Designs, Codes and Cryptography (2022) 90:2271–2299
https://doi.org/10.1007/s10623-022-01069-5

# Blind key-generation attribute-based encryption for general predicates

**Masayuki Abe**[1] · **Miguel Ambrona**[1]

Received: 19 January 2021 / Revised: 24 May 2022 / Accepted: 27 May 2022 /
Published online: 13 August 2022


**Abstract**
Attribute-based encryption (ABE) is a form of public-key encryption that allows fine-grained access control on encrypted data. Blind key-generation (BKG) attribute-based encryption (Rial, DESIGNS, CODES AND CRYPTOGRAPHY 2016) is a variant in which the master authority issues secret keys without learning any information about the attributes associated to them. This extra functionality makes it an appealing building block for several applications. In this work, we extend the generic framework of ABE based on pair encodings (Attrapadung, EUROCRYPT 2014) to support blind key-generation. In particular, we define two new notions of pair encodings that we coin *BKG-compatible* and *algebraic* pair encoding. We show that every encoding satisfies the former without loss of generality, whereas the latter is satisfied by all existing pair encodings from the literature. We then show how to enhance any ABE based on a BKG-compatible pair encoding to achieve honest-but-curious blind key-generation. In the case of algebraic encodings, our protocol admits a very efficient version, secure against malicious parties. The main advantage of our work is generality. Our protocol is designed over the recent and most advanced modular frameworks of ABE that can handle a rich variety of predicates.

**Keywords** ABE · Predicate encryption · Pair encodings · Blind key-generation

**Mathematics Subject Classification** 94A60

## 1 Introduction

Attribute-based encryption (ABE), first conceived by Sahai and Waters [46] and later introduced by Goyal et al. [33], is a form of public-key encryption in which ciphertexts and keys

Communicated by C. Padro.

✉ Miguel Ambrona
mac.ambrona@gmail.com

Masayuki Abe
masayuki.abe.cp@hco.ntt.co.jp

[1] NTT Social Informatics Laboratories, Tokyo, Japan

have attributes attached and the decryption ability of a key on a ciphertext is determined by a potentially complex access control policy involving these attributes. In particular, it is guaranteed that the decryption of a ciphertext $\mathsf{ct}_x$ associated to attribute $x$ with a secret key $\mathsf{sk}_y$ associated to attribute $y$ succeeds if and only if these values satisfy the predicate, that is, $P(x, y) = 1$. Note that this definition generalizes the original definition of ABE [33], introduced as key-policy ABE (KP-ABE), where values $x$ correspond to Boolean vectors, while values $y$ correspond to Boolean functions and the predicate $P(x, y)$ is defined as $y(x) = 1$ (in its dual version, i.e., ciphertext-policy ABE, $x$ represents a Boolean function, while $y$ represents a Boolean vector). Efficient ABE schemes exist for different predicates. For example, identity-based encryption (IBE) [45] can be obtained with the predicate $P(x, y) := (x = y)$, zero-inner product encryption (ZIPE) [34] can be obtained by setting $P(\boldsymbol{x}, \boldsymbol{y}) := (\boldsymbol{x}^\top \boldsymbol{y} = \mathbf{0})$, where $\boldsymbol{x}, \boldsymbol{y}$ belong to some vector space; other examples are span programs [35], hierarchical IBE [39], large universe ABE [44], non-monotonic access structures [41], or regular languages [7, 48].

Predicate encryption (PE) [17, 34] is a generalization of ABE where the main syntactical difference is that ciphertexts do not explicitly include $x$ and, therefore, it gives cause to considering the property of *attribute-hiding* where ciphertexts leak no information about the value $x$ they were associated to. On the other hand, *blind key-generation* attribute encryption (BKG-ABE) [43] allows one to hide the attributes associated to the secret keys from the master authority.

*Motivation.* BKG-ABE is a powerful tool that achieves both fine tuning of access control and privacy protection. It can be of use in ABE scenarios where the user's value $y$ associated to the requested key needs to remain private because, for example, it may leak sensitive information about the user. (It may, however, require zero-knowledge proofs for the authority to verify that the user is eligible for the key.) For example, several works evaluate the suitability of ABE for applications such as enforcing privacy of Electronic Medical Records (EMR) [6] in a system where healthcare organizations export EMRs to external storage locations. Other examples are Sieve [50] or Streamforce [25], systems that provide enforced access control for user data and stream data in untrusted clouds. These scenarios are a clear example where the privacy of parties requesting secret keys can be a concern. They would benefit from a BKG implementation of the underlying ABE.

Another application of BKG-ABE, proposed by Rial [43], is *oblivious transfer with fine-grained access control* (OT-AC) [18, 19]. It is a generalization of OT and $k$-out-of-$n$ OT, where a sender associates an access policy $f_i$ to every message $m_i$, for $i \in [n]$, so that a receiver who owns a certificate on some Boolean vector $y$ produced by a credential issuer obtains messages $m_i$ if and only if $f_i(y) = 1$. Furthermore, the sender gains no information about $y$.

We further propose to use BKG-ABE on the framework of secure computation to achieve more security guarantees. The strongest notions of security for secure computation capture no leakage of information against malicious parties arbitrarily deviating from the protocol. However, it may be the case that the correctness of the input is defined according to some context and this is out of the scope of the secure computation defined to be modular for generality. Accordingly, an extra mechanism for defining and guaranteeing the correctness of inputs will be performed to prevent malicious parties from lying on their input to the computation. Now, consider a scenario of secure computation between two parties, where Alice (with input $x$) owns a commitment $\sigma$ to Bob's input $y$ and wants to make sure that Bob uses such input in the execution of the secure computation of $f(x, y)$. This could be achieved by using an extra zero-knowledge proof. Other less modular solution would be to modify the circuit to be computed in such a way that it takes $\sigma$ as an extra input from Alice; it takes the

opening information $\delta$ as an extra input from Bob; and returns $f(x, y)$ if $\delta$ is a valid opening of $\sigma$ or $\perp$ otherwise. These approaches however can introduce a prohibitive overhead in the performance of the original protocol. BKG-ABE would provide a solution with reasonable efficiency. Especially if the commitment scheme used is a Pedersen-like commitment as in the first round of our BKG protocol (Sect. 3). For example, consider an ABE scheme for the predicate $P((i, \gamma), \boldsymbol{y}) := (\boldsymbol{y}_i = \gamma)$, for $\mathcal{X} := [n] \times \{0, 1\}$ and $\mathcal{Y} = \{0, 1\}^n$ (an encoding for this predicate can be found in [2]). Alice could create a garbled circuit for $f$, use ABE to encrypt every input label $\ell_b^{(i)}$ (corresponding to Bob's input) under value $x := (i, b)$ for all $i \in [n]$, $b \in \{0, 1\}$. Then, she could send everything to Bob; together with the answer of the second round of our BKG protocol, having interpreted the commitment $\sigma$ that she owns as the message from the first round. This way Bob will only be able to recover labels $\ell_{y_i}^{(i)}$, for $i \in [n]$ (and use them to evaluate the garbled circuit), as desired.

We note that this application is not specific to BKG-ABE and could be achieved via OT-AC. However, our particular protocol for BKG-ABE is especially suitable for it.

*Modular frameworks for ABE.* In 2014, Wee [49] and Attrapadung [7] independently proposed two generic and unifying frameworks for designing attribute-based encryption schemes for different predicates. Both works define a simple primitive called *encoding* and follow the dual system methodology by Lewko and Waters [38, 47] to construct a compiler that on input an encoding (for certain predicate $P$), produces a fully secure attribute-based encryption scheme for $P$. Wee defines the so-called *predicate encodings*, an information-theoretic primitive inspired by linear secret sharing, while Attrapadung introduces the notion of *pair encodings*, a similar primitive that admits both information-theoretic and computational security definitions. These frameworks remarkably simplify the design and study of ABE schemes: the designer can focus on the construction of the simpler encodings (for the desired predicate), which require weaker security properties that are more easily verifiable. In fact, the potential of this new frameworks is evidenced by the invention of new constructions and performance improvements on existing primitives. Although these frameworks were designed over composite over groups, they were both extended in [20] and [8] respectively to the prime-order setting (under the Matrix-DH assumption).

Pair encodings are the building block in which Attrapadung's framework relies. Informally, a pair encoding is a function from attributes to polynomials, which will be evaluated in the exponent of an algebraic group by the key-generation and encryption ABE algorithms. Such polynomials are restricted in their form (Sect. 2.2, Definition 2), which is crucial for the key-generation and encryption algorithms to be efficiently computable. In particular, they contain two types of variables: *lone* and *non-lone* variables. Lone variables always appear "alone" in degree-1 monomials (such monomials correspond to secret randomness generated during key-generation and encryption), whereas non-lone variables always appear multiplied by another non-lone variable in a degree-2 monomial (such monomials represent the combination of secret randomness with the randomness contained in public parameters).

### 1.1 Our contribution

We pursue the study of attribute-based encryption in the framework of pair-encodings and provide new results to broaden its scope.

*New definitions about pair encodings.* We define two new notions related to pair encodings: *BKG-compatible* encodings and *algebraic* encodings. The former notion establishes certain conditions on a pair encoding scheme that are sufficient for constructing very efficient blind key-generation protocols. Namely, we require that (i) the key-encoding polynomials that

depend on the master secret $\alpha$ be independent of the value $y$ associated to the key and (ii) the so-called *lone variables* can be linearly removed from all the key-encoding polynomials that depend on $y$. The first condition ensures that the computation of the key-generation that depends on the master secret key is independent of the value $y$ and thus, can be performed by the authority without obtaining any information about $y$. The second condition guarantees that the user who requested the key will be able to remove the blinding factor (introduced by himself) after the communication with the authority. We show that any pair encoding can be transformed with minimal overhead[1] into an equivalent BKG-compatible pair encoding. In fact, we provide a stronger result: every pair encoding can be modified into an equivalent one that *does not contain lone variables* apart from $\alpha$ (Sect. 3.2, Lemma 1). This result is of independent interest, as it could lead to improvements on our understanding of pair-encodings and their expressivity.

*Generic framework for blind key-generation ABE.* We equip the generic framework of attribute-based encryption from pair encodings with a protocol for blind key-generation (Sect. 3.3). Our protocol is two-round: (1) the user sends a key request that information-theoretically hides its value $y$; (2) the authority replies back with a piece of information from which the user can derive a secret key $\mathsf{sk}_y$. The protocol is secure against honest-but-curious adversaries and it is defined for any attribute-based encryption scheme that is based on a $BKG$ compatible pair encoding.

We then show that if the underlying encoding is algebraic, the zero-knowledge proof of knowledge sufficient for achieving active security can be implemented very efficiently with the $\Sigma$-protocol that we provide in Appendix C.

*Implementation.* We implement our protocol and compare its performance with the normal key-generation algorithm (of the ABE framework based on pair encodings). Our experiments show that both rounds of our protocol require a computational cost comparable to the one of the normal key-generation, making it suitable for real applications.

### 1.2 Overview of our protocol for blind key-generation

Normal key-generation for an ABE based on pair encodings requires several computations of the form

$$\widetilde{\mathsf{sk}}_\ell := h_{\mathsf{msk}}^{\phi^{(\ell)}} \cdot \prod_{i\in[\hat{m}]} \hat{h}_{\{i,0\}}^{\phi_i^{(\ell)}} \cdot \prod_{\substack{i\in[m]\\ j\in[n]}} h_{\{i,j\}}^{\phi_{\{i,j\}}^{(\ell)}},$$

where $\widetilde{\mathsf{sk}}_\ell$ is the $\ell$-th element of the secret key being generated; the exponents $\phi$ are the coefficients of the polynomials associated to the encoding of $y$, and the bases $h$ are elements of a dual system group (DSG) [23, 24]. A DSG is an abstraction that contains a bilinear group, i.e., a triple of abelian groups $(G, H, G_t)$ and a non-degenerate bilinear map $e : G \times H \to G_t$, these groups are usually implemented through prime-order groups, typically as vectors of prime-order group elements as in the DSG instantiation from $k$-lin, see Online Appendix. DSG are equipped with additional algorithms that can be used to simulate properties of composite-order groups that are they core of security proofs in the standard model, e.g., the existence of cyclic subgroups of unknown order.

Observe that $h_{\mathsf{msk}}$ is only known to the authority and that the other group elements from the above formula are freshly sampled on every key-generation. On the other hand, the exponents are only known to the user requesting the key. In order to perform blind

[1] The number of variables would be increased by a constant factor, independent of the predicate.

key-generation, it would be sufficient to have a two-party protocol for the functionality $\mathcal{F}_{A,U}((h_1,\ldots,h_n);(x_1,\ldots,x_n)) \to (\perp;\ \prod_i h_i^{x_i})$.

Below, we propose a simple protocol that implements such functionality over a group of prime-order $p$, based on the Diffie-Hellman assumption. (This is not the main construction proposed in this work, but serves as a warm-up example and could be adopted for simple ABE constructions.) Let $g$ be a generator of the group:

1. Party $A$ (Authority) masks every $h_i$ by sampling a random $r_i \xleftarrow{\$} \mathbb{Z}_p$, setting $a_i := (h_i^{r_i}, g^{r_i})$ and sending every $a_i$ to party $U$.
2. Party $U$ (User) now samples $t_i \xleftarrow{\$} \mathbb{Z}_p$ and computes $b_i := (h_i^{r_i})^{x_i} \cdot (g^{r_i})^{t_i}$ for every $i$, sending them back to $A$.
3. Finally, $A$ computes $c := \prod_i b_i^{1/r_i}$ and sends it to $U$, who computes the desired value as $c \cdot \prod_i g^{-t_i}$.

The above protocol can be directly used on ABE constructions implemented over a prime-order group. Such constructions are only known to be secure in the generic group model [1]. For ABE schemes proven secure in the standard model, all existing constructions to date use a dual system group as underlying group. Known instantiations of DSGs [8, 20, 36] are based on the $k$-Lin assumption (or the Matrix-DH assumption [26, 40]) and lead to groups that have prime exponent, but not prime order. (Such groups are typically implemented as vectors of prime-order groups.) In fact, such DS groups are not cyclic and thus, the multiplicative masking $h_i^{r_i}$ does not information-theoretically hide $h_i$. Consequently, the above generic protocol may not be directly applicable to constructions based on DSG. One could, however, apply the technique from the above protocol coordinate-wise on the DSG elements, that way maskings would perfectly hide the corresponding value. (This approach would also require both parties to provide additional zero-knowledge proofs that they are acting consistently in each coordinate, e.g., that the user $U$ has used the same exponent $x_i$ in all coordinates of the corresponding DSG element.) Our main construction is a blink-key generation protocol with better communication complexity (it does not need to operate coordinate-wise and it will not require the additional ZK proofs) and with only two rounds of interaction (note that the above protocol requires three rounds).

The main blind-key generation scheme proposed in this work exploits the specific structure of the dual system groups built from $k$-Lin, at the cost of not being generic for any DSG. In a nutshell, in the $k$-Lin implementation of dual system groups, the elements $h_{\{i,0\}}$ are sampled by taking $\boldsymbol{r}_i \xleftarrow{\$} \mathbb{Z}_p^k$ and computing $h_{\{i,0\}} := [\![B]\!]_2 \boldsymbol{r_i}$,[2] whereas $h_{\{i,j\}}$ is computed as $[\![W_j B]\!]_2 \boldsymbol{r}_i$, for all $i \in [m]$ and all $j \in [n]$. Here, $[\![B]\!]_2$ and $[\![W_j B]\!]_2$ are part of the public key, where $B$ and $W_j$ are matrices with coefficients over $\mathbb{Z}_p$.

For simplicity, we assume here that the encoding does not contain lone variables (i.e., the coefficients of the form $\phi_i^{(\ell)}$ are all zero). The idea of our protocol is that the user will compute $[\![\sum_j \phi_{\{i,j\}}^{(\ell)} W_j B]\!]_2$, for every $i$ and $\ell$, from the public key and will mask them with $[\![Z_i^{(\ell)} B]\!]_2$, for uniformly sampled matrices $Z_i^{(\ell)}$. The authority will instead sample the random vectors $\boldsymbol{r}_i$ and use the previous values to compute

$$\sum_i \left[\!\!\left[ \sum_j \phi_{\{i,j\}}^{(\ell)} W_j B + Z_i^{(\ell)} B \right]\!\!\right]_2 \boldsymbol{r}_i .$$

Vectors $\boldsymbol{r}_i$ will also need to be used by the authority in the computation of the rest of polynomials that are independent of $y$. Our requirements on BKG-compatible pair encodings

[2] Here, $[\![B]\!]_2$ denotes a matrix of group elements of a prime-order bilinear group (actually, of the second subgroup, given the subindex 2). The group elements correspond to the exponentiation of a selected generator to the coefficients of matrix $B$, defined over $\mathbb{Z}_p$. Such group elements are typically part of the public parameters.

guarantee that the extra terms $[\![Z_i^{(\ell)} B]\!] \boldsymbol{r}_i$, can be removed by the user, recovering a genuine secret key for $y$.

