PAPER Special Section on Cryptography and Information Security

Efficient Attribute-Based Signatures for Unbounded Arithmetic Branching Programs*

Pratish DATTA[†a)], ***Nonmember***, **Tatsuaki OKAMOTO**[†b)], ***Fellow***, ***and*** **Katsuyuki TAKASHIMA**[††c)], ***Senior Member***

SUMMARY This paper presents the *first attribute-based signature* (ABS) scheme in which the correspondence between signers and signatures is captured in an *arithmetic* model of computation. Specifically, we design a *fully* secure, i.e., *adaptively* unforgeable and *perfectly* signer-private ABS scheme for signing policies realizable by *arithmetic branching programs* (ABP), which are a *quite expressive* model of arithmetic computations. On a more positive note, the proposed scheme places *no bound* on the *size* and *input length* of the supported signing policy ABP's, and at the same time, supports the use of an input attribute for an *arbitrary* number of times inside a signing policy ABP, i.e., the so called *unbounded multi-use* of attributes. The size of our public parameters is *constant* with respect to the sizes of the signing attribute vectors and signing policies available in the system. The construction is built in (asymmetric) bilinear groups of prime order, and its unforgeability is derived in the standard model under (asymmetric version of) the *well-studied decisional linear* (DLIN) assumption coupled with the existence of standard *collision resistant hash functions*. Due to the use of the arithmetic model as opposed to the boolean one, our ABS scheme not only *excels significantly* over the existing state-of-the-art constructions in terms of *concrete efficiency*, but also achieves *improved applicability* in various practical scenarios. Our principal technical contributions are (a) extending the techniques of Okamoto and Takashima [PKC 2011, PKC 2013], which were originally developed in the context of boolean span programs, to the arithmetic setting; and (b) innovating new ideas to allow unbounded multi-use of attributes inside ABP's, which themselves are of unbounded size and input length.

key words: *attribute-based signatures, arithmetic branching programs, arithmetic span programs, concrete efficiency, unbounded multi-use of attributes, bilinear groups*

1. Introduction

Attribute-based signatures (ABS), introduced in the seminal work of Maji et al. [2], is an ambitious variant of digital signatures [3] that simultaneously enforce fine-grained control over authentication rights and conceal the identity of signers. An ABS scheme is associated with a predicate family $\mathcal{R} = \{R(Y, \cdot) : \mathcal{X} \to \{0, 1\} \mid Y \in \mathcal{Y}\}$, where $\mathcal{X}$ is a universe of possible signing attributes and $\mathcal{Y}$ is a collection of admissible signing policies over the attributes of $\mathcal{X}$. A central authority holds a master signing key and publishes system public parameters. Using its master signing key, the authority can issue restricted signing keys to individual signers corresponding to the attributes $X \in \mathcal{X}$ possessed by them. Such a constrained signing key associated with some attribute $X \in \mathcal{X}$ allows a signer to sign messages under only those signing policies $Y \in \mathcal{Y}$ which are satisfied by X, i.e., for which $R(Y, X) = 1$. The signatures can be verified by any one using solely the public parameters.

In an ABS scheme, by verifying a signature on some message with respect to some claimed signing policy, a verifier gets convinced that the signature is indeed generated by someone holding some attributes satisfying the policy. In particular, generating a valid signature on any message under any signing policy is (computationally) infeasible for any group of colluding signers, none of whom individually possesses a signing attribute that satisfies the signing policy, by pooling their attributes together. This is the so called *unforgeability* property of an ABS scheme. The second property of an ABS scheme, which ensures that given a signature, it is impossible to trace the exact signer or signing attributes used to create it, is known as *signer privacy*.

We refer the above notion of ABS as *signature-policy* ABS in recognition of the fact that in this notion of ABS signing policies are associated with signatures. Another flavor of this notion that interchanges the roles of signing attributes and signing policies, i.e., where signing policies are attached to signing keys and signatures are produced with respect to signing attributes, is usually termed as *key-policy* ABS. In addition to being an exciting cryptographic primitive in its own right, ABS has found countless important practical applications ranging from attribute-based messaging and attribute-based authentication to anonymous credential systems, trust negotiations, and leaking secrets (see [2], [4]–[6] for more details). In this paper, we will deal with the signature-policy variant since this variant is more natural and better suited in most of the aforementioned real-life applications of ABS.

Since their inception, ABS have been intensively studied in a long sequence of interesting works, and just like any other access-control primitive, a central theme of research in those works has been to expand the expressiveness of the allowable class of signing policies in view of implementing this delicate signature paradigm in scenarios where the

Manuscript received March 13, 2020.
Manuscript revised July 3, 2020.

[†]The authors are with the NTT Research, Inc., Sunnyvale, CA 94085, U.S.A.
[††]The author is with the Mitsubishi Electric Corporation, Kamakura-shi, 247-8501 Japan.

*An extended abstract of this paper [1] appeared in the proceedings of PKC 2019. This paper provides significant technical contribution over [1], namely, the complete proof of existential unforgeability of the proposed ABS construction, the most technically challenging part of this work, only a sketch of which was provided in [1]. Please refer to Lemmas 11–26 in Sect. 5.

a) E-mail: pratish.datta@ntt-research.com
b) E-mail: tatsuaki.okamoto@gmail.com
c) E-mail: Takashima.Katsuyuki@aj.MitsubishiElectric.co.jp
DOI: 10.1587/transfun.2020CIP0003

relationship between the signing attributes and policies is more and more sophisticated. Starting with the early works [2], [6]–[9], which can handle threshold signing policies, the class of admissible signing policies has been progressively enlarged to boolean formulas or span programs by Maji et al. [4], Okamoto and Takashima [10], [11] as well as El Kaafarani et al. [12], [13], and further to general circuits by Tang et al. [14], Sakai et al. [15], Tsabary [16], as well as El Kaafarani and Katsumata [17], based on various computational assumptions on bilinear groups and lattices, as well as in different security models such as random oracle model, generic group model, and standard model. Very recently, Datta et al. [18] and Sakai et al. [19] have constructed ABS schemes which can even realize Turing machines as signing policies. On the other hand, Bellare and Fuchsbauer [20] have put forward a versatile signature primitive termed as *policy-based signatures* (PBS) and have presented a generic construction of an ABS scheme from a PBS scheme. This generic construction, when instantiated with their proposed PBS scheme for general NP languages, results in an ABS scheme which can realize any NP relation as signing policy.

Two other important parameters determining the quality and applicability of ABS schemes are (a) supporting signing policies of unbounded polynomial size and input length, and (b) allowing the use of a signing attribute for an unbounded polynomial number of times inside a signing policy, i.e., the so called unbounded multi-use of attributes. Here, the term "unbounded" means not fixed by the public parameters. Out of the existing ABS schemes mentioned above, the only schemes which achieve both these parameters simultaneously and are somewhat practicable are the constructions due to Sakai et al. [15], [19]. While Okamoto and Takashima were able to realize unbounded multi-use of attributes in an updated version of their ABS scheme [10], namely, [21], their scheme cannot handle signing policies of unbounded size and input length. On the other hand, the ABS scheme of Datta et al. [18] features both the above properties, but are based on heavy-duty cryptographic tools such as indistinguishability obfuscation.

From the above review of the available ABS schemes, it is evident that research in the field of ABS has already reached the pinnacle in terms of expressiveness and unboundedness of the supported signing policies, as well as in terms of accommodating unbounded multi-use of attributes. Despite of this massive progress, one significant limitation that still persists in the current state of the art in this area is that all the existing ABS constructions consider the relationship between the signing attributes and policies only in some *boolean* model of computation, i.e., in those schemes the signing attributes are treated as bit strings and the policies are defined by sets of boolean operations. This raises the following natural question:

Can we construct an ABS *scheme which captures the relationship between the signing attributes and policies in some arithmetic model of computation, while at the same time, supports signing policies having unbounded size and input length, as well as unbounded multi-use of attributes?*

In an arithmetic-model-based ABS scheme, signing attributes are considered to be elements of some finite field $\mathbb{F}_q$, and signing policies are represented by collections of field operations, i.e., additions and multiplications over the field $\mathbb{F}_q$. The above question is not only intriguing from a theoretical perspective as the arithmetic model is a more structured one compared to its boolean counter part, it is also of a high significance from several practical view points. Most importantly, since arithmetic computations arise in many real-life scenarios, this question has a natural motivation when the concrete efficiency of most of the applications of ABS discussed above is considered. For instance, note that it is possible to capture any arithmetic relationships between the signing attributes and policies by employing the state-of-the-art ABS schemes of Sakai et al. for general circuits and Turing machines [15], [19] by representing an arithmetic computation by an equivalent boolean computation that replaces each field operation by a corresponding boolean sub-computation. Given the bit representation of the signing attributes, this approach can be used to simulate any arithmetic relation with an overhead which depends on the boolean complexity of the field operations. While providing reasonable asymptotic efficiency in theory (e.g., via fast integer multiplication techniques [22]), the concrete overhead of this approach is enormous. Moreover, scenarios may arise where one does not have access to the bits of the signing attributes and must treat them as atomic field elements. Note that in view of similar efficiency and applicability issues with boolean computations, arithmetic variants of various important cryptographic primitives have already been considered in the last few years. Examples include arithmetic garbled circuits [23], arithmetic multi-party computations [24], verifiable arithmetic computations [25], and so on. An even more fascinating aspect of the above question is to simultaneously support unbounded signing policies and unbounded multi-use of attributes in the arithmetic setting. These properties are especially significant for making the scheme resilient to potential usage situations which may arise after the scheme is setup. It can be readily inferred from the scarcity of existing ABS schemes supporting unbounded signing policies and unbounded multi-use of attributes simultaneously, even in the boolean setting, that achieving both these properties at the same time is a rather challenging task in any computational model.

Our Contribution

In this paper, we provide an *affirmative* answer to the above important question. For the *first* time in the literature, we design an ABS scheme where the relationship between the signing attributes and policies are considered in an *arithmetic* model of computation. More specifically, we construct an ABS scheme in which signing attributes are represented as elements of a finite field $\mathbb{F}_q$ and the signing policies are expressed as *arithmetic branching programs* (ABP) [26], [27] of *unbounded* polynomial *size* and *input length* over $\mathbb{F}_q$. While not capable of capturing most general relations like

Table 1 Comparison of concrete efficiency for 128-bit prime q.

Schemes	Computational Assumptions	Signature Size	Exponentiations Needed in Signing	Pairings Needed in Verification
[15]	SXDH	At least 4102 $\|g\|$	At least 5860	At least 4102
Ours	SXDLIN	26 $\|g\|$	138	30

The values presented in this table is for the signing policy ABP $f : \mathbb{F}_q \to \mathbb{F}_q$ defined by $f(x_1) = x_1 - a_1$, where a_1 is a constant belonging to $\mathbb{F}_q$.
In this table, $|g|$ represents the size of a group element.

arbitrary circuits or Turing machines, ABP's are a *quite powerful* model for realizing a wide range of relations that arise in practice, namely, the relations which can be expressed as polynomials over some finite field. In particular, note that there is a linear-time algorithm that can convert any Boolean formula, Boolean branching program, or arithmetic formula to an ABP only with a constant blow-up in the representation size. Thus, in terms of expressiveness of supported signing policies, our ABS scheme subsumes all the existing ABS schemes except those for general circuits or Turing machines. On a more positive note, we place *no restriction* on the *number of times* an attribute can be used inside the description of a signing policy ABP.

The proposed scheme enjoys *perfect* signer privacy and unforgeability against adversaries which are allowed to make an *arbitrary* polynomial number of signing key and signature queries *adaptively*. Our scheme is built in asymmetric bilinear groups of prime order, and its unforgeability is derived under the *simultaneous external decisional linear* (SXDLIN) assumption [28], which is the asymmetric version of and in fact equivalent to the *well-studied decisional linear* (DLIN) assumption, coupled with the existence of standard *collision resistant hash functions*. Observe that asymmetric bilinear groups of prime order are now considered to be both faster and more secure in the cryptographic community following the recent progress of analysing bilinear groups of composite order [29], [30] and symmetric bilinear groups instantiated with elliptic curves of small characteristics [31]–[34].

While our ABS construction is less expressive compared to the state-of-the-art schemes of Sakai et al. [15], [19], due to the use of the arithmetic model as opposed to the boolean one, our scheme *outperforms* those constructions by a *large margin* in terms of *concrete efficiency*. In fact, as we demonstrate in Table 1 and explain in Remark 2, even for a very simple signing policy such as an equality test over some finite field $\mathbb{F}_q$, where q is a 128-bit prime integer, our scheme can give more than 136 times better results from the view points of signature size and verification time while at least 42 times better performance on the signing time ground compared to the one of [15], which is also built in asymmetric prime-order bilinear group setting under the symmetric external Diffie-Hellman (SXDH) assumption. Hence, it is evident that our scheme is a far advantageous choice in most real-life applications of ABS, which often do not require the most general forms of signing policies but do require high performance.

Our ABS construction is developed *directly* from the scratch. On the technical side, our contribution is two fold: Firstly, we extend the ABS construction techniques devised by Okamoto and Takashima [10], [11] in the context of boolean formulas to the arithmetic setting. Secondly and more interestingly, we develop new ideas to support unbounded multi-use of attributes inside arithmetic signing policies, which themselves can be of an arbitrary size and input length.

2. Preliminaries

In this section we present the backgrounds required for the rest of this paper.

2.1 Notations

Let $\lambda \in \mathbb{N}$ denotes the security parameter and 1^λ be its unary encoding. Let $\mathbb{F}_q$ for any prime $q \in \mathbb{N}$ denotes the finite field of integers modulo q. For $d \in \mathbb{N}$ and $c \in \mathbb{N} \cup \{0\}$ (with $c < d$), we let $[d] = \{1, \ldots, d\}$ and $[c, d] = \{c, \ldots, d\}$. For any set Z, $z \xleftarrow{\cup} Z$ represents the process of uniformly sampling an element z from the set Z, and $\#Z$ signifies the size or cardinality of the set Z. For a probabilistic algorithm $\mathcal{P}$, we denote by $\Pi \xleftarrow{R} \mathcal{P}(\Theta)$ the process of sampling Π from the output distribution of $\mathcal{P}$ with a uniform random tape on input Θ. Similarly, for any deterministic algorithm $\mathcal{D}$, we write $\Pi = \mathcal{D}(\Theta)$ to denote the output of $\mathcal{D}$ on input Θ. We use the abbreviation PPT to mean probabilistic polynomial-time. We assume that all the algorithms are given the unary representation 1^λ of the security parameter λ as input, and will not write 1^λ explicitly as input of the algorithms when it is clear from the context. For any finite field $\mathbb{F}_q$ and $d \in \mathbb{N}$, let $\vec{v}$ denote the (row) vector $(v_1, \ldots, v_d) \in \mathbb{F}_q^d$, where $v_i \in \mathbb{F}_q$ for all $i \in [d]$. The all zero vector in $\mathbb{F}_q^d$ will be denoted by $\vec{0}^d$, while the canonical basis vectors in $\mathbb{F}_q^d$ will be represented by $\vec{e}^{(d,i)} = (\overbrace{0, \ldots, 0}^{i-1}, 1, \overbrace{0, \ldots, 0}^{d-i})$ for $i \in [d]$. For any two vectors $\vec{v}, \vec{w} \in \mathbb{F}_q^d$, $\vec{v} \cdot \vec{w}$ stands for the inner product of the vectors $\vec{v}$ and $\vec{w}$, i.e., $\vec{v} \cdot \vec{w} = \sum_{i \in [d]} v_i w_i \in \mathbb{F}_q$. For any $s \in \mathbb{N}$ and any collection of s vectors $\{\vec{v}^{(i)}\}_{i \in [s]} \subset \mathbb{F}_q^d$, we denote by $\mathsf{SPAN}\langle \vec{v}^{(i)} \mid i \in [s] \rangle$ the subspace of $\mathbb{F}_q^d$ spanned by $\{\vec{v}^{(i)}\}_{i \in [s]}$. For any multiplicative group $\mathbb{G}$, let $\boldsymbol{v}$ represents a d-dimensional (row) vector of

group elements, i.e., $\boldsymbol{v} = (g^{v_1}, \ldots, g^{v_d}) \in \mathbb{G}^d$ for some $d \in \mathbb{N}$, where $\vec{v} = (v_1, \ldots, v_d) \in \mathbb{F}_q^d$. We use $\boldsymbol{M} = \left(m_{k,i}\right)$ to represent a $d \times r$ matrix for some $d, r \in \mathbb{N}$ with entries $m_{k,i} \in \mathbb{F}_q$. By $\boldsymbol{M}^\top$ we will signify the transpose of the matrix $\boldsymbol{M}$ and by $\det(\boldsymbol{M})$ the determinant of the matrix $\boldsymbol{M}$. Let $\mathsf{GL}(d, \mathbb{F}_q)$ denote the set of all $d \times d$ invertible matrices over $\mathbb{F}_q$. A function $\mathsf{negl} : \mathbb{N} \to \mathbb{R}^+$ is said to be *negligible* if for every $c \in \mathbb{N}$, there exists $T \in \mathbb{N}$ such that for all $\lambda \in \mathbb{N}$ with $\lambda > T$, $|\mathsf{negl}(\lambda)| < 1/\lambda^c$.

2.2 Arithmetic Branching Programs and Arithmetic Span Programs

Here we formally define the notions of arithmetic branching programs (ABP) and arithmetic span programs (ASP), and explain the connection between them. These computational models will be used to represent the signing policies in our ABS construction.

Definition 1 (Arithmetic Branching Programs: ABP [26], [27]): A *branching program* (BP) Γ is defined by a 5-tuple $\Gamma = (V, E, \mathsf{v}_0, \mathsf{v}_1, \phi)$, where (V, E) is a directed acyclic graph, $\mathsf{v}_0, \mathsf{v}_1 \in V$ are two special vertices called the source and the sink respectively, and ϕ is a labeling function for the edges in E. An *arithmetic branching program* (ABP) Γ over a finite field $\mathbb{F}_q$ computes a function $f : \mathbb{F}_q^n \to \mathbb{F}_q$ for some $n \in \mathbb{N}$. In this case, the labeling function ϕ assigns to each edge in E either a degree one polynomial function in one of the input variables with coefficients in $\mathbb{F}_q$ or a constant in $\mathbb{F}_q$. Let $\wp$ be the set of all v_0-v_1 paths in Γ. The output of the function f computed by the ABP Γ on some input $\vec{x} = (x_1, \ldots, x_n) \in \mathbb{F}_q^n$ is defined as $f(\vec{x}) = \sum_{P \in \wp} \left[\prod_{\mathsf{e} \in P} \phi(\mathsf{e})|_{\vec{x}} \right]$, where for any $\mathsf{e} \in E$, $\phi(\mathsf{e})|_{\vec{x}}$ represents the evaluation of the function $\phi(\mathsf{e})$ at $\vec{x}$. We refer to $\#V + \#E$ as the size of the ABP Γ.

Ishai and Kushilevitz [26], [27] showed how to relate the computation performed by an ABP to the computation of the determinant of a matrix.

Lemma 1 ([27]): *Given an* ABP $\Gamma = (V, E, \mathsf{v}_0, \mathsf{v}_1, \phi)$ *computing a function* $f : \mathbb{F}_q^n \to \mathbb{F}_q$*, we can efficiently and deterministically compute a function* $\boldsymbol{L}$ *mapping an input* $\vec{x} \in \mathbb{F}_q^n$ *to a* $(\#V - 1) \times (\#V - 1)$ *matrix* $\boldsymbol{L}(\vec{x})$ *over* $\mathbb{F}_q$ *such that the following holds:*

- $\det(\boldsymbol{L}(\vec{x})) = f(\vec{x})$.
- *Each entry of* $\boldsymbol{L}(\vec{x})$ *is either a degree one polynomial in a single input variable* x_i $(i \in [n])$ *with coefficients in* $\mathbb{F}_q$ *or a constant in* $\mathbb{F}_q$.
- $\boldsymbol{L}(\vec{x})$ *contains only* -1*'s in the second diagonal, i.e., the diagonal just below the main diagonal, and 0's below the second diagonal.*

Specifically, $\boldsymbol{L}$ *is obtained by removing the column corresponding to* v_0 *and the row corresponding to* v_1 *in the matrix* $\boldsymbol{A}_\Gamma - \boldsymbol{I}$*, where* $\boldsymbol{A}_\Gamma$ *is the adjacency matrix for* Γ *and* $\boldsymbol{I}$ *is the identity matrix of the same size as* $\boldsymbol{A}_\Gamma$.

Note that there is a linear-time algorithm that converts any Boolean formula, Boolean branching program, or arithmetic formula to an ABP with a constant blow-up in the representation size. Thus, ABP's can be viewed as a stronger computational model than all the others mentioned above.

Definition 2 (Arithmetic Span Programs: ASP [35], [36]): An *arithmetic span program* (ASP) $\mathbb{S} = (\mathbb{U}, \rho)$ over n variables is a collection of pairs of vectors $\mathbb{U} = \{(\vec{y}^{(j)}, \vec{z}^{(j)})\}_{j \in [m]}$ for some $m \in \mathbb{N}$, where for all $j \in [m]$, $(\vec{y}^{(j)}, \vec{z}^{(j)}) \in (\mathbb{F}_q^\ell)^2$ for some $\ell \in \mathbb{N}$, and a function $\rho : [m] \to [n]$. We say that $\vec{x} \in \mathbb{F}_q^n$ satisfies $\mathbb{S}$ if and only if $\vec{e}^{(\ell,\ell)} \in \mathsf{SPAN}\langle x_{\rho(j)} \vec{y}^{(j)} + \vec{z}^{(j)} \mid j \in [m] \rangle$.

The following lemma shows a connection between the two arithmetic computational models defined above.

Lemma 2 ([36]): *There exists an efficient algorithm that given an* ABP $\Gamma = (V, E, \mathsf{v}_0, \mathsf{v}_1, \phi)$ *of size* $m + 1$ *computing some function* $f : \mathbb{F}_q^n \to \mathbb{F}_q$ *for some* $n, m \in \mathbb{N}$*, constructs an* ASP $\mathbb{S} = (\mathbb{U} = \{(\vec{y}^{(j)}, \vec{z}^{(j)})\}_{j \in [m]} \subset (\mathbb{F}_q^{(m+1)})^2, \rho : [m] \to [n])$ *such that for all* $\vec{x} \in \mathbb{F}_q^n$*,* $f(\vec{x}) = 0 \iff \mathbb{S}$ *accepts* $\vec{x}$*.*

Proof: The algorithm starts with constructing a modified ABP Γ' for f from the input ABP Γ, by first replacing each edge $\mathsf{e} \in E$ with a pair of edges labeled $\phi(\mathsf{e})$ and 1, and then adding an edge labeled 1 connecting the sink in Γ to a newly created sink node. Clearly, the modified ABP Γ' has $m + 2$ vertices, where every vertex has at most one incoming edge having a lable of degree 1. Next, it applies the transformation of Lemma 1 to Γ' to obtain the $(m + 1) \times (m + 1)$ matrix representation $\boldsymbol{L}$ of Γ'. By Lemma 1, we clearly have $\det(\boldsymbol{L}(\vec{x})) = f(\vec{x})$ for all $\vec{x} \in \mathbb{F}_q^n$, and $\boldsymbol{L}$ is of the following form:

$$\boldsymbol{L} = \begin{pmatrix} \bigstar & \bigstar & \bigstar & \ldots & \bigstar & \bigstar & 0 \\ -1 & \bigstar & \bigstar & \ldots & \bigstar & \bigstar & 0 \\ 0 & -1 & \bigstar & \ldots & \bigstar & \bigstar & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \ldots & -1 & \bigstar & 0 \\ 0 & 0 & 0 & \ldots & 0 & -1 & 1 \end{pmatrix},$$

where the $\bigstar$'s indicate polynomial functions of degree at most 1 in some input variable x_i $(i \in [n])$. Also, observe that since each vertex in Γ' has at most one incoming edge having a label of degree one, for all $j \in [m]$, each entry of the j^{th} column of the matrix $\boldsymbol{L}$ depends on one and the same input variable x_i $(i \in [n])$ and hence can be expressed as $x_i \vec{y}^{(j)} + \vec{z}^{(j)}$ for some pair of vectors $(\vec{y}^{(j)}, \vec{z}^{(j)}) \in (\mathbb{F}_q^{(m+1)})^2$. Further, it is immediate from the structure of $\boldsymbol{L}$ that the first m columns of $\boldsymbol{L}$ are linearly independent. Now, observe that $f(\vec{x}) = 0 \iff \det(\boldsymbol{L}(\vec{x})) = 0 \iff \vec{e}^{(m+1,m+1)}$, which is the $(m + 1)^{\text{th}}$ column of $\boldsymbol{L}$, lies in the linear span of the first m columns of $\boldsymbol{L}$, i.e., $\vec{e}^{(m+1,m+1)} \in \mathsf{SPAN}\langle x_i \vec{y}^{(j)} + \vec{z}^{(j)} \mid j \in [m] \wedge$ the j^{th} column of $\boldsymbol{L}$ depends on x_i $(i \in [n]) \rangle$. The

algorithm outputs the ASP $\mathbb{S} = (\mathbb{U} = \{(\vec{y}^{(j)}, \vec{z}^{(j)})\}_{j\in[m]} \subset (\mathbb{F}_q^{(m+1)})^2, \rho : [m] \to [n])$, where $\rho : [m] \to [n]$ is defined by $\rho(j) = i$ if the j^{th} column of $\boldsymbol{L}$ depends on x_i. This ASP $\mathbb{S}$ is clearly the desired one by the above explanation. This completes the proof of Lemma 2. ∎

2.3 Bilinear Groups and Dual Pairing Vector Spaces

In this section, we will provide the necessary backgrounds on bilinear groups and dual pairing vector spaces, which are the primary building blocks of our ABS construction.

Definition 3 (Bilinear Group): A bilinear group $\mathsf{params}_{\mathbb{G}} = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e)$ is a tuple of a prime $q \in \mathbb{N}$; cyclic multiplicative groups $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ of order q each with polynomial-time computable group operations; generators $g_1 \in \mathbb{G}_1$, $g_2 \in \mathbb{G}_2$; and a polynomial-time computable non-degenerate bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, i.e., e satisfies the following two properties:

- *Bilinearity*: $e(g_1^{\Upsilon}, g_2^{\hat{\Upsilon}}) = e(g_1, g_2)^{\Upsilon\hat{\Upsilon}}$ for all $\Upsilon, \hat{\Upsilon} \in \mathbb{F}_q$.
- *Non-degeneracy*: $e(g_1, g_2) \neq 1_{\mathbb{G}_T}$, where $1_{\mathbb{G}_T}$ denotes the identity element of the group $\mathbb{G}_T$.

A *bilinear group* is said to be asymmetric if no efficiently computable isomorphism exists between $\mathbb{G}_1$ and $\mathbb{G}_2$. Let $\mathcal{G}_{\mathsf{BPG}}$ be an algorithm that on input the unary encoded security parameter 1^{λ}, outputs a description $\mathsf{params}_{\mathbb{G}} = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e)$ of a bilinear group.

Definition 4 (Dual Pairing Vector Spaces: DPVS [37], [38]): A *dual pairing vector space* (DPVS) $\mathsf{params}_{\mathbb{V}} = (q, \mathbb{V}, \mathbb{V}^*, \mathbb{G}_T, \mathbb{A}, \mathbb{A}^*, e)$ formed by the direct product of a bilinear group $\mathsf{params}_{\mathbb{G}} = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e)$ is a tuple of a prime $q \in \mathbb{N}$; d-dimensional vector spaces $\mathbb{V} = \mathbb{G}_1^d$, $\mathbb{V}^* = \mathbb{G}_2^d$ over $\mathbb{F}_q$ for some $d \in \mathbb{N}$, under vector addition and scalar multiplication defined component-wise in the usual manner; canonical bases $\mathbb{A} = \{\boldsymbol{a}^{(i)} = (\overbrace{1_{\mathbb{G}_1}, \ldots, 1_{\mathbb{G}_1}}^{i-1}, g_1, \overbrace{1_{\mathbb{G}_1}, \ldots, 1_{\mathbb{G}_1}}^{d-i})\}_{i\in[d]}$ and $\mathbb{A}^* = \{\boldsymbol{a}^{*(i)} = (\overbrace{1_{\mathbb{G}_2}, \ldots, 1_{\mathbb{G}_2}}^{i-1}, g_2, \overbrace{1_{\mathbb{G}_2}, \ldots, 1_{\mathbb{G}_2}}^{d-i})\}_{i\in[d]}$ of $\mathbb{V}$ and $\mathbb{V}^*$ respectively, where $1_{\mathbb{G}_1}$ and $1_{\mathbb{G}_2}$ are the identity elements of the groups $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively; and a pairing $e : \mathbb{V} \times \mathbb{V}^* \to \mathbb{G}_T$ defined by $e(\boldsymbol{v}, \boldsymbol{w}) = \prod_{i\in[d]} e(g_1^{v_i}, g_2^{w_i}) \in \mathbb{G}_T$ for all $\boldsymbol{v} = (g_1^{v_1}, \ldots, g_1^{v_d}) \in \mathbb{V}$, $\boldsymbol{w} = (g_2^{w_1}, \ldots, g_2^{w_d}) \in \mathbb{V}^*$. Observe that the newly defined map e is also non-degenerate bilinear, i.e., e also satisfies the following two properties:

- *Bilinearity*: $e(\Upsilon\boldsymbol{v}, \hat{\Upsilon}\boldsymbol{w}) = e(\boldsymbol{v}, \boldsymbol{w})^{\Upsilon\hat{\Upsilon}}$ for all $\Upsilon, \hat{\Upsilon} \in \mathbb{F}_q$, $\boldsymbol{v} \in \mathbb{V}$, and $\boldsymbol{w} \in \mathbb{V}^*$.
- *Non-degeneracy*: If $e(\boldsymbol{v}, \boldsymbol{w}) = 1_{\mathbb{G}_T}$ for all $\boldsymbol{w} \in \mathbb{V}^*$, then $\boldsymbol{v} = (\overbrace{1_{\mathbb{G}_1}, \ldots, 1_{\mathbb{G}_1}}^{d})$. Similar statement also holds with the vectors $\boldsymbol{v}$ and $\boldsymbol{w}$ interchanged.

For any ordered basis $\mathbb{W} = \{\boldsymbol{w}^{(1)}, \ldots, \boldsymbol{w}^{(d)}\}$ of $\mathbb{V}$ (or $\mathbb{V}^*$), and any vector $\vec{v} \in \mathbb{F}_q^d$, let $(\vec{v})_{\mathbb{W}}$ represent the vector in $\mathbb{V}$ (or $\mathbb{V}^*$ accordingly) formed by the linear combination of the members of $\mathbb{W}$ with the components of $\vec{v}$ as the coefficients, i.e., $(\vec{v})_{\mathbb{W}} = \sum_{i\in[d]} v_i \boldsymbol{w}^{(i)} \in \mathbb{V}$ (or $\mathbb{V}^*$ accordingly). Also, for any $s \in \mathbb{N}$ and any collection of s vectors $\{\boldsymbol{v}^{(i)}\}_{i\in[s]}$ of $\mathbb{V}$ (or $\mathbb{V}^*$), we will denote by $\mathsf{SPAN}\langle \boldsymbol{v}^{(i)} \mid i \in [s]\rangle$ the subspace of $\mathbb{V}$ (or $\mathbb{V}^*$ accordingly) spanned by the set of vectors $\{\boldsymbol{v}^{(i)}\}_{i\in[s]}$. The DPVS generation algorithm $\mathcal{G}_{\mathsf{DPVS}}$ takes as input the unary encoded security parameter 1^{λ}, a dimension value $d \in \mathbb{N}$, along with a bilinear group $\mathsf{params}_{\mathbb{G}} = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e) \xleftarrow{\mathsf{R}} \mathcal{G}_{\mathsf{BPG}}()$, and outputs a description $\mathsf{params}_{\mathbb{V}} = (q, \mathbb{V}, \mathbb{V}^*, \mathbb{G}_T, \mathbb{A}, \mathbb{A}^*, e)$ of DPVS with d-dimensional $\mathbb{V}$ and $\mathbb{V}^*$.

We now describe random *dual orthonormal basis* generator $\mathcal{G}_{\mathsf{OB}}$ [37], [38] in Fig. 1. This algorithm will be utilized as a sub-routine in our ABS construction.

2.4 Complexity Assumption

For realizing our ABS construction in asymmetric bilinear groups, we rely on the natural extension of the well-studied decisional linear (DLIN) assumption to the asymmetric bilinear group setting, called the external decisional linear (XDLIN) assumption.

Assumption (External Decisional Linear: XDLIN [28], [39]): For $\jmath \in [2]$, the $\mathsf{XDLIN}_{\jmath}$ problem is to guess the bit $\widehat{\beta} \xleftarrow{\mathsf{U}} \{0,1\}$ given $\varrho_{\widehat{\beta}}^{\mathsf{XDLIN}_{\jmath}} = (\mathsf{params}_{\mathbb{G}}, g_1^{\varpi}, g_1^{\Upsilon}, g_1^{\aleph\varpi}, g_1^{\varsigma\Upsilon}, g_2^{\varpi}, g_2^{\Upsilon}, g_2^{\aleph\varpi}, g_2^{\varsigma\Upsilon}, \mathfrak{R}_{\jmath,\widehat{\beta}})$, where

$$\mathsf{params}_{\mathbb{G}} = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e) \xleftarrow{\mathsf{R}} \mathcal{G}_{\mathsf{BPG}}();$$
$$\varpi, \Upsilon, \aleph, \varsigma, \varepsilon \xleftarrow{\mathsf{U}} \mathbb{F}_q;$$
$$\mathfrak{R}_{\jmath,0} = g_{\jmath}^{(\aleph+\varsigma)}, \mathfrak{R}_{\jmath,1} = g_{\jmath}^{(\aleph+\varsigma)+\varepsilon}.$$

The $\mathsf{XDLIN}_{\jmath}$ assumption states that for any PPT algorithm $\mathcal{S}$, for any security parameter λ, the advantage of $\mathcal{S}$ in deciding the $\mathsf{XDLIN}_{\jmath}$ problem, defined as

$$\mathsf{Adv}_{\mathcal{S}}^{\mathsf{XDLIN}_{\jmath}}(\lambda) = \Big| \Pr\Big[1 \xleftarrow{\mathsf{R}} \mathcal{S}(\varrho_0^{\mathsf{XDLIN}_{\jmath}})\Big] - \Pr\Big[1 \xleftarrow{\mathsf{R}} \mathcal{S}(\varrho_1^{\mathsf{XDLIN}_{\jmath}})\Big] \Big|$$

is negligible in λ, i.e., $\mathsf{Adv}_{\mathcal{S}}^{\mathsf{XDLIN}_{\jmath}}(\lambda) \leq \mathsf{negl}(\lambda)$, where negl is some negligible function. The simultaneous XDLIN (SXDLIN) assumption states that both XDLIN_1 and XDLIN_2 assumptions hold at the same time. For any security parameter λ, we denote the advantage of any probabilistic algorithm $\mathcal{S}$ against SXDLIN as $\mathsf{Adv}_{\mathcal{S}}^{\mathsf{SXDLIN}}(\lambda)$.

Indeed as noted in [28], for all $\jmath \in [2]$, the $\mathsf{XDLIN}_{\jmath}$ assumption is equivalent to the DLIN assumption in the group $\mathbb{G}_{\jmath}$ in the generic bilinear group model [40]. We now define some

$\mathcal{G}_{\mathsf{OB}}(N, (d_0, \ldots, d_N))$: This algorithm takes as input the unary encoded security parameter 1^λ, a number $N \in \mathbb{N}$, and the respective dimensions $d_0, \ldots, d_N \in \mathbb{N}$ of the $N+1$ pairs of bases to be generated. It executes the following operations:

1. It first generates $\mathsf{params}_{\mathbb{G}} = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e) \xleftarrow{\mathsf{R}} \mathcal{G}_{\mathsf{BPG}}()$.
2. Next, it samples $\psi \xleftarrow{\mathsf{U}} \mathbb{F}_q \backslash \{0\}$ and computes $g_T = e(g_1, g_2)^\psi$.
3. Then, for $\iota \in [0, N]$, it performs the following:
   a. It constructs $\mathsf{params}_{\mathbb{V}_\iota} = (q, \mathbb{V}_\iota, \mathbb{V}_\iota^*, \mathbb{G}_T, \mathbb{A}_\iota, \mathbb{A}_\iota^*, e) \xleftarrow{\mathsf{R}} \mathcal{G}_{\mathsf{DPVS}}(d_\iota, \mathsf{params}_{\mathbb{G}})$.
   b. It samples $\boldsymbol{B}^{(\iota)} = \left(b_{k,i}^{(\iota)}\right) \xleftarrow{\mathsf{U}} \mathsf{GL}(d_\iota, \mathbb{F}_q)$.
   c. It computes $\boldsymbol{B}^{*(\iota)} = \left(b_{k,i}^{*(\iota)}\right) = \psi((\boldsymbol{B}^{(\iota)})^{-1})^\top$.
   d. For all $k \in [d_\iota]$, let $\vec{b}^{(\iota,k)}$ and $\vec{b}^{*(\iota,k)}$ represent the k^{th} rows of $\boldsymbol{B}^{(\iota)}$ and $\boldsymbol{B}^{*(\iota)}$ respectively. It computes $\boldsymbol{b}^{(\iota,k)} = (\vec{b}^{(\iota,k)})_{\mathbb{A}_\iota}, \boldsymbol{b}^{*(\iota,k)} = (\vec{b}^{*(\iota,k)})_{\mathbb{A}_\iota^*}$ for $k \in [d_\iota]$, and sets
   $$\mathbb{B}_\iota = \{\boldsymbol{b}^{(\iota,1)}, \ldots, \boldsymbol{b}^{(\iota,d_\iota)}\}, \mathbb{B}_\iota^* = \{\boldsymbol{b}^{*(\iota,1)}, \ldots, \boldsymbol{b}^{*(\iota,d_\iota)}\}.$$
   Clearly $\mathbb{B}_\iota$ and $\mathbb{B}_\iota^*$ form bases of the vector spaces $\mathbb{V}_\iota$ and $\mathbb{V}_\iota^*$ respectively. Also, note that $\mathbb{B}_\iota$ and $\mathbb{B}_\iota^*$ are dual orthonormal in the sense that for all $k, k' \in [d_\iota]$,
   $$e(\boldsymbol{b}^{(\iota,k)}, \boldsymbol{b}^{*(\iota,k')}) = \begin{cases} g_T & \text{if } k = k', \\ 1_{\mathbb{G}_T} & \text{otherwise.} \end{cases}$$
4. Next, it sets $\mathsf{params} = (\{\mathsf{params}_{\mathbb{V}_\iota}\}_{\iota \in [0,N]}, g_T)$.
5. It returns $(\mathsf{params}, \{\mathbb{B}_\iota, \mathbb{B}_\iota^*\}_{\iota \in [0,N]})$.

Fig. 1 Dual orthonormal basis generator $\mathcal{G}_{\mathsf{OB}}$.

decisional problems. We will rely on the hardness of these decisional problems for deriving the unforgeability property of our ABS construction. The hardness of these decisional problems can be reduced to that of the SXDLIN problem, as shown in Lemmas 3–8 below.

