Predicate Encryption with Partial Public Keys

C. Blundo, V. Iovino, and G. Persiano
Dipartimento di Informatica ed Applicazioni
Università di Salerno
I-84084 Fisciano (SA), Italy
{carblu, iovino, giuper}@dia.unisa.it

Abstract

Predicate encryption is a new powerful cryptographic primitive which allows for fine-grained access control for encrypted data: the owner of the secret key can release partial keys, called *tokens*, that can decrypt only a specific subset of ciphertexts. More specifically, in a predicate encryption scheme, ciphertexts and tokens have attributes and a token can decrypt a ciphertext if and only if a certain predicate of the two associated attributes holds.

In this paper, ciphertext attributes are vectors $\vec{x}$ of fixed length ℓ over an alphabet Σ and token attributes, called *patterns*, are vectors $\vec{y}$ of the same length over the alphabet $\Sigma_\star = \Sigma \cup \{\star\}$. We consider the predicate $\mathsf{Match}(\vec{x}, \vec{y})$ introduced by [BW07] which is true if and only if $\vec{x} = \langle x_1, \ldots, x_\ell \rangle$ and $\vec{y} = \langle y_1, \ldots, y_\ell \rangle$ agree in all positions i for which $y_i \neq \star$.

Various security notions are relevant for predicate encryption schemes. First of all, one wants the ciphertexts to hide its attributes (this property is called semantic security). In addition, it makes sense also to consider the property of *token security*, a security notion in which the token is required not to reveal any information on the associated pattern. It is easy to see that predicate privacy is impossible to achieve in a public-key setting. In [SSW09], the authors considered the notion of a predicate encryption scheme in the symmetric-key setting and gave the first construction with token security.

In this paper, we consider the notion of a *partial public key encryption* (as suggested in [SSW09]) in which a partial public key allows a user to generate only a subset of the ciphertexts. We give a construction which is semantically secure and in which a token does not reveal any information on the associated pattern except for the locations of the $\star$'s. The proofs of security of our construction are based

on hardness assumptions in bilinear groups of *prime* order; this greatly improves the efficiency of the construction when compared to previous constructions ([SSW09]) which used groups of composite orders.

Our security proofs do not use random oracles.

1 Introduction

In a predicate encryption scheme, ciphertexts and keys have attributes and a key can decrypt a certain ciphertext if and only if a certain predicate on the two attributes holds. In this paper, ciphertext attributes $\vec{x}$ are vectors of fixed length ℓ over an alphabet Σ and key attributes (also called *patterns*) are vectors of the same length over the alphabet $\Sigma_\star = \Sigma \cup \{\star\}$. We consider the predicate $\mathsf{Match}(\vec{x}, \vec{y})$ which is true if and only if $\vec{x} = \langle x_1, \ldots, x_\ell \rangle$ and $\vec{y} = \langle y_1, \ldots, y_\ell \rangle$ agree in all positions i for which $y_i \neq \star$.

We are interested in two security requirements which, roughly speaking, can be described as follows. We first require that a ciphertext $\tilde{X}$ should hide all information on the associated attribute vector $\vec{x}$ (we call this notion Semantic Security). In addition, we require that a key T (also called a *token*) should hide all information on the associated pattern $\vec{y}$ (we call this notion Token Security). Formal definitions of the two security requirements are found in Section 2. We would like to stress though that Token Security is not achievable in a pure public-key scenario: given token T for an unknown pattern $\vec{y}$ an adversary could check if $\mathsf{Match}(\vec{x}, \vec{y})$ holds by creating a ciphertext C for attribute vector $\vec{x}$ using the public key, and then testing T against C. We thus consider the *partial public key* model in which the key owner can decide on a policy that describes which subset of the ciphertexts can be generated. More specifically, a policy $\mathsf{Pol} = \langle \mathsf{Pol}_1, \ldots, \mathsf{Pol}_\ell \rangle$ is simply a vector of length ℓ of subsets of Σ with the following intended meaning: the public key associated with policy Pol allows to create ciphertexts with attribute vector $\vec{x} = \langle x_1, \ldots, x_\ell \rangle$ iff and only for $i \in [\ell]$ we have that $x_i \in \mathsf{Pol}_i$. The private key scenario corresponds to a policy Pol with $\mathsf{Pol}_i = \emptyset$ for all i's; whereas a public key scenario corresponds to a policy with $\mathsf{Pol}_i = \Sigma$ for all i's. For example, for $\ell = 2, \Sigma = \{0, 1\}$, and policy $\mathsf{Pol} = \langle \{1\}, \{0, 1\} \rangle$, then public key $\mathsf{PPK}_{\mathsf{Pol}}$ associated with Pol allows to create ciphertexts with attribute vector $\vec{x} = \langle 1, 0 \rangle$ but not $\vec{x} = \langle 0, 1 \rangle$. In the formal definition of Token Security we thus require that an adversary is not able to distinguish between tokens with pattern $\vec{y}_0$ or $\vec{y}_1$ with respect to a policy Pol provided that the two patterns have the same value of the predicate Match for all attributes $\vec{x}$ that can be encrypted under policy Pol.

Previous work. The first example of predicate encryption scheme has been given by Boneh et al. [BDOP04] that introduced the concept of an encryption scheme supporting equality test. Roughly speaking, in such an encryption scheme, the owner of the public key can compute, for any message M, a token T_M that allows to test if a given ciphertext encrypts message M without obtaining any additional information. More recently, along this line of research, Goyal et al. [GPSW06] have introduced the concept of an attribute-based encryption scheme (ABE scheme). In an ABE scheme, a ciphertext is labeled with a set of attributes and private keys are associated with a predicate. A private key can decrypt a ciphertext iff the attributes of the ciphertext satisfy the predicate associated with the key. An ABE scheme can thus been seen as a special encryption scheme for which, given the key associated with a predicate P, one can test whether a given ciphertext carries a message M that satisfies predicates P without having to decrypt and without getting any additional information. The construction of [GPSW06] is very general as it supports any predicate that can be expressed as a circuit with threshold gates but attributes associated with a ciphertexts appear in clear in a ciphertext. Boneh and Waters [BW07] were the first to give predicate encryption schemes that guaranteed security of the attributes for the Match predicate and showed that this implies construction for several families of predicates including conjunctions of equality, range predicate and subset predicates. This has been subsequently extended to disjunctions, polynomial equations and inner products [KSW08]. Both constructions are based on hardness assumptions regarding bilinear groups on *composite order*. Iovino and Persiano [IP08] gave more efficient constructions based on hardness assumptions regarding bilinear group of *prime order*. Shen et al. [SSW09] were the first to consider the issue of token security and gave *private-key* predicate encryption schemes for inner product based on hardness assumptions regarding bilinear group of order product of four primes.

Our results. In this paper we give a predicate encryption scheme with partial public keys based on hardness assumptions regarding bilinear group of prime order for the Match predicate. Being able to use prime order groups greatly improves the efficiency of the resulting encryption schemes since, for the same level of security, our constructions uses groups of much smaller order. Our scheme guarantees privacy of the attributes associated with the ciphertexts (see Definition 3). In addition, we also show that tokens only reveal the positions of the $\star$-entries in the associated pattern. More precisely,

for any two patterns $\vec{y}_0$ and $\vec{y}_1$ that have $\star$-entries in the same positions, no probabilistic polynomial time adversary can distinguish a token for $\vec{y}_0$ from a token for $\vec{y}_1$ better than guessing at random (see Definition 4).

2 Predicate Encryption Schemes with Partial Public Keys

In this section we present the notion of a *predicate encryption scheme with partial public keys*. Following [SSW09, KSW08], we present our definitions (and constructions in Section 4) for the case in which the ciphertexts are *predicate-only*; that is, they do not carry any message and only specify the attributes. It is straightforward to extend the definitions (and the constructions) to the case in which ciphertexts carry a message.

In the following we will denote by $[\ell]$ the set $\{1, \ldots, \ell\}$ of natural numbers. We let Σ denote an *alphabet* (that is, a finite set of symbols) and let 2^{Σ} denote its power set (that is, the family of all subsets of Σ). Furthermore, we let $\Sigma_\star$ denote the alphabet Σ augmented with the special symbol $\star$. Finally, we say that function $\nu : \mathbb{N} \to [0, 1]$ is *negligible* if, for all polynomials poly and sufficiently large n, we have that $\nu(n) \leq 1/\mathsf{poly}(n)$.

We start by defining the notion of a *policy* and of an *allowed attribute vector* for a policy.

Definition 1 *Fix the number $\ell > 0$ of attributes and alphabet Σ. A* policy $\mathsf{Pol} = \langle \mathsf{Pol}_1, \ldots, \mathsf{Pol}_l \rangle \in (2^{\Sigma} \setminus \emptyset)^{\ell}$ *is a sequence of ℓ non-empty subsets of Σ. The set $\mathbb{X}_{\mathsf{Pol}}$ of* allowed attribute vectors *for policy* Pol *consists of all vectors $\vec{x} \in \Sigma^{\ell}$ such that for $i \in [\ell]$ we have that $x_i \in \mathsf{Pol}_i$.*

Our predicate encryption schemes are for the predicate $\mathsf{Match} : \Sigma^{\ell} \times \Sigma_{\star}^{\ell} \to \{0, 1\}$ defined as follows: $\mathsf{Match}(\vec{x}, \vec{y}) = 1$ if and only if $\vec{x} = \langle x_1, \ldots, x_\ell \rangle$ and $\vec{y} = \langle y_1, \ldots, y_\ell \rangle$ agree in all positions i for which $y_i \neq \star$. We remark that a predicate encryption scheme for the Match predicate implies efficient constructions for several other predicates (see [BW07] for the descriptions of the reductions).

Definition 2 *A* Predicate Encryption Scheme with Partial Public Keys *for the predicate* Match *consists of five algorithms:*

$\mathsf{Setup}(1^n, 1^{\ell})$*: Given the security parameter n and the number of attributes $\ell = \mathsf{poly}(n)$, procedure* Setup *outputs the* secret key SK*.*

$\mathsf{PPKeyGen}(\mathsf{SK}, \mathsf{Pol})$*: Given the secret key* SK *and the policy* $\mathsf{Pol} \in (2^{\Sigma} \setminus \emptyset)^{\ell}$*, procedure* $\mathsf{PPKeyGen}$ *outputs the* partial public key $\mathsf{PPK}_{\mathsf{Pol}}$ *relative to policy* Pol*. We denote by* PK *the public key relative to policy* $\mathsf{Pol} = \Sigma^{\ell}$*.*

$\mathsf{Encryption}(\mathsf{PPK}_{\mathsf{Pol}}, \vec{x})$*: Given the partial public key* $\mathsf{PPK}_{\mathsf{Pol}}$ *relative to policy* Pol *and the* attribute vector $\vec{x} \in \mathbb{X}_{\mathsf{Pol}}$*, procedure* $\mathsf{Encryption}$ *outputs an* encrypted attribute vector $\tilde{X}$*.*

$\mathsf{GenToken}(\mathsf{SK}, \vec{y})$*: Given the secret key* SK *and the* pattern vector $\vec{y} \in \Sigma_{\star}^{\ell}$*, procedure* $\mathsf{GenToken}$ *outputs* token $T_{\vec{y}}$ *and* $P_{\vec{y}}$ *where* $P_{\vec{y}} = \{i \in [\ell] \mid y_i = \star\}$*.*

$\mathsf{Test}(\tilde{X}, T_{\vec{y}})$*: given the encrypted attribute vector* $\tilde{X}$ *corresponding to attribute vector* $\vec{x}$ *and the token* $T_{\vec{y}}$ *corresponding to pattern* $\vec{y}$*, procedure* Test *returns* $\mathsf{Match}(\vec{x}, \vec{y})$ *with overwhelming probability. More precisely, for all* $\ell = \mathsf{poly}(n)$*, all policies* $\mathsf{Pol} \in (2^{\Sigma} \setminus \emptyset)^{\ell}$*, all attribute vectors* $\vec{x} \in \mathbb{X}_{\mathsf{Pol}}$*, and all patterns* $\vec{y} \in \Sigma_{\star}^{\ell}$*, we have that*

$$\mathrm{Prob}[\,\mathsf{SK} \leftarrow \mathsf{Setup}(1^n, 1^{\ell});\ \mathsf{PPK}_{\mathsf{Pol}} \leftarrow \mathsf{PPKeyGen}(\mathsf{SK}, \mathsf{Pol}) : \mathsf{Test}(\mathsf{Encryption}(\mathsf{PPK}_{\mathsf{Pol}}, \vec{x}), \mathsf{GenToken}(\mathsf{SK}, \vec{y})) \neq \mathsf{Match}(\vec{x}, \vec{y})\,]$$

is negligible in n*.*

Remark 1 *Notice that in our definition of* $\mathsf{GenToken}$*, the algorithm outputs the set* $P_{\vec{y}}$ *of positions where* $y_i = \star$*. This is meant to stress that in our model the tokens could leak information on the* $\star$*'s positions. This is not a problem for many applications. Consider the scenario of a system that allows the search on a remote encrypted database. Then, a token issued by a user could leak 'which' fields (for example, 'Name' and 'City') the query involves but not the actual content of these fields (for example, 'Mario' and 'Roma'). Sometimes, we will ignore this fact and will not explicitly include* $P_{\vec{y}}$ *in the output of* $\mathsf{GenToken}$*.*

Next we state security in the selective attribute model.

