodes respond with "Error," but N1 itself is also broadcasted, so it receives a "Check" response. Therefore, if even one device responds with "Error," the message transmission will not be executed.



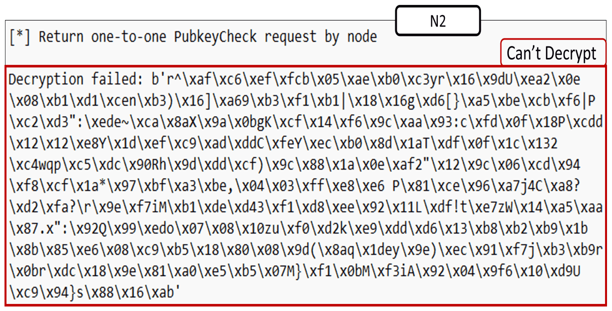
```
[*] Receive check pub request by node

[*] Return check pub request by node

[*] Receive one_to_one_message_Send request by node
```

Figure 10. The experimental interface for Message on Transmission Protocol of broker



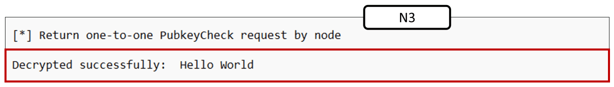
```
[*] Receive one-to-one send request by client

[*] Return one-to-one PubkeyCheck request by node
{'1110': 'Check', '1111': 'Check', '1112': 'Check', '1113': 'Check',
'1114': 'Check'} => PubKey match
```

Figure 11. The experimental interface for Message on Transmission Protocol of Node 1



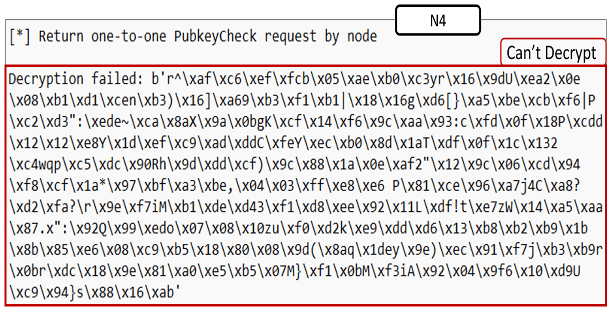
```
[*] Return one-to-one PubkeyCheck request by node
Decryption failed: b'r^\xaf\xc6\xef\xfcb\x05\xae\xb0\xc3yr\x16\x9dU\xea2\x0e
\x08\xb1\xd1\xcen\xb3)\x16]\xa69\xb3\xf1\xb1|\x18\x16g\xd6[}\xa5\xbe\xcb\xf6|P
\xc2\xd3":\xede~\xca\x8aX\x9a\x0bgK\xcf\x14\xf6\x9c\xaa\x93:c\xfd\x0f\x18P\xcdd
\x12\x12\xe8Y\x1d\xef\xc9\xad\xddC\xfeY\xec\xb0\x8d\x1aT\xdf\x0f\x1c\x132
\xc4wqp\xc5\xdc\x90Rh\x9d\xdd\xcf)\x9c\x88\x1a\x0e\xaf2"\x12\x9c\x06\xcd\x94
\xf8\xcf\x1a*\x97\xbf\xa3\xbe,\x04\x03\xff\xe8\xe6 P\x81\xce\x96\xa7j4C\xa8?
\xd2\xfa?\r\x9e\xf7iM\xb1\xde\xd43\xf1\xd8\xee\x92\x11L\xdf!t\xe7zW\x14\xa5\xaa
\x87.x":\x92Q\x99\xedo\x07\x08\x10zu\xf0\xd2k\xe9\xdd\xd6\x13\xb8\xb2\xb9\x1b
\x8b\x85\xe6\x08\xc9\xb5\x18\x80\x08\x9d(\x8aq\x1dey\x9e)\xec\x91\xf7j\xb3\xb9r
\x0br\xdc\x18\x9e\x81\xa0\xe5\xb5\x07M}\xf1\x0bM\xf3iA\x92\x04\x9f6\x10\xd9U
\xc9\x94}s\x88\x16\xab'
```

