sensors

MDPI

Article

End-to-End Encrypted Message Distribution System for the Internet of Things Based on Conditional Proxy Re-Encryption

Shi Lin [1], Li Cui [2] and Niu Ke [1,*]

[1] School of Cryptographic Engineering, Engineering University of PAP, Xi'an 710000, China; slshilin@126.com
[2] School of Information and Communication, National University of Defense Technology, Wuhan 430000, China; lc_licui17@nudt.edu.cn
* Correspondence: niuke@163.com

Abstract: In light of the existing security vulnerabilities within IoT publish–subscribe systems, our study introduces an improved end-to-end encryption approach using conditional proxy re-encryption. This method not only overcomes limitations associated with the reliance on a trusted authority and the challenge of reliably revoking users in previous proxy re-encryption frameworks, but also strengthens data privacy against potential collusion between the broker and subscribers. Through our innovative encryption protocol, unauthorized re-encryption by brokers is effectively prevented, enhancing secure communication between publisher and subscriber. Implemented on HiveMQ, an open-source MQTT platform, our prototype system demonstrates significant enhancements. Comparison to the state-of-the-art end-to-end encryption work, encryption overhead of our scheme is comparable to it, and the decryption cost is approximately half of it. Moreover, our solution significantly improves overall security without compromising the asynchronous communication and decentralized authorization foundational to the publish–subscribe model.

Keywords: Internet of Things; end-to-end encryption; conditional proxy re-encryption; message broker; HiveMQ

Citation: Lin, S.; Cui, L.; Ke, N. End-to-End Encrypted Message Distribution System for the Internet of Things Based on Conditional Proxy Re-Encryption. *Sensors* **2024**, *24*, 438. https://doi.org/10.3390/s24020438

Academic Editor: Jian Li

Received: 29 November 2023
Revised: 1 January 2024
Accepted: 2 January 2024
Published: 10 January 2024

1. Introduction

To realize data communication among a large number of entities, large-scale Internet of Things (IoT) system generally uses the publish–subscribe (pub/sub) paradigm for data distribution. The most commonly used protocols which work in the pub/sub paradigm are Message Queuing Telemetry Transport (MQTT) [1], Advanced Message Queuing Protocol (AMQP) [2], etc. The subscribers can subscribe to a "message topic", and publisher can publish messages to the "message topic". All subscribers will receive the publisher's message through the routing of the message broker between the publisher and subscribers. The pub/sub paradigm decouples the senders and receivers in time and space. They do not need to be directly connected or online simultaneously. It is more flexible, efficient, and scalable than the point-to-point data exchange mode. A typical pub/sub-based IoT system includes three types of components: IoT devices, a message broker, and a user management application. The device and the management application serve as the publisher and subscriber, respectively. The device publishes the data collected by its sensor to a specific topic, and the authorized user of the device subscribes to that topic through the application. The message broker in the middle routes the data to all the authorized users.

At present, Transport Layer Security (TLS) [3,4] is widely used in the industry to protect the data between the client (publisher or subscriber) and the message broker. The broker decrypts the ciphertext from the publisher to obtain the plaintext, encrypts the plaintext again with the key negotiated with each subscriber, and then forwards the corresponding ciphertext to the subscriber. However, the broker can obtain all the data generated by the client and has complete control over the user's data [5]. Furthermore, the message broker maintained by IoT manufacturers is not completely trusted. Recent research

shows that no-security policies are implemented by a large number of accessible message brokers, which allows anyone to receive data or inject messages [6]. Some researchers investigated a specific traffic monitoring system and discovered that the MQTT message broker was disclosing the traffic flow conditions in a specific area of Mexico City [7]. Even if the message broker deploys a security policy, users need to totally trust the broker. Currently, the message broker in pub/sub-based IoT system are deployed either by IoT device manufacturers based on existing commercial message servers (such as EMQX [8], HiveMQ [9] or Solace [10]), or based on the IoT cloud platform provided by third-party cloud computing manufacturers (such as Alibaba cloud [11], Amazon AWS cloud [12]). The message broker is established and maintained by the IoT device manufacturer or the IoT cloud platform [13]. These manufacturers are not entirely reliable. If the administrator operates incorrectly or is bribed by a spy, or if the manufacturer is for profit purposes, user data are likely to be abused or shared with unauthorized entities, which poses a threat to the security of user data.

In addition, at present, most pub/sub-based IoT systems are constructed based on MQTT protocol, which is not designed for hostile environments. Jia Yan et al. [14] found that the MQTT protocol has serious defects. The platform using this protocol can enable adversaries to steal users' private information and forge users' device status.

In order to prevent the threats brought by malicious message brokers, PICADOR [15] uses proxy re-encryption (PRE [16]) technology to provide end-to-end encryption from publishers to subscribers. In PRE, given a re-encryption key to the semi-trusted proxy, the proxy can convert the ciphertext encrypted with the public key of user A into the ciphertext encrypted with the public key of user B without obtaining any plaintext information from the ciphertext. The proxy re-encryption process can be described as: $E(pk_A, m) \xrightarrow{rk_{A\to B}} E(pk_B, m)$]; here, $rk_{A\to B}$ is the re-encryption key from A to B.

In PICADOR, the publisher encrypts its message with its public key. The broker re-encrypts the published message using the re-encryption key of each subscriber and then sends the corresponding ciphertext to each subscriber. The subscriber can decrypt the ciphertext with their private key. PICADOR needs a trusted authority to generate the re-encryption key for each subscriber according to the publisher's private key and each subscriber's public key. When revoking the authorization of a subscriber, the simplest way is that the broker no longer re-encrypts messages for the revoked subscribers. However, the broker is not entirely trusted. If it is compromised and still re-encrypts for the revoked user, then the revoked user can still receive the latest message. Therefore, depending on the broker for user revocation is not completely reliable. If we do not rely on the broker to revoke users, then we can only change the public–private key pair of the publisher once revocation is required and regenerate the re-encryption key for all the remaining authorized users, which would result in frequent changes to the public–private key pair of the publisher. The long-term public–private key of the user is usually used for authentication either, both between users and between users and brokers. If the user's public–private key pair changes frequently, then there will be inconveniences encountered during authentication.

The reason of the problem in PICADOR is that traditional proxy re-encryption allows the proxy to convert all ciphertexts without restriction [17]. As long as the proxy possesses the re-encryption key from A to B ($rk_{A\to B}$), the proxy can convert all ciphertexts encrypted with the public key of A into ciphertexts which could be decrypted using the private key of B. This all-or-nothing feature is not suitable for applications that need fine-grained authorization of decryption capability. Based on this, Weng et al. proposed the concept of Conditional Proxy Re-encryption (CPRE) [18], which allows for the conditional conversion of ciphertext. In CPRE, when generating ciphertext using user A's public key, a condition value is introduced at the same time, and the re-encryption key from A to B are also related to a condition value. Only when the condition value used for generating the ciphertext is equal to the condition value related to the re-encryption key, the proxy can convert the ciphertext encrypted with the public key of A into the ciphertext encrypted with the public key of B. A can prevent the proxy from performing unauthorized re-encryption

by controlling the change of condition value [17]. The process of conditional proxy re-encryption can be described as: $E(pk_A, m, \omega) \overset{rk}{\underset{A}{\overset{\omega'}{\rightarrow}}} E(pk_B, m)\left(\omega = \omega'\right)$.

