ACABELLA: Automated (Crypt)analysis of Attribute-Based Encryption Leveraging Linear Algebra

Antonio de la Piedra[1], Marloes Venema[2,4](✉), and Greg Alpár[3,4]

[1] Kudelski Security Research Team, Cheseaux-sur-Lausanne, Switzerland
[2] University of Wuppertal, Germany
[3] Open University of the Netherlands, Heerlen, the Netherlands
[4] Radboud University, Nijmegen, the Netherlands
antonio.delapiedra@kudelskisecurity.com, venema@uni-wuppertal.de, g.alpar@cs.ru.nl

Abstract. Attribute-based encryption (ABE) is a popular type of public-key encryption that enforces access control cryptographically, and has spurred the proposal of many use cases. To satisfy the requirements of the setting, tailor-made schemes are often introduced. However, designing secure schemes—as well as verifying that they are secure—is notoriously hard. Several of these schemes have turned out to be broken, making them dangerous to deploy in practice.
To overcome these shortcomings, we introduce ACABELLA. ACABELLA simplifies generating and verifying security proofs for pairing-based ABE schemes. It consists of a framework for security proofs that are easy to verify manually and an automated tool that efficiently generates these security proofs. Creating such security proofs generally takes no more than a few seconds. The output is easy to understand, and the proofs can be verified manually. In particular, the verification of a security proof generated by ACABELLA boils down to performing simple linear algebra.
The ACABELLA tool is open source and also available via a web interface. With its help, experts can simplify their proof process by verifying or refuting the security claims of their schemes and practitioners can get an assurance that the ABE scheme of their choice is secure.

Keywords: attribute-based encryption · automated analysis · automated proofs

1 Introduction

Attribute-based encryption (ABE) [23] is a popular cryptographic primitive that associates the keys and ciphertexts with attributes. ABE is attractive for practice, as it cryptographically implements a fine-grained access control on data [13,12,30,17]. Many use cases have been proposed for ABE, e.g., in cloud settings [16,24,31] or the Internet of Things [12,28]. However, due to the difficulty

of designing provably secure schemes, several practical schemes have turned out to be broken [15,27], making them dangerous to deploy in practice.

To simplify the design of provably secure pairing-based schemes, several works have been proposed [5,34,6,1,2,4,3]. These works abstract ABE schemes to pair encoding schemes (PESs), which essentially consider the exponent space of the keys and ciphertexts. In this work, we particularly focus on three frameworks that consider purely algebraic notions of security for the pair encodings to prove security of their ABE instantiations in the pairing-based setting:

- the Agrawal-Chase (AC17) [2] framework: which simplifies the verification of security proofs for a subclass of PESs that we will refer to as "PES-AC17";
- the Ambrona-Barthe-Gay-Wee (ABGW17) [3] framework: which automates the verification of certain algebraic properties used to prove security;
- the Riepel-Wee (RW22) [20] framework: which strengthens the security of the ABGW17 framework by considering a new algebraic property for PES-AC17.

Although these frameworks are strong contributions, they provide various trade-offs in human-verifiability, and simple generation and verification of security proofs. For instance, AC17 provides a method to prove security that is efficient to verify by reducing this effort to simple linear algebra. The algebraic structure of these proofs also allows us to generically transform secure schemes into richer schemes that are also provably secure [2,7]. However, generating these proofs may be difficult, and no automated tools exist that can do this. On the other hand, ABGW17 automates the effort of verifying security, but only in the single-key setting[5] and does not provide a manually verifiable security proof. Lastly, while RW22 improves on ABGW17 by formulating an algebraic property that does imply security in the multiple-key setting, they do not provide an automated tool to prove the algebraic property, and verifying the property cannot be done with simple linear algebra, like AC17.

Another framework of interest is the Venema-Alpár (VA21) [27] framework for manually finding attacks, which uses similar algebraic properties as ABGW17 to attack schemes, rather than prove them secure. Interestingly, they devise methods to utilize known variables in the exponent in their attacks. In contrast, the proof (and attack) techniques in ABGW17 and RW22 do not explicitly consider the possibility that some of the exponents are known. VA21 uses the knowledge of exponents to also cover multi-authority ABE [11], which employs multiple authorities to generate the secret keys instead of employing a single authority.

In this work, we propose ACABELLA, which unifies these four frameworks. From a theoretical standpoint, we propose new algebraic properties that are relatively simple to verify manually and that imply the algebraic properties required to prove security in the other works. Furthermore, we create a tool that automatically generates proofs that these algebraic properties hold. In addition, if it cannot find a security proof, it is likely to find an attack that is even simpler to verify (like in the VA21 framework).

[5] Security in the single-key setting does not provide security against collusion of users.

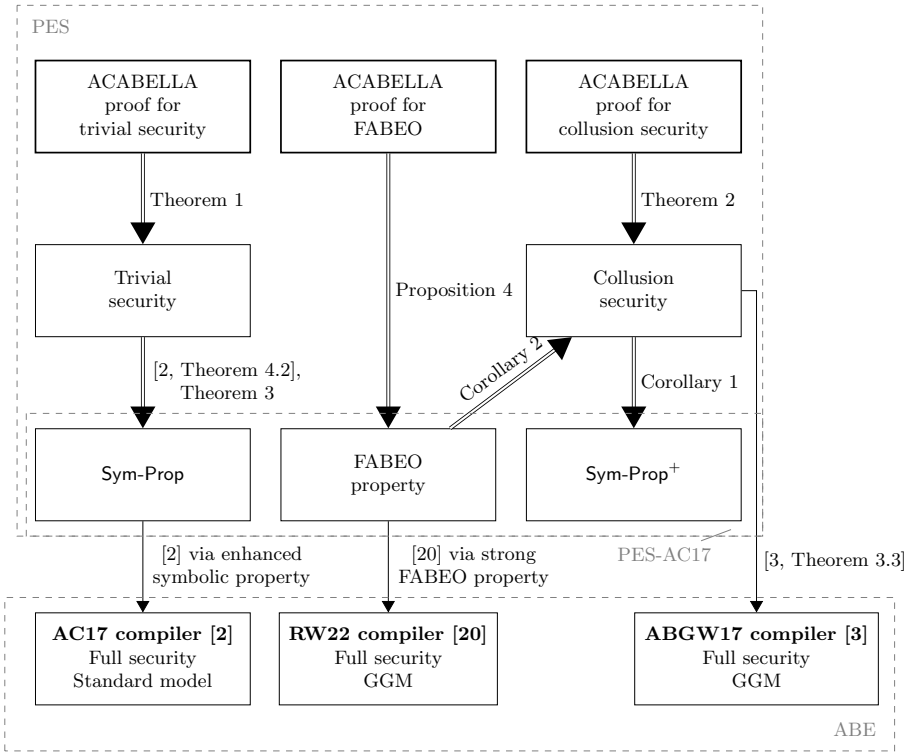

Fig. 1. Overview of the security results proven in this paper and AC17 [2], ABGW17 [3] and RW22 [20]. The double-edged arrows indicate the new results proven in this paper, and the normal-edged arrows indicate the results proven in previous work.

1.1 Our contributions

In this work, we introduce ACABELLA, which consists of a theoretical framework and an automated tool. As part of the theoretical framework, we provide the following contributions:

- We propose three new algebraic properties—associated with existing algebraic security properties—that can be used to generate proofs that are simple to verify manually, i.e., through simple linear algebra;
- We prove that the first algebraic property implies trivial security, meaning that the message cannot be recovered in the single-key setting;
- We prove that the second algebraic property implies collusion security, meaning that the message cannot be recovered in the multiple-key setting;
- We prove that the second algebraic property implies a security proof in the AC17 framework. This was already proven in AC17, but in contrast to this proof, our proof is constructive and can be used to generate a security proof;
- We prove that the third algebraic property implies the security property of RW22, which we refer to as "the FABEO property";

- We prove that the FABEO property implies our second algebraic property;
- We show that trivial security does not imply collusion security by breaking one of the schemes proposed in the ABGW17 framework. This attack directly contradicts Theorem 4.1 in ABGW17, and reduces the scope of their automated tool: it can be used only to prove trivial security.

Figure 1 summarizes these results and shows how they relate to the existing frameworks. As a result of our contributions, the three algebraic properties can be used to readily prove security in the AC17, ABGW17 and RW22 frameworks. At the same time, these proofs can be verified manually with linear algebra.

Using these theoretical results, we have created an automated tool to generate (and verify) security proofs in these frameworks. Conversely, if no security proofs can be found, the tool attempts to find an attack on the scheme, by automating the techniques in the VA21 framework. Like the VA21 framework, our methods explicitly consider the knowledge of exponents in the proofs and attacks, which is currently not supported in ABGW17 and RW22. By extension, our attacking functionality also supports multi-authority ABE (MA-ABE) [11]. Our tool is open source and also available as a web interface. Importantly, our web tool can be used to analyze schemes without needing to install any software, and contains plenty of examples that can be used to learn how the tool works. Furthermore, our tool is efficient: analyzing a scheme often takes at most seconds. In sum, our tool can really help the process of creating new schemes and provide better assurances that existing schemes are truly secure.

1.2 Comparison with AC17, ABGW17 and RW22

As we already mentioned, the AC17, ABGW17 and RW22 frameworks provide various trade-offs. Table 1 summarizes these trade-offs and how they compare with our framework. As the table shows, ours is the first framework that covers the general class of PESs—as also considered by ABGW17—that also provides automated proofs of security that can be verified manually through linear algebra. Furthermore, compared to ABGW17—the only other tool for ABE that can be used to automatically analyze ABE—our automated tool covers both the multiple-key setting (i.e., collusion security) and the multiple-challenge-ciphertext setting (via the FABEO property).

1.3 Organization

This paper is structured as follows. We first introduce some notations and the previous frameworks in Section 2. Then, we present our framework for security proofs and attacks, which applies to the general class of PESs, in Section 3. We show in Section 4 that these results imply security for the class of PES-AC17, in the AC17 and RW22 frameworks. These results are constructive, and are used in the implementation of our tool. We present the ACABELLA tool in Section 5, and give some examples of the proofs generated by the ACABELLA tool in Section 6. Finally, we conclude the paper in Section 7.

Table 1. Comparison of the various frameworks using algebraic properties to prove and verify security. We consider the class of PESs covered by the framework, whether the framework provides an automated tool to prove security and whether the proofs can be verified with linear algebra (SVLA).

Framework	Class	Automated	SVLA
AC17	PES-AC17	✗	✓
ABGW17	general	✓	✗
RW22	PES-AC17	✗	✗
ACABELLA	general	✓	✓

2 Preliminaries

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 $\mathbf{A}$ and $\mathbf{v}$ for matrices and vectors, respectively, where $(\mathbf{A})_{i,j}$ denotes the entry of $\mathbf{A}$ in the i-th row and j-th column, and $(\mathbf{v})_i$ denotes the i-th entry of $\mathbf{v}$. Furthermore, $\mathbf{x}(y_1, y_2, ...)$ denotes a vector, where the entries are polynomials over variables $y_1, y_2, ...$, with coefficients in some specified field. For conciseness, we often write only $\mathbf{x}$. We refer to a polynomial with only one term, or alternatively one term of the polynomial, as a monomial. We denote $a : \mathbf{A}$ to substitute variable a by a matrix $\mathbf{A}$. We define $\mathbf{1}_{i,j}^{d_1 \times d_2} \in \mathbb{Z}_p^{d_1 \times d_2}$ as the matrix with 1 in the i-th row and j-th column, and 0 everywhere else, and similarly $\mathbf{1}_i^{d_1}$ and $\overline{\mathbf{1}}_i^{d_2}$ as the row and column vectors with 1 in the i-th entry and 0 everywhere else. We use $\text{Ker}(\mathbf{A}) = \{\mathbf{v} \in \mathbb{Z}_p^{d_2} \mid \mathbf{A} \cdot \mathbf{v}^\intercal = \mathbf{0}^{d_1}\}$ to denote the kernel of $\mathbf{A}$.

2.1 Pairings

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

2.2 Attribute-based encryption

Definition 1 (Attribute-based encryption (ABE) [2]). *An attribute-based encryption scheme for a predicate $P\colon \mathcal{X} \times \mathcal{Y} \to \{0,1\}$ over a message space $\mathcal{M} = \{M_\lambda\}_{\lambda \in \mathbb{N}}$ consists of four algorithms:*

- $\text{Setup}(\lambda) \to (\text{MPK}, \text{MSK})$*: On input the security parameter λ, this probabilistic algorithm generates the domain parameters, the master public key* MPK *and the master secret key* MSK*.*
- $\text{KeyGen}(\text{MSK}, y) \to \text{SK}_y$*: On input the master secret key* MSK *and some $y \in \mathcal{Y}$, this probabilistic algorithm generates a secret key* SK_y*.*

- $\text{Encrypt}(\text{MPK}, x, M) \to \text{CT}_x$*: On input the master public key* MPK*, some* $x \in \mathcal{X}$ *and message* M*, this probabilistic algorithm generates a ciphertext* CT_x*.*
- $\text{Decrypt}(\text{MPK}, \text{SK}_y, \text{CT}_x) \to M$*: On input the master public key* MPK*, the secret key* SK_y*, and the ciphertext* CT_x*, if* $P(x, y) = 1$*, then it returns* M*. Otherwise, it returns an error message* $\perp$*.*

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

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

Security. We rely on the model for full security against chosen-plaintext attacks (see Appendix A.1 for a definition).

Decryption with the master key. In many ABE schemes, it is possible to use part of the master secret key, e.g., α to decrypt every ciphertext. We call this part of the MSK the master key MK. We also introduce a master-key decryption algorithm that takes as input any ciphertext and the master key and outputs the message:

- $\text{MKDecrypt}(\text{MK}, \text{CT}) \to M$: This deterministic algorithm takes as input a ciphertext CT and the master key MK, and outputs message M.

2.3 Multi-authority ABE

ACABELLA also allows for the cryptanalysis of multi-authority schemes [11], which employs multiple authorities instead of a single one. In multi-authority ABE, the setup is split in two setup phases: the first phase is a global setup in which the global parameters are generated, typically run by a central authority CA, and the second phase is an authority setup, run by an "attribute authority" AA_i. Furthermore, the key generation algorithm can also be split in multiple algorithms, each run by a different attribute authority AA_i. Depending on the security model that is used to prove security, the scheme may allow for corruption. In this work, any such security model allowing for corruption can be used, e.g., [11,18,22].

2.4 Instances of predicates

Identity-based encryption. In identity-based encryption (IBE), the key and ciphertext predicates $x \in \mathcal{X}$ and $y \in \mathcal{Y}$ are identities, where $\mathcal{X}$ and $\mathcal{Y}$ both are sets of identities. It holds that $P(x, y) = 1$ if and only if $x = y$.

Access policies. We represent access policies $\mathbb{A}$ by linear secret sharing scheme (LSSS) matrices.

Definition 2 (Access policies represented by LSSS [14]). *An access policy can be represented as a pair $\mathbb{A} = (\mathbf{A}, \rho)$ such that $\mathbf{A} \in \mathbb{Z}_p^{n_1 \times n_2}$ is an LSSS matrix, where $n_1, n_2 \in \mathbb{N}$, and ρ is a function that maps its rows to attributes in the universe. Then, for some vector with randomly generated entries $\mathbf{v} = (s, v_2, ..., v_{n_2}) \in \mathbb{Z}_p^{n_2}$, the i-th share of secret s generated by this matrix is $\lambda_i = \mathbf{A}_i \mathbf{v}^\intercal$, where $\mathbf{A}_i$ denotes the i-th row of $\mathbf{A}$. In particular, if $\mathcal{S}$ satisfies $\mathbb{A}$, then there exist a set of rows $\Upsilon = \{i \in [n_1] \mid \rho(i) \in \mathcal{S}\}$ and coefficients $\varepsilon_i \in \mathbb{Z}_p$ for all $i \in \Upsilon$ such that $\sum_{i \in \Upsilon} \varepsilon_i \mathbf{A}_i = (1, 0, ..., 0)$, and by extension $\sum_{i \in \Upsilon} \varepsilon_i \lambda_i = s$, holds. If $\mathcal{S}$ does not satisfy $\mathbb{A}$, there exists $\mathbf{w} = (1, w_2, ..., w_{n_2}) \in \mathbb{Z}_p^{n_2}$ such that $\mathbf{A}_i \mathbf{w}^\intercal = 0$ for all $i \in \Upsilon$ [8].*

Key-policy ABE In key-policy ABE (KP-ABE), the ciphertext predicate x is a set of attributes $\mathcal{S}$ over some universe of attributes $\mathcal{U}$, and the key predicate y is an access policy $\mathbb{A} = (\mathbf{A}, \rho)$, in this work represented as LSSS matrices (Definition 2). It holds that $P(x, y) = 1$ if and only if the set satisfies the policy.

Ciphertext-policy ABE In ciphertext-policy ABE (CP-ABE), the key predicate y is a set of attributes $\mathcal{S}$ over some universe of attributes $\mathcal{U}$, and the ciphertext predicate x is an access policy $\mathbb{A} = (\mathbf{A}, \rho)$. It holds that $P(x, y) = 1$ if and only if the set satisfies the policy.

2.5 The Venema-Alpár framework

We briefly review some parts of the Venema-Alpár (VA21) framework.

More concise notation via standard form. Many schemes have a similar structure, captured in frameworks that analyze the exponent space through pair encodings [34,5]. Pair encodings facilitate a shorter notation, and ultimately, a simpler security analysis.

Definition 3 (Standard form of attribute-based encryption [30]). *The standard form of ABE is defined as follows:*

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

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

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

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

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

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

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

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

where $\mathbf{c}$ *denotes two vectors defined over the ciphertext-specific random variables, master public keys and associated access structure.*

- Decrypt(SK, CT)*: Let* $\mathrm{SK} = (y, \mathbf{K} = h^{\mathbf{k}})$ *be a secret key and* $\mathrm{CT} = (x, C = M \cdot e(g,h)^{\alpha s}, \mathbf{C} = g^{\mathbf{c}})$ *a ciphertext such that* $P(x,y) = 1$*. Define* $\mathbf{E}(x,y)$ *as the matrix such that we have* $\mathbf{cEk}^\intercal = \alpha s$*. Then, we retrieve plaintext* M *by computing*

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

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

We will refer to $(\mathrm{mpk}, \mathbf{k}, \mathbf{c})$ *as the pair encoding scheme (PES) associated with the ABE scheme.*

Each encoding $\mathbf{enc}(\mathrm{var})$ denotes a vector of polynomials over variables var.

Modeling knowledge of exponents. We model the "knowledge of exponents" by extending the space from which the entries of $\mathbf{E}$ and $\mathbf{E}_{\mathrm{att},\mathcal{S},\mathbb{A}}$ are chosen: $\mathbb{Z}_p$. 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 $\mathbb{Z}_p(\mathfrak{K})$ is defined as the quotient field of $\mathbb{Z}_p[\mathfrak{K}]$, where $\mathbb{Z}_p[\mathfrak{K}]$ denotes the polynomial ring in variables $\mathfrak{K}$. We write the elements in $\mathbb{Z}_p(\mathfrak{K})$ as $ab^{-1} \pmod p$, where $a, b \in \mathbb{Z}_p[\mathfrak{K}]$ and $b \neq 0$.

The attacks in the concise notations. We formulate the master-key and decryption attacks (Appendix A.2) below.

Definition 4 (Master-key attacks). *A scheme is vulnerable to a master-key attack if there exist* $y_1, y_2, ... \in \mathcal{Y}$ *and the associated key encodings* $\mathbf{k}_{y_i}$*, and there*

exist $\mathbf{e}_i \in \mathbb{Z}_p(\mathfrak{K})^{\ell_i}$*, where* $\ell_i = |\mathbf{k}_{y_i}|$ *denotes the length of the i-th key encoding, such that* $\sum_i \mathbf{k}_i \mathbf{e}_i^\intercal = \mathsf{mk}$*, where* mk *contains the variable* α*. Then, it holds that for all ciphertext encodings* $\mathbf{c}$ *there exists* $\mathbf{e}' \in \mathbb{Z}_p^{\ell'}$ *(with* $|\mathbf{c}| = \ell'$*) such that* $\mathsf{mk} \cdot \mathbf{e}'\mathbf{c}^\intercal = \alpha s$*.*

Definition 5 (Decryption attacks). *A scheme is vulnerable to a decryption attack if there exist* $y_1, y_2, ... \in \mathcal{Y}$ *and* $x \in \mathcal{X}$ *such that* $P(x, y_i) = 0$ *for all i, associated ciphertext encoding* $\mathbf{c}_x$ *and key encodings* $\mathbf{k}_{y_i}$*, for which there exist* $\mathbf{E}_i \in \mathbb{Z}_p(\mathfrak{K})^{\ell_i \times \ell'}$*, where* $\ell_i = |\mathbf{k}_{y_i}|$ *and* $\ell' = |\mathbf{c}_x|$*, such that* $\sum_i \mathbf{k}_{y_i} \cdot \mathbf{E}_i \cdot \mathbf{c}_x^\intercal = \alpha s$*.*

2.6 Trivial and collusion security

We also distinguish between security against single-key and multiple-key attacks. Our notion of trivial security is derived from the notion of trivially broken by Agrawal and Chase [2].

Definition 6 (Trivial and collusion security). *We call a scheme trivially secure if it is secure against decryption attacks (Definition 5) for a single key and ciphertext. If the scheme is secure against decryption attacks for any number of keys and one ciphertext, we call it collusion secure.*

2.7 The ABGW17 framework

In the ABGW17 framework [3], it is proven that any scheme that is collusion secure is fully secure in the generic group model [25]. Furthermore, it is proven in Theorem 4.1 that it is sufficient to show that the scheme is trivially secure. Previously, Riepel and Wee [20] found a flaw in the proof of Theorem 4.1, and further refined the security definition of ABGW17 to fix the mistake. In this work, we give a counterexample that explicitly contradicts Theorem 4.1 in ABGW17, by breaking one of their newly proposed schemes. In particular, this scheme is trivially secure but not collusion secure. Not only does this show that Theorem 4.1 does not hold, this also impacts the scope of their automated tool: it can only be used to prove trivial security and not collusion security.