### 1.3 Related work

*Blind key-generation.* The notion of blind key-generation was first introduced by Green and Hohenberger [28] in the framework of identity-based encryption. The authors provide a protocol for blind key-generation of the Boneh-Boyen IBE [11] and show how to construct very efficient oblivious transfer (OT) [42] protocols from blind key-generation IBE schemes that satisfy some extra conditions. A subsequent work to [28] by Rial [43] extends the notion of blind key-generation to the framework of attribute-based encryption, and shows how to use it for building OT with fine-grained access control. The construction by Rial defines a blind key-generation algorithm for the CP-ABE of [16]. This ABE scheme has the property that secret keys are formed as a concatenation of pieces, each corresponding to one of the attributes included in the key. In a nutshell, Rial's idea for blind key-generation is that the authority will create a key for *all* attributes and the user will get only the parts corresponding to the attributes of their choice via a $k$-out-of-$n$ adaptive OT. Note that Rial's technique allows to upgrade an adaptive $k$-out-of-$n$ OT protocol to OT-AC. Observe that this approach is limited to CP-ABE schemes with the above property of separation between attributes in the key.

*Blind functional encryption.* In a very recent work, Canard et al. [21] considered the notion of blind key-generation in the context of functional encryption. They argue that blind-key generation can be achieved generically (but somewhat inefficiently) by using homomorphic encryption and zero-knowledge proofs and give a specific efficient construction for the inner-product functionality.

The main advantage of our work with respect to these related works is generality. Our protocol is designed over the recent and most advanced modular frameworks of ABE that provide a rich variety of predicates. In particular, it can be applied to all the encodings from the literature [3, 4, 7–9, 20].

## 2 Background

We consider a bilinear group generator $\mathcal{G}$ that, on input the security parameter $\lambda$, outputs an asymmetric bilinear group $(p, G_1, G_2, G_t, g_1, g_2, e : G_1 \times G_2 \to G_t)$, where $g_t := e(g_1, g_2)$ is a generator of $G_t$. For $a \in \mathbb{Z}_p$, we use $[\![a]\!]_i$ to denote the implicit representation $g_i^a$ of $a$ in $G_i$, for $i \in \{1, 2, t\}$, following [26]. For a matrix $A \in \mathbb{Z}_p^{m \times n}$ and a group element $g$, we abuse notation and write $g^A$ or $[\![A]\!]$ to represent the matrix formed by powering $g$ to the elements of $A$. We use $A_{i,j}$ to denote the element of $A$ that appears in the $i$-th row and the $j$-th column of $A$; the $i$-th element of a vector $\boldsymbol{v}$ is denoted as $v_i$. Given a matrix of group elements $A \in G^{m \times n}$ and a vector $\boldsymbol{v} \in \mathbb{Z}_p^n$, we denote by $A\boldsymbol{v}$ the vector of group elements obtained by multiplying each row of $A$ by the scalar vector $\boldsymbol{v}$ as a scalar multi-exponentiation, that is the $k$-th element of $A\boldsymbol{v}$ is defined as $\prod_{i=1}^{n} A_{k,i}^{v_i}$ We denote by $\mathbf{0}_n$ the zero vector of length $n$. We use the $\|$ symbol for concatenation of lists or vectors. For a vector $\boldsymbol{v}$ we denote its length by $|\boldsymbol{v}|$. Given a matrix $A \in \mathbb{Z}_p^{m \times n}$ and a set of indices $\Phi \subseteq [m]$, we define $A_\Phi$ as the matrix formed by the rows of $A$ whose index is in $\Phi$.

Given a predicate $P : \mathcal{X} \times \mathcal{Y} \to \{0, 1\}$, we write $(x, y) \in P$ to denote $P(x, y) = 1$. For a pair of integers $m, n \in \mathbb{N}$ we define $[m, n]$ as the range $\{m, \ldots, n\}$ and we write $[n]$ to represent

$[1, n]$. In case $m > n$, $[m, n]$ is defined as the empty set. For a finite set $S$, $x \xleftarrow{\$} S$ denotes that $x$ is uniformly sampled from $S$. Given a multi-variate polynomial $f(\boldsymbol{X}) \in \mathbb{Z}_p[\boldsymbol{X}]$, we denote by $f(\boldsymbol{x})$ the evaluation of polynomial $f$ on values $\boldsymbol{x}$. This notation is extended to vectors of polynomials $\boldsymbol{f}$ if every polynomial in the vector is defined with respect to the same variables. We sometimes group variables in classes as $f(\boldsymbol{X}, \boldsymbol{Y}) \in \mathbb{Z}_p[\boldsymbol{X}, \boldsymbol{Y}]$. When the variables associated to a polynomial are clear from the context we simply write $f$ instead of $f(\boldsymbol{X}, \boldsymbol{Y})$.

### 2.1 Attribute-based encryption

Attribute-based encryption (ABE) [46] is a form of of public-key encryption that supports fine-grained access control for encrypted data. An attribute-based encryption scheme for a predicate $P$ guarantees that decryption of a ciphertext $\mathsf{ct}_x$ with a secret key $\mathsf{sk}_y$ is allowed if, and only if, the value $x$ associated to the ciphertext $\mathsf{ct}$ and the value $y$ associated to the secret key $\mathsf{sk}$ verify the predicate $P$, i.e., $(x, y) \in P$.

**Definition 1** (Attribute-based encryption) Given a predicate $P : \mathcal{X} \times \mathcal{Y} \to \{0, 1\}$, an attribute-based encryption scheme is tuple of polynomial-time algorithms $\{Setup, Enc, KeyGen, Dec\}$ where:

- $Setup(1^\lambda, \mathcal{X}, \mathcal{Y}) \to (\mathsf{mpk}, \mathsf{msk})$, on input the security parameter $\lambda$ and value universes $\mathcal{X}, \mathcal{Y}$, outputs a master public key and a master secret key.
- $Enc(\mathsf{mpk}, m, x) \to \mathsf{ct}_x$, on input $\mathsf{mpk}$, a message and $x \in \mathcal{X}$, outputs $\mathsf{ct}_x$.
- $KeyGen(\mathsf{msk}, y) \to \mathsf{sk}_y$, on input $\mathsf{msk}$ and $y \in \mathcal{Y}$, outputs a secret key $\mathsf{sk}_y$.
- $Dec(\mathsf{mpk}, \mathsf{sk}_y, \mathsf{ct}_x, x) \to m/\perp$, on input $\mathsf{sk}_y$ and $\mathsf{ct}_x$, outputs a message $m$ if $P(x, y) = 1$ or $\perp$ otherwise.

Without loss of generality and even if we do not make it explicit, we assume $\mathsf{msk}$ always contains $\mathsf{mpk}$. In *blind key-generation* ABE there exists an extra two-party protocol executed between a user $U$ and the master authority $A$:

- $Blind\text{-}KG_{\mathrm{U,A}}((\mathsf{mpk}, y); \mathsf{msk}) \to (\mathsf{sk}_y; \perp)$, on input the master public key and a value $y$ from the user and on input the master secret key from the authority, the user gets a secret key for $y$, whereas the authority learns nothing.

### 2.2 Pair encodings

*Pair encodings* are the building block for constructing ABE on which Attrapadung [7–9] and Agrawal and Chase [3, 4] rely. Note that, in this work, we adopt the most modern definition of pair encodings, introduced in [4], where some of the structural constraints on the polynomials are explicit thanks to the separation between *lone* and *non-lone* variables.

**Definition 2** (Pair encoding) Let $P : \mathcal{X} \times \mathcal{Y} \to \{0, 1\}$ be a binary predicate over finite sets $\mathcal{X}$ and $\mathcal{Y}$. For a prime $p \in \mathbb{N}$ and $\Upsilon = (\bar{w}, w, \hat{w}, \bar{m}, m, \hat{m}, n) \in \mathbb{N}^7$, let

$$\boldsymbol{b} = (b_1, \ldots, b_n) \qquad \boldsymbol{s} = (s_1, \ldots, s_w) \qquad \boldsymbol{r} = (r_1, \ldots, r_m)$$
$$\hat{\boldsymbol{s}} = (\hat{s}_1, \ldots, \hat{s}_{\hat{w}}) \qquad \hat{\boldsymbol{r}} = (\alpha, \hat{r}_1, \ldots, \hat{r}_{\hat{m}})$$

be formal variables. A $\Upsilon$-*pair encoding* scheme (PES) for $P$ consists of three deterministic and efficiently computable algorithms:

- $EncCt(p, x)$: maps $x \in \mathcal{X}$ into a vector of $\bar{w}$ polynomials, $\boldsymbol{c}_x$, in $\mathbb{Z}_p[\boldsymbol{s}, \hat{\boldsymbol{s}}, \boldsymbol{b}]$.
- $EncKey(p, y)$: maps $y \in \mathcal{Y}$ into a vector of $\bar{m}$ polynomials, $\boldsymbol{k}_y$, in $\mathbb{Z}_p[\boldsymbol{r}, \hat{\boldsymbol{r}}, \boldsymbol{b}]$.
- $Pair(p, x, y)$: maps $x \in \mathcal{X}$ and $y \in \mathcal{Y}$ into a pair of matrices $E^{x,y}$, $F^{x,y}$ with coefficients in $\mathbb{Z}_p$ of dimensions $w \times \bar{m}$ and $\bar{w} \times m$ respectively.

We require that the following properties be satisfied (for all $p$ prime):

**reconstructability:** For every $(x, y) \in P$, $\boldsymbol{s}^\top E \boldsymbol{k} + \boldsymbol{c}^\top F \boldsymbol{r} = \alpha s_1$ holds symbolically, where $\boldsymbol{c} \leftarrow EncCt(p, x), \boldsymbol{k} \leftarrow EncKey(p, y)$, $(E, F) \leftarrow Pair(p, x, y)$.

**polynomial constraints:** For every $x \in \mathcal{X}$, the polynomials in $EncCt(p, x)$ only contain monomials of the form $\hat{s}_{i'}$, or $s_i b_j$, for $i' \in [\hat{w}]$, $i \in [w]$, $j \in [n]$. Furthermore, for every $y \in \mathcal{Y}$, the polynomials in $EncKey(p, y)$ only contain monomials of the form $\alpha$, $\hat{r}_{i'}$ or $r_i b_j$ for $i' \in [\hat{m}]$, $i \in [m]$, $j \in [n]$. Thus, $\hat{\boldsymbol{s}}, \hat{\boldsymbol{r}}$ are called *lone* variables, whereas $\boldsymbol{s}, \boldsymbol{r}$ are called *non-lone* variables.

**security (non-reconstructability):** For every $(x, y) \notin P$, and for every pair of matrices $E$ and $F$ with coefficients in $\mathbb{Z}_p$, $\boldsymbol{s}^\top E \boldsymbol{k} + \boldsymbol{c}^\top F \boldsymbol{r} \neq \alpha s_1$, where $\boldsymbol{c} \leftarrow EncCt(p, x)$ and $\boldsymbol{k} \leftarrow EncKey(p, y)$.

When it is clear from the context, we may omit $p$ from the input to the encoding algorithms.

***Remark 1*** There exist several security notions for pair encodings (information-theoretic and also computational) [7]. The one we adopted in Definition 2 was introduced by Agrawal and Chase [4]. Our blind KG protocol works for all pair encodings, independently of the security notion.

***Remark 2*** Other works [3, 4, 9] define pair encodings for a family of predicates $P_\kappa : \mathcal{X}_\kappa \times \mathcal{Y}_\kappa \to \{0, 1\}$, indexed by $\kappa = (p, par)$, where $par$ specifies some parameters (including the security parameter $\lambda$) and consider one extra algorithm, $Param$, that on input $par$, outputs $n \in \mathbb{N}$ specifying the number of common variables. In our definition, for the sake of simplicity, we omit such algorithm. However, one could think of the elements in $\Upsilon$ as polynomials[3] (over $\mathbb{Z}$) in the security parameter $\lambda$. And one could define $Param$ as the algorithm that evaluates the polynomial $n(\lambda)$ on the given security parameter and outputs the corresponding integer.

### 2.3 Attribute-based encryption from pair encodings

The compiler from pair encodings to attribute-based encryption is defined over bilinear groups (implemented as dual system groups) and, roughly, works as follows. To encrypt a message $M \in G_t$, this is multiplied by a blinding factor $g_t^{\alpha s}$, where $s$ is fresh randomness and $g_t^\alpha$ is part of the public key, being $g_t := e(g_1, g_2)$. Ciphertexts contain additional group elements that can be combined with the elements of a valid secret key in order to recover the blinding factor and, consequently, the message. In short, and following the notation from Definition 2, the compiler could be summarized as:

$$\mathsf{mpk} = \left\{g_t^\alpha,\ g_1^{\boldsymbol{b}},\ g_2^{\boldsymbol{b}}\right\} \qquad \mathsf{ct}_x = \left\{g_1^{\boldsymbol{s}},\ g_1^{\boldsymbol{c}_x(\boldsymbol{s}, \hat{\boldsymbol{s}}, \boldsymbol{b})},\ M \cdot g_t^{\alpha s_1}\right\}$$

[3] Actually, we should consider $w$, $\hat{w}$ and $\bar{w}$ as functions that map $x$ into polynomials in $\lambda$; and $m$, $\hat{m}$ and $\bar{m}$ as functions that map $y$ into polynomials in $\lambda$.

$$\mathsf{msk} = \{g_2^{\alpha}\} \qquad \mathsf{sk}_y = \left\{ g_2^{\boldsymbol{r}},\ g_2^{\boldsymbol{k}_y(\boldsymbol{r},(\alpha,\hat{\boldsymbol{r}}),\boldsymbol{b})} \right\}$$

For the purpose of this paper, a formal definition of the setup and the key-generation algorithm is sufficient. For the sake of simplicity, we omit the description of the other algorithms and include a complete definition of the compiler in Online Appendix. (We refer to [4, Sect. 5.4] for further details.) Observe that in the following definition we have already adopted the DSG implementation based on $k$-Lin (see Online Appendix and also [20]), because our blind key-generation protocol is defined for this instantiation.

