Constructing CCA-secure predicate encapsulation schemes from CPA-secure schemes and universal one-way hash functions

Johannes Blömer bloemer@upb.de
Gennadij Liske gennadij.liske@upb.de

Department of Computer Science, University of Paderborn, Germany*

June 28, 2014

Abstract. We present a new transformation of chosen-plaintext secure predicate encryption schemes with public index into chosen-ciphertext secure schemes. Our construction requires only a universal one-way hash function and is selectively secure in the standard model. The transformation is not generic but can be applied to various existing schemes constructed from bilinear groups. Using common structural properties of these schemes we provide an efficient and simple transformation without overhead in form of one-time signatures or message authentication codes as required in the known generic transformations.

Keywords: predicate key encapsulation mechanism, chosen-ciphertext security, predicate encryption with public index, bilinear groups, universal one-way hash function

1 Introduction

Traditionally, encryption schemes are used to ensure confidentiality during one to one communication, whereas modern applications also require encryption schemes which can be applied in much more sophisticated scenarios. Consequently, various types of encryption schemes for different applications have been proposed during the last decades in order to meet novel demands. Many of these schemes fall into the class of predicate encryption schemes with public index (cf. [BSW11]), which will be considered in this paper.

The goal of predicate encryption schemes with public index is still to ensure confidentiality, but the sender does not encrypt messages for some specified receiver anymore. Rather, the encrypter adds some index Y to the ciphertext, whereas the user i obtains a secret key for some index X_i. User i will be able to decrypt a ciphertext if and only if the ciphertext index and the users key index match according to some predicate P, that is if $P(X_i, Y) = 1$. The class of predicate encryption schemes with public index covers such schemes as identity-based encryption (IBE) (e.g. [Wat05, BBG05]), attribute-based encryption (ABE) (e.g. [SW05, GPSW06, Wat11, RW13]), broadcast encryption (BE) (e.g. [BGW05, BW06]) and others.

It is widely accepted in the cryptographic community, that the notion of chosen-ciphertext security (CCA-security) is the right notion of security for encryption schemes. Nevertheless, novel encryption schemes are usually constructed to withstand only chosen-plaintext attacks (CPAs). Achieving CCA-secure schemes is mostly considered separately due to several reasons. On the one hand, CPA-secure constructions are already quite complex. On the other hand, in many contexts there exist generic and quite efficient transformations of CPA-secure encryption schemes into CCA-secure encryption schemes [BCHK06, YAHK11, YAS+12]. Hence, researchers often just propose to

* This work was partially supported by the German Research Foundation (DFG) within the Collaborative Research Centre On-The-Fly Computing (SFB 901).

apply such transformations and the question of finding specific and possibly more efficient CCA-secure constructions for novel schemes is often dropped.

Enhancing of CPA-secure encryption schemes in order to achieve efficient CCA-secure schemes is a laborious but promising work as has been shown for various constructions [BMW05, Kil06a, Kil06b, PPSS13, BL13]. Hence, the aim of our work was to analyze these constructions and to identify structural properties useful to provide efficient CCA-secure constructions for various predicate encryption schemes. Before we discuss our work in more detail we review the related work.

Related work. Fujisaki and Okamoto [FO13] presented a generic transformation to achieve CCA-secure public key encryption schemes from schemes satisfying some weaker notions of security. Even though defined for conventional public key encryption schemes and being secure only in the random oracle model this transformation is often proposed to apply to predicate encryption schemes.

In [BCHK06] a generic transformation from any two-level hierarchical IBE schemes to a CCA-secure IBE scheme is presented. For attribute-based encryption schemes and for predicate encryption schemes generic transformations from CPA-secure schemes to CCA-secure schemes were proposed in [YAHK11] and [YAS$^+$12] respectively. All these schemes add some overhead to the ciphertexts in form of a one-time signature, MAC or commitment scheme with their respective keys. The schemes in [YAHK11, YAS$^+$12] also make some demands on the original schemes such as delegatability, verifiability and additional properties of predicates.

The non-generic approach is followed in [BMW05, Kil06a, KG09, PPSS13, BL13], where CCA-security is achieved directly based on particular CPA-secure encryption schemes. These schemes are more efficient than CCA-secure constructions achieved from original schemes using generic transformations. If one looks carefully at these schemes one observes that they are based on the same general idea and construction. This is the starting point for our work.

Our approach and contribution. We present an efficient and at the same time widely applicable transformation of CPA-secure predicate encryption schemes into CCA-secure schemes. Our transformation works for schemes constructed from bilinear maps. Moreover, we also need the original schemes to satisfy some structural properties. We clearly define these properties and show that many schemes proposed previously have these properties. The resulting constructions are provable secure in the selective security model.

Although efficient generic transformations are known, our non generic transformation shows that, using structural properties of the original schemes, it is possible to achieve CCA-security even at lower costs and surprisingly without a lot of additional work. Concretely, in our constructions we extend the encryption and the public parameters by a single group element respectively. Additionally, we require only a universal one-way hash function and an algorithm that checks some consistency properties of the original encryption. Since we only use universal one-way hash functions we need not extend the users secret keys. Although our constructions are not generic, we show that the required properties are common and many predicate encryption schemes from bilinear groups meet our demands.

2 Background and definitions

In this section we recall some background and security definitions for constructions in use. For some probabilistic polynomial time (ppt) algorithm A we denote by $[A(x)]$ the set of all possible outputs

of A on input x, whereas $y \leftarrow A(x)$ denotes a random choice of y according to the distribution defined through the random choices of A during the execution on input x.

2.1 Predicate key encapsulation mechanism with public index

In this work we consider only predicate encryption schemes with public index, also called payload hiding predicate encryption schemes. This class of public key encryption schemes covers for example identity-based encryption, attribute-based encryption and broadcast encryption. However, we do not consider ciphertext index hiding predicate encryption schemes.

In practice, asymmetric encryption schemes are usually used to encrypt a randomly chosen key for some symmetric encryption scheme. The actual data is then encrypted using this more efficient scheme (also called data encapsulation mechanism in this context). The security requirements for the asymmetric part in this so called hybrid construction are weaker than the usual requirements for fully functional encryption schemes. This is due to the fact that only randomly chosen keys, which will be used once, must be encrypted. Key encapsulation mechanisms meet these weaker security requirements and provide exactly the functionality needed for hybrid constrictions. The formalization of this basic approach goes back to [Sho00].

It is not hard to see, that hybrid constructions for predicate encryption lead to fully functional CCA-secure schemes. The equivalent statement has been proven for identity-based encryption by [BFMLS08] and for attribute-based encryption by [GBN10], with main ideas going back to [CS03]. For predicate encryption the proof is analogous and we drop it in this version of our work. Since CCA-security is widely accepted as the right notion of security for encryption schemes used in practice, CCA-secure schemes are often presented in form of key encapsulation mechanisms (KEMs). Therefore, we provide our constructions in this form, but we note that with minor modifications they also work for fully functional encryption schemes.

Definition 1. *A* relation family $\mathcal{R}$ *is a set of relations* $\mathcal{R} = \{R_i : K_i \times I_i \mapsto \{0,1\} \mid i \in \mathbb{N}\}$*, where* $\{K_i\}$ *and* $\{I_i\}$ *are the key index spaces and the ciphertext index spaces respectively.*[1]

Definition 2. *A* predicate key encapsulation mechanism with public index *for relation family* $\mathcal{R}$ *and for symmetric key space* $\mathbb{K}$ *consists of the following ppt (in* 1^λ*) algorithms:*

- $\text{Setup}\left(1^\lambda, n\right) \to (\text{msk}, \text{pp}_n)$*: Given a security parameter* 1^λ *and* $n \in \mathbb{N}$*, the master secret key and the public parameters supporting* R_n *are generated.*
- $\text{KeyGen}\left(\text{pp}_n, \text{msk}, X\right) \to \text{sk}_X$*: Given a key index* $X \in K_n$*, a secret key for* X *is generated.*
- $\text{Encaps}\left(\text{pp}_n, ind\right) \to (k, c_{ind})$*: Given a ciphertext index* $ind \in I_n$*, an encapsulation of a key* $k \in \mathbb{K}$*, that is chosen uniformly at random, is generated.*
- $\text{Decaps}\left(\text{pp}_n, \text{sk}_X, c_{ind}\right) \to k$*: Given a secret key and an encapsulation, the algorithm outputs a key* $k \in \mathbb{K}$ *or an error symbol* $\perp \notin \mathbb{K}$*.*

Correctness*: We require, that the algorithms satisfy the following correctness property: For every (meaningful)* λ *and* $n \in \mathbb{N}$*, every* $(\text{msk}, \text{pp}_n) \in \left[\text{Setup}\left(1^\lambda, n\right)\right]$*, every* $X \in K_n$ *and* $sk_X \in [\text{KeyGen}\left(\text{pp}_n, \text{msk}, X\right)]$*, every* $ind \in I_n$ *and* $(k, c_{ind}) \in [\text{Encaps}\left(\text{pp}_n, ind\right)]$ *it holds*

$$\text{Decaps}\left(\text{pp}_n, \text{sk}_X, c_{ind}\right) = \begin{cases} k & \text{if } R_n\left(X, ind\right) = 1 \\ \perp & \text{if } R_n\left(X, ind\right) = 0. \end{cases}$$

[1] We implicitly assume the existence of efficient checks for $k \in K_i$ and $\text{ind} \in I_i$ for every $i \in \mathbb{N}$ in the definitions of KeyGen, Encaps and Decaps algorithms.

