Dually Computable Cryptographic Accumulators and Their Application to Attribute Based Encryption

Anaïs Barthoulot*[1,2], Olivier Blazy[3], and Sébastien Canard[4]

[1] Orange Innovation, Caen, France
[2] Université de Limoges, XLim, France
[3] École Polytechnique, Palaiseau, France
[4] Télécom Paris, Palaiseau, France
anais.barthoulot@gmail.com, olivier.blazy@polytechnique.edu
sebastien.canard@telecom-paris.fr

Abstract. In 1993, Benaloh and De Mare introduced cryptographic accumulator, a primitive that allows the representation of a set of values by a short object (the accumulator) and offers the possibility to prove that some input values are in the accumulator. For this purpose, so-called *asymmetric* accumulators require the creation of an additional cryptographic object, called a *witness*. Through the years, several instantiations of accumulators were proposed either based on number theoretic assumptions, hash functions, bilinear pairings or more recently lattices. In this work, we present the first instantiation of an asymmetric cryptographic accumulator that allows private computation of the accumulator but public witness creation. This is obtained thanks to our unique combination of the pairing based accumulator of Nguyen with *dual pairing vector spaces*. We moreover introduce the new concept of *dually computable* cryptographic accumulators, in which we offer two ways to compute the representation of a set: either privately (using a dedicated secret key) or publicly (using only the scheme's public key), while there is a unique witness creation for both cases. All our constructions of accumulators have constant size accumulated value and witness, and satisfy the accumulator security property of *collision resistance*, meaning that it is not possible to forge a witness for an element that is not in the accumulated set. As a second contribution, we show how our new concept of dually computable cryptographic accumulator can be used to build a Ciphertext Policy Attribute Based Encryption (CP-ABE). Our resulting scheme permits policies expressed as disjunctions of conjunctions (without "NO" gates), and is adaptively secure in the standard model. This is the first CP-ABE scheme having both constant-size user secret keys and ciphertexts (i.e. independent of the number of attributes in the scheme, or the policy size). For the first time, we provide a way to use cryptographic accumulators for both key management and encryption process.

Keywords: Cryptographic accumulators · Attribute based encryption · Pairing · Dual pairing vector spaces

1 Introduction

Cryptographic accumulator. Cryptographic accumulators were introduced in 1993 by Benaloh and De Mare [6] as a compact way to represent a set of elements, while permitting to prove the membership for each element in the set. Since their introduction, lots of new functionalities and properties were introduced and we refer interested readers to the work of Derler *et al.* [14] for more details on cryptographic accumulators. In this work, we focus on asymmetric accumulators, which are composed of four algorithms: Gen, the generation algorithm that outputs a public key and a master secret key; Eval, the evaluation algorithm that from a set of elements outputs the compact representation of this set (which is called the "accumulator"); $\mathsf{WitCreate}$, the witness creation algorithm that creates a witness that an element is the set; Verify, the verification algorithm that outputs 1 if the given witness proves that the element is indeed in the accumulated set. If the algorithm Eval (resp. $\mathsf{WitCreate}$) takes as input the master secret key we say that the evaluation (resp. witness generation) is done privately, otherwise it is done publicly. The main purpose of cryptographic accumulators is to produce accumulators and witnesses that have constant size. Regarding security, there are several properties but in this work we will consider the notion of *collision resistance* meaning that given the accumulator public key it is hard for an adversary to find a set $\mathcal{X}$ and a value $y \notin \mathcal{X}$ and build a witness wit_y such that $\mathsf{Verify}(\mathsf{pk}_{\mathsf{acc}}, \mathsf{acc}_{\mathcal{X}}, \mathsf{wit}_y) = 1$, where $\mathsf{acc}_{\mathcal{X}} = \mathsf{Eval}((\mathsf{sk}_{\mathsf{acc}}), \mathsf{pk}_{\mathsf{acc}}, \mathcal{X})$.

Improving accumulator's state of the art. Regarding the literature, one surprising thing is that there is no accumulator with private evaluation and public witness generation: either both evaluation and witness creation are either public [28] or private [19], or witness generation is private while evaluation is public [29]. As soon as the accumulator has been secretly computed and publish, it could be relevant for some use cases to consider the case where anyone can prove that one element is in the chosen set. In the sequel, we show how this property can be used to construct encryption schemes from a cryptographic accumulator. Therefore, we propose the first instantiation of such accumulator, based on asymmetric pairings in prime order group and using dual pairing vector spaces. We also introduce the notion of *dually computable* accumulator, which permits both a private (Eval) and a public ($\mathsf{PublicEval}$) accumulator generation, such that both accumulators are distinguishable. From a unique witness generation algorithm, we add two associated verifications (Verify and $\mathsf{PublicVerify}$ respectively) to verify set membership. Using our previous accumulator instantiation, we derive the first dually computable accumulator scheme. We then show how such new concept can be used to provide an efficient Attribute Based Encryption (ABE) scheme.

Attribute Based Encryption. ABE, introduced by Sahai and Waters in 2005 [34], is an encryption scheme in which secret keys and ciphertexts are associated to some subset of attributes, and decryption is possible if there exists a relation between the secret key's attributes and the ciphertext's attributes. In more details, in a *Ciphertext Policy ABE* (CP-ABE) the ciphertext is associated to an access policy while the secret key is associated to a set of attributes. Decryption becomes possible if the set of attributes satisfies the policy. There exist several ways to define an access policy in the literature: through threshold structure [34], tree-based structure [21], boolean formulas [27], linear secret sharing schemes [39], circuits [9], ... Regarding security, ABE schemes must satisfy *indistinguishability*, meaning that an adversary who is given an encryption of one of two messages he previously chose, cannot tell which message was encrypted. The main aim of research in ABE is to build efficient schemes in terms of both time and space complexities, while supporting complex access policies. Unfortunately, most existing schemes propose ciphertexts with a size linear in the number of attributes in the scheme [21,25,24], while some other constructions succeed in proposing constant size ciphertext, but at the cost of quadratic-size user private key [4].

ABE from dually computable accumulators. In this paper, we propose a way to obtain an ABE scheme for which both the ciphertext and the user secret key are constant, while obtaining very good time complexities. For that, our idea is to use cryptographic accumulators. Curiously, while the purpose of the latter is to make constant the size of cryptographic objects, few attempts have been done to use them for encryption schemes. Indeed, [18,2] propose broadcast encryption schemes that use (RSA based) cryptographic accumulator, and more recently, Wang and Chow [37] present an identity based broadcast encryption scheme that uses a degenarated notion of accumulators. However, [18,2] are using accumulators only to manage users' secret key while [37] is using their notion of accumulator for encryption only, whereas in our scheme, accumulators are used for both secret keys and ciphertexts. Plus, (identity based) broadcast encryption is one particular case of ABE, which makes our scheme more general.

To reach such objective of compactness, our idea is to employ our notion of dually computable accumulators in the following manner: the secret key, computed by the authority, corresponds to a privately computed accumulator of the users' attributes set, while the encryption corresponds to a one-time-pad with a mask derived from a publicly computed accumulator of the access policy. Decryption is then possible if the decryptor can demonstrate that the intersection of their accumulator and the one associated with the ciphertext is not empty, utilizing membership witnesses for both the privately computed and the publicly computed accumulators. However, while it is relatively straightforward to use accumulators to represent sets of attributes, understanding how they can serve as a concise representation of access policies is more complex. In this study, we introduce a way to represent monotone boolean formulas that is compatible with the use of accumulators, and then show how to employ dually computable accumulators to obtain a compact, efficient and secure ABE.

Our contributions. As a summary, our work gives the three following contributions:

- a new accumulator scheme, based on [30]'s work. It is the first accumulator in the literature that has private evaluation while having public witness creation. This scheme is based on asymmetric pairings in prime order groups and dual pairing vector spaces (DPVS) of dimension 2, and satisfies *collision resistance* under the q-Strong Bilinear Diffie-Hellman assumption. This is the first construction of cryptographic accumulators that uses DPVS. See Section 3;
- a new functionality of *dually computable* cryptographic accumulators, together with an instantiation of a such accumulator, based on our first accumulator instantiation. Details are given in Section 4;
- a new bounded CP-ABE scheme, with both constant size for ciphertexts and user secret keys where access policies are monotone NC^1 circuits. Our scheme moreover gives very good time complexities, and is proven adaptively secure in the standard model, under the standard SXDH assumption. See Section 5.

2 Preliminaries

This section introduces the notations, the building blocks and the security assumptions used throughout this paper. Let "PPT" denote "probabilistic polynomial-time". For every finite set S, $x \leftarrow S$ denotes a uniformly random element x from the set S. Vectors are written with **bold face** lower case letters and matrices with **bold face** upper case letters.

2.1 Cryptographic accumulators

In the following we present a simplified definition of accumulator, presenting only properties used in this work, for simplicity of reading. Refer to [14] for a complete definition of cryptographic accumulators.

Definition 1. ***Static accumulator.*** *[6,16,14] A static cryptographic accumulator scheme is a tuple of efficient algorithms defined as follows:*

- $\mathsf{Gen}(1^\kappa, \mathfrak{b})$*: this algorithm takes as input a security parameter* κ *and a bound* $\mathfrak{b} \in \mathbb{N} \cup \{\infty\}$ *such that if* $\mathfrak{b} \neq \infty$ *then the number of elements that can be accumulated is bounded by* $\mathfrak{b}$*. It returns a key pair* $(\mathsf{sk}_{\mathsf{acc}}, \mathsf{pk}_{\mathsf{acc}})$*, where* $\mathsf{sk}_{\mathsf{acc}} = \varnothing$ *if no trapdoor exists and* $\mathsf{pk}_{\mathsf{acc}}$ *contains the parameter* $\mathfrak{b}$*.*
- $\mathsf{Eval}((\mathsf{sk}_{\mathsf{acc}},)\mathsf{pk}_{\mathsf{acc}}, \mathcal{X})$*: this algorithm takes as input the accumulator (secret key* $\mathsf{sk}_{\mathsf{acc}}$ *and) public key* $\mathsf{pk}_{\mathsf{acc}}$ *and a set* $\mathcal{X}$ *to be accumulated. It returns an accumulator* $\mathsf{acc}_{\mathcal{X}}$ *together with some auxiliary information* aux*.*
- $\mathsf{WitCreate}((\mathsf{sk}_{\mathsf{acc}},)\mathsf{pk}_{\mathsf{acc}}, \mathcal{X}, \mathsf{acc}_{\mathcal{X}}, \mathsf{aux}, x)$*: this algorithm takes as input the accumulator (secret key* $\mathsf{sk}_{\mathsf{acc}}$ *and) public key* $\mathsf{pk}_{\mathsf{acc}}$*, an accumulator* $\mathsf{acc}_{\mathcal{X}}$*, the associated set* $\mathcal{X}$*, auxiliary information* aux*, and an element* x*. It outputs a membership witness* $\mathsf{wit}_x^{\mathcal{X}}$ *if* $x \in \mathcal{X}$*, otherwise it outputs a reject symbol* $\perp$*.*
- $\mathsf{Verify}(\mathsf{pk}_{\mathsf{acc}}, \mathsf{acc}_{\mathcal{X}}, \mathsf{wit}_x^{\mathcal{X}}, x)$*: this algorithm takes as input the accumulator public key* $\mathsf{pk}_{\mathsf{acc}}$*, an accumulator* $\mathsf{acc}_{\mathcal{X}}$*, a witness* $\mathsf{wit}_x^{\mathcal{X}}$ *and an element* x*. If* $\mathsf{wit}_x^{\mathcal{X}}$ *is correct it returns* 1*, otherwise it returns* 0*.*

Definition 2. *If in the above definition* x *can be replaced by a set* S*, we say that the accumulator supports* subset queries*. If any element in* $\mathcal{X}$ *can be present more than once, and witnesses can be made to prove that the element is present a given number of times in* $\mathcal{X}$*, we say that the accumulator supports* multisets *setting.*

Note 1. Sometimes $\mathsf{wit}_x^{\mathcal{X}}$ is simply written wit_x.

Informally, the *correctness* property says that for all honestly generated keys, all honestly computed accumulators and witnesses, the Verify algorithm will always return 1.

Definition 3. ***Correctness of accumulators.*** *A static accumulator is said to be* correct *if for all security parameters* κ*, all integer* $\mathfrak{b} \in \mathbb{N} \cup \{\infty\}$*, all set of values* $\mathcal{X}$*, and all element* x *such that* $x \in \mathcal{X}$*:*

$$\Pr\left[\begin{array}{c}\mathsf{sk}_{\mathsf{acc}}, \mathsf{pk}_{\mathsf{acc}} \leftarrow \mathsf{Gen}(1^\kappa, \mathfrak{b}), \mathsf{acc}_{\mathcal{X}}, \mathsf{aux} \leftarrow \mathsf{Eval}((\mathsf{sk}_{\mathsf{acc}},)\mathsf{pk}_{\mathsf{acc}}, \mathcal{X}), \\ \mathsf{wit}_x^{\mathcal{X}} \leftarrow \mathsf{WitCreate}((\mathsf{sk}_{\mathsf{acc}},)\mathsf{pk}_{\mathsf{acc}}, \mathcal{X}, \mathsf{acc}_{\mathcal{X}}, \mathsf{aux}, x): \\ \mathsf{Verify}(\mathsf{pk}_{\mathsf{acc}}, \mathsf{acc}_{\mathcal{X}}, \mathsf{wit}_x, x) = 1\end{array}\right] = 1$$

Regarding security, we will only consider the following definition in this work.

Definition 4. ***Collision resistance.*** *A static accumulator scheme is* collision resistant*, if for all PPT adversaries $\mathcal{A}$ there is a negligible function $\epsilon(.)$ such that:*

$$\Pr\left[\begin{array}{c}\mathsf{sk}_{\mathsf{acc}}, \mathsf{pk}_{\mathsf{acc}} \leftarrow \mathsf{Gen}(1^\kappa, \mathfrak{b}), \mathcal{O} \leftarrow \left\{\mathcal{O}^E, \mathcal{O}^W\right\}, (\mathcal{X}, \mathsf{wit}_x, x) \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{pk}_{\mathsf{acc}}): \\ \mathsf{Verify}(\mathsf{pk}_{\mathsf{acc}}, \mathsf{acc}_{\mathcal{X}}, \mathsf{wit}_x, x) = 1 \wedge x \notin \mathcal{X}\end{array}\right] \leq \epsilon(\kappa),$$

where $\mathsf{acc}_{\mathcal{X}} \leftarrow \mathsf{Eval}((\mathsf{sk}_{\mathsf{acc}},)\mathsf{pk}_{\mathsf{acc}}, \mathcal{X})$ and $\mathcal{A}$ has oracle access to $\mathcal{O}$, where $\mathcal{O}^E$ and $\mathcal{O}^W$ that represent the oracles for the algorithms Eval *and* WitCreate. *An adversary is allowed to query it an arbitrary number of times.*

2.2 Other Preliminaries

Definition 5. ***Asymmetric bilinear pairing groups.*** *[12]* Asymmetric bilinear groups *$\Gamma = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e)$ are tuple of prime p, cyclic (multiplicative) groups $\mathbb{G}_1, \mathbb{G}_2$, $\mathbb{G}_T$ (where $\mathbb{G}_1 \neq \mathbb{G}_2$) of order p, $g_1 \neq 1 \in \mathbb{G}_1$, $g_2 \neq 1 \in \mathbb{G}_2$, and a polynomial-time computable non-degenerate bilinear pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, i.e. $e(g_1^s, g_2^t) = e(g_1, g_2)^{st}$ and $e(g_1, g_2) \neq 1$.*

Note 2. For any group element $g \in \mathbb{G}$, and any vector $\boldsymbol{v}$ of size $l \in \mathbb{N}$, we denote by $g^{\boldsymbol{v}}$ the vector $(g^{v_1}, \cdots, g^{v_l})$. Let $\boldsymbol{u}, \boldsymbol{v}$ be two vectors of length l. Then by $g^{\boldsymbol{u}\cdot\boldsymbol{v}}$, we denote the element g^α, where $\alpha = \boldsymbol{u} \cdot \boldsymbol{v} = u_1 \cdot v_1 + u_2 \cdot v_2 + \cdots + u_l \cdot v_l$. Then we define $e(g_1^{\boldsymbol{v}}, g_2^{\boldsymbol{u}}) := \prod_{i=1}^{l} e(g_1^{v_i}, g_2^{u_i}) = e(g_1, g_2)^{\boldsymbol{v}\cdot\boldsymbol{u}}$.

Definition 6. ***Dual pairing vector spaces (DPVS).*** *[12] For a prime p and a fixed (constant) dimension n, we choose two random bases $\mathbb{B} = (\boldsymbol{b}_1, \cdots, \boldsymbol{b}_n)$ and $\mathbb{B}^* = (\boldsymbol{b}_1^*, \cdots, \boldsymbol{b}_n^*)$ of $\mathbb{Z}_p^n$, subject to the constraint that they are* ***dual orthonormal****, meaning that $\boldsymbol{b}_i \cdot \boldsymbol{b}_j^* = 0 \pmod p$ whenever $i \neq j$, and $\boldsymbol{b}_i \cdot \boldsymbol{b}_i^* = \psi \pmod p$ for all i, where ψ is a uniformly random element of $\mathbb{Z}_p$. Here the elements of $\mathbb{B}, \mathbb{B}^*$ are vectors and $\cdot$ corresponds to the scalar product. We denote such algorithm as* $\mathsf{Dual}(\mathbb{Z}_p^n)$. *For generators $g_1 \in \mathbb{G}_1$ and $g_2 \in \mathbb{G}_2$, we note that $e(g_1^{\boldsymbol{b}_i}, g_2^{\boldsymbol{b}_j^*}) = 1$ whenever $i \neq j$.*

Note 3. In our constructions we will use the notation $(\mathbb{D}, \mathbb{D}^*)$ to also denote dual orthonormal bases, as in our ABE security proof, we will handle more than one pair of dual orthonormal bases at a time, and we think that a different notation will avoid confusion. The notation $(\mathbb{F}, \mathbb{F}^*)$ will also be used in the proof for dual orthonormal bases.

Definition 7. ***Characteristic Polynomial.*** *[17,19]. A set $\mathcal{X} = \{x_1, \cdots, x_n\}$ with elements $x_i \in \mathbb{Z}_p$ can be represented by a polynomial. The following polynomial $\mathsf{Ch}_{\mathcal{X}}[z] = \prod_{i=1}^{n}(x_i + Z)$ from $\mathbb{Z}_p[Z]$, where Z is a formal variable, is called the* characteristic polynomial *of $\mathcal{X}$. In what follows, we will denote this polynomial simply by $\mathsf{Ch}_{\mathcal{X}}$ and its evaluation at a point y as $\mathsf{Ch}_{\mathcal{X}}(y)$.*

Definition 8. ***Elementary symmetric polynomial.*** *The* elementary symmetric polynomial *on $n \in \mathbb{N}$ variables $\{X_i\}$ of degree $k \leq n$ is the polynomial $\sigma_k(X_1, \cdots, X_n) = \sum\limits_{1 \leq i_1 \nleq \cdots \nleq i_k \leq n} X_{i_1} \cdots X_{i_k}$. Notice that $\sigma_1(X_1, \cdots, X_N) = \sum_{i=1}^{n} X_i$ and σ_n is equal to $\prod_{i=1}^{n} X_i$.*

Note 4. Let $\mathcal{X} = \{X_1, \cdots, X_n\}$. Notice that $\mathsf{Ch}_{\mathcal{X}}[Z]$, which is equals to $\prod_{i=1}^{n}(X_i + Z)$ by definition, is also equals to $Z^n + \sigma_1(X_1, \cdots, X_n)Z^{n-1} + \sigma_2(X_1, \cdots, X_n)Z^{n-2} + \cdots + \sigma_n(X_1, \cdots, X_n)$.

Definition 9. ***Decisional Diffie-Hellman assumption in*** $\mathbb{G}_1$ (DDH_1)**.** *[12] Given an asymmetric bilinear pairing group $\Gamma = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e)$, we define the following distribution: $a, b, c \leftarrow \mathbb{Z}_p$, $D = (\Gamma, g_1, g_2, g_1^a, g_2^b)$. We assume that for any PPT algorithm $\mathcal{A}$, $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{DDH}_1}(\lambda) = \left|\Pr\left[\mathcal{A}(D, g_1^{ab})\right] - \Pr\left[\mathcal{A}(D, g_1^{ab+c})\right]\right|$ is negligible in the security parameter λ.*

The dual of above assumption is Decisional Diffie-Hellman assumption in $\mathbb{G}_2$ (denoted as DDH_2), which is identical to DDH_1 with the roles of $\mathbb{G}_1$ and $\mathbb{G}_2$ reversed.

Definition 10. ***Symmetric External Diffie-Hellman (*****SXDH*****).*** *[12] The SXDH assumption holds if DDH problems are intractable in both $\mathbb{G}_1$ and $\mathbb{G}_2$.*

Definition 11. ***q-strong Bilinear Diffie-Hellman (*****q-SBDH*****).*** *[8] Let $\Gamma = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e)$ be a bilinear group. In Γ, the q-SBDH problem is stated as follows: given as input a $(2q+2)$-tuple of elements $(g_1, g_1^{\alpha}, g_1^{(\alpha^2)}, \cdots, g_1^{(\alpha^q)}, g_2, g_2^{\alpha}, g_2^{(\alpha^2)}, \cdots, g_2^{(\alpha^q)}) \in \mathbb{G}_1^{q+1} \times \mathbb{G}_2^{q+1}$, output a pair $(\gamma, e(g_1, g_2)^{1/(\alpha+\gamma)}) \in \mathbb{Z}_p \times \mathbb{G}_T$ for a freely chosen value $\gamma \in \mathbb{Z}_p \setminus \{-\alpha\}$. The q-SBDH assumption states that for any PPT adversary $\mathcal{A}$, there exists negligible function $\epsilon(.)$ such that*

$$\Pr\left[\mathcal{A}(\Gamma, g_1^{\alpha}, g_1^{(\alpha^2)}, \cdots, g_1^{(\alpha^q)}, g_2^{\alpha}, g_2^{(\alpha^2)}, \cdots, g_2^{(\alpha^q)}) = (\gamma, e(g_1, g_2)^{1/(\alpha+\gamma)})\right] \leq \epsilon$$

where the probability is over the random choice of generator $g_1 \in \mathbb{G}_1$ and $g_2 \in \mathbb{G}_2$, the random choice of $\alpha \in \mathbb{Z}_p^*$, and the random bits consumed by $\mathcal{A}$.

Note 5. The above definition is a slightly modified version of the original assumption of [8]. Following the work of [35], our version can be reduced to the original one.

Definition 12. ***Decisional subspace assumption in*** $\mathbb{G}_1$ ***(DS1).*** *[12] Given an asymmetric bilinear group generator $\mathcal{G}(.)$, define the following distribution*

$$\begin{aligned}
&\Gamma = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e) \leftarrow \mathcal{G}(1^\kappa), (\mathbb{B}, \mathbb{B}^*) \leftarrow \mathsf{Dual}(\mathbb{Z}_p^n), \tau_1, \tau_2, \mu_1, \mu_2 \leftarrow \mathbb{Z}_p,\\
&\boldsymbol{u}_1 = g_2^{\mu_1 . \boldsymbol{b}_1^* + \mu_2 . \boldsymbol{b}_{k+1}^*}, \cdots, \boldsymbol{u}_k = g_2^{\mu_1 . \boldsymbol{b}_k^* + \mu_2 \boldsymbol{b}_{2k}^*}, \boldsymbol{v}_1 = g_1^{\tau_1 . \boldsymbol{b}_1}, \cdots, \boldsymbol{v}_k = g_1^{\tau_1 . \boldsymbol{b}_k},\\
&\boldsymbol{w}_1 = g_1^{\tau_1 . \boldsymbol{b}_1 + \tau_2 \boldsymbol{b}_{k+1}}, \cdots, \boldsymbol{w}_k = g_1^{\tau_1 . \boldsymbol{b}_k + \mu_2 \boldsymbol{b}_{2k}}
\end{aligned}$$

and set $\Delta = (\Gamma, g_2^{\boldsymbol{b}_1^*}, \cdots, g_2^{\boldsymbol{b}_k^*}, g_2^{\boldsymbol{b}_{2k+1}^*}, \cdots, g_2^{\boldsymbol{b}_n^*}, g_1^{\boldsymbol{b}_1}, \cdots, g_1^{\boldsymbol{b}_n}, \boldsymbol{u}_1, \cdots, \boldsymbol{u}_k, \mu_2)$, where k, n are fixed positive integers that satisfy $2k \leq n$. We assume that for any PPT algorithm $\mathcal{A}$, the following is negligible in 1^κ.

$$\mathsf{Adv}_{\mathcal{A}}^{DS1}(\kappa) = \left|\Pr\left[\mathcal{A}(\Delta, \boldsymbol{v}_1, \cdots, \boldsymbol{v}_k) = 1\right] - \Pr\left[\mathcal{A}(\Delta, \boldsymbol{w}_1, \cdots, \boldsymbol{w}_k) = 1\right]\right|$$

Lemma 1. *If the decisional Diffie Hellman assumption (DDH) in $\mathbb{G}_1$ holds, then the decisional subspace assumption in $\mathbb{G}_1$ (DS1) also holds.*

For the proof, refer to [12]. The **decisional subspace assumption** in $\mathbb{G}_2$ is defined as identical to DS1 with the roles of $\mathbb{G}_1$ and $\mathbb{G}_2$ reversed. DS2 holds if DDH in $\mathbb{G}_2$ holds. The proof is done as for $\mathbb{G}_1$. Thus, DS1 and DS2 hold if SXDH hold.

3 A New Accumulator Scheme

We here present a new cryptographic accumulator scheme based on a unique combination of Nguyen's accumulator [30] and dual pairing vector spaces. We also briefly compare our scheme to the literature, concluding that this is the first cryptographic accumulator permitting a private evaluation and a public witness generation.

