Puncturable Attribute-Based Encryption for Secure Data Delivery in Internet of Things

Tran Viet Xuan Phuong, Rui Ning, Chunsheng Xin, Hongyi Wu
Center for Cybersecurity Education and Research
Old Dominion University
Norfolk, Virginia, USA
{tphuong,rning001,cxin,h1wu}@odu.edu

***Abstract*—While the Internet of Things (IoT) is embraced as important tools for efficiency and productivity, it is becoming an increasingly attractive target for cybercriminals. This work represents the first endeavor to develop practical Puncturable Attribute Based Encryption schemes that are light-weight and applicable in IoTs. In the proposed scheme, the attribute-based encryption is adopted for fine grained access control. The secret keys are puncturable to revoke the decryption capability for selected messages, recipients, or time periods, thus protecting selected important messages even if the current key is compromised. In contrast to conventional forward encryption, a distinguishing merit of the proposed approach is that the recipients can update their keys by themselves without key re-issuing from the key distributor. It does not require frequent communications between IoT devices and the key distribution center, neither does it need deleting components to expunge existing keys to produce a new key. Moreover, we devise a novel approach which efficiently integrates attribute-based key and punctured keys such that the key size is roughly the same as that of the original attribute-based encryption. We prove the correctness of the proposed scheme and its security under the Decisional Bilinear Diffie-Hellman (DBDH) assumption. We also implement the proposed scheme on Raspberry Pi and observe that the computation efficiency of the proposed approach is comparable to the original attribute-based encryption. Both encryption and decryption can be completed within tens of milliseconds.**

***Index Terms*—Attribute-Based Encryption, Internet-of-Things, Lagrange Polynomial, Linear Secret Sharing**

I. Introduction

In recent years we have witnessed a remarkable proliferation of networked intelligent devices - collectively known as the Internet of Things (IoT). The IoT is a digitization of the physical world by embedding physical devices with electronics, sensors, actuators, and network connectivity that enable them to collect and exchange data without human intervention, and be controlled remotely across existing network infrastructure. It presents vast opportunities for organizations to improve efficiencies, gain a competitive advantage, and build new business models. More IoT devices are coming online each and every day. Experts estimate that the IoT will grow from 12 billion devices in 2015 to more than 50 billion by 2020 [1]. While IoT is embraced as important tools for efficiency and productivity, it is becoming an increasingly attractive target for cybercriminals. Securing IoT is fundamentally challenging due to the vast number of networked devices that are often resource-constrained and exposed in unprotected environment.

A. Attribute Based Encryption (ABE) for IoTs

While it is highly desired to secure data communication between IoT devices, the cost is overwhelming to establish and manage a sheer number of secure connections between each pair of devices. Fortunately, it is unnecessary to always control data access on a per connection basis. In contrast, many data in IoT are accessible by a group of users. The sender does not need to specify the IDs of the receivers. In fact, it is often impossible to do so, since the group of receivers are often dynamic and even unknown, but can only be described by certain properties. For example, in autonomous vehicular networks, an authorized organization or individual may want to send secure messages to a type of controllers on 4x4 vehicles in a region. How to route data packets to them is out of the scope of this work. However, no matter which approach is used for data delivery, we must ensure appropriate access control, i.e., only the targeted control boards can decrypt the data. Likewise, in an industrial IoT setting, a class of sensory data should be accessible by only authorized technicians with prescribed attributes; or in IoT crowdsensing applications, the crowdsensing initiator wants to send the request to qualified participants with the desired properties.

In this research we adopt the Attribute Based Encryption (ABE) for fine grained access control over encrypted data in IoT. Since its introduction in the seminal work of Sahai and Waters [2], ABE has been extensively studied in recent years (e.g., [3]–[10]). There are different ways to define an access structure/policy with ABE. For example, Sahai and Waters [2] presented the fuzzy Identity Based Encryption (IBE) using a specific threshold access policy, which can be treated as the first variant of ABE, the Key Policy-ABE (KP-ABE). Later on, the Linear Secret Sharing Scheme (LSSS) realizable (or monotone) access structure was adopted by several subsequent ABE schemes [3], [4], [6]–[9]. For example, in [3], data are associated with attributes for each of which a public key component is defined. An encryptor associates a set of attributes to the message by encrypting it with the corresponding public key components. Each user is assigned with an access structure, which is usually defined as an access tree over data attributes. A secret key is assigned to a user to reflect the access structure so that the user can decrypt a ciphertext if and only if the data attributes satisfy his access structure. On the contrary to variants of KP-ABE, another scheme, Ciphertext Policy-ABE (CP-ABE) [4], associates a secret key with a user's credentials, and a ciphertext is associated with access policies. If a decryptor wants to decrypt a message successfully, the attributes embedded in the secret key must satisfy the access policies embedded in the ciphertext. In addition, several other access structures can be found in [11]–[19].

B. New Challenges in ABE for Large-Scale IoTs

ABE can effectively support secure data delivery to a group of dynamic or even unknown IoT nodes described by certain attributes. However, the direct application of ABE in IoT induces new challenges. More specifically, a long-term secret key is undesired. An adversary can patiently record encrypted data and wait for opportunities to compromise the secret key. Once it is compromised, all encrypted communications and sessions recorded in the past can be retrieved and decrypted, thus losing confidentiality. An adversary can also record the input and output batches of compromised messages and then replay them. The messages' decryption will remain the same.

To block such attacks, one can add random nonce from timestamps or update keys to revoke their decryption capability for past messages. This property can be seen in the forward encryption [20], [21] which is designed to prevent the compromise of a long-term secret key from affecting the confidentiality of past conversations. Under forward secrecy, a compromised key at time t does not break the confidentiality of the communication that took place prior to t. However, the secret key updating algorithm in the forwarding encryption requires to communicate with the key distribution center, leading to significant cost which is often unaffordable in IoT. With thousands or even hundreds of thousands of IoT devices, frequent secret key updating through a key distribution center is simply impractical.

Motivated by the recent development in puncturable encryption [22], we propose to construct a new ABE scheme, to address the above challenges. The nodes in this proposed scheme repeatedly update their decryption keys to revoke the decryption capability for selected messages, recipients, or time periods. In contrast to forward encryption, a distinguishing merit of the proposed approach is that the recipients can update their keys by themselves without key re-issuing from the key distributor. It does not require frequent communication with the key distribution center, neither does it need deleting components to expunge existing keys to produce a new key.

C. Summary of Our Contributions

This work represents the first endeavor to develop practical Puncturable Attribute Based Encryption schemes that are light-weight and applicable in IoTs. In particular, we propose two new ABE encryption systems named Puncturable-Ciphertext Policy-Attribute Based Encryption (Pt-CP-ABE) and Puncturable-Key Policy-Attribute Based Encryption (Pt-KP-ABE).

The basic idea is to incorporate the principles of attribute-based encryption and puncturable encryption schemes to achieve puncturable attribute based encryption. However, it is nontrivial to realize it efficiently. A naive thought for construction of such puncturable-ABE is to simply combine the two systems, i.e., an ABE and a tag based encryption [22], [23]. However, there does not exist a straightforward approach to integrate the attribute-based key and the punctured keys, because they are generated separately but must be integrated in a coherent way based on the same master random exponent. Moreover, even if this naive approach would be possible, the key size would be the sum of embedded number of attributes or access structure and the punctured keys. The latter is often large, thus resulting in highly inefficient encryption and decryption.

In order to develop practical puncturable attribute based encryption schemes that are applicable in real-world IoTs, we devise a novel approach which efficiently integrates attribute-based key and punctured keys and has the key size being roughly the same as that of the original ABE algorithms. Briefly, to generate a key pair, a node first produces parameters for an ABE user key, which embeds one share (r_a) of the master share α. Then it re-shares r_a from the ABE key by regenerating a share r to create an initial punctured key. By following this way, we can keep the origin of the ABE key, and re-produce an initial key for puncturable encryption. This approach ensures that the user can derive the punctured key without destroying the information of the initial key.

