Generic Transformations of Predicate Encodings: Constructions and Applications

Miguel Ambrona, Gilles Barthe, and Benedikt Schmidt

IMDEA Software Institute, Madrid, Spain
{miguel.ambrona,gilles.barthe,benedikt.schmidt}@imdea.org

Abstract. Predicate encodings are information-theoretic primitives that can be transformed generically into predicate encryption schemes for a broad class of predicates (Wee, TCC 2014; Chen, Gay, Wee, EUROCRYPT 2015). Starting from the observation that predicate encodings admit a purely algebraic characterization equivalent to the original notion, we obtain several new results about predicate encodings. First, we propose two generic optimizations that improve their efficiency. Second, we propose new transformations for boolean combinations of predicate encodings. Third, we develop several new predicate encodings for boolean formulas and arithmetic span programs. In the important case of boolean formulas, our encodings are more efficient and attribute-hiding; moreover, they support revocation and delegation. Finally, we implement our approach and experimentally validate its efficiency gains.

1 Introduction

Predicate Encryption (PE) [11, 23] is a form of public-key encryption that supports fine-grained access control for encrypted data by associating attributes to ciphertexts and keys. In predicate encryption, everyone can create ciphertexts while keys can only be created by the master key owner. Predicate encryption schemes use predicates to model (potentially complex) access control policies and attributes are attached to both ciphertexts and secret keys. A predicate encryption scheme for a predicate P guarantees that decryption of a ciphertext c with a secret key k is allowed if and only if the attribute x associated to the ciphertext c and the attribute y associated to the secret key k verify the predicate P, i.e. $\mathsf{P}(x, y) = 1$. Predicate encryption schemes exist for several useful predicates, such as Zero Inner Product Encryption (ZIPE), where attributes are vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ and the predicate $\mathsf{P}(\boldsymbol{x}, \boldsymbol{y})$ is defined as $\boldsymbol{x}^\top \boldsymbol{y} = 0$. Predicate encryption subsumes several previously defined notions of public-key encryption. For example, Identity-Based Encryption (IBE) [31] can be obtained by defining $\mathsf{P}(x, y)$ as $x = y$ and Attribute-Based Encryption (ABE) [30] can be obtained similarly. More concretely, for Key-Policy ABE (KP-ABE), the attribute x is a boolean vector, the attribute y is a boolean function, and the predicate $\mathsf{P}(x, y)$ is defined as $y(x) = 1$. For Ciphertext-Policy ABE (CP-ABE), the roles of the attributes x and y are swapped.

Modular approaches for PE. In 2014, two independent works by Wee [34] and Attrapadung [4] proposed generic and unifying frameworks for obtaining efficient fully secure PE schemes for a large class of predicates. Both works use the dual system techniques introduced by Lewko and Waters [25, 33] and define a compiler that takes as input a relatively simple symmetric primitive and produces a fully secure PE construction. Wee introduced so-called *predicate encodings*, an information-theoretic primitive inspired from linear secret sharing. Attrapadung introduced so-called *pair encodings* and provides computational and information-theoretic security notions. These approaches greatly simplify the construction and analysis of predicate encryption schemes and share several advantages. First, they provide a good trade-off between *expressivity* and *performance*, while the security relies on standard and well studied assumptions. Second, they unify existing constructions into a single framework, i.e., previous PE constructions can be seen as instantiations of these new compilers with certain encodings. Third, building PE schemes by analyzing and building these simpler encodings is much easier than building PE schemes directly. Compared to full security for PE, the encodings must verify much weaker security requirements. The power of pair and predicate encodings is evidenced by the discovery of new constructions and efficiency improvements over existing ones. However, both approaches were designed over *composite order* bilinear groups. In 2015, Chen, Gay and Wee [12] and Attrapadung [5] respectively extended the predicate encoding and pair encoding compiler to the *prime order* setting. Next, Agrawal and Chase [1] improved on Attrapadung's work by relaxing the security requirement on *pair encodings* and thus, capturing new constructions. In addition, their work also brings both generic approaches closer together, since like [12], the new compiler relies (in a black-box way) on Dual System Groups (DSG) [13, 14].

Predicate encryption from predicate encodings. A predicate encoding for a predicate P consist of encoding functions sE, rE, and kE used for encryption and key generation, and decoding functions used for decryption. In the following, we try to provide some intuition on how predicate encodings are compiled to predicate encryption schemes by the compiler from [12]. The master keys, ciphertexts and secret keys have the following form:

$$\mathsf{msk} = g_2^{\alpha} \qquad \mathsf{ct}_x = (g_1^s, g_1^{s \cdot \mathsf{sE}_x(\boldsymbol{w})}, e(g_1, g_2)^{\alpha s} \cdot m)$$
$$\mathsf{mpk} = (g_1, g_1^{\boldsymbol{w}}, g_2, g_2^{\boldsymbol{w}}, e(g_1, g_2)^{\alpha}) \qquad \mathsf{sk}_y = (g_2^r, g_2^{\alpha \cdot \mathsf{kE}_y + r \cdot \mathsf{rE}_y(\boldsymbol{w})})$$

Note that the encrypted message $m \in \mathbb{G}_t$ is blinded by a random factor $e(g_1, g_2)^{\alpha s}$. The so-called (restricted) α-reconstruction property of predicate encodings ensures that this blinding factor can be recovered for a pair $(\mathsf{ct}_x, \mathsf{sk}_y)$ if $\mathsf{P}(x, y) = 1$. More concretely, restricted α-reconstruction states that, for all x, y with $\mathsf{P}(x, y) = 1$, there exist linear decoding functions $\mathsf{sD}_{x,y}$,$\mathsf{rD}_{x,y}$ such that for all vectors $\boldsymbol{w}$,

$$\mathsf{sD}_{x,y}(\mathsf{sE}_x(\boldsymbol{w})) = \mathsf{rD}_{x,y}(\mathsf{rE}_y(\boldsymbol{w})) \ \wedge \ \mathsf{rD}_{x,y}(\mathsf{kE}_y) = 1.$$

Since decoding functions are linear, it is possible to evaluate these functions in the exponent. To see correctness, note that if such functions are known, one can compute

$$g_1^{s\cdot \mathsf{sD}_{x,y}(\mathsf{sE}_x(\boldsymbol{w}))} \quad \text{and} \quad g_2^{\mathsf{rD}_{x,y}(\alpha\cdot \mathsf{kE}_y + r\cdot \mathsf{rE}_y(\boldsymbol{w}))},$$

obtaining $g_1^{s\beta}$ and $g_2^{\alpha+r\beta}$ for $\beta = \mathsf{sD}_{x,y}(\mathsf{sE}_x(\boldsymbol{w})) = \mathsf{rD}_{x,y}(\mathsf{rE}_y(\boldsymbol{w}))$. Now, it is simple to recover $e(g_1, g_2)^{\alpha s}$ from $e(g_1^s, g_2^{\alpha+r\beta})$ and $e(g_1^{s\beta}, g_2^r)$. To prove security, another property of predicate encodings called α-hiding is used together with decisional assumptions about dual system groups. The α-hiding property states that given certain values derived from the output of the encoding functions for random input, α remains information-theortic hidden.

1.1 Our contributions

We pursue the line of investigations on the strength of (predicate) encodings and establish several general results and new constructions that broaden their scope and improve their efficiency.

Algebraic characterization and optimizations. Our starting point is the observation that the information-theoretic definition of predicate encoding admits an equivalent, purely algebraic, definition. More specifically, we observe that the information-theoretic indistinguishability property used in [12, 34] for capturing security of a predicate encoding is equivalent to the existence of a solution of a linear system of equations. Leveraging this observation, we prove that every triple of encoding functions implicitly defines a unique predicate for which it is a valid predicate encoding. This result sheds a different view on predicate encodings and simplifies the analysis and description of all subsequent results in the paper. In turn, this result leads us to formulate two generic optimizations of predicate encodings that can lead to efficiency improvements and reduce the number of required group elements in keys and ciphertexts. We prove the soundness of the transformations and validate their benefits experimentally on examples from [12, 34]; we successfully apply these simplifications to reduce the size of keys and ciphertexts by up to 50% and to reduce the number of group operations needed in some of the existing encodings.

Combination of predicate encodings. We propose generic methods for building boolean combinations of predicate encodings. Given predicate encodings for two predicates P_1 and P_2, our results give us valid predicate encodings for the *disjunction* predicate $\mathsf{P}_1(x_1, y_1) \vee \mathsf{P}_2(x_2, y_2)$ and for the *conjunction* predicate $\mathsf{P}_1(x_1, y_1) \wedge \mathsf{P}_2(x_2, y_2)$. Furthermore, given a predicate encoding for predicate P, we can produce a predicate encoding for the *negation* $\neg\mathsf{P}(x, y)$ and the dual predicate $\mathsf{P}^\star(y, x) := \mathsf{P}(x, y)$. Moreover, we present a method to flexibly combine the above transformations on predicate encodings. For example, given predicates P_1, P_2 and P_3 we can create a predicate encoding for the predicate $\mathsf{P}_1 \bowtie (\mathsf{P}_2 \bowtie \mathsf{P}_3)$ where the placeholders $\bowtie \in \{\wedge, \vee\}$ are part of the attribute y, i.e., different keys

can enforce different boolean structure on the predicates P_1, P_2 and P_3. Additionally, the transformation to build the dual encoding can be used to convert any KP-ABE scheme into CP-ABE and vice versa for dually related predicates and thus, it unifies both notions in the predicate encoding framework.

Applications. Leveraging on these results, we develop several new applications:

- We show how to combine our generic transformations to build Dual-Policy Attribute-Based Encryption (DP-ABE) [7, 8] from predicate encodings (Section 5.4). DP-ABE combines KP-ABE and CP-ABE into a single construction that allows simultaneously for two access control mechanisms. This primitive has not been considered before in the predicate encoding framework.
- We show how combining a predicate encoding for *broadcast encryption* with another arbitrary predicate encoding is useful to achieve direct revocation of keys (Section 5.5). The former encoding takes care of revocation, while the latter encodes the desired access structure. We also perform an experimental evaluation of the scalability of revocation through this mechanism.
- We extend the attribute-hiding predicate encryption for ZIPE proposed in [12] to non-monotonic boolean formulas (expressed in Disjunctive Normal Form), using our combinations on predicate encodings.
- We define a new methodology that is based on our transformations and optimizations of predicate encodings and that has been successfully applied to simplify existing predicate encodings from [12], building simpler and more efficient predicate encodings with new properties (Section 5.1).
- Finally, we show how to enhance any predicate encoding with support for delegation. In particular, we show how this method can be applied to a predicate encoding for boolean formulas (Section 5.6).

Implementation and evaluation. We implement[1] a general library for predicate encryption with support for the predicate encoding and pair encoding frameworks. We perform a comparison between *predicate encodings* and *pair encodings* in terms of efficiency and number of group operations. Although predicate encodings seem to be more efficient, further investigation is required to obtain a better understanding of these two primitives. Additionally, we implement our constructions and perform an experimental evaluation of their efficiency. We confirm that our improvements on existing predicate encodings have the expected performance. Our scalability experiments show that predicate encodings can be used for real applications.

1.2 Prior work

Predicate encodings have been introduced in [34] and we use a refined version that is defined in [12] as our starting point. Both variants use an information-theoretic definition of the hiding while we show that there is an equivalent algebraic definition. Another related work is [17], initiating a systematic study

[1] source code at https://github.com/miguel-ambrona/abe-relic

of the communication complexity of the so-called conditional secret disclosure primitive, which is closely related to predicate encodings.

Other works also optimize existing predicate encryption schemes, for example many works focus on going from composite order constructions to the more efficient prime order ones [5, 12, 24]. In [12] they also propose performance improvements on Dual System Groups. We believe our optimizations via predicate encoding complement other possible enhancements of predicate encryption. Additionally, a recent work [19] introduces a new generic framework for modular design of predicate encryption that improves on the performance of existing compilers. Their core primitive, *tag based encodings* is really similar to *predicate encodings* and we believe our techniques could be also applied in this new framework.

Boolean combinations of predicates have also been considered in the setting of pair encodings. Attrapadung [7, 8] proposes generic transformations for conjunction and for the dual predicate, but neither for negation nor disjunction. To our best knowledge, we are the first to propose similar constructions for predicate encodings; additionally, we also deal with negation and disjunction.

The main advantage of DP-ABE is the possibility of considering policies over *objective* attributes (associated to data) and policies over *subjective* attributes (associated to user credentials) at the same time. DP-ABE has been considered by Attrapadung in the pair encoding framework [7, 8]. To the best of our knowledge, we are the first to provide DP-ABE in the predicate encoding framework.

Revocation is a desirable property for PE and ABE schemes that has also been considered by many works in the literature. Revocation allows to invalidate a user's secret key in such a way that it becomes useless, even if its associated attribute satisfies the policy associated to the ciphertext. Some attempts [29] propose *indirect* revocation that requires that the master secret owner periodically updates secret keys for non-revoked users. Other attempts achieve *direct* revocation [6, 20, 27, 28], but either rely on strong assumptions or provide only selectively security. Our construction not only allows to achieve revocation in a *fully secure* framework, but it allows to add revocation to arbitrary predicate encodings.

Policy hiding is another property of PE, and ABE in particular, that has been broadly studied. In this context, policies associated to ciphertexts are not attached to them and therefore, unauthorized users will only learn the fact that their key does not satisfy the policy, but nothing else about it. Policy Hiding has been considered in several works [11, 23]. The security of our construction improves on earlier works, thanks to the compiler from [12]. Additionally, our observation extends the expressivity of attribute-hiding predicate encryption for ZIPE proposed in [12] to support policy-hiding for boolean formulas.

