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 of *"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 of *"Attribute-Based Two-Tier Signatures: Definition and Construction"*. This is the full version. The statement on attribute privacy has been corrected.

Proof 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

May 19, 2016

Abstract. We propose a concrete procedure of a Σ-protocol to prove knowledge that satisfies a monotone predicate. Inspired by the high-level proposal by Cramer, Damgård and Schoenmakers at CRYPTO '94, we construct the procedure by extending the so-called OR-proof. Next, using as a witness a signature-bundle scheme of the Fiat-Shamir signature, 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. The latter scheme has a feature of linkable signatures. Finally, applying the two-tier technique of Bellare et al. 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. When instantiated in the RSA setting and the Discrete-Logarithm setting, these schemes are pairing-free.

Keywords: sigma-protocol, proof of knowledge, access structure, Fiat-Shamir transform, two-tier keys

1 Introduction

A Σ-protocol formalized in the doctoral thesis of Cramer [12] is a protocol of a three-round public-coin interactive proof system with completeness, special soundness and honest-verifier zero-knowledge. It is one of the simplest protocols of zero-knowledge interactive proof systems. Simple instantiations of a Σ-protocol have been known as the Schnorr protocol [44] and the Guillou-Quisquater protocol [26]. Also,

* The first and the second authors are partially supported by *kakenhi* Grant-in-Aid for Scientific Research (C) 15K00029 from Japan Society for the Promotion of Science.

a Σ-protocol is a typical proof-of-knowledge system [5]. 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 reduction to a number-theoretic assumption [6]. Besides, an identification scheme by a Σ-protocol can be converted into a signature scheme by the Fiat-Shamir heuristic [19]. The signature scheme can be proved secure against chosen-message attacks in the random oracle model [41], based on the security of the identification scheme against passive attacks [1]. By virtue of these features, a Σ-protocol can be adopted into building blocks of various cryptographic primitives such as anonymous credential systems [11] and group signature schemes [10].

The OR-proof proposed by Cramer, Damgård and Schoenmakers at CRYPTO '94 [13] is a Σ-protocol derived from an original Σ-protocol [14]. It is a witness-hiding protocol [18] by which a prover can convince a verifier that the prover knows one of two (or both) witnesses *hiding* which witness is used. The OR-proof is essentially applied in, for example, the construction of a non-malleable proof of plaintext knowledge [33]. In the paper of Cramer et al. [13], 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 provides a witness hiding protocol in the sense that the prover knows a satisfying set of witnesses *hiding* which satisfying pattern is used. We call the protocol a *boolean proof*. The boolean proof is an extension of the OR-proof to any monotone boolean predicate, and in [13] a high-level construction that employed a "semi-smooth" secret-sharing scheme was given. (As is explained in [13], to remove the restriction of the monotonicity of f looks hard.)

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

Then, we will try to apply our procedure of the boolean proof, $\mathbf{\Sigma}_f$, to construct an attribute-based identification scheme (ABID) as well as an attribute-based signature scheme (ABS) obtained by applying the Fiat-Shamir heuristic to ABID. In ABID, an identification-session is associated with an access structure, where a prover can make a verifier accept only when the prover's set of attributes satisfies the access structure. The access structure is described as a boolean predicate over an attribute universe.

As for an attribute-based signature scheme (ABS), which has been developed since 2008 [27, 45, 36, 35, 37, 32, 17, 39, 21, 31, 40, 28, 16, 15, 16, 22, 29, 43], almost all the constructions are via the approach similar to that of attribute-based encryption schemes (ABE) (for instance, [42]), which uses bilinear maps (that is, pairings) on elliptic curves. Only a few exception are generic constructions by Maji et al. [37] and Bellare et al. [4], and constructions by Herranz in the RSA setting [28] and in the discrete logarithm setting [29].

In contrast to the approach by bilinear maps, we work through a different approach in the Fiat-Shamir paradigm [19], which shares a spirit with [28]. Note that, in this paper, we do not try to attain the property of (usual) attribute-privacy [37, 39, 28] which means that signatures reveals nothing about the identity or attributes of the signer beyond what is explicitly revealed by the satisfied boolean predicate.

[5] In the preliminary version [3], the authors could not refer to this previous work. Now we refer to the work with explanation on the relation.

1.1 Our Construction Idea

To provide a concrete procedure for the above boolean proof system from a given Σ-protocol and a monotone boolean predicate f, we look into the technique employed in the OR-proof [13] 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 assign the string $\textsc{Cha}$ to the root node. If the root node is an $\wedge$-node, then assign the same string $\textsc{Cha}$ to two children. Else if the root node is an $\vee$-node, then divide $\textsc{Cha}$ 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 assign $\textsc{Cha}_{\mathrm{L}}$ and $\textsc{Cha}_{\mathrm{R}}$ to the left child and the right child, respectively. Here $\oplus$ means a bitwise exclusive-OR operation. Then continue to apply this rule at each height, step by step, until we reach to every leaf. Then, 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 boolean proof [13], which is comparable with the original abstract protocol [13]. 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. Especially we show that $\mathbf{\Sigma}_f$ is a protocol to prove knowledge of witnesses that satisfy the boolean predicate f.

Our second contribution is to provide a concrete attribute-based identification scheme (ABID) and a concrete attribute-based signature scheme (ABS), without pairings in both the Discrete-Logarithm setting and the RSA setting. The constructions are by employing the Schnorr identification scheme and the GQ identification scheme [44, 6] as $\mathbf{\Sigma}$, respectively. We again note that signatures of our ABS are linkable, and attribute privacy only holds as a one-time signature.