Based on this idea, we propose two new puncturable attribute-based encryption systems. In the Pt-CP-ABE system, a sender embeds a Linear Secret Sharing access structure $\mathbb{A}$ and attaches a set of tags $t_1, \ldots, t_d$ in the ciphertext. The tag spaces can be the unique message identifiers or timestamps supplementing the sender's ID. A receiver's secret key is an ABE decryption key embedding its attribute set ω and binding a tag t. The decryption will be successful for a ciphertext if ω satisfies $\mathbb{A}$, except the ciphertexts encrypted with t, which cannot be decrypted. As a result, a receiver can securely revoke decryption capability by puncturing the secret key on selected tag(s). As the dual form of Pt-CP-ABE, the roles of ω and $\mathbb{A}$ are swapped in Pt-KP-ABE. The attribute set ω is embedded into the ciphertext, and the access structure $\mathbb{A}$ is embedded into the secret key.

We prove the correctness of the proposed schemes and their security under the Decisional Bilinear Diffie-Hellman (DBDH) assumption. The security proof is nontrivial under the chosen plaintext attack. We have also implemented the proposed schemes on Raspberry Pi. The implementation is built on the JPBC [24] library to execute the design setup, key generation, encryption, punctuation, and decryption algorithms. We observe that the computation efficiency of the proposed approach is comparable to the original ABE scheme.

The rest of the paper is organized as follows. Sec. II introduces the preliminaries about access structures, linear secret sharing, and Lagrange polynomial and interpolation. Sec. III presents the proposed Puncturable-Key Policy-Attribute Based Encryption (Pt-CP-ABE) including the basic definitions, algorithm construction, and correctness and security proof. Sec. IV discusses the Puncturable-Key Policy-Attribute Based Encryption (Pt-KP-ABE) scheme. Sec. V presents performance evaluation and results. Finally, Sec. VI concludes the paper.

II. Preliminaries

A. Access Structures and Linear Secret Sharing

1) Access Structures: 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 $\mathbb{A}$ of non-empty subsets of $\{P_1, P_2, \ldots P_n\}$, i.e, $\mathbb{A} \subseteq 2^{\{P_1, P_2, \ldots, P_n\}} \backslash \{\emptyset\}$. The sets in $\mathbb{A}$ are called the authorized sets, and the sets not in $\mathbb{A}$ are called the unauthorized sets.

2) Linear Secret Sharing Scheme (LSSS): A secret sharing scheme II over a set of parties $\mathcal{P}$ is called linear over $\mathbb{Z}_p$ if

- The shares of each party form a vector over $\mathbb{Z}_p$.
- There exists a matrix M with ℓ rows and n columns called the share-generating matrix for II. For the i'th row of M, $i = 1, \ldots, l$, we let the function ρ label row i as $\rho(i)$, which also defines the party. When we consider the column vector $v = (s, r_2, \ldots, r_n)$, where $s \in \mathbb{Z}_p$ is the secret to be shared, and $r_2, \ldots r_n \in \mathbb{Z}_p$ are randomly chosen, then Mv is the vector of l shares of the secret s according to II. The share $(Mv)_i$ belongs to party $\rho(i)$.

Linear reconstruction: Suppose that II is an LSSS for the access structure $\mathbb{A}$. Let $S \in \mathbb{A}$ be any authorized set, and let $I \subset \{1, 2, \ldots, \ell\}$ be defined as $I = \{i : \rho(i) \in S\}$. Then there exist constants $\{\omega_i \in \mathbb{Z}_p\}_{i \in I}$ such that, if $\{\lambda_i\}$ are valid share of any secret s according to II, then $\sum_{i \in I} \omega_i \lambda_i = s$.

Lemma 1: Let (M, π) be an LSSS for access structure $\mathbb{A}$ over set of parties $\mathcal{P}$, where M is a matrix of size $\ell \times \mathsf{k}$. For all $S \notin \mathbb{A}$, there exists a polynomial time algorithm that outputs a vector $\vec{w}= w_1, w_2, \ldots, w_{\mathsf{k}} \in \mathbb{Z}_p^{\mathsf{k}}$ such that $w_1 = 1$ and for all $i \in [1, \ell]$ where $\pi(i) \in S$ it holds that $M_i \cdot w = 0$.

3) Bilinear Map and Its Assumption: Let $\mathbb{G}$ and $\mathbb{G}_{\mathbb{T}}$ be two multiplicative cyclic groups of same prime order p, and let g be a generator of $\mathbb{G}$. Let $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ be a 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* $\in \mathbb{G}$ and *a*,*b* $\in \mathbb{Z}_p$.
- Non-degeneracy : $e(g, g) \neq 1$.

Notice that the map e is symmetric since $e(g^a, g^b) = e(g, g)^{ab} = e(g^b, g^a)$.

4) Decisional Bilinear Diffie-Hellman (DBDH) Assumption: Let $\mathbb{G}$ be a bilinear group of prime order p. Given a tuple $(g, g^a, g^b, g^c) \in \mathbb{G}^4$ and an element $Z \in \mathbb{G}_T$ as input, the DBDH assumption holds in $\mathbb{G}$ if for any probabilistic polynomial-time algorithm A

$$|\Pr[A(g, g^a, g^b, g^c, e(g, g)^{abc}) = 0] \\ -\Pr[A(g, g^a, g^b, g^c, Z) = 0] \,| \leq \epsilon(\mathsf{k}),$$

where the probability is over the random choice of g in $\mathbb{G}$, the random choice $a, b, c \in \mathbb{Z}_p$, the random choice $Z \in \mathbb{G}_T$, and $\epsilon(k)$ is negligible in the security parameter k.

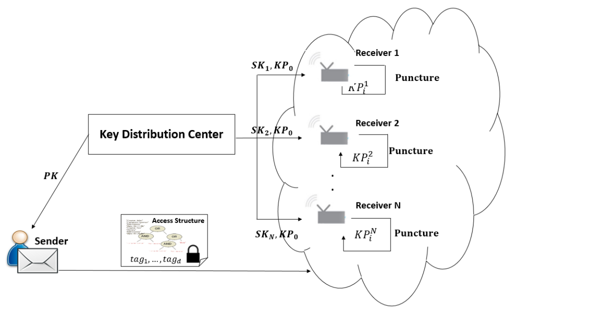

Fig. 1. An overview of Puncturable Ciphertext Policy Attribute Based Encryption (Pt-CP-AB) for IoTs, where Receiver 1 is a 4x4 Jeep with an engine control board of Model SM-02-E; Receiver 2 is a SUV with an engine control board of Model SM-02-A; Receiver N is an unknown vehicle with an engine control board of Model SM-02-X; and the sender is an authorized organization that intends to send secure messages to all Model SM-02 series engine control boards in a given region. The key distribution center is responsible to create and distribute the initial keys only, i.e., the public key PK for the sender, and the secret key $(SK_1, \cdots, SK_N)$ and initial punctured key (KP_0) to the receivers. The subsequent punctured keys are generated by individual receives without communication with the key distribution center.

B. Lagrange Polynomial and Interpolation

Suppose that a polynomial of degree d is uniquely defined by a set of points $(x_0, y_0), (x_1, y_1), \ldots, (x_{d+1}, y_{d+1})$. The Lagrange form of the polynomial allows the computation of a point x on the polynomial using only $d+1$ points as follows:

$$q(x) = L(x, xc, yc) = \sum_{j=0}^{d} (yc[i] \cdot \ell(x, i, xc)), \tag{1}$$

where $xc = [x_0, \ldots, x_{d+1}]$ and $yc = [y_0, \ldots, y_{d+1}]$ and the Lagrange basis polynomial $\ell(\ldots)$ is:

$$\ell(x, j, xc) = \prod_{0 \leq m, m \neq j \leq d} \frac{x - xc[m]}{xc[j] - xc[m]}. \tag{2}$$

Applying the Lagrange polynomial form, a random degree d polynomial $q(x)$ is selected, which consists of sampling d random values $r_1 \ldots, r_d$ from $\mathbb{Z}_p$, setting points $(1, r_1), (2, r_2), \ldots, (d, r_d)$ and setting the final point to $(0, \beta)$ to guarantee $q(0) = \beta$. Lagrange interpolation can compute $V(x)$ without knowledge of the polynomial coefficients by only the public values $g^{q(0)}, \ldots, g^{q(d)}$ as:

$$V(x) = g^{q(x)} = g^{\sum_{j=0}^{d} y_j \ell(x, j, xc)} = \prod_{j=0}^{d} (g^{q(i)})^{\ell(x, j, xc)},$$

where $\ell(x, j, xc)$ is already defined in Eq. (2).