**Definition 3** (Attribute-based encryption from pair encodings) Given a $\Upsilon$-pair encoding $(EncCt, EncKey, Pair)$, with $\Upsilon = (\bar{w}, w, \hat{w}, \bar{m}, m, \hat{m}, n)$, for predicate $P : \mathcal{X} \times \mathcal{Y} \to \{0, 1\}$, we can construct an attribute-based encryption scheme whose setup and key-generation algorithms are as follows:

- $Setup(1^{\lambda}, \mathcal{X}, \mathcal{Y})$: sample a bilinear group $(p, G_1, G_2, G_t, g_1, g_2, \hat{e}) \leftarrow \mathcal{G}(1^{\lambda})$ and sample matrices from the $k$-Lin distribution[4] $(A, \boldsymbol{a}^{\perp}) \leftarrow \mathcal{D}_k$, $(B, \boldsymbol{b}^{\perp}) \leftarrow \mathcal{D}_k$. Sample matrices $W_1, \ldots, W_n$ uniformly over $\mathbb{Z}_p$ of dimension $(k+1)\times(k+1)$. Sample $\boldsymbol{\alpha} \xleftarrow{\$} \mathbb{Z}_p^{k+1}$, let $h_{\mathsf{msk}} := [\![\boldsymbol{\alpha}]\!]_2$ and set

$$\mathsf{mpk} := \begin{pmatrix} (p, G_1, G_2, G_t, g_1, g_2, \hat{e}) & [\![W_1^{\top} A]\!]_1, \ldots, [\![W_n^{\top} A]\!]_1 \\ [\![A]\!]_1,\ [\![B]\!]_2,\ [\![A^{\top}\boldsymbol{\alpha}]\!]_t & [\![W_1 B]\!]_2, \ldots, [\![W_n B]\!]_2 \end{pmatrix} \quad \mathsf{msk} := h_{\mathsf{msk}}\ .$$

  Output $(\mathsf{mpk}, \mathsf{msk})$.
- $KeyGen(\mathsf{msk}, y)$: run $EncKey(p, y)$ to obtain polynomials $\boldsymbol{k}_y(\boldsymbol{r}, \hat{\boldsymbol{r}}, \boldsymbol{b})$. For every $\ell \in [k]$, let the $\ell$-th polynomial in $\boldsymbol{k}_y$ be

$$\phi^{(\ell)}\alpha + \sum_{i\in[\hat{m}]} \phi_i^{(\ell)} \hat{r}_i + \sum_{i\in[m]} \sum_{j\in[n]} \phi_{\{i,j\}}^{(\ell)} r_i b_j$$

  for some coefficients $\phi^{(\ell)}$, $\phi_i^{(\ell)}$ and $\phi_{\{i,j\}}^{(\ell)}$ in $\mathbb{Z}_p$. Now do:

$$\forall i \in [\hat{m}], \hat{\boldsymbol{r}}_{\boldsymbol{i}} \xleftarrow{\$} \mathbb{Z}_p^k \text{ set } \hat{h}_{\{i,0\}} := [\![B\hat{\boldsymbol{r}}_{\boldsymbol{i}}]\!]_2 \text{ and } \forall j \in [n],\ \hat{h}_{\{i,j\}} := [\![W_j B\hat{\boldsymbol{r}}_{\boldsymbol{i}}]\!]_2$$
$$\forall i \in [m], \boldsymbol{r}_{\boldsymbol{i}} \xleftarrow{\$} \mathbb{Z}_p^k \text{ set } h_{\{i,0\}} := [\![B\boldsymbol{r}_{\boldsymbol{i}}]\!]_2 \text{ and } \forall j \in [n],\ h_{\{i,j\}} := [\![W_j B\boldsymbol{r}_{\boldsymbol{i}}]\!]_2.$$

  Let the secret key $\mathsf{sk}_y$ be $\mathsf{sk}_y := \left(\mathsf{sk}_1, \ldots, \mathsf{sk}_m, \widetilde{\mathsf{sk}}_1, \ldots, \widetilde{\mathsf{sk}}_k\right)$ where for every $i \in [m]$, $\mathsf{sk}_i := h_{\{i,0\}}$; and for every $\ell \in [k]$, $\widetilde{\mathsf{sk}}_{\ell}$ is computed as

$$\widetilde{\mathsf{sk}}_{\ell} := h_{\mathsf{msk}}^{\phi^{(\ell)}} \cdot \prod_{i\in[\hat{m}]} \hat{h}_{\{i,0\}}^{\phi_i^{(\ell)}} \cdot \prod_{i\in[m]} \prod_{j\in[n]} h_{\{i,j\}}^{\phi_{\{i,j\}}^{(\ell)}}.$$

  Output $\mathsf{sk}_y$.

## 3 Blind key-generation ABE

In this section we first introduce a formal security definition for blind key-generation ABE (Sect. 3.1), we then present our BKG scheme (Sect. 3.2), which is compatible with the compiler of ABE based on pair encodings [4, 8].

[4] See Definition 1 in Online Appendix for more details about the $k$-lin distribution $\mathcal{D}_k$, which produces a matrix $A$ of dimensions $(k+1) \times k$ and a vector $\boldsymbol{a}^{\perp}$ of length $k+1$, which is orthogonal to every column of $A$.

### 3.1 Security definition

The security notion of blind key-generation ABE is formally captured by the following ideal functionality.

**Definition 4** (Ideal functionality for blind key-generation) The ideal functionality $\mathcal{F}$ for blind key-generation is defined as follows.

1. On input a pair $(\mathsf{mpk}, y)$ from the user, $\mathcal{F}$ sends the master public key $\mathsf{mpk}$ to the authority and waits for a response.
2. If the authority allows the computation, it replies back to $\mathcal{F}$ with a secret key $\mathsf{msk}$ and some randomness $r$.
3. Now, $\mathcal{F}$ checks if $(\mathsf{mpk}, \mathsf{msk}) \in \mathcal{K}$, i.e., if the secret key is valid with respect to the received $\mathsf{mpk}$ from the user. If so, it executes $\mathsf{sk}_y \leftarrow KeyGen(\mathsf{msk}, y; r)$ and sends $\mathsf{sk}_y$ to the user; otherwise it sends $\perp$ to the user.

Here, we denote by $(\mathsf{mpk}, \mathsf{msk}) \in \mathcal{K}$ the fact that there exists some randomness $s$ such that $(\mathsf{mpk}, \mathsf{msk}) \leftarrow Setup(1^\lambda, \mathcal{X}, \mathcal{Y}; s)$.

***Remark 3*** Observe that deciding whether $(\mathsf{mpk}, \mathsf{msk}) \in \mathcal{K}$ can be done efficiently for the ABE scheme from Definition 3. We will assume, without loss of generality, that the master secret key includes the master public key. That way, for every valid $\mathsf{msk}$ there exists exactly one $\mathsf{mpk}$ such that $(\mathsf{mpk}, \mathsf{msk}) \in \mathcal{K}$.

Security of blind key-generation ABE, defined with respect to the above ideal functionality, is schematized in Fig. 1. In words, let A and U be two arbitrary stateful algorithms. Let $Real_{\mathrm{U,A}}(\mathsf{mpk}, y, \mathsf{msk})$ be the following distribution. Algorithm U takes as input $(\mathsf{mpk}, y)$, whereas A takes as input $\mathsf{msk}$; both U and A execute the $Blind\text{-}KG$ protocol on their inputs, but may deviate arbitrarily from the protocol; let $\mathsf{view}_\mathrm{U}$ and $\mathsf{view}_\mathrm{A}$ be their respective final view. Return $(\mathsf{view}_\mathrm{U}, \mathsf{view}_\mathrm{A})$. Let $Ideal_{\mathrm{U,A}}(\mathsf{mpk}, y, \mathsf{msk})$ be the distribution where algorithm U takes as input $(\mathsf{mpk}, y)$ and produces a (possibly modified) pair $(\mathsf{mpk}', y')$ that is sent to the ideal functionality $\mathcal{F}$ (Definition 4); analogously, A takes as input $\mathsf{msk}$, and when it receives $\mathsf{mpk}$ from $\mathcal{F}$, if it decides to allow the computation, it produces a (possibly modified) key $\mathsf{msk}'$ and randomness $r$, which sends to $\mathcal{F}$; otherwise, it sends $\perp$ to $\mathcal{F}$. Let $\mathsf{view}_\mathrm{U}$ and $\mathsf{view}_\mathrm{A}$ be their respective final views. Return $(\mathsf{view}_\mathrm{U}, \mathsf{view}_\mathrm{A})$.

**Definition 5** (Security from malicious users) Protocol $Blind\text{-}KG$ is said to be secure against malicious users if and only if, for every PPT algorithm $\mathcal{U}$ there exists a PPT simulator $\mathcal{S}$ such that for all $(\mathsf{mpk}, y, \mathsf{msk})$,

$$Real_{\mathcal{U},\mathrm{A}_1}(\mathsf{mpk}, y, \mathsf{msk}) \approx_c Ideal_{\mathcal{S},\mathrm{A}_2}(\mathsf{mpk}, y, \mathsf{msk})$$

where $\mathrm{A}_1$ is an algorithm that honestly follows the $Blind\text{-}KG$ protocol (as the authority) and $\mathrm{A}_2$ is an algorithm that passes its input to $\mathcal{F}$ without modification.

***Remark 4*** The above definition expresses the standard simulation-based security which guarantees that all existing attacks in the real world are in fact inherent: they can also be mounted in the ideal world, where parties (authority and user) interact through an incorruptible and trusted third party.

**Definition 6** (Security from malicious authorities) Protocol $Blind\text{-}KG$ is said to be secure against malicious authorities if and only if, for every PPT algorithm $\mathcal{A}$ there exists a PPT simulator $\mathcal{S}$ such that for all $(\mathsf{mpk}, y, \mathsf{msk})$,

$$Real_{\mathrm{U}_1,\mathcal{A}}(\mathsf{mpk}, y, \mathsf{msk}) \approx_c Ideal_{\mathrm{U}_2,\mathcal{S}}(\mathsf{mpk}, y, \mathsf{msk})$$

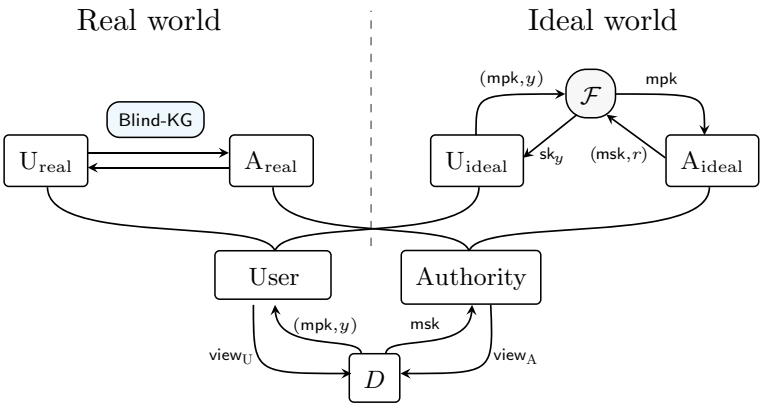


**Fig. 1** Protocol *Blind-KG* is secure if algorithm $D$, who can control mpk, $y$ and msk, cannot distinguish between the real world and the ideal world based on the views ($\mathsf{view}_U$, $\mathsf{view}_A$). For security against *malicious users* (Definition 5), algorithm $U_{real}$ is arbitrary, $U_{ideal}$ is a simulator, and $A_{real}$, $A_{ideal}$ are honest algorithms. For security against *malicious authorities* (Definition 6), $A_{real}$ is arbitrary, $A_{ideal}$ is a simulator, and $U_{real}$, $U_{ideal}$ are honest

where $U_1$ is an algorithm that honestly follows the *Blind-KG* protocol (as a user) and $U_2$ is an algorithm that does not modify its input and passes it to $\mathcal{F}$.

### 3.2 Blind-KG compatible pair encodings

We consider a restricted class of pair encodings, that we coin *blind key-generation compatible* (or *BKG-compatible*, for short) pair encodings. This class is necessary for our blind key-generation algorithm described in the next section.

**Definition 7** (BKG-compatible pair encoding) We say a $\Upsilon$-pair encoding, ($EncCt$, $EncKey$, $Pair$), is *BKG-compatible* if and only if there exist two deterministic and efficiently computable algorithms,

- $EncKey1(p, y)$: maps $(p, y) \in \mathbb{N} \times \mathcal{Y}$ into a vector of polynomials, in $\mathbb{Z}_p[\boldsymbol{r}, \hat{\boldsymbol{r}}, \boldsymbol{b}]$.
- $EncKey2(p)$: maps $p \in \mathbb{N}$ into a vector of polynomials, in $\mathbb{Z}_p[\boldsymbol{r}, \hat{\boldsymbol{r}}, \boldsymbol{b}]$.

satisfying the following conditions, for every $p \in \mathbb{N}$:

(i) $EncKey$ can be expressed as $EncKey(p, y) := EncKey1(p, y) \parallel EncKey2(p)$.
(ii) For every $y \in \mathcal{Y}$, variable $\alpha$ does not appear in $EncKey1(p, y)$.
(iii) For every $i' \in [\hat{m}]$, either

- one of the polynomials in $EncKey2(p)$ is exactly $\hat{r}_{i'}$, or
- for every $y \in \mathcal{Y}$, variable $\hat{r}_{i'}$ does not appear in $EncKey1(p, y)$.

We now show that we can assume, without loss of generality, that a pair encoding is *BKG-compatible*. More precisely, that any pair encoding can be transformed into an equivalent BKG-compatible pair encoding with minimal impact on its size and number of variables. In fact, in Lemma 2, we prove a stronger result: *any pair encoding admits an equivalent encoding that does not have lone variables and such that variable $\alpha$ appears only in one polynomial (which is independent of $y$).*

Our first lemma shows that, without loss of generality, we can consider pair encodings that *do not contain* lone variables (except $\alpha$).

**Lemma 1** *Let* $\Upsilon = (\bar{w}, w, \hat{w}, \bar{m}, m, \hat{m}, n)$ *and let* $(EncCt, EncKey, Pair)$ *be a* $\Upsilon$*-pair encoding for* $P$*. There exists a* $\Upsilon'$*-pair encoding for the same* $P$*, where* $\Upsilon' = (\bar{w},\ \max(w, \hat{w}),\ 0,\ \bar{m},\ \max(m, \hat{m}),\ 0,\ n{+}2)$.

We prove the lemma in Appendix A.

***Remark 5*** Our Lemma 1 suggests that w.l.o.g. we can consider pair encodings that do not contain lone variables. However, in the most recent definitions of pair encodings, all non-lone variables are given "*in the clear*" as part of the key. What our lemma really implies is that, after our transformation, giving out all variables in the clear does not compromise security. Nonetheless, not all variables are used for correctness and so, not all need to be always given. Actually, existing encodings for *large universe ABE* or *regular languages* involve an exponential number of lone variables and giving all of them in the clear after our transformation would be impractical (and unnecessary). When such encodings are used for blind key-generation, the authority must select a subset of variables to be given in the clear. This imposes a bound on the size of the value $y$ that users can request, leaking some information on size of $y$. Observe that this leakage seems inherent and it is usually admitted in cryptography (e.g. encryption schemes hide the content of messages, but not necessarily their length).

We are ready to establish the result on which our blind key-generation protocol from the next section relies.

**Lemma 2** *(BKG-compatibility) Let* $\Upsilon = (\bar{w}, w, \hat{w}, \bar{m}, m, \hat{m}, n)$ *and let* $(EncCt, EncKey, Pair)$ *be a* $\Upsilon$*-pair encoding for predicate* $P$*. There exists a* $\Upsilon'$*-BKG-compatible pair encoding for* $P$*, where*

$$\Upsilon' = (\bar{w}{+}1,\ \max(w{+}1, \hat{w}),\ 0,\ \bar{m}{+}1, \max(m{+}1, \hat{m}),\ 0,\ n{+}4).$$

See Appendix A for a proof.

### 3.3 Blind key-generation ABE from pair encodings

Figure 2 summarizes our blind key-generation protocol, compatible with the ABE compiler from pair encodings [3, 4, 8] (see Definition 2), where for $\Upsilon = (\bar{w}, w, \hat{w}, \bar{m}, m, \hat{m}, n)$, $(EncCt, EncKey1, EncKey2, Pair)$ is a *BKG-compatible* $\Upsilon$-pair encoding scheme.

**Theorem 1** *(Correctness) If both parties honestly execute the* $Blind\text{-}KG$ *protocol described in Fig. 2, on inputs* $(\mathsf{mpk}, y)$ *for the user and* $\mathsf{msk}$ *for the authority, such that* $(\mathsf{mpk}, \mathsf{msk}) \in \mathcal{K}$*, then the final output of the user is identically distributed to the output of* $KeyGen(\mathsf{msk}, y)$ *(see Definition 3), over the coins of the protocol and the* $KeyGen$ *algorithm.*

Our following theorems establish that the protocol is secure against *honest-but-curious* adversaries. In the following, let $\mathrm{U}_1$ be an algorithm that honestly follows the $Blind\text{-}KG$ protocol (as a user) and let $\mathrm{U}_2$ be an algorithm that does not modify its input and passes it to the ideal functionality (as the user). Analogously, let $\mathrm{A}_1$ be an algorithm that honestly follows the $Blind\text{-}KG$ protocol (as the authority) and let $\mathrm{A}_2$ be an algorithm that does not modify its input and passes it (together with true randomness) to the ideal functionality (as the authority).