2.1 Semantic security

Semantic security deals with an adversary that tries to learn information from ciphertexts. We define the security requirement by means of an indistinguishability experiment in which the adversary $\mathcal{A}$ selects two challenge attribute vectors $\vec{z}_0$ and $\vec{z}_1$ and a policy Pol. The adversary $\mathcal{A}$ then receives the partial public key $\mathsf{PPK}_{\mathsf{Pol}}$ and is allowed to issue token queries for patterns $\vec{y}$ such that $\mathsf{Match}(\vec{z}_0, \vec{y}) = \mathsf{Match}(\vec{z}_1, \vec{y}) = 0$. Finally, $\mathcal{A}$ receives encrypted attribute vector $\tilde{X}$ corresponding to a randomly chosen challenge

attribute vector $\vec{z}_\eta$. We require that $\mathcal{A}$ has probability essentially 1/2 of guessing η.

We model the semantic security property by means of the following game $\mathsf{SemanticExp}_{\mathcal{A}}$ between a challenger $\mathcal{C}$ and adversary $\mathcal{A}$.

$\mathsf{SemanticExp}_{\mathcal{A}}(1^n, 1^\ell)$

1. Initialization Phase. The adversary $\mathcal{A}$ announces two challenge attribute vectors $\vec{z}_0, \vec{z}_1 \in \Sigma^\ell$ and policy $\mathsf{Pol} \in (2^\Sigma \setminus \emptyset)^\ell$.
2. Key-Generation Phase. Challenger $\mathcal{C}$ computes the secret key SK by running the Setup procedure on input $(1^n, 1^\ell)$ and the partial public key $\mathsf{PPK}_{\mathsf{Pol}}$ by running $\mathsf{PPKeyGen}(\mathsf{SK}, \mathsf{Pol})$.
   $\mathsf{PPK}_{\mathsf{Pol}}$ is given to $\mathcal{A}$.
3. Query Phase I. $\mathcal{A}$ can make any number of token queries.
   $\mathcal{C}$ answers token query for pattern $\vec{y}$ as follows. If $\mathsf{Match}(\vec{z}_0, \vec{y}) = \mathsf{Match}(\vec{z}_1, \vec{y}) = 0$, then $\mathcal{A}$ receives the output of $\mathsf{GenToken}(\mathsf{SK}, \vec{y})$. Otherwise, $\mathcal{A}$ receives $\perp$.
4. Challenge construction. $\mathcal{C}$ chooses random $\eta \in \{0, 1\}$ and gives the output of $\mathsf{Encryption}(\mathsf{PK}, \vec{z}_\eta)$ to $\mathcal{A}$.
5. Query Phase II. Identical to Query Phase I.
6. Output phase. $\mathcal{A}$ returns η'.
   If $\eta = \eta'$ then the experiments returns 1 else 0.

Notice that in $\mathsf{SemanticExp}_{\mathcal{A}}$ we can assume, without loss of generality, that $\mathcal{A}$ always asks for PK (the public key that allows to encrypt all attribute vectors). We chose the formulation above to keep it similar to the game used to formalize the token security property (see Section 2.2).

Definition 3 *A predicate encryption scheme with partial public keys*

$(\mathsf{Setup}, \mathsf{PPKeyGen}, \mathsf{Encryption}, \mathsf{GenToken}, \mathsf{Test})$ *is* semantically secure*, if for all probabilistic polynomial-time adversaries* $\mathcal{A}$

$$\left|\mathrm{Prob}[\mathsf{SemanticExp}_{\mathcal{A}}(1^n, 1^\ell) = 1] - 1/2\right|$$

is negligible in n *for all* $\ell = \mathsf{poly}(n)$.

2.2 Token security

In this section, we present an experiment that models the fact that a token T gives no information on the associated pattern $\vec{y}$ but the position of the $\star$-entries. We use an indistinguishability experiment in which the adversary $\mathcal{A}$ picks two challenge patterns $\vec{y}_0$ and $\vec{y}_1$ such that $\vec{y}_{0,i} = \star$ iff $\vec{y}_{1,i} = \star$

and a policy Pol such that for all $\vec{x} \in \mathbb{X}_{\mathsf{Pol}}$ we have that $\mathsf{Match}(\vec{x}, \vec{y}_0) = \mathsf{Match}(\vec{x}, \vec{y}_1) = 0$. $\mathcal{A}$ receives the partial public key $\mathsf{PPK}_{\mathsf{Pol}}$ associated with Pol and $\mathcal{A}$ is allowed to issue token queries for patterns $\vec{y}$ of his choice. Finally, $\mathcal{A}$ receives the token associated to a randomly chosen challenge pattern $\vec{y}_\eta$. We require that $\mathcal{A}$ has probability essentially $1/2$ of guessing η.

We model the token security property by means of the following game $\mathsf{TokenExp}_{\mathcal{A}}$ between a challenger $\mathcal{C}$ and adversary $\mathcal{A}$.

$\mathsf{TokenExp}_{\mathcal{A}}(1^n, 1^\ell)$

1. Initialization Phase. The adversary $\mathcal{A}$ announces two challenge patterns $\vec{y}_0, \vec{y}_1 \in \Sigma_\star^\ell$ and a policy Pol such that for all $\vec{x} \in \mathbb{X}_{\mathsf{Pol}}$ we have that $\mathsf{Match}(\vec{x}, \vec{y}_0) = \mathsf{Match}(\vec{x}, \vec{y}_1) = 0$.
   If there exists $i \in [\ell]$ such that $y_{0,i} = \star$ and $y_{1,i} \neq \star$ or if there exists $i \in [\ell]$ such that $y_{1,i} = \star$ and $y_{0,i} \neq \star$ then the experiment returns 0.
2. Key-Generation Phase. The secret key SK is generated by the Setup procedure. The partial public key $\mathsf{PPK}_{\mathsf{Pol}}$ relative to policy Pol is generated running procedure $\mathsf{PPKeyGen}(\mathsf{SK}, \mathsf{Pol})$. $\mathsf{PPK}_{\mathsf{Pol}}$ is given to $\mathcal{A}$.
3. Query Phase I. $\mathcal{A}$ can make any number of token queries that are answered by returning $\mathsf{GenToken}(\mathsf{SK}, \vec{y})$.
4. Challenge construction. η is chosen at random from $\{0, 1\}$ and receives $\mathsf{GenToken}(\mathsf{SK}, \vec{y}_\eta)$.
5. Query Phase II. Identical to Query Phase I.
6. Output phase. $\mathcal{A}$ returns η'.
   If $\eta = \eta'$ then the experiments returns 1 else 0.

Definition 4 *A predicate encryption scheme with partial public keys* $(\mathsf{Setup}, \mathsf{PPKeyGen}, \mathsf{Encryption}, \mathsf{GenToken}, \mathsf{Test})$ *is* token secure *if for all probabilistic polynomial-time adversaries* $\mathcal{A}$*,*

$$\left|\mathrm{Prob}[\mathsf{TokenExp}_{\mathcal{A}}(1^n, 1^\ell) = 1] - 1/2\right|$$

is negligible in n *for all* $\ell = \mathsf{poly}(n)$*.*

Definition 5 *A predicate encryption scheme with partial public keys* $(\mathsf{Setup}, \mathsf{PPKeyGen}, \mathsf{Encryption}, \mathsf{GenToken}, \mathsf{Test})$ *is a* secure predicate encryption scheme with partial public keys *if it is both semantically secure and token secure.*

3 Background and Complexity Assumptions

Linear secret sharing In our assumptions and constructions we use the concept of a (k, n) linear secret sharing scheme (LSSS), for $k \leq n$. A (k, n) LSSS takes as input a *secret* s (typically from a finite field $\mathbb{F}_p$) and returns k *shares* $(s_1, \ldots, s_k)$ with the following properties. Any set of $k-1$ (or fewer) shares are independent among themselves and are independent from the secret s. In addition, the secret s can be expressed as a linear combination of the shares held by any k participants. More precisely, for any $F \subseteq [n]$ of size k there exist reconstruction coefficients α_i such that $s = \sum_{i \in F} \alpha_i s_i$. For instance, in Shamir's secret sharing scheme [Sha79], the reconstruction coefficients are the Lagrange interpolation coefficients. We stress that the reconstruction coefficients depend only on the set F and not on the actual shares.

The symmetric bilinear setting We have two multiplicative groups, the *base* group $\mathbb{G}$ and the *target* group $\mathbb{G}_T$ both of prime order p and a non-degenerate bilinear pairing function $\mathsf{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$. That is, for all $x \in \mathbb{G}$, $x \neq 1$, we have $\mathsf{e}(x, x) \neq 1$ and for all $x, y \in \mathbb{G}$ and all $a, b \in \mathbb{Z}_p$, we have $\mathsf{e}(x^a, y^b) = \mathsf{e}(x, y)^{ab}$. We denote by g and $\mathsf{e}(g, g)$ generators of $\mathbb{G}$ and $\mathbb{G}_T$. We call a tuple $\mathcal{I} = [p, \mathbb{G}, \mathbb{G}_T, g, \mathsf{e}]$ a *symmetric bilinear* instance and assume that there exists an efficient generation procedure that, on input security parameter 1^n, outputs an instance with $|p| = \Theta(n)$.

We now review and justify the hardness assumptions we will use for proving security of our constructions.

Our first two assumptions posit the hardness of distinguishing whether the exponents relative to given bases of a sequence of $(2\ell - 1)$ elements of $\mathbb{G}$ constitute the shares of 0 with respect to an $(\ell, 2\ell - 1)$ LSSS or one of the exponents (the exponent of the *challenge* element, usually denoted by Z in the following) is random. This computational problem is clearly trivial if $\ell - 1$ elements share the same base A with the challenge element Z. Indeed, given an ordered ℓ-subset $F = \langle f_1, \ldots, f_\ell \rangle$ of $[2\ell - 1]$, base A, elements A^{s_i} for $i \in \langle f_1, \ldots, f_{\ell-1} \rangle$ and challenge $Z = A^{s_{f_\ell}}$, checking if the exponents s_i constitute ℓ shares of 0 of an $(\ell, 2\ell - 1)$ LSSS is trivial by the linearity of the secret sharing scheme. In a bilinear setting, the problem remains easy in the base group if $(\ell - 1)$ elements share the same base $A \in \mathbb{G}$ even though this is different from the base $B \in \mathbb{G}$ of the challenge element. Specifically, given bases A and B, elements A^{s_i}, for $i \in \langle f_1, \ldots, f_{\ell-1} \rangle$ and challenge $Z = B^r$ it is possible to check whether the s_i's and r constitute ℓ shares of 0 of an

$(\ell, 2\ell-1)$ LSSS in the following way. First, use linearity to compute $A^{s_{f_\ell}}$ and then use bilinearity to check if $r = s_{f_\ell}$ by comparing $\mathsf{e}(A^{s_\ell}, B)$ and $\mathsf{e}(A, B^r)$. If instead less than $\ell - 1$ elements share the same base then the problem seems to be computationally difficult.

