Designs, Codes and Cryptography (2023) 91:165–220
https://doi.org/10.1007/s10623-022-01093-5

Systematizing core properties of pairing-based attribute-based encryption to uncover remaining challenges in enforcing access control in practice

Marloes Venema[1] · Greg Alpár[1,2] · Jaap-Henk Hoepman[1,3,4]

Received: 13 September 2021 / Revised: 21 March 2022 / Accepted: 16 July 2022 /
Published online: 6 September 2022

Abstract
Attribute-based encryption (ABE) cryptographically implements fine-grained access control on data. As such, data can be stored by an entity that is not necessarily trusted to enforce access control, or an entity that is not even trusted to have access to the plaintext data at all. Instead, access control can be externally enforced by a trusted entity. Additionally, some multi-authority variants of ABE—which do not have a central authority—can effectively and securely implement access control in multiple-domain settings. Furthermore, ABE is the only cryptographic approach to fine-grained access control that does not require an online trusted third party during access requests, and thus provides better availability properties. The actual realization of these theoretical advantages in practice depends on whether current state-of-the-art ABE schemes support the necessary core properties. Much progress has been made in the last two decades in pairing-based ABE schemes, owing to their versatility and efficiency. In fact, it is possible to support most core properties under strong security guarantees, while incurring acceptable storage and computational costs. It is therefore a good time to ask ourselves whether pairing-based ABE has reached its full practical potential. To answer this question, we provide a comprehensive systematized overview of various existing pairing-based ABE schemes and their core properties. We also investigate the relationship between these core properties and real-world access control requirements. We show that a few

Communicated by K. Matsuura.

✉ Marloes Venema
m.venema@cs.ru.nl

Greg Alpár
g.alpar@cs.ru.nl

Jaap-Henk Hoepman
jhh@cs.ru.nl

1 Radboud University, Nijmegen, The Netherlands

2 Open Universiteit of the Netherlands, Heerlen, The Netherlands

3 University of Groningen, Groningen, The Netherlands

4 Karlstad University, Karlstad, Sweden

challenges remain, that must be overcome for ABE to reach its full potential as a mechanism to implement efficient and secure access control in practice.

Keywords Attribute-based encryption · Access control · Systematization of knowledge

Mathematics Subject Classification 94A60 · 68P27 · 68M25

1 Introduction

Would it not be great to be able to encrypt a document so that it can only be decrypted and accessed by e.g., all epidemiologists of the Johns Hopkins Hospital? An encryption scheme that provides this functionality would enable individuals to share data in a secure yet flexible way.

Traditional public key encryption does not effectively provide this functionality. Typically, it is used to allow access to confidential data to one particular entity, which must be known at the time the data are encrypted (by using its public key). Only this entity can later access the data by decrypting the associated ciphertext (by using its secret key). Attribute-based encryption (ABE) [133] is a form of public-key encryption in which the key pairs are associated with *attributes* rather than individual users or entities. For instance, in ciphertext-policy ABE [29], the encrypting user can decide who gets access to the data by specifying a policy during encryption, e.g., any "epidemiologist" at the "Johns Hopkins Hospital". The ability to decrypt the resulting ciphertext is then determined by the attributes owned by the decrypting user, who must be an "epidemiologist" at the "Johns Hopkins Hospital". Thus, data that are encrypted with ABE can be accessed by multiple authorized users, making ABE inherently more flexible than other cryptographic primitives.

Based on its functionality, attribute-based encryption can cryptographically implement fine-grained access control on data [29, 72, 126]. Like most access control mechanisms [85, 134], it relies on a trusted third party (TTP) to help grant or deny users who request access to data. Nevertheless, compared to traditional access control mechanisms, ABE requires less trust in and less reliance on this TTP. First, because ABE is a cryptographic primitive, the data are encrypted and can thus be stored by an entity that is not necessarily trusted to securely enforce access control or to access the data at all. Second, as we will show later, ABE does not require an online TTP during each access request, allowing users to act more autonomously. Importantly, minimizing the role of the TTP in the enforcement of access control in this way fosters the availability of the system.

In ABE, the key generation authority (KGA) constitutes such a trusted third party. The KGA generates the master public keys and the master secret keys, from which it derives secret keys and issues these to eligible users. Once the users have received secret keys, they can decrypt any ciphertexts for which they have a suitable key. In turn, access to data can be managed by the data owner using encryption. Then, access to these data is indirectly enforced by the KGA, which provides only eligible users with keys that can decrypt the resulting ciphertext. In addition, some variants of ABE, called multi-authority ABE (MA-ABE) [40], support the employment of multiple (possibly mutually distrusting) authorities. This allows for the secure enforcement of access control in multiple-domain or cross-organizational settings, e.g., electronic health record (EHR) systems involving hospitals and insurance companies. Owing to all these advantages, ABE has attracted much interest from the practical community [61, 62, 87, 136].

For the past two decades, the theoretical community has made much progress in pairing-based ABE [133], leading to many publications at prominent conferences, and thus establishing itself as an important and popular research topic. Many schemes have been proposed that vary significantly in the core properties, which determine the basic functionality, efficiency, and security. Some examples include the level of fine-grainedness of the access policies, the performance of the scheme, and the underlying cryptographic assumptions. Some core properties may be more desirable for practical applications than others. Ideally, a scheme that supports most or all of these properties is used for these applications. Nowadays, pairing-based ABE has reached a level of maturity such that most of the desirable properties can be achieved simultaneously, whilst attaining both strong security guarantees as well as acceptable storage and computational performance [15, 22, 92, 107]. Given these developments, a natural question that arises is:

To what extent has pairing-based ABE reached its full practical potential?

In this paper, we work towards answering this question. To this end, we focus on the core properties of ABE, as they strongly influence the basic functionality and efficiency of any pairing-based ABE scheme in practice. Concretely, we identify the main core properties as defined through the years. For these properties, we provide unified definitions, and give an overview of prominent schemes and the properties they satisfy. By considering an example of a practical setting, we argue which properties are desirable. Then, we consider if and how these desirable properties can be achieved simultaneously. To obtain a better understanding of the interplay between these various properties, we analyze how these are realized in the existing pairing-based schemes and whether they are compatible with one another. Furthermore, provided that they are compatible, we consider the effect of satisfying all these properties simultaneously on other practical aspects, such as efficiency and availability. Along the way, we uncover a number of remaining challenges, which we pose as directions for future research. We encourage the (theoretical) community to explore these, as it could make ABE even more practical, mitigating the disadvantages of ABE compared to other primitives for implementing access control, whilst amplifying its advantages.

To place properties specific to ABE in a practical context, we will first describe a large-scale medical scenario in which access control to data is enforced through cryptography. In general, the implemented access control mechanism should support properties such as confidentiality, integrity, and availability [135]. In particular, for such large-scale real-world settings, it is important that these properties can be simultaneously guaranteed in the best way possible. On the one hand, properties such as confidentiality and even integrity have been considered at length in the context of ABE. On the other hand, properties such as availability have been treated in much less detail. In this work, we also investigate the relationship between the ABE properties and these three real-world properties. Specifically, we will introduce the new notion of *resilience* in the context of ABE to foster a deeper understanding of availability in real-world settings using ABE. Roughly speaking, resilient ABE minimizes the required interaction between the user and the key generation authority. In particular, we define ABE to be resilient if e.g., the addition or removal of attributes in the system does not cause issues with respect to the correctness or security of the scheme. Such issues would require that e.g., new keys are issued to make the system functional again, subsequently requiring interaction between the user and the KGA. By eliminating such issues and therefore minimizing this required interaction, it is less stringent that the KGA is always available, while access control can still be securely managed. Furthermore, we consider how better availability properties can be achieved in multi-authority ABE (MA-ABE) by analyzing existing work. MA-ABE with such availability properties can provide advantages in implementing access control in the multiple-domain setting compared to other solutions.

Ultimately, the goal of our analyses is to help pairing-based ABE reach its full potential. By considering the interplay between the core properties as well as their relationship with real-world (security) properties, we strive to obtain as much functionality, security and efficiency as possible. At the same time, we want to highlight the advantages of ABE in the implementation of access control compared to other solutions. Therefore, we pose several directions for future research that help ABE become even more practical.

1.1 Our contribution

Our contribution is fourfold.

- **Systematization of knowledge:** We provide an extensive overview and systematization of significant core properties of ABE. To this end, we analyze over fifty important pairing-based ABE schemes—each published at a prominent conference—and their properties.
- **Interplay of properties:** We analyze how the core properties are realized to understand whether and how they can be achieved simultaneously. Furthermore, we analyze the influence of these properties on real-world properties such as availability.
- **New insights:** Sometimes, this analysis leads to deeper, novel insights, explicitly conveyed as *observations* throughout the paper.
- **New research directions:** Based on the analysis, the systematization and observations, we identify several *directions* for future research that are relevant to improve the practical advantages of ABE. We encourage the cryptographic community to explore these directions.

To the best of our knowledge, this is the first in-depth overview of pairing-based ABE that discusses the interplay between different core properties as well as their practical impact.

1.2 Scope and approach

In this work, we focus primarily on pairing-based attribute-based encryption for (non-)monotone span programs (which include Boolean formulas), although the first sections of this paper are general in the sense that they describe a broader class of ABE. The main reason for our focus on pairing-based schemes is that they are more established, efficient and practical than works based on e.g., multilinear maps [68] or post-quantum assumptions [36]. While these are interesting in their own right, we believe that it would be more suitable to address concerns specific to these subfields once they have reached a similar maturity as pairing-based ABE. Furthermore, we consider ABE for (non-)monotone span programs, because these provide sufficient expressivity that is expected in access control mechanisms. ABE for more fine-grained classes of expressivity such as circuits [71] are typically also less efficient or are secure in weaker models. Finally, we focus primarily on the core properties of ABE, and therefore do not extensively discuss other practical extensions, such as those mentioned in Sect. 10. While these extensions are important and may provide some interesting benefits in practice, we leave any systematized analysis of these for future work. We have also excluded any broken schemes (e.g., [142]) or any schemes that lack a proper security analysis.

In this paper, we do not necessarily strive for formality. On the contrary, one of our aims is to make the field of ABE more accessible to the practical community. In particular, we have moved away from the heavy notation and complicated formal concepts in the newer works.

Nevertheless, we have included their accomplishments. This work does contain some formal definitions and models, but only of the most common and established concepts.

For convenience, we refer to the schemes by concatenating the first letter of each author's surname with the last two digits of the year of publication, e.g., the scheme published by Rouselakis and Waters in 2013 [130] is referred to as the RW13 scheme.

1.3 The core properties of ABE

One of the main components of this paper is that we explore and analyze the core properties of ABE, which are

- the level of expressivity the scheme supports in the the access policies;
- whether the scheme is key-policy or ciphertext-policy based;
- whether the scheme supports small or large universes, i.e., which distinguishes between whether it can support any string as attribute or not;
- whether the scheme is bounded in any of the parameters or not.

In addition, we analyze multi-authority ABE, which employs multiple key generation authorities. We highlight this particular extension of ABE because of its unique advantages in enforcing access control in practice.

1.4 Target audience and goal

Our target audience is wide, and includes practitioners, cryptographic engineers and cryptographers (both newcomers to the field of ABE and experts). In part, the reason for this is that our ultimate goal is to investigate to what extent pairing-based ABE has reached its full practical potential with respect to the core properties. To this end, we also explore the basics of ABE and the requirements for ABE when it is employed in practice. Our goal is therefore threefold:

- introducing new cryptographers to the area of ABE;
- informing practitioners about the potential of ABE;
- encouraging experts in ABE (and related areas) to address the future directions.

Note that some of those future directions may be more general in the sense that they extend to other fields of cryptography as well. For instance, they could also pertain to related areas in pairing-based cryptography, or exploring these directions could require expertise in related fields such as cryptographic engineering.

1.5 Organization

Because of the wide range of aspects that this paper considers, we make the dependencies among the sections explicit. In Fig. 1, we illustrate several possible ways to read this paper. In particular,

- **Sections** 2, 3 and 4 describe the general concepts of ABE in a practical context, which might be of interest to any readers who want to know more about the practical advantages of ABE in general, including novel (ABE) cryptographers and practitioners. These concepts are general in the sense that they are applicable not only to pairing-based but to every existing ABE scheme;

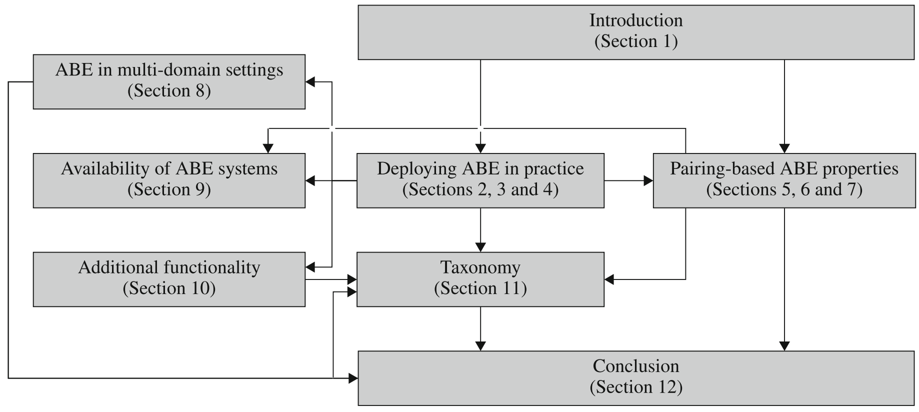
Introduction (Section 1)
ABE in multi-domain settings (Section 8)
Availability of ABE systems (Section 9)
Deploying ABE in practice (Sections 2, 3 and 4)
Pairing-based ABE properties (Sections 5, 6 and 7)
Additional functionality (Section 10)
Taxonomy (Section 11)
Conclusion (Section 12)

Fig. 1 The general structure of this paper

– **Sections** 5, 6 and 7 discuss pairing-based ABE with respect to the general concepts, as well as the efficiency of pairing-based ABE, which may be of interest to cryptographers and cryptographic engineers;
– **Section** 8 focuses on multi-authority ABE, which may be of interest to practitioners who wish to deploy ABE in the multiple-domain setting as well as cryptographers. In this section, we re-contextualize the notions of distributed and decentralized, and systematically classify existing schemes;
– **Section** 9 considers the availability properties of ABE by putting forth the new notion of resilience. This section may be of interest to practitioners and cryptographers;
– **Section** 10 discusses some additional functionality, which may be of interest to practitioners;
– **Section** 11 covers our taxonomy, which may be of interest to practitioners and novel cryptographers;
– **Section** 12 concludes our paper.

2 Practical motivation: access control

ABE allows for the secure and practical enforcement of fine-grained access control on data. On the one hand, ABE ensures that the data are encrypted, such that the storing entity cannot read the plaintext data. On the other hand, by its functionality, it ensures that access control can be enforced externally by a trusted entity. As an example, consider electronic health record (EHR) systems, planned to be used on a large scale [80]. While the use of EHR systems simplifies the sharing of health records across organizations, jurisdictions or countries, it also increases the risk of infringing upon the privacy rights of individuals [82]. To address these privacy concerns, existing solutions often use access control to manage access to the data. It varies, though, which access control model is used, who defines and assigns user roles and access policies, and who grants access to the data [8]. Generally, the most common access control models in such health settings are role-based access control (RBAC) [134] and attribute-based access control (ABAC) [85]. In addition, to ensure confidentiality, these solutions require the data to be encrypted, though security problems may still arise. In practice, the keys are frequently stored by the same entity that stores the data. Effectively, the entity storing the keys enforces access control on the data, and must therefore be highly trusted. Also, this entity needs to be available when access is requested. In multiple-domain settings—in which data pertaining to one individual may be stored or produced by different entities—such a degree of trust may be problematic; especially, if each entity wants to enforce access control regardless of where the data are stored. We show that ABE can provide a secure solution, even in the multiple-domain setting.

We briefly illustrate what such an EHR system may look like in the multiple-domain setting, and what availability and scalability issues may occur. Consider a medical scenario with a hospital and an insurance company that both want to use an EHR system using a traditional form of access control. A patient at the hospital may want to share some of her private data, stored at the hospital, with both her doctor and an employee at her insurance company. Some employee at the insurance company can then request access to the data by contacting the server at the hospital. To grant access, the server needs to know the access policy and contact the insurance company to verify whether the requesting user is an employee. Hence, during an access request, all relevant entities (e.g., the hospital and the insurance company) need to be available. In more complex scenarios, with access policies involving many entities, the required interaction among these entities scales up, amplifying any availability issues.

ABE provides a practical and secure solution, and mitigates the potential availability and scalability issues. Specifically, the KGA indirectly enforces access control by generating the public and secret keys. Furthermore, the ciphertext-policy variant of ABE [29] allows the encrypting users to decide who is allowed access by specifying the access policy. Other users can only successfully decrypt the ciphertext, if they have a set of attributes that satisfies the access policy. In contrast to non-cryptographic access control mechanisms, ABE only requires the enforcing entities, in this case the KGAs, to be available when users request secret keys. A user typically requests a secret key only once: when the user enters the system (and potentially, when the user obtains new attributes). From that point forward, the user can access any data for which she is authorized while requiring no interaction with and between any policy-enforcing entities (only the entities who store the data), effectively mitigating any availability issues. In addition, the use of cryptography ensures that the data can be stored anywhere, and thus can be shared across various domains.

In the multiple-domain setting, multi-authority (MA) ABE [40] can be used. In this variant of ABE, the role of the KGA is shared by multiple entities. Each KGA securely manages a unique set of attributes. Unfortunately, not all MA-ABE schemes are "decentralized" enough. In particular, users need to interact (at some point) with all KGAs associated with a policy. We illustrate the issue by considering an example in more traditional access control mechanisms. For instance, consider an access policy defined over attributes managed by several authorities. Ideally, the decision to grant access is made by verifying with the relevant entities whether the user satisfies the access policy. For example, during the access request of the user in our previous example, the insurance company confirms the employee status to the server that stores the data. It is then not needed for the server to also check with the hospital whether she is a doctor, as the access policy is already satisfied. In contrast, most instantiations of MA-ABE require the decrypting user to request keys from all authorities associated with the access policy, including those authorities for which the user may not have any attributes. Hence, the decision to grant access, however indirectly, needs to be verified with both the insurance company and the hospital. Subsequently, the decrypting user may need to interact with possibly many authorities, which need to be online if the user does not have any keys yet. This negatively impacts the scalability and availability of the system.

Concretely, we distinguish between a decentralized and a distributed access control decision. If the access control mechanism allows that the decision to grant access is made by verifying with only the relevant entities whether the requesting user satisfies the policy, then we call it decentralized. If all entities need to be contacted—effectively distributing the decision—we call it distributed. Essentially, this distinction between decentralized and distributed is determined by the level of autonomy or independence of the entities. This distinction is roughly in line with the terminology in (algorithmic) decision making in systems [57, 114]. In those works, decentralized systems do no require that the nodes have system wide information or need to communicate with all the nodes in the system to correctly make decisions. Decentralized systems subsequently enjoy a level of autonomy and independence in this process, which is in line with our definition of decentralized access control decisions. In contrast, distributed access control decisions can be seen as the opposite of centralized access control decisions, and thus, as the overarching term of any system that allows multiple authorities to make a decision [57]. However, we shall use the term distributed to clearly distinguish distributed but non-decentralized access control decisions from decentralized access control decisions. In addition, for an MA-ABE scheme to be distributed or decentralized, we require that the authorities do not need to trust or rely on one another for confidentiality either. This is especially useful in settings in which some entities have conflicting interests. For instance, consider adding another insurance company in our medical scenario.

To express some of the properties that we informally discussed, we formulate the following two implicit properties that are generally important for practical access control mechanisms.

– **User independence:** authorized users can obtain access even if not all of the authorities are available at the time of an access request.
– **Authority-dependence minimization:** authorized users only need to rely on the authorities associated with their set of attributes. If it satisfies the access policy, they can gain access without needing to interact with other authorities. In addition, the authorities do not have to trust one another to correctly and securely enforce access control.

In contrast to traditional RBAC and ABAC mechanisms, ABE provides a high level of user independence by its functionality, because users can independently obtain access by decrypting ciphertexts for which they are authorized once they have a secret key. Therefore, they do not need to interact with any policy-enforcing entities, which may be unavailable at the time of an access request. To maximize the level of user independence, we assume that the user only needs to request a secret key once. In Sect. 9, we will put forth the new notion of resilience in the context of ABE to foster such user independence. We also show that some MA-ABE schemes satisfy our strong notion of authority-dependence minimization, which makes these schemes especially attractive for implementing access control in the multiple-domain setting. We call these schemes decentralized (MA-)ABE schemes.

3 Attribute-based encryption—core properties

3.1 Attributes and access structures

First, we explain what attributes are, how they are represented, and how they are used as a building block of access structures. An *attribute* is defined as a characteristic of the user, expressed as a type-value pair, e.g., the type of the attribute could be "profession" and its value could be "doctor" [85]. In the examples in this work, we often assume that attributes are represented as strings consisting only of its value (if the type is clear). However, in practice, one may want to use the type as well to avoid confusion. For instance, "doctor" may also refer to a person who holds a PhD in computer science, though, in the context of a medical setting, it is unlikely that this meaning is used.

Attributes are an important building block of *access structures*, or *policies*, which specify which attributes need to be possessed by a user in order to be granted access to a certain resource (in our case: data). Typically, access structures are expressed as Boolean formulas (including thresholds functions). For instance, the policy “doctor $\wedge$ Johns Hopkins Hospital” specifies that access is granted to all doctors that work at the Johns Hopkins Hospital. In the formal sense, access structures can be defined in terms of authorized sets:

Definition 1 (*Access structures* [27]) Let $\{\mathrm{att}_1, ..., \mathrm{att}_n\}$ be a set of attributes. An access structure is a collection $\mathbb{A}$ of non-empty subsets of $\{\mathrm{att}_1, ..., \mathrm{att}_n\}$. The sets in $\mathbb{A}$ are called the authorized sets, and the sets that are not in $\mathbb{A}$ are called the unauthorized sets.

Note that any access policy expressed as a Boolean formula can also be expressed as a set (like in the definition). For instance, consider the formula “(doctor $\vee$ nurse) $\wedge$ Johns Hopkins Hospital”. The associated access structure $\mathbb{A}$ consists of all subsets of attributes in the system that contain the sets {doctor, Johns Hopkins Hospital} or {nurse, Johns Hopkins Hospital}.