Intuition. In a bilinear environment, Nguyen's accumulator for a set $\mathcal{X}$ is the element $\mathsf{acc}_{\mathcal{X}} = g_1^{\prod_{x \in \mathcal{X}}(x+s)}$ where s is the secret key. A witness for an element $\underline{x} \in \mathcal{X}$ is then the object $\mathsf{wit}_{\underline{x}} = g_2^{\prod_{x \in \mathcal{X} \setminus \{\underline{x}\}}(x+s)}$. Verification is done by checking that $e(\mathsf{acc}_{\mathcal{X}}, g_2) = e(g_1^{\underline{x}} \cdot g_1^s, \mathsf{wit}_{\underline{x}})$. If only g_1, g_2 and g_1^s are published, evaluation *and* witness generation are *private*. If the public key contains g_1, g_1^s, $\cdots$, $g_1^{s^q}$, g_2, g_2^s, $\cdots$, $g_2^{s^q}$, then both evaluation *and* witness generation are *public*, using characteristic polynomials (see Definition 7).

One basic idea to obtain a secret evaluation and a public witness generation is to keep secret the elements in $\mathbb{G}_1$ for the evaluation and to publicly use the elements in $\mathbb{G}_2$ for the witness creation. But this does not work as we need to have g_1^s for verification. Our idea is hence to go in a Dual Pairing Vector Space (DPVS) setting, as introduced above, in dimension $n = 2$. By playing with the bases $\boldsymbol{d}_1, \boldsymbol{d}_1^*, \boldsymbol{d}_2$ and $\boldsymbol{d}_2^*$, we can keep secret some elements and publish some others as follows:

- $g_1^{\boldsymbol{d}_1}, g_1^{\boldsymbol{d}_1 s}, \cdots g_1^{\boldsymbol{d}_1 s^q}$ are not publicly given since used for private evaluation;
- $g_2^{\boldsymbol{d}_2^*}, g_2^{\boldsymbol{d}_2^* s}, \cdots g_2^{\boldsymbol{d}_2^* s^q}$ are publicly used for witness creation; and
- $g_2^{\boldsymbol{d}_1^*}, g_1^{\boldsymbol{d}_2}, g_1^{\boldsymbol{d}_2 s}$ are publicly used for verification.

Thanks to that and the transformation from $\prod_{x \in \mathcal{X}}(x+s)$ to $\sum_{i=0}^{q} a_i s^i$, using the characteristic polynomial given in Definition 7, the above public elements are easily computable from the knowledge of the successive powers of s in groups $\mathbb{G}_1$ or $\mathbb{G}_2$, as it is done in Nguyen's. We obtain our scheme below. To be exhaustive, the resulting comparison between our scheme and Nguyen's is given in Table 1.

Table 1. Comparison between Nguyen's accumulator and ours.

Operation	Nguyen [30]	Ours
Evaluation	$\mathsf{acc}_{\mathcal{X}} = g_1^{\prod_{x \in \mathcal{X}}(x+s)}$	$\mathsf{acc}_{\mathcal{X}} = g_1^{\boldsymbol{d}_1 \prod_{x \in \mathcal{X}}(x+s)}$
Witness	$\mathsf{wit}_{\underline{x}} = g_2^{\prod_{x \in \mathcal{X} \setminus \{\underline{x}\}}(x+s)}$	$\mathsf{wit}_{\underline{x}} = g_2^{\boldsymbol{d}_2^* \prod_{x \in \mathcal{X} \setminus \{\underline{x}\}}(x+s)}$.
Verification	$e(\mathsf{acc}_{\mathcal{X}}, g_2) = e(g_1^{\underline{x}} \cdot g_1^s, \mathsf{wit}_{\underline{x}})$	$e(\mathsf{acc}_{\mathcal{X}}, g_2^{\boldsymbol{d}_1^*}) = e(g_1^{\boldsymbol{d}_2 \underline{x}} \cdot g_1^{\boldsymbol{d}_2 s}, \mathsf{wit}_{\underline{x}})$

Regarding efficiency, notice that our scheme is slightly less efficient than Nguyen's scheme [30]. Indeed in the latter accumulators and witnesses are respectively composed of one element of $\mathbb{G}_1$ and $\mathbb{G}_2$ while in our scheme they are respectively composed of *two* elements of $\mathbb{G}_1$ and $\mathbb{G}_2$. Regarding the number of pairing in verification, Nguyen's requires one pairing while our scheme requires *two* pairing.

Construction. Following the above intuition, our full scheme is presented in Figure 1. In a nutshell, our construction is a static, bounded, and supports multisets and subsets queries.

- $\mathsf{Gen}(1^\kappa, q)$: run a bilinear group generation algorithm to get $\Gamma = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2)$. Then choose a random $s \leftarrow \mathbb{Z}_p^*$, and run $\mathsf{Dual}(\mathbb{Z}_p^2)$ to get $\mathbb{D} = (\boldsymbol{d}_1, \boldsymbol{d}_2), \mathbb{D}^* = (\boldsymbol{d}_1^*, \boldsymbol{d}_2^*)$. Let $\psi \in \mathbb{Z}_p$ be the random such that $\boldsymbol{d}_1 \cdot \boldsymbol{d}_1^* = \boldsymbol{d}_2 \cdot \boldsymbol{d}_2^* = \psi$. Set $\mathsf{sk_{acc}} = (s, \mathbb{D}, \mathbb{D}^*)$, $\mathsf{pk_{acc}} = \left(\Gamma, g_1^{\boldsymbol{d}_2}, g_1^{\boldsymbol{d}_2 s}, \cdots, g_1^{\boldsymbol{d}_2 s^q}, g_2^{\boldsymbol{d}_1^*}, g_2^{\boldsymbol{d}_2^*}, g_2^{\boldsymbol{d}_2^* s}, \cdots, g_2^{\boldsymbol{d}_2^* s^q}\right)$, and return $\mathsf{sk_{acc}}, \mathsf{pk_{acc}}$.
- $\mathsf{Eval}(\mathsf{sk_{acc}}, \mathsf{pk_{acc}}, \mathcal{X})$: compute the coefficients $\{a_i\}_{i=0,\cdots,q}$ of the polynomial $\mathsf{Ch}_{\mathcal{X}}[Z] = \prod_{x \in \mathcal{X}}(Z + x)$. Then compute $\mathsf{acc}_{\mathcal{X}} = g_1^{\boldsymbol{d}_1 \sum_{i=0}^{q} a_i s^i}$, and return $\mathsf{acc}_{\mathcal{X}}$.
- $\mathsf{WitCreate}(\mathsf{pk_{acc}}, \mathcal{X}, \mathsf{acc}_{\mathcal{X}}, \mathcal{I})$: let $\{b_i\}_{i=0,\cdots,q}$ be the coefficients of the polynomial $\mathsf{Ch}_{\mathcal{X} \setminus \mathcal{I}}[Z] = \prod_{x \in \mathcal{X} \setminus \mathcal{I}}(x + Z)$. Compute $\mathsf{wit}_{\mathcal{I}} = g_2^{\boldsymbol{d}_2^* \sum_{i=0}^{q} b_i s^i}$, and return $\mathsf{wit}_{\mathcal{I}}$.
- $\mathsf{Verify}(\mathsf{pk_{acc}}, \mathsf{acc}_{\mathcal{X}}, \mathsf{wit}_{\mathcal{I}}, \mathcal{I})$: let $\{c_i\}_{i=0,\cdots,q}$ be the coefficients of the polynomial $\mathsf{Ch}_{\mathcal{I}}[Z] = \prod_{x \in \mathcal{I}}(x + Z)$ and return 1 if $e(\mathsf{acc}_{\mathcal{X}}, g_2^{\boldsymbol{d}_1^*}) = e(g_1^{\boldsymbol{d}_2 \sum_{i=0}^{q} c_i s^i}, \mathsf{wit}_{\mathcal{I}})$, 0 otherwise.

Fig. 1. Our first accumulator scheme, with private evaluation and public witness generation.

Security. In short, the correctness comes from both (i) the one of Nguyen scheme (indeed, the same pairing equation is used), and (ii) the properties of DPVS ($\boldsymbol{b}_i \cdot \boldsymbol{b}_j^* = 0 \pmod p$ whenever $i \neq j$, and $\boldsymbol{b}_i \cdot \boldsymbol{b}_i^* = \psi \pmod p$ for all i). More formally, we prove the following theorem.

Theorem 1. *Our accumulator scheme is correct.*

Proof. Let $\mathcal{X}, \mathcal{I}$ be two sets such that $\mathcal{I} \subset \mathcal{X}$. Let $\{a_i, b_i, c_i\}_{i=0}^{q}$ be respectively the coefficients of polynomials $\mathsf{Ch}_{\mathcal{X}}[Z] = \prod_{x \in \mathcal{X}}(x + Z)$, $\mathsf{Ch}_{\mathcal{X} \setminus \mathcal{I}}[Z] = \prod_{x \in \mathcal{X} \setminus \mathcal{I}}(x + Z)$ and $\mathsf{Ch}_{\mathcal{I}}[Z] = \prod_{x \in \mathcal{I}}[Z](x + Z)$. Let $\mathsf{acc}_{\mathcal{X}} \leftarrow \mathsf{Eval}(\mathsf{sk_{acc}}, \mathcal{X})$ and $\mathsf{mwit}_{\mathcal{I}} \leftarrow \mathsf{WitCreate}(\mathsf{pk_{acc}}, \mathcal{X}, \mathsf{acc}_{\mathcal{X}}, \mathcal{I})$. We have that

$$e(g_1^{\boldsymbol{d}_2 \sum_{i=0}^{q} c_i s^i}, \mathsf{mwit}_{\mathcal{I}}) = e(g_1^{\boldsymbol{d}_2 \sum_{i=0}^{q} c_i s^i}, g_2^{\boldsymbol{d}_2^* \sum_{i=0}^{q} b_i s^i}) = e(g_1, g_2)^{\psi \sum_{i=0}^{q} c_i s^i \cdot \sum_{i=0}^{q} b_i s^i}.$$

As $\mathcal{I} \subset \mathcal{X}$, then $\sum_{i=0}^{q} c_i s^i \cdot \sum_{i=0}^{q} b_i s^i = \sum_{i=0}^{q} a_i s^i$. Thus

$$e(g_1^{\boldsymbol{d}_2 \sum_{i=1}^{q} c_i s^i}, \mathsf{mwit}_{\mathcal{I}}) = e(g_1, g_2)^{\psi \sum_{i=0}^{q} a_i s^i} = e(\mathsf{acc}_{\mathcal{X}}, g_2^{\boldsymbol{d}_1^*}).$$

□

Regarding security, we prove the following theorem.

Theorem 2. *Our scheme satisfies collision resistance under* q-SBDH *assumption.*

- On input $1^\kappa, q \in \mathbb{N}$, $\mathcal{C}$ runs bilinear group generation to get $\Gamma = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2)$ and chooses $\alpha \leftarrow \mathbb{Z}_p^*$. It sends $\Gamma, g_1^\alpha, \cdots, g_1^{\alpha^q}, g_2^\alpha, \cdots, g_2^{\alpha^q}$ to $\mathcal{A}$.
- $\mathcal{A}$ runs $\mathsf{Dual}(\mathbb{Z}_p^2)$ to get $\mathbb{D} = (\boldsymbol{d}_1, \boldsymbol{d}_2)$ and $\mathbb{D}^* = (\boldsymbol{d}_1^*, \boldsymbol{d}_2^*)$ such that $\boldsymbol{d}_1 \cdot \boldsymbol{d}_1^* = \boldsymbol{d}_2 \cdot \boldsymbol{d}_2^* = \psi$, where $\psi \in \mathbb{Z}_p$. Then it sets $\mathsf{pk}_{\mathsf{acc}} = \left(\Gamma, g_1^{\boldsymbol{d}_2}, g_1^{\boldsymbol{d}_2\alpha}, \cdots, g_1^{\boldsymbol{d}_2\alpha^q}, g_2^{\boldsymbol{d}_1^*}, g_2^{\boldsymbol{d}_2^*}, g_2^{\boldsymbol{d}_2^*\alpha}, \cdots, g_2^{\boldsymbol{d}_2^*\alpha^q}\right)$ and sends it to $\mathcal{B}$.
- $\mathcal{B}$ makes an accumulator query: it chooses set $\mathcal{X}$ and sends it to $\mathcal{A}$. The latter uses its knowledge of $\boldsymbol{d}_1$ to return to $\mathcal{B}$ $\mathsf{acc}_\mathcal{X} = g_1^{\boldsymbol{d}_1 \mathsf{Ch}_\mathcal{X}(\alpha)}$. This step can be repeat an unbounded number of times.
- At some point, $\mathcal{B}$ answers with $\underline{\mathcal{X}}, \underline{x}, \mathsf{wit}_{\underline{x}}$ where $\underline{x} \notin \underline{\mathcal{X}}$ and $\mathsf{wit}_{\underline{x}}$ is a membership witness of $\underline{x}$ for set $\underline{\mathcal{X}}$.
- $\mathcal{A}$ returns to $\mathcal{C}$ $(\underline{x}, e(g_1, (\mathsf{wit}_{\underline{x}}^{\boldsymbol{d}_2})^{1/\psi r} \cdot (g_2^{-Q(\alpha)})^{1/r}))$ as its answer to break the assumption.

Fig. 2. Construction of q-SBDH adversary from collision resistance adversary.

Proof. We prove the contrapositive. Let $\mathcal{C}$ be the q-SBDH challenger, $\mathcal{B}$ an adversary against collision resistance of the accumulator, that wins with non-negligible advantage. We build, in Figure 2, $\mathcal{A}$ an adversary against the q-SBDH assumption, using $\mathcal{B}$.

Let us see that the solution output by $\mathcal{A}$ is correct. As $\underline{x} \notin \underline{\mathcal{X}}$, there exist polynomial $Q[Z]$ and integer r such that $\mathsf{Ch}_{\underline{\mathcal{X}}}[Z] = Q[Z](\underline{x} + Z) + r$. As $\mathsf{wit}_{\underline{x}}$ is a membership witness, we have that $e(g_1^{\boldsymbol{d}_2(\underline{x}+\alpha)}, \mathsf{wit}_{\underline{x}}) = e(\mathsf{acc}_{\underline{\mathcal{X}}}, g_2^{\boldsymbol{d}_1^*})$.

Therefore, we have that $e(g_1^{\boldsymbol{d}_2(\underline{x}+\alpha)}, \mathsf{wit}_{\underline{x}}) = e(g_1, g_2)^{\psi(\alpha+\underline{x})Q(\alpha)+\psi r}$ and

$$\begin{aligned} &(e(g_1, (\mathsf{wit}_{\underline{x}}^{\boldsymbol{d}_2})^{1/\psi r} \cdot (g_2^{-Q(\alpha)})^{1/r}))^{(\alpha+\underline{x})} \\ &= e(g_1, g_2)^{\frac{(\alpha+\underline{x})Q(\alpha)}{r}+1} \cdot (g_1, g_2)^{\frac{-(\alpha+\underline{x})Q(\alpha)}{r}} \\ &= e(g_1, g_2) \end{aligned}$$

Notice that $\mathcal{A}$ knows $\boldsymbol{d}_2, \psi$ and r and can compute $g_2^{-Q(\alpha)}$ from the challenge tuple. Thus, $\underline{x}, e(g_1, (\mathsf{wit}_{\underline{x}}^{\boldsymbol{d}_2})^{1/\psi r} \cdot (g_2^{-Q(\alpha)})^{1/r})$ is a solution to the q-SBDH problem.

As $\mathcal{A}$ breaks the assumption when $\mathcal{B}$ breaks the collision resistance of the accumulator, we have that $\mathcal{A}$'s advantage is equal to $\mathcal{B}$'s advantage, meaning that $\mathcal{A}$ breaks the q-SBDH assumption with non-negligible advantage.

$\square$

Comparison. Our accumulator is the first to propose a private evaluation while having a public witness generation. Indeed, we compare in Table 2 for the four families of accumulators instantiations how evaluation and witness creation are done. The only exception could be a construction given by Zhang *et al.* in [40]. More precisely, the studied primitive is called an *Expressive Set Accumulator* and is presented with private evaluation and some kind of public witness creation: their scheme does not have a $\mathsf{WitCreate}$ algorithm but a Query that takes as input some sets along with a set operation query, and returns the result of the query along with a proof of correctness. However, as stated in their work, in their construction the evaluation can actually be done only with the public key.

Table 2. Comparison of evaluation and witness creation according to the type of accumulator instantiation.

Type	Evaluation	Witness Generation
Hash based	Public	Public
	Public	Public
Lattices	Public	Private
Number Theoretic	Public	Public [1]
Pairing based	Public	Public
	Private	Private
Ours	Private	Public

Having both private evaluation and public witness creation helps us build an encryption scheme where the accumulator is used as a secret key computed by an authority, from which user can derive some information (the witness) for decryption. Moreover, accumulators can satisfy a lot of additional properties: universal, dynamic, asynchronous, . . . and verify a lot of security properties: undeniability, indistinguishability, . . . (see e.g., [14]). The above construction focuses on static accumulators that satisfy collision resistance, and in this work, we do not consider those additional features and security properties. We leave as an open problem the modifications to satisfy other properties of accumulators. The only exception is when accumulators are used in the context of authenticated set operations [33,28,19]. See Appendix E for more details on sets operations. Regarding pairing-based accumulators, we refer the interested reader to several works such as [5,13,10,1,14,19] among others.

In the next section, we present our main new functionality, namely dually computable accumulator, and show how to transform the above construction into a new one that satisfies it.

4 Dually Computable Cryptographic Accumulators

In this section, we introduce a new kind of cryptographic accumulator that we call *dually computable* accumulator. In such case, there are two separate evaluation algorithms that give two different outputs: the first one (Eval) uses the accumulator secret key while the second one (PublicEval) uses solely the public key. Using the unique unmodified witness generation algorithm, we also define two different verification algorithms, one for each type of accumulator. Following the work done in the previous

[1] Secret key can be given for witness generation in order to improve efficiency. Creation is still possible without it.

section, we focus on accumulator schemes that have private evaluation and public witness generation. We start by formally defining dually computable accumulators, then we present an instantiation.

4.1 Definitions

Definition 13. ***Dually computable accumulator.*** *Starting from a static accumulator* $\mathsf{Acc} = (\mathsf{Gen}, \mathsf{Eval}, \mathsf{WitCreate}, \mathsf{Verify})$*, we say that* Acc *is* dually computable *if it also provides two algorithms* $\mathsf{PublicEval}$*,* $\mathsf{PublicVerify}$ *such that:*

- $\mathsf{PublicEval}(\mathsf{pk}_{\mathsf{acc}}, \mathcal{X})$*: it takes as input the accumulator public key* $\mathsf{pk}_{\mathsf{acc}}$ *and a set* $\mathcal{X}$*. It outputs an accumulator* $\mathsf{accp}_{\mathcal{X}}$ *of* $\mathcal{X}$ *and auxiliary information* auxp.
- $\mathsf{PublicVerify}(\mathsf{pk}_{\mathsf{acc}}, \mathsf{accp}_{\mathcal{X}}, \mathsf{witp}_x, x)$*: it takes as input the accumulator public key* $\mathsf{pk}_{\mathsf{acc}}$*, a publicly computed accumulator* $\mathsf{accp}_{\mathcal{X}}$ *of* $\mathcal{X}$*, an element* x*, a witness* witp_x *for* x*, computed from* $\mathsf{WitCreate}(\mathsf{pk}_{\mathsf{acc}}, \mathcal{X}, \mathsf{accp}_{\mathcal{X}}, \mathsf{auxp}, x)$*. It outputs* 1 *if* witp_x *is a membership witness and* $x \in \mathcal{X}$*,* 0 *otherwise.*

A dually computable accumulator must satisfy four properties: *correctness*, *collision resistance*, *distinguishability* and *correctness of duality*.

Definition 14. ***Correctness of dually computable accumulator.*** *A dually computable accumulator is said to be* correct *if for all security parameters* κ*, all integer* $\mathfrak{b} \in \mathbb{N} \cup \{\infty\}$*, all set of values* $\mathcal{X}$ *and all element* x *such that* $x \in \mathcal{X}$

$$\Pr\left[\begin{array}{c} \mathsf{sk}_{\mathsf{acc}}, \mathsf{pk}_{\mathsf{acc}} \leftarrow \mathsf{Gen}(1^\kappa, \mathfrak{b}), \\ \mathsf{acc}_{\mathcal{X}}, \mathsf{aux} \leftarrow \mathsf{Eval}(\mathsf{sk}_{\mathsf{acc}}, \mathsf{pk}_{\mathsf{acc}}, \mathcal{X}), \\ \mathsf{accp}_{\mathcal{X}}, \mathsf{auxp} \leftarrow \mathsf{PublicEval}(\mathsf{pk}_{\mathsf{acc}}, \mathcal{X}), \\ \mathsf{wit}_x \leftarrow \mathsf{WitCreate}(\mathsf{pk}_{\mathsf{acc}}, \mathcal{X}, \mathsf{acc}_{\mathcal{X}}, \mathsf{aux}, x) \\ \mathsf{witp}_x \leftarrow \mathsf{WitCreate}(\mathsf{pk}_{\mathsf{acc}}, \mathcal{X}, \mathsf{accp}_{\mathcal{X}}, \mathsf{auxp}, x) \colon \\ \mathsf{Verify}(\mathsf{pk}_{\mathsf{acc}}, \mathsf{acc}_{\mathcal{X}}, \mathsf{wit}_x, x) = 1 \\ \wedge \mathsf{PublicVerify}(\mathsf{pk}_{\mathsf{acc}}, \mathsf{accp}_{\mathcal{X}}, \mathsf{witp}_x, x) = 1 \end{array}\right] = 1$$

Definition 15. ***Collision resistance.*** *A dually computable accumulator is* collision resistant*, if for all PPT adversaries* $\mathcal{A}$ *there is a negligible function* $\epsilon(.)$ *such that:*

$$\Pr\left[\begin{array}{c} (\mathsf{sk}_{\mathsf{acc}}, \mathsf{pk}_{\mathsf{acc}}) \leftarrow \mathsf{Gen}(1^\kappa, \mathfrak{b}), (\mathsf{wit}_x, x) \leftarrow \mathcal{A}^{\mathcal{O}^E}(\mathsf{pk}_{\mathsf{acc}}) \colon \\ (\mathsf{Verify}(\mathsf{pk}_{\mathsf{acc}}, \mathsf{acc}_{\mathcal{X}}, \mathsf{wit}_x, x) = 1 \wedge x \notin \mathcal{X}) \\ \vee(\mathsf{PublicVerify}(\mathsf{pk}_{\mathsf{acc}}, \mathsf{accp}_{\mathcal{X}}, \mathsf{wit}_x, x) = 1 \wedge x \notin \mathcal{X}) \end{array}\right] \leq \epsilon(\kappa),$$

where $\mathsf{acc}_{\mathcal{X}} \leftarrow \mathsf{Eval}(\mathsf{sk}_{\mathsf{acc}}, \mathsf{pk}_{\mathsf{acc}}, \mathcal{X})$*,* $\mathsf{accp}_{\mathcal{X}} \leftarrow \mathsf{PublicEval}(\mathsf{pk}_{\mathsf{acc}}, \mathcal{X})$ *and* $\mathcal{A}$ *has oracle access to* $\mathcal{O}^E$ *that represents the oracle for the algorithm* Eval*. An adversary is allowed to query it an arbitrary number of times and can run* $\mathsf{PublicEval}, \mathsf{WitCreate}$ *as the two algorithms only use the accumulator public key, that is known by the adversary.*

Definition 16. ***Distinguishability.*** *A dually computable accumulator satisfies* distinguishability*, if for any security parameter* κ *and integer* $\mathfrak{b} \in \mathbb{N} \cup \{\infty\}$*, any keys* $(\mathsf{sk}_{\mathsf{acc}}, \mathsf{pk}_{\mathsf{acc}})$ *generated by* $\mathsf{Gen}(1^\kappa, \mathfrak{b})$*, and any set* $\mathcal{X}$*,* $\mathsf{acc}_{\mathcal{X}} \leftarrow \mathsf{Eval}(\mathsf{sk}_{\mathsf{acc}}, \mathsf{pk}_{\mathsf{acc}}, \mathcal{X})$ *and* $\mathsf{accp}_{\mathcal{X}} \leftarrow \mathsf{PublicEval}(\mathsf{pk}_{\mathsf{acc}}, \mathcal{X})$ *are distinguishable.*

The last property states that a witness computed for a privately (resp. publicly) computed accumulator as input of the $\mathsf{WitCreate}$ algorithm must pass the $\mathsf{PublicVerify}$ (resp. Verify) algorithm, with publicly (resp. privately) computed accumulator for the same set as the privately (resp. publicly) computed accumulator.

Definition 17. ***Correctness of duality.*** *A dually computable accumulator is said to satisfy* correctness of duality *if for all security parameters κ, all integer $\mathfrak{b} \in \mathbb{N} \cup \{\infty\}$, all set of values $\mathcal{X}$ and all value x such that $x \in \mathcal{X}$*

$$\Pr\left[\begin{array}{c} \mathsf{sk}_{\mathsf{acc}}, \mathsf{pk}_{\mathsf{acc}} \leftarrow \mathsf{Gen}(1^\kappa, \mathfrak{b}), \\ \mathsf{acc}_{\mathcal{X}}, \mathsf{aux} \leftarrow \mathsf{Eval}(\mathsf{sk}_{\mathsf{acc}}, \mathsf{pk}_{\mathsf{acc}}, \mathcal{X}), \\ \mathsf{accp}_{\mathcal{X}}, \mathsf{auxp} \leftarrow \mathsf{PublicEval}(\mathsf{pk}_{\mathsf{acc}}, \mathcal{X}), \\ \mathsf{wit}_x \leftarrow \mathsf{WitCreate}(\mathsf{pk}_{\mathsf{acc}}, \mathcal{X}, \mathsf{acc}_{\mathcal{X}}, \mathsf{aux}, x) \\ \mathsf{witp}_x \leftarrow \mathsf{WitCreate}(\mathsf{pk}_{\mathsf{acc}}, \mathcal{X}, \mathsf{accp}_{\mathcal{X}}, \mathsf{auxp}, x) \textbf{:} \\ (\mathsf{PublicVerify}(\mathsf{pk}_{\mathsf{acc}}, \mathsf{accp}_{\mathcal{X}}, \mathsf{wit}_x, x) = 1) \\ \wedge(\mathsf{Verify}(\mathsf{pk}_{\mathsf{acc}}, \mathsf{acc}_{\mathcal{X}}, \mathsf{witp}_x, x) = 1) \end{array}\right] = 1$$

4.2 Our First Dually Computable Cryptographic Accumulator

We now present our instantiation of a dually computable cryptographic accumulator. We also present some variants in the next section (for our construction of an ABE), and in Appendix B. We consider that the version we propose in this section is the simplest and more efficient one, but the others, as we will see, can be used for different other contexts.