Definition 5 (Problem 1): Problem 1 is to guess the bit $\widehat{\beta} \in \{0,1\}$ given $\varrho_{\widehat{\beta}}^{\mathsf{P1}} = (\mathsf{params}, \{\mathbb{B}_\iota, \widetilde{\mathbb{B}}_\iota^*\}_{\iota \in [0,2]}, \{\boldsymbol{e}^{(\alpha,\nu,\widehat{\beta})}\}_{\alpha \in [2], \nu \in [2]}, \boldsymbol{f}^{(0,\widehat{\beta})}, \{\boldsymbol{f}^{(1,\nu,\widehat{\beta})}\}_{\nu \in [2]})$, where

$$\begin{aligned}
&(\mathsf{params}, \{\mathbb{B}_\iota, \mathbb{B}_\iota^*\}_{\iota \in [0,2]}) \xleftarrow{\mathsf{R}} \mathcal{G}_{\mathsf{OB}}(2, (4, 14, 8)); \\
&\widetilde{\mathbb{B}}_0^* = \{\boldsymbol{b}^{*(0,1)}, \boldsymbol{b}^{*(0,3)}, \boldsymbol{b}^{*(0,4)}\}; \\
&\widetilde{\mathbb{B}}_1^* = \{\boldsymbol{b}^{*(1,1)}, \ldots, \boldsymbol{b}^{*(1,4)}, \boldsymbol{b}^{*(1,7)}, \boldsymbol{b}^{*(1,8)}, \boldsymbol{b}^{*(1,11)}, \ldots, \\
&\qquad \boldsymbol{b}^{*(1,14)}\}; \\
&\widetilde{\mathbb{B}}_2^* = \{\boldsymbol{b}^{*(2,1)}, \boldsymbol{b}^{*(2,2)}, \boldsymbol{b}^{*(2,5)}, \ldots, \boldsymbol{b}^{*(2,8)}\}; \\
&\delta, \tau, \{\theta_\nu, \theta'_\nu\}_{\nu \in [2]}, \gamma_0 \xleftarrow{\mathsf{U}} \mathbb{F}_q, \\
&\{\vec{\gamma}^{(\nu)}, \vec{\gamma}'^{(\nu)}, \vec{\gamma}''^{(\nu)}\}_{\nu \in [2]} \xleftarrow{\mathsf{U}} \mathbb{F}_q^2; \\
&\left.\begin{aligned}
\boldsymbol{e}^{(1,\nu,0)} &= (\vec{0}^4, \vec{0}^6, \vec{0}^2, \vec{\gamma}^{(\nu)})_{\mathbb{B}_1} \\
\boldsymbol{e}^{(1,\nu,1)} &= (\vec{0}^4, \vec{0}^4, \theta_\nu \vec{e}^{(2,\nu)}, \vec{0}^2, \vec{\gamma}^{(\nu)})_{\mathbb{B}_1} \\
\boldsymbol{e}^{(2,\nu,0)} &= (\vec{0}^2, \vec{0}^2, \vec{0}^2, \vec{\gamma}'^{(\nu)})_{\mathbb{B}_2} \\
\boldsymbol{e}^{(2,\nu,1)} &= (\vec{0}^2, \theta'_\nu \vec{e}^{(2,\nu)}, \vec{0}^2, \vec{\gamma}'^{(\nu)})_{\mathbb{B}_2}
\end{aligned}\right\} \text{for } \nu \in [2]; \\
&\boldsymbol{f}^{(0,0)} = (\delta, 0, 0, \gamma_0)_{\mathbb{B}_0}, \boldsymbol{f}^{(0,1)} = (\delta, \tau, 0, \gamma_0)_{\mathbb{B}_0}; \\
&\left.\begin{aligned}
\boldsymbol{f}^{(1,\nu,0)} &= (\vec{0}^2, \delta \vec{e}^{(2,\nu)}, \vec{0}^6, \vec{0}^2, \vec{\gamma}''^{(\nu)})_{\mathbb{B}_1} \\
\boldsymbol{f}^{(1,\nu,1)} &= (\vec{0}^2, \delta \vec{e}^{(2,\nu)}, \tau \vec{e}^{(2,\nu)}, \vec{0}^4, \vec{0}^2, \vec{\gamma}''^{(\nu)})_{\mathbb{B}_1}
\end{aligned}\right\} \text{for } \nu \in [2].
\end{aligned}$$

For any security parameter λ, the advantage of any probabilistic adversary $\mathcal{B}$ in deciding Problem 1 is defined as

$$\mathsf{Adv}_{\mathcal{B}}^{\mathsf{P1}}(\lambda) = \left| \Pr\left[1 \xleftarrow{\mathsf{R}} \mathcal{B}(\varrho_0^{\mathsf{P1}})\right] - \Pr\left[1 \xleftarrow{\mathsf{R}} \mathcal{B}(\varrho_1^{\mathsf{P1}})\right] \right|.$$

Lemma 3: *For any probabilistic algorithm $\mathcal{B}$, there exist probabilistic algorithms $\mathcal{S}_1$ and $\mathcal{S}_2$, whose running times are essentially the same as that of $\mathcal{B}$, such that for any security parameter λ, $\mathsf{Adv}_{\mathcal{B}}^{\mathsf{P1}}(\lambda) \leq \sum_{\nu \in [2]} \mathsf{Adv}_{\mathcal{S}_\nu}^{\mathsf{SXDLIN}}(\lambda) + \mathsf{negl}(\lambda)$, where negl is some negligible function.*

Proof: Observe that Problem 1 is analogous to Problem 1 in [10], [21]. Thus, the proof of Lemma 3 is analogous to that of Lemma 1 in [21]. ∎

Definition 6 (Problem 2): Problem 2 is to guess the bit $\widehat{\beta} \in \{0,1\}$ given $\varrho_{\widehat{\beta}}^{\mathsf{P2}} = (\mathsf{params}, \{\widetilde{\mathbb{B}}_\iota, \mathbb{B}_\iota^*\}_{\iota \in [0,1]}, \mathbb{B}_2, \mathbb{B}_2^*, \boldsymbol{h}^{*(0,\widehat{\beta})}, \boldsymbol{f}^{(0)}, \{\boldsymbol{h}^{*(1,\nu,\widehat{\beta})}, \boldsymbol{f}^{(1,\nu)}\}_{\nu \in [2]}, \{\boldsymbol{h}^{*(2,\nu)}\}_{\nu \in [2]})$, where

$$\begin{aligned}
&(\mathsf{params}, \{\mathbb{B}_\iota, \mathbb{B}_\iota^*\}_{\iota \in [0,2]}) \xleftarrow{\mathsf{R}} \mathcal{G}_{\mathsf{OB}}(2, (4, 14, 8)); \\
&\widetilde{\mathbb{B}}_0 = \{\boldsymbol{b}^{(0,1)}, \boldsymbol{b}^{(0,3)}, \boldsymbol{b}^{(0,4)}\}; \\
&\widetilde{\mathbb{B}}_1 = \{\boldsymbol{b}^{(1,1)}, \ldots, \boldsymbol{b}^{(1,4)}, \boldsymbol{b}^{(1,7)}, \ldots, \boldsymbol{b}^{(1,14)}\}; \\
&\vartheta, \varkappa, \delta, \tau, \xi_0 \xleftarrow{\mathsf{U}} \mathbb{F}_q, \{\vec{\xi}^{(\nu)}\}_{\nu \in [2]} \xleftarrow{\mathsf{U}} \mathbb{F}_q^2; \\
&\boldsymbol{h}^{*(0,0)} = (\vartheta, 0, \xi_0, 0)_{\mathbb{B}_0^*}, \boldsymbol{h}^{*(0,1)} = (\vartheta, \varkappa, \xi_0, 0)_{\mathbb{B}_0^*}; \\
&\boldsymbol{f}^{(0)} = (\delta, \tau, 0, 0)_{\mathbb{B}_0}; \\
&\left.\begin{aligned}
\boldsymbol{h}^{*(1,\nu,0)} &= (\vec{0}^2, \vartheta \vec{e}^{(2,\nu)}, \vec{0}^6, \vec{\xi}^{(\nu)}, \vec{0}^2)_{\mathbb{B}_1^*} \\
\boldsymbol{h}^{*(1,\nu,1)} &= (\vec{0}^2, \vartheta \vec{e}^{(2,\nu)}, \varkappa \vec{e}^{(2,\nu)}, \vec{0}^4, \vec{\xi}^{(\nu)}, \vec{0}^2)_{\mathbb{B}_1^*} \\
\boldsymbol{f}^{(1,\nu)} &= (\vec{0}^2, \delta \vec{e}^{(2,\nu)}, \tau \vec{e}^{(2,\nu)}, \vec{0}^4, \vec{0}^2, \vec{0}^2)_{\mathbb{B}_1}
\end{aligned}\right\} \text{for } \nu \in [2]; \\
&\boldsymbol{h}^{*(2,\nu)} = \vartheta \boldsymbol{b}^{*(2,\nu)} \text{ for } \nu \in [2].
\end{aligned}$$

For any security parameter λ, the advantage of any probabilistic adversary $\mathcal{B}$ in deciding Problem 2 is defined as

$$\mathsf{Adv}^{\mathsf{P2}}_{\mathcal{B}}(\lambda) = \left|\Pr\left[1 \xleftarrow{\mathsf{R}} \mathcal{B}(\varrho_0^{\mathsf{P2}})\right] - \Pr\left[1 \xleftarrow{\mathsf{R}} \mathcal{B}(\varrho_1^{\mathsf{P2}})\right]\right|.$$

Lemma 4: *For any probabilistic algorithm $\mathcal{B}$, there exists a probabilistic algorithm $\mathcal{S}$, whose running time is essentially the same as that of $\mathcal{B}$, such that for any security parameter λ, $\mathsf{Adv}^{\mathsf{P2}}_{\mathcal{B}}(\lambda) \leq \mathsf{Adv}^{\mathsf{SXDLIN}}_{\mathcal{S}}(\lambda) + \mathsf{negl}(\lambda)$, where negl is some negligible function.*

Proof: Observe that Problem 2 is essentially the same as Basic Problem 2 in [41], [42]. Hence, Lemma 4 can be proven in the same way as Lemma 35 in [42]. ∎

Definition 7 (Problem 3): Problem 3 is to guess the bit $\widehat{\beta} \in \{0,1\}$ given $\varrho^{\mathsf{P3}}_{\widehat{\beta}} = (\mathsf{params}, \{\mathbb{B}_\iota, \mathbb{B}^*_\iota\}_{\iota\in\{0,2\}}, \mathbb{B}_1, \widetilde{\mathbb{B}}^*_1, \{\boldsymbol{e}^{(1,\nu,\widehat{\beta})}\}_{\nu\in[2]})$, where

$$\left.\begin{aligned}&(\mathsf{params}, \{\mathbb{B}_\iota, \mathbb{B}^*_\iota\}_{\iota\in[0,2]}) \xleftarrow{\mathsf{R}} \mathcal{G}_{\mathsf{OB}}(2,(4,14,8));\\ &\widetilde{\mathbb{B}}^*_1 = \{\boldsymbol{b}^{*(1,1)}, \ldots, \boldsymbol{b}^{*(1,8)}, \boldsymbol{b}^{*(1,11)}, \ldots, \boldsymbol{b}^{*(1,14)}\};\\ &\{\theta_\nu\}_{\nu\in[2]} \xleftarrow{\mathsf{U}} \mathbb{F}_q, \{\vec{\gamma}^{(\nu)}\}_{\nu\in[2]} \xleftarrow{\mathsf{U}} \mathbb{F}_q^2;\\ &\boldsymbol{e}^{(1,\nu,0)} = (\vec{0}^4, \vec{0}^6, \vec{0}^2, \vec{\gamma}^{(\nu)})_{\mathbb{B}_1}\\ &\boldsymbol{e}^{(1,\nu,1)} = (\vec{0}^4, \vec{0}^4, \theta_\nu \vec{e}^{(2,\nu)}, \vec{0}^2, \vec{\gamma}^{(\nu)})_{\mathbb{B}_1}\end{aligned}\right\} \text{for } \nu \in [2].$$

For any security parameter λ, the advantage of any probabilistic adversary $\mathcal{B}$ in deciding Problem 3 is defined as

$$\mathsf{Adv}^{\mathsf{P3}}_{\mathcal{B}}(\lambda) = \left|\Pr\left[1 \xleftarrow{\mathsf{R}} \mathcal{B}(\varrho_0^{\mathsf{P3}})\right] - \Pr\left[1 \xleftarrow{\mathsf{R}} \mathcal{B}(\varrho_1^{\mathsf{P3}})\right]\right|.$$

Lemma 5: *For any probabilistic algorithm $\mathcal{B}$, there exist probabilistic algorithms $\mathcal{S}_1$ and $\mathcal{S}_2$, whose running times are essentially the same as that of $\mathcal{B}$, such that for any security parameter λ, $\mathsf{Adv}^{\mathsf{P3}}_{\mathcal{B}}(\lambda) \leq \sum_{\nu\in[2]} \mathsf{Adv}^{\mathsf{SXDLIN}}_{\mathcal{S}_\nu}(\lambda) + \mathsf{negl}(\lambda)$, where negl is some negligible function.*

Proof: Observe that Problem 3 is similar to Problem 1 in [10], [21]. Thus, the proof of Lemma 5 is analogous to that of Lemma 1 in [21]. ∎

Definition 8 (Problem 4-α ($\alpha \in [n = \mathsf{p}(\lambda)]$)): Problem 4-$\alpha$ is to guess the bit $\widehat{\beta} \in \{0,1\}$ given $\varrho^{\mathsf{P4}\text{-}\alpha}_{\widehat{\beta}} = (\mathsf{params}, \{\mathbb{B}_\iota, \mathbb{B}^*_\iota\}_{\iota\in\{0,2\}}, \widetilde{\mathbb{B}}_1, \mathbb{B}^*_1, \boldsymbol{f}^{(0)}, \{\boldsymbol{h}^{*(1,\alpha,\nu,\widehat{\beta})}, \boldsymbol{f}^{(1,\nu)}, \boldsymbol{g}^{(1,\nu)}\}_{\nu\in[2]})$, where

$$\left.\begin{aligned}&(\mathsf{params}, \{\mathbb{B}_\iota, \mathbb{B}^*_\iota\}_{\iota\in[0,2]}) \xleftarrow{\mathsf{R}} \mathcal{G}_{\mathsf{OB}}(2,(4,14,8));\\ &\widetilde{\mathbb{B}}_1 = \{\boldsymbol{b}^{(1,1)}, \ldots, \boldsymbol{b}^{(1,6)}, \boldsymbol{b}^{(1,11)}, \ldots, \boldsymbol{b}^{(1,14)}\};\\ &\tau, \{\breve{\sigma}_{\alpha,\nu}\}_{\nu\in[2]}, \{\theta_{\alpha,\nu}\}_{\nu\in[2]} \xleftarrow{\mathsf{U}} \mathbb{F}_q, \{\vec{\xi}^{(\alpha,\nu)}\}_{\nu\in[2]} \xleftarrow{\mathsf{U}} \mathbb{F}_q^2;\\ &\boldsymbol{f}^{(0)} = (0, \tau, 0, 0)_{\mathbb{B}_0};\\ &\boldsymbol{h}^{*(1,\alpha,\nu,0)} = (\breve{\sigma}_{\alpha,\nu}(1,\alpha), \vec{0}^2, \vec{0}^6, \vec{\xi}^{(\alpha,\nu)}, \vec{0}^2)_{\mathbb{B}^*_1}\\ &\boldsymbol{h}^{*(1,\alpha,\nu,1)} = (\breve{\sigma}_{\alpha,\nu}(1,\alpha), \vec{0}^2, -\theta_{\alpha,\nu}\vec{e}^{(2,\nu)},\\ &\qquad \theta_{\alpha,\nu}\vec{e}^{(2,\nu)}, \vec{0}^2, \vec{\xi}^{(\alpha,\nu)}, \vec{0}^2)_{\mathbb{B}^*_1}\\ &\boldsymbol{f}^{(1,\nu)} = (\vec{0}^4, \tau\vec{e}^{(2,\nu)}, \tau\vec{e}^{(2,\nu)}, \vec{0}^2, \vec{0}^2, \vec{0}^2)_{\mathbb{B}_1}\\ &\boldsymbol{g}^{(1,\nu)} = (\vec{0}^4, \vec{0}^4, \tau\vec{e}^{(2,\nu)}, \vec{0}^2, \vec{0}^2)_{\mathbb{B}_1}\end{aligned}\right\} \text{for } \nu \in [2].$$

For any security parameter λ, for any $n = \mathsf{p}(\lambda)$, where p is an arbitrary polynomial, for any $\alpha \in [n]$, the advantage of any probabilistic adversary $\mathcal{B}$ in deciding Problem 4-α is defined as

$$\mathsf{Adv}^{\mathsf{P4}\text{-}\alpha}_{\mathcal{B}}(\lambda) = \left|\Pr\left[1 \xleftarrow{\mathsf{R}} \mathcal{B}(\varrho_0^{\mathsf{P4}\text{-}\alpha})\right] - \Pr\left[1 \xleftarrow{\mathsf{R}} \mathcal{B}(\varrho_1^{\mathsf{P4}\text{-}\alpha})\right]\right|.$$

Lemma 6: *For any probabilistic algorithm $\mathcal{B}$, there exists a probabilistic algorithm $\mathcal{S}$, whose running time is essentially the same as that of $\mathcal{B}$, such that for any security parameter λ and any $n = \mathsf{p}(\lambda)$, $\mathsf{Adv}^{\mathsf{P4}\text{-}\alpha}_{\mathcal{B}}(\lambda) \leq \sum_{\nu\in[2]} \mathsf{Adv}^{\mathsf{SXDLIN}}_{\mathcal{S}_{\alpha\text{-}\nu}}(\lambda) + \mathsf{negl}(\lambda)$ for all $\alpha \in [n]$, where $\mathcal{S}_{\alpha\text{-}\nu}(\cdot) = \mathcal{S}(\alpha, \nu, \cdot)$ for any $\alpha, \nu \in \mathbb{N}$, and negl is some negligible function.*

Proof: Observe that Problem 4-α is essentially the same as Basic Problem 3-p in [41], [42]. Hence, the proof of Lemma 6 is analogous to that of Lemma 36 in [42]. ∎

Definition 9 (Problem 5-α ($\alpha \in [n = \mathsf{p}(\lambda)]$)): Problem 5-$\alpha$ is to guess the bit $\widehat{\beta} \in \{0,1\}$ given $\varrho^{\mathsf{P5}\text{-}\alpha}_{\widehat{\beta}} = (\mathsf{params}, \{\mathbb{B}_\iota, \mathbb{B}^*_\iota\}_{\iota\in\{0,2\}}, \mathbb{B}_1, \mathbb{B}^*_1, \boldsymbol{h}^{*(0)}, \{\boldsymbol{h}^{*(1,\alpha,\nu)}\}_{\nu\in[2]}, \{\boldsymbol{f}^{(1,\iota,\nu,\widehat{\beta})}\}_{\iota\in[n]\setminus\{\alpha\},\nu\in[2]}, \{\check{\boldsymbol{h}}^{*(\nu)}\}_{\nu\in\{5,6,9,10\}})$, where

$$\left.\begin{aligned}&(\mathsf{params}, \{\mathbb{B}_\iota, \mathbb{B}^*_\iota\}_{\iota\in[0,2]}) \xleftarrow{\mathsf{R}} \mathcal{G}_{\mathsf{OB}}(2,(4,14,8));\\ &\widetilde{\mathbb{B}}^*_1 = \{\boldsymbol{b}^{*(1,1)}, \ldots, \boldsymbol{b}^{*(1,6)}, \boldsymbol{b}^{*(1,9)}, \ldots, \boldsymbol{b}^{*(1,14)}\};\\ &\varkappa, \{\breve{\sigma}_{\alpha,\nu}\}_{\nu\in[2]}, \{\breve{\mu}_{\iota,\nu}\}_{\iota\in[n]\setminus\{\alpha\},\nu\in[2]} \xleftarrow{\mathsf{U}} \mathbb{F}_q, \{\vec{\xi}^{(\alpha,\nu)}\}_{\nu\in[2]},\\ &\{\vec{\theta}^{(\iota,\nu)}\}_{\iota\in[n]\setminus\{\alpha\},\nu\in[2]}, \{\vec{\gamma}^{(\iota,\nu)}\}_{\iota\in[n]\setminus\{\alpha\},\nu\in[2]} \xleftarrow{\mathsf{U}} \mathbb{F}_q^2;\\ &\boldsymbol{h}^{*(0)} = \varkappa \boldsymbol{b}^{*(0,2)};\\ &\boldsymbol{h}^{*(1,\alpha,\nu)} = (\breve{\sigma}_{\alpha,\nu}(1,\alpha), \vec{0}^2, \vec{0}^2, \varkappa\vec{e}^{(2,\nu)}, \vec{0}^2, \vec{\xi}^{(\alpha,\nu)}, \vec{0}^2)_{\mathbb{B}^*_1}\\ &\qquad \text{for } \nu \in [2];\\ &\boldsymbol{f}^{(1,\iota,\nu,0)} = (\breve{\mu}_{\iota,\nu}(\iota,-1), \vec{0}^2, \vec{0}^6, \vec{0}^2, \vec{\gamma}^{(\iota,\nu)})_{\mathbb{B}_1}\\ &\boldsymbol{f}^{(1,\iota,\nu,1)} = (\breve{\mu}_{\iota,\nu}(\iota,-1), \vec{0}^2, \vec{0}^2, \vec{\theta}^{(\iota,\nu)}, \vec{0}^2, \vec{0}^2,\\ &\qquad \vec{\gamma}^{(\iota,\nu)})_{\mathbb{B}_1}\end{aligned}\right\} \begin{array}{l}\text{for}\\ \iota \in [n]\setminus\{\alpha\},\\ \nu \in [2];\end{array}$$
$$\check{\boldsymbol{h}}^{*(\nu)} = \varkappa \boldsymbol{b}^{*(1,\nu)} \text{ for } \nu \in \{5,6,9,10\}.$$

For any security parameter λ, for any $n = \mathsf{p}(\lambda)$, where p is an arbitrary polynomial, for any $\alpha \in [n]$, the advantage of any probabilistic adversary $\mathcal{B}$ in deciding Problem 5-α is defined as

$$\mathsf{Adv}^{\mathsf{P5}\text{-}\alpha}_{\mathcal{B}}(\lambda) = \left|\Pr\left[1 \xleftarrow{\mathsf{R}} \mathcal{B}(\varrho_0^{\mathsf{P5}\text{-}\alpha})\right] - \Pr\left[1 \xleftarrow{\mathsf{R}} \mathcal{B}(\varrho_1^{\mathsf{P5}\text{-}\alpha})\right]\right|.$$

Lemma 7: *For any probabilistic algorithm $\mathcal{B}$, there is a probabilistic algorithm $\mathcal{S}$, whose running time is essentially the same as that of $\mathcal{B}$, such that for any security parameter λ and any $n = \mathsf{p}(\lambda)$, $\mathsf{Adv}^{\mathsf{P5}\text{-}\alpha}_{\mathcal{B}}(\lambda) \leq \sum_{\iota\in[n]\setminus\{\alpha\},\nu\in[2]} \mathsf{Adv}^{\mathsf{SXDLIN}}_{\mathcal{S}_{\alpha\text{-}\iota\text{-}\nu}}(\lambda) + \mathsf{negl}(\lambda)$ for all $\alpha \in [n]$, where $\mathcal{S}_{\alpha\text{-}\iota\text{-}\nu}(\cdot) = \mathcal{S}(\alpha, \iota, \nu, \cdot)$ for any $\alpha, \iota, \nu \in \mathbb{N}$ and negl is some negligible function.*

Proof: Observe that Problem 5-α is essentially the same as

Basic Problem 5-p in [41], [42]. Hence, Lemma 4 can be proven in the same way as Lemma 39 in [42]. ∎

Definition 10 (Problem 6-α ($\alpha \in [n = \mathsf{p}(\lambda)]$)): Problem 6-$\alpha$ is to guess the bit $\widehat{\beta} \in \{0,1\}$ given $\varrho_{\widehat{\beta}}^{\mathsf{P6\text{-}}\alpha} = (\mathsf{params}, \{\mathbb{B}_\iota, \mathbb{B}_\iota^*\}_{\iota\in\{0,2\}}, \widetilde{\mathbb{B}}_1, \mathbb{B}_1^*, \{\boldsymbol{h}^{*(1,\alpha,\nu,\widehat{\beta})}\}_{\nu\in[2]})$, where

$$
\begin{aligned}
&(\mathsf{params}, \{\mathbb{B}_\iota, \mathbb{B}_\iota^*\}_{\iota\in[0,2]}) \xleftarrow{\mathsf{R}} \mathcal{G}_{\mathsf{OB}}(2,(4,14,8));\\
&\widetilde{\mathbb{B}}_1 = \{\boldsymbol{b}^{(1,1)}, \ldots, \boldsymbol{b}^{(1,6)}, \boldsymbol{b}^{(1,9)}, \ldots, \boldsymbol{b}^{(1,14)}\};\\
&\{\breve{\sigma}_{\alpha,\nu}\}_{\nu\in[2]}, \{\theta_{\alpha,\nu}\}_{\nu\in[2]} \xleftarrow{\mathsf{U}} \mathbb{F}_q, \{\vec{\xi}^{(\alpha,\nu)}\}_{\nu\in[2]} \xleftarrow{\mathsf{U}} \mathbb{F}_q^2;\\
&\left.\begin{aligned}
\boldsymbol{h}^{*(1,\alpha,\nu,0)} &= (\breve{\sigma}_{\alpha,\nu}(1,\alpha), \vec{0}^2, \vec{0}^6, \vec{\xi}^{(\alpha,\nu)}, \vec{0}^2)_{\mathbb{B}_1^*}\\
\boldsymbol{h}^{*(1,\alpha,\nu,1)} &= (\breve{\sigma}_{\alpha,\nu}(1,\alpha), \vec{0}^2, \vec{0}^2, \theta_{\alpha,\nu}\vec{e}^{(2,\nu)}, \vec{0}^2,\\
&\qquad \vec{\xi}^{(\alpha,\nu)}, \vec{0}^2)_{\mathbb{B}_1^*}
\end{aligned}\right\} \text{for } \nu \in [2].
\end{aligned}
$$

For any security parameter λ, for any $n = \mathsf{p}(\lambda)$, where p is an arbitrary polynomial, for any $\alpha \in [n]$, the advantage of any probabilistic adversary $\mathcal{B}$ in deciding Problem 6-α is defined as

$$
\mathsf{Adv}_{\mathcal{B}}^{\mathsf{P6\text{-}}\alpha}(\lambda) = \left|\mathsf{Pr}\left[1 \xleftarrow{\mathsf{R}} \mathcal{B}(\varrho_0^{\mathsf{P6\text{-}}\alpha})\right] - \mathsf{Pr}\left[1 \xleftarrow{\mathsf{R}} \mathcal{B}(\varrho_1^{\mathsf{P6\text{-}}\alpha})\right]\right|.
$$

Lemma 8: *For any probabilistic algorithm $\mathcal{B}$, there exists a probabilistic algorithm $\mathcal{S}$, whose running time is essentially the same as that of $\mathcal{B}$, such that for any security parameter λ and any $n = \mathsf{p}(\lambda)$, $\mathsf{Adv}_{\mathcal{B}}^{\mathsf{P6\text{-}}\alpha}(\lambda) \le \sum_{\nu\in[2]} \mathsf{Adv}_{\mathcal{S}_{\alpha\text{-}\nu}}^{\mathsf{SXDLIN}}(\lambda) + \mathsf{negl}(\lambda)$ for all $\alpha \in [n]$, where $\mathcal{S}_{\alpha\text{-}\nu}(\cdot) = \mathcal{S}(\alpha,\nu,\cdot)$ for any $\alpha, \nu \in \mathbb{N}$, and negl is some negligible function.*

Proof: Observe that Problem 6-α is essentially the same as Basic Problem 4-p in [41], [42]. Hence, the proof of Lemma 8 is similar to that of Lemma 38 in [42]. ∎

Definition 11 (Problem 7): Problem 7 is to guess the bit $\widehat{\beta} \in \{0,1\}$ given $\varrho_{\widehat{\beta}}^{\mathsf{P7}} = (\mathsf{params}, \{\mathbb{B}_\iota, \mathbb{B}_\iota^*\}_{\iota\in\{0,2\}}, \mathbb{B}_1, \widetilde{\mathbb{B}}_1^*, \{\boldsymbol{e}^{(1,\nu,\widehat{\beta})}\}_{\nu\in[3]})$, where

$$
\begin{aligned}
&(\mathsf{params}, \{\mathbb{B}_\iota, \mathbb{B}_\iota^*\}_{\iota\in[0,2]}) \xleftarrow{\mathsf{R}} \mathcal{G}_{\mathsf{OB}}(2,(4,14,8));\\
&\widetilde{\mathbb{B}}_1^* = \{\boldsymbol{b}^{*(1,1)}, \ldots, \boldsymbol{b}^{*(1,4)}, \boldsymbol{b}^{*(1,7)}, \boldsymbol{b}^{*(1,8)}, \boldsymbol{b}^{*(1,10)}, \ldots,\\
&\qquad \boldsymbol{b}^{*(1,14)}\};\\
&\{\theta_\nu\}_{\nu\in[3]} \xleftarrow{\mathsf{U}} \mathbb{F}_q, \{\vec{\gamma}^{(\nu)}\}_{\nu\in[3]} \xleftarrow{\mathsf{U}} \mathbb{F}_q^2;\\
&\left.\begin{aligned}
\boldsymbol{e}^{(1,\nu,0)} &= (\vec{0}^4, \vec{0}^6, \vec{0}^2, \vec{\gamma}^{(\nu)})_{\mathbb{B}_1}\\
\boldsymbol{e}^{(1,\nu,1)} &= (\vec{0}^4, \theta_\nu\vec{e}^{(2,\nu)}, \vec{0}^4, \vec{0}^2, \vec{\gamma}^{(\nu)})_{\mathbb{B}_1}
\end{aligned}\right\} \text{for } \nu \in [2];\\
&\boldsymbol{e}^{(1,3,0)} = (\vec{0}^4, \vec{0}^6, \vec{0}^2, \vec{\gamma}^{(3)})_{\mathbb{B}_1},\\
&\boldsymbol{e}^{(1,3,1)} = (\vec{0}^4, \vec{0}^4, \theta_3\vec{e}^{(2,1)}, \vec{0}^2, \vec{\gamma}^{(3)})_{\mathbb{B}_1}.
\end{aligned}
$$

For any security parameter λ, the advantage of any probabilistic adversary $\mathcal{B}$ in deciding Problem 7 is defined as

$$
\mathsf{Adv}_{\mathcal{B}}^{\mathsf{P7}}(\lambda) = \left|\mathsf{Pr}\left[1 \xleftarrow{\mathsf{R}} \mathcal{B}(\varrho_0^{\mathsf{P7}})\right] - \mathsf{Pr}\left[1 \xleftarrow{\mathsf{R}} \mathcal{B}(\varrho_1^{\mathsf{P7}})\right]\right|.
$$

Lemma 9: *For any probabilistic algorithm $\mathcal{B}$, there exist probabilistic algorithms $\mathcal{S}_1$, $\mathcal{S}_2$, and $\mathcal{S}_3$, whose running times are essentially the same as that of $\mathcal{B}$, such that for any security parameter λ, $\mathsf{Adv}_{\mathcal{B}}^{\mathsf{P7}}(\lambda) \le \sum_{\nu\in[3]} \mathsf{Adv}_{\mathcal{S}_\nu}^{\mathsf{SXDLIN}}(\lambda) + \mathsf{negl}(\lambda)$, where negl is some negligible function.*

Proof: Observe that Problem 7 is similar to Problem 1 in [10], [21]. Thus, the proof of Lemma 9 is analogous to that of Lemma 1 in [21]. ∎

Definition 12 (Problem 8): Problem 8 is to guess the bit $\widehat{\beta} \in \{0,1\}$ given $\varrho_{\widehat{\beta}}^{\mathsf{P8}} = (\mathsf{params}, \{\widetilde{\mathbb{B}}_\iota, \mathbb{B}_\iota^*\}_{\iota\in\{0,2\}}, \mathbb{B}_1, \mathbb{B}_1^*, \boldsymbol{h}^{*(0,\widehat{\beta})}, \boldsymbol{f}^{(0)}, \{\boldsymbol{h}^{*(2,\nu,\widehat{\beta})}, \boldsymbol{f}^{(2,\nu)}\}_{\nu\in[2]})$, where

$$
\begin{aligned}
&(\mathsf{params}, \{\mathbb{B}_\iota, \mathbb{B}_\iota^*\}_{\iota\in[0,2]}) \xleftarrow{\mathsf{R}} \mathcal{G}_{\mathsf{OB}}(2,(4,14,8));\\
&\widetilde{\mathbb{B}}_0 = \{\boldsymbol{b}^{(0,1)}, \boldsymbol{b}^{(0,3)}, \boldsymbol{b}^{(0,4)}\};\\
&\widetilde{\mathbb{B}}_2 = \{\boldsymbol{b}^{(2,1)}, \boldsymbol{b}^{(2,2)}, \boldsymbol{b}^{(2,5)}, \ldots, \boldsymbol{b}^{(2,8)}\};\\
&\vartheta, \varkappa, \delta, \tau, \xi_0 \xleftarrow{\mathsf{U}} \mathbb{F}_q, \{\vec{\xi}^{(\nu)}\}_{\nu\in[2]} \xleftarrow{\mathsf{U}} \mathbb{F}_q^2, \boldsymbol{X} \xleftarrow{\mathsf{U}} \mathsf{GL}(2,\mathbb{F}_q),\\
&\boldsymbol{Y} = (\boldsymbol{X}^{-1})^\top;\\
&\boldsymbol{h}^{*(0,0)} = (\vartheta, 0, \xi_0, 0)_{\mathbb{B}_0^*}, \boldsymbol{h}^{*(0,1)} = (\vartheta, \varkappa, \xi_0, 0)_{\mathbb{B}_0^*};\\
&\boldsymbol{f}^{(0)} = (\delta, \tau, 0, 0)_{\mathbb{B}_0};\\
&\left.\begin{aligned}
\boldsymbol{h}^{*(2,\nu,0)} &= (\vartheta\vec{e}^{(2,\nu)}, \vec{0}^2, \vec{\xi}^{(\nu)}, \vec{0}^2)_{\mathbb{B}_2^*}\\
\boldsymbol{h}^{*(2,\nu,1)} &= (\vartheta\vec{e}^{(2,\nu)}, \varkappa\vec{e}^{(2,\nu)}\boldsymbol{X}, \vec{\xi}^{(\nu)}, \vec{0}^2)_{\mathbb{B}_2^*}\\
\boldsymbol{f}^{(2,\nu)} &= (\delta\vec{e}^{(2,\nu)}, \tau\vec{e}^{(2,\nu)}\boldsymbol{Y}, \vec{0}^2, \vec{0}^2)_{\mathbb{B}_2}
\end{aligned}\right\} \text{for } \nu \in [2].
\end{aligned}
$$

For any security parameter λ, the advantage of any probabilistic adversary $\mathcal{B}$ in deciding Problem 8 is defined as

$$
\mathsf{Adv}_{\mathcal{B}}^{\mathsf{P8}}(\lambda) = \left|\mathsf{Pr}\left[1 \xleftarrow{\mathsf{R}} \mathcal{B}(\varrho_0^{\mathsf{P8}})\right] - \mathsf{Pr}\left[1 \xleftarrow{\mathsf{R}} \mathcal{B}(\varrho_1^{\mathsf{P8}})\right]\right|.
$$

Lemma 10: *For any probabilistic algorithm $\mathcal{B}$, there exists a probabilistic algorithm $\mathcal{S}$, whose running time is essentially the same as that of $\mathcal{B}$, such that for any security parameter λ, $\mathsf{Adv}_{\mathcal{B}}^{\mathsf{P8}}(\lambda) \le \mathsf{Adv}_{\mathcal{S}}^{\mathsf{SXDLIN}}(\lambda) + \mathsf{negl}(\lambda)$, where negl is some negligible function.*

Proof: Observe that Problem 8 is essentially the same as Problem 3 in [10], [21]. Thus, the proof of Lemma 10 is similar to that of Lemma 3 in [21]. ∎

2.5 Collision-Resistant Hash Functions

Here we will formally describe the notion of collision-resistant hash functions which will be used as an ingredient of our ABS construction.

▷ Syntax

A hash function family $\mathbb{H}$ associated with a bilinear group generator $\mathcal{G}_{\mathsf{BPG}}$ and a polynomial $\mathsf{poly}(\cdot)$ consists of the following two polynomial-time algorithms:

$\mathsf{KGen}()$: The hashing key generation algorithm is a probabilistic algorithm that takes as input the unary encoded

security parameter 1^λ, and samples a hashing key hk from the key space $\mathbb{HK}_\lambda$, which is a probability space over bit strings parameterized by λ.

$\mathsf{H}_{\mathsf{hk}}^{(\lambda,\mathsf{poly})} : \mathbb{D} = \{0,1\}^{\mathsf{poly}(\lambda)} \to \mathbb{F}_q \backslash \{0\}$: A deterministic function that maps an element of $\mathbb{D} = \{0,1\}^{\mathsf{poly}(\lambda)}$ to an element of $\mathbb{F}_q \backslash \{0\}$ with q being the first element of the output $\mathsf{params}_{\mathbb{G}} = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e)$ of $\mathcal{G}_{\mathsf{BPG}}$ on input 1^λ.

▷ Collision Resistance

A hash function family $\mathbb{H}$ associated with $\mathcal{G}_{\mathsf{BPG}}$ and $\mathsf{poly}(\cdot)$ is said to be collision resistant if for any PPT adversary $\mathcal{M}$, for any security parameter λ and any $\mathsf{hk} \xleftarrow{R} \mathsf{KGen}()$, the advantage of $\mathcal{M}$ in finding a collision, defined as

$$\mathsf{Adv}_{\mathcal{M}}^{\mathsf{H,CR}}(\lambda) = \mathsf{Pr}[\Upsilon_1, \Upsilon_2 \in \mathbb{D} = \{0,1\}^{\mathsf{poly}(\lambda)} \wedge \Upsilon_1 \neq \Upsilon_2 \wedge \mathsf{H}_{\mathsf{hk}}^{(\lambda,\mathsf{poly})}(\Upsilon_1) = \mathsf{H}_{\mathsf{hk}}^{(\lambda,\mathsf{poly})}(\Upsilon_2) \mid (\Upsilon_1, \Upsilon_2) \xleftarrow{R} \mathcal{M}(\mathsf{hk}, \mathbb{D})]$$

is negligible, i.e., $\mathsf{Adv}_{\mathcal{M}}^{\mathsf{H,CR}}(\lambda) \leq \mathsf{negl}(\lambda)$, where negl is some negligible function.

2.6 The Notion of Attribute-Based Signatures for Arithmetic Branching Programs

Let for some prime $q \in \mathbb{N}$, $\mathcal{F}_{\mathsf{ABP}}^{(q)}$ denote the class of all functions $f : \mathbb{F}_q^n \to \mathbb{F}_q$ for any $n = \mathsf{p}(\lambda) \in \mathbb{N}$, where p is an arbitrary polynomial, realizable by some ABP of polynomial size over $\mathbb{F}_q$. In this section, we will formally define the notion of an attribute-based signature (ABS) scheme for the predicate family $\mathcal{R}_{\mathsf{Z\text{-}ABP}}^{(q)}$ defined as $\mathcal{R}_{\mathsf{Z\text{-}ABP}}^{(q)} = \{R_{\mathsf{Z\text{-}ABP}}^{(q)}(f,\cdot) : \mathbb{F}_q^n \to \{0,1\} \mid f : \mathbb{F}_q^n \to \mathbb{F}_q \in \mathcal{F}_{\mathsf{ABP}}^{(q)}\}$, where $R_{\mathsf{Z\text{-}ABP}}^{(q)}(f,\vec{x}) = 1$ if $f(\vec{x}) = 0$, and $R_{\mathsf{Z\text{-}ABP}}^{(q)}(f,\vec{x}) = 0$ otherwise for all $f : \mathbb{F}_q^n \to \mathbb{F}_q \in \mathcal{F}_{\mathsf{ABP}}^{(q)}$ and $\vec{x} \in \mathbb{F}_q^n$. As stated in Lemma 2, there exists a polynomial-time algorithm that on input any $f : \mathbb{F}_q^n \to \mathbb{F}_q \in \mathcal{F}_{\mathsf{ABP}}^{(q)}$, constructs an ASP $\mathbb{S} = (\mathbb{U}, \rho)$ such that for any $\vec{x} \in \mathbb{F}_q^n$, it holds that $R_{\mathsf{Z\text{-}ABP}}^{(q)}(f,\vec{x}) = 1 \iff f(\vec{x}) = 0 \iff \mathbb{S}$ accepts $\vec{x}$. Therefore, for the rest of this paper, we will identify predicates $R_{\mathsf{Z\text{-}ABP}}^{(q)}(f,\cdot) \in \mathcal{R}_{\mathsf{Z\text{-}ABP}}^{(q)}$ by their corresponding ASP-representations $\mathbb{S} = (\mathbb{U}, \rho)$ computed using the algorithm of Lemma 2.

▷ Syntax

An *attribute-based signature* (ABS) scheme for some predicate family $\mathcal{R}_{\mathsf{Z\text{-}ABP}}^{(q)}$ consists of an associated message space $\mathbb{M} \subseteq \{0,1\}^*$, a signature space Σ, along with the following PPT algorithms:

$\mathsf{ABS.Setup}()$: The setup algorithm takes as input the unary encoded security parameter 1^λ. It outputs the public parameters MPK and the master signing key MSK.

$\mathsf{ABS.KeyGen}(\mathsf{MPK}, \mathsf{MSK}, \vec{x})$: The signing key generation algorithm takes as input the public parameters MPK, the master signing key MSK, along with a signing attribute vector $\vec{x} \in \mathbb{F}_q^n$ for some $n = \mathsf{p}(\lambda) \in \mathbb{N}$. It outputs a signing key $\mathsf{SK}(\vec{x})$.

$\mathsf{ABS.Sign}(\mathsf{MPK}, \vec{x}, \mathsf{SK}(\vec{x}), \mathbb{S}, \mathsf{MSG})$: The signing algorithm takes as input the public parameters MPK, a signing attribute string $\vec{x} \in \mathbb{F}_q^n$ for some $n = \mathsf{p}(\lambda) \in \mathbb{N}$, a signing key $\mathsf{SK}(\vec{x})$ for $\vec{x}$, a signing policy $R_{\mathsf{Z\text{-}ABP}}^{(q)}(f,\cdot) : \mathbb{F}_q^n \to \{0,1\} \in \mathcal{R}_{\mathsf{Z\text{-}ABP}}^{(q)}$ represented as an ASP $\mathbb{S} = (\mathbb{U}, \rho)$, and a message $\mathsf{MSG} \in \mathbb{M}$. It outputs either a signature $\mathsf{sig} \in \Sigma$ or the distinguished symbol $\perp$ indicating failure.

$\mathsf{ABS.Verify}(\mathsf{MPK}, \mathbb{S}, (\mathsf{MSG}, \mathsf{sig}))$: The verification algorithm takes as input the public parameters MPK, a signing policy $R_{\mathsf{Z\text{-}ABP}}^{(q)}(f,\cdot) \in \mathcal{R}_{\mathsf{Z\text{-}ABP}}^{(q)}$ represented as an ASP $\mathbb{S} = (\mathbb{U}, \rho)$, and a message-signature pair $(\mathsf{MSG}, \mathsf{sig}) \in \mathbb{M} \times \Sigma$. It outputs either 1 or 0.

▷ Correctness

An ABS scheme for some predicate family $\mathcal{R}_{\mathsf{Z\text{-}ABP}}^{(q)}$ is said to be correct if for any security parameter λ, any $n = \mathsf{p}(\lambda) \in \mathbb{N}$, any signing policy predicate $R_{\mathsf{Z\text{-}ABP}}^{(q)}(f,\cdot) : \mathbb{F}_q^n \to \{0,1\} \in \mathcal{R}_{\mathsf{Z\text{-}ABP}}^{(q)}$ represented as an ASP $\mathbb{S} = (\mathbb{U}, \rho)$, any signing attribute vector $\vec{x} \in \mathbb{F}_q^n$, any $(\mathsf{MPK}, \mathsf{MSK}) \xleftarrow{R} \mathsf{ABS.Setup}()$, and any $\mathsf{SK}(\vec{x}) \xleftarrow{R} \mathsf{ABS.KeyGen}(\mathsf{MPK}, \mathsf{MSK}, \vec{x})$, if $\mathbb{S}$ accepts $\vec{x}$, then

$$\mathsf{Pr}[1 \xleftarrow{R} \mathsf{ABS.Verify}(\mathsf{MPK}, \mathbb{S}, (\mathsf{MSG}, \mathsf{sig})) \mid \mathsf{sig} \xleftarrow{R} \mathsf{ABS.Sign}(\mathsf{MPK}, \vec{x}, \mathsf{SK}(\vec{x}), \mathbb{S}, \mathsf{MSG})] \geq 1 - \mathsf{negl}(\lambda),$$

where negl is some negligible function, and the probability is taken over the random coins of ABS.Sign and ABS.Verify.

▷ Signer Privacy

An ABS scheme for some predicate family $\mathcal{R}_{\mathsf{Z\text{-}ABP}}^{(q)}$ is said to achieve *perfect* signer privacy if for any security parameter λ, any $n = \mathsf{p}(\lambda) \in \mathbb{N}$, any message $\mathsf{MSG} \in \mathbb{M}$, any $(\mathsf{MPK}, \mathsf{MSK}) \xleftarrow{R} \mathsf{ABS.Setup}()$, any signing policy $R_{\mathsf{Z\text{-}ABP}}^{(q)}(f,\cdot) : \mathbb{F}_q^n \to \{0,1\} \in \mathcal{R}_{\mathsf{Z\text{-}ABP}}^{(q)}$ having ASP representation $\mathbb{S} = (\mathbb{U}, \rho)$, any two signing attribute vectors $\vec{x}, \vec{x}' \in \mathbb{F}_q^n$ such that $\mathbb{S}$ accepts both $\vec{x}$ and $\vec{x}'$, any signing keys $\mathsf{SK}(\vec{x}) \xleftarrow{R} \mathsf{ABS.KeyGen}(\mathsf{MPK}, \mathsf{MSK}, \vec{x})$, $\mathsf{SK}(\vec{x}') \xleftarrow{R} \mathsf{ABS.KeyGen}(\mathsf{MPK}, \mathsf{MSK}, \vec{x}')$, the distributions of the signatures outputted by $\mathsf{ABS.Sign}(\mathsf{MPK}, \vec{x}, \mathsf{SK}(\vec{x}), \mathbb{S}, \mathsf{MSG})$ and $\mathsf{ABS.Sign}(\mathsf{MPK}, \vec{x}', \mathsf{SK}(\vec{x}'), \mathbb{S}, \mathsf{MSG})$ are equivalent.

▷ Existential Unforgeability

Existential unforgeability of an ABS scheme for some predicate class $\mathcal{R}_{\mathsf{Z\text{-}ABP}}^{(q)}$ against adaptive-predicate-adaptive-message attack is defined through the following experiment between a stateful probabilistic adversary $\mathcal{A}$ and a stateful probabilistic challenger $\mathcal{B}$:

- $\mathcal{B}$ generates $(\mathsf{MPK}, \mathsf{MSK}) \xleftarrow{R} \mathsf{ABS.Setup}()$ and sends MPK to $\mathcal{A}$.

- $\mathcal{A}$ may adaptively make any polynomial number of queries of the following types to $\mathcal{B}$:
  - *Signing Key Generation Query*: When $\mathcal{A}$ requests the generation of a signing key for some signing attribute vector $\vec{x} \in \mathbb{F}_q^n$ for some $n = \mathsf{p}(\lambda) \in \mathbb{N}$, $\mathcal{B}$ generates a signing key $\mathsf{SK}(\vec{x}) \xleftarrow{R} \mathsf{ABS.KeyGen}(\mathsf{MPK}, \mathsf{MSK}, \vec{x})$ and stores the signing key $\mathsf{SK}(\vec{x})$.
  - *Signature Generation Query*: When $\mathcal{A}$ specifies a signing key for some signing attribute vector $\vec{x} \in \mathbb{F}_q^n$ for some $n = \mathsf{p}(\lambda) \in \mathbb{N}$ that it has already requested $\mathcal{B}$ to generate, and requests the generation of a signature using that signing key on some message $\mathsf{MSG} \in \mathbb{M}$ under some signing policy $R^{(q)}_{\mathsf{Z\text{-}ABP}}(f, \cdot) : \mathbb{F}_q^n \to \{0,1\} \in \mathcal{R}^{(q)}_{\mathsf{Z\text{-}ABP}}$ represented as an ASP $\mathbb{S} = (\mathbb{U}, \rho)$ such that $\mathbb{S}$ accepts $\vec{x}$, $\mathcal{B}$ creates a signature $\mathsf{sig} \xleftarrow{R} \mathsf{ABS.Sign}(\mathsf{MPK}, \vec{x}, \mathsf{SK}(\vec{x}), \mathbb{S}, \mathsf{MSG})$ and stores it.
  - *Signing Key/Signature Reveal Query*: When $\mathcal{A}$ requests $\mathcal{B}$ to reveal an already created signing key corresponding to some signing attribute vector $\vec{x} \in \mathbb{F}_q^n$ for some $n = \mathsf{p}(\lambda) \in \mathbb{N}$ or an already created signature on some message $\mathsf{MSG} \in \mathbb{M}$ under some signing policy $R^{(q)}_{\mathsf{Z\text{-}ABP}}(f, \cdot) : \mathbb{F}_q^n \to \{0,1\} \in \mathcal{R}^{(q)}_{\mathsf{Z\text{-}ABP}}$ for some $n = \mathsf{p}(\lambda) \in \mathbb{N}$ represented by an ASP $\mathbb{S} = (\mathbb{U}, \rho)$, $\mathcal{B}$ provides $\mathcal{A}$ with the respective queried item.