Two important aspects in access structures are the *expressivity* and the *monotonicity*, which collectively determine the level of fine-grainedness of a scheme. Informally speaking, expressivity concerns whether the use of all Boolean formulas consisting of conjunctions, disjunctions and threshold functions [29, 72] is allowed. Furthermore, monotonicity concerns whether the use of negations, e.g., “NOT doctor”, is allowed. Monotone access structures do not allow the use of negations in the formula, while non-monotone access structures do allow it. In itself, monotonicity can technically be characterized as an expressivity aspect. However, we consider it as a separate feature, because negations are typically supported using different techniques than the other expressivity features. In addition, depending on the practical setting and the attribute type, non-monotonicity may not be achievable. In particular, for some attribute types, it may be difficult to ascertain whether a user does not possess certain values, especially when a user can possess multiple values. For instance, doctors may work at multiple departments, or they may also be patients at the hospital where they work.

Definition 2 (*Monotone access structures* [27]) An access structure $\mathbb{A} \subseteq 2^{\{\mathrm{att}_1,...,\mathrm{att}_n\}}$ is monotone, if for all B, C holds that if $B \in \mathbb{A}$ and $B \subseteq C$, then also $C \in \mathbb{A}$.

Existing schemes have varying levels of expressivity and monotonicity. The least expressive policies are those that only support *AND-gates* (or: conjunctions) [52]. More expressive structures support a single *threshold function*, which consists of a set of attributes and a threshold (smaller than the size of this set) to indicate the minimal number of these attributes that needs to be in the user’s possession [133]. The most expressive access structures are *(non-)monotone span programs* ((N)MSP), which support any formulas using conjunctions, disjunctions and threshold functions [29, 72]. Here, the distinction between monotone and non-monotone span programs depends on the monotonicity of the access structures. To implement access control on data in line with RBAC [134] or ABAC [85], it is paramount that a scheme supports all Boolean formulas, including conjunctions, disjunctions and negations.

3.2 Key-policy and ciphertext-policy ABE

In ABE, access structures—also known as policies—can be embedded either in the secret keys or the ciphertexts. In *key-policy attribute-based encryption* (KP-ABE) [72], access structures are embedded in the keys. These structures are defined by the key generation authority (KGA),

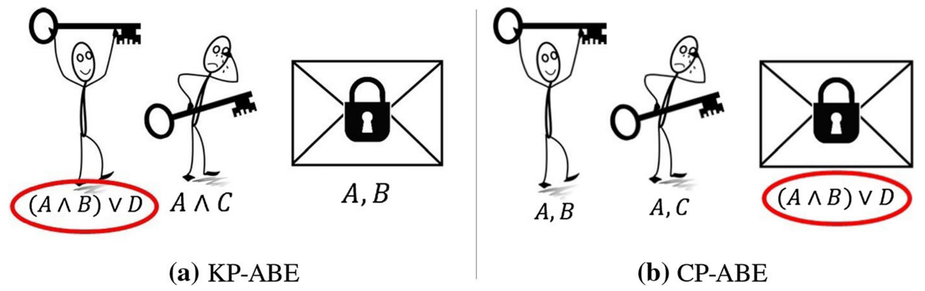

Fig. 2 Key-policy versus ciphertext-policy ABE. In the examples, the access structure is either associated with the keys (i.e., the persons holding a key) or the ciphertext (i.e., the locked envelope). In each example, the person on the left is happy, because he can decrypt the ciphertext, while the person on the right is sad, because he cannot

and subsequently embedded in the keys that are distributed to eligible users. The encrypting user can in turn associate a set of attributes with the ciphertext. This ciphertext can only be decrypted by another user who has a key associated with an access policy satisfied by the set. Conversely, in *ciphertext-policy attribute-based encryption* (CP-ABE) [29], access structures are embedded in the ciphertexts. In particular, the access structures are defined by the encrypting user. The KGA generates secret keys associated with some attributes that the user possesses. Subsequently, decrypting users can decrypt a ciphertext if they possess an *authorized* secret key. A secret key associated with a set of attributes is authorized if the set satisfies the access policy associated with the ciphertext. Figure 2 illustrates the distinction between KP-ABE and CP-ABE with an example.

KP-ABE and CP-ABE allow for the implementation of different types of access control. Specifically, KP-ABE can implement content-based access control using e.g., tags [7, 126]. When the data are created and subsequently encrypted, it may not be clear what the policies are going to be. However, it may be clear what any tags may constitute, e.g., because they relate to the encrypted data rather than who is authorized to access the data. For instance, in the medical setting, suitable tag types may be the patient's name, date of birth or social security number, and the data type (e.g., results of blood tests or scans). When a doctor wants to access some data, she can contact the hospital's KGA. The KGA can then first determine whether the doctor is authorized to access these data. It can then generate a secret key for e.g., the policy "name: Alice $\wedge$ (data type: scans $\vee$ data type: blood test)", so that the doctor can access all test results and scans related to Alice. However, note that this type of access control lets the KGA manage access to the data rather than the data owner. It thus does not allow for the implementation of RBAC and ABAC, which allows data owners to specify who gets access to the data, as required in settings as described in Sect. 2. In contrast, CP-ABE allows for the implementation of more fine-grained access control models such as ABAC. In the CP-ABE setting, the KGA (which may be assigned by some health authorities) distributes the keys associated with the attributes of the user. The encrypting user—which may be the data owner—gets to specify the access policy, and is therefore in control of managing access.

Observation 1 *Some KP-ABE schemes [44, 52, 119, 149]—which are explicitly presented as CP-ABE schemes—"implement" the same functionality as CP-ABE at the cost of restricting the access policies. In particular, these policies are restricted to AND-gates over positive or negative and dummy values for each attribute in the system, where the positive and negative values are used to indicate whether the decrypting user should have it or not, respectively. The dummy value is used to indicate that it does not matter whether the decrypting user has* *it or not. For instance, suppose that $\mathcal{U}$ denotes the set of all attributes, and $\mathcal{S}$ denotes the set of attributes that the user possesses. Then, the user receives a key for all positive values of the attributes in $\mathcal{S}$, all negative values of the attributes in $\mathcal{U} \setminus \mathcal{S}$, and all dummy values of the attributes in $\mathcal{U}$. On a technical level, it is required to generate a key for each attribute in $\mathcal{U}$ in this way, because the scheme implicitly defines a policy over the key: an AND-gate over all attributes in the system. Each clause of the AND-gate is either an OR-gate of the positive and dummy values of the attribute, or an OR-gate of the negative value and dummy value of the attribute, depending on whether the attribute is in $\mathcal{S}$ or not.*

In turn, a ciphertext is associated with a "policy", which is an AND-gate over all attributes in $\mathcal{U}$, in which each clause consists of the positive, negative or dummy value of the attribute. Here, the dummy value indicates that the encrypting user does not care whether the decrypting user has the attribute or not. On a technical level, the ciphertext is associated with a set of attributes of a specific form, i.e., a set which contains for each attribute the positive, negative or dummy value. Indeed, decryption is possible as expected: when the "policy" associated with the ciphertext is satisfied by the "set" associated with the key. This happens exactly when the set associated with the ciphertext satisfies the policy associated with the key. That is, for each attribute in the AND-gate, the key policy specifies an OR-gate over the dummy value and either the positive or negative value. Hence, the ciphertext set satisfies it when it specifies the dummy or the positive/negative value (depending on what is specified in the policy) of this attribute. Conversely, the key "set" always satisfies dummy values, and only satisfies the "policy" if the values for which either the positive or negative value is used all match. As a consequence of this way of implementing CP-ABE, both the keys and ciphertexts scale linearly in the total number of attributes. Not only are these schemes less expressive, they are also less efficient.

We suspect that the reason for implementing CP-ABE via KP-ABE in this way is the security proof. For a long period of time, the BSW07 [29] scheme had been the only expressive CP-ABE scheme (i.e., supporting MSPs), though it only had a proof in the generic group model (Sect. 4.5). The first provably secure CP-ABE schemes that support MSPs were published in 2010 [99] and 2011 [145], which required arguably more complicated proof techniques than previous KP-ABE constructions.

We formally define KP-ABE and CP-ABE by formally defining the notion of *predicate encryption* [88], which is a more general cryptographic primitive that includes KP-ABE and CP-ABE as special cases. Specifically, it is defined for any predicate P, where $P : \mathcal{X} \times \mathcal{Y} \to \{0, 1\}$ is a function that takes as input any pair $(x, y) \in \mathcal{X} \times \mathcal{Y}$, and outputs 1 exactly when the predicate is satisfied, i.e., $P(x, y) = 1$. For KP-ABE, $\mathcal{X}$ denotes the collection of all sets of attributes, and $\mathcal{Y}$ the collection of all policies, and $P(x, y) = 1$ if and only if the set x satisfies the policy y. Conversely, for CP-ABE, $\mathcal{X}$ denotes the collection of policies and $\mathcal{Y}$ the collection of attribute sets.

Definition 3 (*Predicate encryption* (*PE*) [5]) A predicate encryption scheme for a predicate $P : \mathcal{X} \times \mathcal{Y} \to \{0, 1\}$, with some key generation authority (KGA), users and a universe[1] of attributes $\mathcal{U}$ consists of four algorithms:

- Setup(λ): On input the security parameter λ, this randomized algorithm, executed by the KGA, generates the domain parameters, the master public key MPK and the master secret key MSK.
- KeyGen(MSK, y): On input the master secret key MSK and some $y \in \mathcal{Y}$, this randomized algorithm, executed by the KGA, generates a secret key SK_y.

[1] A universe of attributes is the overarching set of all attributes that can be used in the system.

- Encrypt(MPK, x, M): On input the master public key MPK, some $x \in \mathcal{X}$ and message M, this randomized algorithm, executed by the encrypting user, generates a ciphertext CT_x.
- Decrypt(MPK, SK_y, CT_x): On input the master public key MPK, the secret key SK_y, and the ciphertext CT_x, if $P(x, y) = 1$, then it returns M. Otherwise, it returns an error message.

A scheme is called *correct* if decryption of a ciphertext with an authorized key succeeds with overwhelmingly high probability. The scheme is called *secure* if decryption of a ciphertext with any number of unauthorized keys fails with overwhelmingly high probability. We discuss the notion of security in more detail in Sect. 4.

Because of its functionality, we consider CP-ABE as the more favorable of the two, so we focus almost solely on CP-ABE in the remainder of this work.

3.3 The universe of attributes

The set of attributes used in an ABE scheme is called the *universe of attributes*, which can be *small* or *large* [133]. In the formal sense, these distinguish between whether the universe is polynomially bounded in the security parameter or not. In small-universe constructions, the master public key—generated in the setup—depends directly on the universe of attributes. Because the KGA needs to explicitly publish a public key for each attribute, the master public key is consequently polynomially bounded. In large-universe constructions, the master public key is independent of the universe. Any user can uniquely generate the public key associated with some attribute from the master public key and the attribute. Because the number of unique public keys is exponential in the security parameter, the number of attributes in the universe is essentially unbounded.

We consider large-universe constructions to be more practical than small-universe constructions for several reasons. In contrast to small-universe constructions, large-universe allow for the generation of public keys from any input strings. As such, the authority does not need to keep a record of all attributes and their public keys (which may be large!), which makes the system more scalable. In turn, users do not have to locate these public keys before encryption. A secondary advantage of this is that this may also be more privacy friendly. For instance, publishing identifiable information such as names or social security numbers reveals that a person is part of a system. Another secondary advantage is that encrypting users can use attributes for which no keys exist yet without first asking the KGA to generate these [125], which gives them more autonomy and therefore fosters availability. Finally, attributes can also be added to the universe without any consequences with respect to the public keys and previously generated keys and ciphertexts. We show in Sect. 9 that adding attributes may potentially lead to incorrectness or insecurity e.g., in small-universe schemes that associate the entire universe with keys and ciphertexts, such as the schemes that we considered in Observation 1.

3.4 (Completely) unbounded ABE

Sometimes, schemes are bounded in one or more parameters. Indeed, we had already considered the size of the universe, which can be small or large. In addition, schemes can be bounded in the sizes of the sets of attributes or the access policies, and by extension the sizes of the keys or ciphertexts. Furthermore, bounds can be placed on the number of times that an attribute occurs in the policy, which we call *bounded re-use* [99]. If an attribute may be used only once, we say that the scheme suffers from a *one-use restriction*. Conversely, a scheme is multi-use if attributes may appear any number of times in the policy.

If schemes are not bounded in any of these parameters, one might argue that they are called *unbounded*. Remarkably, various works describe different definitions of the term "unbounded". Notable examples include:

– LW11b [101]: requires the scheme to support large universes, and to impose no bounds on the attribute sets;
– AY15 [23], Att19 [15]: require the scheme to impose no bounds on the attribute sets or access policies, including the number of uses of an attribute in the policy. They call a scheme *completely* unbounded, if it also supports large universes;
– CGKW18 [47]: requires the scheme to impose no bounds on the sets or policies.

We also observe that the term "unbounded" is usually only reserved for schemes that avoid the random oracle model (Sect. 4.4). For instance, large-universe schemes that use a full-domain hash (Sect. 5.5)—which additionally pose no restrictions on any of the discussed parameters—are not typically referred to as unbounded. Presumably, this is because the hash is modeled as a random oracle, which can be regarded as a restriction as well. Therefore, rather than classifying a scheme as unbounded or not, we will consider for each of the aforementioned parameters whether they are unbounded.

Observation 2 *In general, an obvious disadvantage of requiring bounds on any of these parameters is that it limits some or all parties in the system, for instance, because the policy that they want to use for encryption is larger than the scheme allows. However, there seems to be an additional, more subtle disadvantage in some cases, which is not necessarily caused directly by imposing these bounds.*

For instance, as we will show in Sects. 5.5, 5.6 and 7.3, some methods used to achieve the large-universe property subsequently result in requiring bounds on the policy (or set) associated with the ciphertext. In addition, these methods also affect the efficiency of the scheme. Typically, the public keys and encryption costs grow by a factor that is linear [143] or even quadratic [21] or cubic [3] in this bound. As such, the efficiency of the scheme is directly dependent on the bound. Increasing the bound makes the scheme more flexible, but less efficient, meaning that this bound cannot simply be chosen as a sufficiently high number.

As another example, we consider schemes with a bounded re-use of an attribute in a policy. To mitigate the one-use restriction, some works [97, 99] make multiple copies of each attribute. The idea is that, for each use of the same attribute in the policy, another copy of the attribute is used. However, the number of copies is fixed after the setup is run, meaning that it is bounded. Furthermore, the efficiency of the scheme depends directly on this bound [4]. In this case, the public keys and key generation costs grow by a factor that is linear in the bound, and thus yield similar flexibility-efficiency trade-offs as the previous example.

4 Security of ABE

4.1 Collusion resistance

An important property of ABE is that it is required to be *collusion resistant*. If any number of users are not individually able to decrypt a ciphertext, they should not be able to do this collectively either. For example, a doctor who works at the Mayo clinic and a nurse who works at Johns Hopkins Hospital should not be able to individually decrypt a ciphertext with policy "doctor $\wedge$ Johns Hopkins Hospital". In addition, they should not be able to collude and decrypt the ciphertext together either.

When access control is enforced in practice, an important aspect is that access is only granted to authorized users. In traditional systems, the authority ensures this by verifying whether a requesting user possesses a set of attributes that satisfies the policy [85]. To do this properly, the attributes need to be authenticated. This means that the authority needs to be certain that the attributes are actually in the possession of a single user (and not, say, in the possession of multiple colluding users). To enforce access control with ABE, the employed scheme should ensure this as well, which is the case when it is collusion resistant.

4.2 Security models

In the context of ABE, the security models capture security against chosen-plaintext attacks (CPA) and collusion resistance (Sect. 4.1). The strongest notion of security is provided by the *full security* [99] model, then the *semi-adaptive security* [50] model and then the *selective security* [133] model. Other models include co-selective security [20] and static security [131], but these are used much less often. The basic models consider security against chosen-plaintext attacks but can easily be extended to model chosen-ciphertext attacks (Sect. 4.3). The full security model is formally defined as follows.

Definition 4 (*Full security against chosen-plaintext attacks* (*CPA*) [5]) We define the security game between challenger and attacker as follows:

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

The advantage of the attacker is defined as $|\Pr[\beta' = \beta] - \frac{1}{2}|$. A scheme is fully secure if all polynomial-time attackers have at most a negligible advantage in this security game.

In the selective security model, the challenge x^* (in CP-ABE: access structure) is announced before the challenger runs the setup. In the semi-adaptive security model, this happens afterwards, but before the attacker is allowed to query secret keys. While selective and semi-adaptive security certainly prove a level of security—or at least, they inspire some confidence in that the scheme is secure—neither does accurately model real-world dangers to security breaches [42]. It is unreasonable to assume that an attacker is going to announce which e.g., access policies it is going to attack before a system setup.

A natural question would however be: are fully secure schemes truly more secure than selectively secure schemes *in practice*? From a theoretical standpoint, this seems to be true. Intuitively, one could provide a security reduction of a selectively secure scheme in the full security model by protecting against every conceivable access policy, resulting in an exponential security loss [133]. In fact, Lewko and Waters [103] formally prove this by showing that any such black-box reduction leads to an exponential security loss. Nevertheless, so far, no practical attacks exist that can break any specific selectively secure scheme with a significant advantage over its fully secure variant. As such, it is unclear whether selective security is simply an artifact of the used proof technique, or whether these schemes are truly less secure in practice; and if so, how much less. This gives rise to the following direction.

Direction 1 (Selective versus full security in practice) *To investigate the relationship between specific instantiations of selectively secure ABE schemes and their fully secure counterparts with respect to their security in practice.*

4.3 Security against chosen-ciphertext attacks

As noted, the basic security models only provide security against chosen-plaintext attacks (CPA). However, in practice, a scheme often also has to be secure against *chosen-ciphertext attacks* (CCA) [127]. The model in Definition 3 can easily be adapted to model this, i.e., by including a decryption oracle. The attacker is allowed to query this oracle with ciphertexts other than the challenge ciphertext.

4.4 The random oracle model

Some schemes are proven secure in the *random oracle model* (ROM) [28], i.e., its security proofs use random oracles. In practice, the random oracles are replaced by cryptographic hash functions [129], which are then assumed to behave randomly. While this is an idealized functionality of a hash function, Bellare and Rogaway [28] argue that it is sufficiently random for its purpose in most cases. However, Canetti, Goldreich and Halevi [38] show that schemes exist that are secure in the ROM, but for which no implementation of a hash function can be found that yields a secure scheme. Although such insecure schemes differ substantially from real-world constructions [89, 94], it is unclear if such problems may translate to any established ABE schemes, and whether they can be exploited in practice (Direction 2).

4.5 Static and parametrized complexity assumptions and the generic group model

ABE schemes are often proven secure by reducing a complexity assumption to its security. Some proofs use *static* assumptions such as the *decisional bilinear Diffie–Hellman* (DBDH) [133], the *(symmetric) external Diffie–Hellman* ((S)XDH) [50], *decisional linear* (DLIN) [4] and *subgroup assumptions* [99]. Other proofs use *parametrized* or *q-type assumptions*, which grow linearly in some parameter q that is often dependent on some system parameters. Many of the q-type assumptions used in these proofs can be generalized under the "uber-assumption" [33, 35], which is shown to generically hold in the *generic bilinear group model* (GGM) [138]. However, the security level of a scheme may decrease as q increases [51], which makes them less attractive than static assumptions. Another difference between static and parametrized assumptions is the analysis of their security. Static assumptions are often concise, simple to understand and derived from well-understood assumptions. In contrast, parametrized assumptions consist of many inputs and are often tailored to prove security of one specific scheme. As such, each assumption needs to be carefully studied.

Finally, note that some schemes directly prove security in the GGM [10, 29]. Much like the random oracle model (ROM), the GGM is considered an idealized security model and a proof herein provides only a basic level of confidence. Dent [59] shows that schemes exist that are provably secure in the GGM, but can be broken in practice. Regardless, much like in the ROM setting, it is unclear if this is also the case for ABE. In fact, Ambrona et al. [10] show that a strong relationship exists between generic attacks and the generic security of ABE schemes. (They additionally show that several selectively secure schemes can be proven fully secure in the GGM, with only polynomial security loss, which in part addresses Direction 1.) However, this does not include attacks that are non-generic, e.g., that exploit the used hash function (in the ROM) or the underlying group structure (in the GGM).

Direction 2 (Security of ABE with proofs in idealized models in practice) *To analyze the security of ABE schemes that are provably secure in the random oracle model or generic (bilinear) group model in real-world implementations, e.g., with respect to their instantiated hashes or underlying groups.*

5 Pairing-based ABE

In this section, we review pairing-based ABE. To this end, we discuss the common structure used in many schemes, and how some of the properties discussed in Sect. 3 can be achieved.

In general, due to the collusion resistance property (Sect. 4.1), ABE requires security guarantees for both the secret keys and ciphertexts. This is unlike more traditional forms of public-key encryption, such as ElGamal [67], which typically only require security guarantees for the ciphertexts. To ensure security of such encryption schemes, the ciphertexts are defined in groups in which the decisional Diffie–Hellman problem [31, 60] is assumed to be hard. Roughly, it should then (provably) hold that illegitimately decrypting the ciphertext is just as hard as solving the discrete-log problem. To ensure that security guarantees can be achieved for both the keys and ciphertexts, it makes sense to place both the keys and ciphertexts in such groups, which however may complicate decryption. To overcome these difficulties, pairings can be used.

5.1 Pairings

A *pairing*—also known as a *bilinear map*—is a map $e\colon \mathbb{G} \times \mathbb{H} \rightarrow \mathbb{G}_T$ defined over three groups $\mathbb{G}$, $\mathbb{H}$ and $\mathbb{G}_T$ of prime order p with generators $g \in \mathbb{G}$, $h \in \mathbb{H}$ such that (i) $e(g^a, h^b) = e(g, h)^{ab}$ for all $a, b \in \mathbb{Z}_p$ (bilinearity), (ii) $e(g, h)$ is not the identity in $\mathbb{G}_T$ (non-degeneracy) and (iii) e is efficiently computable. In some schemes, the order of the groups is a composite of three distinct primes. Because the group and pairing operations in such groups are one to two orders of magnitude less efficient than in its prime-order counterparts [76], prime-order groups are preferred.

Depending on the relationship between $\mathbb{G}$ and $\mathbb{H}$, different *types* of pairings exist. If $\mathbb{G} = \mathbb{H}$, then the pairing is *symmetric* and called a *type-I* pairing. If not, then the pairing is *asymmetric*. Subsequently, if an efficiently computable homomorphism from $\mathbb{H}$ to $\mathbb{G}$ exists, then it is a *type-II* pairing, and otherwise, it is a *type-III* pairing. While most schemes are designed in the type-I setting, type-III pairings should be used in practice, due to computational efficiency [66] and security issues in the type-I setting [65]. In general, schemes designed in the type-I setting can be securely converted to the type-III setting [1, 2, 6].

