Using Predicate Extension for Predicate Encryption to Generically Obtain Chosen-Ciphertext Security and Signatures

Marloes Venema[1] and Leon Botros[2]

[1] University of Wuppertal, Wuppertal, Germany

[2] Radboud University, Nijmegen, The Netherlands

Abstract. Predicate encryption (PE) is a type of public-key encryption that captures many useful primitives such as attribute-based encryption (ABE). Although much progress has been made to generically achieve security against chosen-plaintext attacks (CPA) efficiently, in practice, we also require security against chosen-ciphertext attacks (CCA). Because achieving CCA-security on a case-by-case basis is a complicated task, several generic conversion methods have been proposed, which typically target different subclasses of PE such as ciphertext-policy ABE. As is common, such conversion methods may sacrifice some efficiency. Notably, for ciphertext-policy ABE, all proposed generic transformations incur a significant decryption overhead. Furthermore, depending on the setting in which PE is used, we may also want to require that messages are signed. To do this, predicate signature schemes can be used. However, such schemes provide a strong notion of privacy for the signer, which may be stronger than necessary for some practical settings at the cost of efficiency.

In this work, we propose the notion of predicate extension, which transforms the predicate used in a PE scheme to include one additional attribute, in both the keys and the ciphertexts. Using predicate extension, we can generically obtain CCA-security and signatures from a CPA-secure PE scheme. For the CCA-security transform, we observe that predicate extension implies a two-step approach to achieving CCA-security. This insight broadens the applicability of existing transforms for specific subclasses of PE to cover all PE. We also propose a new transform that incurs slightly less overhead than existing transforms. Furthermore, we show that predicate extension allows us to create a new type of signatures, which we call PE-based signatures. PE-based signatures are weaker than typical predicate signatures in the sense that they do not provide privacy for the signer. Nevertheless, such signatures may be more suitable for some practical settings owing to their efficiency or reduced interactivity. Lastly, to show that predicate extensions may facilitate a more efficient way to achieve CCA-security generically than existing methods, we propose a novel predicate-extension transformation for a large class of pairing-based PE, covered by the pair and predicate encodings frameworks. In particular, this yields the most efficient generic CCA-conversion for ciphertext-policy ABE.

Keywords: predicate encryption · chosen-ciphertext security · signatures · generic transformation · identity-based encryption · attribute-based encryption

E-mail: venema@uni-wuppertal.de (Marloes Venema), l.botros@cs.ru.nl (Leon Botros)

1 Introduction

Predicate encryption (PE) [KSW08] is a paradigm that generalizes multiple powerful cryptographic primitives[1], such as identity-based encryption (IBE) [Sha84, BF01] and attribute-based encryption (ABE) [SW05]. In contrast to traditional public-key encryption, PE allows for the fine-grained access control on data [GPSW06a, BSW07]. In PE, the ciphertexts and secret keys are associated with "entities" x and y, respectively, for which a predicate P determines the ability of the secret key to decrypt the ciphertext. In particular, a secret key for y can decrypt a ciphertext for x if and only if $P(x, y) = 1$ (i.e., "the predicate is true"). For example, in ciphertext-policy ABE [BSW07], x constitutes an access policy and y a set of attributes, and $P(x, y) = 1$ holds if the set of attributes satisfies the policy[2].

Over the years, many works have systematized and improved the techniques to generically and efficiently achieve security for pairing-based PE against chosen-plaintext attacks (CPA) [Wee14, Att14, CGW15, Att16, AC16, AC17, ABGW17, Ven23]. By formalizing the notions of pair encodings [Att14] and predicate encodings [Wee14], a PE scheme can be abstracted to analyze only "what happens in the exponent". This simplifies the effort of proving security to information-theoretic and computational notions of security for vectors of polynomials. These frameworks are incredibly powerful: many PE schemes can be captured in them, and hence, these schemes are provably CPA-secure.

While these frameworks support CPA-security, in practice, it is often recommended or required that the scheme also provides security against chosen-ciphertext attacks (CCA) [NY90, Sho98]. To this end, many works have proposed CCA-secure PE schemes, e.g., [BF01, KG06, Gen06, KV08, TKN21]. Moreover, to achieve CCA-security generically, any of the proposed transformations can be used, e.g.,

- using non-interactive zero-knowledge (NIZK) proofs of well-formedness [BFM88];
- Fujisaki-Okamoto (FO) [FO99, HHK17];
- Canetti-Halevi-Katz (CHK) [CHK04];
- Boneh-Katz (BK) [BK05];
- Abe et al. (ACIK) [ACIK10];
- Yamada et al. (YA(SS)HK) [YAHK11, YAS+12], which consists of two transformations: one for delegatable ABE and one for verifiable ABE;
- Blömer-Liske (BL) [BL16];
- Koppula-Waters (KW) [KW19].

However, each of these generic transformations has a drawback. First, the transformation may be restricted to e.g., hierarchical IBE (HIBE) [CHK04, BK05, ACIK10] or ABE [YAHK11, YAS+12]. Second, the FO-transform, the NIZK approach, and the transformations for verifiable schemes [YAHK11, YAS+12, BL16] incur an additional cost during decryption that is linear in the sizes of x and y, which is a significant cost for many ABE or inner-product encryption [KSW08] schemes. Alternatively, these additional costs

[1] Our definition of predicate encryption is in line with [AC17], which is more general than in some other works [KW19, Att19]. In those works, predicate encryption requires x to be hidden. We will use the notion of attribute hiding (e.g., in Appendix C) to refer to this additional property.

[2] In this example, x could be called a predicate. However, in its dual variant, key-policy ABE, the keys are associated with policies and the ciphertexts with sets of attributes, and thus, the predicate is then associated with the keys, and not the ciphertexts. Similarly, dual-policy ABE [AI09] may specify policies for the keys and ciphertexts. Hence, we refer to both x and y as predicates (or attributes) throughout this work.

may be linear in the security parameter[3], such as in KW [KW19] and the transformations for delegatable CP-ABE [YAHK11]. Notably, for CP-ABE, no CCA-transformations yield a small and constant overhead.

Furthermore, we may require for some practical settings that not only the recipients of some data are authenticated using predicates, but also the sender. Currently, this can be done by using predicate signatures [AHY15], which also covers identity-based [Sha84, BF01] and attribute-based signatures [MPR11, OT11]. Predicate signatures associate signing keys with y and signatures with x, such that the predicate $P(x, y)$ is true. Upon verifying a signature, the verifier learns only that the signer satisfies x, but not which specific y the signer possesses. This is beneficial for the signer in terms of privacy, but the property may also be stronger than needed in some practical settings. It may even be too strong for some settings, as it gives the signer a form of plausible deniability. For example, if x is a set of identities and y is one specific identity, then the signer can deny having signed a certain message and claim that another signer with an identity in the set x has signed it. To avoid such situations, it could be desirable to sign messages directly with a specific y that the signer possesses, and verify signatures for any x such that $P(x, y)$ holds. Furthermore, not requiring such privacy properties may be more efficient, e.g., because the description of y is much shorter than x. Another example of why this may be more efficient is that the signer can sign a message with y without knowing beforehand which x the verifier requires the signer to satisfy. They can therefore already sign messages before any communication has transpired before the first message.

1.1 Our contributions

In this work, we focus on generically achieving CCA-security and signatures from secure PE by introducing the notion of *predicate extension*. Specifically, predicate extension allows us to extend the predicate P of a PE scheme by adding extra attributes x' and y' to x and y (associated with the ciphertexts and secret keys, respectively). The new predicate then holds for inputs (x, x') and (y, y') if both $P(x, y)$ and $x' = y'$ hold. (Alternatively, $y' = *$, in which case, any x' satisfies it.) Via the extended-predicate variant of a scheme, we achieve CCA-security and signatures generically with several novel constructions. Notably, our constructions aim to be as efficient as possible, building on previous works that are shown to be more efficient than others.

CCA-security. We strive to generically achieve CCA-security as efficiently as possible, by splitting any such transformations for CCA-security in two explicit steps. In the first step, the predicate of the scheme is extended using predicate extension. In the second step, the predicate-extended scheme is used to achieve CCA-security for the original scheme. Although several existing transformations take these steps implicitly, explicitly considering them as two steps may lead to more efficient (yet generic) constructions than previous methods allowed. To illustrate that, we propose several novel transformations that perform these two steps efficiently. Furthermore, our approach may provide practitioners with more flexibility in choosing a suitable transformation based on the available implementations of primitives, because it is possible to mix and match approaches for the two steps. Another advantage of explicitly separating the process of extending the predicate from the CCA-security proof is that it may simplify the process of achieving CCA-security in primitives that cannot be captured with existing techniques yet. For example, for post-quantum solutions, it may be possible to design more efficient predicate extensions without having to do an entire CCA-security proof for the new design.

[3]Typically, the security parameter is fixed, e.g., equal to 128. Nevertheless, the additional costs are large.

PE-based signatures. We also put forth the new notion of PE-based signatures, and provide a generic approach to constructing such signatures from PE schemes via predicate extensions. In contrast to existing predicate signatures, our definition of PE-based signatures associates both the signing key and signature with y, and verification can be done for any x such that the predicate $P(x, y)$ is true[4]. Although this definition is weaker than predicate signatures in the sense that the specific y used by the signer is not private, it may be sufficiently strong or even desirable for practice. Furthermore, because our definition of PE-based signatures is weaker, we can directly construct them from PE schemes via predicate extensions. To the best of our knowledge, no (semi-)generic constructions exist yet for predicate signatures, which considerably simplify the design of such schemes.

A new predicate-extension transformation. To optimize the efficiency of concrete constructions, we also give a new predicate-extension transformation in the pair and predicate encodings frameworks. In this way, we can support efficient predicate extension generically for a large class of pairing-based PE. Our transformation yields more efficient constructions than the implicitly-defined predicate-extension methods proposed by previous generic conversion methods. In particular, rather than extending the predicate explicitly by applying a transformation to a PE scheme, these methods exploit the structures of the predicates themselves to embed the predicate extension. For example, for (hierarchical) IBE [CHK04, BK05, ACIK10], this predicate extension is embedded in the (additional "layer" of the) identity, and for delegatable ciphertext-policy ABE (CP-ABE), the bit-representation of the predicate extension is encoded as an AND-policy (and is taken in conjunction with the original policy). Because these transformations exploit the specific structures of the predicates, they are therefore only applicable to those predicates. Furthermore, especially in the case of CP-ABE, this adds a considerable efficiency overhead, as the bit size of a predicate extension is typically 128 or 256 bits.

Mix-and-matching steps and comparison. As part of our contributions, we have surveyed various CCA-security conversion methods and studied if and how they extend the predicates, and how they prove CCA-security using the extended-predicate variant of the scheme. For some of the transformations, e.g., CHK and BK, the second step of the proof is so generic that it can be applied to any predicate-extended scheme, while the first step only applies to a smaller class of PE (e.g., because they exploit the structure of the predicates). Hence, the second step can be applied to any predicate-extended PE, effectively widening the class of applicability for the transform. Table 1 lists those transformations that implicitly use a two-step approach and their applicability. For the predicate-extension techniques, we also list whether they yield a small overhead or not. We consider the overhead incurred by the transform small, if it does not depend on the predicate size or otherwise large numbers that significantly impact any of the costs incurred by the key generation, encryption and decryption algorithms. Roughly, if we analyze the costs of the CPA-secure version of the scheme per attribute in the predicate, then the overhead of the CCA-transform should not be (much) larger than the costs of the CPA-secure version for one attribute in the predicate, regardless of the predicate size. Lastly, note that the most notable distinction between the second-step transformations is in the generality and in the primitives used in the generic construction. Although symmetric primitives such as encapsulation, MAC, PRG and RPC hashes are more desirable than OTS, the distinction between the BK approach and ours is not as obvious. Compared to BK, our transformation is less generic, but it relies on fewer primitives. In practice, this may yield less complex and smaller code that may be easier to maintain.

[4]Note that, in the particular case of identity-based signatures, our definition is equivalent.

Table 1: Comparison of the properties of the several CCA-transformations and our new transformations. For our CCA-transformations, we also consider alternative pathways based on the existing transformations to perform the two steps.

Step	Variant	Primitives	Class	Requirements	Small overhead
Step 1	CHK/BK	-	(H)IBE	-	✓
	YA(SS)HK	-	ABE	delegatable ABE	✗
	Ours	-	All PE	pair/predicate encodings	✓
Step 2	CHK	OTS	All PE	-	
	BK	encapsulation, MAC, PRG	All PE	-	
	ACIK	RPC	(H)IBE	partitioned KEM	
	Ours	RPC	All PE	decomposable PE	

Note: RPC = random-prefix collision-resistant hash function,
OTS = one-time signature, PRG = pseudo-random generator,
MAC = message authentication code

1.2 Technical details

We discuss the technical details of our work on a high level. We first explain how existing transformations for CCA-security work and how our work generalizes these methods. As part of the generalization, we explicitly define the notion of predicate extension. We then show that predicate extensions can also be used to generically construct PE-based signatures in a similar way as we can generically construct regular signatures from identity-based encryption [BF01].

Warm-up: existing CCA-security transformations. Apart from the NIZK, FO and KW transformations, all aforementioned generic transformations for CCA-security exploit the structure of the predicate to efficiently achieve CCA-security. Roughly, they all follow a similar approach: during encryption, the encrypting user commits to some value, which is then embedded in the predicate in addition to the original predicate. Depending on the technique, the value to which is commited is either generated independently of the ciphertext [CHK04, BK05, YAHK11] or by applying a hash with a specific property to the ciphertext [ACIK10, BL16]. Although the latter requires that the ciphertext is of a specific structure (which many pairing-based schemes satisfy), it relies on fewer primitives and yields less storage overhead on the ciphertext.

Our CCA-transformation. On a high level, our approach to achieving CCA-security consists of two steps (which is shown in an overview in Figure 1). First, we transform a CPA-secure PE $\Gamma_{\text{PE,IND-CPA},P}$ for predicate P into a CPA-secure PE $\Gamma_{\text{PE,IND-CPA},P'}$ for extended predicate P'. For this step, we propose novel generic constructions in the pair and predicate encodings frameworks. (We also show that our predicate-encoding transformation preserves the attribute-hiding property in [CGW15].) As a result, many pairing-based PE schemes can be transformed using this construction. Second, we transform any CPA-secure PE $\Gamma_{\text{PE,IND-CPA},P'}$ for extended predicate P' into a CCA-secure PE $\Gamma_{\text{PE,IND-CCA},P}$ for the original predicate P. This step can be done by using similar approaches as CHK and BK. We also give a new transformation based on the ACIK-approach, which is more efficient than CHK and BK. While ACIK only applies to IBE, our new transformation applies to any PE scheme for which the ciphertexts are "decomposable" (which is a similar notion to that of partitioned in [ACIK10]).

Step one: securely extending the predicate. We first extend the predicate P to some predicate $P' = \text{PredEx}[P]$. The idea behind this is similar to the approach for hierarchical IBE [CHK04, BK05, BCHK07] and delegatable KP-ABE by Yamada et al. (YAHK) [YAHK11], and is later also applied using wildcards by Tomida et al. [TKN21]. Roughly,

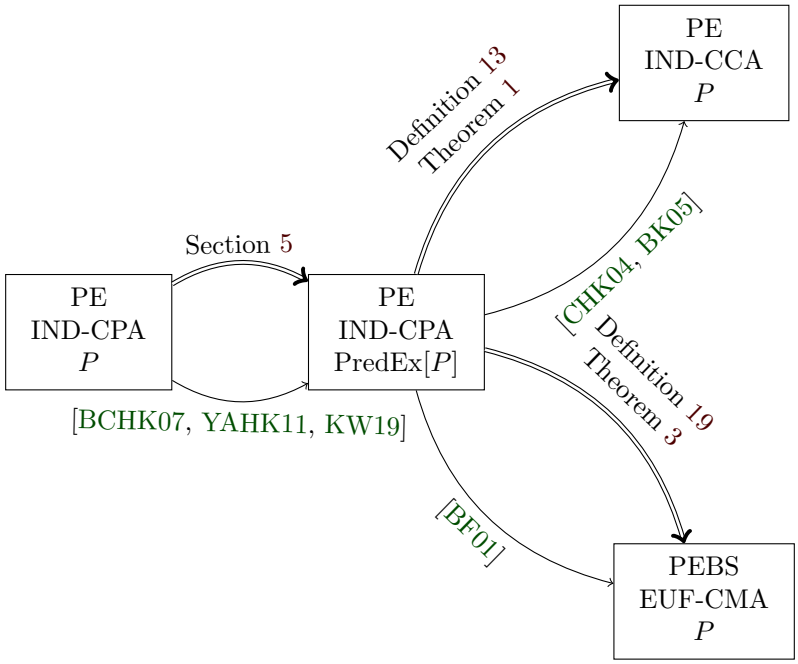

Figure 1: A high-level overview of the transformations and our associated definitions and theorems that prove security of the given transformations. The double-edged arrows indicate that we give a novel provably secure generic transformation in this work, while the normal-edged arrows provide transformations that have been given in other works.

the secret key predicate y is extended to $y \wedge y'$, where y' is either an attribute or a wildcard $*$, and the ciphertext predicate x is extended to (x, x'), where x' is an attribute. The predicate is satisfied if $P(x, y)$ and either $y' = *$ or $y' = x'$ holds. We provide a new transformation in the pair and predicate encodings framework that extends the predicate in this way. The computational overhead incurred by our transformation is a low constant, and unlike YAHK, we do not require the PE scheme to be a delegatable KP-ABE for the transformation to work. Because we generically transform *any* PE into a scheme with this specific extended predicate, we can also efficiently support CCA-security in e.g., CP-ABE. Roughly, we take an AND-composition over the original PE and an "all-or-one-identity" IBE, by using the ideas from Ambrona et al. [ABS17] and Attrapadung [Att19]. For the "all-or-one-identity" IBE, we use the first scheme of Kiltz and Vahlis [KV08] as inspiration, which is essentially implied by a composition of the Boneh-Boyen (BB) IBE [BB04] with a wildcard variant of the same scheme.

Step two: achieving CCA-security. We first consider on a high level what the CCA-transformation looks like. Let $\Gamma = (\mathrm{Setup}, \mathrm{KeyGen}, \mathrm{Encaps}, \mathrm{Decaps})$ be a predicate key-encapsulation scheme (possibly derived from a PE) for the extended predicate, such that that ciphertext is of the form

$$\mathrm{Encaps}(\mathrm{MPK}, (x, x')) = (\mathrm{K}, \mathrm{CT}_1, \mathrm{CT}_{2,(x,x')}),$$

where MPK is the master public key generated in the Setup, K is the encapsulated key to be used to symmetrically encrypt, CT_1 is some randomized part of the ciphertext that is independent of extended predicate (x, x'), and $\mathrm{CT}_{2,(x,x')}$ denotes the rest of the ciphertext. Note that the ciphertext $(\mathrm{CT}_1, \mathrm{CT}_{2,(x,x')})$ can be decapsulated with a key obtained via KeyGen for any predicate (y, y') such that $P'((x, x'), (y, y')) = 1$. Following

the approach by Kiltz and Vahlis [KV08] and Abe et al. (ACIK) [ACIK10], we first split the key-encapsulation algorithm in two parts, and then introduce an authenticated encryption scheme $\mathrm{SE} = (\mathrm{Enc}_\mathrm{K}, \mathrm{Dec}_\mathrm{K})$ and a random-prefix collision-resistant hash function RPC (which takes as input a random prefix k and another input to be hashed), i.e.,

$$\mathrm{Encaps}(\mathrm{MPK}, (x, x')) = (\underbrace{\mathrm{K}, \mathrm{CT}_1}_{\mathrm{Encaps}_1}, \underbrace{\mathrm{CT}_{2,(x,x')}}_{\mathrm{Encaps}_2}).$$

Then, we obtain the CCA-transformed encryption as follows

$$\mathrm{Encrypt}'(\mathrm{MPK}, x, M) = (\mathrm{CT}_\mathrm{sym} = \mathrm{Enc}_\mathrm{K}(M \| \mathrm{CT}_{2,(x,x')}), \mathrm{CT}_1, \mathrm{CT}_{2,(x,x')}, k),$$

where $(\mathrm{K}, \mathrm{CT}_1) \leftarrow \mathrm{Encaps}_1(\mathrm{MPK})$, $k \in_R \{0,1\}^\lambda$, $x' \leftarrow \mathrm{RPC}(k, \mathrm{CT}_1)$, and then $\mathrm{CT}_{2,(x,x')} \leftarrow \mathrm{Encaps}_2(\mathrm{MPK}, (x, x'))$.

Proving CCA-security. We prove CCA-security of the proposed generic construction similarly as other transformations [CHK04, BK05, KV08, ACIK10, YAHK11]. Specifically, the decryption queries are answered as follows. Suppose that $\mathrm{CT}_x = (\mathrm{CT}_\mathrm{sym}, \mathrm{CT}_1, \mathrm{CT}_{2,(x,x')})$ is some ciphertext and y is some predicate such that $P(x, y) = 1$, queried by the attacker. Then, the challenger can generate a secret key for $(y, y' = x')$, and decrypt the ciphertext. The challenger rejects a decryption query if it is similar to the challenge ciphertext $\mathrm{CT}^*_{x^*} = (\mathrm{CT}^*_\mathrm{sym}, \mathrm{CT}^*_1, \mathrm{CT}_{2,(x^*,x'^*)})$, i.e., if $\mathrm{CT}_1 \neq \mathrm{CT}^*_1$ and $x' = x'^*$, or if $\mathrm{K} = \mathrm{K}^*$ and $\mathrm{CT}_\mathrm{sym} \neq \mathrm{CT}^*_\mathrm{sym}$ or $\mathrm{CT}_{2,(x,x')} \neq \mathrm{CT}^*_{2,(x^*,x'^*)}$. Intuitively, the probability that a valid ciphertext is rejected—i.e., the probability that a valid ciphertext satisfies any of these conditions—is negligible due to the random-prefix collision resistance of the hash RPC and the authenticity of the symmetric encryption scheme $\mathrm{SE} = (\mathrm{Enc}_\mathrm{K}, \mathrm{Dec}_\mathrm{K})$.

Alternative pathways to achieving CCA-security. Although the proposed transformations for the two steps are applicable to large classes of existing PE schemes, they do not apply to *all* PE schemes. For example, post-quantum schemes [AFV11, ABV+12, Boy13, GVW13] are not covered by our predicate-extension transformation, and not all schemes may be decomposable and therefore qualify for our second-step transformation. To make our second step more generic, one could also use the BK-approach [BK05], which does not require the extended-predicate scheme to have ciphertexts with a certain structure[5]. However, it does provide more storage overhead and relies on more primitives (i.e., two independent hash functions, a message authentication code and a pseudo-random generator). The latter may be undesirable in practice, e.g., because it increases the code size, yields more complex code (that is more difficult to maintain) and is more complicated to optimize. In this regard, our second-step transformation could provide an effective solution, as it requires only one hash function. Regardless, because the second step *can* be done entirely generically (i.e., via the BK approach), the effort of achieving CCA-security is reduced to finding an efficient predicate extension, instead of performing a full-fledged CCA-security conversion.

Step two: PE-based signatures. Via predicate extension, we can also achieve PE-based signatures (PEBS). In particular, we transform any CPA-secure PE $\Gamma_{\mathrm{PE,IND\text{-}CPA},P'}$ for extended predicate P' into a PE-based signature scheme $\Gamma_{\mathrm{PEBS,CCA},P}$ for the original predicate P (see the overview in Figure 1). We prove the resulting PEBS to be existentially unforgeable under chosen-message attacks (EUF-CMA). The approach can be seen as a generalization of the BF approach to convert an IBE scheme to a signature scheme [BF01].

[5]The security proof of the generalized variant of the BK-transformation is analogous to that of the BK-transformation itself.

In the BF approach, the signer signs a message by embedding it in the identity (using a collision-resistant hash function) and generating a key for that identity. A signature can be verified by first encrypting a random message under the identity that embeds the message (using a hash) and then decrypting the resulting ciphertext, which should yield the same random message.

Generalizing the BF approach to sign with PE keys. We give a high-level explanation of our generalization of the BF approach. In our generic construction, the message is embedded in the predicate extension of the secret keys in a similar way, i.e., (y, y'). Then, a random message is encrypted under (x, x') (where x' is the hashed message), and the resulting ciphertext is decrypted to verify that the key works and therefore constitutes a valid signature. Compared to the BF approach, the main challenge that we have to overcome is that we do not want the users to generate secret keys for the entire key "attribute" (y, y') (where y' corresponds to the hash of the message). Instead, we want the key-generation authority to generate a secret key for a user with "attribute" y, and that the user is then able to delegate the key to the attribute (y, y') when signing a message. To do this, we use a similar approach as in our CCA-security transformation. We generate keys for $(y, *)$, where $*$ denotes a wildcard. However, unlike in our CCA-transformation, we do not want the holder of the key to decrypt for (y, y'). Instead, we want the key holder (i.e., the signer) to delegate the key from $(y, *)$ to (y, y'), so that the verifier can use it to decrypt a randomly generated ciphertext. To facilitate this, we define a delegation function for PE with predicate extension that allows a key for $(y, *)$ to be delegated to a key for (y, y'). We show that our predicate-extension transformation for pair and predicate encodings supports such a delegation function.

Proving EUF-CMA security. We prove that the signature scheme is EUF-CMA-secure if the PE scheme is CPA-secure. To do this, we use a similar approach as in the security proof for our generic CCA-secure construction. In the security reduction for the signature scheme, the EUF-CMA attacker can request signing keys for y, which correspond to key queries for $(y, *)$ to the CPA attacker. The EUF-CMA attacker can also request signatures on messages for y, which correspond to key queries for (y, y'), where y' is the embedded message. Ultimately, the EUF-CMA attacker outputs a forgery on message M^* and "attribute" y^*. This forgery is a secret key of the PE scheme for $(y^*, (y')^*)$, where $(y')^*$ is the hashed message. Because we consider existential unforgeability (and not strong unforgeability), M^* cannot be queried before. Additionally, the hash is collision resistant, so this is the first occurrence of any key with predicate extension $(y')^*$. So, any (x, x') with $x' = (y')^*$ and $P(x, y) = 0$ for all y that are used in the signing-key queries can now be queried by the CPA attacker, as well as any two arbitrary messages. The challenge ciphertext can then be decrypted using the secret key for $(y^*, (y')^*)$.

1.3 Organization

This paper is structured as follows. We first provide some notations and definitions in Section 2. Then, in Section 3, we give the generic transformations from any CPA-secure PE $\Gamma_{\text{PE,IND-CPA},P'}$ for extended predicate P' into a CCA-secure PE $\Gamma_{\text{PE,IND-CCA},P}$ for original predicate P, i.e., step 2. After this, we show how we can use predicate extensions to generically construct PE-based signatures. In Section 4, we show how to construct PE-based signatures from predicate-extended PE. In Section 5, we propose novel generic predicate-extension transformations that transform any CPA-secure PE $\Gamma_{\text{PE,IND-CPA},P}$ for predicate P into a CPA-secure PE $\Gamma_{\text{PE,IND-CPA},P'}$ for extended predicate P', i.e., step 1. We first give the more general steps of the transformation and then the less generic step, mainly due to the "level of genericness". Finally, we compare the performance of our

CCA-transformation in Section 6, and conclude the paper in Sections 7 and 8 by discussing future directions.

2 Preliminaries

2.1 Notation

We use λ to denote the security parameter. We denote a negligible function parametrized by λ by $\mathrm{negl}(\lambda)$. If an element is chosen uniformly at random from a finite set S, then we denote this as $x \in_R S$. For integers $a < b$, we denote $[a, b] = \{a, a+1, ..., b-1, b\}$, $[b] = [1, b]$ and $\overline{[b]} = [0, b]$. We use boldfaced variables $\mathbf{A}$ and $\mathbf{v}$ for matrices and vectors, respectively. We use $a \| b$ to indicate that two strings a and b are concatenated. We denote $a : \mathbf{A}$ to substitute variable a by a matrix or vector $\mathbf{A}$. We define $\mathbf{1}_{i,j} \in \mathbb{Z}_p^{d_1 \times d_2}$ as the matrix with 1 in the i-th row and j-th column, and 0 everywhere else, and similarly $\mathbf{1}_i$ and $\overline{\mathbf{1}}_i^\mathsf{T}$ as the row and column vectors with 1 in the i-th entry and 0 everywhere else.

2.2 Pairings (or bilinear maps)

We define a pairing to be an efficiently computable map e on three groups $\mathbb{G}, \mathbb{H}$ and $\mathbb{G}_T$ of prime order p, so that $e \colon \mathbb{G} \times \mathbb{H} \to \mathbb{G}_T$, with generators $g \in \mathbb{G}, h \in \mathbb{H}$ is such that for all $a, b \in \mathbb{Z}_p$, it holds that $e(g^a, h^b) = e(g, h)^{ab}$ (bilinearity), and for $g^a \neq 1_\mathbb{G}, h^b \neq 1_\mathbb{H}$, it holds that $e(g^a, h^b) \neq 1_{\mathbb{G}_T}$, where $1_{\mathbb{G}'}$ denotes the unique identity element of the associated group $\mathbb{G}'$ (non-degeneracy). We refer to $\mathbb{G}$ and $\mathbb{H}$ as the two source groups, and $\mathbb{G}_T$ as the target group.

2.3 Predicate encryption

Predicate family. A predicate family [Att14] is a set $P = \{P_\kappa\}_{\kappa \in \mathbb{N}^c}$ for some constant c, where $P_\kappa \colon \mathcal{X}_\kappa \times \mathcal{Y}_\kappa \to \{0, 1\}$ is a map over the ciphertext attribute space $\mathcal{X}_\kappa$ and the key attribute space $\mathcal{Y}_\kappa$ such that, for $x \in \mathcal{X}_\kappa, y \in \mathcal{Y}_\kappa$, $P(x, y) = 1$ if and only if the predicate evaluates to true for x and y. For κ, it holds that $\kappa = (p, \mathrm{par})$, where p is a natural number and par denote the rest of the entries.

Definition 1 (Predicate encryption (PE) [AC17])**.** A predicate encryption scheme for a predicate family $P = \{P_\kappa\}_{\kappa \in \mathbb{N}^c}$ over a message space $\mathcal{M} = \{M_\lambda\}_{\lambda \in \mathbb{N}}$ consists of four algorithms:

- $\mathrm{Setup}(\lambda, \mathrm{par})$: On input the security parameter λ and parameters par, this probabilistic algorithm generates the domain parameters, the master public key MPK and the master secret key MSK. In addition, κ is set to $\kappa = (p, \mathrm{par})$, where p denotes a natural number.