2.8 The AC17 framework

In the AC17 framework [2], the PESs have a more restricted form than in Definition 3. In particular, none of the encodings contain any fractions, and the key and ciphertext encodings consist of either polynomials or singleton variables. We also distinguish between two types of key and ciphertext variables: lone and non-lone variables. The lone variables occur only in the polynomials (as monomials). The non-lone variables occur as singletons, and in combination with common variables in the polynomials.

Definition 7 (PES in the AC17 form (PES-AC17) [2]). *A pair encoding scheme for a predicate* $P\colon \mathcal{X} \times \mathcal{Y} \to \{0, 1\}$ *and prime number p, with optionally some additional parameters* par*, is given by four deterministic polynomial-time algorithms as described below.*

- $\text{Param}(\text{par}) \to (n, \mathbf{b})$*: On input* par*, the algorithm outputs* $n \in \mathbb{N}$ *that specifies the number of common variables, which are denoted as* $\mathbf{b} = (b_1, ..., b_n)$*.*
- $\text{EncK}(y, p, \alpha, \mathbf{b}) \to (m_1, m_2, \mathbf{k}(\alpha, \mathbf{r}, \hat{\mathbf{r}}, \mathbf{b}, y))$*: On input* $p \in \mathbb{N}$ *and* $y \in \mathcal{Y}$*, this algorithm outputs a vector of polynomials* $\mathbf{k} = (k_1, ..., k_{m_3})$*, with* $m_3 \in \mathbb{N}$*, defined over non-lone variables* $\mathbf{r} = (r_1, ..., r_{m_1})$ *and lone variables* $\hat{\mathbf{r}} = (\hat{r}_1, ..., \hat{r}_{m_2})$*. Specifically, the polynomial* k_i *is expressed as*

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

  for all $i \in [m_3]$*, where* $\delta_i, \delta_{i,j}, \delta_{i,j,k} \in \mathbb{Z}_p$*.*
- $\text{EncC}(x, p, \mathbf{b}) \to (w_1, w_2, \mathbf{c}(\mathbf{s}, \hat{\mathbf{s}}, \mathbf{b}, x))$*: On input* $p \in \mathbb{N}$ *and* $x \in \mathcal{X}$*, this algorithm outputs a vector of polynomials* $\mathbf{c} = (c_1, ..., c_{w_3})$*, with* $w_3 \in \mathbb{N}$*, defined over non-lone variables* $\mathbf{s} = (s_0 = s, s_1, ..., s_{w_1})$ *and lone variables* $\hat{\mathbf{s}} = (\hat{s}_1, ..., \hat{s}_{w_2})$*. Specifically, the polynomial* c_i *is expressed as*

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

  for all $i \in [w_3]$*, where* $\eta_{i,j}, \eta_{i,j,k} \in \mathbb{Z}_p$*.*
- $\text{Pair}(x, y, p) \to (\mathbf{E}, \overline{\mathbf{E}})$*: On input* $p \in \mathbb{N}$*,* $x \in \mathcal{X}$*, and* $y \in \mathcal{Y}$*, this algorithm outputs two matrices* $\mathbf{E} \in \mathbb{Z}_p^{(w_1+1) \times m_3}$ *and* $\overline{\mathbf{E}} \in \mathbb{Z}_p^{w_3 \times m_1}$*.*

A PES is correct, if for every $x \in \mathcal{X}$ *and* $y \in \mathcal{Y}$ *such that* $P(x, y) = 1$*, it holds that* $\mathbf{s}\mathbf{E}\mathbf{k}^\intercal + \mathbf{c}\overline{\mathbf{E}}\mathbf{r}^\intercal = \alpha s$*.*

A security notion for PES-AC17 is the symbolic property. This property holds if proper vector and matrix substitutions can be found for the common, key and ciphertext variables. In particular, it should hold that substituting the variables with the vectors and matrices ensures that the key and ciphertext polynomials all evaluate to $\mathbf{0}$.

Definition 8 (Symbolic security property (Sym-Prop) [2]). *A pair encoding scheme* $\Gamma = (\text{Param}, \text{EncK}, \text{EncC}, \text{Pair})$ *for a predicate* $P\colon \mathcal{X} \times \mathcal{Y} \to \{0, 1\}$ *satisfies the* (d_1, d_2)*-selective symbolic property for positive integers* d_1 *and* d_2 *if there exist deterministic polynomial-time algorithms* EncB*,* EncS*, and* EncR *such that for all* p*,* par*,* $x \in \mathcal{X}$ *and* $y \in \mathcal{Y}$ *with* $P(x, y) = 0$*, we have that*

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

such that $\mathbf{s}_0 \cdot \mathbf{a}^\intercal \neq 0$*, and if we substitute*

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

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

Similarly, a PES satisfies the (d_1, d_2)-co-selective symbolic security property if there exist EncB, EncR, EncS *that satisfy the above properties but where* EncB *and* EncR *only take y as input, and* EncS *takes x and y as input. A scheme satisfies the (d_1, d_2)-symbolic property if it satisfies the (d'_1, d'_2)-selective and (d''_1, d''_2)-co-selective properties for some $d'_1, d''_1 \leq d_1$ and $d'_2, d''_2 \leq d_2$.*

In [7], Attrapadung defines a slightly stronger version of the symbolic property, called Sym-Prop$^+$*, which additionally requires that $\mathbf{a} = \mathbf{1}_1^{d_1}$. We similarly define the notions of selective and co-selective* Sym-Prop$^+$ *to refer to this additional property in the selective and co-selective proofs.*

Agrawal and Chase [2] prove that any PES satisfying the symbolic property can be transformed in a fully secure ABE scheme. The resulting scheme is fully secure in the standard model under a q-type assumption (that is shown to hold in the generic group model). Furthermore, they prove that any PES that is not vulnerable to any decryption attacks (in the single-key setting) can be transformed into a PES that satisfies the symbolic property.

2.9 The RW22 framework

Recently, Riepel and Wee [20] showed that PES-AC17 schemes (Definition 7) instantiated with Definition 3 that satisfy the following property—which we will refer to as the "FABEO property" but they refer to as the "$(1, 1)$ symbolic security property"—are fully secure in the generic group model, in the multiple-key and multiple-challenge-ciphertext setting. In particular, they prove security by first proving that, if the PES-AC17 satisfies the FABEO property, it is collusion secure.

Definition 9 (The FABEO property [20]). *A PES-AC17 satisfies the FABEO property if, for all $x \in \mathcal{X}$ and $y \in \mathcal{Y}$ with $P(x, y) = 0$, it holds that*

$$\alpha \cdot \mathbf{s} \cdot \mathbf{e}^\intercal \neq \mathbf{s}\mathbf{E}\mathbf{k}^\intercal + \mathbf{c}\overline{\mathbf{E}}\mathbf{r}^\intercal$$

for all $\mathbf{E} \in \mathbb{Z}_p^{(w_1+1)\times m_3}, \overline{\mathbf{E}} \in \mathbb{Z}_p^{w_3 \times m_1}$ and $\mathbf{e} \in \mathbb{Z}_p^{w_1+1}$.

3 Our framework for security proofs and attacks

In this section, we introduce our framework to both finding attacks and proving security. At the core, our framework centers around considering the attacks in a matrix notation, writing the linear-algebraic problems in Definitions 4 and 5 as matrix problems. In this way, we can use important results from linear algebra to prove that no attacks can be found. For trivial security (Definition 6), this follows almost trivially from these results. For collusion security, we require some additional properties to hold. To illustrate these results, we show how they can be applied to an example. The running example that we use is a slightly adapted variant of the Boneh-Boyen IBE1 scheme [10], also given in [7], and is defined as

- $\mathrm{mpk} = (b, b_0, b_1)$
- $\mathbf{k}(\alpha, r, \mathrm{mpk}, y) = (\alpha + rb, r(b_0 + yb_1), r)$
- $\mathbf{c}(s, \mathrm{mpk}, x) = (sb + s'(b_0 + xb_1), s, s')$

3.1 Matrix notation

To automate the security proofs and attacks, we introduce a matrix notation for the encodings. Intuitively, the matrix notation considers the matrix decomposition[6] $\mathbf{p_{enc}}^\intercal = \mathbf{M} \cdot \mathbf{v}^\intercal$ of a vector of encodings $\mathbf{p_{enc}}$ such that:

- each entry of $\mathbf{v}$ consists of a monomial of unknown variables, e.g., $\mathbf{b}, \mathbf{r}$ and $\mathbf{s}$ (as in Definition 3);
- and each entry of $\mathbf{M}$ consists of elements in $\mathbb{Z}_p$ and variables that are known, e.g., x or y, or other elements that are known (e.g., because of corruption).

For the definition of the matrix decomposition, it is important to distinguish between known and unknown variables. For example, the input attribute x to the encryption is often known, while the ciphertext-specific variables $\mathbf{s}$ and the common variables $\mathbf{b}$ are not known to the attacker. To this end, we annotate all variables that are not explicitly written as elements in $\mathbb{Z}_p$ as known or unknown.

Definition 10 (Annotation of variables). *Let* V *denote the set of all variables that occur in some encoding* $\mathbf{p_{enc}}$*. Then, we annotate each variable that is explicitly known by an attacker as* known *and each* $\mathit{unknown}$ *otherwise. This annotation is denoted by a function* $\mathsf{annot}\colon \mathsf{V} \to \{\mathit{known}, \mathit{unknown}\}$*.*

More formally, we define the matrix decomposition as follows.

Definition 11 (Matrix decomposition of encodings). *Let* $\mathbf{p_{enc}}$ *be a vector of encodings and let* annot *be its associated annotation of variables, i.e., each entry of* $\mathbf{p_{enc}}$ *is a polynomial over variables* $\mathsf{var} := \mathsf{annot}^{-1}(\mathit{unknown})$ *and coefficients* $\mathsf{coef} := \mathsf{annot}^{-1}(\mathit{known})$*. We define the matrix decomposition of* $\mathbf{p_{enc}}$ *as*

$$\mathbf{p_{enc}}^\intercal = \mathbf{M} \cdot \mathbf{v}^\intercal,$$

such that each entry of $\mathbf{M}$ *is in the extended field of rational fractions in the variables* coef*, i.e.,* $\mathbb{Z}_p(\mathsf{coef})$*, each entry of* $\mathbf{v}$ *is a monomial over variables* var*, and the entries of* $\mathbf{v}$ *are linearly independent.*

Example. For the example scheme, the unknown variables are $b, b_0, b_1, \alpha, r, s$ and s', whereas the known variables are x and y. We can write the key encodings in matrix notation as follows:

$$\underbrace{(\alpha + rb, r(b_0 + yb_1), r)}_{\mathbf{p_{enc}}} = \underbrace{\begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & y & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}}_{\mathbf{M}} \cdot \underbrace{\begin{pmatrix} \alpha \\ rb \\ rb_0 \\ rb_1 \\ r \end{pmatrix}}_{\mathbf{v}}.$$

[6] Note that vectors are special cases of matrices.

3.2 Finding attacks with the matrix notation

We first explain how the master-key and decryption attacks (Definitions 4 and 5) can be performed in the matrix notation. To this end, we first translate the following equation in the decryption attack to a polynomial notation: $\sum_i \mathbf{k}_{y_i} \cdot \mathbf{E}_i \cdot \mathbf{c}_x^\intercal = \alpha s$. Intuitively, this is equivalent to showing that there exists a linear combination of all products of the entries of $\mathbf{k}_{y_i}$ and the entries of $\mathbf{c}_x$. This follows directly from noting that

$$\mathbf{k}_{y_i} \cdot \mathbf{E}_i \cdot \mathbf{c}_x^\intercal = \sum_{j,k} (\mathbf{E}_i)_{j,k} \cdot (\mathbf{k}_{y_i})_j \cdot (\mathbf{c}_x)_k.$$

Therefore, we define the vector of encodings associated with a PES as follows.

Definition 12 (The vector of encodings associated with a PES). *Let* $(\mathrm{mpk}, \mathbf{k}, \mathbf{c})$ *be a PES for predicate* $P\colon \mathcal{X} \times \mathcal{Y} \to \{0,1\}$*. Then, we define the associated vector of encodings* $\mathbf{p_{enc}}_{x,y}$ *for a decryption attack with* $x \in \mathcal{X}$ *and* $y \in \mathcal{Y}$ *as*

$$\begin{aligned}\mathbf{p_{enc}}_{x,y} = (&\{\mathrm{mpk}_i \cdot \mathbf{k}(\alpha, \mathbf{r}, \mathbf{b}, y)_j\}_{i,j}, \{\mathrm{mpk}_i \cdot \mathbf{c}(\mathbf{s}, \mathbf{b}, x)_j\}_{i,j},\\ &\{\mathbf{k}(\alpha, \mathbf{r}, \mathbf{b}, y)_i \cdot \mathbf{c}(\mathbf{s}, \mathbf{b}, x)_j\}_{i,j})\end{aligned}$$

For a master-key attack for $y \in \mathcal{Y}$*, we define the vector of encodings* $\mathbf{p_{enc}}_y$ *as*

$$\mathbf{p_{enc}}_y = (\{\mathrm{mpk}_i\}_i, \{\mathbf{k}(\alpha, \mathbf{r}, \mathbf{b}, y)_i\}_i).$$

In the case that keys for multiple $y_1, y_2, ... \in \mathcal{Y}$ *are considered, we define*

$$\begin{aligned}\mathbf{p_{enc}}_{x,\{y_1,y_2,...\}} &= (\mathbf{p_{enc}}_{x,y_1}, \mathbf{p_{enc}}_{x,y_2}, ...),\\ \mathbf{p_{enc}}_{\{y_1,y_2,...\}} &= (\mathbf{p_{enc}}_{y_1}, \mathbf{p_{enc}}_{y_2}, ...).\end{aligned}$$

Now, the associated matrix decomposition can be used to define an attack.

Proposition 1. *Suppose that* $\mathbf{p_{enc}}$ *is a vector of encodings, and that the attack dictates that some value* tv *needs to be recovered. Let* $\mathbf{p_{enc}}^\intercal = \mathbf{M} \cdot \mathbf{v}^\intercal$ *be the matrix decomposition, and let* $\mathbf{tv} \cdot \mathbf{v}^\intercal = \mathrm{tv}$ *be the target vector. Then, the following two statements are equivalent.*

(i) There exists $\mathbf{e} \in \mathbb{Z}_p^{|\mathbf{p_{enc}}|}$ *such that* $\mathbf{e} \cdot \mathbf{M} = \mathbf{tv}$*.*
(ii) There exists $\mathbf{e} \in \mathbb{Z}_p^{|\mathbf{p_{enc}}|}$ *such that* $\mathbf{e} \cdot \mathbf{p_{enc}}^\intercal = \mathrm{tv}$*.*

The proof can be found in Appendix B.1.

Example. Suppose that we want to find a master-key attack on the example scheme, i.e., recover α from the key encodings. Then, this is equivalent to showing that there exists a linear combination of the rows of the matrix in the matrix notation of the scheme that yields the target vector $\mathbf{tv} = (1, 0, 0, 0, 0)$, which is the decomposition of α with respect to the vector $\mathbf{v}$.

3.3 Determining whether attacks exist (or not)

Using the matrix decomposition, it becomes easier to argue about the presence or absence of attacks. Specifically, this matrix decomposition also opens up the opportunity to prove that no attacks exist. To this end, we use an important result of linear algebra that is often used in the field of ABE. The most notable example of this is the following property for access structures represented by LSSS matrices (see Definition 2) that many security proofs utilize. If $(1, 0, ..., 0)$ is not in the span of the rows of some matrix $\mathbf{A} \in \mathbb{Z}_p^{n_1 \times n_2}$, then there exists a vector $\mathbf{w} = (1, w_2, ..., w_{n_2}) \in \mathbb{Z}_p^{n_2}$ such that $\mathbf{A} \cdot \mathbf{w}^\intercal = \mathbf{0}^\intercal$. This example is a concrete instantiation of a more general result that was formulated in this context by Goyal et al. [14, §A] (and proven in many works before it):

Proposition 2. *A vector* $\mathbf{tv}$ *is not in the span of the rows of* $\mathbf{M}$ *if and only if there exists a vector* $\mathbf{w}$ *such that* $\mathbf{M} \cdot \mathbf{w}^\intercal = \mathbf{0}^\intercal$ *and* $\mathbf{tv} \cdot \mathbf{w}^\intercal \neq 0$.

From this result, it follows that we can either find an attack, by giving a linear combination of rows, or prove that no trivial attack can be found, by showing that such a vector $\mathbf{w}$ exists. Note that the latter is considerably easier to verify manually than verifying whether some vector $\mathbf{tv}$ is not in the span of some matrix $\mathbf{M}$, as is done in e.g., ABGW17 [3] and RW22 [20].

Example. We can show that the example scheme is secure against master-key attacks by showing that a vector exists that is orthogonal to each row of $\mathbf{M}$ and which is non-zero in the first entry. For example, the vector $\mathbf{w} = (1, -1, 0, 0, 0)$ satisfies this property, as it is non-zero in the first entry (and therefore $\mathbf{tv} \cdot \mathbf{w}^\intercal \neq 0$), and it is easy to see that $\mathbf{M} \cdot \mathbf{w}^\intercal = \mathbf{0}^\intercal$.

3.4 Proving trivial security

We use the result in Proposition 2 to prove that a scheme is trivially secure (Definition 6). In particular, the vector $\mathbf{w}$ in Proposition 2 can function as a witness that proves that no attacks exist in the single-key setting, and thus, that the scheme is trivially secure. Effectively, the vector $\mathbf{w}$ constitutes a proof of trivial security. To verify the proof, one only needs to verify that $\mathbf{M} \cdot \mathbf{w}^\intercal = \mathbf{0}^\intercal$ and $\mathbf{tv} \cdot \mathbf{w}^\intercal$. This is much simpler to do manually than verifying directly that $\mathbf{tv}$ is not in the span of $\mathbf{M}$. Furthermore, this also opens up the opportunity to automatically generate a manually verifiable proof of trivial security with a computer system. In contrast, using a computer system to verify directly whether $\mathbf{tv}$ is in the span of $\mathbf{M}$ requires full trust in the system.

Theorem 1 (Proofs of trivial security). *Let* $(\mathrm{mpk}, \mathbf{k}, \mathbf{c})$ *be a PES for predicate* $P\colon \mathcal{X} \times \mathcal{Y} \to \{0,1\}$*, and consider for all* $x \in \mathcal{X}, y \in \mathcal{Y}$*, the vector of encodings* $\mathbf{p_{enc}}_{x,y}$*, its matrix decomposition* $\mathbf{p_{enc}}_{x,y} = \mathbf{M}_{x,y} \cdot \mathbf{v}_{x,y}^\intercal$ *and target vector* $\mathbf{tv}_{x,y}$ *such that* $\mathbf{tv}_{x,y} \cdot \mathbf{v}_{x,y}^\intercal = \alpha s$*. If for all* $x \in \mathcal{X}, y \in \mathcal{Y}$ *with* $P(x, y) = 0$*, there exists some* $\mathbf{w}_{x,y}^\intercal \in \mathbb{Z}_p(\mathsf{coef})^{\ell_{2,x,y}}$ *(where* $\ell_{2,x,y}$ *is the number of columns of* $\mathbf{M}_{x,y}$*) such that* $\mathbf{M}_{x,y} \cdot \mathbf{w}_{x,y}^\intercal = \mathbf{0}^\intercal$ *and* $\mathbf{tv}_{x,y} \cdot \mathbf{w}_{x,y}^\intercal \neq 0$*, then the PES is trivially secure. We refer to the vectors* $\mathbf{w}_{x,y}$ *as the proof of trivial security.*

The proof can be found in Appendix B.2.

Definition 13 (ACABELLA property for trivial security). *Let* $(\mathrm{mpk}, \mathbf{k}, \mathbf{c})$ *be a PES for predicate* $P\colon \mathcal{X}\times\mathcal{Y} \to \{0,1\}$*. If vectors* $\mathbf{w}_{x,y}$ *exist such as required in Theorem 1, then we say that the PES satisfies the ACABELLA property for trivial security.*

Oftentimes, $\mathbf{p_{enc}}_{x,y}$ consist of many products and even more unique monomials. To reduce the size of the "proof", we can first attempt to reduce the set $\mathbf{p_{enc}}_{x,y}$ by eliminating all elements that are irrelevant to the security analysis. To do this, we can use the following lemma.

Lemma 1. *Let* $\mathbf{p_{enc}}'_{x,y} \subseteq \mathbf{p_{enc}}_{x,y}$ *be the subset of* $\mathbf{p_{enc}}_{x,y}$ *such that* $\mathbf{p_{enc}}'_{x,y}$ *and* $\mathbf{p_{enc}}''_{x,y} = \mathbf{p_{enc}}_{x,y} \setminus \mathbf{p_{enc}}'_{x,y}$ *have no monomials in common and the target monomial* αs *occurs in at least one of the elements in* $\mathbf{p_{enc}}'_{x,y}$*. Then, we have*

$$\mathbf{p_{enc}}_{x,y} = \mathbf{M}_{x,y} \cdot \mathbf{v}_{x,y}^{\intercal} = \begin{pmatrix} \mathbf{M}'_{x,y} & \mathbf{0} \\ \mathbf{0} & \mathbf{M}''_{x,y} \end{pmatrix} \cdot \begin{pmatrix} (\mathbf{v}'_{x,y})^{\intercal} \\ (\mathbf{v}''_{x,y})^{\intercal} \end{pmatrix},$$

where $\mathbf{p_{enc}}'_{x,y} = \mathbf{M}'_{x,y} \cdot (\mathbf{v}'_{x,y})^{\intercal}$ *and* $\mathbf{p_{enc}}'_{x,y} = \mathbf{M}'_{x,y} \cdot (\mathbf{v}'_{x,y})^{\intercal}$*. We also have* $\mathtt{tv}_{x,y} = (\mathtt{tv}'_{x,y}, \mathbf{0})$*, where* $|\mathtt{tv}'_{x,y}| = |\mathbf{v}'_{x,y}|$*. Furthermore, for any* $\mathbf{w}'_{x,y}$ *for which we have* $\mathbf{M}'_{x,y} \cdot (\mathbf{w}'_{x,y})^{\intercal} = \mathbf{0}$ *and* $\mathtt{tv}'_{x,y} \cdot (\mathbf{w}'_{x,y})^{\intercal} \neq 0$*, it holds that* $\mathbf{w}_{x,y} = (\mathbf{w}'_{x,y}, \mathbf{0})$ *is such that* $\mathbf{M}_{x,y} \cdot \mathbf{w}_{x,y}^{\intercal} = \mathbf{0}$ *and* $\mathtt{tv}_{x,y} \cdot \mathbf{w}_{x,y}^{\intercal} \neq 0$*.*

The proof can be found in Appendix B.3.

