Traceable anonymous message authentication scheme for semi-trusted edge-enabled iiot

Ming-Te Chen[1*] and Yu-Che Tsai[1]

ABSTRACT

In recent years, the internet of things (IoT) and industrial factory 4.0 have become important topics in research. IoT devices and chips are now integrated into intelligent machines, allowing them to collect and upload product information to cloud services or edge servers (ESs). This has given rise to a new application called the industrial internet of things (IIoT). In the IIoT architecture, there is a publisher/subscriber model where the publisher and subscriber can communicate with each other through Edge servers as a broker between them. When a subscriber attempts to order some topic from the broker, the broker will forward these topics to the publisher. Then, the publisher could forward the related messages back to the broker and the broker sent these messages to the corresponding subscriber. During the message transmission, if the message does not employee some encryption methodology, then the message may be intercepted or modified by the attacker. I.e., the ordered message's privacy cannot be guaranteed well in this MQTT model.

On the other hand, the message source also cannot be confirmed if there is an attacker which it could modify the message source address and other related information. When the subscriber has received this message, it cannot also confirm the message's source from the broker. In order to solve above problems, there are some related methodologies to provide their solutions in this architecture. There is an approach proposed that it can delegate the encryption method to the ES server if the ES server is assumed to be semi-trusted. In fact, we thought that the ES is assumed to be semi-trusted, it still cannot be trusted fully, i.e., it also could not to perform the delegate operation on its willing even to protect the publisher's data source information either. In order to deal with above problems, we propose our methodology for the IIoT architecture. Our scheme is efficient, provides data protection for subscribed messages, and does not rely on any semi-trusted party assumptions. Moreover, our approach is implemented with an efficient MCL pairing library, making it suitable for IoT devices with constrained resources or limited computing power in the IIoT architecture.

Keywords: Internet of Things, Industrial Internet of Things, Proxy Re-Signature, Data Privacy.

I. INTRODUCTION

With the implementation of Industry 4.0, intelligent machines can now be equipped with internet of things (IoT) chips and devices, giving rise to the industrial internet of things (IIoT) architecture. This architecture integrates many network techniques which it is the cloud computing, edge computing [1], deep learning [2] with IoT sensors and controllers. In this environment, an IoT device can transmit its perceived data to either a cloud service [3], [4] or edge servers [5, 6, 7] for processing. The cloud servers can then store the final processed data in their own databases, which can be accessed by end-users through web cloud services. However, during the data transmission process, IoT devices may upload product-sensitive data without any data confidential protection, making them vulnerable to malicious network attacks that can tamper or intercept the data. Therefore, data integrity, confidentiality, and anonymity are the three critical properties that must be taken into account in the IIoT architecture. **Figure 1** depicts the publisher/subscriber model of IIoT scenario on the cloud as demonstrated in [8]. The publisher/subscriber (Pub/Sub) architecture is also used in this scenario to store, filter, and forward perceived data between the publisher (data owner) and subscriber (data receiver) nodes in communication.

In the IIoT architecture, publishers and subscribers communicate with each other through the Edge servers or cloud servers, as shown in **Figure 1** proposed by [8]. In this paper [8], it is assumed that a key distribution center (KDC) handles the key escrow problem between two sides and the ES. Subscribers can forward their order requests topic to publishers through the ES, and publishers can send the desired content according to the received topic. If there is a matching request, one of the publishers will deliver its related data to some subscribers through the ES. When the ES has received this data from one of publishers, it performs data encryption and then for warded the final result to the subscribers. However, in this situation, the data sent by publishers is not well encrypted, making it vulnerable to interception or modification by malicious attackers during transmission. This can cause

*Corresponding Author: Ming-Te Chen (E-mail: mtchen@ncut.edu.tw).
[1] Department of Computer Science and Information Technology, National Chin-Yi University of Technology, No.57, Sec.2, Zhongshan Rd., Taiping Dist., Taichung, 11311, Taiwan

significant cost damage to industrial companies, especially when the content is secret information related to their products. Additionally, subscribers cannot guarantee content protection, and if the content is modified during transmission, they cannot confirm the data integrity. Several articles have proposed methodologies to address these problems, such as [8] and [9]. However, their efficiency is still inadequate in terms of significant pairing computation, making them unsuitable for IIoT devices with constrained resources or limited computing power. In [8], the authors claimed that they provide data anonymity and is suitable for the IIoT architecture. In [9], the authors showed that their approach using the re-encryption method under the ES server is a semi-trusted party. However, we believe that these schemes' efficiency is not practical, and the ES server cannot be considered a semi-trusted party when it conspires with malicious subscribers or attackers.

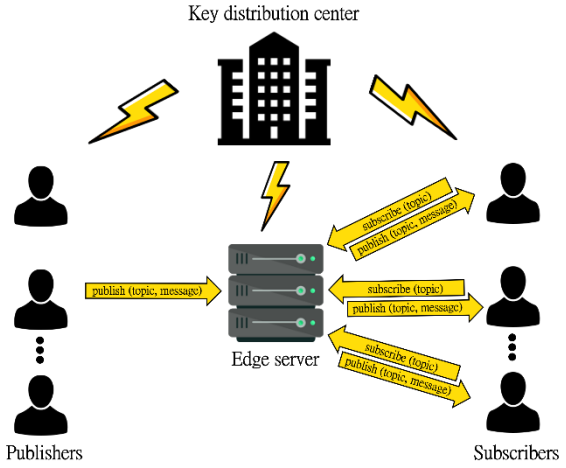

Figure 1. The publisher/subscriber architecture in [1].