Figure 12. The experimental interface for Message on Transmission Protocol of Node 2



```
[*] Return one-to-one PubkeyCheck request by node
Decrypted successfully:  Hello World
```

Figure 13. The experimental interface for Message on Transmission Protocol of Node 3

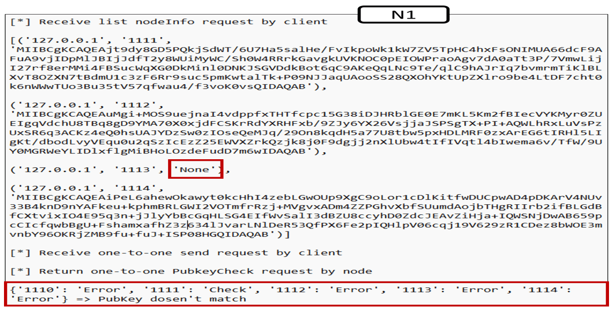


```
[*] Return one-to-one PubkeyCheck request by node
Decryption failed: b'r^\xaf\xc6\xef\xfcb\x05\xae\xb0\xc3yr\x16\x9dU\xea2\x0e
\x08\xb1\xd1\xcen\xb3)\x16]\xa69\xb3\xf1\xb1|\x18\x16g\xd6[}\xa5\xbe\xcb\xf6|P
\xc2\xd3":\xede~\xca\x8aX\x9a\x0bgK\xcf\x14\xf6\x9c\xaa\x93:c\xfd\x0f\x18P\xcdd
\x12\x12\xe8Y\x1d\xef\xc9\xad\xddC\xfeY\xec\xb0\x8d\x1aT\xdf\x0f\x1c\x132
\xc4wqp\xc5\xdc\x90Rh\x9d\xdd\xcf)\x9c\x88\x1a\x0e\xaf2"\x12\x9c\x06\xcd\x94
\xf8\xcf\x1a*\x97\xbf\xa3\xbe,\x04\x03\xff\xe8\xe6 P\x81\xce\x96\xa7j4C\xa8?
\xd2\xfa?\r\x9e\xf7iM\xb1\xde\xd43\xf1\xd8\xee\x92\x11L\xdf!t\xe7zW\x14\xa5\xaa
\x87.x":\x92Q\x99\xedo\x07\x08\x10zu\xf0\xd2k\xe9\xdd\xd6\x13\xb8\xb2\xb9\x1b
\x8b\x85\xe6\x08\xc9\xb5\x18\x80\x08\x9d(\x8aq\x1dey\x9e)\xec\x91\xf7j\xb3\xb9r
\x0br\xdc\x18\x9e\x81\xa0\xe5\xb5\x07M}\xf1\x0bM\xf3iA\x92\x04\x9f6\x10\xd9U
\xc9\x94}s\x88\x16\xab'
```

Figure 14. The experimental interface for Message on Transmission Protocol of Node 4

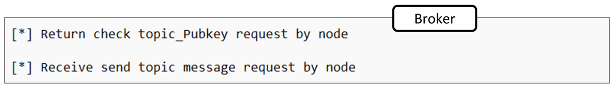
4.2. Empirical Evaluation of the Subject-Specific Communication Protocol

Concerning subject-specific communication protocol grouping, as depicted in Figures 16 to 17, the Broker collects the public keys from various devices. As demonstrated in these figures, when N1 endeavors to dispatch a message to Topic B, the initial phase entails the dissemination of Topic B’s public key for validation. Subsequently, message encryption and transmission are ensured if all nodes respond with an affirmative ”Check” as exemplified in Figure 17.



