

GLUE: Generalizing Unbounded Attribute-Based Encryption for Flexible Efficiency Trade-Offs

Marloes Venema^{1,2}(✉) and Greg Alpar^{2,3}

¹ University of Wuppertal, Wuppertal, Germany

² Radboud University, Nijmegen, the Netherlands

³ Open University of the Netherlands, Heerlen, the Netherlands
mvenemacrypto@gmail.com, g.alpar@cs.ru.nl

Abstract. Ciphertext-policy attribute-based encryption is a versatile primitive that has been considered extensively to securely manage data in practice. Especially completely unbounded schemes are attractive, because they do not restrict the sets of attributes and policies. So far, any such schemes that support negations in the access policy or that have online/offline extensions have an inefficient decryption algorithm.

In this work, we propose GLUE (Generalized, Large-universe, Unbounded and Expressive), which is a novel scheme that allows for the efficient implementation of the decryption while allowing the support of both negations and online/offline extensions. We achieve these properties simultaneously by uncovering an underlying dependency between encryption and decryption, which allows for a flexible trade-off in their efficiency. For the security proof, we devise a new technique that enables us to generalize multiple existing schemes. As a result, we obtain a completely unbounded scheme supporting negations that, to the best of our knowledge, outperforms all existing such schemes in the decryption algorithm.

Keywords: attribute-based encryption · unbounded attribute-based encryption · online/offline attribute-based encryption · non-monotone attribute-based encryption

1 Introduction

Attribute-based encryption (ABE) is an advanced type of public-key encryption in which the key pairs are associated with attributes [47]. In ciphertext-policy (CP) ABE, messages are encrypted under an access policy [17]. The resulting ciphertexts can then be decrypted by a secret key associated with a set of attributes⁴ that satisfies the policy. Conversely, in key-policy (KP) ABE, the keys are associated with access policies and the ciphertexts with sets of attributes

⁴ In this paper, we will use the terms “sets of attributes” and “(attribute) sets” to refer to the attributes associated with the secret keys. We use the term “universe of attributes” to refer to the total set of attributes that can be used in the system.

[26]. To securely and efficiently implement access control on data, especially pairing-based CP-ABE proves to be an attractive primitive [17,34,48,53,36]. In 2018, the European Telecommunications Standards Institute (ETSI) published two technical reports on ABE [24,25], which include detailed descriptions of use cases, varying from cloud settings to mobile networks. In such settings, the computational resources of the encryption and decryption devices may vary. Thus, different use cases may require schemes with different efficiency trade-offs.

According to ETSI, ABE schemes should be efficient and secure. Interestingly, while ETSI proposes ABE to be used to enforce attribute-based access control [32] on data, it explicitly notes that ABE cannot satisfactorily support it, because ABE cannot support negations efficiently [25]. Indeed, the decryption algorithm of most ABE schemes supporting negations is incredibly expensive [43,37,57,11]. Recently, some interesting progress was made, yielding significant speed-ups in decryption time [49,14]. However, those schemes still have a costly decryption [14] or restrict the attribute sets [49].

In this work, we introduce GLUE, which is a new scheme that enables the realization of the following properties:

- (1) Large-universe: any string can be used as an attribute;
- (2) Unbounded: no restrictions on e.g., the sizes of the policies or attributes sets, or the number of times that an attribute may occur in the policy;
- (3) Expressive: support of monotone span programs, ensuring that policies represented as Boolean formulas consisting of conjunctions and disjunctions can be supported;
- (4) Non-monotone: support of non-monotone span programs, ensuring that the policies can use negations;
- (5) Compact: the number of key and ciphertext components depends at most linearly on the set size or the policy length, and in particular does not depend on other parameters (implicit or explicit)⁵.

GLUE is designed to offer a flexible choice in the encryption/decryption efficiency trade-off during the setup of the parameters. More specifically, it is parametrized in variables n_k and n_c , where the encryption costs increase in $n_k + n_c$ and the decryption costs decrease in n_k and n_c , e.g., in the factor n_k if $n_k = n_c$ holds. In this way, the scheme can be fine-tuned to take into account the computational resources of the encryption and decryption devices. In particular, this feature allows for significant speed-ups in the decryption algorithm compared to other schemes that also satisfy the listed properties.

Large-universe and unbounded ABE. The universe of attributes, i.e., the attributes that can be used in the scheme, can be small or large [47]. In small-universe constructions, the number of attributes is bounded after the setup, e.g., because a public key needs to be generated for each attribute. In large-universe

⁵ For example, parameters such as those in schemes with a flexible efficiency trade-off (e.g., [12,52]) or the number of re-uses of the same attribute in the policy (e.g., [35]).

constructions, the universe of attributes is effectively unbounded. Moreover, the public keys do not depend on the attributes in the system, and a user can directly encrypt messages using the master public key and the attribute string. Large-universe ABE is thus more scalable than small-universe ABE, as the encrypting users do not need to first locate the necessary public keys before encrypting. Some large-universe schemes [47,56,13] are however undesirably restrictive [39], as they are bounded in the sizes of the policies or attribute sets, or the number of times that an attribute may occur in the policy. Oftentimes, the scheme’s efficiency depends on such bounds, e.g., the encryption costs grow linearly in the bounds on the policies or sets [55]. Hence, choosing high bounds is not a suitable solution either. Preferably, a scheme is unbounded in all parameters.

Expressivity and non-monotonicity. Most state-of-the-art ABE schemes are expressive in that their policies support monotone span programs (MSPs) [2,49,11,40]. An important subclass of MSPs are Boolean formulas consisting of conjunctions and disjunctions. As mentioned, popular access control models such as attribute-based access control [32] allow the policies to be any Boolean formula, including negations. ABE schemes that support negations are called non-monotone [43,57,11,49,14]. In addition to being more expressive, such schemes readily support revocation systems [37], which is crucial in practice as well.

Different types of non-monotonicity. For large-universe constructions, three types of non-monotonicity exist: OSW-type [43], OT-type [42], and OSWOT-type [14]. In OSW-type negations, e.g., “NOT user: Alice”, the entire attribute set, e.g., “{user: Bob, course: linear algebra, course: calculus}”, is compared with the negated attribute to establish that the set does not contain it. In ABE implementations, this translates in a decryption cost that grows in not only the size of the policy, but also in size of the attribute sets. Such negations may thus not be efficient if the sets are large. In OT-type negations, e.g., “user: NOT Alice”, the attribute labels, e.g., “user”, play a role. In particular, the set must contain an attribute with label “user” and its value, e.g., “Bob”, must differ from the negated attribute. While this is more efficient than OSW-type negations, the set of attributes is allowed to contain only one attribute for each label, e.g., such negations are not supported for the label “course” in our first example. Thus, schemes supporting this type of negations are bounded in the number of label re-uses, which is not always desirable. For instance, like in our example, users may have multiple attributes for labels such as “departments at a hospital”, “courses followed at a university” or “mail addresses”. As a solution, Attrapadung and Tomida [14] introduced OSWOT-type negations, e.g., “course: NOT cryptography”, to extend OT-type negations, such that the negated attribute is compared with all attributes in the set that share the same label, e.g., “{course: linear algebra, course: calculus}”. In this way, the flexibility of OSW-type negations and the efficiency of OT-type negations can be combined.

Table 1: Comparison of large-universe schemes supporting (non-)monotone span programs. For each scheme, we list whether it is unbounded (in the sets $\mathcal{S}$ and policies $\mathbb{A}$, and the number of attribute and label re-uses), whether it supports negations or has a provably secure extension that supports negations, whether it is compact and supports a flexible efficiency trade-off (FET). Note that we have only listed schemes that are structurally different “in the exponent”, i.e., their associated pair encodings [9] are different. For instance, the unbounded schemes in [35,39] have a similar structure and therefore only [39] is listed.

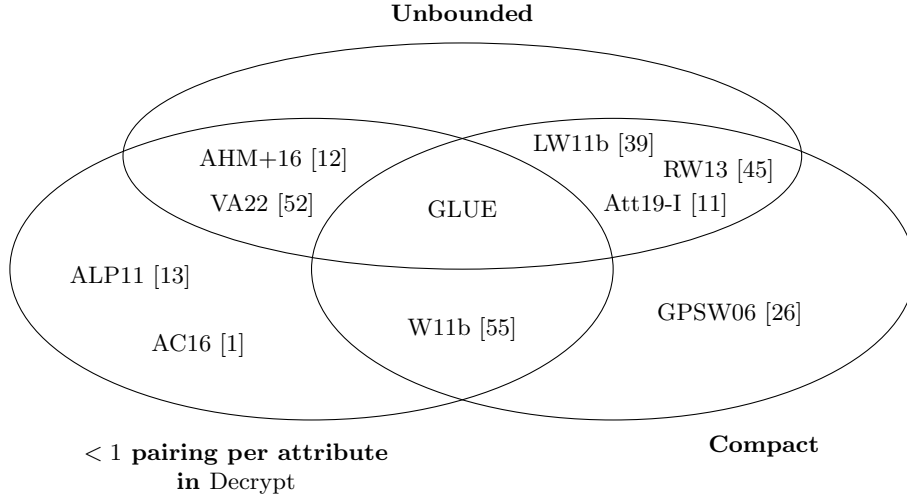
Scheme	KP/CP	Unbounded				Negations			Compact	FET
		$ \mathcal{S} $	$ \mathbb{A} $	ARU	LRU	OSW	OT	OSWOT		
GPSW06-LU [26]	KP	✗	✓	✓	✓	✓ [43]	✗	✗	✓	✗
BSW07 [17]	CP	✓	✓	✓	✓	✗	✗	✗	✓	✗
ALP11 [13]	KP	✗	✓	✓	✓	✓	✗	✗	✗	✗
W11-LU I [56]	CP	✓	✓	✓	✓	✗	✗	✗	✓	✗
W11b-LU [55]	CP	✗	✗	✗	✗	✗	✗	✗	✓	✗
LW11b [39,45]	KP	✓	✓	✓	✓	✓ [37]	✓ [11]	✓ [14]	✓	✗
OT12 [42]	CP	✓	✓	✗	✗	✗	✓	✗	✗	✗
RW13 [45]	CP	✓	✓	✓	✓	✓ [57]	✓ [11]	✓ [14]	✓	✗
AHM+16 [12]	KP	✓	✓	✓	✓	✓ [13]	✗	✗	✗	✓
AC16 [1]	CP	✗	✗	✓	✓	✓ [11,6]	✓ [14,6]	✓ [14,6]	✗	✗
AC17b [2]	CP(/KP)	✓	✓	✗	✗	✗	✗	✗	✓	✗
ABGW17 [7]	KP/CP	✓	✓	✓	✓	✗	✗	✗	✓	✗
Att19-I-CP [11]	CP(/KP)	✓	✓	✓	✓	✓	✓ [14]	✓ [14]	✓	✗
Att19-II-CP [11]	CP(/KP)	✗	✓	✓	✓	✓	✗	✗	✗	✗
Att19-III-CP [11]	CP(/KP)	✓	✗	✓	✓	✓	✗	✗	✗	✗
TKN20 [49]	KP/CP	✓	✓	✓	✗	✗	✓	✗	✓	✗
VA22 [52]	CP	✓	✓	✓	✓	✓ [6]	✗	✗	✗	✓
GLUE	CP(/KP)	✓	✓	✓	✓	✓	✓	✓ [14]	✓	✓

Note: $\mathcal{S}$ = attribute set; $\mathbb{A}$ = access policy;

ARU = attribute re-use in the policies; LRU = label re-use in the sets and policies

Achieving properties (1)-(5) simultaneously. Only a limited number of existing schemes support properties (1)-(5) simultaneously [53]. In fact, all pairing-based schemes that provide non-monotonicity and large-universeness use a polynomial-based hash—also known as a “Boneh-Boyen (BB) hash” [18]—that maps arbitrary attribute strings into the scheme [47,43]. Of those schemes, the only ones that are completely unbounded [37,57,11,14] are based on the schemes by Lewko and Waters (the KP-ABE version) [39] and Rouselakis and Waters (the CP-ABE version) [45] (Table 1). However, all those schemes have an inefficient decryption algorithm compared to other ABE schemes [44], which use a full-domain hash (FDH) to achieve large-universeness. For this reason, such schemes are often favored in practice, despite their inability to support negations [24,25]. Nevertheless, since supporting negations fosters the expressivity of ABE, we aim at improving the decryption efficiency of schemes using a BB hash.

Fig. 1: Overview of large-universe schemes using a BB hash.