5.2 Access structures: representation of (non-)monotone span programs

In pairing-based ABE, (non-)monotone span programs are typically represented by access trees [72] or linear secret sharing scheme (LSSS) matrices [73]. Most often, LSSS matrices are used, which can be efficiently generated with the methods described in e.g., [96].

Definition 5 (*Access structures represented by LSSS matrices* [73]) An access structure can be represented as a pair $\mathbb{A} = (\mathbf{A}, \rho)$ such that $\mathbf{A} \in \mathbb{Z}_p^{n_1 \times n_2}$ is an LSSS matrix, where $n_1, n_2 \in \mathbb{N}$, and ρ is a function that maps the rows of $\mathbf{A}$ to attributes in the universe. Then, for some vector with randomly generated entries $\mathbf{v} = (s, v_2, ..., v_{n_2}) \in \mathbb{Z}_p^{n_2}$, the i-th secret generated by this matrix is $\lambda_i = \mathbf{A}_i \mathbf{v}^\intercal$, where $\mathbf{A}_i$ denotes the i-th row of $\mathbf{A}$. If $\mathcal{S}$ satisfies $\mathbb{A}$, then there exist a set of rows $\Upsilon = \{i \in \{1, ..., n_1\} \mid \rho(i) \in \mathcal{S}\}$ and coefficients $\varepsilon_i \in \mathbb{Z}_p$ for all $i \in \Upsilon$ such that $\sum_{i \in \Upsilon} \varepsilon_i \mathbf{A}_i = (1, 0, ..., 0)$, and by extension, $\sum_{i \in \Upsilon} \varepsilon_i \lambda_i = s$ holds.

5.3 Example: the Wat11 scheme

To illustrate what pairing-based ABE schemes look like, we give an example of a CP-ABE scheme. Arguably the simplest CP-ABE scheme is the first Wat11 [145] scheme. It is the CP-ABE variant of the first expressive KP-ABE scheme, i.e., the GPSW06 [72]. The scheme is originally defined in the prime-order and symmetric setting (and only provides selective security (Sect. 4.2)). This scheme was later improved on in many works [5, 13–15, 92, 99, 107], attaining better levels of security and/or practicality.

Example 1 (The Wat11 [145] scheme)

– Setup(λ): Let $p, \mathbb{G}, \mathbb{G}_T, e, g$ be generated as in Sect. 5.1, such that e is a symmetric pairing and provides sufficient security with respect to security parameter λ. The key generation authority (KGA) also initializes universe $\mathcal{U}$ and randomly generates $\alpha, b, b_{\text{att}} \in_R \mathbb{Z}_p$ for all att $\in \mathcal{U}$. It keeps $\text{MSK} = (\alpha, b, \{b_{\text{att}}\}_{\text{att} \in \mathcal{U}})$ as the master secret key and publishes the master public key

$$\text{MPK} = (p, \mathcal{U}, \mathbb{G}, \mathbb{G}_T, e, g, A = e(g, g)^\alpha, B = g^b, \{B_{\text{att}} = g^{b_{\text{att}}}\}_{\text{att} \in \mathcal{U}})$$

– KeyGen(MSK, $\mathcal{S}$): For a user that possesses a set of attributes $\mathcal{S}$, the KGA randomly generates $r \in_R \mathbb{Z}_p$, and returns as secret key:

$$\text{SK}_{\mathcal{S}} = (\mathcal{S}, K = g^{\alpha - rb}, K' = g^r, \{K_{\text{att}} = g^{rb_{\text{att}}}\}_{\text{att} \in \mathcal{S}}).$$

– Encrypt(MPK, $\mathbb{A}$, M): An encrypting user encrypts message $M \in \mathbb{G}_T$ under access policy $\mathbb{A}$ with $\mathbf{A} \in \mathbb{Z}_p^{n_1 \times n_2}$, $\rho\colon \{1, ..., n_1\} \to \mathcal{U}$. The user then randomly generates integers $s, s_1, ..., s_{n_1}, v_2, ..., v_{n_2} \in_R \mathbb{Z}_p$ and computes the ciphertext as

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

such that λ_i denotes the i-th entry of $\mathbf{A} \cdot (s, v_2, ..., v_{n_2})^\intercal$.
– Decrypt($\text{SK}_{\mathcal{S}}, \text{CT}_{\mathbb{A}}$): Suppose that $\mathcal{S}$ satisfies $\mathbb{A}$, and suppose $\Upsilon = \{j \in \{1, ..., n_1\} \mid \rho(j) \in \mathcal{S}\}$, such that $\{\varepsilon_j \in \mathbb{Z}_p\}_{j \in \Upsilon}$ exist with $\sum_{i \in \Upsilon} \varepsilon_j \mathbf{A}_j = (1, 0, ..., 0)$. Then the plaintext M is retrieved by computing

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

The scheme is correct, i.e., decryption indeed yields M:

$$M \cdot e(g,g)^{\alpha s} / \left(e(g^{\alpha - rb}, g^s) \cdot \prod_{i \in \Upsilon} (e(g^r, B^{\lambda_i} B^{s_i}_{\rho(i)}) / e(g^{rb_{\rho(i)}}, g^{s_i}))^{\varepsilon_i} \right)$$

$$= M \cdot e(g,g)^{\alpha s} / \left(e(g^{\alpha - rb}, g^s) \cdot \prod_{i \in \Upsilon} (e(g^r, g^{\lambda_i b + s_i b_{\rho(i)}}) / e(g^{rb_{\rho(i)}}, g^{s_i}))^{\varepsilon_i} \right)$$

$$= M \cdot e(g,g)^{\alpha s} / \left(e(g,g)^{\alpha s - rsb} \cdot e(g,g)^{\sum_{i \in \Upsilon} \varepsilon_i r \lambda_i b} \right)$$

$$= M \cdot e(g,g)^{\alpha s} / \left(e(g,g)^{\alpha s - rsb} \cdot e(g,g)^{rsb} \right) = M.$$

5.4 Standard form

Many pairing-based schemes have a similar structure as the Wat11 scheme. In particular, the keys and ciphertexts mainly exist in the source group(s)—with the exception of the first ciphertext component, which is almost always $C = M \cdot e(g,g)^{\alpha s}$—and decryption consists of pairing the appropriate key and ciphertext components. This common structure is explicitly considered in frameworks that consider generic compilers (Sect. 6.2), which abstracts the schemes by analyzing the "exponent space". For instance, the exponent space of the secret keys of the Wat11 scheme (Definition 1) can be described as a vector of elements in $\mathbb{Z}_p$, i.e., $\text{sk}_{\mathcal{S}} = (k = \alpha - rb, k' = r, \{k_{\text{att}} = rb_{\text{att}}\}_{\text{att} \in \mathcal{S}})$. In a more general sense, this vector can be expressed as a function of the master-key α, some variables $\mathbf{b}$ associated with the public keys, and some variables $\mathbf{r}$ associated with the secret key. Similarly, such a vector can be defined for the ciphertexts. We summarize these findings by defining the *standard form* of predicate encryption, which is derived from the aforementioned generic compiler frameworks [13, 146].

Definition 6 (*Standard form of predicate encryption* [13, 146]) The standard form of predicate encryption is defined as follows:

– Setup(λ): Taking as input the security parameter λ, the KGA generates three groups $\mathbb{G}, \mathbb{H}, \mathbb{G}_T$ of order p with generators $g \in \mathbb{G}, h \in \mathbb{H}$, and chooses a pairing $e \colon \mathbb{G} \times \mathbb{H} \to \mathbb{G}_T$. The KGA also defines the universe of attributes $\mathcal{U}$, and generates random $\alpha, \mathbf{b} = (b_1, ..., b_n) \in_R \mathbb{Z}_p$ such that $n \in \mathbb{N}$ is some integer. It outputs MSK $= (\alpha, \mathbf{b})$ as its master secret key and publishes the master public key as

$$\text{MPK} = (g, h, e(g,h)^{\alpha}, g^{\mathbf{b}}, h^{\mathbf{b}}).$$

We refer to $\mathbf{b}$ as the *common variables*, because they occur in both the secret keys and ciphertexts. We refer to α as the master-key, as it can be used to decrypt any ciphertext.

– KeyGen(MSK, y): The KGA generates a secret key for y by generating user-specific random integers $\mathbf{r} = (r_1, r_2, ...) \in_R \mathbb{Z}_p$ and computing the secret key as

$$\text{SK}_y = (\mathcal{S}, h^{\mathbf{k}(\alpha, \mathbf{r}, \mathbf{b}, y)}),$$

such that $\mathbf{k}$ denotes a vector defined over the user-specific random variables, master secret keys and associated set of attributes.

– Encrypt(MPK, x, M): An encrypting user encrypts message $M \in \mathbb{G}_T$ for x by generating ciphertext-specific randoms $\mathbf{s} = (s, s_1, s_2, ...) \in_R \mathbb{Z}_p$ and computing the ciphertext as

$$\text{CT}_x = (\mathbb{A}, M \cdot e(g,h)^{\alpha s}, g^{\mathbf{c}(\mathbf{s}, \mathbf{b}, x)}),$$

such that $\mathbf{c}$ denotes two vectors defined over the ciphertext-specific random variables, master public keys and associated access structure.
– Decrypt(SK, CT): Let SK $= (y, \mathbf{K} = h^{\mathbf{k}})$ be a secret key and CT $= (x, C = M \cdot e(g,h)^{\alpha s}, \mathbf{C} = g^{\mathbf{c}})$ a ciphertext such that $P(x, y) = 1$. Define $\mathbf{E}(x, y)$ as the matrix such that we have $\mathbf{c}\mathbf{E}\mathbf{k}^{\intercal} = \alpha s$. Then, we retrieve plaintext M by computing

$$C/\left(\prod_{i,j} e(C_i, K_j)^{\mathbf{E}_{i,j}}\right),$$

where $\mathbf{C} = (C_1, C_2, ...)$ and $\mathbf{K} = (K_1, K_2, ...)$.

Observation 3 *As suggested by Attrapadung [13], the standard form implies a metric that can be used to measure the "similarity" between two schemes. For instance, by analyzing the LOSTW10 [99] and Wat11 [145] schemes, one would conclude that these two schemes have similar structures. In fact, if one were to abstract both schemes to the vectors* $\boldsymbol{b}$, $\boldsymbol{k}$ *and* $\boldsymbol{c}$ *as in Definition 6, they would turn out to be the same. The main difference between the two schemes is the underlying group structure: whereas Wat11 is built on prime-order groups, LOSTW10 is built on composite-order groups. As it turns out, many CP-ABE schemes are structurally similar to the Wat11 scheme, and oftentimes only differ in the underlying group structure [13]. The reason for this is that the Wat11 scheme has an efficient "vector structure"—often referred to as pair encoding—compared to other CP-ABE schemes, but it is provably secure in a weaker model and under less established assumptions than would be desirable (Sect. 4). In contrast, the derived schemes [5, 13–15, 92, 99, 107] are provably secure in stronger models and under more established assumptions, possibly at the cost of some basic functionality and, importantly, the efficiency.*

5.5 Supporting large universes

We analyze the methods that are used to support large universes. As argued in Sect. 3.3, from a practical viewpoint, we prefer large-universe constructions over small-universe schemes. Although pairing-based small-universe schemes are simpler to create, they can often be converted into the large-universe setting [47, 145]. This is because many such small-universe constructions use only the master public key associated with an attribute in the key generation (and not some associated secret key). For example, in the Wat11 scheme (Definition 1), the generator $B_{\text{att}} = g^{b_{\text{att}}}$ is used in the key generation and encryption algorithms. Two techniques exist that allow this generator $g^{b_{\text{att}}}$ to be generated by a function. First, a full-domain hash (FDH) $\mathcal{H}\colon \{0,1\}^* \rightarrow \mathbb{G}$ can be employed that maps strings directly into the group. This FDH is modeled as a random oracle [72]. Second, a collision-resistant hash function can be used to map strings in $\mathbb{Z}_p$, and then an implicit n-degree polynomial is used to map the integer in the group [133]. For instance, the KGA can generate an n-degree polynomial $f(\text{att}) = \sum_{i=0}^{n} a_i x_{\text{att}}^i$, publish "coefficients" $g^{a_0}, ..., g^{a_n}$ and define the mapping F as $F(\text{att}) = \prod_{i=0}^{n} (g^{a_i})^{x_{\text{att}}^i} = g^{f(\text{att})}$, where we assume that x_{att} is the hashed representation of att in $\mathbb{Z}_p$. The latter is sometimes also called a "Boneh-Boyen" hash [32].

We give an overview of several advantages (+) and disadvantages (−) of the two methods, and their relationship with other properties of ABE. For the "FDH-based" method, we identify the following advantages and disadvantages:

+ It is simple to apply to many small-universe schemes, e.g., [126, 145];

+ It typically yields schemes that attain the same structure, and therefore, at first glance, seem to attain the same efficiency as the small-universe counterpart.
− It may however negatively impact the efficiency of the key generation and encryption algorithms compared to its small-universe variant (Observation 7);
− Currently, no techniques exist that allow for the additional support of non-monotonicity (Sect. 5.7);
− It cannot benefit from online/offline techniques like polynomial-based large-universe constructions (Sect. 10.1), and therefore cannot significantly improve the efficiency of the key generation and encryption algorithms in practice.
− Its security is proven in the random oracle model (Sect. 4.4).

For the "polynomial-based" method, we identify the following advantages and disadvantages:

+ Techniques exist that allow for the additional support of non-monotonicity (Sect. 5.7);
+ The mapping is solely determined by group elements g^{a_i}, which may positively influence the efficiency of the key generation and encryption algorithms (Observation 7);
+ The online/offline variants of these schemes (Sect. 10.1) allow for a split of the computational costs into an online and offline phase. The online phase requires almost no computational costs, which significantly improves the efficiency of key generation and encryption in practice;
+ Its security does not rely on the random oracle model (Sect. 4.4).
− Achieving unboundedness is non-trivial (Sect. 5.6);
− Existing schemes typically incur a heavy trade-off in practicality and efficiency, e.g., by being bounded [72, 143] or by having decryption [47, 101, 130] or key generation [16] algorithms that are several factors more computationally costly than the other algorithms (Observation 2 and Sect. 7.3).

5.6 Achieving unboundedness

As we mentioned in Sect. 3.4, some schemes are bounded in one or more system parameters, such as the attribute sets or policies, or the number of times that a specific attribute may occur in the policy. The most notable reason why any of these parameters may be bounded is related to the amount of randomness that is used in the scheme. For instance, in the second Wat11 [143, 145] scheme, the same randomness is used for all attributes in the ciphertext. That is, the ciphertexts in Example 1 are replaced by ciphertexts of the form:

$$\mathrm{CT}_{\mathbb{A}} = (\mathbb{A}, C = M \cdot A^s, C' = g^s, \{C_{1,j} = B^{\lambda_j} B^s_{\rho(j)}\}_{j \in [1,n_1]}),$$

i.e., the $B_{\rho(j)}$ is also randomized with s instead of a fresh randomizer s_j. As a result, the number of times that the attribute $\rho(j)$ can occur in the policy is restricted to one. If it is used more than once, B^{λ_j} is not hidden anymore. Similarly, the sets or policies may be bounded due to the use of insufficient randomness in the keys or ciphertexts. For example, in [72, 133], the authors convert the small-universe constructions into large-universe constructions by replacing the generator B_{att} by an implicit n-degree polynomial. However, such a polynomial provides only sufficient randomness for n attributes. Hence, intuitively, plugging it into e.g., the keys in the Wat11 scheme in Example 1, where each key component is randomized with the same randomness r, places a bound[2] on the size of the attribute set: n.

[2] Actually, the large-universe construction in Appendix B of the full version [143] is bounded in both the sets and policies, because it uses only one randomizer in the keys, and one in the ciphertexts.

To remove those bounds, more randomness can be used. For example, to lift the one-use restriction,[3] one could use a fresh randomizer for each attribute [145] or for each re-use of one specific attribute [5]. For polynomial-based large-universe schemes, this is more difficult. Intuitively, the reason why this is difficult is in the keys. In general, the keys are tied to one specific user by using the same randomness (e.g., see the discussion in Sect. 8.4). For example, in Wat11, the same r is used for all key components. Hence, replacing B_{att} with a polynomial-based hash $F(\text{att})$ yields a bound on the attribute set associated with the keys, as $F(\text{att})^r$ only provides sufficient randomness for n attributes. Lewko and Waters [101] and Rouselakis and Waters [130] overcome this issue by adapting these schemes such that the keys can be tied to one specific user while fresh randomness can be used for each attribute. This subsequently complicates the proof techniques, and fixes the polynomial to the case where the degree $n = 1$.

5.7 Supporting non-monotonicity

We analyze the existing methods used to support non-monotonicity. As mentioned, schemes that support non-monotone access structures are more expressive. Moreover, they may simplify the support of e.g., revocation [95] (Sect. 10.2). In general, we observe that two methods exist to support non-monotonicity in ABE. First, the most straightforward way is to include a negative attribute for each attribute in the universe [52]. During key generation, a key is generated for each attribute in the universe: a positive instance for each attribute in the set, and a negative instance for every other attribute in the universe. The advantage of this method is that it can be applied to any small-universe scheme, and the efficiency of the encryption and decryption algorithms are the same as in the monotone setting. The disadvantage of this technique is that it inherently requires the support of small universes only [15], and the key generation costs grow in the size of the universe. Furthermore, supporting non-monotonicity in this way causes issues when attributes are added to the universe, which we discuss in more detail in Sect. 9.

Ostrovsky et al. [124] devised another method—hereinafter referred to as OSW-method—which also supports large universes. In particular, their method exploits the structure of large-universe constructions that use the implicit n-degree polynomial (Sect. 5.5), and can thus also be applied to completely unbounded schemes [15, 148]. Roughly, this method requires that, during decryption, the entire set of attributes associated with the key is compared with the negated attribute. The decrypting user only satisfies this negation, if all attributes are different from the negated attribute. On a more technical level, this is achieved via the polynomial by exploiting Lagrange interpolation, also frequently used in secret sharing [137]. Intuitively, a partial decryption key is shared among the attributes in the set associated with the key such that one more secret share is needed to recover the decryption key. During decryption with a negated attribute, the decryption key can only be reconstructed if the negated attribute is different from all attributes in the set. The disadvantage of this method is, however, that this comes at the cost of some efficiency. For example, compared to RW13 [130], its non-monotone variant in [148] has keys that are twice as large (and thus the key generation costs are doubled). The decryption costs for negated attributes scale linearly in the size of the set of attributes associated with the key.

[3] In some areas of ABE, this restriction is not as easily lifted as we suggest here. For example, some proof techniques used in the dual system encryption paradigm (Sect. 6.2) require a one-use restriction for a different reason. This restriction was lifted by Kowalczyk and Wee [92] by introducing novel proof techniques.

In the realm of pairing-based ABE, a special subtype of non-monotonicity—which we shall refer to as "labeled non-monotonicity"—was proposed by Okamoto and Takashima [122], and was later further investigated and developed in [22, 141]. These works explicitly use the attribute labels—corresponding to our notion of attribute types—in access policies. For instance, the attribute "doctor" can belong to the sub-universe labeled as "profession". Then, the attribute can be negated in two ways: "NOT profession: doctor" or "profession: NOT doctor". In the first case, a set of attributes satisfies the negation if it does not contain the attribute "profession: doctor". In the second case, a set satisfies the negation if it does contain at least one attribute with the label profession, but not with the value "doctor". In particular, the negation is not satisfied if the set does not contain any attributes with the label. We call the latter type of non-monotonicity "labeled non-monotonicity".

Labeled non-monotonicity can be used to securely implement access control that supports negations in dynamic practical settings [22, 141]. For instance, consider a situation in which a new label is added to a system and used in a negation during encryption. At this point, none of the users have an attribute for this label yet; as such, each user would automatically satisfy the negation "NOT profession: doctor". In contrast, users do not automatically satisfy the negation "profession: NOT doctor", as they need at least one attribute associated with the label "profession". To satisfy it, they would first need to request a new key for a set of attributes that also includes the new label. (Note, however, that the user may possibly not possess any attributes associated with the label. In this case, the user is assigned an empty value, e.g., "profession: none".) We show in Sect. 9.2 that, compared to other techniques achieving non-monotonicity, techniques using labeled non-monotonicity provide the best availability properties without compromising the security of the schemes.

Observation 4 *Currently, large-universe schemes that support non-monotone access structures incur a significant efficiency trade-off. We identify two underlying reasons for this:*

- *Only the second non-monotonicity method—which we dubbed the OSW-method—can simultaneously support large-universeness and non-monotonicity. As such, the resulting schemes suffer from the same decryption inefficiency as unbounded ABE using the polynomial method, as pointed out in Sect. 7.3;*
- *The OSW-method requires that, during decryption, the entire set of attributes is compared to the negated attribute, incurring computational costs that are linear in the set.*

To some extent, labeled non-monotonicity mitigates both these issues. Most obviously, it mitigates the second issue by only requiring that a subset of attributes associated with the same label as the negated attribute is compared. A less obvious reason is that the attribute labels constitute another "layer" of the universe of attributes [22]. For this layer, we do not require non-monotonicity, so we can use both methods to support large-universeness as discussed in Sect. 5.5. In this way, it may be possible to achieve a more desirable efficiency trade-off. For instance, to optimize decryption, we can use an FDH rather than a polynomial. Conversely, to optimize the key generation efficiency or to benefit from online/offline extensions, we can use a polynomial.

For example, the TKN20 [141] scheme supports labeled non-monotonicity by using FDH-based and polynomial-based methods to support large universes. Unlike FDH-based large-universe constructions, the scheme uses an FDH to map the universe labels to the group. Within each labeled universe, it maps the attributes to the group by using the polynomial method (Sect. 5.5). In this way, the non-monotonicity supported with (a simplified version[4]

[4] In the case of TKN20 [141], each attribute label may occur only once in the set of attributes. Attrapadung and Tomida [22] later lift this restriction by applying the OSW-method in the "attribute layer".

of) the OSW-method ensures that not the entire set of attributes needs to be compared to the negated attribute during decryption. Rather, only the subset of attributes associated with the negated attribute's label needs to be compared. While this yields a more efficient decryption algorithm compared to schemes that support non-labeled non-monotonicity, the use of an FDH may decrease the efficiency of the key generation and encryption algorithms (Sect. 7.6). On the other hand, if we use the polynomial method to map the labels to the group, as is proposed in [22], then decryption requires a linear number of pairing operations per matching attribute. Possibly, a more balanced efficiency can be achieved if Direction 5 is explored.