Example. We show that our example scheme is trivially secure. Because $\mathbf{p_{enc}}_{x,y}$ contains 27 products (consisting of even more unique monomials), we first reduce the set $\mathbf{p_{enc}}_{x,y}$ to $\mathbf{p_{enc}}'_{x,y} = (\alpha s + rsb, rs(b_0 + yb_1), \alpha s' + rs'b, rs'(b_0 + yb_1), rsb + rs'(b_0 + xb_1))$. This set has no monomials in common with all other products in $\mathbf{p_{enc}}_{x,y}$, because each combination has either one lone variable and one non-lone variable or two non-lone variables and one common variable, and the other products do not have any monomials of this form. We now consider the vector decomposition

$$\mathbf{p_{enc}}'_{x,y} = \begin{pmatrix} 1&1&0&0&0&0&0&0 \\ 0&0&1&y&0&0&0&0 \\ 0&0&0&0&1&1&0&0 \\ 0&0&0&0&0&0&1&y \\ 0&1&0&0&0&0&1&x \end{pmatrix} \cdot \begin{pmatrix} \alpha s \\ rsb \\ rsb_0 \\ rsb_1 \\ \alpha s' \\ rs'b \\ rs'b_0 \\ rs'b_1 \end{pmatrix},$$

and find that $\mathbf{w}'_{x,y} = (1, -1, 0, 0, 0, 0, \frac{y}{y-x}, \frac{-1}{y-x})$ satisfies the requirements to prove trivial security.

3.5 Proving collusion security

We use a similar approach for proving collusion security by showing that some vector $\mathbf{w}$ exists, for which some additional properties hold that are simple to verify. In general, collusion security is more complicated to prove than trivial security, because it allows any polynomially bounded number of attributes $y_i \in \mathcal{Y}$ to be used. In particular, the attack in Section 6.3 shows that trivial security does not imply collusion security. Riepel and Wee [20] prove collusion security via the FABEO property (Definition 9). However, it is unclear how this property can be generalized to the more general class of pair encodings as considered in this paper and ABGW17 [3]. Hence, we devise a technique that does translate to the more general setting. Furthermore, as we show in Section 4.3, our property is implied by the FABEO property for PES-AC17, and generally appears to be weaker.

To prove collusion security, we need to prove that, for all x and $y_1, y_2, ...$ with $P(x, y_i) = 0$, it holds that $\sum_i \mathbf{k}_{y_i} \cdot \mathbf{E}_i \cdot \mathbf{c}_x^\intercal \neq \alpha s$ for all matrices $\mathbf{E}_i$. With the methods in Section 3.2, we can translate this problem easily to the matrix notation. If we can show that for all such x and $y_1, y_2, ...$, it holds that a vector exists that yields the all-zero vector when multiplied with the matrix and is non-zero when multiplied with the target vector, then we know that the scheme is collusion secure.

Now, rather than finding such a vector for all possible x and $y_1, y_2, ...$, we will show how a suitable vector can be constructed from a vector with special properties for a single x and y_i. To this end, we first note that $\mathbf{penc}_{x,\{y_1,y_2,...\}}$ can be constructed from $\mathbf{penc}_{x,y_i}$ for all i. Their associated matrix decompositions relate to one another similarly, with an important difference: there may be overlap in the entries of their associated vectors $\mathbf{v}_{x,y_i}$. Hence, to create a matrix decomposition for $\mathbf{penc}_{x,\{y_1,y_2,...\}}$ from the matrix decompositions of $\mathbf{penc}_{x,y_i}$, we must identify the entries of vectors $\mathbf{v}_{x,y_i}$ that may occur in more than one vector, and those that are unique to one key. For those that may be shared by more than one vector, we also distinguish between those that occur in the target value and those that do not. To this end, we define the following function.

Definition 14 (Shared-entry function). *Let* $(\mathrm{mpk}, \mathbf{k}, \mathbf{c})$ *be a PES for predicate* $P\colon \mathcal{X} \times \mathcal{Y} \to \{0,1\}$*, and consider the matrix decompositions of* $\mathbf{penc}_{x,y_i}$ *for* $x \in \mathcal{X}$ *and* $y_1, y_2, ... \in \mathcal{Y}$*, and their associated target vectors* $\mathbf{tv}_{x,y_i}$*. Then, we define the shared-entries function* $\mathfrak{f}\colon \mathbb{N} \times \mathbb{N} \to \{0,1,2\}$*, such that* $\mathfrak{f}$ *takes as input a pair* (i,j)*, where* i *corresponds to the identifier of* y_i *and* j *to the* j*-th entry of* $\mathbf{v}_{x,y_i}$*, and outputs*

- 2 *if* $(\mathbf{v}_{x,y_i})_j$ *is divisible by a user-specific random variable (i.e., an entry of* $\mathbf{r}$*), and otherwise*
- 0 *if* $(\mathbf{tv}_{x,y_i})_j \neq 0$ *and*
- 1 *if* $(\mathbf{tv}_{x,y_i})_j = 0$.

We use this function to construct a matrix decomposition for $\mathbf{penc}_{x,\{y_1,y_2,...\}}$ from some matrix decompositions of $\mathbf{penc}_{x,y_i}$ as follows.

Proposition 3. *Let* $(\mathrm{mpk}, \mathbf{k}, \mathbf{c})$ *be a PES for predicate* $P\colon \mathcal{X} \times \mathcal{Y} \to \{0,1\}$*, and consider the matrix decompositions of* $\mathbf{p_{enc}}_{x,y_i}$ *for* $x \in \mathcal{X}$ *and* $y_1, y_2, ... \in \mathcal{Y}$*, and their associated target vectors* $\mathtt{tv}_{x,y_i}$*. Let* $\mathfrak{f}$ *be the shared-entry function for these encodings. Without loss of generality, we assume that the matrix decompositions* $\mathbf{p_{enc}}^{\intercal}_{x,y_i} = \mathbf{M}_{x,y_i} \cdot \mathbf{v}^{\intercal}_{x,y_i}$ *are such that* $\mathbf{v}_{x,y_i}$ *is ordered: the first entries evaluate to* 0 *with* $\mathfrak{f}$*, the entries after that to* 1 *and the last entries to* 2*, i.e.,*

$$\mathbf{v}_{x,y_i} = (\mathbf{v}_{x,y_i,0}, \mathbf{v}_{x,y_i,1}, \mathbf{v}_{x,y_i,2}).$$

Similarly, we split $\mathbf{M}_{x,y_i}$ *into three submatrices* $\mathbf{M}_{x,y_i,j}$ *for* $j \in \{0,1,2\}$ *such that* $\mathbf{M}_{x,y_i,j}$ *denote the columns of* $\mathbf{M}_{x,y_i}$ *that correspond to* $\mathbf{v}_{x,y_i,j}$*. If it holds that* $(\mathbf{v}_{x,y_i,0}, \mathbf{v}_{x,y_i,1}) = (\mathbf{v}_{x,y_j,0}, \mathbf{v}_{x,y_j,1})$ *for all* i, j*, then we can create a matrix decomposition for* $\mathbf{p_{enc}}_{x,\{y_1,y_2,...\}}$ *as follows:*

$$\mathbf{M}_{x,\{y_1,...,y_n\}} = \begin{pmatrix} \mathbf{M}_{x,y_1,0} & \mathbf{M}_{x,y_i,1} & \mathbf{M}_{x,y_i,2} & \mathbf{0} & \mathbf{0} \\ \mathbf{M}_{x,y_2,0} & \mathbf{M}_{x,y_2,1} & \mathbf{0} & \mathbf{M}_{x,y_i,2} & \mathbf{0} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \mathbf{M}_{x,y_n,0} & \mathbf{M}_{x,y_n,1} & \mathbf{0} & \mathbf{0} & \mathbf{M}_{x,y_n,2} \end{pmatrix},$$

$$\mathbf{v}_{x,\{y_1,...,y_n\}} = (\mathbf{v}_{x,y_1,0}, \mathbf{v}_{x,y_1,1}, \mathbf{v}_{x,y_1,2}, \mathbf{v}_{x,y_2,2}, \ldots, \mathbf{v}_{x,y_n,2}).$$

The associated target vector is $\mathtt{tv}_{x,\{y_1,...,y_n\}} = (\mathtt{tv}_{x,y_1,0}, \mathbf{0})$*, where* $\mathtt{tv}_{x,y_1,0}$ *denotes the first entries of* $\mathtt{tv}_{x,y_1}$ *that corresponds to the entries* $\mathbf{v}_{x,y_1,0}$*.*

The proof can be found in Appendix B.4. We use a similar approach to construct a vector that proves collusion security from vectors that prove trivial security. To do this, we need additional properties for the vectors $\mathbf{w}_{x,y}$ that are used in the trivial-security proof (Theorem 1). In particular, we require the first entries of the vectors $\mathbf{w}_{x,y}$ (corresponding to the entries of $\mathbf{v}_{x,y}$ that are shared among all matrix decompositions) to be equal. Then, we can similarly construct a vector $\mathbf{w}_{x,\{y_1,y_2,...\}}$ as in Proposition 3 that proves the absence of the target vector in the span.

Theorem 2 (Proofs of collusion security). *Let* $(\mathrm{mpk}, \mathbf{k}, \mathbf{c})$ *be a PES for predicate* $P\colon \mathcal{X} \times \mathcal{Y} \to \{0,1\}$*, and consider the matrix decompositions of* $\mathbf{p_{enc}}_{x,y_i}$ *for* $x \in \mathcal{X}$ *and* $y_1, y_2, ..., y_n \in \mathcal{Y}$*, and their associated target vectors* $\mathtt{tv}_{x,y_i}$*. Like in Proposition 3, we assume that the matrix decompositions* $\mathbf{p_{enc}}^{\intercal}_{x,y_i} = \mathbf{M}_{x,y_i} \cdot \mathbf{v}^{\intercal}_{x,y_i}$ *are such that* $\mathbf{v}_{x,y_i}$ *is ordered. If for all* $x \in \mathcal{X}, y_1, y_2, ..., y_n \in \mathcal{Y}$ *with* $P(x, y_i) = 0$*,* $(\mathbf{v}_{x,y_i,0}, \mathbf{v}_{x,y_i,1}) = (\mathbf{v}_{x,y_j,0}, \mathbf{v}_{x,y_j,1})$ *for all* i, j*, and there exist vectors* $\mathbf{w}_{x,y_i}$ *(as in Theorem 1) such that the requirements below hold, then the PES is collusion secure. First, we similarly split each vector* $\mathbf{w}_{x,y_i}$ *(which has the same length as* $\mathbf{v}_{x,y_i}$*) in three parts, i.e.,* $\mathbf{w}_{x,y_i} = (\mathbf{w}_{x,y_i,0}, \mathbf{w}_{x,y_i,1}, \mathbf{w}_{x,y_i,2})$*, such that* $|\mathbf{w}_{x,y_i,j}| = |\mathbf{v}_{x,y_i,j}|$ *for* $j \in \{0,1,2\}$*. Then, the requirements are*

- $\mathbf{M}_{x,y_i} \cdot \mathbf{w}^{\intercal}_{x,y_i} = \mathbf{0}^{\intercal}$ *and* $\mathtt{tv}_{x,y_i} \cdot \mathbf{w}^{\intercal}_{x,y_i} \neq 0$ *(i.e., the PES is trivially secure);*
- $\mathbf{w}_{x,y_i,0} = \mathbf{w}_{x,y_j,0}$ *for all* i, j*, and*
- $\mathbf{w}_{x,y_i,1} = \mathbf{0}$*.*

The proof can be found in Appendix B.5.

Definition 15 (ACABELLA property for collusion security). *Let* $(\mathrm{mpk}, \mathbf{k}, \mathbf{c})$ *be a PES for predicate* $P\colon \mathcal{X} \times \mathcal{Y} \to \{0,1\}$*. If vectors* $\mathbf{w}_{x,y_i}$ *exist such as required in Theorem 2, then we say that the PES satisfies the ACABELLA property for collusion security.*

For schemes with rational fractions, the effort of proving collusion security is a little more tricky. To this end, we provide several additional results to find suitable vectors that prove security of the schemes in Appendix C.

Example. We show that our example scheme is also collusion secure. In particular, the vector $\mathbf{w}_{x,y} = (\mathbf{w}'_{x,y}, \mathbf{0})$ satisfies the properties we require for collusion security, because the entry associated with αs is always 1, for any x and y, and the only other shared entry in the vector $\mathbf{v}'_{x,y}$ is $\alpha s'$ and the entry associated with it in $\mathbf{w}'_{x,y}$ is 0. Furthermore, all other entries of $\mathbf{w}_{x,y}$ are 0 and therefore, all entries in $\mathbf{w}_{x,y}$ that are associated with the other shared entries in the vector $\mathbf{v}_{x,y}$ are 0 by definition.

The flaw in the ABGW17 proof of Theorem 4.1. As mentioned in Section 2.7, the proof of Theorem 4.1 in ABGW17 contains a flaw. Specifically, it is assumed in the proof that the number of non-lone ciphertext variables is 1, but it is never shown that the proof also works for more than one non-lone ciphertext variable. Our work underlines this statement for PES-AC17. If a ciphertext has only one non-lone ciphertext variable, then the ACABELLA properties for trivial and collusion security are equivalent. This is more or less because $|\mathbf{w}_{x,y_i,0}| = 1$ and $|\mathbf{w}_{x,y_i,1}| = 0$. If there is more than one non-lone ciphertext variable, then the two properties are not equivalent. Lastly, if the scheme is fractional, it seem less clear what constitutes a non-lone variable, and therefore, it is unclear whether these equivalences hold.

3.6 Notation for the proofs

The vectors in the matrix decomposition and the vectors that constitute the proof of trivial or collusion security are closely related, and imply a simple notation for the proofs. This simple notation allows us to verify the proofs without having to consider the matrix $\mathbf{M}_{x,y}$, but rather allows us to verify the proofs directly in the encodings. In particular, the vectors that constitute the proofs of trivial or collusion security can be seen as substitution vectors for the unknown variables in the encodings, similarly as in the symbolic property (Definition 8). Suppose that $\mathbf{v}$ is the vector of a matrix decomposition, and $\mathbf{w}$ is the associated proof of security, then we can write the substitutions as a list: $(\mathbf{v})_i \ : \ (\mathbf{w})_i$. Furthermore, we will also list the terms associated with the first part of the vector $\mathbf{v}$, i.e., $\mathbf{v}_{x,y,0}$ as "terms associated with the blinding value". Similarly, we call the second part the "special terms that are shared among keys and are

not associated with the blinding value" and the third part the remainder of the terms. We will refer to this list of substitutions as a transcript of the proof.

To verify the proof, one can compute all products of key and ciphertext encodings (and potentially, products of the key and ciphertext encodings with the master public key encodings), substitute the resulting monomials that occur in the polynomial as indicated in the transcript, and verify that the products of encodings evaluate to 0. Because there are potentially many such products, one may want to filter out non-critical products. For some products of key and ciphertext encodings, it is clear that they cannot contribute to an attack. For example, their product may contain a monomial that cannot be created with any other product of key and ciphertext encodings (and does not occur in the blinding value). Intuitively, this monomial cannot be canceled in any linear combination. More formally, the entries in $\mathbf{w}$ associated with this monomial is independent of the entries associated with the other monomials, and can thus be set to any value to ensure that it is orthogonal to all rows of $\mathbf{M}$. We refer to Appendix B.6 for the lemmas and their proofs.

Example. The proof of our example scheme can be written as

$$\begin{pmatrix} \alpha s \\ rsb \\ rsb_0 \\ rsb_1 \\ \alpha s' \\ rs'b \\ rs'b_0 \\ rs'b_1 \end{pmatrix} : \begin{pmatrix} 1 \\ -1 \\ 0 \\ 0 \\ 0 \\ 0 \\ \frac{y}{y-x} \\ \frac{-1}{y-x} \end{pmatrix} \text{ or } \begin{array}{c} \alpha s \ : \ 1 \\ rsb \ : \ -1 \\ rsb_0 \ : \ 0 \\ rsb_1 \ : \ 0 \\ \alpha s' \ : \ 0 \\ rs'b \ : \ 0 \\ rs'b_0 \ : \ \frac{y}{y-x} \\ rs'b_1 \ : \ \frac{-1}{y-x} \end{array} .$$

3.7 Proofs for unbounded-size predicates

Although our tool can currently only handle input predicates of bounded sizes, the security proofs output by the tool can be used to argue that a scheme is secure for predicates of any size. This is in contrast to the ABGW17 tool, which can only handle and prove security for bounded-size predicates. Intuitively, the approach to show that a security proof generalizes to predicates of any size is similar to the approach to prove collusion security.

Recall that, to prove security for multiple key attributes $y_i \in \mathcal{Y}$, we require some properties to hold for the proof vector $\mathbf{w}_{x,y_i}$ for one such attribute y_i. If these properties hold, we can then argue that a vector can be created for any number of attributes y_i. Roughly, the requirements for the proof vector is that, for all entries that are associated with monomials that occur in each vector $\mathbf{v}_{x,y_i}$ (which we call the "shared entries"), the vector $\mathbf{w}_{x,y_i}$ should be the same for all y_i. In this way, we can construct a vector $\mathbf{w}_{x,y_1,y_2,\ldots}$ by simply appending the part of $\mathbf{w}_{x,y_2}$ that corresponds to the non-shared (or: unique) entries to the vector $\mathbf{w}_{x,y_1}$.

We can do something similar to show that a proof can be generalized to a proof for any predicate, and not just those of some bounded size. To do this properly, care needs to be taken in the choice of predicates x and y. For example, for CP-ABE with policies represented by LSSS matrices, it is important that the set contains attributes that do and do not occur in the policy, and vice versa. For example, the set could contain $y = \{\text{att}_1, \text{att}_2\}$ and the policy x could be for attributes att_1 and att_3. Once we have a proof, we can inspect the vector $\mathbf{w}_{x,y}$. If it holds that, for each entry that depends somehow partially on the input predicates x and y, the associated entry in $\mathbf{v}_{x,y}$ consists of a lone or non-lone variable that is unique for these partial predicates. For example, if an entry depends on attributes att_1 and att_3 in our example, the associated entry in $\mathbf{v}_{x,y}$ should be a product of a user-specific variable unique to att_1 and a ciphertext-specific variable unique to att_3. Conversely, this ensures that the entries of $\mathbf{w}_{x,y}$ that are not associated with the attribute-specific non-lone and lone variables are equal. (These entries can be considered the "shared entries" for a single pair (x, y).) If these are independent of the predicates x, y, then the proof generalizes.

4 Security proofs for PES-AC17

For schemes associated with a pair encoding scheme that fits the definition of PES-AC17 (Definition 7), we prove three results. First, we prove that our ACABELLA property for trivial security and the symbolic property are equivalent. We prove this constructively, by devising an algorithm that, given a PES-AC17, generates a symbolic property proof. This algorithm can provably find a proof if the input PES-AC17 is trivially secure. We also prove that our ACABELLA property for collusion security is equivalent to selective $\mathsf{Sym\text{-}Prop}^{+}$. Lastly, we show that the FABEO property (Definition 9) implies the collusion-security property. To prove this, we also show how the FABEO property can be proven in a similar fashion as the trivial-security property.

4.1 Proving the symbolic property

We first show constructively how the symbolic property can be proven if a scheme is trivially secure. Agrawal and Chase [2] have already shown this, but their proof is not sufficient to use it to build matrices and vectors as required by the symbolic property. The most notable reason is that their proof technique requires both the $x \in \mathcal{X}$ and $y \in \mathcal{Y}$ to be known ahead of the generation of the matrices. Although this is sufficient to prove that the symbolic property is implied by trivial security, it is not sufficient to construct the required matrices and vectors. To this end, we improve on their proof by creating a construction-based proof.

We explain how our proof works for the selective symbolic property. Our proof for the co-selective symbolic property is similar. Recall that, in a selective proof, the challenge $x^* \in \mathcal{X}$ is given ahead of the instantiation of the scheme. We therefore obtain it as input to the generation of the master public key MPK and the generation of the secret keys. Thus, we can use it in the construction of

matrices for common variables $\mathbf{b}$ and vectors for key encodings $\mathbf{k}$ (the latter of these also obtaining $y \in \mathcal{Y}$ as input). In particular, we first compute the matrix decomposition of $\mathbf{c}(\mathbf{s}, \hat{\mathbf{s}}, \mathbf{b}, x^*)^\intercal = \mathbf{M}_c \cdot \mathbf{v}_c^\intercal$, and consider a vector basis $\mathcal{V}_c$ of the kernel $\mathrm{Ker}(\mathbf{M}_c)$. For each vector in the kernel, we obtain a substitution vector for the monomials of $\mathbf{c}$, i.e., $\hat{s}_j$ and $s_j b_k$. By the definition of the kernel, this vector yields 0 if the variables $\hat{s}_j$ and $s_j b_k$ are substituted with these entries. For the variables $\hat{s}_j$, we can directly use the basis $\mathcal{V}_c$ to construct the vectors from the associated entries in each basis vector. For the products $s_j b_k$, we need to apply an additional step. Specifically, we set the substitution vector to be the standard basis vector $\mathbf{s}_j = \mathbf{1}_{j+1}^{w_1+1}$. Intuitively, we do this to ensure that multiplying $\mathbf{s}_j$ with the substitution matrix $\mathbf{B}_k$ of b_k selects the $(j+1)$-th row of the substitution matrix. Then, for each common variable b_k, we consider the entries of the vectors in $\mathcal{V}_c$ corresponding to each $s_j b_k$, and construct the $(j+1)$-th row of $\mathbf{B}_k$ using these entries. Now, we have obtained matrices $\mathbf{B}_k$ such that evaluating $\mathbf{s}_j \cdot \mathbf{B}_k$ in the polynomials of $\mathbf{c}$ corresponds to computing the basis vectors $\mathcal{V}_c$. This ensures that the substitutions evaluates to $\mathbf{0}$.

