Online/Offline Public-Index Predicate Encryption for Fine-Grained Mobile Access Control

Weiran Liu[1,2], Jianwei Liu[1], Qianhong Wu[1,3,4], Bo Qin[5](✉), and Kaitai Liang[6]

[1] School of Electronic and Information Engineering, Beihang University, No. 37, XueYuan Road, Haidian District, Beijing 100191, China
liuweiran900217@gmail.com, {liujianwei,qianhong.wu}@buaa.edu.cn

[2] State Key Laboratory of Integrated Services Networks, Xidian University, Xi'an 710071, China

[3] State Key Laboratory of Cryptology, P.O. Box 5159, Beijing 100878, China

[4] State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China

[5] Key Laboratory of Data Engineering and Knowledge Engineering, Ministry of Education, School of Information, Renmin University of China, No. 59, ZhongGuanCun Avenue, Haidian District, Beijing 100872, China
bo.qin@ruc.edu.cn

[6] Department of Computer Science, Aalto University, Konemiehentie 2, 01250 Espoo, Finland
kaitai.liang@aalto.fi

Abstract. Public-Index Predicate Encryption (PIPE) allows users to encrypt according to boolean predicates defined on arbitrary attributes. The expensive algebraic operations are the major efficiency obstacle for PIPE to be applied to mobile clouds. This paper proposes a general Online/Offline PIPE (OO-PIPE) framework to address this issue. First, we propose a generic transformation from a Large Universe PIPE (LU-PIPE) secure against chosen plaintext attack (CPA) to OO-PIPE in the same security model. The challenge is to generate ciphertext without the knowledge of the associated ciphertext attributes in the offline phase. We address the challenge by identifying an interesting *attribute-malleability* property in many LU-PIPE schemes. The property allows an encryptor to efficiently malleate a ciphertext associated with one ciphertext attribute to any assigned ciphertext attribute. Second, we design a generic transformation from CPA-secure LU-PIPE to OO-PIPE secure against adaptively chosen ciphertext attack (CCA2), assuming the underlying LU-PIPE has *attribute-malleability* and *public-verifiability* properties. The main obstacle here is that the online/offline mechanism endogenously implies forgery in the sense that a pre-computed ciphertext must be able to be efficiently malleated to the resulting ciphertext associated with a different ciphertext attribute and a plaintext, while any efficient valid ciphertext forgery is forbidden in CCA2 security. We circumvent

I. Askoxylakis et al. (Eds.): ESORICS 2016, Part II, LNCS 9879, pp. 588–605, 2016.
DOI: 10.1007/978-3-319-45741-3_30

this obstacle by employing a universally collision resistant Chameleon hash, namely, only the original encryptor can malleate the ciphertext to associate with different attributes and provide a hash collision of the ciphertext components.

1 Introduction

Mobile cloud computing becomes more and more attracting in cloud-assisted networks. However, security risks of mobile computing may hinder its wide applications. Since data are outsourced to the cloud, it is required that data must be stored securely, while allowing legal access for authenticated users. Traditional encryption techniques supporting limited access control are not seamlessly applicable to mobile cloud computing. Public-Index Predicate Encryption (PIPE) is an emerging asymmetric encryption allowing fine-grained access control over encrypted data. In a PIPE system, the access control policy is described by a predicate. A ciphertext is associated with a ciphertext attribute, and a secret key is assigned to a key attribute. One can decrypt if and only if the ciphertext attribute specified in the ciphertext matches the key attribute in his/her secret key according to the pre-defined predicate.

PIPE is a general cryptographic concept capturing a wide range of cryptographic primitives, including Identity-Based Encryption (IBE) [5,34], Revocation Encryption (RE) [23], Attribute-Based Encryption (ABE) [33] in Key-Policy settings [15] and in Ciphertext-Policy settings [2]. PIPE is classified into two categories: Small Universe PIPE (SU-PIPE) and Large Universe PIPE (LU-PIPE). In SU-PIPE, the size of attribute is polynomially bounded in security parameter [9,12,15], which poses constraints in practice. LU-PIPE does not suffer from this constraint and its attribute space can be exponentially large [25,31,32]. This desirable feature makes many instances of LU-PIPE, e.g., (H)IBE [19,20], ABE [39], become attractive to secure mobile cloud computing.

There are still hindrances for LU-PIPE to be widely deployed in mobile cloud computing. Most LU-PIPE schemes require time-consuming algebraic operations and encryption time grows with the number of ciphertext attributes. This may limit their efficiency. When the encryption is run on a mobile device, it may raise poor user experience with long latency and meanwhile, exhaust the battery quickly. Moreover, since LU-PIPE is usually suggested to secure data stored on untrusted but powerful servers, a strong security level, i.e., CCA2 security, is necessary for holding against powerful active attackers. Note that CCA2-secure LU-PIPE is less efficient than its CPA-secure counterpart. This further deteriorates the resource consumption and user experience.

Online/offline encryption may mitigate the efficiency problem. The encryption is split into offline and online phases. In the offline phase, an encryptor conducts the majority of the computation task on a high-end computer or when battery recharge before knowing the ciphertext attributes. In the online phase, the encryptor needs only few computations to fulfill the encryption when knowing the corresponding ciphertext attributes. In this way, it is feasible to implement LU-PIPE on resource-limited mobile devices with desirable user experience.

Several online/offline PIPE (OO-PIPE) schemes have been designed in an *ad hoc* way [10,17,18,27,29]. It is desirable to investigate generic OO-PIPE transformation from LU-PIPE. Theoretically, such a work allows a better understanding on LU-PIPE and online/offline mechanism. Practically, it enables one to instantly obtain OO-PIPE with better security and/or efficiency whenever an advantageous LU-PIPE scheme is available.

1.1 Our Contributions

We aim at proposing a framework for constructing OO-PIPE with CCA2 security. Our contribution includes the following aspects.

We start by identifying a useful property, i.e., *attribute-malleability* consisting of *private malleability* and *public malleability*, of many LU-PIPE schemes. The *private malleability* allows an encryptor to malleate a ciphertext associated with one ciphertext attribute to a ciphertext associated with any given ciphertext attribute at a very low cost. In contrast, the *public malleability* states that even through others may malleate a ciphertext associated with one ciphertext attribute to a ciphertext associated with some other ciphertext attribute, they cannot know (the key of) any matching key attribute.

We propose a generic CPA-secure OO-PIPE construction from attribute-malleable CPA-secure LU-PIPE. With *private malleability*, an encryptor prepares a ciphertext under a randomly chosen ciphertext attribute in the offline phase, and then replaces it with the target ciphertext attribute in the online phase. With *public malleability*, we show that the security of the resulting OO-PIPE can be tightly reduced to the CPA security of the underlying LU-PIPE.

We next propose a generic CCA2-secure OO-PIPE construction from any CPA-secure attribute-malleable LU-PIPE with *public-verifiability*. The *public-verifiability* states that there exists a public verification mechanism to verify whether the ciphertext has been honestly generated. This property enables one to establish a built-in LU-PIPE ciphertext validation check mechanism. We further exploit universally collision resistant Chameleon hash for ciphertext validation so that an encryptor can replace the randomly encrypted ciphertext attribute in the offline phase with the target ciphertext attribute in the online phase, while an attacker cannot make such malleation.

Technically, our constructions offer a novel application of Chameleon hash in encryption systems. Chameleon hash was previously used in (online/offline) signature applications [35]. It has been recently used as a security proof tool in constructing CCA2-secure KP-ABE [28]. We strengthen regular Chameleon hash with universal collision resistance and propose a generic universally collision resistant Chameleon hash from a regular Chameleon hash and a standard cryptographic hash. Our work illustrates the unique value of Chameleon hash in online/offline encryption cryptosystems, in contrast to its previous use in online/offline signatures.