6 Security of pairing-based ABE

We review some important techniques and developments in proving security of pairing-based ABE.

6.1 Selective security through "program-and-cancel" strategies

In general, the choice of security model depends on the proof strategy. In many early works [72, 133, 145], the "program-and-cancel" strategy is used to prove security. In this proof strategy, the challenge access structure is embedded in the public keys. During the key query phase, the set of the rows of the access structure is split in two subsets: the set of rows associated with the attributes that are also in the set associated with the key, and its complement. This can only be done if the attacker commits to the challenge access structure before the setup (or key query) phase is run. This strategy is therefore mostly used to prove selective (or semi-adaptive) security. Another characteristic of selective security proofs using the program-and-cancel technique is that, in the ciphertext-policy and prime-order setting, the used complexity assumption is oftentimes q-type [130, 145]. On a high level, this is due to how the policies are embedded in the public keys.

Note that selective proof techniques can also be utilized in full security proofs [102]. In fact, for some schemes, currently the only way to prove full security is to use selective proof techniques [5, 13]. As a consequence, these schemes have, at best, a full security proof under a q-type assumption.

6.2 Full security through dual system encryption

Proving full security is difficult but important. As our taxonomy in Table 3 shows, the minority of schemes is proven fully secure. This is, in part, because selective security is arguably easier to prove, and typically yields more efficient schemes. A more important reason is that many generic frameworks and transformations exist that do not necessarily aim to build one ABE scheme—and are therefore not listed in our taxonomy—but generalize existing structures and transformations to simultaneously achieve certain properties. This simplifies the construction of ABE schemes with many desirable properties while attaining strong security guarantees.

For the past decade, much progress has been made in achieving stronger security guarantees for the existing selectively secure schemes. Currently, the most efficient fully secure versions (e.g., [92, 107]) of their selectively secure counterparts (e.g., [72, 145]) incur roughly twice the storage and computational costs. These works use and improve the *dual system encryption* methodology introduced by Waters [144]. Interestingly, a vast body of literature exists in this area, e.g., [45, 46, 48, 49, 63, 98–100, 121]. We believe that, in itself, this subfield within pairing-based ABE can benefit from a systematized overview. To avoid heavy, technical explanations without entirely avoiding these accomplishments, we merely mention some interesting, recent results.

Generic compilers

To simplify the design and analysis of fully secure schemes, generic frameworks are formulated within the dual system encryption framework, which define generic transformations or *compilers* [3, 5, 13, 14, 45, 146]. These compilers facilitate simplified security proofs by proving security of the underlying group structure generically. They ensure that the designer only needs to prove simple notions of security (such as information-theoretic ones) over the exponent space. Notably, [5] (and by extension [10]) only requires algebraic notions of security, which are derived from selective security proof techniques. Effectively, they prove security generically for any scheme that is not trivially broken. In particular, a scheme is trivially broken if an unauthorized key exists that can decrypt a challenge ciphertext. As a consequence, many selectively secure schemes can be transformed into fully secure schemes (albeit under a q-type assumption). Note that many of these generic frameworks prove stronger notions of security for previously constructed schemes, such as the Wat11 [145] and RW13 [130] schemes.

Generic conversions and compositions

Not only the underlying group structures have been analyzed, but also the structural transformations that are used to achieve certain properties. Several works are dedicated to converting schemes with certain properties into schemes with other properties [17, 23]. These works are built on the generic compiler frameworks of Attrapadung [13, 14]. Other works show that certain transformations on the predicates, e.g., conjunctions or negations, preserve security [9, 15, 22]. In particular, [9, 15] are instantiated in the framework of Agrawal and Chase [5]. As a result, schemes satisfying properties such as complete unboundedness, non-monotonicity and constant-size ciphertexts can be constructed, whilst attaining strong security guarantees (e.g., full security under static assumptions in the standard model [22, 108]).

6.3 Conversion from CPA to CCA-security

Most ABE schemes are only proven CPA-secure, though there are some exceptions [52, 121, 149]. Oftentimes, generic conversion methods can be applied, such as methods using non-interactive zero-knowledge proofs [127], or key-encapsulation techniques such as the Fujisaki-Okamoto transformation [64], which both yield security in the random oracle model. Whereas such key-encapsulation techniques may be more efficient in practice, for completeness, we briefly review other techniques as well.

Some conversion methods avoid random oracles by exploiting specific properties of ABE. Yamada et al. [147] give two methods that use the Canetti-Halevi-Katz transformation [39] to generically obtain CCA-security. The first method considers the *delegatability* of a scheme. A scheme is delegatable if a secret key associated with a set of attributes can be transformed into a secret key associated with a smaller set of attributes. The second method considers the *verifiability* of a scheme. A scheme is verifiable if it can be verified for two sets of attributes whether their associated keys decrypt to the same value. For schemes that are not delegatable or verifiable, Koppula and Waters [90] describe a conversion method that can always be used. In general, it seems that the most efficient CCA-secure schemes (avoiding random oracles) can be constructed from large-universe constructions with the verifiability method, as it only requires a few additional ciphertext components and a few additional pairing operations in the decryption.

7 Efficiency of pairing-based ABE

One of the most important practical aspects of any cryptographic primitive is the efficiency. Compared to other primitives that are used to implement access control (as discussed in Sect. 2), ABE generally requires more computational power on the user side. To narrow the efficiency gap, it is paramount that the computational costs of ABE are minimized. In this section, we critically review some aspects related to efficiency of ABE. We also outline some directions related to making ABE more efficient, and properly measuring and comparing efficiency.

7.1 The storage costs

In general, the efficiency of an ABE scheme is determined by the computational and storage costs. For the *storage costs*, the sizes of the public and secret keys, as well as the ciphertexts are considered. In practice, it may be more important to optimize the size of the ciphertexts, rather than the keys. For instance, the secret keys are stored on the decryption device, which may not need to be updated frequently after key generation. In contrast, this device may frequently receive ciphertexts to decrypt from other data sources. For mobile devices with limited data subscriptions, it might be problematic to have large ciphertexts. For storage-constrained devices such as sensors and other IoT devices that encrypt data using ABE, such ciphertexts may simply be too large, yielding a problem on the encrypting user's side. Therefore, minimizing the ciphertext size may be desired or even required in these settings.

To support ABE on resource-constrained devices, schemes with constant-size or short ciphertexts [3, 5, 81] can be deployed—possibly alongside another scheme that is more suitable for less constrained devices. As we will show in the taxonomy (Table 3), these schemes typically incur various trade-offs in flexibility (imposing bounds on the universe, access policies or sets of attributes) or expressivity (only supporting AND-gates or threshold functions). The only exception is the AHMTY16 [16] scheme, which is expressive, KP-based, achieves complete unboundedness, and has short ciphertexts. The short ciphertexts come with a prominent trade-off: the key size—and by extension the key generation costs—are larger by a factor that is linear in one of the system parameters compared to other popular schemes [130, 145]. Hence, we would recommend that, in settings in which the encryption devices are resource-constrained or powerful, both a scheme with short ciphertexts and a scheme with more balanced sizes and computational costs are deployed. Finally, note that a CP-ABE scheme with similar features as AHMTY16 does not exist yet, even though this may be useful in practice. We therefore formulate the following direction.

Direction 3 (CP-ABE with short ciphertexts) *To design an unbounded CP-ABE scheme with short ciphertexts that supports expressive access policies.*

7.2 Computational costs

For the *computational costs*, the performance of some or all of the algorithms—i.e., the setup, key generation, encryption and decryption—is considered. In practice, some of these algorithms may be performed more often than others. Key generation is ideally run only once for each user, while encryption is performed much more often. In turn, because multiple users can decrypt a ciphertext, decryption may be performed more often than encryption. Furthermore, the encryption and decryption devices may have different computational resources, e.g., the average encryption device may be an IoT device while the average decryption device is a personal computer. It is therefore important to take such practical considerations into account when analyzing the efficiency of a scheme.

7.3 Theoretical performance considerations

Oftentimes, the computational efficiency of ABE is theoretically analyzed by counting the operations required by the algorithms. In this way, we can gather rough estimates on the efficiency of certain schemes without requiring any knowledge on cryptographic engineering. Although this approach is simple and may already give a good view on how certain schemes compare, they may fall short when the efficiency of the required operations cannot be effectively estimated. As an example, we theoretically analyze the efficiency of various large-universe schemes (Sect. 5.5), which can be divided in two categories: schemes that support this via an FDH or a polynomial.

For FDH-based large-universe schemes, we know that the schemes are relatively close to their small-universe counterpart. Because the most efficient small-universe constructions incur only a constant number of pairing operations during decryption [145], their large-universe counterparts incur a similar decryption efficiency [4]. However, as we show in Observation 7, the use of an FDH impedes the key generation and encryption efficiency of the scheme when it is implemented in practice.

For polynomial-based schemes, it is less clear what the most efficient construction is. Some bounded schemes using an n-degree polynomial [143] have an efficient decryption algorithm, as they only require a constant number of pairing operations. However, computing the polynomial-based hash requires n exponentiations. Because encryption needs to evaluate the hash for each attribute, the encryption costs grow linearly in not only the number of attributes, but also the degree of the hash. In contrast, in most unbounded schemes [10, 130], the key generation and encryption costs grow only in the number of attributes, while the number of pairings required during decryption grows in the number of attributes.

As an exception, the unbounded polynomial-based scheme by Attrapadung et al. (AHMTY16) [16] provides a slightly different (but flexible!) efficiency trade-off. Essentially, this scheme is the unbounded variant of the (bounded) constant-size scheme by Attrapadung et al. (ALP11) [21], using techniques of Lewko and Waters [101] to make it unbounded. In this way, the scheme inherits the unbounded features of linear-sized schemes such as [101, 130], while it can enjoy the potentially desirable efficiency trade-offs provided by bounded schemes such as ALP11. Importantly, it provides this latter feature flexibly, allowing practitioners to choose the efficiency trade-offs. While the key generation costs grow in the number of attributes and some chosen parameter, the pairing operations required during decryption decrease by a factor in this parameter compared to most unbounded schemes. Note, however, that the number of exponentiations required during decryption also increases by this factor, as well as the public and secret keys.

In summary, so far, all existing large-universe constructions incur a significant efficiency trade-off. On the one hand, FDH-based schemes typically have less efficient key generation and encryption algorithms, but more efficient decryption algorithms [4, 141, 145]. On the other hand, polynomial-based schemes may allow for very efficient implementations of key generation and encryption, but have much less efficient decryption algorithms [10, 101, 130].

Observation 5 *The polynomial-based method may have the potential to provide a more flexible scheme in terms of efficiency, and perhaps even a generally more efficient scheme in practice. However, in order to show that this is the case, more research needs to be conducted in this area. First, it needs to be investigated whether a scheme can be designed that combines the techniques of unbounded schemes such as [130] and compact*[5] *bounded schemes such as [143]. Possibly, this can be achieved with a similar approach as in [16], which combines the unbounded techniques of [101] and bounded techniques of [21]. Not only may this allow for the design of schemes with a more flexible efficiency trade-off, but also for the design of e.g., unbounded schemes with a balanced efficiency or schemes with a very efficient decryption algorithm. Furthermore, it seems that the online/offline variants [84] of unbounded schemes [101, 130] rather explicitly exploit the polynomial structure, which can potentially be generalized. In this way, for all polynomial-based large-universe constructions, the key generation and encryption algorithms can be implemented efficiently, requiring almost no computational costs in the online phase. As a more ambitious goal, combining these results—splitting an unbounded scheme with an efficient decryption algorithm in an online and offline phase—may lead to a generally very efficient scheme in practice.*

To this end, we formulate the following directions.

Direction 4 (Compact unbounded ABE with flexible efficiency) *To construct completely unbounded ABE using the polynomial method with flexible efficiency trade-offs, such that the scheme is compact, i.e., its keys and ciphertexts grow only in the set sizes or policy lengths.*

Direction 5 (Unbounded ABE with efficient decryption) *To construct unbounded ABE using the polynomial method that minimizes the required number of pairing operations per attribute in the decryption without sacrificing the storage efficiency.*

Direction 6 (Generic online/offline conversions) *To formulate a framework that provides generic conversions for any polynomial-based large-universe scheme to the online/offline setting.*

7.4 Implementing and benchmarking schemes

The most empirical way to evaluate the computational efficiency is to implement the scheme and analyze the costs for various numbers of attributes. In most works, the Charm framework is used to prototype and benchmark new schemes [4, 84, 130, 131]. By default, the Charm framework relies on the PBC library [113], a cryptographic library that supports several pairing-friendly groups and the necessary arithmetic. Unfortunately, the PBC library only implements severely outdated groups [24] that provide *much* fewer bits of security than typically desired (or even required) in practice. Some other works [16, 150] directly use the RELIC toolkit [12]—which also implements groups that provide at least 128 bits of security—but these works only consider one specific group: the BN-254 curve [26]. Although this group

[5] Typically, schemes are considered compact if the asymptotic sizes of the keys and ciphertexts depend, in the worst case, linearly on the sizes of the sets or policies [92], e.g., ciphertexts grow only in the policy length.

provided sufficient security when these works were published, they are currently estimated to provide roughly only 100 bits of security [24]. In contrast, Tomida et al. [141] provided a performance analysis that uses a group with sufficient security, but no source code was published, so it is unclear whether they have used libraries for arithmetic.

Observation 6 *Apart from the low security levels, these implementations may have some other shortcomings, notably with respect to the efficiency. First, the implementation of the access structures in the Charm framework is not the most efficient. In particular, they use threshold functions to implement the OR- and AND-gates [72], which yields coefficients ε_i (conform Definition 5) of large sizes. This means that decryption requires at least one full exponentiation for each attribute (see e.g., Definition 1). Instead, using the conversion methods in Appendix G of [96] (like [4] does) would yield $\varepsilon_i \in \{-1, 1\}$. Thus, decryption only requires a multiplication for each attribute. Second, the Charm framework does not readily support optimized implementations of e.g., products of exponentiations [116]. Furthermore, it does support optimizations of exponentiation with precomputation such as [37], but these are not often used in the benchmarks, unlike in practice, where these optimizations are often used to improve the efficiency of implementations.*

It is important to obtain a more realistic view of the efficiency of secure ABE schemes in practice. Recently, de la Piedra et al. [56] set up a framework for implementing and benchmarking the efficiency of ABE schemes, which investigates several relevant layers of optimization. As a proof of concept, they implement multiple ABE schemes, including Wat11 [145] and RW13 [130]. Although this work provides a good foundation for implementing and benchmarking ABE, de la Piedra et al. describe further open problems in this endeavor.

Direction 7 (Optimized and secure implementations of schemes) *To set up a framework for implementing ABE, which supports any state-of-the-art pairing-friendly group and readily uses high-end optimizations.*

Direction 8 (Benchmarking schemes for practice) *To measure the computational efficiency of existing schemes based on the requirements of some specific practical setting, e.g., by taking into account design goals, the computational resources and the frequency that the algorithms are executed.*

7.5 Comparing efficiency of schemes

The goal of benchmarking can be to compare the efficiency of multiple schemes. Usually, this is done by converting the schemes to the type-III setting, choosing a pairing-friendly group, implementing the converted schemes, and subsequently benchmarking them [4, 75, 84, 130, 131]. In some cases, this is done to illustrate the efficiency of some given optimization [75, 84]. In others, the efficiency of a new scheme is compared with other established schemes to illustrate the feasibility [4, 130, 131].

De la Piedra et al. [56] show how ABE schemes can be compared fairly, i.e., by optimizing each scheme in the same way with respect to some design goal. Such a design goal could be, for instance, to optimize the key generation or encryption efficiency or to balance the encryption and decryption efficiency. More generally, design goals are related to which algorithms should be optimized with respect to the computational costs or whether the keys or ciphertexts need to be optimized in size. These goals can typically be specified once the use case is clear for which an ABE scheme is chosen. For instance, in settings involving resource-constrained devices, it is paramount that the algorithm that runs on those devices, e.g., encryption, is optimized. De la Piedra et al. [56] show that, for various such design goals, schemes may be optimized differently, and thus, their efficiency may consequently differ. Therefore, by optimizing multiple schemes with respect to the same clearly-articulated design goal, such schemes can be compared more fairly. To optimize any scheme, they consider four layers of optimization: the arithmetic and group operations, the chosen pairing-friendly group, the order of computations and the type conversion.

We explain the influence of the design goal on the chosen pairing-friendly group and the type conversion in more detail. First, converting a scheme from the type-I to the type-III setting is not trivial and can be done in many ways [1, 2, 6]. Moreover, the efficiency of a scheme depends directly on the chosen conversion technique. The reason for this is that the arithmetic in the three pairing groups varies [11]. Typically, the arithmetic in the first group is the most efficient, while that in the second group is more costly by at least a factor two. As such, the computational costs incurred by computing a secret key or ciphertext component depend on the group in which it lives. For instance, to optimize the encryption efficiency, it is important to place as many ciphertext components in the first group as possible. Consequently, the key components that need to be paired with these ciphertext components during decryption have to be placed in the second group. The chosen conversion technique is therefore implied by the design goal.

Second, many choices exist for pairing-friendly groups in the type-III setting providing 128 bits of security [77]. Much like the chosen conversion technique, the chosen group directly affects the efficiency of a scheme. The reason for this is that the efficiency of the arithmetic in the groups $\mathbb{G}$, $\mathbb{H}$ and $\mathbb{G}_T$ and the pairing operation varies per choice. Some relevant aspects to consider in this are the computational costs of

- an exponentiation in $\mathbb{G}$, $\mathbb{H}$ and $\mathbb{G}_T$;
- (full-domain) hashing in $\mathbb{G}$ and $\mathbb{H}$;
- one pairing computation;
- a product of pairing computations.

In general, various choices of groups provide different trade-offs in efficiency. For example, curve BN-382 provides efficient hashing at the cost of the exponentiation efficiency, while curve BLS12-381 provides more efficient exponentiation at the cost of the hashing efficiency [11]. This means that different schemes may also perform better with respect to some chosen design goal in some groups than in others. Importantly, two schemes may perform best in two different groups; possibly, using one group benefits one scheme and impedes the other. For a fair comparison of two schemes, it should thus be investigated, given some specific design goal, what the most efficient groups are for each scheme.

In sum, potentially, different schemes may be most suitable for different design goals. To find the best scheme with respect to some design goal, we need to be able to know how to convert the scheme from the type-I to the type-III setting and how to determine the best pairing-friendly group. To this end, de la Piedra et al. [56] propose the following directions.

Direction 9 (Type conversion) *Given some design goal, to determine (e.g., with an automated tool) the optimal conversion from the type-I to the type-III setting.*

Direction 10 (Choosing a pairing-friendly group) *With respect to some specific design goal, to determine (e.g., with an automated tool) the pairing-friendly group that yields the most efficient implementation of a scheme.*

7.6 Performance analysis

We analyze the performance of two CP-ABE schemes, mostly to illustrate the feasibility of implementing ABE in practice given the current state of the art. We additionally want to show that analyzing the performance of a scheme is intricate. Due to efficient implementations of certain computations, schemes may sometimes be more efficient than they seem, while others may actually perform worse than expected. We also aim to inspire e.g., designers to follow similar approaches in the comparison of two schemes.

The two CP-ABE schemes that we compare are the large-universe version of the Wat11 [145] scheme using full-domain hashes [143] and the RW13 scheme [130]. We have chosen these particular schemes despite their seniority, because they have the same "vector structure" as many follow-up schemes and extensions. Oftentimes, these only differ in the underlying group structure, and thus, the additional overhead that these schemes incur is known. For instance, the CP variant of KW19 [92] and the unbounded CP-ABE scheme in [22] incur roughly twice the costs of the Wat11 and RW13 schemes, respectively, when instantiated under the SXDH assumption (Sect. 4.5). While Wat11 and RW13 are both selectively secure (Sect. 4.2) under q-type assumptions, the other two schemes [22, 92] are both fully secure under a static assumption (all in the prime-order setting). In this case, we aim for a fair comparison of the efficiency trade-offs incurred by achieving large-universeness by using a full-domain hash or the polynomial method (Sect. 5.5). The two schemes only differ in how the compared property is achieved. The other properties are the same (e.g., selective security, prime-order setting, no bounded variables). This is important, because these properties may also affect the efficiency.

To give a more accurate view on the performance of ABE schemes nowadays, we give performance estimates of the two schemes in a group that currently provides at least 128 bits of security. Specifically, we use the BLS12-446 curve [25] implemented in RELIC [12]—a cryptographic library for efficient constant-time implementations—which provides approximately 132–134 bits of security [78, 79]. Table 1 shows our performance estimates of the key generation, encryption and decryption algorithms on a 1.6 GHz Intel i5-8250U processor. It also shows the sizes of the secret keys and ciphertexts in kilobytes. For this analysis, we have optimized the ciphertext size (and by extension, the encryption efficiency). In Appendix A, our type-converted variants of the schemes can be found (along with a description of the used access structures). Consequently, encryption is much cheaper than key generation. Furthermore, we have used the optimized implementations of fixed-base and multi-base exponentiations as well as multi-pairings provided by RELIC whenever this was possible, which speeds up the algorithms considerably [12]. The performance analysis shows that, even for large inputs, the schemes are very efficient (i.e., they all perform in less than a tenth of a second).