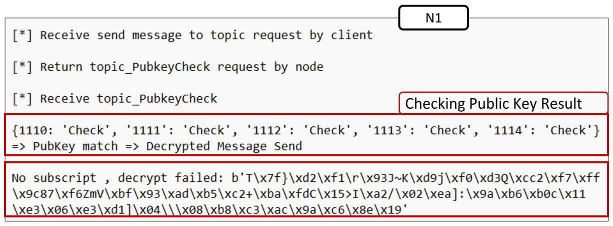
```
[*] Receive list nodeInfo request by client

[('127.0.0.1', '1111',
'MIIBCgKCAQEAjt9dy8GD5PQkjSdWT/6U7Ha5salHe/FvIkpoWk1kW7ZV5TpHC4hxFsONIMUA66dcF9A
FuA9vjIDpMlJBIjJdfT2y8WUiMyWC/Sh0W4RRrkGavgkUVKNOC0pEIOWPraoAgv7dA0aTt3P/7VmwLij
I27rf8erMMi4FBSucWqXG0DkMinl0DNKJSGVDdkBot6qC9AKeQqLNc9Te/qlC9hAJrIq7bvmrmTiKlBL
XvT8OZXN7tBdmU1c3zF6Rr9suc5pmKwtalTk+P09NJJaqUAooSS28QXOhYKtUpZXlro9be4LtDF7cht0
k6nWWwTUo3Bu35tV57qfwau4/f3voK0vsQIDAQAB'),

('127.0.0.1', '1112',
'MIIBCgKCAQEAuMgi+MOS9uejnaI4vdppfxTHTfcpc15G38iDJHRblGE0E7mKL5Km2fBIecVYKMyr0ZU
EIgqVdchU8TBq8gD9YMA70X0xjdFCSKrRdYXRHFxb/9ZJy6YX26VsjjaJSPSgTX+PI+AQWLhRxLuVsPz
UxSR6q3ACKz4eQ0hsUAJYDzSw0zIOseQeMJq/29On8kqdH5a77U8tbw5pxHDLMRF0zxArEG6tIRHl5LI
gKt/dbodLvyVEqu0u2qSzIcEzZ25EWVXZrkQzjk8j0F9dgjj2nXlUbw4tIfIVqtl4bIwema6v/TfW/9U
Y0MGRWeYLIDlxflgMiBHoLOzdeFudD7m6wIDAQAB'),

('127.0.0.1', '1113', 'None'),

('127.0.0.1', '1114',
'MIIBCgKCAQEAiPeL6ahewOkawyt0kcHhI4zebLGwOUp9XgC9oLor1cDlKitfwDUCpwAD4pDKArV4NUv
33B4knD9nYAFkeu+kphmBRLGWI2VOTmfrRzj+MVgvxADm4ZZPGhvXbfSUumdAojbTHgRIIrb2ifBLGdB
fCXtvixIO4E95q3n+jJlyYbBcGqHLSG4EIfWvSalI3dBZU8ccyhD0ZdcJEAvZiHja+IQWSNjDwAB659p
cCIcfqwbBgU+FshamxafhZ3z634lJvarLNlDeR53QfPX6Fe2pIQHlpV06cqj19V629zR1CDez8bWOE3m
vnbY96OKRjZMB9fu+fuJ+ISP08HGQIDAQAB')]

[*] Receive one-to-one send request by client

[*] Return one-to-one PubkeyCheck request by node
{'1110': 'Error', '1111': 'Check', '1112': 'Error', '1113': 'Error', '1114':
'Error'} => PubKey dosen't match
```

Figure 15. In the Message on Transmission Protocol experiment, the public key in N1 device was modified and detected.



```
[*] Return check topic_Pubkey request by node

[*] Receive send topic message request by node
```

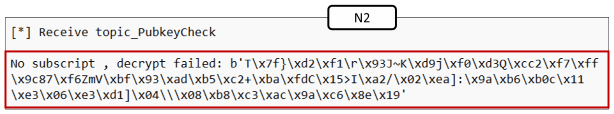
Figure 16. The experimental results of the topic-based communication in broker.



```
[*] Receive send message to topic request by client

[*] Return topic_PubkeyCheck request by node

[*] Receive topic_PubkeyCheck
{1110: 'Check', '1111': 'Check', '1112': 'Check', '1113': 'Check', '1114': 'Check'}
=> PubKey match => Decrypted Message Send

No subscript , decrypt failed: b'T\x7f}\xd2\xf1\r\x93J~K\xd9j\xf0\xd3Q\xcc2\xf7\xff
\x9c87\xf6ZmV\xbf\x93\xad\xb5\xc2+\xba\xfdC\x15>I\xa2/\x02\xea]:\x9a\xb6\xb0c\x11
\xe3\x06\xe3\xd1]\x04\\\x08\xb8\xc3\xac\x9a\xc6\x8e\x19'
```

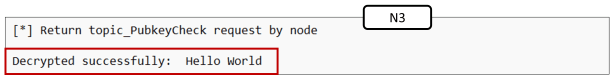
Figure 17. The experimental results of the topic-based communication in Node 1.