1.3 Related Work on ABS

At a high level, our ABS is obtained by the Fiat-Shamir transform of our boolean proof system, where a set of witnesses is the Fiat-Shamir signature bundle (credential bundle [37]). This construction can be compared with the generic construction of the ABS scheme by Maji et al. [37]. They started with a signature bundle (of Boneh-Boyen signatures [9], for instance). Then they employed a non-interactive witness-indistinguishable proof of knowledge system (NIWIPoK) of Groth and Sahai [25] to prove the knowledge of a signature bundle which satisfies a given (monotone) access formula, in the standard model.

Okamoto and Takashima (OT11) [39] gave a scheme of ABS with full-security; security against adaptive target in the standard model under a non-q-type assumption. It can treat 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 [28] provided the first ABS with both collusion resistance (against collecting private secret keys) and (computationally secure) attribute privacy without pairings (pairing-free) in the RSA setting. In the work [28], the concrete procedure was described in detail for threshold-type access formulas. In contrast, our ABS is without pairings and provide a concrete procedure for any access formulas, but does not achieve attribute privacy. Recently, Herranz [29] provided an ABS scheme without pairings in

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.			q-SDH $\wedge$			✓	-		$(2\lambda)\times$	-
[37]	Mono.	Std.	DLIN $\wedge$ CR	✓	✓	(info.)	-	-	$(51l+2r+18\lambda l)$	
OT	Non-		DLIN			✓	-			-
[39]	mono.	Std.	$\wedge$ CR	✓	✓	(info.)	-	-	$(2\lambda)(9l+11)$	
Herranz			q-SRSA $\wedge$			✓	-		$\lambda_{\mathrm{rsa}}(5+\frac{\kappa}{\lambda_{\mathrm{rsa}}})l$	-
[28]	Mono.	R.O.	DDH $\wedge$CR	✓	✓	(comp.)	-	✓	$+\lambda_{\mathrm{rsa}}3-\kappa(\theta-1)$	
Herranz			DL			✓	-		$(2\lambda)l+\lambda(6l-\theta)$	bounded
[29]	Mono.	R.O.	$\wedge$CR	✓	✓	(info.)	-	✓	$+\lambda M(l+1)$	num. keys
Kaafarani			q-SDH $\wedge$ DDH				✓		$(2\lambda)(3l+r+3)$	-
et al. [15]	Mono.	R.O.	$\wedge$DL $\wedge$ CR	✓	✓	-	✓	-	$+\lambda(8l+4)$	
Our ABS			DL				✓		$(2\lambda)(2l)$	-
	Mono.	R.O.	$\wedge$ CR	✓	✓	-	-	✓	$+\lambda 3l$	
Our ABTTS			DL			✓	-			two-tier
(FS-sig.)	Mono.	R.O.	$\wedge$ CR	✓	✓	(info.)	-	✓	$\lambda(3l-1)$	keys
Our ABTTS$'$			q-SDH			✓	-			two-tier
(CL-sig.)	Mono.	Std.	$\wedge$ CR	✓	✓	(info.)	-	-	$\lambda(3l-1)$	keys

the discrete-logarithm setting, but it has a constraint that the number of secret keys is bounded in the set-up phase.

Kaafarani et al. [15] proposed the functionality of "User-Controlled Linkability" (UCL) in the case of attribute-based signatures. UCL property in the work [15] 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 [15] and our ABS do not possess attribute privacy.

1.4 Technical and Efficiency Comparison on ABS: Security, Functionality and Length of Signature

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 [24], 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 for bilinear groups [8]. DLIN means the Decisional Linear assumption for bilinear groups [39]. DDH means the Decisional Diffie-Hellman assumption for a cyclic group [15]. DL means the Discrete-Logarithm assumption for a cyclic group [15]. q-SRSA means the strong RSA assumption with q-type input [11, 28]. DDH in $QR(N)$ means the Decisional Diffie-Hellman assumption for quadratic residues modulo N (the RSA modulus) [28]. In [28, 29], θ is the threshold value of a threshold-type access structure. In [28], κ is a security parameter. In [29], $M=L+N$ is the sum of the bounded number L of users in the set-up phase and the number N of all attributes in the attribute universe. "info." means the information-theoretic security and "comp." means the computational security. "FS-sig." means a scheme that uses the Fiat-Shamir signatures [19] as a witness and "CL-sig." means a scheme that uses the Camenisch-Lysyanskaya signatures [11] as a witness.

The ABS scheme by Maji et al. can be said as the pioneering work. The ABS scheme by Okamoto and Takashima [39] has advantages in the security-proof model, access formula and information-theoretically secure attribute privacy. The scheme by Herranz [28] is the only ABS scheme with collusion resistance, (computational) attribute privacy and pairing-free property, in the RSA setting. Our procedure $\mathbf{\Sigma}_f$ of the boolean proof [13] for any monotone predicate serves as a building block of (the Σ-protocol of) the ABS scheme [28]. Note that the security parameter λ_{rsa} in the RSA setting ([28], our `ABS` in RSA, our `ABTTS` in RSA and our `ABTTS′` in RSA) is almost 9 times longer than λ in the discrete logarithm setting. For example, $\lambda_{\mathrm{rsa}} = 2048$ is almost equivalent to $\lambda = 224$-bit security [47].

Note that the ABS scheme by Herranz [29] 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, `ABTTS` and `ABTTS′`, are in the two-tier setting which means that a secondary secret key and a secondary public key are issued for each signing session and the secondary keys can be used for only one-time use. 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. [15] has a feature of the user-controlled linkability. In contrast, our `ABS` has only the public linkability. It is notable that the ABS scheme [15] uses pairings and can be set up in the multi-authorities setting [40, 16, 22].