Observation 7 *As mentioned in Sect. 5.5, the support of large universes through FDHs (i.e., as in Wat11) may lead to less efficient key generation and encryption algorithms than through polynomials (i.e., RW13). Our performance estimates in Table 1 also confirm this. In the first place, the FDH itself incurs additional costs. In the second place, as a result of the way that an FDH is modeled, no entity (including the key generation authority) knows the discrete logarithm of a hashed output with respect to the generator. This means that the key and ciphertext components involving a hash need to be in the same group, and have at least one variable base in the exponentiation. As such, the type conversion becomes more restricted and practitioners cannot benefit from highly optimized algorithms for exponentiation. In contrast, polynomial-based schemes are more flexible in these regards. Because of the structure of the*

Table 1 Performance estimates of the large-universe schemes Wat11 [143] with full-domain hash and RW13 [130]

	Key generation:				Encryption:				Decryption:	
	Wat11		RW13		Wat11		RW13		Wat11	RW13
# att	Size	Time	Size	Time	Size	Time	Size	Time	Time	Time
10	0.77	16.2	2.43	13.0	2.36	23.2	2.35	17.4	14.0	23.1
30	1.86	46.3	6.84	36.6	5.66	65.1	5.63	47.7	32.3	59.8
50	2.96	76.4	11.26	60.3	8.96	107.0	8.91	78.1	50.7	96.4
70	4.05	106.5	15.67	83.9	12.26	148.9	12.20	108.4	69.0	133.0

The key and ciphertext sizes are expressed in the number of kilobytes, and the timings are expressed in milliseconds

hash, two versions can be specified: one mapping in the first source group, and one mapping in the second, such that they are homomorphically equivalent. That is, if $F(\text{att}) = \prod_{i=0}^{n}(g^{a_i})^{x_{\text{att}}}$ *is the polynomial-based hash in* $\mathbb{G}$*, then* $G(\text{att}) = \prod_{i=0}^{n}(h^{a_i})^{x_{\text{att}}}$ *can be similarly defined in* $\mathbb{H}$*. Furthermore, because the bases are part of the public key, they are fixed for each run of the key generation and encryption algorithms, and thus, these computations can benefit from fixed-base exponentiations.*

Note that more efficient large-universe constructions exist. For instance, the selectively secure variant of the multi-use scheme in [5] would yield a more efficient large-universe scheme when instantiated with a full-domain hash than the considered version of the Wat11 scheme. However, the fully secure variant is, at best, secure under a q-type assumption, and would therefore be less comparable with the fully secure variant of RW13 (as it differs in another property). Similarly, the FDH-based large-universe variant of the second Wat11 scheme would be more efficient than the first Wat11 scheme, but it has a one-use restriction. On the other hand, the polynomial-based large-universe scheme in [10] seems to be more efficient than RW13, and may be more competitive with the aforementioned FDH-based schemes.

Direction 11 (Benchmarking state-of-the-art large-universe schemes) *To compare all prominent large-universe schemes with respect to their efficiency and functionality.*

8 Multi-authority ABE

In some schemes, the role of the KGA is shared by multiple authorities, which is called *multi-authority ABE* (MA-ABE) [40]. Specifically, the setup and key generation algorithms are performed (jointly or independently) by these authorities. In most schemes, each authority is responsible for its own unique set of attributes. Our definition of multi-authority ABE extends the definition of single-authority predicate encryption (for the specific case of CP-ABE), and covers the majority of MA-ABE schemes.

Definition 7 (*Multi-authority CP-ABE* [97, 131]) A multi-authority ciphertext-policy attribute-based encryption (MA-CP-ABE) scheme with users and attribute-authorities and optionally some central authority consists of the following algorithms.

- GlobalSetup(λ): The global setup is a randomized algorithm that takes as input the security parameter λ, and outputs the global parameters GP. It is either executed by the central authority, or jointly by the attribute authorities.

- AuthoritySetup(GP): The authority setup is a randomized algorithm executed by one authority. It generates the authority's master public and secret key. It publishes the master public key MPK and keeps the master secret key MSK.
- KeyGen(MSK, $\mathcal{S}$): The key generation is a randomized algorithm takes as input a set of attributes $\mathcal{S}$, the master secret key MSK of the managing authority and the global parameters GP. Upon some user's request, the authority computes a secret key $\mathrm{SK}_{\mathcal{S}}$ provided that he possesses these attributes.
- Encrypt(GP, $\{\mathrm{MPK}_i\}_i$, $\mathbb{A}$, M): The encryption is a randomized algorithm that takes as input a message M, an access structure $\mathbb{A}$ specified by the encrypting user, the associated master public keys $\{\mathrm{MPK}_i\}_i$ and the global parameters GP. It outputs a ciphertext $\mathrm{CT}_{\mathbb{A}}$.
- Decrypt(GP, $\mathrm{SK}_{\mathcal{S}}$, $\mathrm{CT}_{\mathbb{A}}$): The decryption is a deterministic algorithm that takes as input the ciphertext $\mathrm{CT}_{\mathbb{A}}$ that was encrypted under an access structure $\mathbb{A}$, the user's secret key $\mathrm{SK}_{\mathcal{S}}$ associated with a set of attributes $\mathcal{S}$, and the global parameters GP. It outputs the message only if the set of attributes associated with the keys is authorized with respect to the access structure. Otherwise, it outputs an error message.

8.1 Security against corruption

The security models in the multi-authority setting often capture, in addition to CPA-security and collusion (Sect. 4.2), the notion of *corruption*. Roughly, these models require that CPA-security is preserved with respect to the honest authorities even if some authorities are corrupted. These "additional" security guarantees are desirable or even required in multiple-domain settings in which authorities do not (necessarily) trust one another, like in the example in Sect. 2. The security model is defined as follows.

Definition 8 (*Full CPA-security against (static) corruption* [97]) Let $\mathcal{A}_1, ..., \mathcal{A}_n$ be the n attribute authorities associated with the scheme. We define the game between challenger and attacker as follows:

- **Initialization phase:** The attacker commits to a set $\mathcal{I} \subsetneq \{1, ..., n\}$ of corrupted authorities and sends it to the attacker.
- **Setup phase:** The challenger runs the setup algorithms of the MA-CP-ABE scheme and sends the global parameters, master public keys, as well as the the master secret keys for all corrupted authorities to the attacker.
- **First query phase:** The attacker queries the challenger for secret keys corresponding to sets of attributes $\mathcal{S}_1, ..., \mathcal{S}_{k_1}$, where $k_1 \in \mathbb{N}$ is polynomially bounded in the security parameter.
- **Challenge phase:** The attacker generates two messages M_0 and M_1 of equal length, together with an access structure $\mathbb{A}$ such that none of the queried sets of attributes $\mathcal{S}_i$ (in union with all corrupted attributes) satisfy $\mathbb{A}$. The challenger flips a coin, i.e., $\beta \in_R \{0, 1\}$ and encrypts M_β under $\mathbb{A}$. It sends the resulting ciphertext to the attacker.
- **Second query phase:** The attacker queries the challenger for secret keys corresponding to sets of attributes $\mathcal{S}_{k_1+1}, ..., \mathcal{S}_{k_2}$, where $k_2 \in \mathbb{N}$ is polynomially bounded in the security parameter, with the restriction that none of the sets $\mathcal{S}_i$ (in union with all corrupted attributes) satisfy access structure $\mathbb{A}$.
- **Decision phase:** The attacker outputs a guess β' for β.

The advantage of the attacker is defined as $|\Pr[\beta' = \beta] - \frac{1}{2}|$. A multi-authority ciphertext-policy attribute-based encryption scheme is fully (or adaptively) secure against *static corruption* if all polynomial-time attackers have at most a negligible advantage in this security game. The scheme is secure against *adaptive corruption* if the initialization phase can be postponed until the challenge phase.

8.2 The goal of MA-ABE

In literature, there seems to be little consensus in what constitutes secure MA-ABE and how independent the authorities must be. Furthermore, the reason that the role of the KGA is shared by multiple authorities may differ. The security requirements may therefore also differ. Some common security objectives are:

(i) To increase confidentiality: even if some authorities are corrupted, as long as some are honest, the scheme is secure. In particular, the KGAs cannot individually decrypt any ciphertexts (by using their master-key) [110].
(ii) To mitigate availability issues: even if some authorities are unavailable, users can still request keys for the desired set of attributes [106].
(iii) To increase independence of the (possibly mutually distrusting) authorities: each domain can assign its own trusted authority without requiring to trust the others; encrypting users can securely use attributes managed by one or more authorities [97].

We observe that the objective determines the level of security of the scheme and the interdependence of the authorities. For instance, if the objective is to increase confidentiality (and therefore reduce the trust in the authorities), then the authorities may be more dependent on one another in terms of availability [110]. Conversely, if the objective is to mitigate availability issues, then there may be less security against corruption [106].

8.3 Distributed and decentralized MA-ABE

We define the notions of distributed and decentralized ABE such that they address the goals about increasing confidentiality and the independence of authorities. In general, both distributed and decentralized ABE cover MA-ABE schemes that are secure against corruption. This also means that the schemes should not be centralized in terms of trust, i.e., corruption of one or more authorities should not result in the breach of security towards the other authorities (conform Sect. 8.1). If the scheme is not secure against corruption, the authorities are always and trivially dependent on one another to act honestly, and by extension, the users need to trust all authorities.

Furthermore, we make a clear but important distinction between decentralized and distributed ABE in how access control decisions are made. In particular, the distinction between the two is in whether the scheme supports (albeit indirectly) decentralized access control decisions, as formulated in Sect. 2. In ABE, this means that, for correctness, the decrypting user only needs to request keys for the attributes she possesses and only from the authorities that manage these attributes. In sum, an MA-ABE scheme is decentralized, if it supports decentralized access control decisions, is secure against corruption and does not employ a centralized authority. Therefore, if an MA-ABE scheme is decentralized, it satisfies the authority-dependence minimization (ADM) property (Sect. 2).

Note, however, that it is possible that the policies are so restricted that a decrypting user trivially needs to possess keys for all authorities associated with the access policy. Three types of access policies require this. First, the AND-gate trivially requires a decrypting user to possess all attributes, and therefore the user needs keys from all associated authorities [44]. Second, restricted access policies induced by the multi-authority setting [40]—which require the access policies to be an AND-gate over clauses, in which each clause is an MSP over attributes associated with one authority—also require the decrypting user to request keys from each authority. Third, the disjunctive normal form (DNF)—which requires the access policy to be an OR-gate over AND-gates—essentially requires the plaintext to be encrypted separately for each AND-gate [117, 128], and therefore also trivially requires the user to request keys from each authority associated with one of the AND-gates. While this is an interesting technique, it restricts the scheme significantly, because not all MSPs—and Boolean formulas consisting of AND- and OR-gates only—have an efficient DNF representation. Hence, we consider the ADM property to hold non-trivially only if the scheme efficiently supports all MSPs.

Definition 9 (*Distributed and decentralized MA-ABE*) Consider an MA-CP-ABE scheme (GlobalSetup, AuthoritySetup, KeyGen, Encrypt, Decrypt), such that it

- is secure against corruption conform Definition 8;
- does not have a fully trusted central authority.

If it supports expressive access policies (i.e., (N)MSPs) and decentralized access control decisions, it is *decentralized*. Otherwise, it is *distributed*.

Specifically, the scheme supports decentralized access control decisions (and thus satisfies the ADM property) if for all access policies $\mathbb{A}$ and all sets of attributes $\mathcal{S}$ that satisfy $\mathbb{A}$, such that

- $\mathrm{CT}_{\mathbb{A}}$ is an encryption of a plaintext M under policy $\mathbb{A}$;
- $\mathrm{SK}_{\mathcal{S}}$ is a secret key associated with the set $\mathcal{S}$, generated by only the associated authorities,

it holds that decryption of ciphertext $\mathrm{CT}_{\mathbb{A}}$ with key $\mathrm{SK}_{\mathcal{S}}$ yields the original plaintext M.

Partial and full decentralization

The interdependence of the authorities also depends on the setup, in which the global parameters and master-keys are generated. First, the setup includes the generation of one or more master-keys (from which the users' secret keys are derived), which can be generated either completely independently or distributively. If the master-key is generated distributively, then the users need to request keys from each authority, and therefore the scheme is automatically not decentralized.

Second, the setup includes the generation of the global parameters. These global parameters may be associated, albeit implicitly, with secret information on which the (MA-)security of a scheme relies. Therefore, these parameters should not be generated by a single authority if security against corruption is required. Notable examples of such parameters are a composite group order N [97], or multiple generators within the same group. Oftentimes, knowing the factorization of N or the relative discrete logarithm between two generators enables attacks on the scheme. However, the secrets associated with these parameters do not need to be kept after the global setup has finished (unlike the master-keys). It might therefore be acceptable that some originating authorities jointly and *distributively* generate these parameters in a secured environment, and afterwards, the associated secret data are destroyed. It is then assumed that these authorities cannot retrieve the secret data later. Because these parameters cannot be generated independently, a scheme cannot be fully decentralized. As such, we call a decentralized scheme that generates global parameters associated with secret information *partially* decentralized. Otherwise, it is *fully* decentralized.

Decentralization to foster availability

Interestingly, decentralized ABE provides some additional security with respect to the availability of the authorities. We already covered the notion of corruption in the security model (Sect. 8.1), which ensures that security in the form of confidentiality is still preserved with respect to the honest authorities if some authorities are corrupted. In practice, such corruption may also cause issues with respect to the availability of the authorities [117]. If authorities are suddenly unavailable, they cannot issue keys anymore, which may break the entire system. That is, users entering the system after such corruption may not be able to decrypt any old ciphertexts despite being authorized to do so. Because corruption may be easier to instigate, e.g., through denial-of-service attacks, than the retrieval of an authority's secret keys, security against this type of attack is at least as relevant. Similarly, the availability of authorities may depend on whether they want to leave the system voluntarily, or alternatively, new authorities may want to join.

8.4 Achieving security against corruption: distributed versus decentralized

We briefly review the ways in which security against corruption is achieved in existing schemes. As we have seen in the standard form (Definition 6), a scheme consists of various variables that need to be secret. Notably, the master-key α can be used to decrypt any ciphertext, and knowing some common variables $\mathbf{b}$ may also lead to significant breaks [142]. Most schemes ensure security against corruption by letting each authority hold its own (partial) master-key and its own common variables $\mathbf{b}$ (whereas other schemes distribute the generation of these variables). Furthermore, the user-specific random variables $\mathbf{r}$ introduced in the key generation link the keys to one specific user. It is paramount that this randomness is unique to avert any collusion attacks. In addition, a scheme oftentimes breaks if the randomness is known, so it should not be retrievable by any attackers (which may be a corrupt authority). Hence, to ensure security against corruption, this randomness is either generated distributively by all authorities [41] or provided by a full-domain hash [97].

In general, achieving decentralized ABE is more difficult than achieving distributed ABE. In particular, one of the difficulties is in making the scheme collusion resistant. In centralized schemes, encryption uses only one master-key, while in distributed and decentralized schemes, encryption may need to use master-keys managed by several authorities. Roughly, the part that hides the message, i.e., $e(g, h)^{\alpha s}$, in Definition 6 is replaced by $e(g, h)^{\alpha_i s}$, where α_i is the master-key of the i-th authority. Then, colluding users should not be able to jointly decrypt by individually computing these "randomized master-keys', therefore partially decrypting the ciphertext. Chase and Chow [40, 41, 54] ensure this by generating the system-wide master-key distributively, and they compute a user's secret key by generating a different sharing of the master-key for each user. As such, a decrypting user only obtains the randomized system-wide master-key, i.e., $e(g, h)^{\alpha s} = \prod_i e(g, h)^{\alpha_i s}$, if her own secret keys are used. However, recall that distributing the master-key also ensures that the ADM property cannot hold, because users trivially need to request keys from each authority.

To overcome this restriction, Lewko and Waters [97] use a different method to tie the partially decrypted ciphertexts to one user. Specifically, they include a zero-sharing associated with the access policy in the ciphertext, which can only be canceled, if it is combined with keys that use the same user randomness during decryption. Because this user randomness needs to be the same for each authority and authorities are not supposed to interact with one another, it is implicitly provided by a full-domain hash. As a trade-off, these schemes [97, 131] typically have much larger ciphertexts, because a linear number of elements exists in group $\mathbb{G}_T$. In addition, arithmetic in group $\mathbb{G}_T$ is less efficient, and the use of FDHs may further reduce the efficiency of a scheme (Sect. 7). In contrast, schemes that distribute the master-key like Chase and Chow [40, 41, 54] are structurally often closer to the single-authority version of the scheme, and consequently retain a similar efficiency. Importantly, because all decentralized schemes require a full-domain hash for its user-specific randomness, they cannot benefit from an anonymous key issuance protocol (Sect. 8.5). Removing the full-domain hash may mitigate or even solve some of these issues. As such, we formulate the following directions.

Direction 12 (Decentralized ABE conform standard form) *To devise decentralized ABE that is, in structure, closer to single-authority ABE, i.e., conform the standard form (Definition 5.4), and thus attains a similar efficiency.*

Direction 13 (Decentralized ABE without full-domain hash) *To design a decentralized ABE scheme without requiring full-domain hashes.*

8.5 Authority unlinkability

In MA-ABE, secret keys are requested from different authorities, which opens up the possibility of a linkability attack [58]. While this attack does not necessarily pertain to the confidentiality of the *encrypted data*, it might affect the privacy of the requesting user. An inherent feature of ABE is that a user is associated with a set of attributes, which possibly reveals much information about this user. It might be undesirable [118, 139] or prohibited [120] that certain attributes managed by different authorities are linked together, possibly leading to the identification of users or to other privacy violations. For example, consider a setting in which authorities consist of the government, a university, and a hospital. One user has obtained the attributes “PhD student” and “computer science” from the university. In itself, these attributes might not say much. However, if they are combined with the user’s attribute “female”, which she acquired from the government, this might simplify the identification of the user significantly, especially in some given context. If this particular user is discovered to have requested keys associated with attributes pertaining to her health from the hospital, this results in a privacy violation.

Such *authority-linkability* attacks can be prevented in schemes of a certain form. Chase and Chow [41] devise a privacy-friendly key issuance protocol that allows users to request keys without risking such privacy violations. Their technique can be applied to MA-ABE schemes that satisfy the following two properties [54]. First, the key generation needs to be executable such that no communication with other authorities is required. Second, the keys need to be linked to a user in a certain way. As mentioned in Sect. 8.4, the user-specific randomness is either generated distributively by the authorities [41] or provided by a full-domain hash [97]. If the randomness is generated distributively, the Chase-Chow approach can be used to make the key issuance anonymous. To our knowledge, no secure anonymous key issuance protocols exist for schemes that use a full-domain hash for its randomness yet.

Regardless, as we will show in Sect. 8.6, the only existing decentralized schemes require FDH-based randomness. To give decentralized schemes the same advantages as other (distributed) MA-ABE schemes with respect to unlinkability, an anonymous key issuance protocol with the same security requirements as the Chase-Chow protocol should be devised.

Direction 14 (Anonymous key issuance protocol for FDH-MA-ABE) *To devise an anonymous key issuance protocol for MA-ABE schemes in which the user-specific randomness is provided by an full-domain hash.*

Table 2 Taxonomy of multi-autority ABE

Scheme	KP/CP	MSP	MA-sec	GP	NT-ADM	Dec	No FDH	Unlink
Cha07 [40]	KP	✓	$\checkmark_A$	G	✗	✗	✓	✗
CC09 [41]	KP	✓	✓	-	✗	✗	✓	✓
LHCC+11 [105]	KP	✗	✓	-	✗	✗	✓	✓
LW11 I [97]	CP	✓	✓	N	✓	◑	✗	✗
LW11 II [96]	CP	✓	✓	-	✓	●	✗	✗
LCHWY11 [110]	CP	✓	✓	N	✗	✗	✓	✗
OT13 [123]	CP	✓	✓	G	✓	◑	✗	✗
RW15 [131]	CP	✓	✓	-	✓	●	✗	✗
MJ18 [115]	CP	✗	✓	G	✗	✗	✗	✗

We list whether the scheme is key-policy (KP) or ciphertext-policy (CP) based, whether they support MSPs, whether they are proven to be secure against corruption (MA-sec), whether the identity is embedded into the keys with a full-domain hash (FDH) or not, whether the scheme is authority unlinkable (unlink) and decentralized (dec), and whether it satisfies the ADM property non-trivially (NT-ADM). For the global parameters (GP), we consider whether the scheme requires the generation of a composite order N or multiple generators (G). $\checkmark_A$ = only for the attribute authorities; ◑ = partially, ● = fully; N = composite order; G = multiple generators

8.6 Taxonomy of MA-ABE schemes

Throughout this section, we have analyzed several MA-ABE schemes. We provide a taxonomy for multi-authority ABE in which we evaluate the properties specific to the multi-authority setting. Table 2 lists these results. Note that the table lists much fewer properties than the taxonomy in Sect. 11. For an evaluation of the other properties, we refer to Table 3 in Sect. 11.

Table 2 shows that only two fully and two partially decentralized schemes with expressive access structures exist. They all employ full-domain hashes to generate the secret randomness for each user. As mentioned, no techniques exist to support an anonymous key issuance protocol for FDH-based MA-ABE, meaning that currently no decentralized schemes achieving authority unlinkability exist. Nevertheless, these four (partially) decentralized ABE schemes satisfy the user independence and authority-dependence minimization properties, and therefore provide practical solutions to enforcing access control.

9 Towards (formalizing) resilient ABE

We discuss the resilience of ABE. Many works on ABE have considered the notion of security in the form of confidentiality or integrity, covered by CPA- and CCA-security. However, the notion of availability is also important in practice. We define the notions of attribute resilience and attribute-wise key generation. These two properties capture a level of resilience of a scheme such that availability issues can be mitigated by minimizing the involvement of the authorities.

9.1 Attribute resilience

In practice, the universe of attributes may not be static. For instance, as new users join the system, new attributes (values) for e.g., names may have to be added [85]. In this case, small-universe constructions require that a new public key is generated for each new attribute. This may also impact the previously generated keys and ciphertexts. For example, some schemes associate the keys and ciphertexts with the whole universe, e.g., [52, 119]. Keys generated before the addition of an attribute and its associated public key may not be able to decrypt a newly encrypted message despite its authorized status. As such, all keys and ciphertexts need to be updated to make the system functional again. Large-universe constructions do not have this problem, because the public keys can be generated from any attribute string, and keys and ciphertexts are therefore never associated with the whole universe.

Example 2 The KP-ABE schemes that "implement" CP-ABE [52, 105, 119] mentioned in Observation 1 break when attributes are added to the universe. This is because the keys and ciphertexts are generated with respect to the current universe of attributes. The keys—which are explicitly associated with $\mathcal{S}$—are implicitly associated with the access policy $\bigwedge_{\text{att}\in\mathcal{U}}(\underline{\text{att}} \vee \text{att}^*)$ such that $\underline{\text{att}} = \text{att}$ if $\text{att} \in \mathcal{S}$, and $\underline{\text{att}} = \neg\text{att}$ if $\text{att} \notin \mathcal{S}$, and att^* denotes the dummy value for attribute att. The ciphertexts are explicitly associated with the "access policy" $\bigwedge_{\text{att}\in\mathcal{U}} \underline{\text{att}}$ such that $\underline{\text{att}} \in \{\text{att}, \neg\text{att}, \text{att}^*\}$ specifies whether the encrypting user requires the presence (att) or absence ($\neg$att) of an attribute, or does not care (att^*). If a ciphertext and authorized key are associated with different universes, then some problems arise, either with respect to correctness or security. That is, suppose $\mathcal{U}$ is the universe associated with the key, and $\mathcal{U}'$ the universe associated with the ciphertext. If $\mathcal{U}' \subsetneq \mathcal{U}$, then the implicit ciphertext set never satisfies the implicit key policy. It is therefore incorrect. If $\mathcal{U} \subsetneq \mathcal{U}'$, then the scheme might not be secure. That is, for the attributes in $\mathcal{U}' \setminus \mathcal{U}$, the secret key does not have any values, so a ciphertext that requires the presence of an attribute in $\mathcal{U}' \setminus \mathcal{U}$ may still be decryptable by the key even though it may not be authorized to do so.

