mathematics

Article

Fine-Grained Forward Secure Firmware Update in Smart Home

Qiuxia Zhao [1,2,*], Dong Zheng [3,*], Yinghui Zhang [3] and Yan Ren [2]

1 College of Computer, Qinghai Normal University, Xining 810008, China
2 Maths and Information Technology School, Yuncheng University, Yuncheng 044000, China; renyan-2000@163.com
3 National Engineering Research Center for Secured Wireless, Xi'an University of Posts and Telecommunications, Xi'an 710121, China; yhzhaang@163.com
* Correspondence: qiuxiazhao24@163.com (Q.Z.); zhengdong@xupt.edu.cn (D.Z.)

Abstract: Although the vigorous development of smart homes brings great convenience to people's lives, smart homes usually suffer from various security threats due to firmware vulnerabilities. Firmware update is a possible solution, but existing methods cannot address the security issues during the update process well. To enable secure firmware updates, a Forward Secure Firmware Update (FSFU) system was realized based on the proposed Puncturable-Ciphertext Policy-Attribute-Based Encryption (P-CP-ABE) scheme. In FSFU, when the service provider delivers the latest firmware, it specifies an access policy and some tags to encrypt the data and appends its signature to achieve both fine-grained access control and authentication. Authorized customers can obtain the latest firmware by decrypting the encrypted data through their private key. In particular, after a successful update, each authorized customer can realize forward security by updating his/her puncturable key, which is an important private key component. In addition, FSFU is further enhanced by outsourcing a part of the parameters and computational tasks. Finally, FSFU was proven to be secure under the Decisional Bilinear Diffie–Hellman (DBDH) assumption. Our proposed FSFU is efficient from both the theoretical analysis and the experimental results.

Keywords: firmware update; forward secure; fine-grained access; authentication; outsourcing

MSC: 94A60

check for updates

Citation: Zhao, Q.; Zheng, D.; Zhang, Y.; Ren, Y. Fine-Grained Forward Secure Firmware Update in Smart Home. *Mathematics* **2023**, *11*, 3084. https://doi.org/10.3390/math11143084

Academic Editor: Cheng-Chi Lee

Received: 13 June 2023
Revised: 3 July 2023
Accepted: 11 July 2023
Published: 12 July 2023

1. Introduction

With the continuous development of IoT technology, smart homes have become an attractive application that can provide users with more comfort and convenience in their lives [1,2]. Major manufacturers have developed and launched their smart home systems. They provide consumers with supporting smart devices, cloud platforms, and mobile applications. Some of the more famous are Samsung's SmartThings [3], Amazon's AWS [4], Apple's Home app [5], Alibaba's AliyunloT [6], etc. Manufacturers are eager to develop new features and products to attract consumers and increase their share of the smart home market.

The potential economic benefits of smart homes have attracted many hacker groups to attack vulnerabilities in smart home systems, and many of the attacks are targeting customers' IoT devices. Technically, IoT devices are very small embedded devices that are used for sensing, control, and so on [7]. To protect embedded devices, an effective way is to update the firmware to fix the vulnerabilities. The service provider uploads the latest firmware to the cloud platform for the authorized customers to download.

As shown in Figure 1, there are possible threats, such as pretending and eavesdropping, during the firmware update process, and therefore, its security needs to be considered. However, some existing methods do not address the security issues during firmware updates well [8]. Some state-of-the-art techniques, including differential privacy [9], opacity [10], etc., can achieve security. Differential privacy focuses on protecting the privacy of changes in the database, and opacity focuses on hiding system behavior. In smart home systems, service providers only expect authorized customers meeting certain policies to enjoy the latest firmware updates, hence Ciphertext Policy-Attribute-Based Encryption (CP-ABE) is more suitable for achieving secure firmware updates.

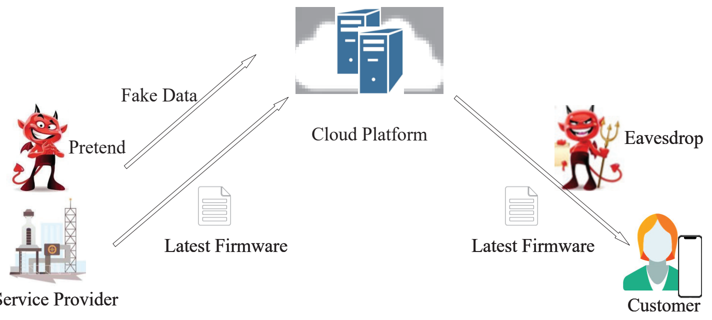

Figure 1. Possible threats in the firmware update process.

1.1. Contribution

To achieve secure firmware updates in the smart home, an FSFU system is proposed based on a proposed Puncturable-Ciphertext Policy-Attribute-Based Encryption (P-CP-ABE) scheme. In FSFU, the service provider is called the Data Owner (DO), and the customer is called the Data User (DU). When the DO supplies the latest firmware, it encrypts the data with a specified access policy and some tags. To enable customers to identify the source of the firmware, the DO attaches a signature to the encrypted data. The authorized DU can obtain plaintext from the encrypted data using his/her private key, which contains a puncturable key component. After a successful update, the DU can renew his/her puncturable key component, and the updated key loses the capability to decrypt the past encrypted data. We also propose extended FSFU, which outsources some parameters and operations to the fog node. Specifically, the features of FSFU are listed below:

- *Authentication:* The signature allows customers to verify that the latest firmware is being delivered by the expected service provider.
- *Fine-grained access control:* Upon receipt of the encrypted data, only authorized users can obtain the plaintext, where authorized users are those whose attribute sets satisfy the policy and the puncturable key does not contain the tag in the ciphertext.
- *Forward security:* After successfully updating to the latest firmware, data users can update their puncturable key component by puncturing a tag attached to the ciphertext. The updated key loses the capability to decrypt the past ciphertext, ensuring the forward security of the encrypted data.
- *Outsourced capability:* In extended FSFU, the DO can outsource some of the encryption work to fog nodes. Similarly, the DU can send part of the private key to the fog node for storage and, thus, outsource part of the decryption work. As a result, the storage and computation costs of the participants, especially DUs, can be reduced.

1.2. Related Work

Sahai and Waters first proposed the concept of Attribute-Based Encryption (ABE) [11], in which DUs are described by their attributes so that ABE can achieve fine-grained access control on encrypted data. In general, there are two main types of ABE: Key Policy-Attribute-Based Encryption (KP-ABE) [12] and Ciphertext Policy-Attribute-Based Encryption (CP-ABE) [13]. The difference between them is that CP-ABE's ciphertext is related to a specified policy (or structure) and the decryption key is related to attributes, while KP-ABE is just the opposite: the ciphertext is related to the attributes, and the decryption key is related to the policy. If the attributes satisfy the policy, the plaintext can be recovered. Ibraimi, Tang,et al. proposed a selectively secure CP-ABE based on the

Decisional Bilinear Diffie–Hellman (DBDH) assumption [14], which can be conveniently modified to outsource the computation during encryption and decryption. There are a large number of studies on attribute-based encryption from [15]. However, very little work considers the forward security of the encrypted data.

The survey [8] investigates various attacks against firmware updates of IoT devices from 2004 to 2018 and discusses measures for secure firmware updates. According to the survey, existing secure firmware updates can be divided into "centralized" and "distributed" categories. Centralized solutions based on the server–client model include [16–18], etc. Decentralized solutions, mostly based on blockchain, include [7,19,20], etc. However, as mentioned above, there is very little work that deals with forward security.

There are some proposals to add forward security to encrypted email, which is a "store and forward" messaging system [21–23]. However, these proposals increase complexity and introduce new points of failure. The distribution of new key components for senders requires a highly available network infrastructure [21]. The premise of secure communication is that an initial message must be exchanged [24]. Reference [25] proposed a Forward Secure-Public Key Encryption scheme (FS-PKE) in which the user could periodically update the key to revoke the decryption capability. However, it is coarse-grained. Puncturable Encryption (PE) [26], proposed in SP 2015, enables fine-grained forward security by puncturing specific tags. In this way, fine-grained revocation of the decryption capability can be achieved. What is more, users can update their keys themselves without the need for the trusted center to reissue a new key. There are some recent works related to PE still emerging [27–30].