  We would like to emphasize that when a signing key or signature generation query is made, $\mathcal{A}$ does not receives the signing key or signature that $\mathcal{B}$ creates. $\mathcal{A}$ receives it only when it makes a reveal query for that signing key or signature.

- At the end of interaction $\mathcal{A}$ outputs a triplet $(\mathbb{S}, \mathsf{MSG}, \mathsf{sig})$, where $\mathbb{S}$ is the ASP-representation of a signing policy $R^{(q)}_{\mathsf{Z\text{-}ABP}}(f, \cdot) : \mathbb{F}_q^n \to \{0,1\} \in \mathcal{R}^{(q)}_{\mathsf{Z\text{-}ABP}}$ for some $n = \mathsf{p}(\lambda) \in \mathbb{N}$, $\mathsf{MSG} \in \mathbb{M}$, and $\mathsf{sig} \in \Sigma$. $\mathcal{A}$ wins if the following conditions hold simultaneously:
  (a) $1 = \mathsf{ABS.Verify}(\mathsf{MPK}, \mathbb{S}, (\mathsf{MSG}, \mathsf{sig}))$.
  (b) $\mathcal{A}$ has not made a signature reveal query on MSG under $\mathbb{S}$.
  (c) $\mathbb{S}$ does not accept any signing attribute string $\vec{x} \in \mathbb{F}_q^n$ for which $\mathcal{A}$ has requested to reveal a signing key.

An ABS scheme for some predicate family $\mathcal{R}^{(q)}_{\mathsf{Z\text{-}ABP}}$ is said to be existentially unforgeable against adaptive-predicate-adaptive-message attack if for any PPT adversary $\mathcal{A}$, for any security parameter λ, the advantage of $\mathcal{A}$ in the above experiment, defined as

$$\mathsf{Adv}^{\mathsf{ABS,UF}}_{\mathcal{A}}(\lambda) = \Pr\left[\mathcal{A} \text{ wins in the unforgeability experiment}\right]$$

is negligible in λ, i.e., $\mathsf{Adv}^{\mathsf{ABS,UF}}_{\mathcal{A}}(\lambda) \leq \mathsf{negl}(\lambda)$, where negl is some negligible function.

Remark 1: Note that following [10], we distinguish between the generation and reveal queries made by the adversary for both signing keys and signatures in the existential unforgeability experiment, as can be seen above. As explained in [10], the reason for making such distinction for the signing keys is as follows. Suppose we do not differentiate between the signing key generation and signing key reveal queries. This means the challenger never stores any signing key it generates for the adversary without revealing it to the adversary. Consequently, according to the restriction of the experiment, the adversary can never request the challenger to generate a signing key corresponding to some signing attribute vector $\vec{x} \in \mathbb{F}_q^n$ which is accepted by the signing policy $R^{(q)}_{\mathsf{Z\text{-}ABP}}(f, \cdot) : \mathbb{F}_q^n \to \{0,1\} \in \mathcal{R}^{(q)}_{\mathsf{Z\text{-}ABP}}$., under which it claims the forgery at the end of the experiment,prior to requesting a signature on some message $\mathsf{MSG} \in \mathbb{M}$ under the signing policy $R^{(q)}_{\mathsf{Z\text{-}ABP}}(f, \cdot)$. In order to answer such a signature query, the challenger then needs to find out some signing attribute vector $\vec{x} \in \mathbb{F}_q^n$ satisfying the signing policy $R^{(q)}_{\mathsf{Z\text{-}ABP}}(f, \cdot)$ on its own. Unfortunately however, the challenger may not always find a suitable $\vec{x} \in \mathbb{F}_q^n$ in a polynomial time since it involves the satisfiability problem for polynomial-size ABPs or ASPs. This issue makes combining the signing key generation and reveal queries problematic. In contrast, in our unforgeability experiment, the adversary can first make a signing key generation query for some attribute vector $\vec{x} \in \mathbb{F}_q^n$ satisfying the signing policy $R^{(q)}_{\mathsf{Z\text{-}ABP}}(f, \cdot)$ to the challenger without ever asking to reveal the generated signing key. After that the adversary can send a signature query to the challenger for some message $\mathsf{MSG} \in \mathbb{M}$ under the signing policy $R^{(q)}_{\mathsf{Z\text{-}ABP}}(f, \cdot)$ and can simply instruct the challenger to use that already generated signing key corresponding to the attribute vector $\vec{x}$ to generate the signature. As a result, the challenger does not need to solve any satisfiability problem to answer the signature query. Note that an analogous approach was also considered by Shi and Waters [43] while formulating the security definition for key-delegation. While the above argument justifies the distinction between the generation and reveal queries for the signing keys, it may still be possible to combine the generation and reveal queries in case of signatures. However, following [10], we chose to make the distinction for the signatures as well without the loss of generality in order to maintain some uniformity in the experiment description.

3. Overview of Our Techniques

In this section, we provide an intuitive exposition to the main technical ideas in this paper. In order to design our ABS scheme for ABP's, we start with the high level approach adopted by Okamoto and Takashima [10], [11]. At the top level of strategy, this approach considers an extension of the Naor's paradigm, which was originally proposed for converting an identity-based encryption (IBE) scheme to a digital signature scheme. The idea is to build a signature-policy

ABS scheme by augmenting a ciphertext-policy attribute-based encryption (ABE) scheme [38], [44].

Just like a signature-policy ABS scheme, a ciphertext-policy ABE scheme has an associated predicate family $\mathcal{R} = \{R(Y, \cdot) : \mathcal{X} \to \{0, 1\} \mid Y \in \mathcal{Y}\}$, where $\mathcal{X}$ and $\mathcal{Y}$ comprise respectively of the admissible decryption attributes and policies. A central authority holds a master secret key and publishes public system parameters. Anyone can encrypt a message, which is also referred to as a payload, with respect to any decryption policy $Y \in \mathcal{Y}$ using solely the public parameters. A decrypter may obtain a restricted decryption key from the authority corresponding to the attributes $X \in \mathcal{X}$ it possesses. Using such a restricted decryption key for $X \in \mathcal{X}$ the decrypter can recover the payload from only those ciphertexts which are generated with respect to a policy $Y \in \mathcal{Y}$ such that $R(Y, X) = 1$. In particular, it is (computationally) infeasible to decrypt a ciphertext generated with respect to some decryption policy $Y \in \mathcal{Y}$ for any collection of colluding decrypters, none of whom individually possesses an attribute that satisfies Y, by pooling their attributes together. An ABE ciphertext contains the associated decryption policy in the clear, and hence this security property of an ABE scheme is referred to as *payload hiding*.

Roughly speaking, in the approach of Okamoto and Takashima [10], [11], a signing key for some signing attribute $X \in \mathcal{X}$ in the ABS scheme corresponds to a decryption key for X in the underlying ABE scheme. On the other hand, a signature on some message MSG under some claimed signing policy $Y \in \mathcal{Y}$ is verified by generating a verification-text that corresponds to a ciphertext of MSG under Y in the underlying ABE scheme. The most challenging part of this approach is that no straightforward counter part of a signature in ABS exists in ABE, and moreover, the privacy property of signatures, which is a vital requirement of an ABS scheme has no corresponding notion in ABE. In order to tackle these issues, Okamoto and Takashima [10], [11] devised a novel technique, which they termed as "rerandomization with specialized delegation", where a signature in the ABS scheme generated with respect to some signing policy Y using a signing key for some attribute X can be interpreted to be a random ABE decryption key specialized to decrypt only those ABE ciphertexts which have Y as the associated decryption policy. As for the security of the resulting ABS scheme, the idea is to reduce the unforgeability of the ABS scheme to the payload-hiding security of the underlying ABE scheme. On the other hand, the signer privacy is ensured by the careful rerandomized delegation procedure employed in the generation of signatures. While this high level description of the approach may sound quite simple, the actual realization, however, is quite delicate and involves many subtle aspects. Okamoto and Takashima [10], [11] addressed those technical hurdles in the context of boolean span programs using various additional ideas.

We first explain how we adopt the above high level construction methodology to the context of ABP's, which is a rather non-trivial task. In order to design our scheme,we utilize the machineries of the *dual pairing vector spaces* (DPVS) [37], [38]. A highly powerful feature of DPVS is that one can completely or partially hide a linear subspace of the whole vector space by concealing the basis of that subspace or the basis of its dual subspace respectively from the public parameters. In DPVS-based constructions, a collection of pairs of mutually dual vector spaces $\{\mathbb{V}_\iota, \mathbb{V}_\iota^*\}_{\iota \in [N]}$ along with a bilinear pairing $e : \mathbb{V}_\iota \times \mathbb{V}_\iota^* \to \mathbb{G}_T$ for all $\iota \in [N]$, constructed from a standard bilinear group $\mathsf{params}_{\mathbb{G}} = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e)$ of prime order q is used. Typically, for all $\iota \in [N]$, a pair of dual orthonormal bases $(\mathbb{B}_\iota, \mathbb{B}_\iota^*)$ of $(\mathbb{V}_\iota, \mathbb{V}_\iota^*)$ is generated using a secret random invertible linear transformation $\boldsymbol{B}^{(\iota)}$ over $\mathbb{F}_q$ during setup, and portions of $(\mathbb{B}_\iota, \mathbb{B}_\iota^*)$, say $(\widehat{\mathbb{B}}_\iota, \widehat{\mathbb{B}}_\iota^*)$ for $\iota \in [N]$ are used as the public parameters. Thus, the remaining portions of the bases $(\mathbb{B}_\iota \backslash \widehat{\mathbb{B}}_\iota, \mathbb{B}_\iota^* \backslash \widehat{\mathbb{B}}_\iota^*)$ for $\iota \in [N]$ remain hidden from the outside world. This provides a strong framework for various kinds of information-theoretic tricks in the public-key setting by exploiting various nice properties of linear transformations.

In order to extend the techniques of Okamoto and Takashima [10], [11] to the setting of ABP's, we first look for a representation of ABP's using some span program like structure, which supports "linear reconstruction". The linear reconstruction property is important for our scheme since we need to reconstruct some secrets in the exponents of group elements. We observe that Ishai and Wee [36] have devised a polynomial-time algorithm that given an ABP f, outputs an *arithmetic span program* (ASP) $\mathbb{S} = (\mathbb{U}, \rho)$ such that for any $\vec{x} \in \mathbb{F}_q^n$, $f(\vec{x}) = 0 \iff \mathbb{S}$ accepts $\vec{x}$. ASP's are the arithmetic counter part of boolean span programs. An ASP $\mathbb{S}$ is described as a pair $\mathbb{S} = (\mathbb{U}, \rho)$, where $\mathbb{U}$ is a set of pairs of vectors $\mathbb{U} = \{(\vec{y}^{(j)}, \vec{z}^{(j)})\}_{j \in [m]} \subset (\mathbb{F}_q^\ell)^2$ for some $\ell, m \in \mathbb{N}$ and ρ is a mapping $\rho : [m] \to [n]$. $\mathbb{S}$ accepts $\vec{x} \in \mathbb{F}_q^n \iff \vec{e}^{(\ell,\ell)} \in \mathsf{SPAN}\langle x_{\rho(j)} \vec{y}^{(j)} + \vec{z}^{(j)} \mid j \in [m]\rangle$, where $\vec{e}^{(\ell,\ell)} = (\overbrace{0, \ldots, 0}^{\ell-1}, 1)$ and SPAN refers to the standard linear span of vectors. With this representation at hand, we proceed to extending the ABS scheme of Okamoto and Takashima [10], [11] to the ABP setting.

The most important difficulty we face here is with the application of the rerandomization with special delegation technique to generate the signatures due to a fundamental difference in the structures of the boolean and arithmetic span programs. Recall that a boolean span program over n boolean variables is represented as $\mathbb{P} = (\boldsymbol{P} \in \mathbb{F}_q^{m \times \ell}, \rho : [m] \to [n])$, and $\mathbb{P}$ accepts a boolean string $\vec{x} \in \mathbb{F}_2^n \iff \vec{e}^{(\ell,\ell)} \in \mathsf{SPAN}\langle \vec{p}^{(j)} \mid j \in [m] \wedge x_{\rho(j)} = 1\rangle$, where $\vec{p}^{(j)} \in \mathbb{F}_q^\ell$ is the j^{th} row vector of $\boldsymbol{P}$. This means while evaluating a boolean span program on some input, the input only determines which vectors are to be included in the linear span and does not affect the description of the included vectors as such. Roughly speaking, in the ABS construction of [10], [11], the randomized special delegation is applied by masking the actual coefficients $(\Omega_j)_{j \in [m]} \in \mathbb{F}_q^m$ of the linear span of the vectors $\{\vec{p}^{(j)}\}_{j \in [m]}$ of a signing policy $\mathbb{P} = (\boldsymbol{P} \in \mathbb{F}_q^{m \times \ell}, \rho)$ resulting in the vector $\vec{e}^{(\ell,\ell)}$ when $\mathbb{P}$ accepts some boolean signing attribute string $\vec{x} \in \mathbb{F}_2^n$, with the coefficients $(\Omega'_j)_{j \in [m]} \in \mathbb{F}_q^m$

of some random linear combination of the vectors $\{\vec{p}^{(j)}\}_{j\in[m]}$ that results in the zero vector $\vec{0}^{\ell}$. More precisely, while generating a signature under $\mathbb{P} = (\boldsymbol{P}, \rho)$ using a secret key for $\vec{x} \in \mathbb{F}_2^n$, one computes $\Omega_j + \Omega'_j$ for all $j \in [m]$. This rerandomization works for ensuring signer privacy, i.e., for erasing the information of the specific signing attribute string $\vec{x} \in \mathbb{F}_2^n$ from the signature for boolean span programs because seeing the rerandomized coefficients $(\Omega_j + \Omega'_j)_{j\in[m]}$, one cannot decide which Ω_j's were 0 in the real linear span, and hence the information of the actual boolean attribute string $\vec{x} \in \mathbb{F}_2^n$ is completely erased via this rerandomization.

This rerandomization technique is, however, no longer sufficient in case of ASP's. This is because, while evaluating an ASP $\mathbb{S} = (\mathbb{U} = \{(\vec{y}^{(j)}, \vec{z}^{(j)})\}_{j\in[m]} \subset (\mathbb{F}_q^{\ell})^2, \rho : [m] \to [n])$ on some input vector $\vec{x} \in \mathbb{F}_q^n$, the description of the vectors, whose linear span needs to be considered, namely, the vectors $\{x_{\rho(j)}\vec{y}^{(j)} + \vec{z}^{(j)}\}_{j\in[m]}$ itself depends on the specific input vector $\vec{x} \in \mathbb{F}_q^n$ used. Therefore, even if the above randomized masking is applied, the result would still leak information of the specific vector $\vec{x}$ used.

In order to overcome this issue, we apply a more clever rerandomization. Roughly speaking, we randomize not only the linear-combination-coefficients, but also the input values $\{x_{\rho(j)}\}_{j\in[m]}$. We consider a random linear combination of the vectors $\{\vec{y}^{(j)}, \vec{z}^{(j)}\}_{j\in[m]}$ that leads to the zero vector $\vec{0}^{\ell}$, i.e., we compute random $((\Omega'_j)_{j\in[m]}, (\Omega''_j)_{j\in[m]})$ such that $\sum_{j\in[m]} (\Omega'_j\vec{y}^{(j)} + \Omega''_j\vec{z}^{(j)}) = \vec{0}^{\ell}$. Then, we use the scalars $(\Omega'_j)_{j\in[m]}$ to mask $(\Omega_j x_{\rho(j)})_{j\in[m]}$ and $(\Omega''_j)_{j\in[m]}$ to mask $(\Omega_j)_{j\in[m]}$, where $(\Omega_j)_{j\in[m]}$ are the coefficients of the vectors $\{x_{\rho(j)}\vec{y}^{(j)} + \vec{z}^{(j)}\}_{j\in[m]}$ in the linear combination resulting in $\vec{e}^{(\ell,\ell)}$. More precisely, while generating a signature under some ASP $\mathbb{S}$ using a signing key for $\vec{x} \in \mathbb{F}_q^n$, we compute $\Omega_j x_{\rho(j)} + \Omega'_j$ and $\Omega_j + \Omega''_j$ for all $j \in [m]$. Observe that this rerandomization not only erases the actual values of the linear combination coefficients $(\Omega_j)_{j\in[m]}$ but also the information of the actual input $\vec{x}$ for which the linear combination is evaluated.

Now, note that unlike the schemes of [10], [11], in which the size and input length of the supported span programs are bounded by the public parameters, our goal is to support ABP's, and hence ASP's by the above discussion, of unbounded size and input length. For this, we start by extending the techniques called "indexing" and "consistent randomness amplification", developed by Okamoto and Takashima in [41] in the context of ABE for boolean span programs, to our setting of ASP's. Roughly speaking, in the ABS constructions of [10], [11], once parts of a set of pairs of dual orthonormal bases $\{\widehat{\mathbb{B}}_\iota, \widehat{\mathbb{B}}^*_\iota\}_{\iota\in[n]}$ are published as the public parameters, the input length of the signing policy span programs becomes fixed to n. The proof of adaptive unforgeability of the scheme follows the so called "dual system encryption" methodology [45], [46], and crucially makes use of certain information-theoretic arguments. The randomness of the secret linear transformations $\{\boldsymbol{B}^{(\iota)}\}_{\iota\in[n]}$ used to generate the bases $\{\mathbb{B}_\iota, \mathbb{B}^*_\iota\}_{\iota\in[n]}$, whose parts are included in the public parameters, acts as the source of entropy for those information-theoretic arguments.

In contrast, in the unbounded setting, the input length of the signing policy span programs are not fixed by the public parameters. In particular, in our unbounded ABS scheme, the public parameters would only consist of a constant number of pairs of dual orthonormal bases. Thus, the randomness contained in the public parameters (which is just a constant amount with respect to the length of the input attribute vectors n) is clearly insufficient for the dual system encryption arguments on adaptive security. To supply the additional randomness required for the security reduction, we adopt the indexing technique of [41], and for all $\iota \in [n]$, embed two dimensional prefix vectors $\sigma_\iota(1, \iota)$ and $\mu_j(\iota, -1)$ within the components corresponding to the ι^{th} attribute in signing keys and verification-texts respectively, where σ_ι and μ_j are freshly sampled random elements of $\mathbb{F}_q$. However, this method of supplying linear-in-n amount of additional randomness is still not sufficient. This is because, for the application of the dual system encryption methodology, such randomness introduced by the indexing technique needs to be expanded to the hidden subspaces of signing keys and verification-texts, and the distribution of the expanded randomness should also be adjusted to the conditions imposed on the queries of the adversary in the unforgeability experiment. To resolve the problem, we attempt to employ the consistent randomness amplification technique similar to [41].

However, recall that our objective is not limited to only supporting signing policies of unbounded size and input length. We additionally want to allow unbounded multi-use of attributes inside the signing policies. As we explain below, the consistent randomness amplification technique of Okamoto and Takashima [41] does not suffice for achieving both these goals simultaneously. Therefore, we need to innovate new technical ideas to accomplish our target. In terms of technicality, this is the most sophisticated part of this paper. In fact, the techniques we devise in this segment are pretty much general, and we strongly believe they will find more applications in various other DPVS-based construction in the future.

Roughly speaking, the single use restriction in DPVS-based adaptively secure constructions of attribute-based primitives arises from the use of a crucial information-theoretic lemma, the so called "pairwise independence lemma" (Lemma 3 in [38]), while employing the dual system encryption paradigm in the security proofs. This technique requires a one-to-one correspondence between a pair of a key part and a verification-text or ciphertext part through the map ρ of the policy span program considered. However, in the multi-use scenario, one key part corresponds to multiple verification-text or ciphertext parts. Even when a generalized version of the pairwise independence lemma [38] is used, the maximum number of times an attribute can be used inside a policy span program remains bounded by the public parameters. In the domain of ABE, some initial attempts were made to mitigate the problem [47], [48], but those were only

partially successful. Very recently however, Kowalczyk and Wee [49], [50] resolved the multi-use issue completely for ABE supporting access policies represented by NC^1 boolean circuits. While very interesting and significantly different from its predecessors [47], [48], the technique of [49], [50] seems to be inherently tied to the boolean structure of the underlying access policies and not suitable for adoption into the arithmetic setting as we are considering in this paper.

On the other hand, Okamoto and Takashima addressed the multi-use issue in the context of ABS in an updated version of [10], namely, [21] by introducing a new technique, which they termed as "one-dimensional localization of inner product values". The main idea of this technique is to embed a specific inner product value for an unbounded (with respect to the public parameters) number of times in a certain one-dimension of the hidden subspace of a signing key or verification-text, while erasing all informations of the inner product value from all the remaining dimensions of the hidden subspace. This technique is applied in two steps. First a "special linear transformation" step is applied over the hidden segments of a signing key and a verification-text. This step localizes the inner product values in certain one-dimension of the hidden subspace. But, some informations of the inner product values still remain in the other dimensions of the hidden subspace. To completely remove those informations, random values are "injected" into those dimensions of the hidden subspace. This second step is executed via a computational transition based on the underlying computational assumption, and thus is not problematic to directly extend to the unbounded setting. However, the first step, i.e., the special linear transformation step is information theoretic, and crucially relies on the secret randomness used to generate the public parameters. Since the public parameters only uses a constant amount of secret randomness in the unbounded setting, such an information-theoretic transition cannot be applied.

The most intuitive way-out to the above issue is to use the indexing and consistent randomness amplification techniques of [41] to supply the additional randomness required for the transition just as it is used to resolve similar issues in extending the dual system encryption proof technique to the unbounded setting. Unfortunately, the consistent randomness amplification technique of [41] is only capable of computationally simulating the application of a *random* linear transformation to the hidden segment of a key component and the corresponding segment of a verification-text component. Such a random linear transformation suffices for the application of the pairwise independence lemma to complete a security proof based on the dual system encryption paradigm. However, the one-dimensional localization technique requires the application of certain *specific* linear transformations over the hidden segments of a signing key and a verification-text that crucially depend on the associated signing attribute vector of the signing key being considered.

To resolve this issue, we devise a more sophisticated technique. Very roughly, we first computationally simulate the effect of random linear transformations over the hidden subspaces on the verification-text side. This step corresponds to the transition between the hybrid experiments Hyb_0 and Hyb_1 in the proof of unforgeability of our ABS construction (proof of Theorem 2). Next, we computationally amplify the randomness provided by the two-dimensional prefix vectors to the hidden subspaces on the signing key side. This is the transition from $\mathsf{Hyb}_{2\text{-}(\chi-1)\text{-}9}$ to $\mathsf{Hyb}_{2\text{-}\chi\text{-}1}$ in the unforgeability proof. After this step, we computationally alter the random linear transformations to specific ones on the verification-text side. This step is executed while moving from $\mathsf{Hyb}_{2\text{-}\chi\text{-}1}$ to $\mathsf{Hyb}_{2\text{-}\chi\text{-}2}$ in the proof of unforgeability. Finally, we computationally adjust the randomness expanded to the hidden segments on the signing key side to match the specific linear transformations to be applied on that side. This transformation is achieved via the transition between $\mathsf{Hyb}_{2\text{-}\chi\text{-}2}$ and $\mathsf{Hyb}_{2\text{-}\chi\text{-}3}$ in our unforgeability proof. We stress that the above explanation of our highly involved techniques is merely a bird's eye-view. For a comprehensive understanding of our techniques refer to our detail security proof presented in Sect. 5.

We would conclude this technical overview with the intuitive description of another small piece of idea due to Okamoto and Takashima [10], [11] that we leverage in the unforgeability proof of our proposed ABS scheme. Note that the overall approach of our unforgeability proof is to start with the real unforgeability experiment, as defined in Sect. 2.6, and through various hybrid steps ultimately reach an experiment in which the adversary has perfectly no chance of producing a forged signature. In order to accomplish that, we introduce a form of correlation between the verification text used to verify a signature and the message-policy pair with respect to which the verification is executed. Similarly, we introduce a correlation between a signature and the message-policy pair for which it is generated as well. We make use of a collision resistant hash function H_{hk} for this purpose. More precisely, we embed the two-dimensional vectors $(1, \mathsf{H}_{\mathsf{hk}}(\mathsf{MSG}\|\mathbb{S}))$ within signatures generated on messages MSG under signing policy ASPs $\mathbb{S} = (\mathbb{U}, \rho)$, while the two dimensional vectors $(-\mathsf{H}_{\mathsf{hk}}(\mathsf{MSG}\|\mathbb{S}), 1)$ within the verification texts used to verify those signatures. During the verification process, we require the verifier to compute the inner product of the two vectors, which clearly evaluates to zero if both the signature and the verification text correspond to the same message-ASP pair. Now, due to the restriction of the unforgeability experiment, the adversary can only output a forged signature on some message-ASP pair $(\mathsf{MSG}, \mathbb{S})$ which is distinct from all those message-ASP pairs $(\mathsf{MSG}_\pi, \mathbb{S}_\pi)$ for which it queries signatures to the challenger during the experiment. Thanks to this restriction and the collision resistance property of the hash function, H_{hk}, the two-dimensional vector $(-\mathsf{H}_{\mathsf{hk}}(\mathsf{MSG}\|\mathbb{S}), 1)$ embedded within the honestly generated verification text used to verify the forged signature would with high probability not be orthogonal to any of the vectors $(1, \mathsf{H}_{\mathsf{hk}}(\mathsf{MSG}_\pi\|\mathbb{S}_\pi))$ embedded within the signatures the adversary obtains from the challenger during the experiment. We leverage this fact in conjunction with the pairwise independence lemma to alter the form of the

verification text used to verify the forged signature to one that embeds a uniformly random two-dimensional vector in stead of $(-\mathsf{H}_{\mathsf{hk}}(\mathsf{MSG}\|\mathbb{S}), 1)$, and thereby make the verification text uncorrelated to the message-ASP pair $(\mathsf{MSG}, \mathbb{S})$ with respect to which the forgery is claimed. Once this modification is achieved, the inner product between the random two dimensional vector embedded within the modified verification text and the vector $(1, \mathsf{H}_{\mathsf{hk}}(\mathsf{MSG}\|\mathbb{S}))$ embedded within the forged signature outputted by the adversary would not be zero except for negligible probability, and therefore, the forged signature outputted by the adversary would not get verified except for negligible probability.

4. The Proposed ABS Scheme

In this section, we will present our ABS scheme for a predicate family $\mathcal{R}^{(q)}_{\mathsf{Z\text{-}ABP}}$ parameterized by some prime $q \in \mathbb{N}$ as defined in Sect. 2.6. Let $\mathbb{M} \subset \{0,1\}^*$ be the message space associated with our ABS scheme. We emphasize that in our construction the functions ρ included within the description of ASP's are not necessarily injective, and thus our ABS scheme supports *unbounded* multi-use of attributes within the signing policies. In our scheme description and in the proof of security $n = \mathsf{p}(\lambda) \in \mathbb{N}$ for an arbitrary polynomial p.

$\mathsf{ABS.Setup}()$: The setup algorithm takes as input the unary encoded security parameter 1^λ. It proceeds as follows:

1. It first generates $(\mathsf{params}, \{\mathbb{B}_\iota, \mathbb{B}^*_\iota\}_{\iota\in[0,2]}) \xleftarrow{\mathsf{R}} \mathcal{G}_{\mathsf{OB}}(2, (4, 14, 8))$.
2. Then, it sets the following:

$$\begin{aligned}
\widehat{\mathbb{B}}_0 &= \{\boldsymbol{b}^{(0,1)}, \boldsymbol{b}^{(0,4)}\},\\
\widehat{\mathbb{B}}^*_0 &= \{\boldsymbol{b}^{*(0,3)}\},\\
\widehat{\mathbb{B}}_1 &= \{\boldsymbol{b}^{(1,1)}, \ldots, \boldsymbol{b}^{(1,4)}, \boldsymbol{b}^{(1,13)}, \boldsymbol{b}^{(1,14)}\},\\
\widehat{\mathbb{B}}^*_1 &= \{\boldsymbol{b}^{*(1,1)}, \ldots, \boldsymbol{b}^{*(1,4)}, \boldsymbol{b}^{*(1,11)}, \boldsymbol{b}^{*(1,12)}\},\\
\widehat{\mathbb{B}}_2 &= \{\boldsymbol{b}^{(2,1)}, \boldsymbol{b}^{(2,2)}, \boldsymbol{b}^{(2,7)}, \boldsymbol{b}^{(2,8)}\},\\
\widehat{\mathbb{B}}^*_2 &= \{\boldsymbol{b}^{*(2,1)}, \boldsymbol{b}^{*(2,2)}, \boldsymbol{b}^{*(2,5)}, \boldsymbol{b}^{*(2,6)}\}.
\end{aligned}$$

3. Next, it samples a hashing key $\mathsf{hk} \xleftarrow{\mathsf{R}} \mathsf{KGen}()$ for a hash function family $\mathbb{H}$ associated with the bilinear group generator $\mathcal{G}_{\mathsf{BPG}}$ used as a subroutine of $\mathcal{G}_{\mathsf{OB}}$ and a polynomial $\mathsf{poly}(\cdot)$, where $\mathsf{poly}(\lambda)$ represents the length of the bit string formed by concatenating a message belonging to $\mathbb{M}$ and the binary representation of an ASP representing a signing policy predicate in $\mathcal{R}^{(q)}_{\mathsf{Z\text{-}ABP}}$.
4. It outputs the public parameters $\mathsf{MPK} = (\mathsf{hk}, \mathsf{params}, \{\widehat{\mathbb{B}}_\iota, \widehat{\mathbb{B}}^*_\iota\}_{\iota\in[0,2]})$ and the master signing key $\mathsf{MSK} = \boldsymbol{b}^{*(0,1)}$.

$\mathsf{ABS.KeyGen}(\mathsf{MPK}, \mathsf{MSK}, \vec{x})$: The signing key generation algorithm takes as input the public parameters MPK, the master signing key MSK, and a signing attribute vector $\vec{x} \in \mathbb{F}^n_q$. It executes the following steps:

1. First, it samples $\omega \xleftarrow{\mathsf{U}} \mathbb{F}_q\backslash\{0\}$, $\varphi_0 \xleftarrow{\mathsf{U}} \mathbb{F}_q$, and computes

$$\boldsymbol{k}^{*(0)} = (\omega, 0, \varphi_0, 0)_{\mathbb{B}^*_0}.$$

2. Next, for $\iota \in [n]$, it samples $\sigma_\iota \xleftarrow{\mathsf{U}} \mathbb{F}_q$, $\vec{\varphi}^{(\iota)} \xleftarrow{\mathsf{U}} \mathbb{F}^2_q$, and computes

$$\boldsymbol{k}^{*(\iota)} = (\sigma_\iota(1, \iota), \omega(1, x_\iota), \vec{0}^6, \vec{\varphi}^{(\iota)}, \vec{0}^2)_{\mathbb{B}^*_1}.$$

3. Then, it samples $\vec{\varphi}^{(n+1,1)}, \vec{\varphi}^{(n+1,2)} \xleftarrow{\mathsf{U}} \mathbb{F}^2_q$, and computes

$$\begin{aligned}
\boldsymbol{k}^{*(n+1,1)} &= (\omega(1, 0), \vec{0}^2, \vec{\varphi}^{(n+1,1)}, \vec{0}^2)_{\mathbb{B}^*_2},\\
\boldsymbol{k}^{*(n+1,2)} &= (\omega(0, 1), \vec{0}^2, \vec{\varphi}^{(n+1,2)}, \vec{0}^2)_{\mathbb{B}^*_2}.
\end{aligned}$$

4. It outputs the signing key $\mathsf{SK}(\vec{x}) = (\boldsymbol{k}^{*(0)}, \ldots, \boldsymbol{k}^{*(n)}, \boldsymbol{k}^{*(n+1,1)}, \boldsymbol{k}^{*(n+1,2)})$.

$\mathsf{ABS.Sign}(\mathsf{MPK}, \vec{x}, \mathsf{SK}(\vec{x}), \mathbb{S}, \mathsf{MSG})$: The signing algorithm takes in the public parameters MPK, a signing attribute string $\vec{x} \in \mathbb{F}^n_q$, a signing key $\mathsf{SK}(\vec{x}) = (\boldsymbol{k}^{*(0)}, \ldots, \boldsymbol{k}^{*(n)}, \boldsymbol{k}^{*(n+1,1)}, \boldsymbol{k}^{*(n+1,2)})$ for $\vec{x}$, a signing policy predicate $R^{(q)}_{\mathsf{Z\text{-}ABP}}(f, \cdot) : \mathbb{F}^n_q \to \{0,1\} \in \mathcal{R}^{(q)}_{\mathsf{Z\text{-}ABP}}$ with ASP representation $\mathbb{S} = (\mathbb{U} = \{(\vec{y}^{(j)}, \vec{z}^{(j)})\}_{j\in[m]} \subset (\mathbb{F}^\ell_q)^2, \rho : [m] \to [n])$, along with a message $\mathsf{MSG} \in \mathbb{M}$. If $\mathbb{S}$ does not accept $\vec{x}$, it outputs $\perp$. Otherwise, i.e., if $\mathbb{S}$ accepts $\vec{x}$, it operates as follows:

1. It first computes $(\Omega_j)_{j\in[m]} \in \mathbb{F}^m_q$ such that $\vec{e}^{(\ell,\ell)} = \sum_{j\in[m]} \Omega_j(x_{\rho(j)}\vec{y}^{(j)} + \vec{z}^{(j)})$.
2. After that, it samples $\xi \xleftarrow{\mathsf{U}} \mathbb{F}_q\backslash\{0\}$, and $((\Omega'_j)_{j\in[m]}, (\Omega''_j)_{j\in[m]}) \xleftarrow{\mathsf{U}} (\mathbb{F}^m_q)^2$ such that $\sum_{j\in[m]} (\Omega'_j\vec{y}^{(j)} + \Omega''_j\vec{z}^{(j)}) = \vec{0}^\ell$.
3. Next, it samples $\boldsymbol{r}^{*(0)} \xleftarrow{\mathsf{U}} \mathsf{SPAN}\langle\boldsymbol{b}^{*(0,3)}\rangle$ and computes

$$\boldsymbol{s}^{*(0)} = \xi\boldsymbol{k}^{*(0)} + \boldsymbol{r}^{*(0)}.$$

4. Then, for $j \in [m]$, it samples $\sigma'_j \xleftarrow{\mathsf{U}} \mathbb{F}_q$, $\boldsymbol{r}^{*(j)} \xleftarrow{\mathsf{U}} \mathsf{SPAN}\langle\boldsymbol{b}^{*(1,11)}, \boldsymbol{b}^{*(1,12)}\rangle$, and computes

$$\begin{aligned}
\boldsymbol{s}^{*(j)} = {}& \xi\Omega_j\boldsymbol{k}^{*(\rho(j))} + \sigma'_j(\boldsymbol{b}^{*(1,1)} + \rho(j)\boldsymbol{b}^{*(1,2)}) +\\
& \Omega''_j\boldsymbol{b}^{*(1,3)} + \Omega'_j\boldsymbol{b}^{*(1,4)} + \boldsymbol{r}^{*(j)}.
\end{aligned}$$

5. Next, it samples $\boldsymbol{r}^{*(m+1)} \xleftarrow{\mathsf{U}} \mathsf{SPAN}\langle\boldsymbol{b}^{*(2,5)}, \boldsymbol{b}^{*(2,6)}\rangle$ and computes

$$\boldsymbol{s}^{*(m+1)} = \xi(\boldsymbol{k}^{*(n+1,1)} + \mathsf{H}^{(\lambda,\mathsf{poly})}_{\mathsf{hk}}(\mathsf{MSG}\|\mathbb{S})\boldsymbol{k}^{*(n+1,2)}) +$$

$$\boldsymbol{r}^{*(m+1)}.$$

6. It outputs the signature $\mathsf{sig} = (\boldsymbol{s}^{*(0)}, \ldots, \boldsymbol{s}^{*(m+1)})$.

$\mathsf{ABS.Verify}(\mathsf{MPK}, \mathbb{S}, (\mathsf{MSG}, \mathsf{sig}))$: The verification algorithm takes as input the public parameters MPK, a signing policy predicate $R^{(q)}_{\mathsf{Z\text{-}ABP}}(f,\cdot) : \mathbb{F}_q^n \to \{0,1\} \in \mathcal{R}^{(q)}_{\mathsf{Z\text{-}ABP}}$ having ASP-representation $\mathbb{S} = (\mathbb{U} = \{(\vec{y}^{(j)}, \vec{z}^{(j)})\}_{j\in[m]} \subset (\mathbb{F}_q^\ell)^2, \rho : [m] \to [n])$, a message-signature pair $(\mathsf{MSG} \in \mathbb{M}, \mathsf{sig} = (\boldsymbol{s}^{*(0)}, \ldots, \boldsymbol{s}^{*(m+1)}))$. It proceeds as follows:

1. It generates a verification-text $(\boldsymbol{c}^{(0)}, \ldots, \boldsymbol{c}^{(m+1)})$ as follows:
   a. It first samples $\vec{u} = (u_1, \ldots, u_\ell) \xleftarrow{\cup} \mathbb{F}_q^\ell$, and computes $s_j = \vec{u} \cdot \vec{y}^{(j)}$, $s'_j = \vec{u} \cdot \vec{z}^{(j)}$ for $j \in [m]$.
   b. Next, it samples $u, \eta_0 \xleftarrow{\cup} \mathbb{F}_q$, and computes
   $$\boldsymbol{c}^{(0)} = (-u - u_\ell, 0, 0, \eta_0)_{\mathbb{B}_0}.$$
   c. Then, for $j \in [m]$, if $\boldsymbol{s}^{*(j)} \notin \mathbb{V}_1^*$, then it outputs 0. Otherwise, it samples $\mu_j \xleftarrow{\cup} \mathbb{F}_q$, $\vec{\eta}^{(j)} \xleftarrow{\cup} \mathbb{F}_q^2$, and computes
   $$\boldsymbol{c}^{(j)} = (\mu_j(\rho(j), -1), (s'_j, s_j), \vec{0}^6, \vec{0}^2, \vec{\eta}^{(j)})_{\mathbb{B}_1}.$$
   d. Then, it samples $\kappa \xleftarrow{\cup} \mathbb{F}_q$, $\vec{\eta}^{(m+1)} \xleftarrow{\cup} \mathbb{F}_q^2$, and computes
   $$\boldsymbol{c}^{(m+1)} = ((u - \kappa \mathsf{H}^{(\lambda,\mathsf{poly})}_{\mathsf{hk}}(\mathsf{MSG}\|\mathbb{S}), \kappa), \vec{0}^2, \vec{0}^2, \vec{\eta}^{(m+1)})_{\mathbb{B}_2}.$$
2. It outputs 0 if $e(\boldsymbol{b}^{(0,1)}, \boldsymbol{s}^{*(0)}) = 1_{\mathbb{G}_T}$.
3. It outputs 1 if $\prod_{j\in[0,m+1]} e(\boldsymbol{c}^{(j)}, \boldsymbol{s}^{*(j)}) = 1_{\mathbb{G}_T}$. It outputs 0 otherwise. Here, $1_{\mathbb{G}_T}$ is the identity element of the group $\mathbb{G}_T$.

▷ Correctness

The correctness of the proposed ABS construction can be verified as follows: For any signature $\mathsf{sig} = (\boldsymbol{s}^{*(0)}, \ldots, \boldsymbol{s}^{*(m+1)})$ on a message $\mathsf{MSG} \in \mathbb{M}$ under a signing policy predicate $R^{(q)}_{\mathsf{Z\text{-}ABP}}(f,\cdot) : \mathbb{F}_q^n \to \{0,1\} \in \mathcal{R}^{(q)}_{\mathsf{Z\text{-}ABP}}$ having ASP representation $\mathbb{S} = (\mathbb{U} = \{(\vec{y}^{(j)}, \vec{z}^{(j)})\}_{j\in[m]} \subset (\mathbb{F}_q^\ell)^2, \rho : [m] \to [n])$ generated using a signing key $\mathsf{SK}(\vec{x}) = (\boldsymbol{k}^{*(0)}, \ldots, \boldsymbol{k}^{*(n)}, \boldsymbol{k}^{*(n+1,1)}, \boldsymbol{k}^{*(n+1,2)})$ for a signing attribute vector $\vec{x} \in \mathbb{F}_q^n$ such that $\mathbb{S}$ accepts $\vec{x}$, and any verification-text $(\boldsymbol{c}^{(0)}, \ldots, \boldsymbol{c}^{(m+1)})$ generated while executing $\mathsf{ABS.Verify}$, we have

$$\begin{aligned}
&\prod_{j\in[0,m+1]} e(\boldsymbol{c}^{(j)}, \boldsymbol{s}^{*(j)}) \\
&= e(\boldsymbol{c}^{(0)}, \boldsymbol{k}^{*(0)})^\xi \prod_{j\in[m]} e(\boldsymbol{c}^{(j)}, \boldsymbol{k}^{*(\rho(j))})^{\xi\Omega_j} \cdot \\
&\quad \prod_{j\in[m]} [e(\boldsymbol{c}^{(j)}, \boldsymbol{b}^{*(1,3)})^{\Omega''_j} e(\boldsymbol{c}^{(j)}, \boldsymbol{b}^{*(1,4)})^{\Omega'_j}] \cdot \\
&\quad [e(\boldsymbol{c}^{(m+1)}, \boldsymbol{k}^{*(n+1,1)}) e(\boldsymbol{c}^{(m+1)}, \boldsymbol{k}^{*(n+1,2)})^{\mathsf{H}^{(\lambda,\mathsf{poly})}_{\mathsf{hk}}(\mathsf{MSG}\|\mathbb{S})}]^\xi \\
&= g_T^{\xi\omega(-u-u_\ell)} \prod_{j\in[m]} g_T^{\xi\omega\Omega_j(x_{\rho(j)}s_j + s'_j)} \prod_{j\in[m]} g_T^{(\Omega'_j s_j + \Omega''_j s'_j)} \cdot \\
&\quad g_T^{\xi\omega u} \\
&= g_T^{\xi\omega(-u-u_\ell)} g_T^{\xi\omega(\vec{u}\cdot\sum_{j\in[m]}\Omega_j(x_{\rho(j)}\vec{y}^{(j)} + \vec{z}^{(j)}))} \cdot \\
&\quad g_T^{\vec{u}\cdot\sum_{j\in[m]}(\Omega'_j\vec{y}^{(j)} + \Omega''_j\vec{z}^{(j)})} g_T^{\xi\omega u} \\
&= g_T^{\xi\omega(-u-u_\ell)} g_T^{\xi\omega(\vec{u}\cdot\vec{e}^{(\ell,\ell)})} g_T^{\vec{u}\cdot\vec{0}^\ell} g_T^{\xi\omega u} \\
&= g_T^{\xi\omega(-u-u_\ell)} g_T^{\xi\omega u_\ell} 1_{\mathbb{G}_T} g_T^{\xi\omega u} \\
&= 1_{\mathbb{G}_T}.
\end{aligned}$$

The above follows from the expressions of $(\boldsymbol{c}^{(0)}, \ldots, \boldsymbol{c}^{(m+1)})$, $(\boldsymbol{s}^{*(0)}, \ldots, \boldsymbol{s}^{*(m+1)})$, $(\boldsymbol{k}^{*(0)}, \ldots, \boldsymbol{k}^{*(n)}, \boldsymbol{k}^{*(n+1,1)}, \boldsymbol{k}^{*(n+1,2)})$, and the dual orthonormality property of $\{\mathbb{B}_\iota, \mathbb{B}^*_\iota\}_{\iota\in[0,2]}$; in conjunction with the facts that $\sum_{j\in[m]} \Omega_j(x_{\rho(j)}\vec{y}^{(j)} + \vec{z}^{(j)}) = \vec{e}^{(\ell,\ell)}$ (since $\mathbb{S}$ accepts $\vec{x}$), and $\sum_{j\in[m]}(\Omega'_j\vec{y}^{(j)} + \Omega''_j\vec{z}^{(j)}) = \vec{0}^\ell$ (by selection).