Because CPRE has significant advantages over PRE in fine-grained authorization, this paper introduces CPRE for the first time to realize end-to-end encryption in pub/sub-based IoT system to prevent broker from performing unauthorized re-encryption. We investigate a large number of existing conditional proxy re-encryption schemes. According to the principles of low computing and communication overhead and high security, the conditional proxy re-encryption algorithm proposed by Weng et al. in 2009 [19] is selected in our system.

In our system, the publisher uses its private key, conditional value, and subscriber's public key to generate the conditional re-encryption key for each subscriber, and sends the conditional re-encryption key to the broker. When publishing a message, the publisher encrypts the message with its public key and condition value and sends the ciphertext to the broker. The broker uses the re-encryption key of each subscriber to re-encrypt the message and then sends the re-encrypted ciphertext to the corresponding subscriber. Finally, the subscriber can obtain the plaintext by decrypting the message with their private key. When a subscriber is revoked, the publisher updates the condition value and generates new condition re-encryption keys for each remaining subscriber. Specifically, the main contributions of our system are as follows:

(1) The conditional proxy re-encryption (CPRE) algorithm is introduced to solve the end-to-end encryption problem in the pub/sub-based IoT system. the re-encryption key is associated with a condition value. By changing the condition value, the publisher can ensure that the proxy can not perform unauthorized re-encryption, thereby achieving reliable revocation of subscribers.

(2) By using an open-source MQTT message server, HiveMQ, we implement a prototype end-to-end encryption system for a pub/sub-based IoT system based on CPRE, and further enhance the system's performance through hybrid encryption and hash chain. Moreover, the performance of the system is tested, which shows that our system is not only easy to implement on existing commercial message servers, but also has high performance.

2. Related Works

2.1. End-to-End Encryption in IoT

At present, a large number of scholars have studied the end-to-end security in pub/sub-based systems [20,21]. This section survey the current research status of end-to-end encryption schemes according to the technology used.

The scheme based on a trusted message broker: Jia Yan [14] proposed MOUCON to solve access control issues in MQTT, the MQTT broker in MOUCON is responsible for verifying each client's access to the message. Clients must fully trust the broker and cannot resolve the security threat to user data caused by an untrusted broker.

The scheme based on the trusted key server: Markus et al. [5] propose an end-to-end security scheme for Cyber–Physical Systems (CPS). The scheme relies on trusted key servers to distribute topic keys for publishers and subscribers. The key server stores the global authorization information of the system and the encryption key. If the key server is compromised by the adversary, all the account information, authorization information, and the encryption keys of the system will be exposed.

The scheme based on identity-based encryption (IBE): JEDI [20] realizes the end-to-end encryption between devices and users in IoT, facilitates asynchronous communication, and supports decentralized authorization for the key. The scheme does not require any modifications to the message broker, which is convenient for deployment. However, this method uses the identity-based encryption algorithm with wildcards (WKD-IBE) [21], the algorithm has a high computational complexity, making it unsuitable for resource-constrained IoT devices.

The scheme based on secret sharing: Sana Belguith et al. [22] proposed an efficient and revocable secure publish–subscribe system. The system divided the broker into three parts to handle the functions of topic matching, routing, and message sending separately. As long as the adversary does not simultaneously break all three brokers, the solution remains secure. However, this solution requires the customization of a special message broker, which is inconvenient to deploy.

The scheme based on special hardware: Segarra et al. [23] restrict the broker to run only in a trusted execution environment (TEE) [24], thus ensuring that the broker functions as intended by the deployer. However, the installation and deployment of TEE requires professional management, as well as its maintenance, which results in high costs. In addition, there are security attacks against the current mainstream TEE [25], so TEE still has some security risks.

The scheme based on proxy re-encryption: PICADOR [15] implements end-to-end encryption between publishers and subscribers using proxy re-encryption. As can be inferred from the above analysis, this scheme depends on a trusted authorization center to generate re-encryption keys and relies on the broker to revoke users. However, this approach has the drawback of unreliable revocation.

2.2. Conditional Proxy Re-Encryption Schemes

Mambo and Okamoto [26] first introduced the concept of decryption capability authorization, which has higher performance than the method of decrypting and then encrypting the ciphertext. In 1998, Blaze, Bleuner, and Strauss formally introduced the concept of proxy re-encryption (PRE) [16], and since then, lots of research has been carried out around PRE. PRE allows a semi-trusted agent to transform the decryption capability of a ciphertext without obtaining any valid information about the ciphertext, and is widely used in encrypted email transmission, secure distributed file systems, and encrypted spam filtering, etc. PRE can be categorized according to different criteria. Based on the direction of re-encryption, PRE can be categorized into one-way PRE and two-way PRE. Additionally, based on the number of re-encryptions allowed, it can be categorized into single-hop PRE and multi-hop PRE.

Traditional proxy re-encryption is unable to provide fine-grained authorization of decryption capabilities. In response, Weng et al. introduced the concept of conditional proxy re-encryption (CPRE) [18] and developed the first CPRE scheme. The re-encryption key of the scheme consists of two parts: the re-encryption key and the conditional key. However, the scheme only considered the security of the second ciphertext layer and not the first ciphertext layer. Weng [27] pointed out that the scheme described in the literature [18] is vulnerable to Chosen-Ciphertext Attacks (CCAs). Weng [27] redefined a more stringent security model for CPRE and proposed a new efficient CPRE scheme. Both Shao [28] and Liang [29] proposed CCA-secure identity-based CPRE under the DBDH (Decisional Bilinear Diffie–Hellman Problem) assumption. However, the literature [30] points out that the scheme given by Liang [29] is insecure.

Fang et al. [31] proposed an anonymous CPRE scheme that enables keyword search. Subsequently, Jae Woo Seo et al. [32] proposed a type-based privacy requirement engineering (Type-based PRE) scheme, where "type" refers to a keyword that is equivalent to "condition" in CPRE. Thus, this scheme is essentially similar to the CPRE scheme, which achieves fine-grained authorization of user decryption capabilities. Son et al. [33] proposed a CPRE for big data sharing on cloud platforms by outsourcing the re-encryption key generation and decryption to the servers. Qiu et al. [19] and Liang et al. [34] proposed CCA-secured CPREs, respectively. Ge et al. [35] proposed an identity-based CPRE scheme. The authors proposed an identity-based CPRE scheme that enables contingent gate computation on conditionals. Hu Xiong et al. [36] introduced a unidirectional multi-hop identity-based CPRE scheme that facilitates flexible and efficient data authorization in cloud computing environments and demonstrated its security in the standard model. Arinjita et al. [37] presented a conditional proxy re-encryption scheme that does not require pairwise operations.

The scheme is not reliant on the bilinear pair operation for construction and has lower computational overhead. However, in this scheme, if the receiver conspires with the agent, the agent can compute the sender's private key as long as it possesses two conditional re-encryption keys. Therefore, the scheme proposed by Arinjita et al. [37] is not able to resist the conspiracy attack between the agent and the receiver.

The following compares various CPRE (Conditional Proxy Re-encryption) schemes, and the results are presented in Tables 1 and 2. The schemes proposed in the literature [18,29,37] exhibit security issues and are therefore not included in the comparison. Let $|G|$ and $|GT|$ denote the bit lengths of elements in groups G and GT, respectively. $|Z_p|$ represents the bit length of elements in the prime field $|Z_p|$. $|m|$ stands for the bit length of the plaintext. t_p and t_e represent the time required for a single bilinear pairing operation and a single exponentiation operation, respectively. $|\sigma|$ represents the bit length of the signature output by a strongly unforgeable one-time signature algorithm. svk is the length of the verification key for the strongly unforgeable one-time signature algorithm. t_v is the time taken to verify a strongly unforgeable one-time signature. t represents the size of the access tree, and w denotes the size of attributes.

