


Article

# A Practical and Efficient Node Blind SignCryption Scheme for the IoT Device Network

Ming-Te Chen <sup>†</sup>  and Hsuan-Chao Huang <sup>\*,†</sup>

Department of Computer Science and Information Engineering, National Chin-Yi University of Technology, Taichung 41170, Taiwan; mtchen@ncut.edu.tw

\* Correspondence: sc100@ncut.edu.tw; Tel.: +886-4-23924505 (ext. 8775)

† These authors contributed equally to this work.

**Abstract:** In recent years, Internet of Things (IoT for short) research has become one of the top ten most popular research topics. IoT devices also embed many sensing chips for detecting physical signals from the outside environment. In the wireless sensing network (WSN for short), a human can wear several IoT devices around her/his body such as a smart watch, smart band, smart glasses, etc. These IoT devices can collect analog environment data around the user's body and store these data into memory after data processing. Thus far, we have discovered that some IoT devices have resource limitations such as power shortages or insufficient memory for data computation and preservation. An IoT device such as a smart band attempts to upload a user's body information to the cloud server by adopting the public-key crypto-system to generate the corresponding cipher-text and related signature for concrete data security; in this situation, the computation time increases linearly and the device can run out of memory, which is inconvenient for users. For this reason, we consider that, if the smart IoT device can perform encryption and signature simultaneously, it can save significant resources for the execution of other applications. As a result, our approach is to design an efficient, practical, and lightweight, blind sign-cryption (SC for short) scheme for IoT device usage. Not only can our methodology offer the sensed data privacy protection efficiently, but it is also fit for the above application scenario with limited resource conditions such as battery shortage or less memory space in the IoT device network.

**Keywords:** sign-cryption; unsign-cryption; cryptography module; IoT device



**Citation:** Chen, M.-T.; Huang, H.-C. A Practical and Efficient Node Blind SignCryption Scheme for IoT Device Network. *Appl. Sci.* **2022**, *12*, 278. <https://doi.org/10.3390/app12010278>

Academic Editor: Gianluca Lax

Received: 8 November 2021

Accepted: 21 December 2021

Published: 28 December 2021

**Publisher's Note:** MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



**Copyright:** © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

## 1. Introduction

In recent years, Internet of Things (IoT for short) devices has widely applied in our daily life. From the life of human beings to industry 4.0, there are many common machines composed of several IoT devices such as the air conditioner, electronic vehicle, mobile phone, etc. These devices can collect physical signal data and transfer these data to a powerful gateway device of the IoT network through the Internet in a digital manner. When the gateway has received the sensed data from a sender node, it preserves these records in a database or cloud storage service. However, such IoT devices have limitations compared with a general gateway server, such as fewer memory space or limited computing power. This situation usually occurs in the communications between nodes of wireless sensing network (WSN for short) and IoT networks. Once an IoT device has collected physical data from a human body, it then must forward these data to the powerful gateway that can preserve the final result data into a database and perform other cryptography operations. From the above scenario, we discover that, if any IoT devices attempt to perform a heavy encryption/decryption computation such as modular exponentiation over a large prime number in a public key algorithm, then they must perform a signature operation later for concrete security protection and authentication on these sensed data. This will lead to fast power consumption and free-memory usage of these nodes.

To solve the above situation, we adopt the sign-cryption approach to let a sensing node perform the lightweight sign-cryption operation and generate the final cipher-text with its own signature simultaneously on the powerful server side. When the gateway server has received this cipher-text from a sensor node, it can decrypt this cipher-text first, then perform the validation of this plain-text with the inside signature's help for data authentication.

We consider the following situation of an IoT device called  $DS_i$  that attempts to transfer sensed data to a receiver called  $\mathcal{R}$ , where  $i = 1 \sim l$  and  $l$  is the total number of all the sensor nodes. To keep the data confidential,  $DS_i$  must encrypt its own data first. At this time, it can adopt an efficient encryption/decryption method to generate a cipher-text. Then,  $DS_i$  can forward this cipher-text to a powerful base station ( $BS$  for short), which it equips with more computing power than all the sensor nodes in the same IoT network. However,  $DS_i$  must also consider its own memory limitation and remaining computing power to perform such encrypting/decryption computation in sequence. The node  $DS_i$  may not be able to perform the signature computation after it has generated encryption if the remaining power is not enough to perform signature generation in this time; thus, it must transfer the heavy computation to a powerful node such as the base station  $BS$ .

Due to the mentioned situations, we concluded that, if there exists an efficient method allowing IoT devices to perform encryption and signature operations on the sensed data in one operation, it could save more computing time and energy, which can then be used for other computations. In the recent literature, sign-cryption was discussed in [1–4]. The authors claimed that the sender can transfer the data only to perform one sign-cryption time, and it can output a cipher-text with a guaranteed signature within. Then, the receiver can decrypt the received cipher-text with a secret random number inside the corresponding signature. When the signature is verified by the receiver successfully, the receiver can obtain the random secret value by applying its own secret key. Finally, the receiver  $\mathcal{R}$  can obtain the final data by inputting this secret random number to decrypt the cipher-text. Unfortunately, their computation efficiency are not practical to fit above situation for IoT device network. There are some research limitations in our proposed scheme. One is that the sender device  $S$  is already authenticated with the receiver  $\mathcal{R}$ ; they both inherently trust each other within the same IoT network environment. The authentication mechanism is beyond the scope of this research. Another limitation is that IoT device management is also beyond our research. We can adopt other proposed authentication mechanisms [5–10] for devices to authenticate with each other in an IoT device network and also construct an IoT devices group with other devices. Our scheme focuses on the efficient signature and encryption scheme for these power limitation IoT devices such as the Zigbee chips or IoT sensor devices embedding less memory.

To provide a mechanism to generate a signature and a cipher-text for IoT devices simultaneously, we propose an efficient and practical, fair sign-cryption scheme based on quadratic residue (QR for short) for the IoT device network. Not only does it offer an efficient and practical solution to IoT devices, but it also reduces the signature and cipher-text generation cost in our methodology. We also offer the formal security proof on our proposed scheme in the Appendix A and evaluate the efficiency of our mechanism in this research.

## 2. Related Work and Security Definitions

### *Related Work*

In this section, we discuss the related research proposed in [1–4]. In [1], the authors propose a *CPAS* scheme for the vehicular sensor network and assume that there exists two TAs, where one is a tracing Authority (TRA for short) and the other is a public key generation center (PKG for short) for tracing the identity and key pairs of all vehicles, respectively. The TRA can produce a pseudo-ID for all vehicles after it has verified the real identity from them. The PKG also can generate the key-pairs for these vehicles. If there is a dispute in the protocol, the TRA can determine the real identity of the pseudo-ID key-pair through the help of the PKG. At this time, each vehicle does not show its real identity

through the above scheme’s methodology. On the other hand, we can discover that the total efficiency computation of this scheme is  $3Pa + 1SM$  for signature verification operation and  $3Pa + (n + 1)SM$  for  $n$  signatures batch verification, where  $Pa$  is a pairing operation and  $SM$  is a symmetric encrypting operation. We consider that the pairing operation is demanding for comparing our scheme with others in Table 1 for Internet of Things (IoT for short) devices. From the efficiency comparison in Table 1, we can see that our approach is much more efficient than [1]. In [2], we observed that authors also claim their scheme is more efficient than those in other articles [3,4]. However, this paper [2] is still slower than our proposed approach in Table 1.

