The preliminary versions of this paper appeared in *Proceedings of the 2nd ACM ASIA Public-Key Cryptography Workshop - ASIAPKC 2014*, pp. 49-58, under the title "Attribute-Based Signatures without Pairings via the Fiat-Shamir Paradigm", and *Proceedings of the 18th Annual International Conference on Information Security and Cryptology - ICISC 2015*, pp. 36-49, Lecture Notes in Computer Science 9558, Springer 2016, under the title "Attribute-Based Two-Tier Signatures: Definition and Construction". This is the full version. The statement on the attribute privacy was corrected.

Proofs of Knowledge on Monotone Predicates and its Application to Attribute-Based Identifications and Signatures*

Hiroaki Anada[1], Seiko Arita[2], and Kouichi Sakurai[3,4]

[1] Department of Information Security, University of Nagasaki
W408, 1-1-1, Manabino, Nagayo-cho, Nishisonogi-gun, Nagasaki, 851-2195 JAPAN
anada@sun.ac.jp
[2] Institute of Information Security
509, 2-14-1, Tsuruya-cho, Kanagawa-ku, Yokohama, 221-0835 JAPAN
arita@iisec.ac.jp
[3] Department of Informatics, Kyushu University
W2-712, 744, Motooka, Nishi-ku, Fukuoka, 819-0395 JAPAN
sakurai@inf.kyushu-u.ac.jp
[4] Institute of Systems, Information Technologies and Nanotechnologies
7F, Fukuoka SRP Center Bldg., 2-1-22, Momochihama, Sawara-ku, Fukuoka, 814-0001 JAPAN

January 12, 2017

Abstract. We propose a concrete procedure of a Σ-protocol proving knowledge that a set of witnesses satisfies a monotone predicate in witness-indistinguishable manner. Inspired by the high-level proposal by Cramer, Damgård and Schoenmakers at CRYPTO '94, we construct the concrete procedure by extending the so-called OR-proof. Next, using as a witness a credential-bundle of the Fiat-Shamir signatures, we provide an attribute-based identification scheme (ABID). Then, applying the Fiat-Shamir transform to our ABID, we obtain an attribute-based signature scheme (ABS). These generic schemes are constructed from a given Σ-protocol, and the latter scheme has a feature of linkable signatures. Applying the two-tier technique proposed at PKC 2007 by Bellare and Shoup to our ABID, we obtain an attribute-based two-tier signature scheme (ABTTS). The scheme has a feature to attain attribute-privacy paying expense of the secondary-key issuing. We provide two directions of instantiation. One is to use the Guillou-Quisquater and the Schnorr Σ-protocols, which produce ABID, ABS and ABTTS schemes with a loose security reduction in the random oracle model without pairing computation. The other is to use the Camenisch-Lysyanskaya Σ-protocols in the RSA setting and discrete-logarithm setting, which produce ABTTS schemes with a tighter security reduction in the standard model.

Keywords: proof of knowledge, access structure, attribute-based, identification, signature, two-tier keys

1 Introduction

A Σ-protocol formalized in the doctoral thesis of Cramer [Cra96] is a protocol of a 3-move public-coin interactive proof system with completeness, special soundness and honest-verifier zero-knowledge. It is one of the simplest

* The first and the second authors are partially supported by *kakenhi* Grant-in-Aid for Scientific Research (C) JP15K00029 from Japan Society for the Promotion of Science.

protocols of zero-knowledge interactive proof systems with an easy simulator. Also, it is one of the most typical proof of knowledge systems [BG92]; witness-extraction property by the special soundness enables us to prove that an identification scheme by a Σ-protocol is secure against active and concurrent attacks via a reduction to a number-theoretic assumption [BP02]. Instantiations of the Σ-protocol have been known as the Schnorr protocol [Sch89] and the Guillou-Quisquater protocol [GQ88] of identification schemes. They can be converted into digital signature schemes by the Fiat-Shamir heuristic [FS86]. The signature scheme can be proved secure against chosen-message attacks in the random oracle model [PS96], based on the security of the identification scheme against passive attacks [AABN02]. By virtue of these features, a Σ-protocol can be adopted into building blocks of various cryptographic primitives such as anonymous credential systems [CL02] and group signature schemes [BBS04].

The OR-proof proposed by Cramer, Damgård and Schoenmakers at CRYPTO '94 [CDS94] is a Σ-protocol derived from an original Σ-protocol [Dam10]. It is a witness-indistinguishable protocol [FS90] by which a prover can convince a verifier that a prover knows one of two (or both) witnesses while even an unbounded verifier cannot tell which witness is used. The OR-proof is essentially applied in, for example, the construction of a non-malleable proof of plaintext knowledge [Kat03]. In the paper of Cramer et al. [CDS94], a more general protocol was proposed[5]; suppose a prover and a verifier are given a monotone boolean predicate f over boolean variables. Here a monotone boolean predicate means a boolean predicate without negation; that is, boolean variables connected by AND-gates and OR-gates, but no NOT-gate is used. '1' (True) is substituted into every variable in f at which the prover knows the corresponding witness, and '0' (False) is substituted into every remaining variable. The protocol attains witness-indistinguishability in the sense that the prover knows a satisfying set of witnesses while even an unbounded verifier cannot tell which satisfying set is used. This protocol is an extension of the OR-proof to any monotone boolean predicate, and in [CDS94] a high-level construction that employed a "semi-smooth" secret-sharing scheme was given. (As is explained in [CDS94], to remove the restriction of the monotonicity of f looks hard.)

In this paper, we provide a concrete procedure of the protocol. We start with a given Σ-protocol $\mathbf{\Sigma}$, and derive a Σ-protocol $\mathbf{\Sigma}_f$ for any monotone boolean predicate f. Then we show that our $\mathbf{\Sigma}_f$ is actually a Σ-protocol with witness-indistinguishability.

Then, we will try to apply the protocol $\mathbf{\Sigma}_f$ to construct an attribute-based identification scheme (ABID) and an attribute-based signature scheme (ABS). In ABID, an identification-session is associated with an access structure described as a boolean predicate over an attribute universe. A prover can make a verifier accept only when the prover's set of attributes satisfies the access structure. ABS, in our strategy, is obtained by applying the Fiat-Shamir heuristic to ABID. The concept of ABS has been developed since 2008 [GZ08,SS09,LAS$^+$10,KABR10,MPR11] [HLR10,EHM11,OT11,GNS12,HLLR12,OT13,Her14,EGK14,ECGD14,EGK14,Gha15,Her16a,SAH16]. However, almost all the constructions are via the approach similar to that of attribute-based encryption schemes (ABE, [SW04]), which uses bilinear maps (that is, pairings on elliptic curves). A few exception are generic constructions by Maji et al. [MPR11] and Bellare et al. [BF14], and concrete constructions by Herranz in the RSA setting [Her14] and in the discrete logarithm setting [Her16a]. In contrast to the approach by bilinear maps, we work through a different approach in the Fiat-Shamir paradigm [FS86], which shares a spirit with [Her14]. Note that, in this paper, we do not try to proceed the usual way to attain attribute-privacy [MPR11,OT11,Her14] which means that signatures reveal nothing about the identity or attributes of the signer beyond what is explicitly revealed by the satisfied boolean predicate, but we will pursue the Fiat-Shamir approach. First, we construct a *linkable* attribute-based signature scheme. Then, after introducing a syntax of attribute-based two-tier signature scheme (ABTTS) [AAS15], we construct ABTTS to attain attribute-privacy paying expense of the secondary-key issuing [BS07].

1.1 Our Construction Idea

To provide a concrete procedure for the above protocol $\mathbf{\Sigma}_f$ with witness-indistinguishability, we look into the technique employed in the OR-proof [CDS94] and expand it so that it can treat any monotone boolean predicate, as follows. First express the boolean predicate f as a binary tree $\mathcal{T}_f$. That is, we put leaves each of which corresponds to each position of a variable in f. We connect two leaves by an $\wedge$-node or an $\vee$-node according to an AND-gate or an OR-gate which is between two corresponding positions in f. Then we connect the resulting nodes by an $\wedge$-node or an $\vee$-node in the same way, until we reach to the root node (which is also an $\wedge$-node or an $\vee$-node). A verification equation of the Σ-protocol $\mathbf{\Sigma}$ is assigned to every leaf. If a challenge string $\textsc{Cha}$ of $\mathbf{\Sigma}$ is given, then the prover assigns the string $\textsc{Cha}$ to the root node. If the root node is an $\wedge$-node, then the prover assigns the same string $\textsc{Cha}$ to two children. Else if the root node is an $\vee$-node, then the prover divides $\textsc{Cha}$ into two random strings $\textsc{Cha}_{\mathrm{L}}$ and $\textsc{Cha}_{\mathrm{R}}$ under the constraint that $\textsc{Cha} = \textsc{Cha}_{\mathrm{L}} \oplus \textsc{Cha}_{\mathrm{R}}$, and assigns $\textsc{Cha}_{\mathrm{L}}$ and $\textsc{Cha}_{\mathrm{R}}$ to the left child and the

[5] In the preliminary version [AAS14], the authors could not refer to this previous work. Now we refer to the work with explanation.

right child, respectively. Here $\oplus$ means a bitwise exclusive-OR operation. Then the prover continues to apply this rule at each height, step by step, until he reaches to every leaf. Basically, the OR-proof technique assures that we can either honestly execute the Σ-protocol $\mathbf{\Sigma}$ or execute the simulator of $\mathbf{\Sigma}$. Only when a set of witnesses satisfies the binary tree $\mathcal{T}_f$, the above procedure succeeds in satisfying verification equations for all leaves.

1.2 Our Contributions

Our first contribution is to provide a concrete procedure of the Σ-protocol of [CDS94], which is comparable with the original abstract protocol [CDS94]. That is, given a Σ-protocol $\mathbf{\Sigma}$ and a monotone boolean predicate f, we construct a concrete procedure $\mathbf{\Sigma}_f$ in a recursive form that is suitable for implementation. Then we show that $\mathbf{\Sigma}_f$ is certainly a Σ-protocol with witness-indistinguishability.

Our second contribution is to provide a concrete schemes in two directions. One is to use the Guillou-Quisquater [GQ88] and the Schnorr [Sch89] Σ-protocols, which produce ABID, ABS and ABTTS schemes with a loose security reduction in the random oracle model without pairing computation. The other is to use the Camenisch-Lysyanskaya Σ-protocols in the RSA setting [CL02] and discrete-logarithm setting [FI05,Oka06,TF12] to exit the drawbacks of the loose reduction. For the purpose, we introduce a syntax of attribute-based two-tier signature scheme [AAS15], and construct concrete ABTTS schemes with a tighter reduction in the standard model.

1.3 Related Work on ABS

At a high level, our ABS is obtained by the Fiat-Shamir transform of our $\mathbf{\Sigma}_f$, where a set of witnesses is the Fiat-Shamir credential-bundle [MPR11]. This construction can be compared with the generic construction of the ABS scheme by Maji et al. [MPR11]. They started with a credential bundle (of Boneh-Boyen signatures [BB04b], for instance), then they employed a non-interactive witness-indistinguishable proof of knowledge system (NIWIPoK) of Groth and Sahai [GS08] to prove the knowledge of a credential bundle which satisfies a given (monotone) access formula, in the standard model.

Okamoto and Takashima (OT11) [OT11] gave an ABS scheme with full-security; security against adaptive target in the standard model under a non-q-type assumption; it can treat any non-monotone access formula and multi-use of attributes, and possesses attribute privacy in the information-theoretic sense. The construction is based on their Dual Pairing Vector Space.

Herranz [Her14] provided the first ABS with both collusion resistance (against collecting private secret keys) and computational attribute privacy without pairings (pairing-free) in the RSA setting. In the work [Her14], the concrete procedure was described in detail for threshold-type access formulas. In contrast, our ABS is without pairings and provides a concrete procedure for any monotone access formulas without attribute privacy. Recently, Herranz [Her16a] provided an ABS scheme without pairings in the discrete-logarithm setting, but it has a constraint that the number of private secret keys is bounded in the set-up phase.

Kaafarani et al. [ECGD14] proposed the functionality of “User-Controlled Linkability” (UCL) in the case of attribute-based signatures. UCL property in the work [ECGD14] can be captured as a kind of public linkability. In general, public linkability is achieved with the expense of loosing attribute privacy in ABS, and hence the scheme [ECGD14] and our ABS scheme do not possess attribute privacy.

1.4 Technical and Efficiency Comparison on ABS

We compare our scheme with the above previously proposed schemes from the view point of security, functionality and length of a signature. The comparison is summarized in Table 1 with notations as follows. A prime of bit length λ (the security parameter) is denoted by p. Though a pairing map e should be analysed for the asymmetric bilinear groups [GKZ14], we simply evaluate for the symmetric case in which both source groups are $\mathbb{G}_p$ of order p. We assume that an element of $\mathbb{G}_p$ is represented by 2λ bits. l and r mean the number of rows and columns of the share-generating matrix for monotone access formula f (that is, an access structure), respectively. CR means the collision resistance of an employed hash function. q-SDH means the Strong Diffie-Hellman assumption with q-type input [BB04a]. DLIN means the Decisional Linear assumption [OT11]. DDH means the Decisional Diffie-Hellman assumption [ECGD14]. DL means the Discrete-Logarithm assumption [ECGD14]. q-SRSA means the strong RSA assumption with q-type input [CL02,Her14]. DDH in $QR(N)$ means the Decisional Diffie-Hellman assumption for quadratic residues modulo N (the RSA modulus) [Her14]. In [Her14,Her16a], θ is the threshold value of a threshold-type access structure. In [Her14], κ is a security parameter. In [Her16a], $M = L + N$ is the sum of the upper bound L of the number of users in the set-up phase and the upper bound N of the number of all attributes in the attribute

Table 1. Technical and Efficiency Comparison on ABS: Security, Functionality and Length of Signature.

Scheme	Access Formula	Security Model	Assumption	Adap. Target	Collu. Resist.	Att. Priv.	Pub.Link. UCL-Link.	Pairing -Free	Length of Signature	Remark
Maji et al. [MPR11]	Mono.	Std.	q-SDH $\wedge$ DLIN $\wedge$ CR	✓	✓	✓ (info.)	- -	-	$(2\lambda)\times$ $(51l+2r+18\lambda l)$	-
OT [OT11]	Non-mono.	Std.	DLIN $\wedge$ CR	✓	✓	✓ (info.)	- -	-	$(2\lambda)(9l+11)$	-
Herranz [Her14]	Mono.	R.O.	q-SRSA $\wedge$ DDH $\wedge$CR	✓	✓	✓ (comp.)	- -	✓	$\lambda_{\mathrm{rsa}}(5+\frac{\kappa}{\lambda_{\mathrm{rsa}}})l$ $+\lambda_{\mathrm{rsa}}3-\kappa(\theta-1)$	-
Herranz [Her16a]	Mono.	R.O.	DL $\wedge$CR	✓	✓	✓ (info.)	- -	✓	$(2\lambda)l+\lambda(6l-\theta)$ $+\lambda M(l+1)$	bounded num. keys
Kaafarani et al. [ECGD14]	Mono.	R.O.	q-SDH $\wedge$ DDH $\wedge$DL $\wedge$ CR	✓	✓	-	✓ ✓	-	$(2\lambda)(3l+r+3)$ $+\lambda(8l+4)$	-
Our `ABS`	Mono.	R.O.	DL $\wedge$ CR	✓	✓	-	✓ -	✓	$(2\lambda)(2l)$ $+\lambda 3l$	-
Our `ABTTS` (FS-sig.)	Mono.	R.O.	DL $\wedge$ CR	✓	✓	✓ (info.)	- -	✓	$\lambda(3l-1)$	two-tier keys
Our `ABTTS`$'$ (CL-sig.)	Mono.	Std.	q-SDH $\wedge$ CR	✓	✓	✓ (info.)	- -	-	$\lambda(3l-1)$	two-tier keys

universe. "info." means the information-theoretic security and "comp." means the computational security. "FS-sig." means a scheme that uses the Fiat-Shamir signatures [FS86] as a witness and "CL-sig." means a scheme that uses the Camenisch-Lysyanskaya signatures [CL02] as a witness.

The rigorous notion of ABS scheme was pioneered by the work of Maji et al. [MPR11]. The ABS scheme by Okamoto and Takashima [OT11] has advantages in the security model, the assumption, the treatable access formulas. The scheme by Herranz [Her14] is the only ABS scheme with the pairing-free feature, and with collusion resistance and computational attribute privacy and , in the RSA setting. Our procedure $\mathbf{\Sigma}_f$ of the Σ-protocol in [CDS94] for any monotone predicate serves as a building block of the Σ-protocol of [Her14]. Note that the security parameter λ_{rsa} in the RSA setting ([Her14], our `ABS`, our `ABTTS` and our `ABTTS`$'$ in RSA) is almost 9 times longer than λ in the discrete logarithm setting. For example, $\lambda_{\mathrm{rsa}} = 2048$ achieves almost equivalent security of $\lambda = 224$ [YSKI12].

Note that the ABS scheme by Herranz [Her16a] which is in the discrete-logarithm setting has a constraint that the number of secret keys is bounded in the set-up phase. Also, our attribute-based two-tier signature schemes [AAS15], `ABTTS` and `ABTTS`$'$, are in the two-tier setting which means that a secondary secret key and a secondary public key have to be issued for each signing session and the secondary keys is used only one time. Hence we believe that there is still an open problem to construct a pairing-free efficient ABS scheme in the discrete-logarithm setting.

The ABS scheme by Kaafarani et al. [ECGD14] has a feature of the user-controlled linkability. In contrast, our `ABS` has only the public linkability. It is notable that the ABS scheme [ECGD14] uses pairings and can be set up in the multi-authorities setting [OT13,EGK14,Gha15].

1.5 Organization of this Paper

In Section 2, we prepare for required tools and notions. In Section 3, we describe a concrete procedure of the Σ-protocol $\mathbf{\Sigma}_f$. In Section 4, by using a credential-bundle of the Fiat-Shamir signatures as a witness of our $\mathbf{\Sigma}_f$, we obtain our ABID. In Section 5, by applying the Fiat-Shamir transform to our ABID, we obtain our ABS. In Section 6, we define the syntax of ABTTS. In Section 7, by applying the technique of two-tier signature [AAS15] to our ABID, we obtain our ABTTS. In Section 8, we conclude our work in this paper. In Appendix A, B, C, D and E, we put the definitions of needed cryptographic primitives. In Appendix F and G, we show concrete instantiations of our ABID, ABS and ABTTS in the RSA setting and the discrete-logarithm setting.

2 Preliminaries

The security parameter is denoted by λ. The bit length of a string a is denoted by $|a|$. The concatenation of a string a with a string b is denoted by $a \parallel b$. A uniform random sampling of an element a from a set S is denoted by $a \in_R S$. The expression $a =_? b$ returns a value 1 (True) when $a = b$ and 0 (False) otherwise. The expression

$a \in_? S$ returns a value 1 when $a \in S$ and 0 otherwise. When an algorithm A with input a outputs z, we denote it as $z \leftarrow A(a)$, or, because of space limitation, $A(a) \rightarrow z$. When a probabilistic polynomial-time (PPT, for short) algorithm A with a random tape R and input a outputs z, we denote it as $z \leftarrow A(a; R)$. When A with input a and B with input b interact with each other and B outputs z, we denote it as $z \leftarrow \langle A(a), B(b)\rangle$. When A has oracle-access to $\mathcal{O}$, we denote it as $A^{\mathcal{O}}$. When A has concurrent oracle-access to n oracles $\mathcal{O}_1, \ldots, \mathcal{O}_n$, we denote it as $A^{\mathcal{O}_i|_{i=1}^n}$. Here "concurrent" means that A accesses oracles in arbitrarily interleaved order of messages. The probability of an event E is denoted by Pr[E]. The probability of an event E on condition that events $\mathrm{E}_1, \ldots, \mathrm{E}_m$ occur in this order is denoted as $\Pr[\mathrm{E}_1, \ldots, \mathrm{E}_m : \mathrm{E}]$.

2.1 Language, Proof of Knowledge and *Σ*-protocol [BG92,CDS94,Dam10]

Language Let $R = \{(x, w)\} \subset \{1,0\}^* \times \{1,0\}^*$ be a binary relation. For a pair $(x, w) \in R$ we call x a statement and w a witness of x. We say that R is polynomially bounded if there exists a polynomial *poly* such that $|w| \leq poly(|x|)$ for any $(x, w) \in R$. We say that R is an NP relation if it is polynomially bounded and, in addition, there exists a polynomial-time algorithm for deciding membership of (x, w) in R.

A *language* for a relation R is defined as:

$$L \stackrel{\text{def}}{=} \{x \in \{1,0\}^*; \exists w \in \{1,0\}^*, (x, w) \in R\}.$$

L is called a NP language if R is an NP relation. Hereafter, we assume that R is an NP relation.

We introduce a *relation-function* $R(\cdot, \cdot)$ associated with the relation R by:

$$\begin{aligned} \mathcal{R}(\cdot, \cdot) : \{1,0\}^* \times \{1,0\}^* &\rightarrow \{1,0\}, \\ (x, w) &\mapsto 1 \text{ if } (x, w) \in R,\ 0 \text{ otherwise.} \end{aligned}$$

Denote the set of witnesses corresponding to a statement x by $W(x)(= \{w \in \{0,1\}^*; R(x, w) = 1\})$.

Proof of Knowledge Informally, an interactive proof system [Bab85,GMR89] is a proof of knowledge system if the knowledge being proved can be efficiently computed by using the prover as a subroutine.

A *proof of knowledge system* (PoK for short) $\Pi = (\mathcal{P}, \mathcal{V})$ on a relation R is a protocol with two interactive PPT algorithms: $\mathcal{P}$, a prover, and $\mathcal{V}$, a verifier. $\mathcal{P}$ takes initial input $(x, w) \in R$ and $\mathcal{V}$ takes initial input x. $\mathcal{V}$ outputs 1 (accept) or 0 (reject) after at most a polynomial-number of moves of interaction and $\mathcal{P}$ and $\mathcal{V}$ satisfy the following two requirements.

Completeness. For any statement $x \in L$ and for any witness $w \in W(x)$, $\mathcal{P}$ with the witness w makes $\mathcal{V}$ accept for the statement x with probability 1:

$$\Pr[1 \leftarrow \langle \mathcal{P}(x, w), \mathcal{V}(x)\rangle] = 1.$$

Knowledge Soundness. There are a PPT algorithm $\mathcal{KE}$ called a *knowledge extractor*, a function $\kappa : \{1,0\}^* \rightarrow [1,0]$ called a *knowledge error function* and a constant $c > 0$ that satisfy the following: If there exists a PPT algorithm $\mathcal{A}$ that satisfies $p(x) := \Pr[1 \leftarrow \langle \mathcal{A}(x), \mathcal{V}(x)\rangle] > \kappa(x)$, then $\mathcal{KE}(x)$ that has oracle-access to $\mathcal{A}(x)$ outputs a witness $w \in W(x)$ within an expected number of steps bounded by: $|x|^c/(p(x) - \kappa(x))$.

Witness-Indistinguishable Proof of Knowledge [FS90,CDS94] Informally, an interactive proof system [Bab85,GMR89] is witness indistinguishable if the verifier cannot tell which witness $w \in W(x)$ the prover is using.