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 boolean proof system, $\mathbf{\Sigma}_f$. In Section 4, by using a signature-bundle scheme of the Fiat-Shamir signature FS($\mathbf{\Sigma}$) as witnesses 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 7, by applying the technique of two-tier signature to our ABID, we obtain our ABTTS. In Section 9, we conclude our work in this paper. In Appendix A, we summarize the notion of NIWI proof of knowledge system. In Appendix B, we state a NIWIPoK system that is obtained from our $\mathbf{\Sigma}_f$. In Appendix 8 and C, 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. Bit length of a string x is denoted as $|x|$. 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 to oracles in arbitrarily interleaved order of messages. We denote a concatenation of a string a with a string b as $a \parallel b$. The expression $a \stackrel{?}{=} b$ returns a value 1 (True) when $a = b$ and 0 (False) otherwise. The expression $a \stackrel{?}{\in} S$ returns a value 1 when $a \in S$ and 0 otherwise. A probability of an event E is denoted by Pr[E]. A 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 [5, 13, 14]

Language Let $R = \{(x, w)\} \subset \{1, 0\}^* \times \{1, 0\}^*$ be a binary relation. We say that R is polynomially bounded if there exists a polynomial *poly* such that $|w| \leq poly(|x|)$ for all $(x, w) \in R$. If $(x, w) \in R$ then we call x a statement and w a witness of x. 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 in R.

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

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

L_R 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:

$$R(\cdot,\cdot) : \{1,0\}^* \times \{1,0\}^* \to \{1,0\},$$
$$(x,w) \mapsto 1 \text{ if } (x,w) \in R,\ 0 \text{ otherwise.}$$

Proof of Knowledge A *proof of knowledge system* (PoK for short) $\Pi = (\mathcal{P}, \mathcal{V})$ for a language L_R is a protocol between interactive PPT algorithms $\mathcal{P}$ and $\mathcal{V}$ on initial input $(x,w) \in R$ for $\mathcal{P}$ and x for $\mathcal{V}$, where $\mathcal{V}$ outputs 1 (accept) or 0 (reject) after finite rounds of interaction. $\mathcal{P}$ is called a prover and $\mathcal{V}$ is called a verifier. In general, a prover $\mathcal{P}$ has unbounded computational power, but in this paper we only consider the case that $\mathcal{P}$ is PPT.

Π must possess the following two properties.

Completeness. For any statement $x \in L_R$ and for any witness w such that $(x,w) \in R$, $\mathcal{P}$ with the witness w can make $\mathcal{V}$ accept for the statement x with probability 1:

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

Knowledge Soundness. There are a PPT algorithm $\mathcal{KE}$ called a *knowledge extractor*, a function $\kappa : \{1,0\}^* \to [1,0]$ 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 $\mathcal{KE}(x)$, employing $\mathcal{A}(x)$ as a subroutine that allows to be rewinded[6], outputs a witness w which satisfies $(x,w) \in R$ within an expected number of steps bounded by: $|x|^c/(p(x) - \kappa(x))$.

Σ-protocol [12, 14] A *Σ-protocol* on a relation R is a public coin 3-move protocol between interactive PPT algorithms $\mathcal{P}$ and $\mathcal{V}$ on initial input $(x,w) \in R$ for $\mathcal{P}$ and x for $\mathcal{V}$. $\mathcal{P}$ sends the first message called a commitment $\textsc{Cmt}$, then $\mathcal{V}$ sends a random bit string called a challenge $\textsc{Cha}$, and $\mathcal{P}$ answers with a third message called a response $\textsc{Res}$. Then $\mathcal{V}$ applies a decision test on $(x, \textsc{Cmt}, \textsc{Cha}, \textsc{Res})$ to return accept (1) or reject (0). If $\mathcal{V}$ accepts, then the triple $(\textsc{Cmt}, \textsc{Cha}, \textsc{Res})$ is said to be an *accepting conversation.* $\textsc{Cha}$ is chosen uniformly at random from $\textsc{ChaSp}(1^\lambda) := \{1,0\}^{l(\lambda)}$ with $l(\cdot)$ being a super-log function.

This protocol is written by a PPT algorithm $\mathbf{\Sigma}$ as follows. $\textsc{Cmt} \leftarrow \mathbf{\Sigma}^1(x,w)$: the process of selecting 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 properties.

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, a cheating prover, can only respond for 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 the witness w, 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 $\mathbf{ZKPoK}[\gamma : \Gamma]$, where γ is a knowledge to be proved and Γ is the condition that γ should satisfy.

Any Σ-protocol can be proved to be a proof of knowledge system ([14]).

[6] In [5], it is described as "oracle-access to $\mathcal{A}_x$" instead of rewinding, which is more general statement but we do not need the generality in this paper.

We will need in this paper a property called *unique answer property* [7] that for legitimately produced commitment $\textsc{Cmt}$ and challenge $\textsc{Cha}$, there exists one and only one response $\textsc{Res} =: w'$ that is accepted by a verifier. Known Σ-protocols such as the Schnorr protocol and the Guillou-Quisquater protocol [44, 6] possess this property. For such a unique answer w' we consider a statement x' such that $(x', w') \in R$. Then, we further assume that both a prover and a verifier can compute, in polynomial-time, such an x' from $(x, \textsc{Cmt}, \textsc{Cha})$. We denote the PPT algorithm as $\mathbf{\Sigma}^{\text{stmtgen}}$. That is;

$\mathbf{\Sigma}^{\text{stmtgen}}(x, \textsc{Cmt}, \textsc{Cha})$:
Compute x' s.t.
$^{\exists 1}w'$ s.t. $[(x', w') \in R \ \wedge (\textsc{Cmt}, \textsc{Cha}, \textsc{Res} := w')$ is an accepting conversation$]$
Return x'

Known Σ-protocols [44, 6] possess this *statement generation property* (see Section 8 and Section C).

