sensors

MDPI

Article

A Novel Certificateless Signature Scheme for Smart Objects in the Internet-of-Things

Kuo-Hui Yeh [1], Chunhua Su [2,*], Kim-Kwang Raymond Choo [3] and Wayne Chiu [1]

1 Department of Information Management, National Dong Hwa University, Hualien 97401, Taiwan; khyeh@gms.ndhu.edu.tw (K.-H.Y.); 410235014@gms.ndhu.edu.tw (W.C.)
2 Division of Computer Science, University of Aizu, Aizu-Wakamatsu, Fukushima Pref. 965-8580, Japan
3 Department of Information Systems and Cyber Security, The University of Texas at San Antonio, San Antonio, TX 78249, USA; raymond.choo@fulbrightmail.org
* Correspondence: su@comm.eng.osaka-u.ac.jp

Academic Editors: Luca Roselli, Federico Alimenti and Stefania Bonafoni
Received: 19 February 2017; Accepted: 27 April 2017; Published: 1 May 2017

Abstract: Rapid advances in wireless communications and pervasive computing technologies have resulted in increasing interest and popularity of Internet-of-Things (IoT) architecture, ubiquitously providing intelligence and convenience to our daily life. In IoT-based network environments, smart objects are embedded everywhere as ubiquitous things connected in a pervasive manner. Ensuring security for interactions between these smart things is significantly more important, and a topic of ongoing interest. In this paper, we present a certificateless signature scheme for smart objects in IoT-based pervasive computing environments. We evaluate the utility of the proposed scheme in IoT-oriented testbeds, i.e., Arduino Uno and Raspberry PI 2. Experiment results present the practicability of the proposed scheme. Moreover, we revisit the scheme of Wang et al. (2015) and revealed that a malicious super type I adversary can easily forge a legitimate signature to cheat any receiver as he/she wishes in the scheme. The superiority of the proposed certificateless signature scheme over relevant studies is demonstrated in terms of the summarized security and performance comparisons.

Keywords: certificateless signature; Internet-of-things (IoT); security; sensors

1. Introduction

The boosting advances on wireless communication and sensing technologies bring universal Internet connectivity, and a more ubiquitous and pervasive computing environment is thus created, called Internet-of-Things (i.e., IoT). Plenty of novel smart objects with specific purposes emerge in IoT to support various innovative applications providing higher intelligence and more convenience to our daily life. Since IoT has attracted significant attention as a key step in furthering intelligent human life in the future, IoT is definitely one of the most promising network paradigms in this computer generation. In an IoT environment, numerous smart objects, such as customized sensors or wearable intelligent devices, can be used to sense, collect, transmit, disseminate, etc., data from the field to a server or other smart things. Unsurprisingly, IoT has wide industrial and individual applications. However, due to the amount and nature of data and potential for exploitation, it is essential to ensure the security of both data-in-transit and data-at-rest [1–4]. In addition, the heterogeneous nature of the IoT network and the presence of (a large number of) specific-purpose sensors embedded within the smart objects complicate efforts to offer effective security. One particular research challenge is to balance the tradeoff between performance efficiency and system security when designing security solutions for smart objects in IoT-based networks.

In the literature, researchers have dedicated significant efforts on refining traditional security techniques as system security solutions for IoT-based network architectures, such as authentication [5–9], signcryption [10–13], and certificateless digital signature [14,15], respectively. First of all, due to the nature of limited processing capability of smart objects, the design of lightweight authentication has been thoroughly investigated as a critical security component in IoT-based network systems. In this category of study, lightweight but robust crypto-modules, such as one way hash function, are embedded into the operation and communication of resource-constrained IoT-based objects to support the security of application operated by objects and backend servers (from service providers). It simultaneously focuses on the computation efficiency and communication robustness of object-to-object and object-to-server data exchange procedures. Secondly, the signcryption technique combines the merits from encryption and digital signature. Most of critical security requirements, such as confidentiality, integrity, unforgeability, and non-repudiation, can be guaranteed in a single logic step. It enjoys better security robustness than other kinds of single-crypto-based security mechanisms. Thirdly, the refinement of certificateless digital signature for protecting IoT-based networks has been studied because of the benefit from the relief on the difficult certificate management in traditional public key infrastructure. Relying on a trusted third party, certificateless public key cryptography facilitates users in establishing a private key and the corresponding public key. It is, thus, more suitable to IoT-based network architecture since there is no need to maintain a centralized server for key/certificate management. In addition, with the decentralized and changed structure, it is believed that we the more efficiency will be guaranteed due to the less of limitation on implementing security mechanism on IoT. Existing certificateless signature schemes can be broadly categorized into certificateless signature schemes with and without bilinear pairing. It has been proven that bilinear pairing is less efficient than ECC (elliptic curve cryptography) point-based crypto-operations, in terms of computation costs [16], although the use of bilinear pairing results in shorter signature message. The latter property makes bilinear pairing-based approach particularly suitable for bandwidth-limited networks, such as traditional wireless sensor networks. Nevertheless, owing to the recent advancements in communication technologies, including those for sensors, the communication environment for existing IoT-based sensors is not as limited by bandwidth restriction as before. Various techniques, such as Bluetooth Low Energy, LoRa, and Zigbee, have been leveraged to build IoT-based communication networks which are bandwidth-guaranteed during sensors-to-server message transmission. Hence, during the design of an efficient and secure certificateless signature scheme for IoT-based smart objects, we argue that computation efficiency takes priority over communication efficiency. For the above observations, in this paper we focus on the design of a certificateless signature scheme with ECC point-based crypto-operations for IoT-based network environments.

The rest of the paper is organized as follows. Section 2 presents relevant background materials. In Section 3, we present the proposed certificateless signature scheme for IoT-based smart objects. We then provide the security analysis and the system implementation of our proposed scheme in Sections 4 and 5, respectively. In Section 6, we review related work and present a comparative summary, in terms of security and performance. Finally, we conclude the paper in Section 7.

2. Preliminary

The objective of this study is to propose a robust and efficient certificateless signature scheme with ECC point-based crypto-operations. ECC is one kind of public key cryptography (PKC)-based techniques, where it is based on the algebraic structure of elliptic curves over finite fields. Normally, ECC requires a smaller key size than other PKC-oriented approaches to provide an equivalent security level. For example, it is generally thought that the same security can be delivered by 256-bit elliptic curve and 3072-bit RSA. Hence, to enjoy higher computation efficiency, we would like to integrate the ECC crypto-technique into our proposed certificateless signature scheme. Furthermore, since the robustness of the proposed scheme is based on the hardness of solving the Elliptic Curve Discrete Logarithm Problem (ECDLP), we present the definition of ECDLP in the following.

- The ECDLP is defined as follows: Let the notation E/E_p denotes an elliptic curve E over a prime finite field E_p, defined by an equation: $y^2 = x^3 + ax + b$, where $a,\ b \in F_p$ are constants such that

$\Delta = 4a^3 + 27b^2 \neq 0$. All points $P_i = (x_i, y_i)$ on E and the infinity point O form a cyclic group G under the operation of point addition $R = P + Q$ defined based on the chord-and-tangent rule. In addition, $t \cdot P = P + P + \ldots + P$ (t times) is defined as a scalar multiplication, where P is a generator of G with order n. The ECDLP is that given a group G of elliptic curve points with prime order n, a generator P of G and a point $x \cdot P$, it is computationally infeasible to derive x, where $x \in Z_n^*$.

The robustness of the proposed certificateless signature scheme is based on the intractability of ECDLP. Next, for better understanding of our proposed scheme, we present the general concepts of the certificateless signature. A certificateless signature scheme generally consists of six phases, i.e., Setup, PartialPrivateKeyExtract, SetSecretValue, SetPublicKey, Sign, and Verify [17]. Note that the four phases, i.e., Setup, PartialPrivateKeyExtract, SetSecretValue, and SetPublicKey, can be treated as a pre-processing stage. In the following, we briefly review the normal process of a general certificateless signature scheme (Figure 1).

- Step 1 (Setup phase): A trusted KGC (key generation center) generates a master secret key $s \in Z_n^*$, a corresponding master public key PK_{KGC} and a set of public parameters, i.e., *params*.
- Step 2 (PartialPrivateKeyExtract phase): With the master secret key s, *params* and the user i's identity ID_i, KGC generates a partial secret key D_i for the user i.
- Step 3: KGC sends D_i to the user i.
- Step 4 (SetSecretValue phase): Upon receiving D_i, the user i examine the correctness of D_i. If it holds, the user i randomly selects a value $x_i \in Z_n^*$ as his/her secret. Otherwise, the session is terminated.
- Step 5 (SetPublicKey phase): With *params* and x_i, the user i generates and outputs his/her public key PK_i.
- Step 6 (Sign phase): With the message m, this phase outputs a signature σ_i which is based on m, s and x_i.
- Step 7: the user i sends σ_i to the verifier.
- Step 8 (Verify phase): With the signature σ_i of the message m, the verifier examine the correctness of σ_i. If the examination holds, the signature is valid. Otherwise, the session is terminated.