- $\mathrm{KeyGen}(\mathrm{MSK}, y)$: On input the master secret key MSK and some $y \in \mathcal{Y}_\kappa$, this probabilistic algorithm generates a secret key SK_y.

- $\mathrm{Encrypt}(\mathrm{MPK}, x, M)$: On input the master public key MPK, some $x \in \mathcal{X}_\kappa$ and message M, this probabilistic algorithm generates a ciphertext CT_x.

- $\mathrm{Decrypt}(\mathrm{MPK}, \mathrm{SK}_y, \mathrm{CT}_x)$: On input the master public key MPK, the secret key SK_y, and the ciphertext CT_x, if $P_\kappa(x, y) = 1$, then it returns M. Otherwise, it returns an error message $\perp$.

Correctness. For all par, $M \in \mathcal{M}_\lambda$, $x \in \mathcal{X}_\kappa$, and $y \in \mathcal{Y}_\kappa$ such that $P_\kappa(x, y) = 1$,

$$\Pr[(\mathrm{MPK}, \mathrm{MSK}) \leftarrow \mathrm{Setup}(\lambda, \mathrm{par}); \\ \mathrm{Decrypt}(\mathrm{MPK}, \mathrm{KeyGen}(\mathrm{MSK}, y), \mathrm{Encrypt}(\mathrm{MPK}, x, M)) \neq M] \leq \mathrm{negl}(\lambda).$$

Key-encapsulation mechanism (KEM). In the key-encapsulation variant (Appendix A), which we call predicate KEM (P-KEM), we replace Encrypt by Encaps and Decrypt by Decaps, where Encaps also outputs a symmetric key, and Decaps outputs a symmetric key instead of a plaintext message.

2.4 Full security against chosen-plaintext attacks

Definition 2 (Full security against chosen-plaintext attacks (CPA) [AC17])**.** We define the security game IND-CPA(λ, par) between challenger and attacker as follows:

- **Setup phase:** The challenger runs Setup(λ) to obtain MPK and MSK, and sends the master public key MPK to the attacker.
- **First query phase:** The attacker queries secret keys for $y \in \mathcal{Y}_\kappa$, and obtains $\mathrm{SK}_y \leftarrow \mathrm{KeyGen}(\mathrm{MSK}, y)$ in response.
- **Challenge phase:** The attacker specifies some $x^* \in \mathcal{X}_\kappa$ such that for all y in the first key query phase, we have $P_\kappa(x^*, y) = 0$, and generates two messages M_0 and M_1 of equal length in $\mathcal{M}_\lambda$, and sends these to the challenger. The challenger flips a coin, i.e., $\beta \in_R \{0, 1\}$, encrypts M_β under x^*, i.e., $\mathrm{CT}_{x^*} \leftarrow \mathrm{Encrypt}(\mathrm{MPK}, x^*, M_\beta)$, and sends the resulting ciphertext CT_{x^*} to the attacker.
- **Second query phase:** This phase is identical to the first query phase, with the additional restriction that the attacker can only query $y \in \mathcal{Y}_\kappa$ such that $P_\kappa(x^*, y) = 0$.
- **Decision phase:** The attacker outputs a guess β' for β.

The advantage of the attacker is defined as $\mathsf{Adv}_{\mathrm{PE,IND\text{-}CPA}} = |\Pr[\beta' = \beta] - \frac{1}{2}|$. A scheme is fully secure if all polynomial-time attackers have at most a negligible advantage in this security game, i.e., $\mathsf{Adv}_{\mathrm{PE,IND\text{-}CPA}} \leq \mathrm{negl}(\lambda)$.

In the selective security model, the attacker commits to the predicate $x^* \in \mathcal{X}_\kappa$ before the Setup phase.

2.5 Full security against chosen-ciphertext attacks

Definition 3 (Full security against chosen-ciphertext attacks (CCA))**.** We define the security game IND-CCA(λ, par) between challenger and attacker as follows:

- **Setup phase:** The challenger runs Setup(λ) to obtain MPK and MSK, and sends the master public key MPK to the attacker.
- **First query phase:** The attacker can make two types of queries:
  - **Key query:** the attacker queries secret keys for $y \in \mathcal{Y}_\kappa$, and obtains $\mathrm{SK}_y \leftarrow \mathrm{KeyGen}(\mathrm{MSK}, y)$ in response.
  - **Decryption query:** the attacker sends a ciphertext CT_x for $x \in \mathcal{X}_\kappa$ and $y \in \mathcal{Y}_\kappa$, with $P_\kappa(x, y) = 1$, to the challenger, who returns the message $M \leftarrow \mathrm{Decrypt}(\mathrm{MPK}, \mathrm{SK}_y, \mathrm{CT}_x)$, where $\mathrm{SK}_y \leftarrow \mathrm{KeyGen}(\mathrm{MSK}, y)$.

- **Challenge phase:** The attacker specifies some $x^* \in \mathcal{X}_\kappa$ such that for all y in the first key query phase, we have $P_\kappa(x^*, y) = 0$, and generates two messages M_0 and M_1 of equal length in $\mathcal{M}_\lambda$, and sends these to the challenger. The challenger flips a coin, i.e., $\beta \in_R \{0,1\}$, encrypts M_β under x^*, i.e., $\mathrm{CT}^*_{x^*} \leftarrow \mathrm{Encrypt}(\mathrm{MPK}, x^*, M_\beta)$, and sends the resulting ciphertext $\mathrm{CT}^*_{x^*}$ to the attacker.
- **Second query phase:** This phase is identical to the first query phase, with the additional restriction that the attacker can only query keys for $y \in \mathcal{Y}_\kappa$ such that $P_\kappa(x^*, y) = 0$, and it cannot make a decryption query for CT^*_{x*}.
- **Decision phase:** The attacker outputs a guess β' for β.

The advantage of the attacker is defined as $\mathsf{Adv}_{\mathrm{PE,IND\text{-}CCA}} = |\Pr[\beta' = \beta] - \frac{1}{2}|$. A scheme is fully secure if all polynomial-time attackers have at most a negligible advantage in this security game, i.e., $\mathsf{Adv}_{\mathrm{PE,IND\text{-}CCA}} \leq \mathrm{negl}(\lambda)$.

2.6 Authenticated symmetric encryption

Definition 4 (Symmetric encryption)**.** Let λ be the security parameter. A symmetric encryption scheme $\mathrm{SE} = (\mathrm{Enc_K}, \mathrm{Dec_K})$, with symmetric key $\mathrm{K} \in \mathcal{K}(\lambda)$, where $\mathcal{K}(\lambda)$ is some key space of size λ, is defined by

- $\mathrm{Enc_K}(M)$: On input message $M \in \{0,1\}^*$, encryption returns a ciphertext $\mathrm{CT_{sym}}$.
- $\mathrm{Dec_K}(\mathrm{CT_{sym}})$: On input ciphertext $\mathrm{CT_{sym}}$, decryption returns a message M or an error message $\perp$.

The scheme is correct if for all keys $\mathrm{K} \in \mathcal{K}(\lambda)$ and all messages $M \in \{0,1\}^*$, we have $\mathrm{Dec_K}(\mathrm{Enc_K}(M)) = M$.

For symmetric encryption, we use the same security notions as in [KV08], i.e., ciphertext indistinguishability and ciphertext authenticity.

Definition 5 (Ciphertext indistinguishability of symmetric encryption)**.** Let λ be a security parameter and let $\mathrm{SE} = (\mathrm{Enc_K}, \mathrm{Dec_K})$ be an (authenticated) symmetric encryption scheme. Consider the following game between challenger $\mathcal{C}$ and attacker $\mathcal{A}$. The challenger first picks a key $\mathrm{K} \in \mathcal{K}(\lambda)$. Then, the attacker specifies two messages M_0, M_1 and gives these to the challenger, who flips a coin $\beta \in_R \{0,1\}$ and returns $\mathrm{CT_{sym}} \leftarrow \mathrm{Enc_K}(M_\beta)$ to the attacker. The attacker $\mathcal{A}$ outputs a guess β' for β. Then, $\mathrm{SE} = (\mathrm{Enc_K}, \mathrm{Dec_K})$ has indistinguishable ciphertexts if for all polynomial-time attackers $\mathcal{A}$ in the game above holds:

$$\mathsf{Adv}_{\mathrm{SE,CIND}} = \left| \Pr[\beta' = \beta] - \frac{1}{2} \right| \leq \mathrm{negl}(\lambda).$$

In this work, we will often assume that $\mathcal{K}(\lambda)$ is the target group $\mathbb{G}_T$. Because most encryption schemes take a key that is a bit string of λ or 2λ bits as input, we use a secure key derivation function $\mathrm{KDF}\colon \mathcal{K}(\lambda) \to \{0,1\}^\lambda$ (or $\{0,1\}^{2\lambda}$) to map the target group elements to strings [CS03].

Definition 6 (Ciphertext authenticity of authenticated encryption)**.** Let λ be a security parameter and let $\mathrm{SE} = (\mathrm{Enc_K}, \mathrm{Dec_K})$ be an (authenticated) symmetric encryption scheme. Consider the following game between challenger $\mathcal{C}$ and attacker $\mathcal{A}$. The challenger first picks a key $\mathrm{K} \in \mathcal{K}(\lambda)$. Then, the attacker specifies one message M and gives it to the challenger, who returns $\mathrm{CT_{sym}} \leftarrow \mathrm{Enc_K}(M)$ to the attacker. The attacker outputs a ciphertext $\mathrm{CT'_{sym}}$. Then, the encryption scheme has ciphertext authenticity if for all such attackers holds that $\mathsf{Adv}_{\mathrm{SE,CAUT}} = \Pr[\mathrm{Dec_K}(\mathrm{CT'_{sym}}) \neq \perp \wedge \mathrm{CT'_{sym}} \neq \mathrm{CT_{sym}}] \leq \mathrm{negl}(\lambda)$.

We define a random-prefix collision-resistant hash function (RPC) as follows.

Definition 7 (Random-prefix collision-resistant hash function (RPC) [ACIK10])**.** Let λ be a security parameter, and let $\mathrm{RPC}\colon \{0,1\}^\lambda \times \mathcal{G} \to \mathcal{Z}$ be a hash function that takes two inputs, one in $\{0,1\}^\lambda$ and one in $\mathcal{G}$, and maps it to an element in $\mathcal{Z}$. Consider the following game between challenger $\mathcal{C}$ and attacker $\mathcal{A}$. The attacker gives the challenger some $g \in \mathcal{G}$. The challenger then picks $k \in \{0,1\}^\lambda$, and gives k and $\mathrm{RPC}(k,g)$ to the attacker. Then, the RPC is random-prefix collision resistant if for all such attackers, it holds that the advantage $\mathsf{Adv}_{\mathrm{RPC}} = \Pr[(k',g') \in \{0,1\}^\lambda \times \mathcal{G} \wedge (k',g') \neq (k,g) \wedge \mathrm{RPC}(k',g') = \mathrm{RPC}(k,g)] \leq \mathrm{negl}(\lambda)$.

In this work, we use the concrete instantiation given by Abe et al. [ACIK10]. In particular, their instantiation of the RPC hash is a second pre-image resistant hash that takes as input a 128-bit string k and an element in $\mathcal{G}$.

2.7 Pair encoding schemes

We give the definitions of pair encoding schemes, and their associated security notions: selective and co-selective symbolic properties.

Definition 8 (Pair encoding schemes (PES) [AC17])**.** A pair encoding scheme for a predicate family $P_\kappa\colon \mathcal{X}_\kappa \times \mathcal{Y}_\kappa \to \{0,1\}$, indexed by $\kappa = (p, \mathrm{par})$, where par specifies some parameters, is given by four deterministic polynomial-time algorithms as described below.

- $\mathrm{Param}(\mathrm{par}) \to n$: On input par, the algorithm outputs $n \in \mathbb{N}$ that specifies the number of common variables, which are denoted as $\mathbf{b} = (b_1, ..., b_n)$.

- $\mathrm{EncKey}(y,p) \to (m_1, m_2, \mathbf{k}(\mathbf{r}, \hat{\mathbf{r}}, \mathbf{b}))$: On input $p \in \mathbb{N}$ and $y \in \mathcal{Y}_\kappa$, this algorithm outputs a vector of polynomials $\mathbf{k} = (k_1, ..., k_{m_3})$ defined over non-lone variables $\mathbf{r} = (r_1, ..., r_{m_1})$ and lone variables $\hat{\mathbf{r}} = (\hat{r}_1, ..., \hat{r}_{m_2})$. Specifically, the polynomial k_i is expressed as

$$k_i = \delta_i \alpha + \sum_{j \in [m_2]} \delta_{i,j} \hat{r}_j + \sum_{j \in [m_1], k \in [n]} \delta_{i,j,k} r_j b_k,$$

  for all $i \in [m_3]$, where $\delta_i, \delta_{i,j}, \delta_{i,j,k} \in \mathbb{Z}_p$.

- $\mathrm{EncCt}(x,p) \to (w_1, w_2, \mathbf{c}(\mathbf{s}, \hat{\mathbf{s}}, \mathbf{b}))$: On input $p \in \mathbb{N}$ and $x \in \mathcal{X}_\kappa$, this algorithm outputs a vector of polynomials $\mathbf{c} = (c_1, ..., c_{w_3})$ defined over non-lone variables $\mathbf{s} = (s, s_2, ..., s_{w_1})$ and lone variables $\hat{\mathbf{s}} = (\hat{s}_1, ..., \hat{s}_{w_2})$. Specifically, the polynomial c_i is expressed as

$$c_i = \sum_{j \in [w_2]} \eta_{i,j} \hat{s}_j + \sum_{j \in \overline{[w_1]}, k \in [n]} \eta_{i,j,k} s_j b_k,$$

  for all $i \in [w_3]$, where $\eta_{i,j}, \eta_{i,j,k} \in \mathbb{Z}_p$.

- $\mathrm{Pair}(x,y,p) \to (\mathbf{E}, \overline{\mathbf{E}})$: On input p, x, and y, this algorithm outputs two matrices $\mathbf{E}$ and $\overline{\mathbf{E}}$ of sizes $(w_1+1) \times m_3$ and $w_3 \times m_1$, respectively.

A PES is correct for every $\kappa = (p, \mathrm{par})$, $x \in \mathcal{X}_\kappa$ and $y \in \mathcal{Y}_\kappa$ such that $P_\kappa(x,y) = 1$, it holds that

$$\mathbf{s}\mathbf{E}\mathbf{k}^\intercal + \mathbf{c}\overline{\mathbf{E}}\mathbf{r}^\intercal = \alpha s.$$

The symbolic property is a powerful security notion for PESs that applies to a large class of predicate encryption schemes.

Definition 9 (Symbolic property ($\mathsf{Sym\text{-}Prop}^{+}$) [AC17, Att19])**.** A pair encoding scheme $\Gamma = (\text{Param}, \text{EncKey}, \text{EncCt}, \text{Pair})$ for a predicate family $P_\kappa \colon \mathcal{X}_\kappa \times \mathcal{Y}_\kappa \to \{0,1\}$ satisfies the (d_1, d_2)-selective symbolic property for positive integers d_1 and d_2 if there exist deterministic polynomial-time algorithms EncB, EncS, and EncR such that for all $\kappa = (p, \text{par})$, $x \in \mathcal{X}_\kappa$ and $y \in \mathcal{Y}_\kappa$ with $P_\kappa(x, y) = 0$, we have

- $\text{EncB}(x) \to \mathbf{B}_1, ..., \mathbf{B}_n \in \mathbb{Z}_p^{d_1 \times d_2}$;
- $\text{EncR}(x, y) \to \mathbf{r}_1, ..., \mathbf{r}_{m_1} \in \mathbb{Z}_p^{d_1}, \mathbf{a}, \hat{\mathbf{r}}_1, ..., \hat{\mathbf{r}}_{m_2} \in \mathbb{Z}_p^{d_2}$;
- $\text{EncS}(x) \to \mathbf{s}_0, ..., \mathbf{s}_{w_1} \in \mathbb{Z}_p^{d_2}, \hat{\mathbf{s}}_1, ..., \hat{\mathbf{s}}_{w_2} \in \mathbb{Z}_p^{d_1}$;

such that $\langle \mathbf{s}_0, \mathbf{a} \rangle \neq 0$ and $\mathbf{a} = (1, \mathbf{0}^{d_2 - 1})$, and if we substitute

$$\hat{s}_{i'} : \hat{\mathbf{s}}_{i'}^{\mathsf{T}} \qquad s_i b_j : \mathbf{B}_j \mathbf{s}_i^{\mathsf{T}} \qquad \alpha : \mathbf{a} \qquad \hat{r}_{k'} : \hat{\mathbf{r}}_{k'} \qquad r_k b_j : \mathbf{r}_k \mathbf{B}_j,$$

for $i \in [w_1], i' \in [w_2], j \in [n], k \in [m_1], k' \in [m_2]$ in all the polynomials of $\mathbf{k}$ and $\mathbf{c}$ (output by EncKey and EncCt, respectively), they evaluate to $\mathbf{0}$.

Similarly, a pair encoding scheme satisfies the (d_1, d_2)-co-selective symbolic security property if there exist $\text{EncB}, \text{EncR}, \text{EncS}$ that satisfy the above properties but where EncB and EncR only take y as input, and EncS takes x and y as input.

A scheme satisfies the (d_1, d_2)-symbolic property if it satisfies the (d_1', d_2')-selective and (d_1'', d_2'')-co-selective properties for $d_1', d_1'' \leq d_1$ and $d_2', d_2'' \leq d_2$.

Agrawal and Chase [AC17] prove that any PES satisfying the (d_1, d_2)-symbolic property can be transformed in a fully secure predicate encryption scheme. The resulting schemes are proven secure under a q-type assumption, which is a security assumption that becomes stronger as some parameter q grows.

In some works [Att14, Att16], the information-theoretic security notion of *perfectly master-key hiding* is used to achieve security under non-parametrized assumptions such as the symmetric external Diffie-Hellman (SXDH).

Definition 10 (Perfectly master-key hiding (PMH) [Att16])**.** A pair encoding scheme $\Gamma = (\text{Param}, \text{EncKey}, \text{EncCt}, \text{Pair})$ for a predicate family $P_\kappa \colon \mathcal{X}_\kappa \times \mathcal{Y}_\kappa \to \{0,1\}$ is perfectly master-key hiding if, for all $\kappa = (p, \text{par})$, $x \in \mathcal{X}_\kappa$ and $y \in \mathcal{Y}_\kappa$ with $P_\kappa(x, y) = 0$, we have that the following distributions are identical:

$$\{\mathbf{k}(\alpha, \mathbf{r}, \hat{\mathbf{r}}, \mathbf{b}, y), \mathbf{c}(\mathbf{s}, \hat{\mathbf{s}}, \mathbf{b}, x)\} \text{ and } \{\mathbf{k}(0, \mathbf{r}, \hat{\mathbf{r}}, \mathbf{b}, y), \mathbf{c}(\mathbf{s}, \hat{\mathbf{s}}, \mathbf{b}, x)\},$$

where all variables $\alpha, \mathbf{r}, \hat{\mathbf{r}}, \mathbf{s}, \hat{\mathbf{s}}, \mathbf{b}$ are taken uniformly at random from $\mathbb{Z}_p$.

Attrapadung [Att16] proves that any pair encoding scheme that is perfectly master-key hiding can be converted to a fully secure predicate encryption scheme. The resulting scheme is then secure under a static assumption such as SXDH.

2.8 Predicate encodings

Predicate encodings are similar to pair encodings, but they consider a slightly more restricted structure.

Definition 11 (Predicate encodings [Wee14, CGW15])**.** A $\mathbb{Z}_p$-bilinear predicate encoding scheme for a predicate family $P_\kappa \colon \mathcal{X}_\kappa \times \mathcal{Y}_\kappa \to \{0,1\}$, indexed by $\kappa = (p, \text{par})$, where par specifies some parameters, is given by five deterministic polynomial-time algorithms $(\text{sE}, \text{rE}, \text{kE}, \text{sD}, \text{rD})$, such that, for all κ, the following properties are satisfied:

- **Linearity:** For all $s(x, y) \in \mathcal{X}_\kappa \times \mathcal{Y}_\kappa$, the functions $\text{sE}(x, \cdot)$, $\text{rE}(y, \cdot)$, $\text{kE}(y, \cdot)$, $\text{sD}(x, y, \cdot)$ and $\text{rD}(x, y, \cdot)$ are $\mathbb{Z}_p$-linear.

- **Restricted α-reconstruction:** For all $(x, y) \in \mathcal{X}_\kappa \times \mathcal{Y}_\kappa$ such that $P_\kappa(x, y) = 1$, and for all $\mathbf{w} \in \mathbb{Z}_p^n$:

$$\mathrm{sD}(x, y, \mathrm{sE}(x, \mathbf{w})) = \mathrm{rD}(x, y, \mathrm{rE}(y, \mathbf{w})) \text{ and } \mathrm{rD}(x, y, \mathrm{kE}(y, \alpha)) = \alpha.$$

- **α-privacy:** For all $(x, y) \in \mathcal{X}_\kappa \times \mathcal{Y}_\kappa$ such that $P_\kappa(x, y) = 0$, and for all $\alpha \in \mathbb{Z}_p$, the following distributions are identically distributed:

$$\{x, y, \alpha, \mathrm{sE}(x, \mathbf{w}), \mathrm{kE}(y, \alpha) + \mathrm{rE}(y, \mathbf{w})\} \text{ and } \{x, y, \alpha, \mathrm{sE}(x, \mathbf{w}), \mathrm{rE}(y, \mathbf{w})\},$$

  where $\mathbf{w} \in_R \mathbb{Z}_p^n$.

3 Our generic CCA-transformation

In this section, we introduce our generic transformation for CCA-secure PE.

3.1 Step one: extending the predicate

Let $\Gamma_{\text{PE,IND-CPA},P}$ be a predicate encryption scheme for the predicate family $P = \{P_\kappa\}_\kappa$ with $P_\kappa \colon \mathcal{X}_\kappa \times \mathcal{Y}_\kappa \to \{0, 1\}$. In the first step of our approach, we transform it into a scheme $\Gamma_{\text{PE,IND-CPA},P'}$ for predicate $P' = \mathrm{PredEx}[P]$, where $\mathrm{PredEx}[P]$ denotes the predicate extension required by the CCA-transformation on predicate P, i.e., $P'_\kappa \colon \mathcal{X}'_\kappa \times \mathcal{Y}'_\kappa \to \{0, 1\}$, where

- $\mathcal{X}'_\kappa = (\mathcal{X}_\kappa, \mathcal{Z})$ and $\mathcal{Y}'_\kappa = (\mathcal{Y}_\kappa, \mathcal{Z} \cup \{*\})$, where $|\mathcal{Z}| \geq 2^{2\lambda}$;
- $P'_\kappa((x, x'), (y, y')) = 1$ if and only if
  - $P_\kappa(x, y) = 1$ and $y' = *$;
  - or $P_\kappa(x, y) = 1$ and $x' = y'$.

In Section 5, we give several predicate-extension transformations that generically transform a CPA-secure PE scheme for the predicate P in a CPA-secure PE scheme for predicate $\mathrm{PredEx}[P]$. Conceptually, we do this by making an AND-composition of the original PE scheme for predicate P with an "all-or-one-identity" IBE. In an "all-or-one-identity" IBE, a user is given either a key for one particular identity $y' \in \mathcal{Z}$ or all identities $y' = *$. These transformations are not fully generic, because they only apply to pairing-based ABE. In particular, they are given in the pair encodings [Att14, AC17] and predicate encodings [Wee14, CGW15] frameworks, since it is relatively simple to generically prove security of such transformations [ABS17], and many PE schemes can be instantiated in these frameworks [AC17, Att19, Amb21].

We note that a scheme with an extended predicate can also be obtained in other ways. For instance, the approaches used for (H)IBE [CHK04, BK05, BCHK07] also apply. Additionally, the generic transformations using delegation by Yamada et al. [YAHK11] yield suitable candidates as well, but only for KP-ABE and CP-ABE. Furthermore, the transformation by Tomida et al. [TKN21] using delegation is similar to our proposed constructions in Section 5, but these only work for their specific KP-ABE and CP-ABE schemes, and are not generic in the sense that they can be applied to any PE. While our transformations in Section 5 are specific to pairing-based PE, a similar approach may also work for PE based on other cryptographic assumptions, for instance, by creating an "all-or-one-identity" IBE from a suitable IBE from post-quantum assumptions [GPV08], and taking an AND-composition with any post-quantum PE [AFV11, ABV+12, Boy13, GVW13].

3.2 Step two: generic CCA-secure construction

Much like in [KV08] and [ACIK10], the predicate extension is generated from part of the ciphertext. To this end, we introduce the notion of "decomposable extended-predicate encryption (EPE)", which we use as input to the CCA-security transformation. In decomposable EPE, we decompose the ciphertext in three parts, such that one of the parts is used to generate the predicate extension with the hash.

Definition 12 (Decomposable EPE)**.** An EPE scheme with encryption algorithm Encrypt is called decomposable if the ciphertexts are decomposable. The ciphertexts $\mathrm{CT}_{(x,x')} \leftarrow \mathrm{Encrypt}(\mathrm{MPK}, (x, x'), M)$ are decomposable if they can be decomposed:

$$\mathrm{CT}_{(x,x')} = (\mathrm{CT}_{\mathrm{M}}, \mathrm{CT}_1, \mathrm{CT}_{2,(x,x')}), \text{ such that}$$

- only $\mathrm{CT}_{2,(x,x')}$ depends on (x, x');
- only CT_{M} contains the message;
- M is uniquely determined by CT_{M}, MPK and CT_1, and conversely, CT_1 is uniquely determined by M, MPK and CT_{M};
- $\mathrm{CT}_1 \in \mathcal{G}$ is generated independently of $\mathrm{CT}_{2,(x,x')}$;
- for any $(\hat{x}, \hat{x}') \in \mathcal{X}'_\kappa$ with $\hat{x}' \neq x'$, we have that any $\mathrm{CT}_{2,(\hat{x},\hat{x}')}$ that is valid for CT_1 is such that $\mathrm{CT}_{2,(\hat{x},\hat{x}')} \neq \mathrm{CT}_{2,(x,x')}$;
- CT_1 is generated uniformly at random over $\mathcal{G}$, such that $\Pr[\mathrm{CT}_1 = \mathrm{CT}'_1 \mid \mathrm{CT}'_1 \in_R \mathcal{G}] \leq \mathrm{negl}(\lambda)$.

In this case, we also define two algorithms for encryption, i.e.,

- $\mathrm{Encrypt}_1(\mathrm{MPK}, M) \to (\mathrm{CT}_{\mathrm{M}}, \mathrm{CT}_1)$;
- $\mathrm{Encrypt}_2(\mathrm{MPK}, (x, x')) \to \mathrm{CT}_{2,(x,x')}$,

such that

$$\mathrm{Encrypt}(\mathrm{MPK}, (x, x'), M) = (\mathrm{Encrypt}_1(\mathrm{MPK}, M), \mathrm{Encrypt}_2(\mathrm{MPK}, (x, x'))).$$

Decomposable EP-KEM. This definition naturally extends to the key-encapsulation variants of EPE, i.e., by replacing CT_{M} by the encapsulated symmetric key K. In this case, K is required to be uniquely determined by MPK and CT_1. We can generically obtain a EP-KEM from an EPE by encrypting a randomly-generated symmetric key K. For PE schemes with a certain algebraic structure, we can also generically obtain a more efficient KEM (Appendix B).

Generic construction. We use a CPA-secure decomposable EP-KEM with an extended predicate to generically construct a CCA-secure hybrid PE for the original predicate.

Definition 13 (Generic CCA-secure construction)**.** Let $\Gamma_{\mathrm{PE}} = (\mathrm{Setup}, \mathrm{KeyGen}, \mathrm{Encaps}, \mathrm{Decaps})$ be a predicate key-encapsulation mechanism for the predicate family $P = \{P_\kappa\}_\kappa$ with $P_\kappa \colon \mathcal{X}_\kappa \times \mathcal{Y}_\kappa \to \{0,1\}$, and suppose $\Gamma_{\text{EP-KEM}} = (\mathrm{Setup}_{\text{EP-KEM}}, \mathrm{KeyGen}_{\text{EP-KEM}}, \mathrm{Encaps}_{\text{EP-KEM}}, \mathrm{Decaps}_{\text{EP-KEM}})$ is a decomposable extended-predicate KEM for predicate $P' = \mathrm{PredEx}[P]$ (e.g., obtained with a predicate-extension transformation (Section 5)). Let $\mathrm{SE} = (\mathrm{Enc}_{\mathrm{K}}, \mathrm{Dec}_{\mathrm{K}})$ be an authenticated symmetric encryption scheme with key space $\mathcal{K}_\lambda$ equal to the space in which CT_{M} lives, and $\mathrm{RPC} \colon \{0,1\}^\lambda \times \mathcal{G} \to \mathcal{Z}$ be a random-prefix collision-resistant hash function. Then, we define $\Gamma'_{\mathrm{PE}} = (\mathrm{Setup}', \mathrm{KeyGen}', \mathrm{Encrypt}', \mathrm{Decrypt}')$ to be the CCA-secure hybrid encryption version of scheme Γ_{PE} for predicate P as follows.

