# A Bunch of Broken Schemes: A Simple yet Powerful Linear Approach to Analyzing Security of Attribute-Based Encryption

Marloes Venema[1] and Greg Alpár[1,2]

[1] Radboud University, Nijmegen, the Netherlands
[2] Open University of the Netherlands
{m.venema,g.alpar}@cs.ru.nl

**Abstract.** We present a linear approach to analyzing security of attribute-based encryption (ABE). We use this approach to algebraically break eleven schemes: two single-authority and nine multi-authority attribute-based encryption (MA-ABE) schemes. These latter attacks illustrate that mistakes are made in transforming single-authority schemes into multi-authority ones. Our linear approach is not only useful in the analysis of existing schemes, but can also be applied during the design and verification of new schemes. As such, it can prevent the design of insecure MA-ABE schemes in the future.

**Keywords:** attribute-based encryption · cryptanalysis · multi-authority attribute-based encryption · attacks.

## 1 Introduction

Attribute-based encryption (ABE) [34] is important in the cryptographic enforcement of access control. Ciphertext-policy (CP) ABE [5] naturally implements a fine-grained access control mechanism that puts the data owner in control, and is therefore often considered in applications involving e.g. cloud environments [30,42,44,43,25,27] or medical settings [33,29,31]. These applications of ABE allow the storage of data to be outsourced to potentially untrusted providers whilst ensuring that data owners can securely manage access to their data. Many such works use the multi-authority (MA) variant [6], which employs multiple authorities to generate and issue secret keys. These authorities can be associated with different organizations, e.g. hospitals, insurance companies or universities. This allows data owners, e.g. patients, to securely share their data with other users from various domains, e.g. doctors, actuaries or medical researchers. Because existing schemes may not sufficiently address the problems that arise in these real-world applications, new schemes are needed. Unfortunately, proving and verifying security of such schemes are difficult, and, perhaps unsurprisingly, several schemes turn out to be vulnerable to attacks.

In this work, we focus on algebraic attacks. In particular, we establish a framework for effectively analyzing schemes by exploiting their similar algebraic structure. These schemes use symmetric pairings that map two prime-order source groups to a target group. The secret keys and ciphertext elements exist in the source groups, and decryption involves the pairing of the key and ciphertext elements such that the blinding value—which exists in the target group and masks the message—can be recovered. To simplify the analysis, one can also consider the exponent spaces of the secret keys and ciphertexts. Then, decryption can be described as a *linear* combination of products of key and ciphertext entries such that the exponent of the blinding value can be recovered. This implies a more concise, intuitive and structured notation that also simplifies the (manual) security analysis of multi-authority schemes, which should be secure against corruption of authorities. To analyze the *insecurity* of schemes, we examine whether the blinding value can be recovered for some ciphertext and unauthorized secret keys. Concretely, we define different types of attacks involving the recovery of a master- or attribute-key, or the unauthorized decryption of a ciphertext, all impacting the overall security of a scheme. Furthermore, we describe a heuristic approach to finding such attacks. Using this, we have found vulnerabilities in several existing schemes that aim to solve real-world problems, rendering these at least partially insecure.

More generally, our goal is not necessarily to attack existing schemes, but to propose a framework that simplifies the design of secure (multi-authority) schemes. While the described approach is manual and non-exhaustive, it does help ruling out algebraic insecurities at an early stage in the design. In the future, our approach may be extended to be exhaustive and automated, such that the security of every new scheme can be verified efficiently. To some extent, such frameworks already exist for single-authority schemes [1,2], though our framework also contributes by providing concrete attacks and a general approach. Because of its importance in solving real-world problems, (MA-)ABE should be simple to securely design. Our framework provides a powerful tool in this endeavor.

### 1.1 Our contribution

Our contribution is twofold. First, we describe a linear approach to algebraic security analysis of ABE based on the common structure of many schemes. We describe three types of attacks, which model the implicit security requirements on the keys and ciphertexts. They model whether the master-key can be recovered, or whether users can collude and decrypt ciphertexts that they cannot individually decrypt. In the multi-authority setting, we also model the notion of corruption. Generally, the ability to perform any of the attacks proves the insecurity of a scheme. Second, we use our approach to show that fifteen works describing eleven different schemes are vulnerable to our attacks, consequently rendering them (partially) insecure. Five of these are insecure in the single-authority security model. The other six are insecure in the multi-authority security model, but are possibly secure if all authorities are assumed to be trusted. Essentially, these schemes provide a comparable level of security as single-authority schemes.

### 1.2 Technical details – a brief overview of the attacks

In CP-ABE, ciphertexts are associated with access policies, and secret keys are associated with sets of attributes. A secret key is authorized to decrypt a ciphertext if its access structure is satisfied by the associated set. These secret keys are generated by a KGA from a master-key, which can be used to decrypt any ciphertext. Users with keys for different sets of attributes should not be able to collude in collectively decrypting a ciphertext that they are individually not able to decrypt. Implicitly, the keys need to be secure in two ways. First, the master-key needs to be sufficiently hidden in the secret keys. Second, combining the secret keys of different users should not result in more decryption capabilities.

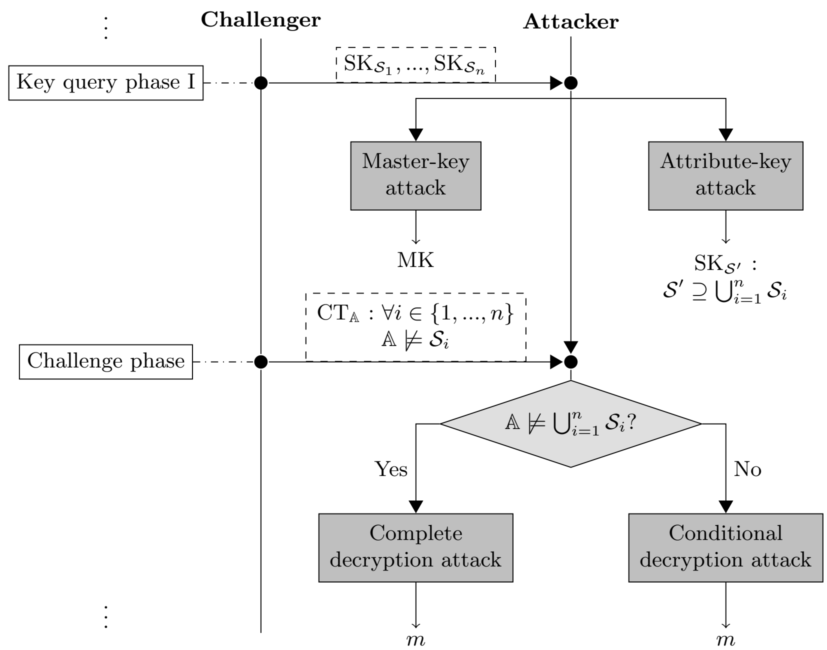
We propose three types of attacks, which all imply attacks on the security model for ABE. This model considers chosen-plaintext attacks (CPA) and collusion of users. Two of our attacks only consider the secret keys issued in the first key query phase of the CPA-security model, while the third also considers the challenge ciphertext. The attacks are informally defined as follows.

- **Master-key attack (MK):** The attacker can extract the KGA's master-key, which can be used to decrypt any ciphertext.
- **Attribute-key attack (AK):** The attacker can generate a secret key for a set $\mathcal{S}'$ that is strictly larger than each set $\mathcal{S}_i$ associated with a key.
- **Decryption attack (D):** The attacker can decrypt a ciphertext for which no authorized key was generated.

In addition, we formulate the notions of complete and conditional decryption attacks. These model the distinction between attacks that can be performed unconditionally, or only under conditions on the access structure and the collective set of attributes possessed by the colluding users. Specifically, complete attacks allow all ciphertexts to be decrypted by all unauthorized keys, while conditional attacks only allow a ciphertext to be decrypted if the associated access structure is satisfied by the collective set. Figure 1 illustrates the relationship between the attacks, and how the attacks relate to the security model. We consider the first key query phase and the challenge phase, which output the secret keys for a polynomial number of sets of attributes, and a ciphertext associated with an access structure such that all keys are unauthorized, respectively.

The security models in the multi-authority setting are similar, but include the notion of corruption. The attacker is allowed to corrupt one or more authorities in an attack, which should not yield sufficient power to enable an attack against the honest authorities. Sometimes, schemes employ a central authority (CA) in addition to employing multiple attribute authorities. This CA is assumed to perform the algorithms as expected, though it can be assumed to be honest or semi-honest. An honest CA is not allowed to collaborate with any other entities, while a semi-honest CA can passively collaborate with other entities in an attack. In this work, we show how to model the corruption of attribute authorities and semi-honest CAs, and how the additional knowledge (e.g. the master secret keys) gained from corrupting an authority can be included in the attacks.

**Fig. 1.** The general attacks and how they relate to one another.




We make some observations about the security models for multi-authority ABE. First, multi-authority ABE was initially designed [6,7,22] to provide security against corruption. This does not only protect honest authorities from corrupt authorities, but it also increases security from the perspective of the encrypting users. Second, not allowing corruption in the security model provides a comparable level of security to that of single-authority ABE. Third, some models are impractical. For instance, they might protect against corruption, but in turn only protect against a bounded number of colluding users [26]. Fourth, in some cases, the informal description of a scheme is ambiguous on whether it provides security against corruption. For instance, schemes are compared with other multi-authority schemes that provide security against corruption, while the proposed scheme does not, though is not explicitly mentioned [31,27].

Table 1 summarizes the schemes we analyzed. We indicate on which scheme it is based, which type of attack we found and whether it is complete, whether it uses collusion or corruption, whether the attack explicitly contradicts the model that the scheme is claimed to be secure in. We also list the conference or journal in which the scheme was published and how many times the paper is cited[3].

[3] According to Google Scholar. These measures were taken at 9 April 2020.

**Table 1.** Schemes for which we provide attacks.

| | Scheme | Based on | CD | Att. | Col. | Cor. | Con. | Venue | Cit. |
|---|---|---|---|---|---|---|---|---|---|
| | ZH10 [47]<br>ZHW13 [48] | - | ✗ | AK | 2 | - | ✓ | CCS<br>NC | 101<br>106 |
| | NDCW15 [30] | Wat11 [38] | ✓ | D | - | - | ✓ | ESORICS | 38 |
| **Multi-authority ABE** | YJ12 [42] | - | ✓ | MK | - | $\mathcal{A}$ | ✓ | NC | 141 |
| | YJR+13 [44]<br>WJB17 [40] | - | ✓ | D | - | - | ✓ | TIFS<br>NC | 452<br>24 |
| | JLWW13 [18]<br>JLWW15 [19] | BSW07 [5] | ✗ | AK | 2 | - | ✓ | NC<br>TIFS | 169<br>146 |
| | QLZ13 [32] | - | ✓ | MK | - | - | ✓ | ICICS | 37 |
| | YJ14 [43] | - | ✓ | D | - | $\mathcal{A}$ | ✓ | NC | 227 |
| | CM14 [9] | - | ✓ | D | - | $\mathcal{A}$ | U | NC | 42 |
| | QLZH15 [33] | Cha07 [6] | - | $\mathcal{C}$ | - | - | ✓ | IJIS | 105 |
| | LXXH16 [25]<br>MST17 [29] | Wat11 [38] | ✓ | MK | - | CA | ✓<br>U | NC<br>AsiaCCS | 95<br>17 |
| | PO17 [31] | - | ✓ | D | - | $\mathcal{A}$ | U | SACMAT | 11 |
| | MGZ19 I [27] | LW11 [22] | ✓ | MK | - | CA | U | Inscrypt | 1 |

CD = complete decryption attack, Att = attack, MK = master-key attack, AK = attribute-key attack, D = decryption attack, $\mathcal{C}$ = correctness attack; Col = collusion, Cor = corruption, Con = contradicts proposed security model, U = unclear, NC = non-crypto venue/journal
For convenience, we refer to the key schemes by concatenating the first letter of each author's surname with the last two digits of the year of publication.

### 1.3 Related work

Prior to this work, several ABE schemes were shown to be insecure. Some schemes were broken with respect to the same types of attacks as we introduce in this work. Others were only broken with respect to additional functionality. Table 2 summarizes each scheme, in which work it was broken, the proposed attack on the scheme, and whether it was later fixed. Also, we provide the venue and number of citations for these schemes. Notably, the schemes broken with respect to additional functionality have been fixed, while the others have not.

## 2 Preliminaries

We provide the necessary preliminaries and notations. If an element is chosen uniformly at random from some finite set $S$, we write $x \in_R S$. If an element $x$ is generated by running algorithm Alg, we write $x \leftarrow \text{Alg}$. We use boldfaced variables for vectors $\mathbf{x}$ and matrices $\mathbf{M}$, where $\mathbf{x}$ denotes a row vector and $\mathbf{y}^{\intercal}$ denotes a column vector. Furthermore, $x_i$ denotes the $i$-th entry of $\mathbf{x}$. If the vector size is unknown, $\mathbf{v} \in_R S$ indicates that for each entry: $v_i \in_R S$. Finally, $\mathbf{x}(y_1, y_2, ...)$ denotes a vector, where the entries are polynomials over variables

**Table 2.** The first three schemes were previously broken with respect to their additional functionality, and the second three were broken with respect to our attacks.

<table>
<tr><th>Scheme</th><th>Broken in</th><th colspan="5">Attacked functionality</th><th>Fixed?</th><th>Venue</th><th>Cit.</th></tr>
<tr><td>LRZW09 [23]</td><td rowspan="3">LHC+11 [24]<br>CDM15 [8]</td><td colspan="5" rowspan="3">Private access policies</td><td rowspan="3">[24]</td><td>ISC</td><td>188</td></tr>
<tr><td>ZCL+13 [46]</td><td>AsiaCCS</td><td>92</td></tr>
<tr><td>XFZ+14 [41]</td><td>NC</td><td>40</td></tr>
<tr><td>YJR+13 [44]</td><td>HXL15 [16]<br>WJB17 [40]</td><td colspan="5">Revocation</td><td>[40]</td><td>TIFS</td><td>452</td></tr>
<tr><td>JLWW15 [19]</td><td>MZY16 [28]</td><td colspan="5">Distributed key generation</td><td>[20]</td><td>TIFS</td><td>146</td></tr>
<tr><th>Scheme</th><th>Broken in</th><th>CD</th><th>Att.</th><th>Col.</th><th>Cor.</th><th>Con.</th><th>Fixed?</th><th>Venue</th><th>Cit.</th></tr>
<tr><td>HSMY12 [13]</td><td>GZZ+13 [11]</td><td>✗</td><td>D</td><td>2</td><td>-</td><td>✓</td><td>✗</td><td>NC</td><td>172</td></tr>
<tr><td>HSM+14 [14]</td><td rowspan="2">WZC15 [37]</td><td rowspan="2">✗</td><td rowspan="2">D</td><td rowspan="2">2</td><td rowspan="2">-</td><td rowspan="2">✓</td><td rowspan="2">✗</td><td>ESORICS</td><td>26</td></tr>
<tr><td>HSM+15 [15]</td><td>TIFS</td><td>108</td></tr>
<tr><td>YCT15 [45]</td><td>TYH19 [35]</td><td>✗</td><td>AK</td><td>2</td><td>-</td><td>✓</td><td>✗</td><td>NC</td><td>158</td></tr>
</table>

CD = complete decryption attack, Att = attack, AK = attribute-key attack, D = decryption attack; Col = collusion, Cor = corruption, Con = contradicts proposed security model, NC = non-crypto venue/journal

$y_1, y_2, ...$, with coefficients in some specified field. We refer to a polynomial with only one term, or alternatively one term of the polynomial, as a monomial.