Remark 1. We assume that, given c_{ind}, ind is known. Indeed, this is not always the case even for predicate encryption schemes with public index. For example, the encapsulations in identity-based schemes do not contain the identity. In fact, we only need to require that, given some key index X and c_{ind}, we can efficiently check if $R_n(\text{ind}, \text{X}) = 1$.[2]

Next, we define the notion of public verifiability of encapsulations, or just public verifiability.

Definition 3. *A predicate key encapsulation mechanism Π with public index for symmetric key space $\mathbb{K}$ is called publicly verifiable if there exist a ppt (in 1^λ) algorithm* PubVrfy *that requires as input* pp_n *and a (possibly malformed) encapsulation c_{ind} and outputs 1 if and only if there exists $k \in \mathbb{K}$ such that $(k, c_{ind}) \in [\text{Encaps}(\text{pp}_n, ind)]$.*

2.2 Security definitions

Formally, in the security experiments against chosen-ciphertext attacks the adversary is provided with a decapsulation oracle. For conventional public key schemes the adversary just specifies an encapsulation for such an oracle, whereas in identity-based settings she additionally outputs an identity. The general settings of predicate encryption schemes are even more complex, since there is no unique secret key in question and there might even exist different secret keys labeled with the same key index. This property will be modeled in the security experiment (compare the security experiments for BE in [PPSS13] and for ABE in [BL13]).

Furthermore, there exist two different security notions for predicate encryptions (PEs), selective security and adaptive security. The first one is easy to achieve, but it does not really reflect reality. Namely, in this model the adversary has to define the target of her attack before the system is initialized. Adaptive security is the right notion of security, where the adversary has to break an existing scheme. The research on selectively secure schemes did not stop even in contexts where adaptively secure schemes are known. Novel techniques and ideas are easy to present in the selective security model and almost all adaptively secure predicate encryption schemes are constructed from schemes first presented in the selective security model.

In Section 3 we will prove that our constructions lead to CCA-secure schemes in the selective security model. Even if the original scheme is adaptively CPA-secure, our constructions do not directly lead to adaptive CCA-secure schemes. However, the adaptively CCA-secure schemes in [KG09, PPSS13] can be seen as extensions of selectively CCA-secure schemes achieved from corresponding CPA-secure schemes using our constructions.

Definition 4. *The* selective indistinguishability experiment $\text{sIND-P-KEM}^{\text{aCCA}}_{\Pi,\mathcal{A}}(\lambda, n)$ *for predicate key encapsulation mechanism Π is as follows:*

- ***Init**: Adversary $\mathcal{A}$ on input 1^λ and n outputs a ciphertext index $ind^* \in I_n$.*
- ***Setup**: The challenger $\mathcal{C}$ generates $(\text{msk}, \text{pp}_n) \leftarrow \text{Setup}(1^\lambda, n)$ and gives pp_n to $\mathcal{A}$. Furthermore, $\mathcal{C}$ initializes an empty set L.*
- ***Phase 1**: $\mathcal{A}$ has access to the following oracles:*
  - ***CoveredKeyGen**(X, i) for $X \in K_n$ and $i \in \mathbb{N}$: The challenger generates $\text{sk}_X \leftarrow \text{KeyGen}(\text{pp}_n, \text{msk}, X)$, (re)stores (i, X, sk_X) in L and returns nothing.*
  - ***Open**(i) for $i \in \mathbb{N}$: If $(i, X, \text{sk}_X) \in L$ and $R_n(X, ind^*) = 0$ the challenger returns sk_X.*
  - ***Decapsulate**(c_{ind}, i) for $i \in \mathbb{N}$: If $(i, X, \text{sk}_X) \in L$ the challenger returns $\text{Decaps}(\text{pp}_n, \text{sk}_X, c_{ind})$.*

[2] We do not require a secret key for X!

- ***Challenge**: When $\mathcal{A}$ asks for the challenge, $\mathcal{C}$ chooses $b \leftarrow \{0,1\}$, generates $k_0 \leftarrow \mathbb{K}$ and $\left(k_1, c^*_{ind^*}\right) \leftarrow \text{Encaps}\left(\text{pp}_n, ind^*\right)$, defines $k^* := k_b$ and returns $\left(k^*, c^*_{ind^*}\right)$.*
- ***Phase 2**: As Phase 1 under the following restriction: **Decapsulate**$\left(c^*_{ind^*}, i\right)$ is not allowed if $(i, X, \text{sk}_X) \in L$ and $R_n\left(X, ind^*\right) = 1$.*
- ***Guess**: $\mathcal{A}$ outputs a bit b'. The output of the experiment is 1 if $b' = b$.*

One can easily adapt this experiment to the case of chosen-plaintext attacks, where the adversary does not have access to the decapsulation oracle. In this case all the oracles from above can be replaced by a single oracle **KeyGen** (X) for $\text{X} \in K_n$ such that $R_n\left(\text{X}, \text{ind}^*\right) = 0$. In order to reply to such a query the challenger generates a new key for every query. The security definitions are as usual and hence, we present only one of the definitions explicitly:

Definition 5. *A predicate key encapsulation mechanism Π is called* selectively secure against adaptive chosen-ciphertext attacks *if for every ppt adversary $\mathcal{A}$ there exists a negligible function* negl *such that* $\Pr\left[\text{sIND-P-KEM}^{\text{aCCA}}_{\Pi,\mathcal{A}}\left(\lambda, n\right) = 1\right] \leq \frac{1}{2} + \text{negl}\left(\lambda\right)$.

For our CCA-secure constructions we will require that the original chosen-plaintext secure (CPA-secure) schemes have some additional properties. One of these properties will be, that the scheme remains CPA-secure even if the ppt adversary is given additional power in form of an oracle to some function f. Next, we define this formally.

Let f be an arbitrary function. We define the experiment $\text{sIND-P-KEM}^{\text{f-CPA}}_{\Pi,\mathcal{A}}\left(\lambda, n\right)$ as extension of $\text{sIND-P-KEM}^{\text{CPA}}_{\Pi,\mathcal{A}}\left(\lambda, n\right)$, where the adversary gets access to an oracle for f in Phase 1 and 2 of the experiment.

Definition 6. *A predicate key encapsulation mechanism Π is called* immune to function f in experiment $\text{sIND-P-KEM}^{\text{CPA}}_{\Pi,\mathcal{A}}\left(\lambda, n\right)$*, if for every ppt adversary $\mathcal{A}$ there exists a negligible function* negl *such that* $\Pr\left[\text{sIND-P-KEM}^{\text{f-CPA}}_{\Pi,\mathcal{A}}\left(\lambda, n\right) = 1\right] \leq \frac{1}{2} + \text{negl}\left(\lambda\right)$.

2.3 Bilinear groups

In this paper we use the standard terminology for bilinear groups (see e.g. [BF03]) and consider only predicate encryption schemes which are constructed from bilinear groups. The security assumptions for these groups are defined according to a group generator $\mathcal{G}$, which given a security parameter outputs the description of a (symmetric) bilinear group: $\left(p, \mathbb{G}, \mathbb{G}_T, e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T, g \in \mathbb{G}\right) \leftarrow \mathcal{G}\left(1^\lambda\right)$, where p is a prime such that $\log\left(p\right) \geq \lambda$; $\mathbb{G}$ and $\mathbb{G}_T$ are cyclic groups of order p; e is a map linear in both components; g is a generator in $\mathbb{G}$. We assume the reader to be familiar with bilinear groups.

2.4 Hash Functions

Our constructions require universal one-way hash functions (UOWHFs) as introduced in [NY89]. The notion of UOWHF was shown to be strictly weaker than collision resistance [RS04]. In fact, UOWHF can be constructed from one-way functions [Rom90], whereas for collision resistant hash functions such a construction is not known. In practice, both types of hash functions are instantiated by dedicated cryptographic hash function like SHA-2 (cf. [CS03]).