Intuition. Using our previous accumulator instantiation (see Section 3), we can obtain a dually computable accumulator scheme by adding $g_2^{\boldsymbol{d}_1^*}, g_2^{\boldsymbol{d}_1^* s}, \cdots, g_2^{\boldsymbol{d}_1^* s^q}$ to the public key. Then, the public evaluation corresponds to the generation of $\mathsf{accp}_{\mathcal{X}} = g_2^{\boldsymbol{d}_1^* \prod_{x \in \mathcal{X}}(x+s)}$. With the description of Eval as in the previous scheme, we directly obtain what we need. Moreover, the two accumulators are easily distinguishable as the secretly computed one is two elements in $\mathbb{G}_1$ while the publicly generated one is two elements in $\mathbb{G}_2$.
From those two accumulators, and the witness as generated in our first accumulator scheme (i.e., $\mathsf{wit}_{\underline{x}} = g_2^{\boldsymbol{d}_2^* \prod_{x \in \mathcal{X} \setminus \{\underline{x}\}}(x+s)}$), we are able to provide two very close verification equations. In fact, we remark that we obtain a sort of symmetry between the two accumulators, as $e(\mathsf{acc}_{\mathcal{X}}, g_2^{\boldsymbol{d}_1^*}) = e(g_1^{\boldsymbol{d}_1}, \mathsf{accp}_{\mathcal{X}})$, which two are equals to $e(g_1^{\boldsymbol{d}_2 \underline{x}} \cdot g_1^{\boldsymbol{d}_2 s}, \mathsf{wit}_{\underline{x}})$, which is computable from the knowledge of the witness[2].

Construction. In Figure 3, we present the full description of our first dually computable scheme, from the above intuition, and using again the characteristic polynomial (see Definition 7).

Security. We can now focus on the security of our construction, by providing the following full theorem.

[2] We could have also chosen to define $\mathsf{PublicEval}$ such that it returns $g_2^{\boldsymbol{d}_2^* a_i s^i}$, and $\mathsf{PublicVerify}$ such that the left part of the equation is $e(g_1^{\boldsymbol{d}_2}, \mathsf{accp})$.

- $\mathsf{Gen}(1^\kappa, q)$: run a bilinear group generation algorithm to get $\Gamma = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2)$. Then choose a random $s \leftarrow \mathbb{Z}_p^*$, and run $\mathsf{Dual}(\mathbb{Z}_p^2)$ to get $\mathbb{D} = (\boldsymbol{d}_1, \boldsymbol{d}_2), \mathbb{D}^* = (\boldsymbol{d}_1^*, \boldsymbol{d}_2^*)$. Let $\psi \in \mathbb{Z}_p$ be the random such that $\boldsymbol{d}_1 \cdot \boldsymbol{d}_1^* = \boldsymbol{d}_2 \cdot \boldsymbol{d}_2^* = \psi$. Set $\mathsf{sk}_{\mathsf{acc}} = (s, \mathbb{D}, \mathbb{D}^*)$,

$$\mathsf{pk}_{\mathsf{acc}} = \begin{pmatrix} \Gamma, g_1^{\boldsymbol{d}_1}, g_1^{\boldsymbol{d}_2}, g_1^{\boldsymbol{d}_2 s}, \cdots, g_1^{\boldsymbol{d}_2 s^q}, g_2^{\boldsymbol{d}_1^*}, \\ g_2^{\boldsymbol{d}_1^* s}, \cdots, g_2^{\boldsymbol{d}_1^* s^q}, g_2^{\boldsymbol{d}_2^*}, g_2^{\boldsymbol{d}_2^* s}, \cdots, g_2^{\boldsymbol{d}_2^* s^q} \end{pmatrix}.$$

  Return $\mathsf{sk}_{\mathsf{acc}}, \mathsf{pk}_{\mathsf{acc}}$.
- $\mathsf{Eval}(\mathsf{sk}_{\mathsf{acc}}, \mathsf{pk}_{\mathsf{acc}}, \mathcal{X})$: compute the coefficients $\{a_i\}_{i=0,\cdots,q}$ of the polynomial $\mathsf{Ch}_{\mathcal{X}}[Z] = \prod_{x \in \mathcal{X}}(Z + x)$. Then compute $\mathsf{acc}_{\mathcal{X}} = g_1^{\boldsymbol{d}_1 \sum_{i=0}^q a_i s^i}$, and return $\mathsf{acc}_{\mathcal{X}}$.
- $\mathsf{PublicEval}(\mathsf{pk}_{\mathsf{acc}}, \mathcal{X})$: compute the coefficients $\{a_i\}_{i=0,\cdots,q}$ of the polynomial $\mathsf{Ch}_{\mathcal{X}}[Z] = \prod_{x \in \mathcal{X}}(Z + x)$. Then compute $\mathsf{accp}_{\mathcal{X}} = g_2^{\boldsymbol{d}_1^* \sum_{i=0}^q a_i s^i}$, and return $\mathsf{accp}_{\mathcal{X}}$.
- $\mathsf{WitCreate}(\mathsf{pk}_{\mathsf{acc}}, \mathcal{X}, \mathsf{acc}_{\mathcal{X}}/\mathsf{accp}_{\mathcal{X}}, \mathcal{I})$: let $\{b_i\}_{i=0,\cdots,q}$ be the coefficients of the polynomial $\mathsf{Ch}_{\mathcal{X} \backslash \mathcal{I}}[Z] = \prod_{x \in \mathcal{X} \backslash \mathcal{I}}(x + Z)$. Compute $\mathsf{wit}_{\mathcal{I}} = \mathsf{witp}_{\mathcal{I}} = g_2^{\boldsymbol{d}_2^* \sum_{i=0}^q b_i s^i}$, and return $\mathsf{wit}_{\mathcal{I}}$.
- $\mathsf{Verify}(\mathsf{pk}_{\mathsf{acc}}, \mathsf{acc}_{\mathcal{X}}, \mathsf{wit}_{\mathcal{I}}, \mathcal{I})$: let $\{c_i\}_{i=0,\cdots,q}$ be the coefficients of the polynomial $\mathsf{Ch}_{\mathcal{I}}[Z] = \prod_{x \in \mathcal{I}}(x + Z)$ and return 1 if $e(\mathsf{acc}_{\mathcal{X}}, g_2^{\boldsymbol{d}_1^*}) = e(g_1^{\boldsymbol{d}_2 \sum_{i=0}^q c_i s^i}, \mathsf{wit}_{\mathcal{I}})$, 0 otherwise.
- $\mathsf{PublicVerify}(\mathsf{pk}_{\mathsf{acc}}, \mathsf{accp}_{\mathcal{X}}, \mathsf{wit}_{\mathcal{I}}, \mathcal{I})$: let $\{c_i\}_{i=0,\cdots,q}$ be the coefficients of the polynomial $\mathsf{Ch}_{\mathcal{I}}[Z] = \prod_{x \in \mathcal{I}}(x + Z)$ and return 1 if $e(g_1^{\boldsymbol{d}_1}, \mathsf{accp}_{\mathcal{X}}) = e(g_1^{\boldsymbol{d}_2 \sum_{i=0}^q c_i s^i}, \mathsf{wit}_{\mathcal{I}})$, 0 otherwise.

Fig. 3. Our first dually computable accumulator scheme.

Theorem 3. *Our scheme is correct, collision resistant under q-*SBDH *assumption, and satisfies both* distinguishability *and correctness of duality.*

Proof. Correctness and collision resistance (for privately and publicly computed accumulators) can be done as for our cryptographic accumulator in Section 3. Indeed, the algorithms $\mathsf{Eval}, \mathsf{WitCreate}$ and Verify are not changed compare to what we provided in Figure 1. For the publicly computed part, the proof still holds. The only modification is that $e(\mathsf{acc}_{\underline{\mathcal{X}}}, g_2^{\boldsymbol{d}_1^*})$ and $e(\mathsf{acc}_{\mathcal{X}}, g_2^{\boldsymbol{d}_1^*})$ are replaced by $e(g_1^{\boldsymbol{d}_1}, \mathsf{acc}_{\underline{\mathcal{X}}})$ and $e(g_1^{\boldsymbol{d}_1}, \mathsf{acc}_{\mathcal{X}})$ respectively.

Additionally, our accumulator satisfies *distinguishability* as a privately computed accumulator is composed of an element in $\mathbb{G}_1$ while a publicly computed accumulator is an element in $\mathbb{G}_2$. In fact, in a bilinear environment, we know that there are efficient algorithms for computing group operations, evaluating the bilinear map, deciding membership of the groups, deciding equality of group elements and sampling generators of the groups (see e.g., [22]).

Correctness of duality is satisfied as we have one unique witness and, as explained above, we have a symmetry between the two accumulators:

$$\underbrace{e(\mathsf{acc}_{\mathcal{X}}, g_2^{\boldsymbol{d}_1^*})}_{\text{from Eval}} = \underbrace{e(g_1^{\boldsymbol{d}_2 \sum_{i=0}^{q} c_i s^i}, \mathsf{wit}_{\mathcal{I}})}_{\text{from WitCreate}} = \underbrace{e(g_1^{\boldsymbol{d}_1}, \mathsf{accp}_{\mathcal{X}})}_{\text{from PublicEval}}.$$

Thus, the proof for $\mathsf{acc}_{\mathcal{X}}$ is exactly the same than in Theorem 1. For $\mathsf{accp}_{\mathcal{X}}$ the proof can proceed as in Theorem 1 by replacing acc_X and Verify by $\mathsf{accp}_{\mathcal{X}}$ and PublicVerify. □

5 Application of Dually Computable Accumulator: Attribute Based Encryption

In this section, our purpose is to show how we can transform our new notion of dually computable cryptographic accumulator to design Attribute Based Encryption (ABE). More precisely, first showing that due to security reasons, it cannot directly be used to obtain an ABE, and then show how to transform it into a dually computable accumulator that can be used to obtain the first Ciphertext Policy ABE (CP-ABE) for monotone NC^1 circuits with both constant size for ciphertexts and secret keys.
We start by formally presenting the notion of ABE, then we explain briefly the intuitions of our construction. Finally we present our scheme and compare it to existing ones.

5.1 Security Definitions for ABE

We start by formally introducing attribute based encryption scheme and the related security notions. In this work we will focus on *bounded* attribute based encryption schemes, meaning that during the setup phase a bound in the number of attributes allowed in the scheme is given and keys and ciphertexts can be created for an arbitrarily number of attributes at the condition that this number is lower than the bound.

Definition 18. ***Bounded (ciphertext policy) attribute based encryption.*** *[34,20] A ciphertext policy attribute based encryption scheme consists of four algorithms:*

- $\mathsf{Setup}(1^\kappa, q) \to (\mathsf{pk}, \mathsf{msk})$*: the setup algorithm takes as input a security parameter* κ *and an integer* q *which represent the bound of the number of attributes, and outputs a master public key* pk *and a master secret key* msk.
- $\mathsf{KeyGen}(\mathsf{pk}, \mathsf{msk}, \Upsilon) \to \mathsf{sk}_\Upsilon$*: the key generation algorithm takes as input the master public key* pk*, the master secret key* msk*, and a key attribute* Υ *and outputs a private key* sk_Υ.
- $\mathsf{Encrypt}(\mathsf{pk}, \Pi, \mathsf{m} \in \mathcal{M}) \to \mathsf{ct}$*: the encryption algorithm takes as input a master public key* pk*, an access policy* Π*, and a message* m *and outputs a ciphertext* ct_Π.
- $\mathsf{Decrypt}(\mathsf{pk}, \mathsf{sk}_\Upsilon, \Upsilon, \mathsf{ct}_\Pi, \Pi) \to \mathsf{m}$ *or* $\perp$*: the decryption algorithm takes as input the master public key* pk*, a private key* sk_Υ *along with the associated set of attributes* Υ*, a ciphertext* ct_Π *and its associated access policy* Π*. It outputs the message* m *if* Υ *satisfies* Π *or reject symbol* $\perp$ *otherwise.*

Informally, *Correctness* of ABE states that for every security parameter, every bound in the number of attributes, every honestly generated secret and public keys, every honestly generated key for any attributes set Υ, every honestly generated ciphertext for any policy Π, such that Υ satisfies Π, the decryption algorithm always returns 1.

Definition 19. ***Correctness of ABE.*** *A CP-ABE scheme is correct if for all security parameter $\kappa \in \mathbb{N}$, all integer q that represents the bound in the number of attributes, all attributes set Υ and all access policy Π such that Υ satisfies Π and for all messages* m*,*

$$\Pr\left[\begin{array}{c}(\mathsf{pk},\mathsf{msk}) \leftarrow \mathsf{Setup}(1^\kappa, q)\\ \mathsf{sk}_\Upsilon \leftarrow \mathsf{KeyGen}(\mathsf{pk},\mathsf{msk},\Upsilon)\\ \mathsf{ct}_\Pi \leftarrow \mathsf{Encrypt}(\mathsf{pk},\Pi,\mathsf{m})\\ \mathsf{Decrypt}(\mathsf{pk},\mathsf{sk}_\Upsilon,\Upsilon,\mathsf{ct}_\Pi,\Pi) = \mathsf{m}\end{array}\right] = 1$$

where the probability is taken over the coins of Setup, KeyGen*, and* $\mathsf{Encrypt}$.

Definition 20. ***Adaptive indistinguishability security.*** *(Ada-IND) A (CP-)ABE scheme is said to satisfy* adaptive indistinguishability *security if for all PPT adversary $\mathcal{A}$, there exists a negligible function $\epsilon(.)$ such that $\mathsf{Adv}_{\mathcal{A}}^{Ada-IND}(1^\kappa) \leq \epsilon(\kappa)$ where $\mathsf{Adv}_{\mathcal{A}}^{Ada-IND}$ (1^κ) is the advantage of $\mathcal{A}$ to win the security game presented in Figure 4, and is defined as $\mathsf{Adv}_{\mathcal{A}}^{Ada-IND}(1^\kappa) = \left|\Pr\left[b^{'} = b\right] - \frac{1}{2}\right|$. Let $\mathcal{C}$ be the challenger.*

1. **Setup phase**: on input $1^\kappa, q$, $\mathcal{C}$ samples $(\mathsf{pk},\mathsf{msk}) \leftarrow \mathsf{Setup}(1^\kappa, q)$ and gives pk to $\mathcal{A}$.
2. **Query phase**: during the game, $\mathcal{A}$ makes the following queries, in an arbitrary order. $\mathcal{A}$ can make unbounded many key queries, but can make only single challenge query.
   (a) **Key Queries**: $\mathcal{A}$ chooses an attributes set Υ and sends it to $\mathcal{C}$ who replies with $\mathsf{sk}_\Upsilon \leftarrow \mathsf{KeyGen}(\mathsf{pk},\mathsf{msk},\Upsilon)$.
   (b) **Challenge Query**: at some point, $\mathcal{A}$ submits a pair of equal length messages $\mathsf{m}_0, \mathsf{m}_1$ and the challenge access policy Π^* to $\mathcal{C}$. The latter samples a random bit $b \leftarrow \{0,1\}$ and replies to $\mathcal{A}$ with $\mathsf{ct}_{\Pi^*} \leftarrow \mathsf{Encrypt}(\mathsf{pk},\Pi^*,\mathsf{m}_b)$.
   We require that Υ does not satisfy Π^* in order to avoid trivial attacks, for any queried Υ.
3. **Output phase**: $\mathcal{A}$ outputs a guess bit $b^{'}$ s the output of the experiment.

Fig. 4. Adaptive indistinguishability security game.

We define the advantage $\mathsf{Adv}_{\mathcal{A}}^{Ada-IND}(1^\kappa)$ of $\mathcal{A}$ in the game as

$$\mathsf{Adv}_{\mathcal{A}}^{Ada-IND}(1^\kappa) = \left|\Pr\left[b^{'} = b\right] - \frac{1}{2}\right|.$$

5.2 ABE from Dualy Computable Accumulator: Intuition

Basic idea. As said previously, having both private evaluation and public witness creation permits us to transform a cryptographic accumulator into an encryption scheme.

More precisely, in our CP-ABE, the user secret key is a privately computed accumulator $\mathsf{acc}_{\mathcal{X}} = g_1^{\boldsymbol{d}_1 \prod_{x \in \mathcal{X}}(x+s)}$, where $\mathcal{X}$ is a representation of the user's attributes. In parallel, the ciphertext is a one-time-pad between the message m and a mask $\boldsymbol{H}$ that is computed using a publicly computable accumulator $\mathsf{accp}_{\mathcal{Y}}$, where $\mathcal{Y}$ is a representation of the access policy. However, with the dually computable accumulator of the previous section as given in Figure 3, this construction is not efficient and secure. We here give only a summary of all the changes we have to make on the accumulator scheme, and we detail them in Appendix A. Before going into those details, we first explain how we can define $\mathcal{X}$ and $\mathcal{Y}$. In the sequel let $Q = 2^q - 1$, where $q \in \mathbb{N}$ is the bound on the number of attributes in the ABE.

Representation of boolean formulas and attributes with cryptographic accumulators. In our ABE, access policies are expressed as disjunctions of conjunctions (DNF), without "NO" gates. Hence, a policy could be noted $\Pi = \pi_1 \vee \pi_2 \vee \cdots \vee \pi_l$, where $l \in \mathbb{N}$, and π_i is a conjunction of attributes. Let $\mathcal{Y}_i$ be the set of attributes present in clause π_i, for $i = 1, \cdots, l$. Our idea is to define $\mathcal{Y}$ as the set $\{\mathcal{H}(\mathcal{Y}_i)\}_{i=1}^{l}$, where $\mathcal{H}$ is a hash function that takes as input a set of elements and returns an element in $\mathbb{Z}_p$, for a prime p. During the encryption process, we create the accumulator $\mathsf{accp}_{\mathcal{Y}}$ using $\mathsf{PublicEval}$ (see below).
For a set Υ of attributes for a given user, we create $\mathcal{X}$ as the set of hash values (using $\mathcal{H}$) of all non-empty subsets of Υ[3]. During the key generation process, the authority hence creates the accumulator $\mathsf{acc}_{\mathcal{X}}$ using Eval.

Encryption and decryption. For a given user, if her set of attributes Υ satisfies the policy Π, it means that there exists a non-empty subset of Υ that corresponds to a clause π_i in Π. As $\mathcal{H}$ is deterministic, it follows that one element, called ξ in the sequel, is present in both accumulators: $\mathsf{acc}_{\mathcal{X}}$ (the one corresponding to the non-empty subsets of Υ) and $\mathsf{accp}_{\mathcal{Y}}$ (the one that corresponds to Π). Based on that, we propose that during the encryption process, the mask $\boldsymbol{H}$ is computed using the public verification equation $\mathsf{PublicVerify}$, as $e(g_1^{\boldsymbol{d}_1}, \mathsf{accp}_{\mathcal{Y}})^{\alpha}$, where α is some randomness.
During decryption, a user having a valid set of attributes precisely knows both the clause π_i and the element in Υ that match together. The next step is then for the user to generate a witness for such element, and thanks to the verification algorithms, retrieve $\boldsymbol{H}$ and then the message. But as both accumulators are not related to the same sets, we cannot directly use the properties of a dually computable accumulator. The user hence needs to compute two witnesses (one for each accumulator), and we need to find a way to combine them appropriately for the decryption to work.

Managing the randomness α and a constant-size ciphertext. The first problem we need to solve is that the element for which the witnesses need to be computed is only known during decryption time, and that we should manage the randomness α. A trivial solution could consist in given as many $g_2^{\alpha \boldsymbol{d}_2^* s^k}$ as necessary to permit the user computing all the possible witnesses. But this option obviously results in (at least) a linear ciphertext. To reach a constant-size ciphertext, we need a way to "anticipate" witnesses during encryption.

[3] It follows that if $|\Upsilon| = k$, then $|\mathcal{X}| = 2^k - 1$.

Here, our trick is to use a specificity of accumulators based on Nguyen's construction, that is the fact that accumulators and witnesses are constructed with the coefficients of polynomials of the form $\mathsf{Ch}[Z] = \prod_{i=1}^{q}(x_i + Z)$. Yet, we know that elementary symmetric polynomials for q variables appear in $\mathsf{Ch}[Z]$ (see Definition 8 and Note 4) and that the coefficient of lowest degree is equal to $\sigma_q = \prod_{i=1}^{q} x_i$. We decide to accumulate in the secret key accumulator a public value, denoted x_0, which is not related to any user attribute, hence having no impact on the decryption capability. From the above observation, we know that x_0 will always be a factor of $\mathsf{Ch}[Z]$'s lowest degree coefficient, no matter the element for which the witness is generated and the user attributes. We proceed similarly for the access policy, introducing the public value y_0 that will be attached to the witness corresponding to the public accumulator. To give the user the possibility to introduce α in the decryption process we then give in the ciphertext the value $\alpha(x_0 + y_0)$.
But this trick necessitates us to modify the way we have computed the witness in our construction in Section 4 so that we can manage the values x_0 and y_0 independently of the other. For that, for a subset $\mathcal{I}$ in $\mathcal{X}$, the witness is now divided into two parts: $\mathsf{wit}_{\mathcal{I}} = (W_1, W_2)$ where $W_1 = g_1^{\boldsymbol{d}_1 b_0}$ and $W_2 = g_2^{\boldsymbol{d}_2^* \sum_{i=1}^{q} b_i s^i}$. This intermediate accumulator is presented in Figure 6, in Appendix B. Again, we proceed similarly for the publicly computable accumulator with witness $\mathsf{witp}_{\mathcal{I}} = (W_1', W_2')$.

Auxiliary information in the ciphertext. From the previous issue, we now know that the ciphertext should include a first auxiliary information to permit decryption: $\mathsf{aux}_1 = g_1^{\boldsymbol{d}_1 \alpha(x_0+y_0)}$. At this step, we also need to give $\mathsf{aux}_2 = g_2^{-\alpha \boldsymbol{d}_1^*}$ with the ciphertext, so that the Verify algorithm, on input such value and the secretly computed accumulator now includes the randomness α.
But from aux_1 and $\left\{ g_2^{\boldsymbol{d}_1^* s^i} \right\}_{i=0}^{Q}$, anyone can compute $e(g_1^{\boldsymbol{d}_1(x_0+y_0)}, \mathsf{accp}_{\mathcal{Y}}) = \boldsymbol{H}^{x_0+y_0}$. As x_0, y_0 are publicly known, this permits to recover $\boldsymbol{H}$ and hence the message. To avoid that, our idea is to split α into two randoms α_1, α_2, and modify the auxiliary information accordingly, as $\mathsf{aux}_1 = g_1^{\boldsymbol{d}_1 \alpha_2(x_0+y_0)}$ and $\mathsf{aux}_2 = g_2^{-\alpha_1\alpha_2 \boldsymbol{d}_1^*}$. For the same reason as above, we cannot directly include α_1 and need to find another trick.
Indeed, we use the same "anticipation" trick that we used for the witnesses. More precisely, we add an additional public value z_0 in both accumulators. The consequence is that, at the time of decryption, the users obtains that the element ξ and the value z_0 are both in the two accumulators. Hence, in the verification process, we necessarily have the term $s^2 + s(z_0 + \xi) + z_0\xi$ which can be divided in two parts: $s^2 + sz_0$ and $s + z_0$. It follows that during encryption, we additionally give the terms $ele_1 = g_2^{\alpha_1 \boldsymbol{d}_1(z_0+s)}$ and $ele_2 = g_2^{\alpha_1 \boldsymbol{d}_1(z_0 s + s^2)}$ that are associated to aux_1 using a pairing during the decryption process. This indirectly brings α_1 to aux_1 without revealing it.

We now have fully treated the case of W_1 and W_1' but we also need to add the randomness (α_1, α_2) to W_2 and W_2'. To solve that we simply need to add two new auxiliary information: $ele_3 = g_1^{\alpha_1\alpha_2 \boldsymbol{d}_2(z_0 s + s^2)}$ and $ele_4 = g_1^{\alpha_1\alpha_2 \boldsymbol{d}_2(z_0+s)}$.

Managing the dual system encryption framework. To prove the security of our ABE, we need to use the dual system encryption framework [38]. In a nutshell, during the security proof, such technique introduces the notion of semi-functional (SF) keys and ciphertexts, which should be indistinguishable from normal keys and ciphertexts. Such new elements behave exactly the same as their normal counterparts, except that no semi-functional key can decrypt an SF ciphertext. During the security proof, the simulator changes all the keys issued to the adversary into SF ones, and make the challenge answer to the adversary an SF ciphertext. This way, the adversary cannot extract any information from the challenge ciphertext: it has no advantage.

To manage semi-functional ciphertexts and secret keys in our own proof, we need to increase by one the dimension of the DPVS. More precisely, we rely on the Decisional Subspace assumptions in $\mathbb{G}_1$ (DS1) and in $\mathbb{G}_2$ (DS2) [12] (which hold if SXDH holds), which necessitate to guess between $g_1^{\tau_1 \boldsymbol{d}_i}$ (resp. $g_2^{\tau_1 \boldsymbol{d}_i^*}$) and $g_1^{\tau_1 \boldsymbol{d}_i + \tau_2 \boldsymbol{d}_{i+k}}$ (resp. $g_2^{\tau_1 \boldsymbol{d}_i^* + \tau_2 \boldsymbol{d}_{i+k}^*}$) for $i = 1, \cdots, k$, where $k \in \mathbb{N}$ is one parameter of the assumption, and $\tau_1, \tau_2 \in \mathbb{Z}_p$ are random elements chosen by the challenger. To avoid disturbance with the base used in the accumulator, we will not use $\boldsymbol{d}_1$ to bring SF space. Instead we consider $\boldsymbol{d}_2$ in the secret key and $\boldsymbol{d}_2^*$ in aux_2. More precisely, we generate two randoms $r, z \in \mathbb{Z}_p$ and generate $r \cdot \boldsymbol{d}_2$ and $z \cdot \boldsymbol{d}_2^*$ to have the same semi-functional part in the ciphertext than the one we have in the secret key. The randoms r and z are used to match the assumptions in which $\boldsymbol{d}_2$ (resp. $\boldsymbol{d}_2^*$) are randomized (by τ_1). But this results in an additional term $e(g_1, g_2)^{\psi r z \alpha_1 \alpha_2}$ during decryption. To avoid this, we need to introduce a new dimension in the DPVS, and then $(\boldsymbol{d_3}, \boldsymbol{d_3}^*)$. It follows that the secret accumulator becomes $\mathsf{acc}_{\mathcal{X}} = g_1^{\boldsymbol{d_1} \sum_{i=0}^{Q} a_i s^i + r(\boldsymbol{d}_2 - \gamma \boldsymbol{d}_3)}$ and $\mathsf{aux}_2 = g_2^{-\boldsymbol{d_1}^* \alpha + z(\gamma \boldsymbol{d}_2^* + \boldsymbol{d}_3^*)}$. This results in our second intermediate accumulator scheme, presented in Figure 7 in Appendix B.