**Theorem 2** *(Security against honest-but-curious authorities) There exists a simulator* $\mathcal{S}$ *such that for all* $(\mathsf{mpk}, y, \mathsf{msk})$ *and every PPT distinguisher,*

$$Real_{\mathrm{U}_1,\mathrm{A}_1}(\mathsf{mpk}, y, \mathsf{msk}) \equiv Ideal_{\mathrm{U}_2,\mathcal{S}}(\mathsf{mpk}, y, \mathsf{msk}).$$

- $\mathsf{Blind\text{-}KG}_{\mathrm{U,A}}((\mathsf{mpk}, y); \mathsf{msk})$:

Round 1 (User)

Run $\boldsymbol{k}_y^{(1)} \leftarrow \mathsf{EncKey1}(p, y)$ and let the $\ell$-th polynomial in $\boldsymbol{k}_y^{(1)}$ be

$$\textstyle\sum_{i\in[\hat{m}]} \phi_i^{(\ell)} \hat{r}_i + \sum_{i\in[m]} \sum_{j\in[n]} \phi_{\{i,j\}}^{(\ell)} r_i b_j$$

for some coefficients $\phi_i^{(\ell)}$, $\phi_{\{i,j\}}^{(\ell)}$ in $\mathbb{Z}_p$. Let $\bar{m}_1 := |\boldsymbol{k}_y^{(1)}|$.
For every $\ell \in [\bar{m}_1]$, and for every $i \in [m]$, sample a random matrix $Z_i^{(\ell)} \xleftarrow{\$} \mathbb{Z}_p^{(k+1)\times(k+1)}$ and compute $M_i^{(\ell)} \in G_2^{(k+1)\times k}$ as

$$M_i^{(\ell)} := \left[\!\left[ \phi_{i,1}^{(\ell)} W_1 B + \cdots + \phi_{i,n}^{(\ell)} W_n B + Z_i^{(\ell)} B \right]\!\right]_2 .$$

Send $\mathsf{mpk}$ and $\{M_i^{(\ell)}\}_{i\in[m],\ell\in[\bar{m}_1]}$ to the authority.

Round 2 (Authority)

If $(\mathsf{mpk}, \mathsf{msk}) \notin \mathcal{K}$, return $\perp$. Otherwise, run $\boldsymbol{k}^{(2)} \leftarrow \mathsf{EncKey2}(p)$. Let the $\ell$-th polynomial in $\boldsymbol{k}^{(2)}$ be

$$\varphi^{(\ell)}\alpha + \textstyle\sum_{i\in[\hat{m}]} \varphi_i^{(\ell)} \hat{r}_i + \sum_{i\in[m]} \sum_{j\in[n]} \varphi_{\{i,j\}}^{(\ell)} r_i b_j$$

for some coefficients $\varphi^{(\ell)}$, $\varphi_i^{(\ell)}$, $\varphi_{\{i,j\}}^{(\ell)}$ in $\mathbb{Z}_p$. Let $\bar{m}_2 := |\boldsymbol{k}^{(2)}|$.
For every $i \in [m]$ and $i' \in [\hat{m}]$, sample $\boldsymbol{r_i} \xleftarrow{\$} \mathbb{Z}_p^k$ and $\hat{\boldsymbol{r}}_{\boldsymbol{i}} \xleftarrow{\$} \mathbb{Z}_p^k$.
Set $\rho_i := [\![B\boldsymbol{r_i}]\!]_2$ and for $\ell \in [\bar{m}_2]$, compute $\sigma_\ell \in G_2^{k+1}$ as

$$\sigma_\ell := h_{\mathsf{msk}}^{\varphi^{(\ell)}} \cdot \left[\!\left[ \textstyle\sum_{i\in[\hat{m}]} \varphi_i^{(\ell)} B\hat{\boldsymbol{r}}_{\boldsymbol{i}} \;+\; \sum_{i\in[m]} \left( \sum_{j\in[n]} \varphi_{\{i,j\}}^{(\ell)} W_i \right) B\boldsymbol{r_i} \right]\!\right]_2 .$$

Finally, for every $\ell \in [\bar{m}_1]$, compute $\tau_\ell \in G_2^{k+1}$ as

$$\tau_\ell := M_1^{(\ell)} \boldsymbol{r_1} + \cdots + M_m^{(\ell)} \boldsymbol{r_m} .$$

Send $\{\rho_i\}_{i\in[m]}$, $\{\sigma_\ell\}_{\ell\in[\bar{m}_2]}$ and $\{\tau_\ell\}_{\ell\in[\bar{m}_1]}$ back to the user.

Finalizing the key (User)

Complete the secret key $\mathsf{sk}_y := (\mathsf{sk}_1, \ldots, \mathsf{sk}_m, \tilde{\mathsf{sk}}_1, \ldots, \tilde{\mathsf{sk}}_{\bar{m}})$ by setting $\mathsf{sk}_i := \rho_i$ for every $i \in [m]$, $\tilde{\mathsf{sk}}_{\bar{m}_1+\ell} := \sigma_\ell$ for every $\ell \in [\bar{m}_2]$ (note that $\bar{m} = \bar{m}_1 + \bar{m}_2$) and for every $\ell \in [\bar{m}]$,

$$\tilde{\mathsf{sk}}_\ell := \tau_\ell - \textstyle\sum_{i\in[m]} Z_i \rho_i + \sum_{i\in[\hat{m}]} \phi_i^\ell \sigma_{\pi(i)} . \qquad (\tilde{\mathsf{sk}}_\ell \in G_2^{k+1})$$

where $\pi(i)$ is the smallest integer $j$ such that $k_j^{(2)} = \hat{r}_i$ or zero if such an integer does not exist (and we define $\sigma_0 = [\![\mathbf{0}_{k+1}]\!]_2$).

**Fig. 2** Description of our *Blind-KG* for ABE for any *BKG*-compatible pair encoding in the framework of dual system groups instantiated from the $k$-Lin assumption

Intuitively, the factor $Z_i^{(\ell)}B$, added by the user, information-theoretically hides the coefficients $\phi_i^{(\ell)}, \phi_{\{i,j\}}^{(\ell)}$, which contain the information about $y$. That is why the authority cannot learn any information about $y$ from the first round of the protocol and it can be easily simulated. (Note that $Z_i^{(\ell)}B$ does not distribute uniformly over $\mathbb{Z}_p^{k+1}$, but it does distribute uniformly over the subspace of $\mathbb{Z}_p^{k+1}$ on which the information about $y$ has been encoded.)

**Theorem 3** *(Security against honest-but-curious users) There exists a simulator $\mathcal{S}$ such that for all $(\mathsf{mpk}, y, \mathsf{msk})$ and every PPT distinguisher,*

$$Real_{\mathrm{U}_1,\mathrm{A}_1}(\mathsf{mpk}, y, \mathsf{msk}) \equiv Ideal_{\mathcal{S},\mathrm{A}_2}(\mathsf{mpk}, y, \mathsf{msk}).$$

We refer to Appendix A for a proof of all three theorems.

## 4 Achieving active security

The standard way of achieving active security on a passively secure protocol is to add *zero-knowledge proofs* of honest behavior (see the Supplementary Material). That way, only honest users will be able to produce valid proofs and honest-but-curious security applies.

For security against malicious users, we could add the following zero-knowledge proof of knowledge

$$\mathrm{PoK}\left\{\begin{array}{ll}(y, Z_i^{\ell}) & EncKey1(y) \to \left\{\sum_{i\in[\hat{m}]}\phi_i^{(\ell)}\hat{r}_i + \sum_{\substack{i\in[m]\\ j\in[n]}}\phi_{\{i,j\}}^{(\ell)} r_i b_j\right\}_{\ell\in[\bar{m}_1]} \wedge \\ \forall i \in [m]: & \\ \ell \in [\bar{m}] & \forall\, \ell \in [\bar{m}_1],\ M_i^{(\ell)} = [\![\phi_{i,1}^{(\ell)} W_1 B + \cdots + \phi_{i,n}^{(\ell)} W_n B + Z_i^{(\ell)} B]\!]_2\end{array}\right\} \quad (1)$$

to the message computed by the user in Round 1 of the protocol from Fig. 2. However, although zero-knowledge proofs of knowledge [14, 30] can be constructed for any NP statement [32], using generic methods to implement the above relation may be inefficient, given that $EncKey1$ can be an arbitrarily complex (still efficiently computable) circuit.

Instead, we introduce the notion of *invertible pair encoding* and use it to significantly simplify the above NP relation.

### 4.1 Invertible pair encodings

For every $p \in \mathbb{N}$, the function $EncKey1(p, \cdot)$ of any $BKG$-compatible pair encoding can be considered *injective*. Note that if $EncKey1(p, y_1) = EncKey1(p, y_2)$ for two different $y_1, y_2$, then, correctness and security of the encoding imply that $\forall x,\ P(x, y_1) \Leftrightarrow P(x, y_2)$ and thus, $y_1$ and $y_2$ can be considered *"equivalent"* under $P$. Consequently, we can consider the inverse function that on input a list, $\boldsymbol{k}$, of polynomials over $\mathbb{Z}_p[\boldsymbol{r}, \hat{\boldsymbol{r}}, \boldsymbol{b}]$, produces a value $y$ such that $\boldsymbol{k} = EncKey1(p, y)$.

**Definition 8** (Invertible pair encoding) We say a $(EncCt, EncKey, Pair)$ encoding is *invertible*[5] if there exists an efficiently computable decoding algorithm $DecKey$ that $\forall p \in \mathbb{N}$, on input a list in $\mathbb{Z}_p[\boldsymbol{r}, \hat{\boldsymbol{r}}, \boldsymbol{b}]$, outputs $y \in \mathcal{Y}$ such that,

$$\forall\, \boldsymbol{k} : \exists y \in \mathcal{Y} \text{ s.t. } EncKey(p, y) = \boldsymbol{k}, \text{ it holds } EncKey(p, DecKey(\boldsymbol{k})) = \boldsymbol{k}\ .$$

[5] For our purpose, we focus on the inversion of the key encoding. For other applications, this definition may be extended to the ciphertext encoding.

We leave as an open problem to (dis)prove that any pair encoding admits an equivalent invertible pair encoding. To the best of our knowledge, all pair encodings from the literature are invertible.

We require pair encodings be invertible so that it is enough for the user to prove that it knows the coefficients that are committed inside the messages $M_i^{(\ell)}$. This way, the simulator (for the malicious user) can extract the coefficients from the proof of knowledge and, because the encoding is invertible, find a value $y \in \mathcal{Y}$ such that $EncKey(p, y)$ equals the polynomials associated to them. However, note that *invertibility* may not be enough: *what happens if there exists no y satisfying the requirements?* Invertibility only guarantees that it is possible to efficiently find $y$ in case it exists. In order to solve this problem, we observe that all pair encodings from the literature are designed in a similar way. Namely, polynomials are built by adding monomials whose coefficient depends on the value $y$, or whose variables are chosen based on the value. We abstract this idea in the notion of *algebraic pair encoding*.

**Definition 9** (Algebraic pair encoding) Let $\Upsilon = (\bar{w}, w, \hat{w}, \bar{m}, m, \hat{m}, n)$. We say a pair encoding $\Upsilon$-pair encoding $(EncCt, EncKey, Pair)$ is *algebraic* if algorithm $EncKey$ defines polynomials as sums of monomials of the following three types:

- *public monomials:* their coefficient and variables are independent of $y$,
- *secret-coefficient monomials:* monomials of the form $\mu_y \hat{r}_{i'}$ or $\mu_y r_i b_j$ for some, $i' \in [\hat{m}]$ or $i \in [m]$, $j \in [n]$ and some $\mu_y \in \mathbb{Z}_p$ (that depends on $y$).
- *secret-variable monomials:* monomials of the form $\hat{r}_{\hat{\eta}_y}$ or $r_{\eta_y} b_j$ or $r_i b_{\nu_y}$ for some $j \in [n]$, $i \in [m]$ and $\hat{\eta}_y \in [\hat{m}]$, $\eta_y \in [m]$ or $\nu_y \in [n]$ (dependent on $y$).

Furthermore, we require an algebraic pair encoding be *surjective*, that is, for all possible values $\boldsymbol{\mu} \in \mathbb{Z}_p^*$ of the coefficients of secret-coefficient monomials, there exists some $y \in \mathcal{Y}$ such that $EncKey(p, y)$ has secret-coefficients matching $\boldsymbol{\mu}$.

Our Example 1 (in Appendix B) illustrates the three types of monomial from the above definition.

***Remark 6*** Note that the structure of an algebraic pair encoding is fixed and only the secret coefficients or the secret variables can depend on value $y \in \mathcal{Y}$. To the best of our knowledge, all existing pair encodings are algebraic.

Given an algebraic pair encoding, the zero-knowledge proof of knowledge from (1) can be done very efficiently. This is because if the pair encoding is algebraic, there exist disjoint sets $\Pi, \Phi \subset \mathbb{N} \times \mathbb{N} \times \mathbb{N}$ of *public coefficients* and *binary coefficients* respectively, there exists a function $f : \Pi \to \mathbb{Z}_p$ and there exist *clusters* $\Lambda = \{(\Lambda_s, g_s, \gamma_s)\}_{s \in \Theta}$, where for every $s \in \Theta$, $\Lambda_s \subset \mathbb{N} \times \mathbb{N} \times \mathbb{N}$, $g_s : \Lambda_s \to \mathbb{Z}_p$ and $\gamma_s \in \mathbb{Z}_p$, such that, for any value of the coefficients $\phi_{\{i,j\}}^{(\ell)} \in \mathbb{Z}_p$ (for $i \in [m]$, $j \in [n]$ and $\ell \in [\bar{m}_1]$) satisfying

$$\forall (i, j, \ell) \in \Pi,\ \phi_{\{i,j\}}^{(\ell)} = f(i, j, \ell) \ \wedge\ \forall (i, j, \ell) \in \Phi,\ \phi_{\{i,j\}}^{(\ell)} \in \{0, 1\}$$
$$\wedge\ \forall s \in \Theta,\ \gamma_s = \textstyle\sum_{(i,j,\ell) \in \Lambda_s} g_s(i, j, \ell) \phi_{\{i,j\}}^{(\ell)}$$

there exists $y \in \mathcal{Y}$ such that

$$EncKey1(y) \to \left\{ \textstyle\sum_{i \in [\hat{m}]} \phi_i^{(\ell)} \hat{r}_i + \sum_{i \in [m]} \sum_{j \in [n]} \phi_{\{i,j\}}^{(\ell)} r_i b_j \right\}_{\ell \in [\bar{m}_1]}.$$

for certain arbitrary $\phi_i^{(\ell)}$. We refer to Appendix B for some examples.

We describe the actively secure version of our protocol in Fig. 3. Note that if the encoding is algebraic, the proof of knowledge from (1) is equivalent to the proof of knowledge described

Modification of Round 1 (User)

The user additionally sends $\pi_1$ to the authority, a zero-knowledge proof of knowledge of the statement:

$$\mathrm{PoK}\left\{\begin{array}{ll}(\phi^{(\ell)}_{\{i,j\}}, Z^{(\ell)}_i)\ \forall \ell\in[\bar{m}_1], & M^{(\ell)}_i = \left[\!\!\left[\left(\sum_{j\in[n]}\phi^{(\ell)}_{\{i,j\}} W_j B\right) + Z^{(\ell)}_i B\right]\!\!\right]_2 \quad \wedge \\ \forall i\in[m] \\ j\in[n] \\ \ell\in[\bar{m}_1] & : \forall (i,j,\ell)\in\Pi, \phi^{(\ell)}_{\{i,j\}} = f(i,j,\ell) \ \wedge\ \forall (i,j,\ell)\in\Phi, \phi^{(\ell)}_{\{i,j\}}\in\{0,1\} \ \wedge \\ & \forall s\in\Theta, \gamma_s = \sum_{(i,j,\ell)\in\Lambda_s} g_s(i,j,\ell)\phi^{(\ell)}_{\{i,j\}}\end{array}\right\}$$

Modification of Round 2 (Authority)

At the beginning of the round, the authority verifies $\pi_1$. If the proof is rejected, it returns $\perp$. Otherwise, it proceeds with the protocol.
At the end of Round 2, the authority additionally sends $\pi_2$ to the user, a zero-knowledge proof of knowledge of the statement:

$$\mathrm{PoK}\left\{\begin{array}{ll}(\mathsf{msk}, \boldsymbol{r_i}, \hat{\boldsymbol{r}}_{\boldsymbol{i'}}) & \forall i\in[m],\ \rho_i = [\![B\boldsymbol{r_i}]\!]_2 \quad \wedge \quad (\mathsf{mpk},\mathsf{msk})\in\mathcal{K} \quad \wedge \\ \forall i\in[m] \\ \forall i'\in[\hat{m}] & : \forall \ell\in[\bar{m}_1],\ \tau_\ell = \sum_{i\in[m]} M^{(\ell)}_i \boldsymbol{r_i} \quad \wedge \quad \forall \ell\in[\bar{m}_2], \\ & \sigma_\ell = h^{\varphi^{(\ell)}}_{\mathsf{msk}} \left[\!\!\left[\sum_{i'}\varphi^{(\ell)}_{i'} B\hat{\boldsymbol{r}}_{\boldsymbol{i'}} + \sum_{i,j}\varphi^{(\ell)}_{\{i,j\}} W_i B\boldsymbol{r_i}\right]\!\!\right]_2\end{array}\right\}$$

Finalizing the key (User)

On input $\pi_2$, the user first verifies that the proof is valid. If it is accepted, it proceeds as in Figure 2, otherwise, it sets $\mathsf{sk}_y := \perp$.