In [12], the authors introduce the notion for *spatial encryption* predicate encodings. We generalize this notion and introduce a transformation that makes delegation possible for every predicate encoding.

Several works evaluate the suitability of ABE for different applications. For example, ABE has been used and benchmarked to enforce privacy of Electronic

Sender encoding:	$\mathsf{sE} : \mathcal{X} \to \mathcal{W} \to \mathcal{S}$
Receiver encoding:	$\mathsf{rE} : \mathcal{Y} \to \mathcal{W} \to \mathcal{R}$
Key encoding:	$\mathsf{kE} : \mathcal{Y} \to \mathcal{R}$
Sender decoding:	$\mathsf{sD} : \mathcal{X} \times \mathcal{Y} \to \mathcal{S} \to \mathbb{Z}_p$
Receiver decoding:	$\mathsf{rD} : \mathcal{X} \times \mathcal{Y} \to \mathcal{R} \to \mathbb{Z}_p$

Fig. 1. Predicate encoding functions and their types.

Medical Records (EMR) [2], in a system where healthcare organizations export EMRs to external storage locations. Other examples are Sieve [32] or Streamforce [15], systems that provide enforced access control for user data and stream data in untrusted clouds. In contrast to these works, we are the first to evaluate predicate encryption and ABE based on modern modular approaches such as the predicate encoding and pair encoding frameworks. The resulting schemes also satisfy a stronger security notion (full vs. selective security) compared to the previously mentioned evaluations. We focus on synthetic case studies, while other works analyze more realistic settings and integration of ABE into bigger systems. Combining our methods with these more practical case studies is a very interesting direction for future work.

2 Background

In this section, we first introduce some mathematical notation and then define predicate encodings, the symmetric primitive used to build predicate encryption.

2.1 Notation

For finite sets S, we use $x \xleftarrow{\$} S$ to denote that x is uniformly sampled from S. For a predicate $\mathsf{P} : \mathcal{X} \times \mathcal{Y} \to \{0,1\}$, we use $(x,y) \in \mathsf{P}$ as a shorthand for $\mathsf{P}(x,y) = 1$. For two functions $f : A \to A'$ and $G : B \to B'$, we define the *cartesian product of f and g* as $(f \times g)(a,b) = (f(a), g(b))$. For $S \subseteq A$ and $f : A \to A'$, we define $f(S) = \{f(s) \mid s \in S\}$ to denote the *image of S under f*. For a finite-dimensional $\mathbb{Z}_p$-vector space $\mathcal{V}$, we define $l_{\mathcal{V}} = \dim(\mathcal{V})$ and use $\mathbf{0}_{\mathcal{V}}$ to denote the zero vector in $\mathcal{V}$. We also assume that $\mathcal{V} = \mathbb{Z}_p^{l_{\mathcal{V}}}$ unless stated otherwise. Furthermore, we use the same conventions for matrix-representations of linear maps on finite-dimensional spaces. We define the *transpose* ${}^{\top}f$ of such maps using the transposed matrix for f. We use $\mathsf{diag}(\boldsymbol{v})$ to denote the diagonal matrix with main diagonal $\boldsymbol{v}$.

2.2 Predicate Encodings

Predicate encodings are a symmetric primitive introduced in [34] that can be used to build predicate encryption schemes for different predicates. We use the refinement of predicate encodings introduced in [12].

Definition 1 (Predicate encoding). *A $\mathbb{Z}_p$-bilinear predicate encoding for a predicate $\mathsf{P} : \mathcal{X} \times \mathcal{Y} \to \{0,1\}$ consists of deterministic algorithms $(\mathsf{sE}, \mathsf{rE}, \mathsf{kE}, \mathsf{sD}, \mathsf{rD})$ with the types given in Figure 1 such that $\mathcal{W}$, $\mathcal{S}$, $\mathcal{R}$ are finite-dimensional vector spaces over $\mathbb{Z}_p$ and the following properties are satisfied:*

Linearity: *sE_x, rE_y, $\mathsf{sD}_{x,y}$, and $\mathsf{rD}_{x,y}$ are $\mathbb{Z}_p$-linear.*

Restricted α-reconstruction: *For all $(x,y) \in \mathsf{P}, \boldsymbol{w} \in \mathcal{W}$:*

$$\mathsf{sD}_{x,y}(\mathsf{sE}_x(\boldsymbol{w})) = \mathsf{rD}_{x,y}(\mathsf{rE}_y(\boldsymbol{w})) \text{ and } \mathsf{rD}_{x,y}(\mathsf{kE}_y) = 1$$

α-privacy: *For all $(x,y) \notin \mathsf{P}, \alpha \in \mathbb{Z}_p$, it holds that*

$$\boldsymbol{w} \xleftarrow{\$} \mathcal{W}; \mathit{return}\ (\mathsf{sE}_x(\boldsymbol{w}), \mathsf{rE}_y(\boldsymbol{w}) + \alpha \cdot \mathsf{kE}_y) \equiv \boldsymbol{w} \xleftarrow{\$} \mathcal{W}; \mathit{return}\ (\mathsf{sE}_x(\boldsymbol{w}), \mathsf{rE}_y(\boldsymbol{w}))$$

where $\equiv$ denotes equality of distributions.

In the remainder of the paper, we say the encoding and decoding functions of predicate encodings are *linear* if they satisfy the *linearity* property. We also abbreviate the *restricted α-reconstruction* property with *reconstructability* and the *α-privacy* property with *privacy*. The reconstructability property allows to recover α from $(x, y, \mathsf{sE}_x(\boldsymbol{w}), \mathsf{rE}_y(\boldsymbol{w}) + \alpha \cdot \mathsf{kE}_y)$ if $(x,y) \in \mathsf{P}$. The α-privacy property ensures that α is perfectly hidden for such tuples if $(x,y) \notin \mathsf{P}$.

Example 1 (IBE predicate encoding). Let $\mathcal{X} = \mathcal{Y} = \mathcal{S} = \mathcal{R} = \mathbb{Z}_p$ and let $\mathcal{W} = \mathbb{Z}_p^2$. We define the encoding functions as follows:

$$\mathsf{sE}_x(w_1, w_2) = x \cdot w_1 + w_2 \qquad \mathsf{rE}_y(w_1, w_2) = y \cdot w_1 + w_2 \qquad \mathsf{kE}_y = 1$$

We can equivalently define the encoding functions using matrix notation:

$$\mathsf{sE}_x = \begin{pmatrix} x & 1 \end{pmatrix} \qquad \mathsf{rE}_y = \begin{pmatrix} y & 1 \end{pmatrix} \qquad \mathsf{kE}_y = \begin{pmatrix} 1 \end{pmatrix}.$$

The above is a predicate encoding for identity-based encryption, i.e., for the predicate $\mathsf{P}(x,y) := x = y$. In the next section, we will see that we can check this using the implicit predicate of the encoding (see Definition 2)

$$\begin{pmatrix} 0 \\ 1 \end{pmatrix} \notin \begin{pmatrix} x & 1 \\ y & 1 \end{pmatrix} (\mathcal{W})$$

which is equivalent to $x = y$. ■

3 Properties of predicate encodings

In this section, we present an alternative definition of predicate encodings. In this new definition, the α-privacy property (which was defined as an indistinguishability statement) becomes a purely algebraic statement. This makes it easy to analyze predicate encodings in a generic way. Furthermore, we unify the reconstructability and privacy properties and then exploit the unified treatment to define optimizations for predicate encodings. In the remainder of this section, we fix attribute sets $\mathcal{X}$ and $\mathcal{Y}$, finite-dimensional $\mathbb{Z}_p$-vector spaces $\mathcal{W}$, $\mathcal{S}$, and $\mathcal{R}$, and linear encoding functions $pe = (\mathsf{sE}, \mathsf{rE}, \mathsf{kE})$ for these types (see Figure 1).

3.1 Alternative definition of predicate encodings

We first define a predicate that can be inferred from the encoding functions.

Definition 2 (Implicit predicate). *The encoding functions* $pe = (\mathsf{sE}, \mathsf{rE}, \mathsf{kE})$ *define the* implicit predicate

$$\mathsf{Pred}_{pe}(x, y) := (\mathbf{0}_{\mathcal{S}}, \mathsf{kE}_y) \notin (\mathsf{sE}_x \times \mathsf{rE}_y)(\mathcal{W}).$$

We can relate the implicit predicate to privacy as follows.

Lemma 1 (Alternative definition of privacy). *The encoding functions* $pe = (\mathsf{sE}, \mathsf{rE}, \mathsf{kE})$ *satisfy the* α*-privacy property for all* $(x, y) \notin \mathsf{Pred}_{pe}$*. We say a function* $\mathsf{rW} : \mathcal{X} \times \mathcal{Y} \to \mathcal{W}$ *is a* witness for α-privacy *if for all* $(x, y) \notin \mathsf{Pred}_{pe}$*, it holds that* $(\mathsf{sE}_x \times \mathsf{rE}_y)(\mathsf{rW}_{x,y}) = (\mathbf{0}_{\mathcal{S}}, \mathsf{kE}_y)$*. Such a witness function always exists.*

Proof. Remember that for a predicate P, a predicate encoding satisfies α-privacy if for all $(x, y) \notin \mathsf{P}$ and $\alpha \in \mathbb{Z}_p$,

$$\boldsymbol{w} \xleftarrow{\$} \mathcal{W}; \mathsf{return}\ (\mathsf{sE}_x(\boldsymbol{w}), \mathsf{rE}_y(\boldsymbol{w}) + \alpha \cdot \mathsf{kE}_y) \equiv \boldsymbol{w} \xleftarrow{\$} \mathcal{W}; \mathsf{return}\ (\mathsf{sE}_x(\boldsymbol{w}), \mathsf{rE}_y(\boldsymbol{w})).$$

This is equivalent to the existence of a bijection $\rho_{x,y,\alpha}$ such that for all $\boldsymbol{w} \in \mathcal{W}$,

$$\mathsf{sE}_x(\boldsymbol{w}) = \mathsf{sE}_x(\rho_{x,y,\alpha}(\boldsymbol{w})) \quad \text{and} \quad \mathsf{rE}_y(\boldsymbol{w}) + \alpha \cdot \mathsf{kE}_y = \mathsf{rE}_y(\rho_{x,y,\alpha}(\boldsymbol{w})).$$

By linearity, this can be rewritten as

$$\mathsf{sE}_x(\rho_{x,y,\alpha}(\boldsymbol{w}) - \boldsymbol{w}) = \mathbf{0}_{\mathcal{S}} \quad \text{and} \quad \alpha \cdot \mathsf{kE}_y = \mathsf{rE}_y(\rho_{x,y,\alpha}(\boldsymbol{w}) - \boldsymbol{w}).$$

Now, the existence of such a bijection is completely equivalent to the existence of a solution in the following linear system on $\boldsymbol{w}^*$:

$$\mathsf{sE}_x(\boldsymbol{w}^*) = \mathbf{0}_{\mathcal{S}} \ \wedge \ \mathsf{rE}_y(\boldsymbol{w}^*) = \alpha \cdot \mathsf{kE}_y$$

To see this, note that if $\rho_{x,y,\alpha}$ is such a bijection, $\rho_{x,y,\alpha}(\boldsymbol{w}_0) - \boldsymbol{w}_0$ is a solution of the system for every $\boldsymbol{w}_0 \in \mathcal{W}$. On the other hand, if $\boldsymbol{w}^*$ is a solution of the system, the bijection $\rho_{x,y,\alpha}(\boldsymbol{w}) = \boldsymbol{w} + \boldsymbol{w}^*$ is a valid one. We have just proved that α-privacy is equivalent to the existence of a solution of the above linear system, which at the same time is equivalent to the existence of solution for the following modified system (independent of α):

$$\mathsf{sE}_x(\boldsymbol{w}^*) = \mathbf{0}_{\mathcal{S}} \ \wedge \ \mathsf{rE}_y(\boldsymbol{w}^*) = \mathsf{kE}_y$$

for every $(x, y) \notin \mathsf{P}$. This proves that α-privacy is equivalent to the existence of the witness function rW from the statement of the lemma (defined for every $(x, y) \notin \mathsf{P}$). The lemma follows from the observation that the predicate $\neg\, \mathsf{Pred}_{pe}$ is equivalent to the existence of such a witness function as well. □

Intuitively, we now prove that we can find decoding functions that work whenever α-privacy does not hold. More formally, this means that each pair (x, y), α is either perfectly hidden or can be reconstructed.

Lemma 2 (Alternative definition of reconstructability). *There exist decoding functions* $(\mathsf{sD}, \mathsf{rD})$ *for the encoding functions* $pe = (\mathsf{sE}, \mathsf{rE}, \mathsf{kE})$ *that achieve* α*-reconstruction for all* $(x, y) \in \mathsf{Pred}_{pe}$*. Furthermore,* $(\mathsf{sD}, \mathsf{rD})$ *can be efficiently computed by Gaussian elimination.*

Proof. To prove Lemma 2 we will use the following helper lemma [10, Claim 2].

Lemma 3. *Let* K *be a field and let* $A \in K^{m \times n}$ *and* $\boldsymbol{b} \in K^n$ *be matrices with coefficients in* K*. The following two statements are equivalent:*

- $\forall \boldsymbol{a} \in \mathbb{Z}_p^m$ *it holds that* $\boldsymbol{b}^\top \neq \boldsymbol{a}^\top A$
- $\exists \boldsymbol{z} \in \mathbb{Z}_p^n$ *such that* $\boldsymbol{z}^\top \boldsymbol{b} = 1 \ \wedge \ A\boldsymbol{z} = \boldsymbol{0}$.