Improving decryption efficiency of schemes using a BB hash. To determine whether we can improve on the decryption efficiency of the existing schemes satisfying properties (1)-(5), we investigate *all* schemes using a BB hash to achieve large-universeness. In particular, if we consider all such schemes, then we see that a scheme that is unbounded, compact and costs less than one pairing operation per attribute during decryption does not exist yet (Figure 1). Because pairing operations are the most expensive operations in pairing-based ABE, it is therefore important to minimize the use of those. In this work, we aim to achieve this: we provide a scheme that satisfies properties (1)-(5), while requiring less than one pairing operation per attribute during decryption.

1.1 Our contributions

We first give a high-level overview of our contributions. Then, we provide more (technical) details about these contributions.

- **New construction:** We present a new unbounded large-universe scheme using a BB hash (thus avoiding random oracles). Its encryption/decryption efficiency trade-off can be fine-tuned by taking into account the computational resources of the devices.
- **Generalizations:** Concretely, the scheme can be considered a generalization of two large-universe schemes: the Rouselakis-Waters (RW13) scheme [45] and the bounded large-universe scheme without random oracles by Waters (W11b) [55]. This generalization also illustrates a deeper connection among various designs.
- **Security proof:** We develop new proof techniques to ensure that the randomness provided by a BB hash can be simultaneously used for the keys and

ciphertexts. To the best of our knowledge, we are the first to achieve this in the unbounded setting, and in the full-security setting.

- **Extensions:** We provide three extensions to the basic scheme: one online/offline [31] and two non-monotone extensions supporting OT-type and OSW-type negations, respectively. Notably, we obtain an online/offline ABE scheme and a scheme supporting OSW-type negations with the most efficient decryption algorithms. This enables us to support OSWOT-type negations more efficiently, which is the most desirable in practice.

1.2 New construction: GLUE

We focus on three schemes that satisfy at least two out of the three depicted properties (see Figure 1): W11b [55], RW13 [45] and AHM+16 [12]. Those three schemes provide a good starting point for GLUE, our new scheme which satisfies all required properties. Intuitively, we apply the partitioning techniques of AHM+16 to combine the unbounded RW13 and the bounded W11b that allows for efficient decryption. However, as we show later, for the secure combination of these techniques, GLUE requires a more intricate approach.

We give a high-level description of the partitioning approach as introduced by AHM+16. First, we partition the attribute sets into smaller subsets. Then, we apply a (bounded) scheme with efficient decryption (in their case, ALP11 [13]) to each subset. Lastly, we use the unbounded techniques of e.g., RW13 or LW11b [39] to securely connect the subsets. In this way, the decryption costs of the scheme can be decreased. Unfortunately, this comes at a cost. Because bounded schemes typically have a more expensive encryption, the encryption costs are increased. From a broader perspective, this approach creates a scheme with a (flexibly) efficiency trade-off feature. As we will show later, this trade-off is determined by some parameter n . The encryption costs are higher by a factor n than those of unbounded schemes such as RW13, whereas the decryption costs are lower by this factor. Because this parameter n can be chosen during setup, it can be fine-tuned for the given practical context. If decryption needs to be efficient (which is often the case), one can choose larger n than in cases in which encryption needs to be efficient.

The main reason why we achieve the compactness property, in contrast to AHM+16, is due to the bounded scheme that is used. Because AHM+16 uses ALP11 [13], a scheme with constant-size ciphertexts and large keys whose sizes depend on the parameter n , its keys are large and its key generation is very expensive. Moreover, although the number of pairing operations required during decryption decreases, the number of exponentiations grows by a factor n for each matching attribute. As we will show, this means that AHM+16 decryption is not much more efficient than unbounded schemes such as RW13. As a solution, we use the W11b scheme, whose decryption costs consist of a constant number of pairing operations and no additional exponentiations. In this way, we achieve a much better speed-up in decryption, and since W11b is compact, the key sizes and key generation costs are not affected.

1.3 Generalizing RW13 by generalizing the hash

The main difference between the partitioning approach as applied by AHM+16 and us is that we have to partition both the key sets and the ciphertext policies. The reason for this is that AHM+16 uses ALP11, which is bounded in only the ciphertexts, whereas we use W11b, which is bounded in both the keys and ciphertexts. By extension, we need to apply some technique to connect the resulting key and ciphertext “parts”. However, we will show that the security proof of W11b does not generalize to the unbounded setting, meaning that we have to devise a new proof technique for W11b that does generalize. Furthermore, it is not possible to apply the exact same approach as that of AHM+16. In particular, to prove security, they embed the scheme in the fully secure key-policy doubly-spatial encryption [30] scheme in [9], and then, they apply the embedding lemma [12]. We cannot use this approach, because, to the best of our knowledge, the W11b scheme cannot be similarly embedded in an existing scheme.

Hence, we take a slightly different approach: we generalize RW13 by generalizing its specific instantiation of the BB hash. A BB hash first takes as input a unique representation x_{att} of an attribute att in the integer set $\mathbb{Z}_p$, where p is the prime order of group $\mathbb{G}$ with generator $g \in \mathbb{G}$. Then, it computes the hash as $F_n(x_{\text{att}}) = \prod_{i=0}^n B_i^{x_{\text{att}}^i}$, where the generators $B_i = g^{b_i}$ implicitly embed the coefficients of the polynomial $f_n(x_{\text{att}}) = \sum_{i=0}^n b_i x_{\text{att}}^i$. Where RW13 (and its unbounded derivatives [31,11]) uses an implicit 1-degree polynomial, we use an implicit n -degree polynomial, like W11b [55]. However, as we will show, simply replacing the 1-degree polynomial by some n -degree polynomial does not immediately yield a secure scheme. To solve this, we replace another public-key variable used in the scheme by a polynomial.

1.4 Security proof

The AC17 framework. To benefit from the strong security guarantees as well as the generic transformations [11,6] within the Agrawal-Chase (AC17) [3] framework, we prove security within it. In general, the AC17 framework considers the pair encoding schemes (PESs) associated with an ABE scheme [9]. Intuitively, PESs are an abstraction of ABE schemes to what happens in the exponent space. If a PES is secure, the AC17 framework transforms it into a fully (also known as adaptively) secure scheme. The security notion for PES is called symbolic security, which consists of two parts: the selective and co-selective symbolic property. These properties hold for any scheme that is not trivially broken, so the AC17 framework simplifies the effort of proving full security considerably by giving such transformations. A small drawback is that the resulting schemes are provably secure under q -type assumptions [19], which are less well-understood than static security assumptions. Regardless, the assumptions used in the AC17 framework are implied by commonly-used q -type assumptions [9]. These have not been shown to yield less secure schemes in practice yet. Another advantage of the AC17 framework is that any symbolically secure PES can be transformed

in a PES that supports negations [6]. It is notoriously difficult to achieve this in the full-security model in combination with the large-universe property [11].

Proving the symbolic property. Proving the (co-)selective symbolic property is similar to proving selective security. In the selective-security model, the attacker commits to the challenge access policy (resp. set of attributes) before seeing the public keys. Many schemes proven security in this model use the “program-and-cancel” strategy [56,45], in which the challenger embeds the policy (resp. set) in the public keys. In the simulation of the secret keys and challenge ciphertext, the components are programmed in a specific way, using that the set does not satisfy the policy (resp. policy is not satisfied by the set). Typically, the components that cannot be programmed are canceled by other non-programmable components. In the AC17 framework, this “programming” is replaced by “substitution”, and the “canceling” is replaced by “evaluating to 0”.

Security proof. One of the main difficulties of our scheme is proving the symbolic property. In the first place, proving security is difficult due to the lack of provably secure schemes that use the randomness provided by the BB hash for *both the keys and ciphertexts*. To the best of our knowledge, previously, only W11b [55] used the hash for both the keys and ciphertexts, but only in the bounded setting and in the selective-security model. However, the proof does not seem to readily generalize to the unbounded setting (see the full version [54]). Hence, we develop a novel technique to prove security. We do this, in part, by combining several techniques.

- **Proof techniques using the hash for the keys:** We generalize the proof techniques used by Agrawal and Chase in [3] to prove full security of their scheme in [1]: the AC16 [1] scheme. AC16 is a CP-ABE scheme with constant-size ciphertexts, in which the randomness provided by the Boneh-Boyen hash is used for the *keys*. In the selective proof, the polynomial embedded in the public keys needs to be used by the secret keys after the public keys are generated. The proof does this by embedding a “reprogrammable” polynomial in the public keys. We call these polynomials to be reprogrammable in the sense that the randomizers of the secret keys can later program it to a suitable target polynomial. We use this general strategy for the keys.
- **Proof techniques using the hash for the ciphertext:** Even though the W11b [55] proof does not generalize to the unbounded setting, we are able to use a part of the proof strategy. In the selective proof, the implicit polynomial embedded in the public key is “programmed” to take into account the attributes that will be used in the challenge ciphertext. We use this general strategy for the ciphertexts.
- **Unbounded proof techniques:** One of the bottlenecks of the two aforementioned strategies is that they are bounded approaches: they use only one randomizer for the keys and one for the ciphertexts. To make them unbounded, we use the general approach of the RW13 [45] proof. This proof

gives us a rough idea of how the implicit polynomial and the randomizers should be programmed. Furthermore, it shows us how to use the polynomial an unbounded number of times: using layering and individual randomness techniques allows us to select the required instance of the polynomial.

Another bottleneck is that the “programmed” and “reprogrammable” approaches are orthogonal, and can therefore not be used simultaneously in the same polynomial without applying a trick. Presumably, this is also the reason why the W11b proof uses the programmed approach for both the keys and the ciphertexts, and applies an algebraic argument to ensure that everything can be simulated as required. We eliminate this bottleneck and combine all these proof techniques, by splitting the polynomial in the product of two smaller polynomials: one “programmed” polynomial and one “reprogrammable” polynomial. For the selective proof, we use the programmed polynomial for the ciphertexts and the reprogrammable polynomial for the keys. For the co-selective proof, the roles of the polynomials are reversed.

1.5 Practical extensions

We provide several extensions to our scheme, most of which can be found in the full version [54]. Because we prove security in the AC17 [3] framework, some of these extensions are automatically provably secure.

- **The key-policy and dual-policy versions**, by applying [11];
- **Online/offline extensions**, by generalizing HW14 [31]. Owing to its generality, these extensions also apply to the following extensions;
- **Non-monotone versions**, by applying [6,11,49]. We give versions supporting OT-type and OSW-type negations in the full version [54].

Online/offline extension. The algebraic structure of large-universe schemes using a BB hash can also be exploited to increase the efficiency. Hohenberger and Waters [31] show how the key generation and encryption of RW13 [45] can be split in an online and offline phase. In this way, most of the computations required by these algorithms can be performed in an offline phase. During the online phase, little computational power is required. This is especially useful in practice when secret keys need to be generated frequently, e.g., in revocation systems [46]. The key generation authority then does not need to have computationally powerful resources to do this in an acceptable time frame. Similarly, the online/offline encryption variant can be used to minimize the encryption costs. This comes however at a cost: the decryption costs increase. Thus, reducing the decryption costs of the basic scheme also helps reducing the decryption costs of the online/offline version. With the online/offline version of GLUE, we can not only improve on the decryption efficiency of existing such schemes, but we can also mitigate the impact of the increase of n on the encryption efficiency.

Table 2: Theoretical efficiency comparison of all compact unbounded large-universe CP-ABE schemes supporting MSPs (that (can) support OSW(OT)-type negations), by analyzing the key generation, encryption and decryption costs with respect to the number of exponentiations c_{exp} and pairings c_{pair} .