1.3. Organization

The rest of this paper is organized as follows. Section 2 presents the system architecture and design goals. Section 3 lists the related preliminaries. Section 4 describes the formal definition of P-CP-ABE and the security model. The details of FSFU (including the extended construction) are given in Section 5. We prove the security and analyze the performance in Section 6. The conclusion of the work is given in Section 7.

2. System Architecture and Design Goal

2.1. System Architecture

The system architecture is shown in Figure 2. There are five entities in the system. The role and responsibilities of each entity are described below:

- The Trusted Authority (TA) is fully trusted. The system's public parameters and secret keys for participants are generated by the TA. We assumed that the TA is responsible for publishing the attribute universe $\Omega = \{a_1, \ldots, a_n\}$ and a collection of possible tags.
- The Cloud Sever (CS) is semi-trusted. The CS can provide a powerful storage service for participants.
- The Fog Node (FN) is semi-trusted. FNs act as caches between the participants and the CS. FNs can provide temporary storage and outsource computing services to participants.
- The Data Owner (DO) is the service provider, which delivers the latest firmware to the DU via the FNs and the CS.
- The Data User (DU) requests data files associated with specific tags and receives them from the FN. If the DU is in the same domain as the DO, the DU can obtain the data file directly from the FN. If the DU is far away from the DO, the FN close to the DU will request the data file from the neighboring FNs and the cloud platform, and the data file will be transmitted to it and eventually forwarded to the DU.

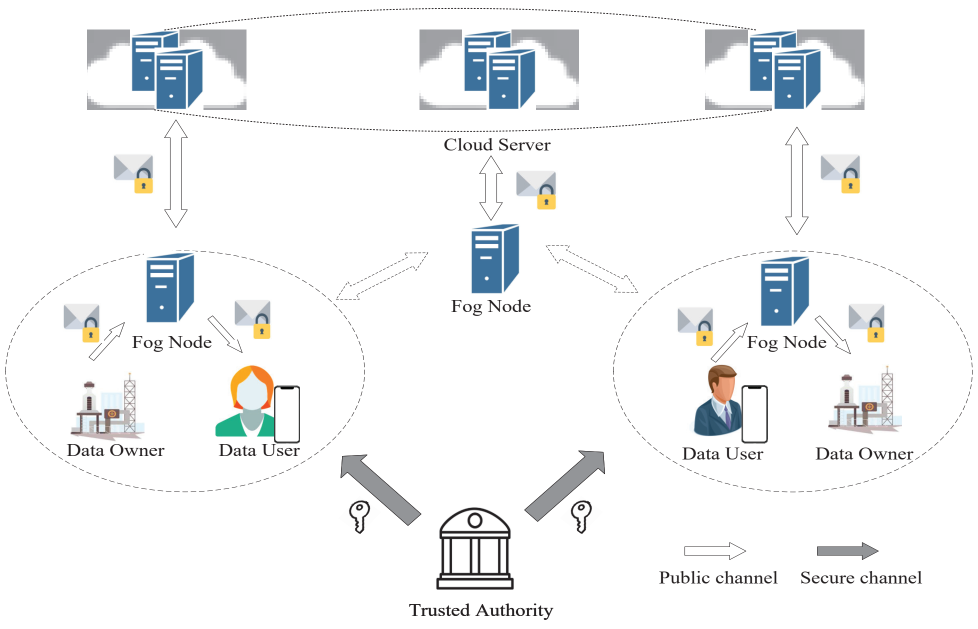

Figure 2. The FSFU system architecture.

An overview of the FSFU system can be found below:

(1) *Initialization:* The system is initialized by the TA, which generates the public parameters and a master key. The public parameters of the system are public to all participants, and the master key is secret.

(2) *Authorization:* The DU authenticates to the TA using his/her own set of attributes, and the TA issues a secret decryption key to the DU based on the set of attributes. Meanwhile, the DO authenticates to the TA using its ID, and the TA issues a signing key associated with the ID to the DO.

(3) *Secure latest firmware delivery:* For the latest firmware, the DO specifies an access policy, uses it along with some tags to encrypt the latest firmware, and embeds its signature. The ciphertext is outsourced to the nearest fog node and then transmitted to the cloud platform.

(4) *Latest fine-grained firmware access:* After receiving the latest encrypted firmware, the authorized DU first verifies the signature. Then, the plaintext is revealed from the ciphertext if each of the embedded tags has never been punctured.

(5) *Revocation of decryption capability of latest firmware:* After revealing the latest firmware and successfully updating it, the DU can revoke the decryption capability for the ciphertext by puncturing a tag attached to it. If the DU has done this, he/she has updated the private key himself/herself, and malicious participants who have stolen the secret key cannot decrypt the current ciphertext.

2.2. Adversary Model and Design Goal

In FSFU, both a malicious DO and an unauthorized DU may attack the system as adversaries. For authentication, a malicious DO may deliver unexpected data, pretending to be a legitimate provider for improper gain. The encrypted data are transmitted over a public channel, so anyone can obtain the ciphertext. For data confidentiality, malicious unauthorized users may collude to recover the latest firmware. In addition, some malicious DUs may attack the devices of authorized DUs to steal private keys. Therefore, we considered the following specific security goals:

- *Data confidentiality:* The latest outsourced firmware should be protected from unauthorized access due to its economic value.
- *Authentication:* When the DO publishes the latest encrypted firmware, the DU should be able to verify that the expected service provider has published it.

- *Collusion resistance:* Unauthorized users with different attribute sets may collude by combining their private keys to obtain the latest firmware for free. For economic reasons, these collusion attacks must be prevented.
- *Forward security:* The DU's device may be hacked, and the private key may be stolen. When the latest firmware is successfully updated, the private decryption key should be updated by the DU itself so that the current ciphertext can no longer be decrypted with the updated key.

3. Preliminaries

Table 1 lists some notations and their descriptions.

Table 1. Notations and descriptions.

Notation	Description
Ω	The universe of attributes.
$\mathbb{G}_1, \mathbb{G}_2$	Two cyclic multiplicative groups.
$\mathbb{Z}_p$	Integer ring with modulus p, where p is a prime number.
$\mathbb{Z}_p^*$	$\mathbb{Z}_p^* = \mathbb{Z}_p \backslash \{0\}$.
$H_i(\cdot), i = 1, 2, 3$	Hash function.
Γ	Access tree associated with the ciphertext.
$\lvert\Gamma\rvert$	Number of leaves in Γ.
S	Attribute set of the DU.
S'	The smallest subset of S that satisfies Γ.
$t_1, \ldots, t_d$	Tags associated with the ciphertext; d is the maximum number.
PK	Public key, including the public system parameters.
MSK	Master key secretly held by the TA.
sk	Participant's private key.
KP	Puncturable key, which is a component of the decryption key.

3.1. Access Structures and Access Tree

Definition 1 (*Access structures* [14])**.** *Let $\{P_1, P_2, \ldots, P_n\}$ be a set of parties. A collection $\mathbb{A} \subseteq 2^{\{P_1, P_2, \ldots, P_n\}}$ is monotone if $\forall B, C$: if $B \in \mathbb{A}$ and $B \subseteq C$, then $C \in \mathbb{A}$. An access structure is a collection $\mathbb{A}$, i.e., $\mathbb{A} \subseteq 2^{\{P_1, P_2, \ldots, P_n\}} \backslash \{\emptyset\}$. An authorized set is a subset in $\mathbb{A}$, and conversely, an unauthorized set is a set that is not in $\mathbb{A}$.*

Access tree: One way to represent an access structure or policy is through an access tree. Each node in the access tree Γ has at most n children. In Γ, the leaves represent attributes, and every other node is a (t, n) threshold gate, which can be AND ($\wedge$) or OR ($\vee$). During the encryption phase, each node is assigned a pair of values (i, s_i), where i is the public index and s_i is the corresponding value [14].