Definition 7. *(cf. [Gol04]) Let $\mathcal{UOWHF} = \left\{h_s : \{0,1\}^* \rightarrow \{0,1\}^{l(|s|)}\right\}_{s \in \{0,1\}^*}$ with $l : \mathbb{N} \rightarrow \mathbb{N}$ be a collection of efficiently computable keyed functions. $\mathcal{UOWHF}$ is called a family of universal one-way hash functions if there exists a ppt algorithm I such that for all ppt adversaries $\mathcal{A}$ the probability to win the following game is negligible in λ:*

– $\mathcal{A}$ *on input* 1^λ *outputs* x.
– $\mathcal{A}$ *is given* $s \leftarrow I(1^\lambda)$.
– $\mathcal{A}$ *outputs* x' *and wins the game if* $x' \neq x$ *but* $h_s(x') = h_s(x)$.

In our constructions we need to hash ciphertexts of predicate encryption schemes into $\mathbb{Z}_p$. Hence, for each concrete scheme we require an injective encoding of ciphertext space through bit strings. We will not explicitly mention this in our constructions. Furthermore, we write UHF $\leftarrow \mathcal{UOWHF}$ and mean with this notation, that a random key is chosen for the hash function.

3 Chosen-ciphertext secure predicate key encapsulation mechanism

In this section we present our method to convert predicate key encapsulation mechanisms (P-KEMs) selectively secure against chosen-plaintext attacks into schemes which withstand chosen-ciphertext attacks. In order to achieve this stronger notion of security, we require a hash function and extend the public parameters and the encapsulations by a single group element respectively. Hence, our constructions are very efficient from this point of view. However, we also require a kind of consistency checks for the encapsulations. The efficiency of the corresponding algorithm depends on the original scheme. In order to achieve an efficient but still widely applicable conversion technique, we explicitly looked at structural properties of predicate encryption schemes based on bilinear groups. These schemes are very similar in their basic structure and we exploit this in our constructions. Next, we formally define the properties necessary for our technique. At first, some of these properties might seem to be very specific, but we will show that many well known schemes have these properties.

3.1 Required properties

Let $\Pi = (\mathrm{Setup}, \mathrm{KeyGen}, \mathrm{Encaps}, \mathrm{Decaps})$ be a selectively CPA-secure P-KEM. Our main construction requires the following properties of Π:

1. Π is defined over prime order bilinear groups $(p, \mathbb{G}, \mathbb{G}_T, e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T, g \in \mathbb{G})$. The symmetric key space of Π is $\mathbb{G}_T$.
2. (Commitment property) The public parameters contain generators $v \in \mathbb{G}$ and $Y \in \mathbb{G}_T$; for every $\mathrm{ind} \in I_n$ and every $(k, c_{\mathrm{ind}}) \in [\mathrm{Encaps}(\mathrm{pp}_n, \mathrm{ind})]$ there exists $s \in \mathbb{Z}_p$ such that $k = Y^s$ and $c' = v^s$, where c' is contained in c_{ind}.[3] Furthermore, s must be explicitly chosen by the encapsulation algorithm.
3. (f-immunity) The public parameters contain some $\omega \in \mathbb{G}$ such that Π is immune to the partial function $f : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$,

$$f(x, y) \mapsto \begin{cases} Y^r & \text{if } \exists r : x = v^r \wedge y = w^r \\ \perp & \text{otherwise} \end{cases}$$

   in experiment $\mathrm{sIND\text{-}P\text{-}KEM}^{\mathrm{CPA}}_{\Pi,\mathcal{A}}(\lambda, n)$.
4. (Verifiability) There is a ppt (in 1^λ) algorithm Vrfy that requires as input pp_n, $\mathrm{X} \in K_n$ and a (possibly malformed) encapsulation c_{ind}. Vrfy outputs 1 if and only if for every $\mathrm{sk}_{\mathrm{X}} \in [\mathrm{KeyGen}(\mathrm{pp}, \mathrm{msk}, \mathrm{X})]$ it holds $\mathrm{Decaps}(\mathrm{pp}, \mathrm{sk}_{\mathrm{X}}, c_{\mathrm{ind}}) = Y^s$, where s is defined through $c' \in c_{\mathrm{ind}}$, $c' = v^s$.

[3] Y can be also defined implicitly through two generators in $\mathbb{G}$.

Note, that the correct form of the input values x, y for the function f can be easily checked using the pairing e: $e(x, \omega) \stackrel{?}{=} e(\upsilon, y)$. Furthermore, f is defined through $\upsilon, \omega, Y \in \mathrm{pp}_n$ and consequently, in Phase 1 and in Phase 2 of security experiment this function is well defined.

The commitment property is based on the fact, that the encapsulation of key Y^s has to provide decrypter with information about s. The element c' can be seen as commitment to s. Almost all PE schemes satisfy the commitment property or can be easily extended in order to satisfy it (see Section 4 for examples). Furthermore, publicly verifiable schemes satisfy the last property through the algorithm PubVrfy. Nevertheless, the required properties of algorithm Vrfy are weaker and for some concrete schemes such an algorithm can be implemented more efficiently (see Section 4 for examples).

The most tricky property is the f-immunity. For some schemes one can directly see, that this property is satisfied, since for an appropriate generator ω the function f will be efficiently computable from the public parameters. But in many cases this is more involved, since we have to consider the original security proof, identify an appropriate generator ω and prove that ppt adversaries can not make use of the oracle f. Some schemes even have to be extended in order to satisfy this property. See Section 4 for more details and examples.

3.2 Construction

Let $\Pi = (\mathrm{Setup}, \mathrm{KeyGen}, \mathrm{Encaps}, \mathrm{Decaps})$ be a predicate key encapsulation mechanism with public index for relation family $\mathcal{R} = \{R_i : K_i \times I_i \to \{0,1\} \mid i \in \mathbb{N}\}$ such that Π has the four properties from Section 3.1. Furthermore, let $\mathcal{UOWHF}$ be a family of universal one-way hash functions. Construct a P-KEM $\Pi' = (\mathrm{Setup}', \mathrm{KeyGen}', \mathrm{Encaps}', \mathrm{Decaps}')$ for $\mathcal{R}$ as follows:

- $\mathrm{Setup}'\left(1^\lambda, n\right)$:
  - Compute $(\mathrm{msk}, \mathrm{pp}_n) \leftarrow \mathrm{Setup}\left(1^\lambda, n\right)$.
  - Choose $h \leftarrow \mathbb{G}$ and $\mathrm{UHF} \leftarrow \mathcal{UOWHF}$ such that $\mathrm{UHF} : \mathcal{C}_\Pi \to \mathbb{Z}_p$, where $\mathcal{C}_\Pi$ is the ciphertext space of Π.
  - Output the master secret msk and the public parameters $\mathrm{pp}'_n = (\mathrm{pp}_n, h, \mathrm{UHF})$.
- $\mathrm{KeyGen}'\left(\mathrm{pp}'_n, \mathrm{msk}', \mathrm{X}\right) = \mathrm{KeyGen}\left(\mathrm{pp}_n, \mathrm{msk}, \mathrm{X}\right)$.
- $\mathrm{Encaps}'\left(\mathrm{pp}'_n, \mathrm{ind}\right)$ with $\mathrm{ind} \in I_n$. Extend the original encapsulation algorithm as follows:
  - Let (Y^s, c_{ind}) be the result of the original computation of $\mathrm{Encaps}\left(\mathrm{pp}_n, \mathrm{ind}\right)$. Compute $t := \mathrm{UHF}\left(c_{\mathrm{ind}}\right)$ and $c'' := \left(\omega^t \cdot h\right)^s$.
  - Output $(Y^s, (c_{\mathrm{ind}}, c''))$.
- $\mathrm{Decaps}'\left(\mathrm{pp}'_n, \mathrm{sk}_\mathrm{X}, (c_{\mathrm{ind}}, c'')\right)$ works as follows:
  - Check $\mathrm{Vrfy}\left(\mathrm{pp}_n, \mathrm{X}, c_{\mathrm{ind}}\right) \stackrel{?}{=} 1$.
  - Compute $t := \mathrm{UHF}\left(c_{\mathrm{ind}}\right)$ and check:

$$e\left(c', \omega^t \cdot h\right) \stackrel{?}{=} e\left(\upsilon, c''\right). \tag{1}$$

  - If both tests succeed, output $\mathrm{Decaps}\left(\mathrm{pp}_n, \mathrm{sk}_\mathrm{X}, c_{\mathrm{ind}}\right)$.

First, we show that Π' is well defined. From the commitment property and f-immunity we know that the original public parameter pp_n contain $\upsilon, \omega \in \mathbb{G}$ and $Y \in \mathbb{G}_T$. Furthermore, by the commitment property the encapsulation of key Y^s contains $c' = \upsilon^s$ and $s \in \mathbb{Z}_p$ is known to the encapsulation algorithm. Hence, c'' can be computed and all the elements in Equation 1 are well defined.