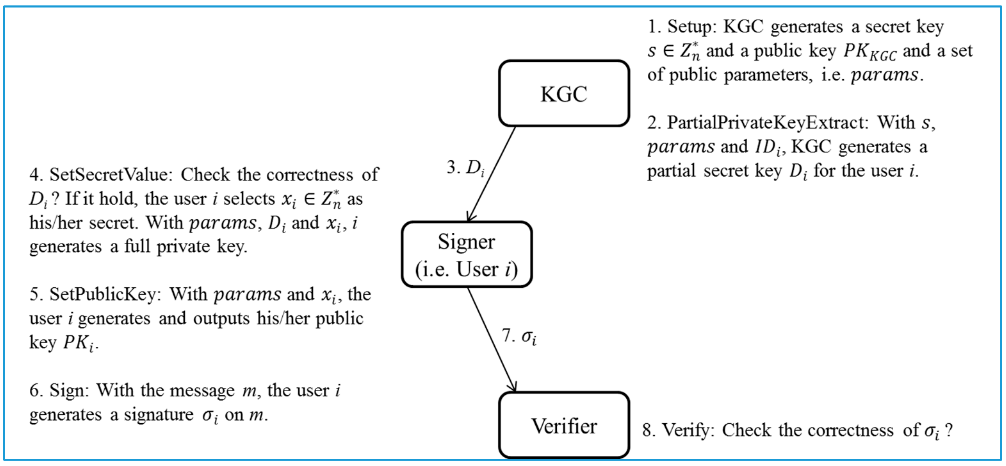

1. Setup: KGC generates a secret key $s \in Z_n^*$ and a public key PK_{KGC} and a set of public parameters, i.e. $params$.
2. PartialPrivateKeyExtract: With s, $params$ and ID_i, KGC generates a partial secret key D_i for the user i.
KGC
3. D_i
4. SetSecretValue: Check the correctness of D_i? If it hold, the user i selects $x_i \in Z_n^*$ as his/her secret. With $params$, D_i and x_i, i generates a full private key.
Signer (i.e. User i)
5. SetPublicKey: With $params$ and x_i, the user i generates and outputs his/her public key PK_i.
7. σ_i
6. Sign: With the message m, the user i generates a signature σ_i on m.
Verifier
8. Verify: Check the correctness of σ_i?

Figure 1. The normal process of a general certificateless signature scheme.

3. The Proposed Certificateless Signature Scheme for IoT-Based Smart Objects

In this section, we propose a new certificateless signature scheme with ECC point-based crypto-operations. The security of the scheme assumes the intractability of ECDLP. In the following, we present the proposed scheme consisting of two phases, i.e., the Pre-processing phase and Sign/Verify phase. Note that three entities, i.e., KGC, the signer and the verifier, are involved.

- Pre-processing phase (Figure 2):

    - Steps 1–4: KGC generates a group G of elliptic curve points with prime order n and determines a generator P of G. Then, KGC chooses a master key $s \in Z_n^*$ and a secure hash function $H_1 : \{0,1\}^* \times G \to Z_q^*$. Next, KGC calculates a master public key $PK_{KGC} = s \cdot P$. Eventually, KGC publishes $params = (G, P, PK_{KGC}, H)$ and keeps s securely. Next, given $params$, s and the identity ID_i of user i, KGC generates a random number $r_i \in Z_n^*$, and calculates $R_i = r_i \cdot P$, $h_i = H(ID_i, PK_{KGC})$ and $s_i = r_i + h_i \cdot s \bmod n$.
    - Steps 5–6: KGC returns a partial private key $D_i = (s_i, R_i)$ to the user i who checks the validity of D_i via whether the equation $s_i \cdot P = R_i + h_i \cdot PK_{KGC} \bmod n$ holds or not. The correctness of D_i is presented as follows:

$$s_i \cdot P = (r_i + h_i \cdot s) \cdot P = r_i \cdot P + h_i \cdot s \cdot P = R_i + h_i \cdot PK_{KGC}$$

    - Steps 7–8: If it holds, the user i picks a random number $x_i \in Z_n^*$ as his/her own secret value. Otherwise, the session is terminated. Then, given $params$ and x_i, the user i computes $PK_i = x_i \cdot P + R_i$ as his/her public key.

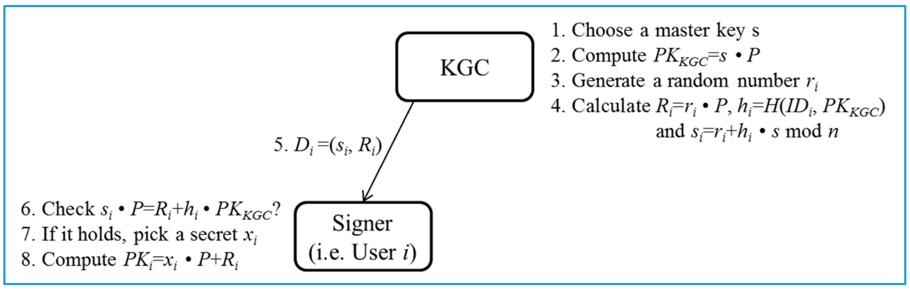
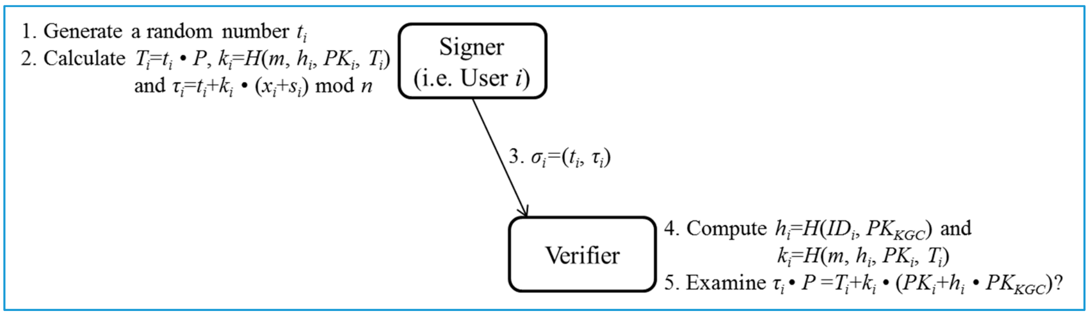

KGC
1. Choose a master key s
2. Compute PK_{KGC}=s • P
3. Generate a random number r_i
4. Calculate R_i=r_i • P, h_i=H(ID_i, PK_{KGC}) and s_i=r_i+h_i • s mod n
5. D_i=(s_i, R_i)
6. Check s_i • P=R_i+h_i • PK_{KGC}?
7. If it holds, pick a secret x_i
8. Compute PK_i=x_i • P+R_i
Signer (i.e. User i)

Figure 2. Pre-processing phase of the proposed certificateless signature scheme.

- Sign/Verify phase (Figure 3):

    - Steps 1–3 (Sign): Given $params$, D_i, x_i and a message m, the user i first chooses a random number $t_i \in Z_n^*$. Then, the user i computes $T_i = t_i \cdot P$, $k_i = H(m, h_i, PK_i, T_i)$ and $\tau_i = t_i + k_i \cdot (x_i + s_i) \bmod n$. Note that the computation of h_i is performed at the Pre-processing phase and thus the cost can be removed. Finally, the user i outputs $\sigma_i = (T_i, \tau_i)$ as the signature of the message m.
    - Steps 4–5 (Verify): Given $params$, ID_i, PK_i, and $\sigma_i = (T_i, \tau_i)$, the verifier first computes $h_i = H(ID_i, PK_{KGC})$ and $k_i = H(m, h_i, PK_i, T_i)$. Next, the verifier examines if $\tau_i \cdot P = T_i + k_i \cdot (PK_i + h_i \cdot PK_{KGC})$ holds. The signature σ_i is accepted if the equation holds. The correctness of the signature $\sigma_i = (T_i, \tau_i)$ is presented as follows:

$$\begin{aligned}\tau_i \cdot P &= (t_i + k_i \cdot (x_i + s_i)) \cdot P \\ &= t_i \cdot P + k_i \cdot (x_i + r_i + h_i \cdot s) \cdot P \\ &= T_i + k_i \cdot (x_i \cdot P + r_i \cdot P + h_i \cdot s \cdot P) \\ &= T_i + k_i \cdot ((x_i \cdot P + R_i) + h_i \cdot PK_{KGC}) \\ &= T_i + k_i \cdot (PK_i + h_i \cdot PK_{KGC})\end{aligned}$$



1. Generate a random number t_i
2. Calculate T_i=t_i • P, k_i=H(m, h_i, PK_i, T_i) and τ_i=t_i+k_i • (x_i+s_i) mod n
Signer (i.e. User i)
3. σ_i=(t_i, τ_i)
Verifier
4. Compute h_i=H(ID_i, PK_{KGC}) and k_i=H(m, h_i, PK_i, T_i)
5. Examine τ_i • P =T_i+k_i • (PK_i+h_i • PK_{KGC})?

Figure 3. Sign/Verify phase of the proposed certificateless signature scheme.

4. Security Analysis

We will now define the adversary model we used to prove the security of our scheme, prior to presenting the security analysis.

4.1. Adversary Model for Certificateless Signature

In the proposed certificateless signature scheme, we considered type I adversary and type II adversaries as defined in [18]. Due to the lack of certificate verification, it is possible for adversaries to replace an entity's public key with one of its choice. Therefore, the type I adversary models an external adversary capable of replacing any entity's public key with specific values chosen by the adversary itself. Nevertheless, the type I adversary does not know the private key of KGC. On the other hand, the type II adversary models a malicious KGC who is able to access the master key, but cannot replace the public keys of other entities. In addition, type I and II adversaries can be further classified into three categories of power levels [17,19], i.e., normal adversary, strong adversary, and super adversary. A normal-level adversary only has the ability to learn a valid verification message. A strong-level adversary is able to replace a public key in order to forge a valid verification message when the adversary possesses a corresponding private value. A super-level adversary is able to learn valid verification messages for a replaced public key without any submission. Normally, the super adversary may issue the following queries.