On one hand, from the data authentication aspect, the gateway is unaware of what the sensor node’s data are in our approach. The sensor node will blind the forward data first before sending these data to the gateway. On the other hand, the gateway also provides its own random parameters during the signature generation of the offline-sign-ryption phase. This means that each signature is generated by the gateway’s signing parameters and the sensor node’s parameters after the above offline-sign-ryption and online-sign-ryption phases. Meanwhile, our approach can guarantee the situation where the signer cannot fully control the signature generation and provide the unlinkability to the signature. In [3], the authors provide an efficient sign-ryption methodology between the traditional public key crypto-system to the identity-based crypto-system and vice versa. This can be applied in the multireceiver construction for the IoT device network and provides a general prototype for this crypto-system transformation. We think that this idea is effective and suitable for the IoT device to transfer sensing data to another crypto-system construction. However, the sensing node still requires great computation effort on the paring operation and can cause a performance bottleneck on these sensor nodes. We also see in [3] that its computation cost is about  $3 Pa$ , where  $Pa$  is a pairing operation on a large prime number  $q$ . Finally, in [4], the authors claim their approach is only about  $4 Mu + 2 Pa$ , where  $Mu$  is the modular multiplication and  $Pa$  is the paring operation. After converting to the final computation approximately, we discover that this scheme still costs  $409 Mu$  more than ours in Table 1. In this approach, our contribution is to construct an efficient methodology that can generate a signature and encryption based on the QR at the same time and also preserve a concrete security proof on well-known hard problems such as the RSA factoring problem [11].

**Table 1.** Performance comparison.

	Sign-Cryption	Unsign-Cryption	Totally	Approx.
[1]	$2Mu + 1Pa$	$3Pa + 1Ad + 1\oplus$	$4Pa + 2Mu + 1Ad + 1\oplus$	$327Mu + 1\oplus$
[2]	$4Mu + 1Ex + 2Ha + 1\oplus$	$1Ex + 2Pa + 2Mu + 2Ha$	$2Ex + 2Pa + 6Mu + 4Ha + 1\oplus$	$647Mu + 1\oplus$
[3]	$4Ha + 1Ex + 2\oplus$	$3Ha + 1Pa + 2\oplus$	$1Ex + 1Pa + 7Ha + 4\oplus$	$322.8Mu + 4\oplus$
[4]	$1Ex + 2Mu + 2Ha + 1\oplus$	$2Pa + 3Ha + 1Ad +$	$1Ex + 2Pa + 2Mu + 1Ad + 5Ha + 1\oplus$	$409Mu + 1\oplus$
Ours	$4Ha + 29Mu + 1\oplus + 1SE$	$1SD + 2Ha + 1\oplus$	$33Mu + 1SE + 1SD + 6Ha + 2\oplus$	$36.2Mu + 2\oplus$

*Ex*—Modular exponentiation, *Ad*—Addition operation, *Mu*—Modular multiplication, *SE*—Symmetric Encryption operation, *Ha*—Hash operation, *SD*—Symmetric Decryption operation, *Pa*—Pairing operation,  $\oplus$ —XOR bit operation.

### 3. The Proposed Scheme

The following is our proposed scheme, which contains four phases: the initial phase, blinding phase, offline-sign-ryption phase, and the unsigned-ryption phase.

#### 3.1. Preliminary

In this subsection, we provide some definitions used in our proposed scheme as follows:

- $n$ : A large prime number, which it computes from two large primes  $p_1$  and  $p_2$  such that  $n = p_1 \cdot p_2$ , where  $p_1 \equiv p_2 \equiv 3 \pmod{4}$ .
- $l$ : The total number of all Internet of Things (IoT for short) nodes.
- $\hat{n}$ : A large prime number, which it computes from two large prime  $p_3$  and  $p_4$  such that  $\hat{n} = p_3 \cdot p_4$ , where  $p_3 \equiv p_4 \equiv 3 \pmod{4}$ .

- $DS_i$ : An IoT data sender, which is a sensor node that forwards collected data to the receiver  $R$ , where  $i = 1 \sim l$  and  $l$  is the number of all sensor nodes.
- $BS$ : A base station, which helps to collect data sent from a sensor node  $DS_i$ , where  $i = 1 \sim l$ .
- $R$ : An IoT data receiver, which receives data from the sender  $DS_i$ .
- $\oplus$ : An exclusive-or operation for symmetric encryption/decryption usage.
- $H_1, H_2$ : Two secure hash functions that each of them maps  $Z_n^* \rightarrow \{0, 1\}^n$  with collision-resistance and outputs the same  $n$ -bits hash strings.
- $E_{pk_j}$ : A symmetric key encryption function for the party  $j$  with the public key  $pk_j$ , where  $j \in \{DS_j, R\}$ , where  $j = 1 \sim l$ .
- $D_{sk_j}$ : A symmetric key decryption function for the party  $j$  with the private key  $sk_j$ , where  $j \in \{DS_j, R\}$ , where  $j = 1 \sim l$ .

### 3.2. Initial Phase

In this phase, an IoT node  $DS_i$  acts as a data sender; it first selects two large, distinct primes, where one is  $p_1$  and the other is  $p_2$  such that  $n = p_1 \cdot p_2$ , where  $l = 1 \sim l$  and  $l$  are totally node numbers.  $DS_i$  also publishes this  $n$  and we could know that given a QR in  $Z_n^*$ ; there are four different square roots (or 2 roots) of the QR in  $Z_n^*$ . From this property, we could derive the  $2^i$ th roots of the QR in  $Z_n^*$ , where  $i$  must be larger than 1 in  $Z_n^*$ . On one hand, we assume that there exists a powerful base station as a signer  $BS$ , which also selects two large primes, where one is  $p_3$  and the other is  $p_4$  in the same IoT network environment. It also computes  $\hat{n} = p_3 \cdot p_4$  and sets up to let  $n < \hat{n}$ . Then, it publishes  $\hat{n}$  and its prefix string  $\Omega$ . In the following, we take Fan and Lei's Scheme [12] as our reference. Nevertheless, the data receiver ( $R$  for short) sets up its own private/public key pair as  $(sk_R, pk_R)$ . When the set-up is finished, it publishes its own public key to the IoT network.