Scheme	Key generation	Encryption	Decryption	
	c_{exp}	c_{exp}	c_{exp}	c_{pair}
RW13 [45]	$4 + 4 \mathcal{S} $	$4 + 10 \mathbb{A} $	$4 \mathcal{T} $	$4 + 4 \mathcal{T} $
Att19-I-CP [11]	$12 + 12 \mathcal{S} $	$4 + 16 \mathbb{A} $	$4 \mathcal{T} $	$8 + 4 \mathcal{T} $
GLUE	$4 + 2 \mathcal{S} + 2 \left\lceil \frac{ \mathcal{S} }{n_k} \right\rceil$	$4 + 2 \mathbb{A} (n_k + 2n_c + 1) + 2 \left\lceil \frac{ \mathbb{A} }{n_c} \right\rceil$	$2 \mathcal{T} $	$4 + 2 \left\lceil \frac{ \mathcal{T} }{n_k} \right\rceil + 2 \left\lceil \frac{ \mathcal{T} }{n_c} \right\rceil$
GLUE-N	$12 + 8 \mathcal{S} + 4 \left\lceil \frac{ \mathcal{S} }{n_k} \right\rceil$	$4 + 2 \mathbb{A} (n_k + 2n_c + 4) + 2 \left\lceil \frac{ \mathbb{A} }{n_c} \right\rceil$	$4 \mathcal{T} $	$8 + 2 \left\lceil \frac{ \mathcal{T} }{n_k} \right\rceil + 2 \left\lceil \frac{ \mathcal{T} }{n_c} \right\rceil$

(a) Costs for non-negated policies

Scheme	c_{exp}	c_{pair}
Att19-I-CP [11]	$4 \mathcal{T} \cdot \mathcal{S} $	$8 + 2 \mathcal{T} + 2 \min(\mathcal{T} , \mathcal{S})$
GLUE-N (worst case)	$4 \mathcal{T} \cdot \mathcal{S} + 2 \left\lceil \frac{ \mathcal{S} }{n_k} \right\rceil \cdot \mathcal{T} $	$8 + 2 \mathcal{T} + 2 \left\lceil \frac{ \mathcal{S} }{n_k} \right\rceil$
GLUE-N (best case)	$4 \left\lceil \frac{ \mathcal{T} }{n_c} \right\rceil \cdot \mathcal{S} + 2 \left\lceil \frac{ \mathcal{S} }{n_k} \right\rceil \cdot \mathcal{T} $	$8 + 2 \left\lceil \frac{ \mathcal{T} }{n_c} \right\rceil + 2 \left\lceil \frac{ \mathcal{S} }{n_k} \right\rceil$

(b) Decryption costs for negated policies

Note: $\mathcal{S}$ = attribute set; $\mathbb{A}$ = access policy; $\mathcal{T}$ = matching attributes;
 n_k, n_c = parameters chosen during setup

1.6 Efficiency comparison with existing schemes supporting (1)-(5)

We generalize RW13 to achieve a scheme that supports or can support properties (1)-(5) whilst being able to achieve a more efficient decryption algorithm. In Table 2, we compare the efficiency of RW13 and its OSW-type non-monotone variant Att19-I-CP with GLUE (which supports MSPs only) and GLUE-N (which additionally supports OSW-type negations). The table shows that, if $n_k = n_c$, the number of pairing operations required during decryption is reduced by roughly a factor of $n_k = n_c$. In Section 6, we give more concrete estimates for the timings in practice and how they compare to existing schemes.

2 Preliminaries

Notation. A negligible function parametrized by λ is denoted as $\text{negl}(\lambda)$. We use $x \in_R S$ to indicate that an element x is chosen uniformly at random from a finite set S . For integers $a < b$, we denote $[a, b] = \{a, a+1, \dots, b-1, b\}$, $[b] = [1, b]$ and $\overline{[b]} = [0, b]$. We denote $a : \mathbf{A}$ to substitute variable a by a matrix or vector $\mathbf{A}$. We define $\mathbf{1}_{i,j}^{d_1 \times d_2} \in \mathbb{Z}_p^{d_1 \times d_2}$ as the matrix with 1 in the i -th row and j -th column, and 0 everywhere else, and $\mathbf{1}_i^{d_1}$ and $\overline{\mathbf{1}}_i^{d_2}$ as the row and column vectors with 1 in the i -th entry and 0 everywhere else.

2.1 Access structures

Definition 1 (Access structures represented by LSSS matrices [27]). We represent an access structure 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 $\mathbf{v} = (s, v_2, \dots, v_{n_2}) \in_R \mathbb{Z}_p^{n_2}$, the i -th share of secret s generated by this matrix is $\lambda_i = \mathbf{A}_i \mathbf{v}^\top$, where $\mathbf{A}_i$ denotes the i -th row of $\mathbf{A}$. In particular, if $\mathcal{S}$ satisfies $\mathbb{A}$, 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 $\sum_{i \in \Upsilon} \varepsilon_i \mathbf{A}_i = (1, 0, \dots, 0)$, and by extension $\sum_{i \in \Upsilon} \varepsilon_i \lambda_i = s$, holds. Otherwise, there exists $\mathbf{w} = (1, w_2, \dots, w_{n_2}) \in \mathbb{Z}_p^{n_2}$ such that $\mathbf{A}_i \mathbf{w}^\top = 0$ for all $i \in \Upsilon$ [16].

2.2 Attribute-based encryption

Predicate family. A predicate family [9] is a set $P = \{P_\kappa\}_{\kappa \in \mathbb{N}^c}$ for some constant c , where $P_\kappa: \mathcal{X}_\kappa \times \mathcal{Y}_\kappa \rightarrow \{0, 1\}$. For κ , it holds that $\kappa = (p, \text{par})$, where p is a natural number and par denote the rest of the entries.

Definition 2 (Attribute-based encryption (ABE) [3]). An attribute-based 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:

- $\text{Setup}(\lambda, \text{par}) \rightarrow (\text{MPK}, \text{MSK})$: 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 $\kappa = (p, \text{par})$, where p denotes a natural number.
- $\text{KeyGen}(\text{MSK}, y) \rightarrow \text{SK}_y$: On input the master secret key MSK and $y \in \mathcal{Y}_\kappa$, this probabilistic algorithm generates a secret key SK_y .
- $\text{Encrypt}(\text{MPK}, x, M) \rightarrow \text{CT}_x$: On input the master public key MPK , $x \in \mathcal{X}_\kappa$ and message M , this probabilistic algorithm generates a ciphertext CT_x .
- $\text{Decrypt}(\text{MPK}, \text{SK}_y, \text{CT}_x) \rightarrow M$: 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 $P_\kappa(x, y) = 1$,

$$\Pr[(\text{MPK}, \text{MSK}) \leftarrow \text{Setup}(1^\lambda); \\ \text{Decrypt}(\text{MPK}, \text{KeyGen}(\text{MSK}, y), \text{Encrypt}(\text{MPK}, x, M)) \neq M] \leq \text{negl}(\lambda).$$

Unbounded large-universe ciphertext-policy ABE. A specific instance of ABE is ciphertext-policy ABE. In this type of ABE, the key predicate y is a set of attributes $\mathcal{S}$ over some universe of attributes $\mathcal{U}$, and the ciphertext predicate x is an access policy $\mathbb{A} = (\mathbf{A}, \rho)$, in this work represented as LSSS matrices (Definition 1). We consider a scheme to be large-universe if it does not impose bounds on the size of the universe. We call it unbounded, if it does not

impose bounds on the sizes of the universe, sets of attributes and access policies, or on the number of times that an attribute occurs in an access policy. The term “unbounded ABE” is more prominently used for schemes that achieve this without requiring random oracles in the proof.

2.3 Full security against chosen-plaintext attacks

Definition 3 (Full security against chosen-plaintext attacks (CPA) [3]).

We define the security game IND-CPA between challenger and attacker as:

- **Setup phase:** The challenger runs $\text{Setup}(\lambda)$ to obtain MPK and MSK, and sends the master public key MPK to the attacker.
- **First query phase:** The attacker queries secret keys for $y \in \mathcal{Y}_\kappa$, and obtains $\text{SK}_y \leftarrow \text{KeyGen}(\text{MSK}, y)$ 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., $\text{CT}_{x^*} \leftarrow \text{Encrypt}(\text{MPK}, x^*, M_\beta)$, 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 attacker’s advantage is defined as $\text{Adv}_{\text{PE}, \text{IND-CPA}} = |\Pr[\beta' = \beta] - \frac{1}{2}|$. A scheme is fully secure if all polynomial-time attackers have at most a negligible advantage in this security game, i.e., $\text{Adv}_{\text{PE}, \text{IND-CPA}} \leq \text{negl}(\lambda)$. In the selective security model, the attacker commits to the predicate $x^* \in \mathcal{X}_\kappa$ before the Setup.

2.4 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 , such that $e: \mathbb{G} \times \mathbb{H} \rightarrow \mathbb{G}_T$, with generators $g \in \mathbb{G}, h \in \mathbb{H}$ 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.

2.5 Pair encoding schemes

Definition 4 (Pair encoding schemes (PES) [3]). A pair encoding scheme for a predicate family $P_\kappa: \mathcal{X}_\kappa \times \mathcal{Y}_\kappa \rightarrow \{0, 1\}$, indexed by $\kappa = (p, \text{par})$, where par specifies some parameters, is given by four deterministic polynomial-time algorithms as described below.

- $\text{Param}(\text{par}) \rightarrow n$: On input par , the algorithm outputs $n \in \mathbb{N}$ that specifies the number of common variables, which are denoted as $\mathbf{b} = (b_1, \dots, b_n)$.
- $\text{EncKey}(y, p) \rightarrow (m_1, m_2, \mathbf{k}(\mathbf{r}, \hat{\mathbf{r}}, \mathbf{b}))$: On input $p \in \mathbb{N}$ and $y \in \mathcal{Y}_\kappa$, this algorithm outputs a vector of polynomials $\mathbf{k} = (k_1, \dots, k_{m_3})$ defined over non-lone variables $\mathbf{r} = (r_1, \dots, r_{m_1})$ and lone variables $\hat{\mathbf{r}} = (\hat{r}_1, \dots, \hat{r}_{m_2})$. 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 $\delta_i, \delta_{i,j}, \delta_{i,j,k} \in \mathbb{Z}_p$.

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

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

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

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

A PES is correct for every $\kappa = (p, \text{par})$, $x \in \mathcal{X}_\kappa$ and $y \in \mathcal{Y}_\kappa$ such that $P_\kappa(x, y) = 1$, it holds that $\mathbf{s}\mathbf{E}\mathbf{k}^\top + \mathbf{c}\bar{\mathbf{E}}\mathbf{r}^\top = \alpha s$.

Definition 5 (Symbolic property [3]). A pair encoding scheme $\Gamma = (\text{Param}, \text{EncKey}, \text{EncCt}, \text{Pair})$ for a predicate family $P_\kappa: \mathcal{X}_\kappa \times \mathcal{Y}_\kappa \rightarrow \{0, 1\}$ 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 that

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

such that $\langle \mathbf{s}_0, \mathbf{a} \rangle \neq 0$, and if we substitute

$$\hat{s}_{i'} : \hat{s}_{i'}^\top, \quad s_i b_j : \mathbf{B}_j \mathbf{s}_i^\top, \quad \alpha : \mathbf{a}, \quad \hat{r}_{k'} : \hat{\mathbf{r}}_{k'}, \quad r_k b_j : \mathbf{r}_k \mathbf{B}_j,$$

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 EncB , EncR , 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 [3] prove that any PES satisfying the (d_1, d_2) -symbolic property can be transformed in a fully secure ABE scheme.

3 Generalizing Rouselakis-Waters

We first show how RW13 [45] can be generalized. On a high level, we do this by substituting the implicit 1-degree polynomial in the RW13 keys and ciphertexts with an n -degree polynomial. Like W11b [55], the randomness provided by this n -degree polynomial will be shared between the keys and ciphertexts. That is, suppose that n_k and n_c are positive integers such that $n = n_k + n_c - 1$, then the n -degree polynomial provides enough randomness for $n_k - 1$ attributes in the keys, and n_c attributes in the ciphertext. To optimally use this randomness, we therefore split the keys and ciphertexts in partitions of at most n_k and n_c attributes, respectively. For instance, if $\mathcal{S}$ denotes the set of attributes for which a key is requested, then $\mathcal{S}$ is split in partitions of maximum size n_k , i.e., $\mathcal{S} = \mathcal{S}_1 \cup \dots \cup \mathcal{S}_m$ such that $|\mathcal{S}_l| \leq n_k$ for each $l \in [m]$. Then, to avoid boundedness, we apply the RW13 trick by introducing one “randomizer” for each partition (both in the keys and ciphertexts).

3.1 The Rouselakis-Waters scheme

First, we briefly review the RW13 scheme [45]. Specifically, the secret keys and ciphertexts are of the form

$$\begin{aligned} \text{SK} &= (K = h^{\alpha - rb}, K' = h^r, \{K_{1,\text{att}} = h^{rb' + r_{\text{att}}(x_{\text{att}}b_1 + b_0)}, K_{2,\text{att}} = h^{r_{\text{att}}}\}_{\text{att} \in \mathcal{S}}), \\ \text{CT} &= (C = M \cdot e(g, h)^{\alpha s}, C' = g^s, \{C_{1,j} = B^{\lambda_j} \cdot (B')^{s_j}, \\ &\quad C_{2,j} = (B_1^{x_{\text{att}_j}} B_0)^{s_j}, C_{3,j} = g^{s_j}\}_{j \in [n_1]}), \end{aligned}$$

where $B = g^b, B_1 = g^{b_1}, B_0 = g^{b_0}$ and $B' = g^{b'}$ denote public keys, $r, r_{\text{att}} \in_R \mathbb{Z}_p$ are randomly chosen integers during the key generation for set $\mathcal{S}$, s, s_j are randomly chosen integers during encryption under access policy $\mathbb{A} = (\mathbf{A}, \rho)$ with $\mathbf{A} \in \mathbb{Z}_p^{n_1 \times n_2}$, and x_{att} is the representation of attribute att in $\mathbb{Z}_p$. We have also denoted $x_{\text{att}_j} = \rho(j)$ to clearly indicate the attributes in the ciphertext.