Empirical observations reveal that devices N3 and N4 can successfully decrypt the transmitted message, as delineated in Figures 18 and 19. In contrast, devices N1 and N2, as presented in Figures 17 and 18, experience decryption failure due to the absence of the requisite corresponding private keys. Figure 21 introduces a scenario where N1’s public key for Topic B has undergone alteration or compromise. Consequently, upon N1’s attempt to transmit a message to Topic B, it broadcasts Topic B’s public key for authentication. However, due to the integrity issues about N1’s public key, all resultant responses are marked as ”Error.”



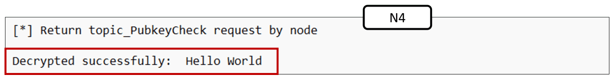
```
[*] Receive topic_PubkeyCheck

No subscript , decrypt failed: b'T\x7f}\xd2\xf1\r\x93J~K\xd9j\xf0\xd3Q\xcc2\xf7\xff
\x9c87\xf6ZmV\xbf\x93\xad\xb5\xc2+\xba\xfdC\x15>I\xa2/\x02\xea]:\x9a\xb6\xb0c\x11
\xe3\x06\xe3\xd1]\x04\\\x08\xb8\xc3\xac\x9a\xc6\x8e\x19'
```

Figure 18. The experimental results of the topic-based communication in Node 2.



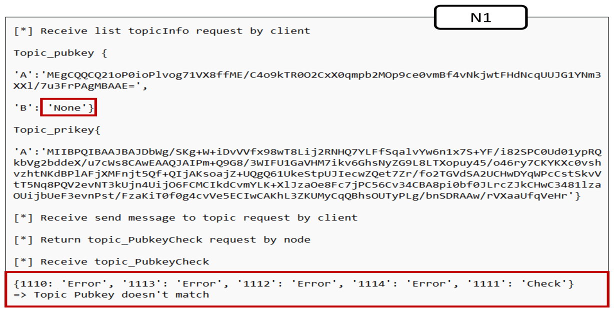
```
[*] Return topic_PubkeyCheck request by node

Decrypted successfully:  Hello World
```

Figure 19. The experimental results of the topic-based communication in Node 3.



```
[*] Return topic_PubkeyCheck request by node

Decrypted successfully:  Hello World
```

Figure 20. The experimental results of the topic-based communication in Node 4.



```
[*] Receive list topicInfo request by client

Topic_pubkey {

'A':'MEgCQQCQ21oP0ioPlvog71VX8ffME/C4o9kTR0O2CxX0qmpb2MOp9ce0vmBf4vNkjwtFHdNcqUUJG1YNm3
XXl/7u3FrPAgMBAAE=',

'B': 'None'}

Topic_prikey{

'A':'MIIBPQIBAAJBAJDbWg/SKg+W+iDvVVfx98wT8Lij2RNHQ7YLFfSqalvYw6n1x7S+YF/i82SPC0Ud01ypRQ
kbVg2bddeX/u7cWs8CAwEAAQJAIPm+Q9G8/3WIFU1GaVHM7ikv6GhsNyZG9L8LTXopuy45/o46ry7CKYKXc0vsh
vzhtNKdBPlAFjXMFnjt5Qf+QIjAKsoajZ+UQgQ61UkeStpUJIecwZQet7Zr/fo2TGVdSA2UCHwDYqWPcCstSkvV
tT5Nq8PQV2evNT3kUjn4UijO6FCMCIkdCvmYLK+XlJzaOe8Fc7jPC56Cv34CBA8pi0bf0JLrcZJkCHwC3481lza
OUijbUeF3evnPst/FzaKiT0f0g4cvVe5ECIwCAKhL3ZKUMyCqQBhsOUTyPLg/bnSDRAAw/rVXaaUfqVeHr'}

[*] Receive send message to topic request by client

[*] Return topic_PubkeyCheck request by node

[*] Receive topic_PubkeyCheck

{1110: 'Error', '1113': 'Error', '1112': 'Error', '1114': 'Error', '1111': 'Check'}
=> Topic Pubkey doesn't match
```