III. Puncturable Ciphertext Policy Attribute Based Encryption for IoT devices

Fig. 1 illustrates an overall architecture of the Puncturable Ciphertext Policy Attribute Based Encryption (Pt-CP-AB) scheme. For example, assume the sender to be an authorized organization that intends to send secure messages to all Model SM-02 series engine control boards in a given region. Receivers are various vehicles. For instance, assume Receiver

1 is a 4x4 Jeep with an engine control board of Model SM-02-E; Receiver 2 is a SUV with an engine control board of Model SM-02-A; and Receiver N is a unknown vehicle with an engine control board of Model SM-02-X. The key distribution center is responsible to create and distribute the initial keys only.

The sender includes a linear access structure $\mathbb{A}$ (which corresponds to Model SM-02 series engine control boards in this example) and a tag (such as a message identifier or time period identifier) in each message sent to given IoT devices. At all times subsequent to initial key generation, an IoT device's secret key is an ABE decryption key embedding its attribute ω (e.g., Model SM-02-E, Model SM-02-A, Model SM-02-X, etc). The goal of the system is to allow the IoT devices, which have attributes ω satisfying the access structure $\mathbb{A}$, to decrypt the messages. Moreover, it allows the corresponding IoT devices to selectively revoke the ability to decrypt messages with specific tags. This is achieved by puncturing a key at a point t, i.e., the receiver updates its existing secret key to derive a new punctured key that embeds the tag t. In this way, even if the newly generated secret key that is used for communications is compromised, the adversary is not able to use it to decrypt other important messages embedded with t. Note that, the creation of punctured keys does not require communications with the key distribution center, neither does it need deleting components to expunge existing keys to produce a new key.

Next, we introduce the basic definitions and algorithm construction of Pt-CP-ABE, followed by correctness and security proofs.

A. Definition

A Pt-CP-ABE scheme is a tuple of five probabilistic algorithms with the following syntax:

- $\mathsf{Setup}(1^{\mathsf{k}}, d, \mathsf{n}_{max})$. This is a randomized algorithm that takes the inputs of a security parameter k and the maximum number of tags per ciphertext d. The parameter d also specifies how many attributes in the system, and n_{max} specifies the maximum number of columns in an LSSS matrix. It outputs the public key PK and master key MSK.
- $\mathsf{Encrypt}(\mathsf{PK}, M, \mathbb{A}, \{t_1, \ldots, t_d\})$. This is randomized algorithm that takes input public key PK, a message M, an access structure $\mathbb{A}$, and a list of tags $\{t_1, \ldots, t_d\}$. It outputs the ciphertext CT.
- $\mathsf{KeyGen}(\mathsf{PK}, \mathsf{MSK}, \omega)$. This is a randomized algorithm that takes an input public key PK, master key MSK, and an attribute set ω. It outputs a secret key SK and an initial punctured key KP_0. Note that, existing authentication schemes can be adopted to ensure that no one can forge attributes [4], [7].
- $\mathsf{Puncture}(\mathsf{PK}, \mathsf{KP}_{i-1}, t)$. This is a randomized algorithm that takes an input public PK, a punctured key KP_{i-1}, and a tag t. It outputs a new punctured key KP_i that can join to decrypt any ciphertexts, except for ciphertext encrypted under tag t.
- $\mathsf{Decrypt}(\mathsf{PK}, \mathsf{CT}, \mathsf{SK}, \mathsf{KP}_i)$. The decryption algorithm inputs a public key PK, a secret key SK, a punctured key KP_i, and ciphertext CT, It then outputs message M, or a symbol $\perp$ indicating a failure decryption.

The security notion for Pt-CP-ABE is defined by the IND-CPA game. We consider selective target security, where the adversary is required to specify the target attribute set, and the set of tags before receiving the public key. We provide the following formal definition as:

- **Init**. The adversary declares the target access structure and the set of tags as $(\mathbb{A}^*, \{t_1^*, \ldots, t_d^*\})$.
- **Setup**. On input the security parameter k and a maximum number of tags d (which also specifies the number of attributes in ciphertext), the challenger initializes two empty sets P and C, and a counter $n = 0$. It then runs $\mathsf{Setup}(1^{\mathsf{k}}, d) \rightarrow \mathsf{PK}, \mathsf{MSK}$, and gives the PK to the adversary.
- **Query Phase 1**. The adversary can repeatedly issue any of the queries, and the challenger answers these queries as follows:
  - The adversary queries for attribute set ω, where $\omega \not\models \mathbb{A}^*$. The challenger computes $\mathsf{KeyGen}(\mathsf{PK}, \mathsf{MSK}, \omega) \rightarrow (\mathsf{SK}, \mathsf{KP}_0)$.
  - The challenger increments n, computes $\mathsf{Puncture}(\mathsf{PK}, \mathsf{KP}_0, t) \rightarrow KP_n$, and adds t to the set P.
  - **Corrupt**() is called in the first time when the adversary issues this query. Then the challenger returns the most recent punctured key KP_n to the adversary and sets $C \leftarrow P$. All subsequent queries return $\perp$. We restrict that **Corrupt()** returns $\perp$ if $\{t_1^*, \ldots, t_d^*\} \cup P = \emptyset$.
- **Challenge**. The adversary submits two messages M_0, M_1. The challenger flips a random bit b and computes the challenge ciphertext $\mathsf{CT}^* \leftarrow \mathsf{Encryption}(\mathsf{PK}, M_b, \mathbb{A}^*, t_1^*, \ldots, t_d^*)$.
- **Query Phase 2**. This phase is identical to Phase 1.
- **Guess**. The adversary output a guess b' of b. The adversary wins if $b = b'$.

The advantage of an adversary in this game is defined as $\Pr[b = b'] = \frac{1}{2}$. We note the above definition can be extended to handle chosen ciphertext attacks by allowing decryption queries in the Query Phase.

Definition 1: A Pt-CP-ABE scheme is secure in the model of indistinguishability against selective-target query attack if all polynomial time adversaries have at most a negligible advantage in the selective-target game.

Intuition of Security Notion. On the input of any t_t, the second step **Puncture** oracle in Query Phase 1 updates the punctured key to revoke t_t. The adversary may query this oracle repeatedly, each time producing a newly punctured key. The **Corrupt** oracle provides the adversary with the most recent state of the secret key held by the challenger.

The adversary may challenge on a pair of messages, targeted access structure $\mathbb{A}^*$, and chosen tags $\{t_1^*, \ldots, t_d^*\}$, subject to the following restriction: the attribute set ω does not satisfy the access structure $\mathbb{A}^*$ in any adversary's queries, or the adversary cannot corrupt the punctured key unless she/he has previously punctured at least one of the tags $\{t_1^*, \dots, t_d^*\}$. This restriction prevents the attacks in which the adversary may trivially decrypt the challenge ciphertext.

B. Construction

In this subsection, we elaborate the Setup, Encrypt, KeyGen, Puncture, and Decrypt algorithms defined above.

- ► $\mathsf{Setup}(1^{\mathsf{k}}, d, \mathsf{n}_{max})$. On input a security parameter k, a maximum number of tags per ciphertext d (which also specifies the number of attributes in the system), and the maximum number of columns in an LSSS matrix n_{max}, the algorithm first chooses a group $\mathbb{G}$ of prime order p, a bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$, and a generator g and a hash function $H : \{0,1\}^* \to \mathbb{Z}_p$. Then it randomly selects exponents $\alpha, \mathsf{a} \in \mathbb{Z}_p$ and random elements $h_{j,x} \in \mathbb{G}$ for $1 \leq j \leq \mathsf{n}_{max}, x \in 1, \dots, d$. Finally, it samples polynomial $q(x)$ of degree d. Then from $i = 1$ to d, it computes $q(i)$, subjects to the constraint that $q(0) = \mathsf{a}$.
Define $V(x) = g^{q(x)}$, and let t_0 be a distinguished tag not used during normal operation, then outputs:

$$\begin{aligned} \mathsf{PK} &= (g, e(g,g)^{\alpha}, g^{\mathsf{a}}, g^{q(1)}, \dots, g^{q(d)}, \\ &\quad \{h_{j,x}\}_{1 \leq j \leq \mathsf{n}_{max}, x \in d}, t_0), \\ \mathsf{MSK} &= g^{\alpha}. \end{aligned}$$