3.2 First attempt: a naive approach

Our first attempt is to directly replace the 1-degree polynomial, $x_{\text{att}}b_1 + b_0$, by an n -degree polynomial, i.e., $f_n(x_{\text{att}}) = \sum_{i=0}^n b_i x_{\text{att}}^i$ (where $n = n_k + n_c - 1$):

$$\begin{aligned} \text{SK} &= (K = h^{\alpha - rb}, K' = h^r, \{K_{1,\text{att}} = h^{rb' + \boxed{r_{\text{att}} f_n(x_{\text{att}})}}, K_{2,\text{att}} = h^{r_{\text{att}}}\}_{\text{att} \in \mathcal{S}}), \\ \text{CT} &= (C = M \cdot e(g, g)^{\alpha s}, C' = g^s, \{C_{1,j} = B^{\lambda_j} \cdot (B')^{s_j}, \\ &\quad C_{2,j} = \boxed{F_n(x_{\text{att}_j})^{s_j}} = \left(\prod_{i=0}^n B_i^{x_{\text{att}_j}^i}\right)^{s_j}, C_{3,j} = g^{s_j}\}_{j \in [n_1]}), \end{aligned}$$

where $B_i = g^{b_i}$ for all $i \in [0, n]$. We split $\mathcal{S}$ in partitions of maximum size n_k , and the rows of $\mathbf{A}$ in partitions of size n_c . We ensure that the same randomizer is used for all attributes in the same partition, i.e., set $r_{\text{att}} = r_{\text{att}'}$ and $s_j = s_{j'}$, if att and att' , and att_j and $\text{att}_{j'}$ are in the same partitions, respectively.

Unfortunately, the resulting scheme is insecure (see the full version [54] for a concrete attack). Roughly, the reason is that $C_{1,j} = g^{\lambda_j b + s_j b'}$ does not sufficiently hide $\lambda_j b$, because the same s_j is used for all attributes in the same partition. Therefore, we need to introduce more randomness.

3.3 Second (successful) attempt

We show how to use another polynomial to introduce enough randomness. Because we only need enough randomness for the ciphertext partitions, we require an $(n_c - 1)$ -degree polynomial. This polynomial, $f'_{n_c-1}(x_{\text{att}}) = \sum_{i=0}^{n_c-1} b'_i x_{\text{att}}^i$, will replace the “0-degree polynomial” b' . Because s_j provides randomness for one attribute, and f'_{n_c-1} provides randomness for $n_c - 1$ attributes in the partition, this sufficiently hides λ_j . The resulting scheme is then

$$\begin{aligned} \text{SK} &= (K = h^{\alpha - rb}, K' = h^r, \{K_{1,\text{att}} = h^{\boxed{rf'_{n_c-1}(x_{\text{att}})} + \boxed{r_{\text{att}}f_n(x_{\text{att}})}}, \\ &\quad K_{2,\text{att}} = h^{r_{\text{att}}}\}_{\text{att} \in \mathcal{S}}), \\ \text{CT} &= (C = M \cdot e(g, h)^{\alpha s}, C' = g^s, \{C_{1,j} = B^{\lambda_j} \cdot \boxed{F'_{n_c-1}(x_{\text{att}_j})^{s_j}}, \\ &\quad C_{2,j} = \boxed{F_n(x_{\text{att}_j})^{s_j}}, C_{3,j} = g^{s_j}\}_{j \in [n_1]}), \end{aligned}$$

where $F'_{n_c-1}(x_{\text{att}}) = \prod_{i=0}^{n_c-1} (B'_i)^{x_{\text{att}}^i} = g^{f'_{n_c-1}(x_{\text{att}})}$, and $B'_i = g^{b'_i}$ for all $i \in [0, n_c - 1]$. Note that this scheme is a generalization of RW13, because setting $n = n_c = n_k = 1$ yields RW13.

3.4 More efficient decryption

Generalizing the polynomial allows for an improved decryption efficiency. To understand why this yields a significant improvement, we briefly review the W11b scheme. We consider the keys and ciphertexts, which are of the form:

$$\begin{aligned} \text{SK} &= (K = h^{\alpha - rb}, K' = h^r, \{K_{\text{att}} = h^{rf_n(x_{\text{att}})}\}_{\text{att} \in \mathcal{S}}), \\ \text{CT} &= (C = M \cdot e(g, h)^{\alpha s}, C' = g^s, \{C_j = B^{\lambda_j} F_n(\rho(j))^s\}_{j \in [n_1]}), \end{aligned}$$

where $r, s \in \mathbb{Z}_p$ are randomly chosen integers, $B = g^b$ is a public key, α is the master key, and $\rho(j)$ is the j -th attribute of the policy of length n_1 , and λ_j is a sharing of s with respect to the policy. To decrypt, one computes

$$C / e(C', K) \cdot \prod_{j \in \mathcal{T}} e(C_j, K')^{\varepsilon_j} / \prod_{j \in \mathcal{T}} e(C', K_{\rho(j)})^{\varepsilon_j},$$

where ε_j for $j \in \mathcal{T} \subseteq [n_1]$ are integers that allow us to reconstruct the secret s . Each such product of pairings can be computed more efficiently by using

the bilinearity property on the shared arguments, e.g., $\prod_j e(K', C_j)^{\varepsilon_j}$ can be computed more efficiently by first multiplying C_j and then taking a pairing:

$$C/e \left(C', K \cdot \prod_{j \in \mathcal{T}} K_{\rho(j)}^{\varepsilon_j} \right) \cdot e \left(\prod_{j \in \mathcal{T}} C_j^{\varepsilon_j}, K' \right).$$

This requires only two pairing operations instead of $2|\mathcal{T}| + 1$. While decryption is very efficient, the drawback of W11b is that it is bounded in both the keys and ciphertexts. Because F_n embeds an n -degree polynomial, its randomized variant only provides sufficient randomness for $n + 1$ attributes shared between the keys and ciphertexts. In contrast, RW13 uses an implicit 1-degree polynomial for the hash. To provide unboundedness, each instance of the hash—which in itself only provides sufficient randomness for one attribute—is randomized. As a result, both the keys and ciphertexts consist of components of the form $(g^{t_i}, F_1(x_{\text{att}})^{t_i})$, such that each randomizer part g^{t_i} needs to be paired with the part involving the hash during decryption. Hence, a linear number of pairing operations is required during decryption instead of a constant. By generalizing the 1-degree polynomial, we can use the same randomizer for multiple attributes. Thus, we can achieve a similar speed-up in decryption efficiency as W11b, whilst benefiting from the unboundedness of RW13.

Note that this also illustrates why it is important that the BB hash is used for both the keys and the ciphertexts. For example, in the GPSW06 large-universe scheme [26, §5], the randomness provided by the hash is used only for the ciphertexts. As a result, the keys require a fresh randomizer for each attribute, and therefore, decryption costs at least one pairing operation per attribute.

4 Our construction

We now present the complete description of our scheme in the selective security setting obtained by using [50] (see the full version [54] for a fully secure version). In this scheme, we also introduce the mappings ι and τ , which map the attributes of the keys and ciphertexts, respectively, into arbitrary partitions of maximum sizes n_k and n_c .

Definition 6 (GLUE). *GLUE is defined as follows.*

- **Setup(λ):** *On 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 $e: \mathbb{G} \times \mathbb{H} \rightarrow \mathbb{G}_T$. It also defines the universe of attributes $\mathcal{U} = \mathbb{Z}_p$, chooses $n_k \in \mathbb{N}$ and $n_c \in \mathbb{N}$ as the maximum partition sizes of the keys and ciphertexts, respectively, and sets $n = n_k + n_c - 1$. It then generates random $\alpha, b, b_i, b'_{i'} \in_R \mathbb{Z}_p$ for all $i \in [0, n], i' \in [0, n_c - 1]$. It outputs $\text{MSK} = (\alpha, b, \{b_i, b'_{i'}\}_{i \in [0, n], i' \in [0, n_c - 1]})$ as its master secret key and publishes the master public key as*

$$\text{MPK} = (g, h, A = e(g, h)^\alpha, B = g^b, \{B_i = g^{b_i}, B'_{i'} = g^{b'_{i'}}\}_{i \in [n], i' \in [n_c - 1]}).$$

- $\text{KeyGen}(\text{MSK}, \mathcal{S})$: On input set of attributes $\mathcal{S}$, the key generation computes $m = \left\lceil \frac{|\mathcal{S}|}{n_k} \right\rceil$, defines $\iota: \mathcal{S} \rightarrow [m]$ such that $|\iota^{-1}(l)| \leq n_k$ for each $l \in [m]$, generates random integers $r, r_1, \dots, r_m \in_R \mathbb{Z}_p$, and outputs the secret key as

$$\text{SK}_{\mathcal{S}} = (K = h^{\alpha - rb}, K' = h^r, \iota, \{K_{1,\text{att}} = h^{r_{\iota(\text{att})}(\sum_{i=0}^n b_i x_{\text{att}}^i) + r(\sum_{i=0}^{n_c-1} b'_i x_{\text{att}}^i)}\}_{\text{att} \in \mathcal{S}}, \{K_{2,l} = h^{r_l}\}_{l \in [m]}).$$

- $\text{Encrypt}(\text{MPK}, \mathbb{A}, M)$: A message $M \in \mathbb{G}_T$ is encrypted under policy $\mathbb{A} = (\mathbf{A}, \rho)$ with $\mathbf{A} \in \mathbb{Z}_p^{n_1 \times n_2}$ and $\rho: [n_1] \rightarrow \mathcal{U}$ by computing $m' = \max\left(\left\lceil \frac{n_1}{n_c} \right\rceil, \max_{j \in [n_1]} |\rho^{-1}(\rho(j))|\right)$ and defining $\tau: [n_1] \rightarrow [m']$ such that $|\tau^{-1}(l')| \leq n_c$ for each $l' \in [m']$ and if $j, j' \in [n_1]$ with $j \neq j'$ such that $\rho(j) = \rho(j')$, then $\tau(j) \neq \tau(j')$, i.e., multiple occurrences of the same attribute are mapped to different partitions. (Note that this works because m' is defined to be at least as large as the maximum number of occurrences of each attribute.) It then generates random integers $s, s_1, \dots, s_{m'}, v_2, \dots, v_{n_2} \in_R \mathbb{Z}_p$ and outputs the ciphertext as

$$\text{CT}_{\mathbb{A}} = (C = M \cdot A^s, C' = g^s, \tau, \{C_{1,j} = B^{\lambda_j} \cdot \prod_{i=0}^{n_c-1} (B'_i)^{s_{\tau(j)} x_{\rho(j)}^i},$$

$$C_{2,j} = \prod_{i=0}^n B_i^{s_{\tau(j)} x_{\rho(j)}^i}\}_{j \in [n_1]}, \{C_{3,l'} = g^{s_{l'}}\}_{l' \in [m']}),$$

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

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

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

This can be computed more efficiently as

$$C / \left(e(C', K) \cdot e\left(\prod_{j \in \mathcal{Y}} C_{1,j}^{\varepsilon_j}, K'\right) \cdot \left(\prod_{l' \in [m']} e(C_{3,l'}, \prod_{j \in \mathcal{Y} \cap \tau^{-1}(l')} K_{1,\rho(j)}^{-\varepsilon_j}) \cdot \prod_{l \in [m]} e\left(\prod_{j \in \mathcal{Y} \cap \rho^{-1}(\iota^{-1}(l))} C_{2,j}^{-\varepsilon_j}, K_{2,l}\right)\right) \right),$$

which costs, on average, $2 + \left\lceil \frac{|\mathcal{Y}|}{n_k} \right\rceil + \left\lceil \frac{|\mathcal{Y}|}{n_c} \right\rceil$ pairing operations.

