sensors

MDPI

Article

An Aggregate Signature Scheme Based on a Trapdoor Hash Function for the Internet of Things

Hong Shu [1,2,3], Fulong Chen [1,2,*], Dong Xie [1,2], Liping Sun [1,2], Ping Qi [3] and Yongqing Huang [3,4]

1 School of Computer and Information, Anhui Normal University, Wuhu 241002, China; shuhongtl@126.com (H.S.); xiedong@ahnu.edu.cn (D.X.); slp620@163.com (L.S.)
2 Anhui Provincial Key Lab of Network and Information Security, Wuhu 241002, China
3 School of Mathematics and Computer, Tongling University, Tongling 244061, China; qiping929@gmail.com (P.Q.); hyq@tlu.edu.cn (Y.H.)
4 Institute of Information Technology & Engineering Management, Tongling University, Tongling 244061, China
* Correspondence: long005@mail.ahnu.edu.cn; Tel.: +86-553-591-0371

Received: 5 August 2019; Accepted: 27 September 2019; Published: 29 September 2019

Abstract: With the rapid development of the Internet of Things (IoT), it becomes challenging to ensure its security. Identity authentication and integrity verification can be achieved by secure hash functions and digital signature algorithms for IoT applications. In order to solve the issues of bandwidth limitation and computational efficiency of secure communication in IoT applications, an aggregate signature scheme based on multi- trapdoor hash function is proposed in this paper. Firstly, to prevent key exposition, based on the elliptic curve discrete logarithm problem (ECDLP), we constructed a double trapdoor hash function (DTH) and proved its reliability. Secondly, the multi-trapdoor hash function (MTH) based on DTH is presented. Finally, an MTH-based aggregate signature scheme (MTH-AS) with constant signature length is proposed. Based on the assumption of ECDLP, the proposed scheme is proven unforgeable against adaptive chosen message attacks with the Forking Lemma. Different from the most signature schemes with bilinear mapping, the proposed scheme has higher computational efficiency and shorter aggregate signature length. Moreover, it is independent of the number of signers. Security analysis and performance evaluation has revealed that the proposed scheme is an ideal solution for secure IoT applications with limited computing power, storage capacity, or limited bandwidth, such as wireless sensor networks, vehicular ad hoc networks, or healthcare sensor networks.

Keywords: Internet of Things (IoT); aggregate signature; trapdoor hash function; elliptic curve discrete logarithm; random oracle model

1. Introduction

With the development of wireless communication technology, sensor network, microchip technology, and pervasive computing, the Internet of Things (IoT) has been applied in more and more areas, including smart home, smart health, wearable equipment, vehicular ad-hoc network, environmental monitoring, and smart grids, etc. [1]. The IoT collects various information from the physical world through radio frequency identification (RFID) devices, infrared sensors, GPS, sensors, etc. It enables people-to-people, people-to-thing, thing-to-thing connection, and communication, which in turn realizes intelligent perception, recognition, decision-making, and control on the physical world [2]. For example, wearable systems can effectively provide patients with seamless monitoring, remote surgery, telemedicine, timely assistance, and turn hospital-centered medical services into

patient-centered care [3]. However, while the Internet of Things provides people with more and more convenient services, its security issues are becoming increasingly prominent. Data security and privacy preservation are major challenges in the applications of the IoT. Integrity, non-repudiation and authenticity have become the key security requirements for the Internet of Things [4].

In IoT applications, identity authentication, encryption, and integrity verification can be achieved by secure hash functions and digital signature algorithms which can ensure data privacy and rooting security [2]. Many scholars have been working on this issue [4–6]. Yeh et al. [5] proposed a certificateless signature scheme based on elliptic curve cryptography (ECC). Since ECC is more efficient than bilinear mapping in terms of computational efficiency [7], the scheme provides a safe and efficient interaction for IoT-based smart objects. When tackling the computational efficiency problem, Kumar et al. [6] did not select bilinear pairing operations either. Instead, they proposed a lightweight digital signature scheme based on quadratic residual theory and proved the security of the scheme under the standard model. The solution, based on the intrusion detection system (IDS), authenticates the crowdsensing data acquired from IoT environment. Meanwhile, to meet security requirements of data integrity and authenticity in IoT environment, Yang et al. [4] proposed a certificateless signature scheme. Based on the collision resistant hash function and computational Diffie-Hellman (CDH) assumption, Yang's scheme was proved to be highly unforgeable under adaptive chosen message attacks in the standard model.

Computing power, battery capacity, and storage resources are the important factors that limit IoT capabilities [2]. In IoT, such as SIoT [8–10] and WIoT [3], hundreds and thousands smart objects are connected. A large number of applications are many-to-one. That is, multiple data senders and one data receiver. Figure 1 shows the many-to-one IoT scenario. In the scenario, smart objects generate or collect information from the physical world and pass data from one node to the other. Finally, the receiver aggregates IoT data and sends it to data center. In the case of smart health, a patient can generate multiple medical records. Some medical records, blood pressure, blood sugar, heart rate, etc., are generated from wearable devices. Other medical records such as medical orders and prescriptions, are from medical personnel. Considering the integrity and authenticity of medical data, each individual medical record should be signed by the corresponding responsible entity. Consequently the digital signatures on different medical data will map to one patient (many-to-one). Aggregate signature can compress the digital signatures of different messages into one short digital signature, thus saving storage space and improving computational efficiency. It is very suitable for IoT applications such as vehicular ad-hoc network, smart grid and wireless sensor network with limited bandwidth and computational resources. This nature of aggregate signatures has attracted interests of more and more scholars [11–13]. Pankaj Kumar et al. [11] proposed a certificateless aggregate signature scheme for medical wireless sensor networks. This scheme features certificateless cryptosystem and aggregate signature, and preserves privacy, non-repudiation and integrity of medical wireless sensor networks. In regard to vehicle-to-infrastructure (V2I) communication in vehicular ad hoc networks, Horng et al. [12] proposed a certificateless aggregate signature. It implements conditional privacy preservation through pseudo identity. Another aggregate signature scheme was proposed by Shen et al. [13] for wireless sensor networks based on identity-based cryptography. That solution has been proved resistant to coalition attacks, capable to reduce energy consumption and ensure security for data acquisition, processing and transmission in wireless sensor networks.

The motivation of our study is to find a solution to identity authentication and integrity verification in many-to-one IoT scenarios where device resources are limited, such as smart health, vehicular ad hoc networks and smart grid et al. In these applications, traditional encryption, authentication and cryptographic algorithms can severely reduce the efficiency of small embedded devices and increase their power consumption. However, secure hash function and lightweight aggregate signature algorithm can effectively solve the problem of battery capacity, computing power and storage capacity limitation while fulfilling security requirements. Therefore, we propose an aggregate signature scheme based on multi-trapdoor hash function in this paper.

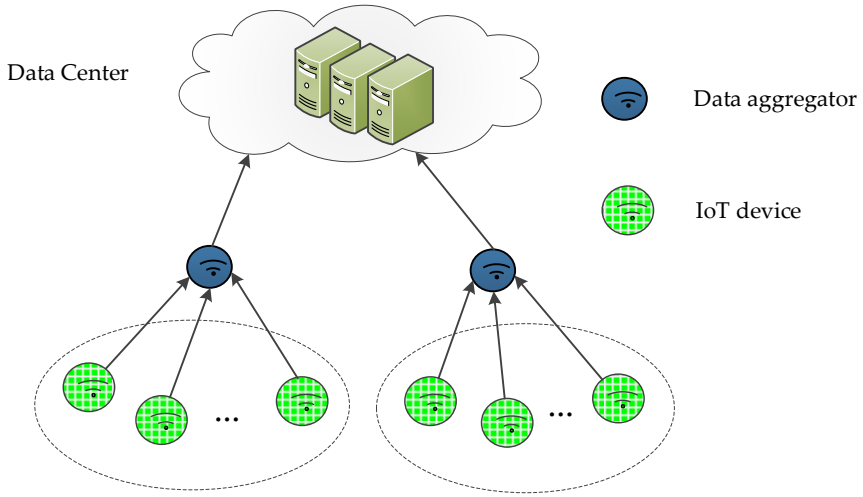

Figure 1. Many-to-one IoT.

The contributions of this paper are as follows:

- Based on ECDLP, we constructed a double trapdoor hash function and a multi-trapdoor hash function respectively. Batch trapdoor collision computation of multi-trapdoor hash function can improve the efficiency of aggregate signature.
- An aggregate signature scheme based on MTH is proposed. With Forking Lemma, the proposed scheme is proven to be secure against the existing unforgeability on adaptively chosen message attacks.
- Compared with other bilinear pairings-based schemes, our ECC-based scheme is more efficient in terms of computational overhead. On the other hand, our MTH-AS scheme has the advantage in storage capacity because the length of the proposed aggregate signature is a constant.
- Due to the above performance, our MTH-AS scheme is suitable for secure IoT applications with limited computing power, storage capacity, and bandwidth.

The rest of this paper is organized as follows. Section 2 discusses the relevant works. The necessary preliminaries and system model are given in Section 3. Section 4 presents the ECDLP-based double trapdoor hash scheme DTH. Section 5 describes the ECDLP-based multi-trapdoor hash function MTH. Thereafter, we demonstrate the MTH- based aggregate signature scheme for IoTs and carry out performance comparison in Section 6. Finally, the conclusion is offered in Section 7.

2. Related Work

Hash functions are one-way and collision resistant. Being a special type of hash function, trapdoor hash function [14] is related to the concept of trapdoor commitment [15]. The trapdoor hash function uses some special information (the trapdoor information) to produce a fixed hash value. People who know the trapdoor information open the trapdoor commitment in different ways, thus opening different collisions [15]. That means, the owner of the trapdoor can calculate the trapdoor collision.