- $\mathrm{Setup}'_{\mathrm{PE}}(\lambda, \mathrm{par})$: On input λ and par, the setup generates $(\mathrm{MPK}, \mathrm{MSK}) \leftarrow \mathrm{Setup}_{\text{EP-KEM}}(\lambda, \mathrm{par})$, and sets $\mathrm{MPK}' = \mathrm{MPK}$ and $\mathrm{MSK}' = \mathrm{MSK}$.
- $\mathrm{KeyGen}'_{\mathrm{PE}}(\mathrm{MSK}', y)$: On input the master secret key MSK' and some $y \in \mathcal{Y}_\kappa$, it returns $\mathrm{SK}'_y \leftarrow \mathrm{KeyGen}_{\text{EP-KEM}}(\mathrm{MSK}, (y, *))$.
- $\mathrm{Encrypt}'_{\mathrm{PE}}(\mathrm{MPK}', x, M)$: On input the master public key MPK', $x \in \mathcal{X}_\kappa$ and message $M \in \{0,1\}^*$, the encrypting user computes $(\mathrm{K}, \mathrm{CT}_1) \leftarrow \mathrm{Encaps}_{1,\text{EP-KEM}}(\mathrm{MPK})$, picks $k \in_R \{0,1\}^\lambda$ and sets $x' = \mathrm{RPC}(k, \mathrm{CT}_1)$, then generates $\mathrm{CT}_{2,(x,x')} \leftarrow \mathrm{Encaps}_{2,\text{EP-KEM}}(\mathrm{MPK}, (x, x'))$, and computes[6] $\mathrm{CT}_{\mathrm{sym}} \leftarrow \mathrm{Enc}_{\mathrm{K}}(M \| \mathrm{CT}_{2,(x,x')})$, and returns
$$\mathrm{CT}'_x = (\mathrm{CT}_{\mathrm{sym}}, \mathrm{CT}_1, \mathrm{CT}_{2,(x,x')}, k).$$
- $\mathrm{Decrypt}'_{\mathrm{PE}}(\mathrm{MPK}', \mathrm{SK}'_y, \mathrm{CT}'_x)$: On input the master public key MPK', the secret key $\mathrm{SK}'_y = \mathrm{SK}_{(y,*)}$, and the ciphertext $\mathrm{CT}'_x = (\mathrm{CT}_{\mathrm{sym}}, \mathrm{CT}_1, \mathrm{CT}_{2,(x,x')}, k)$, if $P_\kappa(x, y) = 1$, then the decrypting user computes $x' = \mathrm{RPC}(k, \mathrm{CT}_1)$ and
$$\mathrm{K}' \leftarrow \mathrm{Decaps}_{\text{EP-KEM}}(\mathrm{MPK}, \mathrm{SK}_{(y,*)}, (\mathrm{CT}_1, \mathrm{CT}_{2,(x,x')})).$$
The user computes $(M' \| \mathrm{CT}'_{2,(x,x')}) \leftarrow \mathrm{Dec}_{\mathrm{K}'}(\mathrm{CT}_{\mathrm{sym}})$, and if $\mathrm{CT}'_{2,(x,x')} = \mathrm{CT}_{2,(x,x')}$, returns M'.

Correctness. The scheme is correct, i.e., if $P_\kappa(x, y) = 1$, then $M' = M$, because the correctness of the P-KEM ensures that $\mathrm{K} = \mathrm{K}'$, and thus, $(M' \| \mathrm{CT}'_{2,(x,x')}) = \mathrm{Dec}_{\mathrm{K}'}(\mathrm{CT}_{\mathrm{sym}}) = \mathrm{Dec}_{\mathrm{K}}(\mathrm{CT}_{\mathrm{sym}}) = \mathrm{Dec}_{\mathrm{K}}(\mathrm{Enc}_{\mathrm{K}}(M \| \mathrm{CT}_{2,(x,x')})) = (M \| \mathrm{CT}_{2,(x,x')})$.

Security. We prove the following theorem.

Theorem 1. *In Definition 13, if* $\Gamma_{\text{EP-KEM}}$ *is a decomposable CPA-secure P-KEM for the extended predicate* $\mathrm{PredEx}[P]$, RPC *is a random-prefix collision-resistant hash function and* $\mathrm{SE} = (\mathrm{Enc}_{\mathrm{K}}, \mathrm{Dec}_{\mathrm{K}})$ *is an authenticated encryption scheme, such that the* RPC *is independent of* $\Gamma_{\text{EP-KEM}}$ *and* SE*, then* Γ'_{PE} *is CCA-secure.*

Proof. We prove this theorem in a series of games in which we start with the real CCA-security game: Game 0. Let $\mathrm{CT}^*_{x^*} = (\mathrm{CT}^*_{\mathrm{sym}}, \mathrm{CT}^*_1, \mathrm{CT}^*_{2,(x^*,x'^*)}, k^*)$ denote the challenge ciphertext (with K^* being the decryption key) for the challenge predicate x^* and message M_β. Let q be the number of decryption queries, and let X_i denote the event that attacker $\mathcal{A}_{\mathrm{CCA}}$ is successful in Game i.

Game 1: In this game, everything is the same as in Game 0, except that, in the first query phase, all decryption queries with $\mathrm{CT}_1 = \mathrm{CT}^*_1$ are rejected. Additionally, in both query phases, the decryption queries with $(\mathrm{CT}_1, k) \neq (\mathrm{CT}^*_1, k^*)$ and $x' = x'^*$ are rejected. The probability that $\mathrm{CT}_1 = \mathrm{CT}^*_1$ holds for any honestly generated ciphertext is $\frac{1}{|\mathcal{G}|}$. Furthermore, the probability that any x' for $(\mathrm{CT}_1, k) \neq (\mathrm{CT}^*_1, k^*)$ is such that $\mathrm{RPC}(k, \mathrm{CT}_1) = x' = x'^* = \mathrm{RPC}(k^*, \mathrm{CT}^*_1)$ is equal to $\Pr[(\mathrm{CT}_1, k) \neq (\mathrm{CT}^*_1, k^*) \wedge \mathrm{RPC}(k, \mathrm{CT}_1) = \mathrm{RPC}(k^*, \mathrm{CT}^*_1)] = \mathsf{Adv}_{\mathrm{RPC}}$. Hence, we have
$$|\Pr[X_0] - \Pr[X_1]| \leq \frac{q}{|\mathcal{G}|} + \mathsf{Adv}_{\mathrm{RPC}}.$$

Game 2: In this game, everything is the same as in Game 1, except that, in the second query phase, all decryption queries are rejected where $\mathrm{CT}_{\mathrm{sym}} \neq \mathrm{CT}^*_{\mathrm{sym}}$ holds, and the key

[6] If one uses an authenticated encryption scheme with associated data [Rog02], one can also treat $\mathrm{CT}_{2,(x,x')}$ as associated data, as it does not need to be secret.

$\mathrm{K} \leftarrow \mathrm{Decaps}_{\text{EP-KEM}}(\mathrm{MPK}, \mathrm{SK}_{(y,*)}, \mathrm{CT}_x)$ is such that $\mathrm{K} = \mathrm{K}^*$. Because this property can only hold if the ciphertext authenticity of the SE is broken, we have

$$|\Pr[X_1] - \Pr[X_2]| \leq \mathsf{Adv}_{\text{SE,CAUT}}.$$

Game 3: In this game, everything is the same as in Game 2, except that, in the second query phase, all valid decryption queries are rejected where $\mathrm{CT}_{2,(x,x')} \neq \mathrm{CT}^*_{2,(x^*,x'^*)}$ holds, and $\mathrm{K} = \mathrm{K}^*$ (and thus, $\mathrm{CT}_1 = \mathrm{CT}^*_1$). Note that this can happen only if the ciphertext authenticity of SE is broken, because the attacker has to generate a valid ciphertext for the same key K^* and another message. Hence, we have

$$|\Pr[X_2] - \Pr[X_3]| \leq \mathsf{Adv}_{\text{SE,CAUT}}.$$

Game 4: At this point, all ciphertexts that are queried in the second phase and that are not rejected are such that, for the keys, it holds that $\mathrm{K} \neq \mathrm{K}^*$. This follows from the fact that K is uniquely determined by MPK and CT_1 (and vice versa), and thus, if $\mathrm{K} = \mathrm{K}^*$, then $\mathrm{CT}_1 = \mathrm{CT}^*_1$. By extension, we have $(\mathrm{CT}_{\text{sym}}, \mathrm{CT}_{2,(x,x')}, k) \neq (\mathrm{CT}^*_{\text{sym}}, \mathrm{CT}^*_{2,(x^*,x'^*)}, k^*)$. (Note that, if $k \neq k^*$, we also have $\mathrm{CT}_{2,(x,x')} \neq \mathrm{CT}^*_{2,(x^*,x'^*)}$, which follows from rejecting all ciphertexts with $x' = x'^*$ in Game 1. From the fact that the EP-KEM is decomposable, it follows that $x' \neq x'^*$ implies $\mathrm{CT}_{2,(x,x')} \neq \mathrm{CT}^*_{2,(x^*,x'^*)}$.) For these cases, we had rejected the decryption queries (in Games 2 and 3). Because this game is the same as Game 3, we have

$$|\Pr[X_3] - \Pr[X_4]| = 0.$$

Game 5: In this game, everything is the same as in Game 4, except that we generate the challenge ciphertext as follows. Let $\mathcal{O}_{\text{RPC}}$ denote the oracle that finds $k \in \{0,1\}^\lambda$ such that $\mathrm{RPC}(k, g) = z$ for any given $(g, z) \in \mathcal{G} \times \mathcal{Z}$. Because RPC is independent of the P-KEM and symmetric encryption scheme, this does not give the attacker any advantage. Then, the challenger generates $(\mathrm{K}^*, \mathrm{CT}_1, \mathrm{CT}_{2,(x^*,x'^*)}) \leftarrow \mathrm{Encaps}_{\text{EP-KEM}}(\mathrm{MPK}, (x^*, x'^*))$ for the challenge predicate x^* and randomly chosen x'^*, and queries the oracle $\mathcal{O}_{\text{RPC}}$ with (CT_1, x'^*), which returns k^* if it exists. (Otherwise, it repeats the process of generating new ciphertexts until the oracle provides some output k^*. This likely succeeds because of the random-prefix collision resistance of the RPC. Intuitively, if many such inputs exist for which the oracle does not return a output, we can also find many g such that there exist at least two k, k' with $\mathrm{RPC}(k, g) = \mathrm{RPC}(k', g)$, which breaks the random-prefix collision resistance of the RPC.) The challenger then outputs the challenge ciphertext as $(\hat{\mathrm{K}}^*, \mathrm{CT}_1, \mathrm{CT}_{2,(x^*,x'^*)}, k^*)$, where $\hat{\mathrm{K}}^*$ is a randomly chosen key that replaces K^*. Because the attacker cannot make decryption queries for K^*, it can only distinguish this game from Game 4 by breaking the CPA-security of the EP-KEM. Therefore, we have

$$|\Pr[X_4] - \Pr[X_5]| \leq \mathsf{Adv}_{\text{EP-KEM,IND-CPA}}.$$

Game 6: In this game, everything is the same as in Game 5, except we replace the challenge message by a randomly generated message of the same length as M_β. By the ciphertext indistinguishability of the symmetric encryption scheme, no attacker can distinguish Game 5 from Game 6, i.e.,

$$|\Pr[X_5] - \Pr[X_6]| \leq \mathsf{Adv}_{\text{SE,CIND}}.$$

Summary: In this final game, because the ciphertext is for a random message, the success probability of the attacker is $\frac{1}{2}$, i.e., $\Pr[X_6] = \frac{1}{2}$. This gives us the following upper bound on the advantage of the attacker in the real security game:

$$\mathsf{Adv}_{\text{PE,IND-CCA}} = \left|\Pr[X_0] - \frac{1}{2}\right|$$

$$\leq \frac{q}{|\mathcal{G}|} + \mathsf{Adv}_{\mathrm{RPC}} + 2\mathsf{Adv}_{\mathrm{SE,CAUT}} + \mathsf{Adv}_{\text{EP-KEM,IND-CPA}} + \mathsf{Adv}_{\mathrm{SE,CIND}}.$$

Since all advantages on the right-hand side are negligible in λ, it holds that $\mathsf{Adv}_{\mathrm{PE,IND\text{-}CCA}}$ is negligible in λ. □

Remark 1. The predicate extension x'^* associated with the ciphertext is determined during encryption by the challenger, and not by the attacker. Possibly, to obtain a fully CCA-secure hybrid PE scheme, one can make an AND-composition of a selectively secure "all-or-one-identity" IB-KEM and a fully secure P-KEM. In this case, a selectively secure IB-KEM is sufficient, because the challenger can generate the predicate extension x'^* before generating the challenge ciphertext. Formalizing such a composition is however not trivial, for instance, because the master public keys of fully secure and selectively secure schemes have a different structure and are thus difficult to split (to build the AND-composition). We therefore believe that such a generic transformation is not as simple to prove generically secure as the proposed transformations in this work. Additionally, it may require a combination of various (complex) proof techniques.

Remark 2. Our proof techniques are similar to but also slightly different from the ACIK techniques. In fact, by feeding $\mathrm{CT}_{2,(x,x')}$ through the authenticated symmetric encryption scheme, a part of the proof is more similar to the BK-approach. However, unlike BK, we use the same key K to encrypt and authenticate the message M, and to authenticate $\mathrm{CT}_{2,(x,x')}$. To ensure that this can be done securely, we require MPK, M, CT_M and CT_1 to be highly dependent on one another. This property is arguably easier to verify than ACIK's rejection property. Furthermore, we explicitly require CT_1 to be sufficiently random (which is a requirement that is inspired by the KV scheme [KV08]). Lastly, note that our property that, for any $(\hat{x}, \hat{x}') \in \mathcal{X}'_\kappa$ with $\hat{x}' \neq x'$, we have that any $\mathrm{CT}_{2,(\hat{x},\hat{x}')}$ that is valid for CT_1 is such that $\mathrm{CT}_{2,(\hat{x},\hat{x}')} \neq \mathrm{CT}_{2,(x,x')}$, is similar to ACIK's unique-split property.

3.3 Variation on the construction: special decomposable EP-KEM

One of the differences between our transformation and the transformation by Abe et al. [ACIK10] is that we do not require the $\mathrm{CT}_{2,(x,x')}$ part to be uniquely defined from CT_1. Instead, we require the encapsulated key K to be uniquely determined by CT_1. We do this, because many PE schemes do not uniquely determine $\mathrm{CT}_{2,(x,x')}$ and do uniquely determine K from CT_1, e.g., the unbounded ABE schemes in [RW13, AC17]. Furthermore, such a deterministic property should also assume that the second ciphertext part $\mathrm{CT}_{2,(x,x')}$ is not delegatable in some way. For example, in many ABE schemes, one can simply drop certain ciphertext components such that this yields a valid ciphertext for a smaller set (in KP-ABE) or a more restricted policy (in CP-ABE).

In many cases, however, we can decompose the ciphertext in such a way that one part is dedicated to the predicate extension x' only. In this case, the ciphertext is of the form

$$\mathrm{CT}_{(x,x')} = (\mathrm{K}, \mathrm{CT}_1, \mathrm{CT}_{2,x}, \mathrm{CT}_{3,x'}),$$

such that only $\mathrm{CT}_{3,x'}$ depends on x' and it is uniquely determined by x', CT_1 and MPK. Furthermore, decryption with a different $\mathrm{CT}'_{3,x'}$ should yield a uniformly distributed output. In this case, we can make two different variants of the generic CCA-secure construction. Instead of including $\mathrm{CT}_{2,(x,x')}$ in the payload of the symmetric encryption scheme, include only $\mathrm{CT}_{2,x}$. Alternatively, we can include $\mathrm{CT}_{2,x}$ in the input to the RPC hash, such that the indistinguishability between Game 2 and 3 follows from the target-collision resistance of the RPC hash. Furthermore, because $\mathrm{CT}_{3,x'}$ is uniquely determined by x', CT_1, and MPK, $\mathrm{CT}_{3,x'}$ cannot differ from the challenge ciphertext if x' and CT_1 are equal to the

challenge ciphertext. If decryption is done with a different $\mathrm{CT}'_{3,x''} \neq \mathrm{CT}_{3,x'}$, then the output key is uniformly distributed. With this latter approach, the key-encapsulation and data-encapsulation mechanisms are also strictly separated, which can be advantageous in the implementation of the schemes.

Special decomposable EPE. A special decomposable EPE is a decomposable EPE (Definition 12) with the additional property that the ciphertext $\mathrm{CT}_{2,(x,x')}$ can be split in two parts $\mathrm{CT}_{2,(x,x')} = (\mathrm{CT}'_{2,x}, \mathrm{CT}'_{3,x'})$, such that $\mathrm{CT}_{3,x'}$ is uniquely determined by x', CT_1 and MPK. We also generate $\mathrm{CT}_{2,x}$ in the first encryption algorithm. Another property that we require is that decryption with a different $\mathrm{CT}'_{3,x'}$ yields an output that is statistically close to uniformly distributed.

Definition 14 (Special decomposable EPE)**.** An EPE scheme with encryption algorithm Encrypt is called special decomposable if the ciphertexts are special decomposable. The ciphertexts $\mathrm{CT}_{(x,x')} \leftarrow \mathrm{Encrypt}(\mathrm{MPK}, (x, x'), M)$ are special decomposable if they can be decomposed:

$$\mathrm{CT}_{(x,x')} = (\mathrm{CT}_{\mathrm{M}}, \mathrm{CT}_1, \mathrm{CT}_{2,x}, \mathrm{CT}_{3,x'}), \text{ such that}$$

- only $\mathrm{CT}_{2,x}$ depends on x, and only $\mathrm{CT}_{3,x'}$ depends on x';
- only CT_{M} contains the message;
- M is uniquely determined by CT_{M}, MPK and CT_1, and conversely, CT_1 is uniquely determined by M, MPK and CT_{M};
- $\mathrm{CT}_{3,x'}$ is uniquely determined by MPK, CT_1 and x', and conversely, x' is uniquely determined by MPK, CT_1 and $\mathrm{CT}_{3,x'}$;
- CT_1 is generated independently of $\mathrm{CT}_{2,x}$ and $\mathrm{CT}_{3,x'}$;
- $(\mathrm{CT}_{\mathrm{M}}, \mathrm{CT}_1, \mathrm{CT}_{2,x})$ is generated in such a way that CT_1 is generated uniformly at random in some space $\mathcal{G}'$, such that $\Pr[\mathrm{CT}_1 = \mathrm{CT}'_1 \mid \mathrm{CT}'_1 \in_R \mathcal{G}'] \leq \mathrm{negl}(\lambda)$;
- For all $\mathrm{CT}'_{3,x''} \neq \mathrm{CT}_{3,x'}$, it holds that $\mathrm{Decrypt}(\mathrm{MPK}, \mathrm{KeyGen}(\mathrm{MSK}, (y, x')), (\mathrm{CT}_{\mathrm{M}}, \mathrm{CT}_1, \mathrm{CT}_{2,x}, \mathrm{CT}'_{3,x''}))$ is statistically close to uniformly distributed.

In this case, we also define two algorithms for encryption, i.e.,

- $\mathrm{Encrypt}_1(\mathrm{MPK}, M, x) \to (\mathrm{CT}_{\mathrm{M}}, \mathrm{CT}_1, \mathrm{CT}_{2,x})$;
- $\mathrm{Encrypt}_2(\mathrm{MPK}, x') \to \mathrm{CT}_{3,x'}$,

such that

$$\mathrm{Encrypt}(\mathrm{MPK}, (x, x'), M) = (\mathrm{Encrypt}_1(\mathrm{MPK}, M), \mathrm{Encrypt}_2(\mathrm{MPK}, (x, x'))).$$

Construction for CCA-secure PE. We define the construction for generic CCA-secure PE as follows. To show that the security proofs for P-KEMs built from PE schemes and P-KEM schemes are the same, we define the construction below using a PE as input.

Definition 15 (Generic CCA-secure PE from special decomposable EPE)**.** Let $\Gamma_{\mathrm{PE}} = (\mathrm{Setup}, \mathrm{KeyGen}, \mathrm{Encrypt}, \mathrm{Decrypt})$ be a PE for the message space $\mathcal{M}_\lambda$ and the predicate family $P = \{P_\kappa\}_\kappa$ with $P_\kappa \colon \mathcal{X}_\kappa \times \mathcal{Y}_\kappa \to \{0, 1\}$, and suppose $\Gamma_{\mathrm{SEPE}} = (\mathrm{Setup}_{\mathrm{SEPE}}, \mathrm{KeyGen}_{\mathrm{SEPE}}, \mathrm{Encrypt}_{\mathrm{SEPE}}, \mathrm{Decrypt}_{\mathrm{SEPE}})$ is a special decomposable extended-predicate encryption scheme for predicate $P' = \mathrm{PredEx}[P]$ (e.g., obtained with a predicate-extension transformation (Section 5)). Let $\mathrm{SE} = (\mathrm{Enc}_{\mathrm{K}}, \mathrm{Dec}_{\mathrm{K}})$ be an authenticated symmetric

encryption scheme with key space $\mathcal{K}_\lambda$ equal to the space in which $\mathrm{CT_M}$ lives, and $\mathrm{RPC}\colon \{0,1\}^\lambda \times \mathcal{G} \to \mathcal{Z}$ be a random-prefix collision-resistant hash function. Then, we define $\Gamma'_{\mathrm{PE}} = (\mathrm{Setup}', \mathrm{KeyGen}', \mathrm{Encrypt}', \mathrm{Decrypt}')$ to be the CCA-secure hybrid encryption version of scheme Γ_{PE} for predicate P as

- $\mathrm{Setup}'_{\mathrm{PE}}(\lambda, \mathrm{par})$: On input λ and par, the setup generates the key pair $(\mathrm{MPK}, \mathrm{MSK}) \leftarrow \mathrm{Setup}_{\mathrm{SEPE}}(\lambda, \mathrm{par})$, and sets $\mathrm{MPK}' = \mathrm{MPK}$ and $\mathrm{MSK}' = \mathrm{MSK}$.
- $\mathrm{KeyGen}'_{\mathrm{PE}}(\mathrm{MSK}', y)$: On input the master secret key MSK' and some $y \in \mathcal{Y}_\kappa$, it returns $\mathrm{SK}'_y \leftarrow \mathrm{KeyGen}_{\mathrm{SEPE}}(\mathrm{MSK}, (y, *))$.
- $\mathrm{Encrypt}'_{\mathrm{PE}}(\mathrm{MPK}', x, M)$: On input the master public key MPK', $x \in \mathcal{X}_\kappa$ and message $M \in \{0,1\}^*$, the user first picks a random message $M' \in \mathcal{M}_\lambda$, and computes $(\mathrm{CT_M}, \mathrm{CT}_1, \mathrm{CT}_{2,x}) \leftarrow \mathrm{Encrypt}_{1,\mathrm{SEPE}}(\mathrm{MPK}, M', x)$, picks $k \in_R \{0,1\}^\lambda$, sets $x' = \mathrm{RPC}(k, \mathrm{CT_M}, \mathrm{CT}_1, \mathrm{CT}_{2,x})$, and generates the last partial ciphertext $\mathrm{CT}_{3,x'} \leftarrow \mathrm{Encrypt}_{2,\text{EP-KEM}}(\mathrm{MPK}, x')$. The user then computes $\mathrm{CT_{sym}} \leftarrow \mathrm{Enc}_{M'}(M)$ and returns
$$\mathrm{CT}'_x = (\mathrm{CT_{sym}}, \mathrm{CT_M}, \mathrm{CT}_1, \mathrm{CT}_{2,x}, \mathrm{CT}_{3,x'}, k).$$
- $\mathrm{Decrypt}'_{\mathrm{PE}}(\mathrm{MPK}', \mathrm{SK}'_y, \mathrm{CT}'_x)$: On input the master public key MPK', the secret key SK'_y, and the ciphertext $\mathrm{CT}'_x = (\mathrm{CT_{sym}}, \mathrm{CT_M}, \mathrm{CT}_1, \mathrm{CT}_{2,x}, \mathrm{CT}_{3,x'}, k)$, if $P_\kappa(x, y) = 1$, then the decrypting user computes $x' = \mathrm{RPC}(k, \mathrm{CT_M}, \mathrm{CT}_1, \mathrm{CT}_{2,x})$ and decrypts
$$K \leftarrow \mathrm{Decrypt}_{\mathrm{SEPE}}(\mathrm{MPK}, \mathrm{SK}_{(y,*)}, \mathrm{CT}'_x),$$
and then outputs
$$M' \leftarrow \mathrm{Dec}_K(\mathrm{CT_{sym}}).$$

Correctness. Correctness follows readily from the correctness of the used PE.

Security. We prove security similarly as that of the construction in Definition 13.

Theorem 2. *In Definition 15, if* Γ_{SEPE} *is a special decomposable CPA-secure PE for the extended predicate* $\mathrm{PredEx}[P]$*, and* RPC *is a random-prefix collision-resistant hash function, such that the* RPC *is independent of* Γ_{SEPE}*, then* Γ'_{PE} *is CCA-secure.*

Proof. We prove this theorem in a series of games in which we start with the real CCA-security game: Game 0. Let $\mathrm{CT}^*_{x^*} = (\mathrm{CT}^*_{\mathrm{sym}}, \mathrm{CT}^*_{\mathrm{M}}, \mathrm{CT}^*_1, \mathrm{CT}^*_{2,x^*}, \mathrm{CT}^*_{3,x'^*}, k^*)$ denote the challenge ciphertext for the challenge predicate x^* and message M_β. Let q be the number of decryption queries, and let X_i denote the event that attacker $\mathcal{A}_{\mathrm{CCA}}$ is successful in Game i.

Game 1: In this game, everything is the same as in Game 0, except that, in the first query phase, all decryption queries with $\mathrm{CT}_1 = \mathrm{CT}^*_1$ are rejected. In both query phases, the decryption queries with $(\mathrm{CT_{sym}}, \mathrm{CT_M}, \mathrm{CT}_1, \mathrm{CT}_{2,x}, k) \neq (\mathrm{CT}^*_{\mathrm{sym}}, \mathrm{CT}^*_{\mathrm{M}}, \mathrm{CT}^*_1, \mathrm{CT}^*_{2,x^*}, k^*)$ and $x' = x'^*$ are rejected. The probability that $\mathrm{CT}_1 = \mathrm{CT}^*_1$ holds for any honestly generated ciphertext is $\frac{1}{|\mathcal{G}|}$. Furthermore, the probability that any x' for $(\mathrm{CT_M}, \mathrm{CT}_1, \mathrm{CT}_{2,x}, k) \neq (\mathrm{CT}^*_{\mathrm{M}}, \mathrm{CT}^*_1, \mathrm{CT}^*_{2,x^*}, k^*)$ is such that $\mathrm{RPC}(k, \mathrm{CT_M}, \mathrm{CT}_1, \mathrm{CT}_{2,x}) = x' = x'^* = \mathrm{RPC}(k^*, \mathrm{CT}^*_{\mathrm{M}}, \mathrm{CT}^*_1, \mathrm{CT}^*_{2,x^*})$ is equal to
$$\Pr[(k, \mathrm{CT_M}, \mathrm{CT}_1, \mathrm{CT}_{2,x}) \neq (\mathrm{CT}^*_{\mathrm{M}}, \mathrm{CT}^*_1, \mathrm{CT}^*_{2,x^*}, k^*)$$
$$\wedge \mathrm{RPC}(k, \mathrm{CT_M}, \mathrm{CT}_1, \mathrm{CT}_{2,x}) = \mathrm{RPC}(k^*, \mathrm{CT}^*_{\mathrm{M}}, \mathrm{CT}^*_1, \mathrm{CT}^*_{2,x^*})] = \mathsf{Adv}_{\mathrm{RPC}}.$$

Hence, we have
$$|\Pr[X_0] - \Pr[X_1]| \leq \frac{q}{|\mathcal{G}|} + \mathsf{Adv}_{\mathrm{RPC}}.$$

Game 2: In this game, everything is the same as in Game 1, except that, in the second query phase, all decryption queries are rejected where $\mathrm{CT}_{\mathrm{sym}} \neq \mathrm{CT}^*_{\mathrm{sym}}$ holds, and the key $\mathrm{K} \leftarrow \mathrm{Decrypt}_{\mathrm{SEPE}}(\mathrm{MPK}, \mathrm{SK}_{(y,*)}, \mathrm{CT}_x)$ is such that $\mathrm{K} = \mathrm{K}^*$. Because this property can only hold if the ciphertext authenticity of the SE is broken, we have

$$|\Pr[X_1] - \Pr[X_2]| \leq \mathsf{Adv}_{\mathrm{SE,CAUT}}.$$

Game 3: At this point, all ciphertexts that are queried in the second phase and that are not rejected are such that, for the keys, it holds that $\mathrm{K} \neq \mathrm{K}^*$. This also follows from the fact that $\mathrm{CT}_{3,x'}$ is uniquely determined by MPK, CT_1 and x' (and vice versa). Furthermore, $\mathrm{K}' \leftarrow \mathrm{Decrypt}_{\mathrm{SEPE}}(\mathrm{MPK}, \mathrm{SK}_{(y,*)}, (\mathrm{CT}^*_{\mathrm{M}}, \mathrm{CT}^*_1, \mathrm{CT}^*_{2,x^*}, \mathrm{CT}'_{3,x''}, k^*))$ is uniformly distributed over the key space. Because this game is the same as Game 3, we have

$$|\Pr[X_3] - \Pr[X_4]| = 0.$$

Game 5: In this game, everything is the same as in Game 4, except that we generate the challenge ciphertext as follows. Let $\mathcal{O}_{\mathrm{RPC}}$ denote the oracle that finds $k \in \{0,1\}^\lambda$ such that $\mathrm{RPC}(k, g) = z$ for any given $(g, z) \in \mathcal{G} \times \mathcal{Z}$. Because RPC is independent of the PE and symmetric encryption scheme, this does not give the attacker any advantage. Then, the challenger generates $(\mathrm{K}^*, \mathrm{CT}_1, \mathrm{CT}_{2,(x^*,x'^*)}) \leftarrow \mathrm{Encaps}_{\text{EP-KEM}}(\mathrm{MPK}, (x^*, x'^*))$ for the challenge predicate x^* and randomly chosen x'^*, and queries the oracle $\mathcal{O}_{\mathrm{RPC}}$ with (CT_1, x'^*), which returns k^* if it exists. (Otherwise, it repeats the process of generating new ciphertexts until the oracle provides some output k^*. This likely succeeds because of the random-prefix collision resistance of the RPC. Intuitively, if many such inputs exist for which the oracle does not return a output, we can also find many g such that there exist at least two k, k' with $\mathrm{RPC}(k, g) = \mathrm{RPC}(k', g)$, which breaks the random-prefix collision resistance of the RPC.) The challenger then outputs the challenge ciphertext as $(\hat{\mathrm{K}}^*, \mathrm{CT}_1, \mathrm{CT}_{2,(x^*,x'^*)}, k^*)$, where $\hat{\mathrm{K}}^*$ is a randomly chosen key that replaces K^*. Because the attacker cannot make decryption queries for K^*, it can only distinguish this game from Game 4 by breaking the CPA-security of the EPE. Therefore, we have

$$|\Pr[X_4] - \Pr[X_5]| \leq \mathsf{Adv}_{\text{EP-KEM,IND-CPA}}.$$

Game 6: In this game, everything is the same as in Game 5, except we replace the challenge message by a randomly generated message of the same length as M_β. By the ciphertext indistinguishability of the symmetric encryption scheme, no attacker can distinguish Game 5 from Game 6, i.e.,

$$|\Pr[X_5] - \Pr[X_6]| \leq \mathsf{Adv}_{\mathrm{SE,CIND}}.$$

Summary: In this final game, because the ciphertext is for a random message, the success probability of the attacker is $\frac{1}{2}$, i.e., $\Pr[X_6] = \frac{1}{2}$. This gives us the following upper bound on the advantage of the attacker in the real security game:

$$\begin{aligned}\mathsf{Adv}_{\text{PE,IND-CCA}} &= \left|\Pr[X_0] - \frac{1}{2}\right| \\ &\leq \frac{q}{|\mathcal{G}|} + \mathsf{Adv}_{\mathrm{RPC}} + 2\mathsf{Adv}_{\mathrm{SE,CAUT}} \\ &\quad + \mathsf{Adv}_{\text{EP-KEM,IND-CPA}} + \mathsf{Adv}_{\mathrm{SE,CIND}}.\end{aligned}$$

Since all advantages on the right-hand side are negligible in λ, it holds that $\mathsf{Adv}_{\text{PE,IND-CCA}}$ is negligible in λ. □

3.4 Variation on the construction: non-decomposable EP-KEM

To convert extended-predicate schemes that do not have decomposable ciphertexts, we can also base our second step of the transformation on a more generic conversion technique than ACIK [ACIK10], such as CHK [CHK04] or BK [BK05]. To apply those techniques, we can treat the extended predicate (x', y') similarly as the identity in those transformations. For example, as in the CHK-transformation, we can embed the verification key in the ciphertext's extended predicate x', and sign the resulting ciphertext with the associated signing key of the one-time signature scheme. Recall, however, that both these methods provide trade-offs in various practical aspects. That is, OTSs provide a significant efficiency trade-off, and the BK-approach induces a higher storage overhead and relies on more primitives. In particular, for $\lambda = 128$, the storage overhead incurred by the BK-approach is 704 bits (i.e., 128 bits for the MAC, 448 bits for the random seed and 128 bits for the public commitment), while our approach incurs only 128 bits overhead (i.e., the random prefix). Furthermore, relying on more primitives may be undesirable in practice, e.g., because it increases the code size, yields more complex code (that is more difficult to maintain) or is more complicated to optimize. Additionally, the specific instantiations of the primitives used in the BK-transform depend strongly on one another, and diverting from the proposed constructions may be prone to error. Specifically, the random seed should be long enough so that the commitment to the random seed statistically hides the key that is derived from it with a hash function (that is later used for the MAC). The random output generated by the pseudo-random generator (using as input the key that was encapsulated with the EP-KEM) needs to hide both the message (e.g., another encapsulated key or a short message) and the random seed. An example of a practical implementation of the BK-transform that diverted from the proposed constructions is the one in the CIRCL library, whose initial random seed size was too short to statistically hide the derived key[7]. In contrast, the instantiations of the primitives used in our transform depend more directly on the security parameter. The complexity in our transform is mostly in how the primitives are applied to the EP-KEM ciphertexts, but as we show in Section 5, this can be done semi-generically.