The scheme is correct, i.e., we have $C / e(C', K) = M \cdot e(g, h)^{\alpha s} \cdot e(g, h)^{-\alpha s + r s b} = M \cdot e(g, h)^{r s b}$ and

$$\begin{aligned} & \prod_{j \in \mathcal{Y}} (e(C_{1,j}, K') / e(C_{3,\tau(j)}, K_{1,\rho(j)}) \cdot e(C_{2,j}, K_{2,\iota(j)}))^{\varepsilon_j} \\ &= \prod_{j \in \mathcal{Y}} (e(g, h)^{r \lambda_j b + r s_{\tau(j)} \sum_{i=0}^{n_c-1} b'_i \rho(j)^i} \end{aligned}$$

$$\begin{aligned}
& \cdot e(g, h)^{-r_{\iota(\rho(j))} s_{\tau(j)} (\sum_{i=0}^n b_i \rho(j)^i) - r s_{\tau(j)} \sum_{i=0}^{n_c-1} b'_i \rho(j)^i} \\
& \cdot e(g, h)^{r_{\iota(\rho(j))} s_{\tau(j)} \sum_{i=0}^n b_i \rho(j)^i)^{\varepsilon_j} \\
& = \prod_{j \in \mathcal{I}} e(g, h)^{r \varepsilon_j \lambda_j b} = e(g, h)^{r b \sum_{j \in \mathcal{I}} \varepsilon_j \lambda_j} = e(g, h)^{r s b},
\end{aligned}$$

which yields the plaintext, i.e., $M \cdot e(g, h)^{r s b} / e(g, h)^{r s b} = M$.

Unique representation of attributes. In the scheme, we assume that any attribute string $\text{att} \in \{0, 1\}^*$ can be uniquely represented in $\mathbb{Z}_p$. In practice, this can be done by using a collision-resistant hash function $\mathcal{H}: \{0, 1\}^* \rightarrow \mathbb{Z}_p$ [47].

4.1 The associated pair encoding scheme

To prove security, we define the pair encoding scheme associated with our scheme in Definition 6, for which we use the variables $n_c, n_k, n, \mathcal{S}, \iota, \rho, \tau, n_1, n_2, \lambda_i, m, m'$ from Definition 6, as follows.

Definition 7 (PES for GLUE).

- $\text{Param}(\text{par}) \rightarrow 2n_c + n_k + 3$. Let $\mathbf{b} = (b, b_0, \dots, b_n, b'_0, \dots, b'_{n_c-1})$, where $n = n_k + n_c - 1$.
- $\text{EncKey}(\mathcal{S}) \rightarrow (\mathbf{r}, k', \{k_{1,\text{att}}\}_{\text{att} \in \mathcal{S}})$. Let $\mathbf{r} = (r, \{r_l\}_{l \in [m]})$, $k' = \alpha - rb$ and $k_{1,\text{att}} = r_{\iota(\text{att})} (\sum_{i=0}^n b_i x_{\text{att}}^i) + r (\sum_{i=0}^{n_c-1} b'_i x_{\text{att}}^i)$.
- $\text{EncCt}((\mathbf{A}, \rho)) \rightarrow (\mathbf{s}, \hat{\mathbf{s}}, \{c_{1,j}, c_{2,j}\}_{j \in [n_1]})$. Let $\mathbf{s} = (s, \{s_{l'}\}_{l' \in [m']})$ and $\hat{\mathbf{s}} = (\hat{v}_2, \dots, \hat{v}_{n_2})$, and $c_{1,j} = \mathbf{A}_j(s\mathbf{b}, \hat{\mathbf{s}})^\top + s_{\tau(j)} \sum_{i=0}^{n_c-1} b'_i \rho(j)^i$ and $c_{2,j} = s_{\tau(j)} \sum_{i=0}^n b_i \rho(j)^i$.

In Section 5, we prove security of the PES.

Theorem 1. *The PES for GLUE in Definition 7 satisfies the symbolic property (Definition 5).*

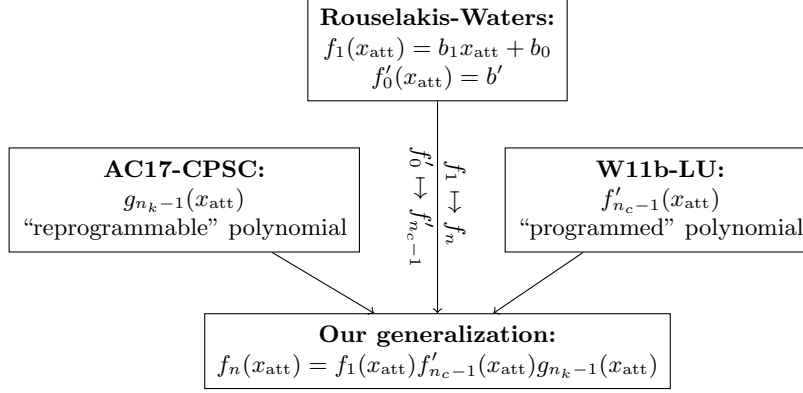
Therefore, instantiating the PES in the AC17 framework yields a fully secure scheme, and instantiating the PES with [50] yields a selectively secure scheme.

5 The security proof

While the construction of the scheme already provides some idea on why it may be secure, the proof requires some additional insights. First, we briefly review some important aspects in the Rouselakis-Waters proof, to gain some deeper understanding of the structure of the selective property proof. Then, we show how existing techniques can be combined to generalize the selective proof.

On a high level, the selective proof consists of the splitting of the n -degree polynomial, which provides randomness for the keys and ciphertexts, into a product of three polynomials f_1, f'_{n_c-1} and g_{n_k-1} . We use g_{n_k-1} for the keys,

Fig. 2: A high-level overview of the polynomials used in the combined proofs and our generalized selective proof.



and $f_1 f'_{n_c-1}$ for the challenge ciphertext. For the key polynomial g_{n_k-1} , we use Agrawal and Chase’s [4] techniques. In their selective proof of the CP-ABE scheme with short ciphertexts, they embed a polynomial in the public keys such that this polynomial can be reprogrammed to some polynomial associated with the set of attributes of the key. We call such polynomials “reprogrammable”. For the ciphertext polynomials f_1, f'_{n_c-1} , we use a combination of the proofs of RW13 [45] and W11b [55]. Roughly, in these proofs, they embed polynomials in the public keys, such that these polynomials are associated with the attributes that occur in the challenge access policy. We call such polynomials “programmed”. These techniques ensure that the polynomials evaluate to the right values when the set and policy attributes are evaluated. Figure 2 depicts the relationship between the existing proofs and ours. A similar approach can be taken in the co-selective proof, by swapping the roles of the two polynomials.

5.1 The Rouselakis-Waters proof

We briefly review the Rouselakis-Waters selective security proof. They use the commonly used “program-and-cancel” technique [26,56], in which the challenge access policy $\mathbb{A} = (\mathbf{A}, \rho)$ is split in two disjoint subsets with respect to the set of attributes $\mathcal{S}$ associated with the queried key. One subset $\mathcal{T} = \{j \in [n_1] \mid \rho(j) \in \mathcal{S}\}$ contains all the rows of matrix $\mathbf{A}$ for which the corresponding attribute is in the set, while the other set $\bar{\mathcal{T}} = \mathcal{S} \setminus \mathcal{T}$ contains the rest of the rows. In the simulation of the key, it then uses the property that if $\mathcal{S}$ does not satisfy $\mathbb{A}$, there exists a vector $\mathbf{w} = (1, w_2, \dots, w_{n_2}) \in \mathbb{Z}_p^{n_2}$ such that $\mathbf{A}_j \mathbf{w}^\top = 0$ for all $j \in \mathcal{T}$. Furthermore, they introduce the layering and individual randomness technique to ensure that the challenge ciphertext can be simulated. Roughly, they embed the attributes that occur in the policy in the public keys as well as their corresponding row in the matrix in a layered fashion, using an individual randomness

for each layer. During the key query and challenge phases, the randomizer embeds a single individual randomness to select the correct attribute layer, such that the key and ciphertext components can be simulated.

The selective symbolic security proof can be analogously structured, such that the substitutions ensure that all polynomials $k_{1,\text{att}}$, $c_{1,j}$ and $c_{2,j}$ evaluate to $\mathbf{0}$. In general, an attribute layer in the public keys is represented as a matrix $\mathbf{1}_{i,j}^{d_1 \times d_2} \in \mathbb{Z}_p^{d_1 \times d_2}$. Then, the appropriate attribute layer can be selected by the vector that represents the associated individual randomness, i.e., $\mathbf{1}_i^{d_1} \in \mathbb{Z}_p^{d_1}$. Multiplying this vector with the matrix yields $\bar{\mathbf{1}}_j^{d_2} \in \mathbb{Z}_p^{d_2}$. Effectively, only one remaining entry needs to evaluate to $\mathbf{0}$. This is done by embedding the policy in the public keys in a certain way. Specifically, in $c_{1,j} = \mathbf{A}_j(sb, \hat{\mathbf{s}})^\top + s_j b'$, where $\mathbf{A}_j(sb, \hat{\mathbf{s}})^\top = \sum_{k \in [n_2]} A_{j,k} v_k$ (where $v_1 = sb$) and $s_j b'$ are supposed to cancel out one another. This can be ensured by embedding all rows $\mathbf{A}_j$ of the policy in b' , and using individual randomness (represented as a vector) to select the appropriate row. More concretely, b' could be substituted by $-\sum_{j \in [n_1], k \in [n_2]} A_{j,k} \mathbf{1}_{j,k}^{d_1 \times d_2}$ and s_j by $\mathbf{1}_j^{d_1}$ such that $s_j b' = -\sum_{k \in [n_2]} A_{j,k} \bar{\mathbf{1}}_k^{d_2}$. Then, if v_k is substituted by $\bar{\mathbf{1}}_{(0,k)}^{d_2}$, s by $\mathbf{1}_0^{d_1}$ and b by $\mathbf{1}_{0,(0,1)}^{d_1 \times d_2}$, then $\mathbf{A}_j(sb, \hat{\mathbf{s}})^\top = \sum_{k \in [n_2]} A_{j,k} \bar{\mathbf{1}}_{(0,k)}^{d_2}$. Similarly, $c_{2,j}$ evaluates to $\mathbf{0}$ by defining b_0 and b_1 such that for each attribute layer associated with row j , the 1-degree polynomial $F_{1,j}(x) = x - \rho(j)$ is embedded. The individual randomness ensures that this polynomial is selected with s_j . To ensure that the key $k_{1,\text{att}} = r_{\text{att}} f_1(x_{\text{att}}) + r b'$ evaluates to $\mathbf{0}$, we embed the vector $\mathbf{w}$ in r and r_{att} , which ensures that all attribute layers that are also in the set $\mathcal{S}$ go to $\mathbf{0}$. For those attribute layers that are not in the set $\mathcal{S}$, i.e., $\bar{\mathcal{Y}}$, we ensure that layers $F_{1,j}(x_{\text{att}}) \mathbf{A}_j \mathbf{w}^\top$ in $r_{\text{att}} f_1(x_{\text{att}})$ cancel out layers $\mathbf{A}_j \mathbf{w}^\top$ in $r b'$. Roughly, this is done by embedding $\frac{1}{F_{1,j}(x_{\text{att}})}$ in r_{att} for all $j \in \bar{\mathcal{Y}}$, such that the $\mathbf{A}_j \mathbf{w}^\top$ in the two summands cancel out one another.

5.2 Generalizing the Rouselakis-Waters proof

We generalize the Rouselakis-Waters proof by layering the policy embedded in the public keys in a partition-wise fashion instead of attribute-wise. In this way, the ciphertext-specific variable $s_{l'}$, which is used for all attributes in the same partition, can select all attributes associated within the l' -th partition. As such, in the computation of $c_{1,j}$ and $c_{2,j}$, $s_{\tau(j)} f'_{n_c-1}(\rho(j))$ needs to cancel out $\mathbf{A}_j(sb, \hat{\mathbf{s}})^\top$ and $s_{\tau(j)} f_n(\rho(j))$ needs to go to $\mathbf{0}$. To this end, we need to substitute f'_{n_c-1} in such a way that it outputs exactly $-\sum_{k \in [n_2]} A_{j,k} \bar{\mathbf{1}}_{(0,k)}^{d_2}$ when $s_{\tau(j)} f'_{n_c-1}(\rho(j))$ is computed. Similarly, the key-specific variable r_l needs to be constructed such that $k_{1,\text{att}}$ goes to $\mathbf{0}$, which happens when $r_{\iota(\text{att})} f_n(x_{\text{att}})$ cancels out $r f'_{n_c-1}(x_{\text{att}})$.