**Fig. 3** Modification of the protocol from Fig. 2 to achieve active security in the case of algebraic pair encodings

in Round 1 of Fig. 3. In fact, we leverage techniques from [12] to design a very efficient $\Sigma$-protocol for implementing such a proof. We present it in Appendix C.

Theorems 4 and 5 are relative to the modification of our protocol, described in Fig. 3. As in the previous section, we define honest algorithms $\mathrm{U}_1$, $\mathrm{A}_1$, that follow the *Blind-KG* protocol and $\mathrm{U}_2$, $\mathrm{A}_2$, honest algorithms that pass their input to $\mathcal{F}$ and do not modify their output.

**Theorem 4** *(Security against malicious authorities) For every algorithm $\mathcal{A}$ (acting as the authority in the real world) there exists a simulator $\mathcal{S}$ s.t. for all $\iota = (\mathsf{mpk}, y, \mathsf{msk})$, all PPT distinguishers $\mathcal{D}$, and all sufficiently large $\lambda\in\mathbb{N}$,*

$$\left|\Pr\left[\mathcal{D}(\mathit{Real}_{\mathrm{U},\mathcal{A}}(\iota))=1\right] - \Pr\left[\mathcal{D}(\mathit{Ideal}_{\mathrm{U}_2,\mathcal{S}}(\iota))=1\right]\right| \le \varepsilon_{zk}(\lambda) + \varepsilon_{ext}(\lambda).$$

**Theorem 5** *(Security against malicious users) For every algorithm $\mathcal{U}$ (acting as a user in the real world) there exists a simulator $\mathcal{S}$ s.t. for all $\iota = (\mathsf{mpk}, y, \mathsf{msk})$, all PPT distinguishers $\mathcal{D}$, and all sufficiently large $\lambda\in\mathbb{N}$,*

$$\left|\Pr\left[\mathcal{D}(\mathit{Real}_{\mathcal{U},\mathrm{A}_1}(\iota))=1\right] - \Pr\left[\mathcal{D}(\mathit{Ideal}_{\mathcal{S},\mathrm{A}_2}(\iota))=1\right]\right| \le \varepsilon_{zk}(\lambda) + \varepsilon_{ext}(\lambda) + Adv^{G_2}_{DLOG}(\lambda)$$

*where $Adv^{G_2}_{DLOG}$ is an upper-bound on the probability of PPT machines solving the discrete logarithm problem in $G_2$.*

## 5 Implementation

We implement a general library for attribute-based encryption based on predicate and pair encodings with support for blind key-generation. Our library is based on the Relic-Toolkit [5] for pairings with a 256-bits Barreto-Naehrig Curve [15]. We evaluate our blind key-generation protocol and compare it to the normal key-generation algorithm. All the experiments were executed on a 8-core machine with 4.00GHz Intel Core i7-4790K CPU and 16GB of RAM. Our source code[6] will be publicly available and open source for reproducibility and verifiability.

Our experimental results, described in Table 1, correspond to an implementation of attribute-based encryption based on the SXDH assumption (1-Lin). The first block corresponds to the pair encoding for zero inner-product described in our Example 2 (in Appendix B). The second, corresponds to an encoding for CP-ABE [20, 22, 33]. The third is an encoding for broadcast encryption, described in our Example 3.

In every block, we compare the three different algorithms of our protocol for blind key-generation described in Fig. 2 (*Round 1*, *Round 2* and *Finalize key*) and the normal key-generation algorithm, in terms of the execution time, size of the output and number of group operations over the group $G_2$. Group operations over $G_2$ are represented by a pair of integers separated by |, corresponding to the number of group law operations and group exponentiations respectively.

We show results for the median times of 10 executions of randomly selected instances of different problem sizes. In the case of inner-product, keys are generated with respect to randomly sampled vectors of the given lengths. In the case of Boolean-formulae, keys are generated with respect to randomly selected sets of attributes of the given cardinalities. In the case of broadcast encryption, keys are generated with respect to a random identity over a system of $n$ identities. Note that in this encoding, keys have approximately $t$ elements, and ciphertexts approximately $\frac{n}{t}$ group elements. Numbers in the above table correspond to the honest-but-curious version of our protocol.

We have also implemented an interactive ZK proof of knowledge based on our $\Sigma$-protocol from Appendix C. In order to give an intuition of the overhead introduced by this proof (to achieve active security), we present the numbers corresponding to proof for the inner-product encoding:

| Vector length: | 100 | 1000 | 10000 |
|---|---|---|---|
| User's total execution time: | 0.17 s | 1.61 s | 16.8 s |
| User's total information transmitted: | 3.58 KB | 32.3 KB | 0.32 MB |
| Authority's total execution time: | 0.15 s | 1.53 s | 16.53 s |
| Authority's total information transmitted: | 96 B | 96 B | 96 B |

[6] At the moment, we will provide it to the program chairs for the reviewing process, upon request.

**Table 1** Evaluation results. Comparison between blind and normal key-generation. For the honest-but-curious version of our protocol

| Inner-product | Vector length: 100 | | | Vector length: 1000 | | | Vector length: 10000 | | |
|---|---|---|---|---|---|---|---|---|---|
| | Time | Size | $G_2$ ops. | Time | Size | $G_2$ ops. | Time | Size | $G_2$ ops. |
| Round 1 | 0.05s | 128B | 202\|204 | 0.52s | 128B | 1996\|1994 | 6.16s | 128B | 19910\|19806 |
| Round 2 | 0.16s | 256B | 2\|206 | 1.67s | 256B | 2\|2006 | 22.2s | 256B | 2\|20006 |
| Finalize key | 4ms | 256B | 4\|4 | 4ms | 256B | 4\|4 | 5ms | 256B | 4\|4 |
| Normal KG | 0.24s | 256B | 200\|404 | 2.17s | 256B | 1994\|3994 | 27.3s | 256B | 19908\|39806 |
| Boolean formulae | N° of attributes: 10 | | | N° of attributes: 100 | | | N° of attributes: 1000 | | |
| | Time | Size | $G_2$ ops. | Time | Size | $G_2$ ops. | Time | Size | $G_2$ ops. |
| Round 1 | 0.04s | 1.37KB | 34\|44 | 0.41s | 12.6KB | 336\|404 | 8.18s | 125KB | 3272\|4004 |
| Round 2 | 0.05s | 1.50KB | 2\|66 | 0.66s | 12.7KB | 2\|606 | 6.85s | 125KB | 2\|6006 |
| Finalize key | 0.04s | 1.50KB | 44\|44 | 0.39s | 12.7KB | 404\|404 | 5.39s | 125KB | 4004\|4004 |
| Normal KG | 0.04s | 1.50KB | 2\|44 | 0.42s | 12.7KB | 2\|404 | 5.53s | 125KB | 2\|4004 |
| Broadcast encryption | $\frac{n}{t} = 10^3, t = 50$ | | | $\frac{n}{t} = 50, t = 10^3$ | | | $\frac{n}{t} = 10^3, t = 10^3$ | | |
| | Time | Size | $G_2$ ops. | Time | Size | $G_2$ ops. | Time | Size | $G_2$ ops. |
| Round 1 | 0.24s | 6.37KB | 208\|204 | 5.29s | 125KB | 4008\|4004 | 6.88s | 125KB | 3272\|4004 |
| Round 2 | 2.16s | 6.50KB | 2\|2206 | 4.19s | 125KB | 2\|4106 | 8.10s | 125KB | 2\|6006 |
| Finalize key | 0.19s | 6.50KB | 204\|204 | 4.09s | 125KB | 4004\|4004 | 4.42s | 125KB | 4004\|4004 |
| Normal KG | 2.11s | 6.50KB | 6\|2104 | 2.15s | 125KB | 6\|2104 | 5.18s | 125KB | 6\|4004 |

## 6 Concluding remarks

We have presented an efficient protocol for blind key-generation, built on top of the general framework for attribute-based encryption introduced by Attrapadung [7]. Our construction can be instantiated with any BKG-compatible pair encoding, a new notion that we define in this work. Furthermore, we provide theoretical results that establish that every pair encoding can be transformed with minimal overhead into a BKG-compatible one. Consequently, our work significantly broadens the scope of blind key-generation ABE in terms of predicate expressivity and it can lead to more efficient constructions of oblivious transfer with fine-grained access control.

Our experimental results show that our blind key-generation protocol introduces a reasonable overhead, comparable to the cost of normal key-generation, being suitable for real applications. We leave for future work to prove explicit asymptotic bounds on the complexity of our construction. Finally, we have focused on synthetic case studies. Combining our methods with more realistic case studies (such as electronic medical records or streaming of data over untrusted clouds) is a very interesting direction for future work.

**Supplementary Information** The online version contains supplementary material available at https://doi.org/10.1007/s10623-022-01069-5.

**Funding** Not applicable.

**Availability of data and material** Not applicable.

## Declarations

**Conflicts of interest** Any person employed by NTT would be in CoI.

**Code availability** Available now upon request (open source in the future).



## A Proofs of the main body

### A.1 Lemmas from Sect. 3

#### Proof of Lemma 1

Note that our notation allows us to handle the different cases $w < \hat{w}$ and $w \geq \hat{w}$ (or $m < \hat{m}$ and $m \geq \hat{m}$) simultaneously. (Observe that the range $[w+1, \hat{w}]$ is defined as empty if $w \geq \hat{w}$.)

***Proof*** Let $\Upsilon = (\bar{w}, w, \hat{w}, \bar{m}, m, \hat{m}, n)$ and let $(EncCt, EncKey, Pair)$ be a $\Upsilon$-pair encoding for $P$. We define a new $\Upsilon'$-pair encoding, $(EncCt', EncKey', Pair')$, below, where

$\Upsilon' = (\bar{w},\ \max(w, \hat{w}),\ 0,\ \bar{m},\ \max(m, \hat{m}),\ 0,\ n{+}2)$ and then argue that it is a valid encoding for the same predicate $P$.

Let $\boldsymbol{b}, \boldsymbol{s}, \hat{\boldsymbol{s}}, \boldsymbol{r}, (\alpha, \hat{\boldsymbol{r}})$ be the variables of the original encoding. (Note that $|\boldsymbol{b}| = n$, $|\boldsymbol{s}| = w$, $|\hat{\boldsymbol{s}}| = \hat{w}$, $|\boldsymbol{r}| = m$, $|\hat{\boldsymbol{r}}| = \hat{m}$.) Consider fresh variables $b_{n+1}, b_{n+2}, s_i, r_j$, for $i \in [w{+}1, \hat{w}]$ and $j \in [m{+}1, \hat{m}]$ and let the variables of the new encoding be $\hat{\boldsymbol{s}}' = \emptyset$, $(\alpha', \hat{\boldsymbol{r}}') = (\alpha, \emptyset)$ and [7]

$$\boldsymbol{b}' = (\boldsymbol{b}, b_{n+1}, b_{n+2}) \quad \boldsymbol{s}' = (\boldsymbol{s}, s_{w+1}, \ldots, s_{\hat{w}}) \quad \boldsymbol{r}' = (\boldsymbol{r}, r_{m+1}, \ldots, r_{\hat{m}}).$$

Define vectors [8] $\tilde{\boldsymbol{s}} = (s_1, \ldots, s_{\hat{w}})$ and $\tilde{\boldsymbol{r}} = (r_1, \ldots, r_{\hat{m}})$, and let the encoding be

- $EncCt'(p, x)$: run $\boldsymbol{c}_x \leftarrow EncCt(p, x)$, output $\boldsymbol{c}'_x(\boldsymbol{s}', \boldsymbol{b}') := \boldsymbol{c}_x(\boldsymbol{s}, b_{n+1}\tilde{\boldsymbol{s}}, \boldsymbol{b})$. [9]
- $EncKey'(p, y)$: run $\boldsymbol{k}_y \leftarrow EncKey(p, y)$, output $\boldsymbol{k}'_y(\boldsymbol{r}', \boldsymbol{b}') := \boldsymbol{k}_y(\boldsymbol{r}, (\alpha, b_{n+2}\tilde{\boldsymbol{r}}), \boldsymbol{b})$.
- $Pair'(p, x, y)$: run $(E, F) \leftarrow Pair(p, x, y)$ and output $(E', F')$ defined as

$$E' := \begin{pmatrix} E \\ 0 \end{pmatrix} \qquad F' := \begin{pmatrix} F & 0 \end{pmatrix}$$

  where $w' := \max(\hat{w}{-}w, 0)$ and $m' := \max(\hat{m}{-}m, 0)$.

Observe that the presented encoding does not contain lone variables (except $\alpha$), since all variables in $\boldsymbol{s}'$ and $\boldsymbol{r}'$ always appear multiplied by $b_j$, for some $j \in [n{+}2]$. Also, observe that all structural constraints are satisfied.

To see reconstructability, note that, due to the correctness of the original encoding, for every $(x, y) \in P$, $\boldsymbol{s}^\top E \boldsymbol{k}_y(\boldsymbol{r}, (\alpha, \hat{\boldsymbol{r}}), \boldsymbol{b}) + \boldsymbol{c}_x(\boldsymbol{s}, \hat{\boldsymbol{s}}, \boldsymbol{b})^\top F \boldsymbol{r} = \alpha s_1$ and, because the right-hand side does not contain variables $\hat{\boldsymbol{s}}$ nor $\hat{\boldsymbol{r}}$, it must be

$$\begin{aligned} \alpha s_1 &= \boldsymbol{s}^\top E \boldsymbol{k}_y(\boldsymbol{r}, (\alpha, b_{n+2}\tilde{\boldsymbol{r}}), \boldsymbol{b}) + \boldsymbol{c}_x(\boldsymbol{s}, b_{n+1}\tilde{\boldsymbol{s}}, \boldsymbol{b})^\top F \boldsymbol{r} \\ &= \boldsymbol{s}'^\top E' \boldsymbol{k}'_y(\boldsymbol{r}', \boldsymbol{b}') + \boldsymbol{c}'_x(\boldsymbol{s}', \boldsymbol{b}')^\top F' \boldsymbol{r}' \end{aligned}$$

where $\boldsymbol{c}'_x(\boldsymbol{s}', \boldsymbol{b}') \leftarrow EncCt'(p, x)$, and $\boldsymbol{k}'_y(\boldsymbol{r}', \boldsymbol{b}') \leftarrow EncKey'(p, y)$ and $(E', F') \leftarrow Pair'(p, x, y)$.

Finally, for security (non-reconstructability), we proceed by contradiction. Assume that there exist $(x, y) \notin P$ and there exist matrices $E_1, E_2, F_1, F_2$ s.t.

$$\boldsymbol{s}^\top E_1 \boldsymbol{k}'_y + (s_{w+1}, \ldots, s_{\hat{w}}) E_2 \boldsymbol{k}'_y + \boldsymbol{c}'^\top_x F_1 \boldsymbol{r} + \boldsymbol{c}'^\top_x F_2 (r_{m+1}, \ldots, r_{\hat{m}})^\top = \alpha s_1. \tag{2}$$

The above is a polynomial equality and so, it must hold when we only consider the monomials that contain $b_{n+2}$, which gives us

$$\boldsymbol{s}^\top E_1 \boldsymbol{k}_y(\boldsymbol{0}_m, (0, \tilde{\boldsymbol{r}}), \boldsymbol{0}_n) + (s_{w+1}, \ldots, s_{\hat{w}}) E_2 \boldsymbol{k}_y(\boldsymbol{0}_m, (0, \tilde{\boldsymbol{r}}), \boldsymbol{0}_n) = 0.$$

But now, looking at the coefficient of $s_i$, for every $i \in [w]$ in the above equality, we get $E_1 \boldsymbol{k}_y(\boldsymbol{0}_m, (0, \tilde{\boldsymbol{r}}), \boldsymbol{0}_n) = \boldsymbol{0}_w$ and, by linearity, we have that

$(*)$ the vector of polynomials $E_1 \boldsymbol{k}_y(\boldsymbol{r}, (\alpha, \hat{\boldsymbol{r}}), \boldsymbol{b})$ does not depend on $\hat{\boldsymbol{r}}$.