3.5 Variation on the construction: CCA-secure PE or P-KEM only

Instead of directly combining a key-encapsulation mechanism with an authenticated encryption scheme, we can also create a CCA-secure PE or P-KEM first (which could be used as-is or combined later with an authenticated encryption scheme). Instead of feeding the $\mathrm{CT}_{2,(x,x')}$-part of the ciphertext through the authenticated encryption scheme, it can be plugged into a one-time secure message authentication code. CCA-security of the PE or P-KEM follows then from the same arguments as in the hybrid case.

4 PE-based signatures: signing with secret keys

In this section, we introduce the new notion of predicate-encryption-based signatures (PEBS). This notion is somewhat related to predicate signatures [AHY15], which also covers identity-based [Sha84, BF01] and attribute-based signatures [MPR11, OT11]. However, our new notion differs in two aspects. In the first place, the definition is different. In our definition, the signing keys and signatures are associated with y, and the verification is done by taking as input x such that $P_\kappa(x, y)$. In contrast, predicate signatures associate the keys with y and the signatures with x, and they verify correctly if $P_\kappa(x, y)$. Note that verification uses always x and not anything else. In the second place, the security assumptions are different. In predicate signatures, the signer has a level of privacy, in

[7] https://github.com/cloudflare/circl/pull/394

that the verifier only learns *that* the signer satisfies a specific policy but not *how*. More specifically, the verifier learns that the signer has some y such that $P_\kappa(x, y) = 1$, but y itself remains hidden. In PEBS, we allow y to be visible to the verifier. This has one advantage: the signer may not be able to hide as easily in a group of other signers (provided that y uniquely identifies a user). Furthermore, it may also be more efficient. Where a predicate-signature signer has to sign a message for the whole predicate x (which could be a set of identities, for example), the PEBS signer needs to sign only for y (which could be an identity in the set) such that $P_\kappa(x, y) = 1$. Similarly, a predicate-signature verifier has to verify the signature for the whole predicate x, while a PEBS verifier has to verify the signature for only the y. Another subtle difference is that a PEBS signer can sign a message without knowing the specific x for which the verifier wants to see a signature from the sender.

4.1 Definitions

Definition 16 (Predicate-encryption-based signature (PEBS) scheme)**.** A predicate-encryption-based signature (PEBS) scheme for a predicate family $P = \{P_\kappa\}_{\kappa \in \mathbb{N}^c}$ over a message space $\mathcal{M} = \{M_\lambda\}_{\lambda \in \mathbb{N}}$ consists of four algorithms:

- PEBS.Setup(λ, par): On input the security parameter λ and parameters par, this probabilistic algorithm generates the domain parameters, the master public key MPK and the master secret key MSK. In addition, κ is set to $\kappa = (p, \text{par})$, where p denotes a natural number.
- PEBS.KeyGen(MSK, y): On input the master secret key MSK and some $y \in \mathcal{Y}_\kappa$, this probabilistic algorithm generates a secret signing key PEBS.SignSK_y.
- PEBS.Sign$(\text{MPK}, \text{PEBS.SignSK}_y, M)$: On input the master public key MPK, a secret signing key PEBS.SignSK_y for some $y \in \mathcal{Y}_\kappa$ and message M, this probabilistic algorithm generates a signature σ_y on message M.
- PEBS.Verify$(\text{MPK}, (\sigma_y, M), x)$: On input the master public key MPK, the signature σ_y on message M, and some $x \in \mathcal{X}_\kappa$, if $P_\kappa(x, y) = 1$, then it returns 1 (accept) or 0 (reject).

Correctness. For all par, $M \in \mathcal{M}_\lambda$, $x \in \mathcal{X}_\kappa$, and $y \in \mathcal{Y}_\kappa$ such that $P_\kappa(x, y) = 1$,

$$\begin{gathered}\Pr[(\text{MPK}, \text{MSK}) \leftarrow \text{PEBS.Setup}(1^\lambda);\\ (\sigma_y, M) \leftarrow \text{PEBS.Sign}(\text{MPK}, \text{PEBS.KeyGen}(\text{MSK}, y), M);\\ \text{PEBS.Verify}(\text{MPK}, (\sigma_y, M), x) \neq 1] \leq \text{negl}(\lambda).\end{gathered}$$

Definition 17 (Existential unforgeability under chosen-message attacks (EUF-CMA) for PEBS)**.** We define the security game EUF-CMA(λ) between challenger and attacker as follows:

- **Setup phase:** The challenger runs PEBS.Setup(λ) to obtain MPK and MSK, and sends the master public key MPK to the attacker;
- **Query phase:** The attacker can make two types of queries:
  - **Key query:** The attacker queries secret signing keys for $y \in \mathcal{Y}_\kappa$ and obtains $\text{PEBS.SignSK}_y \leftarrow \text{PEBS.KeyGen}(\text{MSK}, y)$ in response.
  - **Signing query:** The attacker sends some $y \in \mathcal{Y}_\kappa$ and a message M to the challenger, and receives a signature on the message $(\sigma_y, M) \leftarrow \text{PEBS.Sign}(\text{MPK}, \text{PEBS.KeyGen}(\text{MSK}, y), M)$ in response.

- **Output phase:** The attacker outputs a signature $(\sigma^*_{y^*}, M)$ for $y^* \in \mathcal{Y}_\kappa$ on message M, such that y^* was not queried in key query phase and M that was not signed before in the signing query phase. The attacker also sends some $x^* \in \mathcal{X}_\kappa$ such that $P_\kappa(x^*, y^*) = 1$ and, for all y for which a key query was made, it holds that $P_\kappa(x^*, y) = 0$. If the signature verifies correctly, the attacker wins.

A PEBS scheme is existentially unforgeable under chosen-message attacks (EUF-CMA) if all polynomial-time attackers have at most a negligible success probability to win this security game.

4.2 Generic construction for PEBS

We give a generic construction for PEBS using a PE scheme with predicate extension as input. The additional property that we require for the PE with predicate extension is that a key for $(y, *)$ can be delegated to a key for (y, y') with $y' \in \mathcal{Z}$.

Definition 18 (EPE with delegatable extensions)**.** An EPE scheme has delegatable extensions if there exists an algorithm Delegate such that

- $\text{Delegate}(\text{SK}_{(y,*)}, y')$: On input a secret key $\text{SK}_{(y,*)}$ for $(y, *) \in \mathcal{Y}'_\kappa$ and some $y' \in \mathcal{Z}$, it generates a secret key $\text{SK}_{(y,y')}$ for $(y, y') \in \mathcal{Y}'_\kappa$.

We now use EPE with delegateable extensions to generically construct a PE-based signature scheme.

Definition 19 (Generic PE-based signature construction)**.** Let $\Gamma_{\text{PE}} = (\text{Setup}, \text{KeyGen}, \text{Encrypt}, \text{Decrypt})$ be an extended-predicate encryption scheme for predicate $P' = \text{PredEx}[P]$ (e.g., obtained with a predicate-extension transformation (Section 5)), where $P = \{P_\kappa\}_\kappa$ is the predicate family with $P_\kappa \colon \mathcal{X}_\kappa \times \mathcal{Y}_\kappa \to \{0,1\}$. Let $\mathcal{H} \colon \{0,1\}^* \to \mathcal{Z}$ be a collision-resistant hash function. Then, we define the PEBS scheme $\Gamma_{\text{PEBS}} = (\text{PEBS.Setup}, \text{PEBS.KeyGen}, \text{PEBS.Sign}, \text{PEBS.Verify})$ for predicate P as follows.

- $\text{PEBS.Setup}(\lambda, \text{par})$: This algorithm outputs $\text{Setup}(\lambda, \text{par})$.
- $\text{PEBS.KeyGen}(\text{MSK}, y)$: This algorithm outputs a signing key $\text{PEBS.SignSK}_y \leftarrow \text{KeyGen}(\text{MSK}, (y, *))$.
- $\text{PEBS.Sign}(\text{MPK}, \text{PEBS.SignSK}_y, M)$: This algorithm outputs a signature $\sigma_y \leftarrow \text{Delegate}(\text{PEBS.SignSK}_y, \mathcal{H}(y, M)))$ on message M.
- $\text{PEBS.Verify}(\text{MPK}, (\sigma_y, M), x)$: On input the master public key MPK, the signature σ_y on message M, and some $x \in \mathcal{X}_\kappa$, the algorithm first computes a ciphertext $\text{CT}_{(x,\mathcal{H}(y,M))} \leftarrow \text{Encrypt}(\text{MPK}, (x, \mathcal{H}(y, M)), M')$ for random message M', and then outputs $M' \stackrel{?}{=} \text{Decrypt}(\text{MPK}, \sigma_y, \text{CT}_{(x,\mathcal{H}(y,M))})$ holds.

Note that, instead of using an EPE, we can also use an EP-KEM. In that case, the signature verification first encapsulates a key and then checks whether the decapsulated key obtained by using the decryption key yields the same key. The proofs of correctness and security are the same for this particular construction.

Lemma 1. *The PEBS in Definition 19 is correct if the PE scheme is correct.*

Proof. We have that $\text{PEBS.Verify}(\text{MPK}, (\sigma_y, M), x)$ checks whether

$$\begin{aligned} M' &= \text{Decrypt}(\text{MPK}, \sigma_y, \text{CT}_{(x,\mathcal{H}(y,M))}) \\ &= \text{Decrypt}(\text{MPK}, \text{Delegate}(\text{PEBS.SignSK}_y, \mathcal{H}(y, M)), \text{Encrypt}(\text{MPK}, (x, \mathcal{H}(y, M)), M')) \\ &= \text{Decrypt}(\text{MPK}, \text{SK}_{(y,\mathcal{H}(y,M))}, \text{Encrypt}(\text{MPK}, (x, \mathcal{H}(y, M)), M')) = M', \end{aligned}$$

which is indeed the case. □

Theorem 3. *The PEBS in Definition 19 is EUF-CMA secure if the PE scheme is IND-CPA secure and the hash $\mathcal{H}$ is collision resistant.*

Proof. Suppose that there exists an attacker $\mathcal{A}_{\text{PEBS}}$ that can break the EUF-CMA security of the PEBS. Then, we show that we can also construct an attacker $\mathcal{A}_{\text{PE}}$ on the IND-CPA security of the PE scheme.

- **Setup phase:** The challenger for the PE scheme runs the setup and sends the resulting master public key MPK to the attacker, which it relays to the attacker on the PEBS scheme.
- **Query phase:** The attacker on the PEBS scheme makes two types of queries:
  - **Key query:** When the PEBS attacker requests a key for $y \in \mathcal{Y}_\kappa$, the PE attacker relays the request for $(y, *)$ to the PE challenger, and sends the resulting key to the PEBS attacker.
  - **Signing query:** When the PEBS attacker sends $y \in \mathcal{Y}_\kappa$ and a message M to the challenger, the PE attacker requests a key for $(y, \mathcal{H}(y, M))$ from the PE challenger, and sends the resulting key as the signature on M to the PEBS attacker.
- **Output phase:** Ultimately, the PEBS attacker sends a signature $(\sigma^*_{y^*}, M)$ and some $x^* \in \mathcal{X}^*$ to the PEBS challenger.

The PE attacker then sends $(x^*, \mathcal{H}(y^*, M)) \in \mathcal{X}'_\kappa$, and two arbitrary messages M_0 and M_1 to the challenger, who flips a coin and encrypts one of the two. The PE attacker can then decrypt the ciphertext using the key $\sigma^*_{y^*} = \text{SK}_{(y^*, \mathcal{H}(y^*, M))}$, because $P_\kappa(x^*, y^*) = 1$. Note that, because of the restriction that y^* cannot be queried in the key query phase and none of the y for which a key query was made is allowed to satisfy x^*, message M cannot have been signed before in the signing phase and $\mathcal{H}$ is collision resistant, that none of the requirements for the CPA-security game are violated. Hence, the PE attacker is successful if the PEBS attacker is successful. □

Remark 3. To guarantee unforgeability of the signature scheme, it is important that a fresh ciphertext is generated for the verification of each signature. To minimize the computational costs of verification, we can use PE schemes with an online/offline encryption algorithm [GMC08, HW14, VA23]. In online/offline PE, it is possible to generate an intermediate ciphertext before the ciphertext attribute x is known. Upon actual encryption under x, we require minimal additional computational costs. Similarly, we can precompute ciphertexts for verification, minimizing the online verification costs.

4.3 Example: signatures from ciphertext-policy ABE

An example of a type of predicate encryption for which PEBS could be interesting is ciphertext-policy ABE (CP-ABE). Recall that, in CP-ABE, the secret keys are associated with attribute sets and the ciphertexts with policies. With our generic construction for PEBS, users can also sign messages with their secret (decryption) keys. For example, a key holder could have keys for attributes such as their name, email address, profession, place of residence, etc. They could then sign their messages with that set, and a recipient can verify the signature by creating an AND-statement of all the attributes (or a subset thereof). In this setting, the recipient has the guarantee that the signer possesses those attributes with which they have signed. An advantage of this approach over predicate signatures is also that the signer can do this without knowing the policy for which the recipient may want to verify the signature. It is therefore, in a sense, less interactive than predicate signatures. This could be beneficial in applications, e.g., involving email

encryption or the cloud, where the sender often sends messages before ever interacting with the recipient or where it may not even be clear who the recipient of the message is.

Key delegation. PEBS from CP-ABE is especially interesting in combination with key delegation. Key delegation takes as input a secret key for an attribute set $\mathcal{S}$ and then outputs a secret key for a (possibly smaller) subset $\mathcal{S}' \subseteq \mathcal{S}$. (Note that key delegation is only securely realizable for monotone schemes, so that, if $\mathcal{S}'$ satisfies a policy, then also $\mathcal{S}$ satisfies the policy.) The signer can then sign with any possible subset of attributes they possess. Many monotone CP-ABE schemes, including the RWAC scheme we have implemented and analyzed in Section 6, support such key delegation.

5 New predicate-extension transformations

In this section, we give concrete predicate-extension transformations for pairing-based ABE, which transforms any PE scheme into a decomposable extended-predicate encryption scheme. Its security proofs apply to the security notions for pair and predicate encodings in [Wee14, Att14, AC17, Ven23, dlPVA23]. Roughly, they follow a similar approach as Attrapadung [Att19]. In particular, we take as input a secure PE scheme (satisfying some properties) and perform a predicate transformation on it, i.e., an AND-composition (on the key) of the original scheme and an "all-or-one-identity" IBE scheme. To this end, we adapt the key-policy augmentation transformation of Attrapadung [Att19]. Our adaptation differs from the original in two ways. First, we ensure that, for the extended key predicate $(y, *)$, we can generate a key for all identities (y, y'). Second, we re-use the randomness used in the keys of the original scheme to randomize the partial "all-or-one-identity" key. In this way, we minimize the amount of randomness, and ultimately, the computational costs. For schemes with an admissible pair encoding[8], we use the key randomness r that is used in the polynomial that masks the master-key α, i.e., $\alpha + rb$.

5.1 "All-or-one-identity" IBE

For the predicate extension, we use an "all-or-one-identity" IBE scheme. On a high level, we define the "all-or-one-identity" IBE scheme with identities $x', y' \in \mathbb{Z}_p = \mathcal{Z}$ as follows:

$$\begin{aligned}
\mathrm{MPK}' &= (g, h, e(g,h)^{\alpha}, g^{b'_0}, g^{b'_1}),\\
\mathrm{SK}'_{y'} &= (h^{\alpha + r(b'_0 + y' b'_1)}, h^r), \mathrm{SK}'_{*} = (h^{\alpha + r b'_0}, h^{r b'_1}, h^r),\\
\mathrm{CT}'_{x'} &= (M \cdot e(g,h)^{\alpha s}, g^{s(b'_0 + x' b'_1)}, g^s),
\end{aligned}$$

where $g \in \mathbb{G}, h \in \mathbb{H}$ are generators in the groups $\mathbb{G}, \mathbb{H}$ of prime order p, $e\colon \mathbb{G} \times \mathbb{H} \to \mathbb{G}_T$ is a pairing, and $\alpha, b'_0, b'_1, r, s \in \mathbb{Z}_p$. With SK'_{*}, we can generate $\mathrm{SK}'_{y'}$ for any $y' \in \mathbb{Z}_p$, by computing:

$$h^{\alpha + r b'_0} \cdot \left(h^{r b'_1}\right)^{y'} = h^{\alpha + r(b'_0 + y' b'_1)}.$$

Note that this scheme is similar to the Boneh-Boyen IBE1 scheme [BB04], which is selectively secure, with the modification that it allows for the generation of a secret key that can be used for all identities.

[8] This is a pair encoding with some additional properties, used in [Att19]. Note that any secure pair encoding can be converted into an admissible pair encoding by applying the Layer-Trans transformation in [Att19].

5.2 AND-composition with a PE

The transformation of a PE for predicate P to the PE with extended predicate PredEx[P] consists of an AND-composition with the "all-or-one-identity" IBE. For example, consider the following scheme:

$$\begin{aligned}\mathrm{MPK} &= (g, h, e(g,h)^{\alpha}, g^{\mathbf{b}}),\\ \mathrm{SK}_y &= (h^{\mathbf{r}}, h^{\mathbf{k}(\alpha,\mathbf{r},\hat{\mathbf{r}},\mathbf{b},y)}),\\ \mathrm{CT}_x &= (M \cdot e(g,h)^{\alpha s}, g^{\mathbf{s}}, g^{\mathbf{c}(\mathbf{s},\hat{\mathbf{s}},\mathbf{b},x)}),\end{aligned}$$

where $\mathbf{b}$, $\mathbf{r}, \hat{\mathbf{r}}$, $\mathbf{k}$, $\mathbf{s}$, $\hat{\mathbf{s}}$ and $\mathbf{c}$ denote the vectors that describe the secret key and ciphertext (in line with the definition for pair encodings (Definition 8)), respectively. Then, the transformed scheme is of the form:

$$\begin{aligned}\mathrm{MPK} &= (g, h, e(g,h)^{\alpha}, g^{\mathbf{b}}, g^{b'_0}, g^{b'_1}),\\ \mathrm{SK}_{(y,y')} &= \begin{cases} (h^{\mathbf{r}}, h^{\mathbf{k}(\alpha_1,\mathbf{r},\hat{\mathbf{r}},\mathbf{b},y)}, h^{\alpha-\alpha_1+rb'_0}, h^{rb'_1}) & \text{if } y' = *,\\ (h^{\mathbf{r}}, h^{\mathbf{k}(\alpha_1,\mathbf{r},\hat{\mathbf{r}},\mathbf{b},y)}, h^{\alpha-\alpha_1+r(b'_0+y'b'_1)}) & \text{if } y' \in \mathbb{Z}_p \end{cases}\\ \mathrm{CT}_{(x,x')} &= (M \cdot e(g,h)^{\alpha s}, g^{\mathbf{s}}, g^{\mathbf{c}(\mathbf{s},\hat{\mathbf{s}},\mathbf{b},x)}, g^{s(b'_0+x'b'_1)}),\end{aligned}$$

where $\alpha_1 \in_R \mathbb{Z}_p$. To decrypt, we use the decryption algorithm of the original scheme, which then yields $e(g,h)^{\alpha_1}$, and we apply the following operations to the additional part:

$$e(g^s, h^{\alpha-\alpha_1+r(b'_0+y'b'_1)}) \cdot e(g^{s(b'_0+x'b'_1)}, h^{-r}) = e(g,h)^{(\alpha-\alpha_1)s}.$$

Then, $M = M \cdot e(g,h)^{\alpha s} \cdot (e(g,h)^{(\alpha-\alpha_1)s} \cdot e(g,h)^{\alpha_1})^{-1}$.

Storage and computational overhead. The overhead incurred by this transformation is constant: the key size is increased by two group elements (assuming $y' = *$), and the ciphertexts by one. Further, the key generation costs two extra exponentiations, encryption one and decryption costs two pairing operations and one exponentatiation. These are the actual costs for the scheme when instantiated with the selective-security compiler in [Ven23]. When instantiating the pair and predicate encodings with full-security compilers such as [CGW15, AC17], the overhead is doubled.

Decomposability of the ciphertexts. The resulting extended-predicate encryption scheme is decomposable and even special decomposable:

$$\mathrm{CT}_{(x,x')} = (\underbrace{M \cdot e(g,h)^{\alpha s}}_{\mathrm{CT_M}}, \underbrace{g^{\mathbf{s}}}_{\mathrm{CT}_1}, \underbrace{g^{\mathbf{c}(\mathbf{s},\hat{\mathbf{s}},\mathbf{b},x)}}_{\mathrm{CT}_{2,x}}, \underbrace{g^{s(b'_0+x'b'_1)}}_{\mathrm{CT}_{3,x'}}),$$

and can be easily transformed in a KEM by removing $\mathrm{CT_M}$ and setting the key $\mathrm{K} = e(g,h)^{\alpha s}$. For a fixed master public key MPK, the key K is then uniquely defined by CT_1 and vice versa, and CT_1 is generated uniformly at random over $\mathcal{G} = \mathbb{G}^{|\mathbf{s}|}$ (such that each CT_1 is generated with negligible probability). For $\hat{x}' \neq x'$, we have that $g^{s(b'_0+\hat{x}'b'_1)} \neq g^{s(b'_0+x'b'_1)}$. We can also define a different split, e.g., $\mathrm{CT}_1 = g^s$ and push the rest of $g^{\mathbf{s}}$ in $\mathrm{CT}_{2,x}$. Note that, if one chooses to encapsulate some randomly generated symmetric key K, then $\mathrm{K} \cdot e(g,h)^{\alpha s}$ should be included in CT_1 to ensure that K is uniquely defined by CT_1 and MPK.

Delegatable extensions. The schemes provided with our transform have delegatable extensions (Definition 18). In particular, we define the algorithm

- Delegate$(\mathrm{SK}_{(y,*)}, y')$: On input a secret key

$$\mathrm{SK}_{(y,*)} = (h^{\mathbf{r}}, h^{\mathbf{k}(\alpha_1,\mathbf{r},\hat{\mathbf{r}},\mathbf{b},y)}, h^{\alpha-\alpha_1+rb'_0}, h^{rb'_1})$$

for $(y,*) \in \mathcal{Y}'_\kappa$ and some $y' \in \mathbb{Z}_p$, it generates a secret key

$$\mathrm{SK}_{(y,y')} = (h^{\mathbf{r}}, h^{\mathbf{k}(\alpha_1,\mathbf{r},\hat{\mathbf{r}},\mathbf{b},y)}, h^{\alpha-\alpha_1+r(b'_0+y'b'_1)})$$

for $(y,y') \in \mathcal{Y}'_\kappa$ by computing:

$$h^{\alpha-\alpha_1+rb'_0} \cdot (h^{rb'_1})^{y'} = h^{\alpha-\alpha_1+r(b'_0+y'b'_1)}.$$

Pair and predicate encodings framework. We formulate this transformation in the pair encodings and the predicate encodings frameworks the rest of this section. We prove security of the transformation in several ways. We show that the transformation for pair encodings preserves the symbolic security and perfectly master-key hiding properties. Because we re-use the randomness of the key and ciphertext encodings of the original scheme, the transformation can also be formulated in the predicate encodings framework [Wee14, CGW15], and its security follows from the similarity between the perfectly master-key hiding and α-privacy—the security notion for predicate encodings [ABS17].

5.3 The transformation for pair encodings

We define the transformation more formally for pair encodings as follows.

Definition 20 (PredEx-Trans for PES)**.** Let Γ be a PES for predicate P. Then, we construct a PES for PredEx$[P]$ as follows:

- $\mathrm{Param}'(\mathrm{par}) = \mathrm{Param}(\mathrm{par}) + 2$. The common variables are $\mathbf{b}' = (\mathbf{b}, b'_0, b'_1)$, where $\mathbf{b}$ are the common variables of Γ.
- $\mathrm{EncKey}'((y,y'),p)$. Let $y \in \mathcal{Y}_\kappa$ and $y' \in \mathbb{Z}_p \cup \{*\}$, and generate $\alpha_1 \in_R \mathbb{Z}_p$, and set $\alpha_2 = \alpha - \alpha_1$. Then, compute $\mathbf{k}^{(1)}(\alpha, \mathbf{r}^{(1)}, \hat{\mathbf{r}}^{(1)}, \mathbf{b}, y) \leftarrow \mathrm{EncKey}(y,p)$, and replace each occurrence of α by α_1, yielding $\mathbf{k}^{(2)}(\alpha_1, \mathbf{r}^{(1)}, \hat{\mathbf{r}}^{(1)}, \mathbf{b}, y)$. Additionally, compute

$$\mathbf{k}^{(3)}(\alpha_2, \mathbf{r}^{(1)}, \hat{\mathbf{r}}^{(1)}, \mathbf{b}', y') = \begin{cases} (\alpha_2 + r_1(b'_0 + y'b'_1)), & \text{for } y' \in \mathbb{Z}_p \\ (\alpha_2 + r_1 b'_0, r_1 b'_1)), & \text{for } y' = *. \end{cases}$$

  Output $\mathbf{k}(\alpha, \mathbf{r}, \hat{\mathbf{r}}, \mathbf{b}', (y,y')) = (\mathbf{k}^{(2)}, \mathbf{k}^{(3)})$, where $\mathbf{r} = \mathbf{r}^{(1)}$, and $\hat{\mathbf{r}} = (\alpha_1, \hat{\mathbf{r}}^{(1)})$.
- $\mathrm{EncCt}'((x,x'),p)$. Let $x \in \mathcal{X}_\kappa$ and $x' \in \mathbb{Z}_p$. Compute $\mathbf{c}' = \mathbf{c}'(\mathbf{s}, \hat{\mathbf{s}}, \mathbf{b}, x) \leftarrow \mathrm{EncCt}(x,p)$. Output $\mathbf{c}(\mathbf{s}, \hat{\mathbf{s}}, \mathbf{b}', (x,x')) \leftarrow (\mathbf{c}', s(b'_0 + x'b'_1))$.

Pair/Correctness. Let $(x,x') \in \mathcal{X}'_\kappa$ and $(y,y') \in \mathcal{Y}'_\kappa$ be such that $P'((x,x'),(y,y')) = 1$. In particular, we have $P(x,y) = 1$ and either $y' = *$ or $x' = y'$. Let $(\mathbf{E}', \bar{\mathbf{E}}') \leftarrow \mathrm{Pair}(x,y,p)$, such that $\mathbf{s}\mathbf{E}'(\mathbf{k}'_1)^\intercal + \mathbf{c}'\bar{\mathbf{E}}'\mathbf{r}^\intercal = \alpha_1 s$. If $y' = *$, we recover

$$\alpha_2 s = \mathbf{s} \begin{pmatrix} 1 & x' \\ \mathbf{0}^\intercal & \mathbf{0}^\intercal \end{pmatrix} \mathbf{k}_2^\intercal + (s(b'_0 + x'b'_1)) \begin{pmatrix} 1 & \mathbf{0} \end{pmatrix} \mathbf{r}^\intercal.$$

If $y' \in \mathbb{Z}_p$, we recover

$$\alpha_2 s = \mathbf{s} \begin{pmatrix} 1 \\ \mathbf{0}^\intercal \end{pmatrix} \mathbf{k}_2^\intercal + (s(b'_0 + x'b'_1)) \begin{pmatrix} 1 & \mathbf{0} \end{pmatrix} \mathbf{r}^\intercal.$$

Finally, we recover $\alpha s = \alpha_1 s + \alpha_2 s$. Note that the output of $\mathrm{Pair}'((x,x'),(y,y'))$ is $(\mathbf{E}, \bar{\mathbf{E}})$, where

$$\mathbf{E} = \begin{cases} \left(\mathbf{E}', \begin{pmatrix} 1 & x' \\ \mathbf{0}^\intercal & \mathbf{0}^\intercal \end{pmatrix}\right) & \text{for } y' = *, \\ \left(\mathbf{E}', \begin{pmatrix} 1 \\ \mathbf{0}^\intercal \end{pmatrix}\right) & \text{for } y' \in \mathbb{Z}_p, \end{cases} \qquad \bar{\mathbf{E}} = \begin{pmatrix} \bar{\mathbf{E}}' \\ (1 \quad \mathbf{0}) \end{pmatrix}.$$

5.4 The transformation preserves the symbolic property

We show that the predicate-extension transformation for pair encodings preserves the symbolic property.