A *witness-indistinguishable proof of knowledge system* (WIPoK for short) $\Pi = (\mathcal{P}, \mathcal{V})$ on a relation R is a proof of knowledge system with the following requirement.

Witness-Indistinguishability. For any unbounded algorithm $\mathcal{A}$, we have

$$\begin{aligned} &\Pr[(x, w_0, w_1) \leftarrow \mathcal{A}(1^\lambda), 1 \leftarrow \langle \mathcal{P}(x, w_0), \mathcal{A}\rangle] \\ =&\Pr[(x, w_0, w_1) \leftarrow \mathcal{A}(1^\lambda), 1 \leftarrow \langle \mathcal{P}(x, w_1), \mathcal{A}\rangle] \end{aligned}$$

where $w_0, w_1 \in W(x) \vee w_0, w_1 \notin W(x)$ holds.

***Σ*-protocol [Cra96,Dam10]** Let R be an NP relation. A *Σ-protocol* on a relation R is a public-coin 3-move protocol of a proof of knowledge system $\Pi = (\mathcal{P}, \mathcal{V})$. $\mathcal{P}$ sends the first message called a commitment $\textsc{Cmt}$ to $\mathcal{V}$, then $\mathcal{V}$ sends the second message that is a public random string called a challenge $\textsc{Cha}$ to $\mathcal{P}$, and then $\mathcal{P}$ answers with the third message called a response $\textsc{Res}$ to $\mathcal{V}$. Then $\mathcal{V}$ applies a decision test on $(x, \textsc{Cmt}, \textsc{Cha}, \textsc{Res})$ to return 1 (accept) or 0 (reject). If $\mathcal{V}$ accepts, then the triple $(\textsc{Cmt}, \textsc{Cha}, \textsc{Res})$ is said to be an *accepting conversation on* x. Here $\textsc{Cha}$ is chosen uniformly at random from $\textsc{ChaSp}(1^\lambda) := \{1, 0\}^{l(\lambda)}$ with $l(\cdot)$ being a super-log function.

A Σ-protocol is described by the following PPT algorithm $\mathbf{\Sigma}$. $\textsc{Cmt} \leftarrow \mathbf{\Sigma}^1(x, w)$: the process of generating the first message $\textsc{Cmt}$ according to the protocol $\mathbf{\Sigma}$ on input $(x, w) \in R$. Similarly we denote $\textsc{Cha} \leftarrow \mathbf{\Sigma}^2(1^\lambda)$, $\textsc{Res} \leftarrow \mathbf{\Sigma}^3(x, w, \textsc{Cmt}, \textsc{Cha})$ and $b \leftarrow \mathbf{\Sigma}^{\mathrm{vrfy}}(x, \textsc{Cmt}, \textsc{Cha}, \textsc{Res})$.

Σ-protocol must possess the following three requirements.

Completeness. A prover $\mathcal{P}$ with a witness w can make $\mathcal{V}$ accept with probability 1.

Special Soundness. Any PPT algorithm $\mathcal{P}^*$ without any witness $w \in W(x)$ can respond to only one possible challenge $\textsc{Cha}$. In other words, there is a PPT algorithm called a *knowledge extractor*, $\mathbf{\Sigma}^{\mathrm{KE}}$, which, given a statement x and using $\mathcal{P}^*$ as a subroutine, can compute a witness w satisfying $(x, w^*) \in R$ with at most a negligible error probability, from two accepting conversations of the form $(\textsc{Cmt}, \textsc{Cha}, \textsc{Res})$ and $(\textsc{Cmt}, \textsc{Cha}', \textsc{Res}')$ with $\textsc{Cha} \neq \textsc{Cha}'$.

Honest-Verifier Zero-Knowledge. Given a statement x and a random challenge $\textsc{Cha} \leftarrow \mathbf{\Sigma}^2(1^\lambda)$, we can produce in polynomial-time, without knowing a witness $w \in W(x)$, an accepting conversation $(\textsc{Cmt}, \textsc{Cha}, \textsc{Res})$ whose distribution is the same as the real accepting conversation. In other words, there is a PPT algorithm called a *simulator*, $\mathbf{\Sigma}^{\mathrm{sim}}$, such that $(\textsc{Cmt}, \textsc{Res}) \leftarrow \mathbf{\Sigma}^{\mathrm{sim}}(x, \textsc{Cha})$.

As a zero-knowledge proof of knowledge system, we denote $\mathbf{\Sigma}$ as $\mathbf{ZKPoK}[w : x]$, where w is a witness proved by a prover $\mathcal{P}$ in zero-knowledge manner, and x is a statement for which the prover $\mathcal{P}$ and the verifier $\mathcal{V}$ have conversation. Any Σ-protocol is actually known to be a protocol of a proof of knowledge system ([Dam10]).

We will need in this paper a property called the *unique answer property* [BS07] that for legitimately produced commitment $\textsc{Cmt}$ and challenge $\textsc{Cha}$, there exists one and only one response $\textsc{Res} =: \tilde{w}$ that is accepted by a verifier. Known Σ-protocols such as the Schnorr protocol and the Guillou-Quisquater protocol [Sch89,BP02] possess this property. For such a unique answer $\tilde{w}$ we consider a statement $\tilde{x}$ such that $(\tilde{x}, \tilde{w}) \in R$. Then, we further assume that both a prover and a verifier can compute, in polynomial-time, such an $\tilde{x}$ from $(x, \textsc{Cmt}, \textsc{Cha})$. We denote the PPT algorithm as $\mathbf{\Sigma}^{\mathrm{stmtgen}}$. That is;

$\mathbf{\Sigma}^{\mathrm{stmtgen}}(x, \textsc{Cmt}, \textsc{Cha})$:
  Compute $\tilde{x}$ s.t.
  $^{\exists 1}\tilde{w}$ s.t. $[(\textsc{Cmt}, \textsc{Cha}, \textsc{Res})$ is an accepting conversation on $x \wedge \textsc{Res} = \tilde{w} \wedge (\tilde{x}, \tilde{w}) \in R]$
  Return $\tilde{x}$

Known Σ-protocols [Sch89,BP02] possess this *statement generation property* (see Section F).

The OR-proof [Dam10] Consider the following relation for a boolean predicate $f(X_1, X_2) = X_1 \vee X_2$.

$$\begin{aligned} R_{\mathrm{OR}} = & \{(x = (x_0, x_1), w = (w_0, w_1)) \in \{1, 0\}^* \times \{1, 0\}^*; \\ & f(\mathcal{R}(x_0, w_0), \mathcal{R}(x_1, w_1)) = 1\}. \end{aligned}$$

The corresponding language is

$$L_{\mathrm{OR}} = \{x \in \{1, 0\}^*; \exists w, (x, w) \in R_{\mathrm{OR}}\}.$$

Suppose that a Σ-protocol $\mathbf{\Sigma}$ on a relation R is given. Then we can construct the protocol $\mathbf{\Sigma}_{\mathrm{OR}}$ on a relation R_{OR} as follows. For instance, suppose $(x_0, w_0) \in R$ holds. $\mathcal{P}$ computes $\textsc{Cmt}_0 \leftarrow \mathbf{\Sigma}^1(x_0, w_0)$, $\textsc{Cha}_1 \leftarrow \mathbf{\Sigma}^2(1^\lambda)$, $(\textsc{Cmt}_1, \textsc{Res}_1) \leftarrow \mathbf{\Sigma}^{\mathrm{sim}}(x_1, \textsc{Cha}_1)$ and sends $(\textsc{Cmt}_0, \textsc{Cmt}_1)$ to $\mathcal{V}$. Then $\mathcal{V}$ sends $\textsc{Cha} \leftarrow \mathbf{\Sigma}^2(1^\lambda)$ to $\mathcal{P}$. Then, $\mathcal{P}$ computes $\textsc{Cha}_0 := \textsc{Cha} \oplus \textsc{Cha}_1$, $\textsc{Res}_0 \leftarrow \mathbf{\Sigma}^3(x_0, w_0, \textsc{Cmt}_0, \textsc{Cha}_0)$ answers to $\mathcal{V}$ with $(\textsc{Cha}_0, \textsc{Cha}_1)$ and $(\textsc{Res}_0, \textsc{Res}_1)$. Here $\oplus$ denotes a bitwise exclusive-OR operation. Then both $(\textsc{Cmt}_0, \textsc{Cha}_0, \textsc{Res}_0)$ and $(\textsc{Cmt}_1, \textsc{Cha}_1, \textsc{Res}_1)$ are accepting conversations on x and have the same distribution as real accepting conversations. This protocol $\mathbf{\Sigma}_{\mathrm{OR}}$ can be proved to be a Σ-protocol. We often call $\mathbf{\Sigma}_{\mathrm{OR}}$ the *OR-proof*. The OR-proof is known to be witness-indistinguishable [CDS94].

The Fiat-Shamir Transform [AABN02] Suppose that a cryptographic hash function with collision resistance, $Hash_\mu(\cdot) : \{1,0\}^* \to \{1,0\}^{l(\lambda)}$, is given. We fix the hash key μ hereafter. A Σ-protocol $\mathbf{\Sigma}$ on a relation R can be transformed into a *non-interactive witness-indistinguishable proof of knowledge system* (NIWIPoK for short) [AABN02]. When a Σ-protocol $\mathbf{\Sigma}$ is an identification scheme, the resulting scheme is a digital signature scheme [AABN02]. The transform is described as follows. (Here, in the case of a NIWIPoK, the message m is empty.) On input the security parameter 1^λ, the key-generation algorithm runs the instance generator $\mathrm{Instance}_R$. it generates a pair of a statement and a witness $(x, w) \in R$: $(x, w) \leftarrow \mathrm{Instance}_R(1^\lambda)$. Then x is a public key and w is a secret key. Given a message $m \in \{1,0\}^*$, the signer execute: $\textsc{Cmt} \leftarrow \mathbf{\Sigma}^1(x, w)$, $\textsc{Cha} \leftarrow Hash_\mu(\textsc{Cmt} \parallel m)$, $\textsc{Res} \leftarrow \mathbf{\Sigma}^3(x, w, \textsc{Cmt}, \textsc{Cha})$. Then $\sigma := (\textsc{Cmt}, \textsc{Res})$ is a signature on m. We denote this signing algorithm as $\mathrm{FS}(\mathbf{\Sigma})^{\mathrm{sign}}(x, w, m) \to (\textsc{Cmt}, \textsc{Res}) =: \sigma$. On the other hand, the verifier runs: $\textsc{Cha} \leftarrow Hash_\mu(\textsc{Cmt} \parallel m)$ and returns $b \leftarrow \mathbf{\Sigma}^{\mathrm{vrfy}}(x, \textsc{Cmt}, \textsc{Cha}, \textsc{Res})$. We denote this verification algorithm as $\mathrm{FS}(\mathbf{\Sigma})^{\mathrm{vrfy}}(x, m, \sigma) \to b$.

The signature scheme $\mathrm{FS}(\mathbf{\Sigma}) = (\mathrm{Instance}_R(1^\lambda), \mathrm{FS}(\mathbf{\Sigma})^{\mathrm{sign}}, \mathrm{FS}(\mathbf{\Sigma})^{\mathrm{vrfy}})$ can be proved, in the random oracle model, to be *existentially unforgeable against chosen-message attacks* if and only if the underlying Σ-protocol $\mathbf{\Sigma}$ is secure against *passive attacks* as an identification scheme [AABN02]. More precisely, let q_H denote the maximum number of hash queries issued by a PPT adversary $\mathcal{F}$ on $\mathrm{FS}(\mathbf{\Sigma})$. Then, for any PPT $\mathcal{F}$, there exists a PPT $\mathcal{B}$ which satisfies the following inequality ($\mathrm{neg}(\cdot)$ means a negligible function).

$$\mathbf{Adv}^{\text{euf-cma}}_{\mathrm{FS}(\mathbf{\Sigma}),\mathcal{F}}(\lambda) \le q_H \mathbf{Adv}^{\mathrm{pa}}_{\mathbf{\Sigma},\mathcal{B}}(\lambda) + \mathrm{neg}(\lambda).$$

3 Our Construction of Witness-Indistinguishable Proofs of Knowledge on Monotone Predicates

In this section, we first construct a Σ-protocol $\mathbf{\Sigma}_f$ from a given Σ-protocol $\mathbf{\Sigma}$ and a monotone boolean predicate f so that $\mathbf{\Sigma}_f$ is a protocol of WIPoK on the relation R_f.

3.1 Witness-Indistinguishable Proofs of Knowledge on Monotone Predicates [CDS94,AAS14]

We revisit here the notion of a 3-move public-coin honest-verifier zero-knowledge proof of knowledge system which is also a witness-indistinguishable proof system introduced by Cramer, Damgård and Schoenmakers [CDS94]. Then we restate the definition for concreteness.

Let R be a binary relation. Let $f(X_{i_1}, \ldots, X_{i_a})$ be a boolean predicate over boolean variables $U = \{X_1, \ldots, X_u\}$.

Definition 1 (Cramer, Damgård and Schoenmakers [CDS94], Our Rewritten Form) *A relation R_f is defined by:*

$$\begin{aligned} R_f \stackrel{def}{=} \{&(x = (x_{i_1}, \ldots, x_{i_a}), w = (w_{i_1}, \ldots, w_{i_a})) \in \{1,0\}^* \times \{1,0\}^*;\\ &f(\mathcal{R}(x_{i_1}, w_{i_1}), \ldots, \mathcal{R}(x_{i_a}, w_{i_a})) = 1\}. \end{aligned}$$

R_f is a generalization of the relation R_{OR} for the OR-proof [CDS94,Dam10], where f is a boolean predicate with the single boolean connective OR: $X_1 \vee X_2$. Note that, if R is an NP relation, then R_f is also an NP relation under the assumption that a, the arity of f, is bounded by a polynomial in λ. The corresponding language is

$$L_f \stackrel{\mathrm{def}}{=} \{x \in \{1,0\}^*; \exists w, (x, w) \in R_f\}.$$

In [CDS94], a 3-move public-coin honest-verifier zero-knowledge proof of knowledge system for the language L_f was defined as a witness-indistinguishable proof system on any monotone predicate f satisfied by a set of witnesses. Then, in [CDS94], a Σ-protocol of the WIPoK system on the relation R_f was studied at a high level by using the notion of the dual access structure of the access structure determined by f.

3.2 Our Witness-Indistinguishable Proofs of Knowledge on Monotone Predicates

We will provide a concrete procedure $\mathbf{\Sigma}_f$ of a Σ-protocol of WIPoK on the relation R_f. $\mathbf{\Sigma}_f$ is a 3-move protocol between interactive PPT algorithms $\mathcal{P}$ and $\mathcal{V}$ on input a pair of a statement and a witness (x, w) for $\mathcal{P}$, and x for $\mathcal{V}$, where $(x := (x_{i_j})_{1 \le j \le \mathrm{arity}(f)}$ and $w := (w_{i_j})_{1 \le j \le \mathrm{arity}(f)}) \in R_f$. In our prover algorithm $\mathcal{P}$, there are three PPT subroutines $\mathbf{\Sigma}^{\mathrm{eval}}_f$, $\mathbf{\Sigma}^1_f$ and $\mathbf{\Sigma}^3_f$. On the other hand, in our verifier algorithm $\mathcal{V}$, there are two PPT subroutines

$$
\begin{array}{lcl}
\mathcal{P}(x,w,f): & & \mathcal{V}(x,f): \\
\quad \mathbf{\Sigma}_f^{\mathrm{eval}}(\mathcal{T}_f, S) \to (v_n)_n & & \\
\quad \text{If } v_{r(\mathcal{T}_f)} \neq 1, \text{ then abort} & & \\
\quad \text{else } \textsc{Cha}_{r(\mathcal{T}_f)} := * & & \\
& & \\
\quad \mathbf{\Sigma}_f^1(x,w,\mathcal{T}_f,(v_n)_n,\textsc{Cha}_{r(\mathcal{T}_f)}) & & \\
\quad \to ((\textsc{Cmt}_l)_l, (\textsc{Cha}_n)_n, (\textsc{Res}_l)_l) & \xrightarrow{(\textsc{Cmt}_l)_l} & \\
& & \\
\quad \textsc{Cha}_{r(\mathcal{T}_f)} := \textsc{Cha} & \xleftarrow{\textsc{Cha}} & \textsc{Cha} \leftarrow \mathbf{\Sigma}_f^2(1^\lambda) \\
& & \\
\quad \mathbf{\Sigma}_f^3(x,w,\mathcal{T}_f,(v_n)_n, & & \mathbf{\Sigma}_f^{\mathrm{vrfy}}(x,\mathcal{T}_f,\textsc{Cha}, \\
\quad\quad (\textsc{Cmt}_l)_l, (\textsc{Cha}_n)_n, (\textsc{Res}_l)_l) & & \quad (\textsc{Cmt}_l)_l, (\textsc{Cha}_n)_n, (\textsc{Res}_l)_l) \\
\quad \to ((\textsc{Cha}_n)_n, (\textsc{Res}_l)_l) & \xrightarrow{(\textsc{Cha}_n)_n, (\textsc{Res}_l)_l} & \to b, \text{Return } b
\end{array}
$$

Fig. 1. Our WIPoK $\mathbf{\Sigma}_f$ on the relation R_f.

$\mathbf{\Sigma}_f^2$ and $\mathbf{\Sigma}_f^{\mathrm{vrfy}}$. Moreover, $\mathbf{\Sigma}_f^{\mathrm{vrfy}}$ has two subroutines **VrfyCha** and **VrfyRes**. Fig. 1 shows the construction of our procedure $\mathbf{\Sigma}_f$. (For the tree expressions of a boolean predicate f, see Appendix C.)

Evaluation of Satisfiability The prover $\mathcal{P}$ begins with evaluation of whether and how S satisfies f by running the evaluation algorithm $\mathbf{\Sigma}_f^{\mathrm{eval}}$. It labels each node of $\mathcal{T}$ with a value $v = 1$ (TRUE) or 0 (FALSE). For each leaf l, we label l with $v_l = 1$ if $\rho(l) \in S$ and $v_l = 0$ otherwise. (For the definition of the function ρ, see Appendix C.) For each inner node n, we label n with $v_n = v_{n_\mathrm{L}} \wedge v_{n_\mathrm{R}}$ or $v_n = v_{n_\mathrm{L}} \vee v_{n_\mathrm{L}}$ according to AND/OR evaluation of two labels of its two children $n_\mathrm{L}, n_\mathrm{R}$. The computation is executed for every node from the root to each leaf, recursively, in the following way.

$$
\begin{array}{l}
\mathbf{\Sigma}_f^{\mathrm{eval}}(\mathcal{T}, S): \\
\quad \mathcal{T}_\mathrm{L} := \mathrm{Lsub}(\mathcal{T}), \mathcal{T}_\mathrm{R} := \mathrm{Rsub}(\mathcal{T}) \\
\quad \text{If } r(\mathcal{T}) \text{ is an } \wedge\text{-node, then Return } v_{r(\mathcal{T})} := (\mathbf{\Sigma}_f^{\mathrm{eval}}(\mathcal{T}_\mathrm{L}, S) \wedge \mathbf{\Sigma}_f^{\mathrm{eval}}(\mathcal{T}_\mathrm{R}, S)) \\
\quad \text{else if } r(\mathcal{T}) \text{ is an } \vee\text{-node, then Return } v_{r(\mathcal{T})} := (\mathbf{\Sigma}_f^{\mathrm{eval}}(\mathcal{T}_\mathrm{L}, S) \vee \mathbf{\Sigma}_f^{\mathrm{eval}}(\mathcal{T}_\mathrm{R}, S)) \\
\quad \text{else if } r(\mathcal{T}) \text{ is a leaf, then Return } v_{r(\mathcal{T})} := (\rho(r(\mathcal{T})) \in_? S)
\end{array}
$$

Commitment $\mathcal{P}$ computes a commitment value for each leaf by running the algorithm $\mathbf{\Sigma}_f^1$ described in Fig. 2. Basically, $\mathbf{\Sigma}_f^1$ runs for every node from the root to each leaf, recursively. As a result, $\mathbf{\Sigma}_f^1$ generates for each leaf l a value $\textsc{Cmt}_l$; If $v_l = 1$, then $\textsc{Cmt}_l$ is computed honestly according to $\mathbf{\Sigma}^1$. Else if $v_l = 0$, then $\textsc{Cmt}_l$ is computed in the simulated way according to $\mathbf{\Sigma}^{\mathrm{sim}}$. Other values, $(\textsc{Cha}_n)_n$ and $(\textsc{Res}_l)_l)$, are needed for the simulation. Note that the distinguished symbol $*$ is used to indicate an "honest computation".

$$
\begin{array}{l}
\mathbf{\Sigma}_f^1(x,w,\mathcal{T},(v_n)_n,\textsc{Cha}): \\
\quad \mathcal{T}_\mathrm{L} := \mathrm{Lsub}(\mathcal{T}), \mathcal{T}_\mathrm{R} := \mathrm{Rsub}(\mathcal{T}) \\
\quad \text{If } \quad r(\mathcal{T}) \text{ is } \wedge\text{-node, then } \textsc{Cha}_{r(\mathcal{T}_\mathrm{L})} := \textsc{Cha}, \textsc{Cha}_{r(\mathcal{T}_\mathrm{R})} := \textsc{Cha} \\
\quad\quad \mathrm{Return}(\textsc{Cha}_{r(\mathcal{T}_\mathrm{L})}, \mathbf{\Sigma}_f^1(x,w,\mathcal{T}_\mathrm{L},(v_n)_n,\textsc{Cha}_{r(\mathcal{T}_\mathrm{L})}), \textsc{Cha}_{r(\mathcal{T}_\mathrm{R})}, \mathbf{\Sigma}_f^1(x,w,\mathcal{T}_\mathrm{R},(v_n)_n,\textsc{Cha}_{r(\mathcal{T}_\mathrm{R})})) \\
\quad \text{else if } r(\mathcal{T}) \text{ is } \vee\text{-node, then} \\
\quad\quad \text{If } \quad v_{r(\mathcal{T}_\mathrm{L})} = 1 \wedge v_{r(\mathcal{T}_\mathrm{R})} = 1, \text{ then } \textsc{Cha}_{r(\mathcal{T}_\mathrm{L})} := *, \quad \textsc{Cha}_{r(\mathcal{T}_\mathrm{R})} := * \\
\quad\quad \text{else if } v_{r(\mathcal{T}_\mathrm{L})} = 1 \wedge v_{r(\mathcal{T}_\mathrm{R})} = 0, \text{ then } \textsc{Cha}_{r(\mathcal{T}_\mathrm{L})} := *, \quad \textsc{Cha}_{r(\mathcal{T}_\mathrm{R})} \leftarrow \mathbf{\Sigma}^2(1^\lambda) \\
\quad\quad \text{else if } v_{r(\mathcal{T}_\mathrm{L})} = 0 \wedge v_{r(\mathcal{T}_\mathrm{R})} = 1, \text{ then } \textsc{Cha}_{r(\mathcal{T}_\mathrm{L})} \leftarrow \mathbf{\Sigma}^2(1^\lambda), \textsc{Cha}_{r(\mathcal{T}_\mathrm{R})} := * \\
\quad\quad \text{else if } v_{r(\mathcal{T}_\mathrm{L})} = 0 \wedge v_{r(\mathcal{T}_\mathrm{R})} = 0, \text{ then } \textsc{Cha}_{r(\mathcal{T}_\mathrm{L})} \leftarrow \mathbf{\Sigma}^2(1^\lambda), \textsc{Cha}_{r(\mathcal{T}_\mathrm{R})} := \textsc{Cha} \oplus \textsc{Cha}_{r(\mathcal{T}_\mathrm{L})} \\
\quad\quad \mathrm{Return}(\textsc{Cha}_{r(\mathcal{T}_\mathrm{L})}, \mathbf{\Sigma}_f^1(x,w,\mathcal{T}_\mathrm{L},(v_n)_n,\textsc{Cha}_{r(\mathcal{T}_\mathrm{L})}), \textsc{Cha}_{r(\mathcal{T}_\mathrm{R})}, \mathbf{\Sigma}_f^1(x,w,\mathcal{T}_\mathrm{R},(v_n)_n,\textsc{Cha}_{r(\mathcal{T}_\mathrm{R})})) \\
\quad \text{else if } r(\mathcal{T}) \text{ is a leaf, then} \\
\quad\quad \text{If } \quad v_{r(\mathcal{T})} = 1, \text{ then } \textsc{Cmt}_{r(\mathcal{T})} \leftarrow \mathbf{\Sigma}^1(x_{\rho(r(\mathcal{T}))}, w_{\rho(r(\mathcal{T}))}), \textsc{Res}_{r(\mathcal{T})} := * \\
\quad\quad \text{else if } v_{r(\mathcal{T})} = 0, \text{ then } (\textsc{Cmt}_{r(\mathcal{T})}, \textsc{Res}_{r(\mathcal{T})}) \leftarrow \mathbf{\Sigma}^{\mathrm{sim}}(x_{\rho(r(\mathcal{T}))}, \textsc{Cha}) \\
\quad\quad \mathrm{Return}(\textsc{Cmt}_{r(\mathcal{T})}, \textsc{Res}_{r(\mathcal{T})})
\end{array}
$$

Fig. 2. The subroutine $\mathbf{\Sigma}_f^1$ of our $\mathbf{\Sigma}_f$.