The Linear Secret Sharing Assumption (see Section 3.1 below) makes a formal statement of this fact. Specifically, we are given bases

$U_1, \ldots, U_{2\ell-1} \in \mathbb{G}$, elements $U_1^{a_1}, \ldots, U_\ell^{a_{2\ell-1}} \in \mathbb{G}$ and index $j \in [2\ell-1]$ of the challenge element and we have to decide whether $(a_1, \ldots, a_{2\ell-1})$ constitute an $(\ell, 2\ell-1)$ secret sharing of 0 or the exponent a_j of the challenge element is random. We stress that, for sake of ease of exposition, in stating the Linear Secret Sharing Assumption we have not tried to reduce the number of bases: we have $(2\ell-1)$ bases for $(2\ell-1)$ elements. It is not difficult to see that we could have used only 4 bases to formulate an assumption that is sufficient for proving the security of our constructions.

If we consider the same problem in the target group $\mathbb{G}_T$, it seems that it remains difficult even if $\ell - 1$ elements share the same base which is different from the base used for the challenge element. Indeed in the target group we are not allowed to use the pairing function e and thus we cannot use the same approach employed for the base group. The F-Linear Secret Sharing Assumption (see Section 3.2 below) makes a formal statement of this fact. By looking ahead, in the F-Linear Secret Sharing Assumption we have ℓ shares corresponding to an ordered subset $F = \langle f_1, \ldots, f_\ell \rangle$ of elements of $[2\ell-1]$ which appear as exponents of ℓ elements of $\mathbb{G}_T$: $\ell - 1$ of these elements share the same base $\mathsf{e}(g, g)$ (specifically, in the assumption we have $\mathsf{e}(\bar{U}_{f_j}, V_{f_j}) = \mathsf{e}(g, g)^{a_{f_j}}$ for $2 \leq j \leq \ell$) and the challenge element uses a different base (specifically, $\mathsf{e}(U_{f_1}, V_{f_1}) = \mathsf{e}(U_{f_1}, U_{f_1})^{a_{f_1}}$). The task is to decide whether the a_i's for $i \in F$ constitute an $(\ell, 2\ell-1)$ secret share of 0 or the a_{f_ℓ} is random. We state our assumptions using elements of $\mathbb{G}$ (i.e., the $\bar{U}_j$'s and the V_j's) instead of elements of $\mathbb{G}_T$ (i.e., giving only $\mathsf{e}(\bar{U}_j, V_j)$).

For each of the two above assumptions, we have a *split* version which we call the Split Linear Secret Sharing Assumption (see Section 3.3) and the F-Split Linear Secret Sharing Assumption (see Section 3.4). The split versions of our assumptions are derived by mixing the assumptions based on linear secret sharing with the Decision Linear Assumption (see [BW06]). In the Decision Linear Assumption, the task is to decide, given A, A^r, B, B^s, C, C^z whether $z = r - s$ or z is random. Specifically, in the Split Linear Secret Sharing Assumption we have bases $U_1, \ldots, U_{2\ell-1}$, elements $U_1^{a_1}, \ldots, U_{2\ell-1}^{a_{2\ell-1}}$ and $g^{a_1}, \ldots, g^{a_{2\ell-1}}$ with $(a_1, \ldots, a_{2\ell-1})$ constituting an $(\ell, 2\ell-1)$ LSSS of 0, and $2\ell - 2$ related instances of the Decision Linear Assumptions for a

randomly chosen $j \in [2\ell-1]$: $U_i^u, U_j^{a_j}, W^s$, with $i \in [2\ell-1] \setminus \{j\}$ in which we have to decide whether $s = u - a_j$. In addition, we are also given $\hat{U} = W^{u_j}$ where $U_j = g^{u_j}$. The F-Split Linear Secret Sharing Assumption is obtained is a similar way from the F-Linear Secret Sharing Assumption.

3.1 Linear Secret Sharing Assumption

Consider the following game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$.

$\mathsf{LSSExp}_{\mathcal{A}}(1^n, 1^\ell)$
01. $\mathcal{C}$ computes shares $a_1, \ldots, a_{2\ell-1}$ of 0 using an $(\ell, 2\ell-1)$ LSSS;
02. $\mathcal{C}$ chooses instance $\mathcal{I} = [p, \mathbb{G}, \mathbb{G}_T, g, \mathsf{e}]$ with security parameter 1^n;
03. $\mathcal{C}$ chooses random $j \in [2\ell-1]$;
04. **for** $i \in [2\ell-1]$
    $\mathcal{C}$ chooses random $u_i \in \mathbb{Z}_p$ and sets $U_i = g^{u_i}$ and $V_i = U_i^{a_i}$;
05. $\mathcal{C}$ chooses random $\eta \in \{0,1\}$;
06. **if** $\eta = 1$ **then** $\mathcal{C}$ sets $Z = U_j^{a_j}$ **else** $\mathcal{C}$ chooses random $Z \in \mathbb{G}$;
07. $\mathcal{C}$ runs $\mathcal{A}$ on input $[\mathcal{I}, j, (U_i)_{i\in[2\ell-1]}, (V_i)_{i\in[2\ell-1]\setminus\{j\}}, Z]$;
08. Let η' be $\mathcal{A}$'s guess for η;
09. **if** $\eta = \eta'$ **then** return 1 **else** return 0.

Assumption 1 (LSS Assumption) *The Linear Secret Sharing Assumption states that for all probabilistic polynomial-time algorithms $\mathcal{A}$,*
$\left|\mathrm{Prob}[\mathsf{LSSExp}_{\mathcal{A}}(1^n, 1^\ell) = 1] - 1/2\right|$ *is negligible in n for all $\ell = \mathsf{poly}(n)$.*

3.2 F-Linear Secret Sharing Assumption

Let $F = \langle f_1, \ldots, f_\ell \rangle$ be a sequence of ℓ distinct elements from $[2\ell-1]$. We formalize the F-Linear Secret Sharing Assumption (F-LSS Assumption) by means of the following game between a Challenger $\mathcal{C}$ and an Adversary $\mathcal{A}$.

F-$\mathsf{LSSExp}_{\mathcal{A}}(1^n, 1^\ell)$
01. $\mathcal{C}$ computes shares $a_1, \ldots, a_{2\ell-1}$ of 0 using an $(\ell, 2\ell-1)$ LSSS;
02. $\mathcal{C}$ chooses instance $\mathcal{I} = [p, \mathbb{G}, \mathbb{G}_T, g, \mathsf{e}]$ with security parameter 1^n;
03. **for** $i \in F$
    $\mathcal{C}$ chooses random $u_i \in \mathbb{Z}_p$ and sets $U_i = g^{u_i}$, $\bar{U}_i = g^{1/u_i}$, and $V_i = U_i^{a_i}$;
04. $\mathcal{C}$ chooses random $\eta \in \{0,1\}$;
05. **if** $\eta = 1$ **then** $\mathcal{C}$ sets $Z = U_{f_\ell}^{a_{f_\ell}}$ **else** $\mathcal{C}$ chooses random $Z \in \mathbb{G}$;

06. $\mathcal{C}$ runs $\mathcal{A}$ on input $[\mathcal{I}, F, (U_i)_{i\in F}, (\bar{U}_i)_{i\in F\setminus\{f_1\}}, (V_i)_{i\in F\setminus\{f_\ell\}}, Z]$;
07. Let η' be $\mathcal{A}$'s guess for η;
08. **if** $\eta = \eta'$ **then** return 1 **else** return 0.

Assumption 2 (F-LSS Assumption) *The F-Linear Secret Sharing Assumption states that for all probabilistic polynomial-time algorithms $\mathcal{A}$,*
$\left|\text{Prob}[F\text{-}\mathsf{LSSExp}_{\mathcal{A}}(1^n, 1^\ell) = 1] - 1/2\right|$ *is negligible in n for all $\ell = \mathsf{poly}(n)$.*

The proof of the following theorem is similar to, but simpler than, the proof of Theorem 2. So, we omit it.

Theorem 1 *For any sequences F and K each of ℓ distinct elements from $[2\ell - 1]$, F-LSS implies K-LSS.*

3.3 Split Linear Secret Sharing Assumption

In this section we present the Split Linear Secret Sharing Assumption (the SplitLSS Assumption) which is similar to the Linear Secret Sharing Assumption. The only difference is that whereas in the LSS Assumption the task is to decide whether $V_j = U_j^{a_j}$ or V_j is random, here the task is to decide, whether $Z = W^{u-a_j}$ or Z is a random element of $\mathbb{G}$. We formalize the SplitLSS Assumption by means of the following game between a Challenger $\mathcal{C}$ and an Adversary $\mathcal{A}$.

$\mathsf{SplitLSSExp}_{\mathcal{A}}(1^n, 1^\ell)$
01. $\mathcal{C}$ computes shares $a_1, \ldots, a_{2\ell-1}$ of 0 using an $(\ell, 2\ell - 1)$ LSSS;
02. $\mathcal{C}$ chooses instance $\mathcal{I} = [p, \mathbb{G}, \mathbb{G}_T, g, \mathsf{e}]$ with security parameter 1^n;
03. $\mathcal{C}$ chooses random $u, w \in \mathbb{Z}_p$ and sets $W = g^w$;
04. **for** $i \in [2\ell - 1]$
    $\mathcal{C}$ chooses random $u_i \in \mathbb{Z}_p$ and sets $U_i = g^{u_i}, V_i = U_i^{a_i}$,
    $A_i = g^{a_i}$, and $S_i = U_i^u$;
05 $\mathcal{C}$ picks a random $j \in [2\ell - 1]$ and sets $\hat{U} = U_j^w$;
06. $\mathcal{C}$ chooses random $\eta \in \{0, 1\}$;
07. **if** $\eta = 1$ **then** $\mathcal{C}$ sets $Z = W^{u-a_j}$ **else** $\mathcal{C}$ chooses random $Z \in \mathbb{G}$;
08. $\mathcal{C}$ runs $\mathcal{A}$ on input
    $[\mathcal{I}, j, (U_i)_{i\in[2\ell-1]}, (V_i)_{i\in[2\ell-1]}, (A_i)_{i\in[2\ell-1]}, (S_i)_{i\in[2\ell-1]\setminus\{j\}}, W, \hat{U}, Z]$;
09. Let η' be $\mathcal{A}$'s guess for η;
10. **if** $\eta = \eta'$ **then** return 1 **else** return 0.

Assumption 3 (SplitLSS Assumption) *The Split Linear Secret Sharing Assumption states that for all probabilistic polynomial-time algorithms* $\mathcal{A}$, $\left|\text{Prob}[\mathsf{SplitLSSExp}_{\mathcal{A}}(1^n, 1^\ell) = 1] - 1/2\right|$ *is negligible in* n *for all* $\ell = \mathsf{poly}(n)$.

3.4 F-Split Linear Secret Sharing Assumption

Let $F = \langle f_1, \ldots, f_\ell \rangle$ be a sequence of ℓ distinct elements from $[2\ell - 1]$. We formalize the F-Split Linear Secret Sharing Assumption (F-SplitLSS Assumption) by means of the following game between $\mathcal{C}$ and $\mathcal{A}$.

F-$\mathsf{SplitLSSExp}_{\mathcal{A}}(1^n, 1^\ell)$
01. $\mathcal{C}$ computes shares $a_1, \ldots, a_{2\ell-1}$ of 0 using an $(\ell, 2\ell - 1)$ LSSS;
02. $\mathcal{C}$ chooses instance $\mathcal{I} = [p, \mathbb{G}, \mathbb{G}_T, g, \mathsf{e}]$ with security parameter 1^n;
03. $\mathcal{C}$ chooses random $u \in \mathbb{Z}_p$;
04. **for** $i \in F$,
    $\mathcal{C}$ chooses random $u_i \in \mathbb{Z}_p$ and sets $U_i = g^{u_i}$, $\bar{U}_i = g^{1/u_i}$, $V_i = U_i^{a_i}$, and $S_i = U_i^u$;
05. $\mathcal{C}$ chooses random $w \in \mathbb{Z}_p$ and sets $W = g^w$ and $\bar{W} = g^{1/w}$.
06. $\mathcal{C}$ chooses random $\eta \in \{0, 1\}$;
07. **if** $\eta = 1$ **then** $\mathcal{C}$ sets $Z = W^{u - a_{f_\ell}}$ **else** $\mathcal{C}$ chooses random $Z \in \mathbb{G}$;
08. $\mathcal{C}$ runs $\mathcal{A}$ on input
    $[\mathcal{I}, F, (U_i)_{i \in F}, (\bar{U}_i)_{i \in F \setminus \{f_1\}}, (V_i)_{i \in F}, (S_i)_{i \in F}, W, \bar{W}, Z]$;
09. Let η' be $\mathcal{A}$'s guess for η.
10. **if** $\eta = \eta'$ **then** return 1 **else** return 0.