Remark 2 (Discussion on the Concrete Efficiency of the Proposed ABS Scheme): In order to understand the concrete efficiency gains of our ABS scheme over the state-of-the-art scheme of [15], let us consider the performance of both the schemes for a simple signing policy ABP $f : \mathbb{F}_q \to \mathbb{F}_q$ defined by $f(x_1) = x_1 - a_1$ for all $x_1 \in \mathbb{F}_q$, where q is a 128-bit prime integer and a_1 is a constant belonging to $\mathbb{F}_q$. We have already presented the summary of this efficiency analysis in Table 1 in the Introduction section. For the considered ABP, we have $R^{(q)}_{\mathsf{Z\text{-}ABP}}(f, x_1) = 1 \iff f(x_1) = 0 \iff x_1 = a_1$. By applying the algorithm of [36], we can represent the ABP f by the ASP $\mathbb{S} = (\mathbb{U} = \{(\vec{y}^{(1)} = (1,0), \vec{z}^{(1)} = (-a,-1))\}, \rho \mid 1 \mapsto 1)$. Hence, it can be readily verified from the description of the proposed ABS scheme above that in this scheme, a signature $\mathsf{SIG} = (\boldsymbol{s}^{*(0)}, \boldsymbol{s}^{*(1)}, \boldsymbol{s}^{*(2)})$ on some message $\mathsf{MSG} \in \mathbb{M}$ under $R^{(q)}_{\mathsf{Z\text{-}ABP}}(f,\cdot)$ would consist of only 26 group elements, namely, 4 group elements for $\boldsymbol{s}^{*(0)}$, 14 group elements for $\boldsymbol{s}^{*(1)}$, while 8 group elements for $\boldsymbol{s}^{*(2)}$. On the other hand, to generate the signature a signer would have to compute 138 exponentiations, namely 8 exponentiations for $\boldsymbol{s}^{*(0)}$, 98 for $\boldsymbol{s}^{*(1)}$, while 32 for $\boldsymbol{s}^{*(2)}$; and to verify the signature, a verifier would have to compute 30 pairing operations, namely, 4 pairing operations to verify whether $e(\boldsymbol{b}^{(0,1)}, \boldsymbol{s}^{*(0)}) = 1_{\mathbb{G}_T}$ and 26 pairing operations to verify whether $\prod_{j\in[0,2]} e(\boldsymbol{c}^{(j)}, \boldsymbol{s}^{*(j)}) = 1_{\mathbb{G}_T}$, where $(\boldsymbol{c}^{(0)}, \boldsymbol{c}^{(1)}, \boldsymbol{c}^{(2)})$ is the verification-text computed during the verification procedure.

Now, let us look into the size of a signature for the same signing policy as well as the computations required for its generation and verification in the ABS scheme of Sakai et al. [15]. Observe that in this scheme, signing policies are considered as boolean circuits. So, we must express $R^{(q)}_{\mathsf{Z\text{-}ABP}}(f,\cdot)$ as a boolean circuit. Clearly, the boolean circuit that simulates $R^{(q)}_{\mathsf{Z\text{-}ABP}}(f,\cdot)$ would have 128 input gates to take

as input the bit representation of x_1. Moreover, in order to simulate the equality test $x_1 = a_1$ over $\mathbb{F}_q$ using boolean operations, the circuit would need to implement 127 boolean AND gates, where the first boolean AND gate would connect the first and second bits of x_1, the second one would connect the earlier AND gate with the third bit of x_1, and so on. Also, for all $i \in [128]$, the wire connecting the i^{th} bit of x_1 to an AND gate must pass through a NOT gate if the i^{th} bit of a_1 is 0. For instance, if we represent the i^{th} bit of an element $b \in \mathbb{F}_q$ by $b[i]$ for all $i \in [128]$, and some $a_1 \in \mathbb{F}_q$ has binary representation $110\ldots01$, then the boolean circuit simulating $R^{(q)}_{\mathsf{Z\text{-}ABP}}(f, \cdot)$ with this a_1 would be

$$(((\ldots((x_1[1] \text{ AND } x_1[2]) \text{ AND } (\text{NOT } x_1[3]))\ldots) \text{ AND } (\text{NOT } x_1[127])) \text{ AND } x_1[128]).$$

Hence, it follows that the boolean circuit that realizes $R^{(q)}_{\mathsf{Z\text{-}ABP}}(f, \cdot)$ would have 128 input gates, 127 AND gates along with some additional NOT gates. Further, note that the ABS scheme of [15] considers representing signing policies using boolean circuits consisting of NAND gates only. Since 3 NAND gates are required to simulate each AND gate, and 1 NAND gate is needed to simulate each NOT gate, it follows that the boolean circuit simulating $R^{(q)}_{\mathsf{Z\text{-}ABP}}(f, \cdot)$ using only NAND gates would consist of at least 128 input gates and at least 127 NAND gates. Now, notice that Sakai et al. employs the celebrated Groth-Sahai proof system for pairing product equations [51] as an underlying tool in their ABS construction. A signature in their ABS scheme includes two Groth-Sahai commitments for each wire of the signing policy circuit with respect to which it is being generated and proofs showing the well-formedness of all those commitments. In addition, the signature includes proofs attesting to the correct evaluation of all the NAND gates of the signing policy circuit. On the other hand, verifying a signature involves checking all the proofs comprising it. Therefore, it is immediate from the performance figures presented in Tables 1 and 2 of [15] that a signature on some message with respect to the boolean circuit simulating $R^{(q)}_{\mathsf{Z\text{-}ABP}}(f, \cdot)$ in this scheme would include at least 4102 group elements, and verification of the signature would require at least 4102 pairing operations. Also, since generating a commitment in the Groth-Sahai proof system involves 4 exponentiations while creating a proof involves at least 10 exponentiations, it follows that generating the signature would require at least 5860 exponentiations.

Thus, it is clear that in terms of concrete efficiency, even for a very simple signing policy such as an equality test over $\mathbb{F}_q$, our ABS scheme gives more than 136 times better results from the view points of signature size and verification time while exhibits at least 42 times better performance on the signing time ground compared to the one of [15].

5. Security

Theorem 1 (Signer Privacy): *The proposed* ABS *scheme achieves perfect signer privacy (as per the security model described in Sect. 2.6).*

Proof: In order to prove Theorem 1, we introduce the following signing algorithm, we call ABS.AltSign, that generates signatures on messages using the master signing key MSK and do not use any attribute-specific signing key $\mathsf{SK}(\vec{x})$.

ABS.AltSign($\mathsf{MPK}, \mathsf{MSK}, \mathbb{S}, \mathsf{MSG}$): This algorithm takes in the public parameters MPK, the master signing key MSK, a signing policy predicate $R^{(q)}_{\mathsf{Z\text{-}ABP}}(f, \cdot) : \mathbb{F}_q^n \to \{0,1\} \in \mathcal{R}^{(q)}_{\mathsf{Z\text{-}ABP}}$ having ASP-representation $\mathbb{S} = (\mathbb{U} = \{(\vec{y}^{(j)}, \vec{z}^{(j)})\}_{j\in[m]} \subset (\mathbb{F}_q^\ell)^2, \rho : [m] \to [n])$, and a message $\mathsf{MSG} \in \mathbb{M}$. It proceeds as follows:

1. If
$$S = \{((\widehat{\Omega}_j)_{j\in[m]}, (\widehat{\Omega}'_j)_{j\in[m]}) \in (\mathbb{F}_q^m)^2 \mid \sum_{j\in[m]} (\widehat{\Omega}_j \vec{y}^{(j)} + \widehat{\Omega}'_j \vec{z}^{(j)}) = \vec{e}^{(\ell,\ell)}\} = \varnothing, \tag{1}$$
then it outputs $\perp$ indicating failure. Otherwise, it samples $((\widehat{\Omega}_j)_{j\in[m]}, (\widehat{\Omega}'_j)_{j\in[m]}) \xleftarrow{\cup} S$.
2. Next, it samples $\widehat{\omega} \xleftarrow{\cup} \mathbb{F}_q \backslash \{0\}$, $\widehat{v}_0 \xleftarrow{\cup} \mathbb{F}_q$, and computes
$$\boldsymbol{s}^{*(0)} = (\widehat{\omega}, 0, \widehat{v}_0, 0)_{\mathbb{B}_0^*}.$$
3. For $j \in [m]$, it samples $\widehat{\sigma}_j \xleftarrow{\cup} \mathbb{F}_q$, $\vec{\widehat{v}}_j \xleftarrow{\cup} \mathbb{F}_q^2$, and computes
$$\boldsymbol{s}^{*(j)} = (\widehat{\sigma}_j(1, \rho(j)), (\widehat{\Omega}'_j, \widehat{\Omega}_j), \vec{0}^6, \vec{\widehat{v}}^{(j)}, \vec{0}^2)_{\mathbb{B}_1^*}.$$
4. Then, it samples $\vec{\widehat{v}}^{(m+1)} \xleftarrow{\cup} \mathbb{F}_q^2$ and computes
$$\boldsymbol{s}^{*(m+1)} = (\widehat{\omega}(1, \mathsf{H}^{(\lambda,\text{poly})}_{\mathsf{hk}}(\mathsf{MSG}\|\mathbb{S})), \vec{0}^2, \vec{\widehat{v}}^{(m+1)}, \vec{0}^2)_{\mathbb{B}_2^*}.$$
5. It outputs the signature $\mathsf{sig} = (\boldsymbol{s}^{*(0)}, \ldots, \boldsymbol{s}^{*(m+1)})$.

Remark 3: Note that using the ABS.AltSign algorithm, one can generate a correctly verifiable signature on any message $\mathsf{MSG} \in \mathbb{M}$ under any signing policy predicate $R^{(q)}_{\mathsf{Z\text{-}ABP}}(f, \cdot) : \mathbb{F}_q^n \to \{0,1\} \in \mathcal{R}^{(q)}_{\mathsf{Z\text{-}ABP}}$ having ASP-representation $\mathbb{S} = (\mathbb{U} = \{(\vec{y}^{(j)}, \vec{z}^{(j)})\}_{j\in[m]} \subset (\mathbb{F}_q^\ell)^2, \rho : [m] \to [n])$ even without knowing any signing attribute string $\vec{x} \in \mathbb{F}_q^n$ accepted by $\mathbb{S}$. However, in order to execute this algorithm, one should have access to the master signing key MSK – something which a signer does not have access to in the real world (and an adversary in the unforgeability experiment). Hence, the above algorithm should only be viewed as a virtual one used in the security proof. Also, note that if the set S defined in the ABS.AltSign algorithm above is empty, then it is impossible that there exists some signing attribute string $\vec{x} \in \mathbb{F}_q^n$ accepted by $\mathbb{S}$, and hence no signature can ever be generated under $\mathbb{S}$, even in the real world.

Clearly, in order to prove Theorem 1 it is enough to show that the following statement is true:
For any security parameter $\lambda \in \mathbb{N}$, *any message* $\mathsf{MSG} \in \mathbb{M}$, *any signing attribute string* $\vec{x} \in \mathbb{F}_q^n$, *any signing policy predicate* $R^{(q)}_{\mathsf{Z\text{-}ABP}}(f,\cdot) : \mathbb{F}_q^n \to \{0,1\} \in \mathcal{R}^{(q)}_{\mathsf{Z\text{-}ABP}}$ *having* ASP*-representation* $\mathbb{S} = (\mathbb{U} = \{(\vec{y}^{(j)}, \vec{z}^{(j)})\}_{j\in[m]} \subset (\mathbb{F}_q^\ell)^2, \rho : [m] \to [n])$ *such that* $\mathbb{S}$ *accepts* $\vec{x}$, *any* $(\mathsf{MPK}, \mathsf{MSK}) \xleftarrow{R} \mathsf{ABS.Setup}(1^n)$, *and any* $\mathsf{SK}(\vec{x}) \xleftarrow{R} \mathsf{ABS.KeyGen}(\mathsf{MPK}, \mathsf{MSK}, \vec{x})$, *the distributions of the signatures outputted by* $\mathsf{ABS.Sign}(\mathsf{MPK}, \vec{x}, \mathsf{SK}(\vec{x}), \mathbb{S}, \mathsf{MSG})$ *and those outputted by* $\mathsf{ABS.AltSign}(\mathsf{MPK}, \mathsf{MSK}, \mathbb{S}, \mathsf{MSG})$ *are equivalent.*
In the proposed ABS scheme, $\mathsf{sig} = (\boldsymbol{s}^{*(0)}, \ldots, \boldsymbol{s}^{*(m+1)}) \xleftarrow{R} \mathsf{ABS.Sig}(\mathsf{MPK}, \vec{x}, \mathsf{SK}(\vec{x}), \mathbb{S}, \mathsf{MSG})$ is computed as

$$\begin{aligned}
\boldsymbol{s}^{*(0)} &= (p_0, 0, 0, \upsilon_0)_{\mathbb{B}_0^*},\\
\boldsymbol{s}^{*(j)} &= (\bar{\sigma}_j(1, \rho(j)), \vec{p}^{(j)}, \vec{0}^6, \vec{v}^{(j)}, \vec{0}^2)_{\mathbb{B}_1^*} \text{ for } j \in [m],\\
\boldsymbol{s}^{*(m+1)} &= (\vec{p}^{(m+1)}, \vec{0}^2, \vec{v}^{(m+1)}, \vec{0}^2)_{\mathbb{B}_2^*},
\end{aligned}$$

such that $p_0 = \xi\omega$, $\bar{\sigma}_j = \xi\sigma_{\rho(j)}\Omega_j + \sigma'_j$, $\vec{p}^{(j)} = (\xi\omega\Omega_j + \Omega''_j, \xi\omega x_{\rho(j)}\Omega_j + \Omega'_j)$ for $j \in [m]$, and $\vec{p}^{(m+1)} = \xi\omega(1, \mathsf{H}^{(\lambda,\mathsf{poly})}_{\mathsf{hk}}(\mathsf{MSG}\|\mathbb{S}))$, where $\omega, \xi \xleftarrow{U} \mathbb{F}_q\backslash\{0\}$, $\{\sigma_\iota\}_{\iota\in[n]}, \{\sigma'_j\}_{j\in[m]}, \upsilon_0 \xleftarrow{U} \mathbb{F}_q$, $\{\vec{v}^{(j)}\}_{j\in[m+1]} \xleftarrow{U} \mathbb{F}_q^2$, $(\Omega_j)_{j\in[m]} \in \mathbb{F}_q^\ell$ with $\sum_{j\in[m]} \Omega_j(x_{\rho(j)}\vec{y}^{(j)} + \vec{z}^{(j)}) = \vec{e}^{(\ell,\ell)}$, and $((\Omega'_j)_{j\in[m]}, (\Omega''_j)_{j\in[m]}) \xleftarrow{U} (\mathbb{F}_q^m)^2$ with $\sum_{j\in[m]} (\Omega'_j\vec{y}^{(j)} + \Omega''_j\vec{z}^{(j)}) = \vec{0}^\ell$.

On the other hand $\mathsf{sig} = (\boldsymbol{s}^{*(0)}, \ldots, \boldsymbol{s}^{*(m+1)}) \xleftarrow{R} \mathsf{ABS.AltSign}(\mathsf{MPK}, \mathsf{MSK}, \mathbb{S}, \mathsf{MSG})$ is computed as

$$\begin{aligned}
\boldsymbol{s}^{*(0)} &= (\widehat{p}_0, 0, \widehat{\upsilon}_0, 0)_{\mathbb{B}_0^*},\\
\boldsymbol{s}^{*(j)} &= (\widehat{\sigma}_j(1, \rho(j)), \vec{\widehat{p}}^{(j)}, \vec{0}^6, \vec{\widehat{v}}^{(j)}, \vec{0}^2)_{\mathbb{B}_1^*} \text{ for } j \in [m],\\
\boldsymbol{s}^{*(m+1)} &= (\vec{\widehat{p}}^{(m+1)}, \vec{0}^2, \vec{\widehat{v}}^{(m+1)}, \vec{0}^2)_{\mathbb{B}_2^*},
\end{aligned}$$

such that $\widehat{p}_0 = \widehat{\omega}$, $\vec{\widehat{p}}^{(j)} = (\widehat{\Omega}'_j, \widehat{\Omega}_j)$ for $j \in [m]$, and $\vec{\widehat{p}}^{(m+1)} = \widehat{\omega}(1, \mathsf{H}^{(\lambda,\mathsf{poly})}_{\mathsf{hk}}(\mathsf{MSG}\|\mathbb{S}))$, where $\widehat{\omega} \xleftarrow{U} \mathbb{F}_q\backslash\{0\}$, $\{\widehat{\sigma}_j\}_{j\in[m]}, \widehat{\upsilon}_0 \xleftarrow{U} \mathbb{F}_q$, $\{\vec{\widehat{v}}^{(j)}\}_{j\in[m+1]} \xleftarrow{U} \mathbb{F}_q^2$, and $((\widehat{\Omega}_j)_{j\in[m]}, (\widehat{\Omega}'_j)_{j\in[m]}) \xleftarrow{U} S = \{((\widehat{\Omega}_j)_{j\in[m]}, (\widehat{\Omega}'_j)_{j\in[m]}) \in (\mathbb{F}_q^m)^2 \mid \sum_{j\in[m]} (\widehat{\Omega}_j\vec{y}^{(j)} + \widehat{\Omega}'_j\vec{z}^{(j)}) = \vec{e}^{(\ell,\ell)}\}$.

Observe that the distributions

$$\left\{ \begin{array}{l} (\xi\omega, (\xi\omega x_{\rho(j)}\Omega_j + \Omega'_j)_{j\in[m]}, (\xi\omega\Omega_j + \Omega''_j)_{j\in[m]}) \mid \\ \omega, \xi \xleftarrow{U} \mathbb{F}_q\backslash\{0\}, ((\Omega'_j)_{j\in[m]}, (\Omega''_j)_{j\in[m]}) \xleftarrow{U} (\mathbb{F}_q^m)^2 \\ \text{with } \sum_{j\in[m]} (\Omega'_j\vec{y}^{(j)} + \Omega''_j\vec{z}^{(j)}) = \vec{0}^\ell, (\Omega_j)_{j\in[m]} \in \mathbb{F}_q^m \\ \text{with } \sum_{j\in[m]} \Omega_j(x_{\rho(j)}\vec{y}^{(j)} + \vec{z}^{(j)}) = \vec{e}^{(\ell,\ell)} \end{array} \right\} \tag{2}$$

and

$$\left\{ \begin{array}{l} (\widehat{\omega}, (\widehat{\Omega}_j)_{j\in[m]}, (\widehat{\Omega}'_j)_{j\in[m]}) \mid \widehat{\omega} \xleftarrow{U} \mathbb{F}_q\backslash\{0\}, \\ ((\widehat{\Omega}_j)_{j\in[m]}, (\widehat{\Omega}'_j)_{j\in[m]}) \xleftarrow{U} S \end{array} \right\} \tag{3}$$

are equivalent.

Remark 4: Note that the set S defined in Eq. (1) above is an affine subspace of $\mathbb{F}_q^{2m}$. Hence, it follows that other than the first component (which is uniformly distributed over $\mathbb{F}_q\backslash\{0\}$), the distribution specified in Eq. (3) above is essentially the uniform distribution over an affine subspace of $\mathbb{F}_q^{2m}$. On the other hand, leaving aside the first component (which is uniformly distributed over $\mathbb{F}_q\backslash\{0\}$ as well), the distribution specified in Eq. (2) is an affine shift of a uniform distribution over a linear subspace of $\mathbb{F}_q^{2m}$.

Also, the distributions

$$\left\{ \begin{array}{l} (\bar{\sigma}_j = \xi\Omega_j\sigma_{\rho(j)} + \sigma'_j)_{j\in[m]} \mid \xi \xleftarrow{U} \mathbb{F}_q\backslash\{0\}, \\ \{\sigma_\iota\}_{\iota\in[n]}, \{\sigma'_j\}_{j\in[m]} \xleftarrow{U} \mathbb{F}_q, (\Omega_j)_{j\in[m]} \in \mathbb{F}_q^m \\ \text{with } \sum_{j\in[m]} \Omega_j(x_{\rho(j)}\vec{y}^{(j)} + \vec{z}^{(j)}) = \vec{e}^{(\ell,\ell)} \end{array} \right\}$$

and

$$\{(\widehat{\sigma}_j)_{j\in[m]} \mid \{\widehat{\sigma}_j\}_{j\in[m]} \xleftarrow{U} \mathbb{F}_q\}$$

are equivalent. Thus, the distributions of $\mathsf{sig} \xleftarrow{R} \mathsf{ABS.Sign}(\mathsf{MPK}, \vec{x}, \mathsf{SK}(\vec{x}), \mathbb{S}, \mathsf{MSG})$ and that of $\mathsf{sig} \xleftarrow{R} \mathsf{ABS.AltSign}(\mathsf{MPK}, \mathsf{MSK}, \mathbb{S}, \mathsf{MSG})$ are equivalent. This completes the proof of Theorem 1. ∎

Theorem 2 (Existential Unforgeability): *The proposed* ABS *scheme is existentially unforgeable against adaptive-predicate-adaptive-message attack (as per the security model described in Sect. 2.6) under the* SXDLIN *assumption.*

Proof: In order to prove Theorem 2, we consider a sequence of hybrid experiments which differ from one another in the construction of the signing keys/signatures queried by the adversary $\mathcal{A}$ and/or the verification-text used by the challenger $\mathcal{B}$ to verify the validity of the forged signature outputted by $\mathcal{A}$ at the end of the experiment. The first hybrid corresponds to the real unforgeability experiment described in Sect. 2.6, while the last hybrid corresponds to one in which the probability that a forged signature outputted by $\mathcal{A}$ passes the verification is negligible. We argue that $\mathcal{A}$'s winning probability changes only by a negligible amount in each successive hybrid experiment, thereby establishing Theorem 2. The overall structure of our reduction is demonstrated in Fig. 2. Let q_{key} and q_{sig} be the total number of signing keys and signatures $\mathcal{A}$ requests $\mathcal{B}$ to reveal during the experiment. The sequence of hybrid experiments are described below. In the description of the hybrids a part framed by a box indicates coefficients which are altered in a transition from its previous hybrid.

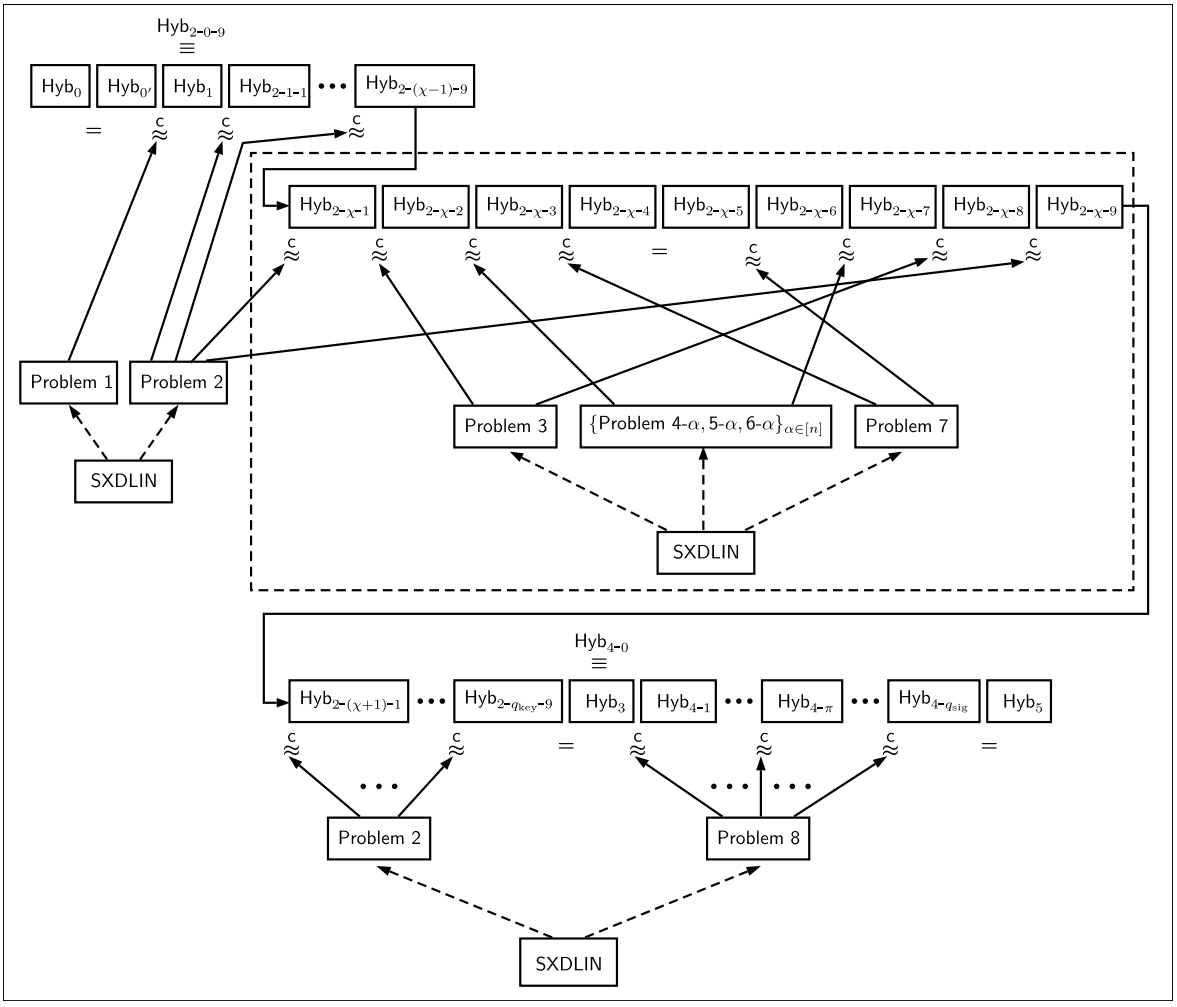


Fig. 2 Structure of the hybrid reduction for the proof of Theorem 2.

▷ Sequence of Hybrid Experiments

$\mathsf{Hyb}_{\mathbf{0}}$

This is the real unforgeability experiment described in Sect. 2.6.

$\mathsf{Hyb}_{\mathbf{0'}}$

This experiment is the same as Hyb_0 except the following:

1. When $\mathcal{A}$ makes a signing key generation query for some signing attribute string $\vec{x} \in \mathbb{F}_q^n$, $\mathcal{B}$ only records $\vec{x}$, but creates no actual signing key.
2. When a signature query is made by $\mathcal{A}$ on some message $\mathsf{MSG} \in \mathbb{M}$ under some signing policy predicate $R_{\mathsf{Z\text{-}ABP}}^{(q)}(f, \cdot) : \mathbb{F}_q^n \to \{0,1\} \in \mathcal{R}_{\mathsf{Z\text{-}ABP}}^{(q)}$ having ASP-representation $\mathbb{S} = (\mathbb{U}, \rho)$ to be created using a signing key for some signing attribute string $\vec{x} \in \mathbb{F}_q^n$ for which it has already made a signing key generation query, $\mathcal{B}$ simply records the triple $(\mathsf{MSG}, \mathbb{S}, \vec{x})$, but creates no actual signature.
3. When $\mathcal{A}$ issues a signing key reveal query for some signing attribute string $\vec{x} \in \mathbb{F}_q^n$ which has been already recorded, $\mathcal{B}$ creates the queried signing key as $\mathsf{SK}(\vec{x}) \xleftarrow{\mathsf{R}} \mathsf{ABS.KeyGen}(\mathsf{MPK}, \mathsf{MSK}, \vec{x})$, and returns it to $\mathcal{A}$. On the other hand, when $\mathcal{A}$ issues a signature reveal query for some triple $(\mathsf{MSG}, \mathbb{S}, \vec{x}) \in \mathbb{M} \times \mathcal{R}_{\mathsf{Z\text{-}ABP}}^{(q)} \times \mathbb{F}_q^n$ which has been already recorded, $\mathcal{B}$ creates the queried signature as $\mathsf{sig} \xleftarrow{\mathsf{R}} \mathsf{ABS.AltSign}(\mathsf{MPK}, \mathsf{MSK}, \mathbb{S}, \mathsf{MSG})$, where the ABS.AltSign algorithm is described in the proof of Theorem 1, and hands sig to $\mathcal{A}$.

Thus, in this experiment for $h \in [q_{\mathrm{key}}]$, the h^{th} signing key for signing attribute string $\vec{x}^{(h)} \in \mathbb{F}_q^n$ requested by $\mathcal{A}$ to reveal is generated as $\mathsf{SK}(\vec{x}^{(h)}) = (\boldsymbol{k}^{*(h,0)}, \ldots, \boldsymbol{k}^{*(h,n)}, \boldsymbol{k}^{*(h,n+1,1)}, \boldsymbol{k}^{*(h,n+1,2)})$ such that

$$
\begin{aligned}
\boldsymbol{k}^{*(h,0)} &= (\omega_h, 0, \varphi_{h,0}, 0)_{\mathbb{B}_0^*},\\
\boldsymbol{k}^{*(h,\iota)} &= (\sigma_{h,\iota}(1,\iota), \omega_h(1, x_\iota^{(h)}), \vec{0}^6, \vec{\varphi}^{(h,\iota)}, \vec{0}^2)_{\mathbb{B}_1^*}\\
&\qquad \text{for } \iota \in [n],\\
\boldsymbol{k}^{*(h,n+1,1)} &= (\omega_h(1,0), \vec{0}^2, \vec{\varphi}^{(h,n+1,1)}, \vec{0}^2)_{\mathbb{B}_2^*},\\
\boldsymbol{k}^{*(h,n+1,2)} &= (\omega_h(0,1), \vec{0}^2, \vec{\varphi}^{(h,n+1,2)}, \vec{0}^2)_{\mathbb{B}_2^*},
\end{aligned}
\tag{4}
$$

where $\omega_h \xleftarrow{\cup} \mathbb{F}_q\backslash\{0\}$, $\{\sigma_{h,\iota}\}_{\iota\in[n]}, \varphi_{h,0} \xleftarrow{\cup} \mathbb{F}_q$, $\{\vec{\varphi}^{(h,\iota)}\}_{\iota\in[n]}, \vec{\varphi}^{(h,n+1,1)}, \vec{\varphi}^{(h,n+1,2)} \xleftarrow{\cup} \mathbb{F}_q^2$.

On the other hand, for $t \in [q_{\mathrm{sig}}]$, the t^{th} signature associated with the triple $(\mathsf{MSG}_t, \mathbb{S}_t, \vec{x}^{(t)}) \in \mathbb{M} \times \mathcal{R}_{\mathsf{Z\text{-}ABP}}^{(q)} \times \mathbb{F}_q^n$ that $\mathcal{A}$ requests to reveal, where $\mathbb{S}_t = (\mathbb{U}_t = \{(\vec{y}^{(t,j)}, \vec{z}^{(t,j)})\}_{j\in[m_t]} \subset (\mathbb{F}_q^{\ell_t})^2, \rho_t : [m_t] \to [n])$, is created as $\mathrm{sig}_t = (\boldsymbol{s}^{*(t,0)}, \ldots, \boldsymbol{s}^{*(t,m_t+1)})$ such that

$$
\begin{aligned}
\boldsymbol{s}^{*(t,0)} &= (\widehat{\omega}_t, 0, \widehat{\upsilon}_{t,0}, 0)_{\mathbb{B}_0^*},\\
\boldsymbol{s}^{*(t,j)} &= (\widehat{\sigma}_{t,j}(1,\rho_t(j)), (\widehat{\Omega}'_{t,j}, \widehat{\Omega}_{t,j}), \vec{0}^6, \vec{\upsilon}^{(t,j)}, \vec{0}^2)_{\mathbb{B}_1^*}\\
&\qquad \text{for } j \in [m_t],\\
\boldsymbol{s}^{*(t,m_t+1)} &= (\widehat{\omega}(1, \mathsf{H}_{\mathsf{hk}}^{(\lambda,\mathsf{poly})}(\mathsf{MSG}_t \| \mathbb{S}_t)), \vec{0}^2, \vec{\upsilon}^{(t,m_t+1)}, \vec{0}^2)_{\mathbb{B}_2^*},
\end{aligned}
\tag{5}
$$

where $\widehat{\omega}_t \xleftarrow{\cup} \mathbb{F}_q\backslash\{0\}$, $\{\widehat{\sigma}_{t,j}\}_{j\in[m_t]}, \widehat{\upsilon}_{t,0} \xleftarrow{\cup} \mathbb{F}_q$, $\{\vec{\upsilon}^{(t,j)}\}_{j\in[m_t+1]} \xleftarrow{\cup} \mathbb{F}_q^2$, and $((\widehat{\Omega}_{t,j})_{j\in[m_t]}, (\widehat{\Omega}'_{t,j})_{j\in[m_t]}) \xleftarrow{\cup} S_t = \{((\widehat{\Omega}_{t,j})_{j\in[m_t]}, (\widehat{\Omega}'_{t,j})_{j\in[m_t]}) \in (\mathbb{F}_q^{m_t})^2 \mid \sum_{j\in[m_t]} (\widehat{\Omega}_{t,j}\vec{y}^{(t,j)} + \widehat{\Omega}'_{t,j}\vec{z}^{(t,j)}) = \vec{e}^{(\ell_t,\ell_t)}\}$.

Finally, in this experiment, the verification-text used to verify the forged signature outputted by $\mathcal{A}$ on some message $\mathsf{MSG} \in \mathbb{M}$ under some signing policy predicate $R_{\mathsf{Z\text{-}ABP}}^{(q)}(f,\cdot) : \mathbb{F}_q^n \to \{0,1\} \in \mathcal{R}_{\mathsf{Z\text{-}ABP}}^{(q)}$ having ASP-representation $\mathbb{S} = (\mathbb{U} = \{(\vec{y}^{(j)}, \vec{z}^{(j)})\}_{j\in[m]} \subset (\mathbb{F}_q^{\ell})^2, \rho : [m] \to [n])$ is generated as $(\boldsymbol{c}^{(0)}, \ldots, \boldsymbol{c}^{(m+1)})$ such that

$$
\begin{aligned}
\boldsymbol{c}^{(0)} &= (-u - u_\ell, 0, 0, \eta_0)_{\mathbb{B}_0},\\
\boldsymbol{c}^{(j)} &= (\mu_j(\rho(j), -1), (s'_j, s_j), \vec{0}^6, \vec{0}^2, \vec{\eta}^{(j)})_{\mathbb{B}_1} \text{ for } j \in [m],\\
\boldsymbol{c}^{(m+1)} &= ((u - \kappa \mathsf{H}_{\mathsf{hk}}^{(\lambda,\mathsf{poly})}(\mathsf{MSG} \| \mathbb{S}), \kappa), \vec{0}^2, \vec{0}^2, \vec{\eta}^{(m+1)})_{\mathbb{B}_2},
\end{aligned}
\tag{6}
$$

where $\vec{u} = (u_1, \ldots, u_\ell) \xleftarrow{\cup} \mathbb{F}_q^\ell$, $s_j = \vec{u} \cdot \vec{y}^{(j)}$, $s'_j = \vec{u} \cdot \vec{z}^{(j)}$ for $j \in [m]$, $u, \{\mu_j\}_{j\in[m]}, \kappa, \eta_0 \xleftarrow{\cup} \mathbb{F}_q$, and $\{\vec{\eta}^{(j)}\}_{j\in[m+1]} \xleftarrow{\cup} \mathbb{F}_q^2$.

Here $\{\mathbb{B}_\iota, \mathbb{B}_\iota^*\}_{\iota\in[0,2]}$ is the collection of dual orthonormal bases generated by $\mathcal{B}$ during the setup phase of the experiment.

Hyb_1

This experiment is analogous to $\mathsf{Hyb}_{0'}$ except that in this experiment, the verification-text used to verify the forged signature outputted by $\mathcal{A}$ on some message $\mathsf{MSG} \in \mathbb{M}$ under some signing policy predicate $R_{\mathsf{Z\text{-}ABP}}^{(q)}(f,\cdot) : \mathbb{F}_q^n \to \{0,1\} \in \mathcal{R}_{\mathsf{Z\text{-}ABP}}^{(q)}$ having ASP-representation $\mathbb{S} = (\mathbb{U} = \{(\vec{y}^{(j)}, \vec{z}^{(j)})\}_{j\in[m]} \subset (\mathbb{F}_q^{\ell})^2, \rho : [m] \to [n])$ is generated as $(\boldsymbol{c}^{(0)}, \ldots, \boldsymbol{c}^{(m+1)})$ such that

$$
\begin{aligned}
\boldsymbol{c}^{(0)} &= (-u - u_\ell, \boxed{-\widetilde{u}_\ell}, 0, \eta_0)_{\mathbb{B}_0},\\
\boldsymbol{c}^{(j)} &= (\mu_j(\rho(j), -1), (s'_j, s_j), \boxed{(\widetilde{s}'_j, \widetilde{s}_j)}, \vec{0}^2, \boxed{\vec{r}^{(j)}},\\
&\qquad \vec{0}^2, \vec{\eta}^{(j)})_{\mathbb{B}_1} \text{ for } j \in [m],\\
\boldsymbol{c}^{(m+1)} &= ((u - \kappa \mathsf{H}_{\mathsf{hk}}^{(\lambda,\mathsf{poly})}(\mathsf{MSG} \| \mathbb{S}), \kappa), \boxed{\vec{r}^{(m+1)}},\\
&\qquad \vec{0}^2, \vec{\eta}^{(m+1)})_{\mathbb{B}_2},
\end{aligned}
\tag{7}
$$

where $\vec{\widetilde{u}} = (\widetilde{u}_1, \ldots, \widetilde{u}_\ell) \xleftarrow{\cup} \mathbb{F}_q^\ell$, $\widetilde{s}_j = \vec{\widetilde{u}} \cdot \vec{y}^{(j)}$, $\widetilde{s}'_j = \vec{\widetilde{u}} \cdot \vec{z}^{(j)}$ for $j \in [m]$, $\{\vec{r}^{(j)}\}_{j\in[m+1]} \xleftarrow{\cup} \mathbb{F}_q^2$, and all the other variables are generated as in $\mathsf{Hyb}_{0'}$.

$\mathsf{Hyb}_{2\text{-}\chi\text{-}1}$ ($\chi \in [q_{\mathrm{key}}]$)

$\mathsf{Hyb}_{2\text{-}0\text{-}9}$ coincides with Hyb_1. This experiment is the same as $\mathsf{Hyb}_{2\text{-}(\chi-1)\text{-}9}$ with the only exception that in this experiment, the χ^{th} signing key for signing attribute string $\vec{x}^{(\chi)} \in \mathbb{F}_q^n$ requested by $\mathcal{A}$ to reveal is generated as $\mathsf{SK}(\vec{x}^{(\chi)}) = (\boldsymbol{k}^{*(\chi,0)}, \ldots, \boldsymbol{k}^{*(\chi,n)}, \boldsymbol{k}^{*(\chi,n+1,1)}, \boldsymbol{k}^{*(\chi,n+1,2)})$ such that $\boldsymbol{k}^{*(\chi,n+1,1)}, \boldsymbol{k}^{*(\chi,n+1,2)}$ are given by Eq. (4), and

$$
\begin{aligned}
\boldsymbol{k}^{*(\chi,0)} &= (\omega_\chi, \boxed{\widetilde{\omega}_\chi}, \varphi_{\chi,0}, 0)_{\mathbb{B}_0^*},\\
\boldsymbol{k}^{*(\chi,\iota)} &= (\sigma_{\chi,\iota}(1,\iota), \omega_\chi(1, x_\iota^{(\chi)}), \boxed{\widetilde{\omega}_\chi(1, x_\iota^{(\chi)})},\\
&\qquad \vec{0}^4, \vec{\varphi}^{(\chi,\iota)}, \vec{0}^2)_{\mathbb{B}_1^*} \text{ for } \iota \in [n],
\end{aligned}
\tag{8}
$$

where $\widetilde{\omega}_\chi \xleftarrow{\cup} \mathbb{F}_q\backslash\{0\}$ and all the other variables are generated as in $\mathsf{Hyb}_{2\text{-}(\chi-1)\text{-}9}$.

$\mathsf{Hyb}_{2\text{-}\chi\text{-}2}$ ($\chi \in [q_{\mathrm{key}}]$)

This experiment is analogous to $\mathsf{Hyb}_{2\text{-}\chi\text{-}1}$ except that in this experiment, the verification-text used to verify the forged signature outputted by $\mathcal{A}$ on some message $\mathsf{MSG} \in \mathbb{M}$ under some signing policy predicate $R_{\mathsf{Z\text{-}ABP}}^{(q)}(f,\cdot) : \mathbb{F}_q^n \to \{0,1\} \in \mathcal{R}_{\mathsf{Z\text{-}ABP}}^{(q)}$ having ASP-representation $\mathbb{S} = (\mathbb{U} = \{(\vec{y}^{(j)}, \vec{z}^{(j)})\}_{j\in[m]} \subset (\mathbb{F}_q^{\ell})^2, \rho : [m] \to [n])$ is generated as $(\boldsymbol{c}^{(0)}, \ldots, \boldsymbol{c}^{(m+1)})$ such that $\boldsymbol{c}^{(0)}, \boldsymbol{c}^{(m+1)}$ have the same form as in Eq. (7) and

$$
\begin{aligned}
\boldsymbol{c}^{(j)} &= (\mu_j(\rho(j), -1), (s'_j, s_j), (\widetilde{s}'_j, \widetilde{s}_j), \vec{0}^2,\\
&\qquad \boxed{(\widetilde{s}'_j, \widetilde{s}_j)\boldsymbol{Z}^{(\rho(j))}}, \vec{0}^2, \vec{\eta}^{(j)})_{\mathbb{B}_1} \text{ for } j \in [m],
\end{aligned}
\tag{9}
$$

where $\boldsymbol{Z}^{(\iota)} \in \{\boldsymbol{Z} \in \mathsf{GL}(2, \mathbb{F}_q) \mid \vec{e}^{(2,2)} = (1, x_\iota^{(\chi)})(\boldsymbol{Z}^{-1})^\top\}$ for $\iota \in [n]$, and all the other variables are generated as in $\mathsf{Hyb}_{2\text{-}\chi\text{-}1}$.

$\mathsf{Hyb}_{2\text{-}\chi\text{-}3}$ ($\chi \in [q_{\mathrm{key}}]$)

This experiment is the same as $\mathsf{Hyb}_{2\text{-}\chi\text{-}2}$ with the only exception that in this experiment, the χ^{th} signing key for signing attribute string $\vec{x}^{(\chi)} \in \mathbb{F}_q^n$ requested by $\mathcal{A}$ to reveal is generated

as $\mathsf{SK}(\vec{x}^{(\chi)}) = (\boldsymbol{k}^{*(\chi,0)}, \ldots, \boldsymbol{k}^{*(\chi,n)}, \boldsymbol{k}^{*(\chi,n+1,1)}, \boldsymbol{k}^{*(\chi,n+1,2)})$ such that $\boldsymbol{k}^{*(\chi,0)}$ is given by Eq. (8), $\boldsymbol{k}^{*(\chi,n+1,1)}, \boldsymbol{k}^{*(\chi,n+1,2)}$ are given by Eq. (4), and

$$\boldsymbol{k}^{*(\chi,\iota)} = (\sigma_{\chi,\iota}(1,\iota), \omega_\chi(1, x_\iota^{(\chi)}), \boxed{\vec{0}^2}, \vec{0}^2, \boxed{(0, \widetilde{\omega}_\chi)}, \vec{\varphi}^{(\chi,\iota)}, \vec{0}^2)_{\mathbb{B}_1^*} \text{ for } \iota \in [n], \tag{10}$$

where all the variables are generated as in $\mathsf{Hyb}_{2\text{-}\chi\text{-}2}$.