To construct the substitution vectors for the key encodings $\mathbf{k}$, we use that the scheme is trivially secure. In the context of PES-AC17, this means that for every $x \in \mathcal{X}$ and $y \in \mathcal{Y}$ such that $P(x, y) = 0$, it holds that $\mathbf{sEk}^\intercal + \mathbf{c}\overline{\mathbf{E}}\mathbf{r}^\intercal \neq \alpha s$. We similarly apply Proposition 2 as in our proofs for security (Theorem 1). Consider the vector of encodings

$$\mathbf{p_{enc}}'_{x,y} = (\{s_j k_{j'} \mid j \in \overline{[w_1]}, j' \in [m_3]\}, \\ \{r_j c_{j'} \mid j \in [m_1], j' \in [w_3]\}),$$

and the associated matrix composition $\mathbf{M}_{x,y} \cdot \mathbf{v}_{x,y}^\intercal$, where we assume that $(\mathbf{v}_{x,y})_1 = \alpha s$. With Proposition 2, it follows that a vector $\mathbf{w}_{x,y}$ exists with $\mathbf{M}_{x,y} \cdot \mathbf{w}_{x,y}^\intercal = \mathbf{0}$ and $\mathbf{tv}_{x,y} \cdot \mathbf{w}_{x,y}^\intercal \neq 0$. Because $\mathbf{tv}_{x,y} = (1, 0, ..., 0)$, we therefore know that the first entry of $\mathbf{w}_{x,y}$ is non-zero, and specifically, we may assume that it is 1. (If not, we can simply "normalize" the vector.) This vector $\mathbf{w}_{x^*,y}$ implies substitutions for $\hat{r}_j$ and α, by considering the entries associated with $\hat{r}_j s_{j'}$ and $\alpha s_{j'}$ in a similar fashion as for $\hat{s}_j$. In particular, we construct the $(j+1)$-th entry of the vector $\mathbf{a}$ as the entry in $\mathbf{w}_{x,y}$ corresponding to $\alpha s_{j'}$, and similarly create the vectors $\hat{\mathbf{r}}_j$. For the substitution vectors $\mathbf{r}_j$ of r_j, we need to apply some extra steps. We first consider, for each r_j, the entries of $\mathbf{w}_{x,y}$ associated with the entries $r_j s_{j'} b_k$ for all j', k. We use these entries to construct a vector $\mathbf{w}'_{x^*}$, which we show to be in the kernel of the matrix $\mathbf{M}_c$ of the matrix decomposition of $\mathbf{c}$. In particular, this is the case, because the rows of $\mathbf{M}_{x^*,y}$ associated with the products $r_j c_{j'}$ correspond to the row of $\mathbf{M}_c$ associated with $c_{j'}$. It follows from this fact that $\mathbf{w}'_{x^*}$ is in the kernel of $\mathbf{M}_c$. This means that there exists a linear combination of the basis vectors $\mathcal{V}_c$ that is equal to $\mathbf{w}'_{x^*}$. This linear combination implies the substitution vectors $\mathbf{r}_j$ of r_j.

Theorem 3 (Proofs for the symbolic property.). *Let* (Param, EncK, EncC, Pair) *be a PES-AC17 for predicate* $P\colon \mathcal{X} \times \mathcal{Y} \to \{0, 1\}$. *Then, it holds that the PES-AC17 is trivially secure if and only if the symbolic property holds.*

The proof can be found in Appendix B.7.

4.2 Proving Sym-Prop$^+$

From Theorem 3, it follows rather quickly that our ACABELLA property for collusion security and selective Sym-Prop$^+$ are equivalent. From this, it also follows that (selective) Sym-Prop$^+$ implies collusion security.

Corollary 1. *Let* (Param, EncK, EncC, Pair) *be a PES-AC17 for the predicate* $P\colon \mathcal{X} \times \mathcal{Y} \to \{0,1\}$. *Then, it holds that the PES-AC17 is collusion secure if the selective symbolic property* Sym-Prop$^+$ *holds. Furthermore, if our property that we use to prove collusion security holds (Theorem 2), then the PES-AC17 satisfies selective* Sym-Prop$^+$.

The proof can be found in Appendix B.8.

4.3 Proving the FABEO property

We give a similar way to proving the FABEO property as we prove the trivial and collusion security properties. To prove the FABEO property, we need to show that, for all $x \in \mathcal{X}$ and $y \in \mathcal{Y}$ with $P(x,y) = 0$, it holds that

$$\alpha \cdot \mathbf{s} \cdot \mathbf{e}^\intercal \neq \mathbf{sEk}^\intercal + \mathbf{c}\overline{\mathbf{E}}\mathbf{r}^\intercal.$$

This is equivalent to proving that, for the matrix decomposition $\mathbf{M}_{x,y} \cdot \mathbf{v}_{x,y}^\intercal$ as in Equation 4.1, it holds that the intersection of the span of rows of the matrix and the span of rows $\mathfrak{V} = \{\mathbf{1}_1^{|\mathbf{v}|}, ..., \mathbf{1}_{w_1+1}^{|\mathbf{v}|}\}$ is empty, where we assume, without loss of generality, that the first $w_1 + 1$ entries of $\mathbf{v}_{x,y}$ are associated with αs_j.

To prove this, we show that vectors $\bar{\mathbf{w}}_{x,y,1}, ..., \bar{\mathbf{w}}_{x,y,w_1+1}$ exist with $(\bar{\mathbf{w}}_{x,y,i})_i = 1$, such that $\mathbf{M}_{x,y} \cdot \bar{\mathbf{w}}_{x,y,i}^\intercal = \mathbf{0}^\intercal$ for all $i \in [w_1 + 1]$. Then, it follows that no linear combination of vectors in $\mathfrak{V}$ exists that is in the span of $\mathbf{M}_{x,y}$, and thus, the FABEO property holds.

Proposition 4 (Proofs for the FABEO property). *Let* (Param, EncK, EncC, Pair) *be a PES-AC17 for predicate* $P\colon \mathcal{X} \times \mathcal{Y} \to \{0,1\}$. *Let* $\mathbf{M}_{x,y} \cdot \mathbf{v}_{x,y}^\intercal$ *be the matrix decomposition as in Equation 4.1, for which we assume that the first entries of* $\mathbf{v}_{x,y}$ *are* $\alpha s_0, \alpha s_1, ..., \alpha s_{w_1}$. *Then, it holds that the PES-AC17 satisfies the FABEO property if, for all* $x \in \mathcal{X}$ *and* $y \in \mathcal{Y}$ *with* $P(x,y) = 0$, *there exist vectors* $\bar{\mathbf{w}}_{x,y,1}, ..., \bar{\mathbf{w}}_{x,y,w_1+1}$ *with* $(\bar{\mathbf{w}}_{x,y,i})_i = 1$ *and* $(\bar{\mathbf{w}}_{x,y,i})_{i'} = 0$ *for* $i' \in [w_1+1]$ *with* $i' \neq i$, *such that* $\mathbf{M}_{x,y} \cdot \bar{\mathbf{w}}_{x,y,i}^\intercal = \mathbf{0}^\intercal$ *for all* $i \in [w_1 + 1]$.

The proof can be found in Appendix B.9. From Proposition 4, it follows trivially that the collusion security property holds, because $\bar{\mathbf{w}}_{x,y,1}$ also satisfies the properties required for Theorem 2.

Corollary 2. *Let* (Param, EncK, EncC, Pair) *be a PES-AC17 for predicate* $P\colon \mathcal{X} \times \mathcal{Y} \to \{0,1\}$ *that satisfies the FABEO property. Then, it is also collusion secure.*

Definition 16 (ACABELLA property for FABEO). *Let* $(\mathrm{mpk}, \mathbf{k}, \mathbf{c})$ *be a PES-AC17 for predicate* $P\colon \mathcal{X} \times \mathcal{Y} \to \{0,1\}$. *If vectors* $\bar{\mathbf{w}}_{x,y_i,j}$ *exist such as required in Proposition 4, then we say that the PES-AC17 satisfies the ACABELLA property for FABEO.*

5 Description of the ACABELLA tool

The ACABELLA tool[7] analyzes the security of ABE schemes by using the results of our theoretical framework. In this section, we describe the different components of the tool: how it proves the security of an ABE scheme and how it looks for master-key and decryption attacks in case security cannot be proven. The ACABELLA tool is written in Python 3.10 and relies on the linear-algebra methods provided by Sympy 1.11.1 [19] and described in Appendix D. For a direct experience with the ACABELLA tool, we refer to https://www.acabel.la, which provides the ACABELLA tool as a web application. This application also contains many examples in which the security of various ABE schemes is analyzed.

5.1 Analysis functionality

Given the PES of an ABE scheme, the ACABELLA tool automates the analysis of the scheme. Depending on the configuration, it automates the generation and verification of different security properties, and looks for the existence of attacks in case security cannot be proven. More specifically, it performs the following steps.

Security analysis: First, the tool checks whether the PES satisfies the AC17 form (Definition 7). If the PES is a PES-AC17, then it performs the following steps.

- The tool checks if the scheme is trivially and collusion secure, by using the results in Theorem 3 and Corollary 1.
- If the scheme is collusion secure, then it also generates a proof for the symbolic property. It verifies the proofs before giving them as output.
- Lastly, the tool checks whether the PES-AC17 satisfies the FABEO property, by using the results in Proposition 4.

If the PES is not a PES-AC17, then the tool performs the following steps.

- The tool checks if the scheme is trivially and collusion secure, by using the results in Theorems 1 and 2.
- If the scheme is collusion secure, then it also generates a transcript that functions as a proof of trivial and collusion security (Section 3.6).

Finally, using the analysis report, the practitioner can look further into the possibility of attacking the scheme via master-key attacks and decryption attacks.

[7] The ACABELLA tool is published under the GNU General Public License, Version 3 and it is available at https://github.com/abecryptools/ACABELLA

Finding master-key and decryption attacks: To find master-key and decryption attacks, our tool automates some of the techniques given in the VA21 framework (Section 2.5). In particular, it automates the search for master-key and decryption attacks, given the input encodings, using the matrix decomposition (Definition 11). In addition, much like in the VA21 framework, we allow the attacker to corrupt authorities in the multi-authority setting. Note that this works only for multi-authority schemes that fit roughly our definition of pair encodings (Definition 3). Recall that, depending on the scheme, various authorities, e.g., CAs and AA_i, might be deployed to run certain parts of the schemes, notably, the setup and key generation. In the tool, one can specify the knowledge of the various authorities by listing, for each authority, the variables that they learn.

Furthermore, it depends on the security model whether the scheme is supposed to prove security against corruption of these authorities or not. With ACABELLA, it is possible to consider attacks under corruption. This can be done by describing which variables are obtained via corruption when performing the analysis with respect to master-key and decryption attacks.

Our tool additionally supports several ready-to-use corruption models based on the attacks in the VA21 framework [27]. Using these models, the tool explains which authorities need to be corrupted in order to perform the attack. For the master-key attacks, we support the following corruption models:

- **AA:** the scheme employs multiple attribute authorities AA_i, and one attribute authority AA_i is corrupted.
- **Mixed CA corruption:** the scheme employs a central authority and multiple attribute authorities, and the CA is corrupted.
- **Mixed AA corruption:** the scheme employs a central authority and multiple attribute authorities, and one attribute authority AA_i is corrupted.

Furthermore, for the decryption attacks, we support the following corruption models:

- **AA:** the scheme employs multiple attribute authorities AA_i, and one attribute authority AA_i is corrupted.
- **AA extended:** the scheme employs multiple attribute authorities AA_i, and one attribute authority AA_i is corrupted. The ciphertext on which the attack is performed uses master public keys of authorities AA_i and AA_j, and the attack is performed on the honest attribute authority AA_j.

Complete analysis: First, the tool performs the security analysis. If it finds that the scheme is possibly insecure, then it also performs the master-key and decryption attacks.

5.2 Availability of the ACABELLA tool

The ACABELLA tool is available to be used in three ways: directly in Python, or using one of the two tools that rely on the implementation of ACABELLA in Python.

1. In Python:
   - via the `Attack` class, which independently provides methods for finding master-key and decryption attacks and analyze the security of an ABE scheme.
   - via the `Analysis` class, which prepares batches of attacks and run them based on the description of an ABE scheme. It is also possible to use the ACABELLA JSON format, which describes the properties of an ABE scheme and particular corruption environments where security is analyzed.
2. With `acabella_cmd`: a command line tool for analyzing the security of ABE schemes based on the ACABELLA JSON format, which we explain in more detail below. It provides an easy way to access the different checks and analysis that are implemented in the ACABELLA tool.
3. With `acabella_web`: a web interface built with Flask[8] that provides a similar functionality as `acabella_cmd`. It provides several examples to help the user to understand the different inputs required for analyzing an ABE scheme.

5.3 ACABELLA JSON format

The ACABELLA tool takes as input a JSON description of an ABE scheme. In a JSON file, the user describes the ABE scheme according to the type of analysis. For instance, when analyzing the security of a scheme, we need the following parameters:

- Lists of the global parameter, key and ciphertext encodings (Definition 3).
- A list of the unknown variables by the user/attacker (Definition 10).
- The definition of the target, e.g., $\alpha \cdot s$.
- Optionally, a list of variables obtained via corruption.

For master-key and decryption attacks, we also require a description of the corruption model and the list of variables obtained via corruption and their origin, e.g., if they have been obtained from a CA or from a particular attribute authority AA_i.

An example of a JSON input for the unbounded CP-ABE scheme in [3] is described in Appendix E. In the example, we ask the tool to run the security analysis and the master-key and decryption attacks. In the example, we use the following policy matrix for the decryption attack: $\mathbf{A} = \begin{pmatrix} 1 & 1 \\ 0 & -1 \end{pmatrix}$ which is a matrix representation of an AND gate over two attributes, e.g., $x_1 \wedge x_2$.

5.4 Using the ACABELLA tool

In general, when an ABE practitioner wants to analyze an ABE scheme using the ACABELLA tool, she would follow the following steps:

[8] https://flask.palletsprojects.com/en/2.2.x/. We used Version 2.2.2.

1. Identify the pair encoding scheme associated with the ABE scheme (Definition 3).
2. Describe which variables are unknown to the user (Definition 10).
3. Decide which corruption model to use and which variables could be obtained by corruption.
4. Choose the type of analysis, i.e., security analysis, master-key or decryption attacks, or complete analysis.

Finally, the tool also supports a functionality to translate the ACABELLA JSON format to the ABGW17 format. This can be used to analyze the security of the scheme with the ABGW17 tool [3], which might provide additional insights about the security of the scheme.

6 Examples: security proofs and attacks

To illustrate ACABELLA, we give various interesting examples of proofs and an attack. We first show that the Boneh-Boyen [10] IBE and Rouselakis-Waters [21] CP-ABE schemes can be proven secure in the ACABELLA framework by proving them secure using our properties for trivial and collusion security (Theorems 1 and 2), and the FABEO property (Proposition 4). We then give an attack on the unbounded CP-ABE scheme in the ABGW17 framework. This attack also illustrates that schemes exist that are trivially secure but not collusion secure. Lastly, we give a new CP-ABE scheme that improves on the unbounded CP-ABE in the ABGW17 framework in terms of both the efficiency and security, and show with ACABELLA that it is secure.

6.1 The BB-IBE1 scheme

The pair encoding scheme associated with the IBE scheme by Boneh and Boyen (BB-IBE1) [10] is defined as

- $\mathrm{mpk} = (b_0, b_1)$
- $\mathbf{k}(\alpha, r, \mathrm{mpk}, y) = (\alpha + r(b_0 + yb_1), r)$
- $\mathbf{c}(s, \mathrm{mpk}, x) = (s(b_0 + xb_1), s)$

Because this scheme is a PES-AC17, we can prove trivial and collusion security via the symbolic property. In particular, the tool finds the following substitutions for the selective property:

$$b_0 \ : \ -x, \quad b_1 \ : \ 1, \quad s \ : \ 1, \quad r \ : \ \frac{1}{x-y}, \quad \alpha \ : \ 1,$$

for which it indeed holds that the polynomials $\alpha + r(b_0 + yb_1) = 1 + \frac{1}{x-y}(-x + y \cdot 1) = 1 - 1 = 0$ and $s(b_0 + xb_1) = 1 \cdot (-x + x \cdot 1) = 0$ evaluate to 0. Similarly, for the co-selective property, the tool finds the following substitutions:

$$b_0 \ : \ \begin{pmatrix} 1 \\ 0 \end{pmatrix} \quad b_1 \ : \ \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \quad s \ : \ \begin{pmatrix} 1 & -\frac{1}{x} \end{pmatrix}, \quad r \ : \ -1, \quad \alpha \ : \ \begin{pmatrix} 1 \\ y \end{pmatrix},$$

which also ensure that the polynomials evaluate to 0 and $(0,0)$. Furthermore, because the matrix decomposition has only one entry that is shared among all keys, i.e., αs, it automatically follows that the symbolic property implies a proof of the FABEO property (Proposition 4). Hence, for BB-IBE1, we have the following security guarantees:

- selective security via its original description [10];
- full security via the AC17 framework;
- full security in GGM via the ABGW17 framework;
- full security in GGM via the RW22 framework.

6.2 The RW13 scheme

The pair encoding scheme associated with the CP-ABE scheme by Rouselakis and Waters (RW13) [21] is defined as

- $\mathrm{mpk} = (b, b', b_0, b_1)$
- $\mathbf{k}(\alpha, \mathbf{r}, \mathrm{mpk}, \mathcal{S}) = (\alpha + rb, r, \{rb' + r_{\mathrm{att}}(b_0 + y_{\mathrm{att}} b_1), r_{\mathrm{att}}\}_{\mathrm{att}\in\mathcal{S}})$
- $\mathbf{c}(\mathbf{s} = (s, \{s_j\}_{j\in[n_1]}), \mathrm{mpk}, (\mathbf{A}, \rho)) = (\{\lambda_j + s_j b', s_j(b_0 + \rho(j) b_1)\}_{j\in[n_1]})$, with $\lambda_j = \mathbf{A}_j \cdot (s, v_2, ..., v_{n_2})^\intercal$, where $\mathbf{A}_j$ denotes the j-th row of $\mathbf{A}$

Because this scheme is a PES-AC17, we can prove trivial and collusion security via the symbolic property. In particular, the tool finds substitutions for the selective and co-selective property that are similar to such proofs in existing work, e.g., [2,29]. The tool also finds a proof that the FABEO property holds (see Appendix F.2 for a transcript), with $\mathcal{S} = \{y, z\}$ and $\rho(1) = x$, $\rho(2) = z$, and the vector

$$\begin{gathered}\mathbf{v} = (rv_2, brs, b'rs', b_0rs', b_1rs', b'rs'_2, b_0rs'_2, b_1rs'_2, r'v_2, br's,\\ b'r's', b_0r's', b_1r's', b'r's'_2, b_0r's'_2, b_1r's'_2, r'_2v_2, br'_2s, b'r'_2s',\\ b_0r'_2s', b_1r'_2s', b'r'_2s'_2, b_0r'_2s'_2, b_1r'_2s'_2, \alpha s, b_0r's, b'rs,\\ b_1r's, b_0r'_2s, b_1r'_2s, \alpha s', brs', \alpha s'_2, brs'_2).\end{gathered}$$

We have run the code on this set and access policy, to ensure that we have covered all three cases (see Section 3.7): we have one attribute that occurs in the key and not in the ciphertext, one attribute that occurs in the ciphertext and not the key, and one that occurs in both.

Now, the vector with αs being substituted by 1 and $\alpha s'$, $\alpha s'_2$ by 0:

$$\begin{gathered}(0, -1, 1, 0, 0, 0, 0, 0, 0, 0, 0, -x/(x-y), 1/(x-y), 0, 0, 0, 0, 0, 0,\\ -x/(x-z), 1/(x-z), 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0).\end{gathered}$$

Note that this vector is also associated with the selective symbolic proof. The vector with $\alpha s'$ being substituted by 1 and αs, $\alpha s'_2$ by 0:

$$(0^{30}, 1, -1, 0, 0).$$

The vector with $\alpha s_2'$ being substituted by 1 and αs, $\alpha s'$ by 0:

$$(0^{32}, 1, -1).$$

To verify the proof of the FABEO property, one can compute the product of key and ciphertext encodings, and substitute the monomials with the values in the vectors. For the first vector, we already know that it verifies correctly, because the proof for the selective symbolic property verifies correctly. For the other two vectors, we note that almost all combinations are substituted by 0. The only combinations that are not substituted by 0 is $s'(\alpha + rb)$ for the second vector and $s_2'(\alpha + rb)$ for the third. Because

$$\alpha s' \ : \ 1, \quad rs'b \ : \ -1, \text{ and } \alpha s_2' \ : \ 1, \quad rs_2'b \ : \ -1,$$

we have that $\alpha s' + rs'b \ : \ 1 - 1 = 0$ and $\alpha s_2' + rs_2'b \ : \ 1 - 1 = 0$. Also note that these proofs generalize to predicates of any sizes, because the only entries in the vectors that depend on the attributes are associated with entries in $\mathbf{v}$ that contain variables that are unique to those attributes. Hence, for RW13, we have the same security guarantees as for BB-IBE1.