By a very similar analysis based on variable $b_{n+1}$ and the coefficient of $r_i$, for $i \in [m]$, we can deduce that

[7] Observe that if $w \geq \hat{w}$, $\boldsymbol{s}' = \boldsymbol{s}$. Analogously, if $m \geq \hat{m}$, $\boldsymbol{r}' = \boldsymbol{r}$.

[8] Note that if $w \leq \hat{w}$, $\tilde{\boldsymbol{s}} = \boldsymbol{s}'$ and otherwise, $\tilde{\boldsymbol{s}}$ equals the first $\hat{w}$ components of $\boldsymbol{s}$. Analogously for $\tilde{\boldsymbol{r}}$.

[9] $\boldsymbol{c}_x(\boldsymbol{s}, b_{n+1}\tilde{\boldsymbol{s}}, \boldsymbol{b})$ denotes evaluation of polynomial $\boldsymbol{c}_x : \mathbb{Z}_p^w \times \mathbb{Z}_p^{\hat{w}} \times \mathbb{Z}_p^n \to \mathbb{Z}_p$ on value $(\boldsymbol{s}, (b_{n+1}\tilde{s}_1, \ldots, b_{n+1}\tilde{s}_{\hat{w}}), \boldsymbol{b})$.

(∗∗) the vector of polynomials $\boldsymbol{c}_x(\boldsymbol{s}, \hat{\boldsymbol{s}}, \boldsymbol{b})^\top F_1$ does not depend on $\hat{\boldsymbol{s}}$.

Now, substituting variables $s_{w+1}, \ldots, s_{\hat{w}}$ and $r_{m+1}, \ldots, r_{\hat{m}}$ by 0 in Eq. (2), and using facts (∗) and (∗∗), we get

$$\boldsymbol{s}^\top E_1 \boldsymbol{k}_y(\boldsymbol{r}, (\alpha, \hat{\boldsymbol{r}}), \boldsymbol{b}) + \boldsymbol{c}_x(\boldsymbol{s}, \hat{\boldsymbol{s}}, \boldsymbol{b})^\top F_1 \boldsymbol{r} = \alpha s_1$$

which contradicts the security of the original encoding. □

### Proof of Lemma 2

***Proof*** It is known that, without loss of generality, we can consider pair encodings where the first polynomial of $EncKey$ is the only one that contains $\alpha$ and it is of the form $\alpha + r_1 b_1$.

This fact has already been used in the literature [4, 9] for other purposes. It can be easily derived by applying generic dual transformation defined in [10] twice. (Note that the dual transformation is an involution and does not modify the predicate after being applied twice.) Lemma 2 is a direct consequence of the combination of the previous fact with our Lemma 1. □

## A.2 Theorems from Sect. 3.3

### Proof of Theorem 1

***Proof*** Observe that the part of the secret key that does not depend on $y$, i.e., $\boldsymbol{r}$ and $EncKey2(p)$, has been given to the user as $\{\rho_i\}_{i\in[m]}$ and $\{\sigma_\ell\}_{\ell\in[\bar{m}_2]}$ respectively and these values have been computed exactly as in $KeyGen$. The remaining elements of the key have been computed (for every $\ell \in [\bar{m}_1]$) as:

$$\begin{aligned}
\widetilde{\mathsf{sk}}_\ell :=\ & \tau_\ell - \textstyle\sum_{i=1}^{m} Z_i^{(\ell)} \rho_i + \sum_{i=1}^{\hat{m}} \phi_i^{(\ell)} \sigma_{\pi(i)} \\
=\ & \textstyle\sum_{i=1}^{m} \left( Z_i^{(\ell)} + \sum_{j=1}^{n} \phi_{\{i,j\}}^{(\ell)} W_j \right) B\boldsymbol{r_i} - \sum_{i=1}^{m} Z_i^{(\ell)} B\boldsymbol{r_i} + \sum_{i=1}^{\hat{m}} \phi_i^{(\ell)} \sigma_{\pi(i)} \\
=\ & \textstyle\sum_{i=1}^{m} \left( \sum_{j=1}^{n} \phi_{\{i,j\}}^{(\ell)} W_j B\boldsymbol{r_i} \right) + \sum_{i=1}^{\hat{m}} \phi_i^{(\ell)} B\hat{\boldsymbol{r}}_{\boldsymbol{i}}
\end{aligned}$$

which is identically distributed to the output of $KeyGen$. □

### Proof of Theorem 2

***Proof*** Note that $view_{\mathrm{A}_1}$ in the real world contains $\mathsf{mpk}$ and the messages from the user corresponding to $M_i^{(\ell)}$ for $i \in [m]$ and $\ell \in [\bar{m}_1]$, whereas in the ideal world, $view_{\mathcal{S}}$ only contains $\mathsf{mpk}$. The simulator can complete the view by sampling a uniform random matrix $V_i^{(\ell)}$ of dimension $(k+1) \times (k+1)$ (with coefficients over $\mathbb{Z}_p$), for every $i \in [m]$ and $\ell \in [\bar{m}_1]$ and simulating $M_i^{(\ell)}$ as $V_i^{(\ell)} B$. Observe that the simulation is perfect. □

### Proof of Theorem 3

***Proof*** In the real world, if $(\mathsf{mpk}, \mathsf{msk}) \in \mathcal{K}$, the user's view will be formed by

$$view_{\mathrm{U1}} = \left( \{Z_i^\ell\}_{i\in[m],\ell\in[\bar{m}_1]},\quad \{\rho_i\}_{i\in[m]},\quad \{\sigma_\ell\}_{\ell\in[\bar{m}_2]},\quad \{\tau_\ell\}_{\ell\in[\bar{m}_1]} \right)$$

where $Z_i^{(\ell)}$ are an independent uniformly sampled random matrices of dimension $(k+1) \times (k+1)$, and where $\rho_i, \sigma_\ell$ and $\tau_\ell$ are computed as in Fig. 2 (based on $Z_i^{(\ell)}$ and $y$). The simulator can sample matrices $\widetilde{Z}_i^{(\ell)}$ uniformly at random of dimension $(k+1) \times (k+1)$, then call the ideal functionality $\mathcal{F}$ on $(\mathsf{mpk}, y)$, receiving $\mathsf{sk}_y := (\mathsf{sk}_1, \ldots, \mathsf{sk}_m, \widetilde{\mathsf{sk}}_1, \ldots, \widetilde{\mathsf{sk}}_{\bar{m}})$. It now sets

$$\widetilde{\rho}_i := \mathsf{sk}_i \text{ for every } i \in [m] \qquad \widetilde{\sigma}_\ell := \widetilde{\mathsf{sk}}_{\bar{m}_1+\ell} \text{ for every } \ell \in [\bar{m}_2] \text{ and}$$

$$\widetilde{\tau}_\ell := \widetilde{\mathsf{sk}}_\ell + \textstyle\sum_{i=1}^{m} \widetilde{Z}_i \widetilde{\rho}_i - \sum_{i=1}^{\hat{m}} \phi_i^{(\ell)} \widetilde{\sigma}_{\pi(i)} \text{ for every } \ell \in [\bar{m}_1]$$

where $\phi_i^{(\ell)}$ is the coefficient of $\hat{r}_i$ in the $\ell$-th polynomial of $EncKey1(p, y)$, for every $i \in [\hat{m}]$. Because every party is honest, note that $view_{\mathrm{A}_1} \equiv (\mathsf{mpk}, \mathsf{msk}) = view_{\mathrm{A}_2}$ and observe that the simulated view

$$view_{\mathcal{S}} = \left( \{\widetilde{Z}_i^{\ell}\}_{i\in[m],\ell\in[\bar{m}_1]}, \quad \{\widetilde{\rho}_i\}_{i\in[m]}, \quad \{\widetilde{\sigma}_\ell\}_{\ell\in[\bar{m}_2]}, \quad \{\widetilde{\tau}_\ell\}_{\ell\in[\bar{m}_1]} \right)$$

is identically distributed to the real view $view_{\mathrm{U}_1}$ for any $(\mathsf{mpk}, y, \mathsf{msk})$ such that $(\mathsf{mpk}, \mathsf{msk}) \in \mathcal{K}$. Finally, if $(\mathsf{mpk}, \mathsf{msk}) \notin \mathcal{K}$, note that the user's view in the real world will consist only of random matrices (the authority will interrupt the $Blind\text{-}KG$ protocol in Round 2), which can be easily simulated. □

### Proof of Theorem 4, security against malicious authorities

***Proof*** In the real world, $\mathsf{view}_{\mathcal{A}}$ contains $\mathsf{msk}, \mathsf{mpk}$, the messages from the user corresponding to $M_i^{(\ell)}$ for all $i \in [m]$, $\ell \in [\bar{m}_1]$ and the zero-knowledge proof of knowledge $\pi_1$. On the other hand, in the ideal world, the initial view of the simulator only contains $\mathsf{msk}$ and, after receiving the first message from $\mathcal{F}$, $\mathsf{view}_{\mathcal{S}}$ additionally contains $\mathsf{mpk}$. The simulator can complete the view by sampling uniform random matrices $V_i^{(\ell)}$ of dimension $(k+1) \times (k+1)$ for every $i \in [m]$ and $\ell \in [\bar{m}_1]$, simulating $M_i^{(\ell)}$ as $V_i^{(\ell)} B$ and simulating a zero knowledge proof of knowledge $\widetilde{\pi}_1$ for the NP-statement described in Round 1 of Fig. 3. Observe that the simulation is perfect, except for the zero-knowledge simulation error.

Now from these simulated values, $\mathcal{S}$ will proceed exactly as algorithm $\mathcal{A}$ does in the real world, eventually producing some values $\{\widetilde{\rho}_i\}_{i\in[m]}$, $\{\widetilde{\sigma}_\ell\}_{\ell\in[\bar{m}_2]}$ and $\{\widetilde{\tau}_\ell\}_{\ell\in[\bar{m}_1]}$, together with a proof of knowledge $\widetilde{\pi}_2$ for the NP-relation described in Round 2 of Fig. 3 w.r.t these values. The simulator will verify $\widetilde{\pi}_2$ and if the proof is rejected, it will interrupt its execution (it will not send any message to the ideal functionality $\mathcal{F}$). If it is accepted, it will run the extractor algorithm on $\widetilde{\pi}_2$, to obtain values $\mathsf{msk}'$, and $r = \left(\{\boldsymbol{r_i}\}_{i\in[m]}, \{\hat{\boldsymbol{r}}_{\boldsymbol{i}'}\}_{i'\in[\hat{m}]}\right)$ such that $(\mathsf{mpk}, \mathsf{msk}') \in \mathcal{K}$ and

$$\forall\, i \in [m],\ \widetilde{\rho}_i = [\![ B\boldsymbol{r_i} ]\!]_2 \qquad \wedge \qquad \forall\, \ell \in [\bar{m}_1],\ \widetilde{\tau}_\ell = \textstyle\sum_{i\in[m]} V_i^{(\ell)} B\boldsymbol{r_i}$$
$$\forall \ell \in [\bar{m}_2],\ \widetilde{\sigma}_\ell = h_{\mathsf{msk}'}^{\varphi^{(\ell)}} \cdot [\![ \textstyle\sum_{i'} \varphi_{i'}^{(\ell)} B\hat{\boldsymbol{r}}_{\boldsymbol{i}'} + \sum_{i,j} \varphi_{\{i,j\}}^{(\ell)} W_i B\boldsymbol{r_i} ]\!]_2.$$

Then, the simulator will allow the secret key-generation by sending $(\mathsf{msk}', r)$ to the ideal functionality $\mathcal{F}$. Observe that if the extraction is successful, the secret key that $\mathrm{U}_2$ will receive in the ideal world is distributed as the secret key that $\mathrm{U}_1$ will obtain after finishing the protocol from Fig. 3. Note that there is only one $\mathsf{msk}'$ that satisfies the above relation, but there exist many different values $r$. However, observe that all values of $r$ that satisfy the relation will lead to the same secret key, since the actual key can be seen as a function of values $V_i^{(\ell)}$, $\rho_i$, $\sigma_\ell$ and $\tau_\ell$, which are invariant for every $r$.

Therefore, the only difference between the real and the simulated views is due to the zero-knowledge error. To finished the proof note that we could convert any any distinguisher between the real and ideal views into a distinguisher for for the zero-knowledge property of the ZK scheme. To conclude, note that all the steps where conditioned on the event that extraction was successful. Therefore, the final bound must take into account the extraction error $\varepsilon_{ext}$. □

### Proof of Theorem 5, security against malicious users

***Proof*** On input $(\mathsf{mpk}, y)$, the authority will proceed exactly as algorithm $\mathcal{U}$, producing $\mathsf{mpk}'$, some values $\{\widetilde{M}_i^{(\ell)}\}_{\ell\in[\bar{m}_1], i\in[m]}$ and a proof $\widetilde{\pi}_1$, identically distributed to the values that $\mathcal{U}$ will send to $\mathrm{A}_1$ in the real world.

The simulator will then verify $\widetilde{\pi}_1$ with respect to the produced values and the relation described and Round 1 of Fig. 3. If the verification fails, $\mathcal{S}$ will interrupt its execution. Note that this happens with exactly the same probability $\mathrm{A}_1$ rejects the proof in the real world and halts. If the proof is accepted, the simulator will run the extractor to obtain witness values $\{\phi_{\{i,j\}}^{(\ell)}\}_{i\in[m],\, j\in[n]}$ in $\mathbb{Z}_p$ and matrices $\{Z_i^{(\ell)}\}_{\ell\in[\bar{m}_1], i\in[m]}$ in $\mathbb{Z}_p$ of dimension $(k+1)\times(k+1)$ such that

$$\forall\, \ell \in [\bar{m}_1],\ M_i^{(\ell)} = [\![ \big( \textstyle\sum_{j\in[n]} \phi_{\{i,j\}}^{(\ell)} W_j B \big) + Z_i^{(\ell)} B ]\!]_2 \quad \wedge \quad \textstyle\sum_{(i,j)\in\Phi_\ell} \phi_{\{i,j\}}^{(\ell)} = \gamma_\ell$$
$$\forall (i,j,\mu) \in \Pi_\ell,\ \phi_{\{i,j\}}^{(\ell)} = \mu \quad \wedge \quad \forall (i,j) \in \Phi_\ell,\ \phi_{\{i,j\}}^{(\ell)} \in \{0,1\}.$$

Observe that the Pedersen-like commitments $M_i^{(\ell)}$ are computationally binding under the discrete logarithm assumption in $G_2$. Therefore, although there exist several witnesses for the NP relation, the extractor algorithm can only provide one of them (or otherwise, it can be used to solve the discrete logarithm problem). This guarantees that the witness that the simulator extracts is identically distributed to the witness that the malicious user may also extract (from its own proof) in the real world. Now, let $\boldsymbol{k}_1$ be the vector of polynomials

$$\boldsymbol{k}_1 := \big\{ \textstyle\sum_{i\in[\hat{m}]} \phi_i^{(\ell)} \hat{r}_i + \sum_{i\in[m]} \sum_{j\in[n]} \phi_{\{i,j\}}^{(\ell)} r_i b_j \big\}_{\ell\in[\bar{m}_1]}.$$

Because the pair encoding is *algebraic*, there exists $y' \in \mathcal{Y}$ such that $EncKey1(y') \to \boldsymbol{k}_1$. And, because the pair encoding is *invertible*, the simulator can find such $y'$ by running $DecKey(\boldsymbol{k}_1 \parallel EncKey2)$. Now, the simulator sends $(\mathsf{mpk}', y')$ to the ideal functionality, obtaining $\mathsf{sk}_{y'} := (\mathsf{sk}_1, \ldots, \mathsf{sk}_m, \widetilde{\mathsf{sk}}_1, \ldots, \widetilde{\mathsf{sk}}_{\bar{m}})$ or $\perp$. The case of $\perp$ will occur only when the master secret key $\mathsf{msk}$ owned by $\mathrm{A}_2$, and $\mathsf{mpk}'$ do not form a valid pair. In such a case, the simulator halts. Observe that, in the real world, $\mathrm{A}_1$ (who owns the same $\mathsf{msk}$) will detect that $(\mathsf{mpk}', \mathsf{msk})$ at the beginning of Round 2, with exactly the same probability $\mathcal{S}$ received $\perp$ from $\mathcal{F}$. In case of receiving a key, the simulator sets

$$\widetilde{\rho}_i := \mathsf{sk}_i \ \text{ for every } i \in [m] \qquad \widetilde{\sigma}_\ell := \widetilde{\mathsf{sk}}_{\bar{m}_1+\ell} \ \text{ for every } \ell \in [\bar{m}_2] \quad \text{and}$$

$$\widetilde{\tau}_\ell := \widetilde{\mathsf{sk}}_\ell + \textstyle\sum_{i=1}^{m} \widetilde{Z}_i \widetilde{\rho}_i - \sum_{i=1}^{\hat{m}} \phi_i^{(\ell)} \widetilde{\sigma}_{\pi(i)} \ \text{ for every } \ell \in [\bar{m}_1]$$

where $\phi_i^{(\ell)}$ is the coefficient of $\hat{r}_i$ in the $\ell$-th polynomial of $EncKey1(p, y)$, for every $i \in [\hat{m}]$. As in the proof of honest-but-curious security, the above simulated values are identically distributed to the real view $view_{\mathcal{U}}$. We can conclude as in the proof of Theorem 4. □

## B Examples

***Example 1*** Consider the following pair encoding for the IBE predicate. That is, $P(x, y) = 1$ iff $x = y$, for $x, y \in \mathbb{Z}_p$:

$$EncKey(y){:=}(\alpha + r_1 b_1,\ \ yr_1 b_2 + r_1 b_3) \qquad EncCt(x){:=}s_1 b_1 + x s_2 b_2 + s_2 b_3$$

The second polynomial of $EncKey(y)$ contains a secret-coefficient monomial: $yr_1 b_2$. On the other hand, $r_1 b_3$ is a public monomial. Note that this polynomial contains other public monomials (with 0 coefficient), e.g. $0r_1 b_1$.