$\mathsf{Hyb}_{2\text{-}\chi\text{-}4}$ ($\chi \in [q_{\mathrm{key}}]$)

This experiment is identical to $\mathsf{Hyb}_{2\text{-}\chi\text{-}3}$ except that in this experiment, the verification-text used to verify the forged signature outputted by $\mathcal{A}$ on some message $\mathsf{MSG} \in \mathbb{M}$ under some signing policy predicate $R^{(q)}_{\text{Z-ABP}}(f,\cdot) : \mathbb{F}_q^n \to \{0,1\} \in \mathcal{R}^{(q)}_{\text{Z-ABP}}$ having ASP-representation $\mathbb{S} = (\mathbb{U} = \{(\vec{y}^{(j)}, \vec{z}^{(j)})\}_{j\in[m]} \subset (\mathbb{F}_q^\ell)^2, \rho : [m] \to [n])$ is generated as $(\boldsymbol{c}^{(0)}, \ldots, \boldsymbol{c}^{(m+1)})$ such that $\boldsymbol{c}^{(0)}, \boldsymbol{c}^{(m+1)}$ have the same form as in Eq. (7) and

$$\boldsymbol{c}^{(j)} = (\mu_j(\rho(j), -1), (s'_j, s_j), \boxed{\vec{a}^{(j)}}, \vec{0}^2, (\boxed{\widetilde{a}_j}, \vec{\widetilde{u}} \cdot (x^{(\chi)}_{\rho(j)} \vec{y}^{(j)} + \vec{z}^{(j)})), \vec{0}^2, \vec{\eta}^{(j)})_{\mathbb{B}_1} \text{ for } j \in [m], \tag{11}$$

where $\{\widetilde{a}_j\}_{j\in[m]} \xleftarrow{\cup} \mathbb{F}_q$, $\{\vec{a}^{(j)}\}_{j\in[m]} \xleftarrow{\cup} \mathbb{F}_q^2$, and all the other variables are generated as in $\mathsf{Hyb}_{2\text{-}\chi\text{-}3}$.

$\mathsf{Hyb}_{2\text{-}\chi\text{-}5}$ ($\chi \in [q_{\mathrm{key}}]$)

This experiment is the same as $\mathsf{Hyb}_{2\text{-}\chi\text{-}4}$ with the only exception that in this experiment, the χ^{th} signing key for signing attribute string $\vec{x}^{(\chi)} \in \mathbb{F}_q^n$ requested by $\mathcal{A}$ to reveal is generated as $\mathsf{SK}(\vec{x}^{(\chi)}) = (\boldsymbol{k}^{*(\chi,0)}, \ldots, \boldsymbol{k}^{*(\chi,n)}, \boldsymbol{k}^{*(\chi,n+1,1)}, \boldsymbol{k}^{*(\chi,n+1,2)})$ such that $\{\boldsymbol{k}^{*(\chi,\iota)}\}_{\iota\in[n]}$ are given by Eq. (10), $\boldsymbol{k}^{*(\chi,n+1,1)}$, $\boldsymbol{k}^{*(\chi,n+1,2)}$ are given by Eq. (4), and

$$\boldsymbol{k}^{*(\chi,0)} = (\omega_\chi, \boxed{\mathfrak{I}_\chi}, \varphi_{\chi,0}, 0)_{\mathbb{B}_0^*}, \tag{12}$$

where $\mathfrak{I}_\chi \xleftarrow{\cup} \mathbb{F}_q$, and all the other variables are generated as in $\mathsf{Hyb}_{2\text{-}\chi\text{-}4}$.

$\mathsf{Hyb}_{2\text{-}\chi\text{-}6}$ ($\chi \in [q_{\mathrm{key}}]$)

This experiment is analogous to $\mathsf{Hyb}_{2\text{-}\chi\text{-}5}$ except that in this experiment, the verification-text used to verify the forged signature outputted by $\mathcal{A}$ on some message $\mathsf{MSG} \in \mathbb{M}$ under some signing policy predicate $R^{(q)}_{\text{Z-ABP}}(f,\cdot) : \mathbb{F}_q^n \to \{0,1\} \in \mathcal{R}^{(q)}_{\text{Z-ABP}}$ having ASP-representation $\mathbb{S} = (\mathbb{U} = \{(\vec{y}^{(j)}, \vec{z}^{(j)})\}_{j\in[m]} \subset (\mathbb{F}_q^\ell)^2, \rho : [m] \to [n])$ is generated as $(\boldsymbol{c}^{(0)}, \ldots, \boldsymbol{c}^{(m+1)})$ such that $\boldsymbol{c}^{(0)}$, $\boldsymbol{c}^{(m+1)}$ have the same form as in Eq. (7) and $\{\boldsymbol{c}^{(j)}\}_{j\in[m]}$ are given by Eq. (9) where $\widetilde{s}_j = \vec{\widetilde{u}} \cdot \vec{y}^{(j)}$, $\widetilde{s}'_j = \vec{\widetilde{u}} \cdot \vec{z}^{(j)}$ for $j \in [m]$, $\boldsymbol{Z}^{(\iota)} \in \{\boldsymbol{Z} \in \mathsf{GL}(2, \mathbb{F}_q) \mid \vec{e}^{(2,2)} = (1, x_\iota^{(\chi)})(\boldsymbol{Z}^{-1})^\top\}$ for $\iota \in [n]$, and all the other variables are generated as in $\mathsf{Hyb}_{2\text{-}\chi\text{-}5}$.

$\mathsf{Hyb}_{2\text{-}\chi\text{-}7}$ ($\chi \in [q_{\mathrm{key}}]$)

This experiment is analogous to $\mathsf{Hyb}_{2\text{-}\chi\text{-}6}$ with the only exception that in this experiment, the χ^{th} signing key for signing attribute string $\vec{x}^{(\chi)} \in \mathbb{F}_q^n$ requested by $\mathcal{A}$ to reveal is generated as $\mathsf{SK}(\vec{x}^{(\chi)}) = (\boldsymbol{k}^{*(\chi,0)}, \ldots, \boldsymbol{k}^{*(\chi,n)}, \boldsymbol{k}^{*(\chi,n+1,1)}, \boldsymbol{k}^{*(\chi,n+1,2)})$ such that $\boldsymbol{k}^{*(0)}$ is given by Eq. (12), $\{\boldsymbol{k}^{*(\chi,\iota)}\}_{\iota\in[n]}$ are given by Eq. (8), and $\boldsymbol{k}^{*(\chi,n+1,1)}, \boldsymbol{k}^{*(\chi,n+1,2)}$ are given by Eq. (4), where all the variables are generated as in $\mathsf{Hyb}_{2\text{-}\chi\text{-}6}$.

$\mathsf{Hyb}_{2\text{-}\chi\text{-}8}$ ($\chi \in [q_{\mathrm{key}}]$)

This experiment is analogous to $\mathsf{Hyb}_{2\text{-}\chi\text{-}7}$ except that in this experiment, the verification-text used to verify the forged signature outputted by $\mathcal{A}$ on some message $\mathsf{MSG} \in \mathbb{M}$ under some signing policy predicate $R^{(q)}_{\text{Z-ABP}}(f,\cdot) : \mathbb{F}_q^n \to \{0,1\} \in \mathcal{R}^{(q)}_{\text{Z-ABP}}$ having ASP-representation $\mathbb{S} = (\mathbb{U} = \{(\vec{y}^{(j)}, \vec{z}^{(j)})\}_{j\in[m]} \subset (\mathbb{F}_q^\ell)^2, \rho : [m] \to [n])$ is generated as $(\boldsymbol{c}^{(0)}, \ldots, \boldsymbol{c}^{(m+1)})$ such that $\{\boldsymbol{c}^{(j)}\}_{j\in[0,m+1]}$ have the same form as in Eq. (7), where $\{\vec{r}^{(j)}\}_{j\in[m]} \xleftarrow{\cup} \mathbb{F}_q^2$, and all the other variables are generated as in $\mathsf{Hyb}_{2\text{-}\chi\text{-}7}$.

$\mathsf{Hyb}_{2\text{-}\chi\text{-}9}$ ($\chi \in [q_{\mathrm{key}}]$)

This experiment is analogous to $\mathsf{Hyb}_{2\text{-}\chi\text{-}8}$ with the only exception that in this experiment, the χ^{th} signing key for signing attribute string $\vec{x}^{(\chi)} \in \mathbb{F}_q^n$ requested by $\mathcal{A}$ to reveal is generated as $\mathsf{SK}(\vec{x}^{(\chi)}) = (\boldsymbol{k}^{*(\chi,0)}, \ldots, \boldsymbol{k}^{*(\chi,n)}, \boldsymbol{k}^{*(\chi,n+1,1)}, \boldsymbol{k}^{*(\chi,n+1,2)})$ such that $\boldsymbol{k}^{*(0)}$ is given by Eq. (12), and $\{\boldsymbol{k}^{*(\chi,\iota)}\}_{\iota\in[n]}$, $\boldsymbol{k}^{*(\chi,n+1,1)}, \boldsymbol{k}^{*(\chi,n+1,2)}$ are given by Eq. (4), where all the variables are generated as in $\mathsf{Hyb}_{2\text{-}\chi\text{-}8}$.

Hyb_3

This experiment is identical to $\mathsf{Hyb}_{2\text{-}q_{\mathrm{key}}\text{-}9}$ except that in this experiment, the verification-text used to verify the forged signature outputted by $\mathcal{A}$ on some message $\mathsf{MSG} \in \mathbb{M}$ under some signing policy predicate $R^{(q)}_{\text{Z-ABP}}(f,\cdot) : \mathbb{F}_q^n \to \{0,1\} \in \mathcal{R}^{(q)}_{\text{Z-ABP}}$ having ASP-representation $\mathbb{S} = (\mathbb{U} = \{(\vec{y}^{(j)}, \vec{z}^{(j)})\}_{j\in[m]} \subset (\mathbb{F}_q^\ell)^2, \rho : [m] \to [n])$ is generated as $(\boldsymbol{c}^{(0)}, \ldots, \boldsymbol{c}^{(m+1)})$ such that $\{\boldsymbol{c}^{(j)}\}_{j\in[m+1]}$ have the same form as in Eq. (7), and

$$\boldsymbol{c}^{(0)} = (-u - u_\ell, \boxed{v}, 0, \eta_0)_{\mathbb{B}_0}, \tag{13}$$

where $v \xleftarrow{\cup} \mathbb{F}_q$, and all the other variables are generated as in $\mathsf{Hyb}_{2\text{-}q_{\mathrm{key}}\text{-}9}$.

$\mathsf{Hyb}_{4\text{-}\pi}$ ($\pi \in [q_{\mathrm{sig}}]$)

$\mathsf{Hyb}_{4\text{-}0}$ coincides with Hyb_3. This experiment is the same as $\mathsf{Hyb}_{4\text{-}(\pi-1)}$ except that in this experiment, the π^{th} signature associated with the triple $(\mathsf{MSG}_\pi, \mathbb{S}_\pi, \vec{x}^{(\pi)}) \in \mathbb{M} \times \mathcal{R}^{(q,n)}_{\text{Z-ABP}} \times \mathbb{F}_q^n$ that $\mathcal{A}$ requests to reveal, where $\mathbb{S}_\pi = (\mathbb{U}_\pi = \{(\vec{y}^{(\pi,j)}, \vec{z}^{(\pi,j)})\}_{j\in[m_\pi]} \subset (\mathbb{F}_q^{\ell_\pi})^2, \rho_\pi : [m_\pi] \to [n])$, is created as $\mathrm{sig}_\pi = (\boldsymbol{s}^{*(\pi,0)}, \ldots, \boldsymbol{s}^{*(\pi,m_\pi+1)})$ such that $\{\boldsymbol{s}^{*(\pi,j)}\}_{j\in[m_\pi]}$ have the same form as in Eq. (5), and

$$\begin{aligned} \boldsymbol{s}^{*(\pi,0)} &= (\widehat{\omega}_\pi, \boxed{\zeta_{\pi,0}}, \widehat{\upsilon}_{\pi,0}, 0)_{\mathbb{B}_0^*}, \\ \boldsymbol{s}^{*(\pi,m_\pi+1)} &= (\widehat{\omega}_\pi (1, \mathsf{H}^{(\lambda,\mathsf{poly})}_{\mathsf{hk}}(\mathsf{MSG}_\pi \| \mathbb{S}_\pi)), \\ &\quad \boxed{\vec{\zeta}^{(\pi,m_\pi+1)}}, \vec{\widehat{\upsilon}}^{(\pi,m_\pi+1)}, \vec{0}^2)_{\mathbb{B}_2^*}, \end{aligned} \tag{14}$$

where $\zeta_{\pi,0} \xleftarrow{\mathsf{U}} \mathbb{F}_q$, $\vec{\zeta}^{(\pi,m_\pi+1)} \xleftarrow{\mathsf{U}} \mathbb{F}_q^2$, and all the other variables are generated as in $\mathsf{Hyb}_{4\text{-}(\pi-1)}$.

Hyb_{5}

This experiment is identical to $\mathsf{Hyb}_{4\text{-}q_{\mathsf{sig}}}$ except that in this experiment, the verification-text used to verify the forged signature outputted by $\mathcal{A}$ on some message $\mathsf{MSG} \in \mathbb{M}$ under some signing policy predicate $R^{(q)}_{\mathsf{Z\text{-}ABP}}(f,\cdot) : \mathbb{F}_q^n \to \{0,1\} \in \mathcal{R}^{(q)}_{\mathsf{Z\text{-}ABP}}$ having ASP-representation $\mathbb{S} = (\mathbb{U} = \{(\vec{y}^{(j)}, \vec{z}^{(j)})\}_{j\in[m]} \subset (\mathbb{F}_q^\ell)^2, \rho : [m] \to [n])$ is generated as $(\boldsymbol{c}^{(0)}, \ldots, \boldsymbol{c}^{(m+1)})$ such that $\{\boldsymbol{c}^{(j)}\}_{j\in[m+1]}$ have the same form as in Eq. (7), and

$$\boldsymbol{c}^{(0)} = (\boxed{w}, v, 0, \eta_0)_{\mathbb{B}_0}, \tag{15}$$

where $w \xleftarrow{\mathsf{U}} \mathbb{F}_q$, and all the other variables are generated as in $\mathsf{Hyb}_{5\text{-}q_{\mathsf{sig}}}$.

▷ Analysis

Let us now denote by $\mathsf{Adv}^{(i)}_{\mathcal{A}}(\lambda)$ the probability that $\mathcal{A}$ wins in Hyb_i for $i \in \{0, 0', 1, \{2\text{-}\chi\text{-}k\}_{\chi\in[q_{\mathsf{key}}],k\in[9]}, 3, \{4\text{-}\pi\}_{\pi\in[q_{\mathsf{sig}}]}, 5\}$. By definition, we clearly have $\mathsf{Adv}^{\mathsf{ABS,UF}}_{\mathcal{A}}(\lambda) \equiv \mathsf{Adv}^{(0)}_{\mathcal{A}}(\lambda)$, $\mathsf{Adv}^{(1)}_{\mathcal{A}}(\lambda) \equiv \mathsf{Adv}^{(2\text{-}0\text{-}9)}_{\mathcal{A}}(\lambda)$, and $\mathsf{Adv}^{(3)}_{\mathcal{A}}(\lambda) \equiv \mathsf{Adv}^{(4\text{-}0)}_{\mathcal{A}}(\lambda)$. Hence, we have

$$\begin{aligned}
&\mathsf{Adv}^{\mathsf{ABS,UF}}_{\mathcal{A}}(\lambda) \leq \\
&\left|\mathsf{Adv}^{(0)}_{\mathcal{A}}(\lambda) - \mathsf{Adv}^{(0')}_{\mathcal{A}}(\lambda)\right| + \left|\mathsf{Adv}^{(0')}_{\mathcal{A}}(\lambda) - \mathsf{Adv}^{(1)}_{\mathcal{A}}(\lambda)\right| + \\
&\sum_{\chi\in[q_{\mathsf{key}}]} \Big[\left|\mathsf{Adv}^{(2\text{-}(\chi-1)\text{-}9)}_{\mathcal{A}}(\lambda) - \mathsf{Adv}^{(2\text{-}\chi\text{-}1)}_{\mathcal{A}}(\lambda)\right| + \\
&\sum_{k\in[8]} \left|\mathsf{Adv}^{(2\text{-}\chi\text{-}k)}_{\mathcal{A}}(\lambda) - \mathsf{Adv}^{(2\text{-}\chi\text{-}(k+1))}_{\mathcal{A}}(\lambda)\right|\Big] + \\
&\left|\mathsf{Adv}^{(2\text{-}q_{\mathsf{key}}\text{-}9)}_{\mathcal{A}}(\lambda) - \mathsf{Adv}^{(3)}_{\mathcal{A}}(\lambda)\right| + \\
&\sum_{\pi\in[q_{\mathsf{sig}}]} \left|\mathsf{Adv}^{(4\text{-}(\pi-1))}_{\mathcal{A}}(\lambda) - \mathsf{Adv}^{(4,\pi)}_{\mathcal{A}}(\lambda)\right| + \\
&\left|\mathsf{Adv}^{(4\text{-}q_{\mathsf{sig}})}_{\mathcal{A}}(\lambda) - \mathsf{Adv}^{(5)}_{\mathcal{A}}(\lambda)\right| + \mathsf{Adv}^{(5)}_{\mathcal{A}}(\lambda).
\end{aligned} \tag{16}$$

Then Theorem 2 follows from Lemmas 11–26 presented below, in conjunction with Lemmas 3–10 of Sect. 2.4. ∎

Lemma 11: *For any stateful probabilistic adversary $\mathcal{A}$, for any security parameter λ, $\mathsf{Adv}^{(0)}_{\mathcal{A}}(\lambda) = \mathsf{Adv}^{(0')}_{\mathcal{A}}(\lambda)$.*

Proof: Lemma 11 follows directly from Theorem 1. ∎

Lemma 12: *For any stateful probabilistic adversary $\mathcal{A}$, there exists a probabilistic algorithm $\mathcal{B}_1$, whose running time is essentially the same as that of $\mathcal{A}$, such that for any security parameter λ, $\left|\mathsf{Adv}^{(0')}_{\mathcal{A}}(\lambda) - \mathsf{Adv}^{(1)}_{\mathcal{A}}(\lambda)\right| \leq \mathsf{Adv}^{\mathsf{P1}}_{\mathcal{B}_1}(\lambda) + 5/q$.*

Proof: In order to prove Lemma 12, we construct below a probabilistic algorithm $\mathcal{B}_1$ against Problem 1 using as a blackbox sub-routine a stateful probabilistic adversary $\mathcal{A}$ that distinguishes between $\mathsf{Hyb}_{0'}$ and Hyb_1. Suppose $\mathcal{B}_1$ is given an instance of Problem 1

$$\begin{aligned}
\mathcal{O}^{\mathsf{P1}}_{\widehat{\beta}} = (&\mathsf{params}, \{\mathbb{B}_\iota, \widetilde{\mathbb{B}}^*_\iota\}_{\iota\in[0,2]}, \{\boldsymbol{e}^{(\alpha,\nu,\widehat{\beta})}\}_{\alpha\in[2],\nu\in[2]}, \\
&\boldsymbol{f}^{(0,\widehat{\beta})}, \{\boldsymbol{f}^{(1,\nu,\widehat{\beta})}\}_{\nu\in[2]}),
\end{aligned}$$

where

$$\begin{aligned}
&(\mathsf{params}, \{\mathbb{B}_\iota, \mathbb{B}^*_\iota\}_{\iota\in[0,2]}) \xleftarrow{\mathsf{R}} \mathcal{G}_{\mathsf{OB}}(2, (4, 14, 8)); \\
&\widetilde{\mathbb{B}}^*_0 = \{\boldsymbol{b}^{*(0,1)}, \boldsymbol{b}^{*(0,3)}, \boldsymbol{b}^{*(0,4)}\}; \\
&\widetilde{\mathbb{B}}^*_1 = \{\boldsymbol{b}^{*(1,1)}, \ldots, \boldsymbol{b}^{*(1,4)}, \boldsymbol{b}^{*(1,7)}, \boldsymbol{b}^{*(1,8)}, \boldsymbol{b}^{*(1,11)}, \ldots, \\
&\qquad \boldsymbol{b}^{*(1,14)}\}; \\
&\widetilde{\mathbb{B}}^*_2 = \{\boldsymbol{b}^{*(2,1)}, \boldsymbol{b}^{*(2,2)}, \boldsymbol{b}^{*(2,5)}, \ldots, \boldsymbol{b}^{*(2,8)}\}; \\
&\delta, \tau, \{\theta_\nu, \theta'_\nu\}_{\nu\in[2]}, \gamma_0 \xleftarrow{\mathsf{U}} \mathbb{F}_q, \\
&\{\vec{\gamma}^{(\nu)}, \vec{\gamma}'^{(\nu)}, \vec{\gamma}''^{(\nu)}\}_{\nu\in[2]} \xleftarrow{\mathsf{U}} \mathbb{F}_q^2; \\
&\left.\begin{aligned}
\boldsymbol{e}^{(1,\nu,0)} &= (\vec{0}^4, \vec{0}^6, \vec{0}^2, \vec{\gamma}^{(\nu)})_{\mathbb{B}_1} \\
\boldsymbol{e}^{(1,\nu,1)} &= (\vec{0}^4, \vec{0}^4, \theta_\nu \vec{e}^{(2,\nu)}, \vec{0}^2, \vec{\gamma}^{(\nu)})_{\mathbb{B}_1} \\
\boldsymbol{e}^{(2,\nu,0)} &= (\vec{0}^2, \vec{0}^2, \vec{0}^2, \vec{\gamma}'^{(\nu)})_{\mathbb{B}_2} \\
\boldsymbol{e}^{(2,\nu,1)} &= (\vec{0}^2, \theta'_\nu \vec{e}^{(2,\nu)}, \vec{0}^2, \vec{\gamma}'^{(\nu)})_{\mathbb{B}_2}
\end{aligned}\right\} \text{for } \nu \in [2]; \\
&\boldsymbol{f}^{(0,0)} = (\delta, 0, 0, \gamma_0)_{\mathbb{B}_0}, \boldsymbol{f}^{(0,1)} = (\delta, \tau, 0, \gamma_0)_{\mathbb{B}_0}; \\
&\left.\begin{aligned}
\boldsymbol{f}^{(1,\nu,0)} &= (\vec{0}^2, \delta \vec{e}^{(2,\nu)}, \vec{0}^6, \vec{0}^2, \vec{\gamma}''^{(\nu)})_{\mathbb{B}_1} \\
\boldsymbol{f}^{(1,\nu,1)} &= (\vec{0}^2, \delta \vec{e}^{(2,\nu)}, \tau \vec{e}^{(2,\nu)}, \vec{0}^4, \vec{0}^2, \vec{\gamma}''^{(\nu)})_{\mathbb{B}_1}
\end{aligned}\right\} \text{for } \nu \in [2].
\end{aligned}$$

$\mathcal{B}_1$ interacts with $\mathcal{A}$ as follows:

1. At first, $\mathcal{B}_1$ sets

$$\begin{aligned}
\widehat{\mathbb{B}}_0 &= \{\boldsymbol{b}^{(0,1)}, \boldsymbol{b}^{(0,4)}\}, \\
\widehat{\mathbb{B}}^*_0 &= \{\boldsymbol{b}^{*(0,3)}\}, \\
\widehat{\mathbb{B}}_1 &= \{\boldsymbol{b}^{(1,1)}, \ldots, \boldsymbol{b}^{(1,4)}, \boldsymbol{b}^{(1,13)}, \boldsymbol{b}^{(1,14)}\}, \\
\widehat{\mathbb{B}}^*_1 &= \{\boldsymbol{b}^{*(1,1)}, \ldots, \boldsymbol{b}^{*(1,4)}, \boldsymbol{b}^{*(1,11)}, \boldsymbol{b}^{*(1,12)}\}, \\
\widehat{\mathbb{B}}_2 &= \{\boldsymbol{b}^{(2,1)}, \boldsymbol{b}^{(2,2)}, \boldsymbol{b}^{(2,7)}, \boldsymbol{b}^{(2,8)}\}, \\
\widehat{\mathbb{B}}^*_2 &= \{\boldsymbol{b}^{*(2,1)}, \boldsymbol{b}^{*(2,2)}, \boldsymbol{b}^{*(2,5)}, \boldsymbol{b}^{*(2,6)}\},
\end{aligned}$$

   using $\{\mathbb{B}_\iota, \widetilde{\mathbb{B}}^*_\iota\}_{\iota\in[0,2]}$, which are part of the given Problem 1 instance. It also samples a hashing key $\mathsf{hk} \xleftarrow{\mathsf{R}} \mathsf{KGen}()$ for a hash function family $\mathbb{H}$ associated with $\mathcal{G}_{\mathsf{BPG}}$ and the polynomial $\mathsf{poly}(\cdot)$, where $\mathsf{poly}(\lambda)$ represents the length of the bit string formed by concatenating a message belonging to $\mathbb{M}$ and the binary representation of an ASP representing a signing policy predicate in $\mathcal{R}^{(q)}_{\mathsf{Z\text{-}ABP}}$. It provides $\mathcal{A}$ with the public parameters $\mathsf{MPK} = (\mathsf{hk}, \mathsf{params}, \{\widehat{\mathbb{B}}_\iota, \widehat{\mathbb{B}}^*_\iota\}_{\iota\in[0,2]})$.

2. For all $h \in [q_{\mathsf{key}}]$, in response to the h^{th} signing key reveal query of $\mathcal{A}$ for some $\vec{x}^{(h)} \in \mathbb{F}_q^n$, $\mathcal{B}_1$ gives $\mathcal{A}$ a signing key $\mathsf{SK}(\vec{x}^{(h)}) = (\boldsymbol{k}^{*(h,0)}, \ldots, \boldsymbol{k}^{*(h,n)}, \boldsymbol{k}^{*(h,n+1,1)}, \boldsymbol{k}^{*(h,n+1,2)})$ whose components are generated as in Eq. (4) using $\{\widetilde{\mathbb{B}}^*_\iota\}_{\iota\in[0,2]}$ included within the given Problem 1

instance.

3. Similarly, for all $t \in [q_{\mathsf{sig}}]$, in response to the t^{th} signature reveal query of $\mathcal{A}$ for some triple $(\mathsf{MSG}_t, \mathbb{S}_t, \vec{x}^{(t)}) \in \mathbb{M} \times \mathcal{R}^{(q)}_{\mathsf{Z\text{-}ABP}} \times \mathbb{F}_q^n$, $\mathcal{B}_1$ hands $\mathcal{A}$ a signature $\mathsf{sig}_t = (\boldsymbol{s}^{*(t,0)}, \ldots, \boldsymbol{s}^{*(t,m_t+1)})$ whose components are computed as in Eq. (5) using $\{\widetilde{\mathbb{B}}^*_t\}_{t\in[0,2]}$ of the given Problem 1 instance.

4. When $\mathcal{A}$ outputs a forgery sig on some message MSG under some signing policy ASP $\mathbb{S} = (\mathbb{U} = \{(\vec{y}^{(j)}, \vec{z}^{(j)})\}_{j\in[m]} \subset (\mathbb{F}_q^\ell)^2, \rho : [m] \to [n]) \in \mathcal{R}^{(q)}_{\mathsf{Z\text{-}ABP}}$, $\mathcal{B}_1$ computes the verification-text $(\boldsymbol{c}^{(0)}, \ldots, \boldsymbol{c}^{(m+1)})$ as

$$\begin{aligned}
\boldsymbol{c}^{(0)} &= -u^\dagger_\ell \boldsymbol{f}^{(0,\widehat{\beta})} + (-u^\ddagger_\ell - u)\boldsymbol{b}^{(0,1)},\\
\boldsymbol{c}^{(j)} &= \mu_j(\rho(j)\boldsymbol{b}^{(1,1)} - \boldsymbol{b}^{(1,2)}) + s'^\dagger_j \boldsymbol{f}^{(1,1,\widehat{\beta})} +\\
&\quad s^\dagger_j \boldsymbol{f}^{(1,2,\widehat{\beta})} + s'^\ddagger_j \boldsymbol{b}^{(1,3)} + s^\ddagger_j \boldsymbol{b}^{(1,4)} +\\
&\quad \sum_{\nu\in[2]} r^{\dagger(j)}_\nu \boldsymbol{e}^{(1,\nu,\widehat{\beta})} + \sum_{\nu\in[2]} \eta^{\dagger(j)}_\nu \boldsymbol{b}^{(1,12+\nu)}\\
&\quad \text{for } j \in [m],\\
\boldsymbol{c}^{(m+1)} &= u\boldsymbol{b}^{(2,1)} + \kappa(-\mathsf{H}^{(\lambda,\mathsf{poly})}_{\mathsf{hk}}(\mathsf{MSG}\|\mathbb{S})\boldsymbol{b}^{(2,1)} + \boldsymbol{b}^{(2,2)}) +\\
&\quad \sum_{\nu\in[2]} \boldsymbol{e}^{(2,\nu,\widehat{\beta})},
\end{aligned}$$

where $\vec{u}^\dagger = (u^\dagger_1, \ldots, u^\dagger_\ell), \vec{u}^\ddagger = (u^\ddagger_1, \ldots, u^\ddagger_\ell) \xleftarrow{\mathsf{U}} \mathbb{F}_q^\ell$, $s^\dagger_j = \vec{u}^\dagger \cdot \vec{y}^{(j)}$, $s'^\dagger_j = \vec{u}^\dagger \cdot \vec{z}^{(j)}$, $s^\ddagger_j = \vec{u}^\ddagger \cdot \vec{y}^{(j)}$, $s'^\ddagger_j = \vec{u}^\ddagger \cdot \vec{z}^{(j)}$ for $j \in [m]$, $u, \kappa, \{\mu_j\}_{j\in[m]} \xleftarrow{\mathsf{U}} \mathbb{F}_q$, $\{\vec{r}^{\dagger(j)}, \vec{\eta}^{\dagger(j)}\}_{j\in[m]} \xleftarrow{\mathsf{U}} \mathbb{F}_q^2$, and $\{\mathbb{B}_\iota\}_{\iota\in[0,2]}, \{\boldsymbol{e}^{(1,\nu,\widehat{\beta})}\}_{\nu\in[2]}$, $\{\boldsymbol{e}^{(2,\nu,\widehat{\beta})}\}_{\nu\in[2]}$, $\boldsymbol{f}^{(0,\widehat{\beta})}, \{\boldsymbol{f}^{(1,\nu,\widehat{\beta})}\}_{\nu\in[2]}$ are taken from the given Problem 1 instance. $\mathcal{B}_1$ then verifies the validity of the forged signature outputted by $\mathcal{A}$ using the above verification-text, and outputs 1 if the verification succeeds, and 0 otherwise.

It is straightforward to verify that the distribution of $\mathcal{A}$'s view simulated by $\mathcal{B}_1$ given a Problem 1 instance $\varrho^{\mathsf{P1}}_{\widehat{\beta}}$ for $\widehat{\beta} \in \{0,1\}$ coincides with that in $\mathsf{Hyb}_{0'}$ if $\widehat{\beta} = 0$. Similarly, the view of $\mathcal{A}$ simulated by $\mathcal{B}_1$ given a Problem 1 instance $\varrho^{\mathsf{P1}}_{\widehat{\beta}}$ for $\widehat{\beta} \in \{0,1\}$ coincides with that in Hyb_1 in case $\widehat{\beta} = 1$ except when any one of $\tau, \{\theta_\nu, \theta'_\nu\}_{\nu\in[2]}$ is 0, i.e., except with probability $5/q$. This completes the proof of Lemma 12. ∎

Lemma 13: *For any stateful probabilistic adversary $\mathcal{A}$, there exists a probabilistic algorithm $\mathcal{B}_{2\text{-}1}$, whose running time is essentially the same as that of $\mathcal{A}$, such that for any security parameter λ, $\left|\mathsf{Adv}^{(2\text{-}(\chi-1)\text{-}9)}_{\mathcal{A}}(\lambda) - \mathsf{Adv}^{(2\text{-}\chi\text{-}1)}_{\mathcal{A}}(\lambda)\right| \le \mathsf{Adv}^{\mathsf{P2}}_{\mathcal{B}_{2\text{-}\chi\text{-}1}}(\lambda) + 3/q$ for all $\chi \in [q_{key}]$, where $\mathcal{B}_{2\text{-}\chi\text{-}1}(\cdot) = \mathcal{B}_{2\text{-}1}(\chi, \cdot)$ for any $\chi \in \mathbb{N}$.*

Proof: In order to prove Lemma 13, we construct below a probabilistic algorithm $\mathcal{B}_{2\text{-}1}$ against Problem 2 using as a blackbox sub-routine a stateful probabilistic adversary $\mathcal{A}$ that distinguishes between $\mathsf{Hyb}_{2\text{-}(\chi-1)\text{-}9}$ and $\mathsf{Hyb}_{2\text{-}\chi\text{-}1}$. Suppose $\mathcal{B}_{2\text{-}1}$ is given $\chi \in [q_{\mathsf{key}}]$ together with an instance of Problem 2

$$\begin{aligned}
\varrho^{\mathsf{P2}}_{\widehat{\beta}} = (&\mathsf{params}, \{\widetilde{\mathbb{B}}_\iota, \mathbb{B}^*_\iota\}_{\iota\in[0,1]}, \mathbb{B}_2, \mathbb{B}^*_2, \boldsymbol{h}^{*(0,\widehat{\beta})}, \boldsymbol{f}^{(0)},\\
&\{\boldsymbol{h}^{*(1,\nu,\widehat{\beta})}, \boldsymbol{f}^{(1,\nu)}\}_{\nu\in[2]}, \{\boldsymbol{h}^{*(2,\nu)}\}_{\nu\in[2]}),
\end{aligned}$$

where

$$\begin{aligned}
&(\mathsf{params}, \{\mathbb{B}_\iota, \mathbb{B}^*_\iota\}_{\iota\in[0,2]}) \xleftarrow{\mathsf{R}} \mathcal{G}_{\mathsf{OB}}(2, (4, 14, 8));\\
&\widetilde{\mathbb{B}}_0 = \{\boldsymbol{b}^{(0,1)}, \boldsymbol{b}^{(0,3)}, \boldsymbol{b}^{(0,4)}\};\\
&\widetilde{\mathbb{B}}_1 = \{\boldsymbol{b}^{(1,1)}, \ldots, \boldsymbol{b}^{(1,4)}, \boldsymbol{b}^{(1,7)}, \ldots, \boldsymbol{b}^{(1,14)}\};\\
&\vartheta, \varkappa, \delta, \tau, \xi_0 \xleftarrow{\mathsf{U}} \mathbb{F}_q, \{\vec{\xi}^{(\nu)}\}_{\nu\in[2]} \xleftarrow{\mathsf{U}} \mathbb{F}_q^2;\\
&\boldsymbol{h}^{*(0,0)} = (\vartheta, 0, \xi_0, 0)_{\mathbb{B}^*_0}, \boldsymbol{h}^{*(0,1)} = (\vartheta, \varkappa, \xi_0, 0)_{\mathbb{B}^*_0};\\
&\boldsymbol{f}^{(0)} = (\delta, \tau, 0, 0)_{\mathbb{B}_0};\\
&\left.\begin{aligned}
\boldsymbol{h}^{*(1,\nu,0)} &= (\vec{0}^2, \vartheta\vec{e}^{(2,\nu)}, \vec{0}^6, \vec{\xi}^{(\nu)}, \vec{0}^2)_{\mathbb{B}^*_1}\\
\boldsymbol{h}^{*(1,\nu,1)} &= (\vec{0}^2, \vartheta\vec{e}^{(2,\nu)}, \varkappa\vec{e}^{(2,\nu)}, \vec{0}^4, \vec{\xi}^{(\nu)}, \vec{0}^2)_{\mathbb{B}^*_1}\\
\boldsymbol{f}^{(1,\nu)} &= (\vec{0}^2, \delta\vec{e}^{(2,\nu)}, \tau\vec{e}^{(2,\nu)}, \vec{0}^4, \vec{0}^2, \vec{0}^2)_{\mathbb{B}_1}
\end{aligned}\right\} \text{for } \nu \in [2];\\
&\boldsymbol{h}^{*(2,\nu)} = \vartheta\boldsymbol{b}^{*(2,\nu)} \text{ for } \nu \in [2].
\end{aligned}$$

$\mathcal{B}_{2\text{-}1}$ interacts with $\mathcal{A}$ as follows:

1. At first, $\mathcal{B}_{2\text{-}1}$ sets

$$\begin{aligned}
\widehat{\mathbb{B}}_0 &= \{\boldsymbol{b}^{(0,1)}, \boldsymbol{b}^{(0,4)}\},\\
\widehat{\mathbb{B}}^*_0 &= \{\boldsymbol{b}^{*(0,3)}\},\\
\widehat{\mathbb{B}}_1 &= \{\boldsymbol{b}^{(1,1)}, \ldots, \boldsymbol{b}^{(1,4)}, \boldsymbol{b}^{(1,13)}, \boldsymbol{b}^{(1,14)}\},\\
\widehat{\mathbb{B}}^*_1 &= \{\boldsymbol{b}^{*(1,1)}, \ldots, \boldsymbol{b}^{*(1,4)}, \boldsymbol{b}^{*(1,11)}, \boldsymbol{b}^{*(1,12)}\},\\
\widehat{\mathbb{B}}_2 &= \{\boldsymbol{b}^{(2,1)}, \boldsymbol{b}^{(2,2)}, \boldsymbol{b}^{(2,7)}, \boldsymbol{b}^{(2,8)}\},\\
\widehat{\mathbb{B}}^*_2 &= \{\boldsymbol{b}^{*(2,1)}, \boldsymbol{b}^{*(2,2)}, \boldsymbol{b}^{*(2,5)}, \boldsymbol{b}^{*(2,6)}\},
\end{aligned}$$

using $\{\widetilde{\mathbb{B}}_\iota, \mathbb{B}^*_\iota\}_{\iota\in\{0,1\}}$, $\mathbb{B}_2$, and $\mathbb{B}^*_2$, which are part of the given Problem 2 instance. It also samples a hashing key $\mathsf{hk} \xleftarrow{\mathsf{R}} \mathsf{KGen}()$ for a hash function family $\mathbb{H}$ associated with $\mathcal{G}_{\mathsf{BPG}}$ and the polynomial $\mathsf{poly}(\cdot)$, where $\mathsf{poly}(\lambda)$ represents the length of the bit string formed by concatenating a message belonging to $\mathbb{M}$ and the binary representation of an ASP representing a signing policy predicate in $\mathcal{R}^{(q)}_{\mathsf{Z\text{-}ABP}}$. It provides $\mathcal{A}$ with the public parameters $\mathsf{MPK} = (\mathsf{hk}, \mathsf{params}, \{\widehat{\mathbb{B}}_\iota, \widehat{\mathbb{B}}^*_\iota\}_{\iota\in[0,2]})$.

2. For $h \in [q_{\mathsf{key}}]$, in response to the h^{th} signing key reveal query of $\mathcal{A}$ for some $\vec{x}^{(h)} \in \mathbb{F}_q^n$, $\mathcal{B}_{2\text{-}1}$ gives $\mathcal{A}$ a signing key $\mathsf{SK}(\vec{x}^{(h)}) = (\boldsymbol{k}^{*(h,0)}, \ldots, \boldsymbol{k}^{*(h,n)}, \boldsymbol{k}^{*(h,n+1,1)}, \boldsymbol{k}^{*(h,n+1,2)})$ whose components are generated as follows:

   a. ($h < \chi$) $\mathcal{B}_{2\text{-}1}$ computes $\boldsymbol{k}^{*(h,0)}$ as in Eq. (12), while $\boldsymbol{k}^{*(h,1)}, \ldots, \boldsymbol{k}^{*(h,n)}, \boldsymbol{k}^{*(h,n+1,1)}, \boldsymbol{k}^{*(h,n+1,2)}$ as in Eq. (4) using $\{\mathbb{B}^*_\iota\}_{\iota\in[0,2]}$ included within the given Problem 2 instance.

b. $(h = \chi)$ $\mathcal{B}_{2\text{-}1}$ computes $\boldsymbol{k}^{*(\chi,0)}, \ldots, \boldsymbol{k}^{*(\chi,n)}, \boldsymbol{k}^{*(\chi,n+1,1)}, \boldsymbol{k}^{*(\chi,n+1,2)}$ as follows:

$$\begin{aligned}
\boldsymbol{k}^{*(\chi,0)} &= \boldsymbol{h}^{*(0,\widehat{\beta})}, \\
\boldsymbol{k}^{*(\chi,\iota)} &= \sigma_{\chi,\iota}(\boldsymbol{b}^{*(1,1)} + \iota \boldsymbol{b}^{*(1,2)}) + \boldsymbol{h}^{*(1,1,\widehat{\beta})} + \\
&\quad x_\iota^{(\chi)} \boldsymbol{h}^{*(1,2,\widehat{\beta})} + \sum_{\nu\in[2]} \varphi_\nu^{\dagger(\chi,\iota)} \boldsymbol{b}^{*(1,10+\nu)} \\
&\quad \text{for } \iota \in [n], \\
\boldsymbol{k}^{*(\chi,n+1,\nu)} &= \boldsymbol{h}^{*(2,\nu)} + \sum_{\alpha\in[2]} \varphi_\alpha^{(\chi,n+1,\nu)} \boldsymbol{b}^{*(2,4+\alpha)} \\
&\quad \text{for } \nu \in [2],
\end{aligned}$$

where $\{\sigma_{\chi,\iota}\}_{\iota\in[n]} \xleftarrow{\mathsf{U}} \mathbb{F}_q$, $\{\vec{\varphi}^{\dagger(\chi,\iota)}\}_{\iota\in[n]}, \{\vec{\varphi}^{(\chi,n+1,\nu)}\}_{\nu\in[2]} \xleftarrow{\mathsf{U}} \mathbb{F}_q^2$, and $\{\mathbb{B}_\iota^*\}_{\iota\in[0,2]}, \boldsymbol{h}^{*(0,\widehat{\beta})}, \{\boldsymbol{h}^{*(1,\nu,\widehat{\beta})}\}_{\nu\in[2]}, \{\boldsymbol{h}^{*(2,\nu)}\}_{\nu\in[2]}$ are taken from the given Problem 2 instance.

c. $(h > \chi)$ $\mathcal{B}_{2\text{-}1}$ computes $\boldsymbol{k}^{*(h,0)}, \ldots, \boldsymbol{k}^{*(h,n)}, \boldsymbol{k}^{*(h,n+1,1)}, \boldsymbol{k}^{*(h,n+1,2)}$ as in Eq. (4) using $\{\mathbb{B}_\iota^*\}_{\iota\in[0,2]}$ of the given Problem 2 instance.

3. For all $t \in [q_{\mathrm{sig}}]$, in reply to the t^{th} signature reveal query of $\mathcal{A}$ for some triple $(\mathsf{MSG}_t, \mathbb{S}_t, \vec{x}^{(t)}) \in \mathbb{M} \times \mathcal{R}_{\mathsf{Z\text{-}ABP}}^{(q)} \times \mathbb{F}_q^n$, $\mathcal{B}_{2\text{-}1}$ hands $\mathcal{A}$ a signature $\mathrm{sig}_t = (\boldsymbol{s}^{*(t,0)}, \ldots, \boldsymbol{s}^{*(t,m_t+1)})$ whose components are computed as in Eq. (5) using $\{\mathbb{B}_\iota^*\}_{\iota\in[0,2]}$ of the given Problem 2 instance.