Krawczyk et al. [14] first proposed the trapdoor hash function in order to construct chameleon signatures. Thereof, many digital signature schemes were developed based on chameleon signatures. One of the most representative schemes was a solution proposed by Shamir et al. [16] for online/offline signatures. The scheme could resist the adaptive chosen message attacks. However, it encountered key exposure problems of the chameleon hash. That is because collision calculation would lead to exposure of trapdoor information. Focusing on solving this problem, Chen et al. [17] and Atteniese et al. [18] proposed trapdoor hash schemes without key exposure. In 2008, Chen et al. [19] introduced a special double trapdoor hash scheme. This scheme features two trapdoors, long-term trapdoor and temporary trapdoor. It guarantees the safety of long-term trapdoor at the cost of temporary trapdoor leakage.

Chandrasekhar and Singhal et al. [20–23] carried out in-depth study on trapdoor hash function. Based on discrete logarithm, Chandrasekhar et al. [20] proposed a multi-trapdoor hash function without key exposure. This scheme had multiple trapdoors corresponding to different entities. It was suitable for constructing multi-party digital signatures. Chandrasekhar et al. [22] put forward the concept of aggregate signcryption. They proposed a new efficient scheme of aggregate signcryption which could generate the aggregate signcryption text of constant order. The new scheme combined multi-trapdoor hash function with decomposable multiplicative homomorphism ElGamal encryption while providing confidentiality, integrity and identity authentication for many-to-one communication scenarios.

The conception of aggregate signature was first proposed by Boneh et al. [24] in 2003, which has played a significant role in promoting digital signature cryptography technology. Based on trapdoor permutations, sequential aggregate signature was proposed by Lysyanskaya et al. [25] in 2004. In this scheme, before adding his own signature, every signer has to verify all the previously aggregated signatures. It is, however, not suitable for the situation where the signers operate independently of each other. Accordingly, Brogle et al. [26] improved it with the idea of "lazy verification". Their scheme does not require the signer to know the public key of other signers, but the length of the signature grows linearly when the number of signers increases. In order to reduce the interaction between signers, a synchronous aggregation signature scheme with a synchronous clock was proposed by Ahn et al. [27]. The scheme allows a signer to sign at most once in each period of time, and only the signatures in the same period can be aggregate. However, the computation cost is relatively high.

The identity-based aggregation signature scheme proposed by Gentry et al. [28] does not require to store the public key of each signer. The purpose is to minimize the total amount of information for verification. However, it requires an additional trusted third party (e.g., key generation center).

Certificateless aggregation signature has the characteristics of keyless escrow in certificateless cryptosystem and relatively low computation and communication overhead in aggregation signatures. That's why some relevant scholars have made in-depth research on it [29,30]. In 2007, Gong et al. [29] firstly proposed two certificateless identity-based aggregation signature schemes. But there were shortcomings in respect of signature length and verification efficiency. Zhang et al. [30] introduced a certain improvement in computation efficiency. In Zhang's scheme, the verification process, not reliant on the number of aggregate signatures, requires a small set of a constant number of pairing computations. However, the generation of aggregate signatures requires assistance of a synchronous clock.

With the Forking Lemma [31], the scheme proposed by Chen et al. [32] was proven strong security based on the hardness of computational Diffie-Hellman problem (CDHP). It makes use of the bilinear pair and state information. However, the length of signature grew with the number of signers. The scheme proposed by Li et al. [33] drew on the state information of Chen et al. [32]. It was existentially unforgeable against adaptively chosen message attacks without the Forking Lemma. The scheme provides fixed-length aggregate signatures. Zhou et al. [34] and Cheng et al. [35] effectively compensated for the shortcomings of the above two schemes [32,33] by using elliptic curve discrete logarithm problem (ECLDP). Zhou et al. [34] proposed two certificateless aggregate signatures CLAS-1 and CLAS-2, which were proven unforgeable by the discrete logarithm problem (DLP). Compared with CLAS-1, information sharing was used in CLAS-2 to aggregate partial signatures in advance. CLAS-2 provides shorter constant-level signature lengths than CLAS-1. Cui et al. [36] applied the certificateless aggregation signature scheme to vehicle ad hoc network, and used pseudo-identity to provide privacy preservation for vehicle information. The scheme has high computation efficiency. However, the length of signature is related to the number of signers.

Among the above-mentioned schemes, the aggregate signature lengths of the schemes [26,32,35,36], CAS-1 [29], and CLAS-II [34] increase linearly with the number of signers and they are only suitable for low bandwidth network environments. Meanwhile, the schemes [24,29,32] based on bilinear pairs having no advantage in the computational performance because the time overhead of bilinear pair operations is relatively high [7]. The comparisons of relevant aggregate signatures are shown in Table 1.

Table 1. The comparison of relevant aggregate signatures.

Scheme	Bilinear Pair	Constant Signature Length	Hardness Problem
Boneh [24]	Yes	No	Co-CDH
Gong-1 [29]	Yes	Yes	CDHP
Gong-2 [29]	Yes	No	CDHP
Chen [32]	Yes	Yes	CDHP
Zhou-I [34]	No	Yes	DLP
Zhou-II [34]	No	No	DLP
Cheng [35]	Yes	Yes	CDHP
Cui [36]	No	Yes	ECDLP,CDHP
Our proposed scheme	No	Yes	ECDLP

3. Preliminaries

3.1. Symbolic Representation

The symbols used in the proposed scheme are shown in Table 2.

Table 2. Symbol description.

Symbol	Interpretation
$\|x\|$	Length of binary string x
$t \in_R V$	t is uniformly distributed in V
$\{x_i\}_m^n$	$\{x_m, x_{m+1}, \ldots, x_n\}$
$\{x \to \overline{x}\}_m^n \backslash T$	$\{x_m, \ldots, x_{i-1}, \overline{x}_i, x_{i+1}, \ldots, x_n\}$, where $\forall i \in T, T \subset \{m, n\}$, $x_i = \overline{x}_i$, other values of x_i remain unchanged. For example, $\{x_i \to \overline{x}_i\}_m^n \backslash \{p, q\}$ represents $\{x_m, \ldots, x_{p-1}, \overline{x}_p, x_{p+1}, \ldots, x_{q-1}, \overline{x}_q, x_{q+1}, \ldots, x_n\}$.

3.2. Double Trapdoor Hash Function

Different from other hash functions, trapdoor hash function is a probability function with the hash key and the trapdoor key $< HK, TK >$. The collision resistance of the trapdoor hash functions depends on the user's knowledge state [16]. When both TK and HK are known, it is easy to calculate the trapdoor collision. That is to say, when only HK is known, it is difficult to find two different messages M and M' in the message space, and two different auxiliary parameters R and R', which satisfy $TH_{HK}(M, R) = TH_{HK}(M', R')$. However, when both TK and HK are known, it is easy to calculate R' based on M, M' and R such that $TH_{HK}(M, R) = TH_{HK}(M', R')$.

In trapdoor hash function, the calculation of the collision causes the trapdoor key to leak, which is called key exposure. In chameleon signature, the exposure of the trapdoor key affects its transitivity. In online/offline signature scheme, key exposure will result in anyone being able to forge the signature. A hash function without key exposure usually has two trapdoors, i.e., a long-term trapdoor and a temporary trapdoor. Collision calculations only expose the temporary trapdoor, thus preventing the long-term trapdoor key from leaking.

Different from the traditional trapdoor hash function family [16], the trapdoor hash function [20] proposed in this paper is a variant of the double trapdoor hash function family [21], using the temporary key pair $< HK', TK' >$ to generate trapdoor collisions.

Definition 1. *Trapdoor hash function consists of four probability polynomial time (PPT) algorithms* ***< ParGen, KeyGen, HashGen, TrapColGen >****.*

- ***ParGen****: Inputs security parameter k, outputs system parameters Params;*
- ***KeyGen****: Inputs Params, outputs < HK, TK >;*
- ***HashGen****: Inputs Params, message M, and auxiliary parameter R, and outputs the trapdoor hash value* $TH_{HK}(M, R)$*;*

- ***TrapColGen***: *Inputs Params, < HK, TK >, M, R, and new message M′(≠M), and outputs new auxiliary parameter R′ and the temporary hash key HK′ such that:*

$$TH_{HK}(M,\ R) = TH_{HK'}\ (M',\ R');$$

When HK ≠ HK′, < HK′, TK′ > and < HK, TK > are called temporary hash/trapdoor key pair and long-term hash/trapdoor key pair respectively. The properties of double trapdoor hash functions are as follows:

(1) Validity: Given HK and (M, R), $TH_{HK}(M,\ R)$ is calculated in polynomial time.

(2) Collision resistance: Given HK, there is no PPT algorithm that can find HK′ which satisfies:

$$TH_{HK}(M,\ R) = TH_{HK'}\ (M',\ R'),\ M' \neq M.$$

(3) Trapdoor collision: There is a PPT algorithm, given < HK, TK >, (M, R) and new message M′ ≠ M, output HK′ and R′ such that:

$$TH_{HK}(M,\ R) = TH_{HK'}\ (M',\ R').$$

(4) Key exposure freeness: Given the long-term hash key HK′, and (M, R), (M′, R′), M′ ≠ M, there is no PPT algorithm to output long-term trapdoor key TK with non-negligible probability.