Correctness: Let (c_{ind}, c'') be a correctly generated encapsulation of key Y^s. Correctness of Π implies that $\text{Decaps}\,(\text{pp}, \text{sk}_\text{X}, c_{\text{ind}}) = Y^s$ and hence, by definition of verifiability $\text{Vrfy}\,(\text{pp}_n, \text{X}, c_{\text{ind}}) = 1$. Consequently, Decaps' does not abort in the first step. Next, $c' = \upsilon^s$ for some $s \in \mathbb{Z}_p$ by the commitment property and $c'' = \left(\omega^t \cdot h\right)^s$ by construction. Hence, Equation 1 is also satisfied. Last, as already mentioned $\text{Decaps}\,(\text{pp}, \text{sk}_\text{X}, c_{\text{ind}}) = Y^s$ and hence, the output of Decaps' is correct.

Theorem 1. *Let $\mathcal{UOWHF}$ be a family of universal one-way hash functions and Π be a predicate key encapsulation mechanism with four required properties from Subsection 3.1. If Π is selectively CPA-secure under some security assumptions, then Π' constructed from Π is selectively CCA-secure under the same security assumptions.*

Proof. We prove, that if there exists an adversary $\mathcal{A}$ with significant advantage in experiment $\text{sIND-P-KEM}^{\text{aCCA}}_{\Pi',\mathcal{A}}\,(\lambda, n)$, then there exists an algorithm $\mathcal{B}$ with significant advantage in the experiment $\text{sIND-P-KEM}^{\text{f-CPA}}_{\Pi,\mathcal{A}}\,(\lambda, n)$. Assume, that such an algorithm $\mathcal{A}$ exists; algorithm $\mathcal{B}$ is as follows:

Init-Phase: $\mathcal{B}$ on input 1^λ and $n \in \mathbb{N}$ outputs $\text{ind}^* \leftarrow \mathcal{A}\left(1^\lambda, n\right)$.

Setup-Phase: The challenger $\mathcal{C}$ sends $\mathcal{B}$ the public parameters pp_n, that include υ, ω, Y. Then, $\mathcal{B}$ asks for the challenge and gets $\left(k^*, c^*_{\text{ind}^*}\right)$, where $c^*_{\text{ind}^*}$ includes c'^*. Next, $\mathcal{B}$ chooses $\text{UHF} \leftarrow \mathcal{UOWHF}$ and $\xi \leftarrow \mathbb{Z}_p$, computes $t^* := \text{UHF}\left(c^*_{\text{ind}^*}\right)$ and sets $h := \upsilon^\xi \cdot \omega^{-t^*}$. $\mathcal{B}$ sends the public parameters $\text{pp}'_n := (\text{pp}_n, h, \text{UHF})$ to $\mathcal{A}$. Furthermore, $\mathcal{B}$ precomputes $c''^* := (c'^*)^\xi$ for the challenge. The generator h is correctly distributed due to ξ.

Challenge: When $\mathcal{A}$ asks for the challenge, $\mathcal{B}$ outputs $\left(k^*, \left(c^*_{\text{ind}^*}, c''^*\right)\right)$. It is easy to check, that the precomputed c''^* is correct for $c^*_{\text{ind}^*}$:

$$\begin{aligned} c''^* &\overset{\text{def}}{=} \left(\omega^{t^*} \cdot h\right)^{s^*} \\ &= \left(\omega^{\,t^*} \cdot \upsilon^\xi \cdot \omega^{-t^*}\right)^{s^*} \\ &= \left(c'^*\right)^\xi, \end{aligned}$$

where s^* is unknown but uniquely defined through $c'^* = \upsilon^{s^*}$. $\mathcal{B}$ is able to compute the correct value c''^* without knowledge of s^*, due to the definition of h.

Phase 1 and Phase 2: $\mathcal{B}$ initializes an empty set L and answers the queries of $\mathcal{A}$ as follows:

CoveredKeyGen (X, i) with $\text{X} \in K_n$ and $i \in \mathbb{N}$:
- If $R_n\,(\text{X}, \text{ind}^*) = 0$, $\mathcal{B}$ queries its own oracle **KeyGen** (X), receives $\text{sk}_\text{X} \leftarrow \text{KeyGen}\,(\text{pp}_n, \text{msk}, \text{X})$, (re)stores $(i, \text{X}, \text{sk}_\text{X})$ in L and returns nothing.
- If $R_n\,(\text{X}, \text{ind}^*) = 1$, $\mathcal{B}$ (re)stores $(i, \text{X}, \perp)$ in L.

Open (i) with $i \in \mathbb{N}$:
- If $(i, \text{X}, \text{sk}_\text{X}) \in L$ output sk_X. Otherwise output $\perp$.

Decapsulate $\left((c_{\text{ind}}, c'')\,, i\right)$ with $i \in \mathbb{N}$ and $c' = \upsilon^r \in c_{\text{ind}}$ for some unknown $r \in \mathbb{Z}_p$. $\mathcal{B}$ answers as follows:
1. If $(i, \text{X}, \text{sk}_\text{X})$ in L return $\text{Decaps}'\,(\text{pp}'_n, \text{sk}_\text{X}, (c_{\text{ind}}, c''))$. Output $\perp$ if there is no entry for i.
2. If $R_n\,(\text{X}, \text{ind}) \neq 1$ output $\perp$, where $(i, \text{X}, \perp) \in L$.
3. If $(c_{\text{ind}}, c'') = \left(c^*_{\text{ind}^*}, c''^*\right)$ in Phase 1, abort.
4. If $\text{Vrfy}\,(c_{\text{ind}}) = 0$ or $e\left(c', \omega^t \cdot h\right) \neq e\,(\upsilon, c'')$ output $\perp$, where $t := \text{UHF}\,(c_{\text{ind}})$.
5. If $t = t^* \wedge c_{\text{ind}} \neq c^*_{\text{ind}^*}$ abort.

6. Use the oracle for f and return $f\left(c', \left(c'' \cdot (c')^{-\xi}\right)^{(t-t^*)^{-1}}\right)$.

By construction, $\mathcal{B}$ never queries its key generation oracle for X if $R_n(\mathrm{X}, \mathrm{ind}^*) = 1$. Furthermore, the opening oracle simulation is consistent, because $\mathcal{A}$ is not allowed to open keys if $R_n(\mathrm{X}, \mathrm{ind}^*) = 1$ and if there is no entry for i in L.

$\mathcal{B}$ aborts in Step 3 only with negligible probability over the random choice of $c'^* \in c^*_{\mathrm{ind}^*}$, because in Phase 1 the view of $\mathcal{A}$ is independent of this value. In Phase 2 a query with $(c_{\mathrm{ind}}, c'') = \left(c^*_{\mathrm{ind}^*}, c''^*\right) \wedge R_n(\mathrm{X}, \mathrm{ind}) = 1$ is not allowed, whereas $R_n(\mathrm{X}, \mathrm{ind}) \neq 1$ was already caught in Step 2. $\mathcal{B}$ outputs $\perp$ in Step 4 only if one of the tests from the decapsulation algorithm is not satisfied. In Step 5, $\mathcal{B}$ aborts with negligible probability, because of the properties of a universal one way hash function.

The case $t = t^* \wedge c_{\mathrm{ind}} = c^*_{\mathrm{ind}^*}$ is caught in Step 3, since in this case holds $c'' = c''^*$ by construction and thus $(c_{\mathrm{ind}}, c'') = \left(c^*_{\mathrm{ind}^*}, c''^*\right)$. Hence, in the last step $(t - t^*)^{-1}$ is well defined. Furthermore, by the consistency check for c'' in Step 4 we have $c'' = \left(\omega^t \cdot h\right)^r$ and thus

$$\begin{aligned}
\left(c'' \cdot \left(c'\right)^{-\xi}\right)^{(t-t^*)^{-1}} &= \left(\left(\omega^t \cdot h\right)^r \cdot \left(v^r\right)^{-\xi}\right)^{(t-t^*)^{-1}} \\
&= \left(\left(\omega^t \cdot v^{\xi} \cdot \omega^{-t^*}\right)^r \cdot v^{-\xi \cdot r}\right)^{(t-t^*)^{-1}} \\
&= \left(\omega^{(t-t^*)\cdot r}\right)^{(t-t^*)^{-1}} \\
&= \omega^r .
\end{aligned}$$

Hence, the input of the query to oracle f in Step 6 is valid and the oracle returns Y^r. By the verifiability property and the fact that $\mathrm{Vrfy}(c_{\mathrm{ind}}) = 1$ after Step 4 this is the correct answer to the decapsulation query.

Guess: Output the output of $\mathcal{A}$.

The simulation provided by $\mathcal{B}$ is not perfect, since in some cases $\mathcal{B}$ aborts. But this happens only with negligible probability. Furthermore, $\mathcal{B}$ wins if $\mathcal{A}$ wins and hence, the theorem follows. □

3.3 Improvement

In this subsection we present our second construction. For many schemes this construction leads to more efficient CCA-secure schemes. The idea for this construction is to hash only those components of an encapsulation, which uniquely define the remaining elements. In combination with public verifiability this is sufficient to achieve CCA security.