1.2 Related Work

LU-PIPE. The simplest LU-PIPE is IBE that was theoretically introduced by Shamir [34] and practically constructed by Boneh and Franklin [5]. In 2005, Sahai and Waters [33] proposed Fuzzy IBE with a more expressive predicate. The concept of ABE, a versatile type of LU-PIPE, was also introduced in their work. Subsequently, two types of ABE, i.e., KP-ABE and CP-ABE, were respectively proposed by Goyal *et al.* [15] and Bethencourt *et al.* [2]. These schemes are proven secure in the selective security model. Fully secure ABE constructions were provided by Okamoto *et al.* [30,31] and Lewko *et al.* [26]. They follow the dual system encryption methodology due to Waters [36] and Lewko *et al.* [24] to achieve fully security. Another kind of typical LU-PIPE systems, i.e., Revocation Encryption (RE), was introduced by Lewko, Sahai and Waters [23] in the selective and fully security model.

CCA2-Secure LU-PIPE. Many researches have devoted their efforts to the constructions of CCA2-secure LU-PIPE schemes. The Canetti-Halevi-Katz approach [8] is widely used for the CCA2 security transformation at the cost of one-time signatures. Their approach was first applied for converting CPA-secure IBE to CCA2-secure PKE, and converting CPA-secure Hierarchical IBE (HIBE) to CCA2-secure IBE. It was later used to obtain CCA2-secure KP-ABE from CPA-secure KP-ABE by Goyal *et al.* [15], and CCA2-secure CP-ABE from CPA-secure CP-ABE by Cheung *et al.* [2]. Yamada *et al.* [37] generalized this approach and introduced a generic framework to transform CPA-secure ABE to CCA2-secure ABE. Yamada *et al.* [38] further extended this approach into PE settings and showed that any CPA-secure PE scheme could be converted into a CCA2-secure one assuming that the underlying PE scheme is verifiable, i.e., all legitimate receivers of a ciphertext can obtain the same message upon decryption. All the above CCA2-secure PE constructions need one-time signatures. Boyen *et al.* [7] introduced a shrink approach to obtain CCA2-secure PKE from CPA-secure IBE using standard collision resistant hash functions, by exploiting the specific ciphertext structure of the underlying schemes [4]. Recently, Liu *et al.* [28] refined this technique in KP-ABE and proposed a direct CCA2-secure KP-ABE scheme from the Rouselakis-Waters KP-ABE [32].

OO-PIPE. Online/offline cryptosystems were first proposed by Even *et al.* [11]. The goal of design is to have a very short response time after a pre-processing phase in which all expensive operations are pre-computed. They instantiated this technique in the context of digital signatures. The signing process is divided into online and offline phases. Most of the computation work is done in the offline phase without knowledge of the message to be signed. Once the message to be signed is given, the resulting signature can be quickly obtained in the online phase. Shamir and Tauman [35] showed how to use Chameleon hash functions to transform any digital signature scheme to an online/offline signature scheme. In the online phase, one only needs to find a Chameleon hash collision, which usually only requires several modular multiplications [22]. Guo *et al.* [17] considered online/offline variants of the Boneh-Boyen IBE [3] and the Gentry

IBE [14], followed by the work of Liu *et al.* [27]. Subsequently, online/offline HIBE [29] and online/offline Identity-Based Key Encapsulation [10] were proposed. Hohenberger and Waters [18] proposed CPA-secure online/offline ABE schemes based on the Rouselakis-Waters ABE [32]. Most of the existing CCA2-secure OO-PIPE schemes employ the Canetti-Halevi-Katz approach [17,29] with the help of one-time signatures. Chow *et al.* [10] presented a generic transformation to get CCA2-secure OO-IBE from any OO-IBE in the key encapsulation mechanism. Their transformation, inspired by the technique from Fujisaki and Okamoto [13], is actually very efficient, although one needs to model the output of the hash function as a random oracle.

2 Preliminaries

We write $[a, b]$ to denote the set $\{a, a+1, \cdots, b\}$ containing consecutive integers, and $[a]$ as shorthand for $[1, a]$ if there is no ambiguity. For a set S, we use $|S|$ to denote the number of elements in S. We use $s_1, s_2, \cdots, s_n \overset{R}{\leftarrow} S$ for $n \in \mathbb{N}$ to represent that $s_i \overset{R}{\leftarrow} S$ for each $i \in [n]$.

For a randomized algorithm A, we denote $y \leftarrow \mathsf{A}(x; R)$ as the process of running the algorithm A on input x with randomness R to output y, where R is sampled from the space $\mathcal{R}_\mathsf{A}$, i.e., $R \overset{R}{\leftarrow} \mathcal{R}_\mathsf{A}$. We interchangeably use the notations $y \leftarrow \mathsf{A}(x)$ and $y \leftarrow \mathsf{A}(x; R)$, depending on whether we need emphasis on the randomness. We denote $S[\mathsf{A}(x)]$ as the range space of A with the input x.

2.1 Definition of LU-PIPE

We follow the LU-PIPE definition given by Yamada *et al.* [38]. We work in the Key Encapsulation Mechanism (KEM) setting, where the ciphertext hides a symmetric session key key for encryption of regular digital contents. Let $U = \{0,1\}^*$ be an attribute space and $P_n = \{K_n \times E_n \to \{0,1\} | n \in \mathbb{N}\}$ be a large universe predicate family, where n denotes the dimension of the predicate P_n. Let K_n denote the "key attribute" space and E_n denote the "ciphertext attribute" space over U. A Large Universe Public-Index Predicate KEM (LU-PIP-KEM) for P_n consists of four polynomial time algorithms:

$(msk, pp) \leftarrow \mathsf{Setup}(\lambda, n)$. Take as inputs a security parameter $\lambda \in \mathbb{N}$ and a dimension n of the predicate P_n. It outputs a master secret key msk and a public parameter pp.

$sk_x \leftarrow \mathsf{KeyGen}(pp, msk, x)$. Take as inputs the public parameter pp, the master secret key msk, and a key attribute $x \in K_n$. It outputs a secret key sk_x associated with the key attribute x.

$(key, ct_y) \leftarrow \mathsf{Encrypt}(pp, y; R_y)$. Take as inputs the public parameter pp and a ciphertext attribute $y \in E_n$. It outputs a session key key and a ciphertext ct_y associated with the ciphertext attribute y under the randomness R_y.

$key \leftarrow \mathsf{Decrypt}(pp, ct_y, y, sk_x, x)$. Take as inputs the public parameter pp, a ciphertext ct_y associated with the ciphertext attribute $y \in E_n$, and a secret key sk_x associated with the key attribute $x \in K_n$. It outputs the session key key.

A LU-PIP-KEM scheme is correct if for all $(msk, pp) \leftarrow \mathsf{Setup}(\lambda, n)$, all $sk_x \leftarrow \mathsf{KeyGen}(pp, msk, x)$ with $x \in K_n$, and all $(key, ct_y) \leftarrow \mathsf{Encrypt}(pp, y, R_y)$ with $y \in E_n$, it holds that if $P_n(x, y) = 1$, then $\mathsf{Decrypt}(pp, ct_y, y, sk_x, x) = key$; else if $P_n(x, y) = 0$, then $\mathsf{Decrypt}(pp, ct_y, y, sk_x, x) = \perp$.

The chosen plaintext security in LU-PIP-KEM is defined through a game played between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$. Both of them take the security parameter λ and the dimension n of the predicate as input.

Setup. $\mathcal{C}$ runs Setup to generate and give the public parameter pp to $\mathcal{A}$.

Phase 1. $\mathcal{A}$ adaptively submits secret key queries for the key attribute $x \in K_n$. $\mathcal{C}$ generates a secret key sk_x for x and returns it to $\mathcal{A}$.