6.3 The unbounded ABGW17-CP scheme

The pair encoding scheme associated with the unbounded CP-ABE scheme by Ambrona et al. (ABGW17-CP) [3] is defined as

- $\text{mpk} = (b, b_0, b_1, b')$
- $\mathbf{k}(\alpha, r, \text{mpk}, \mathcal{S}) = (\alpha + rb, r, \{\frac{rb'}{b_0 + yb_1}\}_{y \in \mathcal{S}})$
- $\mathbf{c}(\mathbf{s} = (s, \{s_j\}_{j \in [n_1]}), \text{mpk}, (\mathbf{A}, \rho)) = (\{\lambda_j, \lambda_j b + s_j b', s_j(b_0 + x_j b_1)\}_{j \in [n_1]})$, where $\lambda_j = \mathbf{A}_j \cdot (s, v_2, ..., v_{n_2})^\intercal$

The tool outputs that this scheme is trivially secure. However, it is not collusion secure. In particular, we find the following attack. Let $\mathcal{S}_1 = \{y_1\}$ and $\mathcal{S}_2 = \{y_2\}$, and suppose $(\mathbf{A}, \rho)$ is an LSSS matrix for the policy "y_1 AND y_2", i.e.,

$$\mathbf{A} = \begin{pmatrix} 1 & 1 \\ 0 & -1 \end{pmatrix}, \quad \rho(1) = y_1, \rho(2) = y_2.$$

Then, we sample keys and ciphertext as follows:

$$\begin{aligned}
\mathbf{k}_1 = (k_{1,1}, k_{1,2}, k_{1,3}) &= (\alpha + rb, r, \frac{rb'}{b_0 + y_1 b_1}) \\
\mathbf{k}_2 = (k_{2,1}, k_{2,2}, k_{2,3}) &= (\alpha + r'b, r', \frac{r'b'}{b_0 + y_2 b_1}) \\
\mathbf{c}_1 = (c_1, c_2, c_3, c_4, c_5, c_6) &= (s + v_2, (s + v_2)b + s_1 b', s_1(b_0 + y_1 b_1), \\
&\quad - v_2, -v_2 b + s_2 b', s_2(b_0 + y_2 b_1)).
\end{aligned}$$

The attack is

$$\begin{aligned}
&k_{1,1}c_1 - k_{1,2}c_2 + k_{1,3}c_3 + k_{2,1}c_4 - k_{2,2}c_5 + k_{2,3}c_6 \\
&= (\alpha + rb)(s + v_2) - r((s + v_2)b + s_1b') + \frac{rb'}{b_0 + y_1b_1}(s_1(b_0 + y_1b_1)) \\
&+(\alpha + r'b)(-v_2) - r'(-v_2b + s_2b') + \frac{r'b'}{b_0 + y_2b_1}(s_2(b_0 + y_2b_1)) \\
&= \alpha s + \alpha v_2 + rsb + rv_2b - rsb - rv_2b - rs_1b' + rs_1b' \\
&-\alpha v_2 - r'v_2b + r'v_2b - r's_1b' + r's_2b' = \alpha s.
\end{aligned}$$

6.4 A new unbounded CP-ABE scheme

We propose a new unbounded CP-ABE scheme that is both more efficient than ABGW17-CP, and that is provably collusion secure. It is defined as follows.

- $\mathrm{mpk} = (b, b_0, b_1)$
- $\mathbf{k}(\alpha, r, \mathrm{mpk}, \mathcal{S}) = ((\alpha + r)/b, \{\frac{r}{b_0 + yb_1}\}_{y \in \mathcal{S}})$
- $\mathbf{c}(\mathbf{s}, \mathrm{mpk}, (\mathbf{A}, \rho)) = (sb, \{\lambda_j(b_0 + x_jb_1)\}_{j \in [n_1]})$, where $\mathbf{s} = (s, \{s_j\}_{j \in [n_1]})$ and $\lambda_j = \mathbf{A}_j \cdot (s, v_2, ..., v_{n_2})^\intercal$.

A transcript output by the ACABELLA tool that proves trivial and collusion security of this scheme can be found in Appendix F.3, for $\mathcal{S} = \{x_1, y\}$ and $\mathbb{A} = (\mathbf{A}, \rho)$ with $\mathbf{A} = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}$, $\rho(1) = x_1$, $\rho(2) = x_2$. We show that the scheme is collusion secure with Proposition 6. We first have to verify the proof for trivial security, and then that the other three requirements hold.

Trivial security is already verified automatically but can be verified manually as argued in Section 3.6, i.e., by computing the products of key and ciphertext encodings and substituting the monomials with the associated values in $\mathbf{w}_{\mathcal{S},\mathbb{A}}$. Most combinations of key and ciphertext encodings yield non-critical combinations (that can be argued will trivially evaluate to 0, see Section 3.6). Hence, we only have to consider the products of $(\alpha + r)/b$ and sb, and $\frac{r}{b_0 + y'b_1}$ and $\lambda_j(b_0 + x_jb_1)$ for four combinations of y' and x_j. Because the proof is "normalized" by a factor $\lambda = b(b_0 + x_1b_1)(b_0 + yb_1)$, we also need to normalize the computations to find the appropriate substitutions. We show how the verification works for one product and leave the rest up to the reader:

$$\begin{aligned}
\lambda \cdot (\alpha + r)/b \cdot sb = \; & \underbrace{\alpha b b_0^2 s}_{:\ (1+x_1)\frac{-A_{12}x_2 + A_{12}y}{A_{11}}} + \underbrace{\alpha b b_0 b_1 s}_{:\ 0}(x_1 + y) \\
& + \underbrace{\alpha b b_1^2 s}_{:\ 0} x_1 y + \underbrace{b b_0 b_1 r s}_{:\ \frac{-(1+x_2)A_{11}A_{22}x_1 + (1+x_1)A_{12}A_{21}x_2}{A_{11}A_{21}(x_1 - x_2)}}(x_1 + y) \\
& + \underbrace{b b_0^2 r s}_{:\ \frac{(1+x_2)A_{11}A_{22}x_1^2 - (1+x_1)A_{12}A_{21}x_2^2}{A_{11}A_{21}(x_1 - x_2)}} + \underbrace{b b_1^2 r s}_{:\ \frac{(1+x_2)A_{11}A_{22} - (1+x_1)A_{21}A_{12}}{A_{11}A_{21}(x_1 - x_2)}} x_1 y,
\end{aligned}$$

which is substituted by

$$\begin{aligned}
&\frac{-(1+x_1)(x_1-x_2)A_{12}A_{21}x_2+(1+x_1)(x_1-x_2)A_{12}A_{21}y}{A_{11}A_{21}(x_1-x_2)}\\
+&\frac{-(x_1+y)(1+x_2)A_{11}A_{22}x_1+(x_1+y)(1+x_1)A_{12}A_{21}x_2}{A_{11}A_{21}(x_1-x_2)}\\
+&\frac{(1+x_2)A_{11}A_{22}x_1^2-(1+x_1)A_{12}A_{21}x_2^2}{A_{11}A_{21}(x_1-x_2)}\\
+&\frac{x_1y(1+x_2)A_{11}A_{22}-x_1y(1+x_1)A_{21}A_{12}}{A_{11}A_{21}(x_1-x_2)}\\
=&\underbrace{(-x_1(x_1+y)+x_1^2+x_1y)}_{=0}\frac{A_{11}A_{22}(1+x_2)}{A_{11}A_{21}(x_1-x_2)}\\
+&\underbrace{\left(x_2^2-x_1x_2+yx_1-yx_2+x_1x_2+x_2y-x_2^2-x_1y\right)}_{=0}\frac{A_{12}(1+x_1)}{A_{11}(x_1-x_2)}
\end{aligned}$$

We now show that the other three requirements hold. The last one can easily be seen from the transcript, as all the shared entries that are not associated with the target vector are substituted by 0. For the second and third requirements, we first translate the substitution vector to the "regular" setting in which the encodings are not normalized. We do this by dividing each entry of the vector $\mathbf{v}$ by $\lambda = b(b_0 + x_1b_1)(b_0 + yb_1)$. Now, to argue that the second requirement holds, we simply use that the part of the matrix and vector that is associated αs is translated from $\lambda\alpha s$ and then back to αs, so it is always equal to αs regardless of the used predicates. Something similar follows for the part that is associated with shared entries that are not associated with the target αs.

The third requirement holds by showing that we can "normalize" the substitution vector in the only entry associated with the shared entries. This entry corresponds to $\alpha b b_0^2 s$, and can be rewritten as

$$\frac{A_{12}(y-x_2)(1+x_1)}{A_{11}} = \frac{A_{12}(y-x_2)(1+x_1)}{-w_2A_{12}} = \frac{(x_2-y)(1+x_1)}{w_2},$$

because x_1 occurs in both the key and ciphertext, and therefore, there exists a vector $(1, w_2)$ such that $A_{11} + w_2A_{12} = 0$, i.e., $A_{11} = -w_2A_{12}$ (Definition 2). By dividing the vector by $(x_2 - y)\frac{1+x_1}{w_2}$, we obtain a vector that is 1 in the entry associated with $\alpha b b_0^2 s$, and 0 in the rest of the shared entries.

Lastly, to argue that this proof also works for predicates of different sizes and structures, we note that the same randomness r is used for all key entries, and the same randomness s is used for all ciphertext entries. We can therefore not argue, similarly as for the RW13 scheme, that this proof generalizes, because all the entries of the substitution vector that depend on the predicates use the same randomness. However, the fact that the scheme is secure for multiple attributes while still using the same randomness illustrates that it will be secure for any number of attributes. That is, for the polynomial structure that we use to embed

the attribute, i.e., $b_0 + xb_1$, one randomized instance of it usually provides only enough randomness for one attribute (see [29]). Because we have used it for multiple attributes, it should be secure for any number of attributes. Hence, the scheme is trivially and collusion secure.

6.5 More examples

The web tool contains many more examples, e.g., some attacks on the schemes in [27] and several popular IBE and ABE schemes. These examples can be used to learn how the tool works, or to achieve better security guarantees for existing schemes. Notably, we prove security in the ABGW17 framework for the first CP-ABE scheme [9] and the IBE and KP-ABE schemes in [3], and we prove that the FABEO property holds for the PESs associated with [33] and [21].

7 Conclusions

In this work, we have introduced ACABELLA, which simplifies generating and verifying security proofs for pairing-based ABE schemes. It consists of a framework for security proofs that are easy to verify manually, and an automated tool that efficiently generates these security proofs. Using ACABELLA, it is now possible to automatically generate security proofs that imply security results in the AC17, ABGW17 and FABEO frameworks (see Figure 1 for an overview). Importantly, these security proofs are manually verifiable, meaning that we do not have to fully trust the tool to have verified the scheme correctly. In contrast, the ABGW17 tool needs to be fully trusted to have performed the security analysis correctly, and can only prove security in the single-key setting. An additional advantage of our framework is that the security proofs are also much shorter than we are typically used to see in literature, e.g., security proofs using the dual-system encryption paradigm [32]. Lastly, to improve the accessibility and usability of automated verification, we have provided the ACABELLA tool as a web interface. With its help, experts can simplify their proving process by verifying or refuting the security of their schemes and practitioners can get an assurance that the ABE scheme of their choice is secure.

In future work, ACABELLA can be extended with several interesting functionalities. First, although our tool already provides some functionality for multi-authority schemes, it is valuable to extend the functionality for all pairing-based multi-authority schemes, and to flexibly cover all types of corruption models. For example, the recently published compiler by Venema [26] may be used for this. Furthermore, ACABELLA does not currently support the generation of proofs in the multi-authority setting, because none of the frameworks that it builds on does this. Another aspect of the tool that can be improved is the expression of the predicates, e.g., access policies and sets of attributes. By using only linear-algebra methods to generate the proofs and attacks, we cannot manipulate the values of the access policies and sets of attributes. Possibly, this can be done by combining our linear algebra approach with a symbolic solver like ABGW17. Then, it will also become easier to find complex attacks.

Acknowledgments. The authors would like to thank the anonymous reviewers for their insightful comments and suggestions. They would also like to thank Cas Cremers and Jan Schoone for helpful discussions.

References

1. Agrawal, S., Chase, M.: A study of pair encodings: Predicate encryption in prime order groups. In: Kushilevitz, E., Malkin, T. (eds.) TCC. LNCS, vol. 9563, pp. 259–288. Springer (2016)
2. Agrawal, S., Chase, M.: Simplifying design and analysis of complex predicate encryption schemes. In: Coron, J., Nielsen, J.B. (eds.) EUROCRYPT. LNCS, vol. 10210, pp. 627–656. Springer (2017)
3. Ambrona, M., Barthe, G., Gay, R., Wee, H.: Attribute-based encryption in the generic group model: Automated proofs and new constructions. In: Thuraisingham, B.M., Evans, D., Malkin, T., Xu, D. (eds.) CCS. pp. 647–664. ACM (2017)
4. Ambrona, M., Barthe, G., Schmidt, B.: Generic transformations of predicate encodings: Constructions and applications. In: Katz, J., Shacham, H. (eds.) CRYPTO. LNCS, vol. 10401, pp. 36–66. Springer (2017)
5. Attrapadung, N.: Dual system encryption via doubly selective security: Framework, fully secure functional encryption for regular languages, and more. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT. LNCS, vol. 8441, pp. 557–577. Springer (2014)
6. Attrapadung, N.: Dual system encryption framework in prime-order groups via computational pair encodings. In: Cheon, J.H., Takagi, T. (eds.) ASIACRYPT. LNCS, vol. 10032, pp. 591–623. Springer (2016)
7. Attrapadung, N.: Unbounded dynamic predicate compositions in attribute-based encryption. In: Ishai, Y., Rijmen, V. (eds.) EUROCRYPT. LNCS, vol. 11476, pp. 34–67. Springer (2019)
8. Beimel, A.: Secure Schemes for Secret Sharing and Key Distribution. Phd thesis, Ben Gurion University (1996)
9. Bethencourt, J., Sahai, A., Waters, B.: Ciphertext-policy attribute-based encryption. In: S&P. pp. 321–334. IEEE (2007)
10. Boneh, D., Boyen, X.: Efficient selective-ID secure identity-based encryption without random oracles. In: Cachin, C., Camenisch, J. (eds.) EUROCRYPT. LNCS, vol. 3027, pp. 223–238. Springer (2004)
11. Chase, M.: Multi-authority attribute-based encryption. In: Vadhan, S.P. (ed.) TCC. LNCS, vol. 4392, pp. 515–534. Springer (2007)
12. ETSI: ETSI TS 103 458 (V1.1.1). Technical specification, European Telecommunications Standards Institute (ETSI) (2018)
13. Goyal, V., Pandey, O., Sahai, A., Waters, B.: Attribute-based encryption for fine-grained access control of encrypted data. In: Juels, A., Wright, R.N., di Vimercati, S.D.C. (eds.) CCS. ACM (2006)
14. Goyal, V., Pandey, O., Sahai, A., Waters, B.: Attribute-based encryption for fine-grained access control of encrypted data. Cryptology ePrint Archive, Report 2006/309 (2006)
15. Herranz, J.: Attacking pairing-free attribute-based encryption schemes. IEEE Access **8**, 222226–222232 (2020)
16. Kamara, S., Lauter, K.E.: Cryptographic cloud storage. In: Sion, R., Curtmola, R., Dietrich, S., Kiayias, A., Miret, J.M., Sako, K., Sebé, F. (eds.) RLCPS. LNCS, vol. 6054, pp. 136–149. Springer (2010)

17. Ladd, W., Verma, T., Venema, M., Faz-Hérnandez, A., McMillion, B., Wildani, A., Sullivan, N.: Portunus: Re-imagining access control in distributed systems. In: USENIX ATC. pp. 35–52 (2023)
18. Lewko, A., Waters, B.: Decentralizing attribute-based encryption. In: EUROCRYPT. pp. 568–588. Springer (2011)
19. Meurer, A., Smith, C.P., Paprocki, M., Certík, O., Kirpichev, S.B., Rocklin, M., Kumar, A., Ivanov, S., Moore, J.K., Singh, S., Rathnayake, T., Vig, S., Granger, B.E., Muller, R.P., Bonazzi, F., Gupta, H., Vats, S., Johansson, F., Pedregosa, F., Curry, M.J., Terrel, A.R., Roucka, S., Saboo, A., Fernando, I., Kulal, S., Cimrman, R., Scopatz, A.M.: Sympy: symbolic computing in python. PeerJ Comput. Sci. **3**, e103 (2017)
20. Riepel, D., Wee, H.: FABEO: fast attribute-based encryption with optimal security. In: Yin, H., Stavrou, A., Cremers, C., Shi, E. (eds.) CCS. pp. 2491–2504. ACM (2022)
21. Rouselakis, Y., Waters, B.: Practical constructions and new proof methods for large universe attribute-based encryption. In: Sadeghi, A., Gligor, V.D., Yung, M. (eds.) CCS. pp. 463–474. ACM (2013)
22. Rouselakis, Y., Waters, B.: Efficient statically-secure large-universe multi-authority attribute-based encryption. In: Böhme, R., Okamoto, T. (eds.) FC. LNCS, vol. 8975, pp. 315–332. Springer (2015)
23. Sahai, A., Waters, B.: Fuzzy identity-based encryption. In: Cramer, R. (ed.) EUROCRYPT. LNCS, vol. 3494, pp. 457–473. Springer (2005)
24. Santos, N., Rodrigues, R., Gummadi, K.P., Saroiu, S.: Policy-sealed data: A new abstraction for building trusted cloud services. In: USENIX Security Symposium. pp. 175–188. USENIX Association (2012)
25. Shoup, V.: Lower bounds for discrete logarithms and related problems. In: Fumy, W. (ed.) EUROCRYPT. LNCS, vol. 1233, pp. 256–266. Springer (1997)
26. Venema, M.: A practical compiler for attribute-based encryption: New decentralized constructions and more. In: Rosulek, M. (ed.) CT-RSA. LNCS, vol. 13871, pp. 132–159. Springer (2023)
27. Venema, M., Alpár, G.: A bunch of broken schemes: A simple yet powerful linear approach to analyzing security of attribute-based encryption. In: Paterson, K.G. (ed.) CT-RSA. LNCS, vol. 12704, pp. 100–125. Springer (2021)
28. Venema, M., Alpár, G.: TinyABE: Unrestricted ciphertext-policy attribute-based encryption for embedded devices and low-quality networks. In: Batina, L., Daemen, J. (eds.) AFRICACRYPT. LNCS, vol. 13503, pp. 103–129. Springer (2022)
29. Venema, M., Alpár, G.: GLUE: generalizing unbounded attribute-based encryption for flexible efficiency trade-offs. In: Boldyreva, A., Kolesnikov, V. (eds.) PKC. LNCS, vol. 13940, pp. 652–682. Springer (2023)
30. Venema, M., Alpár, G., Hoepman, J.: Systematizing core properties of pairing-based attribute-based encryption to uncover remaining challenges in enforcing access control in practice. Des. Codes Cryptogr. **91**(1), 165–220 (2023)
31. Wang, F., Mickens, J., Zeldovich, N., Vaikuntanathan, V.: Sieve: Cryptographically enforced access control for user data in untrusted clouds. In: Argyraki, K.J., Isaacs, R. (eds.) NSDI. pp. 611–626. USENIX Association (2016)
32. Waters, B.: Dual system encryption: Realizing fully secure IBE and HIBE under simple assumptions. In: Halevi, S. (ed.) CRYPTO. LNCS, vol. 5677, pp. 619–636. Springer (2009)
33. Waters, B.: Ciphertext-policy attribute-based encryption - an expressive, efficient, and provably secure realization. In: Catalano, D., Fazio, N., Gennaro, R., Nicolosi, A. (eds.) PKC. LNCS, vol. 6571, pp. 53–70. Springer (2011)

34. Wee, H.: Dual system encryption via predicate encodings. In: Lindell, Y. (ed.) TCC. LNCS, vol. 8349, pp. 616–637. Springer (2014)
35. Wee, H.: Déjà Q: encore! un petit IBE. In: Kushilevitz, E., Malkin, T. (eds.) TCC. LNCS, vol. 9563, pp. 237–258. Springer (2016)

A Further definitions

A.1 Security against chosen-plaintext attacks

Definition 17 (Full security against chosen-plaintext attacks (CPA) [2]). *We define the security game* IND-CPA(λ, par) *between challenger and attacker as follows:*

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

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

In the selective security model, the attacker commits to the predicate $x^* \in \mathcal{X}_\kappa$ before the Setup phase. In the co-selective security model, the attacker commits to all $y \in \mathcal{Y}_\kappa$ before the Setup phase. In the static security model, the attacker commits to $x^* \in \mathcal{X}_\kappa$ and all $y \in \mathcal{Y}_\kappa$ before the Setup phase.

A.2 The attack models

Venema and Alpár define the following attack models [27].

Definition 18 (Master-key attacks (MKA) [27]). *We define the game between challenger and attacker as follows. First, the initialization, setup and first key query phases are run as in Definition 17. Then:*

- ***Decision phase:*** *The attacker outputs* MK'.

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

Definition 19 (Decryption attacks (DA) [27]). *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 17. Then:*

– ***Decision phase:*** *The attacker outputs plaintext M'.*

The attacker wins the game if $M' = M$.

B Remainder of lemmas and proofs

B.1 Proof of Proposition 1

Proof. First, it follows trivially that finding an attack in the matrix notation also yields an attack in the polynomial notation, because any coefficients found in the matrix notation are also valid coefficients for the polynomial notation: $\mathbf{e} \cdot \mathbf{p_{enc}}^\intercal = \mathbf{e} \cdot (\mathbf{M} \cdot \mathbf{v}^\intercal) = (\mathbf{e} \cdot \mathbf{M}) \cdot \mathbf{v}^\intercal = \mathtt{tv} \cdot \mathbf{v}^\intercal = \mathrm{tv}$. However, the converse follows less trivially: if there exists a linear combination for the polynomial notation, then there exists a linear combination for the matrix notation. This follows from the fact that all entries of $\mathbf{v}$ are linearly independent. Suppose $\mathbf{p_{enc}} = \mathbf{M} \cdot \mathbf{v}^\intercal$ is a matrix decomposition as in Definition 11, and let $\mathtt{tv} \cdot \mathbf{v}^\intercal = \alpha s$ be the target vector. If $\mathbf{e} \in \mathbb{Z}_p(\mathsf{coef})^{|\mathbf{p_{enc}}|}$ exists such that $\mathbf{e} \cdot \mathbf{p_{enc}}^\intercal = \alpha s$, then we have $\mathbf{e}(\mathbf{M} \cdot \mathbf{v}^\intercal) = \mathtt{tv} \cdot \mathbf{v}^\intercal$. Now, assume that $\mathbf{e} \cdot \mathbf{M} \neq \mathtt{tv}$, and consider $\mathbf{w} = \mathbf{e} \cdot \mathbf{M} - \mathtt{tv} \neq \mathbf{0}$. Then, we have

$$\begin{aligned} \mathbf{w} \cdot \mathbf{v}^\intercal &= (\mathbf{e} \cdot \mathbf{M} - \mathtt{tv}) \cdot \mathbf{v}^\intercal \\ &= (\mathbf{e} \cdot \mathbf{M}) \cdot \mathbf{v}^\intercal - \mathtt{tv} \cdot \mathbf{v}^\intercal = \mathbf{0}. \end{aligned}$$

From this, it follows that the entries of $\mathbf{v}$ are not linearly independent, which contradicts the definition of the matrix decomposition. Thus, it must hold that $\mathbf{e} \cdot \mathbf{M} = \mathtt{tv}$.