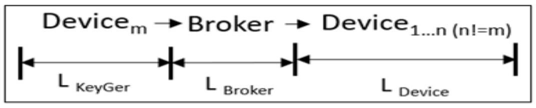
Figure 21. In the experiment, after N1's Topic B public key was tampered with, successful detection of the tampering occurred.

5. Empirical Evaluation of System Performance

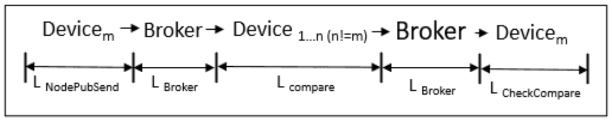
This study conducts a performance evaluation of the proposed messaging protocol on various endpoints under different load conditions. We employ emulation technology to simulate many users and generate multiple virtual devices using other ports on the same physical machine to achieve this. In this experiment, we intentionally exclude the influence of physical network latency factors to focus on assessing the protocol's performance due to the limitations of our research environment. Subsequently, we present the test results of the two protocols mentioned above.

5.1. Performance Evaluation of Transmission Protocol Device Registration Messages

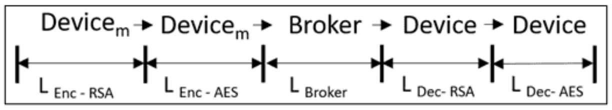
$$L_{Reg} = L_{KeyGer} + L_{Broker} + L_{Device} \quad (1)$$



$$\mathrm{L}_{\mathrm{DevicetoDevice\ (PubCheck)}} = L_{\mathrm{NodePubSend}} + L_{\mathrm{Broker}} + L_{\mathrm{compare}} + L_{\mathrm{Broker}} + L_{\mathrm{CheckCompare}} \quad (2)$$


Device$_m$ → Broker → Device $_{1...n\ (n!=m)}$ → Broker → Device$_m$
L NodePubSend
L Broker
L compare
L Broker
L CheckCompare

$$L_{DevicetoDevice(DataSend)} = L_{Enc-RSA} + L_{Enc-AES} + L_{Broker} + L_{Dec-RSA} + L_{Dec-AES} \quad (3)$$

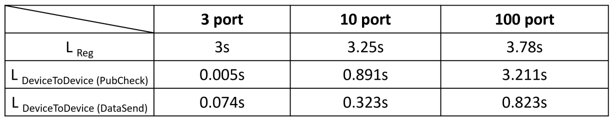


Upon entering the protocol, the device endpoint first registers, generates a key pair, stores the private key, broadcasts its public key to the broker, propagates it to other nodes, and creates equation (1).

A. Public Key Verification Transmission Protocol Message: The transmitting device transmits the target device's public key to the broker, which broadcasts it to other nodes for public key verification and returns the result. The broker also generates equation (2).

B. Transport protocol messages used for data transfer: RSA and AES transport encrypts the 1 MB message and sends it to the agent. First, it uses AES to encrypt the message and generate a symmetric key. It then encrypts the symmetric key using the target device's public key. The transmitter transmits the AES-encrypted data and RSA-encrypted symmetric key to the broker. The broker broadcasts it to all nodes, but only the target device with the private key can successfully decrypt it. The receiving device first decrypts the symmetric key using its private key and then uses the symmetric key to decrypt the ciphertext and generate equation (3).

We record the simulation results of three different quantities in the study results. Except for the "Transport Protocol Message for Public Key Checking," the time difference between "Port 3" and "Port 100" is less than 1 second. The reason why "$\mathrm{L}_{\text{One to One Public key Check}}$" has a significant time difference of 3 seconds is because "$\mathrm{L}_{\text{Broker}}$" must collect the data returned by each connection port before it can continue to execute. Regarding the extension of time, our research found that during multi-point key exchange, data needs to be received and checked one by one, which is also why this part of the time is longer.