To accomplish this, we define f_n and f'_{n_c-1} as mentioned before, i.e., $f_n(x_{\text{att}}) = f_1(x_{\text{att}}) f'_{n_c-1}(x_{\text{att}}) g_{n_k-1}(x_{\text{att}})$. Roughly, we substitute f_1 in the same way as in the Rouselakis-Waters proof, while we use the polynomials f'_{n_c-1} and g_{n_k-1} to ensure that $c_{1,j}$ and $c_{2,j}$, and $k_{1,\text{att}}$ evaluate to $\mathbf{0}$, respectively. Because we are

given the challenge access structure a priori, i.e., as input to EncB, we can program these as required in the substitutions of the polynomials f_1 and f'_{n_c-1} in the public keys. Concretely, we substitute $b_0, \dots, b_n$ such that

$$\begin{aligned} f_n(x_{\text{att}}) &: \sum_{j \in [n_1], k \in [n_2]} A_{j,k} F_{n,j,k}(x_{\text{att}}) \\ &= \sum_{j \in [n_1], k \in [n_2]} A_{j,k} \underbrace{F_{1,j}(x_{\text{att}}) F'_{n_c-1,j}(x_{\text{att}}) \hat{G}_{n_k-1,j,k}(x_{\text{att}})}_{F_{n,j,k}(x_{\text{att}})}, \end{aligned}$$

where $F_{1,j}(x_{\text{att}}) = (x_{\text{att}} - \rho(j))$ and

$$F'_{n_c-1,j}(x_{\text{att}}) = \sum_{i=0}^{n_c-1} d'_{i,j} x_{\text{att}}^i = \prod_{j' \in \chi_j \setminus \{j\}} \frac{x_{\text{att}} - \rho(j')}{\rho(j) - \rho(j')},$$

with $\chi_j = \{j' \in [n_1] \mid \tau(j') = \tau(j)\}$. We refer to these polynomials as the “programmed” polynomials. These ensure that $F_{n,j}(\rho(j')) = 0$ for all $j' \in \chi_j$, $F'_{n_c-1,j}(\rho(j)) = 1$ and $F'_{n_c-1,j'}(\rho(j)) = 0$ for all $j' \in \chi_j \setminus \{j\}$. Then, $c_{1,j}$ and $c_{2,j}$ evaluate to $\mathbf{0}$, if we substitute

$$f'_{n_c-1}(x_{\text{att}}) : \sum_{j \in [n_1], k \in [n_2]} A_{j,k} F'_{n_c-1,j}(x_{\text{att}}) \mathbf{1}_{(1,\tau(j)),(0,k)}^{d_1 \times d_2}.$$

In contrast, the set of attributes associated with a key is given after the public keys have been established, i.e., as input to EncR, so we need to somehow achieve that we can program the polynomial $\hat{G}_{n_k-1,j}$ after the public keys are generated. We do this by setting

$$\hat{G}_{n_k-1,j,k}(x_{\text{att}}) = \sum_{i=0}^{n_k-1} \mathbf{1}_{(1,\tau(j)),(1,i,j,k)}^{d_1 \times d_2} x_{\text{att}}^i,$$

such that $\hat{G}_{n_k-1,j,k}$ constitutes a “reprogrammable” polynomial. It can be reprogrammed by ensuring that r_l consists of the coefficients $u_{i,j,l}$ of some target polynomial(s), i.e., by multiplying

$$\left(\sum_{i=0}^{n_k-1} \mathbf{1}_{(1,\tau(j)),(1,i,j,k)}^{d_1 \times d_2} x_{\text{att}}^i \right) \left(\sum_{i=0}^{n_k-1} u_{i,j,l} \bar{\mathbf{1}}_{(1,i,j,k)}^{d_2} \right) = \sum_{i=0}^{n_k-1} u_{i,j,l} \mathbf{1}_{(1,\tau(j))}^{d_1} x_{\text{att}}^i.$$

We use this to “reprogram” polynomial $\hat{G}_{n_k-1,j,k}(x_{\text{att}})$ for all $j \in \bar{\mathcal{T}}$, which is well-defined, because $\rho(j) \notin \mathcal{S}$. This then yields $F'_{n_c-1,j}(x_{\text{att}})$ and cancels out the $F'_{n_c-1,j}(x_{\text{att}})$ in $r f'_{n_c-1}(x_{\text{att}})$ part in $k_{1,\text{att}}$ for all $j \in \bar{\mathcal{T}}$. Note that, like in Rouselakis-Waters, we have $\mathbf{A}_j \mathbf{w}^\top = 0$ for all $j \in \mathcal{Y}$, so those layers automatically go to $\mathbf{0}$ in the computation of $k_{1,\text{att}}$. Hence, for each partition $\Psi_l = \{\text{att} \in \mathcal{S} \mid \iota(\text{att}) = l\}$ with $l \in [m]$, we define the polynomial

$$G_{n_k-1,j,l}(x_{\text{att}}) = \sum_{i=0}^{n_k-1} u_{i,j,l} x_{\text{att}}^i = \sum_{\text{att}' \in \Psi_l} \frac{1}{F_{1,j}(x_{\text{att}'})} \prod_{\text{att}'' \in \Psi_l \setminus \{\text{att}'\}} \frac{x_{\text{att}} - x_{\text{att}''}}{x_{\text{att}'} - x_{\text{att}''}},$$

for each $j \in \bar{T}$, such that $G_{n_k-1,j,l}(x_{\text{att}}) = \frac{1}{F_{1,j}(x_{\text{att}})}$ for all $\text{att} \in \Psi_l$.

Putting it all together, we substitute $b_0, \dots, b_n$ with coefficients such that the polynomial f_n is substituted by

$$\sum_{j \in [n_1], k \in [n_2]} A_{j,k} F_{n,j,k}(x_{\text{att}}) = \sum_{j \in [n_1], k \in [n_2]} A_{j,k} \sum_{i=0}^n d_{i,j,k} x_{\text{att}}^i,$$

where

$$d_{i,j,k} = \sum_{i' \in [\overline{n_k-1}], i'' \in [\overline{n_c-1}]: i' + i'' = i} d'_{i',j} \mathbf{1}_{(1,\tau(j)), (1,i'',j,k)}^{d_1 \times d_2}.$$

5.3 The selective symbolic property

We prove the selective symbolic property, using m, m', τ, ι as in Section 4 and $F_{n,j,k}, d_{i,j,k}, F'_{n_c-1,j}, d'_{i,j}, G_{n_k-1,j,l}, u_{i,j,l}$ and χ_j as in Section 5.2. For simplicity of notation, we write the second index of $\mathbf{1}^{d_1 \times d_2}$ and the index of $\bar{\mathbf{I}}^{d_2}$ as a tuple $(1, i, j, k)$ (with $i \in [\overline{n_k}], j \in [n_1], k \in [n_2]$) such that it represents a unique integer in $[n_2 + 1, ((n_k + 1)n_1 + 1)n_2]$. Note that we use $(0, k)$ to indicate the first n_2 columns, which are associated with only k and not (i, j) . For the first index of $\mathbf{1}^{d_1 \times d_2}$ and the index of $\mathbf{1}^{d_1}$, we start counting at 0. Note that, therefore, $d_1 = n_2 + 1$ and $d_2 = ((n_k + 1)n_1 + 1)n_2$. The substitutions are, for all $i \in [\overline{n}], i' \in [\overline{n_k}], l \in [m], l' \in [m'], k \in [2, n_2]$:

$$\begin{aligned} b &: \mathbf{1}_{0, (0,1)}^{d_1 \times d_2}, & b_i &: \sum_{j \in [n_1], k \in [n_2]} A_{j,k} d_{i,j,k} \\ b'_{i'} &: \sum_{j \in [n_1], k \in [n_2]} A_{j,k} d'_{i',j} \mathbf{1}_{(1,\tau(j)), (0,k)}^{d_1 \times d_2}, & s &: \mathbf{1}_0^{d_1} \\ s_{l'} &: -\mathbf{1}_{(1,l')}^{d_1}, & \alpha &: \mathbf{1}_0^{d_1}, & \hat{v}_k &: \bar{\mathbf{I}}_{(0,k)}^{d_2}, & r &: \sum_{k' \in [n_2]} w_{k'} \bar{\mathbf{I}}_{(0,k')}^{d_2} \\ r_l &: - \sum_{i' \in [\overline{n_k-1}], j' \in \bar{T}, k' \in [n_2]} w_{k'} u_{i',j',l} \bar{\mathbf{I}}_{(1,i',j',k')}^{d_2}. \end{aligned}$$

Then, $c_{1,j}, c_{2,j}, k'$ and $k_{1,\text{att}}$ indeed go to $\mathbf{0}$ (see the full version [54]).

5.4 Co-selective symbolic property

We prove that the co-selective symbolic property also holds. For this proof, the roles of the reprogrammable and the programmed polynomial are reversed, because we are allowed to use an attribute set $\mathcal{S}$ in the programming of the public keys and secret keys, and the policy $\mathbb{A}$ only for the ciphertext. Similarly as in the selective case, we use the Rouselakis-Waters proof as inspiration for the structure of the proof. In particular, we use the selective security proof of

the KP-ABE variant of the Rouselakis-Waters scheme (which is analogous to the co-selective proof of the CP-ABE variant).

In this proof, we substitute the polynomials with:

$$f_n(x_{\text{att}}) : \sum_{l \in [m]} \left(\underbrace{\hat{F}_{n_c-1,1,l}(x_{\text{att}})G_{n_k,l}(x_{\text{att}})}_{G_{n,l}(x_{\text{att}})} - \hat{F}_{n_c-1,2,l}(x_{\text{att}}) \right),$$

$$f'_{n_c-1}(x_{\text{att}}) : \hat{F}_{n_c-1,2,0}(x_{\text{att}}),$$

where we define $G_{n,l}(x_{\text{att}}) = \sum_{i=0}^n \tilde{u}_{i,l} x_{\text{att}}^i$, and

$$G_{n_k,l}(x_{\text{att}}) = \sum_{i=0}^{n_k} u_{i,l} x_{\text{att}}^i = \prod_{\text{att}' \in \Psi_l} (x_{\text{att}} - x_{\text{att}'}),$$

is a programmed polynomial with $G_{n_k,l}(x_{\text{att}}) = 0$ for $\text{att} \in \Psi_l$, and

$$\hat{F}_{n_c-1,1,l}(x_{\text{att}}) = \sum_{i=0}^{n_c-1} \mathbf{1}_{(1,i,l),l}^{d_1 \times d_2} \text{ and } \hat{F}_{n_c-1,2,l}(x_{\text{att}}) = \sum_{i=0}^{n_c-1} \mathbf{1}_{(2,i),l}^{d_1 \times d_2}$$

are the reprogrammable polynomials, to be reprogrammed to

$$F_{n_c-1,1,j,l}(x_{\text{att}}) = \sum_{i=0}^{n_c-1} \tilde{d}_{i,j,l} x_{\text{att}}^i = \frac{1}{G_{n_k,l}(\rho(j))} F'_{n_c-1,j}(x_{\text{att}}),$$

$$F'_{n_c-1,j}(x_{\text{att}}) = \sum_{i=0}^{n_c-1} d'_{i,j} x_{\text{att}}^i,$$

respectively, for $j \in \bar{\mathcal{T}}$. Note that $F_{n_c-1,1,j,l}(\rho(j)) = \frac{1}{G_{n_k,l}(\rho(j))}$ if $j \in \bar{\mathcal{T}}$ and $F_{n_c-1,1,j',l}(\rho(j)) = 0$ for $j' \in \chi_j$. Concretely, we have

$$\tilde{u}_{i,l} = \sum_{i' \in \overline{[n_c-1]}, i'' \in \overline{[n_k]} : i' + i'' = i} u_{i'',l} \mathbf{1}_{(1,i',l),l}^{d_1 \times d_2}.$$

Then, for $i \in \overline{[n_c-1]}, i' \in [n_c, n], l \in [m], l' \in [m'], k \in [n_2]$ we make the following substitutions:

$$b : \mathbf{1}_{0,0}^{d_1 \times d_2}, \quad b_i : \sum_{l \in [m]} \left(\tilde{u}_{i,l} - \mathbf{1}_{(2,i),l}^{d_1 \times d_2} \right), \quad b_{i'} : \sum_{l \in [m]} \tilde{u}_{i,l}$$

$$b'_i : \mathbf{1}_{(2,i),0}^{d_1 \times d_2}, \quad \alpha : \mathbf{1}_0^{d_1}, \quad \hat{v}_k : w_k \bar{\mathbf{1}}_0^{d_2}, \quad r : \bar{\mathbf{1}}_0^{d_2}, \quad r_l : \bar{\mathbf{1}}_l^{d_2}$$

$$s : \mathbf{1}_0^{d_1}, \quad s_{l'} : - \sum_{i \in \overline{[n_c-1]}, j \in \chi_{l'} \cap \bar{\mathcal{T}}, k \in [n_2]} A_{j,k} w_k \left(d'_{i,j} \mathbf{1}_{(2,i)}^{d_1} + \tilde{d}_{i,j} \right),$$

where $\tilde{d}_{i,j} = \sum_{l \in [m]} \tilde{d}_{i,j,l} \mathbf{1}_{(1,i,l)}^{d_1}$, $d_1 = n_c(m+1) + 1$ and $d_2 = m+1$. For simplicity, we use tuple notations $(1, i, l)$ and $(2, i)$ for the first index of $\mathbf{1}^{d_1 \times d_2}$ and the index of $\mathbf{1}^{d_1}$ for all $i \in \overline{[n_c-1]}, l \in [m]$, which map injectively into the intervals $[2, n_c m + 1]$ and $[n_c m + 2, d_1]$, respectively, and index 0 maps to the first row. Then, $c_{1,j}, c_{2,j}, k'$ and $k_{1,\text{att}}$ indeed go to $\mathbf{0}$ (see the full version [54]).