Theorem 4. *Suppose that* Γ *satisfies* (d_1, d_2)-Sym-Prop$^+$. *Then,* $\Gamma' = \mathrm{PredEx\text{-}Trans}(\Gamma)$ *for* $\mathrm{CCA}[P]$ *satisfies* $(d_1+1, 2d_2)$-Sym-Prop$^+$.

Proof. We show that the PES satisfies both the selective and co-selective symbolic properties. We define the partial predicate $\bar{P}_\kappa$ such that $\bar{P}_\kappa(x',y') = 1$ if and only if $x' = y'$ or $y' = *$. Suppose that $(x,x') \in \mathcal{X}'_\kappa$ and $(y,y') \in \mathcal{Y}'_\kappa$ are such that $P'_\kappa((x,x'),(y,y')) = 0$. This means that $P_\kappa(x,y) = 0$ or $x' \neq y'$ (with $y' \in \mathbb{Z}_p$) holds (or both). (Note that, if $y' = *$, then we necessarily have $P_\kappa(x,y) = 0$.) In particular, EncB, EncR, and EncS output matrix and vector substitutions for the variables α, $\mathbf{b}$, $\mathbf{r}, \hat{\mathbf{r}}$, $\mathbf{s} = (s, s_1, ...)$ and $\hat{\mathbf{s}}$, i.e., $\mathbf{a}, \mathbf{B}^{(1)}, \mathbf{r}^{(1)}, \hat{\mathbf{r}}^{(1)}, \mathbf{s}^{(1)}$, and $\hat{\mathbf{s}}^{(1)}$ (which are vectors of matrices/vectors). For these substitutions, it holds that, if $P_\kappa(x,y) = 0$, then the polynomials in the encodings evaluate to $\mathbf{0}$.

- **The selective symbolic property:** First, we show that the selective symbolic property holds. We use the substitutions of Γ for the selective symbolic property to substitute the variables and polynomials of Γ' as follows:

$$b_i \ : \ \begin{pmatrix} \mathbf{B}_i^{(1)} \\ \mathbf{0} \end{pmatrix}, \qquad b'_0 \ : \ -x'\mathbf{1}_{d_1+1,1}, \qquad b'_1 \ : \ \mathbf{1}_{d_1+1,1},$$

$$r_1 \ : \ \left(\beta \mathbf{r}_1^{(1)}, \frac{\beta'(1-\beta)}{x'-y'}\right) \text{ if } y' \in \mathbb{Z}_p, \qquad r_1 \ : \ \left(\beta \mathbf{r}_1^{(1)}, 0\right) \text{ if } y = *,$$

$$r_{i'} \ : \ (\beta \mathbf{r}_{i'}^{(1)}, 0), \qquad \hat{r}_{i^{(2)}} \ : \ \beta \hat{\mathbf{r}}_{i^{(2)}}^{(1)},$$

$$\alpha \ : \ \mathbf{a}, \qquad \alpha_1 \ : \ (\beta, \mathbf{0})$$

$$s \ : \ \mathbf{s}_0^{(1)}, \qquad s_j \ : \ \mathbf{s}_j^{(1)}, \qquad \hat{s}_{j'} \ : \ \begin{pmatrix} \mathbf{s}_{j'}^{(1)} \\ 0 \end{pmatrix},$$

  for all $i \in [n], i' \in [2, m_1], i^{(2)} \in [m_2], j \in [0, w_1], j' \in [w_2]$, where $\beta = 1 - P_\kappa(x,y)$ and $\beta' = 1 - \bar{P}_\kappa(x',y')$. Note that $\frac{\beta'(1-\beta)}{x'-y'}$ is well-defined, because if $y' = x'$, then $\beta' = 0$. Note that we indeed have $\langle \mathbf{a}, \mathbf{s_0}^{(1)} \rangle \neq 0$, because Γ satisfies Sym-Prop$^+$.

  We show that, for these substitutions, the polynomials evaluate to $\mathbf{0}$. We have $\mathbf{k} = (\mathbf{k}^{(2)}, \mathbf{k}^{(3)})$ and $\mathbf{c} = (\mathbf{c}', s(b'_0 + x'b'_1))$, where the polynomials in $\mathbf{c}'$ and $\mathbf{k}^{(2)}$ in which α_1 does not occur evaluate to $\mathbf{0}$ due to the selective symbolic property of Γ. The polynomials k' in which α_1 does occur can be written as $k'(\alpha_1) = \delta'\alpha_1 + k''$, where $\delta' \in \mathbb{Z}_p$ and k'' is a polynomial in which α_1 does not occur. If $P_\kappa(x,y) = 1$, then $\beta = 0$ and $\bar{P}_\kappa(x',y') = 0$, and thus, $\mathbf{r}$ and $\hat{\mathbf{r}}$ are all-zero, except possibly the last entry of r_1, which may be $\frac{1}{x'-y'}$. Since the only common variables that occur in k'' are b_i, which are substituted by matrices in which the last rows are all-zero, all combinations $r_i b_j$ evaluate to $\mathbf{0}$. Furthermore, $\alpha_1 = \mathbf{0}$, and therefore $k'(\alpha_1) = \mathbf{0}$. On the other hand, if $P_\kappa(x,y) = 0$, then $\beta = 1$, and all combinations of $r_i b_j$ are

substituted as in Γ itself: $r_i b_j = (\mathbf{r}_i^{(1)}, r_{d_1+1}) \begin{pmatrix} \mathbf{B}_j^{(1)} \\ \mathbf{0} \end{pmatrix} = \mathbf{r}_i^{(1)} \mathbf{B}_j^{(1)}$. And, because in this case, the substitutions for α and α_1 are equal, we have $k'(\alpha_1) = k'(\alpha) = \mathbf{0}$.

For the "new" polynomials in $\mathbf{k}^{(3)}$ and $s(b'_0 + x'b'_1)$, we may need to consider whether $y' \in \mathbb{Z}_p$ or $y' = *$. In general, we have $s(b'_0 + x'b'_1) = \mathbf{s}_1^{(1)}(-x'\mathbf{1}_{d_1+1,1} + x'\mathbf{E}\mathbf{1}_{d_1+1,1}) = \mathbf{0}$. For $\mathbf{k}^{(3)}$, and $y' \in \mathbb{Z}_p$, we have:

$$\begin{aligned} &\alpha_2 + r_1(b'_0 + y'b'_1) \\ &= (\mathbf{a}, 0) - (\beta, \mathbf{0}) + \left(\beta \mathbf{r}_1^{(1)}, \frac{\beta'(1-\beta)}{x'-y'}\right)((-x'\mathbf{1}_{d_1+1,1} + y'\mathbf{1}_{d_1+1,1})) \\ &= (1-\beta, \mathbf{0}) + \frac{\beta'(1-\beta)}{x'-y'}(-x'+y', \mathbf{0}). \end{aligned}$$

If $\bar{P}_\kappa(x', y') = 1$, then $P_\kappa(x, y) = 0$, and thus $\beta = 1$. Also, $\beta' = 0$, and therefore we have $\alpha_2 + r_1(b'_0 + y'b'_1) = \mathbf{0}$. Otherwise, $\bar{P}_\kappa(x', y') = 0$, and thus $\beta' = 1$. Then, we have $\alpha_2 + r_1(b'_0 + y'b'_1) = (1-\beta, \mathbf{0}) - (1-\beta, \mathbf{0}) = \mathbf{0}$. For $\mathbf{k}^{(3)}$ and $y = *$, we necessarily have $P_\kappa(x, y) = 0$ and thus, $\beta = 1$. Then,

$$\alpha_2 + r_1 b'_0 = (\mathbf{a}, 0) - (\beta, \mathbf{0}) + \left(\beta \mathbf{r}_1^{(1)}, 0\right) - x'\mathbf{1}_{d_1+1,1} = \mathbf{0},$$

and $r_1 b'_1 = \left(\beta \mathbf{r}_1^{(1)}, 0\right) \mathbf{1}_{d_1+1,1} = \mathbf{0}$.

- **The co-selective property:** We also show that the co-selective property holds. We use the substitutions of Γ for the co-selective symbolic property to substitute the variables and polynomials of Γ' as follows:

$$\begin{gathered} b_i \ : \ \begin{pmatrix} \mathbf{0}^{d_1 \times d_2} & \mathbf{B}_i^{(1)} \\ \mathbf{0}^{1 \times d_2} & \mathbf{0}^{1 \times d_2} \end{pmatrix}, \\ b'_0 \ : \ \mathbf{1}_{d_1+1, d_2+1}^{(d_1+1) \times 2d_2} - \mathbf{1}_{d_1+1,1}^{(d_1+1) \times 2d_2} + y' \left(\mathbf{1}_{d_1+1, d_2+2}^{(d_1+1) \times 2d_2} - \mathbf{1}_{d_1+1,2}^{(d_1+1) \times 2d_2}\right), \text{ if } y' \in \mathbb{Z}_p \\ b'_1 \ : \ \mathbf{1}_{d_1+1, d_2+2}^{(d_1+1) \times 2d_2} - \mathbf{1}_{d_1+1,2}^{(d_1+1) \times 2d_2}, \text{ if } y' \in \mathbb{Z}_p \\ b'_0 \ : \ \mathbf{1}_{d_1+1, d_2+1}^{(d_1+1) \times 2d_2} - \mathbf{1}_{d_1+1,1}^{(d_1+1) \times 2d_2}, \text{ and } b'_1 \ : \ \mathbf{0}^{(d_1+1) \times 2d_2}, \text{ if } y' = * \\ r_1 \ : \ (\mathbf{r}_1^{(1)}, 1), \qquad r_{i'} \ : \ (\mathbf{r}_{i'}^{(1)}, 0), \qquad \hat{r}_{i^{(2)}} \ : \ \left(\mathbf{0}^{d_2 \times 1}, \hat{\mathbf{r}}_{i^{(2)}}^{(1)}\right), \\ \alpha \ : \ \mathbf{a}^\intercal, \qquad \alpha_1 \ : \ \mathbf{1}_{d_2+1}^{2d_2} \\ s \ : \ \beta \begin{pmatrix} \mathbf{s}_0^{(1)} \\ \mathbf{s}_0^{(1)} \end{pmatrix} + \beta' \begin{pmatrix} 1 \\ \frac{1}{x'-y'} \\ \mathbf{0}^{2d_2-2} \end{pmatrix} \qquad s_j \ : \ \begin{pmatrix} \mathbf{s}_j^{(1)} \\ \mathbf{0}^{d_2} \end{pmatrix}, \qquad \hat{s}_{j'} \ : \ \begin{pmatrix} \mathbf{s}_{j'}^{(1)} \\ 0 \end{pmatrix}, \end{gathered}$$

for all $i \in [n], i' \in [2, m_1], i^{(2)} \in [m_2], j \in [0, w_1], j' \in [w_2]$, where $\beta = 1 - P_\kappa(x, y)$, and $\beta' = 1 - \bar{P}_\kappa(x', y')$. Here, we assume that $d_2 \geq 2$. Note that $\alpha s \neq 0$, because the first entry of s is non-zero, which holds because not both β and β' can be 0 (which would hold only if $P'_\kappa((x, x'), (y', y')) = 1$).

We show that, for these substitutions, the polynomials evaluate to $\mathbf{0}$. Like in the selective case, for the polynomials in $\mathbf{c}'$ and $\mathbf{k}^{(2)}$, in which α_1 does not occur, it follows readily that the polynomials evaluate to $\mathbf{0}$. Similarly, for the polynomials k' in $\mathbf{k}^{(2)}$ in which α_1 does occur, we can write these polynomials as $k'(\alpha_1) = \delta'\alpha_1 + k''$, where $k'' = \sum_{j \in [m_2]} \delta_{i,j} \hat{r}_j + \sum_{j \in [m_1], k \in [n]} \delta_{i,j,k} r_j b_k$. For the original substitutions in Γ, we have

$$k'' = \delta'\mathbf{a} + \sum_{j \in [m_2]} \delta''_j \hat{\mathbf{r}}_j^{(1)} + \sum_{j \in [m_1], k \in [n]} \delta''_{j,k} \mathbf{r}_j^{(1)} \mathbf{B}_k^{(1)} = \mathbf{0}$$

For the "new" substitutions, we have

$$\begin{aligned}
k'' &= \delta' \mathbf{1}_{d_2+1}^{2d_2} + \sum_{j\in[m_2]} \delta_{i,j} \left(\mathbf{0}^{1\times d_2}, \hat{\mathbf{r}}_{i^{(2)}}^{(1)}\right) \\
&+ \sum_{j\in[m_1],k\in[n]} \left(\delta_{i,j,k}(\mathbf{r}_j^{(1)}, (\mathbf{r}_1)_1) \begin{pmatrix} \mathbf{0}^{d_1\times d_2} & \mathbf{B}_i^{(1)} \\ \mathbf{0}^{1\times d_2} & \mathbf{0}^{1\times d_2} \end{pmatrix}\right)^\intercal \\
&= (\mathbf{0}^{d_2}, \delta', \mathbf{0}^{d_2-1})^\intercal + \sum_{j\in[m_2]} \delta_{i,j} \left(\mathbf{0}^{1\times d_2}, \hat{\mathbf{r}}_{i^{(2)}}^{(1)}\right) + \sum_{j\in[m_1],k\in[n]} \delta_{i,j,k}(\mathbf{0}^{d_2}, \mathbf{r}_j^{(1)}\mathbf{B}_k^{(1)})^\intercal \\
&= \left(\mathbf{0}^{d_2}, \left(\delta'\mathbf{a} + \sum_{j\in[m_2]} \delta_{i,j}\hat{\mathbf{r}}_j^{(1)} + \sum_{j\in[m_1],k\in[n]} \delta_{i,j,k}\mathbf{r}_j^{(1)}\mathbf{B}_k^{(1)}\right)\right) = \mathbf{0}.
\end{aligned}$$

Now, we show for the "new" polynomials $\mathbf{k}^{(3)}$ and $s(b'_0 + x'b'_1)$ evaluate to $\mathbf{0}$. If $y' \in \mathbb{Z}_p$, then we have

$$\begin{aligned}
s(b'_0 + x'b'_1) &= \left(\beta \begin{pmatrix} \mathbf{s}_0^{(1)} \\ \mathbf{s}_0^{(1)} \end{pmatrix} + \beta' \begin{pmatrix} 1 \\ \frac{1}{x'-y'} \\ \mathbf{0}^{2d_2-2} \end{pmatrix}\right)(b'_0 + x'b_1) \\
&= \left(\mathbf{0}^{d_1}, \beta(\mathbf{s}_0^{(1)})_1(-1+1) + (-y'+y'+x'-x')(\mathbf{s}_0^{(1)})_2\right) \\
&+ (\mathbf{0}^{d_1}, \beta'(-1 - \frac{y'}{x'-y'} + \frac{x'}{x'-y'})) \\
&= \mathbf{0}^{d_1+1} + (\mathbf{0}^{d_1}, \beta'(-1 + \frac{x'-y'}{x'-y'})) = \mathbf{0}^{d_1+1},
\end{aligned}$$

because either we have $x' = y'$ and then, $\beta' = 0$, or we have $(-1 + \frac{x'-y'}{x'-y'}) = 0$. If $y' = *$, then we have $\bar{P}_\kappa(x', y') = 1$ and thus, $\beta' = 0$, and

$$\begin{aligned}
s(b'_0 + x'b'_1) &= \beta \begin{pmatrix} \mathbf{s}_0^{(1)} \\ \mathbf{s}_0^{(1)} \end{pmatrix} (\mathbf{1}_{d_1+1,d_2+1}^{(d_1+1)\times 2d_2} - \mathbf{1}_{d_1+1,1}^{(d_1+1)\times 2d_2} + x'\mathbf{0}^{(d_1+1)\times 2d_2}) \\
&= (\mathbf{0}^{d_1}, \beta(\mathbf{s}_0^{(1)})_1 - \beta(\mathbf{s}_0^{(1)})_1) = \mathbf{0}^{d_1+1}.
\end{aligned}$$

For $\mathbf{k}^{(3)}$ and $y' \in \mathbb{Z}_p$, we have

$$\begin{aligned}
\alpha_2 + r_1(b'_0 + y'b'_1) &= \mathbf{1}_1^{2d_2} - \mathbf{1}_{d_2+1}^{2d_2} + (\mathbf{r}_1^{(1)}, 1)(b'_0 + y'b'_1) \\
&= \mathbf{1}_1^{2d_2} - \mathbf{1}_{d_2+1}^{2d_2} + (-1, -y'+y', \mathbf{0}^{d_2-2}, 1, y'-y', \mathbf{0}^{d_2-2}) \\
&= \mathbf{1}_1^{2d_2} - \mathbf{1}_{d_2+1}^{2d_2} - \mathbf{1}_1^{2d_2} + \mathbf{1}_{d_2+1}^{2d_2} = \mathbf{0}^{2d_2}.
\end{aligned}$$

For $\mathbf{k}^{(3)}$ and $y' = *$, we have

$$\begin{aligned}
\alpha_2 + r_1 b'_0 &= \mathbf{1}_1^{2d_2} - \mathbf{1}_{d_2+1}^{2d_2} + (\mathbf{r}_1^{(1)}, 1)\left(\mathbf{1}_{d_1+1,d_2+1}^{(d_1+1)\times 2d_2} - \mathbf{1}_{d_1+1,1}^{(d_1+1)\times 2d_2}\right) \\
&= \mathbf{1}_1^{2d_2} - \mathbf{1}_{d_2+1}^{2d_2} + \mathbf{1}_{d_2+1}^{2d_2} - \mathbf{1}_1^{2d_2} = \mathbf{0}^{2d_2},
\end{aligned}$$

and $r_1 b'_1 = \mathbf{0}^{2d_2}$.

Thus, Sym-Prop$^+$ holds for Γ'. □

5.5 The transformation preserves perfectly master-key hiding

We show that the predicate-extension transformation for pair encodings preserves the perfect master-key hiding property.

Theorem 5. *Suppose* Γ *is perfectly master-key hiding. Then,* $\Gamma' = \text{PredEx-Trans}(\Gamma)$ *for* $\text{CCA}[P]$ *is also perfectly master-key hiding.*

Proof. Let $(x, x') \in \mathcal{X}'_\kappa$ and $(y, y') \in \mathcal{Y}'_\kappa$ be such that $P'_\kappa((x, x'), (y, y')) = 0$. First, we show that, if $x' \neq y'$ and $y' \neq *$, we have that α_2 is perfectly hidden, i.e., the distributions

$$\{\alpha_2 + r_1(b'_0 + y'b'_1), \hat{\mathbf{r}}^{(1)}, s(b'_0 + x'b'_1)\} \text{ and } \{r_1(b'_0 + y'b'_1), s(b'_0 + x'b'_1)\}$$

are equal. This follows from the fact that, if $x' \neq y'$, then $b'_0 + y'b'_1$ and $b'_0 + x'b'_1$ are pairwise independent [CGW15]. Furthermore, if $P(x, y) = 0$, then α_1 is perfectly hidden by the assumption that Γ is perfectly master-key hidden.

Suppose that $x' = y'$ or $y' = *$. Then, $\alpha_2 = \alpha - \alpha_1$ is not hidden. To ensure that α is hidden, we sample some random $\bar{\alpha} \in_R \mathbb{Z}_p$, which we subtract from α and α_1, i.e., replace α with $\alpha' \leftarrow \alpha - \bar{\alpha}$ and α_1 with $\alpha'_1 \leftarrow \alpha_1 - \bar{\alpha}$. Note that we still have $\alpha_2 = \alpha' - \alpha'_1$, and thus, this does not change the encodings for x' and y'. Because $P(x, y) = 0$, we can switch out α_1 for α'_1 in $\mathbf{k}^{(2)}$, because α_1 is hidden. Therefore, α is hidden.

Suppose now that $P(x, y) = 1$. In this case, α_1 is not hidden. Then, we similarly hide α by subtracting randomly generated $\bar{\alpha} \in_R \mathbb{Z}_p$, i.e., replace α by $\alpha' \leftarrow \alpha - \bar{\alpha}$. Because $x' \neq y'$ and $y' \neq *$, we have that $\alpha'_2 = \alpha' - \alpha_1 = \alpha - \bar{\alpha} - \alpha_1 \neq \alpha_2$ is hidden. Thus, α is hidden. □

5.6 Transformation for predicate encodings

Because our transformation for pair encodings re-uses the randomness $\mathbf{r}$ and $\mathbf{s}$ in the extension, it can also be applied to predicate encodings [Wee14, CGW15]. In particular, if $\mathbf{r}$ and $\mathbf{s}$ are of length 1, then our transformation does not increase the number of key and ciphertext variables, and thus, the transformation yields a (new) predicate encoding. By Theorem 5, the predicate encoding satisfies the α-privacy property, which is similar to the perfectly master-key hiding property [ABS17]. In fact, the encoding for equality given in [CGW15] is the same as our "all-or-one-identity" IBE for $y' \in \mathbb{Z}_p$. It can be simply adjusted to also include the encodings for $y' = *$.

Definition 21 (PredEx-Trans for predicate encodings)**.** Let $\Gamma = (\text{sE}, \text{rE}, \text{kE}, \text{sD}, \text{rD})$ be a predicate encoding for predicate P. Then, we construct a predicate encoding for the extended predicate $\text{CCA}[P]$ as follows:

- The length of $\mathbf{w}'$ is increased by three compared to $\mathbf{w}$ of Γ: $\mathbf{w}' = (\mathbf{w}, w'_0, w'_1, u)$.
- $\text{sE}'((x, x'), \mathbf{w}') = (\mathbf{c} = \text{sE}(x, \mathbf{w}), c' = w'_0 + x'w'_1)$.
- $\text{sD}'((x, x'), (y, y'), (\mathbf{c}, c')) = \text{sD}(x, y, \mathbf{c}) + c'$.
- $\text{rE}'((y, y'), \mathbf{w}') = (\mathbf{k} = \text{rE}(y, \mathbf{w}) + \text{kE}(y, u), \mathbf{k}' = \text{rE}''(y', \mathbf{w}'))$, where

$$\text{rE}''(y', \mathbf{w}') = \begin{cases} (-u + w'_0 + y'w'_1), & \text{for } y' \in \mathbb{Z}_p \\ (-u + w'_0, w'_1), & \text{for } y' = *. \end{cases}$$

- $\text{kE}'((y, y'), \alpha) = (\mathbf{0}^{|\text{kE}(y,\alpha)|}, \text{kE}''(y', \alpha))$, where

$$\text{kE}''(y', \alpha)) = \begin{cases} (\alpha), & \text{for } y' \in \mathbb{Z}_p \\ (\alpha, 0), & \text{for } y' = *. \end{cases}$$

- $\mathrm{rD}'((x,x'),(y,y'),(\mathbf{k},\mathbf{k}')) = \mathrm{rD}(x,y,\mathbf{k}) + \mathrm{rD}''(x',y',\mathbf{k}')$, where

$$\mathrm{rD}''(x',y',\mathbf{k}') = \begin{cases} 1, & \text{for } y' \in \mathbb{Z}_p \\ \mathbf{k}'_1 + x'\mathbf{k}'_2, & \text{for } y' = *. \end{cases}$$

- **Restricted α-reconstruction:** We have

$$\begin{aligned} \mathrm{sD}'((x,x'),(y,y),\mathrm{sE}'((x,x'),\mathbf{w}')) &= \mathrm{sD}'((x,x'),(y,y),(\mathrm{sE}(x,\mathbf{w})),w'_0 + x'w'_1)) \\ &= \mathrm{sD}(x,y,\mathrm{sE}(x,\mathbf{w}))) + w'_0 + x'w'_1, \end{aligned}$$

  which is equal to

$$\begin{aligned} \mathrm{rD}'((x,x'),(y,y),\mathrm{rE}'((y,y'),\mathbf{w}')) &= \mathrm{rD}'((x,x'),(y,y'),(\mathrm{rE}(y,\mathbf{w}) + \mathrm{kE}(y,u),\mathbf{k}')) \\ &= \mathrm{rD}(x,y,\mathrm{rE}(y,\mathbf{w}) + \mathrm{kE}(y,u)) + \mathrm{rD}''(x',y',\mathbf{k}') \\ &= \mathrm{rD}(x,y,\mathrm{rE}(y,\mathbf{w})) + \mathrm{rD}(x,y,\mathrm{kE}(y,u)) \\ &\quad + \begin{cases} -u + w'_0 + y'w'_1, & \text{for } y' \in \mathbb{Z}_p \\ -u + w'_0 + x'w'_1, & \text{for } y' = *. \end{cases} \\ &= \mathrm{sD}(x,y,\mathrm{sE}(y,\mathbf{w})) + u - u + w'_0 + x'w'_1. \end{aligned}$$

- **α-privacy:** The argument is similar as in the case of perfectly master-key hiding for the transformation for pair encodings (Theorem 5).

Remark 4. One of the advantages of predicate encodings over pair encodings is that the CGW compiler [CGW15] also supports an attribute-hiding property, i.e., for some types of predicate encryption such as identity-based encryption, it can hide the attributes in the ciphertext. This means that, in addition to the message being hidden, the predicate x is also hidden. In Appendix C, we show that this property is preserved under our transformation.

6 Performance analysis of CCA-secure constructions

To illustrate the benefits of our transformation with respect to the efficiency compared to other generic transformation techniques, we first analyze the storage and computational costs from a theoretical standpoint, and then of two concrete constructions. On a high level, the efficiency of the transformation depends on the P-KEM part that encapsulates the symmetric key, and the other primitives used. For simplicity, we assume that all transformations considered in the introduction can support KEM variants, which encapsulate a symmetric key in the P-KEM part and use this key to symmetrically encrypt the plaintext.

6.1 Theoretical performance analysis

We first compare the efficiency of our CCA-transformation with the others from a theoretical standpoint. Ours incurs only a small constant overhead in all algorithms and the key and ciphertext sizes in the first step, regardless of the size of the predicate. For the other transformations, this is not the case. Especially for schemes with linear-sized predicates, such as ABE, our construction provides a significant efficiency improvement. In contrast, the other approaches applicable to ABE incur the following efficiency trade-offs:

- FO [FO99, HHK17]: in general, this approach incurs little to no overhead to most algorithms, except for the decryption algorithm, which requires an invocation of the encryption algorithm, whose costs are often linear;

- YAKK-del [YAHK11]: depending on the type of ABE, this transformation for delegatable schemes might either be very efficient or very costly. For KP-ABE, the transformation incurs only a small constant overhead in all algorithms and the key and ciphertext sizes. For CP-ABE, the transformation incurs an additional overhead that is linear in the security parameter in the encryption and decryption algorithms;

- YAHK-ver [YAHK11], BL [BL16]: these transformations for verifiable schemes incur little to no overhead in most of the algorithms and the key and ciphertext sizes, except for the decryption algorithm, which also verifies whether the ciphertexts are well-formed. The costs incurred by the verification step are similar to the decryption costs of the CPA-secure PE scheme, and therefore roughly double the decryption costs of the CCA-secure PE scheme (which are often linear in the predicate size);

- KW [KW19]: this fully-generic transformation is very costly and incurs an overhead in all algorithms and sizes that is linear in the security parameter.

6.2 Benchmarks

We also analyze the efficiency of the P-KEM part by implementing and benchmarking the available CCA-transformed versions of two schemes. (We leave out the KW [KW19] transformation due to its evident blowup in costs as argued earlier this section.) The first scheme is CGW-IBE, the anonymous IBE scheme by Chen, Gay and Wee [CGW15]. The second scheme is RWAC, the fully secure variant [AC17, Att19] of the CP-ABE scheme by Rouselakis and Waters [RW13]. We provide full descriptions of the schemes and their CCA-secure variants in Appendices D and E. For CGW-IBE, we provide a more optimized variant implied by our CCA-secure variant, which, interestingly, resembles the first Kiltz-Vahlis scheme [KV08]. For RWAC, we approximate the efficiency of the FO, delegation and verifiability-based transformations. For FO, we add the encryption costs to the decryption costs (for same-length inputs). Note that this also includes the public-key storage cost in the secret key size as this is required for re-encryption. For verifiability-based transformations, we add one attribute in the ciphertext-policy input, and multiply the decryption costs by a factor 2. For delegation-based transformations, we assume that the length of the verification key of the used OTS is at least 128 bits (at the 128-bit security level), and thus, that the key set is extended with $2 \cdot 128 = 256$ attributes, and the ciphertext policy with 128 attributes. We have implemented the schemes in Rust[9] using the BLS12-381 crate provided by the zkCrypto group [ZkC20].

Table 2 summarizes the benchmarks obtained by running the code on an AMD Ryzen 7 3700X CPU, with a frequency of 4.1 GHz. For CGW-IBE, our keys and ciphertexts are both the smallest. For RWAC, we observe that our ciphertexts are generally the smallest, while the keys are only a little larger than the smallest. For CGW-IBE, we observe that FO key generation is significantly faster than ours, while our decryption is in turn faster than FO. For RWAC, we observe that our decryption is by far the most efficient, i.e., at least a factor 2 than all other variants. Furthermore, the key generation and encryption costs are only milliseconds slower than the most efficient variants. In conclusion, all transformations except for ours incur a significant trade-off: either attaining a large overhead in the key or ciphertext sizes, or incurring a very large overhead in at least one of the algorithms. In contrast, with respect to the decryption algorithm, our transformation outperforms all other transformations, with incredibly little sacrifice in key generation and encryption efficiency.

[9] The code is available at https://github.com/leonbotros/pe_cca.

Table 2: Comparison of the storage and computational costs of the P-KEM part of several CCA-secure variants of CGW-IBE and the fully secure variant of RW13. The storage costs are expressed in bytes and the timings are expressed in milliseconds. The lowest costs are typeset in **bold**, and for 100 attributes, we also include the increase in costs compared to the CPA-secure P-KEM version. For RWAC, we consider inputs of 1, 10 and 100 attributes. (Note that we use compressed point representation to minimize the storage costs.)