More generally, we propose the notion of *attribute resilience*, which protects against such problems. It captures the correctness and security of a scheme with respect to the universe of attributes at any given time. That is, at a certain time, the universe of attributes may be larger than at an earlier point in time. In many settings, it is reasonable to require that e.g., ciphertexts encrypted at the earlier point in time are decryptable by a key generated at a later point, provided that it is authorized to do so. However, certain types of schemes are by definition not resilient, and can therefore not be efficiently used in such dynamic settings.

The definition of attribute resilience can be expressed as an extra condition on the correctness and security of the scheme. On the one hand, a ciphertext should be decryptable by an authorized key, regardless of the associated universes of attributes at the time of generation. On the other hand, a ciphertext should *never* be decryptable by an unauthorized key either. Such an additional requirement on the definition, however, requires that the formal definition and security model of the scheme need to be adjusted, e.g., because the universe needs to be taken as input to the algorithms. We formalize attribute resilience as follows.

Definition 10 (*Attribute-resilient predicate encryption*) A predicate encryption scheme for a predicate $P\colon \mathcal{X} \times \mathcal{Y} \to \{0, 1\}$, with some KGA and users consists of four algorithms:

- Setup(λ): On input the security parameter λ, this randomized algorithm, executed by the KGA, generates the domain parameters, the master public key MPK and the master secret key MSK. The master public key also contains a description of the attribute universe $\mathcal{U}$.
- UpdateUniverse(MPK, $\mathcal{U}, \mathcal{U}'$): This algorithm, executed by the KGA, updates the attribute universe $\mathcal{U}$ to a new attribute universe $\mathcal{U}'$ (such that $\mathcal{U} \subseteq \mathcal{U}'$), and adjusts the master public key accordingly to MPK$'$.
- KeyGen(MSK, $\mathcal{U}$, y): On input the master secret key MSK, universe $\mathcal{U}$, and some $y \in \mathcal{Y}$, this randomized algorithm, executed by the KGA, generates a secret key $\text{SK}_{\mathcal{U},y}$.

- Encrypt(MPK, $\mathcal{U}$, x, M): On input the master public key MPK, universe $\mathcal{U}$, some $x \in \mathcal{X}$ and message M, this randomized algorithm, executed by the encrypting user, generates a ciphertext $\mathrm{CT}_{\mathcal{U},x}$.
- Decrypt(MPK, $\mathrm{SK}_{\mathcal{U},y}$, $\mathrm{CT}_{\mathcal{U}',x}$): On input the master public key MPK, the secret key $\mathrm{SK}_{\mathcal{U},y}$ and its associated universe $\mathcal{U}$, and the ciphertext $\mathrm{CT}_{\mathcal{U}',x}$ and its associated universe $\mathcal{U}'$, if $P(x, y) = 1$, then it returns M. Otherwise, it returns an error message.

A scheme is called *correct* if decryption of a ciphertext with an authorized key (which holds if $P(x, y) = 1$) succeeds with overwhelmingly high probability, regardless of the associated universes $\mathcal{U}$ and $\mathcal{U}'$.

The associated security model also takes into account that the attacker can update the universe, and is defined as follows.

Definition 11 (*Full CPA-security of attribute-resilient predicate encryption*) We define the security game between challenger and attacker as follows:

- **Setup phase:** The challenger runs Setup(λ) to obtain MPK and MSK, and sends the master public key MPK to the attacker.
- **First query phase:** The attacker can make two types of queries:
  - The attacker queries secret keys for $y \in \mathcal{Y}$, and obtains $\mathrm{SK}_{\mathcal{U},y} \leftarrow \mathrm{KeyGen}(\mathrm{MSK}, \mathcal{U}, y)$ in response, where $\mathcal{U}$ denotes the current universe.
  - The attacker queries an update of the universe $\mathcal{U}$ to $\mathcal{U}'$ (with $\mathcal{U} \subseteq \mathcal{U}'$). The challenger returns the new master public key output by UpdateUniverse(MPK, $\mathcal{U}$, $\mathcal{U}'$).
- **Challenge phase:** The attacker specifies some $x^* \in \mathcal{X}$ such that for all y in the first key query phase, we have $P(x^*, y) = 0$, and generates two messages M_0 and M_1 of equal length in $\mathcal{M}_\lambda$, and sends these to the challenger. The challenger flips a coin, i.e., $\beta \in_R \{0, 1\}$, encrypts M_β under x^*, i.e., $\mathrm{CT}_{x^*} \leftarrow \mathrm{Encrypt}(\mathrm{MPK}, \mathcal{U}, x^*, M_\beta)$, where $\mathcal{U}$ denotes the current universe, 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 keys for $y \in \mathcal{Y}$ such that $P(x^*, y) = 0$.
- **Decision phase:** The attacker outputs a guess β' for β.

The advantage of the attacker is defined as $|\Pr[\beta' = \beta] - \frac{1}{2}|$. A scheme is fully secure and attribute resilient if all polynomial-time attackers have at most a negligible advantage in this security game.

9.2 On the resilience of ABE supporting non-monotonicity

We observe that ABE that supports non-monotonicity has issues with regard to the attribute resilience. To show this, we discuss the non-resilience that follows from applying the three methods described in Sect. 5.7.

The first method: negative attributes

Schemes supporting non-monotonicity using the first method as considered in Sect. 5.7 are rendered incorrect when attributes are added. Like in Example 2, suppose that $\mathcal{U}$ and $\mathcal{U}'$ are the universes associated with some key and ciphertext, respectively. Let the access policy of the ciphertext be such that it requires the possession of a negative attribute in $\mathcal{U}' \setminus \mathcal{U}$. Then, the key cannot decrypt the ciphertext, despite possibly not having the attribute and thus satisfying the policy.

The second method: OSW-method

Schemes supporting non-monotonicity using the second method, i.e., the OSW-method, as considered in Sect. 5.7 are rendered insecure when attributes are added. The reason for this is that the OSW-method is combined with the polynomial-based method to support large universes (Sect. 5.5). On the one hand, this allows users to use any conceivable input string as attribute. On the other hand, this lack of control over the universe may yield insecurity in the non-monotonic setting. As we previously discussed, the negations used in the OSW-method require the decrypting user to compare the entire set of attributes associated with the key with the negated attribute. However, the decrypting user could still possess the negated attribute, despite not having the associated key. It is therefore possible that a decrypting user can decrypt a ciphertext despite not being authorized to do so, and the scheme is consequently insecure. (Note that this issue was also pointed out by Attrapadung and Tomida [22], who introduced the notion of labeled non-monotonicity to mitigate such issues.)

The third method: labeled non-monotonicity

Schemes supporting a labeled non-monotonicity with the third method as considered in Sect. 5.7 may provide a more acceptable trade-off in (non-)resilience. As discussed, it is useful that ciphertexts for which the access policy specifies a negation for a label, for which the decrypting user does not have a key yet, cannot be decrypted. This ensures that the owner of the key cannot unjustifiably decrypt a ciphertext despite possessing the negated attribute (value) but not possessing the associated key. In this case, it is reasonable to say that the incorrectness that follows from this lack of attribute(-label) resilience trumps the insecurity that might have otherwise followed. The lack of resilience with respect to the correctness is less disruptive in the third method than in the first method. Whereas in the first method, this non-resilience can occur at any addition of an attribute (value), in the third method, this only happens when an attribute label is added.

9.3 Attribute-wise key generation

To minimize the required computational power of the KGA, we introduce the notion of *attribute-wise key generation*. An additional benefit of decentralized CP-ABE [97, 131] is that the user's secret keys can be generated incrementally, i.e., one partial secret key for each attribute. In contrast, existing single-authority schemes require users to request secret keys for the entire set of attributes they possess. The keys are mathematically linked to a single user by using the same user-specific randomness for each attribute such that users cannot collude. In most decentralized schemes, this randomness is provided by a full-domain hash, which deterministically and implicitly generates randomness for any given identity. It therefore allows the KGA to link the new "attribute keys" incrementally to the identity instead of all at once.

In general, we identify two advantages of such an "attribute-wise key generation" with respect to efficiency, practicality and privacy-friendliness. First, we consider attribute-wise key generation to be more efficient and practical. Especially in dynamic settings in which new attribute values, labels or types may be frequently added, attribute-wise key generation may be more efficient. For instance, suppose a recently added attribute (label or value) is used in a ciphertext access policy, and a decrypting user possesses the attribute but not the associated secret key. Then, a partial secret key can be requested for the new attribute (value) only. This reduces computational costs for the key generation authority significantly. By extension, key requests can be handled more quickly, and thus more key requests can be processed, which fosters availability. Second, attribute-wise key generation provides the first step towards a privacy-friendly key generation [53]. Similar to authority unlinkability (Sect. 8.5), the notion of *attribute unlinkability* can be defined, which ensures that an authority cannot link two separately requested attribute-keys to one user. Like in the multi-authority setting, the ability to link some attributes to one user may reveal sensitive information about this user, even towards a single authority. A privacy-preserving key issuance protocol mitigates this problem, though we leave the construction of such a protocol for ABE for future work:

Direction 15 (Privacy-preserving key generation) *To devise a secure and privacy-preserving key generation protocol that provides attribute unlinkability.*

In general, the construction of a scheme with a attribute-wise key generation is an open problem. Some CP-ABE schemes seem to have a structure that allows for the construction of such an attribute-wise key generation by using an FDH like in the decentralized setting. Alternatively, in the single-authority setting, the user-specific random that ties the keys together can be provided by a hash that maps arbitrary strings to the set of integers $\mathbb{Z}_p$. However, the main difficulty in constructing such a scheme lies in the security proof. The existing security models consider key queries for sets of attributes (which are thus specified before the key is generated). In contrast, the security model associated with a scheme that supports attribute-wise key generation would have to take into account that keys can be queried gradually for any user and attribute. For instance, selective security proofs (e.g., [130, 145]) embed the entire set of attributes in all key components, so it is unlikely that these proofs carry over to any such new security model without some adjustments. To solve this, Rouselakis and Waters [131] use a static security model. In this model, the sets of attributes for which the attacker is going to query keys are determined during the initialization phase (Definition 4). It is unclear, however, if such a scheme can be designed without resorting to weaker models.

Direction 16 (Attribute-wise key generation) *To construct a scheme with attribute-wise key generation that is secure in a non-static model.*

Observation 8 (Non-monotonicity and attribute-wise key generation) *For some methods that are used to support non-monotonicity, attribute-wise key generation is impossible. Intuitively, this follows from the converse of the definition of monotonicity (Definition 2). If an access structure $\mathbb{A}$ is not monotone, then there exist B, C such that $B \in \mathbb{A}$ and $B \subseteq C$, but not $C \in \mathbb{A}$. Suppose now that $\mathbb{A}$ is used during encryption. Then, some user, who possesses a set of attributes C, but only has secret keys for the subset B (because she has not requested keys for the rest of the attributes $C \setminus B$ yet), can decrypt the ciphertext even though she does not satisfy the policy.*

Fortunately, not all three methods discussed in Sect. 5.7 satisfy this definition of non-monotonicity. The first method essentially uses monotone structures by pushing the negation to the value, e.g., "profession: non-doctor" instead of "NOT profession: doctor". The second method, on the other hand, does satisfy this definition of non-monotonicity. We also observe that all schemes that use the OSW-method tie all "attribute keys" together for one user by distributing the user randomness in a certain way. The KGA can therefore not generate these keys independently of the rest of the attribute keys, and we can thus not define an attribute-wise key generation for these schemes. Because the third method uses a labeled non-monotonicity, e.g., "profession: NOT doctor", it is possible to incrementally generate keys for the labels, but not for the attributes values with the same label. This is however only secure, if the KGA *can check whether the user requests keys for all values that she possesses for the label for which she requests keys.*

10 Additional functionality

Some schemes that we have analyzed support some additional functionality, achieving more properties than we have discussed so far. These properties may enhance ABE in certain aspects, which may make them more practical. We briefly introduce these properties below, but leave any systematization of these works for future work. Furthermore, note that the number of properties that we consider in this section may not be exhaustive with respect to the entire realm of ABE.

10.1 Online/offline key generation and encryption

Unbounded ABE schemes [101, 130] based on the polynomial method discussed in Sect. 5.5 can be implemented more efficiently. The key generation and encryption algorithms can be efficiently split in an online and offline phase [84], such that the offline phase covers most of the computational costs, while the online phase requires minimal computations. (In contrast, key generation and encryption of FDH-based large-universe schemes could also be split in two phases, but would always require a linear number of exponentiations during online time.) The key generation efficiency is enhanced, because the KGA can securely precompute large batches of secret key material, and needs to perform only few computations in a key request. As a small trade-off, the user needs to put in more computational effort. The encryption efficiency is similarly enhanced, but at the cost of a diminished decryption efficiency. Furthermore, the ciphertexts increase significantly in size. This implementation of encryption is however especially useful for e.g., devices with limited power resources.

10.2 Revocation

In access control systems, the access privileges of users may be revoked, e.g., because their credentials have been stolen or have expired. A user's secret keys then need to be disabled such that she cannot decrypt any ciphertexts for which she has lost authorization. For ABE, revocation is significantly more troublesome, because—unlike in traditional public-key encryption—several distinct users might hold secret keys associated with the same attributes [126]. Many works have addressed revocation by means of different approaches, which can be classified in two categories: user-based [19, 54, 55, 86, 95, 149] and time-based [18, 109, 132]. Some of these works describe generic approaches to supporting revocation [54, 132]. Because of the importance of revocation in practice and the vast body of (non-generic) work in this subfield of ABE, we recommend that any such approaches are systematized and generalized. Then, these methods can be applied to any scheme, generically, rather than to a single scheme.

Direction 17 (Systematizing and generalizing revocation methods) *To systematize and generalize revocation methods such that they can be applied generically to many or even all ABE schemes.*

10.3 Hidden policies—attribute-hiding ABE

Some schemes are designed such that the access policies can be securely hidden, which is called *attribute-hiding* ABE [34, 88, 119]. Because the predicate (e.g., access policy) associated with a ciphertext may leak information about the sender or receiver, this makes ABE more privacy friendly. Interestingly, the generic framework by Chen, Gay and Wee (CGW15) [45] provides a modular approach to achieving a weaker notion of attribute-hiding called *weakly* attribute-hiding. In this notion of attribute-hiding, the decrypting user only does not learn anything about the access policy, if it is not satisfied by the set of attributes. In contrast, in fully attribute-hiding schemes, the decrypting does not learn anything about the access policy except whether the set of attributes satisfies it, even if the set does satisfy the policy.

10.4 Outsourced decryption

To mitigate the decryption costs on the user side, decryption can be securely outsourced to a third party with better computational resources [75]. The approach was later generalized in [54], and may thus be applicable to various existing schemes. This property can also be combined with server-aided revocation techniques [55]. While outsourced decryption may solve some issues with the decryption efficiency in practice, we believe that this particular property may somewhat undermine the user independence property—which mitigates availability issues in access control systems—and is one of the main advantages of ABE. As such, it should be carefully considered whether the entity assigned for outsourced decryption is sufficiently reliable in terms of availability.

10.5 Traceability

Depending on the setting, users may attempt to share their access privileges with other users (e.g., for financial gain). This can be mitigated with traitor-tracing schemes [105], and allows the authorities to trace these malicious users. Notably, Lai and Tang [93] devised a practical generic approach that can be applied to any existing scheme.

11 Taxonomy: classifying existing schemes

Throughout this work, we have analyzed over fifty ABE schemes with respect to the described properties. These schemes and our analysis are summarized in Table 3. Each property that we have discussed in Sects. 3–8 and 10 is considered in the table. We illustrate the significance of each scheme by highlighting the satisfied properties in green. Note that the table does not include schemes that can be instantiated in the works mentioned in Sect. 6.2, as the underlying choices influence several properties, such as the group order and the level of security.

12 Conclusion

We showed that over the past two decades, much progress has been made in the context of pairing-based ABE. Especially with respect to security, a wide variety of techniques and frameworks has been developed to construct ABE with the required security guarantees. Selectively secure schemes can now be converted into fully secure schemes, incurring relatively little sacrifice in efficiency. Furthermore, such security guarantees can be achieved whilst retaining or achieving all desirable core properties. From a practical viewpoint, we are therefore optimistic that the existing techniques and frameworks are sufficient in the construc-

Table 3 Taxonomy: classification of existing schemes

Scheme	Based on	KP/CP	Expr.	LU	LU-FDH	Bounded	Prime	Type	CS-CT	Model	CCA*	Assump.	ROM	MA-ABE	Add. func.
SW05 [133] I	–	KP	T	✗	✗	-	✓	I	✗	◑	$\checkmark_V$	Static	✗	✗	–
SW05 [133] II	–	KP	T	✓	-	$\|\mathcal{S}\|$	✓	I	✗	◑	$\checkmark_{D,V}$	Static	✗	✗	–
GPSW06 [72] I	[133] I	KP	MSP	✗	✗	-	✓	I	✗	◑	$\checkmark_V$	Static	✗	✗	–
GPSW06 [72] II	[133] II	KP	MSP	✓	-	$\|\mathcal{S}\|$	✓	I	✗	◑	$\checkmark_{D,V}$	Static	✗	✗	–
Cha07 [40] I	[133] I	KP	MA-T	✗	✗	-	✓	I	✗	◑	$\checkmark_V$	Static	✗	◑	–
Cha07 [40] II	[72] II	KP	MA-MSP	✓	-	$\|\mathcal{S}\|$	✓	I	✗	◑	$\checkmark_{D,V}$	Static	✗	◑	-
BSW07 [29]	[72] II	CP	MSP	✓	-	-	✓	I	✗	◑	$\checkmark_{D,V}$	GGM	✓	✗	–
CN07 [52]	[72] I	KP*	AND	✗	✗	-	✓	I	✗	◑	✓	Static	✗	✗	–
OSW07 [124]	[72] II	KP	NMSP	✓	-	$\|\mathcal{S}\|$	✓	I	✗	◑	$\checkmark_V$	Static	✗	✗	–
NYO08 [119] I	[52] I	KP*	AND	✗	✗	-	✓	I	✗	◑	$\checkmark_V$	Static	✗	✗	A-H
NYO08 [119] II	[29]	CP	AND	✗	✗	-	✓	II	✗	◑	$\checkmark_{D,V}$	GGM	✗	✗	A-H
CC09 [41]	[40]	KP	MA-T	✗	✗	-	✓	I	✗	◑	$\checkmark_V$	$q(\|\mathcal{A}\|)$	✗	◑	–
YWRL10 [149]	[52] I	KP*	AND	✗	✗	-	✓	I	✗	◑	✓	Static	✗	✗	Rev.
HLR10 [81]	-	CP	T	✗	✗	-	✓	I	✓	◑	✗	$q(\|\mathcal{U}\|)$	✗	✗	–
LOSTW10 [99]	[144]	CP	MSP	✗	✓	OU	✗	I	✗	●	$\checkmark_{D,V}$	Static	✗	✗	–
OT10 [121]	[99]	KP,CP	NMSP	✓	–	OU,$\|\mathbb{A}\|$	✓	III	✗	●	✓	Static	✗	✗	–

Table 3 continued

Scheme	Based on	KP/CP	Expr.	LU	LU-FDH	Bounded	Prime	Type	CS-CT	Model	CCA*	Assump.	ROM	MA-ABE	Add. func.
Wat11 [145] I	[99]	CP	MSP	✗	✓	-	✓	I	✗	◑	$\checkmark_{D,V}$	$q(\lvert\mathbb{A}\rvert)$	✗	✗	–
Wat11 [145] II	[99]	CP	MSP	✗	✓	OU	✓	I	✗	◑	$\checkmark_{D,V}$	q(AND)	✗	✗	–
Wat11 [145] III	[99]	CP	MSP	✗	✓	OU	✓	I	✗	◑	$\checkmark_{D,V}$	Static	✗	✗	–
LHCC+11 [105]	[41, 119]	KP*	AND	✗	✗	-	✓	II	✗	◑	$\checkmark_{V}$	$q(\lvert\mathcal{A}\rvert)$	✗	◑	A-H, trace.
ALP11 [21]	-	KP	(N)MSP	✓	-	$\lvert\mathcal{S}\rvert$	✓	I	✓	◑	$\checkmark_{D,V}$	$q(\lvert\mathcal{S}\rvert)$	✗	✗	–
LW11 [97] I	[99]	CP	MSP	✗	✗	OU	✗	I	✗	●	✗	Static	✓	●	–
LW11 [97] II	[99]	CP	MSP	✗	✗	OU	✓	I	✗	◑	✗	GGM	✓	●	–
LW11b [101]	[99]	KP	MSP	✓	-	-	✗	I	✗	◑	$\checkmark_{D,V}$	Static	✗	✗	–
GHW11 [75] I	[145]	CP	MSP	✓	-	-	✓	I	✗	◑	RCCA	$q(\lvert\mathbb{A}\rvert)$	✓	✗	Out. dec.
GHW11 [75] II	[72]	KP	MSP	✓	-	-	✓	I	✗	◑	RCCA	Static	✓	✗	Out. dec.
LCHWY11 [110]	[99]	CP	MSP	✗	✗	-	✗	I	✗	●	$\checkmark_{D,V}$	Static	✗	◑	–
LW12 [102]	[99]	CP	MSP	✗	✓	-	✗	I	✗	●	$\checkmark_{D,V}$	q(AND)	✗	✗	–
SSW12 [132]	[99]	KP,CP	MSP	✗	✓	OU	✗	I	✗	●	✗	Static	✗	✗	Rev.
OT12 [122]	[121]	KP,CP	NMSP	✓	-	-	✓	III	✗	●	$\checkmark_{V}$	Static	✗	✗	–
HW13 [83]	[72] II	KP	MSP	✗	✗	-	✓	I	✗	◑	$\checkmark_{D,V}$	$q(\lvert\mathcal{U}\rvert)$	✗	✗	–
CCLZ+13 [43]	[70]	KP	T	✓	–	$\lvert\mathcal{S}\rvert$	✗	I	✓	●	✗	Static	✗	✗	–
OT13 [123]	[121]	KP,CP	NMSP	✓	–	$\lvert\mathbb{A}\rvert$	✓	III	✗	●	$\checkmark_{D,V}$	Static	✓	●	–
LCLJ+13 [104]	[133]	KP	T	✗	✓	-	✓	I	✗	◑	$\checkmark_{D,V}$	Static	✗	✗	Rev., out. dec.
RW13 [130]	[101]	KP,CP	MSP	✓	-	-	✓	I	✗	◑	$\checkmark_{D,V}$	$q(\lvert\mathbb{A}\rvert)$	✗	✗	–
LCW13 [111]	[69, 102]	CP	MSP	✗	✓	-	✗	I	✗	●	$\checkmark_{D,V}$	$q(\lvert\mathbb{A}\rvert)$	✗	✗	Trace.
HW14 [84]	[130]	KP,CP	MSP	✓	–	-	✓	I	✗	◑	$\checkmark_{D,V}$	$q(\lvert\mathbb{A}\rvert)$	✗	✗	O/O
KL15 [91]	[99]	KP	MSP	✗	✗	-	✗	I	✗	●	$\checkmark_{D,V}$	Static	✗	✗	–
RW15 [131]	[97]	CP	MSP	✓	-	-	✓	I	✗	◔	✗	$q(\lvert\mathbb{A}\rvert)$	✓	●	–
CGW15 [45]	[145] II	KP,CP	MSP	✗	✓	OU,$\lvert\mathbb{A}\rvert$	✓	III	✗	●	$\checkmark_{D,V}$	Static	✗	✗	A-H
LW15 [112]	[111, 130]	CP	MSP	✓	-	-	✓	I	✗	◑	$\checkmark_{D,V}$	$q(\lvert\mathbb{A}\rvert)$	✗	✗	Rev., trace.