- First, a node  $DS_i$  randomly chooses its own QR numbers  $(z_1, z_2, z_3)$  from  $Z_n^*$  similar with  $y_1, y_2$  and  $y_3$ , where each of them is computed from  $y_i = (z_i^2 \text{ mod } n)$  and  $i = 1 \sim 3$ , respectively. Then, base station  $BS$  also selects two random QR numbers  $\alpha$  and  $\beta$  such that they allow  $(\beta^2/\alpha^2 \text{ mod } n)$  to belong to QR in  $Z_n^*$ .  $DS_i$  also publishes  $(n, y_1, y_2, y_3)$  to the signer  $BS$ . Once the signer  $BS$  has received them from  $DS_i$ ,  $DS_i$  computes  $\gamma = (\kappa^2 \text{ mod } \hat{n})$  with a random number  $\kappa$  and the identifier  $\hat{z} = H_1(z) \text{ mod } \hat{n}$  with an identifier number  $z$ . After setting up these random numbers,  $BS$  forwards  $(\gamma, \hat{n}, z, \hat{z})$  to  $DS_i$  and enters the offline-signing phase.

### 3.3. Offline-Signing Phase

- When  $DS_i$  has received  $(\gamma, \hat{n}, z, \hat{z})$  from the  $BS$ ,  $DS_i$  also computes the following messages if the checking of  $z$  is valid, where  $\hat{z} = H_1(z) \text{ mod } \hat{n}$ .  $DS_i$  selects a random number  $r \in Z_n^*$  and computes the following:

$$\begin{aligned} C_1 &= E_{pk_R}(r) \\ C_2 &= H_1(r) \oplus m \\ C_3 &= H_1(C_1, C_2, r, \hat{z}, m) \end{aligned} \tag{1}$$

- After computing the above equations,  $DS_i$  also allows  $\beta^2/\alpha^2$  as  $\tau$  and performs the following:

$$\begin{aligned} C'_1 &= C_1 * \tau^2 * \gamma \\ C'_2 &= C_2 * \gamma \\ C'_3 &= C_3 * \gamma \\ h &= H_1(C'_1, C'_2, C'_3) \end{aligned} \tag{2}$$

- From the above equations, we know that  $DS_i$  blinds the sensor data and computes a cipher-text  $(C'_1, C'_2, C'_3)$ . Then,  $DS_i$  forwards  $(C'_1, C'_2, C'_3, h, z, \hat{z})$  to  $BS$ . When  $BS$  has

received these messages from  $DS'_i$ , it verifies above them with  $z$ , checks the  $h$  from  $(C'_1, C'_2, C'_3)$ , and enters the online-signing phase.

### 3.4. Online-Signing Phase

- When  $BS$  obtains  $(C'_1, C'_2, C'_3, h, z, \hat{z})$  from  $DS_i$ , it could perform verification of these cipher-texts. If they are valid, then  $BS$  decrypts them with  $\gamma^{-1}$  as follows:

$$\begin{aligned} C_1 &= C'_1 * \tau^2 * \gamma^{-1} \\ C_2 &= C'_2 * \gamma^{-1} \\ C_3 &= C'_3 * \gamma^{-1} \end{aligned} \tag{3}$$

- After decrypting the above cipher-texts successfully,  $BS$  computes the signature as follows with a QR number  $\lambda$ :

$$\begin{aligned} C'_3 &= C_3^{-2} * \left(\frac{\beta}{\alpha}\right)^{-2} * (\lambda)^2 \\ C''_3 &= C'_3 * y_1 \pmod{n} \\ C''_2 &= C'_2 * y_2 \pmod{n} \\ C''_1 &= C'_1 * y_3 \pmod{n} \end{aligned} \tag{4}$$

- The signer  $BS$  finishes the signing operation and generates the signature  $(C''_1, C''_2, C''_3)$  to the data sender  $DS_i$ . When the node  $DS_i$  has received this signature, it could unblind the signature by computing the following operations:

$$\begin{aligned} C'_1 &= C''_1 * y_3^{-1} \\ C'_2 &= C''_2 * y_2^{-1} \\ C'_3 &= C''_3 * y_1^{-1} \\ C^*_3 &= C'_3 * \left(\frac{1}{\alpha}\right)^2 \\ &= C_3^{-2} * \beta^{-2} * (\lambda)^2 \end{aligned} \tag{5}$$

- Then, the  $DS_i$  computes the final encrypted cipher-text messages  $(C'''_1, C'''_2, C'''_3)$  to the  $BS$  in the following and enters the unsign-encryption phase:

$$\begin{aligned} C'''_1 &= C'_1 * \gamma \\ C'''_2 &= C'_2 * \gamma \\ C'''_3 &= C^*_3 * \gamma \end{aligned} \tag{6}$$

### 3.5. Unsign-Cryption Phase

- When  $BS$  received these cipher-text messages from  $DS_i$ , it can decrypt by the following operations:

$$\begin{aligned} C^*_3 &= C'''_3 * \gamma^{-1} \\ t &= (C^*_3)^2 * (\lambda)^{-4} \\ &= C_3^{-4} * \beta^{-4} \\ t^* &= t * y_1 \end{aligned} \tag{7}$$

- After  $BS$  has computed this signature  $t$  from the above equation, it forwards  $(t^*, z, \hat{z})$  to the node  $DS_i$  and allows the  $DS_i$  to decrypt  $t^*$  and un-blinds this signature  $t$  as follows:

$$\begin{aligned}
 t &= t^* * y_1^{-1} \\
 S_R &= t * \beta^4 \\
 &= C_3^{-4} * \beta^{-4} * \beta^4 \\
 &= C_3^{-4} \pmod n
 \end{aligned}
 \tag{8}$$

- After  $DS_i$  summarizes the above equation, we conclude that the node  $DS_i$  has the final signature  $\sigma_R = (S_R, C_1, C_2, C_3)$ , where  $S_R^4 = C_3 = H_1(C_1, C_2, \gamma, \tau, \hat{z}, m)$ . Then, the node  $DS_i$  can forward the sign-cryption signature  $\sigma_R$  and cipher-text messages  $(C_1, C_2, C_3)$  to the receiver  $R$  of the Internet host.
- Once the receiver  $R$  has obtained this sign-cryption signature  $\sigma_R$  and cipher-text messages  $(C_1, C_2, C_3)$  from  $DS_i$ , it can perform the following steps:

$$\begin{aligned}
 r^* &= D_{sk_R}(C_1) \\
 m &\stackrel{?}{=} C_2 \oplus H_1(r^*) \\
 C_3 &\stackrel{?}{=} H_1(C_1, C_2, r, \hat{z}, m) \\
 S_R^4 &\stackrel{?}{=} C_3
 \end{aligned}
 \tag{9}$$

#### 4. Functionality Comparisons and Security Analysis

In this section, we could provide functionality comparisons with other schemes and security analysis about our proposed scheme.