Assumption 4 (F-SplitLSS Assumption)) *The F-Split Linear Secret Sharing Assumption states that for all probabilistic polynomial-time algorithms* $\mathcal{A}$ $\left|\text{Prob}[F\text{-}\mathsf{SplitLSSExp}_{\mathcal{A}}(1^n, 1^\ell) = 1] - 1/2\right|$ *is negligible in* n *for all* $\ell = \mathsf{poly}(n)$.

Theorem 2 *For any two sequences* F *and* K *each of* ℓ *distinct elements from* $[2\ell - 1]$, *we have that* F*-SplitLSS implies* K*-SplitLSS.*

Proof. Let $F = \langle f_1, \ldots, f_\ell \rangle$ and $K = \langle k_1, \ldots, k_\ell \rangle$ be sequences of ℓ distinct elements from $[2\ell - 1]$. Given an F-SplitLSS instance

$$[\mathcal{I}, F, (U_j)_{j \in F}, (\bar{U}_j)_{j \in F \setminus \{f_1\}}, (V_j)_{j \in F}, (S_j)_{j \in F}, W, \bar{W}, Z]$$

we show how to get from it a K-SplitLSS instance

$$[\mathcal{I}, K, (U'_j)_{j \in K}, (\bar{U}'_j)_{j \in K \setminus \{k_1\}}, (V'_j)_{j \in K}, (S'_j)_{j \in K}, W', \bar{W}', Z'].$$

For $i = 1, \ldots, \ell$, let $\alpha_i = v_{f_i,F}/v_{k_i,K}$, where $v_{f_i,F}$ ($v_{k_i,K}$) is the publicly known value associated to f_i-th (k_i-th) share when participants whose identities are in F (resp., K) collaborate to the reconstruction of the secret in an $(\ell, 2\ell-1)$ LSSS. Set $U'_{k_1} = U_{f_1}$. For $i = 2, \ldots, \ell$, set

$$U'_{k_i} = U_{f_i} \text{ and } \bar{U}'_{k_i} = \bar{U}_{f_i}.$$

For $i = 1, \ldots, \ell$, set

$$V'_{k_i} = V_{f_i}^{\alpha_i} \text{ and } S'_{k_i} = S_{f_i}^{\alpha_\ell}.$$

Set $W' = W$ and $\bar{W}' = \bar{W}$. Finally, set $Z' = Z^{\alpha_\ell}$. It is immediate to see that the values $(U'_j)_{j\in K}, (\bar{U}'_j)_{j\in K\setminus\{k_1\}}, (V'_j)_{j\in K}, (S'_j)_{j\in K}, W', \bar{W}', Z'$ define a K-SplitLSS instance. □

4 The Scheme

In this section, we describe a new proposal for a secure predicate encryption scheme with partial public keys. Our description is for binary alphabets; it is possible to convert our scheme to a scheme for any alphabet by increasing the size of the key, but not the size of ciphertexts and tokens.

The Setup procedure. On input security parameter 1^n and the number of attributes $\ell = \mathsf{poly}(n)$, Setup proceeds as follows.

1. Select a symmetric bilinear instance $\mathcal{I} = [p, \mathbb{G}, \mathbb{G}_T, g, \mathsf{e}]$ with $|p| = \Theta(n)$.
2. For $i \in [2\ell-1]$, choose random $t_{1,i,0}, t_{2,i,0}, t_{1,i,1}, t_{2,i,1} \in \mathbb{Z}_p$ and set

$$\mathsf{K}_i = \begin{pmatrix} T_{1,i,0} = g^{t_{1,i,0}}, & T_{2,i,0} = g^{t_{2,i,0}} \\ T_{1,i,1} = g^{t_{1,i,1}}, & T_{2,i,1} = g^{t_{2,i,1}} \end{pmatrix} \text{ and}$$

$$\bar{\mathsf{K}}_i = \begin{pmatrix} \bar{T}_{1,i,0} = g^{1/t_{1,i,0}}, & \bar{T}_{2,i,0} = g^{1/t_{2,i,0}} \\ \bar{T}_{1,i,1} = g^{1/t_{1,i,1}}, & \bar{T}_{2,i,1} = g^{1/t_{2,i,1}} \end{pmatrix}.$$

3. Return $\mathsf{SK} = [\mathcal{I}, (\mathsf{K}_i, \bar{\mathsf{K}}_i)_{i\in[2\ell-1]}]$.

The $\mathsf{PPKeyGen}$ procedure. On input SK and policy $\mathsf{Pol} = \langle \mathsf{Pol}_1, \ldots, \mathsf{Pol}_\ell \rangle \in (2^{\{0,1\}}\setminus\emptyset)^\ell$ of length ℓ, $\mathsf{PPKeyGen}$ proceeds as follows.

1. For $i = 1, \ldots, \ell$,
   for every $b \in \mathsf{Pol}_i$, add $T_{1,i,b}$ and $T_{2,i,b}$ to PPK_i.
2. For $i = \ell+1, \ldots, 2\ell-1$,
   add $T_{1,i,0}$ and $T_{2,i,0}$ to PPK_i.

3. Return $\mathsf{PPK}_{\mathsf{Pol}} = [(\mathsf{PPK}_i)_{i\in[2\ell-1]}]$.

The Encryption **procedure.** On input partial public key $\mathsf{PPK}_{\mathsf{Pol}}$ and attribute vector $\vec{x} = (x_1, \ldots, x_\ell)$ of length ℓ, Encryption proceeds as follows.

1. If $\vec{x} \notin \mathbb{X}_{\mathsf{Pol}}$ return $\perp$.
2. Extend $\vec{x}$ to a vector with $2\ell-1$ entries by appending $(\ell-1)$ 0-entries.
3. Pick s at random from $\mathbb{Z}_p$.
4. Compute shares $(s_1, \ldots, s_{2\ell-1})$ of 0 using an $(\ell, 2\ell-1)$ linear secret sharing scheme.
5. For $i = 1, \ldots, 2\ell-1$,
   set $X_{1,i} = T_{1,i,x_i}^{s-s_i}$ and $X_{2,i} = T_{2,i,x_i}^{-s_i}$.
6. Return the encoded attribute vector $\tilde{X} = [(X_{1,i}, X_{2,i})_{i\in[2\ell-1]}]$.

Notice that if $\vec{x} \in \mathbb{X}_{\mathsf{Pol}}$, then for every i it holds that $T_{1,i,x_i}, T_{2,i,x_i} \in \mathsf{PPK}_{\mathsf{Pol}}$. Hence, the Encryption procedure will be able to execute the steps above.
In the following will use sometimes the writing
$\mathsf{Encryption}(\mathsf{PPK}_{\mathsf{Pol}}, \vec{x}; s, (s_i)_{i\in[2\ell-1]})$ to denote the encoded attribute vector $\tilde{X}$ output by Encryption on input $\mathsf{PPK}_{\mathsf{Pol}}$ and $\vec{x}$ when using s as random element and $(s_i)_{i\in[2\ell-1]}$ as shares of an $(\ell, 2\ell-1)$ linear secret sharing scheme for the secret 0.

The GenToken **procedure.** On input secret key SK and pattern vector $\vec{y} = (y_1, \ldots, y_\ell)$ of length ℓ, GenToken proceeds as follows. Set $P_{\vec{y}} = \{i \mid y_i = \star\}$.

1. Let h be the number of non-$\star$ entries of $\vec{y}$. Extend $\vec{y}$ to a vector with $(2\ell-1)$ entries by appending $(\ell-h)$ 0-entries and $(h-1)$ $\star$-entries and denote by $S_{\vec{y}}$ the indices of the non-$\star$ entries of the extended vector. Notice that $|S_{\vec{y}}| = \ell$.
2. Compute shares $(r_1, \ldots, r_{2\ell-1})$ of 0 using an $(\ell, 2\ell-1)$ linear secret sharing scheme.
3. Pick random $r \in \mathbb{Z}_p$.
4. For $i \in S_{\vec{y}}$,
   set $Y_{1,i} = \bar{T}_{1,i,y_i}^{r_i}$ and $Y_{2,i} = \bar{T}_{2,i,y_i}^{r-r_i}$.
5. Return $T_{\vec{y}} = [S_{\vec{y}}, (Y_{1,i}, Y_{2,i})_{i\in S_{\vec{y}}}]$ and $P_{\vec{y}}$.

In the following we will sometimes use the writing
$\mathsf{GenToken}(\mathsf{SK}, \vec{y}; r, (r_i)_{i\in S_{\vec{y}}})$ to denote the token $T_{\vec{y}}$ computed by GenToken on input SK and $\vec{y}$ and using r as random element and $(r_i)_{i\in S_{\vec{y}}}$ as ℓ shares of an $(\ell, 2\ell-1)$ LSSS for the secret 0.

The Test **procedure.** On input token $T_{\vec{y}} = [S, (Y_{1,j_1}, Y_{2,j_1}, \ldots, Y_{1,j_\ell}, Y_{2,j_\ell})]$ and attribute vector $\tilde{X} = [(X_{1,i}, X_{2,i})_{i\in[2\ell-1]}]$, Test proceeds as follows. Let $v_{j_1}, \ldots, v_{j_\ell}$ be the reconstruction coefficients for the set $S = \{j_1, \ldots, j_\ell\}$.

Then, the Test procedure returns

$$\prod_{i\in[\ell]} [\mathsf{e}(X_{1,j_i}, Y_{1,j_i}) \cdot \mathsf{e}(X_{2,j_i}, Y_{2,j_i})]^{v_{j_i}}.$$

The proof of next theorem is found in Appendix **??**.

Theorem 3 *The quintuple of algorithms* (Setup, PPKeyGen, Encryption, GenToken, Test) *specified above is a predicate encryption scheme with partial public keys.*

5 Semantic Security

In this section, we show that, if the Linear Secret Sharing Assumption and the Split Linear Secret Sharing Assumption hold, then the scheme presented in Section 4 is semantically secure. Specifically, we show that, for any attribute vector $\vec{z}$ and for any policy Pol, the encoded attribute vector output by the Encryption procedure is indistinguishable from a sequence of $2\cdot(2\ell-1)$ random elements of $\mathbb{G}$ to a polynomial time adversary $\mathcal{A}$ that has the partial public key associated with Pol and oracle access to GenToken for all pattern vectors $\vec{y}$ such that $\mathsf{Match}(\vec{z}, \vec{y}) = 0$. As it is easily seen, this implies semantic security.

The experiments We start by describing 3ℓ experiments with a probabilistic polynomial-time adversary $\mathcal{A}$.

Experiment k **with** $0 \le k \le 2\ell - 1$**.** In this experiment, $\mathcal{A}$ outputs an attribute vector $\vec{z}$ and a policy Pol, receives the partial public key $\mathsf{PPK}_{\mathsf{Pol}}$ relative to Pol, and has oracle access to GenToken for all pattern vectors $\vec{y}$ such that $\mathsf{Match}(\vec{z}, \vec{y}) = 0$. Then $\mathcal{A}$ receives challenge $\tilde{X} = [(X_{1,i}, X_{2,i})_{i\in[2\ell-1]}]$ computed as follows and outputs a bit.

1. Extend $\vec{z}$ to a $2\ell - 1$ vector by appending $(\ell - 1)$ 0-entries.
2. Compute shares $(s_1, \ldots, s_{2\ell-1})$ of 0 using an $(\ell, 2\ell - 1)$ LSSS.
3. For $i = 1, \ldots, k$, randomly choose $X_{1,i} \in \mathbb{G}$ and set $X_{2,i} = T_{2,i,z_i}^{s_i}$.
4. For $i = k + 1, \ldots, 2\ell - 1$, set $X_{1,i} = T_{1,i,z_i}^{s-s_i}$ and $X_{2,i} = T_{2,i,z_i}^{s_i}$.

Experiment $2\ell + k - 1$ **with** $k \in [\ell]$**.** These experiments differ from the previous ones only in the way in which the challenge $\tilde{X}$ is computed. More precisely, $\tilde{X} = [(X_{1,i}, X_{2,i})_{i\in[2\ell-1]}]$ is computed as follows.