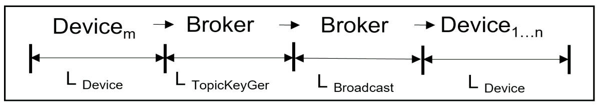


	3 port	10 port	100 port
L_{Reg}	3s	3.25s	3.78s
$L_{DeviceToDevice (PubCheck)}$	0.005s	0.891s	3.211s
$L_{DeviceToDevice (DataSend)}$	0.074s	0.323s	0.823s

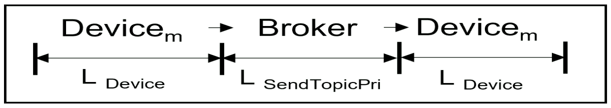
Table 3. Observations of Message on Transmission latency data in the experiment reveal that the time differences between 3 ports and 100 ports are consistently within a margin of almost one second.

5.2. Evaluating the Performance of the Subject-Specific Communication Protocol

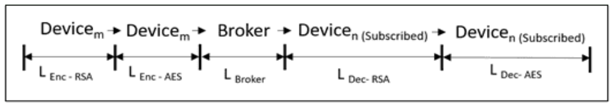
$$L_{TopicRegistration} = L_{Device} + L_{TopicKeyGer} + L_{Broadcast} + L_{Device} \quad (4)$$



$$L_{TopicSubscribe} = L_{Device} + L_{SendTopicPri} + L_{Device} \quad (5)$$

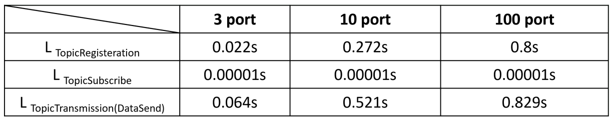


$$L_{TopicTransmission(DataSend)} = L_{Enc-RSA} + L_{Enc-AES} + L_{Broker} + L_{Dec-RSA} + L_{Dec-AES} \quad (6)$$



To evaluate the performance of the topic-specific protocol, we registered one of the devices with the topic for the agent. The broker generated a topic key pair and stored the private key. The broker then broadcasts the public key to all nodes and generates equation (4). The first job process subscribed to the topic and one of the devices subscribed to the topic from the broker. The broker then sent the corresponding private key to the subscribing device and generated an equation (5). The second action was to transmit the topic. The transmitting device sends the public key of the topic to the broker, which broadcasts it to other nodes for public key verification. Concurrent with data transmission, the broker transmits the verification result to the transmitting device. The underlying principles of the topic transmission protocol are identical to those of the single transmission protocol, and the formula can be referenced in equation (2), replacing the device's public key with the topic's public key. For transmitting data using the topic transmission protocol, the transmitting device encrypts a message and then sends it to the broker, which broadcasts it to subscribed devices. Equation (6) represents this process.

Table 4, it is seen that from 3 ports to 100 ports, the time between $L_{\text{Tpic registration}}$, $L_{\text{TpicSubscribe}}$, and $L_{\text{TpicTransmission (DataSend)}}$ is less than 1 second. Therefore, it can be proved that no matter the number of devices in the topic subscription status of different subcollections, the message transmission time is unaffected. When the number of nodes is expanded from a minimum of 3 nodes to 100 nodes, the response feedback time of the transmission time can be less than 1 second, which also proves that based on IoT mqtt communication encryption protocol we propose can be flexibly integrated into a large number of IoT device endpoint environments without affecting the data exchange time.



	3 port	10 port	100 port
$L_{TopicRegisteration}$	0.022s	0.272s	0.8s
$L_{TopicSubscribe}$	0.00001s	0.00001s	0.00001s
$L_{TopicTransmission(DataSend)}$	0.064s	0.521s	0.829s

Table 4. Observations of Subject-Specific Communication latency data in the experiment reveal that the time differences between 3 ports and 100 ports are consistently within a margin of almost one second.