For all $(x, y) \in \mathsf{Pred}_{pe}$, we have

$$(\boldsymbol{0}_{\mathcal{S}}, \mathsf{kE}_y) \notin (\mathsf{sE}_x \times \mathsf{rE}_y)(\mathcal{W})$$

if and only if for all $\boldsymbol{w} \in \mathcal{W}$, it holds that

$$(\boldsymbol{0}_{\mathcal{S}}, \mathsf{kE}_y) \neq (\mathsf{sE}_x(\boldsymbol{w}), \mathsf{rE}_y(\boldsymbol{w})) \Leftrightarrow (\boldsymbol{0}_{\mathcal{S}}, \mathsf{kE}_y) \neq (-\mathsf{sE}_x(\boldsymbol{w}), \mathsf{rE}_y(\boldsymbol{w}))$$

which is equivalent (by Lemma 3) to the existence of $(\boldsymbol{z}_1, \boldsymbol{z}_2) \in \mathcal{S} \times \mathcal{R}$ such that

$$(\boldsymbol{z}_1, \boldsymbol{z}_2)^\top (\boldsymbol{0}_{\mathcal{S}}, \mathsf{kE}_y) = 1 \ \wedge \ (\boldsymbol{z}_1, \boldsymbol{z}_2)^\top (-\mathsf{sE}_x, \mathsf{rE}_y) = \boldsymbol{0}_{\mathcal{W}}^\top$$

In particular we can define the decoding functions as the linear functions $\mathsf{sD} : \mathcal{S} \to \mathcal{W}$ and $\mathsf{rD} : \mathcal{R} \to \mathcal{W}$ defined as $\mathsf{sD}(\boldsymbol{s}) = \boldsymbol{z}_1^\top \boldsymbol{s}$ and $\mathsf{rD}(\boldsymbol{r}) = \boldsymbol{z}_2^\top \boldsymbol{r}$.
In this context, for all $(x, y) \in \mathsf{Pred}_{pe}$, $\boldsymbol{w} \in \mathcal{W}$:

$$\mathsf{sD}_{x,y}(\mathsf{sE}_x(\boldsymbol{w})) = \mathsf{rD}_{x,y}(\mathsf{rE}_y(\boldsymbol{w})) \ \wedge \ \mathsf{rD}_{x,y}(\mathsf{kE}_y) = 1.$$

Finally, note that $\mathsf{sD}_{x,y}$ and $\mathsf{rD}_{x,y}$ can be computed by solving a linear system of equations. □

We can now prove an alternative definition of $\mathbb{Z}_p$-bilinear predicate encodings that keeps the decoding functions implicit and uses the implicit predicate.

Theorem 1 (Alternative definition of predicate encodings). *For the encoding functions* $pe = (\mathsf{sE}, \mathsf{rE}, \mathsf{kE})$*, there exist efficient decoding functions* sD *and* rD *such that* $(\mathsf{sE}, \mathsf{rE}, \mathsf{kE}, \mathsf{sD}, \mathsf{rD})$ *is a* $\mathbb{Z}_p$*-bilinear predicate encoding for* Pred_{pe}*.*

Proof. The statement immediately follows from Lemma 1 and Lemma 2. □

To obtain our results, we often switch between the original definition and our alternative definition. The advantage of our definition is that we can focus on the encoding functions and the implicit predicate yields a simple method to check how modifying the encoding functions affects the predicate. For example, in the next section, we define optimizations that preserve the implicit predicate. The advantage of making the decoding functions explicit is that we can check if they satisfy certain conditions such as independence of x and y, which is important for attribute-hiding. In some transformations, we also use the witness function rW for α-privacy.

3.2 Optimization of predicate encodings

In this section, we present two different methods to simplify predicate encoding functions $pe = (\mathsf{sE}, \mathsf{rE}, \mathsf{kE})$. By combining both methods, we can obtain new predicate encoding functions $pe' = (\mathsf{sE}', \mathsf{rE}', \mathsf{kE}')$ for types $\mathcal{W}'$, $\mathcal{R}'$, and $\mathcal{S}'$.

First, the inputs of sE and rE can be preprocessed using a function $\sigma^w : \mathcal{W}' \to \mathcal{W}$. We define this predicate function transformation as $\mathsf{isimp}_{\sigma^w}(\mathsf{sE}, \mathsf{rE}, \mathsf{kE}) = (\mathsf{sE} \circ \sigma^w, \mathsf{rE} \circ \sigma^w, \mathsf{kE})$. Since the same preprocessing function is used to define both sE and rE, it cannot depend on $\mathcal{X}$ or $\mathcal{Y}$. It is easy to see that this transformation preserves reconstructability, but can destroy privacy.

Second, the outputs of sE, rE, and kE can be postprocessed using functions $\sigma^s : \mathcal{X} \to \mathcal{S} \to \mathcal{S}'$ and $\sigma^r : \mathcal{Y} \to \mathcal{R} \to \mathcal{R}'$. We define this predicate function transformation as $\mathsf{osimp}_{\sigma^s,\sigma^r}(\mathsf{sE}, \mathsf{rE}, \mathsf{kE}) = (\mathsf{sE}', \mathsf{rE}', \mathsf{kE}')$ s.t. for all x, $\mathsf{sE}'_x = \sigma^s_x \circ \mathsf{sE}_x$ and for all y, $\mathsf{rE}'_y = \sigma^r_y \circ \mathsf{rE}_y$ and $\mathsf{kE}'_y = \sigma^r_y(\mathsf{kE}_y)$. For such functions, privacy is obviously preserved, but reconstructability can be destroyed.

These transformations are useful to make predicate encodings simpler and more efficient in different manners. They can be used to:

- Reduce the size of mpk, ct_x and sk_y. If $\mathsf{dim}(\mathcal{W}') < \mathsf{dim}(\mathcal{W})$, the number of elements in mpk will decrease. This will also improve the performance of encryption and key generation because both depend directly on mpk. Additionally, if $\mathsf{dim}(\mathcal{S}') < \mathsf{dim}(\mathcal{S})$ or $\mathsf{dim}(\mathcal{R}') < \mathsf{dim}(\mathcal{R})$, the number of elements in ct_x and sk_y will also decrease respectively.
- Make matrices corresponding to encoding and decoding functions sparser. For example, if we consider σ^s and σ^r as functions that apply matrix *Gaussian elimination* to the matrices associated to sE and $(\mathsf{rE} \times \mathsf{kE})$, many entries from this matrices will be zero. If many entries are zero, fewer group operations will be performed during encryption and key generation, improving the performance.

The above simplifications can be successfully applied to actual predicate encodings proposed in [12]. In Sections 5.1 and 5.2 we apply a simplification mechanism to improve the performance of predicate encodings for *ABE for monotonic boolean formulas* and predicate encodings for *arithmetic span programs*.

Lemma 4 (Input simplification). *For $pe = (\mathsf{sE}, \mathsf{rE}, \mathsf{kE})$ and all linear functions $\sigma^w : \mathcal{W}' \to \mathcal{W}$ s.t. for all $(x, y) \notin \mathsf{P}$,*

$$(\mathbf{0}_{\mathcal{S}}, \mathsf{kE}_y) \in (\mathsf{sE}_x \times \mathsf{rE}_y)(\sigma^w(\mathcal{W}')),$$

it holds that $\mathsf{Pred}_{pe} = \mathsf{Pred}_{pe'}$ for the simplified predicate encoding functions $pe' = \mathsf{isimp}_{\sigma^w}$.

Proof. First, note that the predicate encoding functions pe' are linear since σ^w and the functions in pe are linear. The condition on σ^w yields $\mathsf{Pred}_{pe'}(x, y) \Rightarrow \mathsf{Pred}_{pe}(x, y)$ for all $(x, y) \in \mathcal{X} \times \mathcal{Y}$. The other direction directly follows from $\sigma^w(\mathcal{W}') \subseteq \mathcal{W}$. □

We can prove a similar result for output postprocessing.

Lemma 5 (Output simplification). *For* $pe = (\mathsf{sE}, \mathsf{rE}, \mathsf{kE})$ *and for all functions* $\sigma^s : \mathcal{X} \to \mathcal{S} \to \mathcal{S}'$ *and* $\sigma^r : \mathcal{Y} \to \mathcal{R} \to \mathcal{R}'$ *s.t. for all* $(x, y) \in \mathsf{P}$*,*

$$(\mathbf{0}_{\mathcal{S}'}, \sigma^r_y(\mathsf{kE}_y)) \notin ((\sigma^s_x \circ \mathsf{sE}_x) \times (\sigma^r_y \circ \mathsf{rE}_y))(\mathcal{W}),$$

it holds that $\mathsf{Pred}_{pe} = \mathsf{Pred}_{pe'}$ *for the simplified predicate encoding functions* $pe' = \mathsf{osimp}_{\sigma^s, \sigma^r}$*.*

Proof. First, note that the predicate encoding functions pe' are linear since σ^s, σ^r, and the functions in pe are linear. The condition on σ^w yields $\mathsf{Pred}_{pe}(x, y) \Rightarrow \mathsf{Pred}_{pe'}(x, y)$ for all $(x, y) \in \mathcal{X} \times \mathcal{Y}$. The other direction directly follows from

$$(\mathbf{0}_{\mathcal{S}}, \mathsf{kE}_y) = (\mathsf{sE}_x(\boldsymbol{w}), \mathsf{rE}_y(\boldsymbol{w})) \Rightarrow (\mathbf{0}_{\mathcal{S}'}, \sigma^r_y(\mathsf{kE}_y)) = (\sigma^s_x(\mathsf{sE}_x(\boldsymbol{w})), \sigma^r_y(\mathsf{rE}_y(\boldsymbol{w})))$$

since linearity of σ^s_x implies $\sigma^s_x(\mathbf{0}_{\mathcal{S}}) = \mathbf{0}_{\mathcal{S}'}$. □

4 Combining predicates

Using the new characterization of predicate encodings from the previous section, we define transformations to combine predicate encodings into new predicate encodings for more complex predicates. More formally, given predicate encodings pe^1 and pe^2 for the predicates P^1 and P^2 and a binary operator $\diamond$, we define a predicate encoding transformation $\mathsf{ptrans}_\diamond$ that yields a new predicate encoding $\mathsf{ptrans}_\diamond(pe^1, pe^2)$ for P where $\mathsf{P}((x_1, x_2), (y_1, y_2)) := \mathsf{P}^1(x_1, y_1) \diamond \mathsf{P}^2(x_2, y_2)$. Similarly, we define predicate encoding transformations for unary operators. We first define transformers for disjunction, conjunction, and negation. Then we define a transformer for flexible combination where the receiver can choose between disjunction and conjunction when he performs the encoding. Finally, we define a transformer for the dual predicate. These combinations are useful to create interesting schemes that gather different properties from the more basic building blocks. In Section 5 we propose several constructions that rely on these transformations.

In the remainder of this section, we fix linear predicate encoding functions $pe^1 = (\mathsf{sE}^1, \mathsf{rE}^1, \mathsf{kE}^1)$ for types $\mathcal{X}_1, \mathcal{Y}_1, \mathcal{W}_1, \mathcal{S}_1, \mathcal{R}_1$ and $pe^2 = (\mathsf{sE}^2, \mathsf{rE}^2, \mathsf{kE}^2)$ for types $\mathcal{X}_2, \mathcal{Y}_2, \mathcal{W}_2, \mathcal{S}_2, \mathcal{R}_2$. Using these encoding functions as inputs for the transformations, the result types of our transformations are given in Figure 2.

4.1 Boolean combinations

We first give the transformation for disjunction. Next, we give the transformation for conjunction which requires only a small change to the transformation for disjunction. Finally, we give the transformation for negation, which uses different techniques than the first two transformations.

	$\mathcal{X}$	$\mathcal{Y}$	$\mathcal{W}$	$\mathcal{S}$	$\mathcal{R}$
$\mathsf{ptrans}_\vee$	$\mathcal{X}_1 \times \mathcal{X}_2$	$\mathcal{Y}_1 \times \mathcal{Y}_2$	$\mathcal{W}_1 \times \mathcal{W}_2$	$\mathcal{S}_1 \times \mathcal{S}_2$	$\mathcal{R}_1 \times \mathcal{R}_2$
$\mathsf{ptrans}_\wedge$	$\mathcal{X}_1 \times \mathcal{X}_2$	$\mathcal{Y}_1 \times \mathcal{Y}_2$	$\mathcal{W}_1 \times \mathcal{W}_2 \times \mathbb{Z}_p$	$\mathcal{S}_1 \times \mathcal{S}_2$	$\mathcal{R}_1 \times \mathcal{R}_2$
$\mathsf{ptrans}_\neg$	$\mathcal{X}_1$	$\mathcal{Y}_1$	$\mathcal{S}_1 \times \mathcal{W}_1 \times \mathcal{R}_1$	$\mathcal{W}_1$	$\mathcal{W}_1 \times \mathbb{Z}_p$
$\mathsf{ptrans}_{\bowtie}$	$\mathcal{X}_1 \times \mathcal{X}_2$	$\mathcal{Y}_1 \times \mathcal{Y}_2 \times \{\vee, \wedge\}$	$\mathcal{W}_1 \times \mathcal{W}_2 \times \mathbb{Z}_p$	$\mathcal{S}_1 \times \mathcal{S}_2$	$\mathcal{R}_1 \times \mathcal{R}_2$
$\mathsf{ptrans}_\star$	$\mathcal{Y}_1$	$\mathcal{X}_1$	$\mathcal{W}_1 \times \mathbb{Z}_p$	$\mathcal{R}_1$	$\mathcal{S}_1 \times \mathbb{Z}_p$

Fig. 2. Result types for predicate transformers with input types $\mathcal{X}_1$, $\mathcal{X}_2$, ...