1. Extend $\vec{z}$ to a $2\ell-1$ vector by appending $(\ell-1)$ 0-entries.
2. Compute shares $(s_1,\ldots,s_{2\ell-1})$ of 0 using an $(\ell, 2\ell-1)$ LSSS.
3. For $i=1,\ldots,k$ randomly choose $X_{1,i}, X_{2,i} \in \mathbb{G}$.
4. For $i=k+1,\ldots,2\ell-1$ randomly choose $X_{1,i} \in \mathbb{G}$ and set $X_{2,i} = T_{2,i,z_i}^{s_i}$.

Clearly, in Experiment 0, vector $\tilde{X}$ is a well-formed encryption of $\vec{z}$ whereas in Experiment $3\ell-1$ vector $\tilde{X}$ consists instead of randomly chosen elements from $\mathbb{G}$. We denote by $p_k^{\mathcal{A}}$ the probability that $\mathcal{A}$ outputs 1 when playing Experiment k. We start by proving that, under the Split Linear Secret Sharing Assumption, the difference $|p_k^{\mathcal{A}} - p_{k-1}^{\mathcal{A}}|$ is negligible, for $k \in [2\ell-1]$.

Due to space limit, some proofs are omitted and they can be found in the full version of this paper [BIP10].

Indistiguishability of the first $2\ell-1$ experiments.

Lemma 1 *Assume the Split Linear Secret Sharing Assumption. Then, for $k \in [2\ell-1]$, it holds that $|p_k^{\mathcal{A}} - p_{k-1}^{\mathcal{A}}|$ is negligible for all probabilistic polynomial-time adversaries $\mathcal{A}$.*

Proof. Assume, for sake of contradiction, that there exist a probabilistic polynomial-time $\mathcal{A}$ and an index k for which the lemma does not hold. We construct a successful adversary $\mathcal{B}$ for experiment $\mathsf{SplitLSSExp}$. $\mathcal{B}$ receives instance $[\mathcal{I}, j, (U_i)_{i\in[2\ell-1]}, (V_i)_{i\in[2\ell-1]}, (A_i)_{i\in[2\ell-1]}, (S_i)_{i\in[2\ell-1]\setminus\{j\}}, W, \hat{U}, Z]$ of the SplitLSS assumption from the challenger with $t=\ell$. We denote by $(a_1,\ldots,a_{2\ell-1})$ the shares of 0 associated with the instance and by u and w the secret exponents used by the challenger to compute the S_i's and W.

If $j \neq k$ then $\mathcal{B}$ outputs a random bit. Otherwise, $\mathcal{B}$ executes the instructions specified below. As we shall see, these instructions will have the effect that, depending on whether $Z = W^{u-a_j}$ or Z is random in $\mathbb{G}$, $\mathcal{B}$ simulates Experiment $k-1$ or Experiment k for $\mathcal{A}$ and $\mathcal{A}$'s guess (which is assumed to be significatly better than random) is used by $\mathcal{B}$ to break the Split Linear Secret Sharing Assumption. It is not difficult to see that if $|p_k^{\mathcal{A}} - p_{k-1}^{\mathcal{A}}| \geq 1/\mathsf{poly}(n)$ then $\mathcal{B}$ has probability at least $1/2 + 1/(2\cdot\ell\cdot\mathsf{poly}(n))$ of winning experiment $\mathsf{SplitLSSExp}$.

$\mathcal{B}$ starts by running $\mathcal{A}$ and receives $\vec{z}$ and Pol. For sake of ease of exposition we assume that $\vec{z} = 0^\ell$. It is straightforward to adapt the proof for general $\vec{z}$.

Key-generation Phase. For $i \in [2\ell-1]$, $\mathcal{B}$ picks random $t'_{1,i,0}, t'_{1,i,1}, t'_{2,i,0}, t'_{2,i,1} \in \mathbb{Z}_p$. For $i \in [2\ell-1]\setminus\{k\}$, $\mathcal{B}$ sets $T_{1,i,0} = U_i^{t'_{1,i,0}}$, $T_{1,i,1} = g^{t'_{1,i,1}}$, $T_{2,i,0} = g^{t'_{2,i,0}}$ and $T_{2,i,1} = g^{t'_{2,i,1}}$.

Finally, $\mathcal{B}$ sets $T_{1,k,0} = W^{t'_{1,k,0}}, T_{1,k,1} = g^{t'_{1,k,1}}, T_{2,k,0} = U_k^{t'_{2,k,0}}$, and $T_{2,k,1} = g^{t'_{2,k,1}}$. Notice that, for $i \in [2\ell-1] \setminus \{k\}$, the above settings implicitly define $t_{1,i,0} = u_i t'_{1,i,0}$, $t_{1,i,1} = t'_{1,i,1}$, $t_{2,i,0} = t'_{2,i,0}$ and $t_{2,i,1} = t'_{2,i,1}$ and $t_{1,k,0} = w t'_{1,k,0}$, $t_{1,k,1} = t'_{1,k,1}$, $t_{2,k,0} = u_k t'_{2,k,0}$ and $t_{2,k,1} = t'_{2,k,1}$. In turn, these values implictly define the values $\bar{T}_{1,i,0}, \bar{T}_{1,i,1}, \bar{T}_{2,i,0}$, and $\bar{T}_{2,i,1}$ for $i \in [2\ell-1]$. Therefore, after this step, secret key $\mathsf{SK} = (\mathcal{I}, (\mathsf{K}_i, \bar{\mathsf{K}}_i)_{i\in[2\ell-1]})$ is implicitly defined. It is immediate to see that SK has the same distribution as a secret key given in output by Setup.

Computing $\mathsf{PPK}_{\mathsf{Pol}}$. $\mathcal{B}$ computes the partial public key $\mathsf{PPK}_{\mathsf{Pol}}$ for policy Pol and gives it to $\mathcal{A}$. We stress that the partial public key depends only on the values $T_{1,i,0}, T_{1,i,1}, T_{2,i,0}, T_{2,i,1}$ that are known to $\mathcal{B}$.

Answering $\mathsf{GenToken}$ Queries. We now describe how $\mathcal{B}$ answers $\mathcal{A}$'s queries to the oracle $\mathsf{GenToken}$ for vectors $\vec{y} \in \{0,1,\star\}^\ell$. $\mathcal{B}$ extends $\vec{y}$ to a vector of $(2\ell-1)$ entries by appending 0-entries and $\star$-entries so that the resulting vector has exactly $(\ell-1)$ $\star$-entries and exactly ℓ entries in $\{0,1\}$. With a slight abuse of notation, we call $\vec{y}$ the extended vector and denote by $S_{\vec{y}}$ the set of indices i for which $y_i \in \{0,1\}$. Notice that by construction $|S_{\vec{y}}| = \ell$. Let $h \in [2\ell-1]$ be such that $y_h = 1$. If no such h exists then $\mathsf{Match}(\vec{z}, \vec{y}) = 1$ and $\mathcal{B}$ returns $\perp$. Otherwise, $\mathcal{B}$ computes token $T_{\vec{y}}$ in the following way.

$\mathcal{B}$ starts by picking a random $t \in \mathbb{Z}_p$. Then, for $i \in S_{\vec{y}} \setminus \{h,k\}$, $\mathcal{B}$ picks random $t_i \in \mathbb{Z}_p$ and sets

$$Y_{1,i} = \begin{cases} g^{t_i/t'_{1,i,0}}, & \text{if } y_i = 0; \\ U_i^{t_i/t'_{1,i,1}}, & \text{if } y_i = 1; \end{cases} \quad \text{and} \quad Y_{2,i} = U_k^{t/t'_{2,i,y_i}} \cdot U_i^{-t_i/t'_{2,i,y_i}}.$$

Simple algebraic manipulations show that, by the above settings, we have that, for $i \in S_{\vec{y}} \setminus \{h,k\}$,

$$Y_{1,i} = \bar{T}_{1,i,y_i}^{r_i} \quad \text{and} \quad Y_{2,i} = \bar{T}_{2,i,y_i}^{r-r_i}$$

for $r_i = t_i \cdot u_i$ and $r = t \cdot u_k$. We remark that $\mathcal{B}$ does not know r and the r_i's and that r and the r_i's are randomly chosen from $\mathbb{Z}_p$.

Furthermore, if $k \in S_{\vec{y}}$, $\mathcal{B}$ picks random $t_k \in \mathbb{Z}_p$ and sets

$$Y_{1,k} = \begin{cases} U_k^{t_k/t'_{1,k,0}}, & \text{if } y_k = 0; \\ \hat{U}^{t_k/t'_{1,k,1}}, & \text{if } y_k = 1; \end{cases} \quad \text{and} \quad Y_{2,k} = \begin{cases} g^{t/t'_{2,k,0}} \cdot W^{-t_k/t'_{2,k,0}}, & \text{if } y_k = 0; \\ U_k^{t/t'_{2,k,1}} \cdot \hat{U}^{-t_k/t'_{2,k,1}}, & \text{if } y_k = 1. \end{cases}$$

As before, we can see that

$$Y_{1,k} = \bar{T}_{1,k,y_k}^{r_k} \text{ and } Y_{2,k} = \bar{T}_{2,k,y_k}^{r-r_k}$$

for $r_k = w \cdot t_k \cdot u_k$ and $r = t \cdot u_k$. Also we observe that r_k is randomly distributed in $\mathbb{Z}_p$.

Finally, we are left with computing $Y_{1,h}$ and $Y_{2,h}$. We observe that our construction dictates that

$$Y_{1,h} = \bar{T}_{1,h,1}^{r_h} \qquad \text{and} \qquad Y_{2,h} = \bar{T}_{2,h,1}^{r-r_h} \tag{1}$$

for r_h such that the sequence of $(r_i)_{i \in S_{\vec{y}}}$ is a sequence of ℓ shares of 0 with respect to the underlying $(\ell, 2\ell - 1)$ LSSS. For the linearity of the secret sharing scheme, there exist publicly-known reconstructing coefficients α_i such that

$$r_h = -\frac{1}{\alpha_h} \sum_{i : S_{\vec{y}} \setminus \{h\}} \alpha_i r_i.$$

We stress that the α_i's only depend on the set $S_{\vec{y}}$. Thus, if $k \in S_{\vec{y}}$, $\mathcal{B}$ sets

$$Y_{1,h} = \hat{U}^{-\alpha_k/\alpha_h \cdot t_k / t'_{1,h,1}} \prod_{i \in S_{\vec{y}} \setminus \{h,k\}} U_i^{-\alpha_i/\alpha_h \cdot t_i / t'_{1,h,1}}$$

and

$$Y_{2,h} = U_k^{t/t'_{2,h,1}} \cdot \hat{U}^{-\alpha_k/\alpha_h \cdot t_k / t'_{2,h,1}} \cdot \prod_{i \in S_{\vec{y}} \setminus \{h,k\}} U_i^{-\alpha_i/\alpha_h \cdot t_i / t'_{2,h,1}}$$

otherwise $\mathcal{B}$ sets

$$Y_{1,h} = \prod_{i \in S_{\vec{y}} \setminus \{h\}} U_i^{-\alpha_i/\alpha_h \cdot t_i / t'_{1,h,1}}$$

and

$$Y_{2,h} = U_k^{t/t'_{2,h,1}} \cdot \prod_{i \in S_{\vec{y}} \setminus \{h\}} U_i^{-\alpha_i/\alpha_h \cdot t_i / t'_{2,h,1}}$$

Simple algebraic manipulations show that, in both cases, $Y_{1,h}$ and $Y_{2,h}$ satisfy Equation (1).

We can thus conclude that the replies computed by $\mathcal{B}$ for $\mathcal{A}$'s GenToken queries are correctly computed.

Challenge construction. When $\mathcal{B}$ is asked to provide encrypted attribute vector for $\vec{z}$, $\mathcal{B}$ constructs the tuple $\tilde{X} = [(X_{1,i}, X_{2,i})_{i \in [2\ell-1]}]$ in the following way.

1. For $i = 1, \ldots, k-1$,: $\mathcal{B}$ chooses random $X_{1,i} \in \mathbb{G}$ and sets $X_{2,i} = A_i^{-t'_{2,i,0}}$.
2. $\mathcal{B}$ sets $X_{1,k} = Z^{t'_{1,k,0}}$ and $X_{2,k} = V_k^{-t'_{2,k,0}}$.

3. For $i = k+1, \ldots, 2\ell-1$,: $\mathcal{B}$ sets $X_{1,i} = S_i^{t'_{1,i,0}} \cdot V_i^{-t'_{1,i,0}}$ and $X_{2,i} = A_i^{-t'_{2,i,0}}$.