3.3. Elliptic Curve Discrete Logarithm

Definition 2. *(Elliptic curve discrete logarithm problem (ECDLP) [37]). $E(F_l)$ is an elliptic curve over the finite field F_l. And P is a q-order generator of $E(F_l)$, when $Q \in E(F_l)$ and Q = kP, find the integer k (0 ≤ k ≤ q−1).*

This definition is also known as the onewayness of ECDLP. The probability that algorithm $\mathcal{A}$ successfully solves ECDLP is defined as:

$$Adv_A^{ECDLP}(\varphi) = Pr[A(\ q,\ P,\ Q) = \ k | 0\ <\ k\ \leq\ q-1,\ Q\ =\ kP]$$

It is determined by the random selection of $k\ \in_R\ Z_q^*$ and $\mathcal{A}$.

3.4. Aggregate Signature

Aggregate signatures consist of PPT algorithms: **AS = < Setup, KeyGen, Sign, Verify, Aggregate, Aggregate Verify >**. And the tuple **< Setup, KeyGen, Sign, Verify >** constructs a standard system parameter establishment, key generation, signature, verification of the short signature process, called the standard signature of aggregate signature.

- **Setup**: Inputs security parameter k, outputs system parameter *Params*.
- **KeyGen**: For a particular $ID_i \in U$ (U is a user set), inputs system parameter *Params*, then outputs the private and public key $< y, Y >$.
- **Sign**: For a message M_i to be signed, inputs private key y_i, outputs individual signature σ_i.
- **Verify**: Inputs public key Y_i, message M_i, and individual short signature σ_i, if the verification algorithm is successful, it outputs *ACCEPT*, otherwise, it outputs *REJECT*.
- **Aggregate**: Inputs $\{ID_i\}_1^n \in U$, their signature messages $\{M_i\}_1^n$ and individual signatures $\{\sigma_i\}_1^n$, outputs aggregate signature σ.
- **Aggregate Verify**: Inputs public keys $\{Y_i\}_1^n$, messages $\{M_i\}_1^n$, and aggregate signature σ, if the aggregation validation algorithm is successful, it outputs *ACCEPT*, otherwise, it outputs *REJECT*.

3.5. Security Model

Assuming k is a security parameter, $G_A^{MTH\_AS}(1^k)$ is a game between challenger $\mathcal{B}$ and adversary $\mathcal{V}$. The attack model is shown below:

- *Setup* Inputs the security parameter k, $\mathcal{B}$ runs the *Setup* algorithm and returns the system parameter to $\mathcal{V}$.
- *Query*
  $\mathcal{V}$ adaptively performs the following oracle query.
  - Hash queries: $\mathcal{V}$ makes hash oracle queries to all hash functions in the proposed scheme, and challenger $\mathcal{B}$ returns the corresponding value.
  - Trapdoor hash queries: $\mathcal{V}$ inputs $< m, r >$ for trapdoor hash query and the oracle outputs $TH_Y\ (m,\ r)$.
  - Key queries: $\mathcal{V}$ inputs the message m_i of user i to make key query, and the oracle returns the trapdoor key y of user i to the adversary $\mathcal{V}$.
  - Signature queries: $\mathcal{V}$ inputs the original message/random value pair $<\ m_i\ ,\ r_i >$, new message m_i' and hash key TK_i, the oracle outputs the signature.
- *Forge*
  Finally, $\mathcal{V}$ outputs $\sigma^* =\ (\ K^*,\ C^*)$ as a forged aggregate signature based on new message set $\left\{m_i'^*\right\}_1^n$.
- The adversary $\mathcal{V}$ wins the game if σ^* is a valid signature and $\mathcal{V}$ does not make a key query on at least one user among n users.

3.6. System Model

In many-to-one IoT scenarios where bandwidth, computing power, and storage resources are limited, it is important to improve computational efficiency and storage capacity. Furthermore, it is also vital to protect data from modification and repudiation. Due to its natural compression properties, aggregate signatures are ideal for resource-constrained many-to-one IoT applications. As shown in Figure 2, the system model of the aggregated signature in the IoT environment proposed in this paper consists of five components: the key generation center (KGC), IoT devices, data aggregator, verifier and data center.

- *KGC*

The KGC is responsible for system setup. It is regarded to be trusted in our proposed scheme. The KGC generates system parameters and sends them to all the entities, such as IoT devices, aggregator, verifier and data center. The private keys sk_i are computed by the KGC for each IoT device. Then these private keys are sent to each entity through a secure channel.

- *IoT Devices*

The IoT devices with limited computational and storage capacity are capable to collect real data from the physical world. In order to ensure data integrity, non-repudiation, privacy, and authenticity, with the system parameter and the private key, each IoT device makes individual signature on the original data they collect. Then the IoT devices send message m_i, individual signature σ_i and public key pk_i to the data aggregator.

- *Data aggregator*

The data aggregator may be a node in the system model that verifies all the individual signatures it receives. It checks the validity of the individual signatures, if they are correct, then aggregates them into a single short signature. Finally, the data aggregator sends the aggregate signature to the verifier.

- *Verifier*

The verifier are responsible to check the correctness of the received aggregate signature. It can verify the correctness of all individual signatures by one operation. If the aggregation signature is verified correctly, all the messages and the aggregate signature are sent to the data center.

- *Data Center*

The data center has powerful storage space and computing power, which can store and share the validated aggregate signatures and original messages safely.

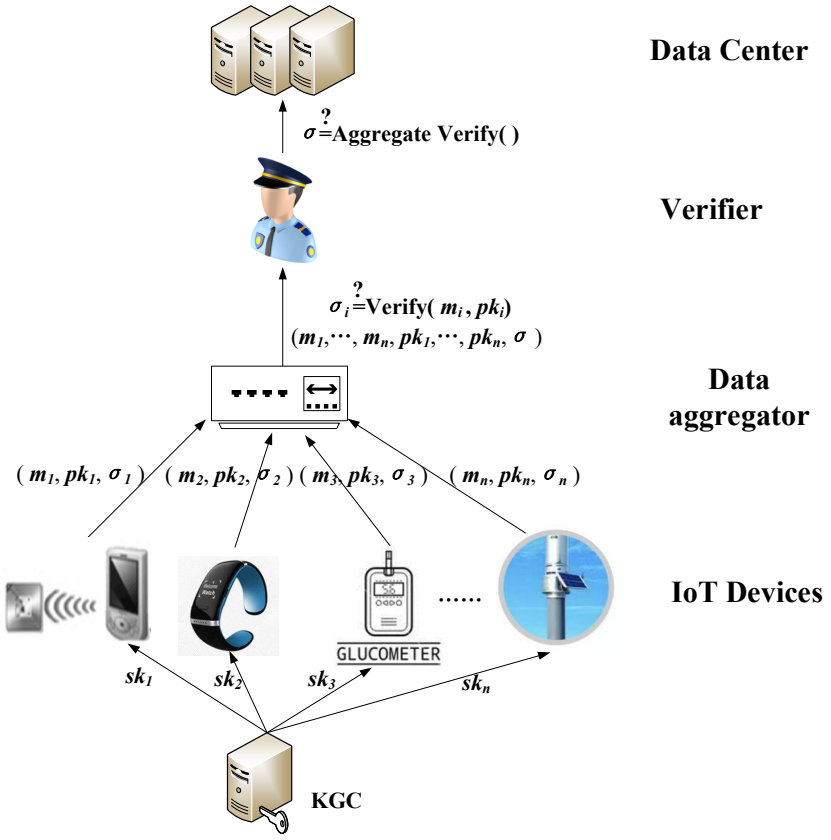

Figure 2. System model.

4. Scheme of Double Trapdoor Hash Function

4.1. Double Trapdoor Hash Scheme Based on ECDLP

In this section, a scheme of double trapdoor hash function based on ECDLP is presented, which is consisted of the tuple: **DTH = < DParGen, DKeyGen, DHashGen, DTrapColGen >**.

- **DParGen**: Select l and a big prime p, where $l = p^m$. Let $E(F_l)$ be an elliptic curve over finite field F_l and G a cyclic subgroup of $E(F_l)$. Let P be a generator of G with prime order q and $H : \{0, 1\}^* \rightarrow Z_q^*$, $f : G \times Z_q^* \times G \rightarrow Z_q^*$, $F : G \rightarrow Z_q^*$ cryptographic hash functions. The system parameters are $params = < G, P, q, H, F, f >$.
- **DKeyGen**: Select randomly $y \in Z_q^*$ and compute $Y = yP$. The trapdoor key is y and the hash key is Y.
- **DHashGen**: Select randomly $t \in Z_q^*$, compute $A = tP$ and $r = F(A)$. The trapdoor hash value is $h = H(m)P + rY$.
- **DTrapColGen**: Select randomly $t' \in Z_q^*$, compute:

$$A' = t' P \text{ and } r' = F(A').$$

The temporary trapdoor key is $y' = r'^{-1} (H(m) - H(m') + ry) \bmod q$ and the temporary hash key is $Y' = y' P$. Compute:

$$k = t' - y * f (h , r' , Y') \bmod q.$$

which then outputs $< k , r', Y' >$. $< k , r' >$ is the signature on $TH_{HK}(m, r) = TH_{HK'} (m', r')$ verifiable under Y [20]. The verification equation expands:

$$\begin{aligned} & kP + f(h, r', Y')Y \\ &= (t' - y * f(h, r', Y'))P + f(h, r', Y') * yP \\ &= t'P \\ &= A' \\ & F(A') = r' \end{aligned}$$

4.2. Security Analysis

(1) *Efficiency*: Given the system parameter *params*, the hash key Y and the message/auxiliary parameter pair $< m, r >$, the trapdoor hash valve $h = H(m)P + rY$ is computable in PPT.