- $CreateUser(ID_t)$: The oracle takes as input a query (ID_t), where ID_t is the party t's identity, and then runs algorithms PartialPrivateKeyExtract, SetSecretValue, and SetPublicKey to obtain the partial private key D_t, the secret value x_t, and the public key PK_t.
- $RequestPublicKey(ID_t)$: The oracle takes as input a query (ID_t). It browses the list L and returns the party t's public key PK_t.
- $ReplacePublicKey\left(ID_t, PK_t, PK_t'\right)$: The oracle takes as input a query $\left(ID_t, PK_t, PK_t'\right)$. This oracle replaces the party t's public key with PK_t' and updates the corresponding information in the list L.
- $ExtractSecret(ID_t)$: The oracle takes as input a query ID_t. It browses the list L and returns the secret values x_t. However, if the party t has been asked the $ReplacePublicKey$ query, it returns $\perp$.
- $ExtractPartialSecret(ID_t)$: The oracle takes as input a query ID_t. It then browses the list L and returns the partial private key $D_t = (s_t, R_t)$.
- $SuperSign(ID_t, m_t)$: The oracle takes as input a query (ID_t, m_t), where m_t denotes the message to be signed. This oracle outputs a signature $\sigma_t = (R_t, T_t, \tau_t)$ such that $true \leftarrow Verify(m_t, \sigma_t, params, ID_t, PK_t)$. If the public key has not been replaced, i.e., $PK_t = PK_t$, PK_t is the public key returned from the oracle $RequestPublicKey(ID_t)$. Otherwise, $PK_t = PK_t'$, where PK_t' is the latest public key value submitted to the oracle $ReplacePublicKey\left(ID_t, PK_t, PK_t'\right)$.

The following two games, i.e., Games 1 and 2, are against super type I and type II adversaries, respectively. Type I adversary models an external adversary who is able to replace any entity's public key with specific values chosen by the adversary itself. On the other hand, type II adversary simulates a malicious KGC who holds the master key and might engage in adversarial activities, such as eavesdropping on signatures and asking signing queries.

Game 1. *This game is performed between a challenger C and a super type I adversary SA_1 interacting within the proposed certificateless signature scheme. First, in the "Initialization" stage, the challenger C runs the Setup algorithm and generates a private key s, and public system parameters params. Next, C keeps s, but gives params to the adversary SA_1. Second, in the "Query" phase, SA_1 can adaptively access oracle queries $CreateUser(ID_t)$, $RequestPublicKey(ID_t)$, $ReplacePublicKey(ID_t, PK_t, PK'_t)$, $ExtractSecret(ID_t)$, $ExtractPartialSecret(ID_t)$ and $SuperSign(ID_t, m_t)$, of C, where t may be the user i. After all necessary queries have been asked, SA_1 outputs a forged signature (ID_t, m_t, σ_t). SA_1 wins in Game 1 if the following three conditions hold*:

(1) *SA_1 has never queried the oracle $ExtractPartialSecret(ID_t)$.*
(2) *SA_1 has never queried the oracle $SuperSign(ID_t, m_t)$.*
(3) *true ← Verify(m_t, σ_t, params, ID_t, PK_t) where PK_t is the current public key of party t and it may be replaced by SA_1.*

Definition 1. *The proposed certificateless signature scheme is existentially unforgeable against a super type I adversary SA_1, if SA_1 runs in polynomial time pt, makes at most q_H queries to the oracle $Hash(.)$, q_{CU} queries to the oracle $CreateUser(ID_t)$, q_{EPS} queries to the oracle $ExtractPartialSecret(ID_t)$, q_{ES} queries to the oracle $ExtractSecret(ID_t)$, q_{PK} queries to the oracle $RequestPublicKey(ID_t)$, q_{RPK} queries to the oracle $ReplacePublicKey(ID_t, PK_t, PK'_t)$ and q_{SS} queries to the oracle $SuperSign(ID_t, m_t)$ and $Succ_{SA_1}$ is negligible, where $Succ_{SA_1}$ is the success probability that SA_1wins in Game 1.*

Game 2. *This game is performed between a challenger C and a super type II adversary SA_2 interacting within the proposed certificateless signature scheme. First, in the "Initialization" phase, the challenger C runs the Setup algorithm and generates a private key s, and public system parameters params. Then, C keeps s, but gives params to the adversary SA_2. Second, in the "Query" phase, SA_2 can adaptively access the oracle queries $CreateUser(ID_t)$, $RequestPublicKey(ID_t)$, $ReplacePublicKey(ID_t, PK_t, PK'_t)$, $ExtractSecret(ID_t)$, $ExtractPartialSecret(ID_t)$ and $SuperSign(ID_t, m_t)$, of C, where t may be the user i. After all necessary queries have been asked, SA_2 outputs a forged signature (ID_t, m_t, σ_t). SA_2 wins in Game 2 if the following three conditions hold:*

(1) *SA_2 has never queried the oracle $ExtractSecret(ID_t)$.*
(2) *SA_2 has never queried the oracle $SuperSign(ID_t, m_t)$.*
(3) *true ← Verify(m_t, σ_t, params, ID_t, PK_t), where PK_t is the original public key of party.*

Definition 2. *The proposed certificateless signature scheme is existentially unforgeable against a super type II adversary SA_2, if SA_2 runs in polynomial time pt, makes at most q_H queries to the oracle $Hash(.)$, q_{CU} queries to the oracle $CreateUser(ID_t)$, q_{EPS} queries to the oracle $ExtractPartialSecret(ID_t)$, q_{ES} queries to the oracle $ExtractSecret(ID_t)$, q_{PK} queries to the oracle $RequestPublicKey(ID_t)$, q_{RPK} queries to the oracle $ReplacePublicKey(ID_t, PK_t, PK'_t)$ and q_{SS} queries to the oracle $SuperSign(ID_t, m_t)$ and $Succ_{SA_2}$ is negligible, where $Succ_{SA_2}$ is the success probability that SA_2 wins in Game 2.*

4.2. Formal Analysis

Assuming the hardness of solving ECDLP, we prove that our proposed scheme is existentially unforgeable against the super type I adversary and super type II adversary, respectively.

Theorem 1. *The proposed certificateless signature scheme is existentially unforgeable against a super type I adversary in the random oracle model, assuming the hardness of solving ECDLP. That is, if there exists a super type I adversary SA_1 who can submit queries to random oracles and win in Game 1 with probability $Succ_{SA_1}$,* *then there is an algorithm β which can solve a random instance of ECDLP in polynomial time with success probability* $Succ_{\beta} \geq \frac{1}{q_{CU}+q_H}\left(1-\frac{1}{q_{CU}+q_H}\right)^{q_{EPS}} Succ_{SA_1}$.

Proof. Let SA_1 be a super type I adversary SA_1 which can compromise our proposed certificateless signature scheme with a non-negligible probability $Succ_{SA_1}$. We then construct a polynomial-time algorithm β which can utilize SA_1 to solve ECDLP. At first, β contains a hash list L_{H_1} and a key list L_{K_1}, which are initially empty.

- Initialization phase: β picks an identity ID^* as the challenged identity in Game 1, sets PK_{KGC} and sends $params = (G, P, PK_{KGC}, H)$ to SA_1.
- Query phase:

  ➢ $CreateUser(ID_t)$: The oracle takes as input a query (ID_t). If ID_t has been created, nothing happens. Otherwise, β runs algorithms PartialPrivateKeyExtract, SetSecretValue, and SetPublicKey to obtain the partial private key D_t, the secret value x_t and the public key PK_t. Next, β returns PK_t to SA_1.

  ➢ *Hash* query:

  (1) When SA_1 accesses a hash query on (ID_t, PK_{KGC}), if the list L_{H_1} contains $< h_t, ID_t, PK_{KGC} >$, β returns h_t to SA_1. Otherwise, β picks a random number $h_t \in Z_n^*$, returns h_t to SA_1, and adds $< h_t, ID_t, PK_{KGC} >$ to L_{H_1}.

  (2) When SA_1 accesses a hash query on (m, h_t, PK_t, T_t), if the list L_{H_1} contains $< k_t, m, h_t, PK_t, T_t >$, β returns k_t to SA_1. Otherwise, β picks a random number $k_t \in Z_n^*$, returns k_t to SA_1, and adds $< k_t, m, h_t, PK_t, T_t >$ to L_{H_1}.

  ➢ $RequestPublicKey(ID_t)$: Upon receiving a *RequestPublicKey* query with an identity ID_t from SA_1, β performs the following steps.

  (1) If $ID_t \neq ID^*$, β selects three random numbers $a_t, b_t, x_t \in Z_n^*$, and performs $s_t \leftarrow a_t$, $h_t \leftarrow b_t$, $R_t \leftarrow a_t \cdot P - b_t \cdot PK_{KGC}$, and $PK_t = x_t \cdot P + R_t$. Then, β adds $\langle ID_t, R_t, h_t \rangle$ to list L_{H_1}, and $\langle ID_t, s_t, R_t \rangle$ and $\langle ID_t, PK_t, x_t \rangle$ to list L_{K_1}, respectively. Finally, β returns PK_t to SA_1.

  (2) Otherwise, β generates three random numbers $a_t, b_t, x_t \in Z_n^*$, and sets $R_t \leftarrow a_t \cdot P$, $h_t \leftarrow b_t$, $s_t \leftarrow \perp$ and $PK_t = x_t \cdot P + R_t$. Then, β adds $\langle ID_t, R_t, h_t \rangle$ to list L_{H_1}, and $\langle ID_t, \perp, R_t \rangle$ and $\langle ID_t, PK_t, x_t \rangle$ to list L_{K_1}, respectively. Finally, β returns PK_t to SA_1.