We have the following simple observations:

1. For $i \in [2\ell-1]$, we have that $X_{2,i} = T_{2,i,0}^{-a_i}$.
2. The sequence $(a_1, \ldots, a_{2\ell-1})$ comprises the shares of 0 with respect to an $(\ell, 2\ell-1)$ LSSS (just as in Experiment k and Experiment $k-1$).
3. For $i \in [k-1]$, $X_{1,i}$ is a random element of $\mathbb{G}$ (just as in Experiment k and Experiment $k-1$).
4. For $i = k+1, \ldots, 2\ell-1$, $X_{1,i} = T_{1,i,0}^{u-a_i}$, for a randomly chosen $u \in \mathbb{Z}_p$ (just as in Experiment k and Experiment $k-1$).
5. If $Z = W^{u-a_k}$, then $X_{1,k} = T_{1,k,0}^{u-a_k}$ (just as in Experiment $k-1$).
6. If Z is random in $\mathbb{G}$, then $X_{1,k}$ is also random in $\mathbb{G}$ (just as in Experiment k).

By the above observations, we have that if $Z = W^{u-a_k}$, then the view of $\mathcal{A}$ in the interaction with $\mathcal{B}$ is exactly the same as the view of $\mathcal{A}$ in Experiment $k-1$; on the other hand, if Z is random in $\mathbb{G}$ then the view of $\mathcal{A}$ in the interaction with $\mathcal{B}$ is exactly the same as the view of $\mathcal{A}$ in Experiment k. The lemma then follows. ∎

Indistiguishability of the last ℓ experiments.

Lemma 2 *Assume the Linear Secret Sharing Assumption. Then, for $k \in [\ell]$, it holds that $|p^{\mathcal{A}}_{2\ell+k-2} - p^{\mathcal{A}}_{2\ell+k-1}|$ is negligible for all probabilistic polynomial-time adversaries $\mathcal{A}$.*

Proof. Assume, for sake of contradiction, that there exist a probabilistic polynomial-time $\mathcal{A}$ and an index k for which the lemma does not hold. We construct a successful adversary $\mathcal{B}$ for experiment LSSExp. $\mathcal{B}$ receives instance $[\mathcal{I}, j, (U_i)_{i\in[2\ell-1]}, (V_i)_{i\in[2\ell-1]\setminus\{j\}}, Z]$ of the LSS assumption from the challenger with $t = \ell$. We denote by $(a_1, \ldots, a_{2\ell-1})$ the shares of 0 associated with the instance.

If $j \neq k$ then $\mathcal{B}$ outputs a random bit. Otherwise, $\mathcal{B}$ executes the instructions specified below. As we shall see, these instruction will have the effect that, depending on whether $Z = U_j^{a_j}$ or Z is random in $\mathbb{G}$, $\mathcal{B}$ simulates Experiment $k-1$ or Experiment k for $\mathcal{A}$. $\mathcal{A}$'s guess is then output by $\mathcal{B}$. It is not difficult to see that, if $|p^{\mathcal{A}}_k - p^{\mathcal{A}}_{k-1}| \geq 1/\mathsf{poly}(n)$ then $\mathcal{B}$ has probability at least $1/2 + 1/(2 \cdot \ell \cdot \mathsf{poly}(n))$ of winning experiment LSSExp.

We next describe $\mathcal{B}$'s instruction for the case $j = k$. $\mathcal{B}$ starts by running $\mathcal{A}$ and receives $\vec{z}$ and Pol. For sake of ease of exposition we assume that

$\vec{z} = 0^\ell$. It is straightforward to adapt the proof for general $\vec{z}$.

Key-generation Phase. For $i \in [2\ell-1]$, $\mathcal{B}$ picks random $t'_{1,i,0}, t'_{1,i,1}, t'_{2,i,0}, t'_{2,i,1} \in \mathbb{Z}_p$ and sets $T_{1,i,0} = g^{t'_{1,i,0}}, T_{1,i,1} = g^{t'_{1,i,1}}, T_{2,i,0} = U_i^{t'_{2,i,0}}$ and $T_{2,i,1} = g^{t'_{2,i,1}}$. Notice that, for $i \in [2\ell-1]$, such a settings implicitly define $t_{1,i,0} = t'_{1,i,0}, t_{1,i,1} = t'_{1,i,1}, t_{2,i,0} = u_i t'_{2,i,0}$, and $t_{2,i,1} = t'_{2,i,1}$. which in turn define the values $\bar{T}_{1,i,0}, \bar{T}_{1,i,1}, \bar{T}_{2,i,0}$ and $\bar{T}_{2,i,1}$. Therefore, after this step, secret key $\mathsf{SK} = (\mathcal{I}, (\mathsf{K}_i, \bar{\mathsf{K}}_i)_{i\in[2\ell-1]})$ is implicitly defined. It is immediate to see that SK has the same distribution as a secret key given in output by Setup.

Computing $\mathsf{PPK}_{\mathsf{Pol}}$. $\mathcal{B}$ computes the partial public key $\mathsf{PPK}_{\mathsf{Pol}}$ for policy Pol and gives it to $\mathcal{A}$. We stress that the partial public key depends only from the values $T_{1,i,0}, T_{1,i,1}, T_{2,i,0}, T_{2,i,1}$ which are known to $\mathcal{B}$.

Answering $\mathsf{GenToken}$ Queries. We now describe how $\mathcal{B}$ answers $\mathcal{A}$'s queries to the oracle $\mathsf{GenToken}$ for vector $\vec{y} \in \{0,1,\star\}^\ell$. $\mathcal{B}$ extends $\vec{y}$ to a vector of $(2\ell-1)$ entries by appending 0-entries and $\star$-entries so that the resulting vector has exactly $(\ell-1)$ $\star$-entries and exactly ℓ entries in $\{0,1\}$. With a slight abuse of notation, we call $\vec{y}$ the extended vector and denote by $S_{\vec{y}}$ the set of indices i for which $y_i \in \{0,1\}$. Notice that $|S_{\vec{y}}| = \ell$. Let $h \in [2\ell-1]$ be such that $y_h = 1$. If no such h exists then $\mathsf{Match}(\vec{z}, \vec{y}) = 1$ and $\mathcal{B}$ returns $\perp$. Otherwise, $\mathcal{B}$ computes token $T_{\vec{y}}$ is the following way.

$\mathcal{B}$ starts by picking a random $t \in \mathbb{Z}_p$. Then, for $i \in S_{\vec{y}} \setminus \{h\}$, $\mathcal{B}$ picks random $t_i \in \mathbb{Z}_p$ and sets

$$Y_{1,i} = g^{t/t'_{1,i,y_i}} \cdot U_i^{-t_i/t'_{1,i,y_i}} \qquad \text{and} \qquad Y_{2,i} = \begin{cases} g^{t_i/t'_{2,i,0}}, & \text{if } y_i = 0; \\ U_i^{t_i/t'_{2,i,1}}, & \text{if } y_i = 1; \end{cases}$$

Simple algebraic manipulations show that, for $i \in S_{\vec{y}} \setminus \{h\}$, $Y_{1,i} = \bar{T}_{1,i,y_i}^{r_i}$ and $Y_{2,i} = \bar{T}_{2,i,y_i}^{r-r_i}$ for $r = t$ and $r_i = t - t_i \cdot u_i$. We remark that $\mathcal{B}$ does not know the r_i's and that r and the r_i's are randomly chosen from $\mathbb{Z}_p$.

Finally, $\mathcal{B}$ is left with computing $Y_{1,h}$ and $Y_{2,h}$. We observe that our construction dictates that

$$Y_{1,h} = \bar{T}_{1,h,1}^{r_h} \text{ and } Y_{2,h} = \bar{T}_{2,h,1}^{r-r_h} \tag{2}$$

for r_h such that the sequence of $(r_i)_{i\in S_{\vec{y}}}$ is a sequence of ℓ shares of 0 with respect to the underlying $(\ell, 2\ell-1)$ LSSS. For the linearity of the secret sharing scheme, there exist publicly-known reconstructing coefficients α_i such that

$$r_h = -\frac{1}{\alpha_h} \sum_{i \in S_{\vec{y}} \setminus \{h\}} \alpha_i r_i.$$

We stress that the α_i's only depend on the set $\mathcal{S}_y$. Thus $\mathcal{B}$ sets

$$Y_{1,h} = g^{-\alpha_i/\alpha_h \cdot t/t'_{1,h,1}} \cdot \prod_{i \in \mathcal{S}_y \setminus \{h\}} U_i^{-\alpha_i/\alpha_h \cdot t_i/t'_{1,h,1}} \quad \text{and} \quad Y_{2,h} = \prod_{i \in \mathcal{S}_y \setminus \{h\}} U_i^{-\alpha_i/\alpha_h \cdot t_i/t'_{2,h,1}}.$$

Simple algebraic manipulation shows that by above settings $Y_{1,h}$ and $Y_{2,h}$ satisfy Equation (2).

We can thus conclude that the replies computed by $\mathcal{B}$ for $\mathcal{A}$'s GenToken queries are correctly computed.

Challenge construction. When $\mathcal{B}$ is asked to provide encrypted attribute vector for $\vec{z}$, $\mathcal{B}$ constructs the tuple $\tilde{X} = [(X_{1,i}, X_{2,i})_{i \in [2\ell-1]}]$ in the following way.

1. For $i = 1, \ldots, k-1$: $\mathcal{B}$ chooses random $X_{1,i}, X_{2,i} \in \mathbb{G}$.
2. $\mathcal{B}$ chooses random $X_{1,k} \in \mathbb{G}$ and sets $X_{2,k} = Z^{-t'_{2,k,0}}$.
3. For $i = k+1, \ldots, 2\ell-1$: $\mathcal{B}$ chooses random $X_{1,i} \in \mathbb{G}$ and sets $X_{2,i} = V_i^{-t'_{2,i,0}}$.

We have the following simple observations.

1. For $i \in [2\ell-1]$, $X_{1,i}$ is a random element of $\mathbb{G}$ (just as in Experiment $2\ell+k-1$ and in Experiment $2\ell+k-2$).
2. For $i \in [k-1]$, $X_{2,i}$ is random element of $\mathbb{G}$ (just as in Experiment $2\ell+k-1$ and in Experiment $2\ell+k-2$).
3. For $i = k+1, \ldots, 2\ell-1$, $X_{2,i} = V_i^{-t'_{2,i,0}} = U_i^{-a_i t'_{2,i,0}} = T_{2,i,0}^{-a_i}$
4. The sequence $(a_1, \ldots, a_{2\ell-1})$ comprises the shares of 0 with respect to an $(\ell, 2\ell-1)$ LSSS (just as in Experiment $2\ell+k-1$ and in Experiment $2\ell+k-2$).
5. If $Z = U_k^{a_k}$, then $X_{2,k} = T_{2,k}^{a_k}$ (just as in Experiment $2\ell+k-2$).
6. If Z is random in $\mathbb{G}$, then $X_{2,k}$ is also random (just as in Experiment $2\ell+k-1$).

Therefore, if $Z = U_k^{a_k}$, then the view of $\mathcal{A}$ in the interaction with $\mathcal{B}$ is exactly the same as the view of $\mathcal{A}$ in Experiment $2\ell+k-2$; on the other hand, if Z is random in $\mathbb{G}$, then the view of $\mathcal{A}$ in the interaction with $\mathcal{B}$ is exactly the same as the view of $\mathcal{A}$ in Experiment $2\ell+k-1$. The lemma then follows. □

Lemma 1 and Lemma 2 imply the following theorem.

Theorem 4 *Assume LSS and SplitLSS. Then, predicate encryption scheme with partial public keys*
(Setup, PPKeyGen, Encryption, GenToken, Test) *is semantically secure.*

6 Token Security

In this section, we show that, if the F-Linear Secret Sharing Assumption and the F-Split Linear Secret Sharing Assumption hold, the scheme presented in Section 4 is token secure. Specifically, let $\vec{z}$ be a pattern and Pol a policy such that $\mathbb{X}_{\mathsf{Pol}}$ does not contain any attribute vector $\vec{x}$ such that $\mathsf{Match}(\vec{x}, \vec{z}) = 1$. Then we show that no probabilistic polynomial-time adversary $\mathcal{A}$ that has oracle access to $\mathsf{GenToken}$ and the public key relative to Pol can distinguish a well formed token for pattern $\vec{z}$ from a sequence of random elements of $\mathbb{G}$. It is straightforward to see that this implies token security.