Table 1. Comparison of the computation overhead of each CPRE algorithm.

Scheme	Re-Encryption Key Generation	Encrypt	Re-Encrypt	Decrypt
[27]	$4t_e$	$3t_e + t_p$	$3t_p$	$2t_e + t_p$
[28]	$5t_e + t_p$	$8t_e + t_p$	$6t_p$	$4t_e + 8t_p$
[31]	$2t_e$	t_e+$5t_p + t_s$	$2t_e + t_p + t_v$	$t_e + t_p$
[32]	t_e	$5t_e + t_p$	$5t_e$+$2t_p$	$2t_e + 4t_p$
[33]	$4t_e$	$3t_e$+$2t_p$	t_e+$2t_p$	$3t_e$+$2t_p$
[19]	$3t_e$	$4t_e + t_p$	$4t_p$	$2t_e + t_p$
[34]	$9t_e + t_p$	$5t_e + t_p$	$6t_e$+$7t_p$	$5t_e$+$12t_p$
[35]	$8t_e + t_p + t_s$	$6t_e + t_p + t_s$	$(9 + 2w + t)t_p + t_v$	$t_e + 8t_p + t_v$
[36]	$6t_e$	$6t_e + 2t_p$	$t_e + 4t_p + t_v$	$t_e + 3t_p + t_v$

Table 2. Comparison of the communication overhead of each CPRE algorithm.

Scheme	Initial Ciphertext	Re-Encryption Key	Ciphertext after Re-Encryption
[27]	$2\|G\| + \|G_T\| + \|m\|$	$2\|G\|$	$2\|G\| + \|G_T\| + \|m\|$
[28]	$4\|G\| + \|G_T\| + \|Z_p\|$	$2\|G\| + 2\|G_T\| + 2\|Z_p\|$	$4\|G\| + \|G_T\| + \|m\|$
[31]	$\|G\| + 3\|G_T\| + svk + \|\sigma\|$	$2\|G\| + 2\|Z_p\|$	$2\|G_T\| + svk + \|m\|$
[32]	$2\|G\| + \|G_T\| + svk + \|\sigma\|$	$\|G\| + \|Z_p\|$	$4\|G\| + \|G_T\| + svk + \|\sigma\|$
[33]	$\|G\| + 2\|G_T\|$	$2\|G\|$	$2\|G\| + 2\|G_T\|$
[19]	$2\|G\| + \|m\|$	$2\|G\|$	$\|G_T\| + \|m\|$
[34]	$4\|G\| + \|G_T\| + svk + \|\sigma\|$	$6\|G\| + \|G_T\| + svk + \|\sigma\|$	$3\|G\| + \|G_T\| + svk + \|\sigma\| + \|m\|$
[35]	$(w + 4)\|G\| + \|G_T\| + w + \|\sigma\| + \|m\|$	$6\|G\| + \|G_T\| + \|\sigma\| + t$	$(w + 6)\|G\| + 2\|G_T\| + 2\|\sigma\| + 2\|m\|$
[36]	$3\|G\| + 2\|G_T\| + svk + \|\sigma\|$	$2\|G\| + \|Z_p\|$	$3\|G\| + 2\|G_T\| + svk + \|\sigma\|$

In the practical implementation of the CPRE algorithm, the re-encryption algorithm is executed by a semi-trusted agent, which is generally deployed on servers or clouds with more abundant resources. In contrast, the encryption and decryption operations are typically performed by IoT devices or personal handheld electronic devices with significantly smaller computational resources than the agent. Therefore, the overall principle in selecting CPRE algorithms is to choose the scheme with lower computational and communication overhead. When computational and communication overheads are comparable, the scheme with lower encryption and decryption overheads is chosen. In terms of security, the schemes in the table are all provably CCA secure in either the standard model or the stochastic prediction model. Although the schemes proven to be secure under the standard model are theoretically more reliable, in this paper, we choose the schemes proven to be secure under the stochastic prediction model. This is because the security of such security schemes depends only on the hash function itself, and no practical attack has yet occurred

that can compromise a practical cryptographic algorithm proven to be secure under the stochastic prediction model (excluding some carefully constructed counterexamples by humans [38]). In addition, these security schemes are computationally more efficient and have a wider range of applications.

Based on the aforementioned principles, this paper utilizes the conditional proxy re-encryption algorithm proposed by Weng et al. [27] in 2009 to achieve the encryption of the publish–subscribe system of this paper from the publisher side to the subscriber side.

3. Preliminaries

The conditional proxy re-encryption CPRE scheme includes the following algorithms, and its workflow is given in Figure 1:

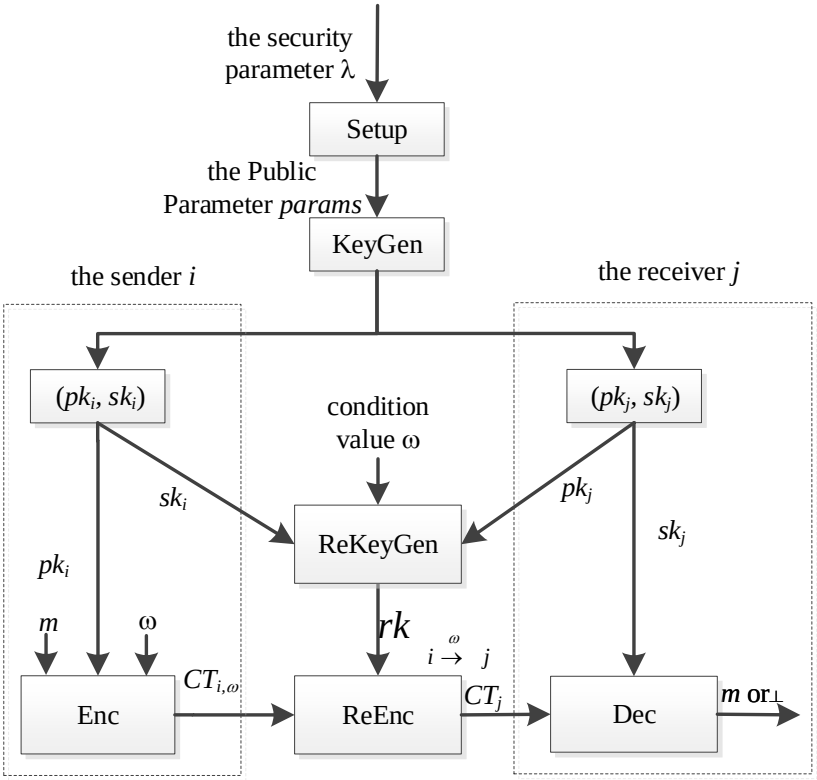

Figure 1. Workflow of CPRE.

$Setup\left(\lambda^k\right)$: Input the security parameter λ^k, and the algorithm outputs the public parameter $params$.

$KeyGen\left(\lambda^k\right)$: Each entity uses this randomized key generation algorithm to generate a public–private key pair (pk_i, sk_i).

$ReKeyGen\left(sk_i, \omega, pk_j\right)$: Input the private key sk_i of the sender, the condition value ω and the public key pk_j of the receiver, the re-encryption key generation algorithm outputs the re-encryption key $rk_{i\xrightarrow{\omega}j}$ from sender i to receiver j.