Challenge. $\mathcal{A}$ outputs a challenge ciphertext attribute $y^* \in E_n$ on which it wishes to be challenged. The challenge ciphertext attribute y^* must satisfy that $P_n(x, y^*) = 0$ for any x that $\mathcal{A}$ has already queried for the secret key sk_x. $\mathcal{C}$ runs $\mathsf{Encrypt}(pp, y^*)$ to obtain (key^*, ct^*). Then, it flips a random coin $b \in \{0, 1\}$. If $b = 0$, $\mathcal{C}$ returns (key^*, ct^*) to $\mathcal{A}$. If $b = 1$, it selects a random session key key_R^* and returns (key_R^*, ct^*).

Phase 2. Phase 1 is repeated with a restriction that $\mathcal{A}$ cannot submit secret key queries for $x \in K_n$ with $P_n(x, y^*) = 1$.

Guess. Finally, $\mathcal{A}$ outputs a guess $b' \in \{0, 1\}$ and wins in the game if $b = b'$.

The advantage of $\mathcal{A}$ in attacking the LU-PIP-KEM system with security parameter λ is defined as $Adv_{\mathcal{A}}^{\text{LU-PIP-KEM}}(\lambda) = \left|\Pr[b' = b] - \frac{1}{2}\right|$.

Definition 1. *A LU-PIP-KEM system is CPA-secure if for any polynomial time adversary $\mathcal{A}$, the advantage of winning the above game is a negligible function ϵ in λ, i.e., $Adv_{\mathcal{A}}^{LU\text{-}PIP\text{-}KEM}(\lambda) < \epsilon$.*

A weaker security notion called selective security can be defined in the above game with an extra **Init** phase in which $\mathcal{A}$ must commit to the challenge ciphertext attribute $y^* \in E_n$ before **Setup**.

We next review the notion of *OR-compatibility* for a predicate. A predicate P_n is said to have *OR-compatibility* if for two ciphertext attributes, the predicate is able to capture the presence of one *or* the other. This property was first introduced by Yamada *et al.* [38]. They commented that *OR-compatibility* is commonly achieved in many concrete LU-PIPE schemes [2,15,21].

Definition 2. *A predicate $P_n = \{K_n \times E_n \to \{0, 1\} | n \in \mathbb{N}\}$ is said to have OR-compatibility if for all $d \in \mathbb{N}$, there exists a map $OR : E_n \times E_d \to E_{n+d}$ and two attribute extension maps $EN : K_n \to K_{n+d}$, $ED : K_d \to K_{n+d}$ such that for all $x_1 \in K_n$, $x_2 \in K_d$, $y_1 \in E_n$, $y_2 \in E_d$,*

$$P_{n+d}(EN(x_1), OR(y_1, y_2)) = P_n(x_1, y_1),$$
$$P_{n+d}(ED(x_2), OR(y_1, y_2)) = P_d(x_2, y_2).$$

2.2 Definition of Public-Verifiability

We review the *public-verifiability* of a LU-PIP-KEM scheme. This property was first defined in the IBE setting [16] and then be extended to FE settings by Yamada *et al.* [37]. Intuitively, a LU-PIP-KEM has *public-verifiability* if there exists a public verification mechanism to verify whether a given ciphertext is honestly generated. As remarked by Abdalla *et al.* [1], any encryption schemes with *public-verifiability* cannot be anonymous (or known as Private Index Predicate Encryption [6]). Hence, *public-verifiability* can only be achieved in PIPE, which is also the focus of this paper.

To define *public-verifiability*, we introduce a polynomial time algorithm Verify.

0 or $1 \leftarrow \mathsf{Verify}(pp, ct_y, y)$. Take as inputs the public parameter pp and a ciphertext $ct_y \in \{0,1\}^*$ under a ciphertext attribute $y \in E_n$. It outputs 0 or 1.

Verify needs to satisfy that for all $(key, ct_y) \in S[\mathsf{Encrypt}(pp, y, R_y)]$, it holds that $\mathsf{Verify}(pp, ct_y, y) = 1$, while for all $(key, ct_y) \notin S[\mathsf{Encrypt}(pp, y, R_y)]$, it must have that $\mathsf{Verify}(pp, ct_y, y) = 0$ except with a negligible probability.

Definition 3. *A LU-PIP-KEM scheme is said to have public-verifiability if there exists an algorithm* Verify *in the LU-PIP-KEM scheme satisfying the completeness requirement defined above.*

3 Modelling OO-PIPE

We formally define OO-PIPE in the KEM setting. An OO-PIP-KEM scheme consists of five polynomial time algorithms OO.Setup, OO.KeyGen, OO.OffEncrypt, OO.OnEncrypt and OO.Decrypt. The definitions of OO.Setup and OO.KeyGen are identical to those of LU-PIP-KEM systems shown in Sect. 2.1. The others are defined as follows.

$ict \leftarrow \mathsf{OO.OffEncrypt}(pp)$. Only take as input the public parameter pp and outputs an intermediate ciphertext ict.

$(key, ct_y) \leftarrow \mathsf{OO.OnEncrypt}(pp, y, ict)$. Take as inputs the public parameter pp, a target ciphertext attribute $y \in E_n$, and an intermediate ciphertext ict. It outputs a session key key and a ciphertext ct_y associated with y.

$key \leftarrow \mathsf{OO.Decrypt}(pp, ct_y, y, sk_x, x)$. Take as inputs the public parameter pp, a ciphertext ct_y associated with the ciphertext attribute $y \in E_n$, and a secret key sk_x associated with the key attribute $x \in K_n$. It outputs the session key key.

The correctness requires that for all $(msk, pp) \leftarrow \mathsf{OO.Setup}(\lambda, n)$, all $x \in K_n$, all $sk_x \leftarrow \mathsf{OO.KeyGen}(pp, msk, x)$, all $ict \leftarrow \mathsf{OO.OffEncrypt}(pp)$, all $y \in E_n$, and all $(key, ct_y) \leftarrow \mathsf{OO.OnEncrypt}(pp, y, ict)$, if $P_n(x, y) = 1$, then we have that

$\mathsf{OO.Decrypt}(pp, ct_y, y, sk_x, x) = key$; else if $P_n(x, y) = 0$, then we have that $\mathsf{OO.Decrypt}(pp, ct_y, y, sk_x, x) = \perp$ except with a negligible probability.

We next define chosen ciphertext security in OO-PIP-KEM. The security model is similarly defined through a game played between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$, both of which are given the parameter λ and the dimension n of the predicate as inputs.

Setup. $\mathcal{C}$ runs $\mathsf{OO.Setup}$ to generate public parameter pp and sends it to $\mathcal{A}$.

Phase 1. $\mathcal{A}$ adaptively issues queries:

– **Secret Key Query**. $\mathcal{A}$ submits a key attribute $x \in K_n$ to $\mathcal{C}$. $\mathcal{C}$ generates and gives a secret key sk_x for x to $\mathcal{A}$.
– **Decryption Query**. $\mathcal{A}$ submits a ciphertext ct_y with ciphertext attribute $y \in E_n$ to $\mathcal{C}$. $\mathcal{C}$ constructs a key attribute $x \in K_n$ with $P_n(x, y) = 1$, and runs $\mathsf{OO.KeyGen}(pp, msk, x)$ to generate a secret key sk_x. It then runs $\mathsf{OO.Decrypt}(pp, ct_y, y, sk_x, x)$ and returns the decryption result to $\mathcal{A}$.

Challenge. $\mathcal{A}$ outputs a challenge ciphertext attribute $y^* \in E_n$ on which it wishes to be challenged. The challenge ciphertext attribute y^* must satisfy that $P_n(x, y^*) = 0$ for any x that $\mathcal{A}$ queried for the secret key sk_x. $\mathcal{C}$ generates a session key key^* and a ciphertext ct^* under the challenge attribute y^*. Then, it flips a random coin $b \in \{0, 1\}$. If $b = 0$, $\mathcal{C}$ returns (key^*, ct^*) to $\mathcal{A}$. Otherwise, it randomly selects a session key key_R^* and returns (key_R^*, ct^*) to $\mathcal{A}$.