6 Performance analysis

We analyze the efficiency of our schemes. An important aspect in this analysis are the parameters n_k and n_c , which are chosen during the setup (e.g., by a practitioner). On a high level, the key generation, encryption and decryption of the selectively secure version of GLUE (via [50]) incur the following costs:

- KeyGen: $2 + |\mathcal{S}| + \left\lceil \frac{|\mathcal{S}|}{n_k} \right\rceil$ exponentiations in $\mathbb{H}$;
- Encrypt: 1 exponentiation in $\mathbb{G}_T$, $1 + \left\lceil \frac{n_1}{n_c} \right\rceil$ exponentiations, n_1 multi-exponentiations with $n_c + 1$ bases and n_1 multi-exponentiations with $n_k + n_c$ bases in $\mathbb{G}$;
- Decrypt: roughly $2 + \left\lceil \frac{|\mathcal{T}|}{n_k} \right\rceil + \left\lceil \frac{|\mathcal{T}|}{n_c} \right\rceil$ pairing operations.

The efficiency of these algorithms depends on the one hand on the efficiency of these operations, and on the other hand on the choices of n_k and n_c . By analyzing these rough costs from a mathematical point of view, the trade-off between the encryption and decryption efficiency is optimal when $n_k = n_c$ (which follows from the arithmetic mean-harmonic mean inequality). However, when the set of attributes $\mathcal{S}$ is large, and n_k is small, it may occur that all matching attributes are in different partitions. As such, choosing n_k to be larger, e.g., $n_k = 10$, ensures that the matching attributes are in the same key partitions with a large probability, and therefore the actual number of pairing operations is higher. In general, it holds that, the larger the partition sizes, the fewer pairing operations are needed during decryption. Unfortunately, the drawback is that encryption becomes more expensive, meaning that we may want to use the online/offline version of the scheme in practice. In the full version [54], we give more details on how a suitable partition size may be chosen. For our analysis, we consider three parameter settings: $(n_k, n_c) \in \{(3, 3), (5, 5), (10, 5)\}$. Furthermore, for the variant that supports OSW-type negations, we consider $|\mathcal{S}| \in \{1, 5\}$.

On the comparability of the schemes. For a fair comparison, we optimize all the schemes in the same way when instantiating the schemes in the asymmetric setting [44]. Specifically, we optimize the decryption and encryption efficiency. For the analysis of the RW13 [45], HW14 [31], Att19-I-CP and Att19-I-CP-OO [11] schemes, we have used the performance analysis of our associated schemes for $n_k = n_c = 1$ (which have the same encodings). We also compare our monotone schemes with AHM+16 [12] and ABGW17 [7] (see the full version [54] for the compared instantiations). To place the costs based on our theoretical analyses of the selectively secure instantiations (via [50]) in the full-security setting, we multiply the costs for each element and operation in $\mathbb{G}$ and $\mathbb{H}$ by a factor 2. This overhead corresponds to the most efficient instantiation of the schemes in the AC17 framework [3]. For all schemes, we also assume that the access policies are Boolean formulas, so that for decryption, it is ensured that $\varepsilon_j \in \{0, 1\}$ [38].

Estimates based on benchmarks in RELIC. We estimate the computational costs of the schemes by obtaining benchmarks of various algorithms and extrapolating the results by analyzing the descriptions of the schemes. We analyze the efficiency in this way for two reasons. First, it allows us to analyze the efficiency of many scheme configurations without having to implement each one, which is a cumbersome and error-prone effort. Second, it allows us to compare the schemes more accurately and more fairly. Currently, the simplest and most popular way [45,2,53] to benchmark schemes is by using Charm [5]. However, Charm only supports curves that do not provide sufficient security anymore, and de la Piedra et al. [44] show that benchmarking the schemes on these curves yields inaccurate and unfair comparisons. To compare the schemes more accurately and fairly, we estimate⁶ the costs of the schemes by applying their approaches [44]. In particular, we have run benchmarks in RELIC [8], a cryptographic library for efficient implementations of pairing-based cryptography on state-of-the-art elliptic curves. This library has implementations for exponentiations, including fixed-base and multi-base variants. In fixed-base exponentiation, the base g in g^x is fixed after setup, and as such, a precomputation table can be made to speed up the computation [20]. In a multi-base exponentiation, the product of multiple exponentiations, e.g., $g_1^{x_1} g_2^{x_2}$, is computed more efficiently [41]. We have run these benchmarks on a 1.6 GHz Intel i5-8250U processor for the BLS12-446 curve [15], which provides approximately 132-134 bits of security [29,28]. These benchmarks can be found in the full version [54] and are used in our analysis.

Comparison. Tables 3a and 3b show the performances of all unbounded schemes using a BB hash that support MSPs and NMSPs. The tables illustrate that the decryption algorithms of our regular schemes are significantly faster than the established schemes. While the encryption costs increase compared to the other schemes, our online/offline versions also provide a solution in this regard, incurring minimal online costs. This comes with a slight trade-off in the ciphertext size and the decryption efficiency compared to the regular version, but overall, our online/offline schemes outperform the established schemes in all algorithms. Importantly, the decryption of our schemes supporting negations with parameters $n_k = n_c = 5$ outperforms the only other unbounded OSW-type non-monotone scheme. Importantly, decryption is faster by a factor 4 for non-negated attributes, and faster by a factor 4-5 for negated attributes and $|\mathcal{S}| = 5$, bringing down the costs from almost two seconds to 382 ms. As a result, our schemes could provide a more attractive building block for OSWOT-type non-monotone schemes, as they support more efficient decryption algorithms for negated and non-negated attributes, and for small and large sets of attributes for each label. Furthermore, owing to the online/offline extensions, the key generation and encryption algorithms do not need to suffer from heavy online computations. Instead, encrypting users need to store only 3.17-10.17 kilobytes per one intermediate ciphertext of the first type and sufficient of the second type for ten attributes (depending on

⁶ Although approximated theoretically, we expect our estimates to be close to the costs of actual implementations [54].

Table 3: Rough estimates of the storage costs of the secret keys and the ciphertexts in kilobytes (KB), where 1 KB = 1024 bytes, and the (online) computational costs incurred by the key generation, encryption and decryption algorithms of $\text{GLUE}_{(n_k, n_c)}$ (and its online/offline (suffixed with “OO”) and OSW-non-monotone (suffixed with “N”) variants) and the other unbounded CP-ABE schemes, expressed in milliseconds (ms), for 10 and 100 attributes. Note that the offline key generation and encryption costs of each online/offline scheme are equal to the key generation and encryption costs of its regular version. The code used to generate these benchmarks is available at [51].

Scheme		Storage costs						Computational costs					
		MPK	SK		CT		KeyGen		Encrypt		Decrypt		
			10	100	10	100	10	100	10	100	10	100	
Regular	RW13 [45]	1.42	4.86	44.58	4.05	33.58	26.0	238.7	32.9	305.9	46.2	375.2	
	AHM+16 [12] ($n_k = 2$)	1.75	5.3	45.02	6.45	55.67	16.5	122.9	40.8	368.3	43.7	317.4	
	ABGW17 [7]	1.42	2.65	22.51	3.94	33.47	14.2	120.5	32.3	305.2	27.9	192.4	
	GLUE _(3,3)	2.08	3.53	30.02	3.39	26.36	18.9	160.7	59.8	571.4	24.3	133.9	
	GLUE _(5,5)	2.74	3.09	26.93	3.17	24.83	16.5	144.2	82.3	800.4	17.0	82.8	
	GLUE _(10,5)	3.28	2.87	24.72	3.17	24.83	15.4	132.3	102.1	998.4	15.1	64.5	
O/O	HW14 [31]	1.42	5.23	48.29	4.79	41.0	0	0	0	0	51.5	416.2	
	GLUE _(3,3)	2.08	3.9	33.73	5.62	48.62	0	0	0	0	33.6	202.6	
	GLUE _(5,5)	2.74	3.46	30.64	6.88	61.94	0	0	0	0	27.6	157.5	
	GLUE _(10,5)	3.28	3.24	28.43	8.74	80.49	0	0	0	0	24.2	123.4	

(a) Schemes supporting MSPs only.

	Scheme	Storage costs					Computational costs									
		MPK	SK		CT		KeyGen		Encrypt		Decrypt					
			10	100	10	100	10	100	10	100	10	100	10	100		
Regular	Att19-I-CP [11]	1.42	10.89	100.01	6.37	55.76	59.0	541.8	66.6	637.5	51.7	380.7	55.3	367.9	216.2	1779.0
	GLUE-N _(3,3)	2.08	7.59	63.66	5.04	41.19	44.8	385.9	90.0	865.1	29.8	139.4	62.3	374.9	109.5	745.5
	GLUE-N _(5,5)	2.74	6.49	55.95	4.6	38.1	40.1	352.8	111.4	1086.0	22.4	88.3	55.3	367.9	55.3	382.5
	GLUE-N _(10,5)	3.28	5.94	50.44	4.6	38.1	37.7	329.2	131.2	1284.0	20.6	70.0	78.5	599.4	78.5	614.1
O/O	Att19-I-CP-OO	1.42	12.01	111.15	7.11	63.18	0	0	0	0	61.0	461.3	64.6	448.5	225.5	1859.6
	GLUE-N _(3,3)	2.08	8.7	74.79	7.27	63.46	0	0	0	0	46.7	275.2	79.3	510.6	126.4	881.3
	GLUE-N _(5,5)	2.74	7.6	67.08	8.31	75.21	0	0	0	0	41.8	235.9	74.7	515.5	74.7	530.1
	GLUE-N _(10,5)	3.28	7.05	61.58	10.17	93.77	0	0	0	0	36.8	185.9	94.7	715.4	94.7	730.0

(b) Schemes supporting OSW-type negations. The decryption costs are for non-negated, and negated policies with $|S| \in \{1, 5\}$, respectively.

the instantiation). This means that, with just a megabyte of space, a user can store at least 100 intermediate ciphertexts for a total of 1000 attributes. For computing devices such as computers and smartphones, which have an abundance of storage space nowadays, this is a more than acceptable trade-off. Similarly, key generation authorities can store intermediate keys for at least 286 users and 2860 attributes with just a megabyte of space. Thus, with gigabytes, an authority can precompute keys for hundreds of thousands of users and millions of attributes.

7 Applying multiple instantiations of GLUE in practice

The flexible efficiency trade-offs that GLUE provides can be exploited in practice. In particular, practitioners can choose one suitable instantiation of GLUE, or choose multiple instantiations of GLUE to support different computational devices. Interestingly, by using the direct sum with parameter reuse transformation of Attrapadung [11], GLUE would be able to support multiple instances of itself simultaneously, such that the size of the master public key is upper-bounded in the maximum size of the public keys of all instances. This may be useful in settings in which the devices have varying computational resources. For instance, in the WLAN use case considered by ETSI [24], the decryption devices may be any mobile device in a network, including more constrained devices such as smartwatches. For those devices, it is more beneficial to use a scheme with fast decryption, e.g., $\text{GLUE}_{(5,5)}$, while for faster devices, it is sufficient to employ a scheme with slower decryption, e.g., RW13. In WLAN systems, the access point sends, for instance, an encrypted WPA2-PSK key to the connecting device, which can decrypt it if it satisfies the policy. Because this exchange is interactive, the connecting device and access point could first negotiate on the particular instance of GLUE for which the connecting device has a secret key before encrypting the WPA2-PSK key. In non-interactive systems, e.g., cloud settings [24], it may be more desirable to use multiple instances in parallel. Powerful devices could, for example, use multiple instances to support less powerful devices that only use the more efficient instances. For example, powerful decryption devices could have keys for both $\text{GLUE}_{(5,5)}$ and RW13, while less powerful encryption devices use RW13 or an online/offline variant of GLUE to encrypt.