Disjunction. Disjunction of predicates can be obtained generically from a pair of predicate encryption schemes by defining the master keys as the concatenations of the original keys and defining the ciphertext (resp. the secret key) for pairs of attributes as the concatenation of the ciphertexts (resp. the secret keys) where the first attribute is used with the first scheme and the second attribute is used with the second scheme. Given a valid secret key for one of the schemes, it is possible to decrypt the corresponding ciphertext in the concatenation. In contrast to the generic approach, we define transformations on the level of predicate encodings. The advantage is that for some transformations, no generic solution is known and the generic disjunction transformation is not compatible with subsequent transformations defined only for predicate encodings. Furthermore, applying the compiler from predicate encodings to predicate encryption only once at the end results in sharing of elements in keys and ciphertexts that would be duplicated with generic approaches.

Definition 3 (Disjunction of predicate encodings). *We define the predicate encoding* $\mathsf{ptrans}_\vee(pe^1, pe^2) = (\mathsf{sE}, \mathsf{rE}, \mathsf{kE})$ *as*

$$\begin{aligned}
\mathsf{sE}_{(x_1,x_2)}(\boldsymbol{w}_1, \boldsymbol{w}_2) &= (\mathsf{sE}^1_{x_1}(\boldsymbol{w}_1), \mathsf{sE}^2_{x_2}(\boldsymbol{w}_2)) \\
\mathsf{rE}_{(y_1,y_2)}(\boldsymbol{w}_1, \boldsymbol{w}_2) &= (\mathsf{rE}^1_{y_1}(\boldsymbol{w}_1), \mathsf{rE}^2_{y_2}(\boldsymbol{w}_2)) \\
\mathsf{kE}_{(y_1,y_2)} &= (\mathsf{kE}^1_{y_1}, \mathsf{kE}^2_{y_2}).
\end{aligned}$$

Theorem 2 (Correctness of disjunction). *For all* $(x_1, y_1) \in \mathcal{X}_1 \times \mathcal{Y}_1$ *and* $(x_2, y_2) \in \mathcal{X}_2 \times \mathcal{Y}_2$*, it holds that*

$$\mathsf{Pred}_{\mathsf{ptrans}_\vee(pe^1,pe^2)}((x_1, x_2), (y_1, y_2)) \Leftrightarrow \mathsf{Pred}_{pe^1}(x_1, y_1) \vee \mathsf{Pred}_{pe^2}(x_2, y_2).$$

Proof. For all $(x_1, y_1) \in \mathcal{X}_1 \times \mathcal{Y}_1$, and $(x_2, y_2) \in \mathcal{X}_2 \times \mathcal{Y}_2$ such that $\mathsf{Pred}_{pe^1}(x_1, y_1) \vee \mathsf{Pred}_{pe^2}(x_2, y_2)$, by Lemma 2, if $\mathsf{Pred}_{pe^1}(x_1, y_1)$, there exist decoding functions $\mathsf{sD}^1_{x_1,y_1}$, $\mathsf{rD}^1_{x_1,y_1}$ for pe^1. Otherwise, $\mathsf{Pred}_{pe^2}(x_2, y_2)$ holds and there exists decoding functions $\mathsf{sD}^2_{x_2,y_2}$, $\mathsf{rD}^2_{x_2,y_2}$ for pe^2. It can be easily checked that the following are valid decoding functions for $\mathsf{ptrans}_\vee(pe^1, pe^2)$:

$$\mathsf{sD}_{(x_1,x_2),(y_1,y_2)}(\boldsymbol{s}_1, \boldsymbol{s}_2) = \begin{cases} \mathsf{sD}^1_{x_1,y_1}(\boldsymbol{s}_1) & \text{if } \mathsf{Pred}_{pe^1}(x_1, y_1) \\ \mathsf{sD}^2_{x_2,y_2}(\boldsymbol{s}_2) & \text{otherwise} \end{cases}$$

$$\mathsf{rD}_{(x_1,x_2),(y_1,y_2)}(\boldsymbol{r}_1,\boldsymbol{r}_2) = \begin{cases} \mathsf{rD}^1_{x_1,y_1}(\boldsymbol{r}_1) & \text{if } \mathsf{Pred}_{pe^1}(x_1,y_1) \\ \mathsf{rD}^2_{x_2,y_2}(\boldsymbol{r}_2) & \text{otherwise.} \end{cases}$$

For all $(x_1, y_1) \in \mathcal{X}_1 \times \mathcal{Y}_1$, and $(x_2, y_2) \in \mathcal{X}_2 \times \mathcal{Y}_2$ such that $\neg(\mathsf{Pred}_{pe^1}(x_1, y_1) \vee \mathsf{Pred}_{pe^2}(x_2, y_2))$, it holds $\neg\,\mathsf{Pred}_{pe^1}(x_1, y_1) \wedge \neg\,\mathsf{Pred}_{pe^2}(x_2, y_2)$ and by Lemma 1 there exist valid privacy witnesses $\mathsf{rW}^1_{x_1,y_1}$, $\mathsf{rW}^2_{x_2,y_2}$ for pe^1 and pe^2 respectively. It can be easily verified that this is a valid privacy witness for $pe^1 \vee pe^2$:

$$\mathsf{rW}_{(x_1,x_2),(y_1,y_2)} = \left(\mathsf{rW}^1_{x_1,y_1}, \mathsf{rW}^2_{x_2,y_2}\right) \qquad \square$$

Note that it is possible to obtain sharing between attributes, e.g., for $\mathcal{X}$ as follows. If $\mathcal{X}_1 = \mathcal{X}_2$ and the sender uses only the subset $\{(x_1, x_1) \mid x_1 \in \mathcal{X}_1\}$ of $\mathcal{X}$, then we obtain an encoding for $\mathsf{P}(x_1, (y_1, y_2)) := \mathsf{Pred}_{pe^1}(x_1, y_1) \vee \mathsf{Pred}_{pe^2}(x_1, y_2)$.

Conjunction. In contrast to disjunction, we cannot use the naive solution that just concatenates secret keys for conjunction. Given keys for attribute pairs (y_1, y_2) and (y_1', y_2'), it would be otherwise possible to recombine the components and obtain a key for (y_1, y_2') leading to collusion attacks. The predicate encoding transformation deals with this problem by "tying" the two components together with additional randomness.

Definition 4 (Conjunction of predicate encodings). *We define the predicate encoding* $\mathsf{ptrans}_\wedge(pe^1, pe^2) = (\mathsf{sE}, \mathsf{rE}, \mathsf{kE})$ *as*

$$\begin{aligned} \mathsf{sE}_{(x_1,x_2)}(\boldsymbol{w}_1, \boldsymbol{w}_2, w) &= (\mathsf{sE}^1_{x_1}(\boldsymbol{w}_1), \mathsf{sE}^2_{x_2}(\boldsymbol{w}_2)) \\ \mathsf{rE}_{(y_1,y_2)}(\boldsymbol{w}_1, \boldsymbol{w}_2, w) &= (\mathsf{rE}^1_{y_1}(\boldsymbol{w}_1) + w \cdot \mathsf{kE}^1_{y_1}, \mathsf{rE}^2_{y_2}(\boldsymbol{w}_2) - w \cdot \mathsf{kE}^2_{y_2}) \\ \mathsf{kE}_{(y_1,y_2)} &= (\mathsf{kE}^1_{y_1}, \mathsf{kE}^2_{y_2}). \end{aligned}$$

Theorem 3 (Correctness of conjunction). *For all* $(x_1, y_1) \in \mathcal{X}_1 \times \mathcal{Y}_1$*, and* $(x_2, y_2) \in \mathcal{X}_2 \times \mathcal{Y}_2$*, it holds that*

$$\mathsf{Pred}_{\mathsf{ptrans}_\wedge(pe^1,pe^2)}((x_1, x_2), (y_1, y_2)) \Leftrightarrow \mathsf{Pred}_{pe^1}(x_1, y_1) \wedge \mathsf{Pred}_{pe^2}(x_2, y_2).$$

Proof. For all $(x_1, y_1) \in \mathcal{X}_1 \times \mathcal{Y}_1$, and $(x_2, y_2) \in \mathcal{X}_2 \times \mathcal{Y}_2$ such that $\mathsf{Pred}_{pe^1}(x_1, y_1) \wedge \mathsf{Pred}_{pe^2}(x_2, y_2)$, by Lemma 2, there exist decoding functions $\mathsf{sD}^1_{x_1,y_1}$, $\mathsf{rD}^1_{x_1,y_1}$ and $\mathsf{sD}^2_{x_2,y_2}$, $\mathsf{rD}^2_{x_2,y_2}$ for pe^1 and pe^2 respectively. It can be easily checked that the following are valid decoding functions for $\mathsf{ptrans}_\wedge(pe^1, pe^2)$:

$$\begin{aligned} \mathsf{sD}_{(x_1,x_2),(y_1,y_2)}(\boldsymbol{s}_1, \boldsymbol{s}_2) &= \frac{1}{2}\mathsf{sD}^1_{x_1,y_1}(\boldsymbol{s}_1) + \frac{1}{2}\mathsf{sD}^2_{x_2,y_2}(\boldsymbol{s}_2) \\ \mathsf{rD}_{(x_1,x_2),(y_1,y_2)}(\boldsymbol{r}_1, \boldsymbol{r}_2) &= \frac{1}{2}\mathsf{rD}^1_{x_1,y_1}(\boldsymbol{r}_1) + \frac{1}{2}\mathsf{rD}^2_{x_2,y_2}(\boldsymbol{r}_2) \end{aligned}$$

For all $(x_1, y_1) \in \mathcal{X}_1 \times \mathcal{Y}_1$, and $(x_2, y_2) \in \mathcal{X}_2 \times \mathcal{Y}_2$ such that $\neg(\mathsf{Pred}_{pe^1}(x_1, y_1) \wedge \mathsf{Pred}_{pe^2}(x_2, y_2))$, if $\neg\,\mathsf{Pred}_{pe^1}(x_1, y_1)$, by Lemma 1 there exists a valid privacy witness $\mathsf{rW}^1_{x_1,y_1}$ for pe^1, otherwise there exists a valid privacy witness $\mathsf{rW}^2_{x_2,y_2}$ for pe^2. It can be easily verified that this is a valid privacy witness for $\mathsf{ptrans}_\wedge(pe^1, pe^2)$:

$$\mathsf{rW}_{(x_1,x_2),(y_1,y_2)} = \begin{cases} (2\,\mathsf{rW}^1_{x_1,y_1}, \mathbf{0}_{\mathcal{W}_2}, & -1) \text{ if } \neg\,\mathsf{Pred}_{pe^1}(x_1, y_1) \\ (\mathbf{0}_{\mathcal{W}_1}, & 2\,\mathsf{rW}^2_{x_2,y_2}, +1) \text{ otherwise.} \end{cases} \qquad \square$$

Negation. To obtain a functionally complete set of boolean predicate encoding transformers, we now define a transformation for negation. This is interesting because it unifies negated predicates like Non-zero Inner Product Encryption (NIPE) and Zero Inner Product Encryption (ZIPE). In Section 5.1 we use this transformation to build optimized predicate encodings. The technique works for predicate encodings where the negation transformation yields a predicate encoding that can be further simplified (using our methods from Section 3.2).

Definition 5 (Negation of predicate encodings). *We define the predicate encoding* $\mathsf{ptrans}_\neg(pe^1) = (\mathsf{sE}, \mathsf{rE}, \mathsf{kE})$ *as*

$$\begin{aligned}
\mathsf{sE}_{x_1}(\boldsymbol{w}', \boldsymbol{w}'', \boldsymbol{w}''') &= {}^\top\mathsf{sE}^1_{x_1}(\boldsymbol{w}') - \boldsymbol{w}'' \\
\mathsf{rE}_{y_1}(\boldsymbol{w}', \boldsymbol{w}'', \boldsymbol{w}''') &= ({}^\top\mathsf{rE}^1_{y_1}(\boldsymbol{w}''') + \boldsymbol{w}'', \mathsf{kE}^\top_{y_1}\boldsymbol{w}''') \\
\mathsf{kE}_{y_1} &= (\mathbf{0}_{\mathcal{W}_1}, 1).
\end{aligned}$$

Theorem 4 (Correctness of negation). *For all* $(x, y) \in \mathcal{X} \times \mathcal{Y}$*, it holds that*

$$\mathsf{Pred}_{\mathsf{ptrans}_\neg(pe^1)}(x, y) \Leftrightarrow \neg\, \mathsf{Pred}_{pe^1}(x, y).$$

Proof. For all $(x_1, y_1) \in \mathcal{X}_1 \times \mathcal{Y}_1$, such that $\neg\, \mathsf{Pred}_{pe^1}(x_1, y_1)$ by Lemma 1, there exists a valid privacy witness $\mathsf{rW}^1_{x_1,y_1}$ for pe^1. It can be easily checked that the following are valid decoding functions for $\mathsf{ptrans}_\neg(pe^1)$:

$$\begin{aligned}
\mathsf{sD}_{x_1,y_1} &= \mathsf{rW}^1_{x_1,y_1} \\
\mathsf{rD}_{x_1,y_1} &= \left(-\mathsf{rW}^1_{x_1,y_1} \mid 1\right)
\end{aligned}$$

For all $(x_1, y_1) \in \mathcal{X}_1 \times \mathcal{Y}_1$, such that $\mathsf{Pred}_{pe^1}(x_1, y_1)$, by Lemma 2 there exist decoding functions $\mathsf{sD}^1_{x_1,y_1}$, $\mathsf{rD}^1_{x_1,y_1}$ for pe^1. It can be easily verified that this is a valid privacy witness for $\mathsf{ptrans}_\neg(pe^1)$:

$$\mathsf{rW}_{x_1,y_1} = \left(-\mathsf{sD}^1_{x_1,y_1}, -{}^\top\mathsf{sE}^1_{x_1}(\mathsf{sD}^1_{x_1,y_1}), \mathsf{rD}^1_{x_1,y_1}\right) \qquad \square$$

4.2 Flexible boolean combinations

We note that the boolean combinations of predicate encodings can be applied dynamically. This is, instead of combining two predicates statically, with the conjunction for example $\mathsf{P} \wedge \mathsf{P}'$, we can combine them by leaving placeholders $\mathsf{P} \bowtie \mathsf{P}'$ that will be chosen during encryption or key generation. The following theorem shows how to dynamically combine two predicates making the combinator part of the secret key sk_y. Thanks to this dynamic combination, we can create predicate encodings for boolean formulas where the leaves are predicates instead of attributes. The only drawback is that the structure of the boolean formula has to be fixed.