Phase 2. $\mathcal{A}$ further adaptively issues the following two kinds of queries:

– **Secret Key Query** for key attributes $x \in K_n$ satisfying $P_n(x, y^*) = 0$.
– **Decryption Query** for the ciphertext ct_y with a constraint that $ct_y \neq ct^*$.

$\mathcal{C}$ responds the same as in **Phase 1**.

Guess. Finally, $\mathcal{A}$ outputs a guess $b' \in \{0, 1\}$ and wins in the game if $b = b'$.

The advantage of $\mathcal{A}$ who issues q_S secret key queries and q_D decryption queries in attacking the OO-PIP-KEM system with security parameter λ is defined as $Adv_{\mathcal{A},q_S,q_D}^{\text{OO-PIP-KEM}}(\lambda) = \left|\Pr[b' = b] - \frac{1}{2}\right|$.

Definition 4. *An OO-PIP-KEM system is CCA2-secure if for any polynomial time adversary $\mathcal{A}$ who makes a total of q_S secret key queries and q_D decryption queries, the advantage of winning the security game defined above is at most negligible function ϵ in λ, i.e., $Adv_{\mathcal{A},q_S,q_D}^{\text{OO-PIP-KEM}}(\lambda) < \epsilon$.*

The CPA security for OO-PIP-KEM system can also be defined as in the preceding game, with a constraint that $\mathcal{A}$ is not allowed to issue decryption queries in **Phase 1** and **Phase 2**.

Definition 5. *An OO-PIP-KEM system is CPA-secure if for any polynomial time adversary $\mathcal{A}$ who makes a total of q_S secret key queries and no decryption query, the advantage of winning the security game defined above is at most negligible function ϵ in λ, i.e., $Adv_{\mathcal{A},q_S,0}^{\text{OO-PIP-KEM}}(\lambda) < \epsilon$.*

Similar to LU-PIP-KEM, the selective security of an OO-PIP-KEM system can be defined in the above game by adding an **Init** phase before **Setup** phase. $\mathcal{A}$ must decide the challenge ciphertext attribute $y^* \in E_n$ in the **Init** phase.

4 CPA-secure OO-PIP-KEM from LU-PIP-KEM

The major challenge in constructing OO-PIP-KEM is that in the offline phase, the encryptor cannot know the ciphertext attribute that a ciphertext will be associated with. We manage to overcome this challenge by identifying a useful property, i.e., *attribute-malleability*, in many LU-PIPE schemes. Coarsely speaking, a LU-PIP-KEM scheme has *attribute-malleability* if an encryptor can malleate a ciphertext ct_{ori} associated with an original ciphertext attribute y_{ori} to a new ciphertext ct_{new} associated with a new ciphertext attribute y_{new} with the same session key key. The ones who have the secret key sk_x with key attribute x satisfying $P(x, y_{new}) = 1$ can also correctly decrypt ct_{new} to recover key.

The *attribute-malleability* enables an encryptor to prepare the ciphertext without knowing the associated ciphertext attribute. In the offline phase, the encryptor randomly chooses a ciphertext attribute y_{ori}, and encapsulates a session key key under that ciphertext attribute to generate a ciphertext ct_{ori}. When the target ciphertext attribute y is available to the encryptor in the online phase, he malleates the ciphertext ct_{ori} with the ciphertext attribute y_{ori} to a target ciphertext ct_y associated with the given ciphertext attribute y with the same session key key. In decryption, the receiver who has the secret key sk_x with the key attribute x satisfying $P(x, y) = 1$ can decrypt the ciphertext ct_y and recover the session key key.

4.1 Definition of Attribute-Malleability

We first introduce three polynomial time algorithms, $\mathsf{PriMalleate}$, $\mathsf{PubMalleate}$, $\mathsf{Combine}$ in LU-PIP-KEM, and their necessary properties.

$y_{mall} \leftarrow \mathsf{PriMalleate}(y_{ori}, y_{new}, R_{ori})$. Take as inputs the original ciphertext attribute $y_{ori} \in E_n$, a new ciphertext attribute y_{new}, and the randomness R_{ori} used to run $(key, ct_{ori}) \leftarrow \mathsf{Encrypt}(pp, y_{ori}; R_{ori})$. It outputs a malleated ciphertext attribute $y_{mall} \in E_n$.

$\widetilde{ct}_{ori} \leftarrow \mathsf{PubMalleate}(pp, \widetilde{ct}_{new}, \widetilde{y}_{mall})$. Take as inputs the public parameter pp, a ciphertext $\widetilde{ct}_{new}$ associated with the new ciphertext attribute y_{new}, and a malleated ciphertext attribute $\widetilde{y}_{mall} \in E_n$. It outputs a ciphertext $\widetilde{ct}_{ori} \in E_n$.

$ct_{new} \leftarrow \mathsf{Combine}(pp, ct_{ori}, y_{mall})$. Take as inputs the public parameter pp, a ciphertext ct_{ori} associated with the ciphertext attribute y_{ori}, and the malleated ciphertext attribute y_{mall}. It outputs a ciphertext ct_{new} associated with the given ciphertext attribute y_{new}.

These algorithms need to meet the following requirements.

- *Private Malleability.* For all $(key, ct_{ori}) \leftarrow \mathsf{Encrypt}(pp, y_{ori}; R_{ori})$ with a randomly chosen ciphertext attribute $y_{ori} \overset{R}{\leftarrow} E_n$ and all ciphertext attribute $y_{new} \in E_n$, if y_{mall} is output by $y_{mall} \leftarrow \mathsf{PriMalleate}(y_{ori}, y_{new}, R_{ori})$, and ct_{new} is generated as $ct_{new} \leftarrow \mathsf{Combine}(pp, ct_{ori}, y_{mall})$, then we have $(key, ct_{new}) = \mathsf{Encrypt}(pp, y_{new}; R_{ori})$.
- *Public Malleability.* For all $(key, \widetilde{ct}_{new}) \leftarrow \mathsf{Encrypt}(pp, y_{new}; R_{new})$ with a ciphertext attribute $y_{new} \in E_n$ and randomly chosen $\widetilde{y}_{mall} \overset{R}{\leftarrow} E_n$, if $\widetilde{ct}_{ori} \leftarrow \mathsf{PubMalleate}(pp, \widetilde{ct}_{new}, \widetilde{y}_{mall})$, then $(key, \widetilde{ct}_{ori}) = \mathsf{Encrypt}(pp, \widetilde{y}_{ori}; R_{new})$. Also, $\widetilde{ct}_{new} = \mathsf{Combine}(pp, \widetilde{ct}_{ori}, \widetilde{y}_{mall})$.
- *Efficiency.* Running $y_{mall} \leftarrow \mathsf{PriMalleate}(y_{ori}, y_{new}, R_{ori})$ for all $y_{ori}, y_{new} \in E_n$ is more efficient than running $(key, ct_{new}) \leftarrow \mathsf{Encrypt}(pp, y_{new}; R_{new})$.

Definition 6. *We say a LU-PIP-KEM scheme has attribute-malleability if there exist polynomial time algorithms* $\mathsf{PriMalleate}$, $\mathsf{PubMalleate}$ *and* $\mathsf{Combine}$ *satisfying private malleability, public malleability and efficiency defined above.*

4.2 Generic Transformation

We now describe our transformation. Let $\Pi' = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ be a CPA-secure LU-PIP-KEM scheme for predicate P_n over the attribute universe $U = \{0,1\}^*$ that has *attribute-malleability* defined in Definition 6. We can construct a CPA-secure OO-PIP-KEM scheme $\Pi =$($\mathsf{OO.Setup}$, $\mathsf{OO.KeyGen}$, $\mathsf{OO.OffEncrypt}$, $\mathsf{OO.OnEncrypt}$, $\mathsf{OO.Decrypt}$) for the same predicate P_n as follows.