We propose a revised methodology that utilizes the proxy re-signing technique [10] as our foundation. Our approach removes the semi-trusted assumption of the ES and designates the GM as the delegator to whom each publisher is a delegate. In this approach, the GM performs the proxy signature re-signing process simultaneously when delegated by some publishers, and forwards the final result to the ES server for transmission to the subscriber. Our approach also demonstrates a formal security proof in the last appendix. Our contributions are outlined below:

- Integrity: Our scheme provides member signature and re-signing functionalities, ensuring the dedicated receiver can confirm the signature and content integrity through signature verification of the received cipher-text.
- Confidentiality: We offer data encryption with time-bound involvement, ensuring that only the designated subscriber can decrypt and retrieve the original content. The subscriber can also verify the signature related to this content, determining its validity.
- Anonymity: During data transmission, only the publisher and the GM know the real data, with the ES server unaware of the content's actuality. The designated subscriber can decrypt the ciphertext successfully, obtaining the original content with the privacy protection.
- Designated subscriber: Only the designated subscriber can obtain the original censored message from the received cipher-text and performs the signature verification with the decrypted message. Attackers cannot decrypt the ciphertext without the corresponding decryption key. Under this architecture, subscribers can confirm that the message is designated for them with data privacy protection, and the assurance that the data is sent from the desired IIoT group.

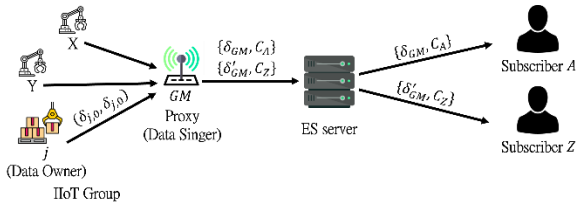

Figure 2. The Proposed Architecture in our approach.

II. RELATED WORKS AND SECURITY DEFINITIONS

2.1 Related works

In the beginning, we review some articles [8, 9] for discussion. In [9], the authors proposed an anonymous message scheme in the IIoT architecture under the assumption that the ES server is semi-trusted. They adopted re-encryption to achieve this goal, with the ES server performing the re-encryption computation upon receiving the content. However, we consider it a strong assumption to assume the ES server is semi-trusted. The ES server's role is to store or forward the encrypted content and corresponding signature to the subscribers. In terms of security objectives, we believe that the ES server may conspire with malicious subscribers or attackers, potentially leaking unencrypted content to the conspired subscribers or dishonestly performing re-encryption. This could cause significant damage to data owners or publishers in this IIoT architecture, particularly when there is a commercial relationship between publishers and subscribers.

To address this issue, we remove this assumption and instead utilize the MCL library [15], as implemented in [13], as our implementation library. We also provide a comparison of each scheme's efficiency in **Table 3**.

Our contribution is in developing a re-signing technique that ensures data protection with stand chosen cipher-text attack-II. Each IoT publisher owns data and aims to provide censored information to the subscribers based on the topic order. In our approach, the GM acts as a data signer within the same group and generates the final cipher-texts to the subscribers directly. Then, GM forwarded the final cipher-texts to the cloud servers or edge servers. In our scheme, a publisher can also generate its own signature on the censored data and forward it to the GM with delegation information. If the GM accepts the delegation, it will request the delegation key from

the corresponding publisher. This ensures data anonymity and prevents the ES server from conspiring with the subscribers.

Moreover, the re-signing operation on the publisher's signature guarantees the subscribed content's integrity. The subscriber and GM can verify the final received signature and member signature, respectively. In terms of efficiency, our scheme outperforms other schemes [8, 9] as shown in **Table 3**. Our approach has a lower computation cost, so subscribers do not need to provide more computing power or resources to access the subscribed content. This makes our approach suitable for the IIoT architecture with constrained limited resources, such as mobile devices like smartwatches or electronic paper as application agents. In conclusion, our approach offers an efficient conclusion, our approach offers an efficient methodology with a formal security proof provided in the appendix.

2.2 Security definitions

In this subsection, we define the security parameters and other cryptography functions in the followings.

Definition 1: Bi-linear Groups

We also define the bi-linear map function and bi-linear group definitions as follows:

- We assume that there are two multiplicative cyclic number groups with a large prime number p. One group is named as $\mathbb{G}_1$ and the other one is named as $\mathbb{G}_2$.
- We also assumed that there exists a group $\mathbb{G}_T$, an isomorphism $\phi: \mathbb{G}_{\neq} \to \mathbb{G}_1$ and a bilinear map $e: \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$.
- We defined that e, ϕ and the group action in $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ can be computed efficiently.

Definition 2: The q-SDH Assumption

The q-SDH problem in $(\mathbb{G}_1, \mathbb{G}_2)$, is defined as follows: given a $(q+2)$ -tuples $\left(g_1, g_2, g_2^x, g_2^{x^2}, \dots, g_2^{x^q}\right)$ as input where as above $g_1 = \phi(g_2)$, output a pair $(c, g^{\frac{1}{x+c}})$ where $c \in Z_p^*$. An algorithm $\mathcal{A}$ has advantage ϵ in solving q-SDH in $(\mathbb{G}_1, \mathbb{G}_2)$ if

$$Pr\left[\mathcal{A}\left(g_1, g_2, g_2^x, g_2^{x^2}, \dots, g_2^{x^q}\right) = \left(c, g^{\frac{1}{x+c}}\right)\right] \geq \epsilon \quad (1)$$

where the probability is over the random choice of generator $g_2 \in \mathcal{G}_{\in}$ with $g_1 = \phi(g_2)$, the random choice of $x \in Z_p^*$ and the random bits consumed by $\mathcal{A}$.

Definition 3: Existential Forgery under a Chosen Message Attack Game (EFCMA Game)

First, we must define a secure against existence unforgeability (EF) with a chosen message attacker of our proposed scheme. We also take the Lemma 3.3 of ref [19] as a building block and assume that it is a black box under this game. Next, we prepare the following experiment. First, assumed that there exists a simulator $\mathcal{A}$ and a forger $\mathcal{B}$ for this scheme. $\mathcal{A}$ will launch the experiment simulation to simulate each query sent from $\mathcal{B}$ with a subscriber as follows:

- **Setup:** Initially, $\mathcal{A}$ will start to simulate the system parameters in this phase. $\mathcal{A}$ is given the public key of (g_1, g_2, u, g_2^y, z) and it will let them as the public key of one publisher, where we assume it as j. After above system setting up completely. $\mathcal{A}$ also prepares the responding for each query made from $\mathcal{B}$ with the help of above black box in the next phase.
- **Phase 1:** In this phase, $\mathcal{B}$ can launch all kinds of queries to $\mathcal{A}$, and $\mathcal{A}$ will generate the corresponding value back to $\mathcal{B}$ as follows:
  - Hash H_1 query: When $\mathcal{B}$ asks the H_1 query on the messages (m_j) and on the subscriber j, it also generates the corresponding hash values back to $H_1(R_j||m_j)$ with the random number R_j. On the other hand, if $\mathcal{B}$ queries on the ciphertext $(C_1, C_2, C_3)$to $\mathcal{A}$, then it returns the $H_1(C_1||C_2||C_3)$ value to $\mathcal{B}$. All above queries message are preserved in the H_1-list of $\mathcal{A}$.
  - Sign-cryption query: When $\mathcal{B}$ submits a signature query on the random message $\omega_1, \dots, \omega_q \in Z_p^*$. First, $\mathcal{A}$ will input each message to check the encryption list of E_1-list. If there is an entry inside, then $\mathcal{A}$ returns the $(C_1, C_2, C_3, \sigma_j, z)$ to $\mathcal{B}$, where $j \in \{1, \dots, q\}$. If there is no sign-cryption query entry found, it also transfers to make the sign-cryption operation. It inputs the query message m_j and the target subscriber's public key pk_i, then $\mathcal{A}$ computes the encryption computation with above random value of H_1 -list to fetch (R_j, r_j) values. With these two values, $\mathcal{A}$ can compute the sign-cryption signature part σ_j on the message m_j. Finally, $\mathcal{A}$ forwards the signature tuple $(C_1, C_2, C_3, \sigma_j, z)$ to the $\mathcal{B}$.

From above game simulation, we assumed that if there exist a forger $\mathcal{B}$ with ϵ' to forge a valid sign-cryption signature in the t' period at most q_s times sign-cryption query. Then we have

$$Adv_B^{EFCMA}(\theta, t') = \varepsilon' \quad (2)$$

where the advantage is over the random values selected by $\mathcal{B}$ and the simulator $\mathcal{A}$ in the polynomial time bound t'.

Theorem 1: In this theorem, we claim that the q -SDH assumption holds if and only if there is no such attacker $\mathcal{A}$ that it can break above EFCMA game with advantage ϵ that

$$\epsilon \geq 2(\epsilon' + q_s \cdot q_e \cdot q_h/p) \approx 2\epsilon'$$
$$t \leq t' - \theta(q_s \cdot q_e \cdot q_h \cdot T), \quad (3)$$

where at most q_s times sign cryption query, at most q_e

times encryption query, at most q_h times hash query, and at most T time for an exponentiation in $\mathbb{G}_1$ and $\mathbb{G}_2$.

Definition 4: Indistinguishable selective identity chosen cipher-text attack (IND-sID-CCA) Game

Here, we continue to define the experiment for the chosen cipher-text attack in the following. We also assume that there exists a malicious attacker $\mathcal{A}$ and a simulator $\mathcal{F}$ whose goal is to use $\mathcal{A}$'s ability to guess the cipher text with non-negligible probability. Then, $\mathcal{A}$ and $\mathcal{F}$ both perform the following queries.

- **Setup:** $\mathcal{F}$ will prepare the simulation parameters params and forward them to $\mathcal{A}$. Then, it keeps the master secret key x by itself with a subscriber i where its public key pk_i.
- **Phase 1:** In this phase, $\mathcal{A}$ can still make a query, which is as follows:
  - Hash H_1 query: When $\mathcal{F}$ makes this type of query, C will find out the record to see if there is any entry saved in the H_1 hash tape list. If not, C will compute the H_1 values with the H_1 function. Then, it returns the hash value back to $\mathcal{F}$ and stores this value into the H_1 hash tape list entry.
  - Cipher-text query $\langle m_j, pk_i \rangle$: Meanwhile, $\mathcal{F}$ also makes the encryption query on (C_1, C_2, C_3, pk_i, z) of identity j, and the subscriber is i. C will check the H_1 list entry to determine if there is any encryption record that it has been asked before. If not, C computes the final cipher-text (C_1, C_2, C_3) with i's public key pk_i. Finally, C stores the encryption value into the H_1 list and returns (C_1, C_2, C_3) to $\mathcal{F}$.
  - Decryption queries $\langle C_1, C_2, C_3 \rangle$: In this type of query, C also searches the H_1 list to see if there is any entry inside. If not and it is not the challenge one, C found the original message m_j from H_1-list. Finally, it returns the result m_j back to $\mathcal{F}$ and stores the result in the H_1 list.
- **Challenge**: When $\mathcal{F}$ has performed the above query, it enters the **Challenge phase**. $\mathcal{F}$ can still make the same query in this phase. Then, $\mathcal{F}$ prepares the challenge message (M_0, M_1) and forwards this message to C. Then, C can perform the coin flip $b' \in \{0,1\}$ to choose one of them, generate the corresponding cipher text $C^{*,b'}_{pk_i^*} = (C_1^{*,b'}, C_2^{*,b'}, C_3^{*,b'}, pk_i^*)$ and let $C^{*,b'}_{pk_i^*}$ be the result of $Encrypt(params, pk_i^*, m_j^{b'})$ with the i's public key pk_i^* and the chosen message M'_b. Finally, it forwards $C^{*,b'}_{pk_i^*}$ to $\mathcal{F}$. Here, the only restriction is that $\mathcal{F}$ cannot ask the decryption query on the challenge cipher-text $C^{*,b'}_{pk_i^*}$ and the private key query of pk_i^* or the prefix key of pk_i^*.
- **Phase 2**: In this phase, $\mathcal{F}$ can still issue other queries with some constrained conditions as follows:
  - Private key query on pk_i^* : At this time, $\mathcal{F}$ can ask only the private key query on pk_i^*, where $i \neq pk_i^*$ or pk_i is not the prefix of pk_i^*.
  - Decryption query on (C_{pk_i}, C^*), where $C_{pk_i} \neq C^{*,b'}_{pk_i^*}$ for pk_i^* or any prefix of pk_i^*: We can see that $\mathcal{F}$ can also make other queries same as in **Phase 1**.
- Finally, $\mathcal{F}$ outputs its own guess result on M_b, where $b \in \{0,1\}$. She/he wins the above experiment if $b' = b$ on the challenge cipher-text $C^{*,b'}_{pk_i^*}$ of identity pk_i^*. Then, we can claim that the attacker $\mathcal{F}$'s advantage is

$$Adv_{\mathcal{F},C}^{IND-sID-CCAII}\left(C^{*,b'}_{pk_i^*}, \theta, t', b, b'\right) =$$

$$|Pr[\mathcal{F} \to b | b' = b] - 1/2|, \quad (4)$$

where the probability is over the random bits chosen by C and the attacker $\mathcal{F}$ in the polynomial time bound t'.