Managing the third bases. There is one last change we need to do in our accumulator. Indeed, in the last part of the CP-ABE security proof, we need to randomize the dual orthonormal bases $(\mathbb{D}, \mathbb{D}^*)$ to new bases $(\mathbb{F}, \mathbb{F}^*)$, so that with the latter, the adversary has no more possibility to win the game. This modification implies that we need to express $\boldsymbol{d}_1$ as $\boldsymbol{f}_1 + \eta \boldsymbol{f}_5$, which means that any element having $\boldsymbol{d}_1$ in the exponent will have a SF part when expressed in bases $(\mathbb{F}, \mathbb{F}^*)$. It results that the elements aux_1 and $g_1^{\boldsymbol{d}_1}$ used in $\boldsymbol{H}$ have now a SF part, while we defined a SF ciphertext such that only aux_2 contains the SF components.

Our idea here is then to replace $\boldsymbol{d}_1$ by $\boldsymbol{d}_3$ in the witness creation: hence, the witness element W_1 goes from $g_1^{\boldsymbol{d}_1 b_0}$ to $g_1^{\boldsymbol{d}_3 b_0}$. To keep the orthonormality of the DPVS, we also replace $\boldsymbol{d}_1^*$ by $\boldsymbol{d}_3^*$ in the public evaluation of the accumulator and the publicly computed accumulator goes from $g_2^{\boldsymbol{d}_1^* \sum_{i=0}^{Q} m_i s^i}$ to $g_2^{\boldsymbol{d}_3^* \sum_{i=0}^{Q} m_i s^i}$. We then change aux_1 to $g_1^{\boldsymbol{d}_3(x_0+y_0)}$, $ele_1 = g_2^{\alpha_1 \boldsymbol{d}_3^* (z_0 s + s^2)}$, and $ele_2 = g_2^{\alpha_1 \boldsymbol{d}_3^* (z_0 + s)}$. This gives us the final dually computable accumulator that we use to design our CP-ABE, fully given in Appendix B (see Figure 8). For our CP-ABE we will use DS1 and DS2 with parameter $k = 3$ and $n = 2k = 6$, and so DPVS of dimension 6.

5.3 Our CP-ABE Scheme

The resulting CP-ABE is fully given in Figure 5. As said above, it permits to manage access policies expressed as disjunctions of conjunctions without "NO" gates. For sake of clarity, we highlight the underlying dually computable accumulator scheme with colors as follows: the privately computed accumulator is in green, the publicly computed accumulator is in blue, the anticipation of the first element of the witnesses is in orange, the second parts of the witnesses are in purple and the anticipation of the intersection of both sets is in red.

Theorem 4. *Our ciphertext policy attribute based encryption scheme is correct.*

Proof.

$$\begin{aligned}
& e(\mathsf{aux}_1^{\delta\delta'}, ele_1 \cdot ele_2^{\xi}) \\
= \quad & e((g_1^{\alpha_2 \boldsymbol{d}_3 (x_0+y_0)})^{\delta\delta'}, g_2^{\alpha_1 \boldsymbol{d}_3^* (z_0 s + s^2)} \cdot (g_2^{\alpha_1 \boldsymbol{d}_3^* (z_0+s)})^{\xi}) \\
= \quad & e(g_1^{\alpha_2 \boldsymbol{d}_3 \delta\delta' (x_0+y_0)}, g_2^{\alpha \boldsymbol{d}_3^* (s^2+s(z_0+\xi)+z_0\xi)}) \\
= \quad & e(g_1, g_2)^{\psi\alpha_1\alpha_2(s^2+s(z_0+\xi)+z_0\xi)c_0\delta'} \cdot e(g_1, g_2)^{\psi\alpha_1\alpha_2(s^2+s(z_0+\xi)+z_0\xi)t_0\delta}
\end{aligned}$$

$$\begin{aligned}
& e(ele_3 \cdot ele_4^{\xi}, W_2^{\delta'} \cdot W_2'^{\delta}) \\
= \quad & e(g_1^{\alpha_1\alpha_2 \boldsymbol{d}_2 (z_0 s + s^2)} \cdot (g_1^{\alpha_1\alpha_2 \boldsymbol{d}_2 (z_0+s)})^{\xi}, (g_2^{\boldsymbol{d}_2^* \sum_{i=1}^{Q} c_i s^i})^{\delta'} \cdot (g_2^{\boldsymbol{d}_2^* \sum_{i=1}^{Q} t_i s^i})^{\delta}) \\
= \quad & e(g_1^{\alpha_1\alpha_2 \boldsymbol{d}_2 (s^2+s(z_0+\xi)+z_0\xi)}, g_2^{\boldsymbol{d}_2^* \delta' \sum_{i=1}^{Q} c_i s^i + \boldsymbol{d}_2^* \delta \sum_{i=1}^{Q} t_i s^i}) \\
= \quad & e(g_1, g_2)^{\psi\alpha_1\alpha_2(s^2+s(z_0+\xi)+z_0\xi)\delta' \sum_{i=1}^{Q} c_i s^i} \\
& \cdot e(g_1, g_2)^{\psi\alpha_1\alpha_2(s^2+s(z_0+\xi)+z_0\xi)\delta \sum_{i=1}^{Q} t_i s^i}
\end{aligned}$$

Therefore

$$\begin{aligned}
& e(\mathsf{aux}_1^{\delta\delta'}, ele_1 \cdot ele_2^{\xi}) \cdot e(ele_3 \cdot ele_4^{\xi}, W_2^{\delta'} \cdot W_2'^{\delta}) \\
= \quad & e(g_1, g_2)^{\psi\alpha_1\alpha_2(s^2+s(z_0+\xi)+z_0\xi)\delta' \sum_{i=0}^{Q} c_i s^i} \\
& \cdot e(g_1, g_2)^{\psi\alpha_1\alpha_2(s^2+s(z_0+\xi)+z_0\xi)\delta \sum_{i=0}^{Q} t_i s^i}
\end{aligned}$$

If ξ belongs to $\mathcal{X}$ and ξ belongs to $\mathcal{Y}$, then

$$\begin{aligned}
& e(\mathsf{aux}_1^{\delta\delta'}, ele_1 \cdot ele_2^{\xi}) \cdot e(ele_3 \cdot ele_4^{\xi}, W_2^{\delta'} \cdot W_2'^{\delta}) \\
= \ & e(g_1, g_2)^{\psi\alpha_1\alpha_2\delta' \sum_{i=0}^{Q} a_i s^i} \cdot e(g_1, g_2)^{\psi\alpha_1\alpha_2\delta \sum_{i=0}^{Q} m_i s^i}
\end{aligned}$$

The last pairing is equal to

$$\begin{aligned}
& e(\mathsf{acc}_{\mathcal{X}}, \mathsf{aux}_2)^{\delta'} \\
= \quad & e(g_1^{\boldsymbol{d}_1 \sum_{i=0}^{Q} a_i s^i + r(\boldsymbol{d}_2 - \gamma \boldsymbol{d}_3)}, g_2^{-\boldsymbol{d}_1^* \alpha_1\alpha_2 + z(\gamma \boldsymbol{d}_2^* + \boldsymbol{d}_3^*)})^{\delta'} \\
= \ & e(g_1, g_2)^{-\alpha_1\alpha_2\psi \sum_{i=0}^{Q} a_i s^i \delta'} \cdot e(g_1, g_2)^{rz\gamma\psi} \cdot e(g_1, g_2)^{-rz\gamma\psi} \\
= \quad & e(g_1, g_2)^{-\alpha_1\alpha_2\psi \sum_{i=0}^{Q} a_i s^i \delta'}
\end{aligned}$$

- $\mathsf{Setup}(1^\lambda, 1^q)$: generate bilinear group $\mathit{\Gamma} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, e, g_1, g_2)$, dual pairing vector spaces $(\mathbb{D}, \mathbb{D}^*) \leftarrow \mathsf{Dual}(\mathbb{Z}_p^6)$ such that $\mathbb{D} = (\boldsymbol{d}_1, \cdots, \boldsymbol{d}_6)$, $\mathbb{D}^* = (\boldsymbol{d}_1^*, \cdots, \boldsymbol{d}_6^*)$ and $\boldsymbol{d}_i \cdot \boldsymbol{d}_i^* = \psi$, for $i = 1, \cdots, 6$ and $\psi \in \mathbb{Z}_p$. Also choose $\gamma, s, x_0, y_0, z_0 \leftarrow \mathbb{Z}_p$ and a hash function $\mathcal{H}$ that takes as input an attributes set and outputs an element of $\mathbb{Z}_p \setminus \{\gamma, s, x_0, y_0, z_0\}$. Set $Q = 2^q - 1$, $\mathsf{msk} = \left(\gamma, s, g_2^{\boldsymbol{d}_2^*}, \left\{g_1^{\boldsymbol{d}_1 s^i}\right\}_{i=0}^{Q}, \left\{g_1^{\boldsymbol{d}_3 s^i}\right\}_{i=1}^{Q}\right)$ and

$$\mathsf{pk} = \left(\begin{matrix} \mathit{\Gamma}, g_1^{\boldsymbol{d}_3}, g_1^{\boldsymbol{d}_2}, g_1^{\boldsymbol{d}_2 s}, \cdots, g_1^{\boldsymbol{d}_2 s^Q}, g_2^{\boldsymbol{d}_1^*}, g_2^{\boldsymbol{d}_1^* s}, \cdots, g_2^{\boldsymbol{d}_1^* s^Q}, g_2^{\boldsymbol{d}_2^* \gamma}, \\ g_2^{\boldsymbol{d}_2^* s}, \cdots, g_2^{\boldsymbol{d}_2^* s^Q}, g_2^{\boldsymbol{d}_3^*}, g_2^{\boldsymbol{d}_3^* s}, \cdots, g_2^{\boldsymbol{d}_3^* s^Q}, \mathcal{H}, x_0, y_0, z_0 \end{matrix}\right).$$

  Return $\mathsf{msk}, \mathsf{pk}$.
- $\mathsf{KeyGen}(\mathsf{pk}, \mathsf{msk}, \mathit{\Upsilon})$: let $k \in \mathbb{N}$ be the number of attributes in $\mathit{\Upsilon}$. Compute $p_1, \cdots, p_{2^k-1}$ all the non-empty subsets of $\mathit{\Upsilon}$ and set $\mathcal{X} = \{\mathcal{H}(p_i)\}_{i=1}^{2^k-1} \cup \{x_0, z_0\}$. Compute $\{a_i\}_{i=0,\cdots,Q}$ the coefficients of the polynomial $\mathsf{Ch}_{\mathcal{X}}[Z] = (x_0 + Z) \cdot (z_0 + Z) \cdot \prod_{i=1}^{2^k-1}(\mathcal{H}(p_i) + Z)$. Pick $r \leftarrow \mathbb{Z}_p$ and set

$$\mathsf{sk}_{\mathit{\Upsilon}} = \mathsf{acc}_{\mathcal{X}} = g_1^{\boldsymbol{d}_1 \sum_{i=0}^{Q} a_i s^i + r(\boldsymbol{d}_2 - \gamma \boldsymbol{d}_3)}.$$

- $\mathsf{Encrypt}(\mathsf{pk}, \mathit{\Pi}, \mathsf{m})$: let $\mathit{\Pi} = \pi_1 \vee \pi_2 \vee \cdots \vee \pi_l$ be the access policy, where $l \in \mathbb{N}$ is the number of clauses in the policy, and π_i for $i = 1, \cdots, l$ is a conjunction of attributes. Define $\mathcal{Y}_i$ for $i = 1, \cdots, l$ as the set of attributes associated to clause π_i and $\mathcal{Y} = \cup_{i=1}^{l} \mathcal{H}(\mathcal{Y}_i) \cup \{y_0, z_0\}$. Let $\{m_i\}_{i=0}^{Q}$ be the coefficients of polynomial $\mathsf{Ch}_{\mathcal{Y}}[Z]$.
  - *Mask computation:* choose $z, \alpha_1, \alpha_2 \leftarrow \mathbb{Z}_p$ and define $\mathsf{accp}_{\mathcal{Y}} = g_2^{\boldsymbol{d}_3^* \sum_{i=0}^{Q} m_i s^i}$ and $\boldsymbol{H} = e(g_1^{\boldsymbol{d}_3}, \mathsf{accp}_{\mathcal{Y}})^{\alpha_1 \alpha_2}$.
  - *Auxiliary information computation:* $\mathsf{aux}_1 = g_1^{\alpha_2 \boldsymbol{d}_3 (x_0 + y_0)}$ and $\mathsf{aux}_2 = g_2^{-\boldsymbol{d}_1^* \alpha_1 \alpha_2 + z(\gamma \boldsymbol{d}_2^* + \boldsymbol{d}_3^*)}$.
  - *Anticipation of the element computation:* $ele_1 = g_2^{\alpha_1 \boldsymbol{d}_3^* (z_0 s + s^2)}$, $ele_2 = g_2^{\alpha_1 \boldsymbol{d}_3^* (z_0 + s)}$, $ele_3 = g_1^{\alpha_1 \alpha_2 \boldsymbol{d}_2 (z_0 s + s^2)}$, and $ele_4 = g_1^{\alpha_1 \alpha_2 \boldsymbol{d}_2 (z_0 + s)}$.

  Set $\mathsf{ct}_{\mathit{\Pi}} = (ele_1, ele_2, ele_3, ele_4, \mathsf{aux}_1, \mathsf{aux}_2, \mathsf{m} \cdot \boldsymbol{H})$ and return $\mathsf{ct}_{\mathit{\Pi}}$.

- $\mathsf{Decrypt}(\mathsf{pk}, \mathsf{sk}_{\mathit{\Upsilon}}, \mathit{\Upsilon}, \mathsf{ct}_{\mathit{\Pi}}, \mathit{\Pi})$: find p_{j^*} (for $j^* \in \{1, \cdots, 2^k - 1\}$) the non-empty subset of $\mathit{\Upsilon}$ that satisfies $\mathit{\Pi}$ (if no such subset exists, then return reject symbol $\perp$). It means that there exist $j \in [1, \cdots, l]$ such that $\mathcal{Y}_j = p_{j^*}$ and $\mathcal{H}(\mathcal{Y}_j) = \mathcal{H}(p_{j^*}) = \xi$. Let $\{c_i\}_{i=0}^{Q}$ be the coefficients of the polynomial $\mathsf{Ch}_{\mathcal{X}}[Z]/((z_0 + Z)(\xi + Z))$. Let $\{t_i\}_{i=0}^{Q}$ be the coefficients of the polynomial $\mathsf{Ch}_{\mathcal{Y}}[Z]/((z_0 + Z)(\xi + Z))$. Find $\delta, \delta' \in \mathbb{Z}_p$ such that $c_0 = x_0 \delta$ and $t_0 = y_0 \delta'$. Set $W_2 = g_2^{\boldsymbol{d}_2^* \sum_{i=1}^{Q} c_i s^i}$, $W_2' = g_2^{\boldsymbol{d}_2^* \sum_{i=1}^{Q} t_i s^i}$ and compute

$$\frac{\mathsf{m} \cdot \boldsymbol{H}}{\left(e(\mathsf{aux}_1^{\delta\delta'}, ele_1 \cdot ele_2^{\xi}) \cdot e(ele_3 \cdot ele_4^{\xi}, W_2^{\delta'} \cdot W_2'^{\delta}) \cdot e(\mathsf{acc}_{\mathcal{X}}, \mathsf{aux}_2)^{\delta'}\right)^{\delta^{-1}}}$$

  to get m or $\perp$.

Fig. 5. Our CP-ABE scheme.

so multiplying it with $e(\mathsf{aux}_1^{\delta\delta'}, ele_1 \cdot ele_2^{\xi}) \cdot e(ele_3 \cdot ele_4^{\xi}, W_2^{\delta'} \cdot W_2'^{\delta})$ gives $e(g_1, g_2)^{\psi\alpha_1\alpha_2\delta\sum_{i=0}^{Q} m_i s^i}$. As we know δ we can recover the mask of the message and then the message. Therefore, the scheme is correct. $\square$

The adaptive indistinguishability is given by the following theorem.

Theorem 5. *Our ABE satisfies adaptive indistinguishability under* SXDH *assumption.*

To prove the security of our scheme, we prove that the encryption of challenge message is indistinguishable from the encryption of a random message. To do so, we use a sequence of games (our proof is inspired of Chen *et al.* [12]'s IBE security proof) and Water's dual system encryption framework [38]. Let $N_q \in \mathbb{N}$ be the number of secret keys that the adversary is allowed to query.[4]

- Game_{Real} is the original security game, as presented in Figure 4.
- Game_0 is the same as Game_{Real} except that the challenge ciphertext is a *semi-functional* ciphertext.
- Game_i for $i = 1, \cdots, N_q$ is the same as Game_0 except that the first i keys are semi-functional.
- Game_{Final} is the same as Game_{N_q} except that the challenge ciphertext is an encryption of a random message.

Now we define semi-functional (SF) keys and ciphertexts. Let $t_5, t_6, z_5, z_6 \leftarrow \mathbb{Z}_p$.

- a semi-functional key for Υ, $\mathsf{sk}_{\Upsilon}^{(SF)}$, is computed from normal key sk_{Υ} as $\mathsf{sk}_{\Upsilon}^{(SF)} = \mathsf{sk}_{\Upsilon} \cdot g_1^{t_5 \boldsymbol{d}_5 + t_6 \boldsymbol{d}_6} = g_1^{\boldsymbol{d}_1 \sum_{i=0}^{Q} a_i s^i + r(\boldsymbol{d}_2 - \gamma \boldsymbol{d}_3) + t_5 \boldsymbol{d}_5 + t_6 \boldsymbol{d}_6}$
- a semi-functional ciphertext for Π, $\mathsf{ct}_{\Pi}^{(SF)}$, is computed as a normal ciphertext ct_{Π} except that $\mathsf{aux}_2^{(SF)} = \mathsf{aux}_2 \cdot g_2^{z_5 \boldsymbol{d}_5^* + z_6 \boldsymbol{d}_6^*}$.

Notice that normal keys can decrypt SF ciphertexts, and normal ciphertexts can be decrypted by SF keys. However, decryption of a SF ciphertext by a SF key leads to an additional term: $1/e(g_1, g_2)^{(t_5 z_5 \psi + t_6 z_6 \psi)\delta^{-1}}$.

Intuition of the proof. The proof is using the two assumptions DS1 and DS2 that hold if SXDH holds, and is done as follows.

First we prove that if there exists an adversary that can distinguish Game_{Real} from Game_0 we can build an adversary that breaks the DS2 assumption with parameters $k = 3$ and $n = 6$. To do so the main idea is to use the assumption's challenge to build the challenge ciphertext. Depending on the value of the challenge we will either obtain a normal form ciphertext or a semi-functional form one.

Then we prove that if there exists an adversary that can distinguish Game_{j-1} from Game_j for $j = 1, \cdots, N_q$ we can build an adversary that breaks the DS1 assumption with $k = 3$ and $n = 6$. The idea is to use the assumption's challenge to build the j-th key. Thus, depending on the value of the challenge we will either obtain a normal

[4] As the number of attributes in the scheme is bounded, so is the number of keys that an adversary can query.

form key or a semi-functional form one. To build the challenge ciphertext, we use the assumption's parameters to obtain a semi-functional ciphertext.

Finally, we prove that Game_{N_q} is computationally indistinguishable from Game_{Final}, with a change of dual orthonormal bases. Doing so, we randomized the coefficient of $\boldsymbol{d}_1$ in the aux_2 term of the ciphertext, thereby severing its link with the blinding factor. That gives us the encryption of a random message.

In the following, the advantage of a PPT adversary $\mathcal{A}$ to win a game Game, is written $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Game}}$. Here is the proof of Theorem 5.

Proof. **Lemma 2.** *If there exists a PPT algorithm $\mathcal{A}$ such that $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Game}_{Real}} - \mathsf{Adv}_{\mathcal{A}}^{\mathsf{Game}_0}$ is non-negligible, then there exists a PPT algorithm $\mathcal{B}$ with non-negligible advantage against assumption DS2 with $k = 3$ and $n = 6$.*

Proof. `INIT`: $\mathcal{B}$ is given $\Delta = (\mathit{\Gamma}, g_1^{\boldsymbol{b}_1}, g_1^{\boldsymbol{b}_2}, g_1^{\boldsymbol{b}_3}, g_2^{\boldsymbol{b}_1^*}, g_2^{\boldsymbol{b}_2^*}, g_2^{\boldsymbol{b}_3^*}, g_2^{\boldsymbol{b}_4^*}, g_2^{\boldsymbol{b}_5^*}, g_2^{\boldsymbol{b}_6^*}, u_1, u_2, u_3, \mu_2)$ along with t_1, t_2, t_3. $\mathcal{B}$ must decide if t_1, t_2, t_3 are distributed as $g_2^{\tau_1 \boldsymbol{b}_1^*}, g_2^{\tau_1 \boldsymbol{b}_2^*}, g_2^{\tau_1 \boldsymbol{b}_3^*}$ or $g_2^{\tau_1 \boldsymbol{b}_1^* + \tau_2 \boldsymbol{b}_4^*}, g_2^{\tau_1 \boldsymbol{b}_2^* + \tau_2 \boldsymbol{b}_5^*}, g_2^{\tau_1 \boldsymbol{b}_3^* + \tau_2 \boldsymbol{b}_6^*}$.

`SETUP`: $\mathcal{B}$ first chooses a random invertible matrix $\boldsymbol{A} \in \mathbb{Z}_p^{3\times 3}$. It implicitly sets dual orthonormal bases $\mathbb{D}, \mathbb{D}^*$ to: $\boldsymbol{d}_1^* = \boldsymbol{b}_1^*$,$\boldsymbol{d}_2^* = \boldsymbol{b}_2^*$, $\boldsymbol{d}_3^* = \boldsymbol{b}_3^*$, $(\boldsymbol{d}_4^*, \boldsymbol{d}_5^*, \boldsymbol{d}_6^*) = (\boldsymbol{b}_4^*, \boldsymbol{b}_5^*, \boldsymbol{b}_6^*) \cdot \boldsymbol{A}$, $\boldsymbol{d}_1 = \boldsymbol{b}_1$, $\boldsymbol{d}_2 = \boldsymbol{b}_2$, $\boldsymbol{d}_3 = \boldsymbol{b}_3$, $(\boldsymbol{d}_4, \boldsymbol{d}_5, \boldsymbol{d}_6) = (\boldsymbol{b}_4, \boldsymbol{b}_5, \boldsymbol{b}_6) \cdot (\boldsymbol{A}^{-1})^\top$.

We note that $\mathbb{D}, \mathbb{D}^*$ are properly distributed and reveal no information about $\boldsymbol{A}$. Notice also that $\mathcal{B}$ cannot produce $g_1^{\boldsymbol{d}_4}, g_1^{\boldsymbol{d}_5}, g_1^{\boldsymbol{d}_6}$, but these will not be needed to create normal keys. $\mathcal{B}$ chooses random values $\gamma, s, x_0, y_0, z_0 \in \mathbb{Z}_p$ and a hash function $\mathcal{H}$ that takes as input attributes set and outputs an element of $\mathbb{Z}_p \setminus \{\gamma, s, x_0, y_0, z_0\}$. $\mathcal{A}$ is given the public key

$$\mathsf{pk} = \begin{pmatrix} \mathit{\Gamma}, g_1^{\boldsymbol{d}_3}, g_1^{\boldsymbol{d}_2}, g_1^{\boldsymbol{d}_2 s}, \cdots, g_1^{\boldsymbol{d}_2 s^Q}, g_2^{\boldsymbol{d}_1^*}, g_2^{\boldsymbol{d}_1^* s}, \cdots, g_2^{\boldsymbol{d}_1^* s^Q}, g_2^{\boldsymbol{d}_2^* \gamma}, \\ g_2^{\boldsymbol{d}_2^* s}, \cdots, g_2^{\boldsymbol{d}_2^* s^Q}, g_2^{\boldsymbol{d}_3^*}, g_2^{\boldsymbol{d}_3^* s}, \cdots, g_2^{\boldsymbol{d}_3^* s^Q}, \mathcal{H}, x_0, y_0, z_0 \end{pmatrix}$$

The master key is $\mathsf{msk} = \left(\gamma, s, g_2^{\boldsymbol{d}_2^*}, \left\{g_1^{\boldsymbol{d}_1 s^i}\right\}_{i=0}^{Q}, \left\{g_1^{\boldsymbol{d}_3 s^i}\right\}_{i=1}^{Q}\right)$.

`KEY QUERY`: msk is known to $\mathcal{B}$, which allows $\mathcal{B}$ to respond to all of $\mathcal{A}$'s key queries by calling the normal key generation algorithm.

`CHALLENGE`: $\mathcal{A}$ sends to $\mathcal{B}$ a challenge policy $\mathit{\Pi}^*$ and two challenge messages $\mathsf{m}_0, \mathsf{m}_1$. $\mathcal{B}$ chooses a random bit $b \in \{0, 1\}$ and encrypts m_b under $\mathit{\Pi}^*$ as follows: $z, \alpha_1, \alpha_2 \leftarrow \mathbb{Z}_p$ and

$$\begin{array}{cc} \mathsf{accp}_{\mathcal{Y}} = g_2^{\boldsymbol{b}_3^* \sum_{i=0}^{Q} m_i s^i} & \boldsymbol{H} = e(g_1^{\boldsymbol{b}_3}, \mathsf{accp}_{\mathcal{Y}})^{\alpha_1 \alpha_2} \\ \mathsf{aux}_1 = g_1^{\alpha_2 \boldsymbol{b}_3 (x_0 + y_0)} & \mathsf{aux}_2 = g_2^{-\boldsymbol{b}_1^* \alpha_1 \alpha_2} \cdot t_2^{\gamma} \cdot t_3 \\ ele_1 = g_2^{\alpha_1 \boldsymbol{b}_3^* (z_0 s + s^2)} & ele_2 = g_2^{\alpha_1 \boldsymbol{b}_3^* (z_0 + s)} \\ ele_3 = g_1^{\alpha_1 \alpha_2 \boldsymbol{b}_2 (z_0 s + s^2)} & ele_4 = g_1^{\alpha_1 \alpha_2 \boldsymbol{b}_2 (z_0 + s)} \end{array}$$

where $\mathcal{Y} = \{\mathcal{H}(\mathcal{Y}_i)\}_{i=1}^{l} \cup \{y_0, z_0\}$, and $\mathcal{Y}_i$ for $i = 1, \cdots, l$ is a set that contains the elements of the clause π_i^*. It gives the ciphertext $\mathsf{ct}^* = (ele_1, ele_2, ele_3, ele_4, \mathsf{aux}_1, \mathsf{aux}_2, \mathsf{m} \cdot \boldsymbol{H})$ to $\mathcal{A}$.