B.2 Proof of Theorem 1

Proof. To prove that the scheme is trivially secure, we must prove that for all $x \in \mathcal{X}, y \in \mathcal{Y}$ with $P(x, y) = 0$, it holds that $\mathbf{k}_y \cdot \mathbf{E}_i \cdot \mathbf{c}_x^\intercal \neq \alpha s$ for all $\mathbf{E}_i$. This is equivalent to proving that $\mathtt{tv}_{x,y}$ is not in the span of the rows of $\mathbf{M}_{x,y}$ (Proposition 1). Because, for all $x \in \mathcal{X}, y \in \mathcal{Y}$, there exists some $\mathbf{w}_{x,y}^\intercal$ such that $\mathbf{M}_{x,y} \cdot \mathbf{w}_{x,y}^\intercal = \mathbf{0}^\intercal$ and $\mathtt{tv}_{x,y} \cdot \mathbf{w}_{x,y}^\intercal \neq 0$, we know by Proposition 2 that $\mathtt{tv}_{x,y}$ is not in the span of $\mathbf{M}_{x,y}$.

B.3 Proof of Lemma 1

Proof. It follows trivially from the assumption that $\mathbf{p_{enc}}'_{x,y}$ and $\mathbf{p_{enc}}''_{x,y}$ have no monomials in common that $\mathbf{M}_{x,y}$ is of the form $\begin{pmatrix} \mathbf{M}'_{x,y} & \mathbf{0} \\ \mathbf{0} & \mathbf{M}''_{x,y} \end{pmatrix}$. Similarly, because α does not occur in $\mathbf{v}''_{x,y}$, the target vector is of the form $\mathtt{tv}_{x,y} = (\mathtt{tv}'_{x,y}, \mathbf{0})$. Then, from the form of $\mathbf{M}_{x,y}$, it follows that $\mathbf{M}_{x,y} \cdot \mathbf{w}_{x,y}^\intercal = \mathbf{M}'_{x,y} \cdot (\mathbf{w}'_{x,y})^\intercal + \mathbf{M}''_{x,y} \cdot (\mathbf{w}''_{x,y})^\intercal = \mathbf{0} + \mathbf{0} = \mathbf{0}$, and $\mathtt{tv}_{x,y} \cdot \mathbf{w}_{x,y}^\intercal = \mathtt{tv}'_{x,y} \cdot (\mathbf{w}'_{x,y})^\intercal + \mathtt{tv}''_{x,y} \cdot (\mathbf{w}''_{x,y})^\intercal = \mathtt{tv}'_{x,y} \cdot (\mathbf{w}'_{x,y})^\intercal + 0 \neq 0$.

B.4 Proof of Proposition 3

Proof. To prove that $\mathbf{M}_{x,\{y_1,y_2,\ldots,y_n\}} \cdot \mathbf{v}^{\intercal}_{x,\{y_1,y_2,\ldots,y_n\}}$ is a matrix decomposition, we first need to show that $\mathbf{M}_{x,\{y_1,y_2,\ldots,y_n\}} \cdot \mathbf{v}^{\intercal}_{x,\{y_1,y_2,\ldots,y_n\}} = \mathbf{p_{enc}}x,\{y_1,y_2,\ldots,y_n\}$. This follows from the fact that

$$\mathbf{M}_{x,\{y_1,y_2,\ldots,y_n\}} \cdot \mathbf{v}^{\intercal}_{x,\{y_1,y_2,\ldots,y_n\}} = \begin{pmatrix} \sum_{j\in\{0,1,2\}} \mathbf{M}_{x,y_1,j} \cdot \mathbf{v}^{\intercal}_{x,y_1,j} \\ \sum_{j\in\{0,1,2\}} \mathbf{M}_{x,y_2,j} \cdot \mathbf{v}^{\intercal}_{x,y_2,j} \\ \vdots \\ \sum_{j\in\{0,1,2\}} \mathbf{M}_{x,y_n,j} \cdot \mathbf{v}^{\intercal}_{x,y_n,j} \end{pmatrix}$$

$$= \left(\mathbf{p_{enc}}x,y_1 \ \mathbf{p_{enc}}x,y_2 \cdots \mathbf{p_{enc}}x,y_n\right)^{\intercal} = \mathbf{p_{enc}}^{\intercal}_{x,\{y_1,\ldots,y_n\}}.$$

Furthermore, it follows trivially that the associated target vector constructed as $\mathbf{tv}_{x,\{y_1,\ldots,y_n\}} = (\mathbf{tv}_{x,y_1,0}, \mathbf{0})$ is correct by verifying that

$$\begin{aligned} &\mathbf{tv}_{x,\{y_1,\ldots,y_n\}} \cdot \mathbf{v}^{\intercal}_{x,\{y_1,\ldots,y_n\}} \\ &= (\mathbf{tv}_{x,y_1,0}, \mathbf{0}) \cdot (\mathbf{v}_{x,y_1,0}, \mathbf{v}_{x,y_1,1}, \mathbf{v}_{x,y_1,2}, \mathbf{v}_{x,y_2,2}, \ldots, \mathbf{v}_{x,y_n,2})^{\intercal} \\ &= \mathbf{tv}_{x,y_1,0} \cdot \mathbf{v}^{\intercal}_{x,y_1,0} = \alpha s. \end{aligned}$$

Then, we need to show that the entries of $\mathbf{v}_{x,\{y_1,y_2,\ldots,y_n\}}$ are linearly independent monomials. First, note that all entries are monomials, because the entries of all $\mathbf{v}_{x,y_i}$ are monomials. Second, they are linearly independent. We prove this by contradiction. Suppose that one of the entries can be written as a linear combination of the others. Then, we can also find a linear combination of the monomials in the entries that is equal to 0. For the terms associated with $\mathbf{v}_{x,y_i,2}$, we know that any non-trivial linear combination yields at least one term that is a multiple of some user-specific random variable $r_{i,k}$. For all $j \neq i$ and k', it holds that all other user-specific random variables $r_{j,k'}$ are such that $r_{i,k} \neq r_{j,k'}$. Because each non-trivial linear combination of the monomials in $\mathbf{v}_{x,y_j,2}$ also have at least one term that is a multiple of a user-specific random variable $r_{j,k'}$, the terms associated with $\mathbf{v}_{x,y_j,2}$ cannot cancel out the terms associated with $\mathbf{v}_{x,y_i,2}$. Thus, each of these terms need to be canceled by a linear combination of $(\mathbf{v}_{x,y_1,0}, \mathbf{v}_{x,y_1,1})$, which contradicts the assumption that $\mathbf{v}_{x,y}$ consists of linearly independent monomials.

B.5 Proof of Theorem 2

Proof. To prove that the PES is collusion secure, we need to show that $\sum_i \mathbf{k}_{y_i} \cdot \mathbf{E}_i \cdot \mathbf{c}^{\intercal}_x \neq \alpha s$. By Proposition 1, this is equivalent to proving that $\mathbf{tv}_{x,\{y_1,\ldots,y_n\}}$ is not in the span of $\mathbf{M}_{x,\{y_1,\ldots,y_n\}}$, where $\mathbf{tv}_{x,\{y_1,\ldots,y_n\}}$ and $\mathbf{M}_{x,\{y_1,\ldots,y_n\}}$ are constructed as in Proposition 3. In turn, this is equivalent to proving that some vector $\mathbf{w}_{x,\{y_1,\ldots,y_n\}}$ exists such that $\mathbf{M}_{x,\{y_1,\ldots,y_n\}} \cdot \mathbf{w}^{\intercal}_{x,\{y_1,\ldots,y_n\}} = \mathbf{0}^{\intercal}$ and $\mathbf{tv}_{x,\{y_1,\ldots,y_n\}} \cdot \mathbf{w}^{\intercal}_{x,\{y_1,\ldots,y_n\}} \neq 0$. We claim that this is the case for

$$\mathbf{w}_{x,\{y_1,\ldots,y_n\}} = (\mathbf{w}_{x,y_1,0}, \mathbf{w}_{x,y_1,1}, \mathbf{w}_{x,y_1,2}, \mathbf{w}_{x,y_2,2}, \ldots, \mathbf{w}_{x,y_n,2}).$$

This follows from verifying that

$$\mathbf{M}_{x,\{y_1,\ldots,y_n\}} \cdot \mathbf{w}^{\intercal}_{x,\{y_1,\ldots,y_n\}} = \begin{pmatrix} \sum_{j\in\{0,1,2\}} \mathbf{M}_{x,y_1,j} \cdot \mathbf{w}^{\intercal}_{x,y_1,j} \\ \sum_{j\in\{0,1,2\}} \mathbf{M}_{x,y_2,j} \cdot \mathbf{w}^{\intercal}_{x,y_2,j} \\ \vdots \\ \sum_{j\in\{0,1,2\}} \mathbf{M}_{x,y_n,j} \cdot \mathbf{w}^{\intercal}_{x,y_n,j} \end{pmatrix}$$
$$= \begin{pmatrix} \mathbf{M}_{x,y_1} \cdot \mathbf{w}^{\intercal}_{x,y_1} \\ \mathbf{M}_{x,y_2} \cdot \mathbf{w}^{\intercal}_{x,y_2} \\ \vdots \\ \mathbf{M}_{x,y_n} \cdot \mathbf{w}^{\intercal}_{x,y_n} \end{pmatrix} = \mathbf{0},$$

and that

$$\mathtt{tv}_{x,\{y_1,\ldots,y_n\}} \cdot \mathbf{w}^{\intercal}_{x,\{y_1,\ldots,y_n\}} = \mathtt{tv}_{x,y_1} \cdot \mathbf{w}^{\intercal}_{x,y_1} \neq 0,$$

which follows from the fact that $\mathtt{tv}_{x,\{y_1,\ldots,y_n\}}$ is only non-zero in the entries corresponding to the first entries of $\mathbf{w}^{\intercal}_{x,\{y_1,\ldots,y_n\}}$, i.e., $\mathbf{w}_{x,y_1,0}$.

B.6 Lemmas for the notation of the proofs

Lemma 2. *Let* $\mathbf{p_{enc}}$ *be a vector of encodings with matrix decomposition* $\mathbf{p_{enc}} = \mathbf{M} \cdot \mathbf{v}^{\intercal}$*, and let* $\mathbf{w}$ *be a vector of the same length as* $\mathbf{v}$*. Then, computing* $\mathbf{M} \cdot \mathbf{w}^{\intercal}$ *is equal to considering each entry of* $\mathbf{p_{enc}}$*, and evaluate the polynomial for which the monomial in each term is substituted as* $(\mathbf{v})_i \ : \ (\mathbf{w})_i$*.*

Proof. Let p_i be the i-th entry of $\mathbf{p_{enc}}$. Then, the decomposition of p_i is the i-th row of $\mathbf{M}$. Note that p_i is a polynomial, and can in particular be written as a linear combination of entries in $\mathbf{v}$, i.e., $p_i = \sum_j M_{i,j}(\mathbf{v})_j$, where $M_{i,j}$ is the j-th entry in the i-th row. Now, substituting each $(\mathbf{v})_j$ with $(\mathbf{w})_j$ yields $\sum_j M_{i,j}(\mathbf{w})_j$. This is equal to computing $(\mathbf{M} \cdot \mathbf{w}^{\intercal})_i = \sum_j M_{i,j}(\mathbf{w})_j$.

Lemma 3. *Let* $\mathbf{p_{enc}}$ *be a vector of encodings with matrix decomposition* $\mathbf{p_{enc}} = \mathbf{M} \cdot \mathbf{v}^{\intercal}$*. Let* $\mathbf{p_{enc}}' = ((\mathbf{p_{enc}})_1, ..., (\mathbf{p_{enc}})_{i-1}, (\mathbf{p_{enc}})_{i+1}, ...)$ *be the vector of encodings* $\mathbf{p_{enc}}$ *without* $(\mathbf{p_{enc}})_i$*, and let* $\mathbf{M}'$ *be the matrix* $\mathbf{M}$ *with the* i*-th row removed. Suppose that* $(\mathbf{p_{enc}})_i$ *is such that some* j *exists with* $(\mathbf{M})_{i,j} \neq 0$ *and* $(\mathbf{M})_{i',j} = 0$ *for all* $i' \neq i$*. Then, if* $\mathbf{w}'$ *is a vector such that* $\mathbf{M}' \cdot (\mathbf{w}')^{\intercal} = \mathbf{0}^{\intercal}$*, then the vector* $\mathbf{w} = ((\mathbf{w}')_1, ..., (\mathbf{w}')_{i-1}, -\frac{1}{(\mathbf{M})_{i,j}} \sum_{j'\neq j} (\mathbf{M})_{i,j'}(\mathbf{w}')_{j'}, (\mathbf{w}')_i, (\mathbf{w}')_{i+1}, ...)$ *is such that* $\mathbf{M} \cdot \mathbf{w}^{\intercal} = \mathbf{0}^{\intercal}$*.*

Proof. For all $i' \neq i$, we have that

$$(\mathbf{M} \cdot (\mathbf{w}')^{\intercal})_{i'} = (\mathbf{M}' \cdot \mathbf{w}^{\intercal})_{i'} - \frac{(\mathbf{M})_{i',j}}{(\mathbf{M})_{i,j}} \sum_{j'\neq j} (\mathbf{M})_{i,j'}(\mathbf{w}')_{j'} = 0,$$

because $(\mathbf{M}' \cdot \mathbf{w}^{\intercal})_{i'} = 0$ and $M_{i',j} = 0$. For i, we have that

$$(\mathbf{M} \cdot (\mathbf{w}')^{\intercal})_{i} = (\mathbf{M}' \cdot \mathbf{w}^{\intercal})_{i} - \frac{(\mathbf{M})_{i,j}}{(\mathbf{M})_{i,j}} \sum_{j'\neq j} (\mathbf{M})_{i,j'}(\mathbf{w}')_{j'} = 0,$$

because $(\mathbf{M}' \cdot \mathbf{w}^{\intercal})_{i} = \sum_{j'\neq j} (\mathbf{M})_{i,j'}(\mathbf{w}')_{j'}$.

B.7 Proof of Theorem 3

Proof. We first prove the implication that PES-AC17 is trivially secure if the symbolic property holds. This actually already follows from the fact that the selective symbolic property holds. We prove this again by showing that a vector $\mathbf{w}_{x,y}$ exists such that $\mathbf{M}_{x,y} \cdot \mathbf{w}_{x,y}^{\intercal} = \mathbf{0}^{\intercal}$ and $\mathtt{tv}_{x,y} \cdot \mathbf{w}_{x,y}^{\intercal} \neq 0$, where $\mathbf{M}_{x,y} \cdot \mathbf{v}_{x,y}^{\intercal}$ denotes a matrix decomposition for $\mathbf{p_{enc}}'_{x,y}$ and $\mathtt{tv}_{x,y}$ is the associated target vector. First, we obtain vectors $\mathbf{a}, \mathbf{r}_i, \hat{\mathbf{r}}_{i'}, \mathbf{s}_j, \hat{\mathbf{s}}_{j'}$ and matrices $\mathbf{B}_k$ as in Definition 8 by running EncB(x), EncR(x, y) and EncS(x). Without loss of generality, we assume that $\mathbf{a}_1 \neq 0$. (Note that, by definition, at least one of the entries should be non-zero.) Then, we set the entries of $\mathbf{w}_{x,y}$ to $\mathbf{v}_{x,y}$, but each variable is substituted as in Definition 8. Because $\mathtt{tv}_{x,y}$ is only 1 in the entry corresponding to the entry in $\mathbf{v}_{x,y}$ with αs, and the corresponding entry in $\mathbf{w}_{x,y}$ is determined as $\mathbf{s}_0 \cdot \mathbf{a}^{\intercal}$, for which, by the symbolic property, $\mathbf{s}_0 \cdot \mathbf{a}^{\intercal} \neq 0$ holds, we have that $\mathtt{tv}_{x,y} \cdot \mathbf{w}_{x,y}^{\intercal} \neq 0$. Furthermore, computing $\mathbf{M}_{x,y} \cdot \mathbf{w}_{x,y}^{\intercal}$ is equivalent to substituting all encodings in $\mathbf{p_{enc}}'_{x,y}$ with the vectors and matrices. Here, all entries are of the form $s_j k_{j'}$ and $r_j c_{j'}$, which evaluate to 0 because $k_{j'}$ and $c_{j'}$ evaluate to 0. Thus, $\mathbf{M}_{x,y} \cdot \mathbf{w}_{x,y}^{\intercal} = \mathbf{0}^{\intercal}$.

The implication in the other direction is in line with our explanation above. We first compute the matrix decomposition of $\mathbf{c}(\mathbf{s}, \hat{\mathbf{s}}, \mathbf{b}, x^*)^{\intercal} = \mathbf{M}_c \cdot \mathbf{v}_c^{\intercal}$, and consider a vector basis $\mathcal{V}_c = \{\mathtt{v}_1, ..., \mathtt{v}_{d_2}\}$ of the kernel Ker($\mathbf{M}_c$), where d_2 is the number of vectors in $\mathcal{V}_c$. We set

$$\begin{aligned} \hat{\mathbf{s}}_j &= ((\mathtt{v}_1)_l, ..., (\mathtt{v}_{d_2})_l) && \text{where } \mathbf{v}_l = \hat{s}_j \\ \mathbf{s}_j &= \mathbf{1}_{j+1}^{d_1} \\ (\mathbf{B}_k)_{l,m} &= (\mathtt{v}_m)_n && \text{where } \mathbf{v}_n = s_{l-1} b_k \end{aligned}$$

where $d_1 = w_1 + 1$. Note that, by the definition of a kernel, substituting the polynomials in $\mathbf{c}$ with these matrices and vectors yields $\mathbf{0}$.

Furthermore, we consider the vector of encodings $\mathbf{p_{enc}}'_{x,y}$ as in Equation 4.1, and its associated matrix decomposition $\mathbf{M}_{x,y} \cdot \mathbf{v}_{x,y}^{\intercal}$ and target vector $\mathtt{tv}_{x,y}$, where we assume, without loss of generality, that $(\mathbf{v}_{x,y})_1 = \alpha s$. From Proposition 2 and the trivial security of the PES-AC17, it follows that there exists a vector $\mathbf{w}_{x,y}$ such that $\mathbf{M}_{x,y} \cdot \mathbf{w}_{x,y}^{\intercal} = \mathbf{0}^{\intercal}$ and $\mathtt{tv}_{x,y} \cdot \mathbf{w}_{x,y}^{\intercal} \neq 0$, and in particular, that $(\mathbf{w}_{x,y})_1 = 1$. Then, we set

$$\begin{aligned} \mathbf{a}_l &= (\mathbf{w}_{x,y})_m && \text{where } (\mathbf{v}_{x,y})_m = \alpha s_{l+1} \\ \hat{\mathbf{r}}_l &= (\mathbf{w}_{x,y})_m && \text{where } (\mathbf{v}_{x,y})_m = \hat{r} s_{l+1}. \end{aligned}$$

To determine $\mathbf{r}_j$, we first consider the sub-vector $\mathbf{w}'_{x,y,j}$, where $(\mathbf{w}'_{x,y,j})_l = (\mathbf{w}_{x,y})_m$, such that $r_j (\mathbf{v}_c)_l = (\mathbf{v}_{x,y})_m$. Then, matrix $\mathbf{M}_{x,y}$ can be similarly reduced to the smaller sub-matrix $\mathbf{M}'_{x,y,j}$ corresponding to the rows associated with $r_j c_{j'}$ and the entries of $\mathbf{v}_{x,y}$ in which r_j occurs, e.g., $r_j s_{j'} b_k$. We assume that this sub-matrix is ordered in such a way that $\mathbf{M}'_{x,y,j} \cdot \mathbf{v}_c^{\intercal} = \mathbf{M}_c \cdot \mathbf{v}_c^{\intercal}$. For this sub-matrix, it holds that $\mathbf{M}'_{x,y,j} \cdot (\mathbf{w}'_{x,y,j})^{\intercal} = \mathbf{0}^{\intercal}$, and because $\mathbf{M}'_{x,y,j} \cdot \mathbf{v}_c^{\intercal} = \mathbf{M}_c \cdot \mathbf{v}_c^{\intercal}$, we therefore have $\mathbf{M}_c \cdot (\mathbf{w}'_{x,y,j})^{\intercal} = \mathbf{0}^{\intercal}$. Because $\mathbf{w}'_{x,y,j}$ is in Ker($\mathbf{M}_c$), it follows

that it can be written as a linear combination of the vectors in $\mathcal{V}_c$. In particular, we set $\mathbf{r}_j$ to be the vector for which holds that $\mathbf{w}'_{x,y,j} = \sum_{i\in[d_2]}(\mathbf{r}_j)_i \cdot \mathsf{v}_i$. Note that, therefore, substituting $r_j b_k$ with $\mathbf{B}_k \cdot \mathbf{r}_j^\intercal$ yields the entries of $\mathbf{w}'_{x,y,j}$ associated with $r_j s_{j'} b_k$. Hence, making the proper substitutions in $\mathbf{k}$ yields evaluations to $\mathbf{0}$. Lastly, it holds that $\mathbf{a} \cdot \mathbf{s}_0^\intercal \neq 0$, which follows from the fact that $\mathbf{s}_0 = \mathbf{1}_1^{d_1}$ and $\mathbf{a}_1 = (\mathbf{w}_{x,y})_1 = 1$, because the first entry of $\mathbf{v}_{x,y}$ is assumed to be αs.