### 2.1 Access structures

In this work, we only consider monotone access structures[4]. If a set $\mathcal{S}$ satisfies access structure $\mathbb{A}$, then we denote this as $\mathbb{A} \models \mathcal{S}$. We denote the $i$-th attribute in the access structure as $\text{att}_i \sim \mathbb{A}$.

### 2.2 Pairings

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

### 2.3 Formal definition of (multi-authority) ciphertext-policy ABE

In this work, we focus primarily on ciphertext-policy attribute-based encryption.

**Definition 1 (Ciphertext-policy ABE).** *A ciphertext-policy attribute-based encryption (CP-ABE) scheme with some authorities $\mathcal{A}_1, ..., \mathcal{A}_n$ (where $n \in \mathbb{N}$) such that each $\mathcal{A}_i$ manages universe $\mathcal{U}_i$, users and a universe of attributes $\mathcal{U} = \bigcup_{i=1}^{n} \mathcal{U}_i$ may consist of the following algorithms.*

[4] A formal definition of (monotone) access structures can be found in Appendix A.

- GlobalSetup($\lambda$) $\rightarrow$ GP*: This randomized algorithm takes as input the security parameter* $\lambda$*, and outputs the public global system parameters* GP *(independent of any attributes).*
- MKSetup(GP) $\rightarrow$ (GP, MK)*: This randomized algorithm takes as input the global parameters* GP*, and outputs the (secret) master-key* MK *(independent of any attributes) and updates the global parameters by adding the public key associated with* MK.
- AttSetup(att, MK, GP) $\rightarrow$ ($\mathrm{MSK}_{\mathrm{att}}$, $\mathrm{MPK}_{\mathrm{att}}$)*: This randomized algorithm takes as input an attribute, possibly the master-key and the global parameters, and outputs a master secret* $\mathrm{MSK}_{\mathrm{att}}$ *and public key* $\mathrm{MPK}_{\mathrm{att}}$ *associated with attribute* att.
- UKeyGen(id, MK, GP) $\rightarrow$ $\mathrm{SK}_{\mathrm{id}}$*: This randomized algorithm takes as input the identifier* id*, the master-key* MK *and the global parameters* GP*, and outputs the secret key* $\mathrm{SK}_{\mathrm{id}}$ *associated with* id.
- AttKeyGen($\mathcal{S}$, GP, MK, $\mathrm{SK}_{\mathrm{id}}$, $\{\mathrm{MSK}_{\mathrm{att}}\}_{\mathrm{att}\in\mathcal{S}}$) $\rightarrow$ $\mathrm{SK}_{\mathrm{id,att}}$*: This randomized algorithm takes as input an attribute* att *possessed by some user with identifier* id*, and the global parameters, the master-key* MK*, the secret key* $\mathrm{SK}_{\mathrm{id}}$ *and master secret key* $\mathrm{MSK}_{\mathrm{att}}$*, and outputs a user-specific secret key* $\mathrm{SK}_{\mathrm{id,att}}$.
- Encrypt($m$, $\mathbb{A}$, GP, $\{\mathrm{MPK}_{\mathrm{att}}\}_{\mathrm{att}\sim\mathbb{A}}$) $\rightarrow$ $\mathrm{CT}_{\mathbb{A}}$*: This randomized algorithm is run by any encrypting user and takes as input a message* $m$*, access structure* $\mathbb{A}$ *and the relevant public keys. It outputs the ciphertext* $\mathrm{CT}_{\mathbb{A}}$.
- Decrypt($\mathrm{SK}_{\mathrm{id},\mathcal{S}}$, $\mathrm{CT}_{\mathbb{A}}$) $\rightarrow$ $m$*: This deterministic algorithm takes as input a ciphertext* $\mathrm{CT}_{\mathbb{A}}$ *and secret key* $\mathrm{SK}_{\mathrm{id},\mathcal{S}} = \{\mathrm{SK}_{\mathrm{id}}, \mathrm{SK}_{\mathrm{id,att}}\}_{\mathrm{att}\in\mathcal{S}}$ *associated with an authorized set of attributes, i.e.* $\mathbb{A} \models \mathcal{S}$*, and outputs plaintext* $m$*. Otherwise, it aborts.*
- MKDecrypt(MK, CT) $\rightarrow$ $m$*: This deterministic algorithm takes as input a ciphertext* CT *and the master-key* MK*, and outputs plaintext* $m$.

*The scheme is called correct if decryption outputs the correct message for a secret key associated with a set of attributes that satisfies the access structure.*

*In the single-authority setting (i.e.* $n = 1$*), the* GlobalSetup, MKSetup *and* AttSetup *are described in one* Setup*, and the* UKeyGen *and* AttKeyGen *have to be run in one* KeyGen*. In the multi-authority setting (i.e.* $n > 1$*), the* GlobalSetup *is run either jointly or by some additional central authority (CA).* MKSetup *can either be run distributively or independently by each* $\mathcal{A}_i$*.* AttSetup *can be run distributively or individually by* $\mathcal{A}_i$ *for the managed attributes* $\mathcal{U}_i$*.* UKeyGen *is run either distributively, individually for each* $\mathcal{A}_i$*, or implicitly (e.g. by using a hash).* AttKeyGen *is run by the* $\mathcal{A}_i$ *managing the set of attributes.* MKDecrypt *is typically run jointly by an authorized set of authorities.*

### 2.4 The security model and our attacks

We consider the notion of chosen-plaintext attack (CPA) security for ABE [5].

**Definition 2 (Full CPA-security for CP-ABE [5]).** *Let* $\mathfrak{C}$ = (GlobalSetup, ..., MKDecrypt) *be a CP-ABE scheme for authorities* $\mathcal{A}_1, ..., \mathcal{A}_n$ *(*$n \in \mathbb{N}$*) conform Definition 1. We define the game between challenger and attacker as follows.*

- ***Initialization phase:*** *The attacker corrupts a set* $\mathcal{I} \subsetneq \{1, ..., n\}$ *of authorities, and sends* $\mathcal{I}$ *to the challenger. In the selective security game, the attacker also commits to an access structure* $\mathbb{A}$*.*
- ***Setup phase:*** *The challenger runs the* GlobalSetup, MKSetup *for all authorities, and* AttSetup *for all attributes. It sends the global parameters* GP*, master public keys* $\{\mathrm{MPK}_{\mathrm{att}}\}_{\mathrm{att}\in\mathcal{U}}$*, and corrupted master secret keys* $\{\mathrm{MSK}_{\mathrm{att}}\}_{\mathrm{att}\in\mathcal{U}_\mathcal{I}}$ *to the attacker, where* $\mathcal{U}_\mathcal{I} = \bigcup_{i\in\mathcal{I}} \mathcal{U}_i$*.*
- ***Key query phase I:*** *The attacker queries secret keys for sets of attributes* $(\mathrm{id}_1, \mathcal{S}_1), ..., (\mathrm{id}_{n_1}, \mathcal{S}_{n_1})$*. The challenger runs* UKeyGen *and* AttKeyGen *for each* $(\mathrm{id}_j, \mathcal{S}_j)$ *and sends* $\mathrm{SK}_{\mathrm{id}_1,\mathcal{S}_1}$,...,$\mathrm{SK}_{\mathrm{id}_{n_1},\mathcal{S}_{n_1}}$ *to the attacker.*
- ***Challenge phase:*** *The attacker generates two messages* $m_0$ *and* $m_1$ *of equal length, together with an access structure* $\mathbb{A}$ *such that* $\mathcal{S}_j \cup \mathcal{U}_\mathcal{I}$ *does not satisfy* $\mathbb{A}$ *for all* $j$*. The challenger flips a coin* $b \in_R \{0, 1\}$ *and encrypts* $m_b$ *under* $\mathbb{A}$*. It sends the resulting challenge ciphertext* $\mathrm{CT}_\mathbb{A}$ *to the attacker.*
- ***Key query phase II:*** *The same as the first key query phase, with the restriction that the queried sets* $\mathcal{S}_{n_1+1}, ..., \mathcal{S}_{n_2}$ *are such that* $\mathbb{A} \not\models \mathcal{S}_j \cup \mathcal{U}_\mathcal{I}$*.*
- ***Decision phase:*** *The attacker outputs a guess* $b'$ *for* $b$*.*

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

We formally define our attacks in line with the chosen-plaintext attacks above and Figure 1, such that CPA-security also implies security against these attacks. Conversely, the ability to find such attacks implies insecurity in this model. While this follows intuitively, we prove this in Appendix B.

**Definition 3 (Master-key attacks (MKA)).** *We define the game between challenger and attacker as follows. First, the initialization, setup and first key query phases are run as in Definition 2. Then:*

- ***Decision phase:*** *The attacker outputs* $\mathrm{MK}'$*.*

*The attacker wins the game if for all messages* $m$*, decryption of ciphertext* $\mathrm{CT} \leftarrow \mathrm{Encrypt}(m, ...)$ *yields* $m' \leftarrow \mathrm{MKDecrypt}(\mathrm{MK}', \mathrm{CT})$ *such that* $m = m'$*.*

**Definition 4 (Attribute-key attacks (AKA)).** *We define the game between challenger and attacker as follows. First, the initialization, setup and first key query phases are run as in Definition 2. Then:*

- ***Decision phase:*** *The attacker outputs* $\mathrm{SK}_{\mathcal{S}'}$*, where* $\mathcal{S}' \supsetneq \mathcal{S}_j$ *for all* $j \in \{1, ..., n_1\}$*, and* $\mathcal{S}' \supseteq \bigcup_{j=1}^{n_1} \mathcal{S}_j$*.*

*The attacker wins the game if* $\mathrm{SK}_{\mathrm{id}',\mathcal{S}'}$ *is a valid secret key for some arbitrary identifier* $\mathrm{id}'$ *and set* $\mathcal{S}'$*.*

**Definition 5 (Decryption attacks (DA)).** *We define the game between challenger and attacker as follows. First, the initialization, setup, first key query and challenge phases are run as in Definition 2. Then:*

– ***Decision phase:*** *The attacker outputs plaintext* $m'$.

*The attacker wins the game if* $m' = m$. *A decryption attack is conditional if* $\mathbb{A} \models \bigcup_{j=1}^{n_1} \mathcal{S}_j$. *Otherwise, it is complete.*

# 3 Our methodology

Our methodology consists of a systemized approach to finding attacks on a scheme. First, we review a common structure of many ABE schemes, which implies a more concise notation for schemes to simplify cryptanalysis (Section 3.1). Then, we consider the effect of learning one or more values 'in the exponent', e.g. by corrupting an authority, and how such knowledge can be modeled in any attacks (Section 3.2). Using this, we formally define our attacks (Sections 3.3 and 3.4). Finally, we describe a heuristic approach that should simplify the effort of finding attacks (Section 3.5).

## 3.1 The standard form implies a more concise notation

Many pairing-based ABE schemes have a similar form. This form is explicitly defined and used in generic frameworks that simplify the design and analysis of ABE [39,3]. Our goal is to simplify the *cryptanalysis* of ABE. We have adapted the definition to match our own extended definition of CP-ABE (Definition 1).

**Definition 6 (Standard form of CP-ABE).** *The standard form of a CP-ABE scheme is defined as follows. Let* $\mathcal{A}_1, ..., \mathcal{A}_n$ *be some authorities (where* $n \in \mathbb{N}$*) such that each* $\mathcal{A}_i$ *manages universe* $\mathcal{U}_i$*, and* $\mathcal{U} = \bigcup_{i=1}^{n} \mathcal{U}_i$ *denotes the collective universe of attributes.*

– $\text{GlobalSetup}(\lambda)$*: This algorithm generates three groups* $\mathbb{G}, \mathbb{H}, \mathbb{G}_T$ *of prime order* $p$ *with generators* $g \in \mathbb{G}, h \in \mathbb{H}$*, and chooses a pairing* $e \colon \mathbb{G} \times \mathbb{H} \to \mathbb{G}_T$*. It may also select* ***attribute-independent common variables*** $\mathbf{b} \in_R \mathbb{Z}_p$*. It publishes the global parameters*

$$\text{GP} = (p, \mathbb{G}, \mathbb{H}, \mathbb{G}_T, g, h, \mathcal{U}, g^{\mathbf{gp}(\mathbf{b})}),$$

*where we refer to* $\mathbf{gp}$ *as the* ***global parameter encoding****.*

– $\text{MKSetup}(\text{GP})$*: This algorithm selects* $\alpha \in_R \mathbb{Z}_p$*, updates* $\text{GP} \leftarrow \text{GP} \cup \{e(g,h)^\alpha\}$*, publishes* GP *and keeps master-key* $\text{MK} = \alpha$ *secret.*

– $\text{AttSetup}(\text{att}, \text{MK}, \text{GP})$*: This algorithm selects integers* $\mathbf{b}_{\text{att}} \in_R \mathbb{Z}_p$ *as secret* $\mathbf{MSK}_{\text{att}} = \mathbf{b}_{\text{att}}$*, and publishes*

$$\mathbf{MPK}_{\text{att}} = g^{\mathbf{mpk}_a(\mathbf{b}_{\text{att}}, \mathbf{b})},$$

*where we refer to* $\mathbf{mpk}_a$ *as the* ***master attribute-key encoding****.*

– $\text{UKeyGen}(\text{id}, \text{MK}, \text{GP})$*: This algorithm selects user-specific random integers* $\mathbf{r}_u \in_R \mathbb{Z}_p$ *and computes partial user-key*

$$\text{SK}_{\text{id}} = h^{\mathbf{k}_u(\text{id}, \alpha, \mathbf{r}_u, \mathbf{b})},$$