– If $(t_1, t_2, t_3) = (g_2^{\tau_1 \boldsymbol{b}_1^*}, g_2^{\tau_1 \boldsymbol{b}_2^*}, g_2^{\tau_1 \boldsymbol{b}_3^*})$, we have a normal ciphertext with randomness $z = \tau_1$.

$$\begin{aligned} \mathsf{accp}_{\mathcal{Y}} &= g_2^{\boldsymbol{d}_3^* \sum_{i=0}^{Q} m_i s^i} & \boldsymbol{H} &= e(g_1^{\boldsymbol{d}_3}, \mathsf{accp}_{\mathcal{Y}})^{\alpha_1 \alpha_2} \\ \mathsf{aux}_1 &= g_1^{\alpha_2 \boldsymbol{d}_3 (x_0 + y_0)} & \mathsf{aux}_2 &= g_2^{-\boldsymbol{d}_1^* \alpha_1 \alpha_2 + \tau_1(\gamma \boldsymbol{d}_2^* + \boldsymbol{d}_3^*)} \cdot t_2^{\gamma} \cdot t_3 \\ ele_1 &= g_2^{\alpha_1 \boldsymbol{d}_3^* (z_0 s + s^2)} & ele_2 &= g_2^{\alpha_1 \boldsymbol{d}_3^* (z_0 + s)} \\ ele_3 &= g_1^{\alpha_1 \alpha_2 \boldsymbol{d}_2 (z_0 s + s^2)} & ele_4 &= g_1^{\alpha_1 \alpha_2 \boldsymbol{d}_2 (z_0 + s)} \end{aligned}$$

Thus $\mathcal{B}$ has properly simulated Game_{Real}.

– If $(t_1, t_2, t_3) = (g_2^{\tau_1 \boldsymbol{b}_1^* + \tau_2 \boldsymbol{b}_4^*}, g_2^{\tau_1 \boldsymbol{b}_2^* + \tau_2 \boldsymbol{b}_5^*}, g_2^{\tau_1 \boldsymbol{b}_3^* + \tau_2 \boldsymbol{b}_6^*})$, then we have that aux_2 is equal to $g_2^{-\boldsymbol{d}_1^* \alpha_1 \alpha_2 + \tau_1(\gamma \boldsymbol{d}_2^* + \boldsymbol{d}_3^*) + \tau_2 \gamma \boldsymbol{b}_5^* + \tau_2 \boldsymbol{b}_6^*}$.

This ciphertext has an additional term with coefficients in bases $\boldsymbol{b}_5^*, \boldsymbol{b}_6^*$, which form the vector $\tau_2(\gamma, 1)$. To compute coefficients in the bases $(\boldsymbol{d}_5^*, \boldsymbol{d}_6^*)$ we multiply the matrix $\boldsymbol{A}^{-1}$ by the transpose of this vector. Since $\boldsymbol{A}$ is random, these new coefficients are uniformly random. Thus, in this case, the ciphertext is SF (with coefficients in the base $\mathbb{D}$) and $\mathcal{B}$ has properly simulated Game_0. This allows $\mathcal{B}$ to leverage $\mathcal{A}$'s non-negligible difference in advantage between Game_{Real} and Game_0 to achieve a non-negligible advantage against DS2. □

Lemma 3. *If there exists a PPT algorithm $\mathcal{A}$ such that $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Game}_{j-1}} - \mathsf{Adv}_{\mathcal{A}}^{\mathsf{Game}_j}$ (for $j = 1, \cdots, N_q$) is non-negligible, then there exists a PPT algorithm $\mathcal{B}$ with non-negligible advantage against assumption DS1 with $k = 3$ and $n = 6$.*

Proof. `INIT`: $\mathcal{B}$ is given $\Delta = (\mathit{\Gamma}, g_2^{\boldsymbol{b}_1^*}, g_2^{\boldsymbol{b}_2^*}, g_2^{\boldsymbol{b}_3^*}, g_1^{\boldsymbol{b}_1}, g_1^{\boldsymbol{b}_2}, g_1^{\boldsymbol{b}_3}, g_1^{\boldsymbol{b}_4}, g_1^{\boldsymbol{b}_5}, g_1^{\boldsymbol{b}_5}, u_1, u_2, u_3, \mu_2)$ along with t_1, t_2, t_3, distributed either as $g_1^{\tau_1 \boldsymbol{b}_1}, g_1^{\tau_1 \boldsymbol{b}_2}, g_1^{\tau_1 \boldsymbol{b}_3}$ or $g_1^{\tau_1 \boldsymbol{b}_1 + \tau_2 \boldsymbol{b}_3}, g_1^{\tau_1 \boldsymbol{b}_2 + \tau_2 \boldsymbol{b}_4}, g_1^{\tau_1 \boldsymbol{b}_3 + \tau_2 \boldsymbol{b}_6}$.

`SETUP`: $\mathcal{B}$ chooses a random invertible matrix $\boldsymbol{A} \in \mathbb{Z}_q^{3\times 3}$. Then it implicitly sets dual orthonormal bases $\mathbb{D}, \mathbb{D}^*$ to: $\boldsymbol{d}_1^* = \boldsymbol{b}_1^*$, $\boldsymbol{d}_2^* = \boldsymbol{b}_2^*$, $\boldsymbol{d}_3^* = \boldsymbol{b}_3^*$ $(\boldsymbol{d}_4^*, \boldsymbol{d}_5^*, \boldsymbol{d}_6^*) = (\boldsymbol{b}_4^*, \boldsymbol{b}_5^*, \boldsymbol{b}_6^*) \cdot \boldsymbol{A}$, $\boldsymbol{d}_1 = \boldsymbol{b}_1^*$, $\boldsymbol{d}_2^* = \boldsymbol{b}_2^*$, $\boldsymbol{d}_3^* = \boldsymbol{b}_3^*$, $(\boldsymbol{d}_4, \boldsymbol{d}_5, \boldsymbol{d}_6) = (\boldsymbol{b}_4, \boldsymbol{b}_5, \boldsymbol{b}_6) \cdot (\boldsymbol{A}^{-1})^{\top}$.

We note that $\mathbb{D}, \mathbb{D}^*$ are properly distributed and reveal no information about $\boldsymbol{A}$. $\mathcal{B}$ chooses random values $\gamma, s, x_0, y_0, z_0 \in \mathbb{Z}_p$ and a hash function $\mathcal{H}$ that takes as input attributes set and outputs an element of $\mathbb{Z}_p \setminus \{\gamma, s, x_0, y_0, z_0\}$. $\mathcal{A}$ is given the public key

$$\mathsf{pk} = \begin{pmatrix} \mathit{\Gamma}, g_1^{\boldsymbol{d}_3}, g_1^{\boldsymbol{d}_2}, g_1^{\boldsymbol{d}_2 s}, \cdots, g_1^{\boldsymbol{d}_2 s^Q}, g_2^{\boldsymbol{d}_1^*}, g_2^{\boldsymbol{d}_1^* s}, \cdots, g_2^{\boldsymbol{d}_1^* s^Q}, g_2^{\boldsymbol{d}_2^* \gamma}, \\ g_2^{\boldsymbol{d}_2^* s}, \cdots, g_2^{\boldsymbol{d}_2^* s^Q}, g_2^{\boldsymbol{d}_3^*}, g_2^{\boldsymbol{d}_3^* s}, \cdots, g_2^{\boldsymbol{d}_3^* s^Q}, \mathcal{H}, x_0, y_0, z_0 \end{pmatrix}$$

The master key is $\mathsf{msk} = \left(\gamma, s, g_2^{\boldsymbol{d}_2^*}, \left\{g_1^{\boldsymbol{d}_1 s^i}\right\}_{i=0}^{Q}, \left\{g_1^{\boldsymbol{d}_3 s^i}\right\}_{i=1}^{Q}\right)$.

`KEY QUERY`: $\mathcal{B}$ knows msk and $g_1^{\boldsymbol{d}_5}, g_1^{\boldsymbol{d}_6}$, thus can easily call the key generation algorithm or produce semi-functional keys. It allows $\mathcal{B}$ to answer to all $\mathcal{A}$'s key queries.

– To answer the first j-1 key queries that $\mathcal{A}$ makes, $\mathcal{B}$ runs the semi-functional key generation algorithm to produce semi-functional keys.
– To answer to the j-th key query for Υ^j, $\mathcal{B}$ responds with:

$$\mathsf{sk}_{\Upsilon^j} = g_1^{\boldsymbol{b}_1 \sum_{i=0}^{Q} a_i s^i} \cdot t_2 \cdot t_3^{-\gamma}$$

where $\{a_i\}_{i=0}^{Q}$ are the coefficients of polynomial $\mathsf{Ch}_{\mathcal{X}}[Z]$ and $\mathcal{X} = \left\{\mathcal{H}(p_i^j)\right\}_{i=1}^{2^k-1} \cup \{x_0, z_0\}$, $k \in \mathbb{N}$ is the size of Υ^j and $\left\{p_i^j\right\}_{i=1}^{2^k-1}$ are all the non-empty parties of Υ^j.

• If $t_1, t_2, t_3 = g_1^{\tau_1 \boldsymbol{b}_1}, g_1^{\tau_1 \boldsymbol{b}_2}, g_1^{\tau_1 \boldsymbol{b}_3}$, then sk_{Υ^j} is a normal key with randomness $r = \tau_1$: $\mathsf{sk}_{\Upsilon^j} = g_1^{\boldsymbol{d}_1 \sum_{i=0}^{Q} a_i s^i + \tau_1(\boldsymbol{d}_2 - \gamma \boldsymbol{d}_3)}$. Thus $\mathcal{B}$ has properly simulated Game_{j-1}.
• If $t_1, t_2, t_3 = g_1^{\tau_1 \boldsymbol{b}_1 + \tau_2 \boldsymbol{b}_3}$, $g_1^{\tau_1 \boldsymbol{b}_2 + \tau_2 \boldsymbol{b}_4}$, $g_1^{\tau_1 \boldsymbol{b}_3 + \tau_2 \boldsymbol{b}_6}$, then we have that sk_{Υ^j} is equal to $g_1^{\boldsymbol{d}_1 \sum_{i=0}^{Q} a_i s^i + \tau_1(\boldsymbol{d}_2 - \gamma \boldsymbol{d}_3) + \tau_2(\boldsymbol{b}_4 - \gamma \boldsymbol{b}_6)}$.

– For the remaining key queries, $\mathcal{B}$ runs the normal key generation algorithm.

`CHALLENGE`: At some point, $\mathcal{A}$ sends to $\mathcal{B}$ two challenge messages $\mathsf{m}_0, \mathsf{m}_1$ and a challenge policy $\Pi^* = \pi_1^* \vee \cdots \vee \pi_l^*$. $\mathcal{B}$ chooses a random bit $b \in \{0,1\}$ and encrypts m_b under Π^* as follows: $z, \alpha_1, \alpha_2 \leftarrow \mathbb{Z}_p$ and

$$\begin{aligned}
\mathsf{accp}_{\mathcal{Y}} &= g_2^{\boldsymbol{b}_3^* \sum_{i=0}^{Q} m_i s^i} & \boldsymbol{H} &= e(g_1^{\boldsymbol{b}_3}, \mathsf{accp}_{\mathcal{Y}})^{\alpha_1 \alpha_2} \\
\mathsf{aux}_1 &= g_1^{\alpha_2 \boldsymbol{b}_3 (x_0 + y_0)} & \mathsf{aux}_2 &= g_2^{-\boldsymbol{b}_1^* \alpha_1 \alpha_2} \cdot u_2^{\gamma} \cdot u_3 \\
ele_1 &= g_2^{\alpha_1 \boldsymbol{b}_3^* (z_0 s + s^2)} & ele_2 &= g_2^{\alpha_1 \boldsymbol{b}_3^* (z_0 + s)} \\
ele_3 &= g_1^{\alpha_1 \alpha_2 \boldsymbol{b}_2 (z_0 s + s^2)} & ele_4 &= g_1^{\alpha_1 \alpha_2 \boldsymbol{b}_2 (z_0 + s)}
\end{aligned}$$

which is equal to

$$\begin{aligned}
\mathsf{accp}_{\mathcal{Y}} &= g_2^{\boldsymbol{d}_3^* \sum_{i=0}^{Q} m_i s^i} & \boldsymbol{H} &= e(g_1^{\boldsymbol{d}_3}, \mathsf{accp}_{\mathcal{Y}})^{\alpha_1 \alpha_2} \\
\mathsf{aux}_1 &= g_1^{\alpha_2 \boldsymbol{d}_3 (x_0 + y_0)} & \mathsf{aux}_2 &= g_2^{-\boldsymbol{d}_1^* \alpha_1 \alpha_2} \cdot u_2^{\gamma} \cdot u_3 \\
ele_1 &= g_2^{\alpha_1 \boldsymbol{d}_3^* (z_0 s + s^2)} & ele_2 &= g_2^{\alpha_1 \boldsymbol{d}_3^* (z_0 + s)} \\
ele_3 &= g_1^{\alpha_1 \alpha_2 \boldsymbol{d}_2 (z_0 s + s^2)} & ele_4 &= g_1^{\alpha_1 \alpha_2 \boldsymbol{d}_2 (z_0 + s)}
\end{aligned}$$

where $\mathcal{Y} = \{\mathcal{H}(\mathcal{Y}_i)\}_{i=1}^{l} \cup \{y_0, z_0\}$, and $\mathcal{Y}_i$ for $i = 1, \cdots, l$ is a set that contains the elements of the clause π_i^*.

Suppose that $\mathcal{B}$ decides not to be honest, and find the nature of the j-th key by herself. To do so, she creates a SF ciphertext for a policy $\mathit{\Pi}$ such that Υ^j satisfies $\mathit{\Pi}$. She tries to decrypt it with sk_{Υ^j} to learn if sk_{Υ^j} is a normal or a SF key (a normal key will decrypt correctly while a SF key will with high probability fail to decrypt). Let's see that by construction even if sk_{Υ^j} is SF it will decrypt correctly.

Suppose that $t_1, t_2, t_3 = g_1^{\tau_1 \boldsymbol{b}_1 + \tau_2 \boldsymbol{b}_3}, g_1^{\tau_1 \boldsymbol{b}_2 + \tau_2 \boldsymbol{b}_4}, g_1^{\tau_1 \boldsymbol{b}_3 + \tau_2 \boldsymbol{b}_6}$. During decryption, $\mathcal{B}$ computes $e(\mathsf{sk}_{\Upsilon^j}, \mathsf{aux}_2)$ which is equal to

$$e\left(g_1^{\boldsymbol{b}_1 \sum_{i=0}^{Q} a_i s^i + \tau_1 \boldsymbol{b}_2 + \tau_2 \boldsymbol{d}_5 + \gamma(-\tau_1 \boldsymbol{b}_3 - \tau_2 \boldsymbol{d}_6)}, g_2^{-\boldsymbol{b}_1^* \alpha\alpha_2 + \gamma(\mu_1 \boldsymbol{b}_2^* + \mu_2 \boldsymbol{b}_5^*) + \mu_1 \boldsymbol{b}_3^* + \mu_2 \boldsymbol{b}_6^*}\right)$$

This can be decomposed as

$$e\left(g_1^{\boldsymbol{b}_1 \sum_{i=0}^{Q} a_i s^i}, g_2^{-\boldsymbol{b}_1^* \alpha\alpha_2}\right) \cdot e\left(g_1^{\tau_1 \boldsymbol{b}_2}, g_2^{\gamma\mu_1 \boldsymbol{b}_2^*}\right) \cdot e\left(g_1^{\tau_2 \boldsymbol{b}_5}, g_2^{\gamma\mu_2 \boldsymbol{b}_5^*}\right)$$
$$\cdot e\left(g_1^{-\gamma\tau_1 \boldsymbol{b}_3}, g_2^{\mu_1 \boldsymbol{b}_3^*}\right) \cdot e\left(g_1^{-\gamma\tau_2 \boldsymbol{b}_6}, g_2^{\mu_2 \boldsymbol{b}_6^*}\right)$$

thanks to dual pairing vector spaces properties.

As $\mathit{\Pi}$ is satisfied by Υ^j, the first pairing will cancel itself with the rest of the verification equation. And by construction, the four others cancel with each other. Thus, it will decrypt, and $\mathcal{B}$ will have no information about the j-th key's nature.

Note 6. Notice that in order to create an SF ciphertext, $\mathcal{B}$ must use elements u_2 and u_3 of the assumption, as she does not know $g_2^{\boldsymbol{d}_5^*}$ and $g_2^{\boldsymbol{d}_6^*}$.

In the authorized case, Υ^j does not satisfy $\mathit{\Pi}^*$. Let us see that when $t_1, t_2, t_3 = g_1^{\tau_1 \boldsymbol{b}_1 + \tau_2 \boldsymbol{b}_3}, g_1^{\tau_1 \boldsymbol{b}_2 + \tau_2 \boldsymbol{b}_4}, g_1^{\tau_1 \boldsymbol{b}_3 + \tau_2 \boldsymbol{b}_6}$, the extra coefficients in bases $(\boldsymbol{b}_5^*, \boldsymbol{b}_6^*)$ of the ciphertext and the extra coefficients in bases $(\boldsymbol{b}_5, \boldsymbol{b}_6)$ of the key are distributed as random vectors in the spans of $(\boldsymbol{d}_5^*, \boldsymbol{d}_6^*)$ and $(\boldsymbol{d}_5, \boldsymbol{d}_6)$ respectively. To express them in bases $(\boldsymbol{d}_5^*, \boldsymbol{d}_6^*)$ and $(\boldsymbol{d}_5, \boldsymbol{d}_6)$ respectively, we multiply them by $\boldsymbol{A}^{-1}$ and $\boldsymbol{A}^{\top}$ respectively. Since the distribution of everything given to $\mathcal{A}$ except for the j-th key and the challenge ciphertext is independent of the random matrix $\boldsymbol{A}$ and Υ^j does not satisfy $\mathit{\Pi}^*$, we can conclude that these coefficients are uniformly random. Thus, $\mathcal{B}$ has properly simulated Game_j in this case.

If $t_1, t_2, t_3 = g_1^{\tau_1 \boldsymbol{b}_1}, g_1^{\tau_1 \boldsymbol{b}_2}, g_1^{\tau_1 \boldsymbol{b}_3}$ then the coefficients of the semi-functional part of the ciphertext are uniformly random. Thus, $\mathcal{B}$ has properly simulated Game_{j-1} in this case. Therefore, $\mathcal{B}$ can leverage $\mathcal{A}$'s non-negligible difference in advantage between these games to obtain a non-negligible advantage against DS1. □

Lemma 4. *For any PPT adversary $\mathcal{A}$,* $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Game}_{Final}} \leq \mathsf{Adv}_{\mathcal{A}}^{\mathsf{Game}_{N_q}}$.

We prove this lemma, by randomizing the coefficient of $\boldsymbol{d}_1^*$ in the aux_2 term of the ciphertext, thereby severing its link with the blinding factor.

Proof. We pick $\eta \in \mathbb{Z}_p$ and define new dual orthonormal bases $\mathbb{F} = (\boldsymbol{f}_1, \cdots, \boldsymbol{f}_6)$ and $\mathbb{F}^* = (\boldsymbol{f}_1^*, \cdots, \boldsymbol{f}_6^*)$ as follows:

$$\begin{array}{l} \boldsymbol{f}_1^* = \boldsymbol{d}_1^*, \quad \boldsymbol{f}_2^* = \boldsymbol{d}_2^*,\ \boldsymbol{f}_3^* = \boldsymbol{d}_3^*,\ \boldsymbol{f}_4^* = \boldsymbol{d}_4^*,\ \boldsymbol{f}_5^* = \eta\boldsymbol{d}_1^* + \boldsymbol{d}_5^*,\ \boldsymbol{f}_6^* = \boldsymbol{d}_6^* \\ \boldsymbol{f}_1 = \boldsymbol{d}_1 - \eta\boldsymbol{d}_5,\ \boldsymbol{f}_2 = \boldsymbol{d}_2,\ \boldsymbol{f}_3 = \boldsymbol{d}_3,\ \boldsymbol{f}_4 = \boldsymbol{d}_4, \quad \boldsymbol{f}_5 = \boldsymbol{d}_5, \quad \boldsymbol{f}_6 = \boldsymbol{d}_6 \end{array}$$

It is easy to see that $\mathbb{F}$ and $\mathbb{F}^*$ are also dual orthonormal, and are distributed the same as $\mathbb{D}$ and $\mathbb{D}^*$.

Then, the public key, challenge ciphertext, and queried secret keys in Game_{N^q} are expressed over bases $\mathbb{D}$ and $\mathbb{D}^*$:

$$\mathsf{pk} = \left(\begin{array}{c} \mathit{\Gamma}, g_1^{\boldsymbol{d}_3}, g_1^{\boldsymbol{d}_2}, g_1^{\boldsymbol{d}_2 s}, \cdots, g_1^{\boldsymbol{d}_2 s^Q}, g_2^{\boldsymbol{d}_1^*}, g_2^{\boldsymbol{d}_1^* s}, \cdots, g_2^{\boldsymbol{d}_1^* s^Q}, g_2^{\boldsymbol{d}_2^* \gamma}, \\ g_2^{\boldsymbol{d}_2^* s}, \cdots, g_2^{\boldsymbol{d}_2^* s^Q}, g_2^{\boldsymbol{d}_3^*}, g_2^{\boldsymbol{d}_3^* s}, \cdots, g_2^{\boldsymbol{d}_3^* s^Q}, \mathcal{H}, x_0, y_0, z_0 \end{array}\right)$$

$$\mathsf{ct}_{\mathit{\Pi}} = \left(\begin{array}{cc} \mathsf{accp}_{\mathcal{Y}} = g_2^{\boldsymbol{d}_3^* \sum_{i=0}^{Q} m_i s^i} & \boldsymbol{H} = e(g_1^{\boldsymbol{d}_3}, g_2^{\boldsymbol{d}_3^* \sum_{i=0}^{Q} m_i s^i})^{\alpha_1\alpha_2} \\ \mathsf{aux}_1 = g_1^{\alpha_2 \boldsymbol{d}_3 (x_0 + y_0)} & \mathsf{aux}_2 = g_2^{-\boldsymbol{d}_1^* \alpha_1\alpha_2 + z(\gamma\boldsymbol{d}_2^* + \boldsymbol{d}_3^*) + z_5\boldsymbol{d}_5^* + z_6\boldsymbol{d}_6^*} \\ ele_1 = g_2^{\alpha_1 \boldsymbol{d}_3^* (z_0 s + s^2)} & ele_2 = g_2^{\alpha_1 \boldsymbol{d}_3^* (z_0 + s)} \\ ele_3 = g_1^{\alpha_1\alpha_2 \boldsymbol{d}_2 (z_0 s + s^2)} & ele_4 = g_1^{\alpha_1\alpha_2 \boldsymbol{d}_2 (z_0 + s)} \end{array}\right)$$

$$\{\mathsf{sk}_{\Upsilon^j}\}_{j\in[N_q]} = \left\{ \mathsf{acc}_{\mathcal{X}} \cdot g_1^{t_5^j \boldsymbol{d}_5 + t_6^j \boldsymbol{d}_6} = g_1^{\boldsymbol{d}_1 \sum_{i=0}^{Q} a_i^j s^i + r^j(\boldsymbol{d}_2 - \gamma\boldsymbol{d}_3) + t_5^j \boldsymbol{d}_5 + t_6^j \boldsymbol{d}_6} \right\}_{j\in[N_q]}$$

Then we can express them over bases $\mathbb{F}, \mathbb{F}^*$ as:

$$\mathsf{pk} = \left(\begin{array}{c} \mathit{\Gamma}, g_1^{\boldsymbol{f}_3}, g_1^{\boldsymbol{f}_2}, g_1^{\boldsymbol{f}_2 s}, \cdots, g_1^{\boldsymbol{f}_2 s^Q}, g_2^{\boldsymbol{f}_1^*}, g_2^{\boldsymbol{f}_1^* s}, \cdots, g_2^{\boldsymbol{f}_1^* s^Q}, g_2^{\boldsymbol{f}_2^* \gamma}, \\ g_2^{\boldsymbol{f}_2^* s}, \cdots, g_2^{\boldsymbol{f}_2^* s^Q}, g_2^{\boldsymbol{f}_3^*}, g_2^{\boldsymbol{f}_3^* s}, \cdots, g_2^{\boldsymbol{f}_3^* s^Q}, \mathcal{H}, x_0, y_0, z_0 \end{array}\right)$$

$$\mathsf{ct}_{\mathit{\Pi}} = \left(\begin{array}{cc} \mathsf{accp}_{\mathcal{Y}} = g_2^{\boldsymbol{f}_3^* \sum_{i=0}^{Q} m_i s^i} & \boldsymbol{H} = e(g_1^{\boldsymbol{f}_3}, g_2^{\boldsymbol{f}_3^* \sum_{i=0}^{Q} m_i s^i})^{\alpha_1\alpha_2} \\ \mathsf{aux}_1 = g_1^{\alpha_2 \boldsymbol{f}_3 (x_0 + y_0)} & \mathsf{aux}_2 = g_2^{-\boldsymbol{f}_1^* \alpha' + z(\gamma\boldsymbol{f}_2^* + \boldsymbol{f}_3^*) + z_5\boldsymbol{f}_5^* + z_6\boldsymbol{f}_6^*} \\ ele_1 = g_2^{\alpha_1 \boldsymbol{f}_3^* (z_0 s + s^2)} & ele_2 = g_2^{\alpha_1 \boldsymbol{f}_3^* (z_0 + s)} \\ ele_3 = g_1^{\alpha_1\alpha_2 \boldsymbol{f}_2 (z_0 s + s^2)} & ele_4 = g_1^{\alpha_1\alpha_2 \boldsymbol{f}_2 (z_0 + s)} \end{array}\right)$$

$$\{\mathsf{sk}_{\Upsilon^j}\}_{j\in[N_q]} = \left\{ \mathsf{acc}_{\mathcal{X}} \cdot g_1^{t_5^{j'} \boldsymbol{f}_5 + t_6^j \boldsymbol{f}_6} = g_1^{\boldsymbol{f}_1 \sum_{i=0}^{Q} a_i^j s^i + r^j(\boldsymbol{d}_2 - \gamma\boldsymbol{d}_3) + t_5^{j'} \boldsymbol{f}_5 + t_6^j \boldsymbol{f}_6} \right\}_{j\in[N_q]}$$

where $\alpha' = \alpha_1\alpha_2 - z_5\eta$, $\left\{ t_5^{'j} = t_5^j + \eta \sum_{i=0}^{Q} a_i^j s^i \right\}_{j\in[N_q]}$, which are all uniformly distributed.