4. When $\mathcal{A}$ outputs a forgery sig on some message $\mathsf{MSG} \in \mathbb{M}$ under some signing policy $\mathsf{ASP}\ \mathbb{S} = (\mathbb{U} = \{(\vec{y}^{(j)}, \vec{z}^{(j)})\}_{j\in[m]} \subset (\mathbb{F}_q^\ell)^2, \rho : [m] \to [n]) \in \mathcal{R}_{\mathsf{Z\text{-}ABP}}^{(q)}$, $\mathcal{B}_{2\text{-}1}$ computes the verification-text $(\boldsymbol{c}^{(0)}, \ldots, \boldsymbol{c}^{(m+1)})$ as

$$\begin{aligned}
\boldsymbol{c}^{(0)} &= -u_\ell^\dagger \boldsymbol{f}^{(0)} + (-u_\ell^\ddagger - u)\boldsymbol{b}^{(0,1)} + \eta_0 \boldsymbol{b}^{(0,4)}, \\
\boldsymbol{c}^{(j)} &= \mu_j(\rho(j)\boldsymbol{b}^{(1,1)} - \boldsymbol{b}^{(1,2)}) + s_j'^\dagger \boldsymbol{f}^{(1,1)} + \\
&\quad s_j^\dagger \boldsymbol{f}^{(1,2)} + s_j'^\ddagger \boldsymbol{b}^{(1,3)} + s_j^\ddagger \boldsymbol{b}^{(1,4)} + \\
&\quad \sum_{\nu\in[2]} r_\nu^{(j)} \boldsymbol{b}^{(1,8+\nu)} + \sum_{\nu\in[2]} \eta_\nu^{(j)} \boldsymbol{b}^{(1,12+\nu)} \\
&\quad \text{for } j \in [m], \\
\boldsymbol{c}^{(m+1)} &= u\boldsymbol{b}^{(2,1)} + \kappa(-\mathsf{H}_{\mathsf{hk}}^{(\lambda,\mathsf{poly})}(\mathsf{MSG}\|\mathbb{S})\boldsymbol{b}^{(2,1)} + \boldsymbol{b}^{(2,2)}) + \\
&\quad \sum_{\nu\in[2]} r_\nu^{(m+1)} \boldsymbol{b}^{(2,2+\nu)} + \sum_{\nu\in[2]} \eta_\nu^{(m+1)} \boldsymbol{b}^{(2,6+\nu)},
\end{aligned}$$

where $\vec{u}^\dagger = (u_1^\dagger, \ldots, u_\ell^\dagger), \vec{u}^\ddagger = (u_1^\ddagger, \ldots, u_\ell^\ddagger) \xleftarrow{\mathsf{U}} \mathbb{F}_q^\ell$, $s_j^\dagger = \vec{u}^\dagger \cdot \vec{y}^{(j)}$, $s_j'^\dagger = \vec{u}^\dagger \cdot \vec{z}^{(j)}$, $s_j^\ddagger = \vec{u}^\ddagger \cdot \vec{y}^{(j)}$, $s_j'^\ddagger = \vec{u}^\ddagger \cdot \vec{z}^{(j)}$ for $j \in [m]$, $u, \kappa, \{\mu_j\}_{j\in[m]}, \eta_0 \xleftarrow{\mathsf{U}} \mathbb{F}_q$, $\{\vec{r}^{(j)}, \vec{\eta}^{(j)}\}_{j\in[m+1]} \xleftarrow{\mathsf{U}} \mathbb{F}_q^2$, and $\boldsymbol{f}^{(0)}, \{\boldsymbol{f}^{(1,\nu)}\}_{\nu\in[2]}$ are taken from the given Problem 2 instance. $\mathcal{B}_{2\text{-}1}$ then verifies the validity of the forged signature outputted by $\mathcal{A}$ using the above verification-text, and outputs 1 if the verification succeeds, and 0 otherwise.

It is straightforward to verify that the distribution of $\mathcal{A}$'s view simulated by $\mathcal{B}_{2\text{-}1}$ given $\chi \in [q_{\mathrm{key}}]$ and a Problem 2 instance $\mathcal{O}_{\widehat{\beta}}^{\mathsf{P2}}$ for $\widehat{\beta} \in \{0,1\}$ coincides with that in $\mathsf{Hyb}_{2\text{-}(\chi-1)\text{-}9}$ if $\widehat{\beta} = 0$ except when $\tau = 0$, i.e., except with probability $1/q$. Similarly, the view of $\mathcal{A}$ simulated by $\mathcal{B}_{2\text{-}1}$ given $\chi \in [q_{\mathrm{key}}]$ and a Problem 2 instance $\mathcal{O}_{\widehat{\beta}}^{\mathsf{P2}}$ for $\widehat{\beta} \in \{0,1\}$ coincides with that in $\mathsf{Hyb}_{2\text{-}\chi\text{-}1}$ in case $\widehat{\beta} = 1$ except when any one of $\varkappa, \tau$ is 0, i.e., except with probability $2/q$. This completes the proof of Lemma 13. ∎

Lemma 14: *For any stateful probabilistic adversary $\mathcal{A}$, there exists a probabilistic algorithm $\mathcal{B}_{2\text{-}2}$, whose running time is essentially the same as that of $\mathcal{A}$, such that for any security parameter λ, $\left|\mathsf{Adv}_{\mathcal{A}}^{(2\text{-}\chi\text{-}1)}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{(2\text{-}\chi\text{-}2)}(\lambda)\right| \leq \mathsf{Adv}_{\mathcal{B}_{2\text{-}\chi\text{-}2}}^{\mathsf{P3}}(\lambda) + 2/q$ for all $\chi \in [q_{key}]$, where $\mathcal{B}_{2\text{-}\chi\text{-}2}(\cdot) = \mathcal{B}_{2\text{-}2}(\chi, \cdot)$ for any $\chi \in \mathbb{N}$.*

Proof: In order to prove Lemma 14, we construct below a probabilistic algorithm $\mathcal{B}_{2\text{-}2}$ against Problem 3 using as a blackbox sub-routine a stateful probabilistic adversary $\mathcal{A}$ that distinguishes between $\mathsf{Hyb}_{2\text{-}\chi\text{-}1}$ and $\mathsf{Hyb}_{2\text{-}\chi\text{-}2}$. Suppose $\mathcal{B}_{2\text{-}2}$ is given $\chi \in [q_{\mathrm{key}}]$ together with an instance of Problem 3

$$\mathcal{O}_{\widehat{\beta}}^{\mathsf{P3}} = (\mathsf{params}, \{\mathbb{B}_\iota, \mathbb{B}_\iota^*\}_{\iota\in\{0,2\}}, \mathbb{B}_1, \widetilde{\mathbb{B}}_1^*, \{\boldsymbol{e}^{(1,\nu,\widehat{\beta})}\}_{\nu\in[2]}),$$

where

$$\begin{aligned}
&(\mathsf{params}, \{\mathbb{B}_\iota, \mathbb{B}_\iota^*\}_{\iota\in[0,2]}) \xleftarrow{\mathsf{R}} \mathcal{G}_{\mathsf{OB}}(2, (4, 14, 8)); \\
&\widetilde{\mathbb{B}}_1^* = \{\boldsymbol{b}^{*(1,1)}, \ldots, \boldsymbol{b}^{*(1,8)}, \boldsymbol{b}^{*(1,11)}, \ldots, \boldsymbol{b}^{*(1,14)}\}; \\
&\{\theta_\nu\}_{\nu\in[2]} \xleftarrow{\mathsf{U}} \mathbb{F}_q, \{\vec{\gamma}^{(\nu)}\}_{\nu\in[2]} \xleftarrow{\mathsf{U}} \mathbb{F}_q^2; \\
&\left.\begin{aligned}
\boldsymbol{e}^{(1,\nu,0)} &= (\vec{0}^4, \vec{0}^6, \vec{0}^2, \vec{\gamma}^{(\nu)})_{\mathbb{B}_1} \\
\boldsymbol{e}^{(1,\nu,1)} &= (\vec{0}^4, \vec{0}^4, \theta_\nu \vec{e}^{(2,\nu)}, \vec{0}^2, \vec{\gamma}^{(\nu)})_{\mathbb{B}_1}
\end{aligned}\right\} \text{for } \nu \in [2].
\end{aligned}$$

$\mathcal{B}_{2\text{-}2}$ interacts with $\mathcal{A}$ as follows:

1. At first, $\mathcal{B}_{2\text{-}2}$ sets

$$\begin{aligned}
\widehat{\mathbb{B}}_0 &= \{\boldsymbol{b}^{(0,1)}, \boldsymbol{b}^{(0,4)}\}, \\
\widehat{\mathbb{B}}_0^* &= \{\boldsymbol{b}^{*(0,3)}\}, \\
\widehat{\mathbb{B}}_1 &= \{\boldsymbol{b}^{(1,1)}, \ldots, \boldsymbol{b}^{(1,4)}, \boldsymbol{b}^{(1,13)}, \boldsymbol{b}^{(1,14)}\}, \\
\widehat{\mathbb{B}}_1^* &= \{\boldsymbol{b}^{*(1,1)}, \ldots, \boldsymbol{b}^{*(1,4)}, \boldsymbol{b}^{*(1,11)}, \boldsymbol{b}^{*(1,12)}\}, \\
\widehat{\mathbb{B}}_2 &= \{\boldsymbol{b}^{(2,1)}, \boldsymbol{b}^{(2,2)}, \boldsymbol{b}^{(2,7)}, \boldsymbol{b}^{(2,8)}\}, \\
\widehat{\mathbb{B}}_2^* &= \{\boldsymbol{b}^{*(2,1)}, \boldsymbol{b}^{*(2,2)}, \boldsymbol{b}^{*(2,5)}, \boldsymbol{b}^{*(2,6)}\},
\end{aligned}$$

using $\{\mathbb{B}_\iota, \mathbb{B}_\iota^*\}_{\iota\in\{0,2\}}$, $\mathbb{B}_1$, and $\widetilde{\mathbb{B}}_1^*$, which are part of the given Problem 3 instance. It also samples a hashing key $\mathsf{hk} \xleftarrow{\mathsf{R}} \mathsf{KGen}()$ for a hash function family $\mathbb{H}$ associated with $\mathcal{G}_{\mathsf{BPG}}$ and the polynomial $\mathsf{poly}(\cdot)$, where $\mathsf{poly}(\lambda)$ represents the length of the bit string formed by concatenating a message belonging to $\mathbb{M}$ and the binary representation of an ASP representing a signing policy predicate in $\mathcal{R}_{\mathsf{Z\text{-}ABP}}^{(q)}$. It provides $\mathcal{A}$ with the public parameters $\mathsf{MPK} = (\mathsf{hk}, \mathsf{params}, \{\widehat{\mathbb{B}}_\iota, \widehat{\mathbb{B}}_\iota^*\}_{\iota\in[0,2]})$.

2. For $h \in [q_{\mathrm{key}}]$, in response to the h^{th} signing key reveal query of $\mathcal{A}$ for some $\vec{x}^{(h)} \in \mathbb{F}_q^n$, $\mathcal{B}_{2\text{-}2}$ gives $\mathcal{A}$ a signing key $\mathsf{SK}(\vec{x}^{(h)}) = (\boldsymbol{k}^{*(h,0)}, \ldots, \boldsymbol{k}^{*(h,n)}, \boldsymbol{k}^{*(h,n+1,1)}, \boldsymbol{k}^{*(h,n+1,2)})$ whose components are generated as follows:
   a. $(h < \chi)$ $\mathcal{B}_{2\text{-}2}$ computes $\boldsymbol{k}^{*(h,0)}$ as in Eq. (12), while $\boldsymbol{k}^{*(h,1)}, \ldots, \boldsymbol{k}^{*(h,n)}, \boldsymbol{k}^{*(h,n+1,1)}, \boldsymbol{k}^{(h,n+1,2)}$ as in Eq. (4).
   b. $(h = \chi)$ $\mathcal{B}_{2\text{-}2}$ computes $\boldsymbol{k}^{*(\chi,0)}, \ldots, \boldsymbol{k}^{*(\chi,n)}$ as in Eq. (8), whereas $\boldsymbol{k}^{*(\chi,n+1,1)}, \boldsymbol{k}^{*(\chi,n+1,2)}$ as in Eq. (4)
   c. $(h > \chi)$ $\mathcal{B}_{2\text{-}2}$ computes $\boldsymbol{k}^{*(h,0)}, \ldots, \boldsymbol{k}^{*(h,n)}, \boldsymbol{k}^{*(h,n+1,1)}, \boldsymbol{k}^{*(h,n+1,2)}$ as in Eq. (4).

   In these computations $\mathcal{B}_{2\text{-}2}$ uses $\{\mathbb{B}_t^*\}_{t\in\{0,2\}}$ and $\widetilde{\mathbb{B}}_1^*$ included within the given Problem 3 instance.
3. For all $t \in [q_{\mathrm{sig}}]$, in reply to the t^{th} signature reveal query of $\mathcal{A}$ for some triple $(\mathsf{MSG}_t, \mathbb{S}_t, \vec{x}^{(t)}) \in \mathbb{M} \times \mathcal{R}_{\mathsf{Z\text{-}ABP}}^{(q)} \times \mathbb{F}_q^n$, $\mathcal{B}_{2\text{-}2}$ hands $\mathcal{A}$ a signature $\mathrm{sig}_t = (\boldsymbol{s}^{*(t,0)}, \ldots, \boldsymbol{s}^{*(t,m_t+1)})$ whose components are computed as in Eq. (5) using $\{\mathbb{B}_t^*\}_{t\in\{0,2\}}$ and $\widetilde{\mathbb{B}}_1^*$ of the given Problem 3 instance.
4. When $\mathcal{A}$ outputs a forgery sig on some message $\mathsf{MSG} \in \mathbb{M}$ under some signing policy ASP $\mathbb{S} = (\mathbb{U} = \{(\vec{y}^{(j)}, \vec{z}^{(j)})\}_{j\in[m]} \subset (\mathbb{F}_q^\ell)^2, \rho : [m] \to [n]) \in \mathcal{R}_{\mathsf{Z\text{-}ABP}}^{(q)}$, $\mathcal{B}_{2\text{-}2}$ computes the verification-text $(\boldsymbol{c}^{(0)}, \ldots, \boldsymbol{c}^{(m+1)})$ as

$$\begin{aligned}
\boldsymbol{c}^{(0)} &= (-u - u_\ell, -\widetilde{u}_\ell, 0, \eta_0)_{\mathbb{B}_0},\\
\boldsymbol{c}^{(j)} &= (\mu_j(\rho(j), -1), (s'_j, s_j), (\widetilde{s}'_j, \widetilde{s}_j), \vec{0}^2,\\
&\qquad (\widetilde{s}'_j, \widetilde{s}_j)\boldsymbol{Z}^{(\rho(j))}, \vec{0}^2, \vec{\eta}^{\dagger(j)})_{\mathbb{B}_1} +\\
&\qquad \sum_{\nu\in[2]} r_\nu^{\dagger(j)} \boldsymbol{e}^{(1,\nu,\widehat{\beta})} \text{ for } j \in [m],\\
\boldsymbol{c}^{(m+1)} &= ((u - \kappa \mathsf{H}_{\mathsf{hk}}^{(\lambda,\mathrm{poly})}(\mathsf{MSG}\|\mathbb{S}), \kappa), \vec{r}^{(m+1)},\\
&\qquad \vec{0}^2, \vec{\eta}^{(m+1)})_{\mathbb{B}_2},
\end{aligned}$$

   where $\vec{u} = (u_1, \ldots, u_\ell), \vec{\widetilde{u}} = (\widetilde{u}_1, \ldots, \widetilde{u}_\ell) \xleftarrow{\cup} \mathbb{F}_q^\ell$, $s_j = \vec{u} \cdot \vec{y}^{(j)}$, $s'_j = \vec{u} \cdot \vec{z}^{(j)}$, $\widetilde{s}_j = \vec{\widetilde{u}} \cdot \vec{y}^{(j)}$, $\widetilde{s}'_j = \vec{\widetilde{u}} \cdot \vec{z}^{(j)}$ for $j \in [m]$, $u, \kappa, \{\mu_j\}_{j\in[m]}, \eta_0 \xleftarrow{\cup} \mathbb{F}_q$, $\{\vec{r}^{\dagger(j)}\}_{j\in[m]}, \vec{r}^{(m+1)}, \{\vec{\eta}^{\dagger(j)}\}_{j\in[m]}, \vec{\eta}^{(m+1)} \xleftarrow{\cup} \mathbb{F}_q^2$, $\boldsymbol{Z}^{(\iota)} \in \{\boldsymbol{Z} \in \mathsf{GL}(2, \mathbb{F}_q) \mid \vec{e}^{(2,2)} = (1, x_\iota^{(\chi)})(\boldsymbol{Z}^{-1})^\top\}$ for $\iota \in [n]$, and $\{\mathbb{B}_t\}_{t\in[0,2]}, \{\boldsymbol{e}^{(1,\nu,\widehat{\beta})}\}_{\nu\in[2]}$ are taken from the given Problem 3 instance. $\mathcal{B}_{2\text{-}2}$ then verifies the validity of the forged signature outputted by $\mathcal{A}$ using the above verification-text, and outputs 1 if the verification succeeds, and 0 otherwise.

It is straightforward to verify that the distribution of $\mathcal{A}$'s view simulated by $\mathcal{B}_{2\text{-}2}$ given $\chi \in [q_{\mathrm{key}}]$ and a Problem 3 instance $\varrho_{\widehat{\beta}}^{\mathsf{P3}}$ for $\widehat{\beta} \in \{0,1\}$ coincides with that in $\mathsf{Hyb}_{2\text{-}\chi\text{-}1}$ if $\widehat{\beta} = 1$ except when any one of $\{\theta_\nu\}_{\nu\in[2]}$ is 0, i.e., except with probability $2/q$. Similarly, the view of $\mathcal{A}$ simulated by $\mathcal{B}_{2\text{-}2}$ given $\chi \in [q_{\mathrm{key}}]$ and a Problem 3 instance $\varrho_{\widehat{\beta}}^{\mathsf{P3}}$ for $\widehat{\beta} \in \{0,1\}$ coincides with that in $\mathsf{Hyb}_{2\text{-}\chi\text{-}2}$ in case $\widehat{\beta} = 0$. This completes the proof of Lemma 14. ∎

Lemma 15: *Under the* SXDLIN *assumption, we have for any* PPT *adversary* $\mathcal{A}$, *for any security parameter* λ, $\left|\mathsf{Adv}_{\mathcal{A}}^{(2\text{-}\chi\text{-}2)}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{(2\text{-}\chi\text{-}3)}(\lambda)\right| \le \mathsf{negl}(\lambda)$ *for all* $\chi \in [q_{\mathrm{key}}]$, *where* negl *is some negligible function.*

Proof: In order to prove Lemma 15, we consider a sequence of intermediate hybrid experiments, which differ from one another in the construction of the χ^{th} signing key requested by $\mathcal{A}$ to reveal and/or the verification-text used by the challenger $\mathcal{B}$ to verify the validity of the forged signature outputted by $\mathcal{A}$ at the end of the experiment. The first hybrid corresponds to $\mathsf{Hyb}_{2\text{-}\chi\text{-}2}$, while the last one corresponds to $\mathsf{Hyb}_{2\text{-}\chi\text{-}3}$. As usual, we argue that $\mathcal{A}$'s winning probability changes only by a negligible amount in each successive hybrid experiment, thereby proving Lemma 15. The overall structure of the reduction is demonstrated in Fig. 3. The sequence of intermediate hybrid experiments are described below. As earlier, in the description of these hybrids as well a part framed by a box indicates coefficients which are altered in a transition from its previous hybrid.

▷ Sequence of Intermediate Hybrid Experiments between $\mathsf{Hyb}_{\mathbf{2\text{-}\chi\text{-}2}}$ and $\mathsf{Hyb}_{\mathbf{2\text{-}\chi\text{-}3}}$ $(\boldsymbol{\chi \in [q_{\mathrm{key}}]})$

$\mathsf{Hyb}_{\mathbf{2\text{-}\chi\text{-}2\text{-}\alpha\text{-}1}}$ $(\boldsymbol{\chi \in [q_{\mathrm{key}}], \alpha \in [n]})$

$\mathsf{Hyb}_{2\text{-}\chi\text{-}2\text{-}0\text{-}8}$ coincides with $\mathsf{Hyb}_{2\text{-}\chi\text{-}2}$. This experiment is analogous to $\mathsf{Hyb}_{2\text{-}\chi\text{-}2\text{-}(\alpha-1)\text{-}8}$ except that in this experiment, the verification-text used to verify the forged signature outputted by $\mathcal{A}$ on some message $\mathsf{MSG} \in \mathbb{M}$ under some signing policy predicate $R_{\mathsf{Z\text{-}ABP}}^{(q)}(f, \cdot) : \mathbb{F}_q^n \to \{0,1\} \in \mathcal{R}_{\mathsf{Z\text{-}ABP}}^{(q)}$ having ASP-representation $\mathbb{S} = (\mathbb{U} = \{(\vec{y}^{(j)}, \vec{z}^{(j)})\}_{j\in[m]} \subset (\mathbb{F}_q^\ell)^2, \rho : [m] \to [n])$ is generated as $(\boldsymbol{c}^{(0)}, \ldots, \boldsymbol{c}^{(m+1)})$ such that $\boldsymbol{c}^{(0)}, \boldsymbol{c}^{(m+1)}$ have the same form as in Eq. (7) and

$$\begin{aligned}
\boldsymbol{c}^{(j)} = (\mu_j(\rho(j), -1), (s'_j, s_j), (\widetilde{s}'_j, \widetilde{s}_j), \boxed{(\widetilde{s}'_j, \widetilde{s}_j)},\\
(\widetilde{s}'_j, \widetilde{s}_j)\boldsymbol{Z}^{(\rho(j))}, \vec{0}^2, \vec{\eta}^{(j)})_{\mathbb{B}_1} \text{ for } j \in [m],
\end{aligned} \tag{17}$$

where all the variables are generated as in $\mathsf{Hyb}_{2\text{-}\chi\text{-}2\text{-}(\alpha-1)\text{-}8}$.

$\mathsf{Hyb}_{\mathbf{2\text{-}\chi\text{-}2\text{-}\alpha\text{-}2}}$ $(\boldsymbol{\chi \in [q_{\mathrm{key}}], \alpha \in [n]})$

This experiment is similar to $\mathsf{Hyb}_{2\text{-}\chi\text{-}2\text{-}\alpha\text{-}1}$ with the only exception that in this experiment, the χ^{th} signing key for signing attribute string $\vec{x}^{(\chi)} \in \mathbb{F}_q^n$ requested by $\mathcal{A}$ to reveal is generated as $\mathsf{SK}(\vec{x}^{(\chi)}) = (\boldsymbol{k}^{*(\chi,0)}, \ldots, \boldsymbol{k}^{*(\chi,n)}, \boldsymbol{k}^{*(\chi,n+1,1)}, \boldsymbol{k}^{*(\chi,n+1,2)})$ such that $\boldsymbol{k}^{*(\chi,0)}, \boldsymbol{k}^{*(\chi,\alpha+1)}, \ldots, \boldsymbol{k}^{*(\chi,n)}$ are given by Eq. (8), $\boldsymbol{k}^{*(\chi,1)}, \ldots, \boldsymbol{k}^{*(\chi,\alpha-1)}$ are given by Eq. (10), $\boldsymbol{k}^{*(\chi,n+1,1)}$, $\boldsymbol{k}^{*(\chi,n+1,2)}$ are given by Eq. (4), and

$$\begin{aligned}
\boldsymbol{k}^{*(\chi,\alpha)} = (\sigma_{\chi,\alpha}(1,\alpha), \omega_\chi(1, x_\alpha^{(\chi)}),\\
\boxed{(\widetilde{\omega}_\chi - \mathfrak{I}_{\chi,\alpha,1}), (\widetilde{\omega}_\chi - \mathfrak{I}_{\chi,\alpha,2})x_\alpha^{(\chi)}, \mathfrak{I}_{\chi,\alpha,1}, \mathfrak{I}_{\chi,\alpha,2}x_\alpha^{(\chi)}},\\
\vec{0}^2, \vec{\varphi}^{(\chi,\alpha)}, \vec{0}^2)_{\mathbb{B}_1^*},
\end{aligned} \tag{18}$$

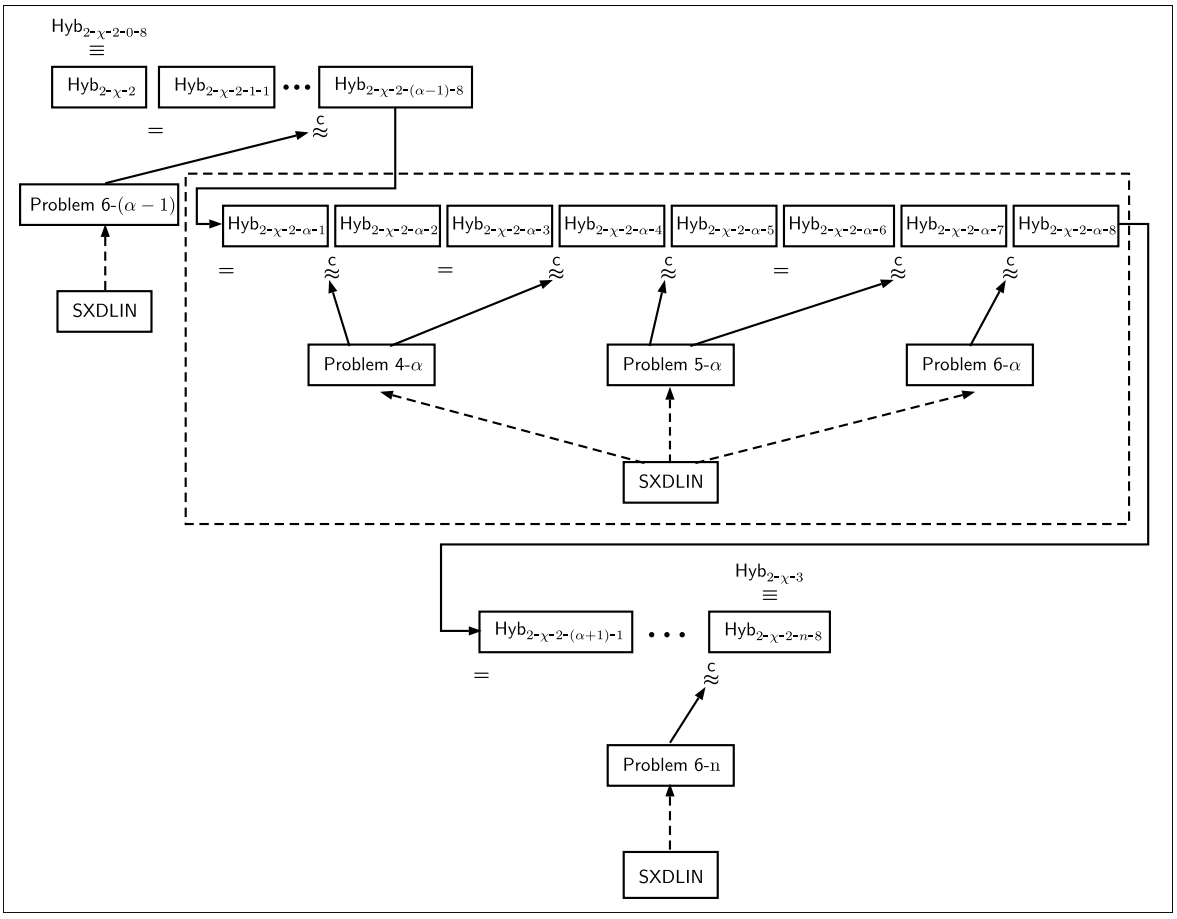


Fig. 3 Structure of the hybrid reduction for the proof of Lemma 15.

where $\{\mathfrak{I}_{\chi,\alpha,\nu}\}_{\nu\in[2]} \xleftarrow{\cup} \mathbb{F}_q$ and all the other variables are formed as in $\mathsf{Hyb}_{2\text{-}\chi\text{-}2\text{-}\alpha\text{-}1}$.

$\mathsf{Hyb}_{2\text{-}\chi\text{-}2\text{-}\alpha\text{-}3}$ ($\chi \in [q_{\mathrm{key}}], \alpha \in [n]$)

This experiment is similar to $\mathsf{Hyb}_{2\text{-}\chi\text{-}2\text{-}\alpha\text{-}2}$ with the only exception that in this experiment, the χ^{th} signing key for signing attribute string $\vec{x}^{(\chi)} \in \mathbb{F}_q^n$ requested by $\mathcal{A}$ to reveal is generated as $\mathsf{SK}(\vec{x}^{(\chi)}) = (\boldsymbol{k}^{*(\chi,0)}, \ldots, \boldsymbol{k}^{*(\chi,n)}, \boldsymbol{k}^{*(\chi,n+1,1)}, \boldsymbol{k}^{*(\chi,n+1,2)})$ such that $\boldsymbol{k}^{*(\chi,0)}, \boldsymbol{k}^{*(\chi,\alpha+1)}, \ldots, \boldsymbol{k}^{*(\chi,n)}$ are given by Eq. (8), $\boldsymbol{k}^{*(\chi,1)}, \ldots, \boldsymbol{k}^{*(\chi,\alpha-1)}$ are given by Eq. (10), $\boldsymbol{k}^{*(\chi,n+1,1)}$, $\boldsymbol{k}^{*(\chi,n+1,2)}$ are given by Eq. (4), and

$$\boldsymbol{k}^{*(\chi,\alpha)} = (\sigma_{\chi,\alpha}(1,\alpha), \omega_\chi(1, x_\alpha^{(\chi)}), \boxed{\mathfrak{I}_{\chi,\alpha,1}, \mathfrak{I}_{\chi,\alpha,2} x_\alpha^{(\chi)}, (\widetilde{\omega}_\chi - \mathfrak{I}_{\chi,\alpha,1}), (\widetilde{\omega}_\chi - \mathfrak{I}_{\chi,\alpha,2}) x_\alpha^{(\chi)}}, \vec{0}^2, \vec{\varphi}^{(\chi,\alpha)}, \vec{0}^2)_{\mathbb{B}_1^*}, \tag{19}$$

where all the variables are generated as in $\mathsf{Hyb}_{2\text{-}\chi\text{-}2\text{-}\alpha\text{-}2}$.

$\mathsf{Hyb}_{2\text{-}\chi\text{-}2\text{-}\alpha\text{-}4}$ ($\chi \in [q_{\mathrm{key}}], \alpha \in [n]$)

This experiment is similar to $\mathsf{Hyb}_{2\text{-}\chi\text{-}2\text{-}\alpha\text{-}3}$ with the only exception that in this experiment, the χ^{th} signing key for signing attribute string $\vec{x}^{(\chi)} \in \mathbb{F}_q^n$ requested by $\mathcal{A}$ to reveal is generated as $\mathsf{SK}(\vec{x}^{(\chi)}) = (\boldsymbol{k}^{*(\chi,0)}, \ldots, \boldsymbol{k}^{*(\chi,n)}, \boldsymbol{k}^{*(\chi,n+1,1)}, \boldsymbol{k}^{*(\chi,n+1,2)})$ such that $\boldsymbol{k}^{*(\chi,0)}, \boldsymbol{k}^{*(\chi,\alpha+1)}, \ldots, \boldsymbol{k}^{*(\chi,n)}$ are given by Eq. (8), $\boldsymbol{k}^{*(\chi,1)}, \ldots, \boldsymbol{k}^{*(\chi,\alpha-1)}$ are given by Eq. (10), $\boldsymbol{k}^{*(\chi,n+1,1)}$, $\boldsymbol{k}^{*(\chi,n+1,2)}$ are given by Eq. (4), and

$$\boldsymbol{k}^{*(\chi,\alpha)} = (\sigma_{\chi,\alpha}(1,\alpha), \omega_\chi(1, x_\alpha^{(\chi)}), \boxed{\vec{0}^2, \widetilde{\omega}_\chi(1, x_\alpha^{(\chi)})}, \vec{0}^2, \vec{\varphi}^{(\chi,\alpha)}, \vec{0}^2)_{\mathbb{B}_1^*}, \tag{20}$$

where all the variables are generated as in $\mathsf{Hyb}_{2\text{-}\chi\text{-}2\text{-}\alpha\text{-}3}$.

$\mathsf{Hyb}_{2\text{-}\chi\text{-}2\text{-}\alpha\text{-}5}$ ($\chi \in [q_{\mathrm{key}}], \alpha \in [n]$

This experiment is similar to $\mathsf{Hyb}_{2\text{-}\chi\text{-}2\text{-}\alpha\text{-}4}$ except that in this experiment, the verification-text used to verify the forged signature outputted by $\mathcal{A}$ on some message $\mathsf{MSG} \in \mathbb{M}$ under some signing policy predicate $R_{\mathsf{Z\text{-}ABP}}^{(q)}(f,\cdot) : \mathbb{F}_q^n \to \{0,1\} \in \mathcal{R}_{\mathsf{Z\text{-}ABP}}^{(q)}$ having ASP-representation $\mathbb{S} = (\mathbb{U} = \{(\vec{y}^{(j)}, \vec{z}^{(j)})\}_{j\in[m]} \subset (\mathbb{F}_q^\ell)^2, \rho : [m] \to [n])$ is generated as $(\boldsymbol{c}^{(0)}, \ldots, \boldsymbol{c}^{(m+1)})$ such that $\boldsymbol{c}^{(0)}$, $\boldsymbol{c}^{(m+1)}$ have the same form

as in Eq. (7), $\{\boldsymbol{c}^{(j)} \mid j \in [m] \wedge \rho(j) = \alpha\}$ are given by Eq. (17), and

$$\boldsymbol{c}^{(j)} = (\mu_j(\rho(j), -1), (s'_j, s_j), (\widetilde{s}'_j, \widetilde{s}_j), \boxed{\vec{\Upsilon}^{(j)}}, (\widetilde{s}'_j, \widetilde{s}_j)\boldsymbol{Z}^{(\rho(j))}, \vec{0}^2, \vec{\eta}^{(j)})_{\mathbb{B}_1} \text{ for } j \in [m] \wedge \rho(j) \neq \alpha, \tag{21}$$

where $\{\vec{\Upsilon}^{(j)} \mid j \in [m] \wedge \rho(j) \neq \alpha\} \xleftarrow{\cup} \mathbb{F}_q^2$ and all the other variables are generated as in $\mathsf{Hyb}_{2\text{-}\chi\text{-}2\text{-}\alpha\text{-}4}$.

$\mathsf{Hyb}_{2\text{-}\chi\text{-}2\text{-}\alpha\text{-}6}$ ($\chi \in [q_{\mathrm{key}}], \alpha \in [n]$)

This experiment is similar to $\mathsf{Hyb}_{2\text{-}\chi\text{-}2\text{-}\alpha\text{-}5}$ with the only exception that in this experiment, the χ^{th} signing key for signing attribute string $\vec{x}^{(\chi)} \in \mathbb{F}_q^n$ requested by $\mathcal{A}$ to reveal is generated as $\mathsf{sk}(\vec{x}^{(\chi)}) = (\boldsymbol{k}^{*(\chi,0)}, \ldots, \boldsymbol{k}^{*(\chi,n)}, \boldsymbol{k}^{*(\chi,n+1,1)}, \boldsymbol{k}^{*(\chi,n+1,2)})$ such that $\boldsymbol{k}^{*(\chi,0)}, \boldsymbol{k}^{*(\chi,\alpha+1)}, \ldots, \boldsymbol{k}^{*(\chi,n)}$ are given by Eq. (8), $\boldsymbol{k}^{*(\chi,1)}, \ldots, \boldsymbol{k}^{*(\chi,\alpha-1)}$ are given by Eq. (10), $\boldsymbol{k}^{*(\chi,n+1,1)}$, $\boldsymbol{k}^{*(\chi,n+1,2)}$ are given by Eq. (4), and

$$\boldsymbol{k}^{*(\chi,\alpha)} = (\sigma_{\chi,\alpha}(1,\alpha), \omega_\chi(1, x_\alpha^{(\chi)}), \vec{0}^2, \boxed{\overline{\omega}_\chi(1, x_\alpha^{(\chi)}), (0, \widetilde{\omega}_\chi)}, \vec{\varphi}^{(\chi,\alpha)}, \vec{0}^2)_{\mathbb{B}_1^*}, \tag{22}$$

while the verification-text used to verify the forged signature outputted by $\mathcal{A}$ on some message $\mathsf{msg} \in \mathbb{M}$ under some signing policy predicate $R^{(q)}_{\mathsf{Z\text{-}ABP}}(f,\cdot) : \mathbb{F}_q^n \to \{0,1\} \in \mathcal{R}^{(q)}_{\mathsf{Z\text{-}ABP}}$ having ASP-representation $\mathbb{S} = (\mathbb{U} = \{(\vec{y}^{(j)}, \vec{z}^{(j)})\}_{j\in[m]} \subset (\mathbb{F}_q^\ell)^2, \rho : [m] \to [n])$ is generated as $(\boldsymbol{c}^{(0)}, \ldots, \boldsymbol{c}^{(m+1)})$ such that $\boldsymbol{c}^{(0)}$, $\boldsymbol{c}^{(m+1)}$ have the same form as in Eq. (7), $\{\boldsymbol{c}^{(j)} \mid j \in [m] \wedge \rho(j) \neq \alpha\}$ are given by Eq. (21), and $\{\boldsymbol{c}^{(j)} \mid j \in [m] \wedge \rho(j) = \alpha\}$ are given by Eq. (9), where $\overline{\omega}_\chi \xleftarrow{\cup} \mathbb{F}_q$ and all the other variables are generated as in $\mathsf{Hyb}_{2\text{-}\chi\text{-}2\text{-}\alpha\text{-}5}$.

$\mathsf{Hyb}_{2\text{-}\chi\text{-}2\text{-}\alpha\text{-}7}$ ($\chi \in [q_{\mathrm{key}}], \alpha \in [n]$)

This experiment is similar to $\mathsf{Hyb}_{2\text{-}\chi\text{-}2\text{-}\alpha\text{-}6}$ except that in this experiment, the verification-text used to verify the forged signature outputted by $\mathcal{A}$ on some message $\mathsf{msg} \in \mathbb{M}$ under some signing policy predicate $R^{(q)}_{\mathsf{Z\text{-}ABP}}(f,\cdot) : \mathbb{F}_q^n \to \{0,1\} \in \mathcal{R}^{(q)}_{\mathsf{Z\text{-}ABP}}$ having ASP-representation $\mathbb{S} = (\mathbb{U} = \{(\vec{y}^{(j)}, \vec{z}^{(j)})\}_{j\in[m]} \subset (\mathbb{F}_q^\ell)^2, \rho : [m] \to [n])$ is generated as $(\boldsymbol{c}^{(0)}, \ldots, \boldsymbol{c}^{(m+1)})$ such that $\boldsymbol{c}^{(0)}$, $\boldsymbol{c}^{(m+1)}$ have the same form as in Eq. (7), and $\boldsymbol{c}^{(1)}, \ldots, \boldsymbol{c}^{(m)}$ are given by Eq. (9), where all the variables are generated as in $\mathsf{Hyb}_{2\text{-}\chi\text{-}2\text{-}\alpha\text{-}6}$.

$\mathsf{Hyb}_{2\text{-}\chi\text{-}2\text{-}\alpha\text{-}8}$ ($\chi \in [q_{\mathrm{key}}], \alpha \in [n]$)

This experiment is similar to $\mathsf{Hyb}_{2\text{-}\chi\text{-}2\text{-}\alpha\text{-}7}$ with the only exception that in this experiment, the χ^{th} signing key for signing attribute string $\vec{x}^{(\chi)} \in \mathbb{F}_q^n$ requested by $\mathcal{A}$ to reveal is generated as $\mathsf{sk}(\vec{x}^{(\chi)}) = (\boldsymbol{k}^{*(\chi,0)}, \ldots, \boldsymbol{k}^{*(\chi,n)}, \boldsymbol{k}^{*(\chi,n+1,1)}, \boldsymbol{k}^{*(\chi,n+1,2)})$ such that $\boldsymbol{k}^{*(\chi,0)}, \boldsymbol{k}^{*(\chi,\alpha+1)}, \ldots, \boldsymbol{k}^{*(\chi,n)}$ are given by Eq. (8), $\boldsymbol{k}^{*(\chi,1)}, \ldots, \boldsymbol{k}^{*(\chi,\alpha)}$ are given by Eq. (10), and $\boldsymbol{k}^{*(\chi,n+1,1)}$, $\boldsymbol{k}^{*(\chi,n+1,2)}$ are given by Eq. (4), where all the variables are generated as in $\mathsf{Hyb}_{2\text{-}\chi\text{-}2\text{-}\alpha\text{-}7}$. Observe that $\mathsf{Hyb}_{2\text{-}\chi\text{-}2\text{-}n\text{-}8}$ coincides with $\mathsf{Hyb}_{2\text{-}\chi\text{-}3}$.