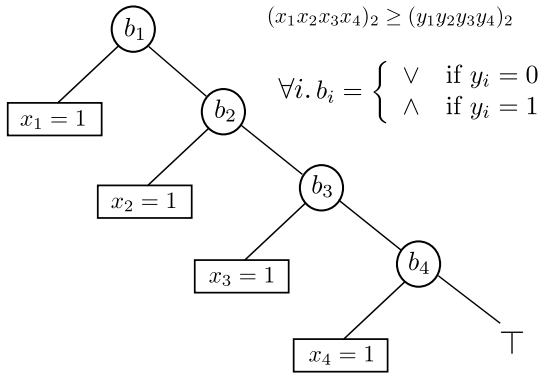

Fig. 3. Example of fixed structure for inequalities

Definition 6 (Flexible combination of predicate encodings). *We define the predicate encoding* $\mathsf{ptrans}_{\bowtie}(pe^1, pe^2) = (\mathsf{sE}, \mathsf{rE}, \mathsf{kE})$ *as*

$$\begin{aligned} \mathsf{sE}_{(x_1,x_2)}(\boldsymbol{w}_1, \boldsymbol{w}_2, w) &= (\mathsf{sE}^1_{x_1}(\boldsymbol{w}_1), \mathsf{sE}^2_{x_2}(\boldsymbol{w}_2)) \\ \mathsf{rE}_{(y_1,y_2,\bowtie)}(\boldsymbol{w}_1, \boldsymbol{w}_2, w) &= (\mathsf{rE}^1_{y_1}(\boldsymbol{w}_1) + f_{\bowtie}(w) \cdot \mathsf{kE}^1_{y_1}, \mathsf{rE}^2_{y_2}(\boldsymbol{w}_2) - f_{\bowtie}(w) \cdot \mathsf{kE}^2_{y_2}) \\ \mathsf{kE}_{(y_1,y_2,\bowtie)} &= (\mathsf{kE}^1_{y_1}, \mathsf{kE}^2_{y_2}). \end{aligned}$$

where $f_{\bowtie}(w)$ *is defined as* w *if* $\bowtie = \wedge$ *and* 0 *if* $\bowtie = \vee$.

Theorem 5 (Correctness of the flexible combination). *For all* $(x_1, y_1) \in \mathcal{X}_1 \times \mathcal{Y}_1$, *and* $(x_2, y_2) \in \mathcal{X}_2 \times \mathcal{Y}_2$, *it holds that*

$$\mathsf{Pred}_{\mathsf{ptrans}_{\bowtie}(pe^1,pe^2)}((x_1, x_2), (y_1, y_2, \bowtie)) \Leftrightarrow \mathsf{Pred}_{pe^1}(x_1, y_1) \bowtie \mathsf{Pred}_{pe^2}(x_2, y_2).$$

Proof. This Theorem follows straightforwardly from Theorems 2 and 3. □

Note that our Theorem 6 gives us an equivalent version of the above theorem, where the placeholder is part of the ciphertext ct_x. Figure 3 presents a possible application of a flexible fixed-structure combination of boolean operators. It encodes the predicate $\mathsf{P}(x, y) = 1$ iff $x \geq y$, where $\mathcal{X} = \mathcal{Y} = \{0, 1\}^4$ (4-bit strings). Note that the leaf nodes are IBE predicate encodings (one of the simplest predicate encodings).

4.3 Dual predicate

In the literature, the notions of KP-ABE and CP-ABE are considered separately. In fact, many works are only valid for one of the two versions of Attribute Based Encryption. Our transformation unifies the notion of KP-ABE and CP-ABE in the framework of predicate encodings. In this context they should not be considered separately or, in other words, our transformation provides a Ciphertext-Policy predicate encoding from any Key-Policy predicate encoding

and vice versa. Note also that this transformation barely modifies the size of the encoding. New ciphertexts have as many elements as secret keys had whereas the size of new secret keys is just one group element (one row in the matrix) bigger that earlier ciphertexts.

Definition 7 (Dual predicate encodings). *We define the predicate encoding* $\mathsf{ptrans}_\star(pe^1) = (\mathsf{sE}, \mathsf{rE}, \mathsf{kE})$ *as*

$$\begin{aligned}\mathsf{sE}_{y_1}(\boldsymbol{w}, w) &= \mathsf{rE}^1_{y_1}(\boldsymbol{w}) + w \cdot \mathsf{kE}_{y_1}\\ \mathsf{rE}_{x_1}(\boldsymbol{w}, w) &= (\mathsf{sE}^1_{x_1}(\boldsymbol{w}), w)\\ \mathsf{kE}_{x_1} &= (\mathbf{0}_{\mathcal{S}_1}, 1).\end{aligned}$$

Theorem 6 (Correctness of dual). *For all* $(x, y) \in \mathcal{X} \times \mathcal{Y}$*, it holds that*

$$\mathsf{Pred}_{\mathsf{ptrans}_\star(pe^1)}(y, x) \Leftrightarrow \mathsf{Pred}_{pe^1}(x, y).$$

Proof. For all $(x_1, y_1) \in \mathcal{X}_1 \times \mathcal{Y}_1$, such that $\mathsf{Pred}_{pe^1}(x_1, y_1)$ by Lemma 1, there exist decoding functions $\mathsf{sD}^1_{x_1,y_1}$, $\mathsf{rD}^1_{x_1,y_1}$ for pe^1. It can be easily checked that the following are valid decoding functions for $\mathsf{ptrans}_\star(pe^1)$:

$$\begin{aligned}\mathsf{sD}_{y_1,x_1}(\boldsymbol{s}) &= \mathsf{rD}^1_{x_1,y_1}(\boldsymbol{s})\\ \mathsf{rD}_{y_1,x_1}(\boldsymbol{r}, r) &= \mathsf{sD}^1_{x_1,y_1}(\boldsymbol{r}) + r\end{aligned}$$

For all $(x_1, y_1) \in \mathcal{X}_1 \times \mathcal{Y}_1$, such that $\neg\, \mathsf{Pred}_{pe^1}(x_1, y_1)$, by Lemma 2 there exists a valid privacy witness $\mathsf{rW}^1_{x_1,y_1}$ for pe^1. It can be easily verified that this is a valid privacy witness for $\mathsf{ptrans}_\neg(pe^1)$:

$$\mathsf{rW}_{y_1,x_1} = \left(-\mathsf{rW}^1_{x_1,y_1}, 1\right) \qquad \square$$

5 Constructions

We provide new instances of predicate encodings to achieve predicate encryption schemes with new properties or better performance.

5.1 Boolean Formulas

In this section and the following we use a common methodology that is useful to simplify existing predicate encodings. Our technique is based on applying Theorem 4 first, to create a predicate encoding for the negation and then apply Lemmas 4 and 5 to simplify the resulting encoding. For some particular predicates, the negated version of the encoding can be simplified into a simpler encoding than the original one. Note that the predicate associated to the new encoding is negated, however if inputs are also negated, the predicate is equivalent to the original one. On the left side of Figure 4 there is an example of a boolean formula KP-ABE for 4 attributes $\{a, b, c, d\}$, where the predicate is $\mathsf{P}(x, y) := x(y) = 1$. On the right side, we show how secret keys and policies can

P	$\bar{\mathsf{P}}$
attributes $= \{a, b, c, d\}$	attributes $= \{\bar{a}, \bar{b}, \bar{c}, \bar{d}\}$
$x = (a \wedge c) \vee d$	$x = (\bar{a} \vee \bar{c}) \wedge \bar{d}$
$y = \{a, c\}$	$y = \{\bar{c}, \bar{d}\}$
$\mathsf{P}(x, y) := x(y)$	$\bar{\mathsf{P}}(x, y) := \neg x(y)$

Fig. 4. Equivalent encodings of a policy using P (CP-ABE) on the left and $\bar{\mathsf{P}}$ (negated CP-ABE) on the right.

be modified in the negated version to make it equivalent. First, the attribute universe is formed by the negated version of all the attributes in the original universe. Second, secret keys are formed by all the attributes (in negated form) that were not part of the original secret key. Finally, policies are the negated original policies expressed in Negation Normal Form.

In [12], the authors propose two predicate encoding (KP and CP versions) for monotonic boolean formulas. In particular, access structures are represented by a linear secret sharing (LSS) scheme.

Definition 8 (Linear Secret Sharing [21]). *A linear secret sharing scheme is a pair (M, ρ) where M is a matrix of size $l_1 \times l_2$ with entries in $\mathbb{Z}_p$ and ρ is a mapping from $[l_1]$ to a universe of attributes $\mathcal{U}$. Given $\boldsymbol{x} = (x_1, \ldots, x_n) \in \{0,1\}^n$, we say that*

$$\boldsymbol{x} \text{ satisfies } (M, \rho) \text{ iff } \mathbf{1}^\top \in \underset{span}{row} \langle M_{\boldsymbol{x}} \rangle$$

where $\mathbf{1}^\top = (1, 0, \ldots, 0) \in \mathbb{Z}_p^{l_2}$ is a row vector and $M_{\boldsymbol{x}}$ denotes the matrix formed by rows $\{M_j : x_{\rho(j)} = 1\}$.

LSS can be used to encode the information of monotonic boolean formulas. In fact, Lewko and Waters proposed an algorithm to efficiently convert any monotonic boolean formula to LSS matrices [26]. Roughly, this algorithm produces a matrix whose number of rows is equal to the number of leaf nodes in the formula and whose number of columns is equal to the number of *and-gates* plus one.

The predicate encoding proposed in [12] needs to fix the dimensions of this matrix a priori and thus, the size of boolean formulas that can be handled is bounded. In order to achieve α-privacy, the authors need to impose the restriction that ρ is a permutation, this is, $l_1 = n$. This implies that every attribute can be only used once in the policy (one could overcome this limitation by duplicating every attribute several times). Therefore, the number of elements in secret keys and ciphertexts is always maximal, it equals the number of (possibly duplicated) attributes, even for small policies. Furthermore, the number of *and-gates* must be fixed a priori too (because it is related the the number of columns in the matrix). The predicate encoding for key-policy monotonic boolean formulas in [12] is the following:

$$\mathcal{X} = \{0,1\}^n, \quad \mathcal{Y} = \mathbb{Z}_p^{n \times l_2}, \quad \mathcal{S} = \mathbb{Z}_p^n, \quad \mathcal{R} = \mathbb{Z}_p^n, \quad \mathcal{W} = \mathbb{Z}_p^n \times \mathbb{Z}_p^{l_2 - 1}$$

$$\mathsf{sE}_{\boldsymbol{x}} = \left(\, \mathsf{diag}(\boldsymbol{x}) \;\middle|\; \mathbf{0} \,\right) \qquad \mathsf{rE}_M = \left(\, I_n \;\middle|\; M_{\{2,\ldots,l_2\}} \,\right) \qquad \mathsf{kE}_M = \left(\, M_{\{1\}} \,\right)$$

Here, $M_{\{1\}}$ denotes the first column of matrix M, whereas $M_{\{2,\ldots,l_2\}}$ represents the rest of the matrix. We propose the following predicate encoding for (negated) key-policy monotonic boolean formulas, which is an equivalent predicate if instantiated with negated inputs:

$$\mathcal{X} = \{0,1\}^n, \quad \mathcal{Y} = \mathbb{Z}_p^{n \times l_2}, \quad \mathcal{S} = \mathbb{Z}_p^n, \quad \mathcal{R} = \mathbb{Z}_p^{l_2}, \quad \mathcal{W} = \mathbb{Z}_p^n$$

$$\mathsf{sE}_{\boldsymbol{x}} = I_n - \mathsf{diag}(\boldsymbol{x}) \qquad \mathsf{rE}_M = M^\top \qquad \mathsf{kE}_M = (1, 0, \ldots, 0)^\top$$

Since the number of columns has been reduced up to half (being half when the bound on the number of *and-gates* is maximal), the new encoding is simpler. Furthermore, it has an additional property: the size of secret keys is proportional to the complexity of the policies. In particular, it is equal to the number of *and-gates* in policy (or equivalently, the number of *or-gates* in the non-negated version). Note that our improvement also works in the ciphertext-policy case.

5.2 Arithmetic span programs.

In [18], Ishai and Wee show how to relate *Arithmetic Branching Programs* (ABP) with *Arithmetic Span Programs*. Roughly speaking, an ABP over a finite field $\mathbb{F}_p$, is an algorithm that computes a function $f : \mathbb{F}_p^n \to \mathbb{F}_p$ defined by using addition and multiplication over the field.