*where we refer to* $\mathbf{k}_u \neq 0$ *as the* ***user-key encoding****.*

– $\mathrm{AttKeyGen}(\mathcal{S}, \mathrm{GP}, \mathrm{MK}, \mathrm{SK}_{\mathrm{id}}, \{\mathrm{MSK}_{\mathrm{att}}\}_{\mathrm{att}\in\mathcal{S}})$*: Parse* $\mathrm{SK}_{\mathrm{id}} = (h_{\mathrm{id},1}, h_{\mathrm{id},2}, ...)$*. This algorithm selects user-specific random integers* $\mathbf{r}_a \in_R \mathbb{Z}_p$ *and computes a key* $\mathrm{SK}_{\mathrm{id},\mathcal{S}} = \{\mathrm{SK}_{\mathrm{id,att}}\}_{\mathrm{att}\in\mathcal{S}}$*, such that for all* $\mathrm{att} \in \mathcal{S}$

$$\mathrm{SK}_{\mathrm{id,att}} = (h_{\mathrm{id},1}^{\mathbf{k}_{a,1}(\mathrm{att},\mathbf{r}_a,\mathbf{b},\mathbf{b}_{\mathrm{att}})}, h_{\mathrm{id},2}^{\mathbf{k}_{a,2}(\mathrm{att},\mathbf{r}_a,\mathbf{b},\mathbf{b}_{\mathrm{att}})}, ...),$$

*where we refer to* $\mathbf{k}_{a,i}$ *as the* ***user-specific attribute-key encodings****.*

– $\mathrm{Encrypt}(m, \mathbb{A}, \mathrm{GP}, \{\mathrm{MPK}_{\mathrm{att}}\}_{\mathrm{att}\sim\mathbb{A}})$*: This algorithm selects ciphertext-specific randoms* $\mathbf{s} = (s, s_1, s_2, ...) \in_R \mathbb{Z}_p$ *and outputs the ciphertext*

$$\mathrm{CT}_{\mathbb{A}} = (\mathbb{A}, m \cdot e(g,h)^{\alpha s}, g^{\mathbf{c}(\mathbb{A},\mathbf{s},\mathbf{b})}, g^{\mathbf{c}_a(\mathbb{A},\mathbf{s},\mathbf{b},\{\mathbf{b}_{\mathrm{att}}\}_{\mathrm{att}\sim\mathbb{A}})}),$$

*where we refer to* $\mathbf{c}$ *as the* ***attribute-independent ciphertext encoding****, and* $\mathbf{c}_a$ *the* ***attribute-dependent ciphertext encoding****. In particular, we assume that each entry of* $\mathbf{c}_a$ *uses a variable in* $\mathbf{b}_{\mathrm{att}}$ *for some* att*.*

– $\mathrm{Decrypt}((\mathrm{SK}_{\mathrm{id}}, \mathrm{SK}_{\mathrm{id},\mathcal{S}}), \mathrm{CT}_{\mathbb{A}})$*: Let* $\mathrm{SK}_{\mathrm{id}} = h^{\mathbf{k}_u(\mathrm{id},\alpha,\mathbf{r}_u,\mathbf{b})} = (h_{\mathrm{id},1}, ...)$*,* $\mathrm{SK}_{\mathrm{id},\mathcal{S}} = \{(h_{\mathrm{id},1}^{\mathbf{k}_{a,1}(\mathrm{att},\mathbf{r}_{a,i},\mathbf{b},\mathbf{b}_{\mathrm{att}})}, h_{\mathrm{id},2}^{\mathbf{k}_{a,2}(\mathrm{att},\mathbf{r}_{a,i},\mathbf{b},\mathbf{b}_{\mathrm{att}})}, ...)\}_{i\in\{1,...,n\},\mathrm{att}\in\mathcal{S}\cap\mathcal{U}_i}$*, and* $\mathrm{CT}_{\mathbb{A}} = (\mathbb{A}, C = m \cdot e(g,h)^{\alpha s}, \mathbf{C} = g^{\mathbf{c}(\mathbb{A},\mathbf{s},\mathbf{b})}, \mathbf{C}_a = g^{\mathbf{c}_a(\mathbb{A},\mathbf{s},\mathbf{b},\{\mathbf{b}_{\mathrm{att}}\}_{\mathrm{att}\sim\mathbb{A}})})$*. Denote* $\mathcal{S}_{\mathbb{A}} = \{\mathrm{att} \sim \mathbb{A} \mid \mathrm{att} \in \mathcal{S}\}$*. Define matrices* $\mathbf{E}$*,* $\mathbf{E}_{\mathrm{att},\mathcal{S},\mathbb{A}}$ *for each* $\mathrm{att} \in \mathcal{S}$ *such that*

$$\mathbf{cEk}_u^{\intercal} + \sum_{\mathrm{att}\in\mathcal{S}_{\mathbb{A}}} (\mathbf{c} \mid \mathbf{c}_a)\mathbf{E}_{\mathrm{att},\mathcal{S},\mathbb{A}}(\mathbf{k}_u \mid \mathbf{k}_a)^{\intercal} = \alpha s.$$

*In particular, we assume that* $\mathbf{E}_{\mathrm{att},\mathcal{S},\mathbb{A}}$ *only has non-zero entries if the associated key or ciphertext encoding depends on* att*. Then the plaintext* $m$ *can be retrieved by recovering* $e(g,h)^{\alpha s}$ *from* $\mathbf{C}, \mathbf{C}_a$ *and* $\mathrm{SK}_{\mathrm{id}}, \mathrm{SK}_{\mathrm{id},\mathcal{S}}$*, and*

$$m = C/e(g,h)^{\alpha s}.$$

– $\mathrm{MKDecrypt}(\mathrm{MK}, \mathrm{CT})$*: Let* $\mathrm{MK} = \alpha$*,* $\mathrm{MK}' = h^{\mathrm{mk}(\alpha,\mathbf{b})}$ *and* $\mathrm{CT} = (C = m \cdot e(g,h)^{\alpha s}, \mathbf{C} = g^{\mathbf{c}(\mathbb{A},\mathbf{s},\mathbf{b})}, \mathbf{C}_a = g^{\mathbf{c}_a(\mathbb{A},\mathbf{s},\mathbf{b},\{\mathbf{b}_{\mathrm{att}}\}_{\mathrm{att}\sim\mathbb{A}})})$*. Define vector* $\mathbf{e}$ *such that* $\mathbf{ce}\mathrm{mk} = \alpha s$*. Then* $m$ *can be retrieved by computing*

$$C/\prod_{\ell} e(C_\ell, \mathrm{MK}')^{e_\ell},$$

*where* $C_\ell$ *and* $e_\ell$ *denote the* $\ell$*-th entry of* $\mathbf{C}$ *and* $\mathbf{e}$*, respectively.*

*The scheme is correct if it holds that* $\alpha s$ *can be retrieved if the secret key associated with* $\mathcal{S}$ *and ciphertext associated with* $\mathbb{A}$ *are such that* $\mathbb{A} \models \mathcal{S}$*.*

*Each encoding* $\mathbf{enc}(\mathrm{var})$ *denotes a vector of polynomials over variables* var*. Depending on the scheme,* MKSetup *may be run distributively or by a CA (in which case there is only one public key* $e(g,h)^{\alpha}$ *associated with the master-keys), or independently and individually (in which case there are multiple public keys* $e(g,h)^{\alpha_i}$*, and we replace the blinding value* $e(g,h)^{\alpha s}$ *by* $e(g,h)^{\sum_{i\in\mathcal{I}}\alpha_i s}$*).*

*Remark 1.* Generators constructed by hash functions [5] are also included in this definition, i.e. by assuming that $\mathcal{H}(\mathrm{att}) = g^{b_{\mathrm{att}}}$ for some implicit $b_{\mathrm{att}}$.

This standard form implies a concise notation. Only the encodings are distinct and therefore sufficient to consider for cryptanalysis. As such, we provide our attacks in terms of encodings rather than group elements.

### 3.2 Modeling knowledge of exponents – extending $\mathbb{Z}_p$

The previously defined notation describes the relationship between the various variables 'in the exponent' of the keys and ciphertexts. The values of most variables are unknown to the attacker. In multi-authority ABE, authorities provide the inputs to some encodings, and therefore know some values, and most importantly, their (part of the) master-key. Hence, corruption of authorities results in the knowledge of some values 'in the exponent'. If the values provided by honest authorities are not well-hidden, it might enable an attack on them.

We model the 'knowledge of exponents' in attacks by extending the space from which the entries of $\mathbf{E}$ and $\mathbf{E}_{\text{att},\mathcal{S},\mathbb{A}}$ are chosen: $\mathbb{Z}_p$ (or some extension with variables associated with $\mathcal{S}$ and $\mathbb{A}$). In fact, the entries of these matrices may be any fraction of polynomials over $\mathbb{Z}_p$ and the known exponents. Let $\mathfrak{K}$ be the set of known exponents, then the extended field of rational fractions is defined as

$$\mathbb{Z}_p(\mathfrak{K}) = \{ab^{-1} \pmod p \mid a, b \in \mathbb{Z}_p[\mathfrak{K}]\},$$

where $\mathbb{Z}_p[\mathfrak{K}]$ denotes the polynomial ring of variables $\mathfrak{K}$.

### 3.3 Formal definitions of the attacks in the concise notations

We formally define our attacks (Definitions 7–9) in the concise notation. For each attack, $\mathfrak{K}$ denotes the set of known variables. We use the following shorthand for a key encoding for a user id with set $\mathcal{S}$ and for a ciphertext encoding for access structure $\mathbb{A}$:

$$\mathbf{k}_{\text{id},\mathcal{S}} := (\mathbf{gp}(\mathbf{b}), \mathbf{mpk}_a(b_{\text{att}}, \mathbf{b}), \mathbf{k}_u(\text{id}, \alpha, \mathbf{r}_u, \mathbf{b}) \mid \mathbf{k}_{a,1}(\text{att}, \mathbf{r}_a, \mathbf{b}, \mathbf{b}_{\text{att}}) \mid ...),$$
$$\mathbf{c}_{\mathbb{A}} := (\mathbf{gp}(\mathbf{b}), \mathbf{mpk}_a(b_{\text{att}}, \mathbf{b}), \mathbf{c}(\mathbb{A}, \mathbf{s}, \mathbf{b}) \mid \mathbf{c}_a(\mathbb{A}, \mathbf{s}, \mathbf{b}, \{\mathbf{b}_{\text{att}}\}_{\text{att}\sim\mathbb{A}})).$$

We first define the master-key attacks. In a master-key attack, the attacker has to retrieve the master-key $\text{m}(\alpha, \mathbf{b})$ such that any ciphertext can be decrypted with MKDecrypt.

**Definition 7 (Master-key attacks).** *A scheme is vulnerable to a master-key attack if there exist* $(\text{id}_1, \mathcal{S}_1), ..., (\text{id}_{n_1}, \mathcal{S}_{n_1})$ *and the associated key encodings* $\mathbf{k}_{\text{id}_i,\mathcal{S}_i}$*, then there exist* $\mathbf{e}_i \in \mathbb{Z}_p(\mathfrak{K})^{\ell_i}$*, where* $\ell_i = |\mathbf{k}_{\text{id}_i,\mathcal{S}_i}|$*, such that* $\sum_i \mathbf{k}_i \mathbf{e}_i^\intercal = \text{mk}(\alpha, \mathbf{b}) \in \mathbb{Z}_p(\alpha, \mathbf{b})$*. Then, it holds that for all attribute-independent ciphertext encodings* $\mathbf{c}$ *there exists* $\mathbf{e}' \in \mathbb{Z}_p^{\ell'}$ *(with* $|\mathbf{c}| = \ell'$*) such that* $\text{mk}\mathbf{e}'\mathbf{c}^\intercal = \alpha s$*.*

We formally define attribute-key attacks. In an attribute-key attack, the attacker has to generate a secret key associated with a set $\mathcal{S}'$ that is strictly larger than any of the sets $\mathcal{S}_i$ associated with the issued keys.

**Definition 8 (Attribute-key attacks).** *A scheme is vulnerable to an attribute-key attack if there exist* $(\text{id}_1, \mathcal{S}_1), ..., (\text{id}_{n_1}, \mathcal{S}_{n_1})$ *such that for the key encodings* $\mathbf{k}_{\text{id}_i,\mathcal{S}_i}$*, it holds that* $\overline{\mathbf{k}}_{\text{id}',\mathcal{S}'}$ *(with user-specific randoms* $\overline{\mathbf{r}}_u$ *and* $\overline{\mathbf{r}}_a$ *constructed linearly from the other user-specific randoms) can be generated such that* $\bigcup_{i=1}^{n_1} \mathcal{S}_i \subseteq \mathcal{S}'$ *and* $\mathcal{S}_i \subsetneq \mathcal{S}'$ *for all* $i \in \{1, ..., n_1\}$*.*

We formally define the complete and conditional decryption attacks. A decryption attack takes as input a ciphertext and a polynomial number of unauthorized keys and outputs the plaintext. The attack is conditional if the collective set of attributes satisfies the access structure associated with the ciphertext. Otherwise it is complete, because it implies that any set of keys can be generated, which can then decrypt any ciphertext.

**Definition 9 (Complete/conditional decryption attacks).** *A scheme is vulnerable to a decryption attack if there exist* $(\mathrm{id}_1, \mathcal{S}_1), ..., (\mathrm{id}_{n_1}, \mathcal{S}_{n_1})$ *and* $\mathbb{A}$ *such that* $\mathbb{A} \not\models \mathcal{S}_i$*, associated ciphertext encoding* $\mathbf{c}_{\mathbb{A}}$ *and key encodings* $\mathbf{k}_{\mathrm{id}_i, \mathcal{S}_i}$*, for which there exist* $\mathbf{E}_i \in \mathbb{Z}_p(\mathfrak{K})^{\ell_i \times \ell'}$*, where* $\ell_i = |\mathbf{k}_{\mathrm{id}_i, \mathcal{S}_i}|$ *and* $\ell' = |\mathbf{c}_{\mathbb{A}}|$*, such that* $\sum_i \mathbf{k}_{\mathrm{id}_i, \mathcal{S}_i} \mathbf{E}_i \mathbf{c}_{\mathbb{A}}^{\intercal} = \alpha s$*. The attack is conditional if it holds that* $\mathbb{A} \models \bigcup_i \mathcal{S}_i$*. Otherwise, it is complete.*

It readily follows that master-key and attribute-key attacks imply decryption attacks. Specifically, master-key attacks and attribute-key attacks for which $\bigcup_{i=1}^{n} S_i \subsetneq \mathcal{S}'$ holds imply complete decryption attacks.

Finally, we formally define attacks on the correctness of a scheme. That is, for some schemes it holds that there exist a ciphertext and an authorized key such that the ciphertext cannot be decrypted. In a sense, such attacks are orthogonal to decryption attacks.

**Definition 10 (Correctness attacks).** *A scheme is incorrect if there exist* $\mathrm{id}, \mathcal{S}$ *and* $\mathbb{A}$ *such that* $\mathbb{A} \models \mathcal{S}$ *for which there exists no* $\mathbf{E} \in \mathbb{Z}_p(\mathfrak{K})^{\ell_1 \times \ell_2}$*, where* $\ell_1 = |\mathbf{k}_{\mathrm{id}, \mathcal{S}}|$ *and* $\ell_2 = |\mathbf{c}_{\mathbb{A}}|$ *such that* $\mathbf{k}_{\mathrm{id}, \mathcal{S}} \mathbf{E} \mathbf{c}_{\mathbb{A}}^{\intercal} = \alpha s$*.*

### 3.4 Definitions of multi-authority-specific attacks

The multi-authority setting yields two additional difficulties in the design of secure schemes. First, the corruption of authorities yields extra knowledge about the exponent space. Second, the distributed structure of the master-key may enable new attacks. Formally, we define attacks under corruption as follows.