  ➢ $ExtractPartialSecret(ID_t)$: Upon receiving an *ExtractPartialSecret* query for an identity ID_t from SA_1, β performs the following steps.

  (1) If $ID_t = ID^*$, β stops the session.

  (2) Otherwise, β looks at L_{H_1} for $< ID_t, s_t, R_t >$. If there exists a record of such a tuple, β returns s_t to SA_1; otherwise, β makes a *RequestPublicKey* query with ID_t and returns s_t to SA_1 accordingly.

  ➢ $ExtractSecret(ID_t)$: When β receives an *ExtractSecret* query for an identity ID_t from SA_1, β looks for $\langle ID_t, PK_t, x_t \rangle$ in the list L_{K_1}. If there is such a tuple, β returns x_t to SA_1. Otherwise, β makes a $ExtractPartialSecret(ID_t)$ query and returns x_t to SA_1.

  ➢ $ReplacePublicKey\left(ID_t, PK_t, PK_t'\right)$: Once β receives a query for some $\left(ID_t, PK_t, PK_t'\right)$ from SA_1, β looks for $\langle ID_t, PK_t, x_t \rangle$in the list L_{K_1}. If there exists such a record, β sets $PK_t = PK_t'$ and $x_t = \perp$. Otherwise, β makes a *RequestPublicKey* query with ID_t and then sets $PK_t = PK_t'$ and $x_t = \perp$.

  ➢ $SuperSign(ID_t, m_t)$: Upon receiving a *SuperSign* query with (ID_t, m_t) from SA_1, β looks for $< ID_t, s_t, R_t >$ and $\langle ID_t, PK_t, x_t \rangle$ in the lists L_{K_1}. Next, β generates a random number

$c_t \in Z_n^*$, and computes $\tau_t \leftarrow c_t$ and $T_t = \tau_t \cdot P - k_t \cdot (PK_t + h_t \cdot PK_{KGC})$. After that, β returns $\sigma_t = (T_t, \tau_t)$ to SA_1.

Finally, SA_1 outputs a forged but valid signature (ID_t, m_t, σ_t). If $ID_t = ID^*$, β terminates the simulation. Otherwise, β looks for $< h_t, ID_t, PK_{KGC} >$, $< k_t, m, h_t, PK_t, T_t >$, $\langle ID_t, s_t, R_t \rangle$, and $\langle ID_t, PK_t, x_t \rangle$ in the lists L_{H_1} and L_{K_1}. On the other hand, based on the forking lemma [20], if we have the polynomial replay of β with the same random tape and different choices of hash oracle, SA_1 is able to output another two valid signatures. Eventually, we will have three valid signatures, i.e., $\sigma_t{}^{(j)} = (T_t{}^{(j)}, \tau_t{}^{(j)})$ with j = 1, 2, 3, satisfying the equations, i.e., $\tau_t{}^{(j)} = t_t{}^{(j)} + k_t{}^{(j)} \cdot (x_t + s_t{}^{(j)}) = t_t{}^{(j)} + k_t{}^{(j)} \cdot (x_t + r_t + h_t{}^{(j)} \cdot s)$ mod n, where j = 1, 2, 3. Note that winning Game 1 requires that SA_1 has never queried the oracles *ExtractPartialSecret* and *SuperSign*. Based on the above three equations, β can derive the three unknown values x_t, r_t, and s, and outputs s as the solution of a random instance $(P, Q = s \cdot P)$ of ECDLP. So far, we have shown that β can solve the given instance of ECDLP. Next, we analyze β's success probability $Succ_\beta$ of winning in Game 1.

E_1: β does not abort in all of the *ExtractPartialSecret* queries.
E_2: SA_1 successfully forges a valid signature (ID_t, m_t, σ_t).
E_3: The forged signature (ID_t, m_t, σ_t) satisfies $ID_t = ID^*$.

The corresponding probabilities of the above three events are presented. That is, $\Pr[\mathrm{E}_1] \geq \left(1 - \frac{1}{q_{CU}+q_H}\right)^{q_{EPS}}$, $\Pr[\mathrm{E}_2|\mathrm{E}_1] \geq Succ_{SA_1}$ and $\Pr[\mathrm{E}_3|\mathrm{E}_1 \wedge \mathrm{E}_2] \geq \frac{1}{q_{CU}+q_H}$, where q_{CU}, q_H and q_{EPS} are the numbers of *CreateUser* queries, *Hash* queries and *ExtractPartialSecret* queries. In that case, the probability of β solving the given instance of ECDLP is $Succ_\beta = \Pr[\mathrm{E}_1 \wedge \mathrm{E}_2 \wedge \mathrm{E}_3] = \Pr[\mathrm{E}_1]\Pr[\mathrm{E}_2|\mathrm{E}_1]\Pr[\mathrm{E}_3|\mathrm{E}_1 \wedge \mathrm{E}_2] \geq \frac{1}{q_{CU}+q_H}\left(1 - \frac{1}{q_{CU}+q_H}\right)^{q_{EPS}} Succ_{SA_1}$. Clearly, β can solve ECDLP with a non-negligible probability $Succ_\beta$ because $Succ_{SA_1}$ is non-negligible. This contradicts the hardness of ECDLP. ■

Theorem 2. *The proposed certificateless signature scheme is existentially unforgeable against a super type II adversary in the random oracle model, assuming the hardness of solving ECDLP. That is, if there exists a super type II adversary SA_2 who can submit queries to random oracles and win in Game 2 with probability $Succ_{SA_2}$, then there is an algorithm β which can solve a random ECDLP instance in polynomial time with success probability $Succ_\beta \geq \frac{1}{q_{CU}+q_H}\left(1 - \frac{1}{q_{CU}+q_H}\right)^{q_{es}} Succ_{SA_2}$.*

Proof. We assume that there is a super type II adversary SA_2 breaking our proposed scheme with a non-negligible probability $Succ_{SA_2}$. Then we want to build a polynomial-time algorithm β which uses SA_2 to solve ECDLP. That is, β receives a random ECDLP instance $(P, Q = x_t \cdot P)$, with β's goal being to derive the secret x_t. Similarly, in the Initialization phase, β picks an identity ID^* as the challenged identity in Game 2, sets PK_{KGC} and sends master key s and $params = (G, P, PK_{KGC}, H)$ to SA_2. Meanwhile, β maintains two lists, i.e., L_{H_2} and L_{K_2}. Next, in the Query phase, β can issue the following oracle queries to SA_2. Here, we skip the same oracle queries as those, i.e., *CreateUser*, *Hash*, *ReplacePublicKey* and *SuperSign*, set out in Theorem 1. In addition, β simulates other oracle queries of SA_2 as follows:

➢ *RequestPublicKey*(ID_t): When SA_2 makes this query with an identity ID_t, β acts as follows:

(1) If $ID_t \neq ID^*$, β generates two random numbers $r_t, x_t \in Z_n^*$, and computes $R_t = r_t \cdot P$, $h_t = H(ID_t, PK_{KGC})$, $s_t = r_t + h_t \cdot s$ mod n and $PK_t = x_t \cdot P + R_t$. Then, β adds $\langle ID_t, R_t, h_t \rangle$, $\langle ID_t, s_t, R_t \rangle$ and $\langle ID_t, PK_t, x_t \rangle$ to the lists L_{H_1} and L_{K_1}, respectively. Finally, β returns PK_t to SA_2.

(2) Otherwise, β selects a random value $r_t \in Z_n^*$, and sets $R_t = r_t \cdot P$, $h_t = H(ID_t, PK_{KGC})$, $s_t = r_t + h_t \cdot s$ mod n and $PK_t = x_t \cdot P + R_t$. Then, β adds $\langle ID_t, R_t, h_t \rangle$, $\langle ID_t, s_t, R_t \rangle$ and $\langle ID_t, PK_t, \perp \rangle$ to the lists L_{H_1} and L_{K_1} respectively. Finally, β returns PK_t to SA_2.

- ➢ $ExtractPartialSecret(ID_t)$: When SA_2 makes this query with an identity ID_t, β looks for $\langle ID_t, s_t, R_t\rangle$ in L_{K_1}. If there exists a record of such a tuple, β returns s_t to SA_2; otherwise, β makes a *RequestPublicKey* query with ID_t and returns s_t to SA_2 accordingly.
- ➢ $ExtractSecret(ID_t)$: When SA_2 makes this query with an identity ID_t, β acts as follows:

  (1) If $ID_t = ID^*$, β terminates the session.

  (2) Otherwise, β looks for $\langle ID_t, PK_t, x_t\rangle$ in L_{K_1}. If there is such a record, β returns x_t to SA_2; otherwise, β makes a *RequestPublicKey* query with ID_t and then returns x_t to SA_2.

Finally, SA_2 outputs a forged but valid signature (ID_t, m_t, σ_t). If $ID_t \neq ID^*$, β stops the simulation. Otherwise, β looks for $\langle ID_t, s_t, R_t\rangle$ and $\langle ID_t, PK_t, x_t\rangle$ in the list L_{K_1}. Based on the forking lemma [20], if we have the polynomial replay of β with the same random tape and different choices of hash oracle, SA_2 can further generate another signature. Eventually, we have two valid signatures, i.e., ${\sigma_t}^{(j)} = ({T_t}^{(j)}, {\tau_t}^{(j)})$ with j = 1, 2, satisfying the equations, i.e., ${\tau_t}^{(j)} = {t_t}^{(j)} + {k_t}^{(j)} \cdot (x_t + {s_t}^{(j)}) = {t_t}^{(j)} + {k_t}^{(j)} \cdot (x_t + r_t + {h_t}^{(j)} \cdot s) \bmod n$, where j = 1, 2. Note that winning Game 2 requires that the oracles *ExtractSecret* and *SuperSign* had never been queried by SA_2. With the above two linear and independent equations, β can derive the two unknown values r_t and x_t, and outputs x_t as the solution of the random ECDLP instance $(P, Q = x_t \cdot P)$. We then analyze β's success probability $Succ_\beta$ of winning in Game 2. We present the events which result in β's success:

E_1: β does not abort in all of the *ExtractSecret* queries.