Definition 9 (Arithmetic Span Program [22]). *An arithmetic span program $(\mathcal{V}, \rho)$ is a collection of vectors $\mathcal{V} = \{(\boldsymbol{y}_j, \boldsymbol{z}_j) : j \in [l]\} \in \mathbb{Z}_p^{l'}$ and $\rho : [l] \to [n]$. We say that*

$$\boldsymbol{x} \in \mathbb{Z}_p^n \text{ satisfies } (\mathcal{V}, \rho) \text{ iff } \left(1, 0, \ldots, 0\right) \in \underset{span}{row} \left\langle x_{\rho(j)} \boldsymbol{y}_j^T + \boldsymbol{z}_j^T \right\rangle .$$

Given an ABP for f, Ishai and Wee present an efficient method to construct an arithmetic span program $(\mathcal{V}, \rho)$ such that $\boldsymbol{x} \in \mathbb{F}_p^n$ satisfies $(\mathcal{V}, \rho)$ iff $f(\boldsymbol{x}) = \mathbf{0}$. They also show that with a small modification on $(\mathcal{V}, \rho)$, they can achieve the negated version of the previous predicate, i.e., $\boldsymbol{x} \in \mathbb{F}_p^n$ satisfies $(\mathcal{V}, \rho)$ iff $f(\boldsymbol{x}) \neq \mathbf{0}$. This means that a predicate encoding for arithmetic span programs is a predicate encoding for the predicate $\mathsf{P}(\boldsymbol{x}, f) := f(\boldsymbol{x}) = 0$ or the predicate $\mathsf{P}(\boldsymbol{x}, f) := f(\boldsymbol{x}) \neq 0$ (where f is an arbitrary polynomial function defined over the field).

The original predicate encoding for arithmetic span programs proposed in [12] is:

$$\mathcal{X} = \mathbb{Z}_p^n, \quad \mathcal{Y} = \mathbb{Z}_p^{n \times l'}, \quad \mathcal{S} = \mathbb{Z}_p^n, \quad \mathcal{R} = \mathbb{Z}_p^n \times \mathbb{Z}_p^n, \quad \mathcal{W} = \mathbb{Z}_p^n \times \mathbb{Z}_p^n \times \mathbb{Z}_p^{l'-1}$$

$$\mathsf{sE}_x = \left(\, \mathsf{diag}(\boldsymbol{x}) \;\middle|\; I_n \;\middle|\; \mathbf{0} \,\right) \quad \mathsf{rE}_{\mathcal{V}} = \left(\begin{array}{c|c|c} I_n & \mathbf{0} & Y_{\{2,\ldots,l\}} \\ \hline \mathbf{0} & I_n & Z_{\{2,\ldots,l\}} \end{array}\right) \quad \mathsf{kE}_{\mathcal{V}} = \left(\begin{array}{c} Y_{\{1\}} \\ \hline Z_{\{1\}} \end{array}\right)$$

where Y (resp. Z) is a matrix formed by concatenating all vectors $\boldsymbol{y}_j$ (resp. $\boldsymbol{z}_j$) as rows, $j \in [n]$. Note that this encoding introduces a one-use restriction, this is the encoding is defined for $n = l$ (from Definition 9). By using our technique we construct a more efficient encoding for arithmetic span programs:

$$\mathcal{X} = \mathbb{Z}_p^n, \quad \mathcal{Y} = \mathbb{Z}_p^{n \times l'}, \quad \mathcal{S} = \mathbb{Z}_p^n, \quad \mathcal{R} = \mathbb{Z}_p^{l'}, \quad \mathcal{W} = \mathbb{Z}_p^n \times \mathbb{Z}_p^n$$

$$\mathsf{sE}_x = \left(\, \mathsf{diag}(\boldsymbol{x}) \;\middle|\; -I_n \,\right) \qquad \mathsf{rE}_{\mathcal{V}} = \left(\, Z^T \middle| Y^T \,\right) \qquad \mathsf{kE}_{\mathcal{V}} = (1, 0, \ldots, 0)^\top.$$

In the conversion proposed in [18], $n \approx l'$, so we can say that this new encoding will reduce the size of secret keys by half, whereas the size of ciphertexts remains the same. On the other hand, setup, encryption and key generation will need $2/3$ as much time as the original encoding.

5.3 Attribute-Hiding

Chen et al. proposed an extension of the compiler in [12] to build weakly attribute-hiding predicate encryption schemes [11, 23]. In a weakly attribute-hiding scheme, the ciphertext attribute x remains secret for unauthorized users, the only thing they learn is the fact that their secret keys are not valid to decrypt but nothing else, even if the collude. This new compiler has to be instantiated with *predicate encodings* satisfying additional properties. The following is the original definition of attribute-hiding encodings:

Definition 10 (Attribute-Hiding Encodings). *A $\mathbb{Z}_p$-bilinear predicate encoding for predicate $\mathsf{P} : \mathcal{X} \times \mathcal{Y} \to \{0, 1\}$ is* attribute-hiding *if it verifies the following additional requirements:*

x-oblivious α-reconstruction: *$\mathsf{sD}_{x,y}$ and $\mathsf{rD}_{x,y}$ are independent of x.*

Attribute-hiding: *For all $(x, y) \notin \mathsf{P}$, it holds that*

$$\boldsymbol{w} \xleftarrow{\$} \mathcal{W}; \mathit{return}\ (\mathsf{sE}_x(\boldsymbol{w}), \mathsf{rE}_y(\boldsymbol{w})) \equiv \boldsymbol{s} \xleftarrow{\$} \mathcal{S}; \boldsymbol{r} \xleftarrow{\$} \mathcal{R}; \mathit{return}\ (\boldsymbol{s}, \boldsymbol{r})$$

where $\equiv$ denotes equality of distributions.

The following lemma relates these two properties with our alternative definition of predicate encodings:

Lemma 6 (Attribute-Hiding). *The* attribute-hiding *property from Definition 10 is equivalent to the following statement:*

$$\textit{For all } (x, y) \notin \mathsf{P},\ \mathsf{dim}(\mathsf{sE}_x \times \mathsf{rE}_y) = \mathsf{dim}(\mathcal{S}) + \mathsf{dim}(\mathcal{R}).$$

Proof. Given $(\boldsymbol{s}, \boldsymbol{r}) \in \mathcal{S} \times \mathcal{R}$, we define $\Omega_{\boldsymbol{s},\boldsymbol{r}} = \{\boldsymbol{w} : \mathsf{sE}_x(\boldsymbol{w}) = \boldsymbol{s} \ \wedge\ \mathsf{rE}_y(\boldsymbol{w}) = \boldsymbol{r}\}$. If $\mathsf{dim}(\mathsf{sE}_x \times \mathsf{rE}_y) = \mathsf{dim}(\mathcal{S}) + \mathsf{dim}(\mathcal{R})$, the cardinality of $\Omega_{\boldsymbol{s},\boldsymbol{r}}$ is independent of $\boldsymbol{s}, \boldsymbol{r}$. In particular, $\#\Omega_{\boldsymbol{s},\boldsymbol{r}} = p^{\mathsf{dim}(\mathcal{W}) - (\mathsf{dim}(\mathcal{S}) + \mathsf{dim}(\mathcal{R}))}$. This implies that the two distributions are identical. □

We note that the *conjunction* and *disjunction* combinations defined in Section 4.1 preserve this notion of attribute-hiding, i.e., if the two original predicates are attribute-hiding, the resulting one will have this property too. However, in the case of *disjunction*, the user has to guess which of the predicates holds. This is because decryption is defined in terms of x, which is hidden in this case. A consequence is that performance during decryption decreases.

We propose a Policy-Hiding ABE scheme for non-monotonic boolean formulas expressed in DNF (Disjunctive Normal Form). It comes from the simple observation that the inner-product can be used to encode conjunctions. Let $\boldsymbol{y} \in \{0,1\}^n \subseteq \mathbb{Z}_p^n$. If we establish that the i-th attribute a_i appears in a secret key for $\boldsymbol{y}$ iff $\boldsymbol{y}_i = 1$, we can encode conjunctions as follows:
Let $S, \bar{S} \subseteq \{a_i\}_{i=1}^n$ such that $S \cap \bar{S} = \emptyset$:

$$\bigwedge_{a \in S} a \ \wedge \ \bigwedge_{a \in \bar{S}} \bar{a}$$

can be encoded as $\boldsymbol{x}^\top \boldsymbol{y} = |S|$, where $\forall i \in [n], \boldsymbol{x}_i$ is defined as

$$\boldsymbol{x}_i = \begin{cases} 1 & \text{if } a_i \in S \\ -1 & \text{if } a_i \in \bar{S} \\ 0 & \text{otherwise} \end{cases}$$

See Appendix A.1 for an attribute-hiding predicate encoding for the predicate $\mathsf{P}((\boldsymbol{x}, \gamma), \boldsymbol{y}) := \boldsymbol{x}^\top \boldsymbol{y} = \gamma$.

In this context, a disjunction of k predicate encodings like the former allows to encode boolean formulas whose representation has at most k disjunction terms. Additionally, decryption will need k tries in the worst case. This is because since $\boldsymbol{x}$ is unknown, the decryptor has to guess which of the disjunctions his key does satisfy (if any).

5.4 Dual-Policy ABE

Dual-Policy Attribute Based Encryption [7, 8] is an interesting primitive that has already been considered in the *pair encodings* framework. It combines KP-ABE and CP-ABE into a single construction that simultaneously allows two access control mechanisms. The main advantage is the possibility of considering policies over *objective* attributes (associated to data) and policies over *subjective* attributes (associated to user credentials) at the same time.

Given an arbitrary predicate encoding for P, applying Theorem 6 we can get a predicate encoding for the dual predicate $\mathsf{P}^\star$. Now, by combining the two with Theorem 3, we get dual policy predicate encoding, for the predicate

$$\mathsf{P}(x_1, y_1) \wedge \mathsf{P}^\star(y_2, x_2)$$

The same construction has been considered and built in the same way in the framework of *pair encodings* [7, 8]. As far as we know, our approach is the first that considers this primitive based on *predicate encodings*.

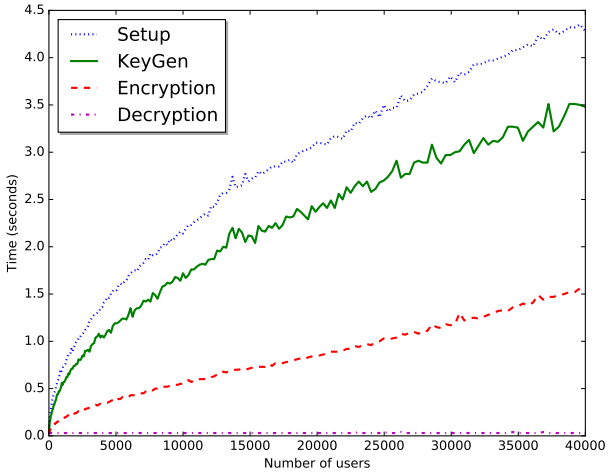

Fig. 5. Scalability of the predicate encoding for revocation

5.5 Revocation

Another application of our techniques is predicate encryption with revocation. Our idea is to combine a *boolean formula predicate encoding* with a *broadcast encryption* predicate encoding. The former is used to encode the actual policy of the scheme, while the latter takes care of the revocation part.

Broadcast encryption has been considered in the literature to approach revocation [16, 20, 27]. In broadcast encryption, a broadcasting authority encrypts a message in such a way that only authorized users will be able to decrypt it. This can be expressed with the following predicate:

$$\mathcal{X} = \{0,1\}^n, \mathcal{Y} = [n] \text{ and } \mathsf{P}(x,y) \Leftrightarrow \boldsymbol{x}_y = 1$$

Note that the number of users in the system, n, has to be fixed and polynomial size. Figure 5 shows that this method can be applied to implement full direct revocation predicate encryption schemes. The system supports thousands of users in reasonable time.

5.6 Delegation

Delegation of keys is a desirable property for every predicate encryption scheme. Roughly, it allows the owner of a secret key to weaken his key and create a new one that is less powerful than the original one. This property can be used to achieve *forward secrecy*, where past sessions are protected from the compromise of future secret keys. To achieve this goal, users can periodically weaken their secret keys and destroy the more powerful ones. In this way, past ciphertexts cannot be decrypted any more and thus, they are protected against the theft of

Let $\mathcal{U} = \{a, b, c\}$ be the set of attributes. Using the predicate encoding for monotonic boolean formulas from [12]:

$$\mathcal{X} = \{0,1\}^3, \quad \mathcal{Y} = \mathbb{Z}_p^{3\times 2}, \quad \mathcal{S} = \mathbb{Z}_p^3, \quad \mathcal{R} = \mathbb{Z}_p^3, \quad \mathcal{W} = \mathbb{Z}_p^3 \times \mathbb{Z}_p^1$$

$$\mathsf{sE}_{\boldsymbol{x}} = \left(\, \mathsf{diag}(\boldsymbol{x}) \mid \mathbf{0} \,\right) \qquad \mathsf{rE}_M = \left(\, I_3 \mid M_{\{2\}} \,\right) \qquad \mathsf{kE}_M = \left(\, M_{\{1\}} \,\right)$$

The following,

$$\mathsf{rE}_M = \left(\begin{array}{ccc|c|c} 1&0&0&1&0\\ 0&1&0&1&0\\ 0&0&1&1&0\\ \hline 0&0&0&0&1 \end{array}\right) \qquad \mathsf{kE}_M = \left(\begin{array}{c} 1\\0\\1\\ \hline 0 \end{array}\right)$$

is a predicate encoding key for the formula $(a \vee c) \wedge b$. Let's assume we want to weaken this key to one for the formula $a \wedge b \wedge c$. Note that in this case we want to make an update of the matrix M:

$$M = \begin{pmatrix} 1&1\\0&1\\1&1 \end{pmatrix} \longrightarrow M' = \begin{pmatrix} 1&1&1\\0&1&1\\1&1&0 \end{pmatrix}$$

That can be done by multiplying R_M from the left by A

$$\underbrace{\begin{pmatrix} 1&0&0&1\\0&1&0&1\\0&0&1&0\\0&0&0&0 \end{pmatrix}}_{A} \cdot \underbrace{\left(\begin{array}{ccc|c|c} 1&0&0&1&0\\ 0&1&0&1&0\\ 0&0&1&1&0\\ \hline 0&0&0&0&1 \end{array}\right)}_{\mathsf{rE}_M} = \left(\begin{array}{ccc|c|c} 1&0&0&1&1\\ 0&1&0&1&1\\ 0&0&1&1&0\\ \hline 0&0&0&0&0 \end{array}\right) \qquad \mathsf{kE}_M = \left(\begin{array}{c} 1\\0\\1\\ \hline 0 \end{array}\right)$$

Fig. 6. Example of delegation of keys for monotonic boolean formulas. Since A is a linear function, it can be computed in the exponent from the given key.

future keys. Delegation is also required for Hierarchical Identity Based Encryption (HIBE).