- ► $\mathsf{Encrypt}(\mathsf{PK}, M, \mathbb{A}, t_1, \dots, t_d)$. On input the public key PK, a message M, an LSSS access structure $\mathbb{A} = (\mathbb{M}, \pi)$, and a set of tags $t_1, \dots, t_d \in \{0,1\}^* \backslash \{t_0\}$, the algorithm first chooses randomly s in $\mathbb{Z}_p$.
Let $\mathbb{M}$ be a $\ell \times \mathsf{n}_{max}$ matrix, it chooses randomly $z_2, \dots, z_{\mathsf{n}_{max}} \in \mathbb{Z}_p^{\mathsf{n}_{max}}$ and lets $\vec{v} = (s, z_2, \dots, z_k) \in \mathbb{Z}_p^{\mathsf{n}_{max}}$. Denotes $\mathbb{M}_{i,j}$ as the i-th row, j-th column of $\mathbb{M}$, then it outputs:

$$\begin{aligned} \mathsf{CT} &= (C_0 = M \cdot e(g,g)^{\alpha s}, C_1 = g^s, \\ &\quad C_{2,i,j} = \{g^{\mathsf{a}\mathbb{M}_{i,j}\vec{v}_j} h_{j,\pi(i)}^{-s}\}_{1 \leq i \leq \ell, 1 \leq j \leq \mathsf{n}_{max}}, \\ &\quad C_{3,j} = \{V(H(t_j))^s\}_{j \in \{1,\dots,d\}}), \end{aligned}$$

along with the tags $t_1, \dots, t_d$.
- ► $\mathsf{KeyGen}(\mathsf{PK}, \mathsf{MSK}, \omega)$. On input the public key PK, a master key MSK, and an attribute set ω, the algorithm randomly chooses $r, r_{\mathsf{a}}, r_1, \dots, r_{\mathsf{n}_{max}}$ in $\mathbb{Z}_p$, and outputs:

$$\begin{aligned} \mathsf{SK} &= (D = g^{\alpha} g^{\mathsf{a}r_1 + \mathsf{a}r_{\mathsf{a}}}, \{D_{1,i} = g^{r_i}\}_{1 \leq i \leq \ell}, \forall x \in \omega, \\ &\quad D_{2,x} = \prod_{j=1,\dots,\mathsf{n}_{max}} h_{j,x}^{r_j}) \\ \mathsf{KP}_0 &= (\mathsf{KP}_{01} = (g^{\mathsf{a}})^{r + r_{\mathsf{a}}}, \mathsf{KP}_{02} = V(H(t_0))^r, \\ &\quad \mathsf{KP}_{03} = g^r, \mathsf{KP}_{04} = t_0). \end{aligned}$$

- ► $\mathsf{Puncture}(\mathsf{PK}, \mathsf{KP}_i, t)$. On input an existing key KP_{i-1} as $\{\mathsf{KP}_0, \mathsf{KP}_1, \dots, \mathsf{KP}_{i-1}\}$, the algorithm chooses λ', and r_0, r_1 randomly from $\mathbb{Z}_p$ and computes:

$$\begin{aligned} &\mathsf{KP}'_{01} = \mathsf{KP}_{01} \cdot g^{\mathsf{a}(r_0 - \lambda')} = (g^{\mathsf{a}})^{r + r_{\mathsf{a}} + r_0 - \lambda'} && , \\ &\mathsf{KP}_{i1} = (g^{\mathsf{a}})^{\frac{\lambda' + r_1}{i}} && , \\ &\mathsf{KP}'_{02} = \mathsf{KP}_{02} \cdot V(H(t_0))^{r_0} = V(H(t_0))^{r + r_0} && , \\ &\mathsf{KP}_{i2} = V(H(t))^{\frac{r_1}{i}} && , \\ &\mathsf{KP}'_{03} = \mathsf{KP}_{03} \cdot g^{r_0} = g^{r + r_0} && , \\ &\mathsf{KP}_{i3} = g^{\frac{r_1}{i}}, \mathsf{KP}'_{04} = t_0, \mathsf{KP}_{i4} = t. \end{aligned}$$

Then it outputs: $\mathsf{KP} = (\mathsf{KP}'_0, \mathsf{KP}_1, \dots, \mathsf{KP}_{i-1}, \mathsf{KP}_i)$.
- ► $\mathsf{Decrypt}(\omega, (\mathbb{M}, \pi), \mathsf{SK}, \mathsf{PK}, \mathsf{CT}, t_1, \dots, t_d, t)$.
Suppose that ω satisfies $(\mathbb{M}, \pi)$. Let $\mathcal{I} = \{i \mid \pi(i) \in \omega\}$. On input the secret key SK, the punctured key KP, a ciphertext CT, and a set of tags $\{t_1, \dots, t_d\}$ associated with the ciphertext, the decryptor then calculates the corresponding set of reconstruction constants $\{i, \nu_i\}_{1 \in \mathcal{I}}$ which has a linear reconstruction property : $\sum_{i \in \mathcal{I}} \nu_i \lambda_i = s$. Note that there could potentially be different ways of choosing the ν_i values to satisfy this. Then it computes the following:

$$\begin{aligned} A &= \frac{e(D, C_1)}{(\prod_{i=1}^{\mathsf{n}_{max}} e(D_{1,i}, \prod_{i \in \mathcal{I}} C_{2,i,j}^{\nu_i})) \cdot \prod_{i \in \mathcal{I}} (K_{\pi(i)}^{\nu_i}, C_1)} \\ &= \frac{e(D, C_1)}{(\prod_{i=1}^{\mathsf{n}_{max}} e(g^{r_i}, g^{\sum i \in \mathcal{I} \mathsf{a} \mathbb{M}_{i,j} v_j \nu_i})} \\ &\quad \cdot \frac{1}{\cdot e(g^{r_i}, \prod_{i \in \mathcal{I}} h_{j,\pi(i)}^{-s\nu_i})) \cdot \prod_{i \in \mathcal{I}} (K_{\pi(i)}^{\nu_i}, C_1)} \\ &= \frac{e(D, C_1)}{(\prod_{i=1}^{\mathsf{n}_{max}} e(g^{r_i}, g^{\sum i \in \mathcal{I} \mathsf{a} \mathbb{M}_{i,j} v_j \nu_i})} \\ &= \frac{e(D, C_1)}{e(g^{r_1}, g^{\sum i \in \mathcal{I} \mathsf{a} \mathbb{M}_{1,j} v_j \nu_i})} = \frac{e(g^{\alpha} g^{\mathsf{a}r_1 + \mathsf{a}r_{\mathsf{a}}}, g^s)}{e(g,g)^{r_1 \mathsf{a} s}} \\ &= e(g,g)^{\alpha s} e(g,g)^{\mathsf{a} r_{\mathsf{a}} s}. \end{aligned}$$

For $j = 0, \dots, i$, parse KP_i as $(\mathsf{KP}_{i1}, \mathsf{KP}_{i2}, \mathsf{KP}_{i3}, \mathsf{KP}_{i4})$. Next compute a set of coefficients $w_1, \dots, w_d, w^*$ such that

$$w^* \cdot q(H(\mathsf{KP}_{i4})) + \sum_{k=1}^{d} (w_k \cdot q(H(t_k))) = q(0) = \mathsf{a}. \quad (3)$$