In other words, the coefficient $\alpha_1\alpha_2$ of $\boldsymbol{d}_1^*$ in the aux_2 term of the challenge ciphertext is changed to random coefficient $\alpha^{'} \in \mathbb{Z}_p$ of $\boldsymbol{f}_1^*$, thus the challenge ciphertext can be viewed as a semi-functional encryption of a random message in $\mathbb{G}_T$. Moreover, the coefficients $\left\{t_5^{j'}\right\}_{j\in[N_q]}$ of $\boldsymbol{f}_5$ in the $\left\{\mathsf{sk}_{\Upsilon^j}^{(SF)}\right\}_{j\in[N_q]}$ are uniformly distributed since $\left\{t_5^j\right\}$ of $\boldsymbol{d}_5$ are all independent random values. Thus $\left(\mathsf{pk}, \mathsf{ct}_{\Pi}^{(SF)}, \left\{\mathsf{sk}_{\Upsilon^j}^{(SF)}\right\}_{i\in[N_q]}\right)$ expressed over bases $\mathbb{F}$ and $\mathbb{F}^*$ is properly distributed as $\left(\mathsf{pk}, \mathsf{ct}_{\Pi_R}^{(SF)}, \left\{\mathsf{sk}_{\Upsilon^j}^{(SF)}\right\}_{i\in[N_q]}\right)$ in Game_{Final}.

In the adversary's view, both $(\mathbb{D}, \mathbb{D}^*)$ and $(\mathbb{F}, \mathbb{F}^*)$ are consistent with the same public parameters. Therefore, the challenge ciphertext and queried secret keys above can be expressed as keys and ciphertext in two ways, in Game_{N_q} over bases $(\mathbb{D}, \mathbb{D}^*)$ and in Game_{Final} over bases $(\mathbb{F}, \mathbb{F}^*)$. Thus, Game_Q and Game_{Final} are statistically indistinguishable. $\square$

Lemma 5. *For any adversary $\mathcal{A}$, $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Game}_{Final}}(\lambda) = 0$.*

Proof. The value of β is independent of the adversary's view in Game_{Final}. Hence, $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Game}_{Final}}(\lambda) = 0$. $\square$

Combining Lemmas 2 to 5 we prove Theorem 5. $\square$

5.4 Comparison

It is known that monotone boolean formulas can be put under DNF form, where the latter represents the minterm of the formula, *i.e.* a minimal set of variables which, if assigned the value 1, forces the formula to take the value 1 regardless of the values assigned to the remaining variables [15]. For more details on the transformation of monotone boolean formulas into DNF and its probable efficiency loss we refer the interested reader to [7,36]. It is also known that the circuit complexity class monotone NC^1 is captured by monotone boolean formulas of log-depth and fan-in two [23]. Therefore, our CP-ABE can directly deal with monotone NC^1 circuits. We present in Table 3 a comparison of (bounded) CP-ABE scheme for monotone NC^1 circuits, based on pairings[5]. All schemes in this table overpass the one-use restriction on attributes, which imposes that each attribute is only present once in the access policy. All schemes are single authority, and secure in the standard model.

As we can see our scheme is the first one to obtain constant size for both ciphertexts and secret keys. However, this is done at the cost of the public key size, which become exponential. This drawback comes from the fact that for accumulating user's attributes set we are running the hash function $\mathcal{H}$ on each non-empty subset of this set. Doing so we obtain an easy way to check if an attributes set verifies an access policy: if it

[5] Some works are expressing their monotone boolean formula through Linear Secret Sharing Scheme (LSSS) matrix, see [26] for more details on this transformation.

Table 3. Comparison of CP-ABE schemes for monotone NC^1 circuits, based on pairings. Here q is the bound on the number of attributes in the scheme, and l is the number of rows in the access matrix when the policy is expressed with LSSS matrix.

Schemes	$\lvert\mathsf{pk}\rvert$	$\lvert\mathsf{ct}\rvert$	$\lvert\mathsf{sk}\rvert$	Adaptive Security	Assumption	Group Order	Pairing
[39]	$O(q)$	$O(l)$	$O(q)$	$\times$	Non Static	Prime	Symmetric
[24]	$O(q)$	$O(l)$	$O(q)$	$\surd$	Static	Composite	Symmetric
[27]	$O(q)$	$O(l)$	$O(q)$	$\surd$	Non Static	Prime	Symmetric
[23]	$O(q)$	$O(q)$	$O(l)$	$\surd$	Static	Prime	Asymmetric
Our	$O(2^q)$	$O(1)$	$O(1)$	$\surd$	Static	Prime	Asymmetric

does, one of non-empty subsets of the set is equal to one clause of the access policy. We argue that the size of the public key is less important than the size of the other parameters, as it can easily be stored on-line. Additionally, while the sets (and access policies) representation might be scary at first glance, this is not an issue in practice as (i) it is not necessary to keep all elements in memory and (ii) for each decryption, only the useful part will have to be computed again. Finding another way to accumulate attributes sets and access policies in order to have efficient membership verification may lead to a more efficient CP-ABE, with shorter public key size. Also notice that our scheme is dealing with DNF access policies which have small expressiveness. We leave as an open problem to reduce the size of the public key in our scheme and also to modify it so that it can deal with fine-grained access policies. We also leave as an open problem the case of unbounded ABE schemes [3,11], and the case of non-monotonic access formulas [31,32], even if we give some intuitions about it in Appendix D. In Appendix C, we also show how the above construction can be transformed into a Key Policy ABE (KP-ABE), in which the secret key is attached to the access policy and the ciphertext is given by a set of attributes.

6 Conclusion

In this work, we improved the state of the art of cryptographic accumulator schemes by proposing a new scheme that has private evaluation while having public generation. This scheme is the first (as far as we know) accumulator that uses dual pairing vector spaces. We also introduced the new notion of *dually computable* cryptographic accumulators, allowing two ways to evaluate an accumulator: either privately or publicly. We instantiate a dually computable accumulator for our first scheme. Furthermore, we built a new CP-ABE scheme, that deals with monotone NC^1 circuits. This is the first scheme in the literature that has both constant size ciphertexts and users secret keys. We achieve such compactness by using cryptographic accumulators for both key management and encryption. Unfortunately, as our construction strongly relies on the fact that Nguyen's accumulator uses polynomial representation of sets, we cannot generalized our idea. Hence, we leave as an open problem the way to generically transform a cryptographic accumulator into an (attribute-based) encryption scheme.

Acknowledgments We would like to thank anonymous reviewers for their helpful discussions and valuable comments. Part of this work has received funding from the French National Research Agency (ANR), PRESTO project number ANR-19-CE39-0011-01.

References

1. Acar, T., Chow, S.S.M., Nguyen, L.: Accumulators and U-prove revocation. In: Sadeghi, A.R. (ed.) FC 2013. LNCS, vol. 7859, pp. 189–196. Springer, Heidelberg (Apr 2013). https://doi.org/10.1007/978-3-642-39884-1_15
2. Asano, T.: A revocation scheme with minimal storage at receivers. In: Zheng, Y. (ed.) ASIACRYPT 2002. LNCS, vol. 2501, pp. 433–450. Springer, Heidelberg (Dec 2002). https://doi.org/10.1007/3-540-36178-2_27
3. Attrapadung, N.: Dual system encryption framework in prime-order groups via computational pair encodings. In: Cheon, J.H., Takagi, T. (eds.) ASIACRYPT 2016, Part II. LNCS, vol. 10032, pp. 591–623. Springer, Heidelberg (Dec 2016). https://doi.org/10.1007/978-3-662-53890-6_20
4. Attrapadung, N., Libert, B., de Panafieu, E.: Expressive key-policy attribute-based encryption with constant-size ciphertexts. In: Catalano, D., Fazio, N., Gennaro, R., Nicolosi, A. (eds.) PKC 2011. LNCS, vol. 6571, pp. 90–108. Springer, Heidelberg (Mar 2011). https://doi.org/10.1007/978-3-642-19379-8_6
5. Au, M.H., Wu, Q., Susilo, W., Mu, Y.: Compact e-cash from bounded accumulator. In: Abe, M. (ed.) CT-RSA 2007. LNCS, vol. 4377, pp. 178–195. Springer, Heidelberg (Feb 2007). https://doi.org/10.1007/11967668_12
6. Benaloh, J.C., de Mare, M.: One-way accumulators: A decentralized alternative to digital sinatures (extended abstract). In: Helleseth, T. (ed.) EUROCRYPT'93. LNCS, vol. 765, pp. 274–285. Springer, Heidelberg (May 1994). https://doi.org/10.1007/3-540-48285-7_24
7. Blais, E., Håstad, J., Servedio, R.A., Tan, L.Y.: On DNF approximators for monotone Boolean functions. In: Esparza, J., Fraigniaud, P., Husfeldt, T., Koutsoupias, E. (eds.) ICALP 2014, Part I. LNCS, vol. 8572, pp. 235–246. Springer, Heidelberg (Jul 2014). https://doi.org/10.1007/978-3-662-43948-7_20
8. Boneh, D., Boyen, X.: Short signatures without random oracles and the SDH assumption in bilinear groups. Journal of Cryptology **21**(2), 149–177 (Apr 2008). https://doi.org/10.1007/s00145-007-9005-7
9. Boneh, D., Gentry, C., Gorbunov, S., Halevi, S., Nikolaenko, V., Segev, G., Vaikuntanathan, V., Vinayagamurthy, D.: Fully key-homomorphic encryption, arithmetic circuit ABE and compact garbled circuits. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 533–556. Springer, Heidelberg (May 2014). https://doi.org/10.1007/978-3-642-55220-5_30
10. Camenisch, J., Kohlweiss, M., Soriente, C.: An accumulator based on bilinear maps and efficient revocation for anonymous credentials. In: Jarecki, S., Tsudik, G. (eds.) PKC 2009. LNCS, vol. 5443, pp. 481–500. Springer, Heidelberg (Mar 2009). https://doi.org/10.1007/978-3-642-00468-1_27
11. Chen, J., Gong, J., Kowalczyk, L., Wee, H.: Unbounded ABE via bilinear entropy expansion, revisited. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018, Part I. LNCS, vol. 10820, pp. 503–534. Springer, Heidelberg (Apr / May 2018). https://doi.org/10.1007/978-3-319-78381-9_19
12. Chen, J., Lim, H.W., Ling, S., Wang, H., Wee, H.: Shorter IBE and signatures via asymmetric pairings. In: Abdalla, M., Lange, T. (eds.) PAIRING 2012. LNCS,

vol. 7708, pp. 122–140. Springer, Heidelberg (May 2013). https://doi.org/10.1007/978-3-642-36334-4_8

13. Damgard, I., Triandopoulos, N.: Supporting non-membership proofs with bilinear-map accumulators. Cryptology ePrint Archive, Report 2008/538 (2008), http://eprint.iacr.org/2008/538
14. Derler, D., Hanser, C., Slamanig, D.: Revisiting cryptographic accumulators, additional properties and relations to other primitives. In: Nyberg, K. (ed.) CT-RSA 2015. LNCS, vol. 9048, pp. 127–144. Springer, Heidelberg (Apr 2015). https://doi.org/10.1007/978-3-319-16715-2_7
15. Elbassioni, K., Makino, K., Rauf, I.: On the readability of monotone boolean formulae. Journal of Combinatorial Optimization p. 293–304 (2011)
16. Fazio, N., Nicolosi, A.: Cryptographic accumulators: Definitions, constructions and applications (2002)
17. Freedman, M.J., Nissim, K., Pinkas, B.: Efficient private matching and set intersection. In: Cachin, C., Camenisch, J. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 1–19. Springer, Heidelberg (May 2004). https://doi.org/10.1007/978-3-540-24676-3_1
18. Gentry, C., Ramzan, Z.: RSA accumulator based broadcast encryption. In: Zhang, K., Zheng, Y. (eds.) ISC 2004. LNCS, vol. 3225, pp. 73–86. Springer, Heidelberg (Sep 2004)
19. Ghosh, E., Ohrimenko, O., Papadopoulos, D., Tamassia, R., Triandopoulos, N.: Zero-knowledge accumulators and set algebra. In: Cheon, J.H., Takagi, T. (eds.) ASIACRYPT 2016, Part II. LNCS, vol. 10032, pp. 67–100. Springer, Heidelberg (Dec 2016). https://doi.org/10.1007/978-3-662-53890-6_3
20. Goyal, V., Jain, A., Pandey, O., Sahai, A.: Bounded ciphertext policy attribute based encryption. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds.) ICALP 2008, Part II. LNCS, vol. 5126, pp. 579–591. Springer, Heidelberg (Jul 2008). https://doi.org/10.1007/978-3-540-70583-3_47
21. 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., De Capitani di Vimercati, S. (eds.) ACM CCS 2006. pp. 89–98. ACM Press (Oct / Nov 2006). https://doi.org/10.1145/1180405.1180418, available as Cryptology ePrint Archive Report 2006/309
22. Groth, J.: On the size of pairing-based non-interactive arguments. In: Fischlin, M., Coron, J.S. (eds.) EUROCRYPT 2016, Part II. LNCS, vol. 9666, pp. 305–326. Springer, Heidelberg (May 2016). https://doi.org/10.1007/978-3-662-49896-5_11
23. Kowalczyk, L., Wee, H.: Compact adaptively secure ABE for NC^1 from k-Lin. In: Ishai, Y., Rijmen, V. (eds.) EUROCRYPT 2019, Part I. LNCS, vol. 11476, pp. 3–33. Springer, Heidelberg (May 2019). https://doi.org/10.1007/978-3-030-17653-2_1
24. 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 / Jun 2010). https://doi.org/10.1007/978-3-642-13190-5_4
25. Lewko, A.B., Sahai, A., Waters, B.: Revocation systems with very small private keys. In: 2010 IEEE Symposium on Security and Privacy. pp. 273–285. IEEE Computer Society Press (May 2010). https://doi.org/10.1109/SP.2010.23
26. Lewko, A.B., Waters, B.: Decentralizing attribute-based encryption. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 568–588. Springer, Heidelberg (May 2011). https://doi.org/10.1007/978-3-642-20465-4_31
27. Lewko, A.B., Waters, B.: New proof methods for attribute-based encryption: Achieving full security through selective techniques. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 180–198. Springer, Heidelberg (Aug 2012). https://doi.org/10.1007/978-3-642-32009-5_12

28. Libert, B., Ramanna, S.C., Yung, M.: Functional commitment schemes: From polynomial commitments to pairing-based accumulators from simple assumptions. In: Chatzigiannakis, I., Mitzenmacher, M., Rabani, Y., Sangiorgi, D. (eds.) ICALP 2016. LIPIcs, vol. 55, pp. 30:1–30:14. Schloss Dagstuhl (Jul 2016). `https://doi.org/10.4230/LIPIcs.ICALP.2016.30`
29. Mahabir, J., Reihaneh, S.N.: Compact accumulator using lattices. International Conference on Security, Privacy, and Applied Cryptography Engineering (2015)
30. Nguyen, L.: Accumulators from bilinear pairings and applications. In: Menezes, A. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 275–292. Springer, Heidelberg (Feb 2005). `https://doi.org/10.1007/978-3-540-30574-3_19`
31. Okamoto, T., Takashima, K.: Fully secure functional encryption with general relations from the decisional linear assumption. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 191–208. Springer, Heidelberg (Aug 2010). `https://doi.org/10.1007/978-3-642-14623-7_11`
32. Okamoto, T., Takashima, K.: Fully secure unbounded inner-product and attribute-based encryption. In: Wang, X., Sako, K. (eds.) ASIACRYPT 2012. LNCS, vol. 7658, pp. 349–366. Springer, Heidelberg (Dec 2012). `https://doi.org/10.1007/978-3-642-34961-4_22`
33. Papamanthou, C., Tamassia, R., Triandopoulos, N.: Optimal verification of operations on dynamic sets. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 91–110. Springer, Heidelberg (Aug 2011). `https://doi.org/10.1007/978-3-642-22792-9_6`
34. Sahai, A., Waters, B.R.: Fuzzy identity-based encryption. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 457–473. Springer, Heidelberg (May 2005). `https://doi.org/10.1007/11426639_27`
35. Tanaka, N., Saito, T.: On the q-Strong Diffie-Hellman problem. IACR Cryptol. ePrint Arch. **2010**, 215 (2010)
36. Venema, M., Alpár, G., Hoepman, J.H.: Systematizing core properties of pairing-based attribute-based encryption to uncover remaining challenges in enforcing access control in practice. Designs CoDes and Cryptography **91**(1), 165–220 (Jan 2023). `https://doi.org/10.1007/s10623-022-01093-5`
37. Wang, X., Chow, S.S.M.: Cross-domain access control encryption: Arbitrary-policy, constant-size, efficient. In: 2021 IEEE Symposium on Security and Privacy. pp. 748–761. IEEE Computer Society Press (May 2021). `https://doi.org/10.1109/SP40001.2021.00023`
38. 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 (Aug 2009). `https://doi.org/10.1007/978-3-642-03356-8_36`
39. Waters, B.: Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization. In: Catalano, D., Fazio, N., Gennaro, R., Nicolosi, A. (eds.) PKC 2011. LNCS, vol. 6571, pp. 53–70. Springer, Heidelberg (Mar 2011). `https://doi.org/10.1007/978-3-642-19379-8_4`
40. Zhang, Y., Katz, J., Papamanthou, C.: An expressive (zero-knowledge) set accumulator. pp. 158–173 (04 2017). `https://doi.org/10.1109/EuroSP.2017.35`

Supplementary material

A More Details on Our CP-ABE Construction

A.1 Intuition of our CP-ABE Construction

Our idea is to use our dually computable accumulator to construct a CP-ABE. Let $q \in \mathbb{N}$ be the bound on the number of attributes in our scheme and $Q = 2^q - 1$. Let $\varUpsilon$ be the user attributes set of size $k \in \mathbb{N}$ with $k \leq q$, $p_1, \cdots p_{2^k-1}$ be all non-empty parties of $\varUpsilon$. Set $\mathcal{X} = \{\mathcal{H}(p_i)\}_{i=1}^{2^k-1}$ and let $\{a_i\}_{i=0}^{q}$ be the coefficients of the polynomial $\mathsf{Ch}_{\mathcal{X}}[Z] = \prod_{i=0}^{2^k-1}(\mathcal{H}(p_i) + Z)$. Let $\varPi = \pi_1 \vee \pi_2 \vee \cdots \vee \pi_l$ be the access policy composed of $l \in \mathbb{N}$ clauses, $\mathcal{Y}_i$ be the set of elements present in clause π_i for $i = 1, \cdots l$. Set $\mathcal{Y} = \cup_{i=1}^{l}\mathcal{H}(\mathcal{Y}_i)$ and let $\{m_i\}_{i=0}^{q}$ be the coefficients of the polynomial $\mathsf{Ch}_{\mathcal{Y}}[Z] = \prod_{i=0}^{l}(\mathcal{H}(\mathcal{Y}_i) + Z)$. Our scheme works as follows:

- KeyGen: the user secret key is a privately computed accumulator of the set $\mathcal{X}$: $\mathsf{acc}_{\mathcal{X}} = g_1^{\boldsymbol{d}_1 \sum_{i=0}^{Q} a_i s^i}$.
- Encrypt: the message is hidden with $e(g_1^{\boldsymbol{d}_1}, g_2^{\boldsymbol{d}_1^* \sum_{i=0}^{Q} m_i s^i})^{\alpha}$ where $g_2^{\boldsymbol{d}_1^* \sum_{i=0}^{Q} m_i s^i} = \mathsf{accp}_{\mathcal{Y}}$ a publicly computed accumulator of the access policy set $\mathcal{Y}$ and α a random.
- Decrypt: we need to be able to reconstruct $\psi\alpha \sum_{i=0}^{Q} m_i s^i$ in exponent of $e(g_1, g_2)$.

Let π_j be the clause in $\varPi$ that is satisfied by $\varUpsilon$, where $j \in [l]$. Let p_{j^*} be the non-empty party of $\varUpsilon$ that satisfies $\varPi$, for $j^* \in [2^k - 1]$. Then $\mathcal{H}(\mathcal{Y}_j) = \mathcal{H}(p_{j^*})$ and we write this element ξ for simplicity.

However, in this construction, this there is nothing that links the ciphertext to the user secret key: anyone can choose to create a witness for one clause of the policy, even if he does not have this clause, as witness creation is public. To avoid this, we need to force user to also prove that the clause is in his accumulator. Let $\mathsf{wit}_{\xi}^{\mathcal{Y}}, \mathsf{wit}_{\xi}^{\mathcal{X}}$ be membership witnesses for ξ and respectively $\mathcal{Y}$, and $\mathcal{X}$. For verification, we compute $A = e(\mathsf{wit}_{\xi}^{\mathcal{Y}}, (g_2^{\boldsymbol{d}_2^*\alpha})^{\xi} \cdot g_2^{\boldsymbol{d}_2^* s\alpha})$ and $B = e(\mathsf{wit}_{\xi}^{\mathcal{X}}, (g_2^{\boldsymbol{d}_2^*\alpha})^{\xi} \cdot g_2^{\boldsymbol{d}_2^* s\alpha})$. Multiplying A and B gives us

$$e(\mathsf{wit}_{\xi}^{\mathcal{Y}} \cdot \mathsf{wit}_{\xi}^{\mathcal{X}}, (g_2^{\boldsymbol{d}_2^*\alpha})^{\xi} \cdot g_2^{\boldsymbol{d}_2^* s\alpha})$$

thanks to bilinear pairing properties. To prove simultaneously that ξ is in both accumulators, we need to "force" the decryptor to compute $e(A \cdot B, (g_2^{\boldsymbol{d}_2^*\alpha})^{\xi} \cdot g_2^{\boldsymbol{d}_2^* s\alpha})$ instead of $e(A, (g_2^{\boldsymbol{d}_2^*\alpha})^{\xi} \cdot g_2^{\boldsymbol{d}_2^* s\alpha}) \cdot (B, (g_2^{\boldsymbol{d}_2^*\alpha})^{\xi} \cdot g_2^{\boldsymbol{d}_2^* s\alpha})$.

An easy way to do that is to give with the ciphertext $(A \cdot B)^{\alpha}$ instead of $g_2^{\boldsymbol{d}_2^*\alpha}, g_2^{\boldsymbol{d}_2^* s\alpha}$. But this implies to know witnesses during encryption whereas they are only known during decryption. Our idea is then to "anticipate" the witnesses or at least a part of them.

From Definition 8 and Note 4 we know that for any set $S = \{s_1, \cdots, s_T\}$, its polynomial representation $\prod_{i=1}^{T}(s + Z)$ is actually composed of the elementary symmetric polynomials for T variables: $\sigma_0 = 1, \sigma_1 = s_1 + s_2 + \cdots s_T, \cdots, \sigma_T = \prod_{i=1}^{T} s_i$. Indeed, $\prod_{i=1}^{T}(s+Z) = \sigma_0 Z^T + \sigma_1 Z^{T-1} + \cdots \sigma_T$. Thus, if we know one element $\tilde{s}$ of S, we know that $\tilde{s}$ is a factor of σ_T. We use this idea to anticipate a part of both witnesses for element ξ.

Let $\{c_i\}_{i=0}^{Q}$ be the coefficients of $\mathsf{Ch}_{\mathcal{X}\setminus\{\xi\}}[Z]$ and $\{t_i\}_{i=0}^{Q}$ are the coefficients of $\mathsf{Ch}_{\mathcal{Y}\setminus\{\xi\}}[Z]$. Our first idea is to separate coefficients c_0 and t_0 of the others. Thus, in our accumulator a witness that $\xi = \mathcal{H}(p_{j^*})$ is accumulated in $\mathsf{acc}_{\mathcal{X}}$ is now equal to $(g_1^{\boldsymbol{d}_1 c_0}, g_2^{\boldsymbol{d}_2^* \sum_{i=1}^{Q} c_i s^i})$ and a witness that $\xi = \mathcal{H}(\mathcal{Y}_j)$ is accumulated in $\mathsf{accp}_{\mathcal{Y}}$ is now equal to $(g_1^{\boldsymbol{d}_1 t_0}, g_2^{\boldsymbol{d}_2^* \sum_{i=1}^{Q} t_i s^i})$. This gives us our first intermediate accumulator, presented in Figure 6 of Appendix B.

But the values c_0 and t_0 depend on $\mathcal{X}\setminus\{\xi\}$ and $\mathcal{Y}\setminus\{\xi\}$ respectively. While $\mathcal{Y}$ is known during encryption, $\mathcal{X}$ and ξ are only known during decryption. Therefore, we cannot anticipate c_0 and t_0 during decryption.

To solve this, we choose two values x_0, y_0 that do not correspond to an output of $\mathcal{H}$ and add them to the sets $\mathcal{X}$ and $\mathcal{Y}$ respectively. As $x_0, y_0 \notin Im(\mathcal{H})$, we know that x_0 and y_0 will always be in sets $\mathcal{X}\setminus\{\xi\}$ and $\mathcal{Y}\setminus\{\xi\}$ respectively. Thus, x_0 is a factor of c_0 while y_0 is a factor of t_0. Therefore, we can anticipate a part of the first element of both witnesses with $g_1^{\boldsymbol{d}_1 x_0}$ for the witness associated to $\mathsf{acc}_{\mathcal{X}}$ and $g_1^{\boldsymbol{d}_1 y_0}$ for the witness associated to $\mathsf{accp}_{\mathcal{Y}}$.