**Definition 11 (Attacks under corruption).** *A scheme is vulnerable to attacks under corruption if an attacker can corrupt a subset* $\mathcal{I} \subsetneq \{1, ..., n\}$ *of authorities* $\mathcal{A}_1, ..., \mathcal{A}_n$ *(where* $n = \mathrm{poly}(n)$*) and thus obtain knowledge of variables* $\mathfrak{K}$ *consisting of all variables and (partial) encodings generated by the corrupt authorities, enabling an attack conform Definitions 7, 8 or 9.*

Oftentimes, the master-key is generated distributively by the KGAs, hence the blinding value is of a distributed form, e.g. $e(g,h)^{\alpha s} = e(g,h)^{\sum_i \alpha_i s}$. If each partial blinding value e.g. $e(g,h)^{\alpha_i s}$ can be recovered independently of the user's randomness, then the scheme is vulnerable to a multi-authority-specific decryption attack. For instance, suppose the blinding value is defined as $(\alpha_1 + \alpha_2)s$. If one user can recover $\alpha_1 s$ (but not $\alpha_2 s$) and another user can recover $\alpha_2 s$ (but not $\alpha_1 s$), then the scheme is vulnerable to a multi-authority-specific decryption attack. They can collectively recover $(\alpha_1 + \alpha_2)s$, while clearly, they cannot do this individually.

**Definition 12 (Multi-authority-specific (MAS) decryption attacks).** *A scheme is vulnerable to a multi-authority-specific decryption attack if the blinding value of the message is of the form* $\sum_i \mathrm{bv}_i(\alpha_i, \mathbf{s}, \mathbf{b})$*, where* $\alpha_i$ *denotes the master-key of authority* $\mathcal{A}_i$*, and* $\mathrm{bv}_i$ *are elements in* $\mathbb{G}_T$*. If there exist ciphertext encoding* $\mathbf{c}_{\mathbb{A}}$ *and sets* $\mathcal{S}_i \subseteq \mathcal{U}_i$ *with key encodings* $\mathbf{k}_{\mathrm{id}_i,\mathcal{S}_i}$ *for which there exist* $\mathbf{E}_i \in \mathbb{Z}_p(\mathfrak{K})^{\ell_i \times \ell'}$*, where* $\ell_i = |\mathbf{k}_{\mathrm{id}_i,\mathcal{S}_i}|$ *and* $\ell' = |\mathbf{c}_{\mathbb{A}}|$*, such that* $\mathbf{k}_{\mathrm{id}_i,\mathcal{S}_i}\mathbf{E}_i\mathbf{c}_{\mathbb{A}}^{\intercal} = \mathrm{bv}_i$*.*

*Remark 2.* A multi-authority-specific decryption attack is also a decryption attack conform Definition 9. That is, the blinding value can be retrieved, even though the individual sets are not authorized to decrypt the ciphertext. Conversely, such attacks do not exist in the single-authority setting, meaning that they are strictly weaker than regular (single-authority) decryption attacks.

*Remark 3.* The attacks of Wang et al. [37] on the HSM+14 [14] scheme—and by extension the HSM+15 [15] scheme—are examples of multi-authority specific decryption attacks.

### 3.5 Our heuristic approach

We devise a targeted approach to finding attacks, which can be applied manually. As the definitions in the previous section imply, finding an attack is equivalent to finding a suitable linear combination—where the linear coefficients are the entries of $\mathbf{e}$ or $\mathbf{E}$—of all products of the key and ciphertext entries. While finding such coefficients is relatively simple, we note that the inputs of the attacks are variable. The number of colluding users and the number of attributes associated with the keys and ciphertexts are effectively unbounded. However, we observe that it often suffices to consider a limited number of inputs, and that for some attacks, only the user-key and attribute-independent ciphertext entries need to be considered. Specifically, Table 3 describes these inputs in terms of encodings, the sets of attributes, and the access policy. Depending on the maximum number of monomials consisting of common variables in any key entry, the attacker might need multiple secret keys for the same set of attributes to recover certain coefficients. For instance, suppose the attacker wants to retrieve $\alpha$ from $\alpha + r_1 b_{\mathrm{att}_1} + r_1' b_{\mathrm{att}_1}'$, where $r_1$ and $r_1'$ are known, user-specific random variables, and $b_{\mathrm{att}_1}$ and $b_{\mathrm{att}_1}'$ denote the common variables associated with attribute $\mathrm{att}_1$. Because of the three unknown, linearly independent monomials, this can only be done if the attacker has three distinct keys for attribute $\mathrm{att}_1$. In general, the maximum number of keys with the same set of attributes can be determined in this way, i.e. by counting the maximum number of linearly independent monomials for each entry.

Similarly, the inputs to multi-authority specific attacks can be limited. First, we consider the attacks under corruption. Corruption of any number of authorities results in the additional knowledge of some otherwise hidden exponents, i.e. the master keys and any random variables generated by these authorities. We assume that it is sufficient to consider one corrupted and one honest authority in the attacks. Further, we use the same descriptions of the inputs to the attacks as in the single-authority setting, with the additional requirement that the

**Table 3.** The inputs of the attacks, and which encodings are needed.

| Attack | Secret keys | | | Ciphertexts | | |
|---|---|---|---|---|---|---|
| | UK | AK | $\mathcal{S}$ | AI | AD | $\mathbb{A}$ |
| Master-key | ✓ | ✗ | - | ✗ | ✗ | - |
| Attribute-key | ✓ | ✓ | $\mathcal{S}_1 = \{\text{att}_1\}, \mathcal{S}_2 = \{\text{att}_2\}$ | ✗ | ✗ | - |
| Complete decryption | ✓ | ✗ | - | ✓ | ✗ | - |
| Conditional decryption | ✓ | ✓ | $\mathcal{S}_1 = \{\text{att}_1\}, \mathcal{S}_2 = \{\text{att}_2\}$ | ✓ | ✓ | $\mathbb{A} = \text{att}_1 \wedge \text{att}_2$ |

UK, AK = user-, attribute-key; AI, AD = attribute-independent, -dependent

**Table 4.** The number of required honest authorities $n$ and the attribute universes $\mathcal{U}_1$ and $\mathcal{U}_2$ managed by authorities $\mathcal{A}_1$ and $\mathcal{A}_2$ in the multi-authority setting.

| Attack | $n$ | $\mathcal{U}_1$ | $\mathcal{U}_2$ |
|---|---|---|---|
| Master-key | 1 | ✗ | ✗ |
| Attribute-key | 1 | $\{\text{att}_1, \text{att}_2\}$ | ✗ |
| Complete decryption | 1 | ✗ | ✗ |
| Conditional decryption | 1 | $\{\text{att}_1, \text{att}_2\}$ | ✗ |
| MAS-decryption | 2 | $\{\text{att}_1\}$ | $\{\text{att}_2\}$ |

input attributes are managed by the honest authority. Second, we consider multi-authority specific decryption attacks. Corruption is not necessary in this setting, so we assume that the authorities are honest. Additionally, we require at least two honest authorities as input to finding any attack, so we let each authority manage one attribute. Table 4 summarizes the additional inputs to the attacks in Table 3. Finally, it may be possible that a semi-honest central authority (CA) is part of the scheme, in which case we also consider whether corruption of this CA enables an attack. A semi-honest CA is assumed to perform all algorithms as required, but it is allowed to be corrupted/collude passively.

We describe a more targeted approach to finding an attack, i.e. the linear coefficients $\mathbf{e}$ and $\mathbf{E}$, given the input encodings. The approach to finding an attack is linear, as we attempt to retrieve the desired output (conform Definitions 7, 8 and 9) by making linear combinations of products of encodings. For the master-key and decryption attacks, the goal is to retrieve master-key $\alpha$, or blinding value $\alpha s$ (or e.g. $(\sum_i \alpha_i)s$ in some multi-authority schemes). Typically, $\alpha$ occurs only in one entry of the keys, while $s$ occurs only in one entry of the ciphertext. Instead of blindly trying all combinations of the key entries with the ciphertext, we formulate a more targeted approach. First, consider the monomials to be canceled, and then which combinations of the key and ciphertext entries can make these monomials. In canceling the previous monomials, it might be the case that new monomials are added, meaning that these in turn also need to be canceled. This process repeats until all monomials are canceled, and $\alpha$ or $\alpha s$ remains—unless such attack cannot be found. For conciseness, we only provide the non-zero coefficients in an attack.

## 4 Detailed examples using the approach

Using examples, we illustrate the way in which our heuristic approach can be applied. In particular, we give several examples of attacks with or without corruption, correctness attacks and a multi-authority specific decryption attack.

### 4.1 Example without corruption: the YJR+13 [44] scheme

The YJR+13 [44] scheme, also known as DAC-MACS, is a multi-authority scheme that supports the revocation of keys. This revocation functionality was already broken in [16,40], but a fix for its revocation functionality was proposed in the latter [40]. We show that the basic scheme—which is also the basic scheme in the 'fixed version' [40]—is vulnerable to a complete decryption attack. The scheme consists of the following encodings, which are sufficient for an attack.

- **Type of attack:** Complete decryption attack;
- **Global parameters:** $\mathbf{gp} = (\mathrm{gp}_1, \mathrm{gp}_2, ...) = (b, 1/b', ...)$;
- **User-key:** $\mathbf{k}_u(\alpha, \mathbf{r}, \mathbf{b}) = (\alpha/x_1 + x_2b + rb/b', rb'/x_1, rb)$;
- **Attribute-independent ciphertext:** $\mathbf{c}(\mathbf{s}, \mathbf{b}) = (s, s/b')$;
- **Known exponents:** $\mathfrak{K} = \{x_1, x_2\}$ (by definition);

The exponents $x_1, x_2$ are known to any decrypting user to enable decryption. We show that knowing these exponents also enables an attack. That is, any decrypting user is trivially able to decrypt any ciphertext, without even considering the attribute-keys. First, we show that it is not possible to retrieve the master-key. We sample a user-key $(k_1, k_2, k_3) \leftarrow \mathbf{k}_u(\alpha, \mathbf{r}, \mathbf{b})$, and observe that master-key $\alpha$ only occurs in $k_1 = \alpha/x_1 + x_2b + rb/b'$. We can cancel $x_2b$, because $x_2$ is known and $\mathrm{gp}_1 = b$ is a global parameter. Unfortunately, we cannot cancel $rb/b'$, it can only be retrieved by combining $k_1$ and $\mathrm{gp}_2$. Second, using these observations, we show that it is possible to perform a decryption attack. We also sample $(c_1, c_2) \leftarrow \mathbf{c}(\mathbf{s}, \mathbf{b})$. To retrieve $\alpha s$, we start by pairing $k_1$ with $c_1$:

$$x_1k_1c_1 = \overbrace{\alpha s}^{\text{Blinding value}} + \overbrace{\underbrace{x_1x_2sb}_{x_1x_2\mathrm{gp}_1c_1} + \underbrace{x_1rsb/b'}_{x_1k_3c_2}}^{\text{to cancel}}$$

Hence, we use these observations to formulate the attack as follows:

$$\alpha s = \underbrace{(k_1, k_2, k_3, \mathrm{gp}_1, \mathrm{gp}_2)}_{\mathbf{k}_u} \underbrace{\begin{pmatrix} x_1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & -x_1 & 0 & 0 \\ -x_1x_2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}}_{\mathbf{E}} \underbrace{\begin{pmatrix} c_1 \\ c_2 \\ \mathrm{gp}_1 \\ \mathrm{gp}_2 \end{pmatrix}}_{\mathbf{c}}$$

$$= x_1k_1c_1 - x_1k_3c_2 - x_1x_2\mathrm{gp}_1c_1.$$

Note that most of the entries of $\mathbf{E}$ are zero, so for brevity, we only write down the non-zero entries of $\mathbf{E}$ below. □

### 4.2 Example without corruption: the JLWW13 [18] scheme

The JLWW13 [18] and JLWW15 [19] schemes, also known as AnonyControl, have the same key generation. Note that the JLWW15 [19] scheme is different from JLWW13 in the encryption algorithm. It is incorrect, because a value of a single user-specific secret key is used in the encryption algorithm. We also show that both schemes are vulnerable to a conditional attribute-key attack under collusion of two users. The encodings of the global parameters, secret keys (both the user-key and attribute-key parts) are defined as follows.

- **Type of attack:** Conditional attribute-key attack, collusion of two users;
- **Global parameters:** $\mathbf{gp} = (b, b'), \mathbf{mpk}_a(\text{att}_i) = b_{\text{att}_i}$;
- **Secret keys:** $\mathbf{k}_u(\alpha, r, \mathbf{b}) = (\alpha + r)$, $\mathbf{k}_a(\text{att}_i, r, r_i, \mathbf{b}) = (r_i b_{\text{att}_i} + r, r_i)$;

We show that the recurrence of $r$ as a monomial in the user-key and attribute-key encoding enables an attack. While it is relatively simple to show that this cannot be exploited in a single-user setting, we show that sampling two keys for two different sets of attributes $\mathcal{S}_1 = \{\text{att}_1\}$ and $\mathcal{S}_2 = \{\text{att}_2\}$ (as in Table 3) enables the generation of a third key for both attributes, i.e. $\mathcal{S}_3 = \{\text{att}_1, \text{att}_2\}$. For $\mathcal{S}_1 = \{\text{att}_1\}$, we sample $k \leftarrow \mathbf{k}_u(\alpha, r, \mathbf{b})$, and $(k_1, k_2) \leftarrow \mathbf{k}_a(\text{att}_1, r, r_1, \mathbf{b})$. For $\mathcal{S}_2 = \{\text{att}_2\}$, we sample $k' \leftarrow \mathbf{k}_u(\alpha, r', \mathbf{b})$, and $(k'_1, k'_2) \leftarrow \mathbf{k}_a(\text{att}_2, r', r_2, \mathbf{b})$.

The goal is to generate a key for set $\mathcal{S}_3 = \{\text{att}_1, \text{att}_2\}$. We aim to generate attribute-keys for the user-key associated with $\mathcal{S}_1$, i.e. $k$, which links the keys together with $r$. Of course, the attribute-key for attribute $\text{att}_1$ can also be readily used for $\mathcal{S}_3$. We generate the attribute-key for $\text{att}_2$ from the other key parts: $\mathbf{k}_a(\text{att}_2, r, r_2, \mathbf{b}) = (k'_1 + k - k', k'_2)$. □

### 4.3 Example with corruption: the YJ14 [43] scheme

The YJ14 [43] scheme is somewhat similar to the YJR+13 [44] scheme in the secret keys. However, the decrypting user knows fewer exponents: instead of sharing $x_2$ (in the YJR+13 scheme) with the user, it is shared with the attribute authorities. Hence, corruption of one authority leads to the knowledge of $x_2$, and thus enables an attack. We define the encodings and attack as follows.