3.2. Lagrange Interpolation and Shamir's Secret-Sharing Scheme

Given t different pairs of values $(x_1, y_1), \ldots, (x_t, y_t)$, a polynomial $f(x)$ of degree $t-1$ can be uniquely defined by the Lagrange interpolation [14], in such a way that:

$$f(x) = \sum_{i=1}^{t} y_i \prod_{j=1, j\neq i}^{t} \frac{(x - x_j)}{(x_i - x_j)}.$$

Suppose a dealer D holds a secret s. It is divided into n shares in Shamir's (t, n) (where $t \leq n$) threshold secret-sharing technique [14]. Any subset with shares greater than or equal to t can jointly reconstruct the secret s, and no subset with shares less than s can obtain the secret s. This is based on polynomial interpolation. The details of this method are given below.

The dealer D chooses a prime $p > \max(s, n)$ and defines $a_0 = s$. Then, D chooses $t-1$ random coefficients $a_1, \cdots, a_{t-1}, 0 \leq a_j \leq p$, and defines the random polynomial over $\mathbb{Z}_p$,

$f(x) = \sum_{j=0}^{t-1} a_j x^j$. According to $f(x)$, D securely sends to user p_i the share $s_i = f(i)$ along with p_i's index i.

Any group of t or more users can reconstruct s using their shares (i, s_i) together by Lagrange interpolation, i.e., $s = f(0) = \sum_{i=1}^{t} s_i l_i(0)$, where $l_i(0) = \prod_{j=1}^{t} \frac{0-j}{i-j}$.

3.3. Ciphertext Policy-ABE

CP-ABE consists of four algorithms [13]:

- Setup (k): Upon the input of a security parameter k, the algorithm outputs the public parameters pk and a master key msk.
- KeyGen (S, msk): Upon the input of the master key msk and an attribute set S, the algorithm outputs a secret key sk_S associated with S.
- Encrypt (m, Γ, pk): Upon the input of the public key pk, a message m, and an access tree Γ, the algorithm outputs the ciphertext c_Γ. The purpose of specifying Γ is to make decryption accessible to authorized users.
- Decrypt (c_Γ, sk_S): Upon the input of a ciphertext c_Γ and a secret key sk_S associated with S, the algorithm outputs a message m or an error symbol $\perp$.

3.4. Puncturable Encryption

A Puncturable Encryption (PE) scheme consists of four algorithms [26]:

- PE.KeyGen(k, d): Upon the input of a security parameter k and a maximum tag number d, the algorithm outputs a public key PK and an initial secret key SK_0.
- PE.Encrypt$(PK, m, t_1, \ldots, t_d)$: Upon the input of a public key PK, a plaintext m, and tags $t_1, \ldots, t_d$, the algorithm outputs the ciphertext c.
- PE.Puncture(PK, SK_{i-1}, t): Upon the input of a secret key SK_{i-1} and a tag t, the algorithm outputs a new secret key SK_i. This new key SK_i revokes the decryption capability on those ciphertexts encrypted with t, and the other decryption capabilities are the same as SK_{i-1}.
- PE.Decrypt$(PK, SK_i, c, t_1, \ldots, t_d)$: Upon the input of a secret key SK_i and a ciphertext c, the algorithm outputs a message m if any tag attached to the ciphertext is not punctured or outputs an error symbol $\perp$.

3.5. Bilinear Pairing and Security Assumption

Definition 2 (*Bilinear pairing* [14])**.** *Let $\mathbb{G}_1, \mathbb{G}_2$ be two multiplicative cyclic groups of prime order p, and let g be a generator of $\mathbb{G}_1$. $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ is a bilinear pairing (or bilinear map) with the following properties:*

– Bilinearity: $e(u^a, v^b) = e(u^b, v^a) = e(u, v)^{ab}$ *for all* $u, v \in \mathbb{G}_1$ *and* $a, b \in \mathbb{Z}_p$.
– Non-degeneracy: $e(g, g) \neq 1$, *where* 1 *is the identity element of* $\mathbb{G}_2$.

Definition 3 (*Computational Diffie–Hellman assumption* [31])**.** *Given $g, g^a, g^b \in \mathbb{G}_1$, for the Computational Diffie–Hellman (CDH) problem in $\mathbb{G}_1$, it is difficult for any PPT adversary to calculate $V = g^{ab}$.*

Definition 4 (*Decisional bilinear Diffie–Hellman assumption* [14])**.** *Given $g, g^a, g^b, g^c \in \mathbb{G}_1$ and a random element $T \in \mathbb{G}_2$, for the Decisional Bilinear Diffie–Hellman (DBDH) problem, it is difficult for any PPT adversary to distinguish the tuple $(g, g^a, g^b, g^c, e(g, g)^{abc})$ from (g, g^a, g^b, g^c, T).*

4. Formal Definition and Security Model

4.1. Definition of Basic P-CP-ABE

There are five algorithms included in our basic P-CP-ABE scheme. The syntax definition is as follows:

- Setup $(k, d) \rightarrow (PK, MSK)$: It takes a security parameter k and a maximum tag number d as the input. The algorithm outputs a public key PK and a master secret key MSK.

- $\mathsf{KeyGen}(PK, MSK, S, ID) \rightarrow (sk_\sigma, sk, KP_0)$: It takes the master secret key MSK, an attribute set S, and a DO identity ID as the input. The algorithm outputs a private key sk_σ associated with ID for signing, a private key sk, and an initial puncture key KP_0 to decrypt the ciphertext together.
- $\mathsf{Encrypt}(PK, M, \Gamma, sk_\sigma, t_1, \ldots, t_d) \rightarrow CT$: It takes a public key PK, a plaintext M, an access tree Γ, a singing key sk_σ associated with ID, and tags $t_1, \ldots, t_d$ as the input. The algorithm outputs the ciphertext CT that contains the signature associated with ID.
- $\mathsf{Puncture}(KP_{i-1}, t) \rightarrow KP_i$: It takes a secret key KP_{i-1} and a tag t as the input. The algorithm outputs an updated key KP_i that cannot decrypt ciphertexts encrypted with t.
- $\mathsf{Decrypt}(CT, S, sk, KP_i) \rightarrow M$ or $\perp$: It takes a ciphertext CT, a secret key sk associated with S, and a puncturable key KP_i as the input. The algorithm outputs a message M or an error symbol $\perp$.

4.2. Security Model

The security of our basic P-CP-ABE is defined by the IND-CPA game played by an adversary $\mathcal{A}$ and a challenger. Here, we considered selective security, i.e., $\mathcal{A}$ specifies the challenge access tree and the set of tags in the initial phase before the public parameters of the system are distributed. The formal definition of the game is given below:

- *Init:* The adversary $\mathcal{A}$ declares the target access tree Γ^* and the set of tags $\{t_1^*, \ldots, t_d^*\}$.
- *Setup:* The challenger initializes a tag set $P = \emptyset$ and a counter $n = 0$. Then, he/she executes $\mathsf{Setup}(k, d) \rightarrow (PK, MSK)$ and gives PK to $\mathcal{A}$.
- *Phase 1:* $\mathcal{A}$ can adaptively issue a polynomially bounded number of queries for any of the following:
  - KeyGen(S): $\mathcal{A}$ queries a secret key for a set of attributes $S = \{a_j | a_j \in \Omega\}$, where $a_j \notin \Gamma^*$.
  - Puncture(t): The challenger sets $n = n + 1$, runs $\mathsf{Puncture}(KP_{n-1}, t) \rightarrow KP_n$, and lets $P = P \cup \{t\}$.
  - Corrupt(): If this is the first time $\mathcal{A}$ issues this query and $\{t_1^*, \ldots, t_d^*\} \cap P = \emptyset$, the challenger sends the current secret key KP_n to $\mathcal{A}$ and sets $C \leftarrow P$. In all other cases, Corrupt() returns $\perp$.
- *Challenge:* $\mathcal{A}$ sends two messages of equal length M_0, M_1 to the challenger. The challenger flips a random coin $\beta \in \{0, 1\}$ and executes $\mathsf{Encrypt}(PK, M_\beta, \Gamma^*, t_1^*, \ldots, t_d^*) \rightarrow CT^*$. The challenge ciphertext CT^* is sent to $\mathcal{A}$.
- *Phase 2:* It is identical to Phase 1.
- *Guess:* $\mathcal{A}$ outputs a guess $\beta' \in \{0, 1\}$.