$Enc_1(pk_i, m)$: Input the public key pk_i of the sender and plaintext m, the first layer encryption algorithm outputs the first layer ciphertext CT_i. The ciphertext cannot be re-encrypted.

$Enc_2(pk_i, m, \omega)$: Input the public key pk_i of the sender, the plaintext m and the condition value ω, the second-layer encryption algorithm will output the second-layer ciphertext $CT_{i,\omega}$. This ciphertext can be re-encrypted using an appropriate re-encryption key into a first-layer ciphertext for different recipients.

$ReEnc\left(CT_{i,\omega}, rk_{i\xrightarrow{\omega}j}\right)$: Input the second-layer ciphertext $CT_{i,\omega}$, the re-encryption key $rk_{i\xrightarrow{\omega}j}$, the proxy runs the re-encryption algorithm to output the first-layer ciphertext CT_j .

$Dec_1(CT_j, sk_j)$: Input the first-layer ciphertext CT_j and private key sk_j, the first-layer decryption algorithm outputs plaintext m or error symbol $\perp$.

$Dec_2(CT_{i,\omega}, sk_i)$: Input the second-layer ciphertext $CT_{i,\omega}$ and private key sk_i, the second-layer decryption algorithm outputs the plaintext message m or error symbol $\perp$.

By introducing a conditional value into the generation of the re-encryption key and the second layer encryption algorithm, the conditional proxy re-encryption algorithm ensures that the proxy cannot perform unauthorized re-encryption.

4. End-to-End Encryption System Based on CPRE

4.1. System Framework

Our CPRE-based end-to-end encryption system consists of three types of entities: IoT devices (referred to as the sender), message broker, and multiple authorized users (referred to as the receiver).

Our system utilizes the conditional proxy re-encryption algorithm proposed by Weng et al. [19] to achieve end-to-end encryption from the publisher to the subscribers in pub/sub-based IoT system. The specific algorithm design can be found in the literature [19]. In our system, the device owner generates a re-encryption key for each authorized user and send the re-encryption key to the Broker. The device acts as the sender and encrypts the message with its public key and a condition value. The broker re-encrypts for each subscriber using the corresponding re-encryption key. All authorized users can decrypt the ciphertext using their private key. The framework of our system is shown in Figure 2.

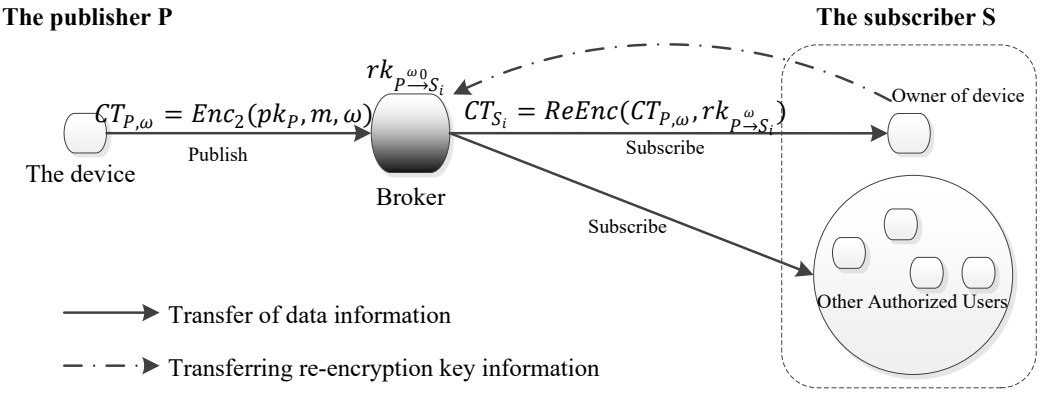

Figure 2. The framework of IoT end-to-end encryption system.

CPRE differs from PRE in that its re-encryption key $rk_{P\xrightarrow{\omega}S_i}$ and the second-layer ciphertext $CT_{P,\omega}$ are both associated with a condition value ω. If the authorization for certain users need to be revoked, the device owner just need to generate a new condition value ω' and send to the device. Then, the device owner generates a new re-encryption key $rk_{P\xrightarrow{\omega'}S_i}$ for each remaining recipient with the device's private key sk_P, the new condition value ω', and the public keys of the remaining authorized users pk_{S_i}. The owner no longer generates re-encryption keys for the revoked users. The device uses the new condition value ω' to generate the second layer ciphertext $CT_{P,\omega'}$. The broker re-encrypts with the new re-encryption key $CT_{P,\omega'}$ for the remaining authorized users, so that the remaining legitimate users can decrypt the message correctly. Since the re-encryption key of the revoked users is not updated, it is still associated with the previous condition value ω, and the new ciphertext corresponds to the updated condition value ω'. Even if the broker is compromised and still re-encrypts the message for the revoked users—as the condition value in the re-encryption key and second layer ciphertext are not equal—the revoked user cannot decrypt the re-encrypted ciphertext correctly with their private key.

4.2. System Workflow

Specifically, the workflow of our CPRE-based end-to-end encryption scheme includes the following steps.

4.2.1. User Registration

When a user wants to utilize our system for device management or monitoring, he must complete the user registration process through the client application. The algorithm $KeyGen\left(1^k\right)$ of CPRE is integrated into the application, allowing the user to generate their public and private key pairs $\left(pk_{S_i}, sk_{S_i}\right)$. The user keeps their private key sk_{S_i} secret.

4.2.2. Device Registration

When a user purchases a new IoT device, the user is the owner of the device and is responsible for device registration and authorization control. Typically, a newly purchased IoT device starts its life cycle with "device discovery" [3]. In this stage, the device owner requests to add a device through the client APP, and the APP establishes a local connection with the device to complete the device registration and the binding of the device and its owner. We assumes that during the "device registration" phase, the device interacts with the client APP of the device owner through a local connection, exchanging basic information and performing mutual authentication. During the device registration process, the device owner utilizes the built-in key generation algorithm $KeyGen(1^k)$ of the APP to generate public and private key pairs (pk_P, sk_P) for the device. Additionally, the owner generates a random initial condition value ω_0, and transfers the initial condition value ω_0 and device's public–private key pair (pk_P, sk_P) to the device through the "Local Connection" established in registration phase. The device owner also keeps the private key sk_P of the device secretly, which is used to generate the conditional re-encryption key for other authorized users.

4.2.3. Authorization Phase

When the device owner wants to authorize the access rights of the device to other users, the device owner uses the private key sk_p of the device, the condition value ω_0, and the public key pk_{S_i} of each authorized user to generate a conditional re-encryption key $rk_{P \xrightarrow{\omega_0} S_i}$ for each authorized user, which will be sent to the broker of the publish–subscribe system.

4.2.4. Message Transmission Stage

The device runs the CPRE encryption algorithm, encrypts the collected information with its public key pk_P and condition value ω_0 and obtain the ciphertext $CT_{P,\omega_0} = Enc_2\left(pk_p, m, \omega_0\right)$, which will be sent to the broker. Then, the broker re-encrypts the ciphertext according to the conditional re-encryption key $rk_{P \xrightarrow{\omega_0} S_i}$ of each authorized user and sends the re-encrypted ciphertext $CT_{S_i} = ReEnc\left(CT_{P,\omega_0}, rk_{P \xrightarrow{\omega_0} S_i}\right)$ to the corresponding authorized user. Finally, each authorized user uses their private key sk_{S_i} to decrypt the ciphertext CT_{S_i} and obtains the plaintext $m = Dec_1\left(CT_{S_i}, sk_{S_i}\right)$.