Theorem 2: In this theorem, we define that our scheme satisfies above IND-sID-CCA secure if and only if there exists no such adversary that it can guess the cipher text successfully with non-negligible probability greater than 1/2 in the experimental time bound t' on above **Indistinguishable selective identity chosen cipher-text attack II (IND-sID-CCA) Game**.

Theorem 3: In this theorem, we also assume that (t, ϵ) the q-SDH assumption holds in $\mathbb{G}_1$, where $\mathbb{G}_1$ is a bilinear group of prime orders p. We can define that if and only if there exist no such adversaries $\mathcal{A}$ and $\mathcal{F}$ that can break our approach with a non-negligible advantage larger than $\frac{\varepsilon}{2} + (q_s \cdot q_e \cdot q_h)/p + \frac{1}{2}\varepsilon'$ in a polynomial time bound $t' < t + \theta(q_s \cdot q_e \cdot q_h \cdot T)$ with at most qs times of signature queries, q_H times of hash queries, and q_e times of encryption queries, then we can claim that our approach satisfies the $(t'', \frac{\varepsilon}{2} + (q_s \cdot q_e \cdot q_h)/p + \frac{1}{2}\epsilon')$-unforgeable criterion and withstands the Ind-sID-CCA cipher-text attack in the IIoT architecture.

III. THE PROPOSED SCHEME

In this section, we take [18] as the building block reference and give the symbol definition list in the followings.

3.1 Symbol Definitions

Table 1. Definition table of the proposed scheme

Symbols	Description
p	A large prime order number that forms a finite primes group.
q	A group order number that contains at most q IoT devices, where each device can be a publisher in the same group.
t	A group order number that contains at most t devices, where each device can be a subscriber in the same group.
$\mathbb{G}_1$	A bilinear group where its order is less than p
$\mathbb{G}_2$	A bilinear group where its order is less than p
$\mathbb{G}_T$	A bilinear group where its order is less than p
ϕ	An isomorphism mapping operation, where $\phi: \mathbb{G}_2 \to \mathbb{G}_1$
e	A bilinear mapping operation, where $e: \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$
j	An IoT publisher that performs data collection, encryption operations and signature operations in a product group, with order q where $j \in \{0, \ldots, q-1\}$.
ID_j	An IoT publisher's identity information, where $j \in \{0, \ldots, q-1\}$.
m_j	The content that one of the IoT devices attempts to publish with other subscribers outside the IoT group, where $j \in \{0, \ldots, q-1\}$ and $m \in Z_p^*$
KDC	A key distribution center which it could generate key pairs to each device of the IoT group.
pk_j	A public key of an IoT device j, where j belongs to the same IoT group, i.e., $j \in \{0, \ldots, q-1\}$
E_i	A secure encryption function that is adopted by input an original plain-text m_j with one subscriber's public key pk_i, where i is one of the subscribers with its public key pk_i,where $i \in \{0, \ldots, t-1\}$
D_i	A secure decryption function that is adopted by input a cipher-text C_j with one subscriber's secret key sk_i, where i is one of the subscribers with its public key pk_i,where $i \in \{0, \ldots, t-1\}$
C_i	A cipher-text that is generated by input an original plain-text m_j. into a secure encryption function E_i.
$\mathrm{H}_1(\cdot)$	A secure hash function that maps the points to a p bits output value, where $\{0,1\}^* \to Z_p^*$.
$\mathrm{H}_2(\cdot)$	A secure hash function that maps the points to a p bits output value, where $\{0,1\}^* \to \{0,1\}^*$.

3.2 Setup phase

In the beginning, there is an IoT device group whose its members is at most q devices in the production line. Each device can be a publisher whose goal is to collect the industrial machine's production information and also it forwards these data for the subscriber outside this group. We also assumed that there is a group manager (GM), which is equipped with more computing resources for resign-cryption operations. The GM performs the following steps:

- We assumed that there exists an IoT device where is j and its setups a random generator $g_2 \in \mathbb{G}_2$ and let $g_1 = \phi(g_2)$. Then it generates its own master secret key $(x_j, y_j) \in Z_p^*$ and the corresponding public key $u_j \leftarrow g_2^{x_j} \in \mathbb{G}_2$, $v_j \leftarrow g_2^{y_j} \in \mathbb{G}_2$, and the bilinear commit $z = e(g_1, g_2)$. After generating the above parameters, it publishes the public key tuple (g_1, g_2, u_j, v_j, z) to the group members and GM.
- Meanwhile, each IoT device is also preparing to collect the subscribed information for their dedicated subscribers. Here, we assume that there exists a subscriber i that it has subscribed content from above IoT device group whose its corresponding publisher is assumed as j. Then, the GM will allocate i's order information to the dedicated publisher j and transfer i's public key pk_i and subscribed time bound T_i to j.

3.3 Sign-cryption phase

In this phase, the publisher j prepares the desired content m_j for the subscriber i with the time period T_i and performs the following computations:

- First, j randomly chooses a random number $(R_j, r_j) \in Z_p^*$ and computes its signature σ_j with the content message m_j, and the time period T_i as follows:

$$C_1 = E_{pk_i}(R_j || H_1(R_j || m_j))$$

$$C_2 = m_j \oplus H_2(R_j)$$

$$C_3 = H_1(m_j \oplus r_j \,||pk_i) \quad (5)$$

$$\sigma_j = g_1^{\frac{1}{(x_j + H_1(C_1||C_2||C_3) + y_j \cdot r_j)}} \in \mathbb{G}_1$$

$$z = e(g_1, g_2)$$

- Above $\frac{1}{(x_j + H_1(C_1||C_2||C_3) + y_j \cdot r_j)}$ is computed from module the large prime p. If $x_j + H_1(C_1||C_2||C_3) + y_j \cdot r_j = 0$, then we repeat choice another random number r_j to computer a valid signature. Hence, we have the final signature tuple $(\sigma_j, r_j, C_1, C_2, C_3, H_1(C_1||C_2||C_3), z, pk_i)$.
- After j has generated this signature $(\sigma_j, r_j, C_1, C_2, C_3, H_1(C_1||C_2||C_3), z, pk_i)$ completely, it forwarded it to the GM and GM performs the following verification.

$$\left(\sigma_j, u_j \cdot g^{H_1(C_1||C_2||C_3)} \cdot v_j^{r_j}\right) =^? e(g_1, g_2) \quad (6)$$

- After GM performed signature verification of above signature tuple sent from j, it forwards the tuple to the subscriber i if above signature is valid.