Required properties: The first three properties remain unchanged. Instead of the verifiability we require public verifiability according to Definition 3. The additional property is:

5. (Hash uniqueness) There exists a deterministic polynomial time (in 1^λ) algorithm HInput that gets as input the public parameters and a correctly generated encapsulation c_{ind} and outputs some $\hat{c}_{\mathrm{ind}}$. We require that $(k_1, c_{\mathrm{ind},1}), (k_2, c_{\mathrm{ind},2}) \in [\mathrm{Encaps}(\mathrm{pp}_n, \mathrm{ind})]$ and $c_{\mathrm{ind},1} \neq c_{\mathrm{ind},2}$ imply $\mathrm{HInput}(\mathrm{pp}_n, c_{\mathrm{ind},1}) \neq \mathrm{HInput}(\mathrm{pp}_n, c_{\mathrm{ind},2})$.

Note, that we do not require that HInput outputs parts of the original encapsulation. Rather, HInput can be any function on encapsulations as long as, restricted to a ciphertext index, it is injective.

Modified construction. Modify the construction of Π' in such a way, that the encapsulation and decapsulation algorithms compute the hash value for encapsulation $c_{\mathrm{ind}} \in \mathcal{C}_{\mathrm{ind}}$ as

$$t := \mathrm{UHF}\left(\mathrm{HInput}\left(c_{\mathrm{ind}}\right)\right)$$

instead of $t := \mathrm{UHF}\left(c_{\mathrm{ind}}\right)$. Furthermore, use the algorithm PubVrfy instead of Vrfy. We call such a modified scheme Π''.

Theorem 2. *Let $\mathcal{UOWHF}$ be a family of universal one-way hash functions and Π be a predicate key encapsulation mechanism with five required properties from above. If Π is selectively CPA-secure under some security assumptions, then Π'' constructed from Π is selectively CCA-secure under the same security assumptions.*

Proof. We have to slightly modify the proof of Theorem 1. The hash value t^* in the setup phase and the hash values t during the decapsulation queries should be computed according to the new construction using HInput. We also use the algorithm PubVrfy instead of the algorithm Vrfy. Furthermore, Case 5 for decapsulation queries should be modified to:

5. If $t = t^*$ but $\mathrm{HInput}\left(c_{\mathrm{ind}}\right) \neq \mathrm{HInput}\left(c^*_{\mathrm{ind}^*}\right)$ abort.

Also in this case $\mathcal{B}$ found a collision and aborts only with negligible probability. All the computations in Step 6 work as before, but we have to consider the case $t = t^* \wedge \mathrm{HInput}\left(c_{\mathrm{ind}}\right) = \mathrm{HInput}\left(c^*_{\mathrm{ind}^*}\right)$. Also for this construction this case can not arise. Namely, due to the public verifiability c_{ind} is correctly formed and hence, by hash uniqueness $c_{\mathrm{ind}} = c^*_{\mathrm{ind}^*}$. This again implies $\left(c_{\mathrm{ind}}, c''\right) = \left(c^*_{\mathrm{ind}^*}, c''^*\right)$. □

4 Application to some known schemes

All schemes considered in this section are defined over prime order groups and thus already satisfy the first required property of our constructions. The public parameters of these schemes includes the description of the bilinear groups $(p, \mathbb{G}, \mathbb{G}_T, e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T, g \in \mathbb{G})$.[4] We briefly recall the schemes in question in their KEM versions and adapt them to our notation. We refer to the original papers for complete descriptions.

Fuzzy-IBE [SW05] and KP-ABE of [GPSW06], the large universe constructions. Both schemes differ only in the key generation algorithms. Hence, we can consider these schemes together. The public parameters are defined as $\mathrm{pp}_n = (g_1, g_2, t_1, \ldots, t_{n+1})$, where all the elements are from group $\mathbb{G}$. The public parameters implicitly define $Y = e\left(g_1, g_2\right)$. The elements $\{t_i\}_{i=1}^{n+1}$ define a publicly computable function $T : \mathbb{Z}_p \to \mathbb{G}$. The encapsulation of key Y^s for identity $\mathrm{ind} \subset \mathbb{Z}_p$, $|\mathrm{ind}| \leq n$ has the form

$$c_{\mathrm{ind}} = \left(\mathrm{ind}, c' = g^s, \{c_i = T\left(i\right)^s\}_{i \in \mathrm{ind}}\right).$$

- The schemes satisfy the commitment property for $\upsilon = g$ and Y as above.
- The scheme has f-immunity for $\omega = g_1$, because given $\upsilon^r = g^r$ and $\omega^r = g_1^r$ the adversary can compute herself $e\left(\omega^r, g_2\right) = e\left(g_1, g_2\right)^r = Y^r$ from ω^r only. Hence, the oracle f is useless for $\mathcal{A}$.

[4] Note, that g is explicitly a part of the public parameters.

– The Vrfy algorithm for these schemes checks for every $i \in \mathrm{X} \cap \mathrm{ind}$:

$$e\left(c', T\left(i\right)\right) \stackrel{?}{=} e\left(g, c_i\right).$$

These checks ensure that all the elements c_i which are relevant for the decapsulation using the secret key with key index X are correctly formed. Hence, the decapsulation algorithm given any $\mathrm{sk}_X \in [\mathrm{KeyGen}\left(\mathrm{pp}_n, \mathrm{msk}, \mathrm{X}\right)]$ outputs Y^s, even if some of the remaining elements are malformed. Note, that this verification algorithm can be performed more efficiently in a randomized manner, combining all the checks in a single one (cf. for example with [BL13]):

- Choose for every $i \in \mathrm{X} \cap \mathrm{ind}$ an element $r_i \leftarrow \mathbb{Z}_p$ and then check:

$$e\left(c', \prod_{i \in \mathrm{X} \cap \mathrm{ind}} T\left(i\right)^{r_i}\right) \stackrel{?}{=} e\left(g, \prod_{i \in \mathrm{X} \cap \mathrm{ind}} c_i^{r_i}\right).$$

Malformed elements pass this randomized test only with negligible probability over the random choice of the elements r_i.

Note, that ind and c' uniquely define all the other elements of the encapsulation. Hence, we can use our second construction and hash only these elements. In this case we have to use the PubVrfy algorithm, which checks all the elements c_i.

The authors of [BL13] presented a CCA-secure extension of key-policy ABE (KP-ABE) of [GPSW06]. Their scheme corresponds to the scheme that we get from our second construction with a single exception. Namely, they even proved that it is sufficient to hash the number of attributes instead of the ciphertext index itself. This is not covered by our second construction, since the inputs of the hash functions are not unique in their construction. The scheme of [BL13] can be seen as an extension of the scheme achieved from our second construction using further specific properties of the scheme.

HIBE of [BBG05] with constant size ciphertext. The structure of this scheme is very similar to that of the large universe construction of [GPSW06]. Note however, that the identity is not contained in the encapsulation. But given an identity, one can easily check if the encapsulation is created for this identity (see also Remark 1). The public parameters are defined as $\mathrm{pp}_n = (g_1, g_2, g_3, h_1, \ldots, h_n)$, where all the elements are from $\mathbb{G}$. The public parameters implicitly define $Y = e\left(g_1, g_2\right)$. The elements $\{h_i\}_{i=1}^n$ and g_3 define a publicly computable function $F : \mathbb{Z}_p^{\leq n} \to \mathbb{G}$. The original encapsulation of key Y^s for identity $\mathrm{ind} \in \mathbb{Z}_p^{\leq n}$ has the form

$$\left(c' = g^s, c = F\left(\mathrm{ind}\right)^s\right),$$

– The scheme satisfies the commitment property for $\upsilon = g$ and Y as above.
– The scheme has f-immunity for $\omega = g_1$. Given $\upsilon^r = g^r$ and $\omega^r = g_1^r$ the adversary can compute herself $e\left(\omega^r, g_2\right) = e\left(g_1, g_2\right)^r = Y^r$ from ω^r only. Hence, the oracle f is useless for $\mathcal{A}$.
– For this scheme the algorithm Vrfy performs a single check:

$$e\left(c', F\left(\mathrm{ind}\right)\right) \stackrel{?}{=} e\left(g, c\right).$$

As long as the identity is not a part of the encapsulation this scheme is not publicly verifiable and hence, our second construction can not be applied. If we extend the encapsulation by identity as suggested in [BSW11], we have to hash the identity too. In this case for our second construction we do not have to hash the element c.

IBE of [Wat05]. The public parameters for this scheme are $\mathrm{pp}_n = (g_1, g_2, t', t_1, \ldots, t_n)$, where all the elements are from group $\mathbb{G}$. The public parameters implicitly define $Y = e(g_1, g_2)$. The elements $\{t_i\}_{i=1}^n$ and t' define a publicly computable function $T : S \to \mathbb{G}$, where S is the power set of $[n]$. The encapsulation of key Y^s for identity $\mathrm{ind} \subseteq [n]$ has the form

$$\left(c' = g^s, c = T(\mathrm{ind})^s\right)$$

The verification of the required properties is as for HIBE from [BBG05].