Now, consider the pair encoding for KP-ABE corresponding to the scheme by Lewko et al. [37]. For attribute universe $\mathcal{U}$, $X \in \mathcal{U}$, a linear secret sharing matrix $A \in \mathbb{Z}_p^{\ell \times k}$ and $\pi : [\ell] \to \mathcal{U}$,

$$EncKey(A, \pi){:=}\{A_i \hat{\boldsymbol{r}} + r_i b_{\pi(i)}\}_{i \in [\ell]} \qquad EncCt(X){:=}\{s_1 b_x\}_{x \in X}$$

where $\hat{\boldsymbol{r}}{:=}(\alpha, \hat{r}_1, \ldots, \hat{r}_{k-1})^\top$, and $P(X, (A, \pi))$ is defined as 1 iff $\exists \Phi \subseteq [\ell] : \pi(\Phi) \subseteq X$ and $(1, 0, \ldots, 0)$ is in the row span of $A_\Phi$.

In this case, every monomial in $A_i \hat{\boldsymbol{r}}$ is a secret-coefficient monomial, whereas monomials of the from $r_i b_{\pi(i)}$ are secret-variable monomials. Note that both pair encodings presented above are *surjective*. □

***Example 2*** (Pair encoding for zero-inner product [11]) Let $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{Z}_p^n$ and let the predicate $P$ be defined as $P(\boldsymbol{x}, \boldsymbol{y}) = 1$ iff $\boldsymbol{x}^\top \boldsymbol{y} = 0$. The following is a pair encoding for $P$:

$$EncKey(\boldsymbol{y}){:=}(\alpha + r_1 b_0,\ \ r_1 b_0 + \textstyle\sum_{i \in [n]} y_i r_1 b_i) \quad EncCt(\boldsymbol{x}){:=}\{x_i s_1 b_{n+1} + s_1 b_i\}_{i \in [n]}$$

In this case, $EncKey1(y)$ is formed by the second polynomial of $EncKey$ and so, $\bar{m}_1 = 1$. Furthermore, the encoding is algebraic, where:

(public coeffs) $\Pi{:=}\{(1,0,1), (1,n+1,1)\}$ and $f(1,0,1){:=}1$, $f(1,n+1,1){:=}0$.
(binary coeffs) $\Phi{:=}\emptyset$. (clusters) $\Lambda{:=}\emptyset$. ■

***Example 3*** (Pair encoding for broadcast encryption [20, 27]) For $t, n \in \mathbb{N}$ such that $t|n$, let $\boldsymbol{x} \in (\{0,1\}^{\frac{n}{t}})^t$, $y \in [t] \times [\frac{n}{t}]$ and let the predicate $P$ be defined as $P((\boldsymbol{x}_1, \ldots, \boldsymbol{x}_t), (y_1, y_2)) = 1$ iff $(\boldsymbol{x}_{y_1})_{y_2} = 1$. The following is a pair encoding for $P$:

$$EncKey(y_1, y_2){:=}(\alpha + r_1 b_0,\ \ \{(\ell = y_2)(r_1 b_0 + r_1 b_{y_1}) + r_1 b_{t+\ell}\}_{\ell \in [\frac{n}{t}]})$$
$$EncCt(\boldsymbol{x}){:=}\{s_1 b_\ell + \textstyle\sum_{j \in [\frac{n}{t}]} (\boldsymbol{x}_\ell)_j s_1 b_{t+j}\}_{\ell \in [t]}$$

In this case, $EncKey1(y)$ is formed all the polynomials of $EncKey$ except the first one and so, $\bar{m}_1 = \frac{n}{t}$. Furthermore, the encoding is algebraic, where:

(public coeffs) $\Pi{:=}\{(1,t+j,\ell)\}_{j,\ell \in [\frac{t}{n}]}$ where $\forall \ell$, $f(1,t+\ell,\ell){:=}1$, and $\forall j, \ell \neq j$, $f(1,t+j,\ell){:=}0$.
(binary coeffs) $\Phi{:=}\Phi_1 \cup \Phi_2$, with $\Phi_1{:=}\{(1, j, \ell)\}_{j,\ell \in [\frac{n}{t}]}$, $\Phi_2{:=}\{(1, 0, \ell)\}_{\ell \in [\frac{n}{t}]}$.
(clusters) $\Lambda{:=}\{(\Phi_1, (\lambda x.1), 1),\ \{(\Lambda_\ell, g, 0)\}_{\ell \in [\frac{n}{t}]}\}$.

where $\lambda x.1$ denotes the constant function 1; $\forall \ell \in [\frac{n}{t}]$, $\Lambda_\ell{:=}\{(1, j, \ell)\}_{j \in [0, \frac{n}{t}]}$; and $g$ is a function defined as $-1$ if the second argument is 0, or 1 otherwise. □

$\mathcal{P}(\mathsf{mpk}, \mathsf{crs}, \{M_i\}_i, \{\phi_{\{i,j\}}\}_{i,j}; \{Z_i\}_i)$ :

$a_{\{i,j\}} \xleftarrow{\$} \mathbb{Z}_p,\ \forall (i,j) \in [m]\times[n]$

$\forall s \in \Theta,$ pick[10] some $(i_s, j_s) \in \Lambda_s$; modify $a_{\{i_s,j_s\}}$ s.t. $\sum_{(i,j)\in\Lambda_s} g_s(i,j) a_{\{i,j\}} = 0$

modify $a_{\{i,j\}} := 0,\ \forall (i,j) \in \Pi$

define $b_{\{i,j\}}$ as $a_{\{i,j\}}(1 - 2\,\phi_{\{i,j\}})$ if $(i,j) \in \Phi$ and 0 otherwise, $\forall (i,j) \in [m]\times[n]$

define $c_{\{i,j\}}$ as $-a^2_{\{i,j\}}$ if $(i,j) \in \Phi$ and 0 otherwise, $\forall (i,j) \in [m]\times[n]$

sample $R_i, S_i, T_i$ uniformly, from $\mathbb{Z}_p^{(k+1)\times(k+1)}, \forall i \in [m]$

$A_i := \mathrm{Com}_{\mathsf{mpk}}(a_{\{i,1\}}, \ldots, a_{\{i,n\}}; R_i)$

$B_i := \mathrm{Com}_{\mathsf{mpk}}(b_{\{i,1\}}, \ldots, b_{\{i,n\}}; S_i)$

$C_i := \mathrm{Com}_{\mathsf{mpk}}(c_{\{i,1\}}, \ldots, c_{\{i,n\}}; T_i)$ $\qquad$ $\mathcal{V}(\mathsf{mpk}, \mathsf{crs}, \{M_i\}_i)$ :

$\xrightarrow{\{A_i, B_i, C_i\}_{i\in[m]}}$

$\xleftarrow{e \xleftarrow{\$} \{0,1\}^\lambda}$

$\beta_{\{i,j\}} := e\,\phi_{\{i,j\}} + a_{\{i,j\}}$

$\Gamma_i := e\,Z_i + R_i$

$\Psi_i := e\,S_i + T_i$ $\qquad$ $\xrightarrow{\{\beta_{\{i,j\}}\}_{i,j},\ \{\Gamma_i, \Psi_i\}_{i\in[m]}}$ Accept if and only if

$$\begin{aligned} &\forall (i,j) \in \Pi,\ \beta_{\{i,j\}} = e\, f(i,j) \\ \wedge\quad &\forall s \in \Theta,\ \textstyle\sum_{(i,j)\in\Lambda_s} g_s(i,j)\beta_{\{i,j\}} = \gamma_s\, e \\ \wedge\quad &\forall i \in [m],\ M_i^e \circ A_i = \mathrm{Com}_{\mathsf{mpk}}(\beta_{\{i,1\}}, \ldots, \beta_{\{i,n\}}; \Gamma_i) \\ \wedge\quad &\forall i \in [m],\ B_i^e \circ C_i = \mathrm{Com}_{\mathsf{mpk}}(\eta_{\{i,1\}}, \ldots, \eta_{\{i,n\}}; \Psi_i) \end{aligned}$$

where $\eta_{\{i,j\}} := \beta_{\{i,j\}}(e - \beta_{\{i,j\}})$ if $(i,j) \in \Phi$ and 0 otherwise.

**Fig. 4** $\Sigma$-protocol for relation $\mathcal{R}_1$. Operator $\circ$ denotes element-wise product

## CΣ-protocol for the PoK of Fig. 3

For coefficients $\phi_j \in \mathbb{Z}_p$ for all $j \in [n]$, and a matrix $Z$ over $\mathbb{Z}_p$ of dimension $(k+1)\times(k+1)$, we define[10]

$$\mathrm{Com}_{\mathsf{mpk}}(\phi_1, \ldots, \phi_n; Z) := [\![ \textstyle\sum_{j\in[n]} \phi_j W_j B ) + ZB ]\!]_2.$$

In Fig. 4, we provide a zero-knowledge proof of knowledge for the relation

$$\mathcal{R}_1 := \left\{ \begin{array}{ll} (\phi_{\{i,j\}}, Z_i) & \forall i \in [m],\ M_i = \mathrm{Com}_{\mathsf{mpk}}(\phi_{\{i,1\}}, \ldots, \phi_{\{i,n\}}; Z_i) \quad \wedge \\ \forall i \in [m] : & \forall (i,j) \in \Pi,\ \phi_{\{i,j\}} = f(i,j) \quad \wedge \quad \forall (i,j) \in \Phi,\ \phi_{\{i,j\}} \in \{0,1\} \quad \wedge \\ j \in [n] & \forall s \in \Theta,\ \gamma_s = \textstyle\sum_{(i,j)\in\Lambda_s} g_s(i,j)\phi_{\{i,j\}} \end{array} \right\}$$

Such a protocol can be used to prove the relation described in the first round of Fig. 3. Note that, for the sake of simplicity in the notation, we have not considered the $\ell \in [\bar{m}_1]$ indices in $\mathcal{R}_1$. In the majority of pair encodings, where the *clusters* do not involve elements between different polynomials, the generic conjunction of $\Sigma$-protocols is enough to handle the necessary relation. For the rest of pair encodings (an example is the encoding from

[10] This can be always done if $(i_s, j_s)$ is such that $g_s(i_s, j_s) \neq 0$. If $g_s$ is always 0, nothing needs to be done.

Example 3), it is not complicated to modify our $\Sigma$-protocol consider all polynomials at the same time.

**Lemma 3** *The protocol described in Fig. 4 is perfectly* complete*, perfect* special honest verifier zero-knowledge *and* 3-special sound.

For the sake of space, we do not include an explicit protocol for the relation in Round 2 of Fig. 3. We note that such a proof is necessary to satisfy the strong simulation-based security. In practice, such a proof is not so critical. The user is interested in receiving a valid secret key for $y$ and it can check the validity of the obtained key by encrypting messages for different values $x$ and decrypting them with the key, verifying that decryption is successful whenever $(x, y) \in P$.

***Proof Of Lemma 3*** • *Perfect completeness*. Note that if the prover knows a valid witness and follows the protocol all verification equations are satisfied. In particular, $\forall (i,j) \in \Pi$, $a_{\{i,j\}}=0$, and so, for the first verification equation,

$$\forall (i,j) \in \Pi,\ \beta_{\{i,j\}} = e\,\phi_{\{i,j\}} + a_{\{i,j\}} = e\,\phi_{\{i,j\}} = e\,f(i,j).$$

For the second equation,

$$\begin{aligned}\forall s \in \Theta,\ \textstyle\sum_{(i,j)\in\Lambda_s} g_s(i,j)\beta_{\{i,j\}} &= \textstyle\sum_{(i,j)\in\Lambda_s} \left(g_s(i,j)e\,\phi_{\{i,j\}} + a_{\{i,j\}}\right)\\ &= e\textstyle\sum_{(i,j)\in\Lambda_s} g_s(i,j)\phi_{\{i,j\}} = e\,\gamma_s.\end{aligned}$$

For the third, for all $i \in [m]$:

$$\begin{aligned}M_i^e \circ A_i &= \mathsf{Com}_{\mathsf{mpk}}(\{\phi_{\{i,j\}}\}_{j\in[n]};\, Z_i)^e \circ \mathsf{Com}_{\mathsf{mpk}}(\{a_{\{i,j\}}\}_{j\in[n]};\, R_i)\\ &\overset{11}{=} \mathsf{Com}_{\mathsf{mpk}}(\{e\,\phi_{\{i,j\}} + a_{\{i,j\}}\}_{j\in[n]};\, e\,Z_i + R_i) = \mathsf{Com}_{\mathsf{mpk}}(\beta_{\{i,1\}}, \ldots, \beta_{\{i,n\}};\, \Gamma_i)\,.\end{aligned}$$

Finally, the fourth also holds, for all $i \in [m]$:

$$\begin{aligned}B_i^e \circ C_i &= \mathsf{Com}_{\mathsf{mpk}}(\{b_{\{i,j\}}\}_{j\in[n]};\, S_i)^e \circ \mathsf{Com}_{\mathsf{mpk}}(\{c_{\{i,j\}}\}_{j\in[n]};\, T_i)\\ &= \mathsf{Com}_{\mathsf{mpk}}(\{e\,b_{\{i,j\}} + c_{\{i,j\}}\}_{j\in[n]};\, e\,S_i + T_i) = \mathsf{Com}_{\mathsf{mpk}}(\eta_{\{i,1\}}, \ldots, \eta_{\{i,n\}};\, \Psi_i)\,.\end{aligned}$$

because, $\forall (i,j) \notin \Phi$, $b_{\{i,j\}} = c_{\{i,j\}} = \eta_{\{i,j\}} = 0$ and $\forall (i,j) \in \Phi$, we have $\phi_{\{i,j\}} \in \{0,1\}$ and so, $\phi_{\{i,j\}}^2 = \phi_{\{i,j\}}$. Therefore,[11]

$$\begin{aligned}e\,b_{\{i,j\}} + c_{\{i,j\}} &= e\,a_{\{i,j\}}(1 - 2\,\phi_{\{i,j\}}) - a_{\{i,j\}}^2\\ &= e\,a_{\{i,j\}}(1 - 2\,\phi_{\{i,j\}}) - a_{\{i,j\}}^2 + e^2(\phi_{\{i,j\}} - \phi_{\{i,j\}}^2)\\ &= (e\,\phi_{\{i,j\}} + a_{\{i,j\}})(e - (e\,\phi_{\{i,j\}} + a_{\{i,j\}})) = \beta_{\{i,j\}}(e - \beta_{\{i,j\}}) = \eta_{\{i,j\}}.\end{aligned}$$

- *Special honest verifier zero-knowledge*. If the challenge $e$ is known. A simulator can produce a valid transcript that is identically distributed to a transcript created by an honest prover, as follows.
  - (i) For every $(i,j) \in \Pi$, set $\beta_{\{i,j\}} := e\,f(i,j)$ and $\forall (i,j) \notin \Pi$, sample $\beta_{\{i,j\}} \xleftarrow{\$} \mathbb{Z}_p$.
  - (ii) For every $s \in \Theta$, pick some $(i_s, j_s) \in \Lambda_s$ such that $g_s(i_s, j_s) \neq 0$ and modify the value of $\beta_{\{i,j\}}$ so that $\sum_{(i,j)\in\Lambda_s} g_s(i,j)\beta_{\{i,j\}} = \gamma_s\, e$.
  - (iii) Define $\eta_{\{i,j\}}$ as $\eta_{\{i,j\}}(e - \eta_{\{i,j\}})$ for every $(i,j) \in \Phi$ and 0 otherwise.
  - (iv) Sample matrices $\Gamma_i$, $\Psi_i$, $S_i$ uniformly from $\mathbb{Z}_p^{(k+1)\times(k+1)}$ for every $i \in [m]$.
  - (v) Set $B_i := [\![S_i\, B]\!]_2$ for every $i \in [m]$.