The experiments We start by describing 4ℓ experiments with a probabilistic polynomial-time adversary $\mathcal{A}$.

Experiment j with $0 \leq j \leq 2\ell - 1$. In this experiment, $\mathcal{A}$ outputs a pattern $\vec{z} \in \{0, 1, \star\}^\ell$ and a policy Pol, receives the partial public key $\mathsf{PPK}_{\mathsf{Pol}}$ relative to Pol and has oracle access to $\mathsf{GenToken}$ for all pattern vectors $\vec{y}$. If there exists an attribute vector $\vec{x} \in \mathbb{X}_{\mathsf{Pol}}$ such that $\mathsf{Match}(\vec{x}, \vec{z}) = 1$ then, $\mathcal{A}$ receives $\perp$; otherwise, $\mathcal{A}$ receives challenge $T_{\vec{z}}$ computed as follows. In both cases $\mathcal{A}$ outputs a bit.

1. Let h be the number of non-$\star$ entries of $\vec{z}$. Extend $\vec{z}$ to a vector with $(2\ell - 1)$ entries by appending $(\ell - h)$ 0-entries and $(h - 1)$ $\star$-entries. With a slight abuse of notation, we call $\vec{z}$ the extended vector and denote by $S_{\vec{z}}$ the set of indices i such that $z_i \in \{0, 1\}$. Notice that $|S_{\vec{z}}| = \ell$.
2. Choose random $r \in \mathbb{Z}_p$ and compute shares $(r_1, \ldots, r_{2\ell-1})$ of 0 using an $(\ell, 2\ell - 1)$ LSSS.
3. For $i \in S_{\vec{z}}$ and $i \leq j$, set $Y_{1,i} = g^{r_i / t_{1,i,z_i}}$ and $Y_{2,i} = g^{(r - r_i)/t_{2,i,z_i}}$.
4. For $i \in S_{\vec{z}}$ and $i > j$, set $Y_{1,i} = g^{r_i / t_{1,i,z_i}}$ and choose random $Y_{2,i} \in \mathbb{G}$.
5. Set $T_{\vec{z}} = [S_{\vec{z}}, (Y_{1,i}, Y_{2,i})_{i \in S_{\vec{z}}}]$.

Experiment j with $2\ell \leq j \leq 4\ell - 1$. The experiments differ from the previous ones only in the way the challenge $T_{\vec{z}}$ is computed. More precisely, the challenge $T_{\vec{z}}$ is computed as follows.

1. Let h be the number of non-$\star$ entries of $\vec{z}$. Extend $\vec{z}$ to a vector with $(2\ell - 1)$ entries by appending $(\ell - h)$ 0-entries and $(h - 1)$ $\star$-entries. With a slight abuse of notation, we call $\vec{z}$ the extended vector and denote by $S_{\vec{z}}$ the set of indices i such that $z_i \in \{0, 1\}$. Notice that $|S_{\vec{z}}| = \ell$.

2. Choose random $r \in \mathbb{Z}_p$ and compute shares $(r_1, \ldots, r_{2\ell-1})$ of 0 using an $(\ell, 2\ell-1)$ LSSS.
3. For $i \in S_{\vec{z}}$, set $Y_{2,i}$ to a random element in $\mathbb{G}$.
4. For $i \in S_{\vec{z}}$ and $i \leq j$, set $Y_{1,i} = g^{r_i/t_{1,i,z_i}}$.
5. For $i \in S_{\vec{z}}$ and $i > j$, set $Y_{1,i}$ to a random element in $\mathbb{G}$.
6. Set $T_{\vec{z}} = [S_{\vec{z}}, (Y_{1,i}, Y_{2,i})_{i \in S_{\vec{z}}}]$.

Clearly in Experiment 0, $T_{\vec{z}}$ is a well formed token for pattern $\vec{z}$ whereas in Experiment $4\ell - 1$, $T_{\vec{z}}$ consists of 2ℓ randomly chosen elements of $\mathbb{G}$. We denote by $p_j^{\mathcal{A}}$ the probability that $\mathcal{A}$ outputs 1 when playing Experiment j. We start by proving that, under the F-Linear Secret Sharing, the difference $|p_j^{\mathcal{A}} - p_{j-1}^{\mathcal{A}}|$ is negligible for $j \in [2\ell - 1]$.

Indistinguishability of the first 2ℓ experiments.

Lemma 3 *Assume F-Split Linear Secret Sharing holds. Then, for $j \in [2\ell-1]$, it holds that $|p_j^{\mathcal{A}} - p_{j-1}^{\mathcal{A}}|$ is negligible for all probabilistic polynomial-time adversary $\mathcal{A}$.*

Proof. Assume, for sake of contradiction, that there exist a probabilistic polynomial-time adversary $\mathcal{A}$ and $j \in [2\ell - 1]$ for which the lemma does not hold. We construct a successful adversary $\mathcal{B}$ for experiment F-$\mathsf{SplitLSSExp}$. $\mathcal{B}$ receives an istance
$[\mathcal{I}, F, (U_i)_{i \in F}, (\bar{U}_i)_{i \in F \setminus \{f_1\}}, (V_i)_{i \in F}, (S_i)_{i \in F}, W, \bar{W}, Z]$
of the F-Split Linear Secret Sharing Assumption and runs $\mathcal{A}$ receiving pattern $\vec{z} \in \{0, 1, \star\}$ and policy Pol. If there exists $\vec{x} \in \mathbb{X}_{\mathsf{Pol}}$ for which $\mathsf{Match}(\vec{x}, \vec{z}) = 1$ then $\mathcal{B}$ returns $\perp$ to $\mathcal{A}$ and outputs $\mathcal{A}$'s output. Otherwise $\mathcal{B}$ constructs token T in the following way.

$\mathcal{B}$ extends $\vec{z}$ to a vector with $(2\ell - 1)$ entries by appending 0-entries and $\star$-entries so that the resulting vector has exactly ℓ entries in $\{0, 1\}$. With a slight abuse of notation, we call $\vec{z}$ the extended vector and denote by $S_{\vec{z}}$ the set of indices i such that $z_i \in \{0, 1\}$. For ease of exposition, we assume that for $i \in S_{\vec{z}}$ we have $\vec{z}_i = 0$. It is straightforward to adapt the proof for general $\vec{z}$. We denote by k an index such that $k \in S_{\vec{z}}$ and $0 \notin \mathsf{Pol}_k$. Observe that such a k exists for otherwise there would exist $\vec{x} \in \mathbb{X}_{\mathsf{Pol}}$ that matches $\vec{z}$. We assume that $k \neq j$ (the proof for the case $k = j$ is simpler and is omitted). We further assume that $S_{\vec{z}}$ coincides with $F = \langle f_1, \ldots, f_\ell \rangle$, that $f_1 = k$, and that $f_\ell = j$. This is without loss of generality by Theorem 2. $\mathcal{B}$ starts by simulating the Key-generation Phase.

Key-generation Phase. For $i \in [2\ell-1]$, $\mathcal{B}$ chooses random $t'_{1,i,0}, t'_{1,i,1}, t'_{2,i,0}, t'_{2,i,1} \in \mathbb{Z}_p$. Then, for $i \notin S_{\vec{z}}$, $\mathcal{B}$ sets $T_{1,i,0} = g^{1/t'_{1,i,0}}, T_{1,i,1} = g^{1/t'_{1,i,1}}, T_{2,i,0} = g^{1/t'_{2,i,0}}$, and $T_{2,i,1} = g^{1/t'_{2,i,1}}$, for $i \in S_{\vec{z}} \setminus \{k, j\}$, $\mathcal{B}$ sets $T_{1,i,0} = \bar{U}_i^{1/t'_{1,i,0}}, T_{1,i,1} = g^{1/t'_{1,i,1}}, T_{2,i,0} = \bar{U}_i^{1/t'_{2,i,0}}, T_{2,i,1} = g^{1/t'_{2,i,1}}, \bar{T}_{1,i,0} = U_i^{t'_{1,i,0}}, \bar{T}_{1,i,1} = g^{t'_{1,i,1}}, \bar{T}_{2,i,0} = U_i^{t'_{2,i,0}}, \bar{T}_{2,i,1} = g^{t'_{2,i,1}}$, and, finally, $\mathcal{B}$ sets $T_{1,j,0} = \bar{U}_j, T_{1,j,1} = g^{1/t'_{1,j,1}}, T_{2,j,0} = \bar{W}, T_{2,j,1} = g^{1/t'_{2,j,1}}, T_{1,k,1} = g^{1/t'_{1,k,1}}, T_{2,k,1} = g^{1/t'_{2,k,1}}, \bar{T}_{1,j,0} = U_j, \bar{T}_{1,j,1} = g^{t'_{1,j,1}}, \bar{T}_{2,j,0} = W$, $\bar{T}_{2,j,1} = g^{t'_{2,j,1}}, \bar{T}_{1,k,0} = U_k^{t'_{1,k,0}}, \bar{T}_{1,k,1} = g^{t'_{1,k,1}}, \bar{T}_{2,k,0} = U_k^{t'_{2,k,0}}, \bar{T}_{2,k,1} = g^{t'_{2,k,1}}$. After this step secret key $\mathsf{SK} = [\mathcal{I}, (\mathsf{K}_i, \bar{\mathsf{K}}_i)_{i \in [2\ell-1]}]$ is *implicitly* defined. It is immediate to see that SK has the same distribution as a secret key output by Setup. We stress that $\mathcal{B}$ does not completely know SK since $\mathcal{B}$ has not computed $T_{1,k,0}$ and $T_{2,k,0}$ (which however are implicitly defined by U_k). However we observe that, since $0 \notin \mathsf{Pol}_k$, $\mathcal{B}$ can construct the partial public key $\mathsf{PPK}_{\mathsf{Pol}}$ for $\mathcal{A}$. In addition $\mathcal{B}$ knows all values $\bar{T}$ and can thus answer any $\mathsf{GenToken}$ query.

Challenge construction. When $\mathcal{B}$ is asked to provide token $T_{\vec{z}}$, $\mathcal{B}$ constructs the token $T_{\vec{z}} = [S_{\vec{z}}, (Y_{1,i}, Y_{2,i})_{i \in S_{\vec{z}}}]$ as follows. The construction is deterministic and uses the randomness present in challenge; specifically, u and the a_i's (see Steps 01 and 03 of the F-Split Linear Secret Sharing). As we shall see, the effect of the following instructions will be to construct a token in which $r = u$ and $r_i = a_i$, for $i \in [2\ell - 1]$,

1. For $i = 1, \ldots, j-1$: If $i \in S_z$, then $\mathcal{B}$ sets $Y_{1,i} = V_i^{t'_{1,i,0}}$ and $Y_{2,i} = S_i^{t'_{2,i,0}} \cdot V_i^{-t'_{2,i,0}}$;
2. $\mathcal{B}$ sets $Y_{1,j} = V_j$ and $Y_{2,j} = Z$;
3. For $i = j+1, \ldots, 2\ell - 1$: If $i \in S_z$, then $\mathcal{B}$ sets $Y_{1,i} = V_i^{t'_{1,i,0}}$ and chooses random $Y_{2,i} \in \mathbb{G}$.

We have the following simple observations.

1. For $i \in S_{\vec{z}} \setminus \{j\}$, we have that $Y_{1,i} = V_i^{t'_{1,i,0}} = \bar{T}_{1,i,0}^{a_i}$ (just as in Experiment j and Experiment $j-1$);
2. $Y_{1,j} = V_j = U_j^{a_j} = \bar{T}_{1,j,0}^{a_j}$ (just as in Experiment j and Experiment $j-1$);
3. For $i \in S_z$ and $i < j$, we have that $Y_{2,i} = S_i^{t'_{2,i,0}} \cdot V_i^{-t'_{2,i,0}} = U_i^{t'_{2,i,0}(u-a_i)} = \bar{T}_{2,i,0}^{u-a_i}$ (just as in Experiment j and Experiment $j-1$);
4. For $i \in S_z$ and $i > j$, we have that $Y_{2,i}$ is a random element of $\mathbb{G}$ (just as in Experiment j and Experiment $j-1$);
5. If $Z = W^{u-a_j}$, then $Y_{2,j} = \bar{T}_{2,j,0}^{u-a_j}$ and (just as in Experiment $j-1$);

6. If Z is random in $\mathbb{G}$, then $Y_{2,j}$ is random in $\mathbb{G}$ (just as in Experiment j).

By the above observations, we have that if $Z = W^{u-a_j}$, then the view of $\mathcal{A}$ in the interaction with $\mathcal{B}$ is exactly the same as the view of $\mathcal{A}$ in Experiment $j-1$; on the other hand, if Z is random, then the view of $\mathcal{A}$ in the interaction with $\mathcal{B}$ is exactly the same as the view of $\mathcal{A}$ in Experiment j. The lemma then follows. □

Indistinguishability of last 2ℓ experiments.