The OR-proof [14] Consider the following relation for a boolean predicate $f(X_1, X_2) = X_1 \vee X_2$.

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

The corresponding language for the relation R_{OR} is given as follows.

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

The OR-proof is defined as an interactive proof system for the language $L_{R_{\text{OR}}}$.

Suppose that a Σ-protocol $\mathbf{\Sigma}$ on a relation R is given. Then we can construct a new protocol, $\mathbf{\Sigma}_{\text{OR}}$, on a relation R_{OR} as follows. For instance, suppose $(x_0, w_0) \in R$ holds. $\mathcal{P}$ computes $\textsc{Cmt}_0 \leftarrow \mathbf{\Sigma}^1(x_0, w), \textsc{Cha}_1 \leftarrow \mathbf{\Sigma}^2(1^\lambda)$, $(\textsc{Cmt}_1, \textsc{Res}_1) \leftarrow \mathbf{\Sigma}^{\text{sim}}(x_1, \textsc{Cha}_1)$ 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 and have the same distribution as real accepting conversations. This protocol $\mathbf{\Sigma}_{\text{OR}}$ can be proved to be a Σ-protocol. We often call this Σ-protocol $\mathbf{\Sigma}_{\text{OR}}$ the *OR-proof*.

The Fiat-Shamir Transform [1] Suppose that a cryptographic hash function with collision resistance, $\mathit{Hash}_\mu(\cdot) : \{1,0\}^* \to \{1,0\}^{l(\lambda)}$, is given. We fix a hash key μ hereafter. A Σ-protocol $\mathbf{\Sigma}$ on a relation R can be transformed into a non-interactive zero-knowledge proof of knowledge system (NIZKPoK) with its knowledge extractor in the random oracle model. Hence a non-interactive witness-indistinguishable proof of knowledge system (NIWIPoK) can be obtained. When a Σ-protocol $\mathbf{\Sigma}$ is an identification scheme, the resulting scheme is a digital signature scheme. The transform is described as follows. (Here, a message m is omitted in the case of a NIWIPoK.) Given a message $m \in \{1,0\}^*$, execute: $a \leftarrow \mathbf{\Sigma}^1(x, w)$, $c \leftarrow \mathit{Hash}_\mu(a \parallel m)$, $z \leftarrow \mathbf{\Sigma}^3(x, w, a, c)$. Then $\sigma := (a, z)$ is a signature on m. We denote the above signing algorithm as $\text{FS}(\mathbf{\Sigma})^{\text{sign}}(x, w, m) \to (a, z) =: \sigma$. The verification algorithm $\text{FS}(\mathbf{\Sigma})^{\text{vrfy}}(x, m, \sigma)$ is given as: $c \leftarrow \mathit{Hash}_\mu(a \parallel m)$, Return $b \leftarrow \mathbf{\Sigma}^{\text{vrfy}}(x, a, c, z)$.

The signature scheme $\text{FS}(\mathbf{\Sigma}) = (R, \text{FS}(\mathbf{\Sigma})^{\text{sign}}, \text{FS}(\mathbf{\Sigma})^{\text{vrfy}})$ 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 [1]. More precisely, let q_H denote the maximum number of hash queries issued by the adversary on $\text{FS}(\mathbf{\Sigma})$. 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).

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

2.2 Signature-Bundle Scheme (Credential-Bundle Scheme [37])

A signature-bundle scheme $\mathtt{SB}$ is an extended notion of signature scheme. It consists of three algorithms: $\mathtt{SB} = (\mathbf{SB}.\mathbf{KG}, \mathbf{SB}.\mathbf{Sign}, \mathbf{SB}.\mathbf{Vrfy})$. Below n is bounded by a polynomial in λ.
$\mathbf{SB}.\mathbf{KG}(1^\lambda) \to (\mathrm{PK}, \mathrm{SK})$. This PPT algorithm for key generation takes as input 1^λ. It returns a public key PK and a secret key SK.
$\mathbf{SB}.\mathbf{Sign}(\mathrm{PK}, \mathrm{SK}, (m_1, \ldots, m_n)) \to (\tau, (\sigma_1, \ldots, \sigma_n))$. This PPT algorithm for signing takes as input PK, SK and n messages $m_1, \ldots, m_n$. It returns a *tag* τ and n signatures $\sigma_1, \ldots, \sigma_n$.
$\mathbf{SB}.\mathbf{Vrfy}(\mathrm{PK}, (m_1, \ldots, m_n), (\tau, (\sigma_1, \ldots, \sigma_n))) \to 1/0$. This deterministic polynomial-time algorithm for verification takes as input PK, n messages $m_1, \ldots, m_n$, a tag τ and n signatures $\sigma_1, \ldots, \sigma_n$. It returns 1 or 0.

Suppose that we are given a digital signature scheme (KG, Sign, Vrfy). Then we can construct a signature-bundle scheme as follows (according to [37]). $\mathbf{SB}.\mathbf{KG}$ takes as input 1^λ and it runs $\mathrm{KG}(1^\lambda)$ to get (PK, SK). it outputs (PK, SK). $\mathbf{SB}.\mathbf{Sign}$ takes as input PK, SK and a set of messages $(m_i)_{1\le i\le n}$. It chooses a tag τ of length λ at random. Then it executes Sign on each tagged message $(\tau \parallel m_i), i = 1, \ldots, n$ and outputs signatures $\sigma_i, i = 1, \ldots, n$, respectively. $\mathbf{SB}.\mathbf{Vrfy}$ takes as input PK, $(m_i)_{1\le i\le n}$, τ and $(\sigma_i)_{1\le i\le n}$. Then it executes Vrfy on each tagged message and signature, $((\tau \parallel m_i), \sigma_i), i = 1, \ldots, n$. It returns 1 if and only if Vrfy returns 1 for all i, $i = 1, \ldots, n$.