##### 4.1. Fast Sign-Cryption Operation

The proposed scheme only needs three hash operations, one  $\oplus$  operations, five multiplication operations, and one symmetric encryption in the offline-signing phase. In this situation, our proposed scheme is more efficient than [2]. In addition, the sensor node  $DS_i$  can blind the sensed data to the base station efficiently and with data confidence. The base station  $BS$  cannot be aware of the sensed data content. If the base station is compromised by a malicious attacker,  $DS_i$  can also protect this data to prevent its exposure outside the IoT network. At the same time, it also guarantees the protection of user’s personal information.

##### 4.2. Signer Fair Signature Operation

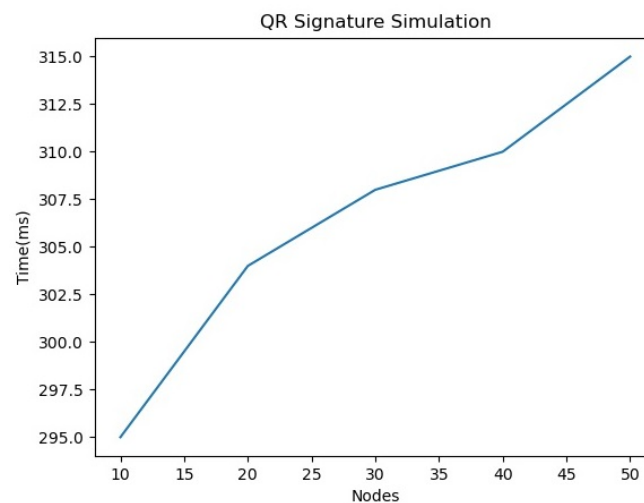
Our proposed scheme can offer the signature of sensed data after the base station  $BS$  has received the encrypted sensed data from the user. In this time,  $BS$  only can apply the square root operation on these sensed data to generate the corresponding signature under these blind and encrypted data. In the online-signing operation, the IoT device can perform lightweight operations on the user’s sensed data and obtain the signing result after the offline-signing phase performed by the signer  $BS$ . From the two signing phases above, we know that the IoT device and the base station can present some random numbers in these phases to prevent the unfair situation that the signature generation is controlled by a certain party.

##### 4.3. User Data Protection

In our proposed scheme, we use the sign-cryption method to generate the encryption data with the corresponding signature within. In this time, the signer cannot know what the plain-text is without the corresponding decryption key. Only the receiver is aware of the corresponding decryption key to decrypt this cipher-text. Thus, our sign-cryption scheme could offer privacy protection of the user’s personal sensed information.

#### 4.4. Efficiency Comparisons

In this section, we evaluate the efficiency of our approach in the following. First, there is an assumption that the prime numbers  $p_1, p_2, p_3$  and  $p_4$  are 1024 bits in length;  $Ha$  is computation time for one hash computation;  $SE$  is the time for a symmetric encryption operation, and  $SD$  is time for a symmetric decryption operation. Meanwhile, we also define that  $Ex$  is the computation time for one modular exponential operation in a 1024-bit module,  $Mu$  is the time for one modular multiplication in a 1024-bit module,  $M_{ecc}$  is the time for a number performing another point addition over an elliptic curve [13], and  $Pa$  is the time for the computation time of a bilinear pairing operation of two elements over an elliptic curve. Then, we assume that  $Ex \approx 8.24M_{ecc}$  for the ARM CPU to process at 200 Mhz in [14]. From the above assumption, we can discover that there exists some relation in the following, where  $Ex \approx 240Mu = 600Ha \approx 3Pa$  and  $Ad \approx 5Mu$  in [15–21]. From the above computation time evaluation, we can see that our approach total computation time is  $33Mu + 6Ha + 2 \oplus + 1SE + 1SD$ . Then, the result is approximate to  $36 Mu$  modular multiplication operations. Comparing with [2], we can see that our approach is much faster under the 1024-bit prime numbers. In the following two simulation results shown in Figures 1 and 2, our approach provides the QR-signature simulation and RSA signature simulation, respectively. On the other hand, we implemented our approach on a Ubuntu 20.04 operating system with Intel Core i5-1135G7 CPU @ Base 2.4 GHz up to 4.2 GHz CPU and 8 GB memory. This simulation is carried out by using GO language and python language with “crypto/encoding/Matplotlib” library on the 10 nodes to 50 nodes, where are shown in Figures 1 and 2, respectively.



**Figure 1.** QR Signature Simulation on 10 nodes to 50 nodes.

#### 4.5. Security Definitions

##### 4.5.1. QR Signature Security

We provide the definition on the digital signature’s security as follows: In the initial phase, we assume that there exists some functions used in our proposed scheme; one is the signature generating function  $Sig(\cdot)$  and the other is the verification function  $Ver(\cdot)$ , where the signer  $S$  can input her/his signing key  $sk_S$  into this signing function with the message  $m$ . Then, we can claim that  $\sigma$  is the resulting output from the signing function by  $S$  and the receiver  $R$  can verify  $\sigma$  by the verification function  $Ver(\cdot)$  with the message  $m$  and the signer’s public key  $pk_S$ . The above scheme is based on well-known hard problems such as the RSA factoring problem. If there exists an attacker  $\mathcal{F}$  whose goal is to forge a valid signature  $S'$  on the message  $m$  and pass the verification, i.e.,  $Ver(S', m, pk_S) = 1$ , then  $\mathcal{F}$  outputs it successfully with non-negligible probability larger than  $\epsilon$ , we can use  $\mathcal{F}$ ’s ability to factor the RSA factoring problem. However, in fact, the attacker  $\mathcal{F}$ ’s advantage is less than  $\epsilon$ .

This means that the probability of  $\mathcal{F}$  to output a forged signature and for this signature to pass the verification function with non-negligible probability is less than  $\epsilon$ .

$$Adv[S'_i \leftarrow \mathcal{F}^{Sig(sk_s, m)} | Ver(S'_i, m, pk_s) = 1] < \epsilon.$$

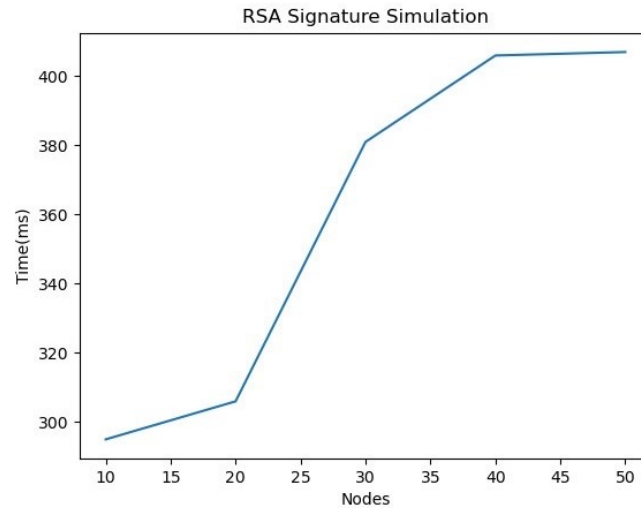


Figure 2. RSA Signature Simulation on 10 nodes to 50 nodes.