8 Future work

For future work, it would be interesting to investigate the following. First, we have proven our scheme secure in the AC17 framework, which yields full security under a q -type assumption. Although frameworks exist that prove security generically under static assumptions [9,21,10], these use a strong security notion called the master-key hiding property. Like other unbounded ABE using a BB hash, ours does not satisfy this property [9]. To achieve such strong notions of security, more intricate proof techniques need to be devised, such as [22]. Second, we have analyzed the efficiency of the schemes on the BLS12-446 curve. Presumably, the encryption and decryption costs can improve if curves such as KSS16-339 [33] are used, which provide faster arithmetic in $\mathbb{G}$ and provide more efficient products of pairing operations [23]. GLUE (and RW13) may also benefit from fixed-base multi-base exponentiations [41], which RELIC does not support. Finally, while we have given the first steps towards realizing more efficient schemes supporting OSWOT-type negations, we have not explicitly specified these schemes. Our analysis in Section 6 indicates that any such schemes would benefit from the efficiency of our schemes, including those supporting OSW-type negations (see the full version [54] for more details).

9 Conclusion

We have proposed GLUE, a new unbounded large-universe scheme with flexible efficiency trade-off. This scheme is a generalization of RW13 [45] and W11b [55], in that it supports polynomials of any degree for the Boneh-Boyen hash. To optimally use the randomness provided by the hash, we use the partitioning approach (previously also used by AHM+16 [12]), splitting the sets of attributes and the policies in partitions of maximum sizes n_k and n_c , respectively. This allows for a decreased number of pairing operations required during decryption compared to RW13 (and related variants). Roughly, the pairing costs decrease by a factor $n_k = n_c$ (if chosen to be equal). Along the way, we have also introduced new proof techniques. These ensure that the randomness provided by the BB hash can be used for both the keys and ciphertexts in the unbounded setting. Finally, we have shown that our schemes indeed outperform existing schemes using a BB hash in the decryption, and notably, all schemes supporting OSW-type negations. Because our non-monotone schemes are unbounded and faster than 1.2 seconds in all algorithms on a laptop, even for large policies and sets, they are more suitable for practice than existing non-monotone schemes.

References

1. Agrawal, S., Chase, M.: A study of pair encodings: Predicate encryption in prime order groups. In: TCC. pp. 259–288. Springer (2016)
2. Agrawal, S., Chase, M.: FAME: fast attribute-based message encryption. In: CCS. pp. 665–682. ACM (2017)
3. Agrawal, S., Chase, M.: Simplifying design and analysis of complex predicate encryption schemes. In: EUROCRYPT. pp. 627–656. Springer (2017)
4. Agrawal, S., Chase, M.: Simplifying design and analysis of complex predicate encryption schemes. Cryptology ePrint Archive, Report 2017/233 (2017)
5. Akinyele, J.A., Garman, C., Miers, I., Pagano, M.W., Rushanan, M., Green, M., Rubin, A.D.: Charm: a framework for rapidly prototyping cryptosystems. J. Cryptogr. Eng. **3**(2), 111–128 (2013)
6. Ambrona, M.: Generic negation of pair encodings. In: Garay, J.A. (ed.) PKC. LNCS, vol. 12711, pp. 120–146. Springer (2021)
7. Ambrona, M., Barthe, G., Gay, R., Wee, H.: Attribute-based encryption in the generic group model: Automated proofs and new constructions. In: CCS. pp. 647–664. ACM (2017)
8. Aranha, D.F., Gouvêa, C.P.L., Markmann, T., Wahby, R.S., Liao, K.: RELIC is an Efficient LIBrary for Cryptography. <https://github.com/relic-toolkit/relic> (2020)
9. Attrapadung, N.: Dual system encryption via doubly selective security: Framework, fully secure functional encryption for regular languages, and more. In: EUROCRYPT. pp. 557–577. Springer (2014)
10. Attrapadung, N.: Dual system encryption framework in prime-order groups via computational pair encodings. In: ASIACRYPT. pp. 591–623. Springer (2016)
11. Attrapadung, N.: Unbounded dynamic predicate compositions in attribute-based encryption. In: EUROCRYPT. pp. 34–67. Springer (2019)

12. Attrapadung, N., Hanaoka, G., Matsumoto, T., Teruya, T., Yamada, S.: Attribute based encryption with direct efficiency tradeoff. In: ACNS. pp. 249–266. Springer (2016)
13. Attrapadung, N., Libert, B., de Panafieu, E.: Expressive key-policy attribute-based encryption with constant-size ciphertexts. In: PKC. pp. 90–108. Springer (2011)
14. Attrapadung, N., Tomida, J.: Unbounded dynamic predicate compositions in ABE from standard assumptions. In: ASIACRYPT. pp. 405–436. Springer (2020)
15. Barreto, P.S.L.M., Lynn, B., Scott, M.: Constructing elliptic curves with prescribed embedding degrees. In: SCN. pp. 257–267. Springer (2002)
16. Beimel, A.: Secure Schemes for Secret Sharing and Key Distribution. Phd thesis, Ben Gurion University (1996)
17. Bethencourt, J., Sahai, A., Waters, B.: Ciphertext-policy attribute-based encryption. In: S&P. pp. 321–334. IEEE (2007)
18. Boneh, D., Boyen, X.: Efficient selective-id secure identity-based encryption without random oracles. In: EUROCRYPT. pp. 223–238. Springer (2004)
19. Boneh, D., Boyen, X., Goh, E.J.: Hierarchical identity based encryption with constant size ciphertext. In: EUROCRYPT. pp. 440–456. Springer (2005)
20. Brickell, E.F., Gordon, D.M., McCurley, K.S., Wilson, D.B.: Fast exponentiation with precomputation. In: EUROCRYPT. pp. 200–207. Springer (1992)
21. Chen, J., Gay, R., Wee, H.: Improved dual system ABE in prime-order groups via predicate encodings. In: EUROCRYPT. pp. 595–624. Springer (2015)
22. Chen, J., Gong, J., Kowalczyk, L., Wee, H.: Unbounded ABE via bilinear entropy expansion, revisited. In: EUROCRYPT. pp. 503–534. Springer (2018)
23. Clarisse, R., Duquesne, S., Sanders, O.: Curves with fast computations in the first pairing group. In: CANS. pp. 280–298. Springer (2020)
24. ETSI: ETSI TS 103 458 (V1.1.1). Technical specification, European Telecommunications Standards Institute (ETSI) (2018)
25. ETSI: ETSI TS 103 532 (V1.1.1). Technical specification, European Telecommunications Standards Institute (ETSI) (2018)
26. Goyal, V., Pandey, O., Sahai, A., Waters, B.: Attribute-based encryption for fine-grained access control of encrypted data. In: CCS. ACM (2006)
27. Goyal, V., Pandey, O., Sahai, A., Waters, B.: Attribute-based encryption for fine-grained access control of encrypted data. Cryptology ePrint Archive, Report 2006/309 (2006)
28. Guillevis, A.: A short-list of pairing-friendly curves resistant to special TNFS at the 128-bit security level. In: PKC. pp. 535–564. Springer (2020)
29. Guillevis, A., Singh, S.: On the alpha value of polynomials in the tower number field sieve algorithm. Cryptology ePrint Archive, Report 2019/885 (2019)
30. Hamburg, M.: Spatial encryption. Cryptology ePrint Archive, Report 2011/389 (2011)
31. Hohenberger, S., Waters, B.: Online/offline attribute-based encryption. In: PKC. pp. 293–310. Springer (2014)
32. Hu, C.T., Ferraiolo, D.F., Kuhn, D.R., Schnitzer, A., Sandlin, K., Miller, R., Scarfone, K.: Guide to attribute based access control (ABAC) definition and considerations (2019), https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=927500
33. Kachisa, E.J., Schaefer, E.F., Scott, M.: Constructing brezing-weng pairing-friendly elliptic curves using elements in the cyclotomic field. In: Galbraith, S.D., Paterson, K.G. (eds.) Pairing. LNCS, vol. 5209, pp. 126–135. Springer (2008)
34. Kamara, S., Lauter, K.E.: Cryptographic cloud storage. In: FC. pp. 136–149. Springer (2010)

35. Kowalczyk, L., Wee, H.: Compact adaptively secure ABE for nc^1 from k -lin. In: EUROCRYPT. pp. 3–33. Springer (2019)
36. Ladd, W., Venema, M., Verma, T.: Portunus: Re-imagining access control in distributed systems. Cryptology ePrint Archive, Paper 2023/094 (2023)
37. Lewko, A., Sahai, A., Waters, B.: Revocation systems with very small private keys. In: IEEE S & P. pp. 273–285 (2010)
38. Lewko, A., Waters, B.: Decentralizing attribute-based encryption. In: EUROCRYPT. pp. 568–588. Springer (2011)
39. Lewko, A.B., Waters, B.: Unbounded HIBE and attribute-based encryption. In: EUROCRYPT. pp. 547–567. Springer (2011)
40. Lin, H., Luo, J.: Compact adaptively secure ABE from k -lin: Beyond nc^1 and towards NL. In: EUROCRYPT. pp. 247–277. Springer (2020)
41. Möller, B.: Algorithms for multi-exponentiation. In: SAC. pp. 165–180. Springer (2001)
42. Okamoto, T., Takashima, K.: Fully secure unbounded inner-product and attribute-based encryption. In: ASIACRYPT. pp. 349–366. Springer (2012)
43. Ostrovsky, R., Sahai, A., Waters, B.: Attribute-based encryption with non-monotonic access structures. In: CCS. pp. 195–203. ACM (2007)
44. de la Piedra, A., Venema, M., Alpár, G.: ABE squared: Accurately benchmarking efficiency of attribute-based encryption. TCHES **2022**(2), 192–239 (2022)
45. Rouselakis, Y., Waters, B.: Practical constructions and new proof methods for large universe attribute-based encryption. In: CCS. pp. 463–474. ACM (2013)
46. Sahai, A., Seyalioglu, H., Waters, B.: Dynamic credentials and ciphertext delegation for attribute-based encryption. In: CRYPTO. pp. 199–217. Springer (2012)
47. Sahai, A., Waters, B.: Fuzzy identity-based encryption. In: EUROCRYPT. pp. 457–473. Springer (2005)
48. Santos, N., Rodrigues, R., Gummadi, K.P., Saroiu, S.: Policy-sealed data: A new abstraction for building trusted cloud services. In: USENIX Security Symposium. pp. 175–188. USENIX Association (2012)
49. Tomida, J., Kawahara, Y., Nishimaki, R.: Fast, compact, and expressive attribute-based encryption. In: PKC. pp. 3–33. Springer (2020)
50. Venema, M.: A practical compiler for attribute-based encryption: New decentralized constructions and more. In: To appear at CT-RSA 2023. Springer (2023)
51. Venema, M., Alpár, G.: Performance estimates for the GLUE paper. <https://github.com/mtcvenema/glue>
52. Venema, M., Alpár, G.: TinyABE: Unrestricted ciphertext-policy attribute-based encryption for embedded devices and low-quality networks. In: Batina, L., Daemen, J. (eds.) AFRICACRYPT. LNCS, vol. 13503, pp. 103–129. Springer (2022)
53. Venema, M., Alpár, G., Hoepman, J.: Systematizing core properties of pairing-based attribute-based encryption to uncover remaining challenges in enforcing access control in practice. Des. Codes Cryptogr. **91**(1), 165–220 (2023)
54. Venema, M., Alpár, G.: Glue: Generalizing unbounded attribute-based encryption for flexible efficiency trade-offs. Cryptology ePrint Archive, Paper 2022/613 (2022)
55. Waters, B.: Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization. Cryptology ePrint Archive, Report 2008/290 (2008)
56. Waters, B.: Ciphertext-policy attribute-based encryption - an expressive, efficient, and provably secure realization. In: PKC. pp. 53–70. Springer (2011)
57. Yamada, S., Attrapadung, N., Hanaoka, G., Kunihiro, N.: A framework and compact constructions for non-monotonic attribute-based encryption. In: PKC. pp. 275–292. Springer (2014)