$\mathsf{OO.Setup}(\lambda, n)$. The setup algorithm imply invokes $(msk, pp) \leftarrow \mathsf{Setup}(\lambda, n)$ and outputs the master secret key and the public parameter as (msk, pp).

$\mathsf{OO.KeyGen}(pp, msk, x)$. Given a key attribute $x \in K_n$, the key generation algorithm simply calls $sk_x \leftarrow \mathsf{KeyGen}(pp, msk, x)$ and outputs the secret key sk_x.

$\mathsf{OO.OffEncrypt}(pp)$. The offline encryption algorithm will generate a ciphertext under a randomly chosen ciphertext attribute and treat it as an intermediate ciphertext. In detail, it randomly chooses $y_{ori} \overset{R}{\leftarrow} E_n$. Then, it runs $(key, ct_{ori}) \leftarrow \mathsf{Encrypt}(pp, y_{ori}; R_{ori})$ with randomly chosen randomness R_{ori} to obtain a session key and a ciphertext associated with the original ciphertext attribute y_{ori}. The intermediate ciphertext is $ict = (key, y_{ori}, ct_{ori}, R_{ori})$.

$\mathsf{OO.OnEncrypt}(pp, y, ict)$. When knowing the target ciphertext attribute $y \in E_n$, the online encryption algorithm first runs $y_{mall} \leftarrow \mathsf{PriMalleate}(y_{ori}, y, R_{ori})$ to obtain a malleated ciphertext attribute $y_{mall} \in E_n$. The session key key is unchange. The ciphertext associated with the ciphertext attribute y is $ct_y = (ct_{ori}, y_{mall})$. Note that the online encryption procedure only involves operations for running algorithm $\mathsf{PriMalleate}$.

$\mathsf{OO.Decrypt}(pp, ct_y, y, sk_x, x)$. If $P_n(x, y) = 0$, then the key attribute x does not satisfy the predicate P_n for the ciphertext attribute y and the decryption algorithm simply outputs $\perp$. Otherwise, it first parses ct_y as (ct_{ori}, y_{mall}). Then, it runs $ct_y \leftarrow \mathsf{Combine}(pp, ct_{ori}, y_{mall})$ and gets a ciphertext ct_y associated with the ciphertext attribute y. It runs $key \leftarrow \mathsf{Decrypt}(pp, ct_y, y, sk_x, x)$ to recover the session key key.

Correctness. Due to the *private malleability*, for the session key and the ciphertext generated by calling $(key, ct_{ori}) \leftarrow \mathsf{Encrypt}(pp, y_{ori}; R_{ori})$ in $\mathsf{OO.OffEncrypt}$ with the randomly chosen $y_{ori} \overset{R}{\leftarrow} E_n$ and for $y_{mall} \leftarrow \mathsf{PriMalleate}(y_{ori}, y, R_{ori})$, we get a LU-PIP-KEM ciphertext associated with the ciphertext attribute y by running $ct_y \leftarrow \mathsf{Combine}(pp, ct_{ori}, y_{mall})$ in the decryption algorithm. Therefore, if a secret key associated with key attribute $x \in K_n$ satisfies $P_n(x, y) = 1$, then the decryption algorithm can correctly recover the session key by running $key \leftarrow \mathsf{Decrypt}(pp, ct_y, y, sk_x, x)$.

Performance. Only operations for running $\mathsf{PriMalleate}$ are required in the online encryption procedure, whereas in the original LU-PIP-KEM, the encryption procedure involves running algorithm $\mathsf{Encrypt}$. With the *efficiency* requirement, for all $y_{new} \in E_n$, running $\mathsf{PriMalleate}$ is more efficient than running $\mathsf{Encrypt}$. Therefore, the efficiency of the online encryption procedure is improved.

4.3 Security Analysis

The CPA security of our OO-PIP-KEM relies on the CPA security of the underlying LU-PIP-KEM. The major obstacle in the security proof is how to convert the challenge LU-PIP-KEM ciphertext into a challenge OO-PIP-KEM ciphertext in the **Challenge** phase. We overcome this obstacle by exploiting the *public malleability* implied by *attribute-malleability*.

When obtaining the challenge LU-PIP-KEM session key $\widetilde{key}^*$ and ciphertext $\widetilde{ct}^*$ associated with the challenge ciphertext attribute y^* from the LU-PIP-KEM challenger, we randomly choose a malleated ciphertext attribute $\widetilde{y}^*_{mall} \in E_n$ and calls $\widetilde{ct}^*_{ori} \leftarrow \mathsf{PubMalleate}(pp, \widetilde{ct}^*, \widetilde{y}^*_{mall})$ to obtain a ciphertext $\widetilde{ct}^*_{ori}$. We then construct the challenge OO-PIP-KEM ciphertext as $ct^* = (\widetilde{ct}^*_{ori}, \widetilde{y}^*_{mall})$.

- Since $\widetilde{ct}^*_{ori} \leftarrow \mathsf{Encrypt}(pp, \widetilde{y}^*_{ori})$, $\widetilde{ct}^*_{ori}$ is a LU-PIP-KEM ciphertext.
- Since $\widetilde{ct}^* = \mathsf{Combine}(pp, \widetilde{ct}^*_{ori}, \widetilde{y}^*_{mall})$, $\widetilde{ct}^*$ is associated with y^*.

Therefore, ct^* is a well-formed challenge OO-PIP-KEM ciphertext for the ciphertext attribute y^* due to the *public malleability*. In this way, the challenge ciphertext simulation in the **Challenge** phase goes through. The formal proof is shown in the full version of the paper.

Theorem 1. *If the underlying LU-PIP-KEM for predicate P_n is CPA-secure and attribute-malleable, then the proposed OO-PIP-KEM scheme is CPA-secure for the same predicate P_n.*

5 CCA2-secure OO-PIP-KEM from LU-PIP-KEM

5.1 Universally Collision Resistant Chameleon Hash Function

Collision Resistant Chameleon Hash. A Chameleon hash [22] has a hash key chk and a trapdoor td. Anyone knowing the hash key chk can efficiently compute the hash value for any given input. There also exists an efficient algorithm for the holder of the trapdoor td to find collisions for every given input. However, it is impossible for others unaware of td to compute collisions for any given input, except with a negligible probability.

A Chameleon hash function [22] family CH with hash value space $\mathcal{H}$ consists of three polynomial time algorithms CHGen, CHash and Coll defined as follows.

$(chk, td) \leftarrow \mathsf{CHGen}(\lambda)$. Take the security parameter $\lambda \in \mathbb{N}$ as input, and outputs a Chameleon hash key/trapdoor pair (chk, td).

$H \leftarrow \mathsf{CHash}(chk, m, r)$. Take as inputs the Chameleon hash key chk, a message m, and an auxiliary random parameter r. It outputs the hash value $H \in \mathcal{H}$ for the given message m.

$r' \leftarrow \mathsf{Coll}(td, m, r, m')$. Take as inputs the Chameleon hash trapdoor td, a message m with its auxiliary random parameter r for previously calculating the hash value H, and another message $m' \neq m$. It outputs another auxiliary random parameter r' such that

$$\mathsf{CHash}(chk, m, r) = \mathsf{CHash}(chk, m', r') = H$$

A Chameleon hash function should satisfy the *collision resistance* requirement, i.e., given the Chameleon hash key chk as input, no efficient algorithm can find two pairs $(m, r) \neq (m', r')$ such that $\mathsf{CHash}(chk, m, r) = \mathsf{CHash}(chk, m', r')$ except with a negligible probability.