4.2.5. Revocation Phase

When the device owner needs to revoke the access permission of one user, first, the device owner randomly selects a new condition value ω_1; then, the owner generates a new conditional re-encryption key for each remaining authorized user with the user's public key pk_{S_j}, the device's private key of sk_P, and the new condition value ω_1. Furthermore, the updated re-encryption keys are sent to the broker. Finally, the device owner encrypts the new condition value with the public key of the device and sends the ciphertext to the device.

The device decrypts with its private key sk_P to obtain the new condition value ω_1, it updates the condition value to the new one. Similarly, when the broker receives the new conditional re-encryption key $rk_{P \xrightarrow{\omega_1} S_j}$ distributed by the device owner, the conditional

re-encryption key will also be updated from $rk_{P \xrightarrow{\omega_0} S_i}$ to $rk_{P \xrightarrow{\omega_1} S_j}$. Then, the device encrypts the message with the new condition value, and the broker re-encrypts with the new conditional re-encryption key.

4.3. System Optimization

4.3.1. Hybrid Encryption

Most IoT devices are low-power devices with limited resources, so we use hybrid encryption to further reduce device-side overhead. Before encrypting the collected messages, the device first selects a random symmetric key k, encrypts the key with CPRE and sends to the broker. Each authorized subscriber can decrypt the re-encrypted ciphertext with their private key, thereby obtaining the same symmetric key k. Since then, subsequent communications between the device and each subscriber can be encrypted using the symmetric key k.

4.3.2. Hash Chain

Whenever the set of authorized users changes (such as new users join or old users are revoked), if CPRE is used to generate new conditional re-encryption keys for all remaining authorized users, when the authorized user set changes frequently, generating and transmitting conditional re-encryption keys imposes significant overhead on the device owner. Therefore, when a new user joins, a symmetric key is distributed to the new user by means of the hash chain [39]. The CPRE algorithm is used only when the user is revoked.

Specifically, the process of distributing a symmetric key to a new user by using a hash chain is as follows: Assume that the symmetric key shared between the device and each subscribing user is k before the new user joins. Before a new user joins, the device owner uses k as the input of the one-way trapdoor function to obtain a new session key $k_1 = Hash(k)$, and sends the new session key to the new joined user. At the same time, the device owner broadcasts the key update command, so that the device and other legitimate users can also update their shared key k to k_1 through the one-way trapdoor function. Then, the consistency of the shared session key between the device and all its authorized users can be ensured.

4.4. System Analysis

Below, we analyze our CPRE-based end-to-end encryption system in IoT.

4.4.1. Satisfy Confidentiality

When a new user is authorized to join, the session key is updated through the hash chain. Based on the unidirectionality of the hash function, the new user cannot deduce the session key k before joining by using the session key k_1.

When the user is revoked, the system uses CPRE to update the session key. The device owner randomly selects a new condition value, and generates new conditional re-encryption keys for the remaining users, but no longer generates new conditional re-encryption keys for revoked users. Therefore, when the device uses CPRE encryption to transmit a randomly selected new session key, the new condition value is used, and the broker also uses the new conditional re-encryption key for re-encryption, so that the remaining legitimate users can use their own private key to decrypt and obtain the new session key. The conditional re-encryption key of the revoked user is also related to the old condition value, even if the broker colludes with the revoked user, still re-encrypts with the old conditional re-encryption key. As the condition value in the cipertext of the device is not equal to the condition value in the revoked user's re-encryption key, so the revoked user cannot decrypt or obtain the new session key. Based on the above analysis, the revoked user cannot obtain a new session key even if he colludes with the broker. In addition, our scheme is constructed based on the CPRE algorithm, and the broker can only use the re-encryption key to convert the ciphertext, and cannot obtain any plaintext message of the ciphertext based on the security proof of CPRE in [19].

4.4.2. Support Asynchronous Communication

When a new user joins, the session key is updated through the hash chain, and the offline user does not affect the session key update process between the online user and the device. When the user is revoked, the update of the conditional re-encryption key involves the device owner. As long as the device owner is online, the broker can obtain the latest conditional re-encryption key, and the device can obtain the latest conditional value, offline users do not affect the process. In short, offline users do not affect the key update process of online users, and our scheme supports asynchronous communication.

4.4.3. Support Decentralized Authorization

In our scheme, the authorization and revocation of device access rights are only controlled by its owner and do not rely on trusted third parties. If a device owner is compromised by an adversary, only the security of the owner and its managed devices will be affected, and the security of other devices and users will not be affected.

4.4.4. Support the Decoupling of Publishers and Subscribers

Our scheme is designed based on the CPRE scheme. The device uses its public key to encrypt the data. The device does not need to encrypt for each authorized user, and does not need to care about which users subscribed to its data. The broker completes the conversion and distribution of the ciphertext sent by the device. Therefore, the publisher (ie, device) and subscriber (ie, user) in our scheme are decoupled, and their relationship is controlled and managed by the device owner.

5. Prototype Implementation and Performance Analysis

This section implements the above-mentioned IoT end-to-end encryption system and tests the system's performance.

5.1. Implementation of the Prototype System

We implement each module based on the Java language. Our prototype system includes three types of entities: the publisher, multiple subscribers, and the message broker which can perform re-encryption operations. The implementation of the CPRE algorithm [19] is based on the JPBC (the Java Pairing-Based Cryptography Library) [40]. The establishment of the message broker is based on the open-source MQTT server HiveMQ [9], which supports customized function extensions. The development of the publisher and the subscriber is based on the Eclipse Paho Java Client library [41]. The prototype system runs on a laptop configured with an Intel Core i7-5600U 2.6GHZ CPU and 8G RAM (Intel, Santa Clara, CA, USA). The computer operating system is Windows 7. The development environment of the publisher, subscriber, and Broker is IntelliJ IDEA 2018.1.6. The functions of the publisher and subscriber are simulated using a Java console program.

5.1.1. CPRE Implementation

The CPRE scheme used in our system is constructed based on the symmetric bilinear group. Therefore, we use the configuration file "a.properties" provided by the JPBC library to generate a type A symmetric bilinear group based on a prime-order elliptic curve $y^2 = x^3 + xmodp(p \equiv 3mod4)$, while the base field size is 512 bits, and the embedding degree is 2.

5.1.2. Implementation of Message Broker

The construction of Broker is based on the open-source MQTT server HiveMQ Community Edition [42], which provides the SDK (HiveMQ Extension SDK) [43] that supports extension development. Through this, users can develop custom business logic to extend the functions of HiveMQ with the SDK, such as intercepting or controlling MQTT messages, integrating other services, statistics, and adding fine-grained security solutions. We use the core part of HiveMQ to implement message routing and forwarding, implements message

re-encryption in a customized extension, and re-encrypts the message payload for each subscriber based on the re-encryption key of each subscriber.

HiveMQ includes multiple types of Interceptors, which provide convenience for intercepting and modifying MQTT messages in extensions. Our customized extension is implemented through HiveMQ Interceptors. The goal of our customized extension is to re-encrypt the payload of the message according to the subscriber's re-encryption key after the message is routed, but before the message is forwarded to the subscriber. Therefore, our customized extension is implemented using the **Publish Outbound Interceptor** in HiveMQ, which allows the extension to intercept the PUBLISH message after the broker is routed, and allows different modifications to the payload corresponding to each subscriber.