Variant	$\lvert\mathrm{MPK}\rvert$	$\lvert\mathrm{SK}_{\mathcal{S}}\rvert$	$\lvert\mathrm{CT}_{\mathbb{A}}\rvert$	KeyGen	Encrypt	Decrypt
CPA	576	448	192	4.10	4.50	1.56
FO	**576**	1024	480	**4.10**	**4.50**	6.06
Ours	672	**576**	**208**	6.13	**4.50**	**4.46**

(a) CGW-IBE [CGW15], the fully secure and anonymous variant of BB-IBE1 [BB04]

Variant	$\lvert\mathrm{MPK}\rvert$	$\lvert\mathrm{SK}_{\mathcal{S}}\rvert$				$\lvert\mathrm{CT}_{\mathbb{A}}\rvert$			
		1	10	100	Increase	1	10	100	Increase
CPA	768	768	4,224	38,784	-	384	2,976	28,896	-
FO	**768**	1,536	4,992	39,552	2%	672	3,264	29,184	1%
Del.	**768**	99,072	102,528	137,088	253%	37,248	39,840	65,760	128%
Ver.	**768**	**768**	**4,224**	**38,784**	0%	672	3,264	29,184	1%
Ours	960	1,152	4,608	39,168	1%	**480**	**3,072**	**29,008**	0.4%

(b) RWAC, the fully secure variant of RW13 [RW13] in AC17 [AC17, Att19] (storage costs)

Variant	KeyGen				Encrypt				Decrypt			
	1	10	100	Increase	1	10	100	Increase	1	10	100	Increase
CPA	8.40	46.2	424	-	6.73	46.6	445	-	3.84	16.6	145	-
FO	8.40	46.2	424	0.4%	**6.73**	**46.6**	**445**	0%	10.6	63.1	590	307%
Del.	1082	1121	1499	255%	573	614	1013	127%	186	200	329	127%
Ver.	**8.37**	**46.0**	**422**	0%	11.1	51.0	449	0.9%	10.5	35.8	292	101%
Ours	12.5	50.1	427	1%	8.21	48.2	448	0.7%	**6.0**	**18.7**	**148**	2%

(c) RWAC, the fully secure variant of RW13 [RW13] in AC17 [AC17, Att19] (computational costs)

7 Future work

Throughout this work, we have mentioned several directions for future work. First, because the second step of the CCA-transformation can be done fully generically, it may be used to convert (decomposable) post-quantum PE as well, e.g., by making an AND-composition of a post-quantum PE and "all-or-one-identity" IBE. Second, we might be able to obtain even more efficient CCA-transformations by using a selectively secure "all-or-one-identity" IBE (Remark 1). Third, it would be interesting to consider whether typical use cases for predicate encryption might also benefit from a signing functionality as proposed in this work. Fourth, although we have considered separate constructions for CCA-security and PE-based signatures, we have not considered how they can be combined securely, e.g., in the spirit of signcryption [Zhe97] or identity-based signcryption [Boy03]. Finally, while this work focuses on predicate encryption, it might be applicable to an even larger class of encryption schemes, e.g., functional encryption [BSW11], which contains PE.

8 Conclusion

We have presented the new notion of predicate extension for predicate encryption schemes. This new notion allows us to achieve CCA-security and PE-based signatures generically.

PE-based signatures are, in a sense, a weaker variant of predicate signatures that do not provide privacy of the signer. This could, however, be considered a feature, because the signer could then not deny having signed some specific message. Furthermore, by considering a two-step approach to achieving CCA-security generically in PE schemes, we aim to convert PE schemes as efficiently as possible. To show this, for each of these steps, we have proposed a new transformation. For the second-step transformation, we have generalized the ACIK-transform [ACIK10], which can now be applied to any PE scheme that is decomposable and for which the predicate can be securely extended. Compared to the more generic CHK- and BK-approaches, ACIK provides less storage overhead and relies on fewer primitives. For the first-step transformation, we have proposed a new predicate-extension transformation that can be applied to any pairing-based schemes that can be captured in the pair and predicate encodings frameworks. Compared to existing (implicitly-described) predicate-extension techniques, ours is more efficient. Notably, for CP-ABE, existing such techniques are very inefficient. To show that our predicate-extension transformation indeed yields interesting improvements on existing ones, we have implemented two schemes. The results show that, especially for linear-sized predicates, our transformations provide a significant improvement in efficiency.

Acknowledgments. The authors would like to thank Eike Kiltz for helping out with a part of the security proof, Greg Alpár for proofreading the paper, and Tobias Handirk and Doreen Riepel for helpful discussions. This work is partly funded by the NWO under the project "Encryption for all" (project number CS.002) of the research programme "Topsectoren 18-19 Cybersecurity", which is (partly) financed by the Dutch Research Council (NWO).

References

[ABGW17] M. Ambrona, G. Barthe, R. Gay, and H. Wee. Attribute-based encryption in the generic group model: Automated proofs and new constructions. In B. M. Thuraisingham, D. Evans, T. Malkin, and D. Xu, editors, *CCS*, pages 647–664. ACM, 2017.

[ABS17] M. Ambrona, G. Barthe, and B. Schmidt. Generic transformations of predicate encodings: Constructions and applications. In J. Katz and H. Shacham, editors, *CRYPTO*, volume 10401 of *LNCS*, pages 36–66. Springer, 2017.

[ABV+12] S. Agrawal, X. Boyen, V. Vaikuntanathan, P. Voulgaris, and H. Wee. Functional encryption for threshold functions (or fuzzy IBE) from lattices. In M. Fischlin, J. Buchmann, and M. Manulis, editors, *PKC*, volume 7293 of *LNCS*, pages 280–297. Springer, 2012.

[AC16] S. Agrawal and M. Chase. A study of pair encodings: Predicate encryption in prime order groups. In E. Kushilevitz and T. Malkin, editors, *TCC*, volume 9563 of *LNCS*, pages 259–288. Springer, 2016.

[AC17] S. Agrawal and M. Chase. Simplifying design and analysis of complex predicate encryption schemes. In J.-S. Coron and J. B. Nielsen, editors, *EUROCRYPT*, volume 10210 of *LNCS*, pages 627–656. Springer, 2017.

[ACIK10] M. Abe, Y. Cui, H. Imai, and E. Kiltz. Efficient hybrid encryption from ID-based encryption. *Des. Codes Cryptogr.*, 54(3):205–240, 2010.

[AFV11] S. Agrawal, D. M. Freeman, and V. Vaikuntanathan. Functional encryption for inner product predicates from learning with errors. In D. H. Lee and

X. Wang, editors, *ASIACRYPT*, volume 7073 of *LNCS*, pages 21–40. Springer, 2011.

[AHY15] N. Attrapadung, G. Hanaoka, and S. Yamada. Conversions among several classes of predicate encryption and applications to ABE with various compactness tradeoffs. In T. Iwata and J. H. Cheon, editors, *ASIACRYPT*, volume 9452 of *LNCS*, pages 575–601. Springer, 2015.

[AI09] N. Attrapadung and H. Imai. Conjunctive broadcast and attribute-based encryption. In H. Shacham and B. Waters, editors, *Pairing*, volume 5671 of *LNCS*, pages 248–265. Springer, 2009.

[Amb21] M. Ambrona. Generic negation of pair encodings. In J. A. Garay, editor, *PKC*, volume 12711 of *LNCS*, pages 120–146. Springer, 2021.

[Att14] N. Attrapadung. Dual system encryption via doubly selective security: Framework, fully secure functional encryption for regular languages, and more. In P. Q. Nguyen and E. Oswald, editors, *EUROCRYPT*, volume 8441 of *LNCS*, pages 557–577. Springer, 2014.

[Att16] N. Attrapadung. Dual system encryption framework in prime-order groups via computational pair encodings. In J. H. Cheon and T. Takagi, editors, *ASIACRYPT*, volume 10032 of *LNCS*, pages 591–623. Springer, 2016.

[Att19] N. Attrapadung. Unbounded dynamic predicate compositions in attribute-based encryption. In Y. Ishai and V. Rijmen, editors, *EUROCRYPT*, volume 11476 of *LNCS*, pages 34–67. Springer, 2019.

[BB04] D. Boneh and X. Boyen. Efficient selective-ID secure identity-based encryption without random oracles. In C. Cachin and J. Camenisch, editors, *EUROCRYPT*, volume 3027 of *LNCS*, pages 223–238. Springer, 2004.

[BCHK07] D. Boneh, R. Canetti, S. Halevi, and J. Katz. Chosen-ciphertext security from identity-based encryption. *SIAM J. Comput.*, 36(5):1301–1328, 2007.

[Bei96] A. Beimel. *Secure Schemes for Secret Sharing and Key Distribution.* Phd thesis, Ben Gurion University, 1996.

[BF01] D. Boneh and M. K. Franklin. Identity-based encryption from the weil pairing. In J. Kilian, editor, *CRYPTO*, volume 2139 of *LNCS*, pages 213–229. Springer, 2001.

[BFM88] M. Blum, P. Feldman, and S. Micali. Non-interactive zero-knowledge and its applications. In J. Simon, editor, *Proceedings of the 20th Annual ACM Symposium on Theory of Computing*, pages 103–112. ACM, 1988.

[BK05] D. Boneh and J. Katz. Improved efficiency for CCA-secure cryptosystems built using identity-based encryption. In Alfred Menezes, editor, *CT-RSA*, volume 3376 of *LNCS*, pages 87–103. Springer, 2005.

[BL16] J. Blömer and G. Liske. Construction of fully cca-secure predicate encryptions from pair encoding schemes. In K. Sako, editor, *CT-RSA*, volume 9610 of *LNCS*, pages 431–447. Springer, 2016.

[Boy03] X. Boyen. Multipurpose identity-based signcryption (A swiss army knife for identity-based cryptography). In D. Boneh, editor, *CRYPTO*, volume 2729 of *LNCS*, pages 383–399. Springer, 2003.

[Boy13] X. Boyen. Attribute-based functional encryption on lattices. In A. Sahai, editor, *TCC*, volume 7785 of *LNCS*, pages 122–142. Springer, 2013.

[BSW07] J. Bethencourt, A. Sahai, and B. Waters. Ciphertext-policy attribute-based encryption. In *S&P*, pages 321–334. IEEE, 2007.

[BSW11] D. Boneh, A. Sahai, and B. Waters. Functional encryption: Definitions and challenges. In Y. Ishai, editor, *TCC*, volume 6597 of *LNCS*, pages 253–273. Springer, 2011.

[BW06] X. Boyen and B. Waters. Anonymous hierarchical identity-based encryption (without random oracles). In C. Dwork, editor, *CRYPTO*, volume 4117 of *LNCS*, pages 290–307. Springer, 2006.

[BW07] D. Boneh and B. Waters. Conjunctive, subset, and range queries on encrypted data. In S. P. Vadhan, editor, *TCC*, volume 4392 of *LNCS*, pages 535–554. Springer, 2007.

[CGW15] J. Chen, R. Gay, and H. Wee. Improved dual system ABE in prime-order groups via predicate encodings. In E. Oswald and M. Fischlin, editors, *EUROCRYPT*, volume 9057 of *LNCS*, pages 595–624. Springer, 2015.

[CHK04] R. Canetti, S. Halevi, and J. Katz. Chosen-ciphertext security from identity-based encryption. In C. Cachin and J. Camenisch, editors, *EUROCRYPT*, volume 3027 of *LNCS*, pages 207–222. Springer, 2004.

[CS03] R. Cramer and V. Shoup. Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. *SIAM J. Comput.*, 33(1):167–226, 2003.

[CW14] J. Chen and H. Wee. Dual system groups and its applications — compact HIBE and more. Cryptology ePrint Archive, Report 2014/265, 2014.

[dlPVA23] A. de la Piedra, M. Venema, and G. Alpár. ACABELLA: automated (crypt)analysis of attribute-based encryption leveraging linear algebra. *To appear at CCS*, 2023.

[FO99] E. Fujisaki and T. Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In M. J. Wiener, editor, *CRYPTO*, volume 1666 of *LNCS*, pages 537–554. Springer, 1999.

[Gen06] C. Gentry. Practical identity-based encryption without random oracles. In S. Vaudenay, editor, *EUROCRYPT*, volume 4004 of *LNCS*, pages 445–464. Springer, 2006.

[GMC08] F. Guo, Y. Mu, and Z. Chen. Identity-based online/offline encryption. In Gene Tsudik, editor, *FC*, volume 5143 of *LNCS*, pages 247–261. Springer, 2008.

[GPSW06a] V. Goyal, O. Pandey, A. Sahai, and B. Waters. Attribute-based encryption for fine-grained access control of encrypted data. In A. Juels, R. N. Wright, and S. De Capitani di Vimercati, editors, *CCS*. ACM, 2006.

[GPSW06b] V. Goyal, O. Pandey, A. Sahai, and B. Waters. Attribute-based encryption for fine-grained access control of encrypted data. Cryptology ePrint Archive, Report 2006/309, 2006.

[GPV08] C. Gentry, C. Peikert, and V. Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In C. Dwork, editor, *STOC*, pages 197–206. ACM, 2008.

[GVW13] S. Gorbunov, V. Vaikuntanathan, and H. Wee. Attribute-based encryption for circuits. In D. Boneh, T. Roughgarden, and J. Feigenbaum, editors, *STOC*, pages 545–554. ACM, 2013.

[HHK17] D. Hofheinz, K. Hövelmanns, and E. Kiltz. A modular analysis of the Fujisaki-Okamoto transformation. In Y. Kalai and L. Reyzin, editors, *TCC*, volume 10677 of *LNCS*, pages 341–371. Springer, 2017.

[HW14] S. Hohenberger and B. Waters. Online/offline attribute-based encryption. In Hugo Krawczyk, editor, *PKC*, volume 8383 of *LNCS*, pages 293–310. Springer, 2014.

[KG06] E. Kiltz and D. Galindo. Direct chosen-ciphertext secure identity-based key encapsulation without random oracles. In L. M. Batten and R. Safavi-Naini, editors, *ACISP*, volume 4058 of *LNCS*, pages 336–347. Springer, 2006.

[KSW08] J. Katz, A. Sahai, and B. Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In N. P. Smart, editor, *EUROCRYPT*, volume 4965 of *LNCS*, pages 146–162. Springer, 2008.

[KV08] E. Kiltz and Y. Vahlis. CCA2 secure IBE: standard model efficiency through authenticated symmetric encryption. In Tal Malkin, editor, *CT-RSA*, volume 4964 of *LNCS*, pages 221–238. Springer, 2008.

[KW19] V. Koppula and B. Waters. Realizing chosen ciphertext security generically in attribute-based encryption and predicate encryption. In A. Boldyreva and D. Micciancio, editors, *CRYPTO*, volume 11693 of *LNCS*, pages 671–700. Springer, 2019.

[LW10] A. Lewko and B. Waters. Decentralizing attribute-based encryption. Cryptology ePrint Archive, Report 2010/351, 2010.

[MPR11] H. K. Maji, M. Prabhakaran, and M. Rosulek. Attribute-based signatures. In A. Kiayias, editor, *CT-RSA*, volume 6558 of *LNCS*, pages 376–392. Springer, 2011.

[NY90] M. Naor and M. Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In H. Ortiz, editor, *STOC*, pages 427–437. ACM, 1990.

[OT11] T. Okamoto and K. Takashima. Efficient attribute-based signatures for non-monotone predicates in the standard model. In D. Catalano, N. Fazio, R. Gennaro, and A. Nicolosi, editors, *PKC*, volume 6571 of *LNCS*, pages 35–52. Springer, 2011.

[Rog02] P. Rogaway. Authenticated-encryption with associated-data. In V. Atluri, editor, *CCS*, pages 98–107. ACM, 2002.

[RW13] Y. Rouselakis and B. Waters. Practical constructions and new proof methods for large universe attribute-based encryption. In A.-R. Sadeghi, V. D. Gligor, and M. Yung, editors, *CCS*, pages 463–474. ACM, 2013.

[Sha84] A. Shamir. Identity-based cryptosystems and signature schemes. In G. R. Blakley and D. Chaum, editors, *CRYPTO*, volume 196 of *LNCS*, pages 47–53. Springer, 1984.

[Sho98] V. Shoup. Why chosen ciphertext security matters. IBM Research Report RZ 3076, November 1998.

[SW05] A. Sahai and B. Waters. Fuzzy identity-based encryption. In R. Cramer, editor, *EUROCRYPT*, volume 3494 of *LNCS*, pages 457–473. Springer, 2005.

[TKN21] J. Tomida, Y. Kawahara, and R. Nishimaki. Fast, compact, and expressive attribute-based encryption. *Des. Codes Cryptogr.*, 89(11):2577–2626, 2021.

[VA23] M. Venema and G. Alpár. GLUE: generalizing unbounded attribute-based encryption for flexible efficiency trade-offs. In A. Boldyreva and V. Kolesnikov, editors, *PKC*, volume 13940 of *LNCS*, pages 652–682. Springer, 2023.

[Ven23] M. Venema. A practical compiler for attribute-based encryption: New decentralized constructions and more. In M. Rosulek, editor, *CT-RSA*, volume 13871 of *LNCS*, pages 132–159. Springer, 2023.

[Wee14] H. Wee. Dual system encryption via predicate encodings. In Y. Lindell, editor, *TCC*, volume 8349 of *LNCS*, pages 616–637. Springer, 2014.

[YAHK11] S. Yamada, N. Attrapadung, G. Hanaoka, and N. Kunihiro. Generic constructions for chosen-ciphertext secure attribute based encryption. In D. Catalano, N. Fazio, R. Gennaro, and A. Nicolosi, editors, *PKC*, volume 6571 of *LNCS*, pages 71–89. Springer, 2011.

[YAS+12] S. Yamada, N. Attrapadung, B. Santoso, J. C. N. Schuldt, G. Hanaoka, and N. Kunihiro. Verifiable predicate encryption and applications to CCA security and anonymous predicate authentication. In M. Fischlin, J. Buchmann, and M. Manulis, editors, *PKC*, volume 7293 of *LNCS*, pages 243–261. Springer, 2012.

[Zhe97] Y. Zheng. Digital signcryption or how to achieve cost(signature & encryption) << cost(signature) + cost(encryption). In B. S. Kaliski Jr., editor, *CRYPTO*, volume 1294 of *LNCS*, pages 165–179. Springer, 1997.

[ZkC20] ZkCrypto. Zero-knowledge cryptography in Rust. https://github.com/zkcrypto, 2020.

A Predicate key encapsulation

In the key-encapsulation variant of predicate encryption (Definition 1), which we call predicate KEM (P-KEM), we replace Encrypt by Encaps and Decrypt by Decaps, where Encaps also outputs a symmetric key, and Decaps outputs a symmetric key instead of a plaintext message. This symmetric key is used to symmetrically encrypt the data.

Definition 22 (Predicate key-encapsulation mechanism (P-KEM))**.** A predicate key-encapsulation mechanism for a predicate family $P = \{P_\kappa\}_{\kappa\in\mathbb{N}^c}$ over a message space $\mathcal{M} = \{M_\lambda\}_{\lambda\in\mathbb{N}}$ consists of four algorithms:

- $\mathrm{Setup}(\lambda, \mathrm{par}) \to (\mathrm{MPK}, \mathrm{MSK})$: On input the security parameter λ and parameters par, this probabilistic algorithm generates the domain parameters, the master public key MPK and the master secret key MSK. In addition, κ is set to $\kappa = (p, \mathrm{par})$, where p denotes a natural number.

- $\mathrm{KeyGen}(\mathrm{MSK}, y) \to \mathrm{SK}_y$: On input the master secret key MSK and some $y \in \mathcal{Y}_\kappa$, this probabilistic algorithm generates a secret key SK_y.
- $\mathrm{Encaps}(\mathrm{MPK}, x) \to (\mathrm{K}, \mathrm{CT}_x)$: On input the master public key MPK and some $x \in \mathcal{X}_\kappa$, this probabilistic algorithm generates an encapsulated symmetric key K and a ciphertext CT_x.
- $\mathrm{Decaps}(\mathrm{MPK}, \mathrm{SK}_y, \mathrm{CT}_x) \to \mathrm{K}$: On input the master public key MPK, the secret key SK_y, and the ciphertext CT_x, if $P_\kappa(x, y) = 1$, then it returns the encapsulated symmetric key K. Otherwise, it returns an error message $\perp$.

Correctness. For all par, $M \in \mathcal{M}_\lambda$, $x \in \mathcal{X}_\kappa$, and $y \in \mathcal{Y}_\kappa$ such that $P_\kappa(x, y) = 1$,

$$\Pr[(\mathrm{MPK}, \mathrm{MSK}) \leftarrow \mathrm{Setup}(1^\lambda); (\mathrm{K}, \mathrm{CT}_x) \leftarrow \mathrm{Encaps}(\mathrm{MPK}, x); \\ \mathrm{Decaps}(\mathrm{MPK}, \mathrm{KeyGen}(\mathrm{MSK}, y)), \mathrm{CT}_x) \neq \mathrm{K}] \leq \mathrm{negl}(\lambda).$$

Full security against chosen-plaintext attacks. The full security model for P-KEM is defined similarly as that for PE (Definition 2). The crucial difference between the two is that the goal of the attacker is to distinguish a symmetric key produced by the encapsulation algorithm from a randomly generated key.

Definition 23 (CPA-security for P-KEM)**.** We define the security game IND-CPA(λ) between challenger and attacker as follows:

- **Setup phase:** The challenger runs $\mathrm{Setup}(\lambda)$ to obtain MPK and MSK, and sends the master public key MPK to the attacker.
- **First query phase:** The attacker queries secret keys for $y \in \mathcal{Y}$, and obtains $\mathrm{SK}_y \leftarrow \mathrm{KeyGen}(\mathrm{MSK}, y)$ in response.
- **Challenge phase:** The attacker specifies some $x^* \in \mathcal{X}$ such that for all y in the first key query phase, we have $P(x^*, y) = 0$, and sends these to the challenger. The challenger first encapsulates a key under x^*, i.e., $(\mathrm{K}^*, \mathrm{CT}_{x^*}) \leftarrow \mathrm{Encaps}(\mathrm{MPK}, x^*)$, and then flips a coin $\beta \in_R \{0, 1\}$. If $\beta = 0$, the key K^* is replaced by a value that is selected uniformly at random from the key space. The challenger then sends the resulting encapsulation key K^* and ciphertext CT_{x^*} to the attacker.
- **Second query phase:** This phase is identical to the first query phase, with the additional restriction that the attacker can only query $y \in \mathcal{Y}$ such that $P(x^*, y) = 0$.
- **Decision phase:** The attacker outputs a guess β' for β.

The advantage of the attacker is defined as $\mathsf{Adv}_{\text{P-KEM,IND-CPA}} = |\Pr[\beta' = \beta] - \frac{1}{2}|$. A scheme is fully secure if all polynomial-time attackers have at most a negligible advantage in this security game, i.e., $\mathsf{Adv}_{\text{P-KEM,IND-CPA}} \leq \mathrm{negl}(\lambda)$.

In the selective security model, the attacker commits to the predicate $x^* \in \mathcal{X}$ before the Setup phase.

B More efficient transformation from PE to P-KEM

Let $\Gamma_{\mathrm{PE}} = (\mathrm{Setup}, \mathrm{KeyGen}, \mathrm{Encaps}, \mathrm{Decaps})$ be a decomposable predicate encryption scheme for the predicate family $P = \{P_\kappa\}_\kappa$ with $P_\kappa \colon \mathcal{X}_\kappa \times \mathcal{Y}_\kappa \to \{0, 1\}$. Suppose that the operation on the group in which CT_M lives is multiplicative[10] and its operator is $\cdot$, and

[10] Something similar works for other algebraic groups such as additive groups as well.

in particular, that $\mathrm{CT_M} = M \cdot \mathrm{rand}$, where rand is some random element in the group in which $\mathrm{CT_M}$ lives. Let id denote the identity in this group. Then, we can generically define Encaps and Decaps from Encrypt and Decrypt as follows.

- Encaps(MPK, x): Let $(\mathrm{CT_M}, \mathrm{CT}_1, \mathrm{CT}_{2,x}) \leftarrow \mathrm{Encrypt}(\mathrm{MPK}, x, \mathrm{id})$. Then, this algorithm outputs $\mathrm{K} = \mathrm{CT_M}$ as the symmetric key and $(\mathrm{CT}_1, \mathrm{CT}_{2,x})$ as the rest of the ciphertext.
- Decaps$(\mathrm{MPK}, \mathrm{SK}_y, \mathrm{CT}_x)$: This algorithm outputs the decapsulated symmetric key as $\mathrm{K}' \leftarrow \mathrm{Decrypt}(\mathrm{MPK}, \mathrm{SK}_y, (\mathrm{id}, \mathrm{CT}_1, \mathrm{CT}_{2,x}))^{-1}$.

The correctness of the P-KEM follows from the correctness of the PE:

$$\begin{aligned}\mathrm{Decaps}(\mathrm{MPK}, \mathrm{SK}_y, \mathrm{CT}_x) &= \mathrm{Decrypt}(\mathrm{MPK}, \mathrm{SK}_y, (\mathrm{id}, \mathrm{CT}_1, \mathrm{CT}_{2,x}))^{-1} \\ &= \mathrm{K} \cdot \mathrm{Decrypt}(\mathrm{MPK}, \mathrm{SK}_y, (\mathrm{id} \cdot \mathrm{K}, \mathrm{CT}_1, \mathrm{CT}_{2,x}))^{-1} \\ &= \mathrm{K} \cdot \mathrm{Decrypt}(\mathrm{MPK}, \mathrm{SK}_y, \mathrm{Encrypt}(\mathrm{MPK}, x, \mathrm{id}))^{-1} \\ &= \mathrm{K} \cdot \mathrm{id}^{-1} = \mathrm{K}.\end{aligned}$$

The CPA-security of the P-KEM also follows readily from the PE. Let $\mathcal{A}_{\text{P-KEM}}$ be an attacker on the P-KEM, i.e., which can distinguish for a given $(\mathrm{K}, \mathrm{CT}_1, \mathrm{CT}_{2,x})$ whether K is a symmetric key or K is random. Then, it can be used to construct an attacker $\mathcal{A}_{\text{PE}}$ for the PE scheme. Suppose $(\mathrm{CT_M}, \mathrm{CT}_1, \mathrm{CT}_{2,x})$ is the challenge ciphertext for M_0 or M_1. Then, pick $\beta \in_R \{0, 1\}$ and send $(\mathrm{K} = \mathrm{CT_M}/M_\beta, \mathrm{CT}_1, \mathrm{CT}_{2,x})$ to attacker $\mathcal{A}_{\text{P-KEM}}$. If it outputs that K is a symmetric key, then attacker $\mathcal{A}_{\text{PE}}$ outputs β as the guess, and otherwise, it outputs $1 - \beta$ as the guess. The advantage of $\mathcal{A}_{\text{P-KEM}}$ is equal to the advantage of $\mathcal{A}_{\text{PE}}$.

C The transformations preserve attribute-hiding

We show that our transformations preserve the attribute-hiding property [BW07, KSW08] (which includes anonymous IBE [BW06] as a special case). In anonymous/attribute-hiding PE, the attribute x of the ciphertext is hidden, and cannot be inferred from the ciphertext either. Intuitively, the reasoning behind why our transformations preserve this property is simple. Because the extended-predicate functionality is independent of the original predicate functionality and does not reveal any additional information about the original predicate, the transformed scheme is also anonymous or attribute-hiding. More formally, we prove this by reducing the anonymity/attribute-hiding security of the resulting scheme to the original scheme. To this end, we first give a definition of weakly attribute-hiding PE. Then, we show how a scheme is created from a predicate encoding, and what the original and resulting scheme looks like.

C.1 Attribute-hiding PE

Definition 24 (Attribute-hiding and fully CPA-secure PE [CGW15])**.** Let $\Gamma = (\mathrm{Setup}, \mathrm{KeyGen}, \mathrm{Encrypt}, \mathrm{Decrypt})$ be a PE scheme for predicate P. We define the security game IND-CPA-AH(λ, par) between challenger and attacker as follows:

- **Setup phase:** The challenger runs Setup(λ) to obtain MPK and MSK, and sends the master public key MPK to the attacker.
- **First query phase:** The attacker queries secret keys for $y \in \mathcal{Y}_\kappa$, and obtains $\mathrm{SK}_y \leftarrow \mathrm{KeyGen}(\mathrm{MSK}, y)$ in response.
- **Challenge phase:** The attacker specifies some $x_0^*, x_1^* \in \mathcal{X}_\kappa$ such that for all y in the first key query phase, we have $P(x_0^*, y) = P(x_1^*, y) = 0$, and generates two

messages M_0 and M_1 of equal length in $\mathcal{M}_\lambda$, and sends these to the challenger. The challenger flips a coin, i.e., $\beta \in_R \{0,1\}$, encrypts M_β under x^*_β, i.e., $\mathrm{CT}_{x^*_\beta} \leftarrow \mathrm{Encrypt}(\mathrm{MPK}, x^*_\beta, M_\beta)$, and sends the resulting ciphertext $\mathrm{CT}_{x^*_\beta}$ to the attacker.

- **Second query phase:** This phase is identical to the first query phase, with the additional restriction that the attacker can only query $y \in \mathcal{Y}_\kappa$ such that $P(x^*_0, y) = P(x^*_1, y) = 0$.
- **Decision phase:** The attacker outputs a guess β' for β.

The advantage of the attacker is defined as $\mathsf{Adv}_{\mathrm{PE,IND\text{-}CPA\text{-}AH}} = |\Pr[\beta' = \beta] - \frac{1}{2}|$. A scheme is fully secure and attribute-hiding if all polynomial-time attackers have at most a negligible advantage in this security game, i.e., we have $\mathsf{Adv}_{\mathrm{PE,IND\text{-}CPA\text{-}AH}} \leq \mathrm{negl}(\lambda)$.

In the selective security model, the attacker commits to the predicate $x^* \in \mathcal{X}_\kappa$ before the Setup phase.

C.2 Generic compiler from dual system groups

Chen, Gay and Wee [CGW15] devised a generic compiler that transforms any predicate encoding into a fully secure PE using dual system groups (DSG) [CW14] from k-Lin. We will consider their specific instantiation for $k = 1$, i.e., SXDH, which is the most efficient.