Challenge $\mathcal{V}$ chooses a challenge value (that is, a public coin) by $\mathbf{\Sigma}^2$.

$$\mathbf{\Sigma}^2_f(1^\lambda) : \textsc{Cha} \leftarrow \mathbf{\Sigma}^2(1^\lambda), \mathrm{Return}(\textsc{Cha})$$

Response $\mathcal{P}$ computes a response value for each leaf by running the algorithm $\mathbf{\Sigma}^3_f$ described in Fig. 3. Basically, the algorithm $\mathbf{\Sigma}^3_f$ runs for every node from the root to each leaf, recursively. As a result, $\mathbf{\Sigma}^3_f$ generates values, $(\textsc{Cha}_t)_t$ and $(\textsc{Res}_l)_l$). Note that the computations of all challenge values $(\textsc{Cha}_t)_t$ are completed (according to the "division rule" described in Section 1.1).

$$
\begin{array}{l}
\mathbf{\Sigma}^3_f(x, w, \mathcal{T}, (v_n)_n, (\textsc{Cmt}_l)_l, (\textsc{Cha}_n)_n, (\textsc{Res}_l)_l) : \\
\quad \mathcal{T}_{\mathrm{L}} := \mathrm{Lsub}(\mathcal{T}), \mathcal{T}_{\mathrm{R}} := \mathrm{Rsub}(\mathcal{T}) \\
\quad \text{If } \quad r(\mathcal{T}) \text{ is } \wedge\text{-node, then } \textsc{Cha}_{r(\mathcal{T}_{\mathrm{L}})} := \textsc{Cha}_{r(\mathcal{T})}, \textsc{Cha}_{r(\mathcal{T}_{\mathrm{R}})} := \textsc{Cha}_{r(\mathcal{T})} \\
\quad\quad \mathrm{Return}(\textsc{Cha}_{r(\mathcal{T}_{\mathrm{L}})}, \mathbf{\Sigma}^3_f(x, w, \mathcal{T}_{\mathrm{L}}, (v_n)_n, (\textsc{Cmt}_l)_l, (\textsc{Cha}_n)_n, (\textsc{Res}_l)_l), \\
\quad\quad\quad\quad \textsc{Cha}_{r(\mathcal{T}_{\mathrm{R}})}, \mathbf{\Sigma}^3_f(x, w, \mathcal{T}_{\mathrm{R}}, (v_n)_n, (\textsc{Cmt}_l)_l, (\textsc{Cha}_n)_n, (\textsc{Res}_l)_l)) \\
\quad \text{else if } r(\mathcal{T}) \text{ is } \vee\text{-node, then} \\
\quad\quad \text{If } \quad v_{r(\mathcal{T}_{\mathrm{L}})} = 1 \wedge v_{r(\mathcal{T}_{\mathrm{R}})} = 1, \text{ then } \textsc{Cha}_{r(\mathcal{T}_{\mathrm{L}})} \leftarrow \mathbf{\Sigma}^2(1^\lambda), \quad \textsc{Cha}_{r(\mathcal{T}_{\mathrm{R}})} := \textsc{Cha}_{r(\mathcal{T})} \oplus \textsc{Cha}_{r(\mathcal{T}_{\mathrm{L}})} \\
\quad\quad \text{else if } v_{r(\mathcal{T}_{\mathrm{L}})} = 1 \wedge v_{r(\mathcal{T}_{\mathrm{R}})} = 0, \text{ then } \textsc{Cha}_{r(\mathcal{T}_{\mathrm{L}})} := \textsc{Cha} \oplus \textsc{Cha}_{r(\mathcal{T}_{\mathrm{R}})}, \textsc{Cha}_{r(\mathcal{T}_{\mathrm{R}})} := \textsc{Cha}_{r(\mathcal{T}_{\mathrm{R}})} \\
\quad\quad \text{else if } v_{r(\mathcal{T}_{\mathrm{L}})} = 0 \wedge v_{r(\mathcal{T}_{\mathrm{R}})} = 1, \text{ then } \textsc{Cha}_{r(\mathcal{T}_{\mathrm{L}})} := \textsc{Cha}_{r(\mathcal{T}_{\mathrm{L}})}, \quad \textsc{Cha}_{r(\mathcal{T}_{\mathrm{R}})} := \textsc{Cha}_{r(\mathcal{T})} \oplus \textsc{Cha}_{r(\mathcal{T}_{\mathrm{L}})} \\
\quad\quad \text{else if } v_{r(\mathcal{T}_{\mathrm{L}})} = 0 \wedge v_{r(\mathcal{T}_{\mathrm{R}})} = 0, \text{ then } \textsc{Cha}_{r(\mathcal{T}_{\mathrm{L}})} := \textsc{Cha}_{r(\mathcal{T}_{\mathrm{L}})}, \quad \textsc{Cha}_{r(\mathcal{T}_{\mathrm{R}})} := \textsc{Cha}_{r(\mathcal{T}_{\mathrm{R}})} \\
\quad\quad \mathrm{Return}(\textsc{Cha}_{r(\mathcal{T}_{\mathrm{L}})}, \mathbf{\Sigma}^3_f(x, w, \mathcal{T}_{\mathrm{L}}, (v_n)_n, (\textsc{Cmt}_l)_l, (\textsc{Cha}_n)_n, (\textsc{Res}_l)_l), \\
\quad\quad\quad\quad \textsc{Cha}_{r(\mathcal{T}_{\mathrm{R}})}, \mathbf{\Sigma}^3_f(x, w, \mathcal{T}_{\mathrm{R}}, (v_n)_n, (\textsc{Cmt}_l)_l, (\textsc{Cha}_n)_n, (\textsc{Res}_l)_l)) \\
\quad \text{else if } r(\mathcal{T}) \text{ is a leaf, then} \\
\quad\quad \text{If } \quad v_{r(\mathcal{T})} = 1, \text{ then } \textsc{Res}_{r(\mathcal{T})} \leftarrow \mathbf{\Sigma}^3(x_{\rho(r(\mathcal{T}))}, w_{\rho(r(\mathcal{T}))}, \textsc{Cmt}_{r(\mathcal{T})}, \textsc{Cha}_{r(\mathcal{T})}) \\
\quad\quad \text{else if } v_{r(\mathcal{T})} = 0, \text{ then } \textsc{Res}_{r(\mathcal{T})} \leftarrow \textsc{Res}_{r(\mathcal{T})} \\
\quad\quad \mathrm{Return}(\textsc{Res}_{r(\mathcal{T})})
\end{array}
$$

Fig. 3. The subroutine $\mathbf{\Sigma}^3_f$ of our $\mathbf{\Sigma}_f$.

Verification $\mathcal{V}$ computes a decision by running from the root to each leaf, recursively, the following algorithm $\mathbf{\Sigma}^{\mathrm{vrfy}}_f$.

$$
\begin{array}{l}
\mathbf{\Sigma}^{\mathrm{vrfy}}_f(x, \mathcal{T}, \textsc{Cha}, (\textsc{Cmt}_l)_l, (\textsc{Cha}_n)_n, (\textsc{Res}_l)_l) : \\
\quad \mathrm{Return}(\mathbf{VrfyCha}(\mathcal{T}, \textsc{Cha}, (\textsc{Cha}_n)_n) \wedge \mathbf{VrfyRes}(x, \mathcal{T}, (\textsc{Cmt}_l, \textsc{Cha}_l, \textsc{Res}_l)_l))
\end{array}
$$

$$
\begin{array}{l}
\mathbf{VrfyCha}(\mathcal{T}, \textsc{Cha}, (\textsc{Cha}_n)_n) : \\
\quad \mathcal{T}_{\mathrm{L}} := \mathrm{Lsub}(\mathcal{T}), \mathcal{T}_{\mathrm{R}} := \mathrm{Rsub}(\mathcal{T}) \\
\quad \text{If } r(\mathcal{T}) \text{ is an } \wedge\text{-node} \\
\quad\quad \text{then Return } ((\textsc{Cha} =_? \textsc{Cha}_{r(\mathcal{T}_{\mathrm{L}})}) \wedge (\textsc{Cha} =_? \textsc{Cha}_{r(\mathcal{T}_{\mathrm{R}})}) \\
\quad\quad\quad \wedge \mathbf{VrfyCha}(\mathcal{T}_{\mathrm{L}}, \textsc{Cha}_{r(\mathcal{T}_{\mathrm{L}})}, (\textsc{Cha}_n)_n) \wedge \mathbf{VrfyCha}(\mathcal{T}_{\mathrm{R}}, \textsc{Cha}_{r(\mathcal{T}_{\mathrm{R}})}, (\textsc{Cha}_n)_n)) \\
\quad \text{else if } r(\mathcal{T}) \text{ is an } \vee\text{-node,} \\
\quad\quad \text{then Return } ((\textsc{Cha} =_? \textsc{Cha}_{r(\mathcal{T}_{\mathrm{L}})} \oplus \textsc{Cha}_{r(\mathcal{T}_{\mathrm{R}})}) \\
\quad\quad\quad \wedge \mathbf{VrfyCha}(\mathcal{T}_{\mathrm{L}}, \textsc{Cha}_{r(\mathcal{T}_{\mathrm{L}})}, (\textsc{Cha}_n)_n) \wedge \mathbf{VrfyCha}(\mathcal{T}_{\mathrm{R}}, \textsc{Cha}_{r(\mathcal{T}_{\mathrm{R}})}, (\textsc{Cha}_n)_n)) \\
\quad \text{else if } r(\mathcal{T}) \text{ is a leaf,} \\
\quad\quad \text{then Return } (\textsc{Cha} \in_? \textsc{ChaSp}(1^\lambda))
\end{array}
$$

$$
\begin{array}{l}
\mathbf{VrfyRes}(x, \mathcal{T}, (\textsc{Cmt}_l, \textsc{Cha}_l, \textsc{Res}_l)_l) : \\
\quad \text{For } l \in \mathrm{Leaf}(\mathcal{T}) : \text{If } \mathbf{\Sigma}^{\mathrm{vrfy}}(x_{\rho(l)}, \textsc{Cmt}_l, \textsc{Cha}_l, \textsc{Res}_l) = 0, \text{then Return } (0) \\
\quad \text{Return } (1)
\end{array}
$$

Now we have to check that $\mathbf{\Sigma}_f$ is certainly a Σ-protocol on the relation R_f.

Proposition 1 (Completeness) *Completeness holds for our $\mathbf{\Sigma}_f$.*

Proof. Suppose that $v_{r(\mathcal{T}_f)} = 1$. We show that, for every node in $\mathrm{Node}(\mathcal{T}_f)$, either $v_n = 1$ or $\textsc{Cha}_n \neq *$ holds after executing $\mathbf{\Sigma}_f^1$. The proof is by induction on the height of $\mathcal{T}_f$. The case of height 0 follows from $v_{r(\mathcal{T}_f)} = 1$ and the completeness of $\mathbf{\Sigma}$. Suppose that the case of height k holds and consider the case of height $k+1$. The construction of $\mathbf{\Sigma}_f^1$ assures the case of height $k+1$. □

Proposition 2 (Special Soundness) *Special soundness holds for our* $\mathbf{\Sigma}_f$*.*

We can construct a knowledge extractor $\mathbf{\Sigma}_f^{\mathrm{KE}}$ from a knowledge extractor $\mathbf{\Sigma}^{\mathrm{KE}}$ of the underlying Σ-protocol $\mathbf{\Sigma}$ as follows.

$\mathbf{\Sigma}_f^{\mathrm{KE}}(x, (\textsc{Cmt}_l, \textsc{Cha}_l, \textsc{Res}_l)_l, (\textsc{Cmt}_l, \textsc{Cha}'_l, \textsc{Res}'_l)_l):$
For $1 \leq j \leq \mathrm{arity}(f) : w^*_{i_j} := *$
For $l \in \mathrm{Leaf}(\mathcal{T}_f)$
If $\textsc{Cha}_l \neq \textsc{Cha}'_l$, then $w^*_{\rho(l)} \leftarrow \mathbf{\Sigma}^{\mathrm{KE}}(x_{\rho(l)}, (\textsc{Cmt}_l, \textsc{Cha}_l, \textsc{Res}_l), (\textsc{Cmt}_l, \textsc{Cha}'_l, \textsc{Res}'_l))$
else If $w^*_{\rho(l)} = *$, then $w^*_{\rho(l)} \leftarrow \{1, 0\}^*$
Return $(w^* := (w^*_{i_j})_{1 \leq j \leq \mathrm{arity}(f)})$

Then Lemma 1 assures the proposition.

Lemma 1 (Witness Extraction) *The string* w^* *output by* $\mathbf{\Sigma}_f^{KE}$ *satisfies* $(x, w^*) \in R_f$*.*

Proof. Induction on the number of all $\vee$-nodes in $\mathrm{iNode}(T_f)$. First remark that $\textsc{Cha} \neq \textsc{Cha}'$.

Suppose that all nodes in $\mathrm{iNode}(T_f)$ are $\wedge$-nodes. Then the above claim follows immediately because $\textsc{Cha}_l \neq \textsc{Cha}'_l$ holds for all leaves.

Suppose that the case of k $\vee$-nodes holds and consider the case of $k+1$ $\vee$-nodes. Look at one of the lowest height $\vee$-node and name the height and the node as h^* and n^*, respectively. Then $\textsc{Cha}_{n^*} \neq \textsc{Cha}'_{n^*}$ because all nodes with height less than h^* are $\wedge$-nodes. So at least one of children of n^*, say n^*_L, satisfies $\textsc{Cha}_{n^*_L} \neq \textsc{Cha}'_{n^*_L}$. Divide the tree $\mathcal{T}_f$ into two subtrees by cutting the branch right above n^*, and the induction hypothesis assures the claim. □

Proposition 3 (Honest-Verifier Zero-Knowledge) *Honest-verifier zero-knowledge property holds for our* $\mathbf{\Sigma}_f$*.*

Proof. This is the immediate consequence of honest-verifier zero-knowledge property of $\mathbf{\Sigma}$. That is, we can construct a polynomial-time simulator $\mathbf{\Sigma}_f^{\mathrm{sim}}$ which, on input $(\mathrm{PK}, \textsc{Cha})$, outputs commitment and response message of $\mathbf{\Sigma}_f$. □

We summarize the above results into the following theorem and corollary.

Theorem 1 ($\mathbf{\Sigma}_f$ is a Σ-protocol) *Our procedure* $\mathbf{\Sigma}_f$ *obtained from a* Σ*-protocol* $\mathbf{\Sigma}$ *on the relation* R *and a boolean predicate* f *is a* Σ*-protocol on the relation* R_f*.*

Theorem 2 ($\mathbf{\Sigma}_f$ is WIPoK) *Our* Σ*-protocol* $\mathbf{\Sigma}_f$ *is a witness-indistinguishable proof of knowledge system on the relation* R_f*.*

Proof. For a fixed statement x and two witnesses w_1 and w_2 satisfying $R(x, w_1) = R(x, w_2) = 1$ or $R(x, w_1) = R(x, w_2) = 0$, $\mathcal{P}(x, w)$ and $\mathcal{V}(x)$ of $\mathbf{\Sigma}_f$ generate transcripts $((\textsc{Cmt}_l)_l, \textsc{Cha}, (\textsc{Cha}_n)_n, (\textsc{Res}_l)_l)$ that has the same distribution. □

3.3 Our Non-interactive Witness-Indistinguishable Proofs of Knowledge on Monotone Predicates

The Fiat-Shamir transform $\mathrm{FS}(\cdot)$ can be applied to any Σ-protocol $\mathbf{\Sigma}$ ([FS86,AABN02]). Therefore, the non-interactive version of our procedure $\mathbf{\Sigma}_f$ is obtained.

Theorem 3 ($\mathrm{FS}(\mathbf{\Sigma}_f)$ is NIWIPoK) *Our* $FS(\mathbf{\Sigma}_f)$ *is a non-interactive witness-indistinguishable proof of knowledge system on the relation* R_f*. A knowledge extractor is constructed in the random oracle model.*

3.4 Discussion

As is mentioned in [CDS94], the Σ-protocol $\mathbf{\Sigma}_f$ can be considered as a proto-type of an attribute-based identification scheme. Also, the non-interactive version $\mathrm{FS}(\mathbf{\Sigma}_f)$ can be considered a proto-type of an attribute-based signature scheme. That is, $\mathbf{\Sigma}_f$ and $\mathrm{FS}(\mathbf{\Sigma}_f)$ are ABID and ABS *without collusion resistance about private secret keys*, respectively.

4 Our Attribute-Based Identification Scheme

In this section, we provide a verifier-policy attribute-based identification scheme (ABID) by combining our Σ-protocol $\mathbf{\Sigma}_f$ in Section 3 with a credential bundle of the Fiat-Shamir signatures. Our credential bundle prevents the collusion attacks about private secret keys, whereas it makes transcripts of interaction (between a fixed single prover and more than one verifiers) linkable.

4.1 Our ABID

By using a credential-bundle (see Appendix A) as a witness of our WIPoK system $\mathbf{\Sigma}_f$ in Section 3, we obtain a verifier-policy attribute-based identification scheme, `ABID` [AAHI13]. Our `ABID` is collusion resistant against collecting private secret keys. Fig. 4 shows our construction: $\mathtt{ABID} = (\mathbf{ABID}.\mathbf{Setup}, \mathbf{ABID}.\mathbf{KG}, \mathcal{P}, \mathcal{V})$.

ABID.Setup takes as input 1^λ and $\mathcal{U}$. It chooses a pair $(x_{\mathrm{mst}}, w_{\mathrm{mst}})$ at random from $R = \{(x, w)\}$ by running $\mathrm{Instance}_R(1^\lambda)$, where $|x|$ and $|w|$ are bounded by a polynomial in λ. It also chooses a hash key μ at random from the hash-key space *Hashkeysp*(λ). It returns a public key $\mathrm{PK} = (x_{\mathrm{mst}}, \mathcal{U}, \mu)$ and a master secret key $\mathrm{MSK} = (w_{\mathrm{mst}})$.

$$
\begin{aligned}
&\mathbf{ABID}.\mathbf{Setup}(1^\lambda, \mathcal{U}) : \\
&\quad (x_{\mathrm{mst}}, w_{\mathrm{mst}}) \leftarrow \mathrm{Instance}_R(1^\lambda), \mu \in_R \mathit{Hashkeysp}(\lambda) \\
&\quad \mathrm{PK} := (x_{\mathrm{mst}}, \mathcal{U}, \mu), \mathrm{MSK} := (w_{\mathrm{mst}}) \\
&\quad \mathrm{Return}(\mathrm{PK}, \mathrm{MSK})
\end{aligned}
$$

ABID.KG takes as input $\mathrm{PK}, \mathrm{MSK}, S$. It chooses a PRF key k from the key space *PRFkeysp*(λ) at random and a random string τ from $\{1, 0\}^\lambda$ at random. Then it applies the credential-bundle technique [MPR11] for each message $m_i := (\tau \parallel i), i \in S$. Here we employ the Fiat-Shamir signing algorithm $\mathrm{FS}(\mathbf{\Sigma})^{\mathrm{sign}}$ (see 2.1). It returns SK_S.

$$
\begin{aligned}
&\mathbf{ABID}.\mathbf{KG}(\mathrm{PK}, \mathrm{MSK}, S) : \\
&\quad k \in_R \mathit{PRFkeysp}(\lambda), \tau \in_R \{1, 0\}^\lambda \\
&\quad \text{For } i \in S : \\
&\quad\quad m_i := (\tau \parallel i), a_i \leftarrow \mathbf{\Sigma}^2(x_{\mathrm{mst}}, w_{\mathrm{mst}}) \\
&\quad\quad c_i \leftarrow \mathit{Hash}_\mu(a_i \parallel m_i), w_i \leftarrow \mathbf{\Sigma}^3(x_{\mathrm{mst}}, w_{\mathrm{mst}}, a_i, c_i) \\
&\quad \mathrm{SK}_S := (k, \tau, (a_i, w_i)_{i \in S}), \mathrm{Return}\ \mathrm{SK}_S.
\end{aligned}
$$

$\mathcal{P}$ and $\mathcal{V}$ takes as input $(\mathrm{PK}, \mathrm{SK}_S, f)$ and (PK, f), respectively. Then $\mathcal{P}$ and $\mathcal{V}$ execute the following interaction.

First, $\mathcal{P}$ uses the following supplementary algorithm **Supp** and a statement-generator algorithm **StmtGen**.