Now that we can anticipate a part of the witnesses, we can combine them by setting $\mathsf{aux}_1 = g_1^{\boldsymbol{d}_1 \alpha(x_0+y_0)}$. Let $\delta, \delta' \in \mathbb{N}$ such that $c_0 = x_0\delta$ and $t_0 = y_0\delta'$. We can compute

$$\begin{aligned}
&e(\mathsf{aux}_1^{\delta\delta_0}, (g_2^{\boldsymbol{d}_1^*})^{\xi} \cdot g_2^{\boldsymbol{d}_1^* s}) \\
&= e((g_1^{\boldsymbol{d}_1 \alpha(x_0+y_0)})^{\delta\delta'}, g_2^{\boldsymbol{d}_1^*(\xi+s)}) \\
&= e(g_1^{\boldsymbol{d}_1 \alpha\delta\delta'(x_0+y_0)}, g_2^{\boldsymbol{d}_1^*(s+\xi)}) \\
&= e(g_1, g_2)^{\psi\alpha\delta' c_0(s+\xi)} \cdot e(g_1, g_2)^{\psi\alpha\delta t_0(s+\xi)}
\end{aligned}$$

As the verification that $\xi \in \mathcal{X}$ will give $e(g_1, g_2)^{\psi\alpha \sum_{i=0}^{Q} a_i s^i}$ (if ξ is indeed in the set) we have to give in encryption the auxiliary information $\mathsf{aux}_2 = g_2^{-\alpha \boldsymbol{d}_1^*}$ to remove this extra term and recover the mask. As we can see aux_2 will work only with privately computed accumulator, and will be used for the verification of membership in the accumulator of the secret key.

We now have to compute the rest of witness such that it is randomized by α. The trivial solution is to give $g_2^{\alpha \boldsymbol{d}_2^* s}, \cdots, g_2^{\alpha \boldsymbol{d}_2^* s^Q}$ but this will result in a linear size for the ciphertext. Thus, it seems more efficient to give $g_1^{\alpha \boldsymbol{d}_2(s+\xi)}$. But as ξ is unknown at the time of encryption, we have to give $g_1^{\alpha \boldsymbol{d}_2 s}$ and $g_1^{\alpha \boldsymbol{d}_2}$. With the latter it is possible to cheat: with $g_2^{\boldsymbol{d}_2^*}, g_2^{\boldsymbol{d}_2^* s}, \cdots g_2^{\boldsymbol{d}_2^* s^Q}$ we can compute $g_2^{\boldsymbol{d}_2^* \sum_{i=0}^{Q} m_i s^i}$ and recover the mask.

Our idea to avoid this is to anticipate the value of ξ. We do as we did to anticipate c_0 and t_0. We choose another value z_0 that is not in $Im(\mathcal{H})$ that we add in $\mathcal{X}$ and $\mathcal{Y}$.

Then z_0 is an element accumulated in both $\mathsf{acc}_{\mathcal{X}}$ (the secret key) and $\mathsf{accp}_{\mathcal{Y}}$ (used in the mask of the message). During decryption, we prove the membership of $\{\xi, z_0\}$, and thus we need the polynomial $Z^2 + Z(\xi + z_0) + \xi \cdot z_0 = Z^2 + Zz_0 + \xi(Z + z_0)$. Therefore, we give with the ciphertext the auxiliary information $ele_3 = g_1^{\alpha \boldsymbol{d}_2(z_0 s + s^2)}$ and $ele_4 = g_1^{\alpha \boldsymbol{d}_2(z_0+s)}$. As s is secret, there is no way to cheat.

Unfortunately, as is, the scheme is not secure. Indeed, from $\mathsf{aux}_1 = g_1^{\boldsymbol{d}_1 \alpha(x_0+y_0)}$ and $\left\{ g_2^{\boldsymbol{d}_1^* s^i} \right\}_{i=0}^{Q}$, anyone can compute $(e(g_1, g_2)^{\psi\alpha \sum_{i=0}^{Q} m_i s^i})^{x_0+y_0}$. As x_0, y_0 are publicly known, anyone can recover $e(g_1, g_2)^{\psi\alpha \sum_{i=0}^{Q} m_i s^i} = \boldsymbol{H}$ and thus the message.

To correct this we set: $\alpha = \alpha_1 \cdot \alpha_2$ for α_1, α_2 two randoms, $\mathsf{aux}_1 = g_1^{\boldsymbol{d}_1 \alpha_2(x_0+y_0)}$, $ele_3 = g_1^{\alpha_1\alpha_2 \boldsymbol{d}_2(z_0 s + s^2)}$, $ele_4 = g_1^{\alpha_1\alpha_2 \boldsymbol{d}_2(z_0+s)}$, and $\mathsf{aux}_2 = g_2^{-\alpha_1\alpha_2 \boldsymbol{d}_1^*}$. To have correctness during membership verification, we need more auxiliary information (ele_1, ele_2) are equal to $(g_2^{\boldsymbol{d}_1^* \alpha_1(z_0 s + s^2)}, g_2^{\boldsymbol{d}_1^* \alpha_1(z_0+s)})$.

Unfortunately, we were not able to prove security of the build CP-ABE. Therefore, we add to modify the underlying accumulator, as we explain in the next subsection.

A.2 Adding Security to Our ABE Scheme

Here we first present the transformation from our first intermediate accumulator to our second intermediate accumulator, then to our last accumulator, which is the one we need to use in our CP-ABE construction.

As already said, we will prove adaptive security of our scheme with the dual system encryption framework, and the decisional subspace assumption in $\mathbb{G}_1$ and $\mathbb{G}_2$, as it relies on the hidden subspaces of dual pairing vector spaces. We recall that our CP-ABE secret key for attributes sets $\varUpsilon$ is equal to $\mathsf{acc}_{\mathcal{X}} = g_1^{\boldsymbol{d}_1 \sum_{i=0}^{Q} a_i s^i}$ and in the ciphertext we provide $\mathsf{aux}_2 = g_2^{-\alpha_1\alpha_2 \boldsymbol{d}_1^*}$, where $\alpha_1, \alpha_2 \leftarrow \mathbb{Z}_p$. We now have to define *semi-functional* keys and ciphertexts, that will be used in the security proof. To do so, we need to double the dimension of DPVS used: we now have $\mathbb{D} = (\boldsymbol{d}_1, \boldsymbol{d}_2, \boldsymbol{d}_3, \boldsymbol{d}_4)$ and $\mathbb{D}^* = (\boldsymbol{d}_1^*, \boldsymbol{d}_2^*, \boldsymbol{d}_3^*, \boldsymbol{d}_4^*)$, where $\boldsymbol{d}_3, \boldsymbol{d}_4, \boldsymbol{d}_3^*, \boldsymbol{d}_4^*$ will be used for semi-functional space. Thus, trivially we can define for a secret key $\mathsf{sk}_{\varUpsilon}$ and ciphertext auxiliary information aux_2 their semi-functional forms as:

$$\mathsf{sk}_{\varUpsilon}^{(SF)} = \mathsf{sk}_{\varUpsilon} \cdot g_1^{\boldsymbol{d}_3 t_3} \text{ and } \mathsf{aux}_2^{(SF)} = \mathsf{aux}_2 \cdot g_2^{\boldsymbol{d}_3^* z_3} \text{ for } t_3, z_3 \leftarrow \mathbb{Z}_p$$

When using the DS2 assumption to change challenge ciphertext from normal form to semi-functional, we will use the element T_1, which is equal either to $g_2^{\tau_1 \boldsymbol{d}_1^*}$ or to $g_2^{\tau \boldsymbol{d}_1^* + \tau_2 \boldsymbol{d}_3^*}$, to build either aux_2 or $\mathsf{aux}_2^{(SF)}$. However, the random τ_1 will have to appear in other parts of the ciphertext as $e(\mathsf{acc}_{\mathcal{X}}, \mathsf{aux}_2) = e(\mathsf{acc}_{\mathcal{X}}, g_2^{\boldsymbol{d}_1^*})^{\tau_1}$ thus for membership verification we have to be able to reconstruct $\tau_1 \sum_{i=0}^{q} a_i s^i$. And as τ_1 is only given in exponent of the assumption's challenge we do not know it and will not be able to use it for other parts of the ciphertext, especially because in the ciphertext there are elements of $\mathbb{G}_1$ and we only have τ_1 as exponent of an element of $\mathbb{G}_2$.

Thus, we have to change the way we define semi-functional keys and ciphertexts. Let us now define normal and semi-functional keys and ciphertexts as follows:

$$\begin{array}{rl} \mathsf{sk}_\Upsilon = g_1^{\boldsymbol{d}_1 \sum_{i=0}^{q} a_i s^i + z_2 \boldsymbol{d}_2} & , \quad \mathsf{aux}_2 = g_2^{r\boldsymbol{d}_1^* + t_2 \boldsymbol{d}_2^*} \\ \mathsf{sk}_\Upsilon^{(SF)} = g_1^{\boldsymbol{d}_1 \sum_{i=0}^{q} a_i s^i + z_2 \boldsymbol{d}_2 + z_4 \boldsymbol{d}_4} & , \mathsf{aux}_2^{(SF)} = g_2^{r\boldsymbol{d}_1^* + t_2 \boldsymbol{d}_2^* + t_4 \boldsymbol{d}_4^*} \end{array}$$

where $z_2, z_4, t_2, t_4 \leftarrow \mathbb{Z}_p$. We easily notice that during membership verification between a normal key and a normal ciphertext, we have an extra term $e(g_1, g_2)^{\psi z_2 t_2}$. To remove this extra term, we can add in the key $g_1^{-\boldsymbol{d}_2 t_2}$ and in the ciphertext $g_2^{\boldsymbol{d}_2^* z_2}$.
But as we add an element to normal keys and one to normal ciphertexts, we have to modify the semi-functional keys and ciphertext by adding them $g_1^{-\boldsymbol{d}_2 t_2 - \boldsymbol{d}_4 t_4}$ and $g_2^{\boldsymbol{d}_2^* z_2 + \boldsymbol{d}_4^* z_4}$ respectively. Notice that we keep the same randoms as coefficients of $\boldsymbol{d}_4$ and $\boldsymbol{d}_4^*$ in both parts of the semi-functional key and ciphertext (as the assumption's challenge gives us only one coefficient for $\boldsymbol{d}_4, \boldsymbol{d}_4^*$ and if we randomized it for the second part of SF keys and ciphertext, again we will not be able to remove the extra term). But doing so we obtain that a semi-functional key always decrypt a semi-functional ciphertext, which should not be possible.

To fix this issue, we can define normal keys and ciphertexts as follows:

$$\mathsf{sk}_\Upsilon = g_1^{\boldsymbol{d}_1 \sum_{i=0}^{q} a_i s^i + (\boldsymbol{d}_1 - \boldsymbol{d}_2)} , \ \mathsf{aux}_2 = g_2^{r\boldsymbol{d}_1^* + (\boldsymbol{d}_1^* + \boldsymbol{d}_2^*)}.$$

With this definition, we obtain in the accumulator verification $e(g_1, g_2)^{\psi\gamma} \cdot e(g_1, g_2)^{-\psi\gamma}$, as we wanted. But we also obtain $e(g_1, g_2)^{\psi\gamma \sum_{i=0}^{q} a_i s^i}$, and extra term we cannot remove.

At this point, our idea is to increase the dimension of the used DPVS of the accumulator to 3 (and thus 6 for the ABE to have semi-functional spaces). Then, we define normal and semi-functional keys and ciphertexts as:

$$\begin{array}{rl} \mathsf{sk}_\Upsilon = g_1^{\boldsymbol{d}_1 \sum_{i=0}^{q} a_i s^i + (\boldsymbol{d}_2 - \boldsymbol{d}_3)} & , \quad \mathsf{aux}_2 = g_2^{r\boldsymbol{d}_1^* + (\boldsymbol{d}_2^* + \boldsymbol{d}_3^*)} \\ \mathsf{sk}_\Upsilon^{(SF)} = g_1^{\boldsymbol{d}_1 \sum_{i=0}^{q} a_i s^i + (\boldsymbol{d}_2 - \boldsymbol{d}_3) + z_5 \boldsymbol{d}_5 + z_6 \boldsymbol{d}_6} & , \mathsf{aux}_2^{(SF)} = g_2^{r\boldsymbol{d}_1^* + (\boldsymbol{d}_2^* + \boldsymbol{d}_3^*) + t_5 \boldsymbol{d}_5^* + t_6 \boldsymbol{d}_6^*} \end{array}$$

where $z_5, z_6, t_5, t_6 \leftarrow \mathbb{Z}_p$. Decryption of a normal ciphertext by a normal or SK key will work as no extra term will be in the result and decryption of a SF ciphertext by a normal key will also work. However, decryption of a SF ciphertext by a SK key will not work as it has an extra term: $e(g_1, g_2)^{\psi(t_5 z_5 + t_6 z_6)}$[6].

Though there is one problem when defining keys and ciphertexts like this. In the security proof, we use the challenge of DS2 assumption T_2, T_3 to build the challenge

[6] This idea is inspired by the IBE of [12].

ciphertext. (T_2, T_3) are either equals to $(g_2^{\tau_1 \boldsymbol{d}_2^*}, g_2^{\tau_1 \boldsymbol{d}_3^*})$ or to $(g_2^{\tau_1 \boldsymbol{d}_2^* + \tau_2 \boldsymbol{d}_5^*}, g_2^{\tau_1 \boldsymbol{d}_3^* + \tau_2 \boldsymbol{d}_6^*})$. We set $\mathsf{aux}_2 = g_2^{-\alpha_1 \alpha_2 \boldsymbol{d}_1^*} \cdot T_2 \cdot T_3$. In the both case, we have that $\boldsymbol{d}_2^*, \boldsymbol{d}_3^*$ are randomized by τ_1. Thus we need to define $\mathsf{aux}_2 = g_2^{r \boldsymbol{d}_1^* + z(\boldsymbol{d}_2^* + \boldsymbol{d}_3^*)}$ for $z \leftarrow \mathbb{Z}_p$. As the same goes when using the challenge of DS1 assumption to build the challenge key, we have to define $\mathsf{sk}_\Upsilon = g_1^{\boldsymbol{d}_1 \sum_{i=0}^q a_i s^i + r(\boldsymbol{d}_2 - \boldsymbol{d}_3)}$ for $r \leftarrow \mathbb{Z}_p$. We carry these modifications in $\mathsf{sk}_\Upsilon^{(SF)}$ and $\mathsf{aux}_2^{(SF)}$.

But notice that with way of building the challenge ciphertext, when T_2, T_3 are equals to $g_2^{\tau_1 \boldsymbol{d}_2^* + \tau_2 \boldsymbol{d}_5^*}, g_2^{\tau_1 \boldsymbol{d}_3^* + \tau_2 \boldsymbol{d}_6^*}$ we have that $t_5 = t_6 = \tau_2$ (and the same goes for the challenge key where $z_5 = z_6 = \tau_2$). Thus, we do not obtain an SF ciphertext (or SF key). To solve this issue we actually randomized $\boldsymbol{d}_3$ in the keys and $\boldsymbol{d}_2^*$ in the ciphertext, with the same random. Therefore, we define normal and SF keys and ciphertext as follows:

$$\begin{array}{rl} \mathsf{sk}_\Upsilon = g_1^{\boldsymbol{d}_1 \sum_{i=0}^q a_i s^i + (\boldsymbol{d}_2 - \gamma \boldsymbol{d}_3)} & , sk_\Upsilon^{(SF)} = g_1^{\boldsymbol{d}_1 \sum_{i=0}^q a_i s^i + (\boldsymbol{d}_2 - \gamma \boldsymbol{d}_3) + z_5 \boldsymbol{d}_5 + z_6 \boldsymbol{d}_6} \\ \mathsf{aux}_2 = g_2^{r \boldsymbol{d}_1^* + (\gamma \boldsymbol{d}_2^* + \boldsymbol{d}_3^*)} & , \quad \mathsf{aux}_2^{(SF)} = g_2^{r \boldsymbol{d}_1^* + (\gamma \boldsymbol{d}_2^* + \boldsymbol{d}_3^*) + t_5 \boldsymbol{d}_5^* + t_6 \boldsymbol{d}_6^*} \end{array}$$

This gives us our second accumulator, presented in Figure 7, in Appendix B.

Finally, to conclude our security proof, we will do a change of bases from $(\mathbb{D}, \mathbb{D}^*)$ to $(\mathbb{F}, \mathbb{F}^*)$. By the way we define it, we obtain $\boldsymbol{f}_1 = \boldsymbol{d}_1 - \eta \boldsymbol{d}_5$ where $\eta \leftarrow \mathbb{Z}_p$. It means that each part of the ciphertext that uses $\boldsymbol{d}_1$ will have a semi-functional part in bases $\mathbb{F}$, and our ciphertext will no longer be a correct SF ciphertext. Indeed, we defined (and we need for the other parts of the proof) a SF ciphertext as being a normal ciphertext with only element aux_2 having a semi-functional part. Therefore, we need to replace $\boldsymbol{d}_1$ by $\boldsymbol{d}_3$ in ciphertexts to avoid this issue. As in our CP-ABE the anticipation of the first element of the membership witness, aux_1 uses $\boldsymbol{d}_1$, we modify our accumulator so that W_1 has in exponent $\boldsymbol{d}_3$. This gives us our last accumulator, presented in Figure 8, in Appendix 6. Plus, as in the CP-ABE ciphertext the mask of the message is $e(g_1^{\boldsymbol{d}_1}, \mathsf{accp}_\mathcal{Y})$, we have to change the way to publicly computed accumulators: we now use $\left\{\boldsymbol{b}_3^* s^i\right\}_{i=0}^q$ instead of $\left\{\boldsymbol{b}_1^* s^i\right\}_{i=0}^q$. We do not include $g_2^{\boldsymbol{d}_2^* \gamma + \boldsymbol{d}_3^*}$ in the publicly computed accumulator we do not need a semi-functional form of it. That gives us our second accumulator scheme, presented in Figure 8. We also change elements ele_1, ele_2 in our CP-ABE ciphertext: we replace $\boldsymbol{d}_1^*$ by $\boldsymbol{d}_3^*$ to keep correctness.

B Intermediates Accumulator Schemes

In this section, we present the intermediates accumulators that helped us move from our first dually computable accumulator, presented in Figure 3, to the dually computable accumulator that we use in our CP-ABE construction.

B.1 First Intermediate Accumulator: Dividing Witnesses

In Figure 6 we present our first intermediate accumulator. It is the first step of upgrading our first accumulator (like explained in Section 5.2 and Appendix A). This accumulator is the same as the accumulator of Figure 3 except that witnesses are now divided in two parts and thus $g_2^{\boldsymbol{d}_2^*}$ is no longer need in the public key.

- $\mathsf{Gen}(1^\kappa, q)$: run a bilinear group generation algorithm to get $\Gamma = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2)$. Then choose a random $s \leftarrow \mathbb{Z}_p^*$, and run $\mathsf{Dual}(\mathbb{Z}_p^2)$ to get $\mathbb{D} = (\boldsymbol{d}_1, \boldsymbol{d}_2), \mathbb{D}^* = (\boldsymbol{d}_1^*, \boldsymbol{d}_2^*)$. Let $\psi \in \mathbb{Z}_p$ be the random such that $\boldsymbol{d}_1 \cdot \boldsymbol{d}_1^* = \boldsymbol{d}_2 \cdot \boldsymbol{d}_2^* = \psi$. Set $\mathsf{sk}_{\mathsf{acc}} = (s, \mathbb{D}, \mathbb{D}^*)$,
$$\mathsf{pk}_{\mathsf{acc}} = \begin{pmatrix} \Gamma, g_1^{\boldsymbol{d}_1}, g_1^{\boldsymbol{d}_2}, g_1^{\boldsymbol{d}_2 s}, \cdots, g_1^{\boldsymbol{d}_2 s^q}, g_2^{\boldsymbol{d}_1^*}, \\ g_2^{\boldsymbol{d}_1^* s}, \cdots, g_2^{\boldsymbol{d}_1^* s^q}, g_2^{\boldsymbol{d}_2^* s}, \cdots, g_2^{\boldsymbol{d}_2^* s^q} \end{pmatrix}.$$
Return $\mathsf{sk}_{\mathsf{acc}}, \mathsf{pk}_{\mathsf{acc}}$.
- $\mathsf{Eval}(\mathsf{sk}_{\mathsf{acc}}, \mathcal{X})$: compute the coefficients $\{a_i\}_{i=0,\cdots,q}$ of the polynomial $\mathsf{Ch}_{\mathcal{X}}[Z] = \prod_{x\in\mathcal{X}}(Z+x)$. Then compute $\mathsf{acc}_{\mathcal{X}} = g_1^{\boldsymbol{d}_1 \sum_{i=0}^q a_i s^i}$, and return $\mathsf{acc}_{\mathcal{X}}$.
- $\mathsf{PublicEval}(\mathsf{pk}_{\mathsf{acc}}, \mathcal{X})$: compute the coefficients $\{a_i\}_{i=0,\cdots,q}$ of the polynomial $\mathsf{Ch}_{\mathcal{X}}[Z] = \prod_{x\in\mathcal{X}}(Z+x)$. Then compute $\mathsf{accp}_{\mathcal{X}} = g_2^{\boldsymbol{d}_1^* \sum_{i=0}^q a_i s^i}$, and return $\mathsf{accp}_{\mathcal{X}}$.
- $\mathsf{WitCreate}(\mathsf{pk}_{\mathsf{acc}}, \mathcal{X}, \mathsf{acc}_{\mathcal{X}}/\mathsf{accp}_{\mathcal{X}}, \mathcal{I})$: let $\{b_i\}_{i=0,\cdots,q}$ be the coefficients of the polynomial $\mathsf{Ch}_{\mathcal{X}\setminus\mathcal{I}}[Z] = \prod_{x\in\mathcal{X}\setminus\mathcal{I}}(x+Z)$. Compute $W_1 = g_1^{\boldsymbol{d}_1 b_0}$ and $W_2 = g_2^{\boldsymbol{d}_2^* \sum_{i=1}^q b_i s^i}$, and return $\mathsf{wit}_{\mathcal{I}} = \mathsf{witp}_{\mathcal{I}} = (W_1, W_2)$.
- $\mathsf{Verify}(\mathsf{pk}_{\mathsf{acc}}, \mathsf{acc}_{\mathcal{X}}, \mathsf{wit}_{\mathcal{I}}, \mathcal{I})$: let $\{c_i\}_{i=0,\cdots,q}$ be the coefficients of the polynomial $\mathsf{Ch}_{\mathcal{I}}[Z] = \prod_{x\in\mathcal{I}}(x+Z)$ and the return 1 if $e(\mathsf{acc}_{\mathcal{X}}, g_2^{\boldsymbol{d}_1^*}) = e(W_1, g_2^{\boldsymbol{d}_1^* \sum_{i=0}^q c_i s^i}) \cdot e(g_1^{\boldsymbol{d}_2 \sum_{i=0}^q c_i s^i}, W_2)$, 0 otherwise.
- $\mathsf{PublicVerify}(\mathsf{pk}_{\mathsf{acc}}, \mathsf{accp}_{\mathcal{X}}, \mathsf{wit}_{\mathcal{I}}, \mathcal{I})$: let $\{c_i\}_{i=0,\cdots,q}$ be the coefficients of the polynomial $\mathsf{Ch}_{\mathcal{I}}[Z] = \prod_{x\in\mathcal{I}}(x+Z)$ and return 1 if $e(g_1^{\boldsymbol{d}_1}, \mathsf{accp}_{\mathcal{X}}) = e(W_1, g_2^{\boldsymbol{d}_1^* \sum_{i=0}^q c_i s^i}) \cdot e(g_1^{\boldsymbol{d}_2 \sum_{i=0}^q c_i s^i}, W_2)$, 0 otherwise.

Fig. 6. The first intermediate accumulator scheme.

B.2 Second Intermediate Accumulator: Increasing DPVS Dimension

In Figure 7 we present our second intermediate accumulator. This accumulator is the based on the previous one, presented in Figure 6, except that we increased the DPVS dimension from 2 to 3. We also modified Eval, WitCreate and Verify algorithm in the following manner: we added elements of bases $\boldsymbol{d}_2^*, \boldsymbol{d}_3^*$ into the accumulator and elements of bases $\boldsymbol{d}_2, \boldsymbol{d}_3$ into the membership verification. Public key of the scheme was changed accordingly to the algorithms changes. The changes are explained in Section 5.2 and detailed in Appendix A.

- $\mathsf{Gen}(1^\kappa, q)$: run a bilinear group generation algorithm to get $\Gamma = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2)$. Then choose randoms $s, \gamma \leftarrow \mathbb{Z}_p^*$, and run $\mathsf{Dual}(\mathbb{Z}_p^3)$ to get $\mathbb{D} = (\boldsymbol{d}_1, \boldsymbol{d}_2, \boldsymbol{d}_3), \mathbb{D}^* = (\boldsymbol{d}_1^*, \boldsymbol{d}_2^*, \boldsymbol{d}_3^*)$. Let $\psi \in \mathbb{Z}_p$ be the random such that $\boldsymbol{d}_1 \cdot \boldsymbol{d}_1^* = \boldsymbol{d}_2 \cdot \boldsymbol{d}_2^* = \boldsymbol{d}_3 \cdot \boldsymbol{d}_3^* = \psi$. Set $\mathsf{sk_{acc}} = (s, \gamma, \mathbb{D}, \mathbb{D}^*)$,
$$\mathsf{pk_{acc}} = \begin{pmatrix} \Gamma, g_1^{\boldsymbol{d}_1}, g_1^{\boldsymbol{d}_2}, g_1^{\boldsymbol{d}_2 s}, \cdots, g_1^{\boldsymbol{d}_2 s^q}, g_2^{\boldsymbol{d}_1^*}, g_2^{\boldsymbol{d}_1^* s}, \\ \cdots, g_2^{\boldsymbol{d}_1^* s^q}, g_2^{\boldsymbol{d}_2^* \gamma}, g_2^{\boldsymbol{d}_2^* s}, \cdots, g_2^{\boldsymbol{d}_2^* s^q}, g_2^{\boldsymbol{d}_3^*} \end{pmatrix}.$$
Return $\mathsf{sk_{acc}}, \mathsf{pk_{acc}}$.