- **Type of attack:** Complete decryption attack, under corruption of one $\mathcal{A}$;
- **Global parameters:** $\mathbf{gp} = (b, b')$;
- **Master secret key $\mathcal{A}_i$:** $\mathfrak{msk}_i = (\alpha_i, x)$;
- **User-key:** $\mathbf{k}_u(\alpha_i, \mathbf{r}, \mathbf{b}) = (\alpha_i + xb + rb', r)$;
- **Attribute-independent ciphertext:** $\mathbf{c}(\mathbf{s}, \mathbf{b}) = (s, sb', ...)$;
- **Blinding value:** $(\sum_i \alpha_i)s$;
- **Known variables:** $\mathfrak{K} = \{x\}$ (by corrupting $\mathcal{A}'$);
- **The goal:** Recover $\alpha_i s$ from $(k_{1,i}, k_{2,i}) \leftarrow \mathbf{k}_u(\alpha_i, \mathbf{r}, \mathbf{b})$, $(c_1, c_2) \leftarrow \mathbf{c}(\mathbf{s}, \mathbf{b})$;
- **The attack:** $\alpha_i s = k_{1,i}c_1 - k_{2,i}c_2 - x\text{mpk}_1 c_1$. □

### 4.4 Example of correctness attack: the QLZH15 [33] scheme

The QLZH15 [33] scheme is essentially the CP-ABE variant of the CC09 [7] scheme (or the multi-authority variant of [17]). We show that it is incorrect, and that it cannot be fixed without making it vulnerable to another attack. To ensure 'correctness', we assume that each authority has a dummy attribute for which every user receives a secret key.

*Remark 4.* For this scheme, it is important to know how we implement the access structures $\mathbb{A}$, e.g. $\mathbb{A} = \text{att}_1 \wedge \text{att}_2$. In particular, we mathematically implement a conjunction by computing two randoms $s_1, s_2 \in_R \mathbb{Z}_p$ associated with attributes $\text{att}_1$ and $\text{att}_2$ and set $s \leftarrow s_1 + s_2 \pmod p$ as the shared secret.

- **Type of attack:** Incorrect;
- **Global parameters:** $\mathbf{gp} = (b)$;
- **Master key pair of $\mathcal{A}_i$:** $\mathfrak{msk}_i(\text{att}) = (b_{\text{att}})$, $\mathbf{mpk}_i(\text{att}) = (b_{\text{att}})$;
- **Secret key:** $\mathbf{k}_u(\alpha, \mathbf{r}, \mathbf{b}) = (\alpha + (\sum r_i)b)$, $\mathbf{k}_a(\text{att}_i, r_i, \mathbf{b}) = (r_i b / b_{\text{att}_i})$;
- **Ciphertext:** $\mathbb{A} = \text{att}_1 \wedge \text{att}_2$, $\mathbf{c}(\mathbb{A}, s_1, s_2, \mathbf{b}) = (s_1 + s_2, s_1 b_{\text{att}_1}, s_2 b_{\text{att}_2})$;
- **Input (keys):** For $\mathcal{S} = \{\text{att}_1, \text{att}_2\}$ such that $\text{att}_1 \in \mathcal{U}_1$, $\text{att}_2 \in \mathcal{U}_2$, authority $\mathcal{A}_1$ and $\mathcal{A}_2$ securely and jointly generate $k \leftarrow \mathbf{k}_u(\alpha, r_1, r_2, \mathbf{b})$, where $r_1, r_2 \in_R \mathbb{Z}_p$ are generated by $\mathcal{A}_1$ and $\mathcal{A}_2$, respectively, and $k_i \leftarrow \mathbf{k}_a(\text{att}_i, r_i, \mathbf{b})$;
- **Input (ciphertext):** $\mathbb{A} = \text{att}_1 \wedge \text{att}_2$, $(c_1, c_2, c_3) \leftarrow \mathbf{c}(\mathbb{A}, s_1, s_2, \mathbf{b})$;
- **Blinding value:** $\alpha(s_1 + s_2)$;
- **The goal:** Showing that, even though $\mathbb{A} \models \mathcal{S}$, $\alpha(s_1+s_2)$ cannot be recovered;
- **The attack:** $\mathbb{A} \models \mathcal{S}$, so $\mathbf{k}$ should be able to decrypt $\mathbf{c}$, yet:

$$\begin{aligned} kc_1 - k_1 c_2 - k_2 c_3 &= \alpha(s_1 + s_2) + (r_1 + r_2)(s_1 + s_2)b - r_1 s_1 b - r_2 s_2 b \\ &= \alpha(s_1 + s_2) + r_1 s_2 b + r_2 s_1 b \neq \alpha(s_1 + s_2). \end{aligned}$$

We observe that the issue is that both the secret key and the ciphertext provide a different random for each attribute. To allow cancellation of $(r_1 + r_2)(s_1 + s_2)b$, either we require that the same random $s = s_1 + s_2$ on each attribute-dependent ciphertext element is used, or the same random $r = r_1 + r_2$ on each attribute-dependent secret key element is used. The next two subsections will discuss these modifications.

### 4.5 Example of a MAS-decryption attack: modified QLZH15 I

We modify the QLZH15 scheme such that it uses the same random $s$ for all ciphertext parts. That is, we define each access policy as $\mathbb{A} = \bigwedge_{\mathcal{A}_i} \mathbb{A}_i$ such that $\mathbb{A}_i$ consists only of attributes managed by $\mathcal{A}_i$. We show that this scheme is vulnerable to a multi-authority specific decryption attack.

- **Type of attack:** Multi-authority specific decryption attack;
- **Global parameters:** $\mathbf{gp} = (b)$;
- **Master key pair of $\mathcal{A}_i$:** $\mathfrak{msk}_i = (b_{\text{att}})$, $\mathbf{mpk}_i = (b_{\text{att}})$;
- **Secret key:** $\mathbf{k}_u(\alpha, \mathbf{r}, \mathbf{b}) = (\alpha + (\sum r_i)b)$, $\mathbf{k}_a(\text{att}_i, r_i, \mathbf{b}) = (r_i b / b_{\text{att}_i})$;

- **Ciphertext:** $\mathbb{A} = \text{att}_1 \wedge \text{att}_2$, $\mathbf{c}(\mathbb{A}, s, \mathbf{b}) = (s, sb_{\text{att}_1}, sb_{\text{att}_2})$;
- **Input (keys):** For $\mathcal{S}_1 = \{\text{att}_1\}$, and $\mathcal{S}_2 = \{\text{att}_2\}$ such that $\text{att}_1 \in \mathcal{U}_1$, $\text{att}_2 \in \mathcal{U}_2$. For $\mathcal{S}_1$, authority $\mathcal{A}_1$ and $\mathcal{A}_2$ securely and jointly generate $k \leftarrow \mathbf{k}_u(\alpha, r, \mathbf{b})$, where $r \in_R \mathbb{Z}_p$ is jointly generated by $\mathcal{A}_1$ and $\mathcal{A}_2$, and $k_1 \leftarrow \mathbf{k}_a(\text{att}_1, r, \mathbf{b})$. They also generate $k' \leftarrow \mathbf{k}_u(\alpha, r', \mathbf{b})$ and $k'_2 \leftarrow \mathbf{k}_a(\text{att}_2, r', \mathbf{b})$;
- **Input (ciphertext):** $\mathbb{A} = \text{att}_1 \wedge \text{att}_2$, $(c_1, c_2, c_3) \leftarrow \mathbf{c}(\mathbb{A}, s, \mathbf{b})$;
- **Blinding value:** $\alpha s$;
- **The goal:** Recover $\alpha s$ even though neither $\mathbb{A} \not\models \mathcal{S}_1$ and $\mathbb{A} \not\models \mathcal{S}_2$;
- **The attack:**

$$\begin{aligned} \alpha s &= \tfrac{1}{2}(kc_1 - k_1 c_2) + \tfrac{1}{2}(k'c_1 - k'_2 c_3) \\ &= \tfrac{1}{2}(\alpha s + rsb - rsb) + \tfrac{1}{2}(\alpha s + r'sb - r'sb). \end{aligned}$$

□

### 4.6 Example of other issues: modified QLZH15 II and III

We present two modifications of the QLZH15 scheme such that it uses the same random $r$ for all keys (like in [17] and [38], respectively). We show that the first modification has vulnerabilities under corruption. For the second modification, we use Chow's [10] multi-authority version of Wat11 [38], which uses his generalization of the Chase-Chow [7] approach. However, we point out that there might be some other vulnerabilities that may be exploitable.

*Modification II:* Naively, we modify QLZH15 such that it uses the same random $r$ for all key elements. We show that this proposed modification cannot be secure against corruption, regardless of how the key generation is implemented.

- **Type of attack:** Complete master-key attack under corruption;
- **Global parameters:** $\mathbf{gp} = (b)$;
- **Master key pair of $\mathcal{A}_i$:** $\mathfrak{msk}_i = (b_{\text{att}})_{\text{att} \in \mathcal{U}_i}$, $\text{mpk}_i = (b_{\text{att}})_{\text{att} \in \mathcal{U}_i}$;
- **Secret key:** $\mathbf{k}_u(\alpha, \mathbf{r}, \mathbf{b}) = (\alpha + rb)$, $\mathbf{k}_a(\text{att}, r, \mathbf{b}) = (rb/b_{\text{att}})$;
- **Known variables:** $\mathfrak{K} = \{b_{\text{att}_1}\}$ (by corrupting $\mathcal{A}_1$);
- **The goal:** Recover $\alpha$ from $k \leftarrow \mathbf{k}_u(\alpha, \mathbf{r}, \mathbf{b})$, $k_1 \leftarrow \mathbf{k}_a(\text{att}_1, r, \mathbf{b})$;
- **The attack:** $\alpha = k - b_{\text{att}_1} k_1$. □

*Modification III:* This is the multi-authority version of the Wat11 [38] scheme as presented by Chow [10]. In particular, the key generation is run in two rounds: the first round involves the user-key and the second round the attribute-keys.

- **Type of attack:** Exploit of two-phase key generation;
- **Global parameters:** $\mathbf{gp} = (b)$;
- **Master key pair of $\mathcal{A}_i$:** $\mathfrak{msk}_i = (b_{\text{att}})_{\text{att} \in \mathcal{U}_i}$, $\text{mpk}_i = (b_{\text{att}})_{\text{att} \in \mathcal{U}_i}$;
- **Secret key round I:** $\mathbf{k}_u(\alpha, \mathbf{r}, \mathbf{b}) = (\alpha + rb, r)$;
- **Secret key round II:** $\mathbf{k}_a(\text{att}, r, \mathbf{b}) = (rb_{\text{att}})$ (generated from $r$);
- **Ciphertext:** $\mathbb{A} = \text{att}_1 \wedge \text{att}_2$, $\mathbf{c}(s_1, s_2, \mathbf{b}) = (s_1 + s_2)$, $\mathbf{c}_a(\text{att}_i, s_i, s'_i, \mathbf{b}) = (s_i b + s'_i b_{\text{att}_i}, s'_i)$;

- **Blinding value:** $\alpha s$;
- **Input:** For $\mathcal{S}_1 = \{\text{att}_1\}$, we run the key generation as defined: $(k_1, k_2) \leftarrow \mathbf{k}_u(\alpha, r, \mathbf{b})$, $k_3 \leftarrow \mathbf{k}_a(\text{att}_1, r, \mathbf{b})$. Define $\mathbb{A} = \text{att}_1 \wedge \text{att}_2$;
- **The goal:** Recover $\alpha(s_1 + s_2)$ from $(k_1, k_2, k_3)$, $c \leftarrow \mathbf{c}(\mathbb{A}, s_1, s_2, \mathbf{b})$, $(c_i, c'_i) \leftarrow \mathbf{c}_a(\text{att}_i, s_i, s'_i, \mathbf{b})$ and $\mathcal{S}_2 = \{\text{att}_2\}$;
- **The attack:** For $\mathcal{S}_2$, we initiate the first round of the key generation: $(k'_1, k'_2) \leftarrow \mathbf{k}_u(\alpha, r', \mathbf{b})$. However, for the second round, we give as input $c'_2$ instead of $k'_2$. So $k'_3 \leftarrow \mathbf{k}_a(\text{att}_2, s'_2, \mathbf{b}) = s'_2 b_{\text{att}_2}$. Then

$$\begin{aligned}\alpha(s_1 + s_2) &= k_1 c - k_2 c_1 + k_3 c'_1 - c_2 + k'_3 \\ &= \alpha(s_1 + s_2) + r(s_1 + s_2)b - rs_1 b - rs'_1 b_{\text{att}_1} + rs'_1 b_{\text{att}_1} \\ &\quad -s_2 b - s'_2 b_{\text{att}_2} + s'_2 b_{\text{att}_2}.\end{aligned}$$

□

This attack can be averted by ensuring that the second round of the key generation only allows actual outputs of the first round as input. For instance, these outputs can be signed by all authorities, such that the second round can only be instantiated if the given input verifies correctly. Furthermore, $r$ needs to be generated such that none of the authorities can *actively* cheat (e.g. by ensuring that $r$ is $s'_i$ like in the attack). Hence, this scheme is considerably more difficult to secure than schemes with a one-round key generation such as [22].

## 5 Attacks on several schemes

We present attacks on several existing schemes (some of which were already shown in Section 4). For each scheme, we describe the secret keys, and possibly the global parameters and master keys, the ciphertext, and the form of the blinding value in the concise notation introduced in Section 3.1. Furthermore, we show whether collusion between users and corruption of any entities are required for the attack. Such corruption results in extra knowledge of exponents, so $\mathbb{Z}_p$ is extended with the known variables conform Section 3.2.

### 5.1 Single-authority ABE

*The ZH10 [47] and ZHW13 [48] schemes:* In these schemes, three generators are defined for each attribute att, indicating the presence (att), or absence (¬att) of an attribute, and a dummy value of the attribute ∗att. For each user, the secret key consists of a part associated with the positive or negative attribute (depending on whether att is in the user's possession) and the dummy value.