E_2: SA_2 successfully forges a valid signature (ID_t, m_t, σ_t).

E_3: The forged signature (ID_t, m_t, σ_t) satisfies $ID_t = ID^*$.

The probabilities of the following equations are presented. That is, $\Pr[\mathrm{E}_1] \geq \left(1 - \frac{1}{q_{CU}+q_H}\right)^{q_{ES}}$, $\Pr[\mathrm{E}_2|\mathrm{E}_1] \geq Succ_{SA_2}$, and $\Pr[\mathrm{E}_3|\mathrm{E}_1 \wedge \mathrm{E}_2] \geq \frac{1}{q_{CU}+q_H}$, where q_{CU}, q_H, and q_{ES} are the numbers of *CreateUser* queries, *Hash* queries and *ExtractSecret* queries. Hence, the probability of β solving the given instance of the ECDLP is $Succ_\beta = \Pr[\mathrm{E}_1 \wedge \mathrm{E}_2 \wedge \mathrm{E}_3] = \Pr[\mathrm{E}_1]\Pr[\mathrm{E}_2|\mathrm{E}_1]\Pr[\mathrm{E}_3|\mathrm{E}_1 \wedge \mathrm{E}_2] \geq \frac{1}{q_{CU}+q_H}\left(1 - \frac{1}{q_{CU}+q_H}\right)^{q_{es}} Succ_{SA_2}$. Now, β is able to solve ECDLP with a non-negligible probability $Succ_\beta$ because $Succ_{SA_2}$ is non-negligible. This contradicts the hardness of ECDLP. ■

5. System Implementation and Performance Evaluation

To evaluate the performance of the proposed certificateless scheme, we adopt two IoT-based testbeds, i.e., Arduino Uno and Raspberry PI 2 platforms, as the major evaluation platforms in the experiments. The Arduino Uno is a microcontroller board based on the ATmega328P, i.e., an 8-bit AVR RISC-based microchip with 32 KB EEPROM and 2 KB RAM. It is a tiny platform at very low cost, and thus is suitable to evaluate the performance of IoT-based schemes. On the other hand, the Raspberry PI is a card-sized single-board computer which offers an ARM GNU/Linux kernel and 1 GB RAM and 16 GB storage. Generally speaking, the Arduino Uno platform is usually simulated as a resource-constrained device while the Raspberry PI platform is simulated as a smart object which is more powerful on computation efficiency. Hence, in our experiment the Arduino Uno is adopted as resource-constrained objects in IoT networks and the Raspberry PI 2 platform is operated as smart objects (or the mobile IoT-based gateway associated with the resource-constrained objects). The implementation environment is outlined in Table 1. It is known that current techniques for solving ECDLP need $\mathrm{O}(\sqrt{n})$ steps, which depend on the size of the underlying field. NIST has recommended five levels of prime fields for certain prime n of sizes, i.e., 192, 224, 256, 384, and 512-bit [21] with associated and recommended elliptic curves. A prime field is the field $\mathbb{GF}(n)$, which contains a prime number n of elements, and the security strength of which is dependent on the length of the binary expansion of n. Normally, an elliptic curve over $\mathbb{GF}(n)$, where $n \approx 2^{256}$, can be contrasted with finite-field cryptography (e.g., DSA) with a 3072-bit public key and a 256-bit private key, and integer factorization cryptography (e.g., RSA) with a 3072-bit value of n. Therefore, to strike the best balance between protocol efficiency, security robustness and system scalability, the following two conditions are considered in our system implementation.

(1) Condition (1). For the Arduino Uno, we adopt elliptic curve points over a prime field $\mathbb{GF}(n)$ with a 192-bit prime n, a random number generator with a 96-bit output sequence and a secure one-way hash function, i.e., SHA-3 (512-bit) [22] as the underlying crypto-modules in our proposed certificateless scheme.
(2) Condition (2). For the Raspberry PI 2 platform, the elliptic curve is with a 384-bit prime n and the random number generator is with 96-bit output sequence. In addition, SHA-3 (512-bit) is implemented as the one-way hash function.

Table 1. Implementation environment.

Environment	Description
Arduino Uno	Atmel ATmega328P 8-Bit 16MHz AVR Architecture Memory 2 KB RAM/32 KB EEPROM
Raspberry PI 2	Broadcom BCM2836 @ 1 GHz Quad-Core ARM Cortex-A7 Architecture with 1 GB DDR2 RAM and SanDisk 16 GB Class 10 SD Card
Programming Language	(For Raspberry PI 2) Eclipse 3.8 with Oracle Java 8 ARM (For Arduino Uno) ANSI C
Crypto API	(For Raspberry PI 2) The Bouncy Castle Crypto APIs [23] (For Arduino Uno) Fackelmann/SHA3 [24], Kmackay/micro-ecc [25], AESLib [26]

Table 2 describes the computation cost of our proposed certificateless signature scheme implemented on the Arduino Uno with a 192-bit elliptic curve, a 96-bit random number generator and a 512-bit SHA-3, in terms of execution time of required computation components. In the pre-processing phase, we need 4.414 ms for generating four random numbers, 0.2 ms for computing $h_i = H(ID_i, PK_{KGC})$ via a SHA-3 operation with a 288-bit input sequence, 14.4 s for calculating four values PK_{KGC}, R_i, s_i and PK_i via ECC scalar multiplication operations, and 8.64 s for verifying the equation $s_i \cdot P = R_i + h_i \cdot PK_{KGC}$. The total computation cost of the pre-processing phase is 23.044 s. Next, during each normal operation of our proposed scheme, we require 11.537 s and 14.416 s for the sign phase and the verify phase, respectively. In the sign phase, we require 1.104 ms to generate a 96-bit t_i, and 2.88 s and 16.2 ms to compute T_i and k_i, respectively. Note that we assume that the size of the signed message m is 512-bit and, thus, the input sequence of k_i is 1408-bit. Finally, 8.64 s is needed to compute the signature value τ_i. On the other hand, in the verify phase, we need 16.4 ms to complete the executions of h_i and k_i, and 14.4 s to verify the equation, i.e., $\tau_i \cdot P = T_i + k_i \cdot (PK_i + h_i \cdot PK_{KGC})$. Thus, we require 25.953 s in total to execute the processes of our proposed certificateless signature scheme. According to the above simulation results, we can see that the practicability of the proposed scheme is not convinced. However, in a general IoT scenario, resource-constrained sensors usually perform simple task (or command), such as the sensing and transmission of environmental parameters. This kind of data is always meaningless when it is transmitted alone. Therefore, we argue that only reasonable security density is required to guarantee basic robustness. Based on our implementation results, we find that the execution of ECC scalar multiplication operations dominates the computation cost of the proposed scheme. For better performance efficiency, we suggest that the elliptic curve points with a 64/96/160-bit prime n, the 64/96-bit random number generator and the SHA-3 128/256-bit can be considered during the implementation of practical applications. Table 3 shows the implementation results of the experiment with the elliptic curve with a 160-bit prime n. If we adopt the elliptic curve with a 160-bit prime n, a 96-bit random number generator and a SHA-3 with 512-bit output, around 53% of computation cost can be deducted from the case with the 192-bit elliptic curve. That is, as shown in Table 3, we only require 10.812 s, 5.421 s, and 6.771 s for executing the pre-processing phase, the sign phase, and the verify phase of our proposed scheme, respectively. It is believed that the best balance of system robustness and performance efficiency can be achieved by appropriately adjusting the system parameters of the adopted crypto-modules. Furthermore, when higher security robustness is needed, the proposed certificateless signature method can be adopted to support a key exchange (or key agreement) process and produce a session key for later secure communication via symmetric encryption (e.g., the performance of AES implementation on Arduino Uno is shown in Table 4). It is obvious that both higher security and better performance can, thus, be delivered. In brief, for resource-constrained devices, we suggest to exploit our proposed certificateless signature mechanism with 160-bit elliptic curve to construct a robust key exchange (or key agreement) process, and enjoy the performance efficiency from the symmetric encryption with an exchanged (or agreed) session key while preserving the security. Note that the same security level can be achieved via 160-bit elliptic curve and 1028-bit RSA, respectively [27].

Table 2. The computation cost of our proposed certificateless signature scheme implemented on the Arduino Uno with Condition (1).

Phase	Computation Cost	Execution Time	Total
Pre-processing	Generate s, r_i, x_i, ID_i (96-bit)	4.414 ms	23.044 s
	Compute h_i (SHA-3 with 288 bit input sequence)	0.2 ms	
	Compute PK_{KGC}, R_i, s_i, PK_i (ECC 192-bit)	14.4 s	
	Verify $s_i \cdot P = R_i + h_i \cdot PK_{KGC}$ (ECC 192-bit)	8.64 s	
Sign	Generate t_i (96-bit)	1.104 ms	11.537 s
	Compute k_i (SHA-3 with 1408-bit input sequence) [1]	16.2 ms	
	Compute T_i (ECC with 192-bit)	2.88 s	
	Compute $\tau_i = t_i + k_i \cdot (x_i + s_i)$ (ECC 192-bit)	8.64 s	
Verify	Compute h_i (SHA-3 with 288-bit input sequence)	0.2 ms	14.416 s
	Compute k_i (SHA-3 with 1408-bit input sequence) [1]	16.2 ms	
	Verify $\tau_i \cdot P = T_i + k_i \cdot (PK_i + h_i \cdot PK_{KGC})$ (ECC 192-bit)	14.4 s	

[1] Suppose the size of message m is 512-bit.