Supp runs for $j,\ 1 \le j \in \mathrm{arity}(f)$, and generates simulated keys (a_{i_j}, w_{i_j}) for $i_j \notin S$.

$$
\begin{aligned}
&\mathbf{Supp}(\mathrm{PK}, \mathrm{SK}_S, f) : \\
&\quad \text{For } j = 1 \text{ to } \mathrm{arity}(f) : \\
&\quad\quad \text{If } i_j \notin S, \text{ then} \\
&\quad\quad\quad m_{i_j} := (\tau \parallel i_j), c_{i_j} \leftarrow \mathit{PRF}_k(m_{i_j} \parallel 0) \\
&\quad\quad\quad (a_{i_j}, w_{i_j}) \leftarrow \mathbf{\Sigma}^{\mathrm{sim}}(x_{\mathrm{mst}}, c_{i_j}; \mathit{PRF}_k(m_{i_j} \parallel 1)) \\
&\quad \mathrm{Return}\ (a_{i_j}, w_{i_j})_{1 \le j \le \mathrm{arity}(f)}
\end{aligned}
$$

StmtGen generates, for each j, $1 \le j \in \text{arity}(f)$, a statement x_{i_j}. Note that we employ here the algorithm $\mathbf{\Sigma}^{\text{stmtgen}}$ which is associated with $\mathbf{\Sigma}$, and whose existence is assured by our assumption (see Section 2.1).

$$\begin{array}{l}
\textbf{StmtGen}(\text{PK}, \tau, (a_{i_j})_{1 \le j \le \text{arity}(f)}): \\
\quad \text{For } j = 1 \text{ to arity}(f): \\
\quad\quad m_{i_j} := (\tau \parallel i_j), c_{i_j} \leftarrow \mathit{Hash}_\mu(a_{i_j} \parallel m_{i_j}) \\
\quad\quad x_{i_j} \leftarrow \mathbf{\Sigma}^{\text{stmtgen}}(x_{\text{mst}}, a_{i_j}, c_{i_j}) \\
\quad \text{Return } (x_{i_j})_{1 \le j \le \text{arity}(f)}
\end{array}$$

Note that $(x_i, w_i) \in R$ for $i \in S$ but $\Pr[(x_i, w_i) \in R] = \text{neg}(\lambda)$ for $i \notin S$.

The above procedures are needed to input a pair of statement and witness, $(x = (x_{i_j})_{1 \le j \le \text{arity}(f)}, w = (w_{i_j})_{1 \le j \le \text{arity}(f)})$, to $\mathbf{\Sigma}^1_f$, into the prover of our procedure $\mathbf{\Sigma}_f$. Note here that $(x_{i_j}, w_{i_j}) \in R$ for any $i_j \in S$. On the other hand, $(x_{i_j}, w_{i_j}) \notin R$ for any $i_j \notin S$, without a negligible probability, $\text{neg}(\lambda)$. Note also that $\mathcal{P}$ has to send a string τ and elements $(a_{i_j})_{1 \le j \le \text{arity}(f)}$ to the verifier $\mathcal{V}$.

Second, $\mathcal{V}$ runs **StmtGen** on input PK, τ and $(a_{i_j})_{1 \le j \le \text{arity}(f)}$ to generate the statement x. Note that τ and $(a_{i_j})_{1 \le j \le \text{arity}(f)}$ can be sent as a part of the message on the first move.

Finally, $\mathcal{P}$ and $\mathcal{V}$ of our `ABID` execute the prover and the verifier of our procedure $\mathbf{\Sigma}_f$, respectively. $\mathcal{V}$ returns 1 or 0 according to the return of the verifier of $\mathbf{\Sigma}_f$.

$$\begin{array}{ll}
\textbf{ABID}.\textbf{Setup}(1^\lambda, \mathcal{U}): & \textbf{ABID}.\textbf{KG}(\text{PK}, \text{MSK}, S): \\
\quad (x_{\text{mst}}, w_{\text{mst}}) \leftarrow \text{Instance}_R(1^\lambda) & \quad k \in_R \mathit{PRFkeysp}(\lambda), \tau \in_R \{1,0\}^\lambda \\
\quad \mu \in_R \mathit{Hashkeysp}(\lambda) & \quad \text{For } i \in S \\
\quad \text{PK} := (x_{\text{mst}}, \mathcal{U}, \mu), \text{MSK} := (w_{\text{mst}}) & \quad\quad m_i := (\tau \parallel i), a_i \leftarrow \mathbf{\Sigma}^1(x_{\text{mst}}, w_{\text{mst}}) \\
\quad \text{Return}(\text{PK}, \text{MSK}) & \quad\quad c_i \leftarrow \mathit{Hash}_\mu(a_i \parallel m_i), w_i \leftarrow \mathbf{\Sigma}^3(x_{\text{mst}}, w_{\text{mst}}, a_i, c_i) \\
 & \quad \text{SK}_S := (k, \tau, (a_i, w_i)_{i \in S}) \\
 & \quad \text{Return SK}_S
\end{array}$$

$$\begin{array}{lcl}
\mathcal{P}(\text{PK}, \text{SK}_S, f): & & \mathcal{V}(\text{PK}, f): \\
\quad \textbf{Supp}(\text{PK}, \text{SK}_S, f) \to (a_{i_j}, w_{i_j})_j & & \\
\quad w := (w_{i_j})_j & & \\
\quad \textbf{StmtGen}(\text{PK}, \tau, (a_{i_j})_j) & & \\
\to (x_{i_j})_j =: x & & \\
 & & \\
\mathbf{\Sigma}^{\text{eval}}_f(\mathcal{T}_f, S) \to (v_n)_n & & \\
\text{If } v_{r(\mathcal{T}_f)} \neq 1, \text{ then abort} & & \\
\text{else } \textsc{Cha}_{r(\mathcal{T}_f)} := * & & \\
 & & \\
\quad \mathbf{\Sigma}^1_f(x, w, \mathcal{T}_f, (v_n)_n, \textsc{Cha}_{r(\mathcal{T}_f)}) & & \\
\to ((\textsc{Cmt}_l)_l, (\textsc{Cha}_n)_n, (\textsc{Res}_l)_l) & \tau, (a_{i_j})_j, (\textsc{Cmt}_l)_l & \textbf{StmtGen}(\text{PK}, \tau, (a_{i_j})_j) \\
 & \longrightarrow & \to (x_{i_j})_j =: x \\
 & & \\
\quad \textsc{Cha}_{r(\mathcal{T}_f)} := \textsc{Cha} & \textsc{Cha} & \textsc{Cha} \leftarrow \mathbf{\Sigma}^2_f(1^\lambda) \\
 & \longleftarrow & \\
\quad \mathbf{\Sigma}^3_f(x, w, \mathcal{T}_f, (v_n)_n, & & \\
\quad\quad (\textsc{Cmt}_l)_l, (\textsc{Cha}_n)_n, (\textsc{Res}_l)_l) & & \mathbf{\Sigma}^{\text{vrfy}}_f(x, \mathcal{T}_f, \textsc{Cha}, \\
\to ((\textsc{Cha}_n)_n, (\textsc{Res}_l)_l) & (\textsc{Cha}_n)_n, (\textsc{Res}_l)_l & \quad (\textsc{Cmt}_l)_l, (\textsc{Cha}_n)_n, (\textsc{Res}_l)_l) \\
 & \longrightarrow & \to b, \text{ Return } b
\end{array}$$

Fig. 4. The scheme of our `ABID`.

4.2 Security of Our ABID

Theorem 4 (Concurrent Security) *If the employed signature scheme $FS(\mathbf{\Sigma})$ is existentially unforgeable against chosen-message attacks, then our `ABID` is secure against concurrent attacks. More precisely, for any PPT algorithm $\mathcal{A}$, there exists a PPT algorithm $\mathcal{F}$ which satisfies the following inequality (neg$(\cdot)$ means a negligible function).*

$$\mathbf{Adv}^{ca}_{\mathit{ABID},\mathcal{A}}(\lambda, \mathcal{U}) \le (\mathbf{Adv}^{\mathit{euf\text{-}cma}}_{FS(\mathbf{\Sigma}),\mathcal{F}}(\lambda))^{1/2} + \mathit{neg}(\lambda).$$

Note that FS($\mathbf{\Sigma}$) is only known to be secure in the random oracle model.

Proof. Employing any given adversary $\mathcal{A}$ as subroutine, we construct a signature forger $\mathcal{F}$ on FS($\mathbf{\Sigma}$) as follows. $\mathcal{F}$ can answer to $\mathcal{A}$'s key-extraction queries for a secret key SK_S because $\mathcal{F}$ can query his signing oracle about $(m_i := \tau \parallel i; i \in S)$, where $\mathcal{F}$ chooses τ at random. $\mathcal{F}$ can simulate any concurrent prover with SK_S which $\mathcal{A}$ invokes because $\mathcal{F}$ can generate SK_S in the above way. After the learning phase, $\mathcal{A}$ begins the impersonation phase. $\mathcal{F}$ simulates a verifier with which $\mathcal{A}$ interacts as a prover. After a completion of a verification, $\mathcal{F}$ rewinds $\mathcal{A}$ to the timing right after receiving a commitment. By running $\mathbf{\Sigma}_f^{\mathrm{KE}}$, $\mathcal{F}$ obtains a witness w^*, a set of attributes S^* and a target access formula f^* with $f^*(S^*) = 1$, Finally, $\mathcal{F}$ succeeds in making at least one valid signature (a_i, w_i) for $i \in S^*$ due to $f^*(S^*) = 1$ and the special soundness. By the Reset Lemma [BP02], the advantage $\mathbf{Adv}^{\mathrm{ca}}_{\mathtt{ABID},\mathcal{A}}(\lambda,\mathcal{U})$ is reduced to $\mathbf{Adv}^{\text{euf-cma}}_{\mathrm{FS}(\mathbf{\Sigma}),\mathcal{F}}(\lambda)$ with a loss of exponent by 1/2. □

Corollary 1 (Passive Security) *If the employed signature scheme FS($\mathbf{\Sigma}$) is existentially unforgeable against chosen-message attacks, then our* `ABID` *is secure against passive attacks. More precisely, for any PPT algorithm $\mathcal{A}$, there exists a PPT algorithm $\mathcal{F}$ which satisfies the following inequality (neg($\cdot$) means a negligible function).*

$$\mathbf{Adv}^{pa}_{ABID,\mathcal{A}}(\lambda,\mathcal{U}) \leq (\mathbf{Adv}^{\textit{euf-cma}}_{FS(\mathbf{\Sigma}),\mathcal{F}}(\lambda))^{1/2} + neg(\lambda).$$

Proof. This is deduced by the observation that $\mathbf{Adv}^{\mathrm{pa}}_{\mathtt{ABID},\mathcal{A}}(\lambda,\mathcal{U}) \leq \mathbf{Adv}^{\mathrm{ca}}_{\mathtt{ABID},\mathcal{A}}(\lambda,\mathcal{U})$, which is from the definitions of both attacks in Section D.1.

More on Reduction of Concurrent Security We mean by "a number theoretic problem" the discrete-logarithm problem or the RSA-inverse problem ([BP02]). There exists the following (very loose) security reduction to a number theoretic problem.

$$\mathbf{Adv}^{\mathrm{ca}}_{\mathtt{ABID},\mathcal{A}}(\lambda,\mathcal{U}) \leq q_H^{1/2}(\mathbf{Adv}^{\mathrm{num.prob.}}_{\mathbf{Grp},\mathcal{S}}(\lambda))^{1/4} + \mathrm{neg}(\lambda). \tag{1}$$

Here we denote q_H as the maximum number of hash queries issued by forger $\mathcal{F}$ on FS($\mathbf{\Sigma}$) in the random oracle model. This is because (as is in Section 2.1) we can reduce the advantage $\mathbf{Adv}^{\text{euf-cma}}_{\mathrm{FS}(\mathbf{\Sigma}),\mathcal{F}}(\lambda)$ to the advantage $\mathbf{Adv}^{\mathrm{pa}}_{\mathbf{\Sigma},\mathcal{B}}(\lambda)$ of passive security of the underlying Σ-protocol $\mathbf{\Sigma}$, in the random oracle model, with a loss factor q_H. Applying the Reset Lemma [BP02], we can reduce $\mathbf{Adv}^{\mathrm{pa}}_{\mathbf{\Sigma},\mathcal{B}}(\lambda)$ to the advantage $\mathbf{Adv}^{\mathrm{num.prob.}}_{\mathbf{Grp},\mathcal{S}}(\lambda)$ of a PPT solver $\mathcal{S}$ of a number theoretic problem, with a loss of exponent by 1/2.

5 Our Attribute-Based Signature Scheme

In this section, we provide an attribute-based signature scheme (ABS) by applying the Fiat-Shamir transform (Section 2.1) to our `ABID` in Section 4. Our ABS is collusion resistant against collecting private secret keys, and EUF-CMA secure in the random oracle model. We note that our ABS signatures are linkable when they are generated with the same private secret key.

5.1 Our ABS

By applying FS($\cdot$) to our `ABID` in Section 4.1, we obtain an ABS scheme, `ABS`. Fig. 5 shows our construction: `ABS` = (**ABS**.**Setup**, **ABS**.**KG**, **ABS**.**Sign**, **ABS**.**Vrfy**).

ABS.**Setup** and **ABS**.**KG** are the same as **ABID**.**Setup** and **ABID**.**KG**, respectively.

ABS.**Sign** takes as input $\mathrm{PK}, \mathrm{SK}_S$ and (m, f). It runs **Supp**($\mathrm{PK}, \mathrm{SK}_S, f$), **StmtGen** and the prover of our procedure $\mathbf{\Sigma}_f$ with a challenge string $\textsc{Cha}$ obtained by hashing the string $(x \parallel (\textsc{Cmt}_l)_l \parallel m)$. It returns a signature

$$\sigma = (\tau, (a_{i_j})_j, (\textsc{Cmt}_l)_l, (\textsc{Cha}_n)_n, (\textsc{Res}_l)_l).$$

ABS.**Vrfy** takes as input PK, (m, f) and σ. It utilizes **StmtGen** and $\mathbf{\Sigma}_f^{\mathrm{vrfy}}$ to check validity of the pair (m, f) and the signature σ under the public key PK.

ABS.Setup$(1^\lambda, \mathcal{U})$:

$(x_{\mathrm{mst}}, w_{\mathrm{mst}}) \leftarrow \mathrm{Instance}_R(1^\lambda)$
$\mu \in_R$ *Hashkeysp*(λ)
$\mathrm{PK} := (x_{\mathrm{mst}}, \mathcal{U}, \mu), \mathrm{MSK} := (w_{\mathrm{mst}})$
Return(PK, MSK)

ABS.KG(PK, MSK, S):

$k \in_R$ *PRFkeysp*$(\lambda), \tau \in_R \{1,0\}^\lambda$
For $i \in S$
  $m_i := (\tau \parallel i), a_i \leftarrow \mathbf{\Sigma}^1(x_{\mathrm{mst}}, w_{\mathrm{mst}})$
  $c_i \leftarrow \mathit{Hash}_\mu(a_i \parallel m_i), w_i \leftarrow \mathbf{\Sigma}^3(x_{\mathrm{mst}}, w_{\mathrm{mst}}, a_i, c_i)$
$\mathrm{SK}_S := (k, \tau, (a_i, w_i)_{i \in S})$
Return SK_S

ABS.Sign$(\mathrm{PK}, \mathrm{SK}_S, (m, f))$:

Supp$(\mathrm{PK}, \mathrm{SK}_S, f) \to (a_{i_j}, w_{i_j})_j$
$w := (w_{i_j})_j$
StmtGen$(\mathrm{PK}, \tau, (a_{i_j})_j)$
$\to (x_{i_j})_j =: x$

$\mathbf{\Sigma}_f^{\mathrm{eval}}(\mathcal{T}_f, S) \to (v_n)_n$
If $v_{r(\mathcal{T}_f)} \neq 1$, then abort
else $\textsc{Cha}_{r(\mathcal{T}_f)} := *$

$\mathbf{\Sigma}_f^1(x, w, \mathcal{T}_f, (v_n)_n, \textsc{Cha}_{r(\mathcal{T}_f)})$
$\to ((\textsc{Cmt}_l)_l, (\textsc{Cha}_n)_n, (\textsc{Res}_l)_l)$

$\textsc{Cha} \leftarrow \mathit{Hash}_\mu(x \parallel (\textsc{Cmt}_l)_l \parallel m)$
$\textsc{Cha}_{r(\mathcal{T}_f)} := \textsc{Cha}$

$\mathbf{\Sigma}_f^3(x, w, \mathcal{T}_f, (v_n)_n,$
$(\textsc{Cmt}_l)_l, (\textsc{Cha}_n)_n, (\textsc{Res}_l)_l)$
$\to ((\textsc{Cha}_n)_n, (\textsc{Res}_l)_l)$

Return $\sigma := (\tau, (a_{i_j})_j,$
$(\textsc{Cmt}_l)_l, (\textsc{Cha}_n)_n, (\textsc{Res}_l)_l)$

ABS.Vrfy$(\mathrm{PK}, (m, f), \sigma := (\tau, (a_{i_j})_j,$
$(\textsc{Cmt}_l)_l, (\textsc{Cha}_n)_n, (\textsc{Res}_l)_l))$:

StmtGen$(\mathrm{PK}, \tau, (a_{i_j})_j)$
$\to (x_{i_j})_j =: x$

$\textsc{Cha} \leftarrow \mathit{Hash}_\mu(x \parallel (\textsc{Cmt}_l)_l \parallel m)$

$\mathbf{\Sigma}_f^{\mathrm{vrfy}}(x, \mathcal{T}_f, \textsc{Cha},$
$(\textsc{Cmt}_l)_l, (\textsc{Cha}_n)_n, (\textsc{Res}_l)_l)$
$\to b$, Return b

Fig. 5. The scheme of our `ABS`.

5.2 Security of Our ABS

Applying the standard technique in the work of Abdalla et al. [AABN02] shows that the security of our ABS is equivalent to the security of an attribute-based identification scheme, ABID, against passive attacks, where our ABID is obtained by combining our Σ-protocol $\mathbf{\Sigma}_f$ with the credential-bundle scheme of the Fiat-Shamir signature $\mathrm{FS}(\mathbf{\Sigma})$.

Theorem 5 (Unforgeability) *Our attribute-based signature scheme ABS is existentially unforgeable against chosen-message attacks in the random oracle model, based on the passive security of ABID. More precisely, let q_H denote the maximum number of hash queries issued by a forger $\mathcal{F}$ on ABS. Then, for any PPT algorithm $\mathcal{F}$, there exists a PPT algorithm $\mathcal{B}$ which satisfies the following inequality (neg$(\cdot)$ means a negligible function).*

$$\mathbf{Adv}^{euf\text{-}cma}_{ABS,\mathcal{F}}(\lambda,\mathcal{U}) \le q_H \mathbf{Adv}^{pa}_{ABID,\mathcal{B}}(\lambda,\mathcal{U}) + neg(\lambda). \tag{2}$$

Proof. First, our ABS is considered to be obtained by applying the Fiat-Shamir transform to our ABID. This is because, in the first message of our ABID, the tag τ and the elements $(a_{i_j})_{1\le j\le \mathrm{arity}(f)}$ are fixed even when the 3-move protocol is repeated between the prover $\mathcal{P}$ with a secret key SK_S and the verifier $\mathcal{V}$ with an access structure f.

As is discussed in Section 2.1, we can reduce the advantage $\mathbf{Adv}^{\text{euf-cma}}_{\mathtt{ABS},\mathcal{F}}(\lambda,\mathcal{U})$ to the advantage $\mathbf{Adv}^{\text{pa}}_{\mathtt{ABID},\mathcal{B}}(\lambda,\mathcal{U})$ of passive security of the underlying ABID, in the random oracle model, with a loss factor q_H. This is because $\mathcal{B}$ can simulate key-extraction queries of $\mathcal{F}$ perfectly with the aid of the key-generation oracle of $\mathcal{B}$. □

More on Reduction of Unforgeability Let q_H denote the maximum number of hash queries issued by a forger $\mathcal{F}$ on ABS and a forger $\mathcal{F}'$ on $\mathrm{FS}(\mathbf{\Sigma})$. Combining the inequality (2) with the inequalities (4) and (1) in Section D and Section 4, we obtain the following (very loose) security reduction of advantages.

$$\mathbf{Adv}^{\text{euf-cma}}_{\mathtt{ABS},\mathcal{F}}(\lambda,\mathcal{U}) \le q_H^{3/2}(\mathbf{Adv}^{\text{num.prob.}}_{\mathbf{Grp},\mathcal{S}}(\lambda))^{1/4} + \mathrm{neg}(\lambda).$$

Attribute Privacy Our ABS does not have attribute privacy defined in Section E.2 because of its linkability; that is, the constant components $\tau, (a_{i_j})_j$ make two signatures linkable when they are generated with the same private secret key.

6 Attribute-Based Two-Tier Signature: Syntax

In this section, we define a syntax of attribute-based two-tier signature scheme (ABTTS) [AAS15] according to the syntax of the two-tier signature scheme [BS07]. Then, we define a chosen-message attack on ABTTS by which an adversary makes an existential forgery, and define the existential unforgeability security against chosen-message attacks (EUF-CMA) .

An attribute-based two-tier signature scheme, ABTTS, consists of five PPT algorithms: ABTTS = (**ABTTS**.**Setup**, **ABTTS**.**PKG**, **ABTTS**.**SKG**, **ABTTS**.**Sign**, **ABTTS**.**Vrfy**).

ABTTS.**Setup**$(1^\lambda,\mathcal{U}) \to (\mathrm{MSK},\mathrm{PK})$. This PPT algorithm for setting up master and public keys takes as input the security parameter 1^λ and the attribute universe $\mathcal{U}$. It returns a master secret key MSK and a public key PK.

ABTTS.**PKG**$(\mathrm{MSK},\mathrm{PK},S) \to \mathrm{SK}_S$. This PPT algorithm for primary-key generation takes as input the master secret key MSK, the public key PK and an attribute set $S \subset \mathcal{U}$. It returns a secret key SK_S that corresponds to S.

ABTTS.**SKG**$(\mathrm{MSK},\mathrm{PK},\mathrm{SK}_S,f) \to (\mathrm{SSK}_{S,f},\mathrm{SPK}_f)$. This PPT algorithm for secondary-key generation takes as input the master secret key MSK, the public key PK, a secret key SK_S and an access formula f. It returns a pair $(\mathrm{SSK}_{S,f},\mathrm{SPK}_f)$ of a secondary secret key and a secondary public key.

ABTTS.**Sign**$(\mathrm{PK},\mathrm{SK}_S,\mathrm{SSK}_{S,f},\mathrm{SPK}_f,(m,f)) \to \sigma$. This PPT algorithm for signing takes as input the public key PK, a secret key SK_S, a secondary secret key $\mathrm{SSK}_{S,f}$, a secondary public key SPK_f and a pair (m,f) of a message $m \in \{1,0\}^*$ and an access formula f. It returns a signature σ.

ABTTS.**Vrfy**$(\mathrm{PK},\mathrm{SPK}_f,(m,f),\sigma) \to 1/0$. This deterministic polynomial-time algorithm for verification takes as input the public key PK, a secondary public key SPK_f, a pair (m,f) of a message and an access formula and a signature σ. It returns a decision 1 or 0. When it is 1, we say that $((m,f),\sigma)$ is *valid*. When it is 0, we say that $((m,f),\sigma)$ is *invalid*.