3.4 Decryption phase

When the publisher i has received this encrypted message from GM, she/he performs decryption on the received cipher texts and the signature verification as follows:

$$\left(\sigma_j, u_j \cdot g^{H_1(C_1||C_2||C_3)} \cdot v_j^{r_j}\right) =^? e(g_1, g_2)$$

$$D(C_1) =^? D(E_{pk_i}(R_j || H_1(R_j || m_j)))$$

$$= (R_j || H_1(R_j || m_j))$$

$$C_2 \oplus H_2(R_j) =^? m_j$$

$$C_3 =^? H_1(m_j \oplus r_j||pk_i)$$

$$H_1(R_j||m_j) =^? H_1(R_j||m_j) \quad (7)$$

3.5 Tracing phase

When the GM has discovered that there is some content m_{j^*} provided by some anonymous whistle-blowers. It can search its own record list to find out if there is any record existed or not by performing the following equation.

$$C_3^* =^? H_1(m_{j^*} \oplus r_{j^*}||pk_{i^*}) \quad (8)$$

If there is a record matched in the list, the GM can find out the related subscriber's public key pk_i^*, where $i^* \in \{1,,,t-1\}$. If above C_3^* is found, the GM also found the related signature assumed it as σ_j^* in its own database. Then, it could perform the signature verification on this σ_j^*. If the result is valid, then it can find the related cipher-texts and signature on the message m_{j^*} assumed it as $(\sigma_{j^*}, C_1^*, C_2^*, C_3^*, z)$. The GM could upload the above tuple to the judge and ask the judge to help find out who is the real traitor. The judge can ask the subscriber i with the public key pk_{i^*} to decrypt the C_1^* and C_2^* to see if there is a match with the distributed message m_{j^*}.

IV. FUNCTIONAL AND SECURITY ANALYSIS

In this section, we offer an efficiency comparison with other schemes and provide a functional analysis in the followings.

4.1 Undeniable property

In this proposed scheme, only the subscriber z can obtain the final subscribed content M and the corresponding signature from the GM. She/he can verify signature σ_j with encrypted message content (C_1, C_2, C_3). from the equation 7.

4.2 Designated Cipher-text

In our methodology, the attacker can not know the real message from the intercepted cipher-text $(C_1, C_2, C_3, \sigma_j, z)$ without the subscriber i's private key. Only the subscriber i can correctly decrypt above cipher-text (C_1, C_2, C_3) after verifying the signature σ_j with z.

4.3 Efficiency comparison

In this approach, our scheme can be implemented efficiently by algorithms such as the elliptic curve. Initially, we take the elliptic curve (ECC) algorithm as our building block and set up the key length to be 256 bits long. At this time, we adopt the Barreto-Naehrig [13](BN) curve as our implementation algorithm. Its security level is comparable to that of the RSA algorithm with 2048 bits key length.

After setting up, we define each operation in our methodology. First, we let ec_{mul} be the elliptic curve point multiplication operation, ec_{add} be the point addition operation, ec_p be the pairing operation, and E_{pki} be the public key encryption under the public key pk_i, where $i \in \{0,\dots,q\}$. H is the hash operation, and mu is the multiplication operation in a big prime modulo in [13] and [14]. On the other hand, we also take the MCL [15] library and four raspberry pi 4 development boards which each of them is an ARMv8 Cortex-A72 processor as one of the publishers IoT devices for implementation simulation. We also select one of them as the group manager to perform the group signing operation and transfers packet to the subscribers through the help of Edge server. We also take [13]'s result as our approach's performance time evaluation reference. From **Table V** of [13], we define that the computation time cost of ec_{add} is approximately 1 ms, that of ec_{mul} is approximately 0.75 ms, that of ec_{Inv} is approximately 1 ms and that of e is approximately 3.88 ms. Then, we summarize the efficiency of our approach and compare it with those of other schemes in **Table 3**. From **Table 3**, we discover that our scheme's performance is better than that of others. In **Figure 4**, our approach's cost is the lowest compared to those of other schemes [8], [9] if the IoT subscriber number is up to 500 nodes. From the **Figure 5**, we also can see that our approach is about 3 times faster than [9] in the publisher's side during the transferring packet size in the mega-bits message length. In addition, our approach can speed up to 6 times faster than [9] in the subscriber's side of the **Figure 6** in the communication overhead. In **Figure 4**, our approach is about 3.78 times faster than [9] under 5000 nodes simulation.

Table 2. Testbed of the proposed scheme

Devices	Description
Publishers	3 Raspberry pi-4 Development boards
Edge Server	1 PC (With Intel i7 CPU-3Ghz 4GB Memory)
Subscribers	1 laptop (With Intel i5 CPU-2.44Ghz 4GB Memory)

Table 3. Performance cost comparison of other three schemes

schemes	publisher	edge server	subscriber
Esposito et al. [8]	12EXP+3Pa	0	12EXP+5Pa
Cui et al. [9]	13EXP+3Pa	1EXP	13EXP+5Pa
mTLS [16]	12EXP+3Pa	0	12EXP+5Pa
Our scheme	5EXP+4Mu	0	3EXP+1Mu

q: A group order that contains at most q IoT devices
mTLS: An anonymous TLS (mTLS) scheme implemented from [9]
EXP: The exponential modular operation
Mu: The multiplication operation
Pa: The pairing operation

V. CONCLUSIONS

In this section, we show that our approach is an efficient and practical message re-signature scheme for the IIoT publisher/subscriber model. Our approach can offer data protection on the subscribed topic content and our proposed scheme can provide the re-signing property for the group manager or the powerful trusted parties to regenerate the signature with data anonymity protection to the desired subscriber. On the other hand, the subscriber's content can be traced with a re-signature, which can be used to revoke subscribers who distribute its content to other subscribers without continuing order. This tractability is beyond this research scope. In the future, we will continue to design a hierarchical content designated signature scheme for the IIoT architecture that can allow different level subscribers to order their specified designated content under one cipher-text received in the subscriber side. At the same time, we will also provide a security discussion in future works.