Table 3. The computation cost of our proposed signature scheme implemented on the Arduino Uno with a 160-bit elliptic curve, a 96-bit random number generator, and a 512-bit SHA-3.

Phases of the Proposed Scheme	Total Execution Time
Pre-processing phase	10.812 s
Sign phase	5.421 s
Verify phase	6.771 s

Table 4. The computation cost of AES implemented on the Arduino Uno.

Input Sequence of AES	Encryption/Decryption
AES-128 with 32/64/128/256 Bytes Input Sequence	0.63 ms
AES-256 with 32/64/128/256 Bytes Input Sequence	0.87 ms

Similarly, Table 5 describes the computation cost of our proposed certificateless signature scheme implemented on the Raspberry PI 2 platform in which the elliptic curve points is with a 384-bit prime n, the random number generator is a 96-bit output sequence, and the one-way hash function is SHA-3 (512-bit). In the pre-processing phase, 0.276 ms is required for four random number generations, 0.0051 ms is required for computing a SHA-3 operation with a 480-bit input sequence, i.e., $h_i = H(ID_i, PK_{KGC})$, 0.355 ms is required for the calculation of four values, i.e., PK_{KGC}, R_i, s_i and PK_i via ECC scalar multiplication operations, and 0.213 ms is required for verifying the equation $s_i \cdot P = R_i + h_i \cdot PK_{KGC}$. In total, we need 0.895 ms to execute the pre-processing phase. Next, 1.549 ms and 1.556 ms are required for executing the sign phase and the verify phase, respectively. In the sign phase, we require 1.336 ms to generate a 96-bit t_i, and to compute T_i and k_i. Note that the input sequence of k_i is 1792-bit. Finally, 0.213 ms is needed for computing the value τ_i. In the verify phase, we need 1.2011 ms to compute h_i and k_i, and 0.355 ms to verify the equation, i.e., $\tau_i \cdot P = T_i + k_i \cdot (PK_i + h_i \cdot PK_{KGC})$. In brief, we require 3.105 ms in total to execute the processes of our proposed certificateless signature scheme.

Based on our implementation results, the performance bottleneck occurs at the execution of the SHA-3 hash function with a 1792-bit input sequence, i.e., about 77% ($\approx (1.196 \times 2)/(1.549 + 1.556)$) of total computation cost is dominated by this operation. Nevertheless, the computation cost of executing h_i, derived via a SHA-3 hash function with a 480-bit input sequence, is almost negligible when compared to the total computation cost. This observation inspires us to further investigate the performance evaluation of the SHA-3 hash function on the Raspberry PI 2 platform. From Table 6, we observe that the performance of SHA-3 hash function will degrade once the input sequence exceeds multiple of 576-bit, which is one of the defaulted block sizes of the SHA-3 hash function. In other words, it appears that SHA-3 hash function is more suitable for communication protocols with short messages. Normally, in a sensor-based IoT environment, communication messages operated by sensors cannot be too long, due to power consumption limitations. We, thus, argue that the proposed scheme is suitable for current IoT-based communication networks.

Table 5. The computation cost of our proposed certificateless signature scheme implemented on the Raspberry PI 2 with Condition (2).

Phase	Computation Cost	Execution Time	Total
Pre-processing	Generate s, r_i, x_i, ID_i (96-bit)	0.276 ms	0.895 ms
	Compute h_i (SHA-3 with 480-bit input sequence)	0.0051 ms	
	Compute PK_{KGC}, R_i, s_i, PK_i (ECC 384-bit)	0.355 ms	
	Verify $s_i \cdot P = R_i + h_i \cdot PK_{KGC}$ (ECC 384-bit)	0.213 ms	
Sign	Generate t_i (96-bit)	0.069 ms	1.549 ms
	Compute k_i (SHA-3 with 1792-bit input sequence) [1]	1.196 ms	
	Compute T_i (ECC with 384-bit)	0.071 ms	
	Compute $\tau_i = t_i + k_i \cdot (x_i + s_i)$ (ECC 384-bit)	0.213 ms	
Verify	Compute h_i (SHA-3 with 480-bit input sequence)	0.0051 ms	1.556 ms
	Compute k_i (SHA-3 with 1792-bit input sequence) [1]	1.196 ms	
	Verify $\tau_i \cdot P = T_i + k_i \cdot (PK_i + h_i \cdot PK_{KGC})$ (ECC 384-bit)	0.355 ms	

[1] Suppose the size of message m is 512-bit.

Table 6. The computation cost of SHA-3 with different length input sequences on Raspberry PI 2.

SHA-3 Operation	Execution Time
SHA-3 with 576-bit input sequence	0.412 ms
SHA-3 with 1152-bit input sequence	0.939 ms
SHA-3 with 1728-bit input sequence	1.194 ms
SHA-3 with 2304-bit input sequence	1.726 ms
SHA-3 with 2880-bit input sequence	2.260 ms
SHA-3 with 3456-bit input sequence	2.407 ms
SHA-3 with 4032-bit input sequence	2.807 ms
SHA-3 with 4608-bit input sequence	3.215 ms
SHA-3 with 5184-bit input sequence	4.084 ms
SHA-3 with 5760-bit input sequence	4.430 ms

Based on the above results, we find that there exists one limitation in our experiment. In order to examine the practicability of the proposed scheme, the experiment adopts the Arduino Uno and Raspberry PI 2 as the evaluation platforms. However, the adopted crypto-libraries are not consistent in which Bouncy Castle Crypto APIs [23] is adopted for the Raspberry PI 2, and Fackelmann/SHA3 [24], Kmackay/micro-ecc [25] and AESLib [26] are for the Arduino Uno. In general, the evaluation platforms with different processors certainly influence the performance. On the other hand, the crypto-library may also be elegantly-tuned to fit specific processors and gain better performance efficiency. In our experiment, the Bouncy Castle Crypto APIs are generic crypto-libraries for general processors and the others (i.e., Fackelmann/SHA3 [24], Kmackay/micro-ecc [25], and AESLib [26]) are well-configured for the feasible implementation on the Arduino Uno. The performance evaluation is, thus, not under the same evaluation criteria. Fortunately, the practicability and feasibility of the proposed certificateless signature scheme is demonstrated by the experiments. Nevertheless, this limitation existed. Therefore, we suggest that this limitation can be as one of the future research directions. In addition, to pursue the best balance between the performance efficiency and security robustness, we suggest that, in the resource-constrained objects, the proposed scheme with 160-bit elliptic curve can be exploited to construct a robust key exchange process and support secure communications. Tuning the ECC crypto-module with a 192-bit (or 224/256/384/512-bit) elliptic curve to fit the resource-constrained objects is suggested as another interesting future research direction.

6. Related Work

In recent years, designing certificateless signature schemeswithout bilinear pairings has been extensively studied due to its effectiveness in solving the key escrow problem in identity-based cryptography, and its potential for deployment in an environment comprising resource-limited mobile devices. In this section, we first present the state-of-the-art of certificateless signature before revealing a previously unknown weakness in a recent certificateless signature mechanism proposed by Wang et al. [28]. We then present a comparative summary of our proposed scheme and relevant schemes.

6.1. Review of Certificateless Signature Schemes

Since Al-Riyami and Paterson [18] first proposed certificateless public key cryptography in 2003 to solve the key-escrow problem in identity-based public key cryptography, certificateless cryptography has been widely investigated for different network types. Huang et al. [17], in 2007, refined the security model presented by Al-Riyami and Paterson, and introduced type I and type II adversaries with three different power levels, namely: a normal adversary, strong adversary, and super adversary. The authors then presented a robust scheme based on bilinear pairing, and proved the security of the scheme against type I and II adversaries. Later, Gong and Li [29] introduced a provably-secure certificateless signature scheme without the use of bilinear pairing. The authors claimed that their proposed scheme is more robust than previous schemes, in terms of resilience to super type I and II adversaries. While security proof in the random oracle was presented, Yeh et al. [30,31] pointed out that the scheme is vulnerable to super type I attacker, contrary to the claims. The authors then proposed a countermeasure for the identified attacks in which the robustness against super type I and II adversaries can be guaranteed. In a latter work, Wang et al. [28] re-designed the communication procedures of the certificateless signature mechanism proposed by Yeh et al. [30,31] to enhance the computation efficiency. Specifically, costs associated with ECC-based scalar multiplication and addition operations of points are removed. However, in the next subsection, we reveal that a malicious super type I adversary can easily forge a legitimate signature to cheat any receiver as he/she wishes in this scheme.