If $\beta' = \beta$, the adversary $\mathcal{A}$ wins the game. The advantage of adversary $\mathcal{A}$ is characterized as $\epsilon = |\Pr[\beta' = \beta] - \frac{1}{2}|$.

Definition 5. *A P-CP-ABE scheme is selectively secure if any polynomial time adversary has at most a negligible advantage in the above selective game.*

5. Forward Secure Firmware Update System

The details of FSFU are described in this section, and then, we extend it to outsource some parameters and computation to fog nodes.

5.1. Design Details of FSFU

(1) *Initialization:* The TA first takes a security parameter k and a maximum number d as the input and generates a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$, where $\mathbb{G}_1$ is a group of prime order p and g is a generator. Then, the TA chooses three hash functions $H_1, H_2 : \{0,1\}^* \rightarrow \mathbb{G}_1, H_3 : \{0,1\}^* \rightarrow \mathbb{Z}_p$. The TA sets the attribute universe as $\Omega = \{a_1, a_2, \ldots, a_n\}$ and executes the following Setup algorithm:

– Setup(k,d): The TA selects elements $\alpha, n_1, n_2, \ldots, n_n \in \mathbb{Z}_p^*$ at random. Compute $y = e(g,g)^{\alpha}$ and $T_j = g^{n_j}, j = 1, \ldots, n$. Then, the TA selects a random element $a \in \mathbb{Z}_p^*$ and chooses a d-degree polynomial $q(\cdot)$ such that $q(0) = a$. Then, the TA defines $V(x) = g^{q(x)}$. t_0 is set as an initial tag that will not be used later in the encryption operation. The public system parameters are published as PK, and the master key is MSK:

$$PK = (e, g, g^a, y = e(g,g)^{\alpha}, T_j(j = 1, \ldots, n),$$
$$g^{q(l)}, (l = 1, \ldots, d), t_0, H_1, H_2, H_3),$$
$$MSK = (\alpha, a, n_j(j = 1, \ldots, n)).$$

(2) *Authorization:* As shown in Figure 3, the TA grants access rights by issuing a private key to the DU based on its attribute set S and issuing a singing key sk_{σ} to the DO associated with its ID. The TA implements these operations by executing the KeyGen algorithm:

– KeyGen(PK, MSK, S, ID): The TA first computes $P = H_1(ID)$ and sends $sk_{\sigma} = P^a$ to the DO. Then, the TA samples random elements $r, r_a, r' \in \mathbb{Z}_p^*$ and lets

$$sk = (D = g^{\alpha - r} \cdot (g^a)^{r_a}, \forall a_j \in S, D_j = g^{rn_j^{-1}}),$$
$$KP_0 = ([kp_0^1, kp_0^2, kp_0^3, kp_0^4])$$
$$= ([(g^a)^{-r_a + r'}, V(H_3(t_0))^{r'}, g^{r'}, t_0]).$$

The TA sends (sk, KP_0) to the DU.

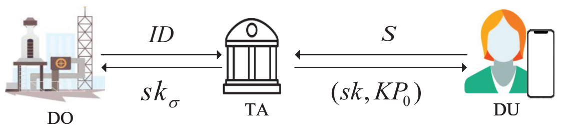

Figure 3. Authorization in FSFU.

(3) *Latest secure firmware delivery:* The DO chooses a symmetric encryption scheme and uses it to encrypt the latest firmware. Then, the DO specifies an access tree Γ and some tags $t_1, \ldots, t_d$ and encrypts the symmetric key, which plays the role of plaintext $M \in \mathbb{G}_2$ in the Encrypt algorithm described below. The encryption process is considered to be two-level. The first level of encryption is associated with M and d tags, where the DO's signature is attached. The second level of encryption is associated with access tree Γ. Finally, as shown in Figure 4, the ciphertext data are outsourced to the local fog node and then transmitted to the cloud platform:

– Encrypt$(PK, M, \Gamma, sk_{\sigma}, t_1, \ldots, t_d)$: The DO first performs the first stage of encryption by selecting a random element $s \in \mathbb{Z}_p^*$ and calculating $C_0 = M \cdot e(g,g)^{\alpha s}, C_1 = g^s, \{C_3^k = V(H_3(t_k))^s, k = 1, \ldots, d\}$.

$$CT_{first} = (C_0, C_1, \{C_3^k, k = 1, \ldots, d\})$$
$$= (g^s, M \cdot e(g,g)^{rs}, \{V(H_3(t_k))^s, k = 1, \ldots, d\}).$$

Then, the DO generates the signature. The DO samples a random element $r^* \in \mathbb{Z}_p^*$ and calculates $P_m = H_2(CT_{first})$. The signature $\sigma = (Sig = P_m^{r*} \cdot P^a, T = g^{r*})$.

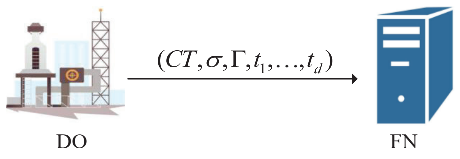

Figure 4. Latest secure firmware delivery in FSFU.

The DO then performs the second-level encryption. In an access tree, consider an AND($\wedge$) node as an (n, n) threshold and an OR($\vee$) node as a $(1, n)$ threshold, where n is the number of its children. Assign values to the nodes of the access tree Γ using a top-down recursive approach. For the root node, set its value to s so that the root is marked as assigned and all other nodes are unassigned.
Recursively, for each inner node i marked assigned, if its children are marked unassigned, its share s_i is divided among its n children by Shamir's (t, n) secret-sharing scheme. Each shared secret $s_j = f(j)$ is assigned to each child node, and thus, this node is marked as assigned. For the leaf node $a_{j,i} \in \Gamma$, which represents an attribute, calculate $c_{j,i} = T_j^{s_i}$, where i is the unique attribute index according to the access tree. This process is illustrated in Figure 5.
Finally, the encrypted data are

$$CT = (\Gamma, C_0, C_1, \{c_{j,i}, \forall a_{j,i} \in \Gamma\}, \{C_3^k, k = 1, \ldots, d\}, \sigma).$$

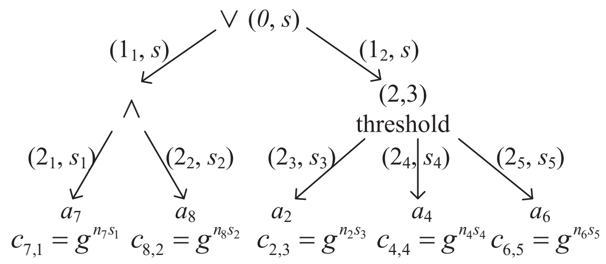

Figure 5. Encryption for leaves of the access tree.

(4) *Latest fine-grained firmware access:* As shown in Figure 6, after the ciphertext is verified by the signature, the authorized DU can reveal the symmetric key that was used to encrypt the latest firmware so that the DU can obtain the latest firmware through the symmetric key. The authorized DU executes the Decrypt algorithm as shown below:

– Decrypt(CT, S, sk, KP_i): If S does not satisfy Γ or $\{t_k, k = 1, \ldots, d\}$ contains the punctured tags, return $\perp$. Otherwise, the algorithm behaves as follows.
The DU first verifies the signature. Check whether the equation $e(Sig, g) = e(T, P_m) \cdot e(g^a, P)$ holds. If not, discard the data; if true, the ciphertext is indeed from the expected DO.
Then, the DU performs the first-level decryption. Choose the smallest $S' \subseteq S$ that satisfies Γ. For each attribute $a_j \in S'$, calculate

$$\begin{aligned}
\widetilde{A} &= \prod_{a_j \in S'} e(D_j, c_{j.i})^{L_i(0)} \\
&= \prod_{a_j \in S'} e(g^{rn_j^{-1}}, T_j^{s_i})^{L_i(0)} \\
&= \prod_{a_j \in S'} e(g^{t_j s_i}, g^{rn_j^{-1}})^{L_i(0)} \\
&= e(g, g)^{r \sum_{a_j \in S'} s_i L_i(0)} \\
&= e(g, g)^{rs},
\end{aligned}$$

where $L_i(0)$ is a Lagrange coefficient and can be calculated by the index of the attribute in the access tree. Then, calculate

$$\begin{aligned}
A &= e(C_1, D) \cdot \widetilde{A} \\
&= e(C_1, D) \cdot e(g, g)^{rs} \\
&= e(g^s, g^{\alpha - r} \cdot (g^a)^{r_a}) \cdot e(g, g)^{rs}
\end{aligned}$$

$$= e(g,g)^{\alpha s} \cdot e(g,g)^{a r_a s}.$$

Simultaneously, the DU decrypts the puncturable part. For $j = 0,1,\ldots,i$, find $w_{j1},\ldots,w_{jd},w_j*$ such that $w_{j*} \cdot q(H_3(kp_j{}^4)) + \sum_{k=1}^{d}(w_{jk} \cdot q(H_3(t_k))) = q(0) = a$ and compute

$$\begin{aligned}
\overline{B} &= \frac{e(kp_0^1, C_0)}{e(kp_0^3, \prod_{k=1}^{d}(C_3^k)^{w_{0k}}) \cdot e(kp_0^2, C_0)^{w_0*}} \\
&= \frac{e((g^a)^{r'-r_a+r_{01}-\lambda_1{}'+\ldots+r_{0i}-\lambda_i{}'}, g^s)}{e(g^{r'+r_{01}+\ldots+r_{0i}}, \prod_{k=1}^{d}(V(H_3(t_k)))^{sw_{0k}}) \cdot e(V(H_3(t_0))^{r'+r_{01}+\ldots+r_{0i}}, g^s)^{w_0*}} \\
&= \frac{e(g,g)^{a(r'-r_a+r_{01}-\lambda_1{}'+\ldots+r_{0i}-\lambda_i{}')s}}{e(g,g)^{s(r'+r_{01}+\ldots+r_{0i})(\sum_{k=1}^{d} w_{0k}\cdot q(H_3(t_k))+w_0*q(H_3(t_0)))}} \\
&= \frac{e(g,g)^{a(r'-r_a+r_{01}-\lambda_1{}'+\ldots+r_{0i}-\lambda_i{}')s}}{e(g,g)^{s(r'+r_{01}+\ldots+r_{0i})a}} \\
&= e(g,g)^{a(-r_a-\lambda_1{}'-\ldots-\lambda_i{}')s}, \\
\widetilde{B} &= \prod_{j=1}^{i} \frac{e(kp_j{}^1, C_0)}{e(kp_j^3, \prod_{k=1}^{d}(C_3^k)^{w_jk}) \cdot e(kp_j^{(2)}, C_0)^{w_j*}} \\
&= \prod_{j=1}^{i} \frac{e((g^a)^{\lambda_j{}'+r_{1j}}, g^s)}{e(g^{r_{1j}}, \prod_{k=1}^{d}(V(H_3(t_k)))^{sw_{jk}}) \cdot e(V(H_3(t_0))^{r_{1j}}, g^s)^{w*}} \\
&= \prod_{j=1}^{i} \frac{e(g,g)^{a(\lambda_j{}'+r_{1j})s}}{e(g,g)^{sr_{1j}(\sum_{k=1}^{d} w_{jk}\cdot q(H_3(t_k))+w_j*q(H_3(t_0)))}} \\
&= \prod_{j=1}^{i} \frac{e(g,g)^{a(\lambda_j{}'+r_{1j})s}}{e(g,g)^{sr_{1j}a}} \\
&= e(g,g)^{a(\lambda_1{}'+\ldots+\lambda_i{}')s},
\end{aligned}$$

$$B = \overline{B} \cdot \widetilde{B} = e(g,g)^{-r_a as}.$$

After that, the DU executes the second-level decryption. Output

$$\begin{aligned}
M &= \frac{C_0}{A \cdot B} \\
&= \frac{M \cdot e(g,g)^{\alpha s}}{e(g,g)^{\alpha s} \cdot e(g,g)^{a r_a s} \cdot e(g,g)^{-r_a as}}.
\end{aligned}$$

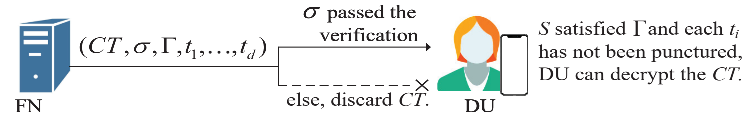
$(CT, \sigma, \Gamma, t_1, \ldots, t_d)$
σ passed the verification
else, discard CT.
FN
DU
S satisfied Γ and each t_i has not been punctured, DU can decrypt the CT.

Figure 6. Latest fine-grained firmware access in FSFU.

(5) *Revocation of decryption capability of latest firmware:* After revealing the latest firmware and successfully updating it, as shown in Figure 7, the DU can revoke the capability to decrypt the ciphertext through the Puncture algorithm, as shown below:

– Puncture(KP_{i-1}, t): First, parse $KP_{i-1} = (kp_0, kp_1, \ldots, kp_{i-1})$, and then, parse $kp_0 = [kp_0^1, kp_0^2, kp_0^3, kp_0^4]$. The DU selects $\lambda_i{}', r_{0i}, r_{1i} \in \mathbb{Z}_p^*$ at random and lets

$$\begin{aligned}
kp_0{}' &= [kp_0^1 \cdot (g^a)^{r_{0i}-\lambda_i{}'}, kp_0^2 \cdot V(H_3(t_0))^{r_{0i}}, kp_0^3 \cdot g^{r_{0i}}, t_0] \\
&= [(g^a)^{r'-r_a+r_{01}-\lambda_1{}'+\ldots+r_{0i}-\lambda_i{}'}, V(H_3(t_0))^{r'+r_{01}+\ldots+r_{0i}}, g^{r'+r_{01}+\ldots+r_{0i}}, t_0].
\end{aligned}$$

$$kp_i = [(g^a)^{\lambda_i{}' + r_{1i}}, V(H_3(t_0))^{r_{1i}}, g^{r_{1i}}, t].$$

Finally, the updated key is

$$KP_i = (kp_0{}', kp_1, \ldots, kp_{i-1}, kp_i).$$

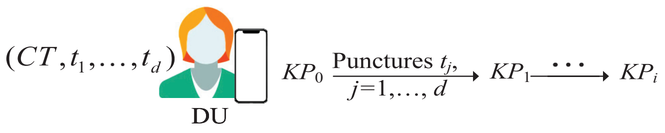

Figure 7. Revocation of decryption capability of latest firmware in FSFU.

5.2. Extended Construction

When firmware updates occur in the smart home system, a major problem is the limited computing power and memory of the data user nodes in the network. In the cloud–fog paradigm, some operations and parameters can be processed by fog nodes to reduce the computing and memory load of participants. In this section, we extend the basic FSFU to outsource some of the computation and parameters to the fog nodes; see Figure 8. The extended construction is described below:

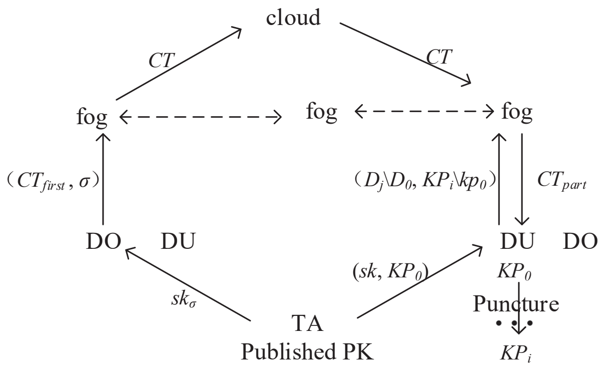

Figure 8. Extended system with outsourcing.

(1) *Initialization:* It is identical to the basic system.

(2) *Authorization:* It is identical to the basic system, except that the TA chooses an additional random element $n_0 \in \mathbb{Z}_p^*$ associated with a virtual attribute and generates $D_0 = g^{rn_0^{-1}}$.