5.1.3. Implementation of the Client

The implementation of the publisher and the subscriber is relatively easy, which is based on the Eclipse Paho Java Client library and the JPBC library is added to support the encryption and decryption of the published and received messages based on CPRE.

5.2. Performance Analysis

This section tests the performance of each module of our system based on the prototype system built in the previous section.

5.2.1. Overhead of Distributing Session Keys Using CPRE

When the device receives a new condition value, it will randomly generate a new session key. The key will be encrypted with the device's public key and the new condition value, and then ciphertext will be sent to the Broker. The broker re-encrypts the ciphertext for each subscriber and sends the re-encrypted ciphertext to each subscriber. Each subscriber decrypts the re-encrypted ciphertext with its private key, and obtains the session key. When the device uses CPRE to transmit an AES (Advanced Encryption Standard) [44] key (128 bits), the computational overhead of each module (device, broker, and subscriber) in our system is shown in Figure 3.

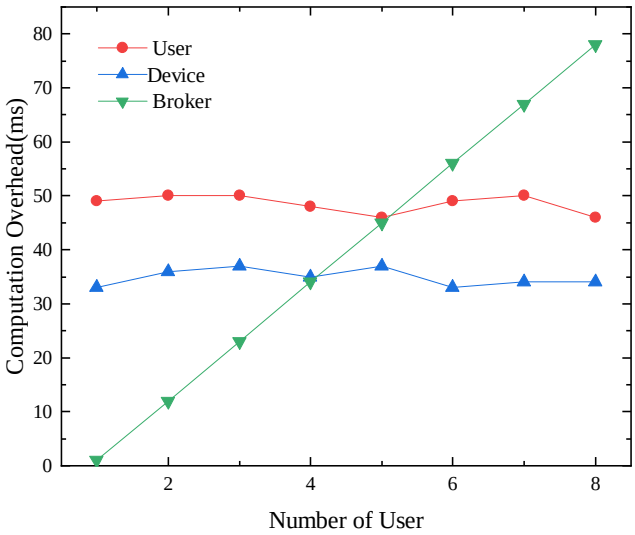

Figure 3. Computation overhead of each module to distribute session key based on CPRE.

The computation overhead of the device is approximately 46 ms. The overhead of the subscriber to is approximately 35 ms. The processing time required by the broker increases linearly with the number of users. As shown in Figure 3, for each additional user, the processing time of the Broker increases by approximately 9 ms. The Broker is generally deployed on a server or cloud platform with rich resources, which can easily cope with

this overhead increase. The computation overhead of the device and users does not change with the number of users, making it suitable for IoT devices with limited resources.

5.2.2. Overhead of Secure Communication

After the device and the subscriber have established a shared symmetric key, the communication between the device and the subscribers are encrypted by the symmetric encryption algorithm AES. We compare the computation overhead of the publisher and subscriber when they use AES for message transmission than when they transmit in plaintext.

For multiple message sizes (128 B, 512 B, 1 KB, 2 KB), Figure 4 shows that encrypted transmission between the publisher and the subscriber incurs an increased overhead of approximately 0.2–0.3 ms per message. Therefore, the use of symmetric keys for secure transmission brings only a slightly increase in computation cost.

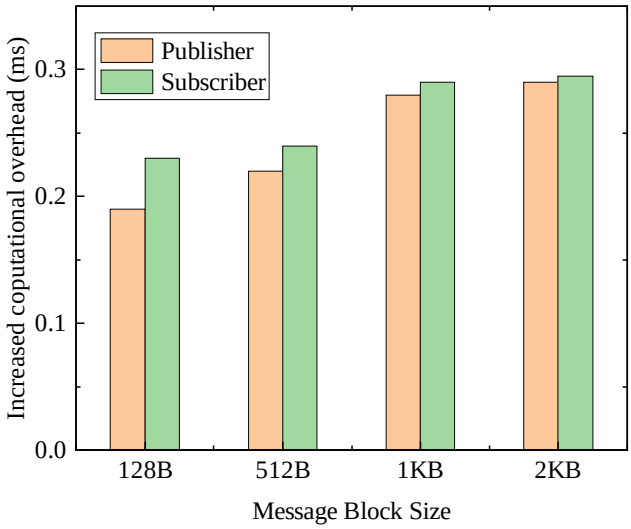

Figure 4. Increased computation overhead for encrypted transfers.

5.2.3. Comparison with Related Schemes

Table 3 compares the current end-to-end encryption schemes in IoT in terms of security, whether it supports decentralized authorization, whether it is convenient to deploy, and performance.

Table 3. Comparison of existing end-to-end encryption scheme.

Scheme	Confidentiality	Decentralization Authorization	Easy Deploy	Performance
[14]	No	No	Yes	-
[5]	Yes	No	Yes	Fast
[22]	Yes	No	No	-
[23]	Yes	No	No	-
[15]	Yes	No	Yes	Comparable to [20]
[20]	Yes	Yes	Yes	Slow (encrypt $\approx$ 42 ms, decrypt $\approx$ 62 ms)
Our scheme	Yes	Yes	Yes	Faster than [20] (encrypt $\approx$ 46 ms, decrypt $\approx$ 35 ms)

Note: - represents the performance of these schemes was not evaluated on our prototype.

- Security: The message brokers in most IoT systems are not completely trusted. In a scheme that completely relies on the message broker, the broker can obtain all the information of the user, which does not meet the confidentiality requirements.
- Support decentralized authorization: If relying on a third-party trusted key server, the authorization and revocation of device access rights must be completed through the key server, and decentralized authorization is not supported. PICADOR relies on a trusted authority to generate re-encryption keys for all users of the system and does not support decentralized authorization too.
- Easy to deploy: Reference [22] divides the functions of a broker into multiple ones, and new brokers need to be customized to meet the corresponding functions. Reference [39] needs to install special hardware, which is difficult to deploy.
- Performance: The schemes that rely on trusted brokers do not meet the confidentiality requirements, and the literature [22,23] is difficult to deploy. Therefore, these schemes are not re-implemented on our experimental platform. Ref. [5] relies on a trusted key server, the scheme uses symmetric key to establish session key, so [5] distributes symmetric keys faster than our scheme. However, in the secure communication stage, the overhead of using symmetric keys to encrypt and transmit messages is equal to our scheme. Ref. [20] implements the scheme of PICADOR, whose performance is comparable to [20]. In order to have a fair comparison with [20], we re-implement the WKD-IBE algorithm in [20] using our crypto library. In [20], the encryption algorithm takes almost 42 ms to encrypt 128 bits of data, and the decryption algorithm takes about 62 ms to decrypt and obtain the plaintext (the decryption time contains the time to generate a decryption key for the encrypted pattern and time to decrypt the ciphertext. When testing the computation overhead, we use a pattern of 20 attributes representing the URI and the last six attributes representing the time.). The encryption overhead of our scheme is comparable to that presented in [20], while the decryption cost is approximately half that presented in [20].

Above all, some existing schemes rely on the trustworthiness of the broker, and their security assumptions are strong, which cannot meet the confidentiality requirements; some schemes rely on a third-party trusted server for authorization and revocation, and do not support decentralized authorization; some solutions need a customized broker or rely on special hardware, which is inconvenient to deploy. Ref. [20] is a solution with good security and deployability. However, it relies on the more complex WKD-IBE algorithm, and its performance is lower compared to our proposed solution.