- **Type of attack:** Conditional attribute-key attack, collusion of two users;
- **Global parameters:** $\mathbf{gp} = (b)$, $\mathbf{mpk}_a(\text{att}_i) = (b_{\text{att}_i}, b_{\neg\text{att}_i}, b_{*\text{att}_i})$;
- **Secret keys:** Define $\overline{\text{att}} = \text{att}$ if $\text{att} \in \mathcal{S}$ and otherwise $\overline{\text{att}} = \neg\text{att}$, $\mathbf{k}_u(\sum r_i, b) = ((\sum_{\text{att}_i \in \mathcal{U}} r_i)b)$, and $\mathbf{k}_a(\overline{\text{att}}_i, r_i, \mathbf{b}) = (r_i b + b b_{\overline{\text{att}}_i}, r_i b + b b_{*\text{att}_i})$;
- **Input:** $\mathcal{S}_1 = \{\text{att}_1, \neg\text{att}_2\}$, $k_u \leftarrow \mathbf{k}_u(r_1 + r_2, b)$, $(k_{1,i}, k_{2,i}) \leftarrow \mathbf{k}_{a,1}(\overline{\text{att}}_i, r_i, \mathbf{b})$, $\mathcal{S}_2 = \{\neg\text{att}_1, \text{att}_2\}$, with $k'_u \leftarrow \mathbf{k}_u(r'_1 + r'_2, b)$, $(k'_{1,i}, k'_{2,i}) \leftarrow \mathbf{k}_a(\overline{\text{att}}_i, r'_i, \mathbf{b})$;

- **The goal:** Generate a key for $\mathcal{S}_3 = \{\text{att}_1, \text{att}_2\}$;
- **The attack:** $\mathbf{k}_u(r'_1 + r'_2, \mathbf{b}) = k'_u$, $\mathbf{k}_a(\overline{\text{att}}_1, r'_1, \mathbf{b}) = (k_{1,1} + k'_{2,1} - k_{2,1}, k'_{2,1})$, and $\mathbf{k}_a(\overline{\text{att}}_2, r'_2, \mathbf{b}) = (k'_{1,2}, k'_{2,2})$. □

*The NDCW15 [30] scheme:* This scheme implements a white-box tracing scheme such that the KGA can trace the secret keys of misbehaving users to the identity. To this end, some of the exponents (i.e. $x_1, x_2, x_3$ below) are known to the user. Note that this scheme is given in a composite-order setting, despite the fact that our framework is defined in the prime-order setting. However, the attack also readily extends to the composite-order setting. Furthermore, it should be noted that the keys as considered below correspond with those given in the second step of the key generation in [30]. We have also removed all unnecessary global parameters and ciphertext elements.

- **Type of attack:** Complete decryption attack;
- **Global parameters:** $\mathbf{gp} = (b_1, b_2)$;
- **User-key:** $\mathbf{k}_u(\alpha, \mathbf{b}) = (\frac{\alpha}{b_1+x_3} + x_2\frac{b_2}{b_1+x_3}, x_1, x_1b_1)$;
- **Attribute-independent ciphertext:** $\mathbf{c}(\mathbf{s}, \mathbf{b}) = (s, sb_1, sb_2)$;
- **Known variables:** $\mathfrak{K} = \{x_1, x_2, x_3\}$ (by definition);
- **The goal:** Recover $\alpha s$ from $(k_1, k_2, k_3) \leftarrow \mathbf{k}_u(\alpha, \mathbf{b}), (c_1, c_2, c_3) \leftarrow \mathbf{c}(\mathbf{s}, \mathbf{b})$;
- **The attack:** $\alpha s = x_3k_1c_1 + k_1c_2 - x_2c_3$. □

### 5.2 Multi-authority ABE

*The YJ12 [42] scheme:* This scheme employs a certificate authority (CA), which is assumed to be fully trusted, and (corruptable) attribute authorities ($\mathcal{A}_i$), which are responsible for the generation of the secret keys. The CA generates user-specific random $r$, meaning that each authority $\mathcal{A}_i$ needs to know the master secret key $b/b'$ in order to compute its user-key element. For the definition of the key encodings, it is important to know that we assume that the master public keys are generated as $\mathcal{H}(\text{att})^{\alpha_i}$ rather than $g^{\alpha_i \mathcal{H}'(\text{att})}$, as proposed in [42]. The latter trivially enables complete attribute-key attacks if $\mathcal{H}'$ is public, while the former ensures that $\mathcal{H}(\text{att})^{\alpha_i} = g^{\alpha_i b_{\text{att}}}$ such that $b_{\text{att}}$ is unknown to everyone and therefore protects against these attacks.

- **Type of attack:** Complete master-key attack, corruption of one $\mathcal{A}$;
- **Global parameters:** $\mathbf{gp} = (b', 1/b')$;
- **Master secret key $\mathcal{A}_i$:** $\mathfrak{msk}_i = (\alpha_i, b/b')$;
- **User-key:** $\mathbf{k}(\alpha_i, \mathbf{r}, \mathbf{b}) = (r, rb/b' + \alpha_i/b')$;
- **Attribute-independent ciphertext:** $\mathbf{c}(\mathbf{s}, \mathbf{b}) = (sb')$;
- **Blinding value:** $(\sum_i \alpha_i)s$, so $\text{mk}(\alpha_i, \mathbf{b}) = \alpha_i/b'$;
- **Known exponents:** $\mathfrak{K} = \{\alpha', b/b'\}$ (by corrupting $\mathcal{A}'$);
- **The goal:** Recover $\text{mk}(\alpha_i, \mathbf{b})$ from $(k_{1,i}, k_{2,i}) \leftarrow \mathbf{k}(\alpha_i, \mathbf{r}, \mathbf{b})$;
- **The attack:** $\text{mk}(\alpha_i, \mathbf{b}) = k_{2,i} - b/b'k_{1,i}$. □

Note that this scheme cannot easily be fixed. Even if it is assumed that $r$ and $b/b'$ cannot be retrieved via corruption of one authority $\mathcal{A}'$, the fact that each user-key encoding uses the same $r$ and $b/b'$ ensures that any $\mathcal{A}'$ can use its knowledge of $\alpha_i$ to recover $rb/b'$. Therefore, collusion with a user leads to the recovery of $\alpha_j/b'$ of each other authority that issued a key to this user.

*The YJR+13 [44] and WJB17 [40] schemes:* A more detailed description of this scheme and our attack can be found in Section 4.1.

- **Type of attack:** Complete decryption attack;
- **Global parameters:** $\mathbf{gp} = (\text{gp}_1, \text{gp}_2, ...) = (b, 1/b', ...)$;
- **User-key:** $\mathbf{k}_u(\alpha, \mathbf{r}, \mathbf{b}) = (\alpha/x_1 + x_2 b + rb/b', rb'/x_1, rb)$;
- **Attribute-independent ciphertext:** $\mathbf{c}(\mathbf{s}, \mathbf{b}) = (s, s/b')$;
- **Known exponents:** $\mathfrak{K} = \{x_1, x_2\}$ (by definition);
- **The goal:** Recover $\alpha s$ from $(k_1, k_2, k_3) \leftarrow \mathbf{k}_u(\alpha, \mathbf{r}, \mathbf{b})$, $(c_1, c_2) \leftarrow \mathbf{c}(\mathbf{s}, \mathbf{b})$;
- **The attack:** $\alpha s = x_1 k_1 c_1 - x_1 x_2 \text{gp}_1 c_1 - x_1 k_3 c_2$. □

The YJ14 [43] scheme (which we discuss later) was proposed as a 'more secure' version of this scheme. In this scheme, $x_2$ is removed from the secret key of the user, and instead shared with each attribute authority $\mathcal{A}_i$.

*The JLWW13 [18] and JLWW15 [19] schemes:* JLWW15 [19] is different from JLWW13 in the encryption algorithm. It is incorrect, because a value of a single user-specific secret key is used in the encryption algorithm. More details on this attack can be found in Section 4.2.

- **Type of attack:** Conditional attribute-key attack, collusion of two users;
- **Master attribute-key:** $\mathbf{mpk}_a(\text{att}_i) = b_{\text{att}_i}$;
- **Secret keys:** $\mathbf{k}_u(\alpha, r, \mathbf{b}) = (\alpha + r)$, $\mathbf{k}_a(\text{att}_i, r, r_i, \mathbf{b}) = (r_i b_{\text{att}_i} + r, r_i)$;
- **Input:** $\mathcal{S}_1 = \{\text{att}_1\}$, $k \leftarrow \mathbf{k}_u(\alpha, r, \mathbf{b})$, $(k_1, k_2) \leftarrow \mathbf{k}_a(\text{att}_1, r, r_1, \mathbf{b})$, $\mathcal{S}_2 = \{\text{att}_2\}$, $k' \leftarrow \mathbf{k}_u(\alpha, r', \mathbf{b})$, $(k'_1, k'_2) \leftarrow \mathbf{k}_a(\text{att}_2, r', r_2, \mathbf{b})$;
- **The goal:** Generate a key for $\mathcal{S}_3 = \{\text{att}_1, \text{att}_2\}$;
- **The attack:** $\mathbf{k}_u(\alpha, r, \mathbf{b}) = k$, $\mathbf{k}_a(\text{att}_1, r, r_1, \mathbf{b}) = (k_1, k_2)$, $\mathbf{k}_a(\text{att}_2, r, r_2, \mathbf{b}) = (k'_1 + k - k', k'_2)$. □

This scheme was derived from the BSW07 [5] scheme, however, the user-key part of the secret key is defined as $(\alpha + r)/b$, and the ciphertext also includes $sb$. While the JLWW15 [19] seems to address this issue, it does this by including a user-specific $b$, meaning that a ciphertext can only be decrypted by one user. Moreover, encryption can only be performed by a user that knows this specific value. Because the generation of $\alpha$ and $r$ are distributed, the generation of any 'fixed' version that replaces $\alpha + r$ by $(\alpha + r)/b$ needs to be distributed as well. It is unclear if this can be done (in any way that preserves the rest of the scheme).

*The QLZ13 [32] scheme:* This scheme addresses real-world privacy concerns involving the authorities and other users. In particular, it implements hidden access structures, and supports a blind key generation. However, we show that the secret keys trivially leak the master-key of all authorities. Note that we only list those parts of the user-key that are relevant to the attack.

- **Type of attack:** Complete master-key attack;
- **Global parameters:** $\mathbf{gp} = (b, b_1, b', ...)$;
- **User-key:** $\mathbf{k}_u(\alpha, \mathbf{r}, \mathbf{b}) = (\alpha + rb + \frac{b_1}{x+b'}, rb - r'b_1, (r' + \frac{1}{x+b'})b_1)$;
- **Known variables:** $\mathfrak{K} = \{x\}$ (by definition);
- **The goal:** Recover $\alpha$ from $(k_1, k_2, k_3) \leftarrow \mathbf{k}_u(\alpha, \mathbf{r}, \mathbf{b})$;
- **The attack:** $\alpha = k_1 - k_2 + k_3$. □

*The YJ14 [43] scheme:* Details about this attack can be found in Section 4.3.

- **Type of attack:** Complete decryption attack, under corruption of one $\mathcal{A}$;
- **Global parameters:** $\mathbf{gp} = (b, b')$;
- **Master secret key $\mathcal{A}_i$:** $\mathfrak{msk}_i = (\alpha_i, x)$;
- **User-key:** $\mathbf{k}_u(\alpha_i, \mathbf{r}, \mathbf{b}) = (\alpha_i + xb + rb', r)$;
- **Attribute-independent ciphertext:** $\mathbf{c}(\mathbf{s}, \mathbf{b}) = (s, sb', ...)$;
- **Blinding value:** $(\sum_i \alpha_i)s$;
- **Known variables:** $\mathfrak{K} = \{x\}$ (by corrupting $\mathcal{A}'$);
- **The goal:** Recover $\alpha_i s$ from $(k_{1,i}, k_{2,i}) \leftarrow \mathbf{k}_u(\alpha_i, \mathbf{r}, \mathbf{b})$, $(c_1, c_2) \leftarrow \mathbf{c}(\mathbf{s}, \mathbf{b})$;
- **The attack:** $\alpha_i s = k_{1,i}c_1 - k_{2,i}c_2 - x\text{mpk}_1 c_1$. □

*The CM14 [9] scheme:* This scheme is a multi-authority version of the Wat11 [38] scheme.

- **Type of attack:** Complete decryption attack, under corruption of one $\mathcal{A}$;
- **Master key pair of $\mathcal{A}_i$: $\mathbf{mpk}_i$** $= (b_i)$, $\mathfrak{msk}_i = (b_i)$;
- **User-key:** $\mathbf{k}_u(\alpha_i, \mathbf{r}, \mathbf{b}) = (\frac{\alpha_i + r}{b_i}, r)$;
- **Attribute-independent ciphertext:** $\mathbf{c}(\mathbf{s}, \mathbf{b}) = (sb_i)$;
- **Blinding value:** $(\sum_i \alpha_i)s$;
- **Known variables:** $\mathfrak{K} = \{b_1\}$ (by corrupting $\mathcal{A}_1$);
- **The goal:** Recover $\alpha_i s$ from $(k_{1,i}, k_{2,i}) \leftarrow \mathbf{k}_u(\alpha_i, \mathbf{r}, \mathbf{b})$, $c_1 \leftarrow \mathbf{c}(\mathbf{s}, \mathbf{b})$;
- **The attack:** $\alpha_i s = k_{1,i}c_1 - 1/b_1 k_{2,i}c_1$ such that $i \neq 1$. □

The issue in this scheme is that $s$ (which is part of the blinding value of the message) can be recovered by every authority associated with the access policy. It is unclear if the scheme can be fixed without creating other issues e.g. with security or the revocation functionality.

*The QLZH15 [33] scheme:* Details of this attack can be found in Section 4.4.

- **Type of attack:** Incorrect;
  item **Global parameters:** $\mathbf{gp} = (b)$;
- **Master key pair of $\mathcal{A}_i$:** $\mathfrak{msk}_i(\text{att}) = (b_{\text{att}})$, $\mathbf{mpk}_i(\text{att}) = (b_{\text{att}})$;

- **Secret key:** $\mathbf{k}_u(\alpha, \mathbf{r}, \mathbf{b}) = (\alpha + (\sum r_i)b)$, $\mathbf{k}_a(\text{att}_i, r_i, \mathbf{b}) = (r_i b / b_{\text{att}_i})$;
- **Input:** For $\mathcal{S} = \{\text{att}_1, \text{att}_2\}$ such that $\text{att}_1 \in \mathcal{U}_1$, $\text{att}_2 \in \mathcal{U}_2$, authority $\mathcal{A}_1$ and $\mathcal{A}_2$ securely and jointly generate $k \leftarrow \mathbf{k}_u(\alpha, r_1, r_2, \mathbf{b})$, where $r_1, r_2 \in_R \mathbb{Z}_p$ are generated by $\mathcal{A}_1$ and $\mathcal{A}_2$, respectively, and $k_i \leftarrow \mathbf{k}_a(\text{att}_i, r_i, \mathbf{b})$;
- **Ciphertext:** $\mathbb{A} = \text{att}_1 \wedge \text{att}_2$, $\mathbf{c}_{\mathbb{A}}(s_1, s_2, \mathbf{b}) = (s_1 + s_2, s_1 b_{\text{att}_1}, s_2 b_{\text{att}_2})$;
- **Blinding value:** $\alpha(s_1 + s_2)$;
- **The goal:** Showing that, even though $\mathbb{A} \models \mathcal{S}$, $\alpha(s_1+s_2)$ cannot be recovered;
- **Incorrectness:**

$$\begin{aligned} kc_1 - k_1c_2 - k_2c_3 &= \alpha(s_1 + s_2) + (r_1 + r_2)(s_1 + s_2)b - r_1s_1b - r_2s_2b \\ &= \alpha(s_1 + s_2) + r_1s_2b + r_2s_1b \neq \alpha(s_1 + s_2). \end{aligned}$$

□

*The LXXH16 [25] and MST17 [29] schemes:* These schemes are similar. The LXXH16 scheme employs a semi-honest (i.e. corruptable) certificate authority CA to run the global setup. In the MST17 scheme, it is unclear which entity runs it and therefore generates the $b$ below.