Finally, it computes:

$$\begin{aligned} B &= \prod_{j=0}^{i} \frac{e(\mathsf{KP}_{j1}, C_1)}{e(\mathsf{KP}_{j3}, \prod_{k=1}^{d} C_{3,k}^{w_k}) \cdot e(\mathsf{KP}_{j2}, C_1)^{w^*}} \\ &= \frac{e(g^{\mathsf{a}(r + r_{\mathsf{a}} + r_0 - \lambda')}, g^s)}{e(g^{r + r_0}, \prod_{k=1}^{d} V(H(t_k))^{w_k}) \cdot e(V(H(t_0))^{r + r_0}, g^s)^{w^*}} \\ &\quad \cdots \frac{e(g^{\mathsf{a}(\lambda' + r_1)}, g^s) \cdot e(g^{\mathsf{a}(\lambda' + r_1)}, g^s)}{e(g^{\frac{r_1}{i}}, \prod_{k=1}^{d} V(H(t_k))^{w_k}) \cdot e(V(H(t))^{\frac{r_1}{i}}, g^s)^{w^*}} \end{aligned}$$

$$\begin{aligned}
&= \frac{e(g, g^{s\mathsf{a}(r+r_{\mathsf{a}}+r_0-\lambda')})}{e(g,g)^{s\mathsf{a}(r+r_0)}} \cdots \frac{e(g,g)^{\frac{\lambda' s(\mathsf{a})}{i}} \cdot e(g,g)^{\frac{\mathsf{a}r_1 s}{i}}}{e(g,g)^{\frac{r_1 s\mathsf{a}}{i}}} \\
&= e(g,g)^{\mathsf{a}(r_{\mathsf{a}}-\lambda')s} \cdot e(g,g)^{\frac{\mathsf{a}\lambda' s}{i}} \cdots e(g,g)^{\frac{\mathsf{a}\lambda' s}{i}} \\
&= e(g,g)^{\mathsf{a}(r_{\mathsf{a}}-\lambda')s} \cdot e(g,g)^{\mathsf{a}\lambda' s} = e(g,g)^{\mathsf{a}r_{\mathsf{a}}s},
\end{aligned}$$

and outputs message as:

$$M = \frac{C_0 \cdot B}{A} = \frac{M \cdot e(g,g)^{\alpha s} \cdot e(g,g)^{\mathsf{a}r_{\mathsf{a}}s}}{e(g,g)^{\alpha s} \cdot e(g,g)^{\mathsf{a}r_{\mathsf{a}}s}}. \quad (4)$$

C. Correctness Proof

The condition for the recovery coefficient is to find $w_0, \ldots, w_d, w^*$ such that

$$w^* \cdot q(H(\mathsf{KP}_{i4})) + \sum_{k=1}^{d}(w_k \cdot q(H(t_k))) = q(0) = \mathsf{a}.$$

It is effective asking for the coefficients necessary to compute $q(x)$ at given points on the polynomial $((t_1, q(t_1)), \ldots, (t_d, q(t_d)))$. Since we only need the recovery coefficients, we simply need the Lagrange basics. To get this, we do not need the y coordinates at all. As a result, we can compute by Eq. (2) as $w_i = \ell(t_i, i, [t_1, \ldots, t_d, \mathsf{PK}_{i4}])$ and $w^* = \ell(\mathsf{PK}_{i4}, d+1, [t_1, \ldots, t_d, \mathsf{PK}_{i4}])$.

D. Security Proof

Theorem 1: Assume that the DBDH assumption holds, then no polynomial-time adversary against our Pt-CP-ABE scheme can have a non-negligible advantage over random guess in the Selective IND-CPA security game.

Proof: Suppose that there exists an adversary $\mathcal{A}$ which can attack our scheme with non-negligible advantage ϵ, we construct another algorithm $\mathcal{B}$ called simulator which uses $\mathcal{A}$ to solve the DBDH problem. On input (g, g^a, g^b, g^c), the goal of $\mathcal{B}$ is to determine whether $Z = e(g,g)^{abc}$ or a random element of $\mathbb{G}_T$. $\mathcal{B}$ proceeds as follows.

Init: $\mathcal{A}$ gives B the target LSSS access structure $W^* = (\mathbb{M}^*, \pi^*)$, where $\mathbb{M}^*$ is a $\ell \times \mathsf{n}^*$ matrix, with $\mathsf{n}^* = \mathsf{n}_{max}$. $\mathcal{A}$ also announces to $\mathcal{B}$ the target tag set $(t_1^*, \ldots, t_d^*)$ that it intends to attack at the beginning of the game.

Setup: The simulator $\mathcal{B}$ chooses random $\alpha' \in \mathbb{Z}_p$ and implicitly sets $\alpha = ab + \alpha'$ by letting $e(g,g)^\alpha = e(g^a, g^b) \cdot e(g,g)^{\alpha'}$. $\mathcal{B}$ also randomly chooses $\mathsf{a}' \in \mathbb{Z}_p$, and implicitly sets $\mathsf{a} = a$. It continuously initializes two empty sets P, C and a counter $\tau = 0$. Then it chooses random value $z_{x,j} \in \mathbb{Z}_p$ where $1 \le x \le d$ and $1 \le j \le \mathsf{n}_{max}$. If there exists i such that $\pi^*(i) = x, i \le \mathsf{n}^*$, then let $h_{j,x} = g^{z_{j,x}} g^{a\mathbb{M}^*_{i,j}}$. Otherwise, let $h_x = g^{z_{x,j}}$.
Next, $\mathcal{B}$ chooses $d+1$ points $\theta_{t_0}, \theta_{t_1}, \ldots, \theta_{t_d}$ uniformly at random from $\mathbb{Z}_p$, in which θ_{t_0} is a distinguished value not used in normal simulation. Then it implicitly sets $q(0) = a$, while $q(t_i) = \theta_{t_i}$, then $V(H(t_i)) = g^{q(t_i)} = g^{\theta_{t_i}}$.
$\mathcal{B}$ sets public key as:

$$\mathsf{PK} = (g, e(g,g)^\alpha, g^{\mathsf{a}}, g^{q(1)}, \ldots, g^{q(d)}, \{h_{j,x}\}_{1 \le j \le \mathsf{n}_{max}, x \in d}, t_0),$$

Then $\mathcal{B}$ sends it to adversary and keeps $\mathsf{MSK} = g^{\mathsf{a}}$ for itself.

Phase 1: $\mathcal{A}$ adaptively makes requests for several attribute sets and tags. Suppose that $\mathcal{A}$ requests the secret key for attribute set ω and tag t to be added to P. We consider the following two cases in Phase 1 as in the proof of Pt-CP-ABE.

- **Case 1:** $\omega \models W^*$. Due to the condition in this case and by **Lemma 1**, there must exist a vector $\vec{w} = (w_1, \ldots, w_k) \in \mathbb{Z}_p^k$ where k is maximum column of $\mathbb{M}$ such that $w_1 = -1$, and for all i where $\pi^*(i) \in \omega$, it holds that $\mathbb{M}^*_i \cdot \vec{w} = 0$.
  $\mathcal{B}$ randomly chooses $r, r_{\mathsf{a}}, r'_2, \ldots, r'_{\mathsf{n}_{max}} \in \mathbb{Z}_p$. Note we can set $w_j = 0$ and consider $\mathbb{M}^*_{i,j} = 0$ for $\mathsf{n}^* \le j \le \mathsf{n}_{max}$.
  $\mathcal{B}$ calculates all $D_{1,j}$ as $D_{1,j} = g^{r'_j}(g^b)^{w_j}$, which implicitly let $r_j = r'_j + b \cdot w_j$.
  Then $\mathcal{B}$ simulates D as

  $$D = g^{ab} g^{\alpha'} g^{a(r'_1 + w_1 b)} g^{ar_{\mathsf{a}}} = g^\alpha g^{\mathsf{a}(r_1 + r_{\mathsf{a}})},$$

  where $r_1 = r'_1 - b$ with $w_1 = -1$. Then $\mathcal{B}$ can simulate the unknown term g^α.
  For $\forall x \in \omega$, then there is no i such that $\pi^*(i) = x$. $\mathcal{B}$ can simulate simply: $D_{2,x} = \prod_{j=1,\ldots,\mathsf{n}_{max}} D^{r_j}_{j,x}$, due to $M_i \cdot w = 0$ and $\phi(\pi(i)) = 0$. Then KP can simply set:

  $$\begin{aligned}\mathsf{KP}_0 \;=\; &(\mathsf{KP}_{01} = (g^{\mathsf{a}})^{r+r_{\mathsf{a}}}, \mathsf{KP}_{02} = g^{\theta_{t_0} r}, \\ &\mathsf{KP}_{03} = g^r, \mathsf{KP}_{04} = t_0).\end{aligned}$$