2.3 Pseudorandom Function Family [34]

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 [34].)

2.4 Access Structure [23]

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:

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

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:

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

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 only consider a *monotone* access formula f.[7]

[7] 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.

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 an attribute map $\rho(\cdot)$ is defined as:

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

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.

2.5 Attribute-Based Identification Scheme [2]

An attribute-based identification scheme, `ABID`, consists of four PPT algorithms [2]: $\mathtt{ABID} = (\mathbf{ABID}.\mathbf{Setup}, \mathbf{ABID}.\mathbf{KG}, \mathcal{P}, \mathcal{V})$.

$\mathbf{ABID}.\mathbf{Setup}(1^\lambda, \mathcal{U}) \to (\mathbf{PK}, \mathbf{MSK})$. This PPT algorithm for setting up 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.

$\mathbf{ABID}.\mathbf{KG}(\mathbf{PK}, \mathbf{MSK}, S) \to \mathbf{SK}_S$. 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 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 $\mathbf{ABID}.\mathbf{KG}(\mathrm{PK}, \mathrm{MSK}, S)$. $\mathcal{V}$ takes as input the public key PK and an access formula f. $\mathcal{P}$ and $\mathcal{V}$ interact with each other for at most constant rounds. 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 `ABID` that, for any λ, and if $f(S) = 1$, $\Pr[(\mathrm{PK}, \mathrm{MSK}) \leftarrow \mathbf{ABID}.\mathbf{Setup}(1^\lambda, \mathcal{U})$, $\mathrm{SK}_S \leftarrow \mathbf{ABID}.\mathbf{KG}(\mathrm{PK}, \mathrm{MSK}, S), b \leftarrow \langle \mathcal{P}(\mathrm{PK}, \mathrm{SK}_S), \mathcal{V}(\mathrm{PK}, f)\rangle : b = 1] = 1$.

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

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

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

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

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

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

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

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

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 $\mathcal{KG}(\mathrm{PK},\mathrm{MSK},\cdot)$ for the secret key SK_{S_i}. In addition, $\mathcal{A}$ invokes provers $\mathcal{P}_j(\mathrm{PK},\mathrm{SK}_{\cdot},\cdot)$, $j = 1,\dots,q'_{\mathrm{p}},\dots,q_{\mathrm{p}}$, 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 $P_j(\mathrm{PK},\mathrm{SK}_{S_j}, f_j)$ 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. Two restrictions are imposed on $\mathcal{A}$ concerning f^*. In key-extraction queries, each attribute set S_i must satisfy $f^*(S_i) = 0$. In interactions with each prover, $f^*(S_j) = 0$. The number of key-extraction queries and the number of invoked provers are at most q_{k} and q_{p} in total, respectively, which are bounded by a polynomial in λ.

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

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

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

$$\mathbf{Adv}^{\mathrm{pa}}_{\texttt{ABID},\mathcal{A}}(\lambda,\mathcal{U}) \leq \mathbf{Adv}^{\mathrm{ca}}_{\texttt{ABID},\mathcal{B}}(\lambda,\mathcal{U}). \tag{1}$$

2.6 Attribute-Based Signature Scheme [37, 39]

An attribute-based signature scheme, $\texttt{ABS}$, consists of four PPT algorithms [39]: $\texttt{ABS} =$ $(\mathbf{ABS}.\mathbf{Setup}, \mathbf{ABS}.\mathbf{KG}, \mathbf{ABS}.\mathbf{Sign}, \mathbf{ABS}.\mathbf{Vrfy})$.

$\mathbf{ABS}.\mathbf{Setup}(1^{\lambda},\mathcal{U}) \rightarrow (\mathbf{PK}, \mathbf{MSK})$. This PPT algorithm for setting up 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.

$\mathbf{ABS}.\mathbf{KG}(\mathbf{PK}, \mathbf{MSK}, S) \rightarrow \mathbf{SK}_S$. 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.

$\mathbf{ABS}.\mathbf{Sign}(\mathbf{PK}, \mathbf{SK}_S, (m, f)) \rightarrow \sigma$. 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.

$\mathbf{ABS}.\mathbf{Vrfy}(\mathbf{PK}, (m, f), \sigma)$. 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 $f(S) = 1$, $\Pr[(\mathrm{PK}, \mathrm{MSK}) \leftarrow \mathbf{ABS}.\mathbf{Setup}(1^\lambda, \mathcal{U}), \mathrm{SK}_S \leftarrow \mathbf{ABS}.\mathbf{KG}(\mathrm{PK}, \mathrm{MSK}, S), \sigma \leftarrow \mathbf{ABS}.\mathbf{Sign}(\mathrm{PK}, \mathrm{SK}_S, (m, f)), b \leftarrow \mathbf{ABS}.\mathbf{Vrfy}(\mathrm{PK}, (m, f), \sigma) : b = 1] = 1$.

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 $\mathbf{Exprmt}^{\text{euf-cma}}_{\mathtt{ABS},\mathcal{F}}(\lambda, \mathcal{U})$ of a forger $\mathcal{F}$ defines the *chosen-message attack on ABS to make an existential forgery*.

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

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, f) of a message and an access formula, $\mathcal{F}$ queries $\mathcal{SIGN}(\mathrm{PK}, \mathrm{SK}_\cdot, (\cdot, \cdot))$ for a signature σ that satisfies $\mathbf{ABS}.\mathbf{Vrfy}(\mathrm{PK}, (m, f), \sigma) = 1$ when $f(S_j) = 1$.