We demand correctness of ABTTS that, for any λ, any $\mathcal{U}$, any $S \subset \mathcal{U}$ and any (m,f) such that $f(S)=1$, $\Pr[(\mathrm{MSK},\mathrm{PK}) \leftarrow \mathbf{ABTTS.Setup}(1^\lambda,\mathcal{U})$, $\mathrm{SK}_S \leftarrow \mathbf{ABTTS.PKG}(\mathrm{MSK},\mathrm{PK},S)$, $(\mathrm{SSK}_{S,f},\mathrm{SPK}_f)$
$\leftarrow \mathbf{ABTTS.SKG}(\mathrm{MSK},\mathrm{PK},\mathrm{SK}_S,f)$, $\sigma \leftarrow \mathbf{ABTTS.Sign}(\mathrm{SK}_S,\mathrm{PK},\mathrm{SSK}_{S,f},\ \mathrm{SPK}_f,(m,f))$,
$b \leftarrow \mathbf{ABS.Vrfy}(\mathrm{PK},\mathrm{SPK}_f,(m,f),\sigma) : b=1] = 1$.

6.1 Chosen-Message Attack on ABTTS and Security Definition

A PPT adversary $\mathcal{F}$ tries to make a forgery $((m^*, f^*), \sigma^*)$ that consists of a message, a target access formula and a signature. The following experiment $\mathbf{Exp}^{\text{euf-cma}}_{\mathtt{ABTTS},\mathcal{F}}(1^\lambda, \mathcal{U})$ of a forger $\mathcal{F}$ defines the *chosen-message attack on ABTTS making an existential forgery.*

$$\begin{aligned}
&\mathbf{Exp}^{\text{euf-cma}}_{\mathtt{ABTTS},\mathcal{F}}(1^\lambda, \mathcal{U}):\\
&\quad (\mathrm{PK}, \mathrm{MSK}) \leftarrow \mathbf{ABTTS}.\mathbf{Setup}(1^\lambda, \mathcal{U})\\
&\quad ((m^*, f^*), \sigma^*) \leftarrow \mathcal{F}^{\mathcal{PKG}(\mathrm{MSK},\mathrm{PK},\cdot),\mathcal{SPKG}(\cdot,\cdot),\mathcal{SIGN}(\mathrm{PK},\mathrm{SK}_\cdot,\mathrm{SSK}_{S,f},\mathrm{SPK}_f(\cdot,\cdot))}(\mathrm{PK})\\
&\quad \text{If } \mathbf{ABTTS}.\mathbf{Vrfy}(\mathrm{PK}, \mathrm{SPK}_f, (m^*, f^*), \sigma^*) = 1\\
&\quad\quad \text{then Return W}\textsc{in}\text{ else Return L}\textsc{ose}
\end{aligned}$$

In the experiment, $\mathcal{F}$ issues key-extraction queries to its oracle $\mathcal{PKG}$, secondary public key queries to its oracle $\mathcal{SPKG}$ and signing queries to its oracle $\mathcal{SIGN}$. Giving an attribute set S_i, $\mathcal{F}$ queries $\mathcal{PKG}(\mathrm{MSK}, \mathrm{PK}, \cdot)$ for a secret key SK_{S_i}. Giving an attribute set S and an access formula f, $\mathcal{F}$ queries $\mathcal{SPKG}(\cdot, \cdot)$ for a secondary public key SPK_f. Giving an attribute set S_j and a pair (m_j, f_j) of a message and an access formula, $\mathcal{F}$ queries $\mathcal{SIGN}(\mathrm{PK}, \mathrm{SK}_\cdot, \mathrm{SSK}_{\cdot,\cdot}, \mathrm{SPK}_\cdot, (\cdot, \cdot))$ for a valid signature σ_j when $f_j(S_j) = 1$. As a rule of the two-tier signature, each published secondary public key SPK_f can be used only once to obtain a signature from $\mathcal{SIGN}$ [BS07].

f^* is called a *target access formula* of $\mathcal{F}$. Here we consider the *adaptive target* case in the sense that $\mathcal{F}$ is allowed to choose f^* after seeing PK and issuing three queries. Two restrictions are imposed on $\mathcal{F}$ concerning f^*. For all key-extraction queries (i.e. for $^\forall i$), $f^*(S_i) = 0$. For all signing queries (for $^\forall j$), $f^*(S_j) = 0$ or $m^* \neq m_j$. The numbers of key-extraction queries and signing queries are at most q_k and q_s, respectively, which are bounded by a polynomial in λ. The *advantage* of $\mathcal{F}$ over `ABTTS` is defined as

$$\mathbf{Adv}^{\text{euf-cma}}_{\mathtt{ABTTS},\mathcal{F}}(\lambda, \mathcal{U}) \stackrel{\text{def}}{=} \Pr[\mathbf{Exp}^{\text{euf-cma}}_{\mathtt{ABTTS},\mathcal{F}}(1^\lambda, \mathcal{U}) \text{ returns W}\textsc{in}].$$

Definition 2 (EUF-CMA of ABTTS) *`ABTTS` is called* existentially unforgeable against chosen-message attacks *if, for any PPT $\mathcal{F}$ and any $\mathcal{U}$, $\mathbf{Adv}^{\textit{euf-cma}}_{\mathtt{ABTTS},\mathcal{F}}(\lambda, \mathcal{U})$ is negligible in λ.*

Then we define *attribute privacy* of ABTTS.

Definition 3 (Attribute Privacy of ABTTS) *`ABTTS` is called to have* attribute privacy *if, for all $(PK, MSK) \leftarrow \mathbf{ABTTS}.\mathbf{Setup}(1^\lambda, \mathcal{U})$, for all message m, for all attribute sets S_1, S_2, for all primary secret keys $SK_{S_1} \leftarrow \mathbf{ABTTS}.\mathbf{PKG}(PK, MSK, S_1)$, $SK_{S_2} \leftarrow \mathbf{ABTTS}.\mathbf{PKG}(PK, MSK, S_2)$, for all secondary secret keys $(SSK_{S_1,f}, SPK_f) \leftarrow \mathbf{ABTTS}.\mathbf{SKG}(MSK, PK, SK_{S_1}, f)$, $(SSK_{S_2,f}, SPK_f) \leftarrow \mathbf{ABTTS}.\mathbf{SKG}(MSK, PK, SK_{S_2}, f)$ and for all access formula f such that $[f(S_1) = 1 \wedge f(S_2) = 1]$, two distributions $\sigma_1 \leftarrow \mathbf{ABTTS}.\mathbf{Sign}(PK, SK_{S_1}, SSK_{S_1,f}, SPK_f, (m, f))$ and $\sigma_2 \leftarrow \mathbf{ABTTS}.\mathbf{Sign}(PK, SK_{S_2}, SSK_{S_2,f}, SPK_f, (m, f))$ are identical.*

7 Our Attribute-Based Two-Tier Signature Scheme

In this section, we provide an attribute-based two-tier signature scheme (ABTTS) [AAS15] by applying the two-tier framework in Section 6 to our `ABID` in Section 4.1. Collusion resistance against collecting private secret keys is assured by the issuer of second secret / public keys. Attribute privacy is assured by the witness-indistinguishability of the underlying procedure $\mathbf{\Sigma}_f$.

7.1 Our ABTTS

By applying the two-tier framework in Section 6 to our `ABID` in Section 4.1, we obtain the ABTTS scheme. Our ABTTS enjoys collusion resistance, EUF-CMA security and attribute privacy. The critical point is that the secondary key generator **ABTTS.SKG** can issue a legitimate statement x for the procedure $\mathbf{\Sigma}_f$. Hence our ABTTS can avoid collusion attacks on secret keys.

Fig. 6 shows our construction: `ABTTS` = (**ABTTS.Setup**, **ABTTS.PKG**, **ABTTS.SKG**, **ABTTS.Sign**, **ABTTS.Vrfy**).

ABTTS.Setup and **ABTTS.PKG** are the same as **ABID.Setup** and **ABID.KG** in Section 4, respectively.

ABTTS.SKG$(\mathrm{MSK}, \mathrm{PK}, \mathrm{SK}_S, f)$ takes as input MSK, PK, SK_S and f. It uses a supplementary algorithm **Supp** and a statement-generator algorithm **StmtGen** to generate a statement x and a corresponding witness w. The usage is the same as in our ABID in Section 4. Then, it runs the prover $\mathcal{P}$ according to $\mathbf{\Sigma}_f$ to generate the first message as

$$((\textsc{Cmt}_l)_l, st) \leftarrow \mathbf{\Sigma}_f^1(x, w, \mathcal{T}_f, (v_n)_n, \textsc{Cha}_{r(\mathcal{T}_f)}).$$

Then it puts $\mathrm{SSK}_{S,f} := (w, (\textsc{Cmt}_l)_l \parallel st)$ and $\mathrm{SPK}_f := (x, (\textsc{Cmt}_l)_l)$. Here st denotes the inner state of $\mathcal{P}$. It returns $\mathrm{SSK}_{S,f}$ and SPK_f. Note that the secondary public key SPK_f should be issued by a key-issuing center [BS07].
ABTTS.Sign$(\mathrm{PK}, \mathrm{SK}_S, \mathrm{SSK}_{S,f}, \mathrm{SPK}_f, (m, f)) \to \sigma$. Given PK, SK_S, the secondary secret key $\mathrm{SSK}_{S,f}$, the secondary public key SPK_f, and a pair (m, f) of a message and an access formula f, it computes a challenge Cha by hashing the string $(\textsc{Cmt}_l)_l \parallel m$. Then, it runs the prover $\mathcal{P}$ according to $\mathbf{\Sigma}_f$ as

$$((\textsc{Cha}_n)_n, (\textsc{Res}_l)_l) \leftarrow \mathbf{\Sigma}_f^3(x, w, \mathcal{T}_f, (v_n)_n, (\textsc{Cmt}_l)_l, (\textsc{Cha}_n)_n, (\textsc{Res}_l)_l; st)$$

Finally, it returns a signature

$$\sigma := ((\textsc{Cha}_n)_n, (\textsc{Res}_l)_l).$$

ABTTS.Vrfy$(\mathrm{PK}, \mathrm{SPK}_f, (m, f), \sigma) \to 1/0$. Given PK, the secondary public key SPK_f, a pair (m, f) and a signature σ, it computes a challenge Cha by hashing the string $(\textsc{Cmt}_l)_l \parallel m$. Then, it runs the verifier $\mathcal{V}$ according to $\mathbf{\Sigma}_f$ as

$$1 \text{ or } 0 \leftarrow \mathbf{\Sigma}_f^{\mathrm{vrfy}}(x, \mathcal{T}_f, \textsc{Cha}, (\textsc{Cmt}_l)_l, (\textsc{Cha}_n)_n, (\textsc{Res}_l)_l).$$

It returns 1 or 0 accordingly.

7.2 Security of Our ABTTS

The security of our ABTTS is derived from the security of the underlying attribute-based identification scheme, ABID, against concurrent attacks [BS07].

Theorem 6 (Unforgeability) *Our attribute-based two-tier signature scheme ABTTS is existentially unforgeable against chosen-message attacks in the standard model, based on the concurrent security of ABID. More precisely, let q_H denote the maximum number of hash queries issued by a forger $\mathcal{F}$ on ABTTS. Then, for any PPT algorithm $\mathcal{F}$, there exists a PPT algorithm $\mathcal{B}$ which satisfies the following inequality (neg$(\cdot)$ means a negligible function).*

$$\mathbf{Adv}^{\text{euf-cma}}_{\text{ABTTS},\mathcal{F}}(\lambda, \mathcal{U}) \leq q_H \mathbf{Adv}^{\text{ca}}_{\text{ABID},\mathcal{B}}(\lambda, \mathcal{U}) + \text{neg}(\lambda). \tag{3}$$

Proof. We just note that the same argument in [BS07] is applied to our ABTTS. □

Theorem 7 (Attribute Privacy) *Our attribute-based two-tier signature scheme ABTTS has attribute privacy.*

Proof. A valid signature of ABTTS, $\sigma := ((\textsc{Cha}_n)_n, (\textsc{Res}_l)_l)$, is a part of a valid proof of $\mathbf{\Sigma}_f$. According to the witness-indistinguishability of $\mathbf{\Sigma}_f$, the attribute privacy holds. □

8 Conclusions

We provided a concrete procedure $\mathbf{\Sigma}_f$ of a Σ-protocol of the WIPoK system on monotone predicates. Our $\mathbf{\Sigma}_f$ can be considered as a proto-type of an attribute-based identification scheme, and also, $\mathrm{FS}(\mathbf{\Sigma}_f)$ can be considered a proto-type of an attribute-based signature scheme [CDS94], without collusion resistance on private secret keys. Then we provided a generic construction of an attribute-based identification scheme ABID, an attribute-based signature scheme ABS, and an attribute-based two-tier signature scheme ABTTS. It must be noted that our ABS does not possess attribute-privacy and our ABTTS assumes the secondary public key in the two-tier framework [BS07].

Our procedure $\mathbf{\Sigma}_f$ of WIPoK on any monotone predicate serves as a building block of the Σ-protocol of the ABS scheme [Her14] that is pairing-free.

Acknowledgements We appreciate sincere comments from Javier Herranz via e-mail communication [Her16b] on the topic in this paper. We would like to thank to Keita Emura and Takahiro Matsuda for their sincere comments and encouragements on the construction of attribute-based signature schemes. We would like to thank to Shingo Hasegawa and Masayuki Fukumitsu for their sincere comments on the construction of the Σ-protocol on monotone predicates.

ABTTS.Setup$(1^\lambda, \mathcal{U})$:
$(x_{\text{mst}}, w_{\text{mst}}) \leftarrow \text{Instance}_R(1^\lambda)$
$\mu \leftarrow \mathit{Hashkeysp}(\lambda)$
$\text{PK} := (x_{\text{mst}}, \mathcal{U}, \mu), \text{MSK} := (w_{\text{mst}})$
Return(PK, MSK)

ABTTS.PKG(PK, MSK, S):
$k \leftarrow \mathit{PRFkeysp}(\lambda), \tau \leftarrow \{1,0\}^\lambda$
For $i \in S$
$m_i := (\tau \parallel i), a_i \leftarrow \mathbf{\Sigma}^1(x_{\text{mst}}, w_{\text{mst}})$
$c_i \leftarrow \mathit{Hash}_\mu(a_i \parallel m_i), w_i \leftarrow \mathbf{\Sigma}^3(x_{\text{mst}}, w_{\text{mst}}, a_i, c_i)$
$\text{SK}_S := (k, \tau, (a_i, w_i)_{i \in S})$
Return SK_S

ABTTS.SKG$(\text{MSK}, \text{PK}, \text{SK}_S, f) \rightarrow (\text{SSK}_{S,f}, \text{SPK}_f)$:
Supp$(\text{PK}, \text{SK}_S, f) \rightarrow (a_{i_j}, w_{i_j})_j$
$w := (w_{i_j})_j$
StmtGen$(\text{PK}, \tau, (a_{i_j})_j)$
$\rightarrow (x_{i_j})_j =: x$

$\mathbf{\Sigma}_f^{\text{eval}}(\mathcal{T}_f, S) \rightarrow (v_n)_n$
If $v_{r(\mathcal{T}_f)} \neq 1$, then abort
else $\text{CHA}_{r(\mathcal{T}_f)} := *$

$\mathbf{\Sigma}_f^1(x, w, \mathcal{T}_f, (v_n)_n, \text{CHA}_{r(\mathcal{T}_f)})$
$\rightarrow ((\text{CMT}_l)_l, (\text{CHA}_n)_n, (\text{RES}_l)_l; \mathit{st})$

$\text{SSK}_{S,f} := (w, (\text{CMT}_l)_l \parallel \mathit{st})$
$\text{SPK}_f := (x, (\text{CMT}_l)_l)$
Return$(\text{SSK}_{S,f}, \text{SPK}_f)$

ABTTS.Sign$(\text{PK}, \text{SK}_S, \text{SSK}_{S,f}, \text{SPK}_f, (m, f))$:
$\text{CHA} \leftarrow \mathit{Hash}_\mu((\text{CMT}_l)_l \parallel m)$
$\text{CHA}_{r(\mathcal{T}_f)} := \text{CHA}$

$\mathbf{\Sigma}_f^3(x, w, \mathcal{T}_f, (v_n)_n,$
$(\text{CMT}_l)_l, \ (\text{CHA}_n)_n, (\text{RES}_l)_l; \mathit{st})$
$\rightarrow ((\text{CHA}_n)_n, (\text{RES}_l)_l)$
Return $\sigma := ((\text{CHA}_n)_n, (\text{RES}_l)_l)$

ABTTS.Vrfy$(\text{PK}, \text{SPK}_f, (m, f),$
$\sigma := ((\text{CHA}_n)_n, (\text{RES}_l)_l))$:

$\text{CHA} \leftarrow \mathit{Hash}_\mu((\text{CMT}_l)_l \parallel m)$

$\mathbf{\Sigma}_f^{\text{vrfy}}(x, \mathcal{T}_f, \text{CHA},$
$(\text{CMT}_l)_l, (\text{CHA}_n)_n, (\text{RES}_l)_l)$
$\rightarrow b$, Return b

Fig. 6. The scheme of our `ABTTS`.

References

[AABN02] Michel Abdalla, Jee Hea An, Mihir Bellare, and Chanathip Namprempre. From identification to signatures via the fiat-shamir transform: Minimizing assumptions for security and forward-security. In *Advances in Cryptology - EUROCRYPT 2002, International Conference on the Theory and Applications of Cryptographic Techniques, Amsterdam, The Netherlands, April 28 - May 2, 2002, Proceedings*, pages 418–433, 2002.

[AAHI13] Hiroaki Anada, Seiko Arita, Sari Handa, and Yosuke Iwabuchi. Attribute-based identification: Definitions and efficient constructions. In *Information Security and Privacy - 18th Australasian Conference, ACISP 2013, Brisbane, Australia, July 1-3, 2013. Proceedings*, pages 168–186, 2013.

[AAS14] Hiroaki Anada, Seiko Arita, and Kouichi Sakurai. Attribute-based signatures without pairings via the fiat-shamir paradigm. In *ASIAPKC2014*, volume 2 of *ACM-ASIAPKC*, pages 49–58. ACM, 2014.

[AAS15] Hiroaki Anada, Seiko Arita, and Kouichi Sakurai. Attribute-based two-tier signatures: Definition and construction. In *Information Security and Cryptology - ICISC 2015 - 18th International Conference, Seoul, South Korea, November 25-27, 2015, Revised Selected Papers*, pages 36–49, 2015.

[Bab85] László Babai. Trading group theory for randomness. In *Proceedings of the 17th Annual ACM Symposium on Theory of Computing, May 6-8, 1985, Providence, Rhode Island, USA*, pages 421–429, 1985.

[BB04a] Dan Boneh and Xavier Boyen. Efficient selective-id secure identity-based encryption without random oracles. In *Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004, Proceedings*, pages 223–238, 2004.

[BB04b] Dan Boneh and Xavier Boyen. Short signatures without random oracles. In *Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004, Proceedings*, pages 56–73, 2004.

[BBS04] Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In *Advances in Cryptology - CRYPTO 2004, 24th Annual International CryptologyConference, Santa Barbara, California, USA, August 15-19, 2004, Proceedings*, pages 41–55, 2004.

[BF14] Mihir Bellare and Georg Fuchsbauer. Policy-based signatures. In *Public-Key Cryptography - PKC 2014 - 17th International Conference on Practice and Theory in Public-Key Cryptography, Buenos Aires, Argentina, March 26-28, 2014. Proceedings*, pages 520–537, 2014.

[BG92] Mihir Bellare and Oded Goldreich. On defining proofs of knowledge. In *Advances in Cryptology - CRYPTO '92, 12th Annual International Cryptology Conference, Santa Barbara, California, USA, August 16-20, 1992, Proceedings*, pages 390–420, 1992.

[BP02] Mihir Bellare and Adriana Palacio. GQ and Schnorr identification schemes: Proofs of security against impersonation under active and concurrent attacks. In *Advances in Cryptology - CRYPTO 2002, 22nd Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 2002, Proceedings*, pages 162–177, 2002.

[BS07] Mihir Bellare and Sarah Shoup. Two-tier signatures, strongly unforgeable signatures, and fiat-shamir without random oracles. In *Public Key Cryptography - PKC 2007, 10th International Conference on Practice and Theory in Public-Key Cryptography, Beijing, China, April 16-20, 2007, Proceedings*, pages 201–216, 2007.

[CDS94] Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *CRYPTO '94*, pages 174–187. Springer-Verlag, 1994.

[CL02] Jan Camenisch and Anna Lysyanskaya. A signature scheme with efficient protocols. In *Security in Communication Networks, Third International Conference, SCN 2002, Amalfi, Italy, September 11-13, 2002. Revised Papers*, pages 268–289, 2002.

[Cra96] Ronald Cramer. *Modular Designs of Secure, yet Practical Cyptographic Protocols*. PhD thesis, University of Amsterdam, Amsterdam, the Netherlands, 1996.

[Dam10] Ivan Damgård. On σ-protocols. In Course Notes, http://cs.au.dk/ ivan/CPT.html, 2010.

[ECGD14] Ali El Kaafarani, Liqun Chen, Essam Ghadafi, and James H. Davenport. Attribute-based signatures with user-controlled linkability. In *Cryptology and Network Security - 13th International Conference, CANS 2014, Heraklion, Crete, Greece, October 22-24, 2014. Proceedings*, pages 256–269, 2014.

[EGK14] Ali El Kaafarani, Essam Ghadafi, and Dalia Khader. Decentralized traceable attribute-based signatures. In *Topics in Cryptology - CT-RSA 2014 - The Cryptographer's Track at the RSA Conference 2014, San Francisco, CA, USA, February 25-28, 2014. Proceedings*, pages 327–348, 2014.

[EHM11] Alex Escala, Javier Herranz, and Paz Morillo. Revocable attribute-based signatures with adaptive security in the standard model. In *Progress in Cryptology - AFRICACRYPT 2011 - 4th International Conference on Cryptology in Africa, Dakar, Senegal, July 5-7, 2011. Proceedings*, pages 224–241, 2011.

[FI05] Jun Furukawa and Hideki Imai. An efficient group signature scheme from bilinear maps. In *Information Security and Privacy, 10th Australasian Conference, ACISP 2005, Brisbane, Australia, July 4-6, 2005, Proceedings*, pages 455–467, 2005.

[FS86] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Advances in Cryptology - CRYPTO '86, Santa Barbara, California, USA, 1986, Proceedings*, pages 186–194, 1986.

[FS90] Uriel Feige and Adi Shamir. Witness indistinguishable and witness hiding protocols. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing, May 13-17, 1990, Baltimore, Maryland, USA*, pages 416–426, 1990.

[Gha15] Essam Ghadafi. Stronger security notions for decentralized traceable attribute-based signatures and more efficient constructions. In *Topics in Cryptology - CT-RSA 2015, The Cryptographer's Track at the RSA Conference 2015, San Francisco, CA, USA, April 20-24, 2015. Proceedings*, pages 391–409, 2015.

[GKZ14] Robert Granger, Thorsten Kleinjung, and Jens Zumbrägel. Breaking '128-bit secure' supersingular binary curves - (or how to solve discrete logarithms in $f_{2^4 1223}$ and $f_{2^{12} 367}$). In *Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part II*, pages 126–145, 2014.

[GMR89] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM J. Comput.*, 18(1):186–208, 1989.