#### 4.5.2. Unforgeability

In this proposed scheme, we provide the signature definition of our sign-cryption scheme. From the above digital signature definition, we discuss the case where there exists a forger  $\mathcal{F}$  with the ability to forge a valid QR-signature on our scheme. We assume that there are some functions such that  $\mathcal{F}$  can make the hash query to the hash functions  $H_1(\cdot)$  and  $H_2(\cdot)$ , symmetric encryption  $Enc_{pk_R}(\cdot)$  function and the signing function  $Sig(\cdot)$ . After preparing these functions,  $\mathcal{F}$  can make its own query on these functions.  $\mathcal{F}$  can ask  $i$  times query, where  $i = 1 \sim l$  and  $l$  is the total number of IoT nodes. After the above  $q_s$  times query, if  $\mathcal{F}$  can output  $q_s + 1$  signatures on our proposed scheme, we can use  $\mathcal{F}$  to break the RSA factoring problem.

$$Adv_{\mathcal{F}^{Sig(\cdot), H_1(\cdot), H_2(\cdot), RO_1, E_{pk_R}(\cdot)}}^{Unf}(\theta, t') \leq \frac{1}{2^l \cdot q_s \cdot q_e \cdot q_d} + \epsilon'.$$

**Lemma 1.** First, we assume that there exists a secure digital signature function  $Sig(\cdot)$  and a secure hash function  $H_1(\cdot)$ , which could be replaced with a random oracle  $RO_1$  and a secure hash function  $H_2$  in our proposed scheme. We also claim that our proposed scheme with the above unforgeability (Unf for short) satisfies the following situations. In other words, if our scheme is  $(t', \epsilon')$  unforgeable, then

$$Adv_{\mathcal{F}^{Sig(\cdot), H_1(\cdot), H_2(\cdot), RO_1, E_{pk_R}(\cdot)}}^{Unf}(\theta, t') \leq \frac{1}{2^l \cdot q_s \cdot q_e \cdot q_d} + \epsilon'.$$

where  $t'$  is total experiment simulation time, including simulating  $l$  as an upper bound on the number of IoT devices, at most signature oracle  $q_s$  times query, at most encryption oracle  $q_e$  times query, at most decryption oracle  $q_d$  times query, and  $\epsilon'$  has taken over the coin toss of our scheme.

#### 4.5.3. Indistinguishability

In this definition, we assume the Indistinguishable (Ind for short) game where there exists an attacker  $\mathcal{A}$  in the following simulation, which is controlled by a simulator  $\mathcal{S}$ . First, we defined that there is a symmetric encryption/decryption function  $E_{pk_i}(\cdot)/D_{sk_i}(\cdot)$ , where  $i \in \{DS_j, BS, R\}, j = 1 \sim l$ , in which  $DS_j$  is one of the  $l$  IoT devices;  $BS$  is the base station, and  $R$  is the receiver of the outside network. The simulator  $\mathcal{S}$  will prepare all



set-up parameters including key pairs for the above parties. After set-up is complete,  $\mathcal{S}$  will launch the proposed scheme simulation with  $\mathcal{A}$ .  $\mathcal{A}$  can perform the encryption/decryption on the chosen message  $m$ .  $\mathcal{S}$  also can reply the cipher-text  $C = E_{pk_i}(m)$  and the original message  $m$  to  $\mathcal{A}$ . After the above game simulation,  $\mathcal{S}$  can replace the encryption/decryption functions to an encryption/decryption oracle  $(\tau, \tau^{-1})$ , which performs the same action as our above symmetric encryption/decryption function. Through the above training phase,  $\mathcal{A}$  sends a chosen target message  $(M_0, M_1)$  to  $\mathcal{S}$ ;  $\mathcal{S}$  will perform a coin flip  $b$  on the message  $(M_b, M_{1-b})$ . Then,  $\mathcal{S}$  inputs the  $M_b$  to the encryption oracle  $E_{pk_i}$  to obtain the final result  $C_b$ .  $\mathcal{S}$  forwards  $C_b$  to  $\mathcal{A}$  to guess whether  $M_b$  is  $M_0$  or  $M_1$  on its coin flip  $b'$ —that is,

$$\Pr[b' \leftarrow \mathcal{A}^{(E_{pk_i}(\cdot), D_{sk_i}(\cdot), \tau, \tau^{-1})} | b = b'] < \frac{1}{2} + \varepsilon'.$$

#### 4.5.4. Indistinguishable-Chosen Cipher-Text Attack (Ind-CCA for Short)

In this proposed scheme, we continue to define the chosen cipher-text attack security of our SC approach. There also exists an attacker  $\mathcal{A}$ , whose goal is to distinguish the cipher-text of our sign-cryption scheme. First, we assume that there is a simulator  $\mathcal{A}$  to control the environment situational parameters including key pairs, security parameters, and hash length. After setting up,  $\mathcal{S}$  defines the experiment in which  $\mathcal{A}$  can make a query as follows.

- Phase 1: In this phase, the attacker  $\mathcal{A}$  could make the encryption/decryption query on the chosen message  $m$ . If  $\mathcal{A}$  makes the encryption query on the  $m$  of the IoT device  $i$ , where  $i = 1 \sim l$ , then  $\mathcal{S}$  inputs the  $m$  into  $C_{i,1} = E_{pk_i}(\gamma_i)$ ,  $C_{i,2} = m \oplus H_1(\gamma_i)$  and  $C_{i,3} = H_1(C_{i,1}, C_{i,2}, \gamma_i, m)$ , where  $i = 1 \sim l$ . Here,  $\mathcal{S}$  will preserve these parameters into the encryption oracle list  $\mathcal{E}_i$  entry. On the other hand,  $\mathcal{A}$  asks the decryption query on the cipher-text  $(C_{i,1}, C_{i,2}, C_{i,3})$ ,  $\mathcal{S}$  will check if there are any parameters matching this cipher-text in the  $\mathcal{E}_i$  entry. If the answer is yes,  $\mathcal{S}$  forwards the original message back to  $\mathcal{A}$  and keeps this query in the decryption oracle  $\mathcal{D}_i$  entry.
- Challenge: In this phase, if  $\mathcal{A}$  chooses a target IoT device  $j^*$  and a message pair  $(M_0^*, M_1^*)$ , where  $M_0^*$  and  $M_1^*$  are never asked the encryption query and decryption query before,  $j^* \neq i$  and  $i = 1 \sim l$ . In this time,  $\mathcal{S}$  will toss the coin flip  $b$  and inputs the  $M_b^*$  into the encryption oracle  $E_{pk_{j^*}}(\cdot)$ . Finally,  $\mathcal{S}$  returns the target cipher-text  $(C_{1,j^*}, C_{2,j^*}, C_{3,j^*})$  to  $\mathcal{A}$ . When  $\mathcal{A}$  has received this target cipher-text, it still can make the decryption query on other cipher-texts except  $(C_{1,j^*}, C_{2,j^*}, C_{3,j^*})$ .