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^*. In key-extraction queries, S_i that satisfies $f^*(S_i) = 1$ was never queried. In signing queries, (m^*, f^*) was never queried. The number of key-extraction queries and the number of signing queries are at most q_{k} and q_{s} in total, respectively, which are bounded by a polynomial in λ.

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

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

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.

Attribute Privacy of ABS Roughly speaking, ABS is called to have attribute privacy if any unconditional cheating verifier 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 1 (Attribute Privacy (Perfect Privacy [37, 39])) *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* $f(S_1) = 1$ *and* $f(S_2) = 1$ *or* $f(S_1) \neq 1$ *and* $f(S_2) \neq 1$*, two distributions* $\mathbf{ABS}.\mathbf{Sign}(PK, SK_{S_1}, (m, f))$ *and*
$\mathbf{ABS}.\mathbf{Sign}(PK, SK_{S_2}, (m, f))$ *are identical.*

3 Our Construction of Boolean Proof

In this section, we first construct a proof system $\mathbf{\Sigma}_f$ from a given Σ-protocol $\mathbf{\Sigma}$ and a boolean predicate f. Then we prove that our $\mathbf{\Sigma}_f$ is a Σ-protocol on the relation R_f. That is, we prove that our $\mathbf{\Sigma}_f$ is a Σ-protocol that is an boolean proof for the language L_f. In Appendix B, we apply the Fiat-Shamir transform $\mathrm{FS}(\cdot)$ to our Σ-protocol $\mathbf{\Sigma}_f$ to obtain a non-interactive witness-indistinguishable proof of knowledge (NIWIPoK) system for the language L_f.

3.1 The Boolean Proof [13, 3]

We revisit the notion of a public coin interactive proof of knowledge system for the language L_f introduced by Cramer, Damgård and Schoenmakers [13], which we call a *boolean proof system*. Then we restate the definitions for the sake of clarity.

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 2 (Cramer, Damgård and Schoenmakers [13], Our Rewritten Form) *A relation R_f is defined by:*

$$R_f \overset{def}{=} \{(x = (x_{i_1}, \ldots, x_{i_a}), w = (w_{i_1}, \ldots, w_{i_a})) \in \{1,0\}^* \times \{1,0\}^*;\\ f(R(x_{i_1}, w_{i_1}), \ldots, R(x_{i_a}, w_{i_a})) = 1\}.$$

R_f is a generalization of the relation R_{OR} for the OR-proof [13, 14], where f is a boolean predicate with the single boolean connective: $X_1 \vee X_2$. 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 for the relation R_f is given as follows.

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

Finally, we achieve the following definition.

Definition 3 *A boolean proof system is an interactive proof system for the language L_f.*

We will provide a concrete procedure $\mathbf{\Sigma}_f$ of a Σ-protocol of a boolean proof system.

3.2 Our Procedure of Boolean Proof for the Language L_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 $\mathbf{\Sigma}_f^{\mathrm{eval}}$, $\mathbf{\Sigma}_f^1$ and $\mathbf{\Sigma}_f^3$. On the other hand, in our verifier-algorithm $\mathcal{V}$, there are two PPT subroutines $\mathbf{\Sigma}_f^2$ and $\mathbf{\Sigma}_f^{\mathrm{vrfy}}$. Moreover, $\mathbf{\Sigma}_f^{\mathrm{vrfy}}$ has two subroutines **VrfyCha** and **VrfyRes**. Fig. 1 shows our construction of boolean proof: $\mathbf{\Sigma}_f$.

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 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,

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

Fig. 1. Our Boolean Proof System $\boldsymbol{\Sigma}_f$ for the language L_f.

in the following way.

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

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

$$
\begin{array}{l}
\boldsymbol{\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}})}, \boldsymbol{\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}})}, \boldsymbol{\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 \boldsymbol{\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 \boldsymbol{\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 \boldsymbol{\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}})}, \boldsymbol{\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}})}, \boldsymbol{\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 \boldsymbol{\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 \boldsymbol{\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 $\boldsymbol{\Sigma}_f^1$ of our $\boldsymbol{\Sigma}_f$.