Lemma 4 *Assume F-Linear Secret Sharing holds. Then, for $j = 2\ell, \ldots, 4\ell - 1$, it holds that $|p_j^{\mathcal{A}} - p_{j-1}^{\mathcal{A}}|$ is negligible for all probabilistic polynomial-time adversary $\mathcal{A}$.*

Proof. Assume, for sake of contradiction, that there exist a probabilistic polynomial-time adversary $\mathcal{A}$ and an index j, where $2\ell \leq j \leq 4\ell - 1$, for which the lemma does not hold. We construct a successful adversary $\mathcal{B}$ for experiment F-LSSExp. In such an experiment, $\mathcal{B}$ receives an istance $[\mathcal{I}, F, (U_i)_{i\in F}, (\bar{U}_i)_{i\in F\setminus\{f_1\}}, (V_i)_{i\in F\setminus\{f_\ell\}}, Z]$ of the F-Linear Secret Sharing Assumption and runs $\mathcal{A}$ receiving pattern $\vec{z} \in \{0, 1, \star\}$ and policy Pol. If there exists $\vec{x} \in \mathbb{X}_{\mathsf{Pol}}$ for which $\mathsf{Match}(\vec{x}, \vec{z}) = 0$ then $\mathcal{B}$ returns $\perp$ to $\mathcal{A}$ and outputs $\mathcal{A}$'s output. Otherwise $\mathcal{B}$ constructs the token T in the following way. $\mathcal{B}$ extends $\vec{z}$ to a vector with $2\ell - 1$ entries by appending 0-entries and $\star$-entries so that the resulting vector has exactly ℓ entries in $\{0, 1\}$. With a slight abuse of notation, we call $\vec{z}$ the extended vector and denote by $S_{\vec{z}}$ the set of indices i such that $z_i \in \{0, 1\}$. For ease of exposition, we assume that for $i \in S_{\vec{z}}$ we have $\vec{z}_i = 0$. It is straightforward to adapt the proof for general $\vec{z}$. We denote by k an index such that $k \in S_{\vec{z}}$ and $0 \notin \mathsf{Pol}_k$. Observe that such a k exists for otherwise there exists $\vec{x} \in \mathbb{X}_{\mathsf{Pol}}$ that matches $\vec{z}$. We assume that $k \neq j$ (the proof for the case $k = j$ is simpler and we omit it). We further assume that $S_{\vec{z}}$ coincides with $F = \langle f_1, \ldots, f_\ell \rangle$, that $f_1 = k$, and that $f_\ell = j$. This is without loss of generality by Theorem 2. $\mathcal{B}$ starts by simulating the Key-generation Phase.

Key-generation Phase. For $i \in [2\ell-1]$, $\mathcal{B}$ chooses random $t'_{1,i,0}, t'_{1,i,1}, t'_{2,i,0}, t'_{2,i,1} \in \mathbb{Z}_p$. Then, for $i \notin S_{\vec{z}}$, $\mathcal{B}$ sets $T_{1,i,0} = g^{1/t'_{1,i,0}}, T_{1,i,1} = g^{1/t'_{1,i,1}}, T_{2,i,0} = g^{1/t'_{2,i,0}}$, and $T_{2,i,1} = g^{1/t'_{2,i,1}}$, for $i \in S_{\vec{z}}\setminus\{k, j\}$, $\mathcal{B}$ sets $T_{1,i,0} = \bar{U}_i^{1/t'_{1,i,0}}, T_{1,i,1} = g^{1/t'_{1,i,1}}, T_{2,i,0} = \bar{U}_i^{1/t'_{2,i,0}}, T_{2,i,1} = g^{1/t'_{2,i,1}}, \bar{T}_{1,i,0} = U_i^{t'_{1,i,0}}, \bar{T}_{1,i,1} = g^{t'_{1,i,1}}, \bar{T}_{2,i,0} = U_i^{t'_{2,i,0}}, \bar{T}_{2,i,1} = g^{t'_{2,i,1}}$, Finally, $\mathcal{B}$ sets $T_{1,j,0} = \bar{U}_j^{1/t'_{1,j,0}}, T_{1,j,1} = g^{1/t'_{1,j,1}}, T_{2,j,0} = \bar{U}_j^{1/t'_{1,j,0}}, T_{2,j,1} = g^{1/t'_{2,j,1}}, T_{1,k,1} = g^{1/t'_{1,k,1}}, T_{2,k,1} = g^{1/t'_{2,k,1}}, \bar{T}_{1,j,0} = U_j^{t'_{1,j,0}}, \bar{T}_{1,j,1} =$

$g^{t'_{1,j,1}}, \bar{T}_{2,j,0} = U_j^{t'_{1,j,0}}, \bar{T}_{2,j,1} = g^{t'_{2,j,1}}, \bar{T}_{1,k,0} = U_k^{t'_{1,k,0}}, \bar{T}_{1,k,1} = g^{t'_{1,k,1}}, \bar{T}_{2,k,0} = U_k^{t'_{2,k,0}}, \bar{T}_{2,k,1} = g^{t'_{2,k,1}}$. After this step secret key $\mathsf{SK} = [\mathcal{I}, (\mathsf{K}_i, \bar{\mathsf{K}}_i)_{i \in [2\ell-1]}]$ is *implicitly* defined. It is immediate to see that SK has the same distribution as a secret key output by Setup. We stress that $\mathcal{B}$ does not completely know SK since $\mathcal{B}$ has not computed $T_{1,k,0}$ and $T_{2,k,0}$ (which however are implicitly defined by U_k). However we observe but, since $0 \notin \mathsf{Pol}_k$, $\mathcal{B}$ can construct the partial public key $\mathsf{PPK}_{\mathsf{Pol}}$ for $\mathcal{A}$. In addition $\mathcal{B}$ knows all values $\bar{T}$ and can thus answer any $\mathsf{GenToken}$ query.

Challenge construction. When $\mathcal{B}$ is asked to provide token T, $\mathcal{B}$ constructs the token $T_{\vec{z}} = [S_{\vec{z}}, (Y_{1,i}, Y_{2,i})_{i \in S_{\vec{z}}}]$ as follows. The construction is deterministic and uses the randomness present in challenge; specifically, the a_i's (see Step 01 of the F-Linear Secret Sharing). As we shall see, the effect of the following instructions will be to construct a token in which $r_i = a_i$, for $i \in [2\ell - 1]$,

1. For $i \in S_{\vec{z}}$, $\mathcal{B}$ chooses random $Y_{2,i} \in \mathbb{G}$;
2. For $i = 1, \ldots, j-1$: If $i \in S_{\vec{z}}$, then $\mathcal{B}$ sets $Y_{1,i} = V_i^{t'_{1,i,0}}$;
3. $\mathcal{B}$ sets $Y_{1,j} = Z^{t'_{1,j,0}}$;
4. For $i = j+1, \ldots, 2\ell - 1$: If $i \in S_{\vec{z}}$, then $\mathcal{B}$ chooses random $Y_{1,i} \in \mathbb{G}$.

We have the following simple observations.

1. For each $i \in S_{\vec{z}}$, we have that $Y_{2,i}$ is a random element of $\mathbb{G}$ (just as in Experiment j and Experiment $j-1$).
2. For $i \in S_z$ and $i < j$, we have that $Y_{1,i} = V_i^{t'_{1,i,0}} = U_i^{a_i t'_{1,i,0}} = \bar{T}_{1,i,0}^{a_i}$ (just as in Experiment j and Experiment $j-1$).
3. For $i \in S_z$ and $i > j$, we have that $Y_{1,i}$ is a random element of $\mathbb{G}$ (just as in Experiment j and Experiment $j-1$).
4. If $Z = U_j^{a_j}$, then $Y_{1,j} = \bar{T}_{1,j,0}^{a_j}$ (just as in Experiment $j-1$).
5. If Z is random in $\mathbb{G}$, then $Y_{1,j}$ is random in $\mathbb{G}$ (just as in Experiment j).

By the above observations, we have that if $Z = U_j^{a_j}$, then the view of $\mathcal{A}$ in the interaction with $\mathcal{B}$ is exactly the same as the view of $\mathcal{A}$ in Experiment $j-1$; on the other hand, if Z is random, then the view of $\mathcal{A}$ in the interaction with $\mathcal{B}$ is exactly the same as the view of $\mathcal{A}$ in Experiment j. The lemma then follows. ∎

Next theorem holds.

Theorem 5 *Assume F-Linear Secret Sharing and F-Split Linear Secret Sharing. Then predicate encryption*

*(*Setup,PPKeyGen,Encryption,GenToken,Test*) is token secure.*

7 Acknowledgments

This work is partially founded by the Italian Ministry of University and Research Project PRIN 2008 *PEPPER: Privacy and Protection of Personal Data* (prot. 2008SY2PH4). We thank Dingding Jia for pointing us out an error in a previous definition of game F-$\mathsf{LSSExp}(1^n, 1^\ell)$.

References

[BDOP04] Dan Boneh, Giovanni Di Crescenzo, Rafail Ostrovsky, and Giuseppe Persiano. Public key encryption with keyword search. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology – EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 506–522, Interlaken, Switzerland, May 2–6, 2004. Springer-Verlag, Berlin, Germany.

[BIP10] Carlo Blundo, Vincenzo Iovino, and Giuseppe Persiano. Predicate encryption with partial public keys. Cryptology ePrint Archive, Report 2010/539, 2010. `http://eprint.iacr.org/`.

[BW06] Xavier Boyen and Brent Waters. Anonymous Hierarchical Identity-Based Encryption (Without Random Oracles). In Cynthia Dwork, editor, *Advances in Cryptology – CRYPTO 2006*, volume 4117 of *Lecture Notes in Computer Science*, pages 290–307, Santa Barbara, CA, USA, August 20–24, 2006. Springer-Verlag, Berlin, Germany.

[BW07] Dan Boneh and Brent Waters. Conjunctive, subset and range queries on encrypted data. In Salil P. Vadhan, editor, *TCC 2007: 4th Theory of Cryptography Conference*, volume 4392 of *Lecture Notes in Computer Science*, pages 535–554, Amsterdam, The Netherlands, February 21–24, 2007. Springer-Verlag, Berlin, Germany.

[GPSW06] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-Based Encryption for Fine-Grained Access Control for Encrypted Data. In *ACM CCS 06: 13th Conference on Computer and Communications Security*, pages 89–98, Alexandria, VA, USA, October 30 - November 3, 2006. ACM Press.

[IP08] Vincenzo Iovino and Giuseppe Persiano. Hidden-vector encryption with groups of prime order. In Steven D. Galbraith and Kenneth G. Paterson, editors, *Pairing-Based Cryptography - Pairing 2008, Second International Conference. Prooceedings*, volume 5209 of *Lecture Notes in Computer Science*, pages 75–88, Egham, UK, September 1–3, 2008. Springer-Verlag, Berlin, Germany.

[KSW08] Jonathan Katz, Amit Sahai, and Brent Waters. Predicate Encryption Supporting Disjunction, Polynomial Equations, and Inner Products. In Nigel Smart, editor, *Advances in Cryptology – EUROCRYPT 2008*, volume 4965 of *Lecture Notes in Computer Science*, pages 146–162, Istanbul, Turkey, April 13–17, 2008. Springer-Verlag, Berlin, Germany.

[Sha79] Adi Shamir. How to share a secret. *Communications of the Association for Computing Machinery*, 22(11):612–613, November 1979.

[SSW09] Emily Shen, Elaine Shi, and Brent Waters. Predicate privacy in encryption systems. In Omer Reingold, editor, *TCC 2009: 6th Theory of Cryptography Conference*, volume 5444 of *Lecture Notes in Computer Science*, pages 457–473, San Francisco, CA, USA, 2009. Springer-Verlag, Berlin, Germany.