6. Conclusions

In the publish–subscribe-based IoT system, the communication between devices and users is not one-to-one direct communication, but one-to-many asynchronous communication and forwarded by the broker located in the middle. Currently, TLS is commonly employed to safeguard the security of data transmission between the device and the broker. However, the broker has the ability to access the plaintext of all messages, rendering this method ineffective in preventing the security and privacy risks posed by untrustworthy brokers to device data. We present and implement a new end-to-end encryption system based on conditional proxy re-encryption. Theoretical analysis and experimental results demonstrate that our proposed scheme is not only theoretically safe, but also highly practical, feasible, and efficient in practice. However, our system cannot efficiently revoke users, which will form the focus of our future research.

Author Contributions: Conceptualization, S.L. and L.C.; methodology, S.L. and N.K.; software, S.L.; validation, S.L., L.C. and N.K.; writing—original draft preparation, L.C.; writing—review and editing, S.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data used to support the findings of this study are included in this article.

Acknowledgments: We thank anonymous reviewers for their useful comments.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

IoT	Internet of Things
CPRE	conditional proxy re-encryption
MQTT	Message Queuing Telemetry Transport
AMQP	Advanced Message Queuing Protocol
TLS	Transport Layer Security
PRE	proxy re-encryption
CPS	Cyber–Physical Systems
WKD-IBE	identity-based encryption with wildcards
JPBC	The Java Pairing Based Cryptography Library
SDK	HiveMQ Extension SDK
AES	Advanced Encryption Standard

References

1. Banks, A.; Briggs, E.; Borgendale, K.; Gupta, R. *MQTT*, Version 5.0; Technical Report, OASIS Standard; OASIS: Burlington, MA, USA, 2019.
2. Godfrey, R.; Ingham, D.S.R. *Advanced Message Queuing Protocol (AMQP)*, Version 1.0; Technical Report, OASIS Standard; OASIS: Burlington, MA, USA, 2012.
3. Zhou, W.; Jia, Y.; Yao, Y.; Zhu, L.; Guan, L.; Mao, Y.; Liu, P.; Zhang, Y. Discovering and Understanding the Security Hazards in the Interactions between IoT Devices, Mobile Apps, and Clouds on Smart Home Platforms. In Proceedings of the 28th USENIX Security Symposium, USENIX Security 2019, Santa Clara, CA, USA, 14–16 August 2019; Heninger, N., Traynor, P., Eds.; USENIX Association: Berkeley, CA, USA, 2019; pp. 1133–1150.
4. Wilson, J.; Wahby, R.S.; Corrigan-Gibbs, H.; Boneh, D.; Levis, P.A.; Winstein, K. Trust but Verify: Auditing the Secure Internet of Things. In Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services, MobiSys'17, Niagara Falls, NY, USA, 19–23 June 2017; Choudhury, T., Ko, S.Y., Campbell, A., Ganesan, D., Eds.; ACM: New York, NY, USA, 2017; pp. 464–474. [CrossRef]
5. Dahlmanns, M.; Pennekamp, J.; Fink, I.B.; Schoolmann, B.; Wehrle, K.; Henze, M. Transparent End-to-End Security for Publish/Subscribe Communication in Cyber-Physical Systems. In Proceedings of the SAT-CPS@CODASPY 2021, Proceedings of the 2021 ACM Workshop on Secure and Trustworthy Cyber-Physical Systems, Virtual Event, 28 April 2021; Gupta, M., Abdelsalam, M., Mittal, S., Eds.; ACM: New York, NY, USA, 2021; pp. 78–87. [CrossRef]
6. Maggi, F.; Vosseler, R.; Quarta, D. *The Fragility of Industrial IoT's Data Backbone: Security and Privacy Issues in MQTT and CoAP Protocols*; Technical Report, Trend Micro Research; Trend Micro Inc.: Tokyo, Japan, 2018.
7. Huq, N.; Vosseler, R.; Swimmer, M. *Cyberattacks against Intelligent Transportation Systems*; Technical Report, Trend Micro Research; 2017.
8. EMQ. *EMQX Broker Docs*; v4.3; Technical report; EMQ: Hong Kong, China, 2021.
9. HiveMQ. *HiveMQ Documentation*; v4.7; Technical Report; HiveMQ: Landshut, Germany, 2021.
10. Solace. *Solace Cloud*; Technical Report; Solace: Ottawa, ON, Canada, 2021.
11. Alibaba. *Alibaba Cloud Iot Platform*; Technical report; Alibaba: Hangzhou, China, 2021.
12. Amazon. *Aws Iot Core*; Technical Report; Amazon: Bellevue, WA, USA, 2021.
13. Henze, M.; Matzutt, R.; Hiller, J.; Mühmer, E.; Ziegeldorf, J.H.; van der Giet, J.; Wehrle, K. Complying With Data Handling Requirements in Cloud Storage Systems. *IEEE Trans. Cloud Comput.* **2022**, *10*, 1661–1674. [CrossRef]
14. Jia, Y.; Xing, L.; Mao, Y.; Zhao, D.; Wang, X.; Zhao, S.; Zhang, Y. Burglars' IoT Paradise: Understanding and Mitigating Security Risks of General Messaging Protocols on IoT Clouds. In Proceedings of the 2020 IEEE Symposium on Security and Privacy, SP 2020, San Francisco, CA, USA, 18–21 May 2020; IEEE: Piscataway, NJ, USA, 2020; pp. 465–481. [CrossRef]
15. Borcea, C.; Gupta, A.D.; Polyakov, Y.; Rohloff, K.; Ryan, G.W. PICADOR: End-to-end encrypted Publish-Subscribe information distribution with proxy re-encryption. *Future Gener. Comput. Syst.* **2017**, *71*, 177–191. [CrossRef]