  Next, the simulator $\mathcal{B}$ increments n, computes $\mathsf{KP}_n = \mathsf{Puncture}(\mathsf{KP}_{n-1}, t)$, and adds t to set P. In the case $\omega^* \models W$, we consider :
  - **Corrupt**() query and $\{t_1^*, \ldots, t_d^*\} \cap C = \emptyset$. $\mathcal{B}$ now randomly chooses $r'_0, r'_1, \lambda \in \mathbb{Z}_p$, and implicitly sets $r_0 = \lambda + r'_0, r_1 = -\lambda + r'_1$. Thus it outputs the following:

    $$\begin{aligned}
    &\mathsf{KP}'_{01} = \mathsf{KP}_{01} \cdot g^{\mathsf{a}(r_0 - \lambda')} = (g^{\mathsf{a}})^{r + r_{\mathsf{a}} + r'_0} &,\\
    &\mathsf{KP}_{i1} = (g^{\mathsf{a}})^{\frac{r'_1}{i}} &,\\
    &\mathsf{KP}'_{02} = \mathsf{KP}_{02} \cdot V(H(t_0))^{r_0} = g^{\theta_{t_0} r(\lambda + r'_0)} &,\\
    &\mathsf{KP}_{i2} = V(H(t))^{\frac{-\lambda + r'_1}{i}} &,\\
    &\mathsf{KP}'_{03} = \mathsf{KP}_{03} \cdot g^{r_0} = g^{r + \lambda + r'_0} &,\\
    &\mathsf{KP}_{i3} = g^{\frac{-\lambda + r'_1}{i}} &,\\
    &\mathsf{KP}'_{04} = t_0 &,\\
    &\mathsf{KP}_{i4} = t.
    \end{aligned}$$

    Corrupt() is called in the first time; the adversary issues this query. Then the challenger returns the most recent punctured key KP_n to the adversary and sets $C \leftarrow P$. All subsequent queries return $\perp$.
- **Case 2:** $\omega \not\models W^*$. We can similarly simulate $D, D_{1,j}$ as in case 1. Since $\omega \not\models W^s$, we simulate $D_{2,x}$ as follows. If $x \in \omega$, we calculate $D_{2,x}$ in the same way as in Case 1. If $x \notin \omega$, then $h_{j,x} = g^{z_{j,x}} g^{a\mathbb{M}^*_{i,j}}$ with additional constraint of $\pi^*(i) \in \omega$ for all i. It holds that $\mathbb{M}^*_i \cdot \vec{w} = 0$.

We show how $\mathcal{B}$ can simulate $D_{2,x}$ without knowing g^{ab} as

$$\begin{aligned} D_{2,x} &= \prod_{j=1,\ldots,\mathsf{n}_{max}} D_{j,x}^{r_j} \\ &= \prod_{j=1,\ldots,\mathsf{n}_{max}} g^{z_{x,j}r_j} \cdot g^{bz_{x,j}} \cdot g^{a\mathbb{M}^*_{i,j}r_j}. \end{aligned}$$

Then KP can simply set:

$$\begin{aligned} \mathsf{KP}_0 &= (\mathsf{KP}_{01} = (g^{\mathsf{a}})^{r+r_{\mathsf{a}}}, \mathsf{KP}_{02} = g^{\theta_{t_0}r}, \\ &\quad \mathsf{KP}_{03} = g^r, \mathsf{KP}_{04} = t_0). \end{aligned}$$

Next, the simulator $\mathcal{B}$ increments n, computes $\mathsf{KP}_n = \mathsf{Puncture}(\mathsf{KP}_{n-1}, t)$, and adds t to set P. In the case $\omega^* \not\models W$, we consider :

- **Corrupt**() is called in the first time as case 1; the adversary issues this query. Then the challenger returns the most recent punctured key KP_n to the adversary and sets $C \leftarrow P$. All subsequent queries return $\perp$.
- **Corrupt()** does not query or $\{t_1^*, \ldots, t_d^*\} \cap C \neq \emptyset$. $\mathcal{B}$ now chooses randomly $r_0, r_1, \lambda \in \mathbb{Z}_p$. Thus it outputs the following:

$$\begin{aligned} &\mathsf{KP}'_{01} = \mathsf{KP}_{01} \cdot g^{\mathsf{a}(r_0-\lambda')} = (g^{\mathsf{a}})^{r+r_{\mathsf{a}}+r_0-\lambda'} &, \\ &\mathsf{KP}_{i1} = (g^{\mathsf{a}})^{\frac{\lambda'+r_1}{i}} &, \\ &\mathsf{KP}'_{02} = \mathsf{KP}_{02} \cdot V(H(t_0))^{r_0} = V(H(t_0))^{r+r_0} & \\ &= (g^{\theta_{t_0}})^{r+r_0} &, \\ &\mathsf{KP}_{i2} = V(H(t))^{\frac{r_1}{i}} = (g^{\theta_t})^{\frac{r_1}{i}} &, \\ &\mathsf{KP}'_{03} = \mathsf{KP}_{03} \cdot g^{r_0} = g^{r+r_0} &, \\ &\mathsf{KP}_{i3} = g^{\frac{r_1}{i}} &, \\ &\mathsf{KP}'_{04} = t_0 &, \\ &\mathsf{KP}_{i4} = t. \end{aligned}$$

Challenge: The adversary gives two messages M_0 and M_1 to $\mathcal{B}$. Then $\mathcal{B}$ flips a coin b and generates the challenge ciphertext by randomly choosing $y'_2, \ldots, y'_k \in \mathbb{Z}_p$ and lets $\vec{v} = (s, s + y'_2, \ldots, s + y'_{\mathsf{n}^*}) \in \mathbb{Z}_p^{\mathsf{n}^*}$. Then it generates $C_1 = g^c$, and for $i = 1, \ldots, n$, $C_{2,i,j}$ is generated as:

$$C_{2,j} = g^{\mathsf{a}\mathbb{M}^*_{i,j}\vec{v}_j} h^{-s}_{j,\pi(i)} = (g^a)^{\mathbb{M}^*_{i,j}y'_j}(g^c)^{-z_{\pi^*(i),j}},$$

and $C_{3,j} = (g^c)^{\theta_{t_j}}$.

$\mathcal{B}$ then sends the following challenge ciphertext to $\mathcal{A}$

$$CT^* = (M_b Z, C_1, \{C_{2,i,j}\}_{1\le i\le \ell, 1\le j\le \mathsf{n}_{max}}, \{C_{3,j}\}_{j\in\{1,\ldots,d\}}).$$

Phase 2 is identical to Phase 1.
Guess: $\mathcal{A}$ outputs $b' \in \{0, 1\}$. If $b' = b$ then $\mathcal{B}$ outputs 1, otherwise outputs 0.
Analysis: If $Z = e(g, g)^{abc}$, then the simulation is the same as in the real game. Hence, $\mathcal{A}$ will have the probability $\frac{1}{2} + \epsilon$ to guess b correctly. If Z is a random element of $\mathbb{G}_T$, then $\mathcal{A}$ will have probability $\frac{1}{2}$ to guess b correctly. Therefore, $\mathcal{B}$ can solve the DBDH assumption also with advantage ϵ. This contradicts the proven hardness of DBDH. Therefore, the assumption of existence of an adversary $\mathcal{A}$ which can attack our scheme with non-negligible advantage ϵ is invalid. Thus the theorem is proven.

IV. Puncturable Key Policy Attribute Based Encryption for IoT devices

In the Puncturable Ciphertext Policy Attribute Based Encryption (Pt-CP-ABE), the sender includes a linear access structure $\mathbf{A}$ and a tag in each message sent to given IoT devices. On the other hand, an IoT device's secret key is an ABE decryption key embedding its attribute ω. It allows the IoT devices, which have attributes ω satisfying the access structure $\mathbb{A}$, to decrypt the messages.

In this section, we introduce a dual form of Pt-CP-ABE by swapping the roles of ω and $\mathbb{A}$ such that the attribute set ω is embedded into the ciphertext and the access structure $\mathbb{A}$ into the secret key. It is named Puncturable Key Policy Attribute Based Encryption (Pt-KP-ABE).

Consider a similar application setting as illustrated in Fig. 1. Suppose that each engine control board is equipped with a card that contains the access structure $\mathbb{A}$. It is installed by the manufacturer and cannot be modified by the user or a third part. The access structure specifies the types of messages that the control board is allowed to receive. The sender embeds attributes ω into messages. A receiver is able to decrypt a message only if its access structure is satisfied by ω. In addition, the sender also includes a "tag" such as a message identifier or time period identifier in each message. Similar to Pt-CP-ABE, a receiver can puncture a key at a point t, i.e., updates its existing secret key to derive a new punctured key that embeds the tag t, in order to selectively revoke the ability to decrypt messages with specific tags. As a result, it protects the messages embedded with t even if the current communication is compromised. Again, the creation of punctured keys does not require communication with the key distribution center, which is highly desired in large-scale IoTs.