The original scheme is proven to be *adaptively* CPA-secure. Furthermore, Galindo and Kiltz [KG09] presented an extension, which is also *adaptively* CCA-secure. Their construction is exactly the same as the resulting construction from our first scheme with a single exception that they hash only c'. This element is independent of ciphertext index and hence, the hash value for the challenge can be computed in the setup phase without knowledge of ind^*. Furthermore, we note that the second element of the encapsulation does not have to be hashed due to the security properties of this concrete scheme. Namely, given c' and c, the adversary is not able to compute the corresponding value c for another identity. This shows that in some cases our resulting constructions can be enhanced to preserve adaptive security.

Broadcast encryption of [BGW05]. The public parameters in this scheme supporting n users are defined as $\mathrm{pp}_n = (v, g_1, \ldots, g_n, g_{n+2}, \ldots, g_{2n})$, where all the elements are from $\mathbb{G}$; the element Y is implicitly set to $e(g_1, g_n)$. The elements $\{g_i\}_{i=1}^n$ and v define a publicly computable function $T : S \to \mathbb{G}$, where S is the power set of $[n]$. The encapsulation of key Y^s for $\mathrm{ind} \subseteq [n]$ is:

$$\left(c' = g^s, c = T(\mathrm{ind})^s\right)$$

The verification of the required properties is similar to the last two constructions.

In the original work the CPA-secure scheme is enhanced to achieve CCA-security applying the ideas of [BCHK06]. Later, a direct CCA-secure variant has been proposed in [PPSS13]. Their technique is similar to ours, but they did not add the element c'' to the encapsulation, but included the hash value $t = \mathrm{UHF}(c')$ directly into the value c. In fact, for their scheme c is defined as $\left(g_1^t \cdot T(\mathrm{ind})\right)^s$. Hence, in their scheme the ciphertext space is not extended. Furthermore, in [PPSS13], this scheme was also proven adaptively secure but under some extended security assumptions. We did not consider such extensions in our work.

KP-ABE of [GPSW06], small universe constructions: The original public parameters are defined as $\mathrm{pp}_n = (t_1, \ldots, t_n, Y)$, where for every attribute i in the universe $\mathcal{U}$ there exists an element $t_i \in \mathbb{G}$ in the public parameters. The original encapsulation of key Y^s for identity $\mathrm{ind} \subseteq \mathcal{U}$ has the form:

$$\left(\mathrm{ind}, \{c_i = t_i^s\}_{i \in \mathrm{ind}}\right).$$

One can easily see, that these schemes do not satisfy our required properties. But we can extend the original scheme such that our transformation can be applied:

- Add a generator $\omega \leftarrow \mathbb{G}$ to the public parameters.
- Add the element $c' = g^s$ to the encapsulation.

The commitment property is satisfied directly for $v = g$ and Y from above. Now, we draft the proof that this extended scheme is CPA-secure and satisfies f-immunity for ω from above. Indeed, we can extend the original security proof as follows:

– Set $\omega := g^a$. This is possible, since originally Y is simulated as $e\left(g^a, g^b\right)$ and thus Y and ω stay independent due to the choice of b.
– Set the additional element of the challenge c'^* to g^c. This is consistent with the definition of the original challenge.
– Given an oracle query for υ^r and ω^r the simulator can compute $f\left(\upsilon^r, \omega^r\right) := e\left(\omega^r, g^b\right) = e\left(g^a, g^b\right)^r = Y^r$.
– The verification algorithm Vrfy given an encapsulation $\left(\mathrm{ind}, c', \{c_i\}_{i\in\mathrm{ind}}\right)$ and $\mathrm{X} \subseteq \mathcal{U}$ checks $\forall i \in \mathrm{X} \cap \mathrm{ind} : e\left(c', t_i\right) \stackrel{?}{=} e\left(\upsilon, c_i\right)$.

  The algorithm PubVrfy simply checks all the elements c_i.

KP-ABE for non-monotonic access structures of [OSW07] The proof of required properties is very similar to the proof for the large universe KP-ABE of [GPSW06]. Only the Vrfy algorithm has to be extended.

The public parameters of the scheme supporting encryption with exactly n attributes are defined as $\mathrm{pp}_n = \left(g_1, \{v_i\}_{i=0}^n, \{t_i\}_{i=0}^n\right)$, where all the elements are from group $\mathbb{G}$. The public parameters implicitly define $Y = e\left(g_1, v_0\right)$. The elements $\{t_i\}_{i=0}^n$ and $\{v_i\}_{i=0}^n$ define two publicly computable functions $T : \mathbb{Z}_p \to \mathbb{G}$ and $V : \mathbb{Z}_p \to \mathbb{G}$. The encapsulation of key Y^s for identity $\mathrm{ind} \subset \mathbb{Z}_p$, $|\mathrm{ind}| = n$ has the form

$$c_{\mathrm{ind}} = \left(\mathrm{ind}, c' = g^s, \{c_{i,1} = T\left(i\right)^s\}_{i\in\mathrm{ind}}, \{c_{i,2} = V\left(i\right)^s\}_{i\in\mathrm{ind}}\right).$$

All the properties excepting verifiability are as for the KP-ABE of [GPSW06]. Analogously to the values $c_{i,1}$ the verification algorithm has to check the values $c_{i,2}$:

$$e\left(c', V\left(i\right)\right) \stackrel{?}{=} e\left(g, c_{i,2}\right).$$

Hence, using the randomized test one can check all the element as follows:

– Choose for every $i \in \mathrm{X} \cap \mathrm{ind}$ elements $r_{i,1}, r_{i,2} \leftarrow \mathbb{Z}_p$ and then check:

$$e\left(c', \prod_{i\in\mathrm{X}\cap\mathrm{ind}} T\left(i\right)^{r_{i,1}} \cdot V\left(i\right)^{r_{i,2}}\right) \stackrel{?}{=} e\left(g, \prod_{i\in\mathrm{X}\cap\mathrm{ind}} c_{i,1}^{r_{i,1}} \cdot c_{i,2}^{r_{i,2}}\right).$$

Malformed elements pass this randomized test only with negligible probability over the random choice of the elements $r_{i,1}, r_{i,2}$.

HIBE of [BB04] based on DBDH assumption. The public parameters supporting n level are defined as $\mathrm{pp}_n = \left(g_1, g_2, h_1, \ldots, h_n\right)$, where all the elements are from group $\mathbb{G}$. The public parameters implicitly define $Y = e\left(g_1, g_2\right)$. The generator g_1 and the elements $\{h_i\}_{i=1}^n$ define publicly computable functions $\{F_i : \mathbb{Z}_p \to \mathbb{G}\}_{i=1}^n$. The encapsulation of key Y^s for identity $\mathrm{ind} \in \mathbb{Z}_p^l$ has the form

$$c_{\mathrm{ind}} = \left(c' = g^s, \{c_i = F_i\left(\mathrm{ind}_i\right)^s\}_{i=1}^n\right),$$

– The schemes satisfy the commitment property for $\upsilon = g$ and Y as above.
– The scheme has f-immunity for $\omega = g_1$. Given $\upsilon^r = g^r$ and $\omega^r = g_1^r$ the adversary can compute herself $e\left(\omega^r, g_2\right) = e\left(g_1, g_2\right)^r = Y^r$ from ω^r only. Hence, the oracle f is useless for $\mathcal{A}$.
– The Vrfy algorithm for this scheme checks for every $i \in \{1, \ldots, n\}$:

$$e\left(c', F_i\left(\mathrm{ind}_i\right)\right) \stackrel{?}{=} e\left(g, c_i\right).$$

  Also for this scheme an efficient randomized test is possible.