(3) *Latest secure firmware delivery:* It is identical to the basic scheme, except:

- The DO generates the first-level ciphertext CT_{first} and a signature σ on it. Then, the DO passes (CT_{first}, σ) to the local fog node.
- The fog node verifies the signature as the DU did above. If the ciphertext CT_{first} is authenticated, the fog node and the DO perform the second-level encryption together to generate the final ciphertext.
- Second-level encryption: The DO divides s into s_0, s_1 such that $s = s_0 + s_1$. The DO generates $c_{0,0} = T_0^{s_0}$ and passes Γ and s_1 as the root value to the fog node to generate ciphertext for each leaf. The fog node generates $\{c_{j,i}, \forall a_{j,i} \in \Gamma\}$.

Finally, output

$$CT = (CT_{first}, c_{0,0} = T_0{}^{s_0}, c_{j,i} = T_j^{s_i}, \forall a_{j,i} \in \Gamma).$$

(4) *Latest fine-grained firmware access:* Some computations can be performed at the fog node. The details are described below:

- Decryption by fog node:
  * The fog node computes $\widetilde{A}$ through $\{c_{j,i} = T_j^{s_i}, \forall a_{j,i} \in \Gamma\}$ and $\{D_j, \forall a_j \in S\}$, which is received from the data-user-obtained attribute set S. The fog node computes $\widetilde{A} = e(g,g)^{rs_1}$.

- * The fog node computes $\widetilde{B} = e(g,g)^{a(\lambda_1{}' + \ldots + \lambda_i{}')s}$ by $\{C_3^k, k = 1, \cdots, d\}$ and $KP_i \backslash kp_0$.
  * $CT_{part} = (C_0, C_1, c_{0,0} = T_0^{s_0}, \widetilde{A}, \widetilde{B})$, then the fog node passes CT_{part} to the data user.
- Decryption by the DU: After receiving CT_{part}, the DU computes

$$\begin{aligned}\overline{A} &= e(T_0^{n_0}, D_0) \cdot \widetilde{A} \\ &= e(g^{s_0 n_0}, g^{r n_0^{-1}}) \cdot e(g,g)^{r s_1} \\ &= e(g,g)^{rs}, \\ A &= e(C_1, D) \cdot \overline{A},\end{aligned}$$

$$\overline{B} = \frac{e(kp_0^1, C_1)}{e(kp_0^3, \prod_{k=1}^{d} (C_3^k)^{w_{0k}}) \cdot e(kp_0^2, C_1)^{w_0{}^*}},$$

and $B = \overline{B} \cdot \widetilde{B} = e(g,g)^{-r_a a s}$.

Finally, the DU can obtain the plaintext:

$$M = \frac{C_0}{A \cdot B}.$$

(5) *Revocation of decryption capability of latest firmware:* This process is identical to the basic system, except that the DO keeps kp_0 and passes $KP_i \backslash kp_0$ to the fog node for external decryption.

6. Security and Performance Analysis

6.1. Security Analysis

Our basic P-CP-ABE scheme proposed above includes the CP-ABE scheme, the identity-based signature and puncturable encryption scheme, so its security is related to these schemes. The security analysis is given below.

6.2. Authentication of P-CP-ABE

Lemma 1. *Suppose the CDH assumption holds in $\mathbb{G}_1$; an IBS scheme is secure against existential forgery [31], and so is the P-CP-ABE scheme.*

Since the signature is attached to the components associated with the message in the ciphertext, the authenticity of the source of the message can be verified by this signature.

6.3. Data Confidentiality of P-CP-ABE

Lemma 2 ([14])**.** *Suppose that the DBDH assumption holds. Then, no polynomial adversary can break the CP-ABE scheme with a challenge access tree Γ^*.*

Theorem 1. *Our basic P-CP-ABE scheme proposed above is selectively CPA-secure in the selective game mentioned in Section 4, based on the assumption in Lemma 2.*

Proof. Suppose that there is an adversary $\mathcal{A}$ who can break our basic P-CP-ABE scheme with a non-negligible advantage ϵ. $\mathcal{A}$ can be used to break the CP-ABE scheme proposed in [14], which we will denote as $\Pi_{CP} = (\mathsf{Setup}_{CP}, \mathsf{KeyGen}_{CP}, \mathsf{Encrypt}_{CP}, \mathsf{Decrypt}_{CP})$, with a simulator $\mathcal{B}$. The simulator plays the role of the challenger and interacts with $\mathcal{A}$.

Init: $\mathcal{A}$ gives $\mathcal{B}$ the challenge access tree Γ^* together with the tag set $\{t_1^*, \ldots, t_d^*\}$ before the public parameters are established. Then, $\mathcal{B}$ forwards them to Π_{CP}.

Setup: $\mathcal{B}$ receives $PK = (e, g, g^a, g^c, y = e(g,g)^{\alpha}, T_j = g_j^{n}, (j = 1, \ldots, n)$ from Π_{CP}. First, $\mathcal{B}$ initializes a tag set $P = \phi$ and a counter $n = 0$. Then, $\mathcal{B}$ uniformly chooses $d+1$ random elements $\theta_0, \theta_1, \ldots, \theta_d$ from $\mathbb{Z}_p^*$, where θ_0 is associated with t_0. This is performed so that $q(0) = a, q(H_3(t_i^*)) = \theta_i$ is implicitly determined, and then, $V(H_3(t_i^*)) = g^{\theta_i}$. $\mathcal{B}$ passes public parameters $PK = (e, g, g^a, y = e(g,g)^\alpha, T_j(j = 1, \ldots, n), g^q(l)(l = 1, \ldots, d), t_0)$ to $\mathcal{A}$.

Phase 1: $\mathcal{A}$ can adaptively issue a polynomially bounded number of queries for any of the following:

- KeyGen(S) query: $\mathcal{B}$ requests the decryption key for the attribute set S from Π_{CP} and contains $sk_S = (d_0, \forall a_j \in S : D_j)$ with the form $d_0 = g^{\alpha - r}$. Then, $\mathcal{B}$ uniformly selects random elements $r_a, r' \in \mathbb{Z}_p^*$ and computes $D = d_0 \cdot (g^a)^{r_a}$ and $kp_0^1 = (g^a)^{-r_a + r'}$, $kp_0^2 = g^{\theta_0 r'}, kp_0^3 = g^{r'}, kp_0^4 = t_0$. $\mathcal{B}$ sends $(sk = (D, \forall a_j \in S : D_j)$, $KP_0 = ([kp_0^1, kp_0^2, kp_0^3, kp_0^4]))$ to $\mathcal{A}$.
- Puncture(t) query: $\mathcal{B}$ increments n and runs Puncture(KP_{n-1}, t), sends KP_i to $\mathcal{A}$, and sets $P = P \cup \{t\}$.
- Corrupt() query: If this is the first time $\mathcal{A}$ issues this query and $\{t_1^*, \ldots, t_d^*\} \cap P = \emptyset$, $\mathcal{B}$ sends the current secret key KP_n to $\mathcal{A}$ and sets $C \leftarrow P$. In all other cases, Corrupt() returns $\perp$.

Challenge: $\mathcal{A}$ sends two equal-length messages M_0, M_1 to $\mathcal{B}$. $\mathcal{B}$ passes them to Π_{CP}. Π_{CP} flips a fair coin $\beta \in 0, 1$ and sends the challenge ciphertext $ct = (C_0, C_1, \forall a_{j,i} \in \Gamma^*, c_{j,i})$ of the form $C_0 = M_\beta \cdot e(g,g)^{\alpha c}, C_1 = g^c$, to $\mathcal{B}$. Then, $\mathcal{B}$ computes $C_3^k = (C_0)^{\theta_k}, k = 1, \cdots, d$ and sends the challenge ciphertext:

$$CT^* = (ct, \{C_3^k, k = 1, \cdots, d\})$$

to $\mathcal{A}$.

Phase 2: It is identical to Phase 1.