(2) *Trapdoor collisions*: Given $< y, Y >$, $< m, r >$ and new message $m' (\neq m) \in \{0,1\}^*$, choose randomly $t' \in Z_q^*$. Then compute

$$A' = t'P, r' = F(A').$$

The temporary trapdoor key is given by

$$y' = r'^{-1}(H(m) - H(m') + ry) \bmod q$$

which satisfies

$$H(m)P + rY = H(m')P + r'Y'.$$

That is to say

$$H(m) + ry = H(m') + r'y'.$$

(3) *Key exposure freeness*: Given two tuples $< Y, m, r >$ and $< Y', m', r' >$ such that:

$$TH_Y(m, r) = TH_{Y'}(m', r').$$

That is to say:

$$H(m) + ry = H(m') + r'y'.$$

In the equation, the long-term trapdoor key y is not computable because y' is unknown. That is, the computation complexity of y' is equivalent to ECDLP because y' is solved by $Y' = y'P$.

(4) *Collision resistance*: The PPT collision forger $\mathcal{E}$ is assumed to resist the DTH scheme with a non-negligible probability. Given *params* and HK, $\mathcal{E}$ runs in polynomial time and outputs $< m, r, m', r', HK', k' >$ with non-negligible probability where the following statements hold:

$$\begin{aligned} & TH_{HK}(m, r) = TH_{HK}(m', r'), \\ & F(k'P + f(h, r', Y')Y) = r', \\ & m' \neq m, HK' \neq HK \text{and } r' \neq r \end{aligned}$$

Suppose $\mathcal{E}$ can construct a PPT algorithm Q for solving ECDLP. Given an instance of ECDLP $< G, P, q, Y >$, Q needs to find a value $z \in Z_q^*$ so that $Z = zP$. The hash function f acts as a random oracle O_f that Q simulates. That means Q provides a random value for each new query to answer any hash query of O_f. Then Q gives two identical answers if the same query is asked twice. Q runs an instance of $\mathcal{E}$ and answers any hash query of O_f until $\mathcal{E}$ produces collision forgery. When $\mathcal{E}$ queries $< TH_Y(m, r), r', Y' >$ to O_f, Q answers x'. With the Oracle replay attack [38], Q rewinds $\mathcal{E}$ to the point when $\mathcal{E}$ queries $< TH_Y(m, r), r', Y' >$ to O_f, and select randomly a new value $x'' \neq x' \in_R Z_q^*$ as the answer to $\mathcal{E}$. Q continues running $\mathcal{E}$ until producing another collision forgery $< m_2, r_2, m'_2, r'_2, Y'', k'' >$. Each instance of $\mathcal{E}$ is randomly

selected. Given $TH_{HK}(m_1,\ r_1)$, $TH_{HK'}\ (m_2,\ r_2)$, $m_1\ \neq\ m_2, r_1\ \neq\ r_2, m_1\ \neq\ m'_1, r_1\ \neq\ r'_1$, which satisfy the following equations:

$$\begin{cases} TH_{HK}(m_1,\ r_1) = TH_{HK'}(m'_1,\ r'_1\ ) \\ TH_{HK}(m_2,\ r_2) = TH_{HK'}(m'_2,\ r'_2\ ) \end{cases}$$
$$\begin{cases} k' = t - y * x' \\ k'' = t - y * x'' \end{cases}$$

According to these two equations, the following can be computed:

$$y = (k'' - k')(x' - x'')^{-1}.$$

This is contrary to the elliptic curve discrete logarithm hypothesis.

5. Multi-Trapdoor Hash functions based on ECDLP

The multi-trapdoor hash function [20] contains many participants $U_1, \ldots, U_n$, each of them having its own trapdoor/hash key pair $\{TK_i\ ,\ HK_i\ \}_1^n$ and original message $\{m_i\ \}_1^n$. It generates the objective multi-trapdoor hash value h according to $\{m_i,\ r_i\ \}_1^n$ and $\{\ HK_i\ \}_1^n$, then the participants can produce hash collisions with h based on new messages $\{m'_i\ \}_1^n$. The multi-trapdoor hash function combines multiple collisions generated by multiple participants to generate a single collision [20], thus saving storage space and bandwidth effectively.

5.1. Formal Definition

The multi-trapdoor hash function is composed of tuples **< MParGen, MKeyGen, MHashGen, MTrapColGen >**.

- **MParGen**: Inputs security parameter k, outputs system parameter *Params*.
- **MKeyGen**: Inputs system parameter *Params*, each participant $U_i\ (i\ \in\ [1,\ n])$ outputs $<\ TK_i\ ,\ HK_i\ >(i\ \in\ [1,\ n])$.
- **MHashGen**: Inputs *Params*, hash key group $\{\ HK_i\ \}_1^n$, message/auxiliary parameter pairs $\{m_i,\ r_i\ \}_1^n$, outputs multi-trapdoor hash value $TH_{\{\ HK_i\ \}_1^n}\left(\{\ m_i,\ r_i\ \}_1^n\right)$.
- **MTrapColGen**: Inputs *Params*, a trapdoor key TK_j, message/auxiliary parameter pairs $\{m_i,\ r_i\ \}_1^n$ and a new message $m_j\ \neq\ m'_j$, outputs collision parameter $<\ r'_j\ ,\ HK'_j\ >$ which satisfies the following equation:

$$TH_{\{\ HK_i\ \}_1^n}\left(\{m_i\ ,\ r_i\ \}_1^n\right) = TH_{\{\ \ HK_i\ \rightarrow\ HK'_i\ \}_1^n\ \setminus\{\ j\ \}}\left(\{m_i\ \rightarrow\ m'_i,\ \ r_i\ \rightarrow\ r'_i\ \}_1^n \setminus \{\ j\ \}\right)$$

5.2. The ECDLP-Based Multi-Trapdoor Hash Function

This section presents an ECDLP-based multi-trapdoor hash function MTH. The algorithm process describes as follows:

- **MParGen**: Similar to *DParGen* in Section 4.
- **MKeyGen**: For each participant U_i, select randomly the long-time trapdoor key $y_i\ \in\ Z_q^*$ and compute long-time hash key:

$$Y_i = y_i P.$$

  then outputs $\{y_i\ ,\ Y_i\ \}_1^n$.
- **MHashGen**: For each U_i, select randomly $t_i\ \in\ Z_q^*$, compute auxiliary parameters:

$$A_i = t_i\ P \text{ and } r_i = F(\ A_i\ ).$$

Trapdoor hash value is $h_i = H(m_i)P + r_iY_i$. Finally, aggregate all the trapdoor hash values as multi-trapdoor hash value:

$$h = \sum_{i=1}^{n} h_i$$

then outputs h.

- **MTrapColGen**: For each U_i, select randomly $t'_i \in Z_q^*$ and compute new auxiliary parameters:

$$A'_i = t'_i P \text{ and } r'_i = F(A'_i).$$

According to trapdoor collision, compute temporary trapdoor/hash key:

$$y'_i = r'^{-1}_i (H(m_i) - H(m'_i) + r_i y_i) \bmod q$$
$$Y'_i = y'_i P.$$

5.3. Security Analysis

Theorem 1. *The proposed multi-trapdoor hash function scheme is collision resistant.*

Proof. The PPT collision forger $\mathcal{E}$ is assumed to resist the MTH scheme with a non-negligible probability. Suppose $\mathcal{E}$ can construct a PPT algorithm Q for solving ECDLP. Given an instance of ECDLP $< G, P, q, Y >$, Q needs to find a value $z \in Z_q^*$ so that $Z = zP$.

Q runs an instance of $\mathcal{E}$ and answers any hash query of O_f until $\mathcal{E}$ produces collision forgery. When $\mathcal{E}$ queries $< TH_{Y_i}(m_i, r_i), r'_i, Y'_i >$ to O_f, Q answers x'_i. With the Oracle replay attack [38], Q rewinds $\mathcal{E}$ to the point when $\mathcal{E}$ queries $< TH_{Y_i}(m_i, r_i), r'_i, Y'_i >$ to O_f, and select randomly a new value $x''_i \neq x'_i \in_R Z_q^*$ as the answer to $\mathcal{E}$. Q continues running $\mathcal{E}$ until producing another collision forgery $< m_{i,2}, r_{i,2}, m'_{i,2}, r'_{i,2}, Y''_i, k''_i >$. Each instance of $\mathcal{E}$ is randomly selected. Given $TH_{Y_i}(m_{i,1}, r_{i,1}), TH_{Y_{i'}}(m_{i,2}, r_{i,2})$, $m_{i,1} \neq m_{i,2}$, $r_{i,1} \neq r_{i,2}$, $m_{i,1} \neq m'_{i,1}$, $r_{i,1} \neq r'_{i,1}$, which satisfy the following equations.

$$\begin{cases} TH_{Y_i}(m_{i,1}, r_{i,1}) = TH_{Y_{i'}}(m'_{i,1}, r'_{i,1}) \\ TH_{Y_i}(m_{i,2}, r_{i,2}) = TH_{Y_{i'}}(m'_{i,2}, r'_{i,2}) \end{cases}$$
$$\begin{cases} k'_i = t_i - y_i * x'_i \\ k''_i = t_i - y_i * x''_i \end{cases}$$

According to these two equations, the following can be computed

$$y_i = (k''_i - k'_i)(x'_i - x''_i)^{-1}.$$

It is contrary to the elliptic curve discrete logarithm hypothesis. Thus, the proposed MTH scheme is collision resistant. □

6. Aggregate Signature Scheme Based on MTH

6.1. The Aggregate Signature Scheme Based on MTH

This section presents an aggregate signature scheme based on MTH, called MTH-AS. The algorithm is presented below.