▷ Analysis

As earlier, let us denote by $\mathsf{Adv}^{(i)}_{\mathcal{A}}(\lambda)$ the probability that $\mathcal{A}$ wins in Hyb_i for $i \in \{2\text{-}\chi\text{-}2\text{-}\alpha\text{-}k\}_{\chi\in[q_{\mathrm{key}}],\alpha\in[n],k\in[8]}$. By definition of these hybrids, we clearly have $\mathsf{Adv}^{(2\text{-}\chi\text{-}2)}_{\mathcal{A}}(\lambda) \equiv \mathsf{Adv}^{(2\text{-}\chi\text{-}2\text{-}0\text{-}8)}_{\mathcal{A}}(\lambda)$ and $\mathsf{Adv}^{(2\text{-}\chi\text{-}3)}_{\mathcal{A}}(\lambda) \equiv \mathsf{Adv}^{(2\text{-}\chi\text{-}2\text{-}n\text{-}8)}_{\mathcal{A}}(\lambda)$ for all $\chi \in [q_{\mathrm{key}}]$. Hence, we have

$$\left|\mathsf{Adv}^{(2\text{-}\chi\text{-}2)}_{\mathcal{A}}(\lambda) - \mathsf{Adv}^{(2\text{-}\chi\text{-}3)}_{\mathcal{A}}(\lambda)\right| \leq \sum_{\alpha\in[n]} \left[\left|\mathsf{Adv}^{(2\text{-}\chi\text{-}2\text{-}(\alpha-1)\text{-}8)}_{\mathcal{A}}(\lambda) - \mathsf{Adv}^{(2\text{-}\chi\text{-}2\text{-}\alpha\text{-}1)}_{\mathcal{A}}(\lambda)\right| + \sum_{k\in[7]} \left|\mathsf{Adv}^{(2\text{-}\chi\text{-}2\text{-}\alpha\text{-}k)}_{\mathcal{A}}(\lambda) - \mathsf{Adv}^{(2\text{-}\chi\text{-}2\text{-}\alpha\text{-}(k+1))}_{\mathcal{A}}(\lambda)\right|\right] \text{ for all } \chi \in [q_{\mathrm{key}}]. \tag{23}$$

Then Lemma 15 follows from Claims 1–8 presented below, the proofs of which are essentially analogous to those of Lemmas 49–56 in [42] respectively, in conjunction with Lemmas 6–8 of Sect. 2.4. ∎

Claim 1: *For any stateful probabilistic adversary $\mathcal{A}$, for any security parameter λ, $\mathsf{Adv}^{(2\text{-}\chi\text{-}2\text{-}(\alpha-1)\text{-}8)}_{\mathcal{A}}(\lambda) = \mathsf{Adv}^{(2\text{-}\chi\text{-}2\text{-}\alpha\text{-}1)}_{\mathcal{A}}(\lambda)$ for all $\chi \in [q_{key}], \alpha \in [n]$.*

Claim 2: *For any stateful probabilistic adversary $\mathcal{A}$, there exists a probabilistic algorithm $\mathcal{B}_{2\text{-}3\text{-}1}$, whose running time is essentially the same as that of $\mathcal{A}$, such that for any security parameter λ, $\left|\mathsf{Adv}^{(2\text{-}\chi\text{-}2\text{-}\alpha\text{-}1)}_{\mathcal{A}}(\lambda) - \mathsf{Adv}^{(2\text{-}\chi\text{-}2\text{-}\alpha\text{-}2)}_{\mathcal{A}}(\lambda)\right| \leq \mathsf{Adv}^{\mathsf{P4}\text{-}\alpha}_{\mathcal{B}_{2\text{-}\chi\text{-}3\text{-}\alpha\text{-}1}}(\lambda)$ for all $\chi \in [q_{key}], \alpha \in [n]$, where $\mathcal{B}_{2\text{-}\chi\text{-}3\text{-}\alpha\text{-}1}(\cdot) = \mathcal{B}_{2\text{-}3\text{-}1}(\chi,\alpha,\cdot)$ for any $\chi, \alpha \in \mathbb{N}$.*

Claim 3: *For any stateful probabilistic adversary $\mathcal{A}$, for any security parameter λ, $\mathsf{Adv}^{(2\text{-}\chi\text{-}2\text{-}\alpha\text{-}2)}_{\mathcal{A}}(\lambda) = \mathsf{Adv}^{(2\text{-}\chi\text{-}2\text{-}\alpha\text{-}3)}_{\mathcal{A}}(\lambda)$ for all $\chi \in [q_{key}], \alpha \in [n]$.*

Claim 4: *For any stateful probabilistic adversary $\mathcal{A}$, there exists a probabilistic algorithm $\mathcal{B}_{2\text{-}3\text{-}2}$, whose running time is essentially the same as that of $\mathcal{A}$, such that for any security parameter λ, $\left|\mathsf{Adv}^{(2\text{-}\chi\text{-}2\text{-}\alpha\text{-}3)}_{\mathcal{A}}(\lambda) - \mathsf{Adv}^{(2\text{-}\chi\text{-}2\text{-}\alpha\text{-}4)}_{\mathcal{A}}(\lambda)\right| \leq \mathsf{Adv}^{\mathsf{P4}\text{-}\alpha}_{\mathcal{B}_{2\text{-}\chi\text{-}3\text{-}\alpha\text{-}2}}(\lambda)$ for all $\chi \in [q_{key}], \alpha \in [n]$, where $\mathcal{B}_{2\text{-}\chi\text{-}3\text{-}\alpha\text{-}2}(\cdot) = \mathcal{B}_{2\text{-}3\text{-}2}(\chi,\alpha,\cdot)$ for any $\chi, \alpha \in \mathbb{N}$.*

Claim 5: *For any stateful probabilistic adversary $\mathcal{A}$, there exists a probabilistic algorithm $\mathcal{B}_{2\text{-}3\text{-}3}$, whose running time is essentially the same as that of $\mathcal{A}$, such that for any security parameter λ, $\left|\mathsf{Adv}^{(2\text{-}\chi\text{-}2\text{-}\alpha\text{-}4)}_{\mathcal{A}}(\lambda) - \mathsf{Adv}^{(2\text{-}\chi\text{-}2\text{-}\alpha\text{-}5)}_{\mathcal{A}}(\lambda)\right| \leq \mathsf{Adv}^{\mathsf{P5}\text{-}\alpha}_{\mathcal{B}_{2\text{-}\chi\text{-}3\text{-}\alpha\text{-}3}}(\lambda)$ for all $\chi \in [q_{key}], \alpha \in [n]$, where $\mathcal{B}_{2\text{-}\chi\text{-}3\text{-}\alpha\text{-}3}(\cdot) = \mathcal{B}_{2\text{-}3\text{-}3}(\chi,\alpha,\cdot)$ for any $\chi, \alpha \in \mathbb{N}$.*

Claim 6: *For any stateful probabilistic adversary $\mathcal{A}$, for any*

security parameter λ, $\mathsf{Adv}_{\mathcal{A}}^{(2\text{-}\chi\text{-}2\text{-}\alpha\text{-}5)}(\lambda) = \mathsf{Adv}_{\mathcal{A}}^{(2\text{-}\chi\text{-}2\text{-}\alpha\text{-}6)}(\lambda)$ *for all* $\chi \in [q_{key}], \alpha \in [n]$.

Claim 7: *For any stateful probabilistic adversary* $\mathcal{A}$*, there exists a probabilistic algorithm* $\mathcal{B}_{2\text{-}3\text{-}4}$*, whose running time is essentially the same as that of* $\mathcal{A}$*, such that for any security parameter* λ, $\left|\mathsf{Adv}_{\mathcal{A}}^{(2\text{-}\chi\text{-}2\text{-}\alpha\text{-}6)}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{(2\text{-}\chi\text{-}2\text{-}\alpha\text{-}7)}(\lambda)\right| \leq \mathsf{Adv}^{\mathsf{P5}\text{-}\alpha}_{\mathcal{B}_{2\text{-}\chi\text{-}3\text{-}\alpha\text{-}4}}(\lambda)$ *for all* $\chi \in [q_{key}], \alpha \in [n]$*, where* $\mathcal{B}_{2\text{-}\chi\text{-}4\text{-}\alpha\text{-}3}(\cdot) = \mathcal{B}_{2\text{-}3\text{-}4}(\chi, \alpha, \cdot)$ *for any* $\chi, \alpha \in \mathbb{N}$.

Claim 8: *For any stateful probabilistic adversary* $\mathcal{A}$*, there exists a probabilistic algorithm* $\mathcal{B}_{2\text{-}3\text{-}5}$*, whose running time is essentially the same as that of* $\mathcal{A}$*, such that for any security parameter* λ, $\left|\mathsf{Adv}_{\mathcal{A}}^{(2\text{-}\chi\text{-}2\text{-}\alpha\text{-}7)}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{(2\text{-}\chi\text{-}2\text{-}\alpha\text{-}8)}(\lambda)\right| \leq \mathsf{Adv}^{\mathsf{P6}\text{-}\alpha}_{\mathcal{B}_{2\text{-}\chi\text{-}3\text{-}\alpha\text{-}5}}(\lambda)$ *for all* $\chi \in [q_{key}], \alpha \in [n]$*, where* $\mathcal{B}_{2\text{-}\chi\text{-}3\text{-}\alpha\text{-}5}(\cdot) = \mathcal{B}_{2\text{-}3\text{-}5}(\chi, \alpha, \cdot)$ *for any* $\chi, \alpha \in \mathbb{N}$.

Lemma 16: *For any stateful probabilistic adversary* $\mathcal{A}$*, there exists a probabilistic algorithm* $\mathcal{B}_{2\text{-}4}$*, whose running time is essentially the same as that of* $\mathcal{A}$*, such that for any security parameter* λ, $\left|\mathsf{Adv}_{\mathcal{A}}^{(2\text{-}\chi\text{-}3)}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{(2\text{-}\chi\text{-}4)}(\lambda)\right| \leq \mathsf{Adv}^{\mathsf{P7}}_{\mathcal{B}_{2\text{-}\chi\text{-}4}}(\lambda) + 3/q$ *for all* $\chi \in [q_{key}]$*, where* $\mathcal{B}_{2\text{-}\chi\text{-}4}(\cdot) = \mathcal{B}_{2\text{-}4}(\chi, \cdot)$ *for any* $\chi \in \mathbb{N}$.

Proof: In order to prove Lemma 16, we construct below a probabilistic algorithm $\mathcal{B}_{2\text{-}4}$ against Problem 7 using as a blackbox sub-routine a stateful probabilistic adversary $\mathcal{A}$ that distinguishes between $\mathsf{Hyb}_{2\text{-}\chi\text{-}3}$ and $\mathsf{Hyb}_{2\text{-}\chi\text{-}4}$. Suppose $\mathcal{B}_{2\text{-}4}$ is given $\chi \in [q_{\mathrm{key}}]$ together with an instance of Problem 7

$$\varrho^{\mathsf{P7}}_{\widehat{\beta}} = (\mathsf{params}, \{\mathbb{B}_\iota, \mathbb{B}^*_\iota\}_{\iota\in\{0,2\}}, \mathbb{B}_1, \widetilde{\mathbb{B}}^*_1, \{\boldsymbol{e}^{(1,\nu,\widehat{\beta})}\}_{\nu\in[3]}),$$

where

$$\begin{aligned}
&(\mathsf{params}, \{\mathbb{B}_\iota, \mathbb{B}^*_\iota\}_{\iota\in[0,2]}) \xleftarrow{\mathsf{R}} \mathcal{G}_{\mathsf{OB}}(2, (4, 14, 8));\\
&\widetilde{\mathbb{B}}^*_1 = \{\boldsymbol{b}^{*(1,1)}, \ldots, \boldsymbol{b}^{*(1,4)}, \boldsymbol{b}^{*(1,7)}, \boldsymbol{b}^{*(1,8)}, \boldsymbol{b}^{*(1,10)}, \ldots, \boldsymbol{b}^{*(1,14)}\};\\
&\{\theta_\nu\}_{\nu\in[3]} \xleftarrow{\mathsf{U}} \mathbb{F}_q, \{\vec{\gamma}^{(\nu)}\}_{\nu\in[3]} \xleftarrow{\mathsf{U}} \mathbb{F}_q^2;\\
&\left.\begin{aligned}
\boldsymbol{e}^{(1,\nu,0)} &= (\vec{0}^4, \vec{0}^6, \vec{0}^2, \vec{\gamma}^{(\nu)})_{\mathbb{B}_1}\\
\boldsymbol{e}^{(1,\nu,1)} &= (\vec{0}^4, \theta_\nu \vec{e}^{(2,\nu)}, \vec{0}^4, \vec{0}^2, \vec{\gamma}^{(\nu)})_{\mathbb{B}_1}
\end{aligned}\right\} \text{for } \nu \in [2];\\
&\begin{aligned}
\boldsymbol{e}^{(1,3,0)} &= (\vec{0}^4, \vec{0}^6, \vec{0}^2, \vec{\gamma}^{(3)})_{\mathbb{B}_1},\\
\boldsymbol{e}^{(1,3,1)} &= (\vec{0}^4, \vec{0}^4, \theta_3 \vec{e}^{(2,1)}, \vec{0}^2, \vec{\gamma}^{(3)})_{\mathbb{B}_1}.
\end{aligned}
\end{aligned}$$

$\mathcal{B}_{2\text{-}4}$ interacts with $\mathcal{A}$ as follows:

1. At first, $\mathcal{B}_{2\text{-}4}$ sets

$$\begin{aligned}
\widehat{\mathbb{B}}_0 &= \{\boldsymbol{b}^{(0,1)}, \boldsymbol{b}^{(0,4)}\},\\
\widehat{\mathbb{B}}^*_0 &= \{\boldsymbol{b}^{*(0,3)}\},\\
\widehat{\mathbb{B}}_1 &= \{\boldsymbol{b}^{(1,1)}, \ldots, \boldsymbol{b}^{(1,4)}, \boldsymbol{b}^{(1,13)}, \boldsymbol{b}^{(1,14)}\},\\
\widehat{\mathbb{B}}^*_1 &= \{\boldsymbol{b}^{*(1,1)}, \ldots, \boldsymbol{b}^{*(1,4)}, \boldsymbol{b}^{*(1,11)}, \boldsymbol{b}^{*(1,12)}\},\\
\widehat{\mathbb{B}}_2 &= \{\boldsymbol{b}^{(2,1)}, \boldsymbol{b}^{(2,2)}, \boldsymbol{b}^{(2,7)}, \boldsymbol{b}^{(2,8)}\},\\
\widehat{\mathbb{B}}^*_2 &= \{\boldsymbol{b}^{*(2,1)}, \boldsymbol{b}^{*(2,2)}, \boldsymbol{b}^{*(2,5)}, \boldsymbol{b}^{*(2,6)}\},
\end{aligned}$$

using $\{\mathbb{B}_\iota, \mathbb{B}^*_\iota\}_{\iota\in\{0,2\}}$, $\mathbb{B}_1$, and $\widetilde{\mathbb{B}}^*_1$, which are part of the given Problem 7 instance. It also samples a hashing key $\mathsf{hk} \xleftarrow{\mathsf{R}} \mathsf{KGen}()$ for a hash function family $\mathbb{H}$ associated with $\mathcal{G}_{\mathsf{BPG}}$ and the polynomial $\mathsf{poly}(\cdot)$, where $\mathsf{poly}(\lambda)$ represents the length of the bit string formed by concatenating a message belonging to $\mathbb{M}$ and the binary representation of an ASP representing a signing policy predicate in $\mathcal{R}^{(q)}_{\mathsf{Z\text{-}ABP}}$. It provides $\mathcal{A}$ with the public parameters $\mathsf{MPK} = (\mathsf{hk}, \mathsf{params}, \{\widehat{\mathbb{B}}_\iota, \widehat{\mathbb{B}}^*_\iota\}_{\iota\in[0,2]})$.

2. For $h \in [q_{\mathrm{key}}]$, in response to the h^{th} signing key reveal query of $\mathcal{A}$ for some $\vec{x}^{(h)} \in \mathbb{F}_q^n$, $\mathcal{B}_{2\text{-}4}$ gives $\mathcal{A}$ a signing key $\mathsf{SK}(\vec{x}^{(h)}) = (\boldsymbol{k}^{*(h,0)}, \ldots, \boldsymbol{k}^{*(h,n)}, \boldsymbol{k}^{*(h,n+1,1)}, \boldsymbol{k}^{*(h,n+1,2)})$ whose components are generated as follows:
   a. $(h < \chi)$ $\mathcal{B}_{2\text{-}4}$ computes $\boldsymbol{k}^{*(h,0)}$ as in Eq. (12), while $\boldsymbol{k}^{*(h,1)}, \ldots, \boldsymbol{k}^{*(h,n)}, \boldsymbol{k}^{*(h,n+1,1)}, \boldsymbol{k}^{(h,n+1,2)}$ as in Eq. (4).
   b. $(h = \chi)$ $\mathcal{B}_{2\text{-}4}$ computes $\boldsymbol{k}^{*(\chi,0)}$ as in Eq. (8), $\boldsymbol{k}^{*(\chi,1)}, \ldots, \boldsymbol{k}^{*(\chi,n)}$ as in Eq. (10), whereas $\boldsymbol{k}^{*(\chi,n+1,1)}$, $\boldsymbol{k}^{*(\chi,n+1,2)}$ as in Eq. (4)
   c. $(h > \chi)$ $\mathcal{B}_{2\text{-}4}$ computes $\boldsymbol{k}^{*(h,0)}, \ldots, \boldsymbol{k}^{*(h,n)}, \boldsymbol{k}^{*(h,n+1,1)}$, $\boldsymbol{k}^{*(h,n+1,2)}$ as in Eq. (4).

   In these computations $\mathcal{B}_{2\text{-}4}$ uses $\{\mathbb{B}^*_\iota\}_{\iota\in\{0,2\}}$ and $\widetilde{\mathbb{B}}^*_1$ included within the given Problem 7 instance.

3. For all $t \in [q_{\mathrm{sig}}]$, in response to a signature reveal query of $\mathcal{A}$ for some triple $(\mathsf{MSG}_t, \mathbb{S}_t, \vec{x}^{(t)}) \in \mathbb{M} \times \mathcal{R}^{(q)}_{\mathsf{Z\text{-}ABP}} \times \mathbb{F}_q^n$, $\mathcal{B}_{2\text{-}4}$ hands $\mathcal{A}$ a signature $\mathrm{sig}_t = (\boldsymbol{s}^{*(t,0)}, \ldots, \boldsymbol{s}^{*(t,m_t+1)})$ whose components are computed as in Eq. (5) using $\{\mathbb{B}^*_\iota\}_{\iota\in\{0,2\}}$ and $\widetilde{\mathbb{B}}^*_1$ of the given Problem 7 instance.

4. When $\mathcal{A}$ outputs a forgery sig on some message $\mathsf{MSG} \in \mathbb{M}$ under some signing policy $\mathbb{S} = (\mathbb{U} = \{(\vec{y}^{(j)}, \vec{z}^{(j)})\}_{j\in[m]} \subset (\mathbb{F}_q^\ell)^2, \rho : [m] \to [n]) \in \mathcal{R}^{(q)}_{\mathsf{Z\text{-}ABP}}$, $\mathcal{B}_{2\text{-}4}$ computes the verification-text $(\boldsymbol{c}^{(0)}, \ldots, \boldsymbol{c}^{(m+1)})$ as

$$\begin{aligned}
\boldsymbol{c}^{(0)} &= (-u - u_\ell, -\widetilde{u}_\ell, 0, \eta_0)_{\mathbb{B}_0},\\
\boldsymbol{c}^{(j)} &= (\mu_j(\rho(j), -1), (s'_j, s_j), (\widetilde{s}'_j, \widetilde{s}_j), \vec{0}^2,\\
&\qquad (\widetilde{s}'_j, \widetilde{s}_j)\boldsymbol{Z}^{(\rho(j))}, \vec{0}^2, \vec{\eta}^{\dagger(j)})_{\mathbb{B}_1} +\\
&\qquad \sum_{\nu\in[2]} a^{\dagger(j)}_\nu \boldsymbol{e}^{(1,\nu,\widehat{\beta})} + \widetilde{a}^\dagger_j \boldsymbol{e}^{(1,3,\widehat{\beta})} \text{ for } j \in [m],\\
\boldsymbol{c}^{(m+1)} &= ((u - \kappa \mathsf{H}^{(\lambda,\mathsf{poly})}_{\mathsf{hk}}(\mathsf{MSG}\|\mathbb{S}), \kappa), \vec{r}^{(m+1)},\\
&\qquad \vec{0}^2, \vec{\eta}^{(m+1)})_{\mathbb{B}_2},
\end{aligned}$$

where $\vec{u} = (u_1, \ldots, u_\ell), \vec{\widetilde{u}} = (\widetilde{u}_1, \ldots, \widetilde{u}_\ell) \xleftarrow{\mathsf{U}} \mathbb{F}_q^\ell$, $s_j = \vec{u} \cdot \vec{y}^{(j)}$, $s'_j = \vec{u} \cdot \vec{z}^{(j)}$, $\widetilde{s}_j = \vec{\widetilde{u}} \cdot \vec{y}^{(j)}$, $\widetilde{s}'_j = \vec{\widetilde{u}} \cdot \vec{z}^{(j)}$ for $j \in [m]$, $u, \kappa, \{\mu_j\}_{j\in[m]}, \{\widetilde{a}^\dagger_j\}_{j\in[m]}, \eta_0 \xleftarrow{\mathsf{U}} \mathbb{F}_q$, $\{\vec{a}^{\dagger(j)}\}_{j\in[m]}, \vec{r}^{(m+1)}, \{\vec{\eta}^{\dagger(j)}\}_{j\in[m]}, \vec{\eta}^{(m+1)} \xleftarrow{\mathsf{U}} \mathbb{F}_q^2$, $\boldsymbol{Z}^{(\iota)} \in \{\boldsymbol{Z} \in \mathsf{GL}(2, \mathbb{F}_q) \mid \vec{e}^{(2,2)} = (1, x_\iota^{(\chi)})(\boldsymbol{Z}^{-1})^\top\}$ for $\iota \in [n]$,

and $\{\mathbb{B}_\iota\}_{\iota\in[0,2]}, \{\boldsymbol{e}^{(1,\nu,\widehat{\beta})}\}_{\nu\in[3]}$ are taken from the given Problem 7 instance. $\mathcal{B}_{2\text{-}4}$ then verifies the validity of the forged signature outputted by $\mathcal{A}$ using the above verification-text, and outputs 1 if the verification succeeds, and 0 otherwise.

It is straightforward to verify that the distribution of $\mathcal{A}$'s view simulated by $\mathcal{B}_{2\text{-}4}$ given $\chi \in [q_{\mathsf{key}}]$ and a Problem 7 instance $\varrho^{\mathsf{P7}}_{\widehat{\beta}}$ for $\widehat{\beta} \in \{0,1\}$ coincides with that in $\mathsf{Hyb}_{2\text{-}\chi\text{-}3}$ if $\widehat{\beta} = 0$. Similarly, the view of $\mathcal{A}$ simulated by $\mathcal{B}_{2\text{-}4}$ given $\chi \in [q_{\mathsf{key}}]$ and a Problem 7 instance $\varrho^{\mathsf{P7}}_{\widehat{\beta}}$ for $\widehat{\beta} \in \{0,1\}$ coincides with that in $\mathsf{Hyb}_{2\text{-}\chi\text{-}4}$ in case $\widehat{\beta} = 1$ except when any one of $\{\theta_\nu\}_{\nu\in[3]}$ is 0, i.e., except with probability $3/q$. This completes the proof of Lemma 16. ∎

Lemma 17: *For any stateful probabilistic adversary $\mathcal{A}$, for any security parameter λ, $\mathsf{Adv}^{(2\text{-}\chi\text{-}4)}_{\mathcal{A}}(\lambda) = \mathsf{Adv}^{(2\text{-}\chi\text{-}5)}_{\mathcal{A}}(\lambda)$ for all $\chi \in [q_{key}]$.*

Proof: In order to prove Lemma 17, we show that the distribution $(\mathsf{MPK}, \{\mathsf{SK}(\vec{x}^{(h)})\}_{h\in[q_{\mathsf{key}}]}, \{\mathsf{SIG}_t\}_{t\in[q_{\mathsf{sig}}]}, (\boldsymbol{c}^{(0)}, \ldots, \boldsymbol{c}^{(m+1)}))$ in $\mathsf{Hyb}_{2\text{-}\chi\text{-}4}$ and that in $\mathsf{Hyb}_{2\text{-}\chi\text{-}5}$ are equivalent, where $\mathsf{MPK} = (\mathsf{hk}, \mathsf{params}, \{\widehat{\mathbb{B}}_\iota, \widehat{\mathbb{B}}^*_\iota\}_{\iota\in[0,2]})$ is the public parameters given to $\mathcal{A}$, $\mathsf{SK}(\vec{x}^{(h)}) = (\boldsymbol{k}^{*(h,0)}, \ldots, \boldsymbol{k}^{*(h,n)}, \boldsymbol{k}^{*(h,n+1,1)}, \boldsymbol{k}^{*(h,n+1,2)})$ is the answer to the h^{th} signing key reveal query of $\mathcal{A}$, $\mathsf{SIG}_t = (\boldsymbol{s}^{*(t,0)}, \ldots, \boldsymbol{s}^{*(t,m_t+1)})$ is the answer to the t^{th} signature reveal query of $\mathcal{A}$, and $(\boldsymbol{c}^{(0)}, \ldots, \boldsymbol{c}^{(m+1)})$ is the verification-text used to check the forged signature outputted by $\mathcal{A}$ at the end of the experiment. By the definition of these hybrids, it is clear that we only need to consider the joint distribution of $\mathsf{SK}(\vec{x}^{(\chi)})$ and $(\boldsymbol{c}^{(0)}, \ldots, \boldsymbol{c}^{(m+1)})$.

The components $\boldsymbol{k}^{*(\chi,0)}, \ldots, \boldsymbol{k}^{*(\chi,n)}, \boldsymbol{k}^{*(\chi,n+1,1)}, \boldsymbol{k}^{*(\chi,n+1,2)}$ of the signing key $\mathsf{SK}(\vec{x}^{(\chi)})$ in $\mathsf{Hyb}_{2\text{-}\chi\text{-}4}$ can be expressed as

$$\begin{aligned}
\boldsymbol{k}^{*(\chi,0)} &= (\omega_\chi, \widetilde{p}_{\chi,0}, \varphi_{\chi,0}, 0)_{\mathbb{B}^*_0},\\
\boldsymbol{k}^{*(\chi,\iota)} &= (\sigma_{\chi,\iota}(1,\iota), \omega_\chi(1, x^{(\chi)}_\iota), \vec{0}^4, (0, \widetilde{p}_{\chi,\iota}),\\
&\qquad \vec{\varphi}^{(\chi,\iota)}, \vec{0}^2)_{\mathbb{B}^*_1} \text{ for } \iota \in [n],\\
\boldsymbol{k}^{*(\chi,n+1,1)} &= (\omega_\chi(1,0), \vec{0}^2, \vec{\varphi}^{(\chi,n+1,1)}, \vec{0}^2)_{\mathbb{B}^*_2},\\
\boldsymbol{k}^{*(\chi,n+1,2)} &= (\omega_\chi(0,1), \vec{0}^2, \vec{\varphi}^{(\chi,n+1,2)}, \vec{0}^2)_{\mathbb{B}^*_2},
\end{aligned}$$

where $\widetilde{p}_{\chi,0} = \cdots = \widetilde{p}_{\chi,n} = \widetilde{\omega}_\chi, \omega_\chi \xleftarrow{\cup} \mathbb{F}_q\backslash\{0\}$, $\{\sigma_{\chi,\iota}\}_{\iota\in[n]}, \varphi_{\chi,0} \xleftarrow{\cup} \mathbb{F}_q$, and $\{\vec{\varphi}^{(\chi,\iota)}\}_{\iota\in[n]}, \vec{\varphi}^{(\chi,n+1,1)}, \vec{\varphi}^{(\chi,n+1,2)} \xleftarrow{\cup} \mathbb{F}^2_q$. On the other hand, the components $\boldsymbol{c}^{(0)}, \ldots, \boldsymbol{c}^{(m+1)}$ of the verification-text are computed in $\mathsf{Hyb}_{2\text{-}\chi\text{-}4}$ as

$$\begin{aligned}
\boldsymbol{c}^{(0)} &= (-u - u_\ell, \widetilde{q}_0, 0, \eta_0)_{\mathbb{B}_0},\\
\boldsymbol{c}^{(j)} &= (\mu_j(\rho(j), -1), (s'_j, s_j), \vec{a}^{(j)}, \vec{0}^2, (\widetilde{a}_j, \widetilde{q}_j),\\
&\qquad \vec{0}^2, \vec{\eta}^{(j)})_{\mathbb{B}_1} \text{ for } j \in [m],\\
\boldsymbol{c}^{(m+1)} &= (u - \kappa \mathsf{H}^{(\lambda,\mathsf{poly})}_{\mathsf{hk}}(\mathsf{MSG}\|\mathbb{S}), \kappa, \vec{r}^{(m+1)}, \vec{0}^2, \vec{\eta}^{(m+1)})_{\mathbb{B}_2},
\end{aligned}$$

where $\vec{\widetilde{u}} = (\widetilde{u}_1, \ldots, \widetilde{u}_\ell) \xleftarrow{\cup} \mathbb{F}^\ell_q$, $\widetilde{q}_0 = -\widetilde{u}_\ell$, $\widetilde{q}_j = \vec{\widetilde{u}} \cdot (x^{(\chi)}_{\rho(j)} \vec{y}^{(j)} + \vec{z}^{(j)})$ for $j \in [m]$, $\vec{u} = (u_1, \ldots, u_\ell) \xleftarrow{\cup} \mathbb{F}^\ell_q$, $s_j = \vec{u}\cdot\vec{y}^{(j)}$, $s'_j = \vec{u}\cdot\vec{z}^{(j)}$ for $j \in [m]$, $u, \kappa, \{\mu_j\}_{j\in[m]}, \{\widetilde{a}_j\}_{j\in[m]}$, $\eta_0 \xleftarrow{\cup} \mathbb{F}_q$, $\vec{r}^{(m+1)}, \{\vec{a}^{(j)}\}_{j\in[m]}$, $\{\vec{\eta}^{(j)}\}_{j\in[m+1]} \xleftarrow{\cup} \mathbb{F}^2_q$, and $\mathbb{S} = (\mathbb{U} = \{(\vec{y}^{(j)}, \vec{z}^{(j)})\}_{j\in[m]} \subset (\mathbb{F}^\ell_q)^2, \rho : [m] \to [n])$ is the ASP representation of the signing policy in $\mathcal{R}^{(q)}_{\mathsf{Z\text{-}ABP}}$ under which $\mathcal{A}$ has outputted the forged signature.

Observe that the only change that occurs in the joint distribution of $\mathsf{SK}(\vec{x}^{(\chi)})$ and $(\boldsymbol{c}^{(0)}, \ldots, \boldsymbol{c}^{(m+1)})$ in the transition from $\mathsf{Hyb}_{2\text{-}\chi\text{-}4}$ to $\mathsf{Hyb}_{2\text{-}\chi\text{-}5}$ is that in $\mathsf{Hyb}_{2\text{-}\chi\text{-}5}$, $\widetilde{p}_{\chi,0}$ in the expression of $\boldsymbol{k}^{*(\chi,0)}$ transforms to a uniformly and independently (from all the other variables) distributed element of $\mathbb{F}_q$. Clearly, in the joint distribution of $\mathsf{SK}(\vec{x}^{(\chi)})$ and $(\boldsymbol{c}^{(0)}, \ldots, \boldsymbol{c}^{(m+1)})$ in $\mathsf{Hyb}_{2\text{-}\chi\text{-}4}$, the only variables with which $\widetilde{p}_{\chi,0}$ is related are $\{\widetilde{p}_{\chi,\iota}\}_{\iota\in[n]}$ and $\{\widetilde{q}_j\}_{j\in[0,m]}$. Hence, it is enough to consider the joint distribution of these variables. First, we observe the joint distribution of the variables $\widetilde{p}_{\chi,0}\widetilde{q}_0 = -\widetilde{\omega}_\chi\widetilde{u}_\ell$ and $\widetilde{p}_{\chi,\rho(j)}\widetilde{q}_j = \widetilde{\omega}_\chi(\vec{\widetilde{u}} \cdot (x^{(\chi)}_{\rho(j)} \vec{y}^{(j)} + \vec{z}^{(j)}))$ for $j \in [m]$. By the restriction on the signing key reveal queries imposed on $\mathcal{A}$, the ASP $\mathbb{S}$ does not accept the signing attribute vector $\vec{x}^{(\chi)}$. Therefore, we must have $\vec{e}^{(\ell,\ell)} \notin \mathsf{SPAN}\langle x^{(\chi)}_{\rho(j)} \vec{y}^{(j)} + \vec{z}^{(j)} \mid j \in [m]\rangle$. Hence, there must exist a vector $\vec{u}^+ = (u^+_1, \ldots, u^+_\ell) \in \mathbb{F}^\ell_q$ such that $\vec{u}^+ \cdot \vec{e}^{(\ell,\ell)} = u^+_\ell \neq 0$ and $\vec{u}^+ \cdot (x^{(\chi)}_{\rho(j)} \vec{y}^{(j)} + \vec{z}^{(j)}) = 0$ for all $j \in [m]$. Now, any vector $\vec{\widetilde{u}} \xleftarrow{\cup} \mathbb{F}^\ell_q$ can be expressed as $\vec{\widetilde{u}} = \Lambda\vec{u}^+ + \vec{u}^{++}$ for some $\Lambda \xleftarrow{\cup} \mathbb{F}_q$ and $\vec{u}^{++} \xleftarrow{\cup} \mathbb{F}^\ell_q$. Thus, $\widetilde{p}_{\chi,0}\widetilde{q}_0$ and $\{\widetilde{p}_{\chi,\rho(j)}\widetilde{q}_j\}_{j\in[m]}$ can be expressed as $\widetilde{p}_{\chi,0}\widetilde{q}_0 = -\widetilde{\omega}_\chi(\Lambda u^+_\ell + u^{++}_\ell)$ and $\widetilde{p}_{\chi,\rho(j)}\widetilde{q}_j = \widetilde{\omega}_\chi(\vec{u}^{++} \cdot (x_{\rho(j)} \vec{y}^{(j)} + \vec{z}^{(j)}))$ for $j \in [m]$. From this representation, it is clear that $\widetilde{p}_{\chi,0}\widetilde{q}_0$ is uniformly and independently distributed from $\{\widetilde{p}_{\chi,\rho(j)}\widetilde{q}_j\}_{j\in[m]}$ since $\Lambda \xleftarrow{\cup} \mathbb{F}_q$. Given this fact, it readily follows that $\widetilde{p}_{\chi,0}$ is uniformly and independently distributed from $\{\widetilde{p}_{\chi,\iota}\}_{\iota\in[n]}$ and $\{\widetilde{q}_j\}_{j\in[0,m]}$. This completes the proof of Lemma 17. ∎

Lemma 18: *For any stateful probabilistic adversary $\mathcal{A}$, there exists a probabilistic algorithm $\mathcal{B}_{2\text{-}5}$, whose running time is essentially the same as that of $\mathcal{A}$, such that for any security parameter λ, $\left|\mathsf{Adv}^{(2\text{-}\chi\text{-}5)}_{\mathcal{A}}(\lambda) - \mathsf{Adv}^{(2\text{-}\chi\text{-}6)}_{\mathcal{A}}(\lambda)\right| \leq \mathsf{Adv}^{\mathsf{P7}}_{\mathcal{B}_{2\text{-}\chi\text{-}5}}(\lambda) + 3/q$ for all $\chi \in [q_{key}]$, where $\mathcal{B}_{2\text{-}\chi\text{-}5}(\cdot) = \mathcal{B}_{2\text{-}5}(\chi, \cdot)$ for any $\chi \in \mathbb{N}$.*

Proof: Lemma 18 can be proven in a manner similar to that of Lemma 16. ∎

Lemma 19: *Under the* SXDLIN *assumption, we have for any* PPT *adversary $\mathcal{A}$, for any security parameter λ, $\left|\mathsf{Adv}^{(2\text{-}\chi\text{-}6)}_{\mathcal{A}}(\lambda) - \mathsf{Adv}^{(2\text{-}\chi\text{-}7)}_{\mathcal{A}}(\lambda)\right| \leq \mathsf{negl}(\lambda)$ for all $\chi \in [q_{\mathsf{key}}]$, where negl is some negligible function.*

Proof: The proof of Lemma 19 is similar to that of Lemma 15. ∎

Lemma 20: *For any stateful probabilistic adversary $\mathcal{A}$,*

there exists a probabilistic algorithm $\mathcal{B}_{2\text{-}7}$, whose running time is essentially the same as that of $\mathcal{A}$, such that for any security parameter λ, $\left|\mathsf{Adv}^{(2\text{-}\chi\text{-}7)}_{\mathcal{A}}(\lambda) - \mathsf{Adv}^{(2\text{-}\chi\text{-}8)}_{\mathcal{A}}(\lambda)\right| \leq \mathsf{Adv}^{\mathsf{P3}}_{\mathcal{B}_{2\text{-}\chi\text{-}7}}(\lambda) + 2/q$ for all $\chi \in [q_{key}]$, where $\mathcal{B}_{2\text{-}\chi\text{-}7}(\cdot) = \mathcal{B}_{2\text{-}7}(\chi, \cdot)$ for any $\chi \in \mathbb{N}$.

Proof: Lemma 20 can be proven in a manner analogous to that of Lemma 14. ∎

Lemma 21: *For any stateful probabilistic adversary $\mathcal{A}$, there exists a probabilistic algorithm $\mathcal{B}_{2\text{-}8}$, whose running time is essentially the same as that of $\mathcal{A}$, such that for any security parameter λ, $\left|\mathsf{Adv}^{(2\text{-}\chi\text{-}8)}_{\mathcal{A}}(\lambda) - \mathsf{Adv}^{(2\text{-}\chi\text{-}9)}_{\mathcal{A}}(\lambda)\right| \leq \mathsf{Adv}^{\mathsf{P2}}_{\mathcal{B}_{2\text{-}\chi\text{-}8}}(\lambda) + 3/q$ for all $\chi \in [q_{key}]$, where $\mathcal{B}_{2\text{-}\chi\text{-}8}(\cdot) = \mathcal{B}_{2\text{-}8}(\chi, \cdot)$ for any $\chi \in \mathbb{N}$.*

Proof: Lemma 21 can be proven in a manner analogous to that of Lemma 13. ∎

Lemma 22: *For any stateful probabilistic adversary $\mathcal{A}$, for any security parameter λ, $\left|\mathsf{Adv}^{(2\text{-}q_{key}\text{-}9)}_{\mathcal{A}}(\lambda) - \mathsf{Adv}^{(3)}_{\mathcal{A}}(\lambda)\right| \leq 1/q$.*

Proof: In order to prove Lemma 22, we show that the distribution $(\mathsf{MPK}, \{\mathsf{SK}(\vec{x}^{(h)})\}_{h\in[q_{\mathrm{key}}]}, \{\mathsf{SIG}_t\}_{t\in[q_{\mathrm{sig}}]}, (\boldsymbol{c}^{(0)}, \ldots, \boldsymbol{c}^{(m+1)}))$ in $\mathsf{Hyb}_{2\text{-}q_{\mathrm{key}}\text{-}9}$ and that in Hyb_3 are equivalent, where $\mathsf{MPK} = (\mathsf{hk}, \mathsf{params}, \{\widehat{\mathbb{B}}_t, \widehat{\mathbb{B}}^*_t\}_{t\in[0,2]})$ is the public parameters given to $\mathcal{A}$, $\mathsf{SK}(\vec{x}^{(h)}) = (\boldsymbol{k}^{*(h,0)}, \ldots, \boldsymbol{k}^{*(h,n)}, \boldsymbol{k}^{*(h,n+1,1)}, \boldsymbol{k}^{*(h,n+1,2)})$ is the answer to $\mathcal{A}$'s h^{th} signing key reveal query for $h \in [q_{\mathrm{key}}]$, $\mathsf{SIG}_t = (\boldsymbol{s}^{*(t,0)}, \ldots, \boldsymbol{s}^{*(t,m_t+1)})$ is the answer to the t^{th} signature reveal query of $\mathcal{A}$ for $t \in [q_{\mathrm{sig}}]$, and $(\boldsymbol{c}^{(0)}, \ldots, \boldsymbol{c}^{(m+1)})$ is the verification-text used to check the forged signature outputted by $\mathcal{A}$ at the end of the experiment. By the definition of these hybrids, it is clear that we only need to consider the joint distribution of the components $\{\boldsymbol{k}^{*(h,0)}\}_{h\in[q_{\mathrm{key}}]}$, $\{\boldsymbol{s}^{*(t,0)}\}_{t\in[q_{\mathrm{sig}}]}$, and $\boldsymbol{c}^{(0)}$. Let us start with the joint distribution of these components in $\mathsf{Hyb}_{2\text{-}q_{\mathrm{key}}\text{-}9}$. Define new dual orthonormal bases $(\mathbb{D}_0, \mathbb{D}^*_0)$ of $(\mathbb{V}_0, \mathbb{V}^*_0)$ from the original bases $(\mathbb{B}_0, \mathbb{B}^*_0)$ used in $\mathsf{Hyb}_{2\text{-}q_{\mathrm{key}}\text{-}9}$ as follows: Generate $\Lambda \xleftarrow{\mathsf{U}} \mathbb{F}_q \backslash \{0\}$, compute

$$\boldsymbol{d}^{(0,2)} = \Lambda \boldsymbol{b}^{(0,2)}, \boldsymbol{d}^{*(0,2)} = \Lambda^{-1} \boldsymbol{b}^{*(0,2)},$$

and set

$$\mathbb{D}_0 = \{\boldsymbol{b}^{(0,1)}, \boldsymbol{d}^{(0,2)}, \boldsymbol{b}^{(0,3)}, \boldsymbol{b}^{(0,4)}\},$$
$$\mathbb{D}^*_0 = \{\boldsymbol{b}^{*(0,1)}, \boldsymbol{d}^{*(0,2)}, \boldsymbol{b}^{*(0,3)}, \boldsymbol{b}^{*(0,4)}\}.$$

It can be readily observed that the new bases $(\mathbb{D}_0, \mathbb{D}^*_0)$ are indeed dual orthonormal, and are distributed the same as the original bases $(\mathbb{B}_0, \mathbb{B}^*_0)$.

First, notice that for all $t \in [q_{\mathrm{sig}}]$, the forms of $\boldsymbol{s}^{*(t,0)}$ in $\mathsf{Hyb}_{2\text{-}q_{\mathrm{key}}\text{-}9}$ and in Hyb_3 are identical, and since the coefficient of $\boldsymbol{b}^{*(0,2)}$ in the expression of $\boldsymbol{s}^{*(t,0)}$ generated in $\mathsf{Hyb}_{2\text{-}q_{\mathrm{key}}\text{-}9}$ is 0, its form remains unaltered under this change of bases. Now, observe that the components $\boldsymbol{c}^{(0)}$ and $\{\boldsymbol{k}^{*(h,0)}\}_{h\in[q_{\mathrm{key}}]}$ in $\mathsf{Hyb}_{2\text{-}q_{\mathrm{key}}\text{-}9}$ can be expressed over the new bases $(\mathbb{D}_0, \mathbb{D}^*_0)$ as follows:

$$\begin{aligned}
\boldsymbol{c}^{(0)} &= (-u - u_\ell, -\widetilde{u}_\ell, 0, \eta_0)_{\mathbb{B}_0} \\
&= (-u - u_\ell, -\widetilde{u}_\ell \Lambda^{-1}, 0, \eta_0)_{\mathbb{D}_0} \\
&= (-u - u_\ell, v, 0, \eta_0)_{\mathbb{D}_0}, \\
\boldsymbol{k}^{*(h,0)} &= (\omega_h, \Im_h, \varphi_{h,0}, 0)_{\mathbb{B}^*_0} \\
&= (\omega_h, \Im_h \Lambda, \varphi_{h,0}, 0)_{\mathbb{D}^*_0} \\
&= (\omega_h, \Im'_h, \varphi_{h,0}, 0)_{\mathbb{D}^*_0} \text{ for } h \in [q_{\mathrm{key}}],
\end{aligned}$$

where $v = -\widetilde{u}_\ell \Lambda^{-1}$ and $\Im'_h = \Im_h \Lambda$ for $h \in [q_{\mathrm{key}}]$. Clearly, for all $h \in [q_{\mathrm{key}}]$, $\Im'_h$ is uniformly and independently (from all the other variables) distributed since $\Im_h \xleftarrow{\mathsf{U}} \mathbb{F}_q$. Further, v is uniformly and independently (from all the other variables) distributed except when $\widetilde{u}_\ell = 0$ since $\Lambda \xleftarrow{\mathsf{U}} \mathbb{F}_q \backslash \{0\}$. Thus, it follows that $\boldsymbol{c}^{(0)}$ and $\{\boldsymbol{k}^{*(h,0)}\}_{h\in[q_{\mathrm{key}}]}$ generated in $\mathsf{Hyb}_{2\text{-}q_{\mathrm{key}}\text{-}9}$ take the form as in Hyb_3 when expressed over the transformed bases $(\mathbb{D}_0, \mathbb{D}^*_0)$.

Clearly, in the view of $\mathcal{A}$, both the original bases $(\mathbb{B}_0, \mathbb{B}^*_0)$ and the transformed bases $(\mathbb{D}_0, \mathbb{D}^*_0)$ are consistent with the public parameters MPK. Therefore, $\mathsf{Hyb}_{2\text{-}q_{\mathrm{key}}\text{-}9}$ can be conceptually changed to Hyb_3 except when $\widetilde{u}_\ell = 0$, i.e., except with probability $1/q$. This completes the proof of Lemma 22. ∎

Lemma 23: *For any stateful probabilistic adversary $\mathcal{A}$, there exists probabilistic algorithms $\mathcal{B}_3$ and $\mathcal{M}$, whose running times are essentially the same as that of $\mathcal{A}$, such that for any security parameter λ, $\left|\mathsf{Adv}^{(4\text{-}(\pi-1))}_{\mathcal{A}}(\lambda) - \mathsf{Adv}^{(4\text{-}\pi)}_{\mathcal{A}}(\lambda)\right| \leq \mathsf{Adv}^{\mathsf{P8}}_{\mathcal{B}_{3\text{-}\pi}}(\lambda) + \mathsf{Adv}^{\mathsf{H,CR}}_{\mathcal{M}_\pi}(\lambda) + 5/q$ for all $\pi \in [q_{sig}]$, where $\mathcal{B}_{3\text{-}\pi}(\cdot) = \mathcal{B}_3(\pi, \cdot)$ and $\mathcal{M}_\pi(\cdot) = \mathcal{M}(\pi, \cdot)$ for any $\pi \in \mathbb{N}$.*

Proof: The proof of Lemma 23 utilizes the following well-known result:

Lemma 24 (Lemma 3 in [38]): *For any $b \in \mathbb{F}_q$ and $d \in \mathbb{N}$, let $\mathbb{C}_b = \{(\vec{v}, \vec{w}) \in (\mathbb{F}^d_q \backslash \{\vec{0}^d\})^2 \mid \vec{v} \cdot \vec{w} = b\}$. For all $(\vec{v}, \vec{w}) \in \mathbb{C}_b$, for all $(\vec{c}, \vec{k}) \in \mathbb{C}_b$, we have*

$$\begin{aligned}
&\mathsf{Pr}[\vec{v}\boldsymbol{F} = \vec{c} \wedge \vec{w}(\boldsymbol{F}^{-1})^\top = \vec{k}] \\
&= \mathsf{Pr}[\vec{v}(\boldsymbol{F}^{-1})^\top = \vec{c} \wedge \vec{w}\boldsymbol{F} = \vec{k}] = 1/\#\mathbb{C}_b,
\end{aligned}$$

where $\boldsymbol{F} \xleftarrow{U} GL(d, \mathbb{F}_q)$.