In the following, we model above the actions as game simulation steps that we played with the attacker  $\mathcal{A}$ .

$$Exp_{\mathcal{A}, SC}^{Ind-CCA-b}(\theta)$$

##### Phase 1

$$i \in \{1, \dots, l\}, M_i \leftarrow \mathcal{A}^{E_{pk_i}(\cdot, \theta), D_{sk_i}(\cdot, \theta), H_1(\cdot)}$$

$$\gamma_i \leftarrow \{0, 1\}^*$$

$$C_{1,i} \leftarrow E_{pk_i}(\gamma_i)$$

$$C_{2,i} \leftarrow M_i \oplus H_1(\gamma_i)$$

$$C_{3,i} \leftarrow H_1(C_{1,i}, C_{2,i}, \gamma_i, M_i)$$

##### Challenge Phase

$$b \in \{0, 1\}, j^* \neq i, (M_b^*, M_{1-b}^*) \leftarrow \mathcal{A}$$

$$M_{b,j^*} \xleftarrow{b} \mathcal{S}$$

$$C_{1,j^*} \leftarrow E_{pk_{j^*}}(\gamma_{j^*})$$

$$C_{2,j^*} \leftarrow M_b^* \oplus H_1(\gamma_{j^*})$$

$$C_{3,j^*} \leftarrow H_1(C_{1,j^*}, C_{2,j^*}, \gamma_{j^*}, M_{b,j^*})$$

$$b' \leftarrow \mathcal{A}^{(E_{pk_{j^*}}(\cdot, \theta), D_{pk_{j^*}}(\cdot, \theta), \tau, \tau^{-1})} (C_{1,j^*}, C_{2,j^*}, C_{3,j^*}, M_b^*, M_{1-b}^*)$$

Return  $b'$ .

The advantage of function of the adversary  $\mathcal{A}$  where it is defined as  $Adv_{\mathcal{A},SC}^{Ind-CCA}(\theta) = |Pr[Exp_{\mathcal{A},SC}^{Ind-CCA-1}(\theta) = 1] - Pr[Exp_{\mathcal{A},SC}^{Ind-CCA-0}(\theta) = 1]| < \epsilon'$ .

**Lemma 2.** We defined that our sign-cryption SC scheme can withstand Ind-CCA attacks if there exists no such attacker  $\mathcal{A}$  that could guess the cipher-text during above experiment  $Exp$  with non-negligible probability than  $\epsilon'$ , i.e.,

$$Adv_{\mathcal{A},SC}^{Ind-CCA}(\theta, t) < \frac{1 + \epsilon'}{2 \cdot q_e \cdot q_d},$$

where at most  $t$  time bound, at most  $q_e$  times encryption query, at most  $q_d$  times decryption query under the  $\theta$  security parameter.

**Theorem 1.** First, we assume that our sign-cryption SC scheme is an Ind-CCA secure symmetric encryption/decryption scheme with a secure hash random oracle  $H_1$  and also satisfied with the unforgeability (Unf) in the following. Then, we can say that, if SC is  $(t', \epsilon')$  Ind-CCA secure and unforgeable, then

$$Adv_{\mathcal{F},\mathcal{A},SC}^{Unf,Ind-CCA}(\theta, t) \leq \left( \frac{1}{2^l \cdot q_s \cdot q_e \cdot q_d} \cdot \epsilon + \frac{1 + \epsilon'}{2 \cdot q_e \cdot q_d} \right),$$

where  $t$  is the maximum total experiment time including adversary execution time,  $l$  is an upper bound on the number of all IoT devices of at most  $q_s$  times signing query, at most encryption oracle  $q_e$  times query, and at most decryption oracle  $q_d$  times query under the security parameter  $\theta$  in the experiment.

## 5. Conclusions

In the final result, we can see that our approach is suitable for an IoT device to compute the QR signature and encryption simultaneously. From Table 1, we also can see that our approach is more efficient than other schemes [1–4]. Our methodology not only efficiently computes the encryption and signature simultaneously, but can also support the fair protocol of two parties during communication between these IoT devices. This point also prevents allowing a single device such as the powerful gateway being compromised by attackers when IoT devices attempt to perform a signature operation or data exchange with this gateway. At the same time, this approach also provides data privacy protection for users. On one hand, our future goal is to develop a lightweight hierarchical sign-cryption scheme for IoT devices, and it can offer the authentication functionality between different levels of IoT devices with data privacy protection simultaneously. On the other hand, our approach can extend to develop a novel and real practical IoT data migration methodology for the IoT network in the future.

**Author Contributions:** Conceptualization, M.-T.C. and H.-C.H.; methodology, M.-T.C.; software, H.-C.H.; validation, M.-T.C. and H.-C.H.; formal analysis, M.-T.C.; investigation, H.-C.H.; resources, H.-C.H.; data curation, H.-C.H.; writing—original draft preparation, M.-T.C.; writing—review and editing, H.-C.H.; visualization, H.-C.H.; supervision, H.-C.H.; project administration, H.-C.H.; funding acquisition, H.-C.H. 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.

**Acknowledgments:** This study was supported in part by grants from the Ministry of Science and Technology of the Republic of China (Grant No. MOST 109-2221-E-167-028-MY2).

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

### Appendix A

**Proof of Theorem 1.** First, we define experiments of the above two security definitions and each attacker’s ability, respectively. We will provide the proof of Lemma 1 and also define that there exists an attacker  $\mathcal{F}$  whose goal is to forge a signature in the proposed scheme. We also define a simulator  $\mathcal{S}$  that can control the experiment of the proposed scheme. On the other hand,  $\mathcal{S}$  is given a signing oracle  $Sig(\cdot)$ , which can perform the same action as signature generation by the signer in our approach.  $\mathcal{S}$  also prepares all IoT device key pairs, including the receiver’s one.

Before beginning the experiment of digital signature,  $\mathcal{S}$  is given a hard RSA problem in  $n^*$  and its goal is to use the  $\mathcal{F}$ ’s ability to factor this  $n^*$ . During this time,  $\mathcal{S}$  will also prepare the symmetric encryption/decryption function for the  $\mathcal{F}$  encryption/decryption query. The query types are discussed below.