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

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

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

$\mathbf{\Sigma}_f^3(x, w, \mathcal{T}, (v_n)_n, (\text{CMT}_l)_l, (\text{CHA}_n)_n, (\text{RES}_l)_l)$:
$\mathcal{T}_{\text{L}} := \text{Lsub}(\mathcal{T}), \mathcal{T}_{\text{R}} := \text{Rsub}(\mathcal{T})$
If $r(\mathcal{T})$ is $\wedge$-node, then $\text{CHA}_{r(\mathcal{T}_{\text{L}})} := \text{CHA}_{r(\mathcal{T})}, \text{CHA}_{r(\mathcal{T}_{\text{R}})} := \text{CHA}_{r(\mathcal{T})}$
Return$(\text{CHA}_{r(\mathcal{T}_{\text{L}})}, \mathbf{\Sigma}_f^3(x, w, \mathcal{T}_{\text{L}}, (v_n)_n, (\text{CMT}_l)_l, (\text{CHA}_n)_n, (\text{RES}_l)_l),$
$\text{CHA}_{r(\mathcal{T}_{\text{R}})}, \mathbf{\Sigma}_f^3(x, w, \mathcal{T}_{\text{R}}, (v_n)_n, (\text{CMT}_l)_l, (\text{CHA}_n)_n, (\text{RES}_l)_l))$
else if $r(\mathcal{T})$ is $\vee$-node, then
If $v_{r(\mathcal{T}_{\text{L}})} = 1 \wedge v_{r(\mathcal{T}_{\text{R}})} = 1$, then $\text{CHA}_{r(\mathcal{T}_{\text{L}})} \leftarrow \mathbf{\Sigma}^2(1^\lambda)$, $\text{CHA}_{r(\mathcal{T}_{\text{R}})} := \text{CHA}_{r(\mathcal{T})} \oplus \text{CHA}_{r(\mathcal{T}_{\text{L}})}$
else if $v_{r(\mathcal{T}_{\text{L}})} = 1 \wedge v_{r(\mathcal{T}_{\text{R}})} = 0$, then $\text{CHA}_{r(\mathcal{T}_{\text{L}})} := \text{CHA} \oplus \text{CHA}_{r(\mathcal{T}_{\text{R}})}$, $\text{CHA}_{r(\mathcal{T}_{\text{R}})} := \text{CHA}_{r(\mathcal{T}_{\text{R}})}$
else if $v_{r(\mathcal{T}_{\text{L}})} = 0 \wedge v_{r(\mathcal{T}_{\text{R}})} = 1$, then $\text{CHA}_{r(\mathcal{T}_{\text{L}})} := \text{CHA}_{r(\mathcal{T}_{\text{L}})}$, $\text{CHA}_{r(\mathcal{T}_{\text{R}})} := \text{CHA}_{r(\mathcal{T})} \oplus \text{CHA}_{r(\mathcal{T}_{\text{L}})}$
else if $v_{r(\mathcal{T}_{\text{L}})} = 0 \wedge v_{r(\mathcal{T}_{\text{R}})} = 0$, then $\text{CHA}_{r(\mathcal{T}_{\text{L}})} := \text{CHA}_{r(\mathcal{T}_{\text{L}})}$, $\text{CHA}_{r(\mathcal{T}_{\text{R}})} := \text{CHA}_{r(\mathcal{T}_{\text{R}})}$
Return$(\text{CHA}_{r(\mathcal{T}_{\text{L}})}, \mathbf{\Sigma}_f^3(x, w, \mathcal{T}_{\text{L}}, (v_n)_n, (\text{CMT}_l)_l, (\text{CHA}_n)_n, (\text{RES}_l)_l),$
$\text{CHA}_{r(\mathcal{T}_{\text{R}})}, \mathbf{\Sigma}_f^3(x, w, \mathcal{T}_{\text{R}}, (v_n)_n, (\text{CMT}_l)_l, (\text{CHA}_n)_n, (\text{RES}_l)_l))$
else if $r(\mathcal{T})$ is a leaf, then
If $v_{r(\mathcal{T})} = 1$, then $\text{RES}_{r(\mathcal{T})} \leftarrow \mathbf{\Sigma}^3(x_{\rho(r(\mathcal{T}))}, w_{\rho(r(\mathcal{T}))}, \text{CMT}_{r(\mathcal{T})}, \text{CHA}_{r(\mathcal{T})})$
else if $v_{r(\mathcal{T})} = 0$, then $\text{RES}_{r(\mathcal{T})} \leftarrow \text{RES}_{r(\mathcal{T})}$
Return$(\text{RES}_{r(\mathcal{T})})$

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

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

$\mathbf{\Sigma}_f^{\text{vrfy}}(x, \mathcal{T}, \text{CHA}, (\text{CMT}_l)_l, (\text{CHA}_n)_n, (\text{RES}_l)_l)$:
Return$(\mathbf{VrfyCha}(\mathcal{T}, \text{CHA}, (\text{CHA}_n)_n) \wedge \mathbf{VrfyRes}(x, \mathcal{T}, (\text{CMT}_l, \text{CHA}_l, \text{RES}_l)_l))$

$\mathbf{VrfyCha}(\mathcal{T}, \text{CHA}, (\text{CHA}_n)_n)$:
$\mathcal{T}_{\text{L}} := \text{Lsub}(\mathcal{T}), \mathcal{T}_{\text{R}} := \text{Rsub}(\mathcal{T})$
If $r(\mathcal{T})$ is an $\wedge$-node
then Return $((\text{CHA} \stackrel{?}{=} \text{CHA}_{r(\mathcal{T}_{\text{L}})}) \wedge (\text{CHA} \stackrel{?}{=} \text{CHA}_{r(\mathcal{T}_{\text{R}})})$
$\wedge\, \mathbf{VrfyCha}(\mathcal{T}_{\text{L}}, \text{CHA}_{r(\mathcal{T}_{\text{L}})}, (\text{CHA}_n)_n) \wedge \mathbf{VrfyCha}(\mathcal{T}_{\text{R}}, \text{CHA}_{r(\mathcal{T}_{\text{R}})}, (\text{CHA}_n)_n))$
else if $r(\mathcal{T})$ is an $\vee$-node,
then Return $((\text{CHA} \stackrel{?}{=} \text{CHA}_{r(\mathcal{T}_{\text{L}})} \oplus \text{CHA}_{r(\mathcal{T}_{\text{R}})})$
$\wedge\, \mathbf{VrfyCha}(\mathcal{T}_{\text{L}}, \text{CHA}_{r(\mathcal{T}_{\text{L}})}, (\text{CHA}_n)_n) \wedge \mathbf{VrfyCha}(\mathcal{T}_{\text{R}}, \text{CHA}_{r(\mathcal{T}_{\text{R}})}, (\text{CHA}_n)_n))$
else if $r(\mathcal{T})$ is a leaf,
then Return $(\text{CHA} \stackrel{?}{\in} \text{CHASP}(1^\lambda))$

$\mathbf{VrfyRes}(x, \mathcal{T}, (\text{CMT}_l, \text{CHA}_l, \text{RES}_l)_l)$:
For $l \in \text{Leaf}(\mathcal{T})$: If $\mathbf{\Sigma}^{\text{vrfy}}(x_{\rho(l)}, \text{CMT}_l, \text{CHA}_l, \text{RES}_l) = 0$, then Return (0)
Return (1)

Now we have to check that $\mathbf{\Sigma}_f$ is certainly a Σ-protocol for the language L_f.