Universally Collision Resistant Chameleon Hash. Our construction exploits Chameleon hash with universal collision resistance. A Chameleon hash function family is *universal collision resistant* if even though the attacker is allowed to choose the Chameleon hash key chk, it remains hard to find a hash collision for any given input. Roughly speaking, the hash value H can be only computed using the fixed Chameleon hash key chk.

We denote such a Chameleon hash family as UCH consisting of algorithms UCHGen, UCHash, UColl. Formally, UCH is universally collision resistant if, given only a description of the Chameleon hash function family, no efficient algorithm can find two tuples $(chk, m, r) \neq (chk', m', r')$ such that $\mathsf{UCHash}(chk, m, r) = \mathsf{UCHash}(chk', m', r')$ except with a negligible probability.

Generic Construction of UCH. We can construct *universally collision resistant* Chameleon hash functions based on any regular Chameleon hash and a standard cryptographic hash $\mathsf{Hash} : \{0,1\}^* \rightarrow \mathcal{H}$. The construction is as follows.

$\mathsf{UCHGen}(\lambda)$. The hash key/trapdoor pair is $(chk, td) \leftarrow \mathsf{CHGen}(\lambda)$.

$\mathsf{UCHash}(chk, m, r)$. The hash value is $H = \mathsf{Hash}(\mathsf{CHash}(chk, m, r) \| chk)$.

$\mathsf{UColl}(td, m, r, m')$. Directly output $r' \leftarrow \mathsf{Coll}(td, m, r, m')$.

One with the trapdoor td can still find collisions for any given input since

$$\begin{aligned} H = \mathsf{UCHash}(chk, m, r) &= \mathsf{Hash}(\mathsf{CHash}(chk, m, r) \| chk) \\ &= \mathsf{Hash}(\mathsf{CHash}(chk, m', r') \| chk) = \mathsf{UCHash}(chk, m', r') \end{aligned}$$

Without td, any polynomial time algorithm cannot find two tuples $(chk, m, r) \neq (chk', m', r')$ with $H = \mathsf{UCHash}(chk, m, r) = \mathsf{UCHash}(chk', m', r')$. Otherwise,

$$\begin{aligned} &\mathsf{UCHash}(chk, m, r) = \mathsf{Hash}(\mathsf{CHash}(chk, m, r) \| chk) \\ = &\mathsf{UCHash}(chk', m', r') = \mathsf{Hash}(\mathsf{CHash}(chk', m', r') \| chk') \end{aligned}$$

which implies that we find a collision for either Hash or CH, contradicting to their security notion.

5.2 Basic Idea

The *public-verifiability* in LU-PIPE allows a ciphertext verification mechanism, i.e., testing whether the ciphertext is honestly generated with the assigned ciphertext attribute. We can leverage such a built-in verification mechanism to construct OO-PIPE with CCA2 security. Precisely, we add an on-the-fly verification attribute y_v in the ciphertext. We split the attribute universe U into two parts: one is the regular attribute universe $\mathcal{U}$, and another is the verification attribute universe $\mathcal{V}$ for the verification attributes. The verification attribute $y_v \in \mathcal{V}$ is only used for ciphertext verification. In encryption, the encryptor hashes the components of a ciphertext, and treats the result as the ciphertext attribute y_v to encrypt again. In the decryption procedure, the receiver computes the hash result again, and verifies whether the ciphertext is encrypted under the assigned ciphertext attribute, and under the hash ciphertext attribute y_v using the ciphertext verification mechanism.

Similar built-in verification has been used by Boyen *et al.* [7]. However, one may encounter an obstacle when directly employing their technique. The online/offline mechanism implies ciphertext forgery in the sense that a ciphertext with an ciphertext attribute can be efficiently malleated to a target ciphertext with a genuine ciphertext attribute, while any efficient ciphertext forgery must be prevented in CCA2 security. A plausible solution is to follow the technique proposed by Liu *et al.* [28] by replacing the regular hash to a Chameleon hash function. With the help of hash collision algorithm Coll in the Chameleon hash function, it is possible to malleate the ciphertext with an ciphertext attribute to a target ciphertext with the genuine ciphertext attribute, while remaining the verification terms unchange. However, for invoking hash collision algorithm, all encryptors must know the trapdoor of the target Chameleon hash key bounded in the public parameter, which obviously implies security problem.

To circumvent this obstacle, we use a "dynamic" universally collision resistant Chameleon hash to replace the regular Chameleon hash for each ciphertext. In offline encryption, the encryptor generates a Chameleon hash key/trapdoor pair (chk, td), chooses a random ciphertext attribute y_{ori}, and calculates the intermediate ciphertext components for y_{ori} and the temporary hash value y_v. When learning the genuine ciphertext attribute in the online phase, the encryptor replaces the random ciphertext attribute with the genuine one, while leveraging UCHash with the trapdoor td to remain y_v unchange. The cost is an additional Chameleon hash key chk in the ciphertext. In the online phase, the encryptor will run UColl, which is efficient in some Chameleon hash instantiations based on discrete log [22]. In this way, the online encryption cost keeps low.

5.3 Generic Transformation

Let Π' be a CPA-secure LU-PIP-KEM scheme consisting of four algorithms Setup, KeyGen, $\mathsf{Encrypt}$, $\mathsf{Decrypt}$ for predicate P_n over the attribute universe $U = \{0,1\}^*$. Suppose that the predicate P_n has *OR-compatibility* defined in Definition 2, Π' has *attribute-malleability* defined in Definition 6, and Π' has *public-verificability* defined in Definition 3. We below construct a CCA2-secure OO-PIP-KEM scheme Π including the algorithms $\mathsf{CCA.Setup}$, $\mathsf{CCA.KeyGen}$, $\mathsf{CCA.OffEncrypt}$, $\mathsf{CCA.OnEncrypt}$, $\mathsf{CCA.Decrypt}$ for the same predicate P_n over the regular attribute universe $\mathcal{U}$ and the verification attribute universe $\mathcal{V}$ with $|\mathcal{U}| = |\mathcal{V}|$, $\mathcal{U} \cap \mathcal{V} = \emptyset$ and $\mathcal{U} \cup \mathcal{V} = U$.

$\mathsf{CCA.Setup}(\lambda, n)$. The setup algorithm runs $(msk, pp) \leftarrow \mathsf{Setup}(\lambda, n+d)$. Then, it chooses a secure UCH function $\mathsf{UCH} : \{0,1\}^* \rightarrow E_d$ with an auxiliary parameter universe $\mathcal{R}$. The system restricts that E_d is over $\mathcal{V}$. The master secret key is msk. The public parameter is published as $(pp, \mathsf{UCH}, \mathcal{R})$.

$\mathsf{CCA.KeyGen}(pp, msk, x)$. Given the key attribute $x \in K_n$, the algorithm first extends x to $EN(x) \in K_{n+d}$ using the map EN. Then, it runs $sk_{EN(x)} \leftarrow \mathsf{KeyGen}(pp, msk, EN(x))$ and outputs the secret key $sk_x = sk_{EN(x)}$.

$\mathsf{CCA.OffEncrypt}(pp)$. The offline encryption algorithm first randomly chooses an original ciphertext attribute $y_{ori} \overset{R}{\leftarrow} E_n$. Then, it runs $(chk, td) \leftarrow \mathsf{UCHGen}(\lambda)$. It next picks a random $r' \overset{R}{\leftarrow} \mathcal{R}$, and calculates an on-the-fly verification attribute $y_v = \mathsf{UCHash}\,(chk, y_{ori}, r')$. It uses map OR to obtain the ciphertext attribute $OR(y_{ori}, y_v) \in E_{n+d}$ and runs $(key, ct_{ori}) \leftarrow \mathsf{Encrypt}(pp, OR(y_{ori}, y_v); R_{ori})$ with randomness R_{ori} to generate the session key and the ciphertext. The intermediate ciphertext is $ict = (key, y_{ori}, y_v, ct_{ori}, R_{ori}, chk, td, r')$.