For the co-selective property, the strategy is similar. However, instead of starting with $\mathbf{c}$, we start with $\mathbf{k}$ and first compute the substitutions of $r_j, \hat{r}_{j'}$ and b_k.

B.8 Proof of Corollary 1

Proof. That the enhanced selective symbolic property $\mathsf{Sym\text{-}Prop}^+$ implies collusion security follows from the proofs of Theorems 2 and 3. In particular, from the selective symbolic property, it follows that, for all $x \in \mathcal{X}$ and $y_1, y_2, ... \in \mathcal{Y}$, some vector $\mathbf{w}_{x,y_i}$ can be constructed as in the proof of Theorem 3, for which we have already shown that $\mathbf{M}_{x,y_i} \cdot \mathbf{w}_{x,y_i}^\intercal = \mathbf{0}^\intercal$ and $\mathsf{tv}_{x,y_i} \cdot \mathbf{w}_{x,y_i}^\intercal \neq 0$. Then, we want to show that

- $\mathbf{w}_{x,y_i,0} = \mathbf{w}_{x,y_j,0}$ for all i, j, and
- $\mathbf{w}_{x,y_i,1} = \mathbf{0}$,

where $\mathbf{w}_{x,y_i,0}$ corresponds to the entry αs of $\mathbf{v}_{x,y_i,0}$, and $\mathbf{w}_{x,y_i,1}$ corresponds to the entries αs_j with $j \in [w_1]$ of $\mathbf{v}_{x,y_i,0}$. If those equations hold, it follows from Theorem 2 that the PES-AC17 is collusion secure. Now, note that, by $\mathsf{Sym\text{-}Prop}^+$, it holds that $\mathbf{a} = \mathbf{1}_1^{d_1}$. Thus, $\mathbf{w}_{x,y_i,0} = \mathbf{s}_0 \cdot \mathbf{a}^\intercal = 1$, and $\mathbf{w}_{x,y_i,1} = (\mathbf{s}_1 \cdot \mathbf{a}^\intercal, ..., \mathbf{s}_{w_1} \cdot \mathbf{a}^\intercal) = \mathbf{0}$, because $\mathbf{s}_i = \mathbf{1}_{i+1}^{d_1}$.

That our property used to prove collusion security implies selective $\mathsf{Sym\text{-}Prop}^+$ follows similarly. We again follow the proof of Theorem 3, but this time, to obtain the matrices $\mathbf{B}_k$ and vectors $\mathbf{a}, \mathbf{r}_i, \hat{\mathbf{r}}_{i'}, \mathbf{s}_j$ and $\hat{\mathbf{s}}_{j'}$. Note that, for these vectors, the selective symbolic property holds. Thus, we only need to show that $\mathbf{a} = \mathbf{1}_1^{d_1}$ holds. Because the PES-AC17 is collusion secure, we can assume, for the vector $\mathbf{w}_{x,y_i}$ that is used to construct the matrices and vectors, that $\mathbf{w}_{x,y_i,0} = \mathbf{w}_{x,y_j,0}$ holds for all i, j and $\mathbf{w}_{x,y_i,1} = \mathbf{0}$. Without loss of generality, we can assume that $\mathbf{w}_{x,y_j,0} = 1$, because $\mathsf{tv}_{x,y_j} \cdot \mathbf{w}_{x,y_j}^\intercal = \mathbf{w}_{x,y_j,0} \neq 0$, and we can simply normalize $\mathbf{w}_{x,y_j}$ if it is not equal to 1. Since $\mathbf{a}$ is constructed such that

$$\mathbf{a}_l = (\mathbf{w}_{x,y})_m \text{ where } (\mathbf{v}_{x,y})_m = \alpha s_{l+1},$$

we know that $\mathbf{a}$ is indeed 0 in the entries $2, ..., d_1$. Hence, $\mathbf{a} = \mathbf{1}_1^{d_1}$.

B.9 Proof of Proposition 4

Proof. Suppose that such vectors $\bar{\mathbf{w}}_{x,y,1}, ..., \bar{\mathbf{w}}_{x,y,w_1+1}$ exist, and assume that the intersection of the spans of $\mathfrak{V}$ and the row space of $\mathbf{M}_{x,y}$ is not empty, i.e., there exists vector v with $(\mathsf{v})_i \neq 0$ for at least one $i \in [w_1 + 1]$, where $(\mathsf{v})_{i'} = 0$ for all

$i' > w_1$. For vector $\bar{\mathbf{w}}_{x,y,i}$, we then know that $(\mathsf{v})_i \cdot \bar{\mathbf{w}}_{x,y,i} = (\mathsf{v})_i \neq 0$. We will show that this contradicts the assumption that v is in the span of the rows of $\mathbf{M}_{x,y}$, i.e., there exists a vector $\mathbf{e}$ such that $\mathsf{v} = \mathbf{e} \cdot \mathbf{M}_{x,y}$. Then, because $\mathbf{M}_{x,y} \cdot \bar{\mathbf{w}}_{x,y,i}^{\mathsf{T}} = \mathbf{0}^{\mathsf{T}}$, we know that $\mathsf{v} \cdot \bar{\mathbf{w}}_{x,y,i}^{\mathsf{T}} = (\mathbf{e} \cdot \mathbf{M}_{x,y}^{\mathsf{T}}) \cdot \bar{\mathbf{w}}_{x,y,i}^{\mathsf{T}} = \mathbf{e} \cdot (\mathbf{M}_{x,y}^{\mathsf{T}} \cdot \bar{\mathbf{w}}_{x,y,i}^{\mathsf{T}}) = \mathbf{0}$.

C Security proofs for schemes with rational fractions

To use the matrix decompositions more conveniently in pair encoding schemes with rational fractions, we first eliminate the rational fractions. Intuitively, we do this by multiplying the encodings vector $\mathbf{p_{enc}}$ by the product of all denominators. In particular, eliminating the fractions simplifies the construction of the matrix decompositions. To show this, we consider an example, i.e., the variant of the IBE scheme by Wee [35] given by ABGW17, which has the following encodings:

- $\mathbf{mpk} = b$
- $\mathbf{k}(\alpha, r, b, y) = \frac{\alpha}{b+y}$
- $\mathbf{c}(s, b, x) = s(b + x)$

To find an attack, i.e., recover αs, we first consider the products of all key and ciphertext encodings: $\mathbf{p_{enc}} = \frac{\alpha}{b+y} \cdot s(b+x) = \frac{\alpha sb}{b+y} + \frac{\alpha sx}{b+y}$, with $\mathsf{V} = \{\alpha, b, r, s, x, y\}$. The annotation function annot maps $\{\alpha, b, r, s\}$ to $\mathsf{unknown}$ and $\{x, y\}$ to known. Then, the matrix decomposition of $\mathbf{p_{enc}}$ can be

$$\begin{pmatrix} 1 & x \end{pmatrix} \cdot \begin{pmatrix} \frac{\alpha sb}{b+y} \\ \frac{\alpha s}{b+y} \end{pmatrix},$$

but it can also be

$$\begin{aligned} \begin{pmatrix} \frac{x}{y} & (1 - \frac{x}{y}) \end{pmatrix} \cdot \begin{pmatrix} \alpha s \\ \frac{\alpha sb}{b+y} \end{pmatrix} &= \frac{x}{y}\alpha s + \left(1 - \frac{x}{y}\right) \frac{\alpha sb}{b+y} \\ &= \frac{\frac{x}{y}\alpha s(b+y)}{b+y} + \frac{\alpha sb}{b+y} = \frac{\alpha sb}{b+y} + \frac{\alpha sx}{b+y}. \end{aligned}$$

For the above decompositions, the target vectors for the attack are given by

$$\alpha s = \begin{pmatrix} 1 & y \end{pmatrix} \cdot \begin{pmatrix} \frac{\alpha sb}{b+y} \\ \frac{\alpha s}{b+y} \end{pmatrix} = \begin{pmatrix} 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} \alpha s \\ \frac{\alpha sb}{b+y} \end{pmatrix}.$$

The reason that this encoding has multiple different decompositions is that it contains a denominator that is a polynomial, i.e., $b + y$. Although it is not necessarily problematic that encodings may have multiple different compositions, it complicates the construction of an algorithm to decompose the encodings. As the example illustrates, αs is not linearly independent of the entries in the vector $\mathbf{v} = \begin{pmatrix} \frac{\alpha sb}{b+y} \\ \frac{\alpha s}{b+y} \end{pmatrix}$. Hence, to decompose the encodings, it must be checked for each monomial whether it is a linear combination of all other monomials.

To simplify this effort, we eliminate the denominators by multiplying all encodings and the target, e.g., αs, with a product of all denominators, and consider resulting encoding polynomials in the canonical form. In particular, each polynomial consists of monomials of the form $c \cdot \prod_i \mathsf{v}_i^{j_i}$, where $c \in \mathbb{Z}_p(\mathsf{coef})$ is a coefficient, $\mathsf{V} = \{\mathsf{v}_1, \mathsf{v}_2, ...\}$ and $j_i \in \mathbb{N}$. In this case, it is easy to verify whether a newly considered monomial is independent of an existing set of monomials, and the attack found with the transformed encodings also works for original encodings.

Proposition 5. *Let* $\mathbf{p_{enc}}$ *be a vector of encodings, and let* $\lambda \in \mathbb{Z}_p(\mathsf{known}, \mathsf{unknown})$ *be an element in the extended field of rational fractions. Consider the matrix decompositions of* $\mathbf{p_{enc}} = \mathbf{M}_1 \cdot \mathbf{v}_1^\intercal$ *and* $\lambda \cdot \mathbf{p_{enc}} = \mathbf{M}_2 \cdot \mathbf{v}_2^\intercal$*, and let* $\mathbf{tv}_1$ *and* $\mathbf{tv}_2$ *be such that* $\mathbf{tv}_1 \cdot \mathbf{v}_1^\intercal = \alpha s$ *and* $\mathbf{tv}_2 \cdot \mathbf{v}_2^\intercal = \lambda \alpha s$*. Let* $\mathbf{e} \in \mathbb{Z}_p^{|\mathbf{p_{enc}}|}$*. Then,* $\mathbf{e} \cdot \mathbf{M}_1 = \mathbf{tv}_1$ *if and only if* $\mathbf{e} \cdot \mathbf{M}_2 = \mathbf{tv}_2$*.*

Proof. We prove that $\mathbf{e} \cdot \mathbf{M}_2 = \mathbf{tv}_2$ implies that $\mathbf{e} \cdot \mathbf{M}_1 = \mathbf{tv}_1$. The implication in the other way is similar. Suppose that $\mathbf{e} \cdot \mathbf{M}_2 = \mathbf{tv}_2$, and thus, that $\mathbf{e} \cdot (\lambda \cdot \mathbf{p_{enc}}) = \lambda \alpha s$. From this, it follows that $\lambda \cdot (\mathbf{e} \cdot \mathbf{p_{enc}}) = \lambda \alpha s$ and canceling out λ on both sides yields $\mathbf{e} \cdot \mathbf{p_{enc}} = \alpha s$. Therefore, $\mathbf{e}$ constitutes a linear combination of the entries in $\mathbf{p_{enc}}$ that yields αs. Then, it follows from the equivalence of the matrix and the polynomial notation (as proven in Proposition 1) that $\mathbf{e}$ is also a linear combination of the rows of $\mathbf{M}_1$ that yields $\mathbf{tv}_1$, and thus, $\mathbf{e} \cdot \mathbf{M}_1 = \mathbf{tv}_1$.

From this, it also follows that finding a vector $\mathbf{w}$ as in Proposition 2 for the matrix decomposition of the version of the encodings $\mathbf{p_{enc}}$ in which the fractions are eliminated also proves that the target vector is not in the span of the matrix in the matrix decomposition of the regular version of the encodings.

Corollary 3. *Let* $\mathbf{p_{enc}}$ *be a vector of encodings, and let* $\lambda \in \mathbb{Z}_p(\mathsf{known}, \mathsf{unknown})$ *be an element in the extended field of rational fractions. Consider the matrix decompositions of* $\mathbf{p_{enc}} = \mathbf{M}_1 \cdot \mathbf{v}_1^\intercal$ *and* $\lambda \cdot \mathbf{p_{enc}} = \mathbf{M}_2 \cdot \mathbf{v}_2^\intercal$*, and let* $\mathbf{tv}_1$ *and* $\mathbf{tv}_2$ *be such that* $\mathbf{tv}_1 \cdot \mathbf{v}_1^\intercal = \alpha s$ *and* $\mathbf{tv}_2 \cdot \mathbf{v}_2^\intercal = \lambda \alpha s$*. If there exists a vector* $\mathbf{w}$ *such that* $\mathbf{M}_2 \cdot \mathbf{w}^\intercal = \mathbf{0}^\intercal$ *and* $\mathbf{tv}_2 \cdot \mathbf{w}^\intercal \neq 0$*, then* $\mathbf{tv}_1$ *is not in the span of rows of* $\mathbf{M}_1$*.*

Proof. If such a vector $\mathbf{w}$ exists, then $\mathbf{tv}_2$ is not in the span of rows of $\mathbf{M}_2$ (Proposition 2). Because there exists no $\mathbf{e}$ such that $\mathbf{e} \cdot \mathbf{M}_2 = \mathbf{tv}_2$, it follows from Proposition 5 that there exists no $\mathbf{e}$ such that $\mathbf{e} \cdot \mathbf{M}_2 = \mathbf{tv}_2$. Thus, $\mathbf{tv}_1$ is not in the span of rows of $\mathbf{M}_1$.

To prove collusion security, we also need to be able to translate the vector $\mathbf{w}$ that is found for the "normalized" encodings to the original encodings. We can do this by dividing each entry of vector $\mathbf{v}_2$ by λ, i.e., set $\mathbf{v}_3 = \lambda^{-1} \cdot \mathbf{v}_2$. Because of the independence of the original entries, the resulting entries are also independent. Note that the substitutions that we find for the $\mathbf{v}_2$ also work for $\mathbf{v}_3$, as long as λ does not evaluate to 0. Furthermore, we need to ensure that the vectors $\mathbf{v}_{x,y}$ match in the entries $\mathbf{v}_{x,y,0}$ and $\mathbf{v}_{x,y,1}$, as well as $\mathbf{w}_{x,y,0}$ and $\mathbf{w}_{x,y,1}$. To ensure this, we check whether evaluating $\mathbf{M}_{x,y}$ restricted to the entries

associated with $\mathbf{v}_{x,y,0}$ and $\mathbf{v}_{x,y,1}$ yields the same polynomial for each y. We do the same for $\mathbf{w}_{x,y,0}$ and $\mathbf{w}_{x,y,1}$. Then, we can create new matrix decompositions and vectors $\mathbf{w}_{x,y,0}$ and $\mathbf{w}_{x,y,1}$ that are equal for each y, and thus, the collusion security property is satisfied.

Proposition 6 (Proofs of collusion security for schemes with rational fractions). *Let* $(\mathrm{mpk}, \mathbf{k}, \mathbf{c})$ *be a PES for predicate* $P\colon \mathcal{X} \times \mathcal{Y} \to \{0,1\}$*, and consider the matrix decompositions of* $\mathbf{p_{enc}}_{x,y_i}$ *for* $x \in \mathcal{X}$ *and* $y_1, y_2, ..., y_n \in \mathcal{Y}$*, and their associated target vectors* $\mathbf{tv}_{x,y_i}$*. Like in Proposition 3, we assume that the matrix decompositions* $\mathbf{p_{enc}}^{\intercal}_{x,y_i} = \mathbf{M}_{x,y_i} \cdot \mathbf{v}^{\intercal}_{x,y_i}$ *are such that* $\mathbf{v}_{x,y_i}$ *is ordered. If for all* $x \in \mathcal{X}, y_1, y_2, ..., y_n \in \mathcal{Y}$ *with* $P(x, y_i) = 0$*,* $(\mathbf{v}_{x,y_i,0}, \mathbf{v}_{x,y_i,1}) = (\mathbf{v}_{x,y_j,0}, \mathbf{v}_{x,y_j,1})$ *for all* i, j*, and there exist vectors* $\mathbf{w}^{\intercal}_{x,y_i}$ *(as in Theorem 1) such that the requirements below hold, then the PES is collusion secure. First, we similarly split each matrix* $\mathbf{M}_{x,y_i}$ *in three parts, i.e.,* $\mathbf{M}_{x,y_i} = (\mathbf{M}_{x,y_i,0}, \mathbf{M}_{x,y_i,1}, \mathbf{M}_{x,y_i,2})$*, such that the number of columns of* $\mathbf{M}_{x,y_i,j}$ *is equal to* $|\mathbf{w}_{x,y_i,j}| = |\mathbf{v}_{x,y_i,j}|$ *for* $j \in \{0,1,2\}$*. Then, the requirements are*

- $\mathbf{M}_{x,y_i} \cdot \mathbf{w}^{\intercal}_{x,y_i} = \mathbf{0}^{\intercal}$ *and* $\mathbf{tv}_{x,y_i} \cdot \mathbf{w}^{\intercal}_{x,y_i} \neq 0$ *(i.e., the PES is trivially secure);*
- $\mathbf{M}_{x,y_i,l} \cdot \mathbf{v}^{\intercal}_{x,y_i,l} = \mathbf{M}_{x,y_j,l} \cdot \mathbf{v}^{\intercal}_{x,y_j,l}$ *for all* i, j *and* $l \in \{0,1\}$*;*
- $\mathbf{M}_{x,y_i,0} \cdot \mathbf{w}^{\intercal}_{x,y_i,0} = \mathbf{M}_{x,y_j,0} \cdot \mathbf{w}^{\intercal}_{x,y_j,0}$ *for all* i, j*, and*
- $\mathbf{w}_{x,y_i,1} = \mathbf{0}$*.*

Proof. Note that only the second and third requirement differ from the requirements for collusion security in Theorem 2. In fact, we will show that the second and third requirement imply the properties for collusion security. Suppose that $\mathbf{M}_{x,y_i,l} \cdot \mathbf{v}^{\intercal}_{x,y_i,l} = \mathbf{M}_{x,y_j,l} \cdot \mathbf{v}^{\intercal}_{x,y_j,l}$ holds for $l \in \{0,1\}$. Then, we can create new decompositions

$$\mathbf{M}'_{x,y_i} = (\mathbf{M}_{x,y_1,0}, \mathbf{M}_{x,y_1,1}, \mathbf{M}_{x,y_i,2})$$
$$\mathbf{v}'_{x,y_i} = (\mathbf{v}_{x,y_1,0}, \mathbf{v}_{x,y_1,1}, \mathbf{v}_{x,y_i,2})$$

that are equivalent to the original decompositions, i.e., $\mathbf{M}'_{x,y_i} \cdot (\mathbf{v}'_{x,y_i})^{\intercal} = \mathbf{M}_{x,y_i} \cdot \mathbf{v}^{\intercal}_{x,y_i}$. Furthermore, we can create vectors

$$\mathbf{w}'_{x,y_i} = (\mathbf{w}_{x,y_1,0}, \mathbf{w}_{x,y_1,1}, \mathbf{w}_{x,y_i,2}),$$

for which it similarly follows that they satisfy the requirements for collusion security in Theorem 2.

Using these results, we can now more efficiently prove security of the previously considered IBE scheme. First, we eliminate the fractions from $\mathbf{p_{enc}}$ by multiplying all encodings by $(b + y)$. This yields $\mathbf{p_{enc}}' = \alpha sb + \alpha sx$, and target vector $\mathbf{tv}' = \alpha sb + \alpha sy$. We then obtain the matrix decomposition

$$\mathbf{p_{enc}}' = \mathbf{M} \cdot \mathbf{v}^{\intercal} = \begin{pmatrix} x & 1 \end{pmatrix} \cdot \begin{pmatrix} \alpha s \\ \alpha sb \end{pmatrix}, \text{ with } \mathbf{tv}' = \begin{pmatrix} y & 1 \end{pmatrix}.$$

To show that $\mathbf{tv}'$ is not in the span of $\begin{pmatrix} x & 1 \end{pmatrix}$, we devise the vector $\mathbf{w} = \begin{pmatrix} -1 & x \end{pmatrix}$, for which holds that $\mathbf{M} \cdot \mathbf{w}^\intercal = \begin{pmatrix} x & 1 \end{pmatrix} \cdot \begin{pmatrix} -1 \\ x \end{pmatrix} = 0$ and $\mathbf{tv}' \cdot \mathbf{w}^\intercal = \begin{pmatrix} y & 1 \end{pmatrix} \cdot \begin{pmatrix} -1 \\ x \end{pmatrix} = y - x$, which is not equal to 0, because $P(x, y) = 0$ holds if and only if $x \neq y$. Now, note that this vector also satisfies the properties required to prove collusion security via Proposition 6.

D Linear algebra tools

For our tool, we use several linear algebra results to analyze the schemes.

D.1 Computing matrix decompositions

We compute the matrix decompositions, given a vector of encodings $\mathbf{p_{enc}}$, as follows. Let $\mathbb{Z}_p(\mathsf{coef})$ denote the space used for the coefficients (associated with the known variables), and let $\mathfrak{K} = \{\mathsf{v}_1, \mathsf{v}_2, ...\}$ be the set of unknown variables. First, we write each entry of $\mathbf{p_{enc}}$ in the canonical form, such that each entry is written as a sum of monomials of the form $c \cdot \prod_i \mathsf{v}_i^{j_i} / f$, where $c \in \mathbb{Z}_p(\mathsf{coef})$ is a coefficient, $j_i \in \mathbb{N}$ and f is a polynomial with coefficients in $\mathbb{Z}_p(\mathsf{coef})$ and variables in $\mathfrak{K}$. We then construct the vector $\mathbf{v}$ of the decomposition from the unique monomials $\prod_i \mathsf{v}_i^{j_i} / f$, such that they are all linearly independent. For schemes in which all monomials are with $f \in \mathbb{Z}_p(\mathsf{coef})$ (and effectively, $f = 1$, because coefficient $c \in \mathbb{Z}_p(\mathsf{coef})$), this effort is trivial. We show in Section C how this can be done more effectively for schemes with $f \notin \mathbb{Z}_p(\mathsf{coef})$. Once we have the vector $\mathbf{v}$ and the encodings, it is simple to construct the matrix.