As dual forms, Pt-KP-ABE and Pt-CP-ABE are similar in several aspects. Here we outline the Pt-KP-ABE scheme by introducing its main algorithmic steps and highlighting its difference from Pt-CP-ABE.

- ► $\mathsf{Setup}(1^{\mathsf{k}}, d)$. On input a security parameter k, a maximum number of tags per ciphertext d, a parameter d also specifies how many attributes in the system, the algorithm firstly chooses a group $\mathbb{G}$ of prime order p, a bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$, and a generator g and a hash function $H : \{0, 1\}^* \to \mathbb{Z}_p$. Then it selects randomly exponents $\alpha, \mathsf{a} \in \mathbb{Z}_p$ and random elements $h_j \in \mathbb{G}$ for $j \in \{1, \ldots, d\}$.
  Finally, it samples polynomial $q(x)$ of degree d. Then from $i = 1$ to d, it computes $q(i)$, and subjects to the constraint that $q(0) = \mathsf{a}$.
  Define $V(x) = g^{q(x)}$, and let t_0 be a distinguish tag not used during normal operation, then outputs:

$$\begin{aligned} \mathsf{PK} &= (g, e(g, g)^\alpha, g^{q(1)}, \ldots, g^{q(d)}, \{h_j\}_{j\in d}, t_0), \\ \mathsf{MSK} &= (\alpha, \mathsf{a}). \end{aligned}$$

- ► $\mathsf{Encryption}(\mathsf{PK}, M, \omega, t_1, \ldots, t_d)$. On input the public key PK, a message M, a attribute set $\omega \subset \mathbf{Z}_p^*$, and a set of tags $t_1, \ldots, t_d \in \{0,1\}^* \backslash \{t_0\}$, the algorithm chooses randomly s in $\mathbb{Z}_p$ and outputs:

$$\begin{aligned} \mathsf{CT} &= (\omega, C_0 = M \cdot e(g,g)^{\alpha s}, C_1 = g^s, \\ &\quad C_{3,j} = \{h_j^s\}_{j \in \omega}, C_{3,j} = \{V(H(t_j))^s\}_{j \in \{1,\ldots,d\}}), \end{aligned}$$

alongs with the tags $t_1, \ldots, t_d$.

- ▶ $\mathsf{KeyGen}(\mathsf{PK}, \mathsf{MSK}, \mathbb{A} = (\mathbb{M}, \pi))$. On input the public key PK, a master key MSK, and a LSSS access structure $\mathbb{A} = (\mathbb{M}, \pi)$, the algorithm chooses randomly $r_{\mathsf{a}}, z_2, \ldots, z_k \in \mathbb{Z}_p$. Then, the algorithm first shares $\alpha + \mathsf{a} r_{\mathsf{a}}$ with LSSS $\mathbb{A} = (\mathbb{M}, \pi)$, and let $\vec{v} = (\alpha + \mathsf{a} r_{\mathsf{a}}, z_2, \ldots, z_k)$. For $i = 1$ to ℓ, it calculates the share $\lambda_i = \mathbb{M} \cdot \vec{v}$ where $\mathbb{M}_i$ is the vector corresponding to the i-th row of $\mathbb{M}$. It then the randomly chooses $r, r_1, \ldots, r_\ell$ in $\mathbb{Z}_p$, and outputs:

$$\begin{aligned} \mathsf{SK} &= (\{D_{1,i} = g^{\lambda_i} h_{\pi(i)}^{r_i}, D_{2,i} = g^{r_i}\}_{i \in \pi(i)}) \\ \mathsf{KP}_0 &= (\mathsf{KP}_{01} = (g^{\mathsf{a}})^{r + r_{\mathsf{a}}}, \mathsf{KP}_{02} = V(H(t_0))^r, \\ &\quad \mathsf{KP}_{03} = g^r, \mathsf{KP}_{04} = t_0). \end{aligned}$$

- ▶ $\mathsf{Puncture}(\mathsf{PK}, \mathsf{KP}_i, t)$. On input an existing key KP_{i-1} as $\{\mathsf{KP}_0, \mathsf{KP}_1, \ldots, \mathsf{KP}_{i-1}\}$, the algorithm chooses λ', and r_0, r_1 randomly from $\mathbb{Z}_p$ and computes similarly to the Pt-CP-ABE's Puncture algorithm, and outputs: $\mathsf{KP} = (\mathsf{KP}'_0, \mathsf{KP}_1, \ldots, \mathsf{KP}_{i-1}, \mathsf{KP}_i)$.

- ▶ $\mathsf{Decrypt}(\omega, (\mathbb{M}, \pi), \mathsf{SK}, \mathsf{PK}, \mathsf{CT},)$. Suppose that ω satisfies $(\mathbb{M}, \pi)$. Let $\mathcal{I} = \{i \mid \pi(i) \in \omega\}$. On input the secret key SK, the punctured key KP, a ciphertext CT, and a set of tags $\{t_1, \ldots, t_d\}$ are associated with the ciphertext. It then calculates the corresponding set of reconstruction constants $\{i, \nu_i\}_{1 \in \mathcal{I}}$ which has a linear reconstruction property : $\sum_{i \in \mathcal{I}} \nu_i \lambda_i = \alpha + \mathsf{a} r_{\mathsf{a}}$. Then it computes the following:

$$\begin{aligned} A &= \prod_{i=1}^{\ell} \left(\frac{e(D_{1,i}, C_1)}{e(D_{2,i}, C_{2,i})}\right)^{\nu_i} = \prod_{i=1}^{\ell} \left(\frac{e(g^{\lambda_i} h_{\pi(i)}^{r_i}, g^s)}{e(g^{r_i}, (h_{\pi(i)})^s)}\right)^{\nu_i}, \\ B &= \prod_{j=0}^{i} \frac{e(\mathsf{KP}_{j1}, C_1)}{e(\mathsf{KP}_{j3}, \prod_{k=1}^{d} C_{3,k}^{w_k}) \cdot e(\mathsf{KP}_{j2}, C_1)^{w^*}}, \end{aligned}$$

and output message as: $M = \frac{C_0 \cdot B}{A}$.

The proofs for correctness and security of Pt-KP-ABE are similar to the proofs given in Sec. III and thus omitted here.

V. Evaluation

Since we have proven the correctness and security of the proposed schemes, our performance evaluation focuses on computation efficiency. We analytically compare the encryption, decryption cost. As shown in Table II, the cost of the proposed schemes is only marginally higher than the original ABE schemes, where p denotes the pairing operation, d the maximum number of tags, $|\omega|$ the number of attributes in access structure, ℓ the number of attributes in user key, n_{max} the maximum number of columns in an LSSS matrix, and pr the number of punctures in the secret key.

Then, we implement the proposed schemes on Raspberry Pi 3 with an ARM Cortex processor. We adapt two existing libraries [25], [26] to construct Pt-CP-ABE and Pt-KP-ABE. The experiments aim to study their average computing speed on IoT devices. The sizes of public key and master key grow linearly with size of attributes universe. We experience on a 160-bit elliptic curve group constructed on the curve $y^2 = x^3 + x$ over a 512-bit field. The program are tested for a number of rounds until the average key generation, encryption and decryption speed converges.

TABLE I
Average Speed for Punturable key generation.

	Time(ms)
Initial Punturable Key Generation	7
Subsequent Punturable Key Generation	5

Table 1 demonstrates the computation time to generate punctured keys. It takes about 7 and 5 ms respectively to generate the initial punctured key (i.e., KP_0) and each subsequent key. Figs. 2 and 3 illustrates the encryption and decryption times of the original ABE schemes and the proposed Pt-ABE schemes. As can be seen, although the proposed schemes incorporate punctured keys to achieve the revocation capability of target messages, their encryption and decryption times only increase marginally (about 10%) in comparison with the original ABE schemes. The computation time naturally increases with the number of attributes. But even with a large attribute set (e.g., 20 attributes), both encryption and decryption operations can be completed within tens of milliseconds. The experimental results show that the proposed puncturable ABE schemes are well suited for IoT applications.