16. Blaze, M.; Bleumer, G.; Strauss, M. Divertible Protocols and Atomic Proxy Cryptography. In Proceedings of the Advances in Cryptology-EUROCRYPT'98, International Conference on the Theory and Application of Cryptographic Techniques, Espoo, Finland, 31 May–4 June 1998; Nyberg, K., Ed.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 1998; Volume 1403, pp. 127–144. [CrossRef]
17. Lee, E. Improved Security Notions for Proxy Re-Encryption to Enforce Access Control. In Proceedings of the Progress in Cryptology-LATINCRYPT 2017-5th International Conference on Cryptology and Information Security in Latin America, Havana, Cuba, 20–22 September 2017; Revised Selected Papers; Lange, T., Dunkelman, O., Eds.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2017; Volume 11368, pp. 66–85. [CrossRef]
18. Weng, J.; Deng, R.H.; Ding, X.; Chu, C.; Lai, J. Conditional proxy re-encryption secure against chosen-ciphertext attack. In Proceedings of the 2009 ACM Symposium on Information, Computer and Communications Security, ASIACCS 2009, Sydney, Australia, 10–12 March 2009; Li, W., Susilo, W., Tupakula, U.K., Safavi-Naini, R., Varadharajan, V., Eds.; ACM: New York, NY, USA, 2009; pp. 322–332. [CrossRef]
19. Qiu, J.; Hwang, G.; Lee, H. Efficient Conditional Proxy Re-encryption with Chosen-Ciphertext Security. In Proceedings of the Ninth Asia Joint Conference on Information Security, AsiaJCIS 2014, Wuhan, China, 3–5 September 2014; Computer Society; IEEE: Piscataway, NJ, USA, 2014; pp. 104–110. [CrossRef]
20. Kumar, S.; Hu, Y.; Andersen, M.P.; Popa, R.A.; Culler, D.E. JEDI: Many-to-Many End-to-End Encryption and Key Delegation for IoT. In Proceedings of the 28th USENIX Security Symposium, USENIX Security 2019, Santa Clara, CA, USA, 14–16 August 2019; Heninger, N., Traynor, P., Eds.; USENIX Association: Berkeley, CA, USA, 2019; pp. 1519–1536.
21. Abdalla, M.; Catalano, D.; Dent, A.W.; Malone-Lee, J.; Neven, G.; Smart, N.P. Identity-Based Encryption Gone Wild. In Proceedings of the Automata, Languages and Programming, 33rd International Colloquium, ICALP 2006, Venice, Italy, 10–14 July 2006; Bugliesi, M., Preneel, B., Sassone, V., Wegener, I., Eds.; Proceedings, Part II; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2006; Volume 4052, pp. 300–311. [CrossRef]
22. Belguith, S.; Cui, S.; Asghar, M.R.; Russello, G. Secure publish and subscribe systems with efficient revocation. In Proceedings of the 33rd Annual ACM Symposium on Applied Computing, SAC 2018, Pau, France, 9–13 April 2018; Haddad, H.M., Wainwright, R.L., Chbeir, R., Eds.; ACM: New York, NY, USA, 2018; pp. 388–394. [CrossRef]
23. Segarra, C.; Delgado-Gonzalo, R.; Schiavoni, V. MQT-TZ: Secure MQTT Broker for Biomedical Signal Processing on the Edge. In Proceedings of the Digital Personalized Health and Medicine-Proceedings of MIE 2020, Medical Informatics Europe, Geneva, Switzerland, 28 April–1 May 2020; Pape-Haugaard, L.B., Lovis, C., Madsen, I.C., Weber, P., Nielsen, P.H., Scott, P., Eds.; Studies in Health Technology and Informatics; IOS Press: Amsterdam, The Netherlands, 2020; Volume 270, pp. 332–336. [CrossRef]
24. Maene, P.; Götzfried, J.; de Clercq, R.; Müller, T.; Freiling, F.C.; Verbauwhede, I. Hardware-Based Trusted Computing Architectures for Isolation and Attestation. *IEEE Trans. Comput.* **2018**, *67*, 361–374. [CrossRef]
25. Sabt, M.; Achemlal, M.; Bouabdallah, A. Trusted Execution Environment: What It is, and What It is Not. In Proceedings of the 2015 IEEE TrustCom/BigDataSE/ISPA, Helsinki, Finland, 20–22 August 2015; IEEE: Piscataway, NJ, USA, 2015; Volume 1, pp. 57–64. [CrossRef]
26. Mambo, M.; Okamoto, E. Proxy Cryptosystems: Delegation of the Power to Decrypt Ciphertexts (Special Section on Cryptography and Information Security). *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.* **1997**, *80*, 54–63.
27. Weng, J.; Yang, Y.; Tang, Q.; Deng, R.H.; Bao, F. Efficient Conditional Proxy Re-encryption with Chosen-Ciphertext Security. In Proceedings of the Information Security Conference, Pafos, Cyprus, 18–20 May 2009.
28. Shao, J.; Wei, G.; Ling, Y.; Xie, M. Identity-Based Conditional Proxy Re-Encryption. In Proceedings of the 2011 IEEE International Conference on Communications (ICC), Kyoto, Japan, 5–9 June 2011; pp. 1–5.
29. Liang, K.; Liu, Z.; Tan, X.; Wong, D.S.; Tang, C. A CCA-Secure Identity-Based Conditional Proxy Re-Encryption without Random Oracles. In Proceedings of the International Conference on Information Security and Cryptology, Seoul, Republic of Korea, 28–30 November 2012.
30. He, K.; Weng, J.; Deng, R.H.; Liu, J.K. On the security of two identity-based conditional proxy re-encryption schemes. *Theor. Comput. Sci.* **2016**, *652*, 18–27. [CrossRef]
31. Fang, L.; Susilo, W.; Ge, C.; Wang, J. Chosen-ciphertext secure anonymous conditional proxy re-encryption with keyword search. *Theor. Comput. Sci.* **2012**, *462*, 39–58. [CrossRef]
32. Seo, J.W.; Yum, D.H.; Lee, P.J. Proxy-invisible CCA-secure type-based proxy re-encryption without random oracles. *Theor. Comput. Sci.* **2013**, *491*, 83–93. [CrossRef]
33. Son, J.; Kim, D.; Hussain, R.; Oh, H. Conditional proxy re-encryption for secure big data group sharing in cloud environment. In Proceedings of the 2014 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), Toronto, ON, Canada, 27 April–2 May 2014; pp. 541–546.
34. Liang, K.; Susilo, W.; Liu, J.K.; Wong, D.S. Efficient and Fully CCA Secure Conditional Proxy Re-Encryption from Hierarchical Identity-Based Encryption. *Comput. J.* **2015**, *58*, 2778–2792. [CrossRef]
35. Ge, C.; Susilo, W.; Wang, J.; Fang, L. Identity-based conditional proxy re-encryption with fine grain policy. *Comput. Stand. Interfaces* **2017**, *52*, 1–9. [CrossRef]
36. Xiong, H.; Wang, Y.; Li, W.; Chen, C. Flexible, Efficient, and Secure Access Delegation in Cloud Computing. *ACM Trans. Manag. Inf. Syst. (TMIS)* **2019**, *10*, 2. [CrossRef]

37. Paul, A.; Selvi, S.S.D.; Rangan, C.P. A Provably Secure Conditional Proxy Re-Encryption Scheme without Pairing. *J. Internet Serv. Inf. Secur.* **2019**, *11*, 1–21.
38. Katz, J.; Lindell, Y. *Introduction to Modern Cryptography*, 2nd ed.; Computer Science, Mathematics, 2014. Available online: https://api.semanticscholar.org/CorpusID:9506320 (accessed on 28 November 2023).
39. Canetti, R.; Malkin, T.; Nissim, K. Efficient Communication-Storage Tradeoffs for Multicast Encryption. In Proceedings of the Advances in Cryptology-EUROCRYPT'99, International Conference on the Theory and Application of Cryptographic Techniques, Prague, Czech Republic, 2–6 May 1999; Stern, J., Ed.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 1999; Volume 1592, pp. 459–474. [CrossRef]
40. Caro, A.D.; Iovino, V. jPBC: Java pairing based cryptography. In Proceedings of the 16th IEEE Symposium on Computers and Communications, ISCC 2011, Kerkyra, Corfu, Greece, 28 June–1 July 2011; IEEE Computer Society: Piscataway, NJ, USA, 2011; pp. 850–855. [CrossRef]
41. e Foundation. *Eclipse Paho Java Client*; Technical Report; e Foundation: Paris, France, 2021.
42. HiveMQ. *Hivemq-Community-Edition*; Technical Report; HiveMQ: Landshut, Germany, 2021.
43. HiveMQSDK. *HiveMQ Extension SDK 4.7.1 API*; Technical Report; HiveMQSDK: Hongkong, China, 2021.
44. Dirk, F. *AES*; Datenschutz und Datensicherheit, Advanced Encryption Standard (AES). 1999. Available online: https://api.semanticscholar.org/CorpusID:31476420 (accessed on 28 November 2023).