IBE of [BB04] based on BDH Inversion assumption. The public parameters are defined as $\mathrm{pp}_n = (g_1, g_2)$, where $g_1, g_2 \in \mathbb{G}$. The public parameters implicitly define $Y = e(g, g)$. The generators g and g_1 define publicly computable function $F : \mathbb{Z}_p \to \mathbb{G}$. The encapsulation of key Y^s for identity $\mathrm{ind} \in \mathbb{Z}_p$ has the form

$$c_{\mathrm{ind}} = \left(c' = g_2^s, c = F(\mathrm{ind})^s\right),$$

- The schemes satisfy the commitment property for $\upsilon = g_2$ and Y as above.
- The scheme has f-immunity for $\omega = g$. Given $\upsilon^r = g_2^r$ and $\omega^r = g^r$ the adversary can easily compute $e(\omega^r, g) = e(g, g)^r = Y^r$ from ω^r only. Hence, the oracle f is useless for $\mathcal{A}$.
- The Vrfy algorithm for this scheme checks:

$$e\left(c', F(\mathrm{ind})\right) \stackrel{?}{=} e(g_2, c).$$

Non-monotonic KP-ABE of [ALdP11] with short ciphertext. The public parameters supporting encryption with less than n attributes are defined as $\mathrm{pp}_n = (\{h_i\}_{i=1}^n, \{u_i\}_{i=0}^n, Y)$, where $Y \in \mathbb{G}_T$ and all the elements are from group $\mathbb{G}$. The elements $\{h_i\}_{i=1}^n$ and $\{u_i\}_{i=0}^n$ define publicly computable functions $H : \mathbb{Z}_p^{<n} \to \mathbb{G}$ and $U : \mathbb{Z}_p^{<n} \to \mathbb{G}$ respectively. The encapsulation of key Y^s for identity $\mathrm{ind} \in \mathbb{Z}_p^l$ with $l < n$ has the form

$$c_{\mathrm{ind}} = \left(\mathrm{ind}, c' = g^s, c_1 = U(\mathrm{ind})^s, c_2 = H(\mathrm{ind})^s\right),$$

- The schemes satisfy the commitment property for $\upsilon = g$.
- The scheme has f-immunity for $\omega = u_1$. To see this we have to look at the proof for the original CPA-secure scheme (see the full version of [ALdP11]). There, the simulator explicitly defines $Y := e(z_1, z_n)^{\delta_0}$ for some generators $z_1, z_n \in \mathbb{G}$ and $\delta_0 \in \mathbb{Z}_p$. Furthermore, the generator in question u_1 is explicitly defined as $u_1 := z_1 \cdot g^{\theta_1}$ for some $\theta_1 \in \mathbb{Z}_p$. Hence, given an oracle query for $\omega^r = u_1^r$ and $\upsilon^r = g^r$, the simulator can compute

$$\frac{e(\omega^r, z_n)^{\delta_0}}{e(\upsilon^r, z_n)^{\delta_0 \cdot \theta_1}} = \frac{e\left(\left(z_1 \cdot g^{\theta_1}\right)^r, z_n\right)^{\delta_0}}{e(g^{r \cdot \theta_1}, z_n)^{\delta_0}} = Y^r$$

- The Vrfy algorithm for this scheme checks:

$$e\left(c', U(\mathrm{ind})\right) \stackrel{?}{=} e(g, c_1).$$

and

$$e\left(c', H(\mathrm{ind})\right) \stackrel{?}{=} e(g, c_2).$$

Fuzzy-IBE of [SW05], small universe: Originally the scheme is proven CPA-secure under the decision modified BDH assumption, that differ a bit from the standard assumptions. One can show, that the scheme is CPA-secure under the DBDH assumption. Indeed the scheme is a special case of KP-ABE encryption scheme of Gayal et al.[GPSW06], that is proven secure under DBDH assumption. In order to apply our CCA construction, we have to extend it similar to the KP-ABE. See this construction for details.

CP-ABE schemes of [Wat11] Waters presented in his work several constructions from different security assumptions. The structural properties of all these schemes are very similar, hence we show the applicability of our constructions only for the most efficient scheme.

The public parameters supporting attribute universe $\mathcal{U} \subset \mathbb{Z}_p$ are defined as $\text{pp} = \left(g_1, \{h_i\}_{i \in \mathcal{U}}, Y\right)$, where $Y \in \mathbb{G}_T$ and all the elements are from group $\mathbb{G}$. The encapsulation of key Y^s for access structure $\mathbb{A}$ realized through monotone span program $\text{ind} = (M, \rho)$, where $M \in \mathbb{Z}_p^{l \times d}$ and $\rho : \{1, \ldots, l\} \mapsto \mathcal{U}$, has the form

$$c_{\text{ind}} = \left(\text{ind}, c' = g^s, \left\{c_{i,1} = g_1^{\lambda_i} \cdot h_{\rho(i)}^{-r_i}, c_{i,2} = g^{r_i}\right\}_{i=1}^{l}\right)$$

with randomly chosen $r_i \in \mathbb{Z}_p$ and vector of shares $\boldsymbol{\lambda} = M \cdot (s, v_2, \ldots, v_d)^T$ with randomly chosen $v_i \in \mathbb{Z}_p$.

- The scheme satisfies the commitment property for $\upsilon = g$.
- The scheme has f-immunity for $\omega = g_1$. To see this, we have to look at original proof. There, the simulator sets explicitly
$$Y := (g_1, h) \cdot e\,(g, g)^{\alpha'}$$
for some $h \in \mathbb{G}$ and $\alpha' \leftarrow \mathbb{Z}_p$. Hence, given $\upsilon^r = g^r$ and $\omega^r = g_1^r$ the challenger can compute
$$(\omega^r, h) \cdot e\,(\upsilon^r, g)^{\alpha'} = Y^r$$
and thus reply all the queries of adversary.
- For this scheme the algorithm Vrfy is more involved. Construction of PubVrfy was presented in [BL13], see this work for more details. The shares λ_i can be reconstructed in the exponent as follows:
$$\begin{aligned} e\,(c_{i,1}, g) \cdot e\left(c_{i,2}, h_{\rho(i)}\right) &= e\left(g_1^{\lambda_i} \cdot h_{\rho(i)}^{-r_i}, g\right) \cdot e\left(g^{r_i}, h_{\rho(i)}\right) \\ &= e\,(g_1, g)^{\lambda_i}. \end{aligned}$$
Then, the method of [BL13] can be applied to verify these shares. Vrfy has to check only the elements associated with attributes from key index X.

CP-ABE scheme of [RW13] The public parameters supporting attribute universe $\mathcal{U} = \mathbb{Z}_p$ are defined as $\text{pp} = (g_1, g_2, g_3, g_4, Y)$, where $Y \in \mathbb{G}_T$ and all the elements are from group $\mathbb{G}$. The generators g_1 and g_2 define a function $H : \mathcal{U} \mapsto \mathbb{G}$. The encapsulation of key Y^s for access structure $\mathbb{A}$ realized through monotone span program $\text{ind} = (M, \rho)$, where $M \in \mathbb{Z}_p^{l \times d}$ and $\rho : \{1, \ldots, l\} \mapsto \mathcal{U}$, has the form

$$c_{\text{ind}} = \left(\text{ind}, c' = g^s, \left\{c_{i,1} = g_3^{\lambda_i} \cdot g_4^{r_i}, c_{i,2} = H\left(\rho\left(i\right)\right)^{-r_i}, c_{i,3} = g^{r_i}\right\}_{i=1}^{l}\right)$$

with randomly chosen $r_i \in \mathbb{Z}_p$ and vector of shares $\boldsymbol{\lambda} = M \cdot (s, v_2, \ldots, v_d)^T$ with randomly chosen $v_i \in \mathbb{Z}_p$.

- The scheme satisfies the commitment property for $\upsilon = g$.

– The scheme has f-immunity for $\omega = g_3$. To see this, we have to look at original proof. There, the simulator explicitly sets

$$Y := (g_3, h) \cdot e(g, g)^{\alpha'}$$

for some $h \in \mathbb{G}$ and $\alpha' \leftarrow \mathbb{Z}_p$. Hence, given $\upsilon^r = g^r$ and $\omega^r = g_3^r$ the challenger can compute

$$(\omega^r, h) \cdot e(\upsilon^r, g)^{\alpha'} = Y^r$$

and thus reply all the queries of adversary.

– The Vrfy algorithm checks at first for all elements in question:

$$e(c_{i,2}, g)^{-1} \stackrel{?}{=} e(c_{i,3}, H(\rho(i))).$$

Then, it reconstructs the shares λ_i in the exponent as follows:

$$\begin{aligned} e(c_{i,1}, g) \cdot e(c_{i,3}, g_4)^{-1} &= e\left(g_3^{\lambda_i} \cdot g_4^{r_i}, g\right) \cdot e(g^{r_i}, g_4)^{-1} \\ &= e(g_3, g)^{\lambda_i} \end{aligned}$$

Then, the method of [BL13] can be applied to verify these shares. Vrfy has to check only the elements associated with attributes from key index X.

KP-ABE scheme of [RW13] The public parameters supporting attribute universe $\mathcal{U} = \mathbb{Z}_p$ are defined as $\mathrm{pp} = (g_1, g_2, g_3, Y)$, where $Y \in \mathbb{G}_T$ and all the elements are from group $\mathbb{G}$. The generators g_1 and g_2 define a function $H : \mathcal{U} \mapsto \mathbb{G}$. The encapsulation of key Y^s for attribute set $\mathrm{ind} \subset \mathcal{U}$ has the form

$$c_{\mathrm{ind}} = \left(\mathrm{ind}, c' = g^s, \left\{c_{i,1} = g^{r_i}, c_{i,2} = H(i)^{r_i} \cdot g_3^{-s}\right\}_{i \in \mathrm{ind}}\right)$$

– The scheme satisfies the commitment property for $\upsilon = g$.