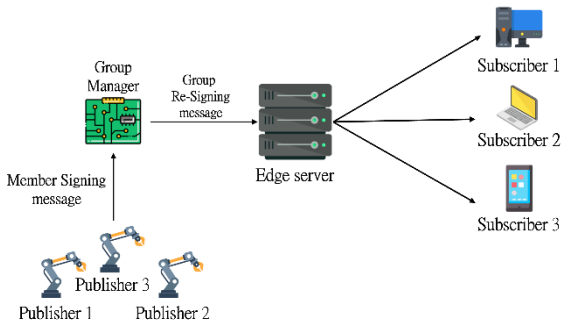

Figure 3. Our Approach Architecture

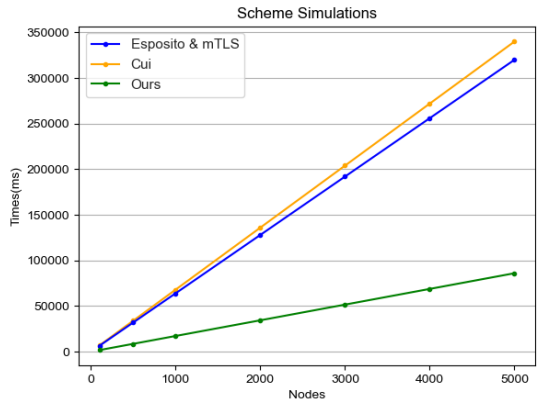

Figure 4. Efficiency comparison with other schemes

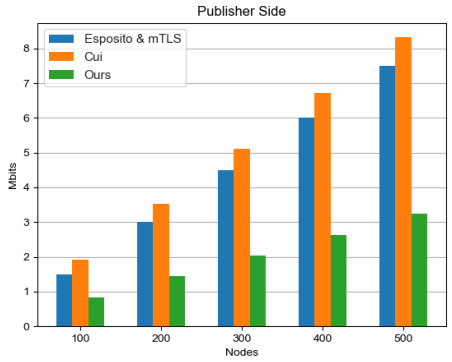

Figure 5. Communication Overhead on the Publisher Side

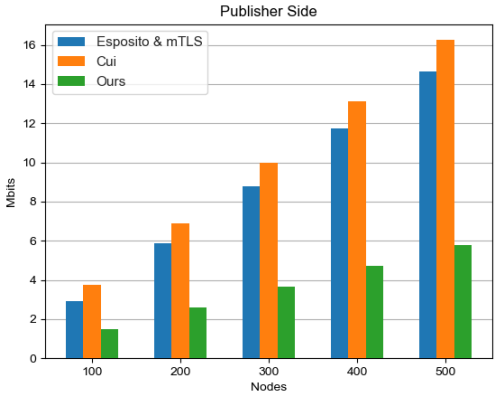

Figure 6. Communication Overhead on the Subscriber Side

ACKNOWLEDGMENTS

The authors would like to thank the anonymous referees for their detailed comments and constructive suggestions on this approach.

APPENDIX

PROOF OF THE PROPOSED SCHEME

Proof 1: **Proof of Theorem 3**

In this section, we prove Theorem 3 in two ways: the proof of Theorem 1 and the proof of Theorem 2.

First, we also assume that there exist some attackers to break the one more unforgeability of our proposed scheme. We give two types of attackers: C, whose goal is to break the $l-wBDHI^*$ problem in $\mathbb{G}$ of our approach, and $\mathcal{A}$, whose ability is to forge a valid signature. Then, C will perform the following phase to generate system parameters and simulate the environment to let $\mathcal{A}$ launch its attacks inside. C also selects two generators $(g, h) \in \mathbb{G}$ from the $l-wBDHI^*$ problem and enters the next phase.

- **Setup phase:** After entering this phase, C will choose its own private key $s \in Z_p^*$ and let s be the group master secret key $gmsk$. Then, it also produces the corresponding group public key gpk and lets $(g, h, g^s, g^t, y_1, \ldots, y_q)$ be gpk. When the above parameters are already set, C forwards these parameters to $\mathcal{A}$ and starts to simulate the environment of our approach in the following.

- **Phase 1**: At this time, C will allow $\mathcal{A}$ to ask some types of queries, such as the following.

  - Hash H_1 query: When $\mathcal{A}$ inputs (T_z, δ'_{GM}) for the target subscriber z' with time bound T_z to make the H_1 query, C will check the H_1 list to see if there is any entry inside. If not, then it will input them into the hash function and generate the corresponding hashed values back to $\mathcal{A}$.

  - Cipher-text query $\langle M, T_z, \delta'_{GM}, N_{GM}, j, pk_z \rangle$: If $\mathcal{A}$ sends this type of query on the above messages to C, C also searches the H1 list to determine if there is any entry of (C_{pk_z}, C^*) inside. If not, C computes (C_{pk_z}, C^*) with z's public key and finally sends it back to $\mathcal{A}$.

  - Signature query: When $\mathcal{A}$ forwards this type of query on the IoT device ID_j, C checks the H_1 list to determine the results. If not, C computes δ_{GM} and returns it back to $\mathcal{A}$.

  - Decryption queries $\langle C_{pk_z}, C_z, j \rangle$: $\mathcal{A}$ can also launch this type of query on the cipher text (C_{pk_z}, C_z, j). At this time, C will prepare to decrypt these cipher texts with the subscriber z's private key.