Notation. Given $a \in \mathbb{Z}_p$, we use $[a]_1$ to denote g^a, $[a]_2$ to denote h^a and $[a]_T$ to denote $e(g,h)^a$. This extends to vectors and matrices in an obvious way, e.g., $[(a_1, a_2, ...)]_1$ denotes $(g^{a_1}, g^{a_2}, ...)$. We define $e([\mathbf{A}]_1, [\mathbf{B}]_2) = [\mathbf{A}^\intercal \mathbf{B}]_T$. Let $\mathcal{D}_1$ denote the distribution over matrices $\mathbf{A} = \begin{pmatrix} a_1 \\ 1 \end{pmatrix}$, where $a_1 \in_R \mathbb{Z}_p$.

Definition 25 (Generic compiler for DSGs from SXDH [CGW15])**.** Let $\Gamma = (\mathrm{sE}, \mathrm{rE}, \mathrm{kE}, \mathrm{sD}, \mathrm{rD})$ be a predicate encoding as in Definition 11.

- Setup(λ): On input the security parameter λ, the PKG first generates domain parameters $(p, \mathbb{G}, \mathbb{H}, \mathbb{G}_T, g, h, e)$. Then, it generates $k_1, k_2 \in \mathbb{Z}_p$, $\mathbf{A}, \mathbf{B} \in \mathcal{D}_1$, and for each entry w_i in vector $\mathbf{w} \in \mathbb{Z}_p^n$, it generates $\mathbf{W} \in_R \mathbb{Z}_p^{2\times 2}$. It sets $\mathrm{MSK} = (\mathbf{k} = \begin{pmatrix} k_1 \\ k_2 \end{pmatrix}, \mathbf{A}, \mathbf{B}, \mathbf{W})$, and outputs the master public key
$$\mathrm{MPK} = (A = e(g,h)^{k_1 a_1 + k_2}, [\mathbf{A}]_1, [\mathbf{B}]_2, \{[\mathbf{W}_i^\intercal \mathbf{A}]_1, [\mathbf{W}_i \mathbf{B}]_2\}_{i\in[n]}).$$
- KeyGen(MSK, y): On input the master secret key MSK and some $y \in \mathcal{Y}_\kappa$, the PKG generates $r \in_R \mathbb{Z}_p$, and output
$$\mathrm{SK}_y = (\mathbf{K} = [\mathbf{B}r]_2, \mathbf{K}' = \mathrm{kE}(y, [\mathbf{k}]_2) \cdot \mathrm{rE}(y, [\mathbf{W}_1 \mathbf{B}r, ..., \mathbf{W}_n \mathbf{B}r]_2))$$
- Encrypt(MPK, x, M): On input the master public key MPK, some $x \in \mathcal{X}_\kappa$ and message M, it generates $s \in_R \mathbb{Z}_p$, and outputs
$$\mathrm{CT}_x = (C = M \cdot A^s, \mathbf{C}' = [\mathbf{A}s]_1, \mathbf{C}'' = \mathrm{sE}(x, [\mathbf{W}_1^\intercal \mathbf{A}s, ..., \mathbf{W}_n^\intercal \mathbf{A}s]_1))$$
- Decrypt($\mathrm{MPK}, \mathrm{SK}_y, \mathrm{CT}_x$): On input the master public key MPK, the secret key SK_y, and the ciphertext CT_x, if $P(x,y) = 1$, then the message can be obtained as
$$M' = C / \left(e(\mathbf{C}', \mathrm{rD}(x, y, \mathbf{K}'))/e(\mathrm{sD}(x, y, \mathbf{C}''), \mathbf{K})\right).$$

C.3 The security proof

Proposition 1. *Let* $\Gamma = (\mathrm{sE}, \mathrm{rE}, \mathrm{kE}, \mathrm{sD}, \mathrm{rD})$ *be a predicate encoding such that the associated PE scheme* $\Psi = (\mathrm{Setup}, \mathrm{KeyGen}, \mathrm{Encrypt}, \mathrm{Decrypt})$ *is attribute-hiding. Then, the PE scheme* Ψ' *associated with the predicate encoding* $\Gamma' = (\mathrm{sE}', \mathrm{rE}', \mathrm{kE}', \mathrm{sD}', \mathrm{rD}')$ *that follows with the transformations in Definitions 21 and 13 is also attribute-hiding.*

Proof. Let $\mathcal{A}_{\Psi'}$ denote the attacker that can break the attribute-hiding property of scheme Ψ' with advantage $\mathsf{Adv}'_{\mathrm{PE,IND\text{-}CPA\text{-}AH}}$. We use it to construct an attacker $\mathcal{A}_{\Psi}$ that can break the attribute-hiding property of scheme Ψ. Let $\mathcal{C}_{\Psi'}$ and $\mathcal{C}_{\Psi}$ denote the respective challengers of attackers $\mathcal{A}_{\Psi'}$ and $\mathcal{A}_{\Psi}$.

- **Setup phase:** Challenger $\mathcal{C}_{\Psi}$ runs the setup algorithm, returning

$$\mathrm{MPK} = (A = e(g,h)^{k_1 a_1 + k_2}, [\mathbf{A}]_1, [\mathbf{B}]_2, \{[\mathbf{W}_i^\intercal \mathbf{A}]_1, [\mathbf{W}_i \mathbf{B}]_2\}_{i \in [n]})$$

  to attacker $\mathcal{A}_{\Psi}$. Challenger then selects target-collision resistant hash TCR and authenticated encryption scheme $\mathrm{SE} = (\mathrm{Enc_K}, \mathrm{Dec_K})$, generates $\mathbf{W}_{n+1}, \mathbf{W}_{n+2}$, $\mathbf{W}_{n+3} \in_R \mathbb{Z}_p^{2\times 2}$, computes $[\mathbf{W}_i^\intercal \mathbf{A}]_1$ and $[\mathbf{W}_i \mathbf{B}]_2$ from $[\mathbf{A}]_1$ and $[\mathbf{B}]_2$, i.e.

$$[\mathbf{W}_i^\intercal \mathbf{A}]_1 = \begin{pmatrix} [a_1]_1^{w_{11}} \cdot [1]_1^{w_{21}} \\ [a_1]_1^{w_{12}} \cdot [1]_1^{w_{22}} \end{pmatrix} \text{ and } [\mathbf{W}_i \mathbf{B}]_2 = \begin{pmatrix} [b_1]_2^{w_{11}} \cdot [1]_2^{w_{12}} \\ [b_1]_2^{w_{21}} \cdot [1]_2^{w_{22}} \end{pmatrix},$$

  and returns to attacker $\mathcal{A}_{\Psi'}$:

$$\mathrm{MPK}' = (A, [\mathbf{A}]_1, [\mathbf{B}]_2, \{[\mathbf{W}_i^\intercal \mathbf{A}]_1, [\mathbf{W}_i \mathbf{B}]_2\}_{i \in [n+3]}).$$

- **First query phase:** For each $y \in \mathcal{Y}_\kappa$ for which attacker $\mathcal{A}_{\Psi'}$ queries a secret key, attacker $\mathcal{A}_{\Psi}$ queries challenger $\mathcal{C}_{\Psi}$ for a secret key:

$$\mathrm{SK}_y = (\mathbf{K} = [\mathbf{B}r]_2, \mathbf{K}' = \mathrm{kE}(y, [\mathbf{k}]_2) \cdot \mathrm{rE}(y, [\mathbf{W}_1 \mathbf{B}r, ..., \mathbf{W}_n \mathbf{B}r]_2)),$$

  which is used to construct

$$\mathrm{SK}'_y = (\mathbf{K} = [\mathbf{B}r]_2, \mathbf{K}'' = \mathrm{kE}'((y,*), [\mathbf{k}]_2) \cdot \mathrm{rE}'((y,*), [\mathbf{W}_1 \mathbf{B}r, ..., \mathbf{W}_{n+3} \mathbf{B}r]_2)).$$

  In particular, note that $\mathrm{kE}''(*, [\mathbf{k}]_2)$ can be computed trivially from SK_y, and $\mathrm{rE}''(*, [\mathbf{W}_1 \mathbf{B}r, ..., \mathbf{W}_{n+3} \mathbf{B}r]_2)$ can be computed by using that

$$[\mathbf{W}_i \mathbf{B}r] = \begin{pmatrix} [b_1 r]_2^{w_{11}} \cdot [r]_2^{w_{12}} \\ [b_1 r]_2^{w_{21}} \cdot [r]_2^{w_{22}} \end{pmatrix}.$$

- **Challenge phase:** At some point, attacker $\mathcal{A}_{\Psi'}$ sends a message $M \in \{0,1\}^*$ and two predicates $x_0^*, x_1^* \in \mathcal{X}_\kappa$. Attacker $\mathcal{A}_{\Psi}$ sends x_0^*, x_1^* and $M' \in_R \mathbb{G}_T$ to $\mathcal{C}_{\Psi}$, who flips a coin $\beta \in_R \{0,1\}$ and returns

$$\mathrm{CT}^*_{x^*_\beta} = (C = M' \cdot A^s, \mathbf{C}' = [\mathbf{A}s]_1, \mathbf{C}'' = \mathrm{sE}(x, [\mathbf{W}_1^\intercal \mathbf{A}s, ..., \mathbf{W}_n^\intercal \mathbf{A}s]_1)).$$

  This is used to construct

$$\bar{\mathrm{CT}}^*_{x^*_\beta} = (\mathrm{CT}^*_{\mathrm{sym}}, \mathbf{C}', \bar{\mathbf{C}}'' = \mathrm{sE}'((x^*_\beta, x'), [\mathbf{W}_1^\intercal \mathbf{A}s, ..., \mathbf{W}_n^\intercal \mathbf{A}s]_1),$$

  where $\mathrm{CT}^*_{\mathrm{sym}} \leftarrow \mathrm{Enc}_{C/M'}(M)$, $x' \leftarrow \mathrm{TCR}([a_1 s]_1)$, and $\mathrm{sE}''(x', [\mathbf{W}_1^\intercal \mathbf{A}s, ..., \mathbf{W}_n^\intercal \mathbf{A}s]_1)$ is generated from $\mathrm{CT}^*_{x^*_\beta}$ in a similar way as in the key generation:

$$[\mathbf{W}_i^\intercal \mathbf{A}s]_1 = \begin{pmatrix} [a_1 s]_1^{w_{11}} \cdot [s]_1^{w_{21}} \\ [a_1 s]_1^{w_{12}} \cdot [s]_1^{w_{22}}. \end{pmatrix}$$

- **Second query phase:** This phase is identical to the first query phase.
- **Decision phase:** Attacker $\mathcal{A}_{\Psi'}$ outputs a guess β' for β, which attacker $\mathcal{A}_{\Psi}$ also outputs as its guess.

The advantage $\mathsf{Adv}_{\mathrm{PE,IND\text{-}CPA\text{-}AH}}$ of attacker $\mathcal{A}_{\Psi}$ is equal to the advantage of attacker $\mathcal{A}_{\Psi'}$: $\mathsf{Adv}_{\mathrm{PE,IND\text{-}CPA\text{-}AH}} = \mathsf{Adv}'_{\mathrm{PE,IND\text{-}CPA\text{-}AH}}$. □

D An anonymous IBE scheme

D.1 Identity-based encryption

A special case of predicate encryption is identity-based encryption.

Definition 26 (Identity-based encryption (IBE) [Sha84, BF01])**.** An identity-based encryption scheme consists of four algorithms:

- Setup(λ): On input the security parameter λ, this probabilistic algorithm, performed by the Private Key Generator (PKG), generates the domain parameters, the master public key MPK and the master secret key MSK. The master public key and domain parameters are published, while the master secret key is kept secret by the PKG.
- KeyGen(MSK, ID): On input the master secret key and some identifier $\mathsf{ID} \in \{0,1\}^*$, this probabilistic algorithm, performed by the PKG, generates a secret key $\mathrm{SK}_{\mathsf{ID}}$ for identifier ID.
- Encrypt(MPK, ID, M): On input the master public key, identifier $\mathsf{ID} \in \{0,1\}^*$ and message M, this probabilistic algorithm generates a ciphertext $\mathrm{CT}_{\mathsf{ID}}$.
- Decrypt(MPK, $\mathrm{CT}_{\mathsf{ID}}, \mathrm{SK}_{\mathsf{ID}'}$): On input the ciphertext $\mathrm{CT}_{\mathsf{ID}}$ for identifier ID and secret key $\mathrm{SK}_{\mathsf{ID}'}$ for identifier ID', if $\mathsf{ID} = \mathsf{ID}'$, then it returns M. Otherwise, it returns an error message $\perp$.

D.2 The CGW anonymous IBE scheme

Definition 27 (CGW-IBE [CGW15])**.** The anonymous identity-based encryption scheme proposed by Chen, Gay and Wee is defined as follows.

- Setup(λ): On input the security parameter λ, the algorithm generates three groups $\mathbb{G}, \mathbb{H}, \mathbb{G}_T$ of prime order p with generators $g \in \mathbb{G}$ and $h \in \mathbb{H}$, and chooses a pairing $e\colon \mathbb{G} \times \mathbb{H} \to \mathbb{G}_T$. It also specifies a collision-resistant hash function $\mathcal{H}\colon \{0,1\}^* \to \mathbb{Z}_p$. It then generates random $k_i, a_i, b_i, w_{0ij}, w_{1ij} \in_R \mathbb{Z}_p$ for all $i, j \in \{1,2\}$. It outputs $\mathrm{MSK} = (\{k_i, a_i, b_i, w_{0ij}, w_{1ij}\}_{i,j\in\{1,2\}})$ as the master secret key and publishes the domain parameters $(p, \mathbb{G}, \mathbb{H}, \mathbb{G}_T, \mathcal{H})$ and the master public key as

$$\mathrm{MPK} = (g, h, A = e(g,h)^{a_1 k_1 + a_2 k_2},$$
$$A_1 = g^{a_1}, A_2 = g^{a_2}, \{W_{l,j} = g^{a_1 w_{l1j} + a_2 w_{l2j}}\}_{j\in\{1,2\}, l\in\{0,1\}}).$$

- KeyGen(MSK, ID): On input identifier $\mathsf{ID} \in \{0,1\}^*$, the PKG first hashes $x = \mathcal{H}(\mathsf{ID})$, and then generates random integer $r \in_R \mathbb{Z}_p$ and computes the secret key as

$$\mathrm{SK}_{\mathsf{ID}} = (\{K_i = h^{r b_i}, K'_i = h^{-k_i - r(b_1 w_{0i1} + b_2 w_{0i2} + x(b_1 w_{1i1} + b_2 w_{1i2}))}\}_{i\in\{1,2\}}).$$

- Encrypt(MPK, ID, M): Message $M \in \mathbb{G}_T$ is encrypted under identifier ID by first hashing $x = \mathcal{H}(\mathsf{ID})$, then picking random integer $s \in_R \mathbb{Z}_p$, and computing the ciphertext as

$$\mathrm{CT}_{\mathsf{ID}} = \big(C = M \cdot A^s, \{C_i = A_i^s, C_i' = (W_{0i} W_{1i}^x)^s\}_{i\in\{1,2\}}\big).$$

- Decrypt(MPK, $\mathrm{SK}_{\mathsf{ID}'}, \mathrm{CT}_{\mathsf{ID}}$): Suppose that $\mathsf{ID} = \mathsf{ID}'$, then the ciphertext can decrypted by computing

$$M' = C \cdot e(C_1, K_1') \cdot e(C_2, K_2') \cdot e(C_1', K_1) \cdot e(C_2', K_2).$$

The scheme is correct, i.e., we have

$$\begin{aligned}
e(C_1, K_1') \cdot e(C_2, K_2') &= \prod_{i\in\{1,2\}} e(A_i^s, h^{-k_i - r(b_1 w_{0i1} + b_2 w_{0i2} + x(b_1 w_{1i1} + b_2 w_{1i2}))}) \\
&= e(g,h)^{-s(a_1 k_1 + a_2 k_2)} \cdot e(g,h)^{-sr(a_1 b_1 w_{011} + a_1 b_2 w_{012} + x(a_1 b_1 w_{111} + a_1 b_2 w_{112})} \\
&\quad \cdot e(g,h)^{-sr(a_2 b_1 w_{021} + a_2 b_2 w_{022} + x(a_2 b_1 w_{121} + a_2 b_2 w_{122}))} \\
&= A^{-s} \cdot e(g,h)^{-srb_1(a_1 w_{011} + a_2 w_{021} + x(a_1 w_{111} + a_2 w_{121})} \\
&\quad \cdot e(g,h)^{-srb_2(a_1 w_{012} + x a_1 w_{112} + a_2 w_{022} + x a_2 w_{122})} \\
&= A^{-s} \cdot e(C_1', K_1)^{-1} \cdot e(C_2', K_2)^{-1}.
\end{aligned}$$

Hence, computing

$$\begin{aligned}
&C \cdot e(C_1, K_1') \cdot e(C_2, K_2') \cdot e(C_1', K_1) \cdot e(C_2', K_2) \\
&= M \cdot A^{-s} \cdot A^{-s} \cdot e(C_1', K_1)^{-1} \cdot e(C_2', K_2)^{-1} \cdot e(C_1', K_1) \cdot e(C_2', K_2) = M
\end{aligned}$$

yields the original plaintext message.

They prove the following:

Proposition 2. *The identity-based encryption scheme in Definition 27 is fully CPA-secure and anonymous.*

D.3 Our CCA-transformation for predicate encodings

Definition 28 (CCA-secure CGW15-IBE)**.** The CCA-secure version of the anonymous identity-based encryption scheme proposed by Chen, Gay and Wee, obtained by applying our CCA-transformations in Sections 3 and 5 is defined as follows. (Note that, because the KEM in Definition 27 is special decomposable and both $\mathrm{CT}_{2,x}$ and $\mathrm{CT}_{3,x'}$ are uniquely determined by CT_1, we use the variation in Section 3.3. In this way, we obtain a strict separation between the KEM and DEM.)

- Setup(λ): On input the security parameter λ, the algorithm generates three groups $\mathbb{G}, \mathbb{H}, \mathbb{G}_T$ of prime order p with generators $g \in \mathbb{G}$ and $h \in \mathbb{H}$, and chooses a pairing $e\colon \mathbb{G} \times \mathbb{H} \to \mathbb{G}_T$. It also specifies a collision-resistant hash function $\mathcal{H}\colon \{0,1\}^* \to \mathbb{Z}_p$, an authenticated symmetric encryption scheme $\mathrm{SE} = (\mathrm{Enc_K}, \mathrm{Dec_K})$ with $\mathcal{K}(\lambda) = \mathbb{G}_T$, and a random-prefix collision-resistant hash function $\mathrm{RPC}\colon \{0,1\}^\lambda \times \mathbb{G} \to \mathbb{Z}_p$. It then generates random $k_i, a_i, b_i, w_{0ij}, w_{1ij}, w'_{0ij}, w'_{1ij}, u_{ij} \in_R \mathbb{Z}_p$ for all $i, j \in \{1,2\}$. It outputs $\mathrm{MSK} = (\{k_i, a_i, b_i, w_{0ij}, w_{1ij}, w'_{0ij}, w'_{1ij}, u_{ij}\}_{i,j\in\{1,2\}})$ as the master secret key and publishes the domain parameters $(p, \mathbb{G}, \mathbb{H}, \mathbb{G}_T, \mathcal{H})$ and the master public key as

$$\begin{aligned}
&\mathrm{MPK} = (g, h, A = e(g,h)^{a_1 k_1 + a_2 k_2}, A_1 = g^{a_1}, A_2 = g^{a_2}, \\
&\{W_{l,j} = g^{a_1 w_{l1j} + a_2 w_{l2j}}\}_{j\in\{1,2\}, l\in\{0,1\}}, \{W'_{l,j} = g^{a_1 w'_{l1j} + a_2 w'_{l2j}}\}_{j\in\{1,2\}, l\in\{0,1\}}).
\end{aligned}$$

- KeyGen(MSK, ID): On input identifier $\mathsf{ID} \in \{0,1\}^*$, the PKG first hashes $x = \mathcal{H}(\mathsf{ID})$, and then generates random integers $r \in_R \mathbb{Z}_p$, sets $k_{i,1} \leftarrow b_1 u_{i1} + b_2 u_{i2}$ and $k_{i,2} \leftarrow k_i - b_1 u_{i1} - b_2 u_{i2}$, and computes the secret key as

$$\mathrm{SK}_{\mathsf{ID}} = (\{K_i = h^{rb_i}, K'_{i,1} = h^{k_{i,1} - r(b_1 w_{0i1} + b_2 w_{0i2} + x(b_1 w_{1i1} + b_2 w_{1i2}))},$$
$$K'_{i,2} = h^{k_{i,2} - r(b_1 w'_{0i1} + b_2 w'_{0i2})}, K''_{i,2} = h^{-r(b_1 w'_{1i1} + b_2 w'_{1i2})}\}_{i \in \{1,2\}}).$$

- Encrypt(MPK, ID, M): Message $M \in \{0,1\}^*$ is encrypted under identifier ID by first hashing $x = \mathcal{H}(\mathsf{ID})$, then picking random integer $s \in_R \mathbb{Z}_p$, and computing the ciphertext as

$$\mathrm{CT}_{\mathsf{ID}} = \big(\mathrm{CT}_{\mathrm{sym}} = \mathrm{Enc}_{\mathrm{K}}(M), \{C_i = A_i^s,$$
$$C'_{i,1} = (W_{0i} W_{1i}^x)^s, C'_{i,2} = (W'_{0i} W_{1i}'^{x'})^s\}_{i \in \{1,2\}}, k\big),$$

  where $k \in_R \{0,1\}^\lambda$, $x' \leftarrow \mathrm{RPC}(k, C_1 \| C_2)$, and $\mathrm{K} \leftarrow A^s$.

- Decrypt(MPK, $\mathrm{SK}_{\mathsf{ID}'}$, $\mathrm{CT}_{\mathsf{ID}}$): Suppose that $\mathsf{ID} = \mathsf{ID}'$, then the ciphertext can decrypted by computing $y' \leftarrow \mathrm{RPC}(k, C_1 \| C_2)$,

$$\mathrm{K}' = e(C_1, K'_{1,1} K'_{1,2} K''^{y'}_{1,2}) \cdot e(C_2, K'_{2,1} K'_{2,2} K''^{y'}_{2,2}) \cdot e(C'_{1,1} C'_{1,2}, K_1) \cdot e(C'_{2,1} C'_{2,2}, K_2),$$

  and retrieving $M' \leftarrow \mathrm{Dec}_{\mathrm{K}'}(\mathrm{CT}_{\mathrm{sym}})$.

Corollary 1. *The scheme in Definition 28 is CCA-secure and anonymous.*

Proof. This follows directly from Theorem 1 and Proposition 1. Note that we use that the ciphertext part $\{C'_{i,1} = (W_{0i} W_{1i}^x)^s, C'_{i,2} = (W'_{0i} W_{1i}'^{x'})^s\}_{i \in \{1,2\}}$ is uniquely defined by C_1 and C_2. □

This scheme can be further optimized. In particular, for its correctness, we do not require that $K_{i,1}$ and $K_{i,2}$, and $C'_{i,1}$ and $C'_{i,2}$ (for $i \in \{1,2\}$) are given separately during key generation and encryption, respectively, for decryption to work. We also show that we do not need these to be separate for its security either. First, we define the optimized version of this scheme:

Definition 29 (Optimized CCA-secure CGW-IBE)**.** The optimized CCA-secure version of the anonymous identity-based encryption scheme proposed by Chen, Gay and Wee, obtained by applying our CCA-transformations in Sections 3 and 5 is defined as follows.

- Setup(λ): On input the security parameter λ, the algorithm generates three groups $\mathbb{G}, \mathbb{H}, \mathbb{G}_T$ of prime order p with generators $g \in \mathbb{G}$ and $h \in \mathbb{H}$, and chooses a pairing $e\colon \mathbb{G} \times \mathbb{H} \to \mathbb{G}_T$. It also specifies a collision-resistant hash function $\mathcal{H}\colon \{0,1\}^* \to \mathbb{Z}_p$, an authenticated symmetric encryption scheme $\mathrm{SE} = (\mathrm{Enc}_{\mathrm{K}}, \mathrm{Dec}_{\mathrm{K}})$ with $\mathcal{K}(\lambda) = \mathbb{G}_T$, and a random-prefix collision-resistant hash function $\mathrm{RPC}\colon \{0,1\}^\lambda \times \mathbb{G} \to \mathbb{Z}_p$. It then generates random $k_i, a_i, b_i, w_{0ij}, w_{1ij}, w'_{ij} \in_R \mathbb{Z}_p$ for all $i, j \in \{1,2\}$. It outputs $\mathrm{MSK} = (\{k_i, a_i, b_i, w_{0ij}, w_{1ij}, w'_{ij}\}_{i,j \in \{1,2\}})$ as the master secret key and publishes the domain parameters $(p, \mathbb{G}, \mathbb{H}, \mathbb{G}_T, \mathcal{H})$ and the master public key as

$$\mathrm{MPK} = (g, h, A = e(g,h)^{a_1 k_1 + a_2 k_2}, A_1 = g^{a_1}, A_2 = g^{a_2},$$
$$\{W_{l,j} = g^{a_1 w_{l1j} + a_2 w_{l2j}}\}_{j \in \{1,2\}, l \in \{0,1\}}, \{W'_j = g^{a_1 w'_{1j} + a_2 w'_{2j}}\}_{j \in \{1,2\}}).$$

- KeyGen(MSK, ID): On input identifier $\mathsf{ID} \in \{0,1\}^*$, the PKG first hashes $x = \mathcal{H}(\mathsf{ID})$, and then generates random integers $r \in_R \mathbb{Z}_p$, and computes the secret key as

$$\mathrm{SK}_{\mathsf{ID}} = (\{K_i = h^{rb_i}, K'_{i,1} = h^{k_i - r(b_1 w_{0i1} + b_2 w_{0i2} + x(b_1 w_{1i1} + b_2 w_{1i2}))},$$
$$K'_{i,2} = h^{-r(b_1 w'_{i1} + b_2 w'_{i2})}\}_{i \in \{1,2\}}).$$

- Encrypt(MPK, ID, M): Message $M \in \{0,1\}^*$ is encrypted under identifier ID by first hashing $x = \mathcal{H}(\mathsf{ID})$, then picking random integer $s \in_R \mathbb{Z}_p$, and computing the ciphertext as

$$\mathrm{CT}_{\mathsf{ID}} = \big(\mathrm{CT}_{\mathrm{sym}} = \mathrm{Enc}_{\mathrm{K}}(M), \{C_i = A_i^s, C_i' = (W_{0i} W_{1i}^x W_i'^{x'})^s\}_{i\in\{1,2\}}, k\big),$$

  where $k \in_R \{0,1\}^\lambda$, $x' \leftarrow \mathrm{RPC}(k, C_1 \| C_2)$, and $\mathrm{K} \leftarrow A^s$.

- Decrypt(MPK, $\mathrm{SK}_{\mathsf{ID}'}, \mathrm{CT}_{\mathsf{ID}}$): Suppose that $\mathsf{ID} = \mathsf{ID}'$, then the ciphertext can decrypted by computing $y' \leftarrow \mathrm{RPC}(k, C_1 \| C_2)$,

$$\mathrm{K}' = e(C_1, K'_{1,1} K'^{y'}_{1,2}) \cdot e(C_2, K'_{2,1} K'^{y'}_{2,2}) \cdot e(C'_1, K_1) \cdot e(C'_2, K_2),$$

  and retrieving $M' \leftarrow \mathrm{Dec}_{\mathrm{K}'}(\mathrm{CT}_{\mathrm{sym}})$.

Proposition 3. *The scheme in Definition 29 is fully CCA-secure.*

Proof. We show this by reducing the CPA-security of the associated EPE variant of the optimized scheme to the CPA-security of the associated EPE variant of the basic scheme (which essentially remove the symmetric encryption scheme from the CCA-secure variants of these schemes).

Let $\mathcal{A}_1$ be an attacker that can break the SEPE scheme associated with the scheme in Definition 29. We construct an attacker $\mathcal{A}_2$ that can break the EPE scheme associated with the scheme in Definition 28. Let $\mathcal{C}_1$ and $\mathcal{C}_2$ be the challengers in the games with $\mathcal{A}_1$ and $\mathcal{A}_2$, respectively.

- **Setup phase:** In the setup, challenger $\mathcal{C}_2$ generates a master public key as

$$\mathrm{MPK} \leftarrow (g, h, A = e(g,h)^{a_1 k_1 + a_2 k_2}, A_1 = g^{a_1}, A_2 = g^{a_2},$$
$$\{W_{l,j} = g^{a_1 w_{l1j} + a_2 w_{l2j}}\}_{j\in\{1,2\}, l\in\{0,1\}}, \{W'_{l,j} = g^{a_1 w'_{l1j} + a_2 w'_{l2j}}\}_{j\in\{1,2\}, l\in\{0,1\}}),$$

  and sends it to attacker $\mathcal{A}_2$. Challenger $\mathcal{C}_1$ sets

$$\overline{\mathrm{MPK}} \leftarrow (g, h, A, A_1, A_2, \{\bar{W}_{0,j} = W_{0,j} \cdot W'_{0,j}, \bar{W}_{1,j} = W_{1,j}, \bar{W}'_j = W'_{1,j}\}_{j\in\{1,2\}}),$$

  and sends it to attacker $\mathcal{A}_1$.

- **First query phase:** For each identity ID for which attacker $\mathcal{A}_1$ requests a secret key, we relay the request to challenger $\mathcal{C}_2$, who generates

$$\mathrm{SK}_{\mathsf{ID}} = (\{K_i = h^{r b_i}, K'_{i,1} = h^{-k_{i,1} - r(b_1 w_{0i1} + b_2 w_{0i2} + x(b_1 w_{1i1} + b_2 w_{1i2}))},$$
$$K'_{i,2} = h^{-k_{i,2} - r(b_1 w'_{0i1} + b_2 w'_{0i2})}, K''_{i,2} = h^{-r(b_1 w'_{1i1} + b_2 w'_{1i2})}\}_{i\in\{1,2\}}).$$

  Challenger $\mathcal{C}_1$ then sets

$$\overline{\mathrm{SK}}_{\mathsf{ID}} = (\{K_i, \bar{K}'_{i,1} = K'_{i,1} \cdot K'_{i,2}, \bar{K}'_{i,2} = K''_{i,2}\}_{i\in\{1,2\}}).$$

- **Challenge phase:** Attacker $\mathcal{A}_1$ sends a challenge identity ID^* and two messages M_0, M_1 to challenger $\mathcal{C}_1$, who relays these to challenger $\mathcal{C}_2$. The challenger flips a coin $\beta \in_R \{0,1\}$ and sends back ciphertext

$$\mathrm{CT}^*_{\mathsf{ID}^*} = \big(C = M_\beta \cdot A^s, \{C_i = A_i^s,$$
$$C'_{i,1} = (W_{0i} W_{1i}^x)^s, C'_{i,2} = (W'_{0i} W'^{x'}_{1i})^s\}_{i\in\{1,2\}}\big).$$

  Challenger $\mathcal{C}_1$ then sends the ciphertext

$$\overline{\mathrm{CT}}^*_{\mathsf{ID}^*} = \big(C, \{C_i, \bar{C}'_i = C'_{i,1} \cdot C'_{i,2}\}_{i\in\{1,2\}}\big)$$

  to attacker $\mathcal{A}_1$.