TABLE II
Analytical Comparison of Encryption and Decryption Cost

Scheme	Puncture Key	Ciphertext Size	Decryption Cost
KP-ABE	X	$\|\mathbb{G}_T\| + (1 + \|\omega\|)\|\mathbb{G}\|$	$2\ell_{\mathcal{I}}\mathsf{p}$
Pt-KP-ABE	√	$\|\mathbb{G}_T\| + (1 + \|\omega\| + d)\|\mathbb{G}\|$	$(2\ell_{\mathcal{I}} + 3 \times pr)\mathsf{p}$
CP-ABE	X	$\|\mathbb{G}_T\| + (1 + (\ell \times \mathsf{n}_{max}))\|\mathbb{G}\|$	$(1 + \mathsf{n}_{max})\mathsf{p}$
Pt-CP-ABE	√	$\|\mathbb{G}_T\| + (1 + (\ell \times \mathsf{n}_{max}) + d)\|\mathbb{G}\|$	$((1 + \mathsf{n}_{max} + 3 \times pr)\mathsf{p}$

VI. Conclusion

This work is the first endeavor to develop practical Puncturable Attribute Based Encryption schemes that are lightweight and applicable in IoTs. We have proposed two new schemes, called Puncturable-Key Policy-Attribute Based Encryption and Puncturable-Ciphertext Policy-Attribute Based Encryption. Both of them support fine grained access control and key punctuation to protect selected important messages even if the current key is compromised. In contrast to conventional forward encryption, the proposed approach enables recipients to update their keys by themselves without communicating with the key distributor. Moreover, the novel approach efficiently integrates attribute-based key and punctured keys such that the key size is roughly the same as that of the original attribute-based encryption. We have proven the correctness of the proposed scheme and its security under the Decisional

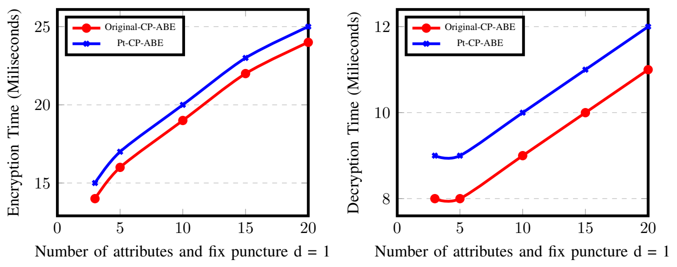

Fig. 2. Comparison of encryption and decryption times between the original CP-ABE and Pt-CP-ABE.

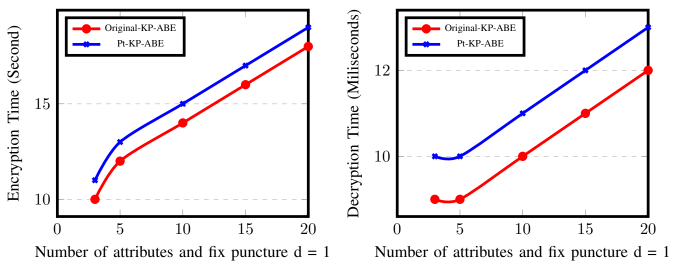

Fig. 3. Comparison of encryption and decryption times between the original KP-ABE and Pt-KP-ABE.

Bilinear Diffie-Hellman (DBDH) assumption. We have also implemented the proposed schemes on Raspberry Pi. The experimental results show that both encryption and decryption can be completed within tens of milliseconds.

Acknowledgments

This work was supported in part by the National Science Foundation under Grants CNS-1528004, CNS-1418012 and CNS-1745632.

References

[1] CISCO, "Cisco IoT System Security: Mitigate Risk, Simplify Compliance, and Build Trust," *White Paper*, 2015.

[2] A. Sahai and B. Waters, "Fuzzy identity-based encryption," in *Proceedings of EUROCRYPT*, pp. 457–473, 2005.

[3] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in *Proceedings of ACM CCS*, pp. 89–98, 2006.

[4] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *Proceedings of IEEE Symposium Security and Privacy*, pp. 321–334, 2007.

[5] L. Cheung and C. Newport, "Provably Secure Ciphertext Policy ABE," in *Proceedings of ACM CCS*, pp. 456–465, 2007.

[6] V. Goyal, A. Jain, O. Pandey, and A. Sahai, "Bounded ciphertext policy attribute based encryption," in *Proceedings of ICALP*, pp. 579–591, 2008.

[7] B. Waters, "Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization," in *Proceedings of PKC*, pp. 53–70, 2011.

[8] N. Attrapadung, B. Libert, and E. Panafieu, "Expressive key-policy attribute-based encryption with constant-size ciphertexts," in *Proceedings of PKC*, pp. 90–108, 2011.

[9] A. B. Lewko and B. Waters, "New proof methods for attribute-based encryption: Achieving full security through selective techniques," in *Proceedings of CRYPTO*, pp. 180–198, 2012.

[10] T. V. X. Phuong, G. Yang, and W. Susilo, "Hidden ciphertext policy attribute-based encryption under standard assumptions," *IEEE Trans. Information Forensics and Security*, vol. 11, no. 1, pp. 35–45, 2016.

[11] T. Nishide, K. Yoneyama, and K. Ohta, "Attribute-based encryption with partially hidden encryptor-specified access structures," in *Proceedings of the International Conference on Applied Cryptography and Network Security*, pp. 111–129, 2008.

[12] K. Emura, A. Miyaji, A. Nomura, K. Omote, and M. Soshi, "A ciphertext-policy attribute-based encryption scheme with constant ciphertext length," in *Proceeding of International Conference on Information Security Practice and Experience*, pp. 13–23, 2009.

[13] Z. Zhou and D. Huang, "On efficient ciphertext-policy attribute based encryption and broadcast encryption: extended abstract," in *Proceedings of ACM CCS*, pp. 753–755, 2010.

[14] C. Chen, Z. Zhang, and D. Feng, "Efficient ciphertext policy attribute-based encryption with constant-size ciphertext and constant computation-cost," in *Proceedings of International Conference on Provable Security*, pp. 84–101, 2011.

[15] N. Attrapadung and H. Imai, "Dual-policy attribute based encryption," in *Proceedings of the International Conference on Applied Cryptography and Network Security*, pp. 168–185, 2009.

[16] J. Lai, R. H. Deng, Y. Yang, and J. Weng, "Adaptable ciphertext-policy attribute-based encryption," in *Proceedings of International Conference on Pairing-Based Cryptography*, pp. 199–214, 2014.

[17] T. Kitagawa, H. Kojima, N. Attrapadung, and H. Imai, "Efficient and fully secure forward secure ciphertext-policy attribute-based encryption," in *Proceedings of ISC*, pp. 87–99, 2015.

[18] R. Canetti, S. Halevi, and J. Katz, "A forward-secure public-key encryption scheme," in *Proceedings of EUROCRYPT*, pp. 255–271, 2003.

[19] Z. Wang, D. D. Yao, and R. Feng, "Adaptive key protection in complex cryptosystems with attributes," *Cryptology ePrint Archive, Report 2012/136*, 2012.

[20] C. G. Günther, "An identity-based key-exchange protocol," in *Proceedings of EUROCRYPT*, pp. 29–37, 1990.

[21] W. Diffie, P. C. Van Oorschot, and M. J. Wiener, "Authentication and authenticated key exchanges," *Des. Codes Cryptography*, vol. 2, pp. 107–125, 1992.

[22] M. D. Green and I. Miers, "Forward secure asynchronous messaging from puncturable encryption," in *Proceedings of IEEE Symposium on Security and Privacy*, pp. 305–320, 2015.

[23] E. Kiltz, "Chosen-ciphertext security from tag-based encryption," in *Proceedings of TCC*, pp. 581–600, 2006.

[24] A. De Caro and V. Iovino, "jPBC: Java pairing based cryptography," in *Proceedings of IEEE Symposium on Computers and Communications*, pp. 850–855, 2011.

[25] J. Wang, "JCPABE java implementation," in *https://github.com/TU-Berlin-SNET/JCPABE.*

[26] A. H. Sánchez and F. Rodríguez-Henríquez, "Neon implementation of an attribute-based encryption scheme," in *Proceedings of the International Conference on Applied Cryptography and Network Security*, pp. 322–338, 2013.