More formally, we say that a predicate $\mathsf{P}(x, y)$, $x \in \mathcal{X}$, $y \in \mathcal{Y}$ supports delegation if there is a partial ordering $\leq$ on $\mathcal{Y}$ such that for every $x \in \mathcal{X}$ the predicate $\mathsf{P}(x, \cdot)$ is monotone, this is

$$(y \leq y') \wedge \mathsf{P}(x, y) = 1 \;\Rightarrow\; \mathsf{P}(x, y') = 1$$

Delegation has been considered in [12] as the property of some predicate encodings. In this section we propose a generic method to convert any predicate encoding into one supporting delegation. The underlying idea is that, as evidenced by Theorem 1, the power of secret keys in every predicate encoding relies on the *row span* of R_y. Therefore, performing row operations and removing some rows from this matrix produces less powerful keys. Our idea is to add a dummy part to the predicate encoding matrix R_y that does not affect the actual scheme, but can be used to weaken the row span of R_y in the desired way.

Theorem 7 (Delegation). *Let* $(\mathsf{sE}^1, \mathsf{rE}^1, \mathsf{kE}^1)$ *be a predicate encoding for* P*. The following is also a valid predicate encoding for* P*:*

$$\begin{aligned}\mathsf{sE}_{x_1}(\boldsymbol{w}, \boldsymbol{w}') &= \mathsf{sE}^1_{x_1}(\boldsymbol{w})\\ \mathsf{rE}_{y_1}(\boldsymbol{w}, \boldsymbol{w}') &= (\mathsf{rE}^1_{y_1}(\boldsymbol{w}), \boldsymbol{w}')\\ \mathsf{kE}_{y_1} &= (\mathsf{kE}^1_{y_1}, \mathbf{0}_{\mathbb{Z}_p^k})\end{aligned}$$

where $\mathcal{S} = \mathcal{S}_1$*,* $\mathcal{R} = \mathcal{R}_1 \times \mathbb{Z}_p^k$ *and* $\mathcal{W} = \mathcal{W}_1 \times \mathbb{Z}_p^k$ *for arbitrary* $k \in \mathbb{N}$*.*

Proof. For all $(x_1, y_1) \in \mathcal{X}_1 \times \mathcal{Y}_1$, such that $\mathsf{P}(x_1, y_1)$, Lemma 1 ensures that there exist decoding functions $\mathsf{sD}^1_{x_1,y_1}$, $\mathsf{rD}^1_{x_1,y_1}$ for pe^1. The following are valid decoding functions for pe:

$$\begin{aligned}\mathsf{sD}_{x_1,y_1}(\boldsymbol{s}) &= \mathsf{sD}^1_{x_1,y_1}(\boldsymbol{s})\\ \mathsf{rD}_{x_1,y_1}(\boldsymbol{r}_1, \boldsymbol{r}_2) &= \mathsf{rD}^1_{x_1,y_1}(\boldsymbol{r}_1)\end{aligned}$$

For all $(x_1, y_1) \in \mathcal{X}_1 \times \mathcal{Y}_1$, such that $\neg\mathsf{P}(x_1, y_1)$, Lemma 2 ensures that there exists a valid privacy witness $\mathsf{rW}^1_{x_1,y_1}$ for pe^1. The following is a valid privacy witness for pe:

$$\mathsf{rW}_{x_1,y_1} = \left(\mathsf{rW}^1_{x_1,y_1}, \mathbf{0}_{\mathbb{Z}_p^k}\right)$$ □

The additional set of not-null rows in rE_y can be used to weaken the linear span of rE^1_y. In particular, this method works really well for monotonic boolean formulas. (See Figure 6 for an example).

6 Implementation

We have implemented and evaluated our results for different policy sizes. All the experiments were executed on a 8-core machine with 2.40GHz Intel Core i7-3630QM CPU and 8GB of RAM. Our tool is written in Ocaml, using the Relic-Toolkit [3] for pairings. We use a 256-bits Barreto-Naehrig Elliptic Curve [9].

6.1 Comparison with existing predicate encodings

We first compare our improved version of predicate encodings (see Sections 5.1 and 5.2) with earlier constructions. To this end, we generate random boolean formulas for different sizes, starting from a random set of leaf nodes and combining them with boolean operators $\vee$ and $\wedge$. Our tables report on the average time.

In Figure 7 we confirm that our new encoding needs 50% less time than the original algorithms for setup, encryption and key generation. For decryption the performance is similar, as explained in Section 3.2. We note that in terms of secret key size, our encoding is smaller. Our simplification lemmas cannot be applied to the old encoding, therefore the size of secret keys using the older encoding is constant (it is equal to the number of attributes in the system).

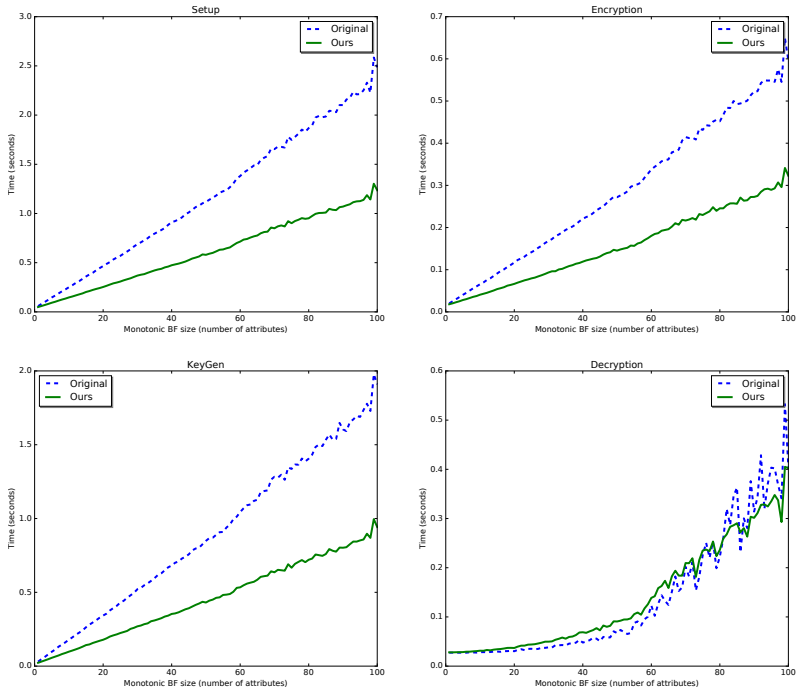

Fig. 7. Improved predicate encoding for boolean formulas vs original encoding

However, in our encoding, the size of secret keys is given by the number of *or-gates* in the policy[2] and thus, simple policies require smaller keys. In the worst case, key size is equal to the old encoding (the number of gates cannot be larger than the number of attributes).

Figure 8 shows the performance of our new encoding for KP-ABE for Arithmetic Span Programs compared to the original encoding from [12]. As we expected, our encoding needs 66% of the time required for the original encoding for setup, encryption and key generation. Additionally, secret key size is halved with our encoding.

6.2 Pair Encodings

As a contribution of independent interest, we implement *pair encodings* in the prime order version of [1] (*pair encodings*). Then we compare the CP-ABE pred-

[2] Note that $\mathsf{rE}_M(\boldsymbol{w}) = M^\top \boldsymbol{w}$. If we consider Lewko and Waters' algorithm to build M from boolean formulas [26], the number of non-zero columns in M is given by the number of *and-gates* in the formula. This is, the number of non-zero rows of $M^\top$ and also the number of elements in the secret key. In practice, as described in Section 5.1, our simpler encoding works with negated policies and thus, instead of the number of *and-gates*, secret keys sizes will be given by the number of *or-gates*.

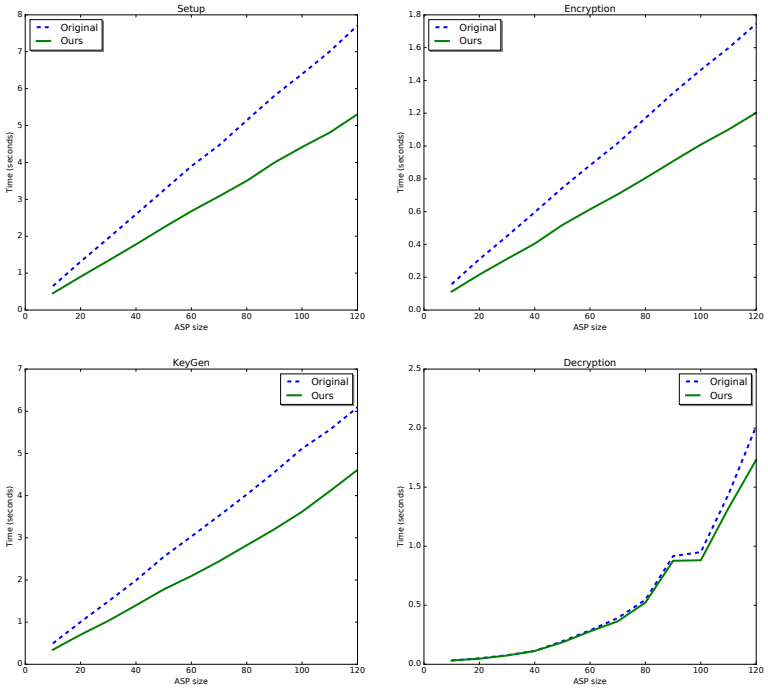

Fig. 8. Improved predicate encoding for ASP vs original encoding

icate encoding for monotonic boolean formulas from [12] and the CP-ABE pair encoding for monotonic boolean formulas (and constant ciphertexts) from [1]. In Figure 9, we show the computation time and operations needed for setup, encryption, key generation and decryption algorithms in ABE supporting attribute universes of different size. Note that, for convenience, vertical axis of key generation and decryption are in log scale. For the setup algorithm we consider the time and the number of random samples needed. For encryption and key generation we measure time and the number of group additions and multiplications in $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively. For decryption we consider time and the number of pairing operation count. Note that the compiler from [12] (pair encodings) needs a constant number of pairing operations (two pairings), while for the compilers from [5, 1] (pair encodings) the number of pairings grows with the size of the constructions (in particular, the number of pairings needed during decryption is the product of the key size and the ciphertext size). This comparison suggests that *predicate encodings* have better performance, but further investigation is required to obtain a better understanding about this two independent primitives.

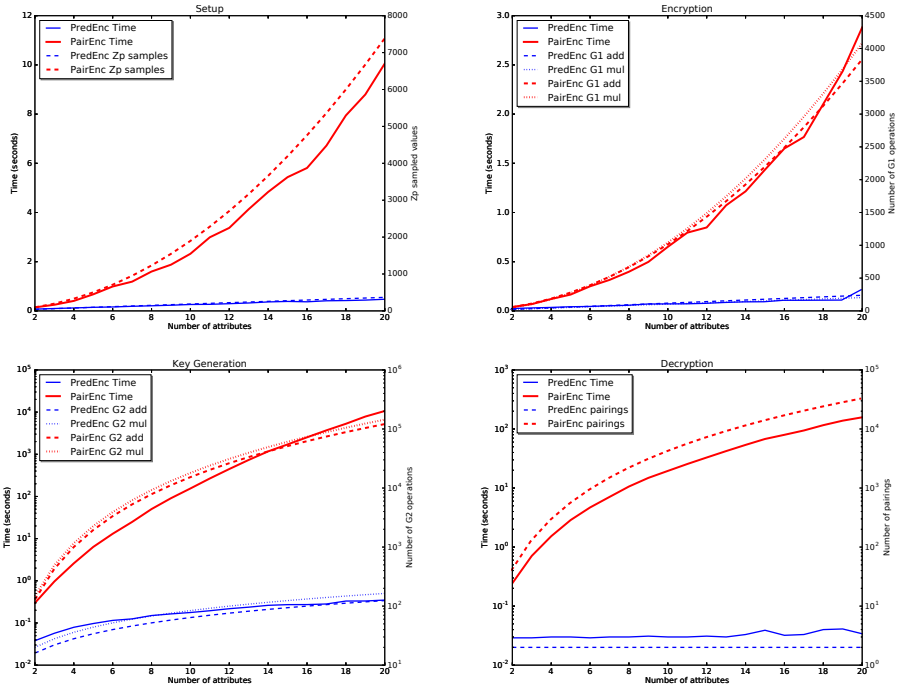

Fig. 9. Comparison between *predicate encodings* and *pair encodings*

Acknowledgements The work presented here was supported by projects S2013/ICE-2731 N-GREENS Software-CM, ONR Grants N000141210914 and N000141512750, as well as FP7 Marie Curie Actions-COFUND 291803.