[GNS12] Martin Gagné, Shivaramakrishnan Narayan, and Reihaneh Safavi-Naini. Short pairing-efficient threshold-attribute-based signature. In *Pairing-Based Cryptography - Pairing 2012 - 5th International Conference, Cologne, Germany, May 16-18, 2012, Revised Selected Papers*, pages 295–313, 2012.

[GPSW06] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *Proceedings of the 13th ACM Conference on Computer and Communications Security, CCS 2006, Alexandria, VA, USA, Ioctober 30 - November 3, 2006*, pages 89–98, 2006.

[GQ88] Louis C. Guillou and Jean-Jacques Quisquater. A "paradoxical" indentity-based signature scheme resulting from zero-knowledge. In *Advances in Cryptology - CRYPTO '88, 8th Annual International Cryptology Conference, Santa Barbara, California, USA, August 21-25, 1988, Proceedings*, pages 216–231, 1988.

[GS08] Jens Groth and Amit Sahai. Efficient non-interactive proof systems for bilinear groups. In *Proceedings of the Theory and Applications of Cryptographic Techniques 27th Annual International Conference on Advances in Cryptology*, EUROCRYPT'08, pages 415–432, Berlin, Heidelberg, 2008. Springer-Verlag.

[GZ08] Shanqing Guo and Yingpei Zeng. Attribute-based signature scheme. In *ISA '08*, pages 509–511. IEEE, 2008.

[Her14] Javier Herranz. Attribute-based signatures from rsa. *Theoretical Computer Science*, 527:73–82, 2014.

[Her16a] Javier Herranz. Attribute-based versions of schnorr and elgamal. *Appl. Algebra Eng. Commun. Comput.*, 27(1):17–57, 2016.

[Her16b] Javier Herranz. Private communication via e-mail, dept. matemàtica aplicada iv, universitat politècnica de catalunya, July 2014, Sept 2015 and May 2016.

[HLLR12] Javier Herranz, Fabien Laguillaumie, Benoît Libert, and Carla Ràfols. Short attribute-based signatures for threshold predicates. In *Topics in Cryptology - CT-RSA 2012 - The Cryptographers' Track at the RSA Conference 2012, San Francisco, CA, USA, February 27 - March 2, 2012. Proceedings*, pages 51–67, 2012.

[HLR10] Javier Herranz, Fabien Laguillaumie, and Carla Ràfols. Constant size ciphertexts in threshold attribute-based encryption. In *Public Key Cryptography - PKC 2010, 13th International Conference on Practice and Theory in Public Key Cryptography, Paris, France, May 26-28, 2010. Proceedings*, pages 19–34, 2010.

[KABR10] Swarun Kumar, Shivank Agrawal, Subha Balaraman, and C. Pandu Rangan. Attribute based signatures for bounded multi-level threshold circuits. In *Public Key Infrastructures, Services and Applications - 7th European Workshop, EuroPKI 2010, Athens, Greece, September 23-24, 2010. Revised Selected Papers*, pages 141–154, 2010.

[Kat03] Jonathan Katz. Efficient and non-malleable proofs of plaintext knowledge and applications. In *Advances in Cryptology - EUROCRYPT 2003, International Conference on the Theory and Applications of Cryptographic Techniques, Warsaw, Poland, May 4-8, 2003, Proceedings*, pages 211–228, 2003.

[KL07] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography.* Chapman and Hall/CRC Press, 2007.

[LAS$^+$10] Jin Li, Man Ho Au, Willy Susilo, Dongqing Xie, and Kui Ren. Attribute-based signature and its applications. In *Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security, ASIACCS 2010, Beijing, China, April 13-16, 2010*, pages 60–69, 2010.

[MPR11] Hemanta K. Maji, Manoj Prabhakaran, and Mike Rosulek. Attribute-based signatures. In *Topics in Cryptology - CT-RSA 2011 - The Cryptographers' Track at the RSA Conference 2011, San Francisco, CA, USA, February 14-18, 2011. Proceedings*, pages 376–392, 2011.

[Oka06] Tatsuaki Okamoto. Efficient blind and partially blind signatures without random oracles. In *Theory of Cryptography, Third Theory of Cryptography Conference, TCC 2006, New York, NY, USA, March 4-7, 2006, Proceedings*, pages 80–99, 2006.

[OT11] Tatsuaki Okamoto and Katsuyuki Takashima. Efficient attribute-based signatures for non-monotone predicates in the Standard Model. In *PKC 2011*, volume 6571 of *LNCS*, pages 35–52. Springer, 2011.

[OT13] Tatsuaki Okamoto and Katsuyuki Takashima. Decentralized attribute-based signatures. In *Public-Key Cryptography - PKC 2013 - 16th International Conference on Practice and Theory in Public-Key Cryptography, Nara, Japan, February 26 - March 1, 2013. Proceedings*, pages 125–142, 2013.

[PS96] David Pointcheval and Jacques Stern. Security proofs for signature schemes. In *Advances in Cryptology - EUROCRYPT '96, International Conference on the Theory and Application of Cryptographic Techniques, Saragossa, Spain, May 12-16, 1996, Proceeding*, pages 387–398, 1996.

[SAH16] Yusuke Sakai, Nuttapong Attrapadung, and Goichiro Hanaoka. Attribute-based signatures for circuits from bilinear map. In *Public-Key Cryptography - PKC 2016 - 19th IACR International Conference on Practice and Theory in Public-Key Cryptography, Taipei, Taiwan, March 6-9, 2016, Proceedings, Part I*, pages 283–300, 2016.

[Sch89] Claus-Peter Schnorr. Efficient identification and signatures for smart cards. In *Advances in Cryptology - CRYPTO '89, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 1989, Proceedings*, pages 239–252, 1989.

[SS09] Siamak Fayyaz Shahandashti and Reihaneh Safavi-Naini. Threshold attribute-based signatures and their application to anonymous credential systems. In *Progress in Cryptology - AFRICACRYPT 2009, Second International Conference on Cryptology in Africa, Gammarth, Tunisia, June 21-25, 2009. Proceedings*, pages 198–216, 2009.

[SW04] Amit Sahai and Brent Waters. Fuzzy identity based encryption. Cryptology ePrint Archive, Report 2004/086, 2004.

[TF12] Isamu Teranishi and Jun Furukawa. Anonymous credential with attributes certification after registration. *IEICE Transactions*, 95-A(1):125–137, 2012.

[YSKI12] Masaya Yasuda, Takeshi Shimoyama, Jun Kogure, and Tetsuya Izu. On the strength comparison of the ECDLP and the IFP. In *Security and Cryptography for Networks - 8th International Conference, SCN 2012, Amalfi, Italy, September 5-7, 2012. Proceedings*, pages 302–325, 2012.

A Credential Bundle Scheme [MPR11]

A credential bundle scheme [MPR11] $\mathtt{CB}$ is an extended notion of a digital signature scheme. It consists of three PPTs: $\mathtt{CB} = (\mathbf{CB}.\mathbf{KG}, \mathbf{CB}.\mathbf{Sign}, \mathbf{CB}.\mathbf{Vrfy})$.
$\mathbf{CB}.\mathbf{KG}(1^\lambda) \to (\mathrm{PK}, \mathrm{SK})$. Given 1^λ as input, it returns a public key PK and a secret key SK.
$\mathbf{CB}.\mathbf{Sign}(\mathrm{PK}, \mathrm{SK}, (m_i)_{i=1}^n) \to (\tau, (\sigma_i)_{i=1}^n)$. Given PK, SK and messages $(m_i)_{i=1}^n$, it returns a *tag* τ and signatures $(\sigma_i)_{i=1}^n$. n is bounded by a polynomial in λ.
$\mathbf{CB}.\mathbf{Vrfy}(\mathrm{PK}, (m_i)_{i=1}^n, (\tau, (\sigma_i)_{i=1}^n)) \to 1/0$. Given PK, messages $(m_i)_{i=1}^n$, a tag τ and signatures $(\sigma_i)_{i=1}^n$, it returns 1 or 0.

A PPT adversary $\mathcal{F}$ tries to make a forgery $((m_i^*)_{i=1}^{n^*}, (\tau^*, (\sigma_i^*)_{i=1}^{n^*}))$. Here τ^* is called a *target tag. An existential forgery by a chosen-message attack* is defined by:

$$\begin{aligned}
&\mathbf{Exp}^{\text{euf-cma}}_{\mathtt{CB},\mathcal{F}}(1^\lambda)\\
&\quad (\mathrm{PK}, \mathrm{SK}) \leftarrow \mathbf{CB}.\mathbf{KG}(1^\lambda), ((m_i^*)_{i=1}^{n^*}, (\tau^*, (\sigma_i^*)_{i=1}^{n^*})) \leftarrow \mathcal{F}^{\mathcal{SBSIGN}}(\mathrm{PK})\\
&\quad \text{If } \mathbf{CB}.\mathbf{Vrfy}(\mathrm{PK}, (m_i^*)_{i=1}^{n^*}, (\tau^*, (\sigma_i^*)_{i=1}^{n^*})) = 1\\
&\quad\quad \text{then Return } \textsc{Win} \text{ else Return } \textsc{Lose}
\end{aligned}$$

Giving a vector of messages $(m_i)_{i=1}^n$, $\mathcal{F}$ queries $\mathcal{SBSIGN}(\mathrm{PK}, \mathrm{SK}, \cdot)$ for a valid credential bundle $(\tau, (\sigma_i)_{i=1}^n)$. τ^* should be different from any queried tag τ, or, whenever τ^* is equal to a queried tag τ, it should hold that $\{m_i^*\}_{i=1}^{n^*} \not\subseteq \{m_i\}_{i=1}^n$ for any queried $(m_i)_{i=1}^n$. The *advantage* of $\mathcal{F}$ over $\mathtt{CB}$ in the experiment of existential forgery by chosen-message attack is defined as $\mathbf{Adv}^{\text{euf-cma}}_{\mathtt{CB},\mathcal{F}}(\lambda) \overset{\text{def}}{=} \Pr[\mathbf{Exp}^{\text{euf-cma}}_{\mathtt{CB},\mathcal{F}}(1^\lambda) \text{ returns } \textsc{Win}]$.

Definition 4 *$\mathtt{CB}$ is called* existentially unforgeable against chosen-message attack *if, for any PPT $\mathcal{F}$, $\mathbf{Adv}^{\text{euf-cma}}_{\mathtt{CB},\mathcal{F}}(\lambda)$ is negligible in λ.*

B Pseudorandom Function Family [KL07]

A pseudorandom function family, $\{PRF_k\}_{k \in PRFkeysp(\lambda)}$, is a function family in which each function $PRF_k : \{1,0\}^* \to \{1,0\}^*$ is an efficiently-computable function that looks random to any polynomial-time distinguisher, where k is called a key and $PRFkeysp(\lambda)$ is called a key space. (See more details in, for example, the book [KL07].)

C Access Structure [GPSW06]

Let $\mathcal{U} = \{1, \ldots, u\}$ be an attribute universe. We must distinguish two cases: the case that $\mathcal{U}$ is small (that is, $|\mathcal{U}| = u$ is bounded by a polynomial in λ) and the case that $\mathcal{U}$ is large (that is, u is not necessarily bounded). We assume the small case in this paper.

Let $f = f(X_{i_1}, \ldots, X_{i_a})$ be a boolean predicate over boolean variables $U = \{X_1, \ldots, X_u\}$. That is, variables $X_{i_1}, \ldots, X_{i_a}$ are connected by boolean connectives; AND-gate ($\wedge$) and OR-gate ($\vee$). For example, $f = X_{i_1} \wedge ((X_{i_2} \wedge X_{i_3}) \vee X_{i_4})$ for some $i_1, i_2, i_3, i_4,\ 1 \le i_1 < i_2 < i_3 < i_4 \le u$. Note that there is a bijective map between boolean variables and attributes:

$$\psi : U \to \mathcal{U},\ \psi(X_i) \overset{\text{def}}{=} i.$$

For $f(X_{i_1}, \ldots, X_{i_a})$, we denote the set of indices (that is, attributes) $\{i_1, \ldots, i_a\}$ by $\mathrm{Att}(f)$. We note the arity of f as $\mathrm{arity}(f)$. Hereafter we use the symbol i_j to mean the following:

$$i_j \stackrel{\text{def}}{=} \text{the index } i \text{ of a boolean variable that is the } j\text{-th argument of } f.$$

Suppose that we are given an access structure as a boolean predicate f. For $S \in 2^{\mathcal{U}}$, we evaluate the boolean value of f at S as follows:

$$f(S) \stackrel{\text{def}}{=} f\big(X_{i_j} \leftarrow [\psi(X_{i_j}) \in_? S]; j = 1, \ldots, \mathrm{arity}(f)\big) \in \{1, 0\}.$$

Under this definition, a boolean predicate f can be seen as a map: $f : 2^{\mathcal{U}} \to \{1, 0\}$. We call a boolean predicate f with this map an *access formula* over $\mathcal{U}$. In this paper, we assume that no NOT-gate ($\neg$) appears in f. In other words, we consider only *monotone* access formulas.[6]

C.1 Access Tree

A monotone access formula f can be represented by a finite binary tree $\mathcal{T}_f$. Each inner node represents a boolean connective, $\wedge$-gate or $\vee$-gate, in f. Each leaf corresponds to a term X_i (not a variable X_i) in f in one-to-one way. For a finite binary tree tree $\mathcal{T}$, we denote the set of all nodes, the root node, the set of all leaves, the set of all inner nodes (that is, all nodes excluding leaves) and the set of all tree-nodes (that is, all nodes excluding the root node) as $\mathrm{Node}(\mathcal{T})$, $r(\mathcal{T})$, $\mathrm{Leaf}(\mathcal{T})$, $\mathrm{iNode}(\mathcal{T})$ and $\mathrm{tNode}(\mathcal{T})$, respectively. Then the attribute map $\rho(\cdot)$ is defined as:

$$\rho : \mathrm{Leaf}(\mathcal{T}) \to \mathcal{U},\ \rho(l) \stackrel{\text{def}}{=} (\text{the attribute } i \text{ that corresponds to } l \text{ through } \psi).$$

If ρ is not injective, then we call the case *multi-use* of attributes.

If $\mathcal{T}$ is of height greater than 0, $\mathcal{T}$ has two subtrees whose root nodes are two children of $r(\mathcal{T})$. We denote the two subtrees by $\mathrm{Lsub}(\mathcal{T})$ and $\mathrm{Rsub}(\mathcal{T})$, which mean the left subtree and the right subtree, respectively.

D Attribute-Based Identification Scheme [AAHI13]

An attribute-based identification scheme, $\mathtt{ABID}$, consists of four PPT algorithms [AAHI13]: $\mathtt{ABID} = (\mathbf{ABID.Setup}, \mathbf{ABID.KG}, \mathcal{P}, \mathcal{V})$.

$\mathbf{ABID.Setup}(1^\lambda, \mathcal{U}) \to (\mathbf{PK}, \mathbf{MSK})$. This PPT algorithm for setting up master and public keys takes as input the security parameter 1^λ and an attribute universe $\mathcal{U}$. It returns a public key PK and a master secret key MSK.

$\mathbf{ABID.KG}(\mathbf{PK}, \mathbf{MSK}, S) \to \mathbf{SK}_S$. This PPT algorithm for key-generation takes as input the public key PK, the master secret key MSK and an attribute set $S \subset \mathcal{U}$. It returns an id-key SK_S corresponding to S.

$\mathcal{P}(\mathbf{PK}, \mathbf{SK}_S, f)$ **and** $\mathcal{V}(\mathbf{PK}, f)$. These interactive PPT algorithms are called a *prover* and a *verifier*, respectively. $\mathcal{P}$ takes as input the public key PK, the secret key SK_S and an access formula f. Here the secret key SK_S is given to $\mathcal{P}$ by an authority that runs $\mathbf{ABID.KG}(\mathrm{PK}, \mathrm{MSK}, S)$. $\mathcal{V}$ takes as input the public key PK and an access formula f. $\mathcal{P}$ and $\mathcal{V}$ interact with each other for at most a polynomial-number of moves. Then, $\mathcal{V}$ returns its decision 1 or 0. When it is 1, we say that $\mathcal{V}$ *accepts* $\mathcal{P}$ for f. When it is 0, we say that $\mathcal{V}$ *rejects* $\mathcal{P}$ for f.

We demand correctness of $\mathtt{ABID}$ that, for any λ, and if $f(S) = 1$, $\Pr[(\mathrm{PK}, \mathrm{MSK}) \leftarrow \mathbf{ABID.Setup}(1^\lambda, \mathcal{U}), \mathrm{SK}_S \leftarrow \mathbf{ABID.KG}(\mathrm{PK}, \mathrm{MSK}, S), b \leftarrow \langle \mathcal{P}(\mathrm{PK}, \mathrm{SK}_S), \mathcal{V}(\mathrm{PK}, f)\rangle : b = 1] = 1$.

D.1 Passive and Concurrent Attacks on ABID and Security Definition

Informally speaking, an adversary $\mathcal{A}$'s objective is impersonation. $\mathcal{A}$ tries to make a verifier $\mathcal{V}$ accept with an access formula f^*.

The following experiment $\mathbf{Exp}^{\mathrm{pa}}_{\mathtt{ABID},\mathcal{A}}(1^\lambda, \mathcal{U})$ of an adversary $\mathcal{A}$ defines the game of *passive attack* on ABID.

$$\begin{aligned}
&\mathbf{Exp}^{\mathrm{pa}}_{\mathtt{ABID},\mathcal{A}}(1^\lambda, \mathcal{U}):\\
&\quad (\mathrm{PK}, \mathrm{MSK}) \leftarrow \mathbf{ABID.Setup}(1^\lambda, \mathcal{U})\\
&\quad (f^*, st) \leftarrow \mathcal{A}^{\mathcal{KG}(\mathrm{PK},\mathrm{MSK},\cdot),\mathrm{Transc}(\mathcal{P}(\mathrm{PK},\mathrm{SK}_\cdot,\cdot),\mathcal{V}(\mathrm{PK},\cdot))}(\mathrm{PK}, \mathcal{U})\\
&\quad b \leftarrow \langle \mathcal{A}(st), \mathcal{V}(\mathrm{PK}, f^*)\rangle\\
&\quad \text{If } b = 1 \text{ then Return } \textsc{Win} \text{ else Return } \textsc{Lose}
\end{aligned}$$

[6] This limitation can be removed by adding *negation attributes* to $\mathcal{U}$ for each attribute in the original $\mathcal{U}$ though the size of the attribute universe $|\mathcal{U}|$ doubles.

In the experiment, $\mathcal{A}$ issues key-extraction queries to its key-generation oracle $\mathcal{KG}$ and transcript queries to its transcript oracle Transc. In a transcript query, giving a pair (S_j, f_j) of an attribute set and an access formula, $\mathcal{A}$ queries $\text{Transc}(\mathcal{P}(\text{PK}, \text{SK}_\cdot, \cdot), \mathcal{V}(\text{PK}, \cdot))$ for a whole transcript of messages interacted between $\mathcal{P}(\text{PK}, \text{SK}_{S_j}, f_j)$ and $\mathcal{V}(\text{PK}, f_j)$.

The *advantage* of $\mathcal{A}$ over `ABID` in the game of a passive attack is defined as

$$\mathbf{Adv}^{\text{pa}}_{\texttt{ABID},\mathcal{A}}(\lambda,\mathcal{U}) \stackrel{\text{def}}{=} \Pr[\mathbf{Exp}^{\text{pa}}_{\texttt{ABID},\mathcal{A}}(1^\lambda,\mathcal{U}) \text{ returns } \textsc{Win}].$$

`ABID` is called *secure against passive attacks* if, for any PPT $\mathcal{A}$ and for any $\mathcal{U}$, $\mathbf{Adv}^{\text{pa}}_{\texttt{ABID},\mathcal{A}}(\lambda,\mathcal{U})$ is negligible in λ.

The following experiment $\mathbf{Exp}^{\text{ca}}_{\texttt{ABID},\mathcal{A}}(1^\lambda,\mathcal{U})$ of an adversary $\mathcal{A}$ defines the game of *concurrent attack* on ABID.

$$\begin{aligned}
&\mathbf{Exp}^{\text{ca}}_{\texttt{ABID},\mathcal{A}}(1^\lambda,\mathcal{U}):\\
&\quad (\text{PK}, \text{MSK}) \leftarrow \mathbf{ABID}.\mathbf{Setup}(1^\lambda,\mathcal{U})\\
&\quad (f^*, st) \leftarrow \mathcal{A}^{\mathcal{KG}(\text{PK},\text{MSK},\cdot),\mathcal{P}_j(\text{PK},\text{SK}_\cdot,\cdot)|_{j=1}^{q_\text{p}}}(\text{PK},\mathcal{U})\\
&\quad b \leftarrow \langle \mathcal{A}(st), \mathcal{V}(\text{PK}, f^*)\rangle\\
&\quad \text{If } b = 1 \text{ then Return } \textsc{Win} \text{ else Return } \textsc{Lose}
\end{aligned}$$

In the experiment, $\mathcal{A}$ issues key-extraction queries to its key-generation oracle $\mathcal{KG}$. Giving an attribute set S_i, $\mathcal{A}$ queries $\mathcal{KG}(\text{PK}, \text{MSK}, \cdot)$ for the secret key SK_{S_i}. In addition, $\mathcal{A}$ invokes provers $\mathcal{P}_j(\text{PK}, \text{SK}_\cdot, \cdot),\ j = 1, \ldots, q_\text{p}$, by giving a pair (S_j, f_j) of an attribute set and an access formula. Acting as a verifier with an access formula f_j, $\mathcal{A}$ interacts with each $P_j(\text{PK}, \text{SK}_{S_j}, f_j)$ concurrently.

The access formula f^* declared by $\mathcal{A}$ is called a *target access formula*. Here we consider the *adaptive* target in the sense that $\mathcal{A}$ is allowed to choose f^* after seeing PK, issuing key-extraction queries and interacting with of provers. A restriction is imposed on $\mathcal{A}$ concerning f^*. For all key-extraction queries (i.e. for $^\forall i$), $f^*(S_i) = 0$. The number of key-extraction queries and the number of invoked provers are at most q_k and q_p, respectively, which are bounded by a polynomial in λ.

The *advantage* of $\mathcal{A}$ over `ABID` in the game of a concurrent attack is defined as

$$\mathbf{Adv}^{\text{ca}}_{\texttt{ABID},\mathcal{A}}(\lambda,\mathcal{U}) \stackrel{\text{def}}{=} \Pr[\mathbf{Exp}^{\text{ca}}_{\texttt{ABID},\mathcal{A}}(1^\lambda,\mathcal{U}) \text{ returns } \textsc{Win}].$$

`ABID` is called *secure against concurrent attacks* if, for any PPT $\mathcal{A}$ and for any $\mathcal{U}$, $\mathbf{Adv}^{\text{ca}}_{\texttt{ABID},\mathcal{A}}(\lambda,\mathcal{U})$ is negligible in λ.