Table 3 continued

Scheme	Based on	KP/CP	Expr.	LU	LU-FDH	Bounded	Prime	Type	CS-CT	Model	CCA*	Assump.	ROM	MA-ABE	Add. func.
ZGTC+16 [151] I	[50, 140]	KP	MSP	✗	✗	-	✓	III	✓	◕	$\checkmark_{D,V}$	Static	✗	✗	–
ZGTC+16 [151] II	[50, 140]	KP	MSP	✓		$\|\mathcal{S}\|$	✓	III	✓	◕	$\checkmark_{D,V}$	Static	✗	✗	-
AHMTY16 [16]	[13, 101]	KP	MSP	✓	-	-	✗	III	✓	●	$\checkmark_{D,V}$	$q(\|\mathbb{A}\|)$	✗	✗	–
CDLQ16 [55]	[130]	CP	MSP	✓	–	-	✓	I	✗	◑	$\checkmark_{D,V}$	$q(\|\mathbb{A}\|)$	✗	✗	Rev., out. dec.
AC17b [4]	[45]	KP,CP	MSP	✓	-	OU	✓	III	✗	●	$\checkmark_{D,V}$	Static	✓	✗	–
CGKW18 [47]	[13, 130]	KP,CP	MSP	✓	–	OU	✓	III	✗	●	$\checkmark_{D,V}$	Static	✗	✗	–
LYZL18 [109]	[33, 145]	CP	MSP	✗	✓	OU	✓	I	✗	◑	$\checkmark_{D,V}$	q(AND)	✗	✗	Rev.
MJ18 [115]	[45]	CP	T	✗	✗	-	✓	III	✗	●	✗	Static	✓	◑	A-H
KW19 [92] I,II	[102]	KP,CP	BF	✗	✗	-	✓	III	✗	●	$\checkmark_{D,V}$	Static	✗	✗	–
KW19 [92] III	[101]	KP	BF	✓	–	-	✓	III	✗	●	$\checkmark_{D,V}$	Static	✗	✗	–
TKN20 [141]	[4, 92]	KP,CP	NM-BF	✓	–	-	✓	III	✗	●	$\checkmark_{D,V}$	Static	✓	✗	–

Core properties
KP = key-policy, CP = ciphertext-policy, KP* = KP implemented as CP expr = expressivity, LU = large-universe,
LU-FDH = extendable to LU with FDH (only evaluated if not already supported), T = threshold function; (NM-)BF = (non-monotone) Boolean formulas;
(N)MSP = (non-)monotone span program; OU = one-use restriction, $|\mathcal{U}|$, $|\mathbb{A}|$, $|\mathcal{S}|$ = the sizes of the universe, the access policy, the set of attributes For boundedness, we consider OU, $|\mathbb{A}|$ and $|\mathcal{S}|$

Efficiency
CS-CT = constant-size ciphertext

Security
CCA* = security against CCA with the generic constructions using verifiability (V) and delegatability (D); assump = security assumption,
ROM = random oracle model, q(par) = q-type assumption depends on par; GGM = generic group model; ● = full, ◕ = semi-adaptive, ◑ = selective, ◔ = static security;

Additional functionality
MA-ABE = multi-authority ABE, A-H = attribute-hiding, out. dec. = outsourced decryption, trace. = traceability, rev. = revocation,
O/O = online/offline ● = decentralized, ◑ = distributed

tion of future schemes. However, we may need to settle for full security under parametrized assumptions rather than under static ones.

Nevertheless, it seems that the focus on improvements in security has resulted in a declined interest in improving the general efficiency and other practical aspects of ABE within the theoretical community. As we have shown, achieving some desirable properties simultaneously may result in the inability to support other properties. For instance, the simultaneous support of large universes and non-monotone access structures negatively affects the resilience and efficiency of a scheme. To shift the attention to the practical issues outlined in this paper, we have proposed several directions for future research that we encourage the community to address. In addition, we encourage the practical community to seriously consider ABE in the design of cryptographically-enforced access control. At this point, ABE is already at a reasonably advanced stage and may be beneficial in settings for which no practical and secure alternatives exist.

In particular, we motivated that ABE provides a practical and secure solution to enforcing access control. We explained the problem through a simple use case of an EHR system in the multiple-domain setting. In Sect. 2, we identified two properties to increase availability and scalability: user independence and authority-dependence minimization. These properties minimize the role of the authorities in the enforcement of access control. In general, ABE provides user independence. After the users have received secret keys from the authority, they can decrypt any ciphertexts for which they are authorized without requiring interaction again. The authority-dependence minimization property is relevant in the multi-authority setting and minimizes the number of authorities with which the user has to interact. In particular, this property ensures that decrypting users do not have to interact with authorities for which they have no relevant credentials. We defined the notion of decentralized ABE such that decentralized schemes satisfy the ADM property. Because decentralized ABE additionally provides security against corruption, it ensures that access control can be securely enforced by various (possibly mutually distrusting) authorities. This makes decentralized ABE especially attractive as a solution in multiple-domain settings. We showed that several schemes are decentralized, but that these can benefit from improvements.

We posed directions for future research (Appendix B) to mitigate the disadvantages of ABE while maintaining and even amplifying its benefits; thus allowing ABE to reach its full potential. Currently, the main disadvantage of ABE is the general inefficiency compared to other cryptographic primitives. In contrast, other primitives such as traditional public-key encryption, proxy re-encryption [30] and identity-based proxy re-encryption [74] are more efficient on the user side. For this reason, many of our directions for future research address the efficiency of ABE. If these can be solved, ABE becomes more efficient, mitigating its main disadvantage. Furthermore, the other directions address features that ABE provides or can potentially provide that other solutions may not be able to sufficiently address. One of the most prominent advantages in this is that ABE does not require an online entity during each access request. By exploring these directions, ABE becomes even more practical and secure, reaching its full potential as a mechanism to implement efficient and secure access control in practice.

Acknowledgements The authors would like to thank Lejla Batina and Jan Schoone, as well as the anonymous reviewers for helpful comments and suggestions.

Appendix A: Performance analysis: the schemes and access structures

In this appendix, we provide some additional elaborations on the performance estimates in Sect. 7.6.

A.1 The type-converted schemes

We provide the converted versions of the Wat11 (with full-domain hashes [143]) and RW13 [130] schemes that we used in the performance estimates in Sect. 7.6. In both conversions, we used the following design approach: we optimize the ciphertext size (and by extension, the efficiency of the encryption algorithm) and not the key size (conform [6]).

Definition 12 (*The Wat11 scheme with full-domain hash* [143]) The large-universe version of the Wat11 scheme using a full-domain hash with optimized encryption algorithm is defined as follows.

- Setup(λ): Taking as input the security parameter λ, the KGA generates three groups $\mathbb{G}, \mathbb{H}, \mathbb{G}_T$ of prime order p with generators $g \in \mathbb{G}, h \in \mathbb{H}$, and chooses a pairing $e\colon \mathbb{G} \times \mathbb{H} \to \mathbb{G}_T$, and a hash function modeled as a random oracle $\mathcal{H}\colon \{0,1\}^* \to \mathbb{G}$. The universe of attributes is $\mathcal{U} = \mathbb{Z}_p$. The KGA also generates randoms $\alpha, b \in_R \mathbb{Z}_p$. It outputs MSK $= (\alpha, b)$ as its master secret key and publishes the master public key as

$$\text{MPK} = (\mathcal{U}, p, \mathbb{G}, \mathbb{H}, \mathbb{G}_T, g, h, A = e(g,h)^{\alpha}, B = g^{b}, \mathcal{H}).$$

- KeyGen(MSK, $\mathcal{S}$): The KGA generates a secret key for a user that possesses a set of attributes $\mathcal{S}$ by generating random integers $r \in_R \mathbb{Z}_p$ and computing the secret key as

$$\text{SK}_{\mathcal{S}} = (K = h^{\alpha - rb}, K' = h^{r}, \{K_{\text{att}} = \mathcal{H}(\text{att})^{r}\}_{\text{att} \in \mathcal{S}}).$$

- Encrypt(M, MPK, $\mathbb{A}$): An encrypting user encrypts message $M \in \mathbb{G}_T$ under $\mathbb{A} = (\mathbf{A}, \rho)$ with $\mathbf{A} \in \mathbb{Z}_p^{n_1 \times n_2}$, $\rho\colon n_1 \to \mathcal{U}$. The user then generates random integers $s, s_1, ..., s_{n_1}, v_2, ..., v_{n_2} \in_R \mathbb{Z}_p$ and computes the ciphertext as

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

  such that λ_i denotes the i-th entry of $\mathbf{A} \cdot (s, v_2, ..., v_{n_2})^{\intercal}$.
- Decrypt($\text{SK}_{\mathcal{S}}, \text{CT}_{\mathbb{A}}$): Suppose that $\mathcal{S}$ satisfies $\mathbb{A}$, and suppose $\Upsilon = \{j \in \{1, ..., n_1\} \mid \rho(j) \in \mathcal{S}\}$, such that $\{\varepsilon_j \in \mathbb{Z}_p\}_{j \in \Upsilon}$ exist with $\sum_{i \in \Upsilon} \varepsilon_j \mathbf{A}_j = (1, 0, ..., 0)$. Then the plaintext M is retrieved by computing

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

Definition 13 (*The RW13 scheme* [130]) The RW13 scheme with optimized encryption is defined as follows.

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

$$\mathrm{MPK} = (\mathcal{U}, p, \mathbb{G}, \mathbb{H}, \mathbb{G}_T, g, h, A = e(g,h)^{\alpha}, B = g^{b}, B_0 = g^{b_0}, B_1 = g^{b_1}, B' = g^{b'}).$$

- KeyGen(MSK, $\mathcal{S}$): The KGA generates a secret key for a user that possesses a set of attributes $\mathcal{S}$ by generating random integers $r, r_{\mathrm{att}} \in_R \mathbb{Z}_p$ for each att $\in \mathcal{S}$, letting $x_{\mathrm{att}} \in \mathbb{Z}_p$ denote the representation[6] of att in $\mathbb{Z}_p$ and computing the secret key as

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

- Encrypt(MPK, $\mathbb{A}$, M): An encrypting user encrypts message $M \in \mathbb{G}_T$ under $\mathbb{A} = (\mathbf{A}, \rho)$ with $\mathbf{A} \in \mathbb{Z}_p^{n_1 \times n_2}$ and $\rho\colon \{1, ..., n_1\} \to \mathcal{U}$ by generating random integers $s, s_1, ..., s_{n_1}, v_2, ..., v_{n_2} \in_R \mathbb{Z}_p$ and computes the ciphertext as $\mathrm{CT}_{\mathbb{A}} =$

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

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

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

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

A.2 Access structure

As mentioned in Observation 6, the access structures used in [96] yield more efficient implementations of the decryption algorithm. We assume that the access structure is represented as a Boolean formula (only consisting of AND- and OR-gates).

Generation of the matrix.

- First, write the Boolean formula as an access tree, in which each node represents an AND- or OR-gate, and the leaves represent the attributes.
- Initialize the root node with vector $\mathbf{x} = (1)$ of length $n_2 = 1$. Note that n_2 is a globally updated counter in the protocol.
- OR-gate: propagate the node vector $\mathbf{x}$ to both children.
- AND-gate: split the node vector $\mathbf{x}$ in two vectors: $(\mathbf{x} \mid 0^{n_2 - |\mathbf{x}|} \mid 1)$ and $(0^{l} \mid -1)$ with length $n_2 + 1$, update $n_2 \leftarrow n_2 + 1$ and propagate the vectors to the children.
- Leaves (after all nodes have been propagated): set the vector $\mathbf{x}$ inherited from the parent to $(\mathbf{x} \mid 0^{n_2 - |\mathbf{x}|})$ if $|\mathbf{x}| < n_2$.

The output of this algorithm is an LSSS matrix $\mathbf{A}$ spanned by the rows in the leaves. The total number of rows corresponds with the length of the policy n_1, and the total number of columns with the resulting n_2 after the protocol run. The secret sharing vector $\mathbf{v}$ is generated by choosing the entries at random, i.e., $\mathbf{v} \in \mathbb{Z}_p^{n_2}$.

Reconstruction of the secret. For each row $j \in \{1, ..., n_1\}$ corresponding with an attribute that is in the set of attributes, the variable $\varepsilon_j = 1$. Otherwise, $\varepsilon_j = 0$.

[6] This can be done, for instance, with a collision-resistant hash function [133].

Appendix B: A list of the future directions

- Direction 1: Selective versus full security in practice
- Direction 2: Security of ABE with proofs in idealized models in practice
- Direction 3: CP-ABE with short ciphertexts
- Direction 4: Compact unbounded ABE with flexible efficiency
- Direction 5: Unbounded ABE with efficient decryption
- Direction 6: Generic online/offline conversions
- Direction 7: Optimized and secure implementations of schemes
- Direction 8: Benchmarking schemes for practice
- Direction 9: Type conversion
- Direction 10: Choosing a pairing-friendly group
- Direction 11: Benchmarking state-of-the-art large-universe schemes
- Direction 12: Decentralized ABE conform standard form
- Direction 13: Decentralized ABE without full-domain hash
- Direction 14: Anonymous key issuance protocol for FDH-MA-ABE
- Direction 15: Privacy-preserving key generation
- Direction 16: Attribute-wise key generation
- Direction 17: Systematizing and generalizing revocation methods

Appendix C: A list of acronyms used throughout the paper

ABE	Attribute-based encryption
ABAC	Attribute-based access control
ADM	Authority-dependence minimization
CCA	Chosen-ciphertext attack
CPA	Chosen-plaintext attack
CP-ABE	Ciphertext-policy attribute-based encryption
CT	Ciphertext
DBDH	Decisional bilinear Diffie–Hellman
DNF	Disjunctive normal form
EHR	Electronic health record
FDH	Full-domain hash
GGM	Generic group model
GP	Global parameters
IBE	Identity-based encryption
IoT	Internet of Things
KGA	Key generation authority
KP-ABE	Key-policy attribute-based encryption
LSSS	Linear secret sharing scheme
MA-ABE	Multi-authority attribute-based encryption
MA-CP-ABE	Multi-authority ciphertext-policy attribute-based encryption
MPK	Master public key
MSK	Master secret key
(N)MSP	(Non-)monotone span program
NT-ADM	Non-trivial authority-dependence minimization
RBAC	Role-based access control
ROM	Random oracle model
SK	Secret key
(S)XDH	(Symmetric) external Diffie–Hellman

References

1. Abe M., Groth J., Ohkubo M., Tango T.: Converting cryptographic schemes from symmetric to asymmetric bilinear groups. In: CRYPTO, pp. 241–260. Springer (2014).
2. Abe M., Hoshino F., Ohkubo M.: Design in type-i, run in type-iii: Fast and scalable bilinear-type conversion using integer programming. In: CRYPTO, pp. 387–415. Springer (2016).
3. Agrawal S., Chase M.: A study of pair encodings: Predicate encryption in prime order groups. In: TCC, pp. 259–288. Springer (2016).
4. Agrawal S., Chase M.: FAME: fast attribute-based message encryption. In: CCS, pp. 665–682. ACM (2017).
5. Agrawal S., Chase M.: Simplifying design and analysis of complex predicate encryption schemes. In: EUROCRYPT, pp. 627–656. Springer (2017).
6. Akinyele J.A., Garman C., Hohenberger S.: Automating fast and secure translations from type-i to type-iii pairing schemes. In: CCS, pp. 1370–1381. ACM (2015).
7. Akinyele J.A., Pagano M.W., Green M.D., Lehmann C.U., Peterson Z.N.J., Rubin A.D.: Securing electronic medical records using attribute-based encryption on mobile devices. In: SPSM, pp. 75–86. ACM (2011).
8. Alemán J.L.F., Señor I.C., Lozoya P.Á.O., Toval A.: Security and privacy in electronic health records: A systematic literature review. J. Biomed. Informatics **46**(3), 541–562 (2013).
9. Ambrona M.: Generic negation of pair encodings. In: Garay J.A. (ed.) PKC, Lecture Notes in Computer Science, vol. 12711, pp. 120–146. Springer (2021).
10. 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).
11. Aranha D.: Pairings are not dead, just resting (2017). https://ecc2017.cs.ru.nl/slides/ecc2017-aranha.pdf
12. 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
13. 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).
14. Attrapadung N.: Dual system encryption framework in prime-order groups via computational pair encodings. In: ASIACRYPT, pp. 591–623. Springer (2016).
15. Attrapadung N.: Unbounded dynamic predicate compositions in attribute-based encryption. In: EUROCRYPT, pp. 34–67. Springer (2019).
16. Attrapadung N., Hanaoka G., Matsumoto T., Teruya T., Yamada S.: Attribute based encryption with direct efficiency tradeoff. In: ACNS, pp. 249–266. Springer (2016).
17. Attrapadung N., Hanaoka G., Yamada S.: Conversions among several classes of predicate encryption and applications to ABE with various compactness tradeoffs. In: ASIACRYPT, pp. 575–601. Springer (2015).
18. Attrapadung N., Imai H.: Attribute-based encryption supporting direct/indirect revocation modes. In: Parker M.G. (ed.) IMACC, LNCS, vol. 5921, pp. 278–300. Springer (2009).
19. Attrapadung N., Imai H.: Conjunctive broadcast and attribute-based encryption. In: Pairing, LNCS, vol. 5671, pp. 248–265. Springer (2009).
20. Attrapadung N., Libert B.: Functional encryption for inner product: Achieving constant-size ciphertexts with adaptive security or support for negation. In: PKC, pp. 384–402. Springer (2010).
21. Attrapadung N., Libert B., de Panafieu E.: Expressive key-policy attribute-based encryption with constant-size ciphertexts. In: PKC, pp. 90–108. Springer (2011).
22. Attrapadung N., Tomida J.: Unbounded dynamic predicate compositions in ABE from standard assumptions. In: ASIACRYPT, pp. 405–436. Springer (2020).
23. Attrapadung N., Yamada S.: Duality in ABE: converting attribute based encryption for dual predicate and dual policy via computational encodings. In: CT-RSA, pp. 87–105. Springer (2015).
24. Barbulescu R., Duquesne S.: Updating key size estimations for pairings. J. Cryptol. **32**(4), 1298–1336 (2019).
25. Barreto P.S.L.M., Lynn B., Scott M.: Constructing elliptic curves with prescribed embedding degrees. In: SCN, pp. 257–267. Springer (2002).
26. Barreto P.S.L.M., Naehrig M.: Pairing-friendly elliptic curves of prime order. In: SAC, pp. 319–331. Springer (2005).
27. Beimel A.: Secure schemes for secret sharing and key distribution (1996).
28. Bellare M., Rogaway P.: Random oracles are practical: A paradigm for designing efficient protocols. In: CCS, pp. 62–73. ACM (1993).