$\mathsf{CCA.OnEncrypt}(pp, y, ict)$. Once the target ciphertext attribute $y \in E_n$ is available, the online encryption algorithm extends the ciphertext attribute $y \in E_n$ to $OR(y, y_v)$ and obtains a malleated ciphertext attribute $y_{mall} \in E_{n+d}$ by running $y_{mall} \leftarrow \mathsf{PriMalleate}(OR(y_{ori}, y_v), OR(y, y_v), R_{ori})$. It next runs $r \leftarrow \mathsf{UColl}(td, y_{ori}, r', ct_{ori} \| y_{mall})$. The session key is key, while the ciphertext ct_y associated with the ciphertext attribute y is $ct_y = (ct_{ori}, y_{mall}, chk, r)$. Note that the online encryption algorithm only needs invocations of $\mathsf{PriMalleate}$ and UColl.

$\mathsf{CCA.Decrypt}(pp, ct_y, y, sk_x, x)$. The decryption algorithm recovers the on-the-fly verification attribute $y_v = \mathsf{UCHash}\,(chk, ct_{ori} \| y_{mall}, r)$. Then, it runs $ct_y \leftarrow \mathsf{Combine}(pp, ct_{ori}, y_{mall})$ to rebuild the ciphertext ct_y with the ciphertext attribute $OR(y, y_v)$. One can verify whether the ciphertext is legitimate by testing

$$\mathsf{Verify}\,(pp, ct_y, OR(y, y_v)) \overset{?}{=} 1$$

The property of Chameleon hash ensures $y_v = \mathsf{UCHash}\,(chk, ct_{ori} \| y_{mall}, r) = \mathsf{UCHash}\,(chk, y_{ori}, r')$ and the on-the-fly verification attribute remains the same in the online encryption procedure. If Verify outputs 0, the ciphertext is invalid and the decryption algorithm simply outputs $\perp$. Otherwise, the decryption algorithm runs $key \leftarrow \mathsf{Decrypt}(pp, ct_y, OR(y, y_v), sk_x, EN(x))$ to recover key.

Correctness. If the ciphertext ct_y is honestly generated by the encryptor with the ciphertext attribute y, then $(key, ct_y) = \mathsf{Encrypt}(pp, OR(y, y_v))$ for $ct_y \leftarrow \mathsf{Combine}(pp, ct_{ori}, y_{mall})$, where y_v can be correctly obtained by invoking $y_v = \mathsf{UCHash}\,(chk, ct_{ori} \| y_{mall}, r)$. Hence, we have that $\mathsf{Verify}(pp, ct_y, OR(y, y_v)) = 1$. The decryption can be done using $sk_x = sk_{EN(x)}$ for $P_{n+d}\,(EN(x), OR(y, y_v)) = P_n(x, y) = 1$. The session key can be correctly recovered with

$$key = \mathsf{Decrypt}(pp, ct_y, OR(y, y_v), sk_x, EN(x)).$$

Performance. Comparing with OO-PIP-KEM, operations for running UColl are additionally required in the online encryption of our CCA2-secure OO-PIP-KEM construction. By properly applying Chameleon hash functions with rather efficient algorithm Coll [22], and by our construction shown in Sect. 5.1, UColl is also efficient. Therefore, the online encryption algorithm remains efficient. The additional communication cost is the extra ciphertext components chk, r, both of which have constant size in all existing Chameleon hash instantiations.

5.4 Security Analysis

Our OO-PIP-KEM is CCA2-secure if the underlying LU-PIP-KEM is CPA-secure. The obstacle in the CCA2 security proof is how to respond the decryption queries for ciphertexts associated with the challenge ciphertext attributes y^*.

We overcome this obstacle by using the extended key attribute $x_d \in K_d$ and the extended verification attribute $y_v \in E_d$. In the **Challenge** phase, the challenge attribute for the LU-PIP-KEM challenger is extended to $OR(y^*, y_v^*)$. When the adversary issues a decryption query for a ciphertext ct_y associated with a ciphertext attribute $OR(y^*, y_v)$, where y_v is its verification ciphertext attribute corresponding to ct_y, we first run Verify to check the validity of the ciphertext. The *public-verifiability* ensures that Verify outputs 1 if and only if the ciphertext is honestly generated. Then, we construct a key attribute $x_v \in E_d$ such that $P(x_v, y_v) = 1$, and issues the secret key associated with $ED(x_v) \in K_{n+d}$ to the LU-PIP-KEM challenger. On one hand, the *OR-compatibility* ensures $P_{n+d}(ED(x_v), OR(y^*, y_v)) = P_d(x_v, y_v)$ so that we can use this secret key to decrypt the ciphertext. On the other hand, the universal collision resistance of UCH implies $y_v \neq y_v^*$ except with a negligible probability. Hence, we have $P(ED(x_v), OR(y^*, y_v^*)) = P_d(x_v, y_v^*) = 0$, and the secret key query is valid to the LU-PIP-KEM challenger. The decryption query is perfectly responded.

The universal collision resistance of UCH is crucial for the security proof. Although chk^* in the challenge ciphertext is chosen by the encryptor, and y_v^* is generated honestly, if the Chameleon hash only hash collision resistance property, it is possible for the adversary to replace chk^* to others of its choice, while remaining y_v^* unchange. In detail, if the Chameleon hash is only collision resistant, after obtaining the challenge ciphertext $ct^* = (\widetilde{ct}^*_{ori}, \widetilde{y}^*_{mall}, chk^*, r^*)$, the adversary can replace chk^* with a hash key $chk'_{\mathcal{A}}$ of its own choice, for which it knows its trapdoor $td'_{\mathcal{A}}$ in order to construct a ciphertext $ct' = (\widetilde{ct}'_{ori}, \widetilde{y}'_{mall}, chk'_{\mathcal{A}}, r'_{\mathcal{A}})$, where $(chk^*, r^*) \neq (chk'_{\mathcal{A}}, r'_{\mathcal{A}})$ but $y'_v = y^*_v$. In this case, the decryption oracle would be stuck. The universal collision resistance of the Chameleon hash family prevents the adversary from such attacks since the hash key chk^* is fixed into the hash value and can be verified by the decryption oracle. The formal security proof is shown in the full version of the paper.

Theorem 2. *The proposed OO-PIP-KEM is CCA2-secure if the underlying CPA-secure LU-PIP-KEM has the properties of attribute-malleability, public-verifiability and OR-compatibility.*

6 Instantiations

Our OO-PIP-KEM transformations can apply to existing LU-PIPE schemes, including OO-IBE schemes proposed by Guo *et al.* [17], and OO-ABE schemes proposed by Hohenberger and Waters [18]. In addition, one can illustratively instantiate a new OO-PIP-KEM scheme by applying our transformation to a LU-PIP-KEM scheme. In 2010, Lewko, Sahai and Waters proposed a revocation encryption (RE) scheme [23]. The ciphertext is associated with an identity set of revoked users. Users who are not in the revoked set can decrypt. It can be shown that their RE satisfies *attribute-malleability* and *public-verifiability*. Hence, one can obtain an OO-RE scheme in the KEM setting by following our generic transformation.

7 Conclusion

We provided a general framework for constructing CCA2-secure OO-PIPE. We proposed a generic transformation from attribute-malleable LU-PIP-KEM to OO-PIP-KEM with CPA security. We further transformed CPA-secure LU-PIP-KEM to CCA2-secure OO-PIP-KEM at the cost of a Chameleon hash, assuming the underlying LU-PIP-KEM has *attribute-malleability* and *public-verifiability*.