Guess: $\mathcal{B}$ decides its own output based on the output of $\mathcal{A}$. When $\beta' \in \{0, 1\}$ is the guess of $\mathcal{A}$, $\mathcal{B}$ outputs the same β' as the guess of β. Thus, $\mathcal{B}$ has the same advantage as ϵ to break the CP-ABE scheme Π_{CP}. $\square$

6.4. Collusion Resistance

In order to reveal a message from the encrypted data, $e(g,g)^{\alpha s}$ should be recovered, and the knowledge of $e(g,g)^{rs}$ is a precondition. Due to the random element r for each DU in the private key distribution phase, it is impossible to combine keys generated for different attribute sets to reconstruct $e(g,g)^{rs}$ because of the different r.

6.5. Forward Security

For a ciphertext concatenated with a list of tags $t_1, \ldots, t_d$, without loss of generality, assume that the puncturable key KP has already punctured t_1 such that $kp_1^4 = t_1$. As a result, whoever kept KP cannot find $w_{11}, \ldots, w_{1d}, w_{1*}$ such that $w_{1*} \cdot q(H_3(kp_1{}^4)) + \sum_{k=1}^{d} (w_{1k} \cdot q(H_3(t_k))) = q(0) = a$. Furthermore, $\widetilde{B}$ cannot be recovered, and it causes KP to lose its capability to decrypt the past ciphertext. This guarantees forward security.

6.6. Security Analysis of Extended Scheme

In the extended scheme, the DO outsources the second-level encryption, which is the encryption for the leaf nodes in the access tree, to the fog node and passes a part of the secret number to it as the value of the root node during encryption. The DU transmits part of the private key components to the fog node, and the fog node performs the external decryption operation, then transmits CT_{part} to the DU such that the DU can perform the final decryption to reveal the plaintext. The extended scheme is secure under the discrete logarithm assumption on the elliptic curve.

6.7. Performance Analysis

The security of the proposed schemes was proven above; in this section, the performance analysis focused on the computational efficiency by comparing the cost of encryption and decryption. Tables 2 and 3 show the feature comparison and performance comparison of different schemes, where the communication cost is for the DO and the DU. In the above tables, Γ is the access tree, l is the row number of the access matrix, while n_{max} is the maximum column number, I is the index set of the authorized attribute set, S' is the smallest attribute subset that satisfies Γ, d is the maximum tag number, and i is the puncture numbers.

Table 2. Feature comparison of different schemes.

Scheme	Access Structure	Outsource Ability	Authentication	Forward Security
Original CP-ABE [14]	Tree	$\times$ [1]	$\times$	$\times$
Pt-CP-ABE [28]	Matrix	$\times$	$\times$	$\surd$ [2]
Our basic scheme	Tree	$\times$	$\surd$	$\surd$
Our extended scheme	Tree	$\surd$	$\surd$	$\surd$

[1] $\times$ indicates that the scheme does not has this capability. [2] $\surd$ indicates that the scheme has this capability.

Table 3. Performance comparison of different schemes.

Scheme	Ciphertext Size		Decryption Cost			Communication Cost	
	$\lvert\mathbb{G}_1\rvert$	$\lvert\mathbb{G}_2\rvert$	$Exp.(\mathbb{G}_1)$	$Exp.(\mathbb{G}_2)$	Pairing	DO	DU
Pt-CP-ABE [28]	$l \times n_{max} + 1 + d$	1	$\lvert I\rvert \times n_{max} + \lvert I\rvert$	/	$n_{max} + \lvert I\rvert + 1 + 3(i+1)$	$(l \times n_{max} + 1 + d)\lvert\mathbb{G}_1\rvert + \lvert\mathbb{G}_2\rvert$	$(l \times n_{max} + 1 + d)\lvert\mathbb{G}_1\rvert + \lvert\mathbb{G}_2\rvert$
Our basic scheme	$\lvert\Gamma\rvert + 3 + d$	1	$d(i+1)$	$\lvert S'\rvert + i + 1$	$S' + 4 + 3(i+1)$	$(\lvert\Gamma\rvert + 3 + d)\lvert\mathbb{G}_1\rvert + \lvert\mathbb{G}_2\rvert$	$(\lvert\Gamma\rvert + 3 + d)\lvert\mathbb{G}_1\rvert + \lvert\mathbb{G}_2\rvert$
Our extended scheme	$2 + d$	1	d	1	5	$(2+d)\lvert\mathbb{G}_1\rvert + \lvert\mathbb{G}_2\rvert + \lvert\mathbb{Z}_p\rvert$	$(5 + \lvert S'\rvert)\lvert\mathbb{G}_1\rvert + 3\lvert\mathbb{G}_2\rvert + \lvert tag\rvert$

The cost of our basic scheme is much lower than that of Pt-CP-ABE [28], as can be clearly seen from Table 3, since in the latter, the product of each element of the access matrix with the corresponding component of the vector is used in encryption and decryption. Since key storage and computation are outsourced, the overhead of the extended scheme does not depend on the size of the universe of attributes and the puncturable numbers, as can be seen from Table 3. Participants may have to make a tradeoff between communication overhead and computation and storage costs, depending on the device.

Then, we conducted the test on a PC: the computer operating system used was Windows 10 Home Chinese Edition, and the processor was an Intel(R) Core(TM) i5-7200U CPU @ 2.50 GHz and 16 GB RAM. The test was based on a 160-bit elliptic curve group constructed on the curve $y^2 = x^3 + x$ over a 512-bit field.

The computation time to generate an initial puncturable key KP_0 and the subsequent puncturable key are listed in Table 4. In smart homes, the latest firmware is not published very often in a short period of time, so the puncturable time is reasonable. It can be seen in Figure 9 that, despite the introduction of the signature and puncturable key, the encryption and decryption were still efficient. In contrast, in the Pt-CP-ABE scheme, both the encryption time and decryption time grew very fast, and the trend was the square of the number of attributes, as analyzed above. In conclusion, our proposed FSFU is efficient from both the theoretical analysis and the experimental results.

Table 4. Puncturable key generation time.

Operation	Time (ms)
Generation for KP_0	57.21
Generation for $KP_{i-1} \rightarrow KP_i$	90.06

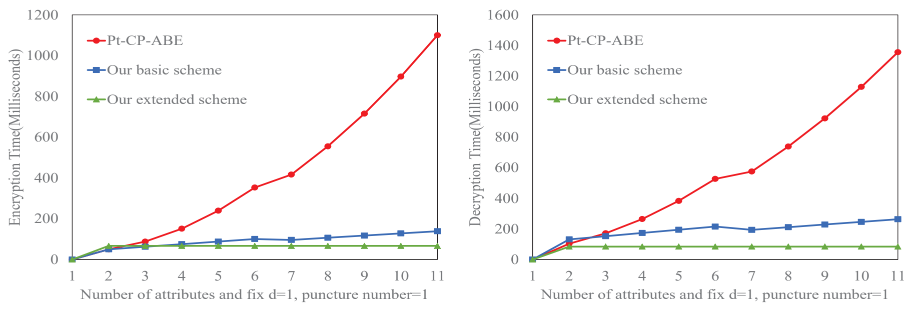

Figure 9. Comparison of encryption and decryption times.

7. Conclusions and Future Work

To enable secure firmware updates in smart home systems, we proposed an FSFU system that invokes the proposed Puncturable-Ciphertext Policy-Attribute-Based Encryption (P-CP-ABE) scheme. In addition, we described the extended version, where some of the computations and parameters were outsourced to fog nodes under the cloud–fog paradigm. In practice, user attributes and firmware tags are often monitored by different independent authorities. In addition, a multi-authority system can better protect the privacy of user identities than a single trusted authority. Key distribution and privacy protection in a multi-authority puncturable-ciphertext policy-attribute-based encryption scheme is a challenging problem for future research.

Author Contributions: Conceptualization, Q.Z. and Y.Z.; methodology, Q.Z.; validation, Y.Z.; formal analysis, Y.Z.; resources, D.Z.; writing—original draft preparation, Q.Z.; supervision, D.Z. and Y.Z.; writing—review and editing, Y.Z. and Y.R. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded in part by the National Natural Science Foundation of China (62072369, 62072371), in part by the Youth Innovation Team of Shaanxi Universities, in part by the Shaanxi Special Support Program Youth Top-notch Talent Program, and in part by the Technology Innovation Leading Program of Shaanxi (2023-YD-CGZH-31).