  - Re-signing key query: At this time, if $\mathcal{A}$ lunches this query on the publisher j, C finds the entry from the H1 list and returns the corresponding re-signing key back to $\mathcal{A}$ if there exists an entry inside. If not, it performs the following steps. C generates the final re-signing key with its own private key s and outputs $(\frac{r_j \cdot s \cdot R_{GM}}{d_j}, h^{R_{GM}})$ to $\mathcal{A}$. Then, C stores this re-signing key into the H_1 list.

  - Group manager signature query: In this query, we assume that $\mathcal{A}$ may launch this query to C with the publisher j's identity ID_j. C can generate the final group signature from j's re-signing key searched from the H_1 list. Finally, C produces the group signature $(\delta_{GM}, C_{pk_z}, C^*, \prod_{t=1}^{q} l_t, h^k, gpk, g, h)$ and forwards it to $\mathcal{A}$.

- **Challenge phase**: After the above queries are made, $\mathcal{A}$ enters this phase and can still make the same queries as above to C. Its only goal is to forge the l+1-th group manager signature after she/he has made l times of group manager signatures to C. If and only if $\mathcal{A}$ can forge the l+1-th group manager signature from our above experiment with a non-negligible probability ε in a polynomial time bound t, then C can use $\mathcal{A}$'s ability to forge the l+1-th group manager signature called δ_{l+1}. Hence, we have

$$Adv_{C,\mathcal{A}}^{One-Unf}(\theta, t)$$

$$= |Pr[\mathcal{A}(\theta, t) \rightarrow (\delta_{l+1})| \, Verf(\delta_{l+1}) = 1]|$$

$$\leq Pr[\mathcal{A}(\theta, t) \rightarrow (\delta_{l+1})]$$

$$\leq \frac{\varepsilon}{2^l \cdot q_s \cdot q_h \cdot q_D \cdot q_{Re}}, \quad (.1)$$

with at most q_s times of IoT member signature queries, q_h times of H_1 and H_2 hash queries, q_D times of cipher-text decryption queries, and q_{Re} times of re-signing key queries in a polynomial time bound t and with probability ε.

Proof 2: **Proof of Theorem 2**

In this subsection, we continue to prove our Theorem 2.

We still assume that there exist some attackers, one of which is the simulator C, which attempts to solve the $l - wBDHI^*$ problem in $\mathbb{G}$, and the other of which is $\mathcal{F}$, which attempts to guess a subscriber z's cipher-text successfully with non-negligible probability. C simulates the following experiment with attacker $\mathcal{F}$.

- **Setup phase**: In this phase, C will choose some random numbers from the $l - wBDHI^*$ problem and set them as one part of the group public key. First, C selects $(y_1, \ldots, y_q)$ from the $l - wBDHI^*$ problem and also chooses a random number $s \in Z_p^*$ as the group master key gmsk. Finally, it computes the corresponding group public key $gpk = g^s$ and forwards $(gpk, y_1, \ldots, y_q, g, h)$ to $\mathcal{F}$. Here, C also simulates the target subscriber is z and its public key is pk_z. After the above parameters are completely set up, C starts to simulate the following phase.

- **Phase 1**: C simulates phase 1 and queries which $\mathcal{F}$ may also ask C in the following.

  - Signature query: When $\mathcal{F}$ forwards this type of query on the IoT device ID_j, C checks the H_1 list to determine the results. If not, C computes δ_{GM} and returns it back to $\mathcal{F}$.

  - Hash H_1 query: In this query, $\mathcal{F}$ can send this request to C. C will search the H_1 list to see if there is any entry inside. If yes, it returns the found hash values to $\mathcal{A}$. Otherwise, C will

accept $\mathcal{F}$'s request and return the hash values to $\mathcal{F}$.

- Cipher-text query $\langle M, T_z, \delta'_{GM}, N_{GM}, j, pk_z \rangle$: If $\mathcal{F}$ has sent this type of query with those parameters, C adopts z's public key and makes the cipher-texts (C_{pk_z}, C^*). Finally, C forwards them back to $\mathcal{F}$

- Decryption queries $\langle C_{pk_z}, C^*, j \rangle$: If $\mathcal{F}$ sends this type of query, C will also check to see if there is any decryption result in the H_1 list and j is equal to z or not. If not, decryption is performed on (C_{pk_z}, C^*) with j's decryption key. Otherwise, C also restores the decryption result to the H_1 list, where it contains the z's plain-text record.

- Re-signing key query: In this type of query, $\mathcal{F}$ can also launch this query to C. C also finds the related records and forwards them back if found. If not, C will input her/his master secret key to perform the re-signing key protocol to generate the final re-signing key $(\frac{r_j \cdot s \cdot R_{GM}}{d_j}, h^{R_{GM}})$. When C has obtained this re-signing key, it sends it back to $\mathcal{F}$.

- Group manager signature query: In this query, $\mathcal{F}$ can ask C to perform the group signature with the input δ_j and the message M. At this time, C generates the final group signature $(\delta_{GM}, C_{pk_z}, C^*, \prod_{t=1}^{q} l_t, h^k, gpk, g, h)$ and forwards it back to $\mathcal{F}$.

- **Challenge phase**: After the above queries have been asked by $\mathcal{F}$, $\mathcal{F}$ enters the next phase. In this challenge phase, $\mathcal{F}$ will prepare a target message pair (M_0, M_1) and the target subscriber ρ and forward them to C. C will start coin flip b and choose M_b to be the challenge cipher-texts $(C''_{pk_{\rho,b}}, C'')$, where $b \in \{0,1\}$. C transmits the challenge cipher-texts $(C''_{pk_{\rho,b}}, C'')$ back to $\mathcal{F}$. Here, $\mathcal{F}$ can still also make the queries, but there is only a restriction that $\mathcal{F}$ does not allow to ask the decryption query on the challenge cipher-texts $(C''_{pk_{\rho,b}}, C'')$ or the decryption key of subscriber ρ. If and only if $\mathcal{F}$ outputs the guess result b' on its coin flip with non-negligible probability ε' in a polynomial time bound t', then we have

$$Adv_{C,\mathcal{F}}^{Ind-sID-CCAII}(\theta, t')$$

$$= |\Pr[\mathcal{F}(\theta, t', C''_{pk_{\rho,b}}, C'') \rightarrow (M_b)|\Pr[b' == b]]|$$

$$\leq |(\Pr[\mathcal{F}(\theta, t', C''_{pk_{\rho,b}}, C'') \rightarrow (M_b)]) * (1 - \Pr[b' \neq b])|$$

$$\leq \frac{1}{2}(\Pr[\mathcal{F}(\theta, t', C''_{pk_{\rho,b}}, C'') \rightarrow (M_{b'})]) \leq \frac{1}{2}\varepsilon'.$$