- **Second query phase:** This phase is identical to the first query phase.
- **Guessing phase:** Attacker $\mathcal{A}_1$ outputs a guess β' for β, which attacker $\mathcal{A}_2$ also outputs as its guess.

□

It also follows quickly (by slightly adjusting the proof of Proposition 1) that the scheme is anonymous.

Corollary 2. *The scheme in Definition 29 is anonymous.*

D.4 CCA-security with the FO-transformation

We compare our CCA-transformed variants of CGW-IBE with a CCA-variant obtained by applying the Fujisaki-Okamoto transform [FO99, HHK17]. In particular, we apply the transformation yielding an explicit rejection during the decapsulation that does not take the ciphertext of the KEM as input to the hash that is used to derive a symmetric key.

Definition 30 (CCA-secure CGW-IBE with FO (CCA-CGW-IBE-FO))**.** The CCA-secure variant of the anonymous identity-based encryption scheme proposed by Chen, Gay and Wee obtained from the FO-transform [HHK17] is defined as follows.

- Setup(λ): On input the security parameter λ, the algorithm generates three groups $\mathbb{G}, \mathbb{H}, \mathbb{G}_T$ of prime order p with generators $g \in \mathbb{G}$ and $h \in \mathbb{H}$, and chooses a pairing $e\colon \mathbb{G} \times \mathbb{H} \to \mathbb{G}_T$. It also specifies a collision-resistant hash function $\mathcal{H}\colon \{0,1\}^* \to \mathbb{Z}_p$, a cryptographic hash function $\mathcal{G}\colon \mathbb{G}_T \to \mathbb{Z}_p$, a key derivation function $\mathrm{KDF}\colon \mathbb{G}_T \to \{0,1\}^{2\lambda}$, and an authenticated encryption scheme $\mathrm{SE} = (\mathrm{Enc_K}, \mathrm{Dec_K})$ with $\mathcal{K}(\lambda) = \{0,1\}^{2\lambda}$. It then generates random $k_i, a_i, b_i, w_{0ij}, w_{1ij} \in_R \mathbb{Z}_p$ for all $i, j \in \{1,2\}$. It outputs $\mathrm{MSK} = (\{k_i, a_i, b_i, w_{0ij}, w_{1ij}\}_{i,j\in\{1,2\}})$ as the master secret key and publishes the domain parameters $(p, \mathbb{G}, \mathbb{H}, \mathbb{G}_T, \mathcal{H})$ and the master public key as
$$\mathrm{MPK} = (g, h, A = e(g,h)^{a_1k_1+a_2k_2},$$
$$A_1 = g^{a_1}, A_2 = g^{a_2}, \{W_{l,j} = g^{a_1w_{l1j}+a_2w_{l2j}}\}_{j\in\{1,2\},l\in\{0,1\}}).$$
- KeyGen($\mathrm{MSK}, \mathsf{ID}$): On input identifier $\mathsf{ID} \in \{0,1\}^*$, the PKG first hashes $x = \mathcal{H}(\mathsf{ID})$, and then generates random integer $r \in_R \mathbb{Z}_p$ and computes the secret key as
$$\mathrm{SK}_{\mathsf{ID}} = (\{K_i = h^{rb_i}, K_i' = h^{-k_i - r(b_1w_{0i1}+b_2w_{0i2}+x(b_1w_{1i1}+b_2w_{1i2}))}\}_{i\in\{1,2\}}).$$
- Encrypt($\mathrm{MPK}, \mathsf{ID}, M$): Message $M \in \{0,1\}^*$ is encrypted under identifier ID by first hashing $x = \mathcal{H}(\mathsf{ID})$, then picking random $M' \in_R \mathbb{G}_T$, setting $s \leftarrow \mathcal{G}(M')$, and computing the ciphertext as
$$\mathrm{CT}_{\mathsf{ID}} \leftarrow (\mathrm{CT}_{\mathrm{sym}} = \mathrm{Enc_K}(M), \mathrm{Encrypt}'(\mathrm{MPK}, \mathsf{ID}, M'; s)),$$
where $\mathrm{K} \leftarrow \mathrm{KDF}(M')$, and
$$\mathrm{Encrypt}'(\mathrm{MPK}, \mathsf{ID}, M'; s) = \big(C = M' \cdot A^s, \{C_i = A_i^s, C_i' = (W_{0i}W_{1i}^x)^s\}_{i\in\{1,2\}}\big).$$
- Decrypt($\mathrm{MPK}, \mathrm{SK}_{\mathsf{ID}'}, \mathrm{CT}_{\mathsf{ID}}$): Suppose that $\mathsf{ID} = \mathsf{ID}'$, then the ciphertext can decrypted by computing
$$M'' = C \cdot e(C_1, K_1') \cdot e(C_2, K_2') \cdot e(C_1', K_1) \cdot e(C_2', K_2),$$
then verifying whether
$$(C, C_1, C_2, C_1', C_2') \stackrel{?}{=} \mathrm{Encrypt}'(\mathrm{MPK}, \mathsf{ID}, M''; \mathcal{G}(M''))$$
holds and, if so, return $M \leftarrow \mathrm{Dec}_{\mathrm{KDF}(M'')}(\mathrm{CT}_{\mathrm{sym}})$.

E A large-universe CP-ABE scheme

E.1 Access structures

Definition 31 (Monotone access structures [Bei96])**.** Let $\{a_1, ..., a_n\}$ be a set of attributes. An access structure is a collection $\mathbb{A}$ of non-empty subsets of $\{a_1, ..., a_n\}$. The sets in $\mathbb{A}$ are called the authorized sets, and the sets that are not in $\mathbb{A}$ are called the unauthorized sets. An access structure $\mathbb{A} \subseteq 2^{\{a_1,...,a_n\}}$ is monotone if for all B, C holds: $B \in \mathbb{A}$ and $B \subseteq C$, then also $C \in \mathbb{A}$.

We represent access policies $\mathbb{A}$ by linear secret sharing scheme (LSSS) matrices, which support monotone span programs [Bei96, GPSW06b].

Definition 32 (Access structures represented by LSSS matrices [GPSW06b])**.** An access structure can be represented as a pair $\mathbb{A} = (\mathbf{A}, \rho)$ such that $\mathbf{A} \in \mathbb{Z}_p^{n_1 \times n_2}$ is an LSSS matrix, where $n_1, n_2 \in \mathbb{N}$, and ρ is a function that maps its rows to attributes in the universe. Then, for some vector with randomly generated entries $\mathbf{v} = (s, v_2, ..., v_{n_2}) \in \mathbb{Z}_p^{n_2}$, the i-th secret generated by this matrix is $\lambda_i = \mathbf{A}_i \mathbf{v}^\intercal$, where $\mathbf{A}_i$ denotes the i-th row of $\mathbf{A}$. In particular, if $\mathcal{S}$ satisfies $\mathbb{A}$, then there exist a set of rows $\Upsilon = \{i \in [n_1] \mid \rho(i) \in \mathcal{S}\}$ and coefficients $\varepsilon_i \in \mathbb{Z}_p$ for all $i \in \Upsilon$ such that $\sum_{i \in \Upsilon} \varepsilon_i \mathbf{A}_i = (1, 0, ..., 0)$, and by extension $\sum_{i \in \Upsilon} \varepsilon_i \lambda_i = s$, holds.

In [LW10], Lewko and Waters devise a way to convert Boolean formulas into LSSS matrices. For our implementations, we use strictly ANDs in our policies, which ensures that the matrix $\mathbf{A}$ is a square matrix in which the number of rows and columns is equal to the length n of the policy, the first row consists of 1s in the first two entries (and the rest is 0), the last row has -1 in the last column and the rest all-zero, and the rest of the rows $i \in [2, n-1]$ is of the form $\mathbf{1}_{i+1}^n - \mathbf{1}_i^n$:

$$\mathbf{A}_1 = \mathbf{1}_1^n + \mathbf{1}_2^n \qquad \mathbf{A}_i = \mathbf{1}_{i+1}^n - \mathbf{1}_i^n \qquad \mathbf{A}_n = -\mathbf{1}_n^n,$$

for all $i \in [2, n-1]$.

E.2 Ciphertext-policy ABE

Definition 33 (Ciphertext-policy ABE [BSW07])**.** A ciphertext-policy ABE (CP-ABE) scheme consists of four algorithms:

- $\text{Setup}(\lambda) \to (\text{MPK}, \text{MSK})$: The setup takes as input a security parameter λ, it outputs the master public-secret key pair (MPK, MSK).
- $\text{KeyGen}(\mathcal{S}, \text{MSK}) \to \text{SK}_\mathcal{S}$: The key generation takes as input a set of attributes $\mathcal{S}$ and the master secret key MSK, and outputs a secret key $\text{SK}_\mathcal{S}$.
- $\text{Encrypt}(M, \mathbb{A}, \text{MPK}) \to \text{CT}_\mathbb{A}$: The encryption takes as input a plaintext message M, an access policy $\mathbb{A}$ and the master public keys MPK. It outputs a ciphertext $\text{CT}_\mathbb{A}$.
- $\text{Decrypt}(\text{CT}_\mathbb{A}, \text{SK}_\mathcal{S}) \to M'$: The decryption takes as input the ciphertext $\text{CT}_\mathbb{A}$ that was encrypted under an access policy $\mathbb{A}$, and a secret key $\text{SK}_\mathcal{S}$ associated with a set of attributes $\mathcal{S}$. It succeeds and outputs the plaintext message M' if $\mathcal{S}$ satisfies $\mathbb{A}$. Otherwise, it aborts.

A scheme is called correct if decryption of a ciphertext with secret key yields the original plaintext message. We consider a scheme to be large-universe if it does not impose bounds on the size of the universe.

E.3 The selectively secure variant of RW13

Definition 34 (The RW13 CP-ABE scheme [RW13])**.** The ciphertext-policy attribute-based encryption scheme by Rouselakis and Waters (RW13) [RW13] is defined as follows.

- Setup(λ): Taking as input the security parameter λ, the setup generates three groups $\mathbb{G}, \mathbb{H}, \mathbb{G}_T$ of prime order p with generators $g \in \mathbb{G}, h \in \mathbb{H}$, and chooses a pairing $e\colon \mathbb{G} \times \mathbb{H} \to \mathbb{G}_T$. The setup also defines the universe of attributes $\mathcal{U} = \mathbb{Z}_p$. It then generates random $\alpha, b, b_0, b_1, b' \in_R \mathbb{Z}_p$. It outputs $\text{MSK} = (\alpha, b, b_0, b_1, b')$ as its master secret key and publishes the master public key as

$$\text{MPK} = (g, h, A = e(g,h)^{\alpha}, B = g^{b}, B_0 = g^{b_0}, B_1 = g^{b_1}, B' = g^{b'}).$$

- KeyGen($\text{MSK}, \mathcal{S}$): On input a set of attributes $\mathcal{S}$, the algorithm generates random integers $r, r_{\text{att}} \in_R \mathbb{Z}_p$ for each $\text{att} \in \mathcal{S}$, letting $x_{\text{att}} \in \mathbb{Z}_p$ denote the representation of att in $\mathbb{Z}_p$ and computes the secret key as

$$\text{SK}_{\mathcal{S}} = (K = h^{\alpha - rb}, K' = h^{r}, \{K_{1,\text{att}} = h^{-r_{\text{att}}(b_1 x_{\text{att}} + b_0) - rb'}, K_{2,\text{att}} = h^{r_{\text{att}}}\}_{\text{att} \in \mathcal{S}}).$$

- Encrypt($M, \text{MPK}, \mathbb{A}$): A message $M \in \mathbb{G}_T$ is encrypted under $\mathbb{A} = (\mathbf{A}, \rho)$ with $\mathbf{A} \in \mathbb{Z}_p^{n_1 \times n_2}$ and $\rho\colon [n_1] \to \mathcal{U}$ by generating random integers $s, s_i, v_j \in_R \mathbb{Z}_p$ for all $i \in [n_1]$ and $j \in [2, n_2]$, and computes the ciphertext as

$$\text{CT}_{\mathbb{A}} = (C = M \cdot A^{s}, C' = g^{s}, \{C_{1,j} = B^{\lambda_j}(B')^{s_j},$$
$$C_{2,j} = \left(B_1^{\rho(j)} B_0\right)^{s_j}, C_{3,j} = g^{s_j}\}_{j \in [1, n_1]}),$$

  such that λ_i denotes the i-th entry of $\mathbf{A} \cdot (s, v_2, ..., v_{n_2})^{\intercal}$.

- Decrypt($\text{SK}_{\mathcal{S}}, \text{CT}_{\mathbb{A}}$): Suppose that $\mathcal{S}$ satisfies $\mathbb{A}$, and suppose $\Upsilon = \{j \in [1, n_1] \mid \rho(j) \in \mathcal{S}\}$, such that $\{\varepsilon_j \in \mathbb{Z}_p\}_{j \in \Upsilon}$ exist with $\sum_{i \in \Upsilon} \varepsilon_j \mathbf{A}_j = (1, 0, ..., 0)$. Then the plaintext M is retrieved by computing

$$C / \left(e(C', K) \cdot e(\prod_{j \in \Upsilon} C_{1,j}^{\varepsilon_j}, K') \prod_{j \in \Upsilon} \left(e(C_{2,j}^{\varepsilon_j}, K_{2,\rho(j)}) \cdot e(C_{3,j}^{\varepsilon_j}, K_{1,\rho(j)}) \right) \right).$$

  Note that, in the case of AND-gates, $\varepsilon_j \in \{0, 1\}$.

E.4 A fully secure variant of RW13

We present a fully secure variant of this scheme, given in the Agrawal-Chase framework (AC17) [AC17].

Definition 35 (The fully secure RW13 CP-ABE scheme (RWAC) [AC17])**.** The ciphertext-policy attribute-based encryption scheme by Rouselakis and Waters (RW13) [RW13] is defined in the Agrawal-Chase framework, using the prime-order dual system groups for SXDH in [CW14], as follows.

- Setup(λ): Taking as input the security parameter λ, the setup generates three groups $\mathbb{G}, \mathbb{H}, \mathbb{G}_T$ of prime order p with generators $g \in \mathbb{G}, h \in \mathbb{H}$, and chooses a pairing $e\colon \mathbb{G} \times \mathbb{H} \to \mathbb{G}_T$. The setup also defines the universe of attributes $\mathcal{U} = \mathbb{Z}_p$. It then generates random $\alpha_1, \alpha_2, d_1, d_2, d_3, d_4, d_5, b_i, b_{0,i}, b_{1,i}, b'_i \in_R \mathbb{Z}_p$ for all $i \in \{1, 2, 3\}$, such that $d_1 d_4 \neq d_2 d_3$. It outputs $\text{MSK} = (\alpha_1, \alpha_2, d_1, d_2, d_3, d_4, d_5, b_i, b_{0,i}, b_{1,i}, b'_i)$ as its master secret key and publishes the master public key as

$$\text{MPK} = (g, h, A = e(g,h)^{\alpha_1 d_1 + \alpha_2 d_2}, \{g_i = g^{d_i}, B_i = g^{b_1 d_i + b_3 d_{i+2}},$$
$$\{B_{l,i} = g^{b_{l,1} d_i + b_{l,3} d_{i+2}}\}_{l \in \{0,1\}}, B'_i = g^{b'_1 d_i + b'_3 d_{i+2}}\}_{i \in \{1,2\}}).$$

- KeyGen(MSK, $\mathcal{S}$): On input a set of attributes $\mathcal{S}$, the algorithm generates random integers $r, r_{\text{att}} \in_R \mathbb{Z}_p$ for each att $\in \mathcal{S}$, letting $x_{\text{att}} \in \mathbb{Z}_p$ denote the representation of att in $\mathbb{Z}_p$ and computes the secret key as

$$\begin{aligned}\text{SK}_{\mathcal{S}} = (\{K_i = h^{\alpha_i - r\bar{b}_i}, K_1' = h^{rd_4d_6}, K_2' = h^{-rd_3d_6},\\ K_{1,\text{att},i} = h^{-r_{\text{att}}(\bar{b}_{1,i}x_{\text{att}} + \bar{b}_{0,i}) - r\bar{b}_i'},\\ K_{2,\text{att},1} = h^{r_{\text{att}}d_4d_6}, K_{2,\text{att},2} = h^{-r_{\text{att}}d_3d_6}\}_{i\in\{1,2\},\text{att}\in\mathcal{S}}),\end{aligned}$$

  where for $l \in \{0,1\}$, we have

$$\begin{aligned}d_6 = \frac{d_5}{d_1d_4 - d_2d_3}\\ \bar{b}_1 = d_6(b_1d_4 - b_2d_2), \bar{b}_2 = d_6(-b_1d_3 + b_2d_1),\\ \bar{b}_{l,1} = d_6(b_{l,1}d_4 - b_{l,2}d_2), \bar{b}_{l,2} = d_6(-b_{l,1}d_3 + b_{l,2}d_1),\\ \bar{b}_1' = d_6(b_1'd_4 - b_2'd_2), \bar{b}_2' = d_6(-b_1'd_3 + b_2'd_1).\end{aligned}$$

- Encrypt(M, MPK, $\mathbb{A}$): A message $M \in \mathbb{G}_T$ is encrypted under $\mathbb{A} = (\mathbf{A}, \rho)$ with $\mathbf{A} \in \mathbb{Z}_p^{n_1 \times n_2}$ and $\rho\colon [n_1] \to \mathcal{U}$ by generating random integers $s, s_i, v_j \in_R \mathbb{Z}_p$ for all $i \in [n_1]$ and $j \in [2, n_2]$, and computes the ciphertext as

$$\begin{aligned}\text{CT}_{\mathbb{A}} = (C = M \cdot A^s, \{C_i' = g_i^s, C_{1,i,j} = B_i^{A_{j,1}s} g_i^{\lambda_j} (B_i')^{s_j},\\ C_{2,i,j} = \left(B_{1,i}^{\rho(j)} B_{0,i}\right)^{s_j}, C_{3,i,j} = g_i^{s_j}\}_{i\in\{1,2\}, j\in[1,n_1]}),\end{aligned}$$

  such that λ_j denotes the j-th entry of $\mathbf{A} \cdot (0, v_2, ..., v_{n_2})^\intercal$.

- Decrypt($\text{SK}_{\mathcal{S}}, \text{CT}_{\mathbb{A}}$): Suppose that $\mathcal{S}$ satisfies $\mathbb{A}$, and suppose $\Upsilon = \{j \in [1, n_1] \mid \rho(j) \in \mathcal{S}\}$, such that $\{\varepsilon_j \in \mathbb{Z}_p\}_{j\in\Upsilon}$ exist with $\sum_{i\in\Upsilon} \varepsilon_j \mathbf{A}_j = (1, 0, ..., 0)$. Then the plaintext M is retrieved by computing

$$C / \prod_{i\in\{1,2\}} \left(e(C_i', K_i) \cdot e(\prod_{j\in\Upsilon} C_{1,i,j}^{\varepsilon_j}, K_i') \prod_{j\in\Upsilon} \left(e(C_{2,i,j}^{\varepsilon_j}, K_{2,\rho(j),i}) \cdot e(C_{3,i,j}^{\varepsilon_j}, K_{1,\rho(j),i})\right)\right).$$

E.5 Our CCA-transformation for PES

We transform the fully CPA-secure scheme in Definition 35 to a fully CCA-secure scheme with the transformation for PES in Section 5. Like for the CGW-IBE, because RWAC is special decomposable, we use the variation in Section 3.3, which hashes the partial ciphertext $\text{CT}_{2,x}$ with the RPC hash.

Definition 36 (The fully CCA-secure RWAC scheme)**.** The CCA-secure version obtained with our transformation in Section 5 of RWAC [RW13, AC17, Att19] is defined as follows.

- Setup(λ): Taking as input the security parameter λ, the setup generates three groups $\mathbb{G}, \mathbb{H}, \mathbb{G}_T$ of prime order p with generators $g \in \mathbb{G}, h \in \mathbb{H}$, and chooses a pairing $e\colon \mathbb{G} \times \mathbb{H} \to \mathbb{G}_T$. It also specifies a collision-resistant hash function $\mathcal{H}\colon \{0,1\}^* \to \mathbb{Z}_p$, an authenticated symmetric encryption scheme $\text{SE} = (\text{Enc}_{\text{K}}, \text{Dec}_{\text{K}})$ with $\mathcal{K}(\lambda) = \mathbb{G}_T$, and a random-prefix collision-resistant hash function $\text{RPC}\colon \{0,1\}^\lambda \times \mathbb{G} \to \mathbb{Z}_p$. The setup also defines the universe of attributes $\mathcal{U} = \mathbb{Z}_p$. It then generates random $\alpha_1, \alpha_2, d_1, d_2, d_3, d_4, d_5, b_i, b_{0,i}, b_{1,i}, b_{0,i}', b_{1,i}', b_i' \in_R \mathbb{Z}_p$ for all $i \in \{1,2,3\}$, such that $d_1d_4 \neq d_2d_3$. It outputs $\text{MSK} = (\alpha_1, \alpha_2, d_1, d_2, d_3, d_4, d_5, b_i, b_{0,i}, b_{1,i}, b_{1,i}', b_i')$ as its master secret key and publishes the master public key as

$$\begin{aligned}\text{MPK} = (g, h, A = e(g,h)^{\alpha_1d_1 + \alpha_2d_2}, \{g_i = g^{d_i}, B_i = g^{b_1d_i + b_3d_{i+2}},\\ \{B_{l,i} = g^{b_{l,1}d_i + b_{l,3}d_{i+2}}, B_{l,i}' = g^{b_{l,1}'d_i + b_{l,3}'d_{i+2}}\}_{l\in\{0,1\}}, B_i' = g^{b_1'd_i + b_3'd_{i+2}}\}_{i\in\{1,2\}}).\end{aligned}$$

- KeyGen(MSK, $\mathcal{S}$): On input a set of attributes $\mathcal{S}$, the algorithm generates random integers $r, \alpha_{1,1}, \alpha_{2,1}, r_{\text{att}} \in_R \mathbb{Z}_p$ for each att $\in \mathcal{S}$, sets $\alpha_{1,2} \leftarrow \alpha_1 - \alpha_{1,1}$ and $\alpha_{2,2} \leftarrow \alpha_2 - \alpha_{2,1}$, letting $x_{\text{att}} = \mathcal{H}(\text{att})$ denote the representation of att in $\mathbb{Z}_p$, and computes the secret key as

$$\begin{gathered}\text{SK}_{\mathcal{S}} = (\{K_i = h^{\alpha_{i,1} - r\bar{b}_i}, K_1' = h^{rd_4d_6}, K_2' = h^{-rd_3d_6}, \\ K_i^{(2)} = h^{\alpha_{i,2} - r\bar{b}'_{0,i}}, K_i^{(3)} = h^{-r\bar{b}'_{1,i}}, K_{1,\text{att},i} = h^{-r_{\text{att}}(\bar{b}_{1,i}x_{\text{att}} + \bar{b}_{0,i}) - r\bar{b}'_i}, \\ K_{2,\text{att},1} = h^{r_{\text{att}}d_4d_6}, K_{2,\text{att},2} = h^{-r_{\text{att}}d_3d_6}\}_{i\in\{1,2\},\text{att}\in\mathcal{S}}),\end{gathered}$$

where for $l \in \{0,1\}$, we have

$$\begin{gathered}d_6 = \frac{d_5}{d_1d_4 - d_2d_3} \\ \bar{b}_1 = d_6(b_1d_4 - b_2d_2), \bar{b}_2 = d_6(-b_1d_3 + b_2d_1), \\ \bar{b}_{l,1} = d_6(b_{l,1}d_4 - b_{l,2}d_2), \bar{b}_{l,2} = d_6(-b_{l,1}d_3 + b_{l,2}d_1), \\ \bar{b}'_{l,1} = d_6(b'_{l,1}d_4 - b'_{l,2}d_2), \bar{b}'_{l,2} = d_6(-b'_{l,1}d_3 + b'_{l,2}d_1), \\ \bar{b}'_1 = d_6(b'_1d_4 - b'_2d_2), \bar{b}'_2 = d_6(-b'_1d_3 + b'_2d_1).\end{gathered}$$

- Encrypt(M, MPK, $\mathbb{A}$): A message $M \in \{0,1\}^*$ is encrypted under $\mathbb{A} = (\mathbf{A}, \rho)$ with $\mathbf{A} \in \mathbb{Z}_p^{n_1 \times n_2}$ and ρ: $[n_1] \to \mathcal{U}$ by generating random integers s,s_j,$v_{j'} \in_R \mathbb{Z}_p$ for all $j \in [n_1]$ and $j' \in [2, n_2]$, and computes the ciphertext as

$$\begin{gathered}\text{CT}_{\mathbb{A}} = (\text{CT}_{\text{sym}} = \text{Enc}_{\text{K}}(M), \{C_i' = g_i^s, C_{1,i,j} = B_i^{A_{j,1}s} g_i^{\lambda_j} (B_i')^{s_j}, \\ C_{2,i,j} = \left(B_{1,i}^{\rho(j)} B_{0,i}\right)^{s_j}, C_{3,i,j} = g_i^{s_j}, C_{4,i} = \left(B_{1,i}'^{x'} B_{0,i}'\right)^s\}_{i\in\{1,2\},j\in[1,n_1]}, k),\end{gathered}$$

such that $k \in \{0,1\}^\lambda$,

$$x' \leftarrow \text{RPC}(k, \{C_i', C_{1,i,j}, C_{2,i,j}, C_{3,i,j}\}_{i\in\{1,2\},j\in[1,n_1]}),$$

$\text{K} \leftarrow A^s$, and λ_j denotes the j-th entry of $\mathbf{A} \cdot (0, v_2, ..., v_{n_2})^\intercal$. Note that, although we use set notation in the input to the hash, the ciphertext components should be concatenated deterministically (in a domain-separated fashion), such that encryption and decryption yield the same output.

- Decrypt($\text{SK}_{\mathcal{S}}, \text{CT}_{\mathbb{A}}$): Suppose that $\mathcal{S}$ satisfies $\mathbb{A}$, and suppose $\Upsilon = \{j \in [1, n_1] \mid \rho(j) \in \mathcal{S}\}$, such that $\{\varepsilon_j \in \mathbb{Z}_p\}_{j\in\Upsilon}$ exist with $\sum_{i\in\Upsilon} \varepsilon_j \mathbf{A}_j = (1, 0, ..., 0)$. Then the plaintext M is retrieved by computing

$$\begin{aligned}\text{K}' \leftarrow \prod_{i\in\{1,2\}} \Bigg(& e(C_i', K_iK_i^{(2)}(K_i^{(3)})^{y'}) \cdot e\left(\left(\prod_{j\in\Upsilon} C_{1,i,j}^{\varepsilon_j}\right) \cdot C_{4,i}, K_i'\right) \\ & \cdot \prod_{j\in\Upsilon} \left(e(C_{2,i,j}^{\varepsilon_j}, K_{2,\rho(j),i}) \cdot e(C_{3,i,j}^{\varepsilon_j}, K_{1,\rho(j),i})\right)\Bigg),\end{aligned}$$

where

$$y' \leftarrow \text{RPC}(k, \{C_i', C_{1,i,j}, C_{2,i,j}, C_{3,i,j}\}_{i\in\{1,2\},j\in[1,n_1]}),$$

and then obtaining $M' \leftarrow \text{Dec}_{\text{K}'}(\text{CT}_{\text{sym}})$.

E.6 CCA-security with other transformations

We compare the efficiency of the CCA-secure scheme using our transformations with several other CCA-secure variants of RWAC. In particular, we consider the

- FO-transformation [FO99] using the techniques in [HHK17];
- YAHK-transformation [YAHK11] using the delegatability property;
- YAHK-transformation [YAHK11] using the verifiability property.

More generic transformations exist, as mentioned in the introduction, but these incur similar computational trade-offs. Rather than implementing fully functional variants of these schemes, we estimate the storage and computational costs based on the operations required in the algorithms of the transformed variants. For instance, the FO-transformed variant calls the encryption algorithm during decryption, and the efficiency of the variants using the YAHK-transformations depends on the efficiency of the chosen OTS. Table 3 estimates the overhead of all variants, and shows that our transformation yields the fastest decryption algorithm. In particular, the additional costs are a small constant.

Table 3: Comparison of the additional storage and computational costs incurred by the CCA-transformation among several CCA-secure variants of RWAC. We do not list the symmetric operations, such as hashes, encryptions and MACs.

Variant	$\lvert\text{MPK}\rvert$	$\lvert\text{SK}_{\mathcal{S}}\rvert$	$\lvert\text{CT}_{\mathbb{A}}\rvert$	KeyGen	Encrypt	Decrypt
FO	-	$s_{\mathbb{G}_T} + 10s_{\mathbb{G}}$	-	-	-	$10\lvert\Upsilon\rvert c_{\text{exp},\mathbb{G}}$
Delegatability	-	$8\lvert\text{vk}\rvert s_{\mathbb{H}}$	$6\lvert\text{vk}\rvert s_{\mathbb{G}}$	$8\lvert\text{vk}\rvert c_{\text{exp},\mathbb{H}}$	$6\lvert\text{vk}\rvert c_{\text{exp},\mathbb{G}}$	$2\lvert\text{vk}\rvert p$
Verifiability	-	-	$6s_{\mathbb{G}}$	-	$10c_{\text{exp},\mathbb{G}}$	$2\lvert\Upsilon\rvert p$
Ours	$4s_{\mathbb{G}}$	$4s_{\mathbb{H}}$	$2s_{\mathbb{G}}$	$4c_{\text{exp},\mathbb{H}}$	$4c_{\text{exp},\mathbb{G}}$	$2c_{\text{exp},\mathbb{H}}$

Note: $c_{\text{exp},\mathbb{G}'}$ = costs of an exponentiation in $\mathbb{G}'$, $s_{\mathbb{G}'}$ = the size of an element in $\mathbb{G}'$, p = the costs of a pairing operation, vk = verification key used in YAHK

Contents