- **AParGen**: Similar to *DParGen* in Section 4.
- **AKeyGen**: For each participant U_i, select randomly the long-time trapdoor key $y_i \in Z_q^*$ and compute long-time hash key:

$$Y_i = y_i P.$$

then outputs $\{y_i, Y_i\}_1^n$.

- **AHashGen**: For each U_i, select randomly $t_i \in Z_q^*$, compute auxiliary parameters:

$$A_i = t_i P \text{ and } r_i = F(A_i).$$

and computes the trapdoor hash value:

$$h_i = H(m_i)P + r_i Y_i.$$

Finally, aggregate all the trapdoor hash values as the multi-trapdoor hash value:

$$h = \sum_{i=1}^{n} h_i$$

Then outputs h.

- **ATrapColGen**: For each U_i, select the latest timestamp $t'_i \in Z_q^*$ and compute new auxiliary parameters:

$$A'_i = t'_i P \text{ and } r'_i = F(A'_i).$$

According to trapdoor collision, compute temporary trapdoor/hash key:

$$y'_i = r'^{-1}_i (H(m_i) - H(m'_i) + r_i y_i) \bmod q$$
$$Y'_i = y'_i P.$$

Computes:

$$k_i = t_i' - y_i * f(h, r_i', Y_i') \bmod q.$$

and generates U_i's individual signature:

$$\sigma_i = (r_i', k_i).$$

outputing $\{\sigma_i, Y_i', A_i', r_i', t_i'\}_1^n$.

- **Verify**: This algorithm verifies the correctness of the individual signature of U_i, computing:

$$B_i = k_i P + f(h, r_i', Y_i')Y_i.$$

If the equation $F(B_i) = r_i'$ holds, it accepts the participant's individual signature and outputs *ACCEPT*, otherwise, outputs *REJECT*.

- **AggSign**: For each participant U_j whose individual signature is accepted, computing:

$$K = K + k_j \bmod q$$
$$C = C + A_j' \bmod q.$$

outputting the aggregate signature:

$$\sigma = (K, C).$$

- **AggVerify**: Let m be the number of participants in the aggregate signature, that is, the number of individual signatures accepted, computing:

$$B = KP + \sum_{j=1}^{m} f(h, r_j', Y_j')Y_j$$

If the equation $F(B) = F(C)$ holds, accepts the aggregate signature and outputs *ACCEPT*, otherwise, outputs *REJECT*.

6.2. The Correctness of Aggregate Verify

The aggregate verify equation expands as follows:

$$\begin{aligned} B &= KP + \sum_{j=1}^{m} f(h, r_j', Y_j')Y_j \\ &= \sum_{j=1}^{m}(t_j' - y_j f(h, r_j', Y_j'))P + \sum_{j=1}^{m} f(h, r_j', Y_j')\, y_j P \\ &= \sum_{j=1}^{m} t_j' P \\ &= \sum_{j=1}^{m} A_j' \\ &= C \end{aligned}$$

6.3. Security Proof

Theorem 2. *Given an adversary makes at most q_f f-hash queries, q_H H-hash queries, q_S signature queries, q_F F-queries, q_K key queries, q_T trapdoor hash queries within a period t in the random oracle model, and wins the game $G_A^{MTH\_AS}(1^k)$ with an non-negligible probability ε, that is, successfully forging the signature of an MTH_AS scheme. Then an algorithm $\mathcal{B}$ can be performed in polynomial time $t' \leq t + O(q_k + 2q_T + 2q_s)T_{ME}$, and solve an instance of ECLDP with probability $\varepsilon' \geq \frac{1}{10^6 q_k \sqrt{q_f}}\varepsilon$. Let T_{ME} be the run time for scalar multiplication in elliptic curve.*

Proof. Given an instance of ECLDP $(P, yP) \epsilon G$, the goal of the algorithm $\mathcal{B}$ is to compute y. Assume the hash key of m^* is yP. The following is a detailed interaction process between algorithm $\mathcal{B}$ and adversary $\mathcal{V}$.

- *Setup*: Challenger $\mathcal{B}$ inputs security parameters 1^k, runs algorithm **APar**$\mathbf{Gen}$, generates system parameters *params*, and sends *params* to adversary $\mathcal{V}$. $\mathcal{B}$ simulates hash functions random oracle D_H, D_F and D_f, key random oracle D_K, trapdoor hash random oracle D_T and signature oracle D_S to answer all the queries from adversary $\mathcal{V}$. $\mathcal{B}$ needs to maintain 6 lists (L_H, L_F, L_f, L_T, L_K, L_S), whose initial values are empty.
- *Query*: $\mathcal{V}$ adaptively performs the following oracle queries.
  - When $\mathcal{V}$ queries D_H with m_i, $\mathcal{B}$ checks whether existing $(m_i, H_i) \in L_H$ or not, if so, $\mathcal{B}$ sends H_i to $\mathcal{V}$. Otherwise, $\mathcal{B}$ selects a random $H_i \in Z_q^*$, sends H_i to $\mathcal{V}$ and saves (m_i, H_i) into the hash list L_H.
  - When $\mathcal{V}$ queries D_F with A_i, $\mathcal{B}$ checks whether existing $(A_i, F_i) \in L_F$ or not, if so, $\mathcal{B}$ returns F_i to $\mathcal{V}$. Otherwise, selects a random $F_i \in Z_q^*$, returns F_i to $\mathcal{V}$ and saves (A_i, F_i) into the hash list L_F.
  - When $\mathcal{V}$ queries D_f with (m_i, h, r_i', Y_i'), $\mathcal{B}$ checks whether existing $(m_i, h, r_i', Y_i', f_i) \in L_f$ or not, if so, $\mathcal{B}$ returns f_i to $\mathcal{V}$. Otherwise, selects a random $f_i \in Z_q^*$, returns f_i to A and saves $(m_i, h, r_i', Y_i', F_i)$ into the hash list L_f.
  - When $\mathcal{V}$ queries D_T with (m_i, r_i), $\mathcal{B}$ checks whether existing $(m_i, r_i, h_i) \in L_T$ or not, if so, $\mathcal{B}$ returns h_i to $\mathcal{V}$. Otherwise, selects a random $y_i \in Z_q^*$ and computes:

    $$h_i = H(m_i)P + r_i y_i P.$$

    returning h_i to $\mathcal{V}$ and saves (m_i, r_i, h_i) into the hash list L_T.
  - When $\mathcal{V}$ queries D_K with m_i, $\mathcal{B}$ checks whether existing $(m_i, Y_i, y_i) \in L_K$ or not, if so, $\mathcal{B}$ returns y_i to $\mathcal{V}$. Otherwise, if $m_i \neq m^*$, selects randomly $y_i \in Z_q^*$, $k_i \in Z_q^*$ and makes a query to D_f. If $(m_i, *, *, *, f_i) \in L_f$, computes ${f_i}^{-1}$. Otherwise, selects randomly ${f_i}^{-1} \in Z_q^*$ and stores $(m_i, *, *, *, f_i)$ into L_f, computing:

    $$y_i = (t_i' - k_i)\, {f_i}^{-1} \text{ and } Y_i = y_i P.$$

then returns (y_i, Y_i) to $\mathcal{V}$ and saves (m_i, Y_i, y_i) into the hash list L_K. If $m_i = m^*$ holds, the game is over and outputs ∇.

– When $\mathcal{V}$ queries D_S with (TK_i, m_i, m_i'), $\mathcal{B}$ checks whether existing $(m_i, m_i', \sigma_i) \in L_S$ or not, if so, $\mathcal{B}$ returns σ_i to $\mathcal{V}$, otherwise, selects a random $t_i' \in Z_q^*$ and computes:

$$\begin{aligned} A'_i &= t'_i P, \\ r'_i &= F(A'_i), \\ y'_i &= r'^{-1}_i (H(m_i) - H(m'_i) + r_i y_i) \bmod q, \\ Y'_i &= y'_i P, \\ k_i &= t_i' - y_i f(h, r_i', Y_i') \bmod q \end{aligned}$$

Then, $\mathcal{B}$ generates individual signature (r_i', k_i) and returns to $\mathcal{V}$.

- *Forge*: After polynomial bounded queries, the attacker $\mathcal{V}$ outputs the aggregate signature $\sigma^* = (K^*, C^*)$ of the new message set $\{m_i'^*\}_1^n$ under the condition of the user's long-term hash key set $\{Y_i^*\}_1^n$ and the original message/auxiliary parameter set $\{m_i^*, r_i^*\}_1^n$, and satisfies the following two conditions at the same time:

– $C = KP + \sum_{i=1}^m f_i Y_i$;

– There is at least a message m_i ($m*$) to which neither a key query nor an individual signature query is performed.