From the above two lemmas, we can conclude that the total advantage is

$$Adv_{C,\mathcal{A},\mathcal{F}}^{One-Unf,Ind-sID-CCAII}(\theta, t'')$$

$$\leq (Adv_{C,\mathcal{A}}^{One-Unf}(\theta, t)$$

$$+Adv_{C,\mathcal{F}}^{Ind-sID-CCAII}(\theta, t')) \quad (.3)$$

$$\leq \left(\frac{\varepsilon}{2^l \cdot q_s \cdot q_h \cdot q_D \cdot q_{Re}} + \frac{1}{2}\varepsilon'\right).$$

ACKNOWLEDGMENTS

The authors would like to thank the anonymous referees for their detailed comments and constructive suggestions on this approach.

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

REFERENCES

[1] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges", IEEE Internet of Things Journal, vol. 3, no.5, pp.637–646, 2016.

[2] G. Shah and A. Tiwari, "Anomaly detection in iiot: A case study using machine learning", in Proceedings of the ACM India Joint International Conference on Data Science and Management of Data, ser. CoDS-COMAD'18, p.295-300, 2018.

[3] J.S. Fu, Y. Liu, H.C. Chao, B. K. Bhargava, and Z.J. Zhang, "Secure data storage and searching for industrial iot by integrating fog computing and cloud computing", IEEE Transactions on Industrial Informatics, vol. 14, no.10, pp.4519–4528, 2018.

[4] C. Stergiou, K. E. Psannis, B.G. Kim, and B. Gupta, "Secure integration of iot and cloud computing", Future Generation Computer Systems, vol.78, pp.964–975, 2018.

[5] Q. Zhang, Q. Zhang, W. Shi, and H. Zhong, "Distributed collaborative execution on the edges and its application to amber alerts", IEEE Internet of Things Journal, vol. 5, no.5, pp.3580–3593, 2018.

[6] N.N. Dao, Y. Lee, S. Cho, E. Kim, K.S. Chung, and C. Keum, "Multi-tier multi-access edge computing: The role for the fourth industrial revolution", in 2017

International Conference on Information and Communication Technology Convergence (ICTC), pp.1280–1282, 2017.

[7] Q. Zhang, Q. Zhang, W. Shi, and H. Zhong, "Firework: Data processing and sharing for hybrid cloud-edge analytics", IEEE Transactions on Parallel and Distributed Systems, vol. 29, no.9, pp.2004–2017, 2018.

[8] C. Esposito, A. Castiglione, F. Palmieri, and A. D. Santis, "Integrity for an event notification within the industrial internet of things by using group signatures", IEEE Trans. Industr. Inform., vol. 14, no. 8, pp. 3669-3678, 2018.

[9] J. Cui, F. Wang, Q. Zhang, Y. Xu and H. Zhong, "Anonymous Message Authentication Scheme for Semitrusted Edge-Enabled IIoT", in IEEE Transactions on Industrial Electronics, vol. 68, no. 12, pp. 12921-12929, Dec. 2021

[10] G. Ateniese and S. Hohenberger, "Proxy re-signatures: New definitions, algorithms, and applications", in Proc. CCS2005, New York, USA, pp. 310-319, 2005.

[11] D. Boneh, X. Boyen, and E.-J. Goh, "Hierarchical identity based encryption with constant size ciphertext", in Proc. EUROCRYPT'05, Berlin, Heidelberg: Springer-Verlag, pp.440-456, 2005.

[12] Z. Li, J. Higgins, and M. Clement, "Performance of finite field arithmetic in an elliptic curve cryptosystem", in Proc. MASCOTS 2001, Cincinnati, Ohio, USA, pp.249-256, 2001.

[13] J. Hajny, P. Dzurenda, S. Ricci, L. Malina, and K. Vrba, "Performance analysis of pairing-based elliptic curve cryptography on constrained devices", in Proc. ICUMT 2018, Moscow, Russia, pp.1-5, 2018.

[14] P. S. L. M. Barreto and M. Naehrig, "Pairing-friendly elliptic curves of prime order", in Proc. SAC 2005, Kingston, ON, Canada, pp.319-331, 2006.

[15] M. S. Herumi, Herumi/mcl: A portable and fast pairing-based cryptography library, https://github.com/herumi/mcl, accessed 2023.

[16] RFC5246: 'T. Dierks and E. Rescorla, The transport layer security (tls) protocol version 1.2.', 2008.

[17] B. Lynn, "The pairing-based cryptography library", 2013.

[18] M. T. Chen, "A Secure Group Data Encryption Scheme in Intelligent Manufacturing System for IIoT", International Journal of Software Innovation, Volume 10, Issue 1, Article 138. (E-SCI), 2022.

[19] D. Boyen and X. Boyen, "Short Signatures Without Random Oracles", in International conference on the theory and applications of cryptographic techniques. Berlin, Heidelberg: Springer Berlin Heidelberg, pp.56-73, 2004.

Ming-Te Chen was born in Tainan, Taiwan on August 2, 1980. He received his M.S. degree in computer science and information engineering from National Sun Yat-sen University, Taiwan, in 2005, and a Ph.D. degree in computer science and information engineering from National Sun Yat-sen University in 2012. In 2018, he joined the faculty of the Department of Computer Science and Information Engineering, National Chin-Yi University Technology, Taichung, Taiwan. His current research interests include information security, applied cryptographic protocols, digital signatures, IoT security, and electronic commerce.

Yu-Che Tsai graduated with a bachelor degree in Department of Computer Science and Information Engineering from the National Chin-Yi University of Technology, Taiwan, R.O.C. He is currently working toward the master's degree in Department of Computer Science and Information Engineering at National Chin-Yi University of Technology, Taiwan, R.O.C. His field of research include information security, digital signature, Blockchain.