– The scheme has f-immunity for $\omega = g_3$. To see this, we have to look at original proof. There, the simulator sets explicitly

$$Y := (g_3, h)$$

for some $h \in \mathbb{G}$. Hence, given $\upsilon^r = g^r$ and $\omega^r = g_3^r$ the challenger can compute

$$(\omega^r, h) = Y^r$$

and thus reply all the queries of adversary.

– The Vrfy algorithm checks for all elements in question:

$$\begin{aligned} e(c_{i,2}, g) \cdot e(c_{i,1}, H(i))^{-1} &= e\left(g_3^{-s} \cdot H(i)^{r_i}, g\right) \cdot e(g^{r_i}, H(i))^{-1} \\ &= e\left(g_3^{-s}, g\right) \\ &\stackrel{?}{=} e\left(g_3, c'\right)^{-1}. \end{aligned}$$

Further schemes: We did not present the complete list of schemes satisfying our required properties. Rather these schemes show that our required properties are widely common.

Conclusion: As we have shown, our transformation is not generic but can be applied for many predicate encryption schemes constructed from bilinear groups. Compared to the generic constructions of [YAHK11] for ABE and [YAS+12] for PE, our constructions use UOWHF and do not add overhead in form of one-time signatures to the encapsulations. Hence, we provide simple and efficient technique to achieve CCA-security for predicate encryption schemes in selective security model. Moreover, for some schemes the resulting constructions can be extended as the examples in [KG09, PPSS13, BL13] show.

References

[ALdP11] Nuttapong Attrapadung, Benoît Libert, and Elie de Panafieu. Expressive key-policy attribute-based encryption with constant-size ciphertexts. In *Public Key Cryptography*, volume 6571 of *Lecture Notes in Computer Science*, pages 90–108. Springer, 2011.

[BB04] Dan Boneh and Xavier Boyen. Efficient selective-id secure identity-based encryption without random oracles. In *EUROCRYPT*, number 3027 in Lecture Notes in Computer Science, pages 223–238. Springer, 2004.

[BBG05] Dan Boneh, Xavier Boyen, and Eu-Jin Goh. Hierarchical identity based encryption with constant size ciphertext. In *EUROCRYPT*, volume 3494 of *Lecture Notes in Computer Science*, pages 440–456. Springer, 2005.

[BCHK06] Dan Boneh, Ran Canetti, Shai Halevi, and Jonathan Katz. Chosen-ciphertext security from identity-based encryption. *SIAM Journal on Computing*, 36(5):1301–1328, 2006.

[BF03] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the Weil pairing. *SIAM Journal on Computing*, 32(3):586–615, 2003.

[BFMLS08] Kamel Bentahar, Pooya Farshim, John Malone-Lee, and Nigel P. Smart. Generic constructions of identity-based and certificateless KEMs. *Journal of Cryptology*, 21:178–199, 2008.

[BGW05] Dan Boneh, Craig Gentry, and Brent Waters. Collusion resistant broadcast encryption with short ciphertexts and private keys. In *CRYPTO*, volume 3621 of *Lecture Notes in Computer Science*, pages 258–275. Springer, 2005.

[BL13] Johannes Blömer and Gennadij Liske. Direct chosen-ciphertext secure attribute-based key encapsulations without random oracles. *IACR Cryptology ePrint Archive*, 2013:646, 2013.

[BMW05] Xavier Boyen, Qixiang Mei, and Brent Waters. Direct chosen ciphertext security from identity-based techniques. In *ACM Conference on Computer and Communications Security*, pages 320–329. ACM, 2005.

[BSW11] Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: Definitions and challenges. In *TCC*, volume 6597 of *Lecture Notes in Computer Science*, pages 253–273. Springer, 2011.

[BW06] Dan Boneh and Brent Waters. A fully collusion resistant broadcast, trace, and revoke system. In *ACM Conference on Computer and Communications Security*, pages 211–220. ACM, 2006.

[CS03] Ronald Cramer and Victor Shoup. Design and analysis of practical public-key encryption schemes secure against adaptive chosen-ciphertext attack. *SIAM Journal on Computing*, 33:167–226, 2003.

[FO13] Eiichiro Fujisaki and Tatsuaki Okamoto. Secure integration of asymmetric and symmetric encryption schemes. *Journal of Cryptology*, 26:80–101, 2013.

[GBN10] M. Choudary Gorantla, Colin Boyd, and Juan Manuel González Nieto. Attribute-based authenticated key exchange. In *ACISP*, volume 6168 of *Lecture Notes in Computer Science*, pages 300–317. Springer, 2010.

[GJPS08] Vipul Goyal, Abhishek Jain, Omkant Pandey, and Amit Sahai. Bounded ciphertext policy attribute based encryption. In *ICALP (2)*, volume 5126 of *Lecture Notes in Computer Science*, pages 579–591. Springer, 2008.

[Gol04] Oded Goldreich. *The Foundations of Cryptography - Volume 2, Basic Applications*. Cambridge University Press, 2004.

[GPSW06] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *ACM Conference on Computer and Communications Security*, pages 89–98. ACM, 2006.

[HW13] Susan Hohenberger and Brent Waters. Attribute-based encryption with fast decryption. In *Public Key Cryptography*, volume 7778 of *Lecture Notes in Computer Science*, pages 162–179. Springer, 2013.

[KG09] Eike Kiltz and David Galindo. Direct chosen-ciphertext secure identity-based key encapsulation without random oracles. *Theoretical Computer Science*, 410(47-49):5093–5111, 2009.

[Kil06a] Eike Kiltz. Chosen-ciphertext security from tag-based encryption. In *TCC*, volume 3876 of *Lecture Notes in Computer Science*, pages 581–600. Springer, 2006.

[Kil06b] Eike Kiltz. On the limitations of the spread of an ibe-to-pke transformation. In *Public Key Cryptography*, volume 3958 of *Lecture Notes in Computer Science*, pages 274–289. Springer, 2006.

[NY89] Moni Naor and Moti Yung. Universal one-way hash functions and their cryptographic applications. In *21st Annual ACM Symposium on Theory of Computing*, pages 33–43. ACM, 1989.

[OSW07] Rafail Ostrovsky, Amit Sahai, and Brent Waters. Attribute-based encryption with non-monotonic access structures. In *ACM Conference on Computer and Communications Security*, pages 195–203. ACM, 2007.

[PPSS13] Duong Hieu Phan, David Pointcheval, Siamak Fayyaz Shahandashti, and Mario Strefler. Adaptive cca broadcast encryption with constant-size secret keys and ciphertexts. *International Journal of Information Security*, 12(4):251–265, 2013.

[Rom90] John Rompel. One-way functions are necessary and sufficient for secure signatures. In *STOC*, pages 387–394. ACM, 1990.

[RS04] Phillip Rogaway and Thomas Shrimpton. Cryptographic hash-function basics: Definitions, implications, and separations for preimage resistance, second-preimage resistance, and collision resistance. In *FSE*, volume 3017 of *Lecture Notes in Computer Science*, pages 371–388. Springer, 2004.

[RW13] Yannis Rouselakis and Brent Waters. Practical constructions and new proof methods for large universe attribute-based encryption. In *ACM Conference on Computer and Communications Security*, pages 463–474. ACM, 2013.

[Sho00] Victor Shoup. Using hash functions as a hedge against chosen ciphertext attack. In Bart Preneel, editor, *EUROCRYPT*, volume 1807 of *Lecture Notes in Computer Science*, pages 275–288. Springer, 2000.

[SW05] Amit Sahai and Brent Waters. Fuzzy identity-based encryption. In *EUROCRYPT*, volume 3494 of *Lecture Notes in Computer Science*, pages 457–473. Springer, 2005.

[Wat05] Brent Waters. Efficient identity-based encryption without random oracles. In *EUROCRYPT*, pages 114–127, 2005.

[Wat11] Brent Waters. Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization. In *Public Key Cryptography*, volume 6571 of *Lecture Notes in Computer Science*, pages 53–70. Springer, 2011.

[YAHK11] Shota Yamada, Nuttapong Attrapadung, Goichiro Hanaoka, and Noboru Kunihiro. Generic constructions for chosen-ciphertext secure attribute based encryption. In *Public Key Cryptography*, volume 6571 of *Lecture Notes in Computer Science*, pages 71–89. Springer, 2011.

[YAS+12] Shota Yamada, Nuttapong Attrapadung, Bagus Santoso, Jacob C. N. Schuldt, Goichiro Hanaoka, and Noboru Kunihiro. Verifiable predicate encryption and applications to cca security and anonymous predicate authentication. In *Public Key Cryptography*, volume 7293 of *Lecture Notes in Computer Science*, pages 243–261. Springer, 2012.