- **Encrypting query:**  $\mathcal{F}$  can make an encrypting query on the chosen message  $m$ , the target receiver  $i$  and the corresponding hash value  $H_1(r'_i)$ . During this time,  $\mathcal{S}$  checks the  $H_1$  list record and determines the random number  $r'_i$ . If there is no hash record on the list,  $\mathcal{S}$  will generate the  $(*, H_1(r'_i), r'_i)$  entry for the random number  $r'_i$  on the list. Then,  $\mathcal{S}$  generates the corresponding cipher-texts in the following:

$$\begin{aligned} C'_1 &= E_{pk_i}(r'_i) \\ C'_2 &= m \oplus H_1(r'_i) \\ C'_3 &= H_1(C'_1, C'_2, r'_i, m). \end{aligned} \tag{A1}$$

Then,  $\mathcal{S}$  forwards this cipher-text  $(C'_1, C'_2, C'_3)$  back to  $\mathcal{F}$  to finish this Encryption query and records  $(C'_1, C'_2, C'_3)$  into the  $H_1$  list to be noted as  $(C'_1, C'_2, C'_3, H_1(r'_i), r'_i)$ .

- **Decrypting query  $Dec(\cdot)$ :** When  $\mathcal{F}$  forwards a cipher-text  $(C'_1, C'_2, C'_3)$  to  $\mathcal{S}$ ,  $\mathcal{S}$  will search the  $H_1$  list to see if there is any entry in this list; if yes,  $\mathcal{S}$  uses the  $H_1(r'_i)$  to decrypt the cipher-text  $(C'_1, C'_2, C'_3)$ . Finally,  $\mathcal{S}$  returns  $m$  back to  $\mathcal{F}$ .
- **QR Signnature query:** When  $\mathcal{F}$  makes the signature query on the chosen message  $m$ ,  $\mathcal{S}$  will generate the following:

$$\begin{aligned} C'_1 &= E_{pk_i}(r'_i) \\ C'_2 &= m \oplus H_1(r'_i) \\ C'_3 &= H_1(C'_1, C'_2, r'_i, m) \\ S'_R{}^4 &= C'_3 \end{aligned} \tag{A2}$$

After generating the signature  $S'_R$  and the corresponding cipher-text  $(C'_1, C'_2, C'_3)$ ,  $\mathcal{S}$  will check the signature list  $s_1$  to see if there is any entry inside; if no,  $\mathcal{S}$  preserves the signature  $S'_R$  into the signature list and stores  $(C'_1, C'_2, C'_3, S'_R, H_1(r'_i), r'_i, m)$  in the  $s_1$  list. Then,  $\mathcal{S}$  transfers  $S'_R$  back to  $\mathcal{F}$ .  $\mathcal{F}$  can make the above signature query several times on the chosen message  $m$ . If  $\mathcal{F}$  has made  $l$  times signature query on the message  $m$ ,  $\mathcal{F}$  can forge  $l + 1$  signatures on the message  $m$ . Then, we can have the probability of adversary  $\mathcal{F}$

$$Adv_{\mathcal{F}, Sig(\cdot), Enc(\cdot), Dec(\cdot)}^{Unf}(\theta, t) \leq \frac{1}{2^l \cdot q_s \cdot q_e \cdot q_d} \cdot \epsilon, \tag{A3}$$

where there is at most  $q_s$  times signature query, at most  $q_e$  times encryption query, and at most  $q_d$  times decryption query in the polynomial  $t$  time bound under security parameter  $\theta$ .

Second, we present the proof of Lemma 2 as follows. We assumed that there exists an attacker  $\mathcal{A}$  whose goal is to distinguish a cipher-text  $(C_1, C_2, C_3)$  from a given message tuple  $(M_0, M_1)$  with non-negligible probability. Before simulating the experiment, we model a simulator  $\mathcal{S}$ , which is given a RSA hard problem  $n^*$  and its goal is to factor  $n^*$  and find the prime factor of  $n^*$ . During this time,  $\mathcal{S}$  also generates all key pairs of IoT devices including

the base gateway  $BS$  and the receiver  $R$ . When everything is ready, the  $S$  also allows  $\mathcal{A}$  to send query types in the following.

- Cipher-text query on  $Enc(\cdot)$ : In this simulation,  $\mathcal{A}$  can also launch a cipher-text query with an input the message  $m$ , the target receiver  $i$ , and the corresponding hash value  $H_1(r_i)$  to  $S$ . When receiving this query,  $S$  checks the  $H_1$  list records and finds out if there exists a random number  $r_i$  and other related records before. If there is no hash record on the list,  $S$  will generate a new entry  $(*, H_1(r_i), r_i)$  for the random number  $r_i$  on the list. Then,  $S$  performs the following steps:

$$\begin{aligned} C_1 &= E_{pk_i}(r_i) \\ C_2 &= m \oplus H_1(r_i) \\ C_3 &= H_1(C_1, C_2, r_i, m) \end{aligned} \tag{A4}$$

Subsequently,  $S$  sends this cipher-text  $(C_1, C_2, C_3)$  back to  $\mathcal{A}$  and stores  $(C_1, C_2, C_3)$  into the  $H_1$  list to be noted as  $(C_1, C_2, C_3, H_1(r_i), r_i)$ .

- Plain-text query on  $Dec(\cdot)$ : When  $\mathcal{A}$  makes a plain-text query on  $S$  with an cipher-text  $(C_1, C_2, C_3)$ ,  $S$  will search the  $H_1$  list first to see if there is any entry inside or not; if yes,  $S$  uses the  $H_1(r_i)$  to decrypt the cipher-text  $(C_1, C_2, C_3)$  and returns  $m$  back to  $\mathcal{A}$ .
- Signing query: When  $\mathcal{A}$  makes an QR signature signing query on the chosen cipher-text  $(C_1, C_2, C_3)$ ,  $S$  will calculate the following equations:

$$\begin{aligned} C_1 &= E_{pk_i}(r_i) \\ C_2 &= m \oplus H_1(r_i) \\ C_3 &= H_1(C_1, C_2, r_i, m) \\ S_R^4 &= C_3 \end{aligned} \tag{A5}$$

After performing the above training, we defined it as the Phase 1 training phase of the experiment in the above definition. In the next phase, the  $\mathcal{A}$  can send a target message tuple  $(M_0^*, M_1^*)$  and forward it to  $S$ . In this time,  $S$  will choose one of them by a coin toss on  $b$ . Then,  $S$  performs signing steps as follows:

$$\begin{aligned} C_1^* &= E_{pk_i}(r_i^*) \\ C_2^* &= M_b^* \oplus H_1(r_i^*) \\ C_3^* &= H_1(C_1^*, C_2^*, r_i^*, M_b^*) \\ S_R^{4*} &= C_3^* \end{aligned} \tag{A6}$$