- **Type of attack:** Complete master-key attack, under corruption of CA;
- **Global parameters:** $\mathbf{gp} = (b)$;
- **User-key:** $\mathbf{k}_u(\alpha, \mathbf{r}, \mathbf{b}) = (\alpha + rb, r)$;
- **Known variables:** $\mathfrak{K} = \{b\}$ (by corrupting CA, and thus the global setup);
- **The goal:** Recover $\alpha$ from $(k_1, k_2) \leftarrow \mathbf{k}_u(\alpha, \mathbf{r}, \mathbf{b})$;
- **The attack:** $\alpha = k_1 - bk_2$. □
- **Note:** This scheme can be fixed by distributing the generation of $b$.

*The PO17 [31] scheme:* This scheme was proposed to address some issues of the Cha07 [6] scheme. In particular, the Cha07 scheme requires that a user receives a key from each authority. However, unlike Cha07, the PO17 scheme does not protect against corruption, so in terms of security, it is closer to any single-authority scheme. It is unclear to us why this scheme uses different master secret keys for each authority if they are assumed to behave correctly and honestly. That is, employing a single-authority scheme, and sharing the master secret keys with all authorities, seems more straightforward, and provides even more redundancy than the proposed solution. In addition, the single-authority variant (e.g. [17]) is generally more efficient.

- **Type of attack:** Complete decryption attack under corruption of one $\mathcal{A}$;
- **Master key pair of $\mathcal{A}_i$: $\mathbf{mpk}_i = (b_i)$**, $\mathfrak{msk}_i = (b_i)$;
- **User-key:** $\mathbf{k}_u(\alpha_i, \mathbf{r}, \mathbf{b}) = (\frac{\alpha_i - r}{b_i}, r)$;
- **Attribute-independent ciphertext:** $\mathbf{c}(\mathbf{s}, \mathbf{b}) = (sb_i)$;
- **Blinding value:** $(\sum_i \alpha_i)s$;
- **Known variables:** $b_1$ (by corrupting $\mathcal{A}_1$);
- **The goal:** Recover $\alpha_i s$ from $(k_{1,i}, k_{2,i}) \leftarrow \mathbf{k}_u(\alpha_i, \mathbf{r}, \mathbf{b})$, $c_1 \leftarrow \mathbf{c}(\mathbf{s}, \mathbf{b})$;
- **The attack:** $\alpha_i s = k_{1,i}c_1 + 1/b_1 k_{2,i}c_1$. □

**Table 5.** Table of the analyzed schemes, and the consequences of our attacks.

| | Scheme | Problem | CPA-security |
|---|---|---|---|
| | ZH10 [47], ZHW13 [48] | Recurring monomials | ✗ |
| | NDCW15 [30] | Known-exponent exploits | ✗ |
| **MA-ABE** | YJ12 [42] | Known-exponent exploits | $✗_{\mathcal{A}}$ |
| | YJR+13 [44], WJB17 [40] | Known-exponent exploits | ✗ |
| | JLWW13 [18], JLWW15 [19] | Recurring monomials | ✗ |
| | QLZ13 [32] | Recurring monomials | ✗ |
| | YJ14 [43] | Known-exponent exploits | $✗_{\mathcal{A}}$ |
| | CM14 [9] | Known-exponent exploits | $✗_{\mathcal{A}}$ |
| | QLZH15 [33] | Incorrect | U |
| | LXXH16 [25], MST17 [29] | Known-exponent exploits | $✗_{\text{CA}}$ |
| | PO17 [31] | Known-exponent exploits | $✗_{\mathcal{A}}$ |
| | MGZ19 I [27] | Known-exponent exploits | $✗_{\text{CA}}$ |

$✗_{\mathcal{A}}$, $✗_{\text{CA}}$ = none under corruption of $\mathcal{A}$, CA; U = unknown

*The first MGZ19 [27] scheme:* This scheme addresses the issue of using a random oracle in the LW11 [22] scheme. It employs multiple 'central authorities' and attribute authorities. The security model considers corruption of the attribute authorities, however, it does not consider the corruption of one of the CAs. These CAs provide the user-specific randoms in the key generation to avoid random oracles. We show that trusting the CA is necessary to ensure security. In particular, we show that the scheme is vulnerable to an attack, provided that the attribute authorities trust the (corrupted) CA. A goal of the scheme is that the attribute authorities do not even have to be aware of the CAs, implying that the authorities inherently assume all CAs are honest.

- **Type of attack:** Complete master-key attack, under corruption of one CA;
- **Master key pair $\mathcal{A}_i$: $\mathbf{mpk}_{a,i}$**$(\text{att}_j) = (b_{\text{att}_j})$, $\mathfrak{msk}_i(\text{att}_j) = (\alpha_i, b_{\text{att}_j})$;
- **CA$_i$ generates:** $r$;
- **Secret key:** $\mathbf{k}_u(\alpha_i, \mathbf{r}, \mathbf{b}) = (r)$, $\mathbf{k}_a(\text{att}_j, \alpha_i, \mathbf{r}, \mathbf{b}) = (\alpha_i + rb_{\text{att}_j})$;
- **Known variables:** $\mathfrak{K} = \{r\}$ (by corrupting one CA);
- **The goal:** Recover $\alpha_i$ from $k_{i,j} \leftarrow \mathbf{k}_a(\text{att}_j, \alpha_i, \mathbf{r}, \mathbf{b})$, $\text{mpk}_{i,j} \leftarrow \mathbf{mpk}_{a,i}(\text{att}_j)$;
- **The attack:** $\alpha_i = k_{i,j} - r\text{mpk}_{i,j}$. □

## 6 Discussion

We have presented a linear, heuristic approach to analyzing security—consisting of a more concise notation—and applied it to existing schemes. We have shown that several schemes are vulnerable to our attacks, either rendering them fully or partially insecure. Most of the attacks are similar in that they either exploit that one monomial occurs more than once in the keys, or known exponents yield sufficient knowledge to enable an attack. Table 5 lists each attacked scheme and the associated fundamental problem that enables the attack. It also shows whether a scheme is insecure in the basic (CPA-)security model, or only under corruption of the central authority (CA) or attribute authorities ($\mathcal{A}$).

We identify the most prevalent issues with the broken multi-authority schemes. In general, schemes for which we have found an attack without requiring corruption are structurally more complicated than the single-authority schemes on which they are (loosely) based. Schemes insecure under corruption are generally closer to its (provably secure) single-authority variant, but the knowledge of certain exponents enables an attack. If possible, a distributed generation of these exponents could prevent this. For future work, it might be interesting to construct a generic transformation on any single-authority scheme to the multi-authority setting that provably ensures security against corrupted authorities.

**Acknowledgments.** We would like to thank the anonymous reviewers of ACNS for their helpful comments.

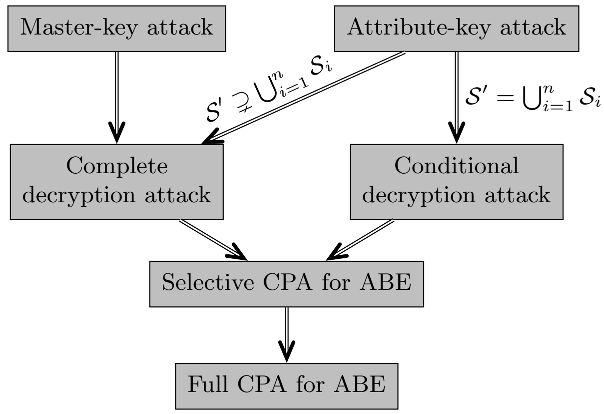
## References

1. S. Agrawal, and M. Chase, "Simplifying Design and Analysis of Complex Predicate Encryption Schemes", in *EUROCRYPT'17*, pp. 627–656, Springer, 2017.
2. M. Ambrona, G. Barthe, R. Gay, and H. Wee, "Attribute-Based Encryption in the Generic Group Model: Automated Proofs and New Constructions", in *CCS'17*, pp. 647–664, ACM, 2017.
3. N. Attrapadung, "Dual System Encryption via Doubly Selective Security: Framework, Fully Secure Functional Encryption for Regular Languages, and More", in *EUROCRYPT'14*, pp. 557–577, Springer, 2014.
4. A. Beimel, "Secure Schemes for Secret Sharing and Key Distribution", PhD Thesis, Ben Gurion University, 1996.
5. J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-Policy Attribute-Based Encryption", in *S&P '07*, pp. 321–334, ACM, 2007.
6. M. Chase, "Multi-Authority Attribute-Based Encryption", in *TCC'07*, pp. 515–534, Springer, 2007.
7. M. Chase, and S. S. M. Chow, "Improving Privacy and Security in Multi-Authority Attribute-based Encryption", in *CCS'09*, pp. 121–130, ACM, 2009.
8. P. Chaudhari, M. L. Das, and A. Mathuria, "On Anonymous Attribute Based Encryption", in *ICISS'15*, pp. 378–392, Springer, 2015.
9. J. Chen, and H. Ma, "Efficient decentralized attribute-based access control for cloud storage with user revocation", in *2014 IEEE International Conference on Communications (ICC)*, pp. 3782–3787, IEEE, 2014.
10. S. S. M. Chow, "A Framework of Multi-Authority Attribute-Based Encryption with Outsourcing and Revocation", in *SACMAT'16*, pp. 215–226, ACM, 2016.
11. A. Ge, J. Zhang, R. Zhang, C. Ma, and Z. Zhang, "Security Analysis of a Privacy-Preserving Decentralized Key-Policy Attribute-Based Encryption Scheme", *Transactions on Parallel and Distributed Systems*, **24**(11), pp. 2319–2321, IEEE, 2013.
12. V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-Based Encryption for Fine-Grained Access Control of Encrypted Data", in *CCS*, pp. 89–98, ACM, 2006.
13. J. Han, W. Susilo, Y. Mu, and J. Yan, "Privacy-Preserving Decentralized Key-Policy Attribute-Based Encryption", in *IEEE Transactions on Parallel and Distributed Systems*, **23**(11), pp. 2150–2162, IEEE, 2012.

14. J. Han, W. Susilo, Y. Mu, J. Zhou, and M. H. A. Au, "PPDCP-ABE: Privacy-Preserving Decentralized Ciphertext-Policy Attribute-Based Encryption", in *ESORICS'14*, pp. 73–90, Springer, 2014.
15. J. Han, W. Susilo, Y. Mu, J. Zhou, and M. H. A. Au, "Improving Privacy and Security in Decentralized Ciphertext-Policy Attribute-Based Encryption", in *Transactions on Information Forensics and Security*, **10**(3), pp. 665–678, IEEE, 2015.
16. J. Hong, K. Xue, and W. Li, "Comments on "DAC-MACS: Effective Data Access Control for Multiauthority Cloud Storage Systems"/Security Analysis of Attribute Revocation in Multiauthority Data Access Control for Cloud Storage Systems", in *IEEE Transactions on Information Forensics and Security*, **10**(6), pp. 1315–1317, IEEE, 2015.
17. L. Ibraimi, Q. Tang, P. Hartel, and W. Jonker, "Efficient and Provable Secure Ciphertext-Policy Attribute-Based Encryption Schemes", in *ISPEC'09*, pp. 1–12, Springer, 2009.
18. T. Jung, X. Y. Li, Z. Wan, and M. Wan, "Privacy Preserving Cloud Data Access With Multi-Authorities", in *INFOCOM'13*, pp. 2625–2633, IEEE, 2013.
19. T. Jung, X. Y. Li, Z. Wan, and M. Wan, "Control Cloud Data Access Privilege and Anonymity with Fully Anonymous Attribute-Based Encryption", in *IEEE Transactions on Information Forensics and Security*, **10**(1), pp. 190–199, IEEE, 2015.
20. T. Jung, X. Y. Li, Z. Wan, and M. Wan, "Rebuttal to "Comments on "Control Cloud Data Access Privilege and Anonymity With Fully Anonymous Attribute-Based Encryption""", in *IEEE Transactions on Information Forensics and Security*, **11**(4), pp. 868–868, IEEE, 2016.
21. A. Lewko, T. Okamoto, A. Sahai, K. Takashima, and B. Waters, "Fully Secure Functional Encryption: Attribute-Based Encryption and (Hierarchical) Inner Product Encryption", in *EUROCRYPT'10*, pp. 62–91, Springer, 2010.
22. A. Lewko, and B. Waters, "Decentralizing Attribute-Based Encryption", in *EUROCRYPT'11*, pp. 568–588, Springer, 2011.
23. J. Li, K. Ren, B. Zhu, and Z. Wan, "Privacy-Aware Attribute-Based Encryption with User Accountability", in *ISC'09*, pp. 347–362, Springer, 2009.
24. J. Li, Q. Huang, X. Chen, S. S. M. Chow, D. S. Wong, and D. Xie, "Multi-Authority Ciphertext-Policy Attribute-Based Encryption with Accountability", in *AsiaCCS'11*, pp. 386–390, ACM, 2011.
25. W. Li, K. Xue, Y. Xue, and J. Hong, "TMACS: A Robust and Verifiable Threshold Multi-Authority Access Control System in Public Cloud Storage", in *IEEE Transactions on Parallel and Distributed Systems*, **27**(5), pp. 1484–1496, IEEE, 2016.
26. H. Lin, Z. Cao, X. Liang, and J. Shao, "Secure Threshold Multi Authority Attribute Based Encryption without a Central Authority", in *INDOCRYPT'08*, pp. 426–436, Springer, 2008.
27. C. Ma, A. Ge, and J. Zhang, "Fully Secure Decentralized Ciphertext-Policy Attribute-Based Encryption in Standard Model", in *Inscrypt'19*, pp. 427–447, Springer, 2019.
28. H. Ma, R. Zhang, and W. Yuan, "Comments on "Control Cloud Data Access Privilege and Anonymity with Fully Anonymous Attribute-Based Encryption"", in *Trans. on Information Forensics and Security*, **11**(4), pp. 866–867, IEEE, 2016.
29. Q. M. Malluhi, A. Shikfa, and V. C. Trinh, "Ciphertext-Policy Attribute-based Encryption Scheme With Optimized Ciphertext Size And Fast Decryption", in *ASIACCS'17*, pp. 230–240, ACM, 2017.
30. J. Ning, X. Dong, Z. Cao, and L. Wei, "Accountable Authority Ciphertext-Policy Attribute-Based Encryption with White-Box Traceability and Public Auditing in the Cloud", in *ESORICS'15*, pp. 270-289, Springer, 2015.