29. Bethencourt J., Sahai A., Waters B.: Ciphertext-policy attribute-based encryption. In: S&P, pp. 321–334. IEEE (2007).
30. Blaze M., Bleumer G., Strauss M.: Divertible protocols and atomic proxy cryptography. In: Nyberg K. (ed.) EUROCRYPT, LNCS, vol. 1403, pp. 127–144. Springer (1998).
31. Boneh D.: The decision diffie-hellman problem. In: Buhler J. (ed.) ANTS-III, LNCS, vol. 1423, pp. 48–63. Springer (1998).
32. Boneh D., Boyen X.: Efficient selective-id secure identity-based encryption without random oracles. In: EUROCRYPT, pp. 223–238. Springer (2004).
33. Boneh D., Boyen X., Goh E.J.: Hierarchical identity based encryption with constant size ciphertext. In: EUROCRYPT, pp. 440–456. Springer (2005).
34. Boneh D., Waters B.: Conjunctive, subset, and range queries on encrypted data. In: Vadhan S.P. (ed.) TCC, LNCS, vol. 4392, pp. 535–554. Springer (2007).
35. Boyen X.: The uber-assumption family – a unified complexity framework for bilinear groups. In: Pairing, pp. 39–56. Springer (2008).
36. Boyen X.: Attribute-based functional encryption on lattices. In: Sahai A. (ed.) TCC, LNCS, vol. 7785, pp. 122–142. Springer (2013).
37. Brickell E.F., Gordon D.M., McCurley K.S., Wilson D.B.: Fast exponentiation with precomputation (extended abstract). In: EUROCRYPT, pp. 200–207. Springer (1992).
38. Canetti R., Goldreich O., Halevi S.: The random oracle methodology, revisited. J. ACM **51**(4), 557–594 (2004).
39. Canetti R., Halevi S., Katz J.: Chosen-ciphertext security from identity-based encryption. In: Cachin C., Camenisch J. (eds.) EUROCRYPT, LNCS, vol. 3027, pp. 207–222. Springer (2004).
40. Chase M.: Multi-authority attribute-based encryption. In: TCC, pp. 515–534. Springer (2007).
41. Chase M., Chow S.S.M.: Improving privacy and security in multi-authority attribute-based encryption. In: Al-Shaer E., Jha S., Keromytis A.D. (eds.) CCS, pp. 121–130. ACM (2009).
42. Chatterjee S., Koblitz N., Menezes A., Sarkar P.: Another look at tightness II: practical issues in cryptography. In: Phan R.C., Yung M. (eds.) Mycrypt, LNCS, vol. 10311, pp. 21–55. Springer (2016).
43. Chen C., Chen J., Lim H.W., Zhang Z., Feng D., Ling S., Wang H.: Fully secure attribute-based systems with short ciphertexts/signatures and threshold access structures. In: CT-RSA, pp. 50–67. Springer (2013).
44. Chen C., Zhang Z., Feng D.: Efficient ciphertext policy attribute-based encryption with constant-size ciphertext and constant computation-cost. In: Boyen X., Chen X. (eds.) ProvSec, LNCS, vol. 6980, pp. 84–101. Springer (2011).
45. Chen J., Gay R., Wee H.: Improved dual system ABE in prime-order groups via predicate encodings. In: EUROCRYPT, pp. 595–624. Springer (2015).
46. Chen J., Gong J.: ABE with tag made easy - concise framework and new instantiations in prime-order groups. In: Takagi T., Peyrin T. (eds.) ASIACRYPT, LNCS, vol. 10625, pp. 35–65. Springer (2017).
47. Chen J., Gong J., Kowalczyk L., Wee H.: Unbounded ABE via bilinear entropy expansion, revisited. In: EUROCRYPT, pp. 503–534. Springer (2018).
48. Chen J., Wee H.: Fully, (almost) tightly secure IBE and dual system groups. In: Canetti R., Garay J.A. (eds.) CRYPTO, LNCS, vol. 8043, pp. 435–460. Springer (2013).
49. Chen J., Wee H.: Dual system groups and its applications — compact hibe and more. Cryptology ePrint Archive, Report 2014/265 (2014).
50. Chen J., Wee H.: Semi-adaptive attribute-based encryption and improved delegation for boolean formula. In: Abdalla M., Prisco R.D. (eds.) SCN, LNCS, vol. 8642, pp. 277–297. Springer (2014).
51. Cheon J.H.: Security analysis of the strong diffie-hellman problem. In: Vaudenay S. (ed.) EUROCRYPT, LNCS, vol. 4004, pp. 1–11. Springer (2006).
52. Cheung L., Newport C.C.: Provably secure ciphertext policy ABE. In: Ning P., di Vimercati S.D.C., Syverson P.F. (eds.) CCS, pp. 456–465. ACM (2007).
53. Chow S.S.M.: Removing escrow from identity-based encryption. In: Jarecki S., Tsudik G. (eds.) PKC, Lecture Notes in Computer Science, vol. 5443, pp. 256–276. Springer (2009).
54. Chow S.S.M.: A framework of multi-authority attribute-based encryption with outsourcing and revocation. In: Wang X.S., Bauer L., Kerschbaum F. (eds.) SACMAT, pp. 215–226. ACM (2016).
55. Cui H., Deng R.H., Li Y., Qin B.: Server-aided revocable attribute-based encryption. In: Askoxylakis I.G., Ioannidis S., Katsikas S.K., Meadows C.A. (eds.) ESORICS, LNCS, vol. 9879, pp. 570–587. Springer (2016).
56. de la Piedra A., Venema M., Alpár G.: ABE squared: Accurately benchmarking efficiency of attribute-based encryption. IACR Trans. Cryptogr. Hardw. Embed. Syst. **2022**(2), 192–239 (2022).
57. de Lemos R., Giese H., Müller H.A., Shaw M., Andersson J., Litoiu M., Schmerl B.R., Tamura G., Villegas N.M., Vogel T., Weyns D., Baresi L., Becker B., Bencomo N., Brun Y., Cukic B., Desmarais

R.J., Dustdar S., Engels G., Geihs K., Göschka K.M., Gorla A., Grassi V., Inverardi P., Karsai G., Kramer J., Lopes A., Magee J., Malek S., Mankovski S., Mirandola R., Mylopoulos J., Nierstrasz O., Pezzè M., Prehofer C., Schäfer W., Schlichting R.D., Smith D.B., Sousa J.P., Tahvildari L., Wong K., Wuttke J.: Software engineering for self-adaptive systems: A second research roadmap. In: de Lemos R., Giese H., Müller H.A., Shaw M. (eds.) Software Engineering for Self-Adaptive Systems II - International Seminar, Dagstuhl Castle, Germany, October 24-29, 2010 Revised Selected and Invited Papers, Lecture Notes in Computer Science, vol. 7475, pp. 1–32. Springer (2010).
58. Deng M., Wuyts K., Scandariato R., Preneel B., Joosen W.: A privacy threat analysis framework: supporting the elicitation and fulfillment of privacy requirements. Requir. Eng. **16**(1), 3–32 (2011).
59. Dent A.W.: Adapting the weaknesses of the random oracle model to the generic group model. In: Zheng Y. (ed.) ASIACRYPT, LNCS, vol. 2501, pp. 100–109. Springer (2002).
60. Diffie W., Hellman M.E.: New directions in cryptography. IEEE Trans. Inf. Theory **22**(6), 644–654 (1976).
61. ETSI: ETSI TS 103 458 (V1.1.1) (2018).
62. ETSI: ETSI TS 103 532 (V1.1.1) (2018).
63. Freeman D.M.: Converting pairing-based cryptosystems from composite-order groups to prime-order groups. In: Gilbert H. (ed.) EUROCRYPT, LNCS, vol. 6110, pp. 44–61. Springer (2010).
64. Fujisaki E., Okamoto T.: Secure integration of asymmetric and symmetric encryption schemes. In: Wiener M.J. (ed.) CRYPTO, LNCS, vol. 1666, pp. 537–554. Springer (1999).
65. Galbraith S.D.: New discrete logarithm records, and the death of type 1 pairings. https://ellipticnews.wordpress.com/2014/02/01/new-discrete-logarithm-records-and-the-death-of-type-1-pairings/ (2014).
66. Galbraith S.D., Paterson K.G., Smart N.P.: Pairings for cryptographers. Discret. Appl. Math. **156**(16), 3113–3121 (2008).
67. Gamal T.E.: A public key cryptosystem and a signature scheme based on discrete logarithms. In: Blakley G.R., Chaum D. (eds.) CRYPTO, LNCS, vol. 196, pp. 10–18. Springer (1984).
68. Garg S., Gentry C., Halevi S., Sahai A., Waters B.: Attribute-based encryption for circuits from multilinear maps. In: Canetti R., Garay J.A. (eds.) CRYPTO, LNCS, vol. 8043, pp. 479–499. Springer (2013).
69. Garg S., Kumarasubramanian A., Sahai A., Waters B.: Building efficient fully collusion-resilient traitor tracing and revocation schemes. In: Al-Shaer E., Keromytis A.D., Shmatikov V. (eds.) CCS, pp. 121–130. ACM (2010).
70. Ge A., Zhang R., Chen C., Ma C., Zhang Z.: Threshold ciphertext policy attribute-based encryption with constant size ciphertexts. In: Susilo W., Mu Y., Seberry J. (eds.) ACISP, LNCS, vol. 7372, pp. 336–349. Springer (2012).
71. Gorbunov S., Vaikuntanathan V., Wee H.: Attribute-based encryption for circuits. J. ACM **62**(6), 45:1–45:33 (2015).
72. Goyal V., Pandey O., Sahai A., Waters B.: Attribute-based encryption for fine-grained access control of encrypted data. In: CCS. ACM (2006).
73. 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).
74. Green M., Ateniese G.: Identity-based proxy re-encryption. In: Katz J., Yung M. (eds.) ACNS, LNCS, vol. 4521, pp. 288–306. Springer (2007).
75. Green M., Hohenberger S., Waters B.: Outsourcing the decryption of ABE ciphertexts. In: USENIX Security Symposium. USENIX Association (2011).
76. Guillevic A.: Comparing the pairing efficiency over composite-order and prime-order elliptic curves. In: Jacobson Jr. M.J., Locasto M.E., Mohassel P., Safavi-Naini R. (eds.) ACNS, LNCS, vol. 7954, pp. 357–372. Springer (2013).
77. Guillevic A.: Pairing-friendly curves. https://members.loria.fr/AGuillevic/pairing-friendly-curves/ (2020).
78. Guillevic 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).
79. Guillevic A., Singh S.: On the alpha value of polynomials in the tower number field sieve algorithm. Cryptology ePrint Archive, Report 2019/885 (2019).
80. Häyrinen K., Saranto K., Nykänen P.: Definition, structure, content, use and impacts of electronic health records: A review of the research literature. Int. J. Medical Informatics **77**(5), 291–304 (2008).
81. Herranz J., Laguillaumie F., Ràfols C.: Constant size ciphertexts in threshold attribute-based encryption. In: Nguyen P.Q., Pointcheval D. (eds.) PKC, LNCS, vol. 6056, pp. 19–34. Springer (2010).

82. Hiller J., McMullen M.S., Chumney W.M., Baumer D.L.: Privacy and security in the implementation of health information technology (electronic health records): U.s. and eu compared. Boston University Journal of Science & Technology Law **17**(1), 1–39 (2011).
83. Hohenberger S., Waters B.: Attribute-based encryption with fast decryption. In: Kurosawa K., Hanaoka G. (eds.) PKC, LNCS, vol. 7778, pp. 162–179. Springer (2013).
84. Hohenberger S., Waters B.: Online/offline attribute-based encryption. In: PKC, pp. 293–310. Springer (2014).
85. 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
86. Ibraimi L., Petkovic M., Nikova S., Hartel P.H., Jonker W.: Mediated ciphertext-policy attribute-based encryption and its application. In: Youm H.Y., Yung M. (eds.) WISA, LNCS, vol. 5932, pp. 309–323. Springer (2009).
87. Kamara S., Lauter K.E.: Cryptographic cloud storage. In: FC, pp. 136–149. Springer (2010).
88. Katz J., Sahai A., Waters B.: Predicate encryption supporting disjunctions, polynomial equations, and inner products. In: Smart N.P. (ed.) EUROCRYPT, LNCS, vol. 4965, pp. 146–162. Springer (2008).
89. Koblitz N., Menezes A.J.: The random oracle model: a twenty-year retrospective. Des. Codes Cryptogr. **77**(2–3), 587–610 (2015).
90. Koppula V., Waters B.: Realizing chosen ciphertext security generically in attribute-based encryption and predicate encryption. In: Boldyreva A., Micciancio D. (eds.) CRYPTO, LNCS, vol. 11693, pp. 671–700. Springer (2019)
91. Kowalczyk L., Lewko A.B.: Bilinear entropy expansion from the decisional linear assumption. In: Gennaro R., Robshaw M. (eds.) CRYPTO, LNCS, vol. 9216, pp. 524–541. Springer (2015).
92. Kowalczyk L., Wee H.: Compact adaptively secure ABE for nc^1 from k-lin. In: EUROCRYPT, pp. 3–33. Springer (2019).
93. Lai J., Tang Q.: Making any attribute-based encryption accountable, efficiently. In: López J., Zhou J., Soriano M. (eds.) ESORICS, LNCS, vol. 11099, pp. 527–547. Springer (2018).
94. Leurent G., Nguyen P.Q.: How risky is the random-oracle model? In: Halevi S. (ed.) CRYPTO, Lecture Notes in Computer Science, vol. 5677, pp. 445–464. Springer (2009).
95. Lewko A., Sahai A., Waters B.: Revocation systems with very small private keys. In: IEEE S & P, pp. 273–285 (2010).
96. Lewko A., Waters B.: Decentralizing attribute-based encryption. Cryptology ePrint Archive, Report 2010/351 (2010).
97. Lewko A., Waters B.: Decentralizing attribute-based encryption. In: EUROCRYPT, pp. 568–588. Springer (2011).
98. Lewko A.B.: Tools for simulating features of composite order bilinear groups in the prime order setting. In: Pointcheval D., Johansson T. (eds.) EUROCRYPT, LNCS, vol. 7237, pp. 318–335. Springer (2012).
99. Lewko A.B., Okamoto T., Sahai A., Takashima K., Waters B.: Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption. In: EUROCRYPT, pp. 62–91. Springer (2010).
100. Lewko A.B., Waters B.: New techniques for dual system encryption and fully secure HIBE with short ciphertexts. In: Micciancio D. (ed.) TCC, LNCS, vol. 5978, pp. 455–479. Springer (2010).
101. Lewko A.B., Waters B.: Unbounded HIBE and attribute-based encryption. In: EUROCRYPT, pp. 547–567. Springer (2011).
102. Lewko A.B., Waters B.: New proof methods for attribute-based encryption: Achieving full security through selective techniques. In: CRYPTO, pp. 180–198. Springer (2012).
103. Lewko A.B., Waters B.: Why proving HIBE systems secure is difficult. In: Nguyen P.Q., Oswald E. (eds.) EUROCRYPT, LNCS, vol. 8441, pp. 58–76. Springer (2014).
104. Li J., Chen X., Li J., Jia C., Ma J., Lou W.: Fine-grained access control system based on outsourced attribute-based encryption. In: Crampton J., Jajodia S., Mayes K. (eds.) ESORICS, LNCS, vol. 8134, pp. 592–609. Springer (2013).
105. Li J., Huang Q., Chen X., Chow S.S.M., Wong D.S., Xie D.: Multi-authority ciphertext-policy attribute-based encryption with accountability. In: Cheung B.S.N., Hui L.C.K., Sandhu R.S., Wong D.S. (eds.) ASIACCS, pp. 386–390. ACM (2011).
106. Lin H., Cao Z., Liang X., Shao J.: Secure threshold multi authority attribute based encryption without a central authority. In: Chowdhury D.R., Rijmen V., Das A. (eds.) INDOCRYPT, LNCS, vol. 5365, pp. 426–436. Springer (2008).
107. Lin H., Luo J.: Compact adaptively secure ABE from k-lin: Beyond nc^1 and towards NL. In: EUROCRYPT, pp. 247–277. Springer (2020).

108. Lin H., Luo J.: Succinct and adaptively secure ABE for ABP from k-lin. In: Moriai S., Wang H. (eds.) ASIACRYPT, LNCS, vol. 12493, pp. 437–466. Springer (2020).
109. Liu J.K., Yuen T.H., Zhang P., Liang K.: Time-based direct revocable ciphertext-policy attribute-based encryption with short revocation list. In: Preneel B., Vercauteren F. (eds.) ACNS, LNCS, vol. 10892, pp. 516–534. Springer (2018).
110. Liu Z., Cao Z., Huang Q., Wong D.S., Yuen T.H.: Fully secure multi-authority ciphertext-policy attribute-based encryption without random oracles. In: Atluri V., Díaz C. (eds.) ESORICS, LNCS, vol. 6879, pp. 278–297. Springer (2011).
111. Liu Z., Cao Z., Wong D.S.: Blackbox traceable CP-ABE: how to catch people leaking their keys by selling decryption devices on ebay. In: Sadeghi A., Gligor V.D., Yung M. (eds.) CCS, pp. 475–486. ACM (2013).
112. Liu Z., Wong D.S.: Practical ciphertext-policy attribute-based encryption: Traitor tracing, revocation, and large universe. In: Malkin T., Kolesnikov V., Lewko A.B., Polychronakis M. (eds.) ACNS, LNCS, vol. 9092, pp. 127–146. Springer (2015).
113. Lynn B.: The stanford pairing based crypto library. http://crypto.stanford.edu/pbc
114. Malek S., Mikic-Rakic M., Medvidovic N.: A decentralized redeployment algorithm for improving the availability of distributed systems. In: Dearle A., Eisenbach S. (eds.) Component Deployment, Lecture Notes in Computer Science, vol. 3798, pp. 99–114. Springer (2005).
115. Michalevsky Y., Joye M.: Decentralized policy-hiding ABE with receiver privacy. In: López J., Zhou J., Soriano M. (eds.) ESORICS, LNCS, vol. 11099, pp. 548–567. Springer (2018).
116. Möller B.: Algorithms for multi-exponentiation. In: SAC, pp. 165–180. Springer (2001).
117. Müller S., Katzenbeisser S., Eckert C.: Distributed attribute-based encryption. In: Lee P.J., Cheon J.H. (eds.) ICISC, LNCS, vol. 5461, pp. 20–36. Springer (2008).
118. Narayanan A., Shmatikov V.: Robust de-anonymization of large sparse datasets. In: S& P, pp. 111–125. IEEE Computer Society (2008).
119. Nishide T., Yoneyama K., Ohta K.: Attribute-based encryption with partially hidden encryptor-specified access structures. In: Bellovin S.M., Gennaro R., Keromytis A.D., Yung M. (eds.) ACNS, LNCS, vol. 5037, pp. 111–129 (2008).
120. of European Union C.: Regulation (eu) 2016/679 of the european parliament and of the council. https://eur-lex.europa.eu/eli/reg/2016/679/oj (2016).
121. Okamoto T., Takashima K.: Fully secure functional encryption with general relations from the decisional linear assumption. In: Rabin T. (ed.) CRYPTO, LNCS, vol. 6223, pp. 191–208. Springer (2010).
122. Okamoto T., Takashima K.: Fully secure unbounded inner-product and attribute-based encryption. In: ASIACRYPT, pp. 349–366. Springer (2012).
123. Okamoto T., Takashima K.: Decentralized attribute-based signatures. In: Kurosawa K., Hanaoka G. (eds.) PKC, LNCS, vol. 7778, pp. 125–142. Springer (2013).
124. Ostrovsky R., Sahai A., Waters B.: Attribute-based encryption with non-monotonic access structures. In: CCS, pp. 195–203. ACM (2007).
125. Paterson K.G., Price G.: A comparison between traditional public key infrastructures and identity-based cryptography. Inf. Secur. Tech. Rep. **8**(3), 57–72 (2003).
126. Pirretti M., Traynor P., McDaniel P.D., Waters B.: Secure attribute-based systems. J. Comput. Secur. **18**(5), 799–837 (2010).
127. Rackoff C., Simon D.R.: Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In: Feigenbaum J. (ed.) CRYPTO, LNCS, vol. 576, pp. 433–444. Springer (1991).
128. Rao Y.S., Dutta R.: Decentralized ciphertext-policy attribute-based encryption scheme with fast decryption. In: Decker B.D., Dittmann J., Kraetzer C., Vielhauer C. (eds.) CMS, LNCS, vol. 8099, pp. 66–81. Springer (2013).
129. Rogaway P., Shrimpton T.: Cryptographic hash-function basics: Definitions, implications, and separations for preimage resistance, second-preimage resistance, and collision resistance. In: Roy B.K., Meier W. (eds.) FSE, LNCS, vol. 3017, pp. 371–388. Springer (2004).
130. Rouselakis Y., Waters B.: Practical constructions and new proof methods for large universe attribute-based encryption. In: CCS, pp. 463–474. ACM (2013).
131. Rouselakis Y., Waters B.: Efficient statically-secure large-universe multi-authority attribute-based encryption. In: Böhme R., Okamoto T. (eds.) FC, LNCS, vol. 8975, pp. 315–332. Springer (2015).
132. Sahai A., Seyalioglu H., Waters B.: Dynamic credentials and ciphertext delegation for attribute-based encryption. In: CRYPTO, pp. 199–217. Springer (2012).
133. Sahai A., Waters B.: Fuzzy identity-based encryption. In: EUROCRYPT, pp. 457–473. Springer (2005).
134. Sandhu R.S., Coyne E.J., Feinstein H.L., Youman C.E.: Role-based access control models. Computer **29**(2), 38–47 (1996).

135. Sandhu R.S., Samarati P.: Access control: principles and practice. IEEE Commun. Mag. **32**(9), 40–48 (1994).
136. 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).
137. Shamir A.: How to share a secret. Commun. ACM **22**(11), 612–613 (1979).
138. Shoup V.: Lower bounds for discrete logarithms and related problems. In: Fumy W. (ed.) EUROCRYPT, LNCS, vol. 1233, pp. 256–266. Springer (1997).
139. Sweeney L.: Weaving technology and policy together to maintain confidentiality. The Journal of Law, Medicine & Ethics **25**(2–3), 98–110 (1997).
140. Takashima K.: Expressive attribute-based encryption with constant-size ciphertexts from the decisional linear assumption. In: SCN, pp. 298–317. Springer (2014).
141. Tomida J., Kawahara Y., Nishimaki R.: Fast, compact, and expressive attribute-based encryption. In: PKC, pp. 3–33. Springer (2020).
142. Venema M., Alpár G.: A bunch of broken schemes: A simple yet powerful linear approach to analyzing security of attribute-based encryption. In: Paterson K.G. (ed.) CT-RSA, LNCS, vol. 12704, pp. 100–125. Springer (2021).
143. Waters B.: Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization. Cryptology ePrint Archive, Report 2008/290 (2008).
144. Waters B.: Dual system encryption: Realizing fully secure IBE and HIBE under simple assumptions. In: Halevi S. (ed.) CRYPTO, LNCS, vol. 5677, pp. 619–636. Springer (2009).
145. Waters B.: Ciphertext-policy attribute-based encryption - an expressive, efficient, and provably secure realization. In: PKC, pp. 53–70. Springer (2011).
146. Wee H.: Dual system encryption via predicate encodings. In: TCC, pp. 616–637. Springer (2014).
147. Yamada S., Attrapadung N., Hanaoka G., Kunihiro N.: Generic constructions for chosen-ciphertext secure attribute based encryption. In: Catalano D., Fazio N., Gennaro R., Nicolosi A. (eds.) PKC, LNCS, vol. 6571, pp. 71–89. Springer (2011).
148. 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).
149. Yu S., Wang C., Ren K., Lou W.: Attribute based data sharing with attribute revocation. In: Feng D., Basin D.A., Liu P. (eds.) ASIACCS, pp. 261–270. ACM (2010).
150. Zeutro: The openabe library - open source cryptographic library with attribute-based encryption implementations in c/c++. https://github.com/zeutro/openabe
151. Zhang K., Gong J., Tang S., Chen J., Li X., Qian H., Cao Z.: Practical and efficient attribute-based encryption with constant-size ciphertexts in outsourced verifiable computation. In: Chen X., Wang X., Huang X. (eds.) ASIACCS, pp. 269–279. ACM (2016).