According to the Forking Lemma [31], the attacker $\mathcal{V}$ simply replaces the hash function f with $\tilde{f}$, a new valid forged signature $\tilde{\sigma} = (\tilde{K}^*, \tilde{C}^*)$ is obtained. When $j \in \{1, 2, \ldots, n\} \backslash \{s\}$, there is $s \in \{1, 2, \ldots, n\}$ such that $f_j^* = \tilde{f}_j^*$. When the following equations holds:

$$j = s,\ f(h^*, r_j'^*, Y_j'^*) = f_j^* \neq \tilde{f}_j^* = \tilde{f}(h^*, r_j'^*, Y_j'^*)$$

we can obtain the following equation set:

$$\begin{aligned} C^* &= K^* P + \sum_{i=1}^m f_i^* Y_i \\ \tilde{C}^* &= \tilde{K}^* P + \sum_{i=1}^n \tilde{f}_i^* Y_i \end{aligned}$$

At the same time, the following calculation is available:

$$\begin{aligned} &K^* - \tilde{K}^* + \sum_{i=1}^n (f_i^* y_i - \tilde{f}_i^* y_i) = \sum_{i=1}^n (t_i^* - \tilde{t}_i^*) \\ &K^* - \tilde{K}^* + (f_s^* - \tilde{f}_s^*) y_s = \sum_{i=1}^n (t_i^* - \tilde{t}_i^*) \\ &y_s = (f_s^* - \tilde{f}_s^*)^{-1} (\sum_{i=1}^n (t_i^* - \tilde{t}_i^*) + \tilde{K}^* - K^*) \end{aligned}$$

The probability of successful breaking of ECDLP by $\mathcal{B}$ is converted into the following three events:

(1) E_1 represents that algorithm $\mathcal{B}$ does not terminate at the query stage;
(2) E_2 indicates that $\mathcal{V}$ successfully forged aggregate signature; and
(3) E_3 indicates the successful application of Forking Lemma.

The probability of solving the ECDLP by algorithm $\mathcal{B}$ is as follows:

$$\begin{aligned} &Pr(E_1 \cap E_2 \cap E_3) = Pr(E_1) Pr(E_2 | E_1) Pr(E_3 | E_1 \cap E_2) \\ &Pr(E_1) \geq \frac{1}{q_k} \\ &Pr(E_2 | E_1) \geq \varepsilon \end{aligned}$$

According to the Forking Lemma [31], we can obtain the following equation:

$$Pr(E_3|\, E_1 \cap E_2) \geq \frac{1}{10^6 \sqrt{q_f}}$$
$$\varepsilon' = Pr(E_1 \cap E_2 \cap E_3) \geq \frac{1}{10^6 q_k \sqrt{q_f}} \varepsilon$$

The running time of $\mathcal{B}$ is equal to the sum of $\mathcal{V}$'s running time, $\mathcal{B}$'s answer querying time and $\mathcal{B}$'s time to successfully break the ECDLP instance with forged signatures. One key query, trapdoor hash query, and signature query respectively requires 1, 2, and 2 scalar multiplication on the group, so we can obtain:

$$t' \leq t + O(q_k + 2q_T + 2q_s)T_{ME}.$$

In summary, the aggregate signature scheme proposed in this paper is $(t_i', \varepsilon', q_H, q_K, q_T, q_S, n)$ existing unforgeable under adaptively chosen message attack. □

6.4. Security Comparisons

As shown in Table 3, the security of our MTH-AS scheme is compared with relevant aggregate signature schemes [11,24,35]. Since our proposed scheme selects the latest timestamp t_i', which is included in the messages $\{t_i', r'_i, A'_i, Y'_i, \sigma_i\}_1^n$. Replay attacks can be found by checking the freshness of the timestamp. Thus, our proposed scheme is resistant against replay attacks. The schemes proposed in literature [11,24,35] are proven to be secure based on the hardness of Co-CDHP, CDHP, CDHP, respectively. As mentioned above, the MTH-AS scheme proposed in this paper is proven to be secure against the existing unforgeability under adaptively chosen message attacks assuming ECDLP is hard. Our MTH-AS scheme could provide message authentication by checking whether the equations $k_iP + f(h, r_i', Y_i')Y_i = A_i', F(A'_i) = r'_i$ hold. Due to the above security analysis, our MTH-AS scheme is suitable for secure communications in IoT applications with limited computing power, storage capacity, and bandwidth.

Table 3. The security comparison of relevant aggregate signatures.

Scheme	Hardness Problem	Message Authentication	Resistance to Replay Attacks
Kumar [11]	CDHP	Yes	No
Boneh [24]	Co-CDHP	Yes	No
Cheng [35]	CDHP	Yes	No
Our proposed scheme	ECDLP	Yes	Yes

6.5. Performance Analysis

In this subsection, performance analysis is mainly carried out from two aspects: the performance comparison of related aggregate signature schemes and the performance comparison of aggregate signatures in IoTs. These schemes are measured by communication cost and computation cost, which are considered in terms of the length of the aggregate signature and the computational complexity of the aggregation verification algorithm respectively.

In this paper, we adopt the performance evaluation method in [39]. The experiments are taken on an Intel I7 3.4 GHz, 4 GB machine with Windows 7 operating system. to obtain the security level of 80 bits, the bilinear pairing $e : G_1 \times G_1 \rightarrow G_2$ is conducted, where G_1 is an additive group generated by a point $\overline{P}$ with the order q_1 on the super singular elliptic curve $E_1 : y^2 = x^3 + x \bmod p_1$ (p_1 and q_1 are 512-bit and 160-bit prime number, respectively) [36]. Accordingly the size of the elements in group G_1 (that is L_{G_1}) is 128 bytes (64 * 2 = 128). For ECC-based aggregate signature schemes, we use an additive group G generated by a point on a non-singular elliptic curve $E : y^2 = x^3 + ax + b \bmod p_2$ with the order q_2 (p_2, q_2 are two 160-bit prime numbers, $a, b \in Z_q^*$). The size of the elements in group G (that is L_G) is 40 bytes $(20 * 2 = 40)$. Let L_q be the size of the elements in Z_q^*, n the number of signers. Table 4 lists the execution time of the encryption operations below [36,39].

Table 4. Different encryption operation execution time.

Encryption Operation	Description	Time (ms)
T_B	The bilinear pair operation	4.2110
T_{MB}	The scalar multiplication in the bilinear pair	1.7090
T_{AB}	The bilinear pair-to-midpoint addition	0.0071
T_{HB}	The hash-to-point operation in bilinear pair	4.4060
T_{ME}	The scalar multiplication in elliptic curve	0.4420
T_{AE}	The point addition operation in elliptic curve	0.0018
T_H	The general hash operation	0.0001

(1) The comparison of aggregate signature schemes

The scheme proposed in this paper is compared with the related aggregate signature schemes from four aspects: individual signature, aggregate verify, aggregate signature length and correlation between signature length and n. The specific performance comparison results of computation cost are presented in Table 5 and Figure 3.

Table 5. The comparison of computation cost.

Scheme	Individual Signature Time	Aggregate Verification Time
Gong-1 [29]	$2T_{MB} + T_{AB} + T_{HB} \approx 7.8311ms$	$(2n + 1)\ T_B + 2n\ T_{HB}$ $\approx 17.234n + 4.211ms$
Gong-2 [29]	$3T_{MB} + 2T_{AB} + 2T_{HB} \approx 13.9532ms$	$(n + 2)\ T_B + nT_{MB} + n\ T_{AB} + 2n\ T_{HB}$ $\approx 14.7391n + 8.422ms$
Chen [32]	$4T_{MB} + 2T_{AB} + 2T_{HB} + 2T_H \approx 15.6624ms$	$4T_B + 2nT_{MB} + (n + 2)T_{HB} + 2nT_H$ $\approx 7.8242n + 25.656ms$
Zhou-I [34]	$T_{ME} + T_H \approx 0.4421ms$	$(2n + 1)T_{ME} + 4nT_{AE} + 2nT_H$ $\approx 0.9166n + 0.422ms$
Zhou-II [34]	$T_{ME} + nT_{AE} + T_H \approx 0.0071n + 1.7091ms$	$(2n + 1)T_{ME} + (3n + 1)T_{AE} + 2nT_H$ $\approx 0.9085n + 0.4438ms$
Cheng [35]	$4T_{MB} + 2T_{AB} + T_{HB} \approx 11.2562ms$	$3T_B + nT_{MB} + nT_{AB} + nT_{HB} + nT_H$ $\approx 6.1222n + 12.633ms$
Our proposed scheme	$2T_{ME} + 3T_H \approx 0.8843ms$	$(n + 1)T_{ME} + (n + 1)T_{AE} + 2T_H$ $\approx 0.4438n + 0.444ms$

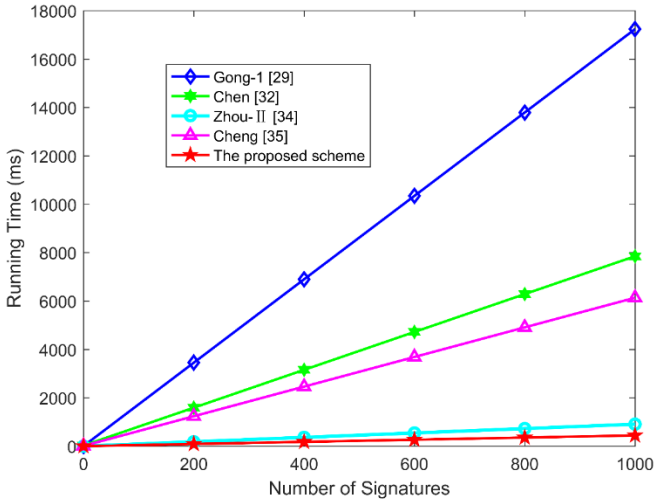

Figure 3. The comparison of aggregate verification time.

As shown in Table 5, the individual signature time of the proposed scheme is 0.8843 ms, which is longer than that of CLAS-I [34]. However, the aggregate signature overhead of our scheme is better than that of CLAS-I [34]. The proposed scheme in this paper is based on ECC which is more efficient than bilinear pairings in computation cost [39]. Figure 3 shows that the aggregate verification

time of our scheme is obviously better than that of the bilinear pairings based aggregate signature schemes [29,32,35], but slightly higher than ECC based schemes [34].