6. Conclusion And Future Work

Our research mitigates the security vulnerabilities of the original MQTT mechanism by integrating the public and private keys and broadcast verification features of blockchain technology with a single transmission protocol and a topic transmission protocol, enabling MQTT to be used for data transmission in a broader range of scenarios. The experimental performance results show that the delay time remains relatively constant regardless of the number of devices or the amount of data transmitted. This result indicates that our software architecture is feasible in the Windows environment. In future research, we will attempt to port the encrypted communication module to the Internet of Things Raspberry Pi environment for verification. We will also increase the number and size of transmitted data samples to refine and improve our proposed encrypted communication architecture.

ACKNOWLEDGMENT

This research was funded by National Science and Technology Council of the Republic of China grant number NSTC 112-2221-E-031-005-.

References

[1] Feroz Ahmad Balouch, Khan Mohammad Wafa, and Ali Ahmad. Internet of things (iot), its application area and combination with gps. *Galaxy International Interdisciplinary Research Journal*, 10(1):725–734, 2022. 2

[2] Masoud Emamian, Aref Eskandari, Mohammadreza Aghaei, Amir Nedaei, Amirmohammad Moradi Sizkouhi, and Jafar Milimonfared. Cloud computing and iot based intelligent monitoring system for photovoltaic plants using machine learning techniques. *Energies*, 15(9):3014, 2022. 2

[3] Arun Rana, Sharad Sharma, Kashif Nisar, Ag Asri Ag Ibrahim, Sachin Dhawan, Bhawani Chowdhry, Samreen Hussain, and Nitin Goyal. The rise of blockchain internet of things (biot): Secured, device-to-device architecture and simulation scenarios. *Applied Sciences*, 12(15):7694, 2022. 2

[4] Edoardo Longo and Alessandro EC Redondi. Design and implementation of an advanced mqtt broker for distributed pub/sub scenarios. *Computer Networks*, 224:109601, 2023. 2

[5] Ahmed J Hintaw, Selvakumar Manickam, Shankar Karuppayah, Mohammad Adnan Aladaileh, Mohammed Faiz Aboalmaaly, and Shams Ul Arfeen Laghari. A robust security scheme based on enhanced symmetric algorithm for mqtt in the internet of things. *IEEE Access*, 2023. 2

[6] Mingyue Xie, Jun Liu, Shuyu Chen, and Mingwei Lin. A survey on blockchain consensus mechanism: research overview, current advances and future directions. *International Journal of Intelligent Computing and Cybernetics*, 16(2):314–340, 2023. 2

[7] Aichih Chang, Nesreen El-Rayes, and Jim Shi. Blockchain technology for supply chain management: A comprehensive review. *FinTech*, 1(2):191–205, 2022. 2

[8] Dinesh Kumar, Sunil Kumar, and Akashdeep Joshi. Assessing the viability of blockchain technology for enhancing court operations. *International Journal of Law and Management*, 2023. 2

[9] Basel Halak, Yildiran Yilmaz, and Daniel Shiu. Comparative analysis of energy costs of asymmetric vs symmetric encryption-based security applications. *IEEE Access*, 10:76707–76719, 2022. 2

[10] Chandu Kota. Secure file storage in cloud using hybrid cryptography. *Available at SSRN 4209511*, 2022. 2

[11] Parveen Sehgal Satinder et al. Design an algorithm for data encryption used in hybrid technique to implementation in database. *Pakistan Heart Journal*, 56(2):1465–1472, 2023. 2

[12] Christian Stohrer and Thomas Lugrin. Asymmetric encryption. *Trends in Data Protection and Encryption Technologies*, pages 11–14, 2023. 2

[13] Prakash Kuppuswamy, Saeed Qasim Yahya Al Khalidi Al, Rajan John, Mohammad Haseebuddin, Ahmed Ali Shaik Meeran, et al. A hybrid encryption system for communication and financial transactions using rsa and a novel symmetric key algorithm. *Bulletin of Electrical Engineering and Informatics*, 12(2):1148–1158, 2023. 2

[14] Roopali Sood and Harpreet Kaur. A literature review on rsa, des and aes encryption algorithms. *Emerging Trends in Engineering and Management*, pages 57–63, 2023. 2