In order to prove Lemma 23, we construct below a probabilistic algorithm $\mathcal{B}_3$ against Problem 8 using as a blackbox sub-routine a stateful probabilistic adversary $\mathcal{A}$ that distinguishes between $\mathsf{Hyb}_{4\text{-}(\pi-1)}$ and $\mathsf{Hyb}_{4\text{-}\pi}$. Suppose $\mathcal{B}_3$ is given $\pi \in [q_{\mathrm{sig}}]$ together with an instance of Problem 8

$$\begin{aligned}
\mathcal{O}^{\mathsf{P8}}_{\widehat{\beta}} = (&\mathsf{params}, \{\widetilde{\mathbb{B}}_t, \mathbb{B}^*_t\}_{t\in\{0,2\}}, \mathbb{B}_1, \mathbb{B}^*_1, \boldsymbol{h}^{*(0,\widehat{\beta})}, \boldsymbol{f}^{(0)}, \\
&\{\boldsymbol{h}^{*(2,\nu,\widehat{\beta})}, \boldsymbol{f}^{(2,\nu)}\}_{\nu\in[2]}),
\end{aligned}$$

where

$(\mathsf{params}, \{\mathbb{B}_t, \mathbb{B}_t^*\}_{t\in[0,2]}) \xleftarrow{\mathsf{R}} \mathcal{G}_{\mathsf{OB}}(2, (4, 14, 8))$;
$\widetilde{\mathbb{B}}_0 = \{\boldsymbol{b}^{(0,1)}, \boldsymbol{b}^{(0,3)}, \boldsymbol{b}^{(0,4)}\}$;
$\widetilde{\mathbb{B}}_2 = \{\boldsymbol{b}^{(2,1)}, \boldsymbol{b}^{(2,2)}, \boldsymbol{b}^{(2,5)}, \ldots, \boldsymbol{b}^{(2,8)}\}$;
$\vartheta, \varkappa, \delta, \tau, \xi_0 \xleftarrow{\mathsf{U}} \mathbb{F}_q$, $\{\vec{\xi}^{(\nu)}\}_{\nu\in[2]} \xleftarrow{\mathsf{U}} \mathbb{F}_q^2$, $\boldsymbol{X} \xleftarrow{\mathsf{U}} \mathsf{GL}(2, \mathbb{F}_q)$,
$\boldsymbol{Y} = (\boldsymbol{X}^{-1})^\top$;
$\boldsymbol{h}^{*(0,0)} = (\vartheta, 0, \xi_0, 0)_{\mathbb{B}_0^*}$, $\boldsymbol{h}^{*(0,1)} = (\vartheta, \varkappa, \xi_0, 0)_{\mathbb{B}_0^*}$;
$\boldsymbol{f}^{(0)} = (\delta, \tau, 0, 0)_{\mathbb{B}_0}$;

$$\left.\begin{array}{l} \boldsymbol{h}^{*(2,\nu,0)} = (\vartheta\vec{e}^{(2,\nu)}, \vec{0}^2, \vec{\xi}^{(\nu)}, \vec{0}^2)_{\mathbb{B}_2^*} \\ \boldsymbol{h}^{*(2,\nu,1)} = (\vartheta\vec{e}^{(2,\nu)}, \varkappa\vec{e}^{(2,\nu)}\boldsymbol{X}, \vec{\xi}^{(\nu)}, \vec{0}^2)_{\mathbb{B}_2^*} \\ \boldsymbol{f}^{(2,\nu)} = (\delta\vec{e}^{(2,\nu)}, \tau\vec{e}^{(2,\nu)}\boldsymbol{Y}, \vec{0}^2, \vec{0}^2)_{\mathbb{B}_2} \end{array}\right\} \text{for } \nu \in [2].$$

$\mathcal{B}_3$ interacts with $\mathcal{A}$ as follows:

1. At first, $\mathcal{B}_3$ sets

$$\begin{aligned} \widehat{\mathbb{B}}_0 &= \{\boldsymbol{b}^{(0,1)}, \boldsymbol{b}^{(0,4)}\}, \\ \widehat{\mathbb{B}}_0^* &= \{\boldsymbol{b}^{*(0,3)}\}, \\ \widehat{\mathbb{B}}_1 &= \{\boldsymbol{b}^{(1,1)}, \ldots, \boldsymbol{b}^{(1,4)}, \boldsymbol{b}^{(1,13)}, \boldsymbol{b}^{(1,14)}\}, \\ \widehat{\mathbb{B}}_1^* &= \{\boldsymbol{b}^{*(1,1)}, \ldots, \boldsymbol{b}^{*(1,4)}, \boldsymbol{b}^{*(1,11)}, \boldsymbol{b}^{*(1,12)}\}, \\ \widehat{\mathbb{B}}_2 &= \{\boldsymbol{b}^{(2,1)}, \boldsymbol{b}^{(2,2)}, \boldsymbol{b}^{(2,7)}, \boldsymbol{b}^{(2,8)}\}, \\ \widehat{\mathbb{B}}_2^* &= \{\boldsymbol{b}^{*(2,1)}, \boldsymbol{b}^{*(2,2)}, \boldsymbol{b}^{*(2,5)}, \boldsymbol{b}^{*(2,6)}\}, \end{aligned}$$

   using $\{\widetilde{\mathbb{B}}_t, \mathbb{B}_t^*\}_{t\in\{0,2\}}$, $\mathbb{B}_1$, and $\mathbb{B}_1^*$, which are part of the given Problem 8 instance. It also samples a hashing key $\mathsf{hk} \xleftarrow{\mathsf{R}} \mathsf{KGen}()$ for a hash function family $\mathbb{H}$ associated with $\mathcal{G}_{\mathsf{BPG}}$ and the polynomial $\mathsf{poly}(\cdot)$, where $\mathsf{poly}(\lambda)$ represents the length of the bit string formed by concatenating a message belonging to $\mathbb{M}$ and the binary representation of an ASP representing a signing policy predicate in $\mathcal{R}_{\mathsf{z\text{-}ABP}}^{(q)}$. It provides $\mathcal{A}$ with the public parameters $\mathsf{MPK} = (\mathsf{hk}, \mathsf{params}, \{\widehat{\mathbb{B}}_t, \widehat{\mathbb{B}}_t^*\}_{t\in[0,2]})$.

2. For all $h \in [q_{\mathrm{key}}]$, in response to the h^{th} signing key reveal query of $\mathcal{A}$ for some $\vec{x}^{(h)} \in \mathbb{F}_q^n$, $\mathcal{B}_3$ gives $\mathcal{A}$ a signing key $\mathsf{SK}(\vec{x}^{(h)}) = (\boldsymbol{k}^{*(h,0)}, \ldots, \boldsymbol{k}^{*(h,n)}, \boldsymbol{k}^{*(h,n+1,1)}, \boldsymbol{k}^{*(h,n+1,2)})$, where it generates $\boldsymbol{k}^{*(h,0)}$ as in Eq. (12), while $\boldsymbol{k}^{*(h,1)}, \ldots, \boldsymbol{k}^{*(h,n)}, \boldsymbol{k}^{*(h,n+1,1)}, \boldsymbol{k}^{*(h,n+1,2)}$ as in Eq. (4) using $\{\mathbb{B}_t^*\}_{t\in[0,2]}$ of the given Problem 8 instance.

3. For $t \in [q_{\mathrm{sig}}]$, in response to the t^{th} signature reveal query of $\mathcal{A}$ for some triple $(\mathsf{MSG}_t, \mathbb{S}_t, \vec{x}^{(t)}) \in \mathbb{M} \times \mathcal{R}_{\mathsf{z\text{-}ABP}}^{(q)} \times \mathbb{F}_q^n$, $\mathcal{B}_3$ hands $\mathcal{A}$ a signature $\mathsf{sig}_t = (\boldsymbol{s}^{*(t,0)}, \ldots, \boldsymbol{s}^{*(t,m_t+1)})$ whose components are computed as follows:
   a. $(t < \pi)$ $\mathcal{B}_3$ computes $\boldsymbol{s}^{*(t,0)}, \boldsymbol{s}^{*(t,m_t+1)}$ as in Eq. (14), while $\boldsymbol{s}^{*(t,1)}, \ldots, \boldsymbol{s}^{*(t,m_t)}$ as in Eq. (5) using $\{\mathbb{B}_t^*\}_{t\in[0,2]}$ included within the given Problem 8 instance.
   b. $(t = \pi)$ $\mathcal{B}_3$ computes $\boldsymbol{s}^{*(\pi,1)}, \ldots, \boldsymbol{s}^{*(\pi,M_\pi)}$ as in Eq. (5), while it computes

$$\boldsymbol{s}^{*(\pi,0)} = \boldsymbol{h}^{*(0,\widehat{\beta})},$$

$$\boldsymbol{s}^{*(\pi,m_\pi+1)} = \boldsymbol{h}^{*(2,1,\widehat{\beta})} + \mathsf{H}_{\mathsf{hk}}^{(\lambda,\mathsf{poly})}(\mathsf{MSG}_\pi \| \mathbb{S}_\pi)\boldsymbol{h}^{*(2,2,\widehat{\beta})},$$

   where $\boldsymbol{h}^{*(0,\widehat{\beta})}, \{\boldsymbol{h}^{*(2,\nu,\widehat{\beta})}\}_{\nu\in[2]}$ are taken from the given Problem 8 instance.
   c. $(t > \pi)$ $\mathcal{B}_3$ computes $\boldsymbol{s}^{*(t,0)}, \ldots, \boldsymbol{s}^{*(t,m_t+1)}$ as in Eq. (5) using $\{\mathbb{B}_t^*\}_{t\in[0,2]}$ of the given Problem 8 instance.

4. When $\mathcal{A}$ outputs a forgery sig on some message $\mathsf{MSG} \in \mathbb{M}$ under some signing policy $\mathbb{S} = (\mathbb{U} = \{(\vec{y}^{(j)}, \vec{z}^{(j)})\}_{j\in[m]} \subset (\mathbb{F}_q^\ell)^2, \rho : [m] \to [n]) \in \mathcal{R}_{\mathsf{z\text{-}ABP}}^{(q)}$, $\mathcal{B}_3$ computes the verification-text $(\boldsymbol{c}^{(0)}, \ldots, \boldsymbol{c}^{(m+1)})$ as

$$\begin{aligned} \boldsymbol{c}^{(0)} &= -u_\ell \boldsymbol{b}^{(0,1)} - (\theta_1^\dagger \boldsymbol{f}^{(0)} + \theta_2^\dagger \boldsymbol{b}^{(0,1)}) + \eta_0 \boldsymbol{b}^{(0,4)}, \\ \boldsymbol{c}^{(j)} &= (\mu_j(\rho(j), -1), (s'_j, s_j), (\widetilde{s}'_j, \widetilde{s}_j), \\ &\qquad \vec{0}^2, \vec{r}^{(j)}, \vec{0}^2, \vec{\eta}^{(j)})_{\mathbb{B}_1} \text{ for } j \in [m], \\ \boldsymbol{c}^{(m+1)} &= (\theta_1^\dagger \boldsymbol{f}^{(2,1)} + \theta_2^\dagger \boldsymbol{b}^{(2,1)}) - \\ &\qquad \mathsf{H}_{\mathsf{hk}}^{(\lambda,\mathsf{poly})}(\mathsf{MSG}\|\mathbb{S})(\theta_1^\ddagger \boldsymbol{f}^{(2,1)} + \theta_2^\ddagger \boldsymbol{b}^{(2,1)}) + \\ &\qquad (\theta_1^\ddagger \boldsymbol{f}^{(2,2)} + \theta_2^\ddagger \boldsymbol{b}^{(2,2)}) + \sum_{\nu\in[2]} \eta_\nu^{(m+1)} \boldsymbol{b}^{(2,6+\nu)}, \end{aligned}$$

   where $\vec{u} = (u_1, \ldots, u_\ell), \vec{\widetilde{u}} = (\widetilde{u}_1, \ldots, \widetilde{u}_\ell) \xleftarrow{\mathsf{U}} \mathbb{F}_q^\ell$, $s_j = \vec{u} \cdot \vec{y}^{(j)}$, $s'_j = \vec{u} \cdot \vec{z}^{(j)}$, $\widetilde{s}_j = \vec{\widetilde{u}} \cdot \vec{y}^{(j)}$, $\widetilde{s}'_j = \vec{\widetilde{u}} \cdot \vec{z}^{(j)}$ for $j \in [m]$, $\{\theta_\nu^\dagger, \theta_\nu^\ddagger\}_{\nu\in[2]}, \{\mu_j\}_{j\in[m]}, \eta_0 \xleftarrow{\mathsf{U}} \mathbb{F}_q$, $\{\vec{r}^{(j)}\}_{j\in[m]}, \{\vec{\eta}^{(j)}\}_{j\in[m+1]} \xleftarrow{\mathsf{U}} \mathbb{F}_q^2$, and $\{\widetilde{\mathbb{B}}_t\}_{t\in\{0,2\}}, \mathbb{B}_1, \boldsymbol{f}^{(0)}, \{\boldsymbol{f}^{(2,\nu)}\}_{\nu\in[2]}$ are taken from the given Problem 8 instance. $\mathcal{B}_3$ then verifies the validity of the forged signature outputted by $\mathcal{A}$ using the above verification-text, and outputs 1 if the verification succeeds, and 0 otherwise.

We now argue that the above simulation of $\mathcal{A}$'s view by $\mathcal{B}_3$ given $\pi \in [q_{\mathrm{key}}]$ and a Problem 8 instance $\varrho_{\widehat{\beta}}^{\mathsf{P8}}$ for $\widehat{\beta} \in \{0, 1\}$ is coincides with that in $\mathsf{Hyb}_{4\text{-}(\pi-1)}$ or $\mathsf{Hyb}_{4\text{-}\pi}$ according as $\widehat{\beta} = 0$ or 1. It is immediate that in order to argue this, it is enough to consider the joint distribution of $(\boldsymbol{c}^{(0)}, \ldots, \boldsymbol{c}^{(m+1)})$ and SIG_π, where $(\boldsymbol{c}^{(0)}, \ldots, \boldsymbol{c}^{(m+1)})$ is the verification-text generated by $\mathcal{B}_3$ to verify the forged signature outputted by $\mathcal{A}$ and $\mathsf{SIG}_\pi = (\boldsymbol{s}^{*(\pi,0)}, \ldots, \boldsymbol{s}^{*(\pi,m_\pi+1)})$ is $\mathcal{B}_3$'s response to the π^{th} signature reveal query of $\mathcal{A}$. Also, a part of the verification-text, namely, $(\boldsymbol{c}^{(1)}, \ldots, \boldsymbol{c}^{(m)})$ and a part of the signature SIG_π, namely, $(\boldsymbol{s}^{*(\pi,1)}, \ldots, \boldsymbol{s}^{*(\pi,m_\pi)})$ are clearly identically distributed to those in $\mathsf{Hyb}_{4\text{-}(\pi-1)}$ and in $\mathsf{Hyb}_{4\text{-}\pi}$. Therefore, we only need to consider the joint distribution of $\boldsymbol{c}^{(0)}, \boldsymbol{c}^{(m+1)}, \boldsymbol{s}^{*(\pi,0)}$, and $\boldsymbol{s}^{*(\pi,m_\pi+1)}$.

When $\widehat{\beta} = 0$, it is straightforward to verify that the joint distribution of $\boldsymbol{c}^{(0)}, \boldsymbol{c}^{(m+1)}, \boldsymbol{s}^{*(\pi,0)}$, and $\boldsymbol{s}^{*(\pi,m_\pi+1)}$ coincides with that in $\mathsf{Hyb}_{4\text{-}(\pi-1)}$ except when ϑ or τ used in the given Problem 8 instance is 0, i.e., except with probability $2/q$.

When $\widehat{\beta} = 1$, $\boldsymbol{c}^{(0)}, \boldsymbol{c}^{(m+1)}, \boldsymbol{s}^{*(\pi,0)}$, and $\boldsymbol{s}^{*(\pi,m_\pi+1)}$ simulated by $\mathcal{B}_3$ take the form

$$\begin{aligned} \boldsymbol{c}^{(0)} &= (-u_\ell - (\delta\theta_1^\dagger + \theta_2^\dagger), -\tau\theta_1^\dagger, 0, \eta_0)_{\mathbb{B}_0} \\ &= (-u_\ell - u, v, 0, \eta_0)_{\mathbb{B}_0}, \end{aligned}$$

$$\begin{aligned}
\boldsymbol{c}^{(m+1)} &= ((\delta\theta_1^{\dagger}+\theta_2^{\dagger})-(\delta\theta_1^{\ddagger}+\theta_2^{\ddagger})\mathsf{H}_{\mathsf{hk}}^{(\lambda,\mathsf{poly})}(\mathsf{MSG}\|\mathbb{S}),\\
&\quad \delta\theta_1^{\ddagger}+\theta_2^{\ddagger},(\tau\theta_1^{\dagger}-\tau\theta_1^{\ddagger}\mathsf{H}_{\mathsf{hk}}^{(\lambda,\mathsf{poly})}(\mathsf{MSG}\|\mathbb{S}),\tau\theta_1^{\ddagger})\boldsymbol{Y},\\
&\quad \vec{0}^2,\vec{\eta}^{(m+1)})_{\mathbb{B}_2}\\
&= (u-\kappa\mathsf{H}_{\mathsf{hk}}^{(\lambda,\mathsf{poly})}(\mathsf{MSG}\|\mathbb{S}),\kappa,\vec{r}^{(m+1)},\\
&\quad \vec{0}^2,\vec{\eta}^{(m+1)})_{\mathbb{B}_2},\\
\boldsymbol{s}^{*(\pi,0)} &= (\vartheta,\varkappa,\xi_0,0)_{\mathbb{B}_0^*}\\
&= (\widehat{\omega}_\pi,\zeta_{\pi,0},\widehat{\upsilon}_{\pi,0},0)_{\mathbb{B}_0^*},\\
\boldsymbol{s}^{*(\pi,m_\pi+1)} &= (\vartheta(1,\mathsf{H}_{\mathsf{hk}}^{(\lambda,\mathsf{poly})}(\mathsf{MSG}_\pi\|\mathbb{S}_\pi)),\\
&\quad \varkappa(1,\mathsf{H}_{\mathsf{hk}}^{(\lambda,\mathsf{poly})}(\mathsf{MSG}_\pi\|\mathbb{S}_\pi))\boldsymbol{X},\vec{\xi}^{(1)}+\\
&\quad \mathsf{H}_{\mathsf{hk}}^{(\lambda,\mathsf{poly})}(\mathsf{MSG}_\pi\|\mathbb{S}_\pi)\vec{\xi}^{(2)},\vec{0}^2)_{\mathbb{B}_2^*}\\
&= (\widehat{\omega}_\pi(1,\mathsf{H}_{\mathsf{hk}}^{(\lambda,\mathsf{poly})}(\mathsf{MSG}_\pi\|\mathbb{S}_\pi)),\\
&\quad \vec{\zeta}^{(\pi,m_\pi+1)},\vec{\upsilon}^{(\pi,m_\pi+1)},\vec{0}^2)_{\mathbb{B}_2^*},
\end{aligned}$$

where $u=\delta\theta_1^{\dagger}+\theta_2^{\dagger}$, $v=\tau\theta_1^{\dagger}$, $\kappa=\delta\theta_1^{\ddagger}+\theta_2^{\ddagger}$, $\vec{r}^{(M+1)}=(\tau\theta_1^{\dagger}-\tau\theta_1^{\ddagger}\mathsf{H}_{\mathsf{hk}}^{(\lambda,\mathsf{poly})}(\mathsf{MSG}\|\mathbb{S}),\tau\theta_1^{\ddagger})\boldsymbol{Y}$, $\widehat{\omega}_\pi=\vartheta$, $\zeta_0=\varkappa$, $\vec{\zeta}^{(\pi,m_\pi+1)}=\varkappa(1,\mathsf{H}_{\mathsf{hk}}^{(\lambda,\mathsf{poly})}(\mathsf{MSG}\|\mathbb{S}))\boldsymbol{X}$, $\widehat{\upsilon}_{\pi,0}=\xi_0$, and $\vec{\upsilon}^{(\pi,m_\pi+1)}=\vec{\xi}^{(1)}+\mathsf{H}_{\mathsf{hk}}^{(\lambda,\mathsf{poly})}(\mathsf{MSG}_\pi\|\mathbb{S}_\pi)\vec{\gamma}^{(2)}$.

Clearly, $\widehat{\omega}_\pi$ is uniformly and independently (from all the other variables) distributed in $\mathbb{F}_q\backslash\{0\}$ except with probability $1/q$ since $\vartheta\xleftarrow{\mathsf{U}}\mathbb{F}_q$. Similarly, u,κ,v, and $\zeta_{\pi,0}$ are uniformly and independently (from all the other variables) distributed in $\mathbb{F}_q$ except when $\tau=0$, i.e., except with probability $1/q$ respectively since $\theta_2^{\dagger},\theta_2^{\ddagger},\theta_1^{\dagger},\varkappa\xleftarrow{\mathsf{U}}\mathbb{F}_q$. Now, observe that since $(\mathsf{MSG},\mathbb{S})\neq(\mathsf{MSG}_\pi,\mathbb{S}_\pi)$ by the restriction imposed on $\mathcal{A}$ in the experiment, $\vec{r}^{(m+1)}\cdot\vec{\zeta}^{(\pi,m_\pi+1)}=\Lambda\varkappa\tau\theta_1^{\ddagger}+\varkappa\tau\theta_1^{\dagger}$ with $\Lambda=\mathsf{H}_{\mathsf{hk}}^{(\lambda,\mathsf{poly})}(\mathsf{MSG}_\pi\|\mathbb{S}_\pi)-\mathsf{H}_{\mathsf{hk}}^{(\lambda,\mathsf{poly})}(\mathsf{MSG}\|\mathbb{S})\neq 0$ except with probability $\mathsf{Adv}_{\mathcal{M}}^{\mathsf{H},\mathsf{CR}}(\lambda)$ for some probabilistic algorithm $\mathcal{M}$ with essentially the same running time as that of $\mathcal{A}$. Hence, $\vec{r}^{(m+1)}\cdot\vec{\zeta}^{(\pi,m_\pi+1)}$ is uniformly and independently (from all the other variables) distributed in $\mathbb{F}_q$ except with probability $\mathsf{Adv}_{\mathcal{M}}^{\mathsf{H},\mathsf{CR}}(\lambda)$ when τ and $\varkappa$ is non-zero since $\theta_1^{\ddagger}\xleftarrow{\mathsf{U}}\mathbb{F}_q$. Moreover, from Lemma 24, the pair of vectors $(\vec{r}^{(m+1)},\vec{\zeta}^{(\pi,m_\pi+1)})$ is uniformly distributed in $\mathbb{C}_{\Lambda\varkappa\tau\theta_1^{\ddagger}+\varkappa\tau\theta_1^{\dagger}}$. Hence, it follows that $\vec{r}^{(m+1)}$ and $\vec{\zeta}^{(\pi,m_\pi+1)}$ are uniformly and independently (from all the other variables) distributed in $\mathbb{F}_q^2$ except with probability $\mathsf{Adv}_{\mathcal{M}}^{\mathsf{H},\mathsf{CR}}(\lambda)$ provided τ and $\varkappa$ is non-zero.

Therefore, it follows that the joint distribution of $\boldsymbol{c}^{(0)},\boldsymbol{c}^{(m+1)},\boldsymbol{s}^{*(\pi,0)}$, and $\boldsymbol{s}^{*(\pi,m_\pi+1)}$ is the same as that in $\mathsf{Hyb}_{4\text{-}\pi}$ except with probability $\mathsf{Adv}_{\mathcal{M}}^{\mathsf{H},\mathsf{CR}}(\lambda)+3/q$ in this case. This completes the proof of Lemma 23 ∎

Lemma 25: *For any stateful probabilistic adversary $\mathcal{A}$, for any security parameter λ,* $\left|\mathsf{Adv}_{\mathcal{A}}^{(4\text{-}q_{\mathsf{sig}})}(\lambda)-\mathsf{Adv}_{\mathcal{A}}^{(5)}(\lambda)\right|\leq 1/q$.

Proof: In order to prove Lemma 25, we show that the distribution $(\mathsf{MPK},\{\mathsf{SK}(\vec{x}^{(h)})\}_{h\in[q_{\mathsf{key}}]},\{\mathsf{SIG}_t\}_{t\in[q_{\mathsf{sig}}]},(\boldsymbol{c}^{(0)},\ldots,\boldsymbol{c}^{(m+1)}))$ in $\mathsf{Hyb}_{4\text{-}q_{\mathsf{sig}}}$ and that in Hyb_5 are equivalent, where $\mathsf{MPK}=(\mathsf{hk},\mathsf{params},\{\widehat{\mathbb{B}}_\iota,\widehat{\mathbb{B}}_\iota^*\}_{\iota\in[0,2]})$ is the public parameters given to $\mathcal{A}$, $\mathsf{SK}(\vec{x}^{(h)})=(\boldsymbol{k}^{*(h,0)},\ldots,\boldsymbol{k}^{*(h,n)},\boldsymbol{k}^{*(h,n+1,1)},\boldsymbol{k}^{*(h,n+1,2)})$ is the answer to the h^{th} signing key reveal query of $\mathcal{A}$, $\mathsf{SIG}_t=(\boldsymbol{s}^{*(t,0)},\ldots,\boldsymbol{s}^{*(t,m_t+1)})$ is the answer to the t^{th} signature reveal query of $\mathcal{A}$, and $(\boldsymbol{c}^{(0)},\ldots,\boldsymbol{c}^{(m+1)})$ is the verification-text used to check the forged signature outputted by $\mathcal{A}$ at the end of the experiment. By the definition of these hybrids, it is clear that we only need to consider the components $\{\boldsymbol{k}^{*(h,0)}\}_{h\in[q_{\mathsf{key}}]}$, $\{\boldsymbol{s}^{*(t,0)}\}_{t\in[q_{\mathsf{sig}}]}$, and $\boldsymbol{c}^{(0)}$. Let us start with the joint distribution of these components in $\mathsf{Hyb}_{4\text{-}q_{\mathsf{sig}}}$. Define new dual orthonormal bases $(\mathbb{D}_0,\mathbb{D}_0^*)$ of $(\mathbb{V}_0,\mathbb{V}_0^*)$ from the original bases $(\mathbb{B}_0,\mathbb{B}_0^*)$ used in $\mathsf{Hyb}_{4\text{-}q_{\mathsf{sig}}}$ as follows: Generate $\Lambda\xleftarrow{\mathsf{U}}\mathbb{F}_q$, compute

$$\boldsymbol{d}^{(0,2)}=\boldsymbol{b}^{(0,2)}+\Lambda\boldsymbol{b}^{(0,1)},\boldsymbol{d}^{*(0,1)}=\boldsymbol{b}^{*(0,1)}-\Lambda\boldsymbol{b}^{*(0,2)},$$

and set

$$\mathbb{D}_0=\{\boldsymbol{b}^{(0,1)},\boldsymbol{d}^{(0,2)},\boldsymbol{b}^{(0,3)},\boldsymbol{b}^{(0,4)}\},$$
$$\mathbb{D}_0^*=\{\boldsymbol{d}^{*(0,1)},\boldsymbol{b}^{*(0,2)},\boldsymbol{b}^{*(0,3)},\boldsymbol{b}^{*(0,4)}\}.$$

It can be readily observed that the new bases $(\mathbb{D}_0,\mathbb{D}_0^*)$ are indeed dual orthonormal, and are distributed the same as the original bases $(\mathbb{B}_0,\mathbb{B}_0^*)$.

Now, observe that the components $\boldsymbol{c}^{(0)}$, $\{\boldsymbol{k}^{*(h,0)}\}_{h\in[q_{\mathsf{key}}]}$, and $\{\boldsymbol{s}^{*(t,0)}\}_{t\in[q_{\mathsf{sig}}]}$ in $\mathsf{Hyb}_{4\text{-}q_{\mathsf{sig}}}$ can be expressed over the new bases $(\mathbb{D}_0,\mathbb{D}_0^*)$ as follows:

$$\begin{aligned}
\boldsymbol{c}^{(0)} &= (-u-u_\ell,v,0,\eta_0)_{\mathbb{B}_0}\\
&= (-u-u_\ell-v\Lambda,v,0,\eta_0)_{\mathbb{D}_0}\\
&= (w,v,0,\eta_0)_{\mathbb{D}_0},\\
\boldsymbol{k}^{*(h,0)} &= (\omega_h,\mathfrak{I}_h,\varphi_{h,0},0)_{\mathbb{B}_0^*}\\
&= (\omega_h,\mathfrak{I}_h+\omega_h\Lambda,\varphi_{h,0},0)_{\mathbb{D}_0^*}\\
&= (\omega_h,\mathfrak{I}_h',\varphi_{h,0},0)_{\mathbb{D}_0^*}\text{ for }h\in[q_{\mathsf{key}}],\\
\boldsymbol{s}^{*(t,0)} &= (\widehat{\omega}_t,\zeta_{t,0},\widehat{\upsilon}_{t,0},0)_{\mathbb{B}_0^*}\\
&= (\widehat{\omega}_t,\zeta_{t,0}+\widehat{\omega}_t\Lambda,\widehat{\upsilon}_{t,0},0)_{\mathbb{D}_0^*}\\
&= (\widehat{\omega}_t,\zeta_{t,0}',\widehat{\upsilon}_{t,0},0)_{\mathbb{D}_0^*}\text{ for }t\in[q_{\mathsf{sig}}],
\end{aligned}$$

where $w=-u-u_\ell-v\Lambda$, $\mathfrak{I}_h'=\mathfrak{I}_h+\omega_h\Lambda$ for $h\in[q_{\mathsf{key}}]$, and $\zeta_{t,0}'=\zeta_{t,0}+\widehat{\omega}_t\Lambda$ for $t\in[q_{\mathsf{sig}}]$. Clearly, for all $h\in[q_{\mathsf{key}}]$, $\mathfrak{I}_h'$ is uniformly and independently (from all the other variables) distributed since $\mathfrak{I}_h\xleftarrow{\mathsf{U}}\mathbb{F}_q$. Similarly, for all $t\in[q_{\mathsf{sig}}]$, $\zeta_{t,0}'$ is uniformly and independently (from all the other variables) distributed since $\zeta_{t,0}\xleftarrow{\mathsf{U}}\mathbb{F}_q$. Finally, w is uniformly and independently (of all the other variables) distributed except when $v=0$, i.e., except with probability $1/q$ since $\Lambda\xleftarrow{\mathsf{U}}\mathbb{F}_q$. Thus, it follows that $\boldsymbol{c}^{(0)}$, $\{\boldsymbol{k}^{*(h,0)}\}_{h\in[q_{\mathsf{key}}]}$, and $\{\boldsymbol{s}^{*(t,0)}\}_{t\in[q_{\mathsf{sig}}]}$ generated in $\mathsf{Hyb}_{4\text{-}q_{\mathsf{sig}}}$ take the form as in Hyb_5 when expressed over the transformed bases $(\mathbb{D}_0,\mathbb{D}_0^*)$ except with probability $1/q$.

Clearly, in the view of $\mathcal{A}$, both the original bases

$(\mathbb{B}_0, \mathbb{B}_0^*)$ and the transformed bases $(\mathbb{D}_0, \mathbb{D}_0^*)$ are consistent with the public parameters MPK. Therefore, $\mathsf{Hyb}_{4\text{-}q_{\mathrm{sig}}}$ can be conceptually changed to Hyb_5 except with probability $1/q$. This completes the proof of Lemma 25. ∎

Lemma 26: *For any stateful probabilistic adversary $\mathcal{A}$, for any security parameter λ, $\mathsf{Adv}^{(5)}_{\mathcal{A}}(\lambda) = 1/q$.*

Proof: Let the forged signature outputted by $\mathcal{A}$ in Hyb_5 be $\mathsf{SIG} = (\boldsymbol{s}^{*(0)}, \ldots, \boldsymbol{s}^{*(m+1)})$. If the coefficient of $\boldsymbol{b}^{*(0,1)}$ in the expression of $\boldsymbol{s}^{*(0)}$ is 0, then $e(\boldsymbol{b}^{(0,1)}, \boldsymbol{s}^{*(0)}) = 1_{\mathbb{G}_T}$ holds, and the verification fails by the specification of the algorithm ABS.Verify. On the other hand, if the coefficient of $\boldsymbol{b}^{*(0,1)}$ in the expression of $\boldsymbol{s}^{*(0)}$ is non-zero, then due to the fact that in Hyb_5 the coefficient w of $\boldsymbol{b}^{(0,1)}$ in the expression of the component $\boldsymbol{c}^{(0)}$ of the verification-text used to verify the forgery is uniformly and independently distributed from all the other variables, the verification also fails except with probability $1/q$. Hence, it follows that $\mathsf{Adv}^{(5)}_{\mathcal{A}}(\lambda) = 1/q$. This completes the proof of Lemma 26. ∎

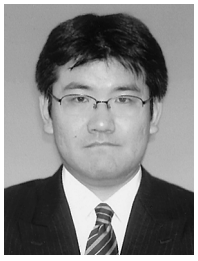
References

[1] P. Datta, T. Okamoto, and K. Takashima, "Efficient attribute-based signatures for unbounded arithmetic branching programs," PKC 2019, pp.127–158, Springer, 2019.
[2] H. Maji, M. Prabhakaran, and M. Rosulek, "Attribute-based signatures: Achieving attribute-privacy and collusion-resistance," Cryptology ePrint Archive, Report 2008/328, 2008.
[3] W. Diffie and M. Hellman, "New directions in cryptography," IEEE Trans. Inf. Theory, vol.22, no.6, pp.644–654, 1976.
[4] H.K. Maji, M. Prabhakaran, and M. Rosulek, "Attribute-based signatures," CT-RSA 2011, pp.376–392, Springer, 2011.
[5] H.K. Maji, M. Prabhakaran, and M. Rosulek, "Attribute-based signatures," Cryptology ePrint Archive, Report 2010/595, 2010. This is the full version of [4].
[6] S.F. Shahandashti and R. Safavi-Naini, "Threshold attribute-based signatures and their application to anonymous credential systems," AFRICACRYPT 2009, pp.198–216, Springer, 2009.
[7] J. Li and K. Kim, "Attribute-based ring signatures," Cryptology ePrint Archive, Report 2008/394, 2008.
[8] J. Li, M.H. Au, W. Susilo, D. Xie, and K. Ren, "Attribute-based signature and its applications," ASIACCS 2010, pp.60–69, ACM, 2010.
[9] J. Herranz, F. Laguillaumie, B. Libert, and C. Ràfols, "Short attribute-based signatures for threshold predicates," CT-RSA 2012, pp.51–67, Springer, 2012.
[10] T. Okamoto and K. Takashima, "Efficient attribute-based signatures for non-monotone predicates in the standard model," PKC-2011, pp.35–52, Springer, 2011.
[11] T. Okamoto and K. Takashima, "Decentralized attribute-based signatures," PKC 2013, pp.125–142, Springer, 2013.
[12] A. El Kaafarani, E. Ghadafi, and D. Khader, "Decentralized traceable attribute-based signatures," CT-RSA 2014, pp.327–348, Springer, 2014.
[13] A. El Kaafarani and R. El Bansarkhani, "Post-quantum attribute-based signatures from lattice assumptions," Cryptology ePrint Archive, Report 2016/823, 2016.
[14] F. Tang, H. Li, and B. Liang, "Attribute-based signatures for circuits from multilinear maps," ISC 2014, pp.54–71, Springer, 2014.
[15] Y. Sakai, N. Attrapadung, and G. Hanaoka, "Attribute-based signatures for circuits from bilinear map," PKC 2016, pp.283–300, Springer, 2016.
[16] R. Tsabary, "An equivalence between attribute-based signatures and homomorphic signatures, and new constructions for both," TCC 2017, pp.489–518, Springer, 2017.
[17] A. El Kaafarani and S. Katsumata, "Attribute-based signatures for unbounded circuits in the ROM and efficient instantiations from lattices," PKC 2018, pp.89–119, Springer, 2018.
[18] P. Datta, R. Dutta, and S. Mukhopadhyay, "Attribute-based signatures for turing machines," Cryptology ePrint Archive, Report 2017/801, 2017.
[19] Y. Sakai, S. Katsumata, N. Attrapadung, and G. Hanaoka, "Attribute-based signatures for unbounded languages from standard assumptions," ASIACRYPT 2018, pp.493–522, Springer, 2018.
[20] M. Bellare and G. Fuchsbauer, "Policy-based signatures," PKC 2014, pp.520–537, Springer, 2014.
[21] T. Okamoto and K. Takashima, "Efficient attribute-based signatures for non-monotone predicates in the standard model," Cryptology ePrint Archive, Report 2011/700, 2011. This is the full version of [10].
[22] M. Fürer, "Faster integer multiplication," SIAM J. Comput., vol.39, no.3, pp.979–1005, 2009.
[23] B. Applebaum, Y. Ishai, and E. Kushilevitz, "How to garble arithmetic circuits," SIAM J. Comput., vol.43, no.2, pp.905–929, 2014.
[24] M. Keller, E. Orsini, and P. Scholl, "MASCOT: Faster malicious arithmetic secure computation with oblivious transfer," ACM-CCS 2016, pp.830–842, ACM, 2016.
[25] B. Parno, J. Howell, C. Gentry, and M. Raykova, "Pinocchio: Nearly practical verifiable computation," Commun. ACM, vol.59, no.2, pp.103–112, 2016.
[26] Y. Ishai and E. Kushilevitz, "Private simultaneous messages protocols with applications," ITCS 1997, pp.174–183, IEEE, 1997.
[27] Y. Ishai and E. Kushilevitz, "Perfect constant-round secure computation via perfect randomizing polynomials," ICALP 2002, pp.244–256, Springer, 2002.
[28] M. Abe, M. Chase, B. David, M. Kohlweiss, R. Nishimaki, and M. Ohkubo, "Constant-size structure-preserving signatures: Generic constructions and simple assumptions," ASIACRYPT 2012, pp.4–24, Springer, 2012.
[29] D.M. Freeman, "Converting pairing-based cryptosystems from composite-order groups to prime-order groups," EUROCRYPT 2010, pp.44–61, Springer, 2010.
[30] A. Guillevic, "Comparing the pairing efficiency over composite-order and prime-order elliptic curves," ACNS 2013, pp.357–372, Springer, 2013.
[31] R. Barbulescu, P. Gaudry, A. Joux, and E. Thomé, "A heuristic quasi-polynomial algorithm for discrete logarithm in finite fields of small characteristic," EUROCRYPT 2014, pp.1–16, Springer, 2014.
[32] F. Göloğlu, R. Granger, G. McGuire, and J. Zumbrägel, "On the function field sieve and the impact of higher splitting probabilities," CRYPTO 2013, pp.109–128, Springer, 2013.
[33] A. Joux, "Faster index calculus for the medium prime case application to 1175-bit and 1425-bit finite fields," EUROCRYPT 2013, pp.177–193, Springer, 2013.
[34] A. Joux, "A new index calculus algorithm with complexity $l(1/4 + o(1))$ in small characteristic," SAC 2013, pp.355–379, Springer, 2013.
[35] M. Karchmer and A. Wigderson, "On span programs," Structure in Complexity Theory Conference 1993, pp.102–111, IEEE, 1993.
[36] Y. Ishai and H. Wee, "Partial garbling schemes and their applications," ICALP 2014, pp.650–662, Springer, 2014.
[37] T. Okamoto and K. Takashima, "Hierarchical predicate encryption for inner-products," ASIACRYPT 2009, pp.214–231, Springer, 2009.
[38] T. Okamoto and K. Takashima, "Fully secure functional encryption with general relations from the decisional linear assumption," CRYPTO 2010, pp.191–208, Springer, 2010.
[39] J. Tomida, M. Abe, and T. Okamoto, "Efficient functional encryption for inner-product values with full-hiding security," ISC 2016, pp.408–425, Springer, 2016.

[40] V. Shoup, "Lower bounds for discrete logarithms and related problems," EUROCRYPT 1997, pp.256–266, Springer, 1997.
[41] T. Okamoto and K. Takashima, "Fully secure unbounded inner-product and attribute-based encryption," ASIACRYPT 2012, pp.349–366, Springer, 2012.
[42] T. Okamoto and K. Takashima, "Fully secure unbounded inner-product and attribute-based encryption," Cryptology ePrint Archive, Report 2012/671, 2012. This is the full version of [41].
[43] E. Shi and B. Waters, "Delegating capabilities in predicate encryption systems," ICALP 2008, pp.560–578, Springer, 2008.
[44] B. Waters, "Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization," PKC 2011, pp.53–70, Springer, 2011.
[45] B. Waters, "Dual system encryption: Realizing fully secure ibe and hibe under simple assumptions," CRYPTO 2009, pp.619–636, Springer, 2009.
[46] A. Lewko and B. Waters, "New techniques for dual system encryption and fully secure hibe with short ciphertexts," TCC 2010, pp.455–479, Springer, 2010.
[47] K. Takashima, "New proof techniques for DLIN-based adaptively secure attribute-based encryption," ACISP 2017, pp.85–105, Springer, 2017.
[48] L. Kowalczyk, J. Liu, T. Malkin, and K. Meiyappan, "Mitigating the one-use restriction in attribute-based encryption," ICISC 2018, pp.23–36, Springer, 2018.
[49] L. Kowalczyk and H. Wee, "Compact adaptively secure abe for NC^1 from k-Lin," EUROCRYPT 2019, pp.3–33, Springer, 2019.
[50] L. Kowalczyk and H. Wee, "Compact adaptively secure abe for NC^1 from k-Lin," J. Cryptol., vol.33, no.3, pp.954–1002, 2020.
[51] J. Groth and A. Sahai, "Efficient non-interactive proof systems for bilinear groups," EUROCRYPT 2008, pp.415–432, Springer, 2008.

Pratish Datta received the B.Sc. degree from St. Xavier's College Kolkata, an autonomous college under the University of Calcutta, Kolkata, India in 2010, while the M.Sc. and Ph.D. degrees from the Indian Institute of Technology Kharagpur, Kharagpur, India in 2012 and in 2017 respectively. He is working as a scientist at the Cryptography and Information Security Laboratories of the NTT Research, Inc, U.S.A since 2019. Before that he was a postdoctoral fellow at the Secure Platform Laboratories of the NTT Corporation, Japan for nearly three years. His research area is cryptography and information security. He is a member of IACR.

Tatsuaki Okamoto received B.E., M.E., and Ph.D. degrees from the University of Tokyo, Tokyo, Japan, in 1976, 1978, and 1988, respectively. With the NTT Corporation since 1978, he is an NTT Fellow and, presently, the Director of the Cryptography and Information Security Laboratories of the NTT Research, Inc., USA. He focuses on research in cryptography and information security. In the past, he served as President of JSIAM, as Director of IACR, and as Program Chair of many international conferences. He received the Best and Life-Time Achievement awards from IEICE, the Distinguished Lecturer award from IACR, the Purple-Ribbon award from Japanese government, the RSA Conference award and the Asahi award.



Katsuyuki Takashima received the B.S., M.S. and Ph.D. degrees from Kyoto University, Kyoto, Japan, in 1993, 1995 and 2009, respectively. He is presently engaged in research on cryptography and information security at Information Technology R&D center, Mitsubishi Electric Corporation, Japan. He received the JSIAM Transactions Best Paper Award in 2003 and 2016, and IEICE Best Paper Award in 2015 and 2016. He is a member of JSIAM, MSJ, IPSJ, and IACR.