References

1. S. Agrawal and M. Chase. *A Study of Pair Encodings: Predicate Encryption in Prime Order Groups*, pages 259–288. Springer Berlin Heidelberg, Berlin, Heidelberg, 2016.
2. J. A. Akinyele, C. U. Lehmann, M. D. Green, M. W. Pagano, Z. N. J. Peterson, and A. D. Rubin. Self-protecting electronic medical records using attribute-based encryption. Cryptology ePrint Archive, Report 2010/565, 2010. `http://eprint.iacr.org/2010/565`.
3. D. F. Aranha and C. P. L. Gouvêa. RELIC is an Efficient LIbrary for Cryptography. `https://github.com/relic-toolkit/relic`.
4. N. Attrapadung. Dual system encryption via doubly selective security: Framework, fully secure functional encryption for regular languages, and more. In P. Q. Nguyen and E. Oswald, editors, *Advances in Cryptology – EUROCRYPT 2014*, volume 8441 of *Lecture Notes in Computer Science*, pages 557–577, Copenhagen, Denmark, May 11–15, 2014. Springer, Heidelberg, Germany.

5. N. Attrapadung. Dual system encryption framework in prime-order groups. Cryptology ePrint Archive, Report 2015/390, 2015. http://eprint.iacr.org/2015/390.
6. N. Attrapadung and H. Imai. *Conjunctive Broadcast and Attribute-Based Encryption*, pages 248–265. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.
7. N. Attrapadung and H. Imai. *Dual-Policy Attribute Based Encryption*, pages 168–185. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.
8. N. Attrapadung and S. Yamada. Duality in abe: Converting attribute based encryption for dual predicate and dual policy via computational encodings. Cryptology ePrint Archive, Report 2015/157, 2015. http://eprint.iacr.org/2015/157.
9. P. S. L. M. Barreto and M. Naehrig. Pairing-friendly elliptic curves of prime order. Cryptology ePrint Archive, Report 2005/133, 2005. http://eprint.iacr.org/2005/133.
10. A. Beimel. *Secret-Sharing Schemes: A Survey*, pages 11–46. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.
11. D. Boneh and B. Waters. Conjunctive, subset, and range queries on encrypted data. In S. P. Vadhan, editor, *TCC 2007: 4th Theory of Cryptography Conference*, volume 4392 of *Lecture Notes in Computer Science*, pages 535–554, Amsterdam, The Netherlands, Feb. 21–24, 2007. Springer, Heidelberg, Germany.
12. J. Chen, R. Gay, and H. Wee. Improved dual system ABE in prime-order groups via predicate encodings. In E. Oswald and M. Fischlin, editors, *Advances in Cryptology – EUROCRYPT 2015, Part II*, volume 9057 of *Lecture Notes in Computer Science*, pages 595–624, Sofia, Bulgaria, Apr. 26–30, 2015. Springer, Heidelberg, Germany.
13. J. Chen and H. Wee. Fully, (almost) tightly secure IBE and dual system groups. In R. Canetti and J. A. Garay, editors, *Advances in Cryptology – CRYPTO 2013, Part II*, volume 8043 of *Lecture Notes in Computer Science*, pages 435–460, Santa Barbara, CA, USA, Aug. 18–22, 2013. Springer, Heidelberg, Germany.
14. J. Chen and H. Wee. Dual system groups and its applications — compact hibe and more. Cryptology ePrint Archive, Report 2014/265, 2014. http://eprint.iacr.org/2014/265.
15. T. T. A. Dinh and A. Datta. 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*, pages 13–24, 2014.
16. S. Garg, A. Kumarasubramanian, A. Sahai, and B. Waters. Building efficient fully collusion-resilient traitor tracing and revocation schemes. In E. Al-Shaer, A. D. Keromytis, and V. Shmatikov, editors, *ACM CCS 10: 17th Conference on Computer and Communications Security*, pages 121–130, Chicago, Illinois, USA, Oct. 4–8, 2010. ACM Press.
17. R. Gay, I. Kerenidis, and H. Wee. Communication complexity of conditional disclosure of secrets and attribute-based encryption. In R. Gennaro and M. J. B. Robshaw, editors, *Advances in Cryptology – CRYPTO 2015, Part II*, volume 9216 of *Lecture Notes in Computer Science*, pages 485–502, Santa Barbara, CA, USA, Aug. 16–20, 2015. Springer, Heidelberg, Germany.
18. Y. Ishai and H. Wee. *Partial Garbling Schemes and Their Applications*, pages 650–662. Springer Berlin Heidelberg, Berlin, Heidelberg, 2014.
19. F. G. Jongkil Kim, Willy Susilo and M. H. Au. A tag based encoding: An efficient encoding for predicate encoding in prime order groups. Cryptology ePrint Archive, Report 2016/655, 2016. http://eprint.iacr.org/2016/655.

20. P. Junod and A. Karlov. An efficient public-key attribute-based broadcast encryption scheme allowing arbitrary access policies. In *Proceedings of the Tenth Annual ACM Workshop on Digital Rights Management*, DRM '10, pages 13–24, New York, NY, USA, 2010. ACM.
21. M. Karchmer and A. Wigderson. On span programs. In *In Proc. of the 8th IEEE Structure in Complexity Theory*, pages 102–111. IEEE Computer Society Press, 1993.
22. M. Karchmer and A. Wigderson. On span programs. In *Structure in Complexity Theory Conference, 1993., Proceedings of the Eighth Annual*, pages 102–111, May 1993.
23. J. Katz, A. Sahai, and B. Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In N. P. Smart, editor, *Advances in Cryptology – EUROCRYPT 2008*, volume 4965 of *Lecture Notes in Computer Science*, pages 146–162, Istanbul, Turkey, Apr. 13–17, 2008. Springer, Heidelberg, Germany.
24. A. B. Lewko. Tools for simulating features of composite order bilinear groups in the prime order setting. In D. Pointcheval and T. Johansson, editors, *Advances in Cryptology – EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 318–335, Cambridge, UK, Apr. 15–19, 2012. Springer, Heidelberg, Germany.
25. A. B. Lewko and B. Waters. New techniques for dual system encryption and fully secure HIBE with short ciphertexts. In D. Micciancio, editor, *TCC 2010: 7th Theory of Cryptography Conference*, volume 5978 of *Lecture Notes in Computer Science*, pages 455–479, Zurich, Switzerland, Feb. 9–11, 2010. Springer, Heidelberg, Germany.
26. A. B. Lewko and B. Waters. Decentralizing attribute-based encryption. In K. G. Paterson, editor, *Advances in Cryptology – EUROCRYPT 2011*, volume 6632 of *Lecture Notes in Computer Science*, pages 568–588, Tallinn, Estonia, May 15–19, 2011. Springer, Heidelberg, Germany.
27. Z. Liu and D. S. Wong. *Practical Ciphertext-Policy Attribute-Based Encryption: Traitor Tracing, Revocation, and Large Universe*, pages 127–146. Springer International Publishing, Cham, 2015.
28. D. Lubicz and T. Sirvent. Attribute-based broadcast encryption scheme made efficient. In S. Vaudenay, editor, *AFRICACRYPT 08: 1st International Conference on Cryptology in Africa*, volume 5023 of *Lecture Notes in Computer Science*, pages 325–342, Casablanca, Morocco, June 11–14, 2008. Springer, Heidelberg, Germany.
29. A. Sahai, H. Seyalioglu, and B. Waters. Dynamic credentials and ciphertext delegation for attribute-based encryption. In R. Safavi-Naini and R. Canetti, editors, *Advances in Cryptology – CRYPTO 2012*, volume 7417 of *Lecture Notes in Computer Science*, pages 199–217, Santa Barbara, CA, USA, Aug. 19–23, 2012. Springer, Heidelberg, Germany.
30. A. Sahai and B. R. Waters. Fuzzy identity-based encryption. In R. Cramer, editor, *Advances in Cryptology – EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 457–473, Aarhus, Denmark, May 22–26, 2005. Springer, Heidelberg, Germany.
31. A. Shamir. Identity-based cryptosystems and signature schemes. In G. R. Blakley and D. Chaum, editors, *Advances in Cryptology – CRYPTO'84*, volume 196 of *Lecture Notes in Computer Science*, pages 47–53, Santa Barbara, CA, USA, Aug. 19–23, 1984. Springer, Heidelberg, Germany.

32. F. Wang, J. Mickens, N. Zeldovich, and V. Vaikuntanathan. Sieve: Cryptographically enforced access control for user data in untrusted clouds. In *13th USENIX Symposium on Networked Systems Design and Implementation (NSDI 16)*, pages 611–626, Santa Clara, CA, Mar. 2016. USENIX Association.
33. B. Waters. Dual system encryption: Realizing fully secure IBE and HIBE under simple assumptions. In S. Halevi, editor, *Advances in Cryptology – CRYPTO 2009*, volume 5677 of *Lecture Notes in Computer Science*, pages 619–636, Santa Barbara, CA, USA, Aug. 16–20, 2009. Springer, Heidelberg, Germany.
34. H. Wee. Dual system encryption via predicate encodings. In Y. Lindell, editor, *TCC 2014: 11th Theory of Cryptography Conference*, volume 8349 of *Lecture Notes in Computer Science*, pages 616–637, San Diego, CA, USA, Feb. 24–26, 2014. Springer, Heidelberg, Germany.

A Appendix

A.1 Predicate encoding for γ-inner product

The following is a valid predicate encoding for the predicate: $\mathsf{P}((\boldsymbol{x},\gamma),\boldsymbol{y}) = \boldsymbol{x}^\top \boldsymbol{y} = \gamma$, where

$$\mathcal{X} = \mathbb{Z}_p^n \times \mathbb{Z}_p, \quad \mathcal{Y} = \mathbb{Z}_p^n, \quad \mathcal{S} = \mathbb{Z}_p^n \times \mathbb{Z}_p, \quad \mathcal{R} = \mathbb{Z}_p^1, \quad \mathcal{W} = \mathbb{Z}_p \times \mathbb{Z}_p^n \times \mathbb{Z}_p$$

$$\mathsf{sE}_{\boldsymbol{x}} = \left(\begin{array}{c|c|c} \boldsymbol{x} & I_n & \boldsymbol{0} \\ \hline \gamma & \boldsymbol{0}^\top & 1 \end{array}\right) \qquad \mathsf{rE}_{\boldsymbol{y}} = \left(\begin{array}{c|c|c} 0 & \boldsymbol{y}^\top & -1 \end{array}\right) \qquad \mathsf{kE}_{\boldsymbol{y}} = \left(1\right)$$

This encoding is *attribute-hiding*. In particular, valid decryption vectors are

$$\mathsf{sD}_{x,y} = \left(\begin{array}{c} \boldsymbol{y} \\ \hline -1 \end{array}\right) \qquad \mathsf{rD}_{x,y} = \left(1\right)$$

(the do not depend on $\boldsymbol{x}$).

A.2 Generalized predicate encoding for broadcast encryption

We generalize the predicate encoding for broadcast encryption from [12]. Let $\mathcal{X} = \mathbb{Z}_p^n$, $\mathcal{Y} = [n] \times \mathbb{Z}_p$, we consider the predicate

$$\mathsf{P}(\boldsymbol{x}, (i,\gamma)) \Leftrightarrow \boldsymbol{x}_i = \gamma.$$

As in [12], it is convenient to express the above predicate as follows:
$\mathcal{X} = (\mathbb{Z}_p^n/t)^t$, $\mathcal{Y} = [t] \times [n/t] \times \mathbb{Z}_p$ and

$$\mathsf{P}((\boldsymbol{x}_1,\ldots,\boldsymbol{x}_t),(i_1,i_2,\gamma)) \Leftrightarrow \boldsymbol{x}_{i_1}^\top \boldsymbol{e}_{i_2} = \gamma$$

where (i_1,i_2) is the unique pair of integers satisfying $i = (i_1-1)\cdot n/t + i_2$ and $0 < i_2 \le n/t$. Also, $(\boldsymbol{e}_1,\ldots,\boldsymbol{e}_{n/t})$ is the standard basis of $\mathbb{Z}_p^{n/t}$. The following is a valid predicate encoding for the above predicate:

$$\mathcal{S} = \mathbb{Z}_p^t, \quad \mathcal{R} = \mathbb{Z}_p^{n/t}, \quad \mathcal{W} = \mathbb{Z}_p^t \times \mathbb{Z}_p^{n/t}$$

$$\mathsf{sE}_x = \left(\frac{I_t}{\boldsymbol{x}_1 \ \ldots \ \boldsymbol{x}_t}\right)^{\top} \quad \mathsf{rE}_{(i_1,i_2,\gamma)} = \big(\, \mathbf{0}\ \underset{\substack{\uparrow \\ i_1}}{\boldsymbol{e}_{i_2}}\ \mathbf{0} \mid \gamma \cdot I_{(n/t)} \,\big) \quad \mathsf{kE}_{(i_1,i_2,\gamma)} = \big(\, \boldsymbol{e}_{i_2} \,\big)\,.$$

This encoding can be used to perform 2-dimensional broadcast encryption. This is, users are divided in n groups and every user i has a unique identifier γ_i. Encryption can be done in such a way that at most one user from every group will be able to decrypt.