The concurrent security means the passive security; for any PPT $\mathcal{A}$, there exists a PPT $\mathcal{B}$ that satisfies the following inequality.

$$\mathbf{Adv}^{\text{pa}}_{\texttt{ABID},\mathcal{A}}(\lambda,\mathcal{U}) \leq \mathbf{Adv}^{\text{ca}}_{\texttt{ABID},\mathcal{B}}(\lambda,\mathcal{U}). \tag{4}$$

E Attribute-Based Signature Scheme [MPR11,OT11]

An attribute-based signature scheme, `ABS`, consists of four PPT algorithms [OT11]: `ABS` = $(\mathbf{ABS}.\mathbf{Setup}, \mathbf{ABS}.\mathbf{KG}, \mathbf{ABS}.\mathbf{Sign}, \mathbf{ABS}.\mathbf{Vrfy})$.

$\mathbf{ABS}.\mathbf{Setup}(1^\lambda,\mathcal{U}) \to (\mathbf{PK}, \mathbf{MSK})$. This PPT algorithm for setting up master and public keys takes as input the security parameter 1^λ and an attribute universe $\mathcal{U}$. It returns a public key PK and a master secret key MSK.

$\mathbf{ABS}.\mathbf{KG}(\mathbf{PK}, \mathbf{MSK}, S) \to \mathbf{SK}_S$. This PPT algorithm for key-generation takes as input the public key PK, the master secret key MSK and an attribute set $S \subset \mathcal{U}$. It returns a signing key SK_S corresponding to S.

$\mathbf{ABS}.\mathbf{Sign}(\mathbf{PK}, \mathbf{SK}_S, (m, f)) \to \sigma$. This PPT algorithm for signing takes as input a public key PK, a private secret key SK_S corresponding to an attribute set S, a pair (m, f) of a message $\in \{1, 0\}^*$ and an access formula. It returns a signature σ.

$\mathbf{ABS}.\mathbf{Vrfy}(\mathbf{PK}, (m, f), \sigma) \to 1/0$. This deterministic polynomial-time algorithm takes as input a public key PK, a pair (m, f) of a message and an access formula, and a signature σ. It returns a decision 1 or 0. When it is 1, we say that $((m, f), \sigma)$ is *valid*. When it is 0, we say that $((m, f), \sigma)$ is *invalid*.

We demand correctness of `ABS` that, for any λ, any $\mathcal{U}$, any $S \subset \mathcal{U}$ and any (m, f) such that $f(S) = 1$, $\Pr[(\text{PK}, \text{MSK}) \leftarrow \mathbf{ABS}.\mathbf{Setup}(1^\lambda,\mathcal{U}), \text{SK}_S \leftarrow \mathbf{ABS}.\mathbf{KG}(\text{PK}, \text{MSK}, S), \sigma \leftarrow \mathbf{ABS}.\mathbf{Sign}(\text{PK}, \text{SK}_S, (m, f)),\ b \leftarrow \mathbf{ABS}.\mathbf{Vrfy}(\text{PK}, (m, f), \sigma) : b = 1] = 1$.

E.1 Chosen-Message Attack on ABS and Security Definition

Informally speaking, an adversary $\mathcal{F}$'s objective is to make an *existential forgery*. $\mathcal{F}$ tries to make a forgery $((m^*, f^*), \sigma^*)$ that consists of a message, a target access structure and a signature. The following experiment $\mathbf{Exp}^{\text{euf-cma}}_{\mathtt{ABS},\mathcal{F}}(1^\lambda, \mathcal{U})$ of a forger $\mathcal{F}$ defines the *chosen-message attack on ABS to make an existential forgery.*

$$
\begin{aligned}
&\mathbf{Exp}^{\text{euf-cma}}_{\mathtt{ABS},\mathcal{F}}(1^\lambda, \mathcal{U}) : \\
&\quad (\mathrm{PK}, \mathrm{MSK}) \leftarrow \mathbf{ABS}.\mathbf{Setup}(1^\lambda, \mathcal{U}) \\
&\quad ((m^*, f^*), \sigma^*) \leftarrow \mathcal{F}^{\mathcal{KG}(\mathrm{PK},\mathrm{MSK},\cdot), \mathcal{SIGN}(\mathrm{PK},\mathrm{SK}_{\cdot},(\cdot,\cdot))}(\mathrm{PK}) \\
&\quad \text{If } \mathbf{ABS}.\mathbf{Vrfy}(\mathrm{PK}, (m^*, f^*), \sigma^*) = 1 \text{ then Return } \textsc{Win} \\
&\quad \text{else Return } \textsc{Lose}
\end{aligned}
$$

In the experiment, $\mathcal{F}$ issues key-extraction queries to its key-generation oracle $\mathcal{KG}$ and signing queries to its signing oracle $\mathcal{SIGN}$. Giving an attribute set S_i, $\mathcal{F}$ queries $\mathcal{KG}(\mathrm{PK}, \mathrm{MSK}, \cdot)$ for the secret key SK_{S_i}. In addition, giving an attribute set S_j and a pair (m_j, f_j) of a message and an access formula, $\mathcal{F}$ queries $\mathcal{SIGN}(\mathrm{PK}, \mathrm{SK}_{\cdot}, (\cdot,\cdot))$ for a signature σ_j that satisfies $\mathbf{ABS}.\mathbf{Vrfy}(\mathrm{PK}, (m_j, f_j), \sigma_j) = 1$ when $f_j(S_j) = 1$.

The access formula f^* declared by $\mathcal{F}$ is called a *target access formula*. Here we consider the *adaptive* target in the sense that $\mathcal{F}$ is allowed to choose f^* after seeing PK and issuing some key-extraction queries and signing queries. Two restrictions are imposed on $\mathcal{F}$ concerning f^*. For all key-extraction queries (i.e. for $^\forall i$), $f^*(S_i) = 0$. For all signing queries (for $^\forall j$), $f^*(S_j) = 0$ or $m^* \neq m_j$. The number of key-extraction queries and the number of signing queries are at most q_{k} and q_{s}, respectively, which are bounded by a polynomial in λ.

The *advantage* of $\mathcal{F}$ over $\mathtt{ABS}$ in the game of chosen-message attack to make existential forgery is defined as

$$\mathbf{Adv}^{\text{euf-cma}}_{\mathtt{ABS},\mathcal{F}}(\lambda, \mathcal{U}) \stackrel{\text{def}}{=} \Pr[\mathbf{Exp}^{\text{euf-cma}}_{\mathtt{ABS},\mathcal{F}}(1^\lambda, \mathcal{U}) \text{ returns } \textsc{Win}].$$

$\mathtt{ABS}$ is called *existentially unforgeable against chosen-message attacks* if, for any PPT $\mathcal{F}$ and for any $\mathcal{U}$, $\mathbf{Adv}^{\text{euf-cma}}_{\mathtt{ABS},\mathcal{F}}(\lambda, \mathcal{U})$ is negligible in λ.

E.2 Attribute Privacy of ABS

Roughly speaking, ABS is called to have attribute privacy if any cheating verifier with unconditional computing power cannot distinguish two distributions of signatures each of which is generated by different attribute set. The following definition is due to Maji et al. and Okamoto-Takashima.

Definition 5 (Attribute Privacy (Perfect Privacy [MPR11,OT11])) *$\mathtt{ABS}$ is called to have* attribute privacy *if, for all $(PK, MSK) \leftarrow$ $\mathbf{ABS}.\mathbf{Setup}(1^\lambda, \mathcal{U})$, for all message m, for all attribute sets S_1 and S_2, for all signing keys $SK_{S_1} \leftarrow \mathbf{ABS}.\mathbf{KG}(PK, MSK, S_1)$ and $SK_{S_2} \leftarrow \mathbf{ABS}.\mathbf{KG}(PK, MSK, S_2)$ and for all access formula f such that $[f(S_1) = 1 \wedge f(S_2) = 1]$, two distributions $\sigma_1 \leftarrow \mathbf{ABS}.\mathbf{Sign}(PK, SK_{S_1}, (m, f))$ and $\sigma_2 \leftarrow \mathbf{ABS}.\mathbf{Sign}(PK, SK_{S_2}, (m, f))$ are identical.*

F Instantiations Using Fiat-Shamir Credential-Bundle as Witness

In this section, we provide instantiations of our procedure $\mathbf{\Sigma}_f$ and $\mathtt{ABID}$ using the Fiat-Shamir signatures [FS86] as a witness. We give two instantiations in the RSA setting and the discrete-logarithm setting.

F.1 Our ABID in RSA Using FS Credential-Bundle as Witness

An RSA modulus of bit length λ is denoted by N. An RSA exponent of odd prime is denoted by e.
ABID.Setup takes as input $(1^\lambda, \mathcal{U})$. Let $R_\lambda := \{(\beta, \alpha) \in \mathbb{Z}_N \times \mathbb{Z}_N; \beta = \alpha^e\}$. Then $\mathrm{Instance}_R(1^\lambda)$ chooses an element $(\beta, \alpha) \in R_\lambda$ at random. **ABID.Setup** returns a public key and a master secret key: $\mathrm{PK} = ((N, e, \beta), \mathcal{U}, \mu)$, $\mathrm{MSK} = \alpha$.
ABID.KG returns SK_S with signatures, for $i \in S$, $\sigma = (a_i = r_i^e,\ w_i = r_i \alpha^{c_i})$. Here we use a key k obtained by $k \leftarrow \mathit{Hash}_\mu(\alpha \parallel \tau)$, put $m_i = \tau \parallel i$, and $r_i \in \mathbb{Z}_N$ is chosen at random according to a random tape: $\mathit{PRF}_k(m_i)$, and c_i is obtained by $c_i \leftarrow \mathit{Hash}_\mu(a_i \parallel m_i)$. $\mathbf{\Sigma}^{\text{stmtgen}}(\beta, a_i, c_i)$ is an algorithm that computes $x_i := a_i \beta^{c_i} \in \mathbb{Z}_N$.

The rest of protocol is executed according to $\mathbf{\Sigma}_f$ on input (x, w) and with the following setting.

$$
\begin{aligned}
&\textsc{Cmt}_l = {r_l}^e, \textsc{Res}_l = r_l (w_{\rho(l)})^{\textsc{Cha}_l}, \\
&\text{Verification Equation} : {\textsc{Res}_l}^e = \textsc{Cmt}_l\ (x_{\rho(l)})^{\textsc{Cha}_l}.
\end{aligned}
$$

F.2 Our ABID in Discrete Log Using FS Credential-Bundle as Witness

A prime of bit length λ is denoted by p. A multiplicative cyclic group of order p is denoted by $\mathbb{G}_p$. We fix a base $g \in \mathbb{G}_p, \langle g \rangle = \mathbb{G}_p$. The ring of the exponent domain of $\mathbb{G}_p$, which consists of integers from 0 to $p-1$ with modulo p operation, is denoted by $\mathbb{Z}_p$.
ABID.**Setup** takes as input $(1^\lambda, \mathcal{U})$. Let $R_\lambda := \{(\beta, \alpha) \in \mathbb{G}_p \times \mathbb{Z}_p; \beta = g^\alpha\}$. Then $\text{Instance}_R(1^\lambda)$ chooses an element $(\beta, \alpha) \in R_\lambda$ at random. **ABID**.**Setup** returns a public key and a master secret key: $\text{PK} = ((g, \beta), \mathcal{U}, \mu)$, $\text{MSK} = \alpha$.
ABID.**KG** returns SK_S with signatures, for $i \in S$, $\sigma_i = (a_i = g^{r_i},\ w_i = r_i + c_i\alpha)$. Here we use a key k obtained by $k \leftarrow \mathit{Hash}_\mu(\alpha \parallel \tau)$, put $m_i = \tau \parallel i$, and $r_i \in \mathbb{Z}_p$ is chosen at random according to a random tape: $\mathit{PRF}_k(m_i)$, and c_i is obtained by $c_i \leftarrow \mathit{Hash}_\mu(a_i \parallel m_i)$. $\mathbf{\Sigma}^{\text{stmtgen}}(\beta, a_i, c_i)$ is an algorithm that computes $x_i := a_i \beta^{c_i} \in \mathbb{G}_p$.

The rest of protocol is executed according to $\mathbf{\Sigma}_f$ on input (x, w) and with the following setting.

$$\textsc{Cmt}_l = g^{r_l}, \textsc{Res}_l = r_l + \textsc{Cha}_l\ w_{\rho(l)},$$
$$\text{Verification Equation} : g^{\textsc{Res}_l} = \textsc{Cmt}_l\ (x_{\rho(l)})^{\textsc{Cha}_l}.$$

G Instantiations Using Camenisch-Lysyanskaya Credential-Bundle as Witness

In this section, we provide another type of instantiations of our procedure $\mathbf{\Sigma}_f$, ABID and ABTTS using the Camenisch-Lysyanskaya Signatures as a witness. We give two instantiations in the RSA setting [CL02] and the discrete-logarithm setting [TF12,FI05,Oka06].

G.1 Our Σ-protocol $\mathbf{\Sigma}_f$ in the Case of CL Credential-Bundle

Our Σ-protocol $\mathbf{\Sigma}_f$ is a zero-knowledge proof of knowledge $\mathbf{ZKPoK}[w = (w_{\rho(l)})_{l\in\text{Leaf}(\mathcal{T}_f)} := (e_{\rho(l)}, s_{\rho(l)})_{l\in\text{Leaf}(\mathcal{T}_f)}, : x = (\mathit{equations})]$ for the language L_f, where the *equations* are:

$$Z_{\rho(l)} = Z_{\rho(l),1}^{e_{\rho(l)}} Z_{\rho(l),2}^{s_{\rho(l)}},\ l \in \text{Leaf}(\mathcal{T}_f). \tag{5}$$

In the above equation, $Z_{\rho(l)}$ is represented by $(e_{\rho(l)}, s_{\rho(l)})$ to the base $(Z_{\rho(l),1}, Z_{\rho(l),2})$. A prover $\mathcal{P}(x, w, f)$ and a verifier $\mathcal{V}(x, f)$ execute $\mathbf{\Sigma}_f$ in the following way.
$\underline{\mathcal{P}(x, w, f)}$. To prove the knowledge of those representations $(e_{\rho(l)}, s_{\rho(l)})$, $\mathcal{P}$ computes the first message, a commitment $(\textsc{Cmt}_l)_l$, as follows. Let $\bar{\mathbb{Z}}$ be the exponent domain for the above expression. To do the computation honestly at a leaf l, $\mathcal{P}$ chooses $\eta_{e,l}, \eta_{s,l} \in_R \bar{\mathbb{Z}}$, and puts $\textsc{Cmt}_l := Z_{\rho(l),1}^{\eta_{e,l}} Z_{\rho(l),2}^{\eta_{s,l}}$. To simulate the honest computation at a leaf l, $\mathcal{P}$ chooses $\eta_{e,l}, \theta_{s,l} \in_R \bar{\mathbb{Z}}$, and in addition, the divided challenge strings $(\textsc{Cha}_n)_n, \textsc{Cha}_n \in \bar{\mathbb{Z}}$, which are in accordance with our procedure $\mathbf{\Sigma}_f$. Then $\mathcal{P}$ puts, for each leaf l, $\theta_{e,l} := \eta_{e,l} + \textsc{Cha}_l e_{\rho(l)}$, and $\textsc{Cmt}_l := Z_{\rho(l)}^{-\textsc{Cha}_l} Z_{\rho(l),1}^{\theta_{e,l}} Z_{\rho(l),2}^{\theta_{s,l}}$. $\mathcal{P}$ sends $(\textsc{Cmt}_l)_l$ to a verifier $\mathcal{V}$.
$\underline{\mathcal{V}(x, f)}$. Receiving $(\textsc{Cmt}_l)_l$, $\mathcal{V}(x, f)$ chooses the second message: a challenge $\textsc{Cha} \in_R \bar{\mathbb{Z}}$, uniformly at random, and sends $\textsc{Cha}$ to $\mathcal{P}$.
$\underline{\mathcal{P}(x, w, f)}$. Receiving $\textsc{Cha}$, $\mathcal{P}$ completes to compute the third message; that is, $\mathcal{P}$ completes the division $(\textsc{Cha}_n)_n$ such that $\textsc{Cha}_{r(\mathcal{T}_f)} = \textsc{Cha}$, and a response $(\textsc{Res}_l := (\theta_{e,l}, \theta_{s,l}))_l$ with $\theta_{e,l} := \eta_{e,l} + \textsc{Cha}_l e_{\rho(l)},\ \theta_{s,l} := \eta_{s,l} + \textsc{Cha}_l s_{\rho(l)}$. $\mathcal{P}$ sends $(\textsc{Cha}_l)_l$ and $(\textsc{Res}_l)_l$ to $\mathcal{V}$.
$\underline{\mathcal{V}(x, f)}$. Receiving $(\textsc{Cha}_l)_l$ and $(\textsc{Res}_l)_l$, $\mathcal{V}$ checks the integrity of the division $(\textsc{Cha}_l)_l$. Then $\mathcal{V}$ verifies:

$$\textsc{Cmt}_l =_? Z_{\rho(l)}^{-\textsc{Cha}_l} Z_{\rho(l),1}^{\theta_{e,l}} Z_{\rho(l),2}^{\theta_{s,l}},\ l \in \text{Leaf}(\mathcal{T}_f). \tag{6}$$

According to the division rule of our procedure $\mathbf{\Sigma}_f$, the integrity of $(\textsc{Cha}_l)_l$ can be checked as follows: From the leaves to the root, and at every inner node $n \in \text{iNode}(\mathcal{T}_f)$ and its two children $\mathit{chd}_1, \mathit{chd}_2$;

- If n is an AND node $(\wedge)$, then verify $\textsc{Cha}_{chd_1} =_? \textsc{Cha}_{chd_2}$. If so, put $\textsc{Cha}_n := \textsc{Cha}_{chd_1}$.
- Else if n is an OR node $(\vee)$, then just put $\textsc{Cha}_n := \textsc{Cha}_{chd_1} + \textsc{Cha}_{chd_2}$.
- If n is the root node, then verify $\textsc{Cha}_n =_? \textsc{Cha}$.
- Repeat until all $n \in \text{iNode}(\mathcal{T}_f)$ are verified.

The above procedure, $\mathbf{\Sigma}_f$, can be shown to possess the three requirements of Σ-protocol: completeness, special soundness and honest-verifier zero-knowledge.

G.2 Our ABID and ABTTS in RSA Using CL Credential-Bundle as Witness

Strong RSA Assumption [CL02] Let $p = 2p' + 1$ denote a *safe prime* (p' is also a prime). Let N denote the *special RSA modulus*; that is, $N = pq$ where $p = 2p' + 1$ and $q = 2q' + 1$ are two safe primes such that $|p'| = |q'| = \lambda - 1$. We denote the probabilistic algorithm that generates such N at random on input 1^λ as $\texttt{RSAmod}$. Let $QR_N \subset \mathbb{Z}_N^*$ denote the set of quadratic residues modulo N; that is, elements $a \in \mathbb{Z}_N^*$ such that $a \equiv x^2 \bmod N$ for some $x \in \mathbb{Z}_N^*$. The strong RSA assumption [CL02] states that for any PPT $\mathcal{A}$, the following advantage is negligible in λ: $\mathbf{Adv}^{\mathrm{srsa}}_{\texttt{RSAmod},\mathcal{S}}(\lambda) := \Pr[N \leftarrow \texttt{RSAmod}(1^\lambda), g \in_R QR_N, (V, e) \leftarrow \mathcal{A}(N, g) : e > 1 \wedge V^e \equiv g \bmod N]$.

CL Credential-Bundle in RSA
Our credential-bundle scheme $\texttt{CB} = (\mathbf{CB}.\mathbf{KG}, \mathbf{CB}.\mathbf{Sign}, \mathbf{CB}.\mathbf{Vrfy})$ is described as follows. Let $l_\mathcal{M}$ be a parameter. The message space $\mathcal{M}$ consists of all binary strings of length $l_\mathcal{M}$. Let $n = n(\lambda)$ denote the maximum number of messages made into a bundle, which is a polynomial in λ.
$\mathbf{CB}.\mathbf{KG}(1^\lambda) \to (\mathrm{PK}, \mathrm{SK})$. Given 1^λ, it chooses a special RSA modulus $N = pq$ of length $l_N = \lambda$, where $p = 2p' + 1$ and $q = 2q' + 1$ are safe primes. For $i = 1$ to n, it chooses $g_{i,0}, g_{i,1}, g_{i,2} \in_R QR_N$. It puts $\mathrm{PK} := (N, (g_{i,0}, g_{i,1}, g_{i,2})_{i=1}^n)$ and $\mathrm{SK} = p$, and returns $(\mathrm{PK}, \mathrm{SK})$.
$\mathbf{CB}.\mathbf{Sign}(\mathrm{PK}, \mathrm{SK}, (m_i)_{i=1}^n) \to (\tau, (\sigma_i)_{i=1}^n)$. Given PK, SK and messages $(m_i)_{i=1}^n$ each of which is of length $l_\mathcal{M}$, it chooses a prime e of length $l_e = l_\mathcal{M} + 2$ at random. For $i = 1$ to n, it chooses an integer s_i of length $l_s = l_N + l_\mathcal{M} + l$ at random, where l is a security parameter, and it computes the value A_i:

$$A_i := (g_{i,0} g_{i,1}^{m_i} g_{i,2}^{s_i})^{\frac{1}{e}}. \tag{7}$$

It puts $\tau = e$ and $\sigma_i = (s_i, A_i)$ for each i and returns $(\tau, (\sigma_i)_{i=1}^n)$.
$\mathbf{CB}.\mathbf{Vrfy}(\mathrm{PK}, (m_i)_{i=1}^n, (\tau, (\sigma_i)_{i=1}^n)) \to 1/0$. Given PK, $(m_i)_{i=1}^n$ and a credential bundle $(\tau, (\sigma_i)_{i=1}^n)$, it verifies whether the following holds:

$$e := \tau \text{ is of length } l_e \text{ and } A_i^e = g_{i,0} g_{i,1}^{m_i} g_{i,2}^{s_i},\ i = 1, \ldots, n. \tag{8}$$

Theorem 8 (Unforgeability of Our CB) *Our credential-bundle scheme* $\texttt{CB}$ *is existentially unforgeable against chosen-message attacks under the Strong RSA assumption.*

Proof. Basically the proof goes in the same way as the Camenisch-Lysyanskaya signature scheme [CL02]. The difference only arises in the case that the simulation of the credential-bundle oracle needs precomputation.

Let $\mathcal{F}$ be a given PPT forger on our $\texttt{CB}$. We construct a PPT solver $\mathcal{S}$ of any instance (N, g) of the Strong RSA problem. To describe three cases of $\mathcal{F}$'s behavior, suppose that $\mathcal{F}$ issues at most q credential-bundle queries $(m_{j,i})_{i=1}^{n_j}, j = 1, \ldots, q$. Suppose that the credential-bundle oracle $\mathcal{SBSIGN}$ replies the tags (that is, exponents) $e_1, \ldots, e_q$ in answer to $\mathcal{F}$'s queries, which are primes of length l_e. Suppose that $\mathcal{F}$'s forgery is $(m_i^*)_{i=1}^{n^*}, \tau^* = e^*, (\sigma_i^* = (s_i^*, A_i^*))_{i=1}^{n^*}$. Let us distinguish three types of forgeries.