- $\mathsf{Eval}(\mathsf{sk_{acc}}, \mathsf{pk_{acc}}, \mathcal{X})$: compute the coefficients $\{a_i\}_{i=0,\cdots,q}$ of the polynomial $\mathsf{Ch}_{\mathcal{X}}[Z] = \prod_{x \in \mathcal{X}}(Z + x)$. Then pick $r \leftarrow \mathbb{Z}_p$ and compute $\mathsf{acc}_{\mathcal{X}} = g_1^{\boldsymbol{d}_1 \sum_{i=0}^q a_i s^i + r(\boldsymbol{d}_2 - \gamma \boldsymbol{d}_3)}$, and return $\mathsf{acc}_{\mathcal{X}}$.

- $\mathsf{PublicEval}(\mathsf{pk_{acc}}, \mathcal{X})$: compute the coefficients $\{a_i\}_{i=0,\cdots,q}$ of the polynomial $\mathsf{Ch}_{\mathcal{X}}[Z] = \prod_{x \in \mathcal{X}}(Z + x)$. Then compute $\mathsf{acc}_{\mathcal{X}} = g_2^{\boldsymbol{d}_1^* \sum_{i=0}^q a_i s^i}$, and return $\mathsf{acc}_{\mathcal{X}}$.

- $\mathsf{WitCreate}(\mathsf{pk_{acc}}, \mathcal{X}, \mathsf{acc}_{\mathcal{X}}/\mathsf{accp}_{\mathcal{X}}, \mathcal{I})$: let $\{b_i\}_{i=0,\cdots,q}$ be the coefficients of the polynomial $\mathsf{Ch}_{\mathcal{X} \setminus \mathcal{I}}[Z] = \prod_{x \in \mathcal{X} \setminus \mathcal{I}}(x + Z)$. Compute $W_1 = g_1^{\boldsymbol{d}_1 b_0}$ and $W_2 = g_2^{\boldsymbol{d}_2^* \sum_{i=1}^q b_i s^i}$, and return $\mathsf{wit}_{\mathcal{I}} = \mathsf{witp}_{\mathcal{I}} = (W_1, W_2)$.

- $\mathsf{Verify}(\mathsf{pk_{acc}}, \mathsf{acc}_{\mathcal{X}}, \mathsf{wit}_{\mathcal{I}}, \mathcal{I})$: let $\{c_i\}_{i=0,\cdots,q}$ be the coefficients of the polynomial $\mathsf{Ch}_{\mathcal{I}}[Z] = \prod_{x \in \mathcal{I}}(x + Z)$ and return 1 if $e(\mathsf{acc}_{\mathcal{X}}, g_2^{\boldsymbol{d}_1^*} \cdot g_2^{\boldsymbol{d}_2^* \gamma} \cdot g_2^{\boldsymbol{d}_3^*}) = e(W_1, g_2^{\boldsymbol{d}_1^* \sum_{i=0}^q c_i s^i}) \cdot e(g_1^{\boldsymbol{d}_2 \sum_{i=0}^q c_i s^i}, W_2)$, 0 otherwise.
- $\mathsf{PublicVerify}(\mathsf{pk_{acc}}, \mathsf{accp}_{\mathcal{X}}, \mathsf{wit}_{\mathcal{I}}, \mathcal{I})$: let $\{c_i\}_{i=0,\cdots,q}$ be the coefficients of the polynomial $\mathsf{Ch}_{\mathcal{I}}[Z] = \prod_{x \in \mathcal{I}}(x + Z)$ and return 1 if $e(g_1^{\boldsymbol{d}_1}, \mathsf{accp}_{\mathcal{X}}) = e(W_1, g_2^{\boldsymbol{d}_1^* \sum_{i=0}^q c_i s^i}) \cdot e(g_1^{\boldsymbol{d}_2 \sum_{i=0}^q c_i s^i}, W_2)$, 0 otherwise.

Fig. 7. The second intermediate accumulator scheme.

B.3 The dually Computable Accumulator Used in Our CP-ABE

Theorem 6. *These three dually computable accumulator schemes are correct, and satisfies collision resistance under the q-SBDH assumption, distinguishability and correctness of duality.*

Proofs of correctness and security (for publicly and privately computed accumulators) can be done as proofs of Theorems 1 and 2 respectively. For distinguishabality and correctness of duality we use the same arguments as the ones in Section 4 for our dually computable accumulator of Figure 3.

C Our KP-ABE Scheme

In this section we present a KP-ABE scheme, which is built as our CP-ABE of Section 5, and we compare it to existing schemes. Our KP-ABE is presented in Figure 9.

- $\mathsf{Gen}(1^\kappa, q)$: run a bilinear group generation algorithm to get $\Gamma = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2)$. Then choose randoms $s, \gamma \leftarrow \mathbb{Z}_p^*$, and run $\mathsf{Dual}(\mathbb{Z}_p^3)$ to get $\mathbb{D} = (\boldsymbol{d}_1, \boldsymbol{d}_2, \boldsymbol{d}_3), \mathbb{D}^* = (\boldsymbol{d}_1^*, \boldsymbol{d}_2^*, \boldsymbol{d}_3^*)$. Let $\psi \in \mathbb{Z}_p$ be the random such that $\boldsymbol{d}_1 \cdot \boldsymbol{d}_1^* = \boldsymbol{d}_2 \cdot \boldsymbol{d}_2^* = \boldsymbol{d}_3 \cdot \boldsymbol{d}_3^* = \psi$. Set $\mathsf{sk}_{\mathsf{acc}} = (s, \gamma, \mathbb{D}, \mathbb{D}^*)$,
$$\mathsf{pk}_{\mathsf{acc}} = \begin{pmatrix} \Gamma, g_1^{\boldsymbol{d}_3}, g_1^{\boldsymbol{d}_2}, g_1^{\boldsymbol{d}_2 s}, \cdots, g_1^{\boldsymbol{d}_2 s^q}, g_2^{\boldsymbol{d}_1^*}, g_2^{\boldsymbol{d}_1^* s}, \cdots, g_2^{\boldsymbol{d}_1^* s^q}, \\ g_2^{\boldsymbol{d}_2^* \gamma}, g_2^{\boldsymbol{d}_2^* s}, \cdots, g_2^{\boldsymbol{d}_2^* s^q}, g_2^{\boldsymbol{d}_3^*}, g_2^{\boldsymbol{d}_3^* s}, \cdots, g_2^{\boldsymbol{d}_3^* s^q} \end{pmatrix},$$
and return $\mathsf{sk}_{\mathsf{acc}}, \mathsf{pk}_{\mathsf{acc}}$.
- $\mathsf{Eval}(\mathsf{sk}_{\mathsf{acc}}, \mathsf{pk}_{\mathsf{acc}}, \mathcal{X})$: compute the coefficients $\{a_i\}_{i=0,\cdots,q}$ of the polynomial $\mathsf{Ch}_{\mathcal{X}}[Z] = \prod_{x \in \mathcal{X}}(Z + x)$. Then pick $r \leftarrow \mathbb{Z}_p$ and compute $\mathsf{acc}_{\mathcal{X}} = g_1^{\boldsymbol{d}_1 \sum_{i=0}^q a_i s^i + r(\boldsymbol{d}_2 - \gamma \boldsymbol{d}_3)}$, and return $\mathsf{acc}_{\mathcal{X}}$.
- $\mathsf{PublicEval}(\mathsf{pk}_{\mathsf{acc}}, \mathcal{X})$: compute the coefficients $\{a_i\}_{i=0,\cdots,q}$ of the polynomial $\mathsf{Ch}_{\mathcal{X}}[Z] = \prod_{x \in \mathcal{X}}(Z + x)$. Then compute $\mathsf{acc}_{\mathcal{X}} = g_2^{\boldsymbol{d}_3^* \sum_{i=0}^q a_i s^i}$, and return $\mathsf{accp}_{\mathcal{X}}$.
- $\mathsf{WitCreate}(\mathsf{pk}_{\mathsf{acc}}, \mathcal{X}, \mathsf{acc}_{\mathcal{X}}/\mathsf{accp}_{\mathcal{X}}, \mathcal{I})$: let $\{b_i\}_{i=0,\cdots,q}$ be the coefficients of the polynomial $\mathsf{Ch}_{\mathcal{X} \setminus \mathcal{I}}[Z] = \prod_{x \in \mathcal{X} \setminus \mathcal{I}}(x + Z)$. Compute $W_1 = g_1^{\boldsymbol{d}_3 b_0}$ and $W_2 = g_2^{\boldsymbol{d}_2^* \sum_{i=1}^q b_i s^i}$, and return $\mathsf{wit}_{\mathcal{I}} = \mathsf{witp}_{\mathcal{I}} = (W_1, W_2)$.
- $\mathsf{Verify}(\mathsf{pk}_{\mathsf{acc}}, \mathsf{acc}_{\mathcal{X}}, \mathsf{wit}_{\mathcal{I}}, \mathcal{I})$: let $\{c_i\}_{i=0,\cdots,q}$ be the coefficients of the polynomial $\mathsf{Ch}_{\mathcal{I}}[Z] = \prod_{x \in \mathcal{I}}(x + Z)$ and return 1 if $e(\mathsf{acc}_{\mathcal{X}}, g_2^{\boldsymbol{d}_1^*} \cdot g_2^{\boldsymbol{d}_2^* \gamma} \cdot g_2^{\boldsymbol{d}_3^*}) = e(W_1, g_2^{\boldsymbol{d}_3^* \sum_{i=0}^q c_i s^i}) \cdot e(g_1^{\boldsymbol{d}_2 \sum_{i=0}^q c_i s^i}, W_2)$, 0 otherwise.
- $\mathsf{PublicVerify}(\mathsf{pk}_{\mathsf{acc}}, \mathsf{accp}_{\mathcal{X}}, \mathsf{wit}_{\mathcal{I}}, \mathcal{I})$: let $\{c_i\}_{i=0,\cdots,q}$ be the coefficients of the polynomial $\mathsf{Ch}_{\mathcal{I}}[Z] = \prod_{x \in \mathcal{I}}(x + Z)$ and return 1 if $e(g_1^{\boldsymbol{d}_3}, \mathsf{acc}_{\mathcal{X}}) = e(W_1, g_2^{\boldsymbol{d}_3^* \sum_{i=0}^q c_i s^i}) \cdot e(g_1^{\boldsymbol{d}_2 \sum_{i=0}^q c_i s^i}, W_2)$, 0 otherwise.

Fig. 8. The dually computable accumulator used in our CP-ABE scheme.

Theorem 7. *Our scheme is correct and satisfies adaptive indistinguishability under* SXDH.

Correctness and security proofs of our KP-ABE can be done as for our CP-ABE.

In Table 4 we compare our KP-ABE with other KP-ABE schemes. All schemes are for single authority, secure in the standard model, bounded and in the pairing settings.

As we notice for our CP-ABE, there exist schemes that are unbounded or deal with non-monotonic access policies. We leave as an open problem to modify our KP-ABE to achieve such properties.

- $\mathsf{Setup}(\lambda, 1^q)$: generate bilinear group $\varGamma = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, e, g_1, g_2)$, dual pairing vector spaces $(\mathbb{D}, \mathbb{D}^*) \leftarrow \mathsf{Dual}(\mathbb{Z}_p^6)$ such that $\mathbb{D} = (\boldsymbol{d}_1, \cdots, \boldsymbol{d}_6)$, $\mathbb{D}^* = (\boldsymbol{d}_1^*, \cdots, \boldsymbol{d}_6^*)$ and $\boldsymbol{d}_i \cdot \boldsymbol{d}_i^* = \psi$, for $i = 1, \cdots, 6$ and $\psi \in \mathbb{Z}_p$. Also choose $\gamma, s, x_0, y_0, z_0 \leftarrow \mathbb{Z}_p$ and a hash function $\mathcal{H}$ that takes as input an attributes set and outputs an element of $\mathbb{Z}_p \setminus \{\gamma, s, x_0, y_0, z_0\}$. Set $Q = 2^q - 1$, $\mathsf{msk} = \left(\gamma, s, g_2^{\boldsymbol{d}_2^*}, \left\{g_1^{\boldsymbol{d}_1 s^i}\right\}_{i=0}^{Q}, \left\{g_1^{\boldsymbol{d}_3 s^i}\right\}_{i=1}^{Q}\right)$ and

$$\mathsf{pk} = \begin{pmatrix} \varGamma, g_1^{\boldsymbol{d}_3}, g_1^{\boldsymbol{d}_2}, g_1^{\boldsymbol{d}_2 s}, \cdots, g_1^{\boldsymbol{d}_2 s^Q}, g_2^{\boldsymbol{d}_1^*}, g_2^{\boldsymbol{d}_1^* s}, \cdots, g_2^{\boldsymbol{d}_1^* s^Q}, g_2^{\boldsymbol{d}_2^* \gamma}, \\ g_2^{\boldsymbol{d}_2^* s}, \cdots, g_2^{\boldsymbol{d}_2^* s^Q}, g_2^{\boldsymbol{d}_3^*}, g_2^{\boldsymbol{d}_3^* s}, \cdots, g_2^{\boldsymbol{d}_3^* s^Q}, \mathcal{H}, x_0, y_0, z_0 \end{pmatrix}.$$

  Return $\mathsf{msk}, \mathsf{pk}$.
- $\mathsf{KeyGen}(\mathsf{pk}, \mathsf{msk}, \varPi)$: let $\varPi = \pi_1 \vee \pi_2 \vee \cdots \vee \pi_l$ be the access policy, where $l \in \mathbb{N}$ is the number of clauses in the policy, and π_i for $i = 1, \cdots, l$ is a conjunction of attributes. Define $\mathcal{Y}_i$ for $i = 1, \cdots, l$ as the set of attributes associated to clause π_i and $\mathcal{Y} = \cup_{i=1}^{l} \mathcal{H}(\mathcal{Y}_i) \cup \{y_0, z_0\}$. Let $\{m_i\}_{i=0}^{Q}$ be the coefficients of polynomial $\mathsf{Ch}_{\mathcal{Y}}[Z]$. Pick $r \leftarrow \mathbb{Z}_p$ and set $\mathsf{sk}_{\varPi} = \mathsf{acc}_{\mathcal{Y}} = g_1^{\boldsymbol{d}_1 \sum_{i=0}^{Q} m_i s^i + r(\boldsymbol{d}_2 - \gamma \boldsymbol{d}_3)}$
- $\mathsf{Encrypt}(\mathsf{pk}, \varUpsilon, \mathsf{m})$: let $k \in \mathbb{N}$ be the number of attributes in $\varUpsilon$. Compute $p_1, \cdots, p_{2^k - 1}$ all the non-empty parties of $\varUpsilon$ and set $\mathcal{X} = \{\mathcal{H}(p_i)\}_{i=1}^{2^k - 1} \cup \{x_0, z_0\}$. Compute $\{a_i\}_{i=0,\cdots,Q}$ the coefficients of the polynomial $\mathsf{Ch}_{\mathcal{X}}[Z] = (x_0 + Z) \cdot (z_0 + Z) \cdot \prod_{i=1}^{2^k - 1} (\mathcal{H}(p_i) + Z)$. Choose $z, \alpha_1, \alpha_2 \leftarrow \mathbb{Z}_p$ and do
  - *Mask computation:* define $\mathsf{accp}_{\mathcal{X} = g_2^{\boldsymbol{d}_3^* \sum_{i=0}^{Q} a_i s^i}}$ and $\boldsymbol{H} = e(g_1^{\boldsymbol{d}_3}, \mathsf{accp}_{\mathcal{X}})^{\alpha_1 \alpha_2}$.
  - *Anticipation fo the witnesses and auxiliary information computation:* set $\mathsf{aux}_1 = g_1^{\alpha_2 \boldsymbol{d}_3 (x_0 + y_0)}$ and $\mathsf{aux}_2 = g_2^{-\boldsymbol{d}_1^* \alpha_1 \alpha_2 + z(\gamma \boldsymbol{d}_2^* + \boldsymbol{d}_3^*)}$.
  - *Anticipation of the element computation:* set $ele_1 = g_2^{\alpha_1 \boldsymbol{d}_3^* (z_0 s + s^2)}$, $ele_2 = g_2^{\alpha_1 \boldsymbol{d}_3^* (z_0 + s)}$, $ele_3 = g_1^{\alpha_1 \alpha_2 \boldsymbol{d}_2 (z_0 s + s^2)}$ and $ele_4 = g_1^{\alpha_1 \alpha_2 \boldsymbol{d}_2 (z_0 + s)}$

  Set $\mathsf{ct}_{\varUpsilon} = (ele_1, ele_2, ele_3, ele_4, \mathsf{aux}_1, \mathsf{aux}_2, \mathsf{m} \cdot \boldsymbol{H})$ and return $\mathsf{ct}_{\varUpsilon}$.
- $\mathsf{Decrypt}(\mathsf{pk}, \mathsf{sk}_{\varPi}, \varPi, \mathsf{ct}_{\varUpsilon}, \varUpsilon)$: Find p_{j^*} (for $j^* \in \left\{1, \cdots, 2^k - 1\right\}$) such that $\varUpsilon$ satisfies $\varPi$ (if no party exists, then return reject symbol $\perp$). It means that there exist $j \in [1, \cdots, l]$ such that $p_{j^*} = \mathcal{Y}_j$ and $\mathcal{H}(p_{j^*}) = \mathcal{H}(\mathcal{Y}_j) = \zeta$. Let $\{c_i\}_{i=0}^{Q}$ be the coefficients of the polynomial $\mathsf{Ch}_{\mathcal{X}}[Z]/(z_0 + Z)(\zeta + Z))$. Let $\{t_i\}_{i=0}^{Q}$ be the coefficients of the polynomial $\mathsf{Ch}_{\mathcal{Y}}[Z]/((z_0 + Z)(\zeta + Z))$. Find $\delta, \delta' \in \mathbb{Z}_p$ such that $c_0 = x_0 \delta$ and $t_0 = y_0 \delta'$. Set $W_2 = g_2^{\boldsymbol{d}_2^* \sum_{i=1}^{Q} c_i s^i}$, $W_2' = g_2^{\boldsymbol{d}_2^* \sum_{i=1}^{Q} t_i s^i}$ and compute

$$\frac{\mathsf{m} \cdot \boldsymbol{H}}{\left(e(\mathsf{aux}_1^{\delta\delta'}, ele_1 \cdot ele_3^{\zeta}) \cdot e(ele_2 \cdot ele_4^{\zeta}, W_2^{\delta'} \cdot W_2'^{\delta}) \cdot e(\mathsf{acc}_{\mathcal{X}}, \mathsf{aux}_2)^{\delta'}\right)^{\delta^{-1}}}$$

  to get m or $\perp$.

Fig. 9. Our KP ABE scheme.

Table 4. Comparison of KP-ABE schemes for monotone NC^1 circuits, based on pairings. Here q is the bound on the number of attributes in the scheme, and l is the number of rows in the access matrix when the policy is expressed with LSSS matrix. "Sec." means "security".

Schemes	$\lvert\mathsf{pk}\rvert$	$\lvert\mathsf{ct}\rvert$	$\lvert\mathsf{sk}\rvert$	Sec. Adaptive	Assumption	One-Use	Group Order	Pairing
[21]	$O(q)$	$O(l)$	$O(q)$	$\times$	Static	Yes	Prime	Symmetric
[24]	$O(q)$	$O(q)$	$O(l)$	$\surd$	Static	No	Composite	Symmetric
[27]	$O(q)$	$O(q)$	$O(l)$	$\surd$	Non Static	No	Prime	Symmetric
[23]	$O(n)$	$O(n)$	$O(l)$	$\surd$	Static	No	Prime	Asymmetric
Our	$O(2^q)$	$O(1)$	$O(1)$	$\surd$	Static	No	Prime	Asymmetric

D Non-Monotonic Access Policy

To improve our CP-ABE scheme so that it deals with "NO" gates, we might need to use *universal* accumulators. A universal accumulator scheme provides both membership and non-membership proofs. We might use non-membership proofs to deal with "NO" gates. The dually computable feature can easily be defined for universal accumulator schemes. However, we were not able to construct such schemes. Our accumulator of Figure 1 can be made universal, following [19]'s idea for non-membership proofs: the use of Bezout's coefficients. Using Extended Euclidean algorithm, compute polynomials $q_1[Z], q_2[Z]$ such that $\mathsf{Ch}_{\mathcal{X}}[Z]q_1[Z] + \mathsf{Ch}_{\mathcal{I}}[Z]q_2[Z] = 1$ (at the condition that $\mathcal{I} \cap \mathcal{X} = \varnothing$ otherwise the gcd of their associate polynomials is not equal to 1). Then, set $W_1 = g_2^{\boldsymbol{d}_1 q_1(s)}$ and $W_2 = g_2^{\boldsymbol{d}_2 q_2(s)}$. However, when universal, our accumulator is no longer *dually computable*: in the non-membership verification, we have $e(\mathsf{acc}_{\mathcal{X}}, W_1)$. Therefore, as $\mathsf{acc}_{\mathcal{X}}$ is replaced by $\mathsf{accp}_{\mathcal{X}}$ which is composed of two elements of $\mathbb{G}_2$, the pairing with W_1 cannot work. To keep it working, we would have to modify the witness, and thus we would no longer satisfies correctness of duality. Plus, the modification requires the use of private elements.

E Sets Operations

In this section we show that our accumulator presented in Figure 1, in Section 3, can be used for sets operations. Indeed, our above accumulator construction can deal with subset queries, meaning that he can provide short (constant size) witness for the intersection, the union and the set difference of two accumulators.

Subset. As we already saw in Section 3, our accumulator of Figure 1 can deal with subset queries, meaning that he can provide short (constant size) witness that a given set is a subset of another set.

Union. Our dually computable accumulator in Figure 3 can provide constant size proof of sets union. Let $\mathcal{X}_1, \mathcal{X}_2$ be two sets and $\mathsf{acc}_{\mathcal{X}_1} \leftarrow \mathsf{Eval}(\mathsf{sk}_{\mathsf{acc}}, \mathsf{pk}_{\mathsf{acc}}, \mathcal{X}_1) = g_1^{\boldsymbol{d}_1 \sum_{i=0}^{q} a_i s^i}$, $\mathsf{accp}_{\mathcal{X}_2} = g_2^{\boldsymbol{d}_1^* \sum_{i=0}^{q} b_i s^i}$. Compute $\mathsf{accp}_{\mathcal{X}_1 \cup \mathcal{X}_2} = g_1^{\boldsymbol{d}_2 \sum_{i=0}^{q} c_i s^i}$ and $\mathsf{aux} = g_2^{\boldsymbol{d}_2^* \sum_{i=0}^{q} e_i s^i}$ where $\{c_i, e_i\}_{i=0}^{q}$ are respectively the coefficients of polynomials $\prod_{x \in \mathcal{X}_1 \cup \mathcal{X}_2} (x + Z)$ and $\prod_{x \in \{\mathcal{X}_1 \uplus \mathcal{X}_2\} \setminus \{\mathcal{X}_1 \cup \mathcal{X}_2\}} (x + Z)$. We have that $(\mathsf{acc}_{\mathcal{X}_1}, \mathsf{accp}_{\mathcal{X}_2}, \mathsf{accp}_{\mathcal{X}_1 \cup \mathcal{X}_2}, \mathsf{aux})$ is a constant size witness of union: verification is done by checking if $e(\mathsf{accp}_{\mathcal{X}_1 \cup \mathcal{X}_2}, \mathsf{aux})$ is equal to $e(\mathsf{acc}_{\mathcal{X}_1}, \mathsf{accp}_{\mathcal{X}_2})$.

However, this is only working when the intersection of two sets is queried, it does not hold for more than two sets. For more than two sets, we can apply [19]'s methodology with either our first accumulator (Figure 1) or our dually computable accumulator (Figure 3): first we prove that each set is a subset of the union, then we prove that the intersection is a subset of the multiset union (i.e. $\uplus$).

- For the first step, we compute a (publicly computable) accumulator of the union of the sets and create a subset witness for each of the sets.
- For the second step, we compute a (public accumulator) of the multiset union of the sets, then we create a subset witness for each set and the multiset union.

Intersection. With our accumulators (dually computable or not) we can do as in [19] to prove that a set is the intersection of two given sets: first we prove that the intersection is a subset of all the sets, then the *completeness* of the intersection (i.e. all elements of the intersection are in the answer).

- For the first step, we compute an (public) accumulator of the intersection, then create a subset witness to prove that the intersection is a subset of all sets.
- For the second step, we compute for each set a (public) accumulator of the set minus the intersection of the sets, and then we find with Extended Euclidean algorithm the Bezout's coefficients that proves that the gcd of the new sets is 1.

Set Difference. To prove that a set correspond to the set difference of two sets given as input, there are two ways to do: as in [33] or as in [19] depending on the privacy we want to reach. Let $\mathcal{X}, \mathcal{S}$ be two sets such that $\mathcal{S} \subset \mathcal{X}$ and $\mathcal{D} = \mathcal{X} \setminus \mathcal{S}$. Notice that both our accumulator of Figure 1 and our dually computable of Figure 3 can be used.

To do as in [33], we first prove that $\mathcal{D}$ is a subset of $\mathcal{X}$, then we prove that the intersection of $\mathcal{X}$ and $\mathcal{X} \setminus \mathcal{D}$ is equal to $\mathcal{X} \cap \mathcal{S}$.

[19] does the same, but combine it with a non-interactive zero knowledge protocol.

Comparison.

- In terms of sets operations, our accumulator can deal with subset, union, intersection and set difference, as in [33] and [19]. [40] deals with more operations but is less efficient.
- Making our accumulator dynamic will lead to dynamic sets operations verification, as in [33] and [19] or [40]. We leave this as an open problem, such as batch updates (updating witnesses after that several elements have been added or removed from teh accumulator).
- Our scheme is quite similar to [19], therefore following the latter's idea we may be able to make our accumulator satisfy zero-knowledge and then obtain privacy preserving sets operations verification.

- [28] establish the following open problem: “Construct a pairing-based accumulator supporting set operations with constant-size witnesses achieving security under simple assumptions.” They partially answer it with their accumulator dealing with subset queries. The key word in their problem is “simple assumption” which means “static assumptions”. Our accumulator is secure under a non-static assumption, therefore we do not answer their problem.
- Finally, notice that [33] and [19] combine their accumulators with an *accumulation tree* to protect the integrity of the evaluated values.