31. H. S. G. Pussewalage, and V. A. Oleshchuk, "A Distributed Multi-Authority Attribute Based Encryption Scheme for Secure Sharing of Personal Health Records", in *SACMAT'17*, pp. 255–262, ACM, 2017.
32. H. Qian, J. Li, and Y. Zhang, "Privacy-Preserving Decentralized Ciphertext-Policy Attribute-Based Encryption with Fully Hidden Access Structure", in *ICICS'13*, pp. 363–372, Springer, 2013.
33. H. Qian, J. Li, Y. Zhang, and J. Han, "Privacy-Preserving Personal Health Record Using Multi-Authority Attribute-Based Encryption with Revocation", in *International Journal of Information Security*, 14(6), pp. 487–497, Springer, 2015.
34. A. Sahai, and B. Waters, "Fuzzy Identity-Based Encryption", in *EUROCRYPT'05*, pp. 457–473, Springer, 2005.
35. S.-Y. Tan, K.-W. Yeow, and S. O. Hwang, "Enhancement of a Lightweight Attribute-Based Encryption Scheme for the Internet of Things", in *Internet of Things Journal*, **6**(4), pp. 6384–6395, IEEE, 2019.
36. The full version of this paper: "A Bunch of Broken Schemes: A Simple yet Powerful Linear Approach to Analyzing Security of Attribute-Based Encryption", available at: `https://surfdrive.surf.nl/files/index.php/s/K0tQcIA5pYs6JTu`
37. M. Wang, Z. Zhang, and C. Chen, "Security Analysis of a Privacy-Preserving Decentralized Ciphertext-Policy Attribute-Based Encryption Scheme", in *Concurrency and Computation: Practice and Experience*, **28**(4), pp. 1237–1245, Wiley Online Library, 2015.
38. B. Waters, "Ciphertext-Policy Attribute-Based Encryption - An Expressive, Efficient, and Provably Secure Realization", in *PKC'11*, pp. 53–70, Springer, 2011.
39. H. Wee, "Dual System Encryption via Predicate Encodings", in *TCC'14*, pp. 616–637, Springer, 2014.
40. X. Wu, R. Jiang, and B. Bhargava, "On the Security of Data Access Control for Multiauthority Cloud Storage Systems", in *IEEE Transactions on Services Computing*, **10**(2), pp. 258–272, IEEE, 2017.
41. F. Xhafa, J. Feng, Y. Zhang, X. Chen, and J. Li, "Privacy-aware attribute-based PHR sharing with user accountability in cloud computing", in *Journal of Supercomputering*, **71**, pp. 1607–1619, Springer, 2014.
42. K. Yang, and X. Jia, "Attribute-Based Access Control for Multi-Authority Systems in Cloud Storage", in *2012 32nd IEEE International Conference on Distributed Computing Systems*, pp. 536–545, IEEE, 2012.
43. K. Yang, and X. Jia, "Expressive, Efficient, and Revocable Data Access Control for Multi-Authority Cloud Storage", in *IEEE Transactions on Parallel and Distributed Systems*, **25**(7), IEEE, 2014.
44. K. Yang, X. Jia, K. Ren, B. Zhang, and R. Xie, "DAC-MACS: Effective Data Access Control for Multiauthority Cloud Storage Systems", in *IEEE Transactions on Information Forensics and Security*, **8**(11), pp. 1790–1801, IEEE, 2013.
45. X. Yao, Z. Chen, and Y. Tian, "A lightweight attribute-based encryption scheme for the Internet of Things", in *Journal on Future Generation Computer Systems*, **49**, pp. 104–112, Elsevier, 2015.
46. Y. Zhang, X. Chen, J. Li, D. Wong, and H. Li, "Anonymous attribute-based encryption supporting efficient decryption test", in *AsiaCCS*, pp. 511–516, ACM, 2013.
47. Z. Zhou, and D. Huang, "On Efficient Ciphertext-Policy Attribute Based Encryption and Broadcast Encryption", in *CCS'10*, pp. 753–755, ACM, 2010.
48. Z. Zhou, D. Huang, and Z. Wang, "Efficient Privacy-Preserving Ciphertext-Policy Attribute Based-Encryption and Broadcast Encryption", in *IEEE Transactions on Computers*, **64**(1), pp. 126–138, IEEE, 2013.

**Fig. 2.** The relationship between our proposed attacks and chosen-plaintext attacks.




## A Formal definition of access structures

**Definition 13 (Access structures [4]).** *Let $\{a_1, ..., a_n\}$ be a set of attributes. An access structure is a collection $\mathbb{A}$ of non-empty subsets of $\{a_1, ..., a_n\}$. The sets in $\mathbb{A}$ are called the authorized sets, and the sets that are not in $\mathbb{A}$ are called the unauthorized sets. An access structure $\mathbb{A} \subseteq 2^{\{a_1,...,a_n\}}$ is monotonic if for all $B, C$ holds: $B \in \mathbb{A}$ and $B \subseteq C$, then also $C \in \mathbb{A}$.*

## B Proofs of implications

For completeness, we give formal proofs of the implications between the definitions of the attacks (i.e. Definitions 2, 3, 4, and 5). More specifically, we prove that the master-key attacks (MKA) and attribute-key attacks (AKA) imply decryption attacks (DA), and decryption attacks imply selective chosen-plaintext attacks (sCPA). Furthermore, it is a well-known fact that selective chosen-plaintext attacks imply full chosen-plaintext attacks [34] (and conversely, full CPA-security implies selective CPA-security). The relationship between the *attacks* is summarized in Figure 2. For instance, if we can perform a master-key attack (i.e. define a polynomial-time algorithm that computes a master-key that can decrypt any ciphertext), then we can also perform a complete decryption attack (i.e. define a polynomial-time algorithm that decrypts any ciphertext with any number of unauthorized secret keys).

**Lemma 1 (MKA implies complete DA).** *If some polynomial-time attacker $\mathcal{B}_{\text{MKA}}$ exists that can win the master-key attack (Definition 3) game, then a polynomial-time attacker $\mathcal{B}_{\text{CDA}}$ exists that can win the complete decryption attack (Definition 4) game.*

*Proof.* Let $\mathcal{B}_{\text{CDA}}$ be the attacker that plays the complete decryption game with the challenger. Suppose $\mathcal{B}_{\text{MKA}}$ denotes a polynomial-time attacker that can win the master-key attack game.

- **Setup phase:** The challenger runs the setup of the scheme, and sends the master public key to attacker $\mathcal{B}_{\text{CDA}}$, which relays it to attacker $\mathcal{B}_{\text{MKA}}$.
- **Key query phase I:** Attacker $\mathcal{B}_{\text{MKA}}$ generates sets $\mathcal{S}_1, ..., \mathcal{S}_{n_1}$ and sends these to attacker $\mathcal{B}_{\text{CDA}}$, which relays these to the challenger. For each set $\mathcal{S}_i$, the challenger generates a secret key $\text{SK}_{\mathcal{S}_i}$ and sends it back to attacker $\mathcal{B}_{\text{CDA}}$, which relays it to attacker $\mathcal{B}_{\text{MKA}}$.
- **Intermission:** The decision phase of attacker $\mathcal{B}_{\text{MKA}}$ yields as output the master-key $\text{MK}'$, which is sent to attacker $\mathcal{B}_{\text{CDA}}$.
- **Challenge phase:** Attacker $\mathcal{B}_{\text{CDA}}$ can then define any access structure $\mathbb{A}$ such that $\mathbb{A} \not\models \mathcal{S}_i$ for all $i \in \{1, ..., n_1\}$. The challenger encrypts a random message $m$ under this access structure, and sends the resulting challenge ciphertext CT to attacker $\mathcal{B}_{\text{CDA}}$.
- **Decision phase:** Attacker $\mathcal{B}_{\text{CDA}}$ uses $\text{MK}'$ to decrypt CT with the master-key decryption algorithm MKDecrypt conform Definition 1, yielding plaintext $m'$, and sends this to the challenger.

Because it was assumed that attacker $\mathcal{B}_{\text{MKA}}$ wins the MKA-game, $\text{MK}'$ is such that master-key decryption works on any ciphertext, and by extension resulting in a correct recovery of plaintext, i.e. $m' = m$. □

**Lemma 2 (AKA implies DA).** *If some polynomial-time attacker $\mathcal{B}_{\text{AKA}}$ exists that can win the attribute-key attack game, then a polynomial-time attacker $\mathcal{B}_{DA}$ exists that can win the decryption attack game. Furthermore, if the set $\mathcal{S}'$ for which the attacker recovers a secret key is strictly larger than the collective set of attributes used in the key query phase, then the decryption attack is complete. Otherwise, it is conditional.*

*Proof.* Let $\mathcal{B}_{\text{DA}}$ be the attacker that plays the decryption game with the challenger. Suppose $\mathcal{B}_{\text{AKA}}$ denotes a polynomial-time attacker that can win the attribute-key attack game.

- **Setup phase:** The challenger runs the setup of the scheme, and sends the master public key to attacker $\mathcal{B}_{\text{DA}}$, which relays it to attacker $\mathcal{B}_{\text{AKA}}$.
- **Key query phase I:** Attacker $\mathcal{B}_{\text{AKA}}$ generates sets $\mathcal{S}_1, ..., \mathcal{S}_{n_1}$ and sends these to attacker $\mathcal{B}_{\text{DA}}$, which relays these to the challenger. For each set, the challenger generates a secret key $\text{SK}_{\mathcal{S}_i}$ and sends it back to attacker $\mathcal{B}_{\text{DA}}$, which relays it to attacker $\mathcal{B}_{\text{AKA}}$.
- **Intermission:** The decision phase of attacker $\mathcal{B}_{\text{AKA}}$ yields as output $\text{SK}_{\mathcal{S}'}$ for set $\mathcal{S}'$ such that $\mathcal{S}' \supsetneq \mathcal{S}_i$ for all $i \in \{1, ..., n_1\}$. Then two cases may occur, for which attacker $\mathcal{B}_{\text{DA}}$ defines access structure $\mathbb{A}$ with $\mathbb{A} \not\models \mathcal{S}_i$ for all $i \in \{1, ..., n_1\}$ as follows:
  - $\mathcal{S}' = \bigcup_{i=1}^{n_1} \mathcal{S}_i$, in which case the attack game becomes conditional, and attacker $\mathcal{B}_{\text{DA}}$ defines $\mathbb{A}$ such that $\mathbb{A} \models \mathcal{S}'$;
  - $\mathcal{S}' \supsetneq \bigcup_{i=1}^{n_1} \mathcal{S}_i$, in which case the attack game becomes complete, and attacker $\mathcal{B}_{\text{DA}}$ defines $\mathbb{A}$ such that $\mathbb{A} \models \mathcal{S}'$ **and** $\mathbb{A} \not\models \bigcup_{i=1}^{n_1} \mathcal{S}_i$.
- **Challenge phase:** Attacker $\mathcal{B}_{\text{DA}}$ sends $\mathbb{A}$ to the challenger, which generates a random message $m$, encrypts it under the access structure $\mathbb{A}$ and sends the resulting challenge ciphertext CT to attacker $\mathcal{B}_{\text{DA}}$.

- **Decision phase:** Because $\mathbb{A} \models \mathcal{S}'$ holds, attacker $\mathcal{B}_{\text{DA}}$ can decrypt CT with secret key $\text{SK}_{\mathcal{S}'}$, yielding plaintext $m'$.

Because it was assumed that attacker $\mathcal{B}_{\text{AKA}}$ wins the AKA-game, $\text{SK}_{\mathcal{S}'}$ is valid, and therefore decryption yields the correct plaintext, i.e. $m' = m$. □

**Theorem 1 (DA implies Selective CPA).** *If some polynomial-time attacker $\mathcal{B}_{\text{DA}}$ exists that can win the decryption attack game, then a polynomial-time attacker $\mathcal{B}_{\text{sCPA}}$ exists that can win the selective chosen-plaintext attack game.*

*Proof.* Let $\mathcal{B}_{\text{sCPA}}$ be the attacker that plays the selective CPA game with the challenger. Suppose $\mathcal{B}_{\text{DA}}$ denotes a polynomial-time attacker that can win the decryption attack game.

- **Initialization phase:** Attacker $\mathcal{B}_{\text{sCPA}}$ commits to an access structure $\mathbb{A}$ to be used in the challenge phase, and sends it to the challenger.
- **Setup phase:** The challenger runs the setup and sends the master public key MPK to attacker $\mathcal{B}_{\text{sCPA}}$, which relays it to attacker $\mathcal{B}_{\text{DA}}$.
- **Key query phase I:** Attacker $\mathcal{B}_{\text{DA}}$ then defines sets $\mathcal{S}_1, ..., \mathcal{S}_{n_1}$ such that $\mathbb{A} \not\models \mathcal{S}_i$ for all $i \in \{1, ..., n_1\}$. Depending on whether attacker $\mathcal{B}_{\text{DA}}$ wins complete or conditional attack games, it also ensures $\mathbb{A} \not\models \bigcup_{i=1}^{n_1} \mathcal{S}_i$ or $\mathbb{A} \models \bigcup_{i=1}^{n_1} \mathcal{S}_i$, respectively. Attacker $\mathcal{B}_{\text{DA}}$ sends the sets to attacker $\mathcal{B}_{\text{sCPA}}$, which relays them to the challenger. Then, the challenger generates secret keys $\text{SK}_{\mathcal{S}_1}, ..., \text{SK}_{\mathcal{S}_{n_1}}$ and sends them back to attacker $\mathcal{B}_{\text{sCPA}}$, which relays them to attacker $\mathcal{B}_{\text{DA}}$.
- **Challenge phase:** Attacker $\mathcal{B}_{\text{sCPA}}$ generates two messages $m_0, m_1$ of equal length and sends these to the challenger, which flips a coin $b \in_R \{0, 1\}$, encrypts one of the messages $m_b$ under the previously chosen access structure and sends the resulting challenge ciphertext CT to attacker $\mathcal{B}_{\text{sCPA}}$, which relays it to attacker $\mathcal{B}_{\text{DA}}$.
- **Intermission:** The decision phase of attacker $\mathcal{B}_{\text{DA}}$ then yields as output the plaintext $m'$, and sends it to attacker $\mathcal{B}_{\text{sCPA}}$.
- **Key query phase II:** This phase may be skipped. As such, decryption attacks also imply selective CPA with non-adaptive key queries.
- **Decision phase:** Depending on whether $m' = m_0$ or $m' = m_1$, it outputs guess $b'$.

From the assumed success of attacker $\mathcal{B}_{\text{DA}}$, it follows that $m' = m_b$, from which it follows that attacker $\mathcal{B}_{\text{sCPA}}$ guesses correctly, i.e. $b' = b$. □