Acknowledgement. This paper is supported by the Natural Science Foundation of China through projects 61370190, 61272501, 61402029, 61472429, 61202465 and 61532021, by the Guangxi natural science foundation through project 2013GXNSFBB053005. K. Liang is supported by privacy-aware retrieval and modelling of genomic data (No. 13283250), the Academy of Finland.

References

1. Abdalla, M., et al.: Searchable encryption revisited: consistency properties, relation to anonymous IBE, and extensions. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 205–222. Springer, Heidelberg (2005)

2. Bethencourt, J., Sahai, A., Waters, B.: Ciphertext-policy attribute-based encryption. In: S&P 2007, pp. 321–334 (2007)
3. Boneh, D., Boyen, X.: Efficient selective-ID secure identity-based encryption without random oracles. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 223–238. Springer, Heidelberg (2004)
4. Boneh, D., Boyen, X., Goh, E.-J.: Hierarchical identity based encryption with constant size ciphertext. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 440–456. Springer, Heidelberg (2005)
5. Boneh, D., Franklin, M.: Identity-based encryption from the weil pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer, Heidelberg (2001)
6. Boneh, D., Sahai, A., Waters, B.: Functional encryption: definitions and challenges. In: Ishai, Y. (ed.) TCC 2011. LNCS, vol. 6597, pp. 253–273. Springer, Heidelberg (2011)
7. Boyen, X., Mei, Q., Waters, B.: Direct chosen ciphertext security from identity-based techniques. In: CCS 2005, pp. 320–329. ACM (2005)
8. Canetti, R., Halevi, S., Katz, J.: Chosen-ciphertext security from identity-based encryption. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 207–222. Springer, Heidelberg (2004)
9. Cheung, L., Newport, C.: Provably secure ciphertext policy abe. In: CCS 2007, pp. 456–465. ACM (2007)
10. Chow, S.S.M., Liu, J.K., Zhou, J.: Identity-based online/offline key encapsulation and encryption. In: Cheung, B.S.N., Hui, L.C.K., Sandhu, R.S., Wong, D.S. (eds.) ASIACCS 2011, pp. 52–60. ACM (2011)
11. Even, S., Goldreich, O., Micali, S.: On-line/off-line digital signatures. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 263–275. Springer, Heidelberg (1990)
12. Fiat, A., Naor, M.: Broadcast encryption. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 480–491. Springer, Heidelberg (1994)
13. Fujisaki, E., Okamoto, T.: Secure integration of asymmetric and symmetric encryption schemes. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 537–554. Springer, Heidelberg (1999)
14. Gentry, C.: Practical identity-based encryption without random oracles. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 445–464. Springer, Heidelberg (2006)
15. Goyal, V., Pandey, O., Sahai, A., Waters, B.: Attribute-based encryption for fine-grained access control of encrypted data. In: CCS 2006, pp. 89–98. ACM (2006)
16. Green, M., Hohenberger, S.: Blind identity-based encryption and simulatable oblivious transfer. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 265–282. Springer, Heidelberg (2007)
17. Guo, F., Mu, Y., Chen, Z.: Identity-based online/offline encryption. In: Tsudik, G. (ed.) FC 2008. LNCS, vol. 5143, pp. 247–261. Springer, Heidelberg (2008)
18. Hohenberger, S., Waters, B.: Online/offline attribute-based encryption. In: Krawczyk, H. (ed.) PKC 2014. LNCS, vol. 8383, pp. 293–310. Springer, Heidelberg (2014)
19. Huan, J., Yang, Y., Huang, X., Yuen, T.H., Li, J., Cao, J.: Accountable mobile e-commerce scheme via identity-based plaintext-checkable encryption. Inf. Sci. **345**, 143–155 (2016)
20. Huang, X., Liu, J.K., Tang, S., Xiang, Y., Liang, K., Xu, L., Zhou, J.: Cost-effective authentic and anonymous data sharing with forward security. IEEE Trans. Comput. **64**(4), 971–983 (2015)
21. Katz, J., Sahai, A., Waters, B.: Predicate encryption supporting disjunctions, polynomial equations, and inner products. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 146–162. Springer, Heidelberg (2008)

22. Krawczyk, H., Rabin, T.: Chameleon signatures. In: NDSS 2000. The Internet Society (2000)
23. Lewko, A., Sahai, A., Waters, B.: Revocation systems with very small private keys. In: S&P 2010, pp. 273–285. IEEE (2010)
24. Lewko, A., Waters, B.: New techniques for dual system encryption and fully secure HIBE with short ciphertexts. In: Micciancio, D. (ed.) TCC 2010. LNCS, vol. 5978, pp. 455–479. Springer, Heidelberg (2010)
25. Lewko, A., Waters, B.: Decentralizing attribute-based encryption. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 568–588. Springer, Heidelberg (2011)
26. Lewko, A., Waters, B.: New proof methods for attribute-based encryption: achieving full security through selective techniques. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 180–198. Springer, Heidelberg (2012)
27. Liu, J.K., Zhou, J.: An efficient identity-based online/offline encryption scheme. In: Abdalla, M., Pointcheval, D., Fouque, P.-A., Vergnaud, D. (eds.) ACNS 2009. LNCS, vol. 5536, pp. 156–167. Springer, Heidelberg (2009)
28. Liu, W., Liu, J., Wu, Q., Qin, B., Zhou, Y.: Practical direct chosen ciphertext secure key-policy attribute-based encryption with public ciphertext test. In: Kutyłowski, M., Vaidya, J. (eds.) ICAIS 2014, Part II. LNCS, vol. 8713, pp. 91–108. Springer, Heidelberg (2014)
29. Liu, Z., Xu, L., Chen, Z., Mu, Y., Guo, F.: Hierarchical identity-based online/offline encryption. In: ICYCS 2008, pp. 2115–2119. IEEE (2008)
30. Okamoto, T., Takashima, K.: Fully secure functional encryption with general relations from the decisional linear assumption. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 191–208. Springer, Heidelberg (2010)
31. Okamoto, T., Takashima, K.: Fully secure unbounded inner-product and attribute-based encryption. In: Wang, X., Sako, K. (eds.) ASIACRYPT 2012. LNCS, vol. 7658, pp. 349–366. Springer, Heidelberg (2012)
32. Rouselakis, Y., Waters, B.: Practical constructions and new proof methods for large universe attribute-based encryption. In: CCS 2013, pp. 463–474. ACM (2013)
33. Sahai, A., Waters, B.: Fuzzy identity-based encryption. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 457–473. Springer, Heidelberg (2005)
34. Shamir, A.: Identity-based cryptosystems and signature schemes. In: Blakely, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 47–53. Springer, Heidelberg (1985)
35. Shamir, A., Tauman, Y.: Improved online/offline signature schemes. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 355–367. Springer, Heidelberg (2001)
36. Waters, B.: Dual system encryption: realizing fully secure IBE and HIBE under simple assumptions. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 619–636. Springer, Heidelberg (2009)
37. Yamada, S., Attrapadung, N., Hanaoka, G., Kunihiro, N.: Generic constructions for chosen-ciphertext secure attribute based encryption. In: Catalano, D., Fazio, N., Gennaro, R., Nicolosi, A. (eds.) PKC 2011. LNCS, vol. 6571, pp. 71–89. Springer, Heidelberg (2011)
38. Yamada, S., Attrapadung, N., Santoso, B., Schuldt, J.C.N., Hanaoka, G., Kunihiro, N.: Verifiable predicate encryption and applications to CCA security and anonymous predicate authentication. In: Fischlin, M., Buchmann, J., Manulis, M. (eds.) PKC 2012. LNCS, vol. 7293, pp. 243–261. Springer, Heidelberg (2012)
39. Yeh, L., Huang, J.: Pbs: a portable billing scheme with fine-grained access control for service-oriented vehicular networks. IEEE Trans. Mob. Comput. **13**(11), 2606–2619 (2014)