After generating the above cipher-text  $(C_1^*, C_2^*, C_3^*, S_R^{4*})$ ,  $S$  returns it back to the  $\mathcal{A}$ . During this time,  $\mathcal{A}$  can make the decryption query except on the target cipher-text  $(C_1^*, C_2^*, C_3^*, S_R^{4*})$ . If  $\mathcal{A}$  can distinguish the cipher-text  $(C_1^*, C_2^*, C_3^*, S_R^{4*})$  computed from  $M_b^*$ , we can have

$$\begin{aligned} Adv_{\mathcal{A}, SC}^{Ind-CCA}(\theta) &= |Pr[Exp_{\mathcal{A}, SC}^{Ind-CCA-1}(\theta) = 1] - Pr[Exp_{\mathcal{A}, SC}^{Ind-CCA-0}(\theta) = 1]| \\ &= Pr[Exp_{\mathcal{A}, SC}^{Ind-CCA-1}(\theta) = 1] - (1 - Pr[Exp_{\mathcal{A}, SC}^{Ind-CCA-1}(\theta) = 1]) \\ &< \epsilon'. \end{aligned} \tag{A7}$$

Then, we can obtain that

$$Adv_{\mathcal{F}, \mathcal{A}, SC}^{Ind-CCA}(\theta, t) = Pr[Exp_{\mathcal{F}, \mathcal{A}, SC}^{Ind-CCA-1}(\theta) = 1] \leq \frac{1 + \epsilon'}{2 \cdot q_e \cdot q_d},$$

where at most  $q_e$  times encryption query and at most  $q_d$  times decryption query in the polynomial  $t$  time bound under the security parameter  $\theta$ . The probability that  $\mathcal{A}$  can

distinguish the above target cipher-text ( $C_1^*, C_2^*, C_3^*$ ) is less than  $\varepsilon'$ . We have summarized the above proofs of Lemmas 1 and 2. We can obtain

$$Adv_{\mathcal{F}, \mathcal{A}, \mathcal{SC}}^{Unf, Ind-CCA}(\theta, t) \leq \left( \frac{1}{2^l \cdot q_s \cdot q_e \cdot q_d} \cdot \varepsilon + \frac{1 + \varepsilon'}{2 \cdot q_e \cdot q_d} \right).$$

□

## References

1. Shim, K.A. CPAS: An Efficient Conditional Privacy-Preserving Authentication Scheme for Vehicular Sensor Networks. *IEEE Trans. Veh. Technol.* **2012**, *61*, 1874–1883. [\[CrossRef\]](#)
2. Naresh, V.S.; Reddi, S.; Kumari, S.; Allavarpu, V.D.; Kumar, S.; Yang, M.H. Practical Identity Based Online/Off-Line Signcryption Scheme for Secure Communication in Internet of Things. *IEEE Access* **2021**, *9*, 21267–21278. [\[CrossRef\]](#)
3. Sun, Y.; Li, H. Efficient signcryption between TPKC and IDPKC and its multi-receiver construction. *Sci. China Inf. Sci.* **2010**, *53*, 557–566. [\[CrossRef\]](#)
4. Li, F.; Xiong, P. Practical secure communication for integrating wireless sensor networks into the Internet of Things. *IEEE Sens. J.* **2013**, *13*, 3677–3684. [\[CrossRef\]](#)
5. Hammi, B.; Fayad, A.; Khatoun, R.; Zeadally, S.; Begriche, Y. A Lightweight ECC-Based Authentication Scheme for Internet of Things (IoT). *IEEE Syst. J.* **2020**, *3*, 3440–3450. [\[CrossRef\]](#)
6. Choi, S.; Ko, J.; Kwak, J. A Study on IoT Device Authentication Protocol for High Speed and Lightweight. In Proceedings of the 2019 International Conference on Platform Technology and Service (PlatCon), Jeju, Korea, 28–30 January 2019; pp. 1–5.
7. Ning, H.; Liu, H.; Yang, L.T. Aggregated-Proof Based Hierarchical Authentication Scheme for the Internet of Things. *IEEE Trans. Parallel Distrib. Syst.* **2015**, *3*, 657–667. [\[CrossRef\]](#)
8. Kim, B.; Yoon, S.; Kang, Y.; Choi, D. PUF based IoT Device Authentication Scheme. In Proceedings of the 2019 International Conference on Information and Communication Technology Convergence (ICTC), Jeju Island, Korea, 16–18 October 2019; pp. 1460–1462.
9. Lounis, K.; Zulkernine, M. T2T-MAP: A PUF-Based Thing-to-Thing Mutual Authentication Protocol for IoT. *IEEE Access* **2021**, *9*, 137384–137405. [\[CrossRef\]](#)
10. Taher, B.H.; Jiang, S.; Yassin, A.A.; Lu, H. Low-Overhead Remote User Authentication Protocol for IoT Based on a Fuzzy Extractor and Feature Extraction. *IEEE Access* **2019**, *7*, 148950–148966. [\[CrossRef\]](#)
11. Rivest, R.; Shamir, A.; Adleman, L. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM* **1978**, *21*, 120–126. [\[CrossRef\]](#)
12. Fan, C.I.; Lei, C.L. A User Efficient Fair Blind Signature Scheme for Untraceable Electronic Cash. *J. Inf. Sci. Eng.* **2002**, *18*, 47–58.
13. Kobitz, N.; Menezes, A.; Vanstone, S. The state of Elliptic curve cryptography. *Des. Codes Cryptogr.* **2000**, *19*, 173–193. [\[CrossRef\]](#)
14. Lauter, K. The Advantages of Elliptic curve cryptography for wireless security. *IEEE Wirel. Commun.* **2004**, *11*, 62–67. [\[CrossRef\]](#)
15. Bertino, G.; Breveglieri, L.; Chen, L.; Fragneto, P.; Harrison, K.; Pelosi, G. A pairing SW implementation for smart cards. *J. Syst. Softw.* **2008**, *81*, 1240–1247. [\[CrossRef\]](#)
16. Hankerson, D.; Menezes, A.; Scott, M. Software Implementation of pairings. *Identity-Based Cryptogr. Cryptol. Inf. Secur.* **2008**, *2*, 188.
17. Hohenberger, S. Advances in Signatures, Encryption, and E-Cash from Bilinear Groups. Ph.D. Dissertation, Massachusetts Institute of Technology, Cambridge, MA, USA, 2006.
18. Li, Z.; Higgins, J.; Clement, M. Performance of Finite Field Arithmetic in an Elliptic Curve Cryptosystem. In Proceedings of the 9th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems (MASCOTS'01), Cincinnati, OH, USA, 15–18 August 2001; pp. 249–256.
19. Ramachandran, A.; Zhou, Z.; Huang, D. Computing cryptography algorithm in Portable and embedded devices. In Proceedings of the IEEE International Conference on Portable Information Devices, Orlando, FL, USA, 25–29 May 2007; pp. 1–7.
20. Schneier, B. *Applied Cryptography*, 2nd ed.; John Wiley & Sons: New York, NY, USA, 1996.
21. Takashima, K. Scaling Security of Elliptic Curves with Fast Pairing Using Efficient Endomorphisms. *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.* **2007**, *90*, 152–159. [\[CrossRef\]](#)