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

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

References

1. Yaqoob, I.; Ahmed, E.; Hashem, I.A.T.; Ahmed, A.I.A.; Gani, A.; Imran, M.; Guizani, M. Internet of Things Architecture: Recent Advances, Taxonomy, Requirements, and Open Challenges. *IEEE Wirel. Commun.* **2017**, *24*, 10–16. [CrossRef]
2. Mehmood, Y.; Ahmad, F.; Yaqoob, I.; Adnane, A.; Imran, M.; Guizani, S. Internet-of-Things-Based Smart Cities: Recent Advances and Challenges. *IEEE Commun. Mag.* **2017**, *55*, 16–24. [CrossRef]
3. Samsung. SmartThing. Available online: http://www.smartthings.com/ (accessed on 12 July 2023).
4. Amazon. AWS. Available online: https://aws.amazon.com/cn/iot/ (accessed on 12 July 2023).
5. Apple. Home-App. Available online: https://www.apple.com/home-app/ (accessed on12 July 2023).
6. Alibaba. AliyunloT. Available online: https://iot.aliyun.com/ (accessed on 12 July 2023).
7. Lee, B.; Lee, J.H. Blockchain-based secure firmware update for embedded devices in an Internet of Things environment. *J. Supercomput.* **2017**, *73*, 1152–1167. [CrossRef]
8. Bettayeb, M.; Nasir, Q.; Talib, M.A. Firmware Update Attacks and Security for IoT Devices: Survey. In Proceedings of the the ArabWIC 6th Annual International Conference Research Track (ArabWIC 2019), Rabat, Morocco, 7–9 March , 2019; Volume 4, pp. 1–6.
9. Han, S.; Topcu, U.; Pappas, G.J. Differentially Private Distributed Constrained Optimization. *IEEE Trans. Autom. Control* **2017**, *62*, 50–64. [CrossRef]
10. An, L.; Yang, G.H. Enhancement of opacity for distributed state estimation in cyber-physical systems. *Automatica* **2022**, *136*, 110087. [CrossRef]
11. Sahai, A.; Waters, B. Fuzzy identity-based encryption. In Proceedings of the Advances in Cryptology–EUROCRYPT 2005, Aarhus, Denmark, 22–26 May; Springer: Berlin/Heidelberg, Germany, 2005; pp. 457–473.

12. Goyal, V.; Pandey, O.; Sahai, A.; Waters, B. Attribute-based encryption for fine-grained access control of encrypted data. In Proceedings of the 13th ACM conference on Computer and communications Security, Alexandria, VA, USA, 30 October–3 November 2006; Association for Computing Machinery: New York, NY, USA, 2006; pp. 89–98.
13. Bethencourt, J.; Sahai, A.; Waters, B. Ciphertext-Policy Attribute-Based Encryption. In Proceedings of the 2007 IEEE Symposium on Security and Privacy (SP '07), Berkeley, CA, USA, 20–23 May 2007; pp. 321–334.
14. Ibraimi, L.; Tang, Q.; Hartel, P.; Jonker, W. Efficient and provable secure ciphertext-policy attribute-based encryption schemes. In Proceedings of the Information Security Practice and Experience, ISPEC 2009, Xi'an, China, 13–15 April 2009; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2009; pp. 1–12.
15. Zhang, Y.; Deng, R.H.; Xu, S.; Sun, J.; Li, Q.; Zheng, D. Attribute-Based Encryption for Cloud Computing Access Control: A Survey. *ACM Comput. Surv.* **2020**, *53*, 1–41 . [CrossRef]
16. Choi, B.C.; Lee, S.H.; Na, J.C.; Lee, J.H. Secure firmware validation and update for consumer devices in home networking. *IEEE Trans. Consum. Electron.* **2016**, *62*, 39–44. [CrossRef]
17. Zaware, P.G.; Shinde, S.V. Wireless monitoring, controlling and firmware upgradation of embedded devices using Wi-Fi. In Proceedings of the 2014 International Conference on Advances in Communication and Computing Technologies (ICACACT 2014), Mumbai, India, 10–11 August 2014; pp. 1–6.
18. Hong, S.G.; Kim, N.S.; Heo, T. A smartphone connected software updating framework for IoT devices. In Proceedings of the 2015 International Symposium on Consumer Electronics (ISCE), Madrid, Spain, 24–26 June 2015; pp. 1–2.
19. Lee, B.; Malik, S.; Wi, S.; Lee, J.H. Firmware verification of embedded devices based on a blockchain. In Proceedings of the Quality, Reliability, Security and Robustness in Heterogeneous Networks, QShine 2016, Seoul, Republic of Korea, 7–8 July 2016; Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering; Springer: Berlin/Heidelberg, Germany, 2017; pp. 52–61.
20. Roy, G.G.R.; Britto Ramesh Kumar, S. An Architecture to Enable Secure Firmware Updates on a Distributed-Trust IoT Network Using Blockchain. In *International Conference on Computer Networks and Communication Technologies*; Springer: Singapore, 2019; pp. 671–679.
21. Schneier, B.; Hall, C. An improved e-mail security protocol. In Proceedings of the 13th Annual Computer Security Applications Conference, San Diego, CA, USA, 8–12 December 1997; pp. 227–230.
22. Barreto, P.S.L.M.; Naehrig, M. Pairing-Friendly Elliptic Curves of Prime Order. In Proceedings of the Selected Areas in Cryptography, Kingston, ON, Canada, 11–12 August 2005; Springer: Berlin/Heidelberg, Germany, 2006; pp. 319–331.
23. Sun, H.M.; Hsieh, B.T.; Hwang, H.J. Secure E-mail protocols providing perfect forward secrecy. *IEEE Commun. Lett.* **2005**, *9*, 58–60.
24. Santesson, S.; Myers, M.; Ankney, R.; Malpani, A.; Galperin, S.; Adams, C. *X. 509 Internet Public Key Infrastructure Online Certificate Status Protocol-OCSP*; Technical Report; Internet Engineering Task Force: Fremont, CA, USA, 2013.
25. Canetti, R.; Halevi, S.; Katz, J. A forward-secure public-key encryption scheme. In Proceedings of the Advances in Cryptology–EUROCRYPT 2003, Warsaw, Poland, 4–8 May 2003; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2003; pp. 255–271.
26. Green, M.D.; Miers, I. Forward Secure Asynchronous Messaging from Puncturable Encryption. In Proceedings of the 2015 IEEE Symposium on Security and Privacy, San Jose, CA, USA, 17–21 May 2015; pp. 305–320.
27. Wei, J.; Chen, X.; Wang, J.; Hu, X.; Ma, J. Forward-Secure Puncturable Identity-Based Encryption for Securing Cloud Emails. In Proceedings of the Computer Security—ESORICS 2019, Luxembourg, 23–27 September 2019; Springer International Publishing: Cham, Switzerland, 2019; pp. 134–150.
28. Xuan Phuong, T.V.; Ning, R.; Xin, C.; Wu, H. Puncturable Attribute-Based Encryption for Secure Data Delivery in Internet of Things. In Proceedings of the IEEE INFOCOM 2018—IEEE Conference on Computer Communications, Honolulu, HI, USA, 16–19 April 2018; pp. 1511–1519.
29. Xue, L.; Ni, J.; Huang, C.; Lin, X.; Shen, X. Forward Secure and Fine-grained Data Sharing for Mobile Crowdsensing. In Proceedings of the 2019 17th International Conference on Privacy, Security and Trust (PST), Fredericton, NB, Canada, 26–28 August 2019; pp. 1–9.
30. Sun, J.; Xu, G.; Zhang, T.; Alazab, M.; Deng, R.H. A Practical Fog-Based Privacy-Preserving Online Car-Hailing Service System. *IEEE Trans. Inf. Forensics Secur.* **2022**, *17*, 2862–2877. [CrossRef]
31. Gentry, C.; Ramzan, Z. Identity-based aggregate signatures. In Proceedings of the Public Key Cryptography—PKC 2006, New York, NY, USA, 24–26 April 2006; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2006; pp. 257–273.