There have been other attempts to design lightweight certificateless signcryption schemes for low-cost sensors. For example, in a 2014 work, Shi et al. [32] proposed a certificateless signcryption scheme without bilinear pairing, and proved the security of the scheme against type I and II adversaries assuming the hardness of the discrete logarithm problem. Shingh et al. [14] and Sharma et al. [15] demonstrated a RSA-based certificateless signature scheme for wireless sensor networks, which attempted to integrate RSA cryptography in certificateless signature scheme for securing resource-limited sensors. However, an exponential multiplication operation under a discrete logarithm is less efficient than point multiplication operations on elliptic curves over $\mathbb{GF}(n)$ under the same security level [27]. Hence, there is potential to improve the performance in the schemes reported in [14,15,32] without invalidating the security claims. Pang et al. [33] presented a bilinear pairing-based certificateless signature scheme and proved its security in the standard model. However, the proposed scheme requires significant computation due to the inherent nature of bilinear pairing. In [16], Tsai proposed a certificateless short signature scheme using bilinear pairing. It was claimed that the proposed scheme is suitable for low-bandwidth communication environment (or power-constrained devices). However, the tradeoff between communication and computation costs is not rigorously investigated, in terms of power consumption of target devices. Hence, this claim is debatable. Moreover, the current sensor-based communication environment is not as bandwidth-limited compared to a decade ago, as we have previously discussed. Therefore, we posit that computational efficiency should be prioritized over communication efficiency when designing an efficient certificateless signature scheme for IoT-based smart objects.

6.2. Previously Unknown Weakness in Wang et al's (2015) CLS Scheme

We now revisit Wang et al.'s certificateless signature scheme [28], and demonstrate that the scheme is insecure against a super type I adversary.

- Revisiting the scheme:
  - ➢ In the Setup phase, KGC generates a group G of elliptic curve points with prime order n and determines a generator P of G, prior to randomly selecting a master secret key $s \in Z_p^*$ and computing the master public key $PK_{KGC} = s \cdot P$. Then, KGC chooses two secure hash functions $H_1 : \{0,1\}^* \times G \times G \to Z_q^*$ and $H_1 : \{0,1\}^* \times \{0,1\}^* \times G \times G \times G \times G \to Z_q^*$, and publishes a set of system parameters, i.e., $params = (G, P, PK_{KGC}, H_1, H_2)$.
  - ➢ In the PartialPrivateKeyExtract phase, given $params$, s and the user i's identity ID_i, KGC selects a random number $r_i \in Z_n^*$, and computes $R_i = r_i \cdot P$, $h_i = H_1(ID_i, R_i, PK_{KGC})$ and $s_i = r_i + h_i \cdot s \bmod n$. Next, KGC returns the partial private key $D_i = (s_i, R_i)$ to the user i. Upon receiving D_i, i is able to verify D_i by examining whether two values, i.e., $s_i \cdot P$ and $R_i + h_i \cdot PK_{KGC}$, are identical or not since $s_i \cdot P = (r_i + h_i \cdot s) \cdot P = R_i + h_i \cdot PK_{KGC}$.
  - ➢ In the SetSecretValue phase, given $params$, the user i randomly selects $x_i \in Z_n^*$ as his/her secret value.
  - ➢ In the SetPublicKey phase, given $params$ and x_i, the user i computes his/her public key as $PK_i = x_i \cdot P$.
  - ➢ In the Sign phase, given $params$, D_i, x_i and a message m, the user i selects a random value $t_i \in Z_n^*$, and outputs a signature $\sigma_i = (R_i, T_i, \tau_i)$ with a series of computed values $T_i = t_i \cdot P$, $k_i = H_2(ID_i, m, T_i, PK_i, R_i, PK_{KGC})$ and $\tau_i = t_i + k_i \cdot x_i + s_i \bmod n$.
  - ➢ In the Verify phase, Given $params$, ID_i, PK_i, m and $\sigma_i = (R_i, T_i, \tau_i)$, the verifier computes $h_i = H_1(ID_i, R_i, PK_{KGC})$ and $k_i = H_2(ID_i, m, T_i, PK_i, R_i, PK_{KGC})$, and then checks whether the equation $\tau_i \cdot P = T_i + k_i \cdot PK_i + R_i + h_i \cdot PK_{KGC}$ holds. Note that σ_i is accepted if the equation holds. That is, $\tau_i \cdot P = (t_i + k_i \cdot x_i + s_i) \cdot P = t_i \cdot P + k_i \cdot x_i \cdot P + s_i \cdot P = t_i \cdot P + k_i \cdot x_i \cdot P + (r_i + h_i \cdot s) \cdot P = T_i + k_i \cdot PK_i + R_i + h_i \cdot PK_{KGC}$.
- Cryptanalysis
  - ➢ Suppose there exists a malicious super type I adversary j which seeks to forge a valid signature $\sigma_i = \left(R_i, T_i', \tau_i'\right)$ on a message m' chosen by the adversary j. The adversary j eavesdrops a valid signature $\sigma_i = (R_i, T_i, \tau_i)$ with message m issued by the user i from any previous session, where $T_i = t_i \cdot P$, $R_i = r_i \cdot P$, $PK_i = x_i \cdot P$, $PK_{KGC} = s \cdot P$, $h_i = H_1(ID_i, R_i, PK_{KGC})$, $k_i = H_2(ID_i, m, T_i, PK_i, R_i, PK_{KGC})$, $s_i = r_i + h_i \cdot s \bmod n$, and $\tau_i = t_i + k_i \cdot x_i + s_i \bmod n$.
  - ➢ Since the adversary j is a super type I adversary, j is able to issue an oracle query of *ExtractSecretValue*(i) and replace any entity's public key including KGC's public key. With the eavesdropped values, i.e., T_i, R_i and τ_i, and public values, i.e., PK_i and PK_{KGC}, the adversary j chooses a random number $t_a \in Z_n^*$, and derives $T_a = t_a \cdot P$,

$T_i' = T_a + T_i$, $k_i' = H_2(ID_i, m', T_i', PK_i, R_i, PK_{KGC})$ and $\tau_i' = \tau_i - k_i \cdot x_i + k_i' \cdot x_i + t_a = (t_i + k_i \cdot x_i + s_i) - k_i \cdot x_i + k_i' \cdot x_i + t_a = (t_i + t_a) + k_i' \cdot x_i + s_i \bmod n$. Note that the secret x_i is retrieved via $ExtractSecretValue(i)$ oracle query.

➢ So far, the adversary j can forge a valid signature $\sigma_i' = (R_i, T_i', \tau_i')$ on the chosen message m'. It is obvious that the equation $\tau_i' \cdot P = [(t_i + t_a) + k_i' \cdot x_i + s_i] \cdot P = (t_i + t_a) \cdot P + k_i' \cdot x_i \cdot P + (r_i + h_i \cdot s) \cdot P = (T_a + T_i) + k_i' \cdot PK_i + r_i \cdot P + h_i \cdot s \cdot P = T_i' + k_i' \cdot PK_i + R_i + h_i \cdot PK_{KGC}$ holds. Therefore, the resistance to signature forgery attack cannot be guaranteed under the assumption of existing a malicious super type I adversary.

6.3. Security and Performance Comparative Summary

We now benchmark the security and performance of the proposed certificateless signature with those of Gong and Li [29], Wang et al. [28] and Tsai [16]. From Table 7, we observe that our proposed scheme and Tsai's scheme [16] enjoy the same security level–resilience to super type I and II adversaries. However, Gong and Li's scheme [29] still suffers from vulnerability to signature forgery attack via super type I adversary [30] as does Wang et al.'s scheme [28], as presented in Section 6.2.

A comparative summary of performance efficiency is presented in Table 8, where the evaluation metrics are of the inverse operation (T_{inv}), bilinear pairing operation (T_{bp}), ECC-based scalar multiplication operation for points (T_{em}), ECC-based addition operation for points (T_{eadd}), multiplication operation (T_m), addition operation (T_{add}), one-way hash function (T_h), and random number generator operation (T_g). It is clear that our proposed scheme outperforms Gong and Li's scheme [29] and Wang et al.'s scheme [28] by eliminating the computation costs of ($1T_m$, $1T_h$, $2T_{eadd}$, $2T_h$) and ($1T_{eadd}$), respectively. When compared to Tsai's scheme [16], the tradeoff between the computation cost ($1T_m$, $1T_{add}$, $1T_h$) and ($1T_{inv}$, $2T_{bp}$) is observed. It is clear that bilinear pairing operation is more inefficient than ECC point-based operations, i.e., scalar multiplication and addition. Hence, we can claim that our proposed scheme is more efficient and practical than Tsai's scheme [16] with a better performance efficiency.

Table 7. A comparative summary: security.

	Gong & Li's Scheme [29]	**Wang et al's Scheme [28]**	**Tsai's Scheme [16]**	**Our proposed Scheme**
Resistance to Super Type I Adversary	No	No	Yes	Yes
Resistance to Super Type II Adversary	Yes	Yes	Yes	Yes

Table 8. A comparative summary: performance.