The comparisons of communication cost are presented in Table 6 and Figure 4. Since the size of L_{G_1}, L_G, L_q are 128 bytes, 40 bytes, and 20 bytes, respectively, the lengths of Gong et al.'s *CAS*-1 and *CAS*-2 scheme [29] are $128 * n + 128$ bytes and $2 * 128 = 256$ bytes. The communication cost of Zhou et al.'s CLAS-I scheme [34] is $128 * n + 20$ bytes. The proposed scheme in this paper has the same constant aggregate signature length as CLAS-II [34], which is 60 bytes, obviously superior to that of *CAS*-1 [29] and CLAS-I [34]. The proposed scheme in this paper has great advantages in communication efficiency.

Table 6. The comparison of communication cost.

Scheme	Aggregate Signature Length	Correlation between Signature Length and n
Gong-1 [29]	$(n+1)L_{G_1}$	Yes
Gong-2 [29]	$2L_{G_1}$	No
Chen [32]	$(n+1)L_{G_1}$	Yes
Zhou-I [34]	$nL_G + L_q$	Yes
Zhou-II [34]	$L_G + L_q$	No
Cheng [35]	$(n+1)L_{G_1}$	Yes
Our proposed scheme	$L_G + L_q$	No

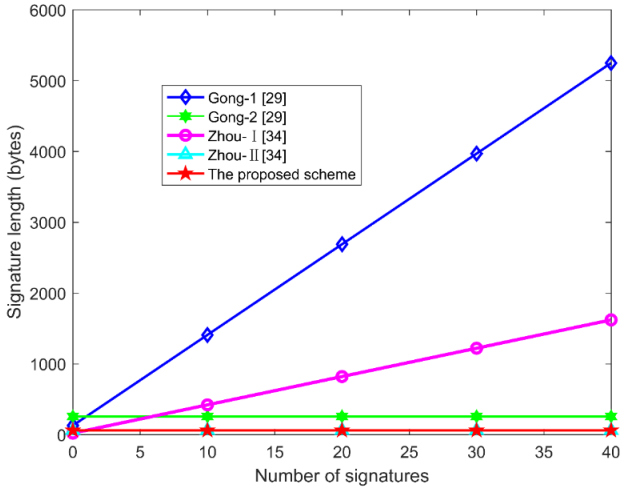

Figure 4. The comparison of signature length.

Performance improvements are compared in Table 7. For example, compared to individual signatures of *CAS*-1 [29], the efficiency of performance improvement is approximately $\frac{7.8311-0.8843}{7.8311} \approx 88.70\%$. Other performance improvements are calculated in the same way, assuming that the number of signatures is 50. As shown in Table 7, in terms of individual signature, the computation cost of our scheme is inferior to that of Zhou et al.'s CLAS-I scheme [34]. However, in terms of aggregate verify, the computation cost of our scheme is superior to all the other schemes [29,32,34,35]. Therefore, the proposed scheme in this paper is more efficient in aggregation verification.

Table 7. The comparison of computational costs with other schemes.

Scheme	Individual Signature Time (%) ($n = 50$)	Aggregate Verification Time (%)($n = 50$)
Gong-1 [29]	88.70	97.39
Gong-2 [29]	93.66	96.96
Chen [32]	94.35	94.57
Zhou-I [34]	−100.00	51.06
Zhou-II [34]	57.16	50.65
Cheng [35]	92.14	92.90

(2) The comparison of aggregate signatures in IoTs

In IoT applications, it incurs lower computation cost and communication cost due to the limited battery capacity, computing power, bandwidth and storage capacity requirements of IoT devices. In this part, we compare the performance of the proposed scheme and other related IoT-based aggregate signature schemes. The cost of signature and verification is an important factor affecting the computing power of the IoT devices. Table 8 and Figure 5 show the comparison of computation cost in IoT-based aggregate signatures. From Figure 5, we can see that the aggregate verification delay of this proposed scheme is similar to that of Cui's scheme [36], but is obviously superior to that of the other schemes [11,12], because bilinear pairings are not used in our scheme and Cui's scheme [36].

Table 8. The comparison of computation cost in IoTs.

Scheme	Individual Signature Time	Aggregate Verification Time
Kumar [11]	$3T_{MB} + 2T_{AB} + T_{HB} + T_H \approx 9.5437ms$	$3T_B + nT_{MB} + 3nT_{AB} \approx 1.7303n + 12.633ms$
Horng [12]	$2T_{MB} + T_{AB} + T_{HB} \approx 3.4252ms$	$3T_B + nT_{MB} + nT_{AB} + nT_{HB} + nT_H \approx 6.1222n + 12.633ms$
Cui [36]	$T_{ME} + T_{AE} + T_H \approx 0.4439ms$	$(n+2)T_{ME} + 2nT_{AE} + 2nT_H \approx 0.4458n + 0.884\ ms$
Our proposed scheme	$2T_{ME} + 3T_H \approx 0.8843ms$	$(n+1)T_{ME} + (n+1)T_{AE} + 2T_H \approx 0.4438n + 0.444\ ms$

The signature length affects the communication capability of IoT devices, which is an important factor in IoT applications. As shown in Table 9 and Figure 6, the signature length of our scheme is a constant (60 bytes), while the signature length of other schemes [11,12,36] are correlate with the IoT devices. In other words, as the number of the signed IoT devices increases, the signature length increases too. Therefore, the scheme proposed in this paper is more suitable for the application of IoTs because it has great advantages in saving bandwidth and storage capacity.

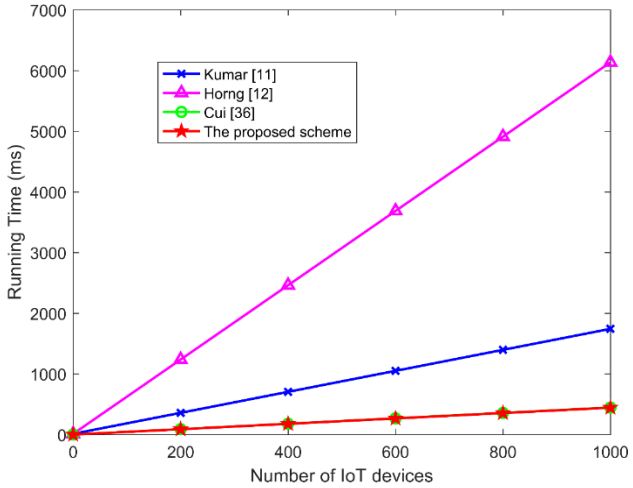

Figure 5. The comparison of aggregate verification cost in IoTs.

Table 9. The comparison of communication cost in IoTs.

Scheme	Aggregate Signature Length	Correlation between Signature Length and n
Kumar [11]	$(n+1)L_{G_1}$	Yes
Horng [12]	$(n+1)L_{G_1}$	Yes
Cui [36]	$(n+1)L_G$	Yes
Our proposed scheme	$L_G + L_q$	No

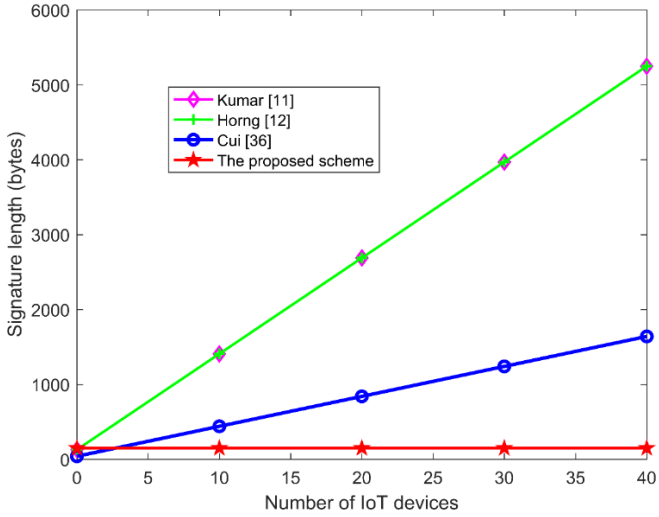

Figure 6. The comparison of signature length in IoTs.

7. Conclusions

In this paper, an aggregate signature scheme is proposed based on ECDLP and MTH, which could be used for secure communication in many-to-one IoT applications. In the random oracle model, the proposed scheme is proven to be secure against the existing unforgeability under adaptively chosen message attacks. The proposed scheme has the characteristics of batch trapdoor collision calculation with multi-trapdoor hash functions and does not use bilinear pair operations. Therefore, it is more efficient than other schemes in terms of computation cost. On the other hand, the length of aggregate signature in this proposed scheme does not depend on the number of the signed IoT devices. Thus, the storage space and bandwidth are greatly saved. In summary, the scheme proposed in this paper is suitable for IoT applications with limited computing speed, storage capacity and bandwidth, such as wireless sensor networks, vehicular ad hoc networks, and healthcare sensor networks, etc.

Author Contributions: Conceptualization: H.S. and D.X.; methodology: H.S.; validation: P.Q. and Y.H.; formal analysis: H.S., P.Q. and Y.H.; investigation: H.S. and L.S.; writing—original draft preparation: H.S.; writing—review and editing: D.X. and F.C.; supervision: L.S.; project administration: H.S.; funding acquisition: F.C.

Funding: This research was funded by the National Natural Science Foundation of China (nos. 61972438, 61672039, 61602009, and 61801004); the Natural Science Foundation of Anhui Province (no. 1808085QF211); the Natural Science Foundation of Universities of Anhui Province (nos. KJ2019A0702 and KJ2019A0704).

Acknowledgments: The authors thank for the help of reviewers and editors.