[11] By linearity of the commitments.

(vi) Set $A_i := M_i^{-e} \circ \mathsf{Com}_{\mathsf{mpk}}(\beta_{\{i,1\}}, \ldots, \beta_{\{i,n\}}; \Gamma_i)$ for every $i \in [m]$.
(vi) Set $C_i := B_i^{-e} \circ \mathsf{Com}_{\mathsf{mpk}}(\eta_{\{i,1\}}, \ldots, \eta_{\{i,n\}}; \Psi_i)$ for every $i \in [m]$.

Note that the simulation is perfect because in both a real transcript and the simulated transcript variables $\beta_{\{i,j\}}$ for all $(i,j) \notin \Pi$ and matrices $\Gamma_i$ and $\Psi_i$ for all $i \in [m]$ are independent and uniformly random, modulo the condition $\sum_{(i,j)\in\Lambda_s} g_s(i,j)\beta_{\{i,j\}} = \gamma_s e$ for all $s \in \Theta$. Furthermore, the value of $\beta_{\{i,j\}}$ for all $(i,j) \in \Pi$ is completely determined, as well as matrices $A_i$ and $C_i$ for all $i \in [m]$, in both kinds of proof.

- *3-special soundness*. Note that two answers, say $\{\beta_{\{i,j\}}\}_{i,j}$, $\{\Gamma_i, \Psi_i\}_{i\in[m]}$ and $\{\hat{\beta}_{\{i,j\}}\}_{i,j}$, $\{\hat{\Gamma}_i, \hat{\Psi}_i\}_{i\in[m]}$ to 2 different challenges, say $e$ and $\hat{e}$ (respectively), with the same first message lead to a valid witness. It is enough to set

$$\phi_{\{i,j\}} := \frac{\beta_{\{i,j\}} - \hat{\beta}_{\{i,j\}}}{e - \hat{e}} \ \forall (i,j) \in [m]\times[n] \quad \text{and} \quad Z_i = \frac{\Gamma_i - \hat{\Gamma}_i}{e - \hat{e}} \ \forall i \in [m]$$

Note that such an opening satisfies,

$$\forall (i,j) \in \Pi,\ \phi_{\{i,j\}} = \frac{\beta_{\{i,j\}} - \hat{\beta}_{\{i,j\}}}{e - \hat{e}} = \frac{e\, f(i,j) - \hat{e}\, f(i,j)}{e - \hat{e}} = f(i,j)$$

Furthermore,

$$\forall s \in \Theta,\ \sum_{(i,j)\in\Lambda_s} g_s(i,j)\phi_{\{i,j\}} = \sum_{(i,j)\in\Lambda_s} g_s(i,j)\frac{(\beta_{\{i,j\}} - \hat{\beta}_{\{i,j\}})}{e - \hat{e}} = \gamma_s$$

Finally, the last verification equation tells us that for every $(i,j) \in \Phi$, the following polynomial in $x$

$$x^2(\phi_{\{i,j\}} - \phi_{\{i,j\}}^2) + x(a_{\{i,j\}}(1 - 2\,\phi_{\{i,j\}} - b_{\{i,j\}}) - a_{\{i,j\}}^2 - c_{\{i,j\}}$$

evaluates to 0 in both $e$ and $\hat{e}$. If there existed a third valid answer to a different challenge but the first same message, the polynomial would have three roots, but its degree is at most 2. Therefore, it must be the zero polynomial ($\mathbb{Z}_p$ is a field), and thus, the coefficient for $x^2$ must be zero: $\phi_{\{i,j\}} \in \{0, 1\}$. □

## References


1. 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.) ACM CCS 2017, pp. 647–664. ACM Press (October/November 2017)
2. Ambrona, M., Gilles, B., Benedikt, S.: Generic transformations of predicate encodings: constructions and applications. In: Jonathan, K., Hovav, S. (eds.) CRYPTO 2017, Part I, LNCS, vol. 10401, pp. 36–66. Springer, Heidelberg (2017)
3. Agrawal S., Chase M.: A study of pair encodings: predicate encryption in prime order groups. In: Kushilevitz E., Malkin T. (eds.) TCC 2016-A, Part II, LNCS, vol. 9563, pp. 259–288. Springer, Heidelberg (2016).
4. Agrawal, S., Chase, M.: Simplifying design and analysis of complex predicate encryption schemes. In: Coron, J.-S., Nielsen, J.B. (eds.) EUROCRYPT 2017, Part I, LNCS, vol. 10210, pp. 627–656. Springer, Heidelberg (April/May 2017)
5. Aranha, D.F., Gouvêa, C.P.L., Markmann, T., Wahby, R.S., Liao, K.: RELIC is an Efficient LIbrary for Cryptography. https://github.com/relic-toolkit/relic
6. Akinyele, J.A., Lehmann, C.U., Green, M.D., Pagano, M.W., Peterson, Z.N.J., Rubin, A.D.: Self-protecting electronic medical records using attribute-based encryption. Cryptology ePrint Archive, Report 2010/565, 2010. http://eprint.iacr.org/2010/565

7. 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 2014, LNCS, vol. 8441, pp. 557–577. Springer, Heidelberg (2014).
8. Attrapadung, N.: Dual system encryption framework in prime-order groups via computational pair encodings. In: J.H. Cheon, T. Takagi (eds.) ASIACRYPT 2016, Part II, LNCS, vol. 10032, pp. 591–623. Springer, Heidelberg (December 2016)
9. Attrapadung N.: Unbounded dynamic predicate compositions in attribute-based encryption. In: Ishai Y., Rijmen V. (eds.) EUROCRYPT 2019, Part I, LNCS, vol. 11476, pp. 34–67. Springer, Heidelberg (2019).
10. Attrapadung N., Yamada S.: Duality in ABE: converting attribute based encryption for dual predicate and dual policy via computational encodings. In: Nyberg K. (ed.) CT-RSA 2015, LNCS, vol. 9048, pp. 87–105. Springer, Heidelberg (2015).
11. Boneh D., Boyen X.: Efficient selective-ID secure identity based encryption without random oracles. In: Cachin C., Camenisch J. (eds.) EUROCRYPT 2004, LNCS, vol. 3027, pp. 223–238. Springer, Heidelberg (2004).
12. Bootle, J., Cerulli, A., Chaidos, P., Ghadafi, E., Groth, J., Petit, C.: Short accountable ring signatures based on DDH. In: Pernul, G., Ryan, EP.Y.A., Weippl, R. (eds.) ESORICS 2015, Part I, LNCS, Vol. 9326, pp. 243–265. Springer, Heidelberg (September 2015)
13. Blum, M., Feldman, P., Micali, S. Non-interactive zero-knowledge and its applications (extended abstract). In: 20th ACM STOC, pp. 103–112. ACM Press (May 1988)
14. Bellare M., Goldreich O.: On defining proofs of knowledge. In: Brickell E.F. (ed.) CRYPTO'92, LNCS, vol. 740, pp. 390–420. Springer, Heidelberg (1993).
15. Barreto P.S.L.M., Naehrig M.: Pairing-friendly elliptic curves of prime order. In: Preneel B., Tavares S. (eds.) SAC 2005, LNCS, vol. 3897, pp. 319–331. Springer, Heidelberg (2006).
16. Bethencourt, J., Sahai, A., Waters, B.: Ciphertext-policy attribute-based encryption. In: 2007 IEEE Symposium on Security and Privacy, pp. 321–334. IEEE Computer Society Press (May 2007)
17. Boneh D., Waters B.: Conjunctive, subset, and range queries on encrypted data. In: Vadhan S.P. (ed.) TCC 2007, LNCS, vol. 4392, pp. 535–554. Springer, Heidelberg (2007).
18. Camenisch, J., Dubovitskaya, M., Neven, G.: Oblivious transfer with access control. In: Al-Shaer, E., Jha, S., Keromytis, A.D. (eds.) ACM CCS 2009, pp. 131–140. ACM Press (November 2009)
19. Coull S.E., Green M., Hohenberger S.: Controlling access to an oblivious database using stateful anonymous credentials. In: Jarecki S., Tsudik G. (eds.) PKC 2009, LNCS, vol. 5443, pp. 501–520. Springer, Heidelberg (2009).
20. Chen J., Gay R., Wee H.: Improved dual system ABE in prime-order groups via predicate encodings. In: Oswald E., Fischlin M. (eds.) EUROCRYPT 2015, Part II, LNCS, vol. 9057, pp. 595–624. Springer, Heidelberg (2015).
21. Canard, S., Hamdi, A., Laguillaumie, F.: Blind functional encryption. In: ICICS (2020)
22. Cheung, L., Newport, C.C.: Provably secure ciphertext policy ABE. In: Ning, P., Capitani di Vimercati, S.D., Syverson, P.F. (eds.) ACM CCS 2007, pp. 456–465. ACM Press (October 2007)
23. Chen J., Wee H.: Fully, (almost) tightly secure IBE and dual system groups. In: Canetti R., Garay J.A. (eds.) CRYPTO 2013, Part II, LNCS, vol. 8043, pp. 435–460. Springer, Heidelberg (2013).
24. Chen, J., Wee, H.: Dual system groups and its applications—compact HIBE and more. Cryptology ePrint Archive, Report 2014/265, 2014. http://eprint.iacr.org/2014/265
25. Dinh, T.T.A., Datta, A.:. Streamforce: outsourcing access control enforcement for stream data to the clouds. In: Fourth ACM Conference on Data and Application Security and Privacy, CODASPY'14, San Antonio, TX, USA - March 03–05, 2014, pp. 13–24 (2014)
26. Escala A., Herold G., Kiltz E., Ràfols C., Villar J.: An algebraic framework for Diffie-Hellman assumptions. In: Canetti R., Garay J.A. (eds.) CRYPTO 2013, Part II, LNCS, vol. 8043, pp. 129–147. Springer, Heidelberg (2013).
27. Fiat A., Naor M.: Broadcast encryption. In: Stinson D.R. (ed.) CRYPTO'93, LNCS, vol. 773, pp. 480–491. Springer, Heidelberg (1994).
28. Green M., Hohenberger S.: Blind identity-based encryption and simulatable oblivious transfer. In: Kurosawa K. (ed.) ASIACRYPT 2007, LNCS, vol. 4833, pp. 265–282. Springer, Heidelberg (2007).
29. Groth J., Kohlweiss M.: One-out-of-many proofs: or how to leak a secret and spend a coin. In: Oswald E., Fischlin M. (eds.) EUROCRYPT 2015, Part II, LNCS, vol. 9057, pp. 253–280. Springer, Heidelberg (2015).
30. Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof-systems (extended abstract). In: 17th ACM STOC, pp. 291–304. ACM Press (May 1985)
31. Goldwasser S., Micali S., Rackoff C.: The knowledge complexity of interactive proof systems. SIAM J. Comput. **18**(1), 186–208 (1989).

32. Goldreich O., Micali S., Wigderson A.: How to prove all NP-statements in zero-knowledge, and a methodology of cryptographic protocol design. In: Odlyzko A.M. (ed.) CRYPTO'86, LNCS, vol. 263, pp. 171–185. Springer, Heidelberg (1987).
33. 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., Capitani di Vimercati, S.D. (eds.) ACM CCS 2006, pp. 89–98. ACM Press (October/November 2006). Available as Cryptology ePrint Archive Report 2006/309.
34. Katz J., Sahai A., Waters B.: Predicate encryption supporting disjunctions, polynomial equations, and inner products. In: Smart N.P. (ed.) EUROCRYPT 2008, LNCS, vol. 4965, pp. 146–162. Springer, Heidelberg (2008).
35. Karchmer, M., Wigderson, A.: On span programs. In: Proceedings of Structures in Complexity Theory, pp. 102–111 (1993)
36. Lewko A.B.: Tools for simulating features of composite order bilinear groups in the prime order setting. In: Pointcheval D., Johansson T. (eds.) EUROCRYPT 2012, LNCS, vol. 7237, pp. 318–335. Springer, Heidelberg (2012).
37. Lewko, A.B., Okamoto, T., Sahai, A., Takashima, K., Waters, B.: Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption. In: Gilbert, H. (ed.) EUROCRYPT 2010, LNCS, vol. 6110, pp. 62–91. Springer, Heidelberg (May/June 2010)
38. Lewko A.B., Waters B.: New techniques for dual system encryption and fully secure HIBE with short ciphertexts. In: Micciancio D. (ed.) TCC 2010, LNCS, vol. 5978, pp. 455–479. Springer, Heidelberg (2010).
39. Lewko A.B., Waters B.: Unbounded HIBE and attribute-based encryption. In: Paterson K.G. (ed.) EUROCRYPT 2011, LNCS, vol. 6632, pp. 547–567. Springer, Heidelberg (2011).
40. Morillo, P., Ràfols, C., Villar, J.L.: The kernel matrix Diffie-Hellman assumption. In: Cheon, J.H., Takagi, T., (eds.) ASIACRYPT 2016, Part I, LNCS, vol. 10031, pp. 729–758. Springer, Heidelberg (December 2016)
41. Ostrovsky, R., Sahai, A., Waters, B.: Attribute-based encryption with non-monotonic access structures. In: Ning, P., Capitani di Vimercati, S.D., Syverson, P.F. (eds.) ACM CCS 2007, pp. 195–203. ACM Press (October 2007)
42. Rabin, M.: How to exchange secrets by oblivious transfer. Technical Report Tech. Memo TR-81, Aiken Computation Laboratory (1981)
43. Rial A.: Blind attribute-based encryption and oblivious transfer with fine-grained access control. Des. Codes Cryptogr. **81**(2), 179–223 (2016).
44. Rouselakis, Y., Waters, B.: Practical constructions and new proof methods for large universe attribute-based encryption. In: Sadeghi, A.-R., Gligor, V.D., Yung, M., (eds.) ACM CCS 2013, pp. 463–474. ACM Press (November 2013)
45. Shamir, A.: Identity-based cryptosystems and signature schemes. In: Blakley, G.R., Chaum, D. (eds.) CRYPTO'84, LNCS, vol. 196, pp 47–53. Springer, Heidelberg (August 1984)
46. Sahai A., Waters B.R.: Fuzzy identity-based encryption. In: Cramer R. (ed.) EUROCRYPT 2005, LNCS, vol. 3494, pp. 457–473. Springer, Heidelberg (2005).
47. Waters B.: Dual system encryption: Realizing fully secure IBE and HIBE under simple assumptions. In: Halevi S. (ed.) CRYPTO 2009, LNCS, vol. 5677, pp. 619–636. Springer, Heidelberg (2009).
48. Waters B.: Functional encryption for regular languages. In: Safavi-Naini R., Canetti R. (eds.) CRYPTO 2012, LNCS, vol. 7417, pp. 218–235. Springer, Heidelberg (2012).
49. Wee H.: Dual system encryption via predicate encodings. In: Lindell Y. (ed.) TCC 2014, LNCS, vol. 8349, pp. 616–637. Springer, Heidelberg (2014).
50. Wang, F., Mickens, J., Zeldovich, N., Vaikuntanathan, V.: Sieve: Cryptographically enforced access control for user data in untrusted clouds. In: 13th USENIX Symposium on Networked Systems Design and Implementation (NSDI 16), pp. 611–626. Santa Clara, CA, March 2016. USENIX Association.