1. e^* is relatively prime to any of $\{e_j\}_j$.
2. e^* is not relatively prime to some of $\{e_j\}_j$, and $g_{i,1}^{m_i^*} g_{i,2}^{s_i^*} \equiv g_{i,1}^{m_{j,i}} g_{i,2}^{s_{j,i}}$ for at least one j s.t. $\gcd(e^*, e_j) \neq 1$ and at least one i.
3. e^* is not relatively prime to some of $\{e_j\}_j$, and $g_{i,1}^{m_i^*} g_{i,2}^{s_i^*} \not\equiv g_{i,1}^{m_{j,i}} g_{i,2}^{s_{j,i}}$ for any j s.t. $\gcd(e^*, e_j) \neq 1$ and any i.

By $\mathcal{F}_1, \mathcal{F}_2$ and $\mathcal{F}_3$ let us denote the forger who runs $\mathcal{F}$ but then only returns its forgery if it is of Type 1, Type 2 and Type 3, respectively. On input an instance (N, g) of the Strong RSA problem, $\mathcal{S}$ first guesses one of the three types at random (hence the advantage of $\mathcal{S}$ reduces by the factor of 1/3 here).

When $\mathcal{F}$ is of Type 1 or Type 2, simulations of $\mathcal{F}$'s credential-bundle oracle $\mathcal{SBSIGN}$ and the extraction of an answer of an instance (N, g) go in the same way as the Camenisch-Lysyanskaya signature scheme [CL02].

When $\mathcal{F}$ is of Type 3, the simulation of $\mathcal{SBSIGN}$ needs slight enhancement. $\mathcal{S}$ chooses q primes $\{e_j\}_{j=1}^q$ of length l_e. Then $\mathcal{S}$ chooses $j^* \in \{1, \ldots, q\}$ at random, and for each $i = 1$ to n, puts $E := \prod_{1 \le j \le q, j \neq j^*} e_j$. Then, for each $i = 1$ to n, $\mathcal{S}$ chooses $r_i, t_i, u_i, \bar{\alpha}_i \in \mathbb{Z}$ of length l_s at random, where $\gcd(\bar{\alpha}_i, e_{j^*}) = 1$, puts $E_i := E\bar{\alpha}_i$, and puts $g_{i,2} :\equiv g^{E_i}, g_{i,1} :\equiv g_{i,2}^{r_i}, g_{i,0} :\equiv g_{i,2}^{e_{j^*} t_i - u_i}$. $\mathcal{S}$ sets $\mathrm{PK} := (N, (g_{i,0}, g_{i,1}, g_{i,2})_{i=1}^n)$ and give PK to $\mathcal{F}$.

For $j \neq j^*$, the simulation of $\mathcal{SBSIGN}$ for a query $(m_{j,i})_i$ issued by $\mathcal{F}$ goes in the same way as in [CL02].

For j^*, $\mathcal{S}$ puts $s_i := u_i - r_i m_{j^*,i}$ and $A_i := g_{i,2}^{t_i}$ for each i. Note that the following holds.

$$A_i^{e_{j^*}} = (g_{i,2}^{t_i})^{e_{j^*}} = g_{i,2}^{e_{j^*} t_i - u_i + u_i} = g_{i,2}^{e_{j^*} t_i - u_i + u_i} = g_{i,0} g_{i,2}^{r_i m_{j^*,i} + s_i} = g_{i,0} g_{i,1}^{m_{j^*,i}} g_{i,2}^{s_i}.$$

When $\mathcal{F}$ returns a forgery $(m_i^*)_{i=1}^{n^*}, (\tau^* = e^*, (\sigma_i^* = (s_i^*, A_i^*))_{i=1}^{n^*})$, the extraction of an answer to the instance goes in the same way as in [CL02]. Note that $e^* = e_{j^*}$ holds with at least a non-negligible probability $1/q$. $\square$

Our ABID in RSA Using CL-CB as Witness

ABID.Setup$(1^\lambda, \mathcal{U}) \to (\mathrm{MSK}, \mathrm{PK})$. Given the security parameter 1^λ and an attribute universe $\mathcal{U}$, it chooses a special RSA modulus $N = pq, p = 2p' + 1, q = 2q' + 1$ of length $l_N = 2\lambda$. For $i \in \mathcal{U}$, it chooses $g_{i,0}, g_{i,1}, g_{i,2} \in_R QR_N$ and a hash key $\mu \in_R \mathit{Hashkeysp}(\lambda)$ of a hash function Hash_μ with the value in $\mathbb{Z}_{\phi(N)}$. It puts $\mathrm{PK} := (N, (g_{i,0}, g_{i,1}, g_{i,2})_{i \in \mathcal{U}}, \mu, \mathcal{U})$ and $\mathrm{MSK} := p$. It returns PK and MSK.

ABID.KG$(\mathrm{MSK}, \mathrm{PK}, S) \to \mathrm{SK}_S$. Given PK, MSK and an attribute subset S, it chooses a prime e of length l_e. For $i \in S$, it computes $a_i \leftarrow \mathit{Hash}_\mu(i)$, $s_i \in_R \mathbb{Z}$ of length l_e, $A_i := (g_{i,0} g_{i,1}^{a_i} g_{i,2}^{-s_i})^{\frac{1}{e}}$. It puts $\mathrm{SK}_S := (e, (s_i, A_i)_{i \in S})$.

$\mathcal{P}(\mathrm{SK}_S, \mathrm{PK}, f)$ and $\mathcal{V}(\mathrm{PK}, f)$ execute $\mathbf{\Sigma}_f$ with the following precomputation. For $i \in \mathrm{Att}(f)$, $\mathcal{P}$ chooses $r_i \in_R \mathbb{Z}$ of length l_e. If $i \in S$ then $s'_i := s_i + e r_i$, $A'_i := A_i g_{i,2}^{-r_i}$. Otherwise $s'_i \in_R \mathbb{Z}$ of length l_e, $A'_i \in_R \mathbb{Z}_N^*$. $\mathcal{P}$ puts

$$Z_i := g_{i,0} g_{i,1}^{a_i}, Z_{i,1} := A'_i, Z_{i,2} := g_{i,2}.$$

Then the statement for $\mathbf{\Sigma}_f$ is $x := (x_i := (Z_i, Z_{i,1}, Z_{i,2}))_i$ and the witness is $w := (\tau := e, (w_i := s'_i)_i)$, where $i \in \mathrm{Att}(f)$ for x and w. $\mathcal{P}$ sends the re-randomized values $(A'_i)_i$ to $\mathcal{V}$ for $\mathcal{V}$ to be able to compute the statement x.

After the above precomputation, $\mathcal{P}$ and $\mathcal{V}$ can execute $\mathbf{\Sigma}_f$ on the relation R_f. In other words, $\mathcal{P}$ and $\mathcal{V}$ execute $\mathbf{ZKPoK}[(e_{\rho(l)}, s'_{\rho(l)})_{l \in \mathrm{Leaf}(\mathcal{T}_f)}, : \mathit{equations}]$, for the language L_f, where the *equations* are:

$$Z_{\rho(l)} = Z_{\rho(l),1}^{e_{\rho(l)}} Z_{\rho(l),2}^{s'_{\rho(l)}}, l \in \mathrm{Leaf}(\mathcal{T}_f). \tag{9}$$

Note that $\mathcal{V}$ verifies whether or not the verification equations hold for all the leaves:

$$\mathrm{CMT}_l = Z_{\rho(l)}^{-\mathrm{CHA}_l} Z_{\rho(l),1}^{\theta_{e,l}} Z_{\rho(l),2}^{\theta_{s',l}}, l \in \mathrm{Leaf}(\mathcal{T}_f). \tag{10}$$

$\mathcal{V}$ returns 1 or 0 accordingly.

Security of Our ABID

Claim 1 (Concurrent Security under a Single Tag) *Our* ABID *is secure against concurrent attacks if our credential-bundle scheme* CB *is existentially unforgeable against chosen-message attacks and if the extracted values e by the extractor of the underlying Σ-protocol $\mathbf{\Sigma}_f$ is a common single value.*

Proof. All the answers of the oracles to queries of a PPT adversary $\mathcal{A}$ on ABID can be perfectly simulated by using the oracles of CB. As for the extraction of a credential bundle, we can do it under the condition that the extracted value e is a common single value. □

Note that Claim 1 is needed only as an intermediate result. That is, the assumption that the extracted value e is a common single value is assured by the two-tier key-issuer, **ABTTS.SKG**, in the next section.

Our ABTTS in RSA Using CL-CB as Witness

ABTTS.Setup and **ABTTS.PKG** are the same as **ABID.Setup** and **ABID.KG** in Section G.2, respectively. **ABTTS.SKG**, **ABTTS.Sign** and **ABTTS.Vrfy** are obtained along the design principle of two-tier signature schemes for the canonical identification schemes [BS07]. That is, on input MSK, PK, a primary secret key SK_S and an access formula f, **ABTTS.SKG** first computes a statement x and a corresponding witness w. Then, on input (x, w), the prover $\mathcal{P}$ is executed in **ABTTS.SKG** to obtain the commitment $(\mathrm{CMT}_l)_l$, and the inner state st of $\mathcal{P}$ with the commitment is included in the secondary secret key; $\mathrm{SSK}_{S,f} := (w, (\mathrm{CMT}_l)_l \parallel st)$, $\mathrm{SPK}_f := (x, (\mathrm{CMT}_l)_l)$. **ABTTS.Sign** and **ABTTS.Vrfy** run the remaining protocol of our ABID in the two-tier framework [BS07] as in Section 7. The signature is:

$$\sigma := ((\mathrm{CHA}_n)_n, (\mathrm{RES}_l)_l).$$

Security of Our ABTTS in RSA Using CL-CB

Theorem 9 (Unforgeability) *Our attribute-based two-tier signature scheme* ABTTS′ *is existentially unforgeable against chosen-message attacks under the Strong RSA assumption in the standard model.*

Proof. According to the same discussion in Bellare et al. [BS07] as well as Theorem 8 and Claim 1, we deduce the claim. □

Theorem 10 (Attribute Privacy) *Our attribute-based two-tier signature scheme* ABTTS′ *has attribute privacy.*

Proof. The witness-indistinguishability of $\mathbf{\Sigma}_f$ assures the attribute privacy. □

G.3 Our ABID and ABTTS in Discrete Log Using CL Credential-Bundle as Witness

Strong Diffie-Hellman Assumption [BB04a] Let p denote a prime of bit length λ. Let $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ denote bilinear groups of order p, where $\mathbb{G}_1$ is generated by g, $\mathbb{G}_2$ is generated by h and $\mathbb{G}_T$ is generated by $e(g,h) \neq 1_{\mathbb{G}_T}$. We denote the probabilistic algorithm that generates such parameters params $:= (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ on input 1^λ as $\texttt{BlGrp}$. Let q denote a number that is less than a fixed polynomial in λ. The strong Diffie-Hellman assumption [BB04a] states that for any PPT $\mathcal{A}$, the following advantage is negligible in λ: $\mathbf{Adv}^{\mathrm{sdh}}_{\texttt{BlGrp},\mathcal{S}}(\lambda) := \Pr[\mathrm{params} \leftarrow \texttt{BlGrp}(1^\lambda), \alpha \in_R \mathbb{Z}_p, \;\; (u,e) \leftarrow \mathcal{A}(\mathrm{params}, (g, g^\alpha, g^{\alpha^2}, \ldots, g^{\alpha^q}, h, h^\alpha)) : u^{\alpha+e} = g]$.

CL Credential-Bundle in DL

We propose a credential-bundle scheme in the discrete-logarithm setting by modifying the pairing-based CL signature scheme [TF12,FI05,Oka06]. Our pairing-based credential-bundle scheme, $\texttt{CB} = (\mathbf{CB}.\mathbf{KG}, \mathbf{CB}.\mathbf{Sign}, \mathbf{CB}.\mathbf{Vrfy})$, is described as follows.

$\mathbf{CB}.\mathbf{KG}(1^\lambda) \to (\mathrm{PK}, \mathrm{SK})$. Given 1^λ as input, it runs a group generator $\texttt{BlGrp}(1^\lambda)$ to get $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e(\cdot,\cdot))$. For $i = 1$ to n, it chooses $g_{i,0}, g_{i,1}, g_{i,2} \in_R \mathbb{G}_1, h_0 \in_R \mathbb{G}_2, \alpha \in_R \mathbb{Z}_p$ and it puts $h_1 := h_0^\alpha$. It puts $\mathrm{PK} := ((g_{i,0}, g_{i,1}, g_{i,2})_{i=1}^n, h_0, h_1)$ and $\mathrm{SK} := \alpha$, and returns $(\mathrm{PK}, \mathrm{SK})$.

$\mathbf{CB}.\mathbf{Sign}(\mathrm{PK}, \mathrm{SK}, (m_i)_{i=1}^n) \to (\tau, (\sigma_i)_{i=1}^n)$. Given PK, SK and messages $(m_i)_{i=1}^n$ each of which is of length $l_{\mathcal{M}}$, it chooses $e \in_R \mathbb{Z}_p$. For $i = 1$ to n, it chooses $s_i \in_R \mathbb{Z}_p$, and it computes the value A_i:

$$A_i := (g_{i,0} g_{i,1}^{m_i} g_{i,2}^{s_i})^{\frac{1}{\alpha+e}}. \tag{11}$$

It puts $\tau = e$ and $\sigma_i = (s_i, A_i)$ for each i and returns $(\tau, (\sigma_i)_{i=1}^n)$.

$\mathbf{CB}.\mathbf{Vrfy}(\mathrm{PK}, (m_i)_{i=1}^n, (\tau, (\sigma_i)_{i=1}^n)) \to 1/0$. Given PK, $(m_i)_{i=1}^n$ and $(\tau, (\sigma_i)_{i=1}^n)$, it verifies whether the following holds:

$$e(A_i, h_0^e h_1) = e(g_{i,0} g_{i,1}^{m_i} g_{i,2}^{s_i}, h_0), \;\; i = 1, \ldots, n. \tag{12}$$

Theorem 11 (Unforgeability of Our CB) *Our credential-bundle scheme $\texttt{CB}$ is existentially unforgeable against chosen-message attack under the Strong Diffie-Hellman assumption.*

Proof. Everything can be done as in [Oka06] except the following slight enhancement.

$\mathcal{S}$ chooses q elements $e_j \in \mathbb{Z}_p, j = 1, \ldots, q$, at random. Then $\mathcal{S}$ chooses $j^* \in \{1, \ldots, q\}$ at random and puts:

$$f(X) := \prod_{j \in S} (X + e_j), f_{j^*}(X) := f(X)/(X + e_{j^*}).$$

Then, for each $i = 1$ to n, $\mathcal{S}$ chooses $r_i, t_i, u_i, \bar{\alpha_i} \in_R \mathbb{Z}_p$ and implicitly puts $\alpha_i := \bar{\alpha_i}\alpha$, and puts $g_{i,2} := g^{f_{j^*}(\alpha_i)}$, $g_{i,1} := g_{i,2}^{r_i}$, $g_{i,0} := g_{i,2}^{(\alpha_i + e_{i^*})t_i - u_i} = (g^{f_{j^*}(\alpha_i)})^{(\alpha_i + e_{j^*})t_i - u_i} = g^{f(\alpha_i)t_i} g^{-u_i f_{j^*}(\alpha_i)}$, $s_{j^*} := u_i - r_i m_{j^*}$, $A_{j^*} := g_{i,2}^{t_i}$. Then,

$$A_{j^*}^{\alpha_i + e_{j^*}} = (g_{i,2}^{t_i})^{\alpha_i + e_{j^*}} = g_{i,2}^{(\alpha_i + e_{j^*})t_i - u_i + u_i} = g_{i,0} g_{i,2}^{u_i} = g_{i,0} g_{i,2}^{r_i m_{j^*} + s_{j^*}} = g_{i,0} g_{i,1}^{m_{j^*}} g_{i,2}^{s_{j^*}}.$$

This completes the simulation of the credential-bundle oracle $\mathcal{SBSIGN}$.

The extraction of the answer to an instance of the Strong Diffie-Hellman assumption can be done in the same way as [Oka06] with division by $\bar{\alpha_i}$. □

Our ABID in DL Using CL-CB as Witness

$\mathbf{ABID}.\mathbf{Setup}(1^\lambda, \mathcal{U}) \to (\mathrm{MSK}, \mathrm{PK})$. Given the security parameter 1^λ and an attribute universe $\mathcal{U}$, it executes a group generator $\texttt{BlGrp}(1^\lambda)$ to get $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e(\cdot,\cdot))$. For $i \in \mathcal{U}$, it chooses $g_{i,0}, g_{i,1}, g_{i,2} \in_R \mathbb{G}_1, h_0 \in_R \mathbb{G}_2, \alpha \in_R \mathbb{Z}_p, h_1 := h_0^\alpha$ and a hash key $\mu \in_R \mathit{Hashkeysp}(\lambda)$ of a hash function Hash_μ with the value in $\mathbb{Z}_p$. It puts $\mathrm{PK} := ((g_{i,0}, g_{i,1}, g_{i,2})_{i \in \mathcal{U}}, h_0, h_1, \mu, \mathcal{U})$ and $\mathrm{MSK} := \alpha$. It returns PK and MSK.

$\mathbf{ABID}.\mathbf{KG}(\mathrm{MSK}, \mathrm{PK}, S) \to \mathrm{SK}_S$. Given PK, MSK and an attribute subset S, it chooses $e \in_R \mathbb{Z}_p$. For $i \in S$, it computes $a_i \leftarrow \mathit{Hash}_\mu(i)$, $s_i \in_R \mathbb{Z}_p$, $A_i := (g_{i,0} g_{i,1}^{a_i} g_{i,2}^{-s_i})^{\frac{1}{\alpha+e}} \in \mathbb{G}_1$. It puts $\mathrm{SK}_S := (e, (s_i, A_i)_{i \in S})$.

$\mathcal{P}(\mathrm{SK}_S, \mathrm{PK}, f)$ and $\mathcal{V}(\mathrm{PK}, f)$ execute $\mathbf{\Sigma}_f$ with the following precomputation. For $i \in \mathrm{Att}(f)$, $\mathcal{P}$ chooses $r_i \in_R \mathbb{Z}_p$. If $i \in S$ then $s_i' := s_i + e r_i, A_i' := A_i g_{i,2}^{-r_i} \in \mathbb{G}_1$. Otherwise $s_i' \in_R \mathbb{Z}_p$, $A_i' \in_R \mathbb{G}_1$. $\mathcal{P}$ puts

$$Z_i := e(g_{i,0} g_{i,1}^{a_i}, h_0) e(A_i', h_1)^{-1}, Z_{i,1} := e(A_i', h_0), Z_{i,2} := e(g_{i,2}, h_0), Z_{i,3} := e(g_{i,2}, h_1).$$

Then the statement for $\mathbf{\Sigma}_f$ is $x := (x_i := (Z_i, Z_{i,1}, Z_{i,2}, Z_{i,3}))_i$ and the witness is $w := (\tau := e, (w_i := s'_i)_i)$, where $i \in \mathrm{Att}(f)$ for x and w. $\mathcal{P}$ sends the re-randomized values $(A'_i)_i$ to $\mathcal{V}$ for $\mathcal{V}$ to be able to compute the statement x.

After the above precomputation, $\mathcal{P}$ and $\mathcal{V}$ can execute $\mathbf{\Sigma}_f$ on the relation R_f. In other words, $\mathcal{P}$ and $\mathcal{V}$ execute $\mathbf{ZKPoK}[(e_{\rho(l)}, s'_{\rho(l)})_{l \in \mathrm{Leaf}(\mathcal{T}_f)} : \mathit{equations}]$, for the language L_f, where the *equations* are:

$$Z_{\rho(l)} = Z_{\rho(l),1}^{e_{\rho(l)}} Z_{\rho(l),2}^{s'_{\rho(l)}} Z_{\rho(l),3}^{r_{\rho(l)}}, l \in \mathrm{Leaf}(\mathcal{T}_f). \tag{13}$$

Note that $\mathcal{V}$ verifies whether or not the verification equations hold for all the leaves:

$$\mathrm{C{\scriptsize MT}}_l = Z_{\rho(l)}^{-\mathrm{C{\scriptsize HA}}_l} Z_{\rho(l),1}^{\theta_{e,l}} Z_{\rho(l),2}^{\theta_{s',l}} Z_{\rho(l),3}^{\theta_{r,l}}, l \in \mathrm{Leaf}(\mathcal{T}_f). \tag{14}$$

$\mathcal{V}$ returns 1 or 0 accordingly.

Security of Our ABID

Claim 2 (Concurrent Security under a Single Tag) *Our* `ABID` *is secure against concurrent attacks if our credential-bundle scheme* `CB` *is existentially unforgeable against chosen-message attacks and if the extracted values* e *by the extractor of the underlying* Σ*-protocol* $\mathbf{\Sigma}_f$ *is a common single value.*

Proof. All the answers of the oracles to queries of a PPT adversary $\mathcal{A}$ on `ABID` can be perfectly simulated by using the oracles of `CB`. As for the extraction of a credential bundle, we can do it under the condition the extracted value e is a common single value. □

Note that Claim 2 is needed only as an intermediate result. That is, the assumption that the extracted value e is a common single value is assured by the two-tier key-issuer, **ABTTS**.**SKG**, in the next section.

Our ABTTS in DL Using CL-CB as Witness

ABTTS.**Setup** and **ABTTS**.**PKG** are the same as **ABID**.**Setup** and **ABID**.**KG** in Section G.2, respectively. **ABTTS**.**SKG**, **ABTTS**.**Sign** and **ABTTS**.**Vrfy** are obtained along the design principle of two-tier signature schemes for the canonical identification schemes [BS07]. That is, on input MSK, PK, a primary secret key SK_S and an access formula f, **ABTTS**.**SKG** first computes a statement x and a corresponding witness w. Then, on input (x, w), the prover $\mathcal{P}$ is executed in **ABTTS**.**SKG** to obtain the commitment $(\mathrm{C{\scriptsize MT}}_l)_l$, and the inner state *st* of $\mathcal{P}$ with the commitment is included in the secondary secret key; $\mathrm{SSK}_{S,f} := (w, (\mathrm{C{\scriptsize MT}}_l)_l \parallel st)$, $\mathrm{SPK}_f := (x, (\mathrm{C{\scriptsize MT}}_l)_l)$. **ABTTS**.**Sign** and **ABTTS**.**Vrfy** run the remaining protocol of our `ABID` in the two-tier framework [BS07] as in Section 7. The signature is:

$$\sigma := ((\mathrm{C{\scriptsize HA}}_n)_n, (\mathrm{R{\scriptsize ES}}_l)_l).$$

Security of Our ABTTS in DL Using CL-CB

Theorem 12 (Unforgeability) *Our attribute-based two-tier signature scheme* `ABTTS'` *is existentially unforgeable against chosen-message attacks under the Strong Diffie-Hellman assumption in the standard model.*

Proof. According to the same discussion in Bellare et al. [BS07] as well as Theorem 11 and Claim 2, we deduce the claim. □

Theorem 13 (Attribute Privacy) *Our attribute-based two-tier signature scheme* `ABTTS'` *has attribute privacy.*

Proof. The witness-indistinguishability of $\mathbf{\Sigma}_f$ assures the attribute privacy. □