	Sign Phase	**Verify Phase**	**In Total**
Gong & Li's scheme [29]	$1T_{em} + 2T_m + 2T_{add} + 2T_h + 1T_g$	$4T_{em} + 3T_{eadd} + 3T_h$	$5T_{em} + 2T_m + 3T_{eadd} + 2T_{add} + 5T_h + 1T_g$
Wang et al's scheme [28]	$1T_{em} + 1T_m + 2T_{add} + 1T_h + 1T_g$	$3T_{em} + 3T_{eadd} + 2T_h$	$4T_{em} + 1T_m + 3T_{eadd} + 2T_{add} + 3T_h + 1T_g$
Tsai's scheme [16]	$1T_{inv} + 1T_{em} + 1T_m + 1T_{add} + 1T_h$	$2T_{bp} + 2T_{em} + 2T_{eadd} + 2T_h$	$1T_{inv} + 2T_{bp} + 3T_{em} + 1T_m + 2T_{eadd} + 1T_{add} + 3T_h$
Our proposed scheme	$1T_{em} + 1T_m + 2T_{add} + 1T_h + 1T_g$	$3T_{em} + 2T_{eadd} + 2T_h$	$4T_{em} + 1T_m + 2T_{eadd} + 2T_{add} + 3T_h + 1T_g$

7. Conclusions

In this paper, we presented a new certificateless signature scheme for IoT-based smart objects. We proved the security of the proposed scheme against the super type I and II adversaries, as well as demonstrating the utility of the scheme in IoT-oriented testbeds. For passive objects with constrained computation ability and limited power capability, we argued that the proposed certificateless signature scheme with 160-bit elliptic curve can be exploited to construct a key exchange (or key agreement) process with a reasonable security robustness. Around 5.421 s and 6.771 s are required for performing the sign phase and the verify phase of our proposed scheme, respectively. For active objects with powerful computation efficiency, we suggested considering the proposed certificateless signature scheme with at least 384-bit elliptic curve and SHA-3 (512-bit) to pursue the highest security due the affordability of computation cost on the Raspberry PI platform. Findings from the implementation showed that low computation cost, i.e., 1.549 ms and 1.556 ms, is required to perform the execution processes of the sign phase and verify phase, respectively. Moreover, we compared the security and performance of our scheme with those of Gong and Li [29], Wang et al. [28] and Tsai [16], as well as revealing a previously unknown vulnerability in Wang et al.'s scheme [28] (where a malicious super type I adversary can easily forge a valid signature on any message and cheat receivers at will).

Acknowledgments: This work was supported in part by the Academia Sinica, in part by the Taiwan Information Security Centre, and in part by the Ministry of Science and Technology, Taiwan under grant MOST 105-2221-E-259-014-MY3, grant MOST 105-2221-E-011-070-MY3, grant MOST 105-2923-E-182-001-MY3, grant MOST 104-2218-E-001-002, and grant MOST 105-2218-E-001-001. In addition, Chunhua Su is supported by JSPS KAKENHI Grant Number 15K16005.

Author Contributions: Kuo-Hui Yeh, Chunhua Su and Kim-Kwang Raymond Choo are responsible for the design and the proof of the proposed certificateless signature scheme. In addition, Kuo-Hui Yeh and Wayne Chiu are responsible for the implementation the proposed scheme.

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

References

1. Yang, Y.; Lu, J.; Choo, K.K.R.; Liu, J. On Lightweight Security Enforcement in Cyber-physical Systems. In Proceedings of the International Workshop on Lightweight Cryptography for Security & Privacy (LightSec 2015), Bochum, Germany, 10–11 September 2015.
2. D'Orazio, C.; Choo, K.-K.R. Circumventing iOS Security Mechanisms for APT Forensic Investigations: A Security Taxonomy for Cloud Apps. *Future Gener. Comput. Syst.* **2016**. [CrossRef]
3. D'Orazio, C.; Choo, K.-K.R. A Technique to Circumvent SSL/TLS Validations on iOS Devices. *Future Gener. Comput. Syst.* **2016**. [CrossRef]
4. D'Orazio, C.; Choo, K.-K.R.; Yang, L.T. Data Exfiltration from Internet of Things Devices: iOS Devices as Case Studies. *IEEE Internet Things J.* **2017**, *4*, 524–535. [CrossRef]
5. Yao, X.; Han, X.; Du, X.; Zhou, X. A lightweight multicast authentication mechanism for small scale IoT applications. *IEEE Sens. J.* **2013**, *13*, 3696–3701. [CrossRef]
6. Kawamoto, Y.; Nishiyama, H.; Kato, N.; Shimizu, Y.; Takahara, A.; Jiang, T. Effectively collecting data for the location-based authentication in the Internet of Things. *IEEE Sens. J.* **2015**. [CrossRef]
7. Hernandez-Ramos, J.L.; Pawlowski, M.P.; Jara, A.J.; Skarmeta, A.F.; Ladid, L. Toward a lightweight authentication and authorization framework for smart objects. *IEEE J. Sel. Areas Commun.* **2015**, *33*, 690–702. [CrossRef]
8. Gope, P.; Hwang, T. BSN-Care: A Secure IoT-based modern healthcare system using body sensor network. *IEEE Sens. J.* **2016**, *16*, 1368–1376. [CrossRef]
9. Yang, Y.; Cai, H.; Wei, Z.; Lu, H.; Choo, K.-K.R. Towards Lightweight Anonymous Entity Authentication for IoT Applications. In Proceedings of the 21st Australasian Conference on Information Security and Privacy-CISP 2016, Melbourne, Australia, 4–6 July 2016; pp. 265–280.
10. Nguyen, K.; Ouahla, N.; Laurent, M. Lightweight Certificateless and Provably-Secure Signcryptosystem for the Internet of Things. In Proceedings of the 14th IEEE International Conference on Trust, Security and Privacy in Computing and Communications (IEEE TrustCom-15), Helsinki, Finland, 20–22 August 2015.
11. Toorani, M.; Beheshti, A.A. An elliptic curve-based signcryption scheme with forward secrecy. *J. Appl. Sci.* **2009**, *9*, 1025–1035. [CrossRef]
12. Dutta, M.; Singh, A.K.; Kumar, A. An Efficient Signcryption Scheme based on ECC with Forward Secrecy and Encrypted Message Authentication. In Proceedings of the IEEE 3rd International Advance Computing Conference (IACC), Ghaziabad, India, 22–23 February 2013.
13. Yu, G.; Yang, H.; Fan, S.; Han, W. Efficient Certificateless Signcryption Scheme from Weil Pairing. *J. Netw.* **2011**, *6*, 1280–1287. [CrossRef]
14. Singh, J.; Kumar, V.; Kumar, R. An RSA based Certificateless Signature Scheme for Wireless Sensor Networks. In Proceedings of the 2015 International Conference on Green Computing and Internet of Things (ICGCIoT), Noida, India, 8–10 October 2016.
15. Sharma, G.; Bala, S.; Verma, A.K. An improved RSA Based Certificateless Signature Scheme for Wireless Sensor Networks. *Int. J. Network Secur.* **2016**, *18*, 82–89.

16. Tsai, J.-L. A New Efficient Certificateless Short Signature Scheme Using Bilinear Pairings. *IEEE Syst. J.* **2015**. [CrossRef]
17. Huang, X.; Mu, Y.; Susilo, W.; Wong, D.S.; Wu, W. Certificateless signature revisited. In Proceedings of the 12th Australasian Conference on Information Security and Privacy (ACISP), Townsville, Australia, 2–4 July 2007; pp. 308–322.
18. Al-Riyami, S.; Paterson, K. Certificateless public key cryptography. In Proceedings of the International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT), Taipei, Taiwan, 30 November–4 December 2003; pp. 452–473.
19. Huang, X.; Mu, Y.; Susilo, W.; Wong, D.S.; Wu, W. Certificateless Signatures: New Schemes and Security Models. *Comput. J.* **2012**, *55*, 457–474. [CrossRef]
20. Pointcheval, D.; Stern, J. Security Proofs for Signature Schemes. In Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT), Saragossa, Spain, 12–16 May 1996; pp. 387–398.
21. FIPS PUB 186–4, Digital Signature Standard (DSS), National Institute of Standards and Technology, June 2009. Available online: http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf (accessed on 1 May 2017).
22. Dworkin, M.J. SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions, NIST FIPS-202. August 2015. Available online: http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.202.pdf (accessed on 24 December 2016).
23. The Bouncy Castle Crypto APIs. 2016. Available online: https://www.bouncycastle.org/ (accessed on 24 December 2016).
24. Fackelmann/SHA3. Available online: https://github.com/Fackelmann/SHA3 (accessed on 20 March 2017).
25. Kmackay/micro-ecc. Available online: https://github.com/kmackay/micro-ecc (accessed on 20 March 2017).
26. AESLib. Available online: https://github.com/DavyLandman/AESLib (accessed on 21 April 2017).
27. Maletsky, K. RSA vs ECC Comparison for Embedded Systems (White Paper), Atmel. Available online: http://www.atmel.com/Images/Atmel-8951-CryptoAuth-RSA-ECC-Comparison-Embedded-Systems-WhitePaper.pdf (accessed on 18 January 2017).
28. Wang, L.; Chen, K.; Long, Y.; Mao, X.; Wang, H. A Modified Efficient Certificateless Signature Scheme without Bilinear Pairings. In Proceedings of the 2015 International Conference on Intelligent Networking and Collaborative Systems (INCOS), Taipei, Taiwan, 2–4 September 2015. [CrossRef]
29. Gong, P.; Li, P. Further Improvement of a Certificateless Signature Scheme without Pairing. *Int. J. Commun. Syst.* **2014**, *27*, 2083–2091. [CrossRef]
30. Yeh, K.-H.; Tsai, K.-Y.; Kuok, R.-Z.; Wu, T.-C. Robust Certificateless Signature Scheme without Bilinear Pairings. In Proceedings of the International Conference on IT Convergence and Security (ICITCS 2013), Macau, China, 16–18 December 2013.
31. t Yeh, K.-H.; Tsai, K.-Y.; Fan, C.-Y. An Efficient Certificateless Signature Scheme without Bilinear Pairings. *Multimed. Tools Appl.* **2015**, *74*, 6519–6530. [CrossRef]
32. Shi, W.; Kumar, N.; Gong, P.; Zhang, Z. Cryptanalysis and Improvement of a Certificateless Signcryption Scheme without Bilinear Pairing. *Front. Comput. Sci.* **2014**, *8*, 656–666. [CrossRef]
33. Pang, L.; Hu, Y.; Liu, Y.; Xu, K.; Li, H. Efficient and Secure Certificateless Signature Scheme in the Standard Model. *Int. J. Commun. Syst.* **2017**, *30*. [CrossRef]