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

References

1. Atzori, L.; Iera, A.; Morabito, G. The Internet of Things: A survey. *Comput. Netw.* **2010**, *54*, 2787–2805. [CrossRef]
2. Yang, Y.C.; Wu, L.F.; Yin, G.S.; Li, L.J.; Zhao, H.B. A Survey on Security and Privacy Issues in Internet-of-Things. *IEEE Internet Things J.* **2017**, *4*, 1250–1258. [CrossRef]
3. Hiremath, S.; Geng, Y.; Mankodiya, K. Wearable Internet of Things: Concept, Architectural Components and Promises for Person-Centered Healthcare. In Proceedings of the 5th Eai International Conference on Wireless Mobile Communication & Healthcare, London, UK, 14–16 October 2015; pp. 304–307.
4. Yang, X.D.; Pei, X.Z.; Chen, G.L.; Li, T.; Wang, M.D.; Wang, C.F. A Strongly Unforgeable Certificateless Signature Scheme and Its Application in IoT Environments. *Sensors* **2019**, *19*, 2692. [CrossRef] [PubMed]
5. Yeh, K.-H.; Su, C.; Choo, K.R.; Chiu, W. A novel certificateless signature scheme for smart objects in the internet-of-things. *Sensors* **2017**, *17*, 1001. [CrossRef] [PubMed]

6. Kumar, M.; Verma, H.K.; Sikka, G. A secure lightweight signature based authentication for Cloud-IoT crowdsensing environments. *Trans. Emerg. Telecommun. Technol.* **2018**, *30*, 3292–3306. [CrossRef]
7. Chen, L.; Cheng, Z.; Smart, N.P. Identity-based key agreement protocols from pairings. *Int. J. Inf. Secur.* **2007**, *6*, 213–241. [CrossRef]
8. Amin, F.; Ahmad, A.; Sang Choi, G.S. Towards Trust and Friendliness Approaches in the Social Internet of Things. *Appl. Sci.* **2019**, *9*, 166. [CrossRef]
9. Amin, F.; Abbasi, R.; Rehman, A.; Choi, G.S. An Advanced Algorithm for Higher Network Navigation in Social Internet of Things Using Small-World Networks. *Sensors* **2019**, *19*, 2007. [CrossRef] [PubMed]
10. Amin, F.; Ahmad, A.; Choi, G.S. Community Detection and Mining Using Complex Networks Tools in Social Internet of Things. In Proceedings of the 2018 IEEE Region 10 Conference, Jeju Island, Korea, 28–31 October 2018; pp. 2086–2091.
11. Kumar, P.; Kumari, S.; Sharma, V.; Sangaiah, A.K.; Wei, J.H.; Li, X. A certificateless aggregate signature scheme for healthcare wireless sensor network. *Sust. Comput.* **2017**, *18*, 80–89. [CrossRef]
12. Horng, S.J.; Tzeng, S.F.; Huang, P.H.; Wang, X.; Li, T.; Khan, M.K. An efficient certificateless aggregate signature with conditional privacy-preserving for vehicular sensor networks. *Inf. Sci.* **2015**, *317*, 48–66. [CrossRef]
13. Shen, L.; Ma, J.; Liu, X.; Wei, F.; Miao, M. A Secure and Efficient ID-Based Aggregate Signature Scheme for Wireless Sensor Networks. *IEEE Internet Things J.* **2017**, *4*, 546–554. [CrossRef]
14. Krawczyk, H.M.; Rabin, T.D. Chameleon signatures. In Proceedings of the Network and Distributed System Security Symposium (NDSS 2000), San Diego, CA, USA, 2–4 February 2000; pp. 143–154.
15. Wu, C.H. Trapdoor Commitment, Trapdoor Hash and Their Applications. Ph.D. Thesis, Sun Yat-Sen University, Guangzhou, China, 2010.
16. Shamir, A.; Tauman, Y. Improved Online/Offline Signature Schemes. In Proceedings of the 21th Annual International Cryptology Conference (CRYPTO 2001), Santa Barbara, CA, USA, 19–23 August 2001; pp. 355–367.
17. Chen, X.; Zhang, F.G.; Kim, K. Chameleon Hashing Without Key Exposure. In Proceedings of the 7th International Information Security Conference (ISC 2004), Palo Alto, CA, USA, 27–29 September 2004; pp. 87–98.
18. Ateniese, G.; Medeiros, B.D. On the Key Exposure Problem in Chameleon Hashes. In Proceedings of the 4th International Conference on Security in Communication Networks (SCN 2004), Amalfi, Italy, 8–10 September 2004; pp. 165–179.
19. Chen, X.F.; Zhang, F.G.; Susilo, W.; Mu, Y. Efficient Generic On-Line/Off-Line Signatures Without Key Exposure. In Proceedings of the 5th International Conference on Applied Cryptography and Network Security, Zhuhai, China, 5–8 June 2007; pp. 18–30.
20. Chandrasekhar, S.; Singhal, M. Multi-trapdoor hash functions and their applications in network security. In Proceedings of the 2nd IEEE Conference on Communications and Network Security, San Francisco, CA, USA, 29–31 October 2014; pp. 463–471.
21. Chandrasekhar, S.; Chakrabarti, S.; Singhal, M. A trapdoor hash-based mechanism for stream authentication. *IEEE Trans. Dependable Secur. Comput.* **2012**, *9*, 699–713. [CrossRef]
22. Chandrasekhar, S.; Singhal, M. Efficient and scalable aggregate signcryption scheme based on multi-trapdoor hash functions. In Proceedings of the 1st Workshop on Security and Privacy in the Cloud, Florence, Italy, 28–30 September 2015; pp. 610–618.
23. Chandrasekhar, S.; Ibrahim, A.; Singhal, M. A novel access control protocol using proxy signatures for cloud-based health information exchange. *Comput. Secur.* **2017**, *67*, 73–88. [CrossRef]
24. Boneh, D.; Gentry, C.; Lynn, B.; Shacham, H. Aggregate and Verifiably Encrypted Signatures from Bilinear Maps. In Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT 2003), Warsaw, Poland, 4–8 May 2003; pp. 416–432.
25. Lysyanskaya, A.; Micali, S.; Reyzin, L.; Shacham, H. Sequential Aggregate Signatures from Trapdoor Permutations. In Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT 2004), Interlaken, Switzerland, 2–6 May 2004; pp. 74–90.
26. Brogle, K.; Goldberg, S.; Reyzin, L. Sequential Aggregate Signatures with Lazy Verification from Trapdoor Permutations. In Proceedings of the 18th International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT 2012), Beijing, China, 2–6 December 2012; pp. 644–662.

27. Ahn, J.H.; Green, M.; Hohenberger, S. Synchronized aggregate signatures: New definitions, constructions and applications. In Proceedings of the 17th ACM conference on Computer and communications security, Chicago, IL, USA, 4–8 October 2010; pp. 473–484.
28. Gentry, C.; Ramzan, Z. Identity-Based aggregate signatures. In Proceedings of the International Conference on Theory & Practice of Public-key Cryptography, New York, NY, USA, 24–26 April 2006; pp. 257–273.
29. GONG, Z.; LONG, Y.; HONG, X.; CHEN, K. Two Certificateless Aggregate Signatures From Bilinear Maps. In Proceedings of the 8th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD 2007), Qingdao, China, 30 July–1 August 2007; pp. 2093–2106.
30. Zhang, L.; Qin, B.; Wu, Q.H.; Zhang, F.T. Efficient many-to-one authentication with certificateless aggregate signatures. *Comput. Netw.* **2010**, *54*, 2482–2491. [CrossRef]
31. Pointcheval, D.; Stern, J. Security arguments for digital signatures and blind signatures. *J. Cryptol.* **2000**, *13*, 361–396. [CrossRef]
32. Chen, H.; Wei, S.M.; Zhu, C.J.; Yang, Y. Secure certificateless aggregate signature scheme. *J. Softw.* **2015**, *26*, 1173–1180.
33. Li, Y.P.; Nie, H.H.; Zhou, Y.W.; Yang, B. A novel and provably secure certificateless aggregate signature scheme. *J. Cryptologic. Res.* **2015**, *2*, 526–535.
34. Zhou, Y.W.; Yang, B.; Zhang, W.Z. Efficient and provide security certificateless aggregate signature scheme. *J. Softw.* **2015**, *26*, 3204–3214.
35. Cheng, L.; Wen, Q.Y.; Jin, Z.P.; Zhang, H.; Zhou, L.M. Cryptanalysis and improvement of a certificateless aggregate signature scheme. *Inf. Sci.* **2015**, *295*, 337–346. [CrossRef]
36. Cui, J.; Zhang, J.; Zhong, H.; Shi, R.H.; Xu, Y. An efficient certificateless aggregate signature without pairings for vehicular ad hoc networks. *Inf. Sci.* **2018**, *451*, 1–15. [CrossRef]
37. Johnson, D.; Menezes, A.; Vanstone, S. The Elliptic Curve Digital Signature Algorithm (ECDSA). *Int. J. Inf. Secur.* **2001**, *1*, 36–63. [CrossRef]
38. Pointcheval, D.; Stern, J. Security proofs for signature schemes. In Proceedings of the International Conference on the Theory & Applications of Cryptographic Techniques (EUROCRYPT 1996), Saragossa, Spain, 12–16 May 1996; pp. 387–398.
39. He, D.B.; Zeadally, S.; Xu, B.W.; Huang, X.Y. An Efficient Identity-Based Conditional Privacy-Preserving Authentication Scheme for Vehicular Ad Hoc Networks. *IEEE Trans. Inf. Forensic Secur.* **2015**, *10*, 2681–2691. [CrossRef]