D.2 Finding linear combinations

In our framework, the effort of finding attacks is equivalent to finding a linear combination of rows of a matrix that yields the target vector. In linear algebra, there exist various methods to find such a linear combination, e.g., Gaussian elimination. In our tool, we use another trick, using the kernel of the matrix.

Lemma 4. *Let* $\mathbf{M}$ *be a matrix, and* $\mathbf{tv}$ *be a target vector. Consider the kernel* $\mathrm{Ker}(\mathbf{M}')$ *of* $\mathbf{M}' = \begin{pmatrix} \mathbf{M} \\ \mathbf{tv} \end{pmatrix}^\intercal$*. Then, if there exists a vector* $\mathbf{e}' \in \mathrm{Ker}(\mathbf{M}')$ *in which the last entry is* -1*, then* $\mathbf{e}' = (\mathbf{e}, -1)$ *is such that* $\mathbf{e}$ *is a linear combination with* $\mathbf{e} \cdot \mathbf{M} = \mathbf{tv}$*. Furthermore, such* $\mathbf{e}'$ *exists, if* $\mathbf{tv}$ *is in the span of the rows of* $\mathbf{M}$*.*

Proof. By definition, $\mathrm{Ker}(\mathbf{M}') = \{\mathbf{w} \mid \mathbf{M}' \cdot \mathbf{w}^\intercal = \mathbf{0}^\intercal\}$. Hence, for $\mathbf{e}'$, it holds that $\mathbf{M}' \cdot (\mathbf{e}')^\intercal = \mathbf{0}^\intercal$. Because $\mathbf{M}' \cdot (\mathbf{e}')^\intercal = \mathbf{M}^\intercal \cdot \mathbf{e} + \mathbf{tv}^\intercal \cdot (-1) = \mathbf{0}^\intercal$, we have that $\mathbf{M}^\intercal \cdot \mathbf{e} = \mathbf{tv}^\intercal$, and thus, $\mathbf{e} \cdot \mathbf{M} = \mathbf{tv}$. Conversely, if $\mathbf{tv}$ is in the span of $\mathbf{M}$, then there exists $\mathbf{e}$ such that $\mathbf{e} \cdot \mathbf{M} = \mathbf{tv}$, and thus, $\mathbf{M}^\intercal \cdot \mathbf{e} = \mathbf{tv}^\intercal$. From this, it follows that $\mathbf{M}^\intercal \cdot \mathbf{e} + \mathbf{tv}^\intercal \cdot (-1) = \mathbf{0}^\intercal = \mathbf{M}' \cdot (\mathbf{e}')^\intercal = \mathbf{0}^\intercal$. Hence $\mathbf{e}'$ is in the kernel of $\mathbf{M}'$.

D.3 Finding all vectors with certain properties in the kernel

To prove the properties required to prove trivial and collusion security, we need to find a vector $\mathbf{w}$ such that it is in the kernel of $\mathbf{M}$ and satisfies the property that $\mathbf{tv} \cdot \mathbf{w}^\intercal \neq 0$. For collusion security, we also require that certain entries of $\mathbf{w}$ need to be 0. We do this via the following lemmas.

Lemma 5. *Let $\mathbf{M}$ be a matrix and $\mathbf{tv}$ be a target vector that is not in the span of the rows of $\mathbf{M}$. Let $\mathcal{V} = \{\mathsf{v}_1, ..., \mathsf{v}_n\}$ be a basis for the kernel of $\mathbf{M}$. Then, $\mathcal{W} \subseteq \mathcal{V}$ with $\mathcal{W} = \{\mathsf{v} \in \mathcal{V} \mid \mathbf{tv} \cdot \mathbf{w}^\intercal \neq 0\}$ is not empty.*

Proof. Suppose that $\mathcal{W} = \emptyset$. Then, for all i, it holds that $\mathbf{tv} \cdot \mathsf{v}_i^\intercal = 0$. Furthermore, for all $\mathbf{w} \in \mathrm{Ker}(\mathbf{M})$, because we have that $\mathbf{w} = \sum_i c_i \mathsf{v}_i$, it holds that $\mathbf{tv} \cdot \mathbf{w}^\intercal = \mathbf{tv} \cdot (\sum_i c_i \mathsf{v}_i)^\intercal = \sum_i c_i (\mathbf{tv} \cdot \mathsf{v}_i) = 0$. However, because $\mathbf{tv}$ is not in the span of $\mathbf{M}$, it should be that some vector $\mathbf{w} \in \mathrm{Ker}(\mathbf{M})$ exists such that $\mathbf{tv} \cdot \mathbf{w}^\intercal \neq 0$ (Proposition 2), this is a contradiction. Thus, $\mathcal{W} \neq \emptyset$.

Lemma 6. *Let $\mathbf{M}$ be a matrix and $\mathbf{tv}$ be a target vector that is not in the span of the rows of $\mathbf{M}$. Let $\mathcal{I}$ be a set of indices such that $(\mathbf{tv})_i = 0$ for all $i \in \mathcal{I}$. (Here, $\mathcal{I}$ does not need to contain all indices for which $\mathbf{tv}$ is 0.) Let $\mathcal{W}$ be a basis for $\{\mathsf{v} \in \mathrm{Ker}(\mathbf{M}) \mid \forall i \in \mathcal{I} \;\; [(\mathsf{v})_i = 0]\}$. Let $\mathcal{W}' \subseteq \mathcal{W}$ be such that $\mathcal{W}' = \{\mathsf{v} \in \mathcal{W} \mid \mathbf{tv} \cdot \mathsf{v}^\intercal \neq 0\}$. Then, if some vector $\bar{\mathbf{w}} \in \mathrm{Ker}(\mathbf{M})$ exists with $\mathbf{tv} \cdot \bar{\mathbf{w}}^\intercal \neq 0$ and $(\bar{\mathbf{w}})_i = 0$ for all $i \in \mathcal{I}$, then $\mathcal{W}' \neq \emptyset$.*

Proof. Suppose $\mathcal{W} = \{\mathsf{w}_1, ..., \mathsf{w}_n\}$ for some $n \in \mathbb{N}$. We use that some vector $\bar{\mathbf{w}} \in \mathrm{Ker}(\mathbf{M})$ exists with $\mathbf{tv} \cdot \bar{\mathbf{w}}^\intercal \neq 0$ and $(\bar{\mathbf{w}})_i = 0$ for all $i \in \mathcal{I}$. In particular, this means that $\bar{\mathbf{w}} = \sum_i c_i \mathsf{w}_i$ for some coefficients c_i. Then, similarly as in the proof of Lemma 5, because $\mathbf{tv} \cdot \bar{\mathbf{w}}^\intercal \neq 0$, it follows that there must be at least one i for which $\mathbf{tv} \cdot \mathsf{w}_i^\intercal \neq 0$. Hence, $\mathcal{W}' \neq \emptyset$.

Lemma 7. *Let $\mathbf{M}$ be a matrix and $\mathbf{tv}$ be a target vector that is not in the span of the rows of $\mathbf{M}$. Let $\mathcal{V} = \{\mathsf{v}_1, ..., \mathsf{v}_n\}$ be a basis for the kernel of $\mathbf{M}$, and let $\mathcal{I}$ be a set of indices such that $(\mathbf{tv})_i = 0$ for all $i \in \mathcal{I}$. Let $\mathcal{V}' = \{\mathsf{v}'_1, ..., \mathsf{v}'_n\}$ be the set of vectors in $\mathcal{V}$ truncated to only consider the indices in $\mathcal{I}$, i.e., $\mathsf{v}'_j = (\{(\mathsf{v}_j)_i\}_{i \in \mathcal{I}})$. Let $\mathbf{V} = (\mathsf{v}_1^\intercal, ..., \mathsf{v}_n^\intercal)$ and $\mathbf{V}' = ((\mathsf{v}'_1)^\intercal, ..., (\mathsf{v}'_n)^\intercal)$ be the matrices associated with $\mathcal{V}$ and $\mathcal{V}'$. Consider then a vector basis $\mathcal{W}$ of $\mathrm{Ker}(\mathbf{V}') = \{\mathsf{w} \mid \mathbf{V}' \cdot \mathsf{w}^\intercal = \mathbf{0}^\intercal\}$. Then, $\mathcal{W}' = \{\mathbf{V} \cdot \mathsf{w}^\intercal \mid \mathsf{w} \in \mathcal{W}\}$ is a basis for $\{\mathsf{v} \in \mathrm{Ker}(\mathbf{M}) \mid \forall i \in \mathcal{I} \;\; [(\mathsf{v})_i = 0]\}$.*

Proof. Let $\mathsf{w} \in \{\mathsf{v} \in \mathrm{Ker}(\mathbf{M}) \mid \forall i \in \mathcal{I} \;\; [(\mathsf{v})_i = 0]\}$. Then, $\mathsf{w} = \sum_j c_j \mathsf{v}_j$ for some coefficients c_j. For these coefficients, it holds that $\mathsf{c} = (c_1, ..., c_n)$ is a vector such that $\mathbf{V}' \cdot \mathsf{c}^\intercal = \mathbf{0}^\intercal$. Thus, $\mathsf{c} = \sum_k d_k \bar{\mathsf{w}}_k$, where $\mathcal{W} = \{\bar{\mathsf{w}}_1, ..., \bar{\mathsf{w}}_{n'}\}$. Note that $\mathcal{W}' = \{\mathbf{V} \cdot \bar{\mathsf{w}}_1^\intercal, ..., \mathbf{V} \cdot \bar{\mathsf{w}}_{n'}^\intercal\}$, where $\mathbf{V} \cdot \bar{\mathsf{w}}_k^\intercal = \sum_j (\bar{\mathsf{w}}_k)_j \mathsf{v}_j$. Substituting $c_j = \sum_k d_k (\bar{\mathsf{w}}_k)_j$ then yields

$$\begin{aligned} \mathsf{w} &= \sum_j c_j \mathsf{v}_j = \sum_j \left(\sum_k d_k (\bar{\mathsf{w}}_k)_j \right) \mathsf{v}_j \\ &= \sum_{j,k} d_k (\bar{\mathsf{w}}_k)_j \mathsf{v}_j = \sum_k d_k \mathbf{V} \cdot (\bar{\mathsf{w}}_k^\intercal) \,. \end{aligned}$$

Hence, $\mathcal{W}'$ is a basis for $\{\mathsf{v} \in \mathrm{Ker}(\mathbf{M}) \mid \forall i \in \mathcal{I} \;\; [(\mathsf{v})_i = 0]\}$.

E Example of a JSON input

Listing 1.1. Description of the unbounded ABGW17-CP-ABE scheme in ACABELLA format

```
{
    "scheme_id": "abgw17-cp-abe-complete",
    "security":{
        "analysis": "security",
        "k": ["alpha + b*r", "r", "b'*r/(b0 + b1*y)",
            ↪ "b'*r/(b0 + b1*x1)"],
        "c": ["A11*s + A12*sp", "b*(A11*s + A12*sp) +
            ↪ b'*s1", "s1*(b0 + b1*x1)"],
        "mpk": ["b", "b0", "b1", "b'", "1"],
        "key" : "alpha * s",
        "unknown_vars" :  ["alpha", "b", "b'", "b0", "b1",
            ↪ "r", "rp", "r0", "r1", "s", "sp", "s1", "s2"],
        "corruptable_vars": []
    },
    "master_key":{
        "analysis": "master_key",
        "k": ["alpha + b*r", "r", "b'*r/(b0 + b1*y)",
            ↪ "b'*r/(b0 + b1*x1)"],
        "master_key": "alpha",
        "unknown_vars" :  ["alpha", "b", "b'", "b0", "b1",
            ↪ "r", "rp", "r0", "r1", "s", "sp", "s1", "s2"],
        "corruption_model" : "NoCorruption",
        "corruptable_vars": [],
        "MPK_CA": ["alpha"],
        "MPK_AA": [],
        "MPK_vars": ["b", "b0", "b1", "b'"],
        "GP_vars": []
    },
    "decryption":{
        "analysis": "decryption",
        "k": ["alpha + b*r", "r", "b'*r/(b0 + b1*y)",
            ↪ "b'*r/(b0 + b1*x1)", "alpha + b*rp", "rp",
            ↪ "b'*rp/(b0 + b1*y)", "b'*rp/(b0 + b1*x2)"],
        "c": ["s + sp", "b*(s + sp) + b'*s1", "s1*(b0 +
            ↪ b1*x1)", "-sp", "-b*sp + b'*s2", "s2*(b0 +
            ↪ b1*x2)"],
        "mpk": ["b", "b0", "b1", "b'", "1"],
        "gp": [],
        "key" : "alpha * s",
        "unknown_vars" :  ["alpha", "b", "b'", "b0", "b1",
            ↪ "r", "rp", "r0", "r1", "s", "sp", "s1", "s2"],
        "corruption_model": "NoCorruption",
        "corruptable_vars": [],
        "MPK_AAi": [],
        "MPK_AAj": [],
```

```
        "misc_vars": []
    }
}
```

F ACABELLA outputs

F.1 The BB-IBE1 scheme

Listing 1.2. Transcript of the FABEO property for the BB-IBE1 scheme

```
        MPK encodings:                    [b0, b1]
        Key encodings:                    [alpha + b0*r +
            ↪ b1*r*y, r]
        Ciphertext encodings:   [b0*s + b1*s*x, s]

        Generating transcript that proves that the FABEO
            ↪ property holds..
              For the transcript, we use the following
                  ↪ reference vector of monomials:
                      [b0*r*s, b1*r*s, alpha*s]

              The vector with 1 in the entry
                  ↪ corresponding to alpha*s and 0 in
                  ↪ the entry corresponding to  is:
                      [x/(-x + y), -1/(-x + y), 1]
```

F.2 The RW13 scheme

Listing 1.3. Transcript of the FABEO property for the RW13 scheme

```
              MPK encodings:                    [b0, b1, b,
                  ↪ bp]
              Key encodings:                    [alpha +
                  ↪ b*r, b0*rp + b1*rp*y + bp*r, r, rp,
                  ↪ b0*rp2 + b1*rp2*z + bp*r, rp2]
              Ciphertext encodings:   [b*s + bp*sp - v2,
                  ↪ b0*sp + b1*sp*x, s, sp, bp*sp2 + v2,
                  ↪ b0*sp2 + b1*sp2*z, sp2]

        Generating transcript that proves that the FABEO
            ↪ property holds..
              For the transcript, we use the following
                  ↪ reference vector of monomials:
                      [r*v2, b*r*s, bp*r*sp, b0*r*sp,
                          ↪ b1*r*sp, bp*r*sp2, b0*r*sp2,
                          ↪ b1*r*sp2, rp*v2, b*rp*s,
                          ↪ bp*rp*sp, b0*rp*sp, b1*rp*sp,
```

```
                    ↪ bp*rp*sp2, b0*rp*sp2,
                    ↪ b1*rp*sp2, rp2*v2, b*rp2*s,
                    ↪ bp*rp2*sp, b0*rp2*sp,
                    ↪ b1*rp2*sp, bp*rp2*sp2,
                    ↪ b0*rp2*sp2, b1*rp2*sp2,
                    ↪ alpha*s, b0*rp*s, bp*r*s,
                    ↪ b1*rp*s, b0*rp2*s, b1*rp2*s,
                    ↪ alpha*sp, b*r*sp, alpha*sp2,
                    ↪ b*r*sp2]

                The vector with 1 in the entry
                    ↪ corresponding to alpha*s and 0 in
                    ↪ the entries corresponding to
                    ↪ alpha*sp,alpha*sp2 is:
                        [0, -1, 1, 0, 0, 0, 0, 0, 0, 0, 0,
                            ↪ -x/(x - y), 1/(x - y), 0, 0, 0
                            ↪ , 0, 0, 0, -x/(x - z), 1/(x -
                            ↪ z), 0, 0, 0, 1, 0, 0, 0, 0, 0
                            ↪ , 0, 0, 0, 0]

                The vector with 1 in the entry
                    ↪ corresponding to alpha*sp and 0 in
                    ↪ the entries corresponding to
                    ↪ alpha*s,alpha*sp2 is:
                        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                            ↪ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                            ↪ 0, 0, 0, 0, 0, 0, 0, 0, 1,
                            ↪ -1, 0, 0]

                The vector with 1 in the entry
                    ↪ corresponding to alpha*sp2 and 0 in
                    ↪ the entries corresponding to
                    ↪ alpha*s,alpha*sp is:
                        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                            ↪ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                            ↪ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                            ↪ 1, -1]
```

F.3 A new unbounded CP-ABE scheme

Listing 1.4. Transcript for the new unbounded CP-ABE scheme

```
MPK encodings:                  [b, b0, b1, 1]
Key encodings:                  [(alpha + r)/b, r,
    ↪ r/(b0 + b1*x1), r/(b0 + b1*y)]
Ciphertext encodings:   [b*s, (b0 + b1*x1)*(A11*s +
    ↪ A12*sp), (b0 + b1*x2)*(A21*s + A22*sp)]
```

```
Substitutions for the terms associated with the blinding
    ↪ value:
        - alpha*b*b0**2*s : (-A12*x2 + A12*y)/A11 +
            ↪ (-A12*x1*x2 + A12*x1*y)/A11
        - alpha*b*b0*b1*s : 0
        - alpha*b*b1**2*s : 0

Substitutions for the special terms that are shared among
    ↪ keys and are not associated with the blinding value:
        - alpha*b0**3*s : 0
        - alpha*b0**3*sp : 0
        - alpha*b0*b1**2*s : 0
        - alpha*b0**2*b1*s : 0
        - alpha*b1**3*s : 0
        - alpha*b0*b1**2*sp : 0
        - alpha*b0**2*b1*sp : 0
        - alpha*b1**3*sp : 0
        - alpha*b*b0**2 : 0
        - alpha*b*b0*b1 : 0
        - alpha*b*b1**2 : 0
        - alpha*b0**3 : 0
        - alpha*b0**2*b1 : 0
        - alpha*b0*b1**2 : 0
        - alpha*b1**3 : 0
        - alpha*b0**2 : 0
        - alpha*b0*b1 : 0
        - alpha*b1**2 : 0
        - b**3*b0**2*s : 0
        - b**3*b0*b1*s : 0
        - b**3*b1**2*s : 0
        - b**2*b0**3*s : 0
        - b**2*b0**2*b1*s : 0
        - b**2*b0*b1**2*s : 0
        - b**2*b1**3*s : 0
        - b**2*b0**2*s : 0
        - b**2*b0*b1*s : 0
        - b**2*b1**2*s : 0
        - b**2*b0**3*sp : 0
        - b**2*b0*b1**2*sp : 0
        - b**2*b0**2*b1*sp : 0
        - b**2*b1**3*sp : 0
        - b*b0**4*s : 0
        - b*b0**4*sp : 0
        - b*b0**3*b1*s : 0
        - b*b0**2*b1**2*s : 0
        - b*b0**3*b1*sp : 0
        - b*b0**2*b1**2*sp : 0
        - b*b0*b1**3*s : 0
        - b*b0*b1**3*sp : 0
        - b*b1**4*s : 0
```

```
- b*b1**4*sp : 0
- b*b0**3*s : 0
- b*b0**3*sp : 0
- b*b0*b1**2*s : 0
- b*b0**2*b1*s : 0
- b*b1**3*s : 0
- b*b0*b1**2*sp : 0
- b*b0**2*b1*sp : 0
- b*b1**3*sp : 0

 Substitutions for the rest of the terms:
- b*b0**2*r*s : (A11*A22*x1**2 -
    ↪ A12*A21*x2**2)/(A11*A21*x1 - A11*A21*x2) +
    ↪ (A11*A22*x1**2*x2 -
    ↪ A12*A21*x1*x2**2)/(A11*A21*x1 - A11*A21*x2)
- b*b0*b1*r*s : (-A11*A22*x1 +
    ↪ A12*A21*x2)/(A11*A21*x1 - A11*A21*x2) +
    ↪ (-A11*A22*x1*x2 + A12*A21*x1*x2)/(A11*A21*x1
    ↪ - A11*A21*x2)
- b*b1**2*r*s : (A11*A22 - A12*A21)/(A11*A21*x1 -
    ↪ A11*A21*x2) + (A11*A22*x2 -
    ↪ A12*A21*x1)/(A11*A21*x1 - A11*A21*x2)
- b0**3*r*s : 0
- b0**3*r*sp : 0
- b0*b1**2*r*s : 0
- b0**2*b1*r*s : 0
- b1**3*r*s : 0
- b0*b1**2*r*sp : 0
- b0**2*b1*r*sp : 0
- b1**3*r*sp : 0
- b**2*b0**2*r*s : 0
- b**2*b0*b1*r*s : 0
- b**2*b1**2*r*s : 0
- b*b0**3*r*s : 0
- b*b0**3*r*sp : 0
- b*b0*b1**2*r*s : 0
- b*b0**2*b1*r*s : 0
- b*b1**3*r*s : 0
- b*b0*b1**2*r*sp : 0
- b*b0**2*b1*r*sp : 0
- b*b1**3*r*sp : 0
- b**2*b0*r*s : 0
- b**2*b1*r*s : 0
- b*b0**2*r*sp : -x1*x2 - x1 - x2
- b*b0*b1*r*sp : 1
- b*b1**2*r*sp : 1
- b*b0**2*r : 0
- b*b0*b1*r : 0
- b*b1**2*r : 0
- b0**3*r : 0
```

```
    - b0**2*b1*r : 0
    - b0*b1**2*r : 0
    - b1**3*r : 0
    - b0**2*r : 0
    - b0*b1*r : 0
    - b1**2*r : 0
    - b**2*b0**2*r : 0
    - b**2*b0*b1*r : 0
    - b**2*b1**2*r : 0
    - b*b0**3*r : 0
    - b*b0**2*b1*r : 0
    - b*b0*b1**2*r : 0
    - b*b1**3*r : 0
    - b**2*b0*r : 0
    - b**2*b1*r : 0
    - b*b0*r : 0
    - b*b1*r : 0
```