Proposition 1 (Completeness) *Completeness holds for our* $\mathbf{\Sigma}_f$. *More precisely, Suppose that* $v_{r(\mathcal{T}_f)} = 1$. *Then, for every node in* $Node(\mathcal{T}_f)$, *either* $v_n = 1$ *or* $\textsc{Cha}_n \neq *$ *holds after executing* $\mathbf{\Sigma}_f^1$.

Proof. 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.

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

Then Lemma 1 assures the proposition.

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

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

Suppose that all nodes in $\text{iNode}(T_f)$ 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.

Corollary 1 *Our procedure* $\mathbf{\Sigma}_f$ *is a boolean proof system for the language* L_f.

It is notable that our Σ-protocol of boolean proof, $\mathbf{\Sigma}_f$, 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 [13]) *without collusion resistance on secret keys.*

4 Our Attribute-Based Identification Scheme

In this section, we provide an attribute-based identification scheme (ABID) by combining our Σ-protocol $\mathbf{\Sigma}_f$ of a boolean proof system in Section 3 with a signature bundle of the Fiat-Shamir signatures. Our ABID is verifier-policy scheme. Our ABID has a feature that it can be constructed without pairings when it is instantiated in, for example, the RSA setting or the discrete-logarithm setting (see Section 8 and Appendix C).

4.1 Our ABID

By combining our $\mathbf{\Sigma}_f$ in Section 3 with a signature-bundle scheme FS($\mathbf{\Sigma}$), we obtain a scheme of VP-ABID, `ABID`. Our `ABID` is collusion resistant against collecting private secret keys. Our `ABID` has a feature that it can be constructed without pairings. Fig. 4 shows our construction: `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 $(x_{\rm mst}, w_{\rm mst})$ 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 a hash-key space $\mathit{Hashkeysp}(\lambda)$. It returns a public key $\mathrm{PK} = (x_{\rm mst}, \mathcal{U}, \mu)$ and a master secret key $\mathrm{MSK} = (w_{\rm mst})$.

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

ABID.**KG** takes as input PK, MSK, S. It chooses a PRF key k from $\mathit{PRFkeysp}(\lambda)$ at random and a random string τ from $\{1, 0\}^\lambda$ at random. Then it applies the signature-bundle technique [37] for each message $m_i := (\tau \parallel i), i \in S$. Here we employ the Fiat-Shamir signing algorithm $\mathrm{FS}(\mathbf{\Sigma})^{\rm sign}$ (see 2.1). It returns SK_S.

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

$\mathcal{P}$ 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 $i_j \notin S$.

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

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

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

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, $(x = (x_{i_j})_{1 \leq j \leq \text{arity}(f)}, w = (w_{i_j})_{1 \leq j \leq \text{arity}(f)})$, to $\boldsymbol{\Sigma}_f^1$, into the prover of our boolean proof system $\boldsymbol{\Sigma}_f$. 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 $(a_{i_j})_{1 \leq j \leq \text{arity}(f)}$ to the verifier $\mathcal{V}$.

Second, $\mathcal{V}$ runs **StmtGen** on input PK, τ and $(a_{i_j})_{1 \leq j \leq \text{arity}(f)}$ to generate the statement x. Note that τ and $(a_{i_j})_{1 \leq j \leq \text{arity}(f)}$ 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 boolean proof system $\boldsymbol{\Sigma}_f$, respectively. $\mathcal{V}$ returns 1 or 0 according to the return of the verifier of $\boldsymbol{\Sigma}_f$.

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

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

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

4.2 Security of Our ABID

Theorem 2 (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).*

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

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 $\mathbf{\Sigma}^{\mathrm{KE}}_f$, $\mathcal{F}$ 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 [6], the advantage $\mathbf{Adv}^{\mathrm{ca}}_{\mathtt{ABID},\mathcal{A}}(\lambda,\mathcal{U})$ is reduced to $\mathbf{Adv}^{\mathrm{euf\text{-}cma}}_{\mathrm{FS}(\mathbf{\Sigma}),\mathcal{F}}(\lambda,\mathcal{U})$ with a loss of exponent by 1/2. □

Corollary 2 (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).*

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

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 2.5.

More on Reduction of Concurrent Security We mean by "a number theoretic problem" the discrete-logarithm problem or the RSA-inverse problem ([6]). There exists the following security reduction to a number theoretic problem.

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

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.

Proof. As is discussed in Section 2.1, we can reduce the advantage $\mathbf{Adv}^{\mathrm{euf\text{-}cma}}_{\mathrm{FS}(\mathbf{\Sigma}),\mathcal{F}}(\lambda,\mathcal{U})$ to the advantage $\mathbf{Adv}^{\mathrm{pa}}_{\mathbf{\Sigma},\mathcal{B}}(\lambda,\mathcal{U})$ of passive security of the underlying Σ-protocol, in the random oracle model, with a loss factor q_H. Applying the Reset Lemma [6], we can reduce $\mathbf{Adv}^{\mathrm{pa}}_{\mathbf{\Sigma},\mathcal{B}}(\lambda,\mathcal{U})$ to the advantage $\mathbf{Adv}^{\mathrm{num.prob.}}_{\mathbf{Grp},\mathcal{S}}(\lambda,\mathcal{U})$ 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 4.1. Our ABS is collusion resistant against collecting private secret keys, and EUF-CMA secure in the random oracle model. We note that our ABS has attribute privacy only as one-time signature because of its linkability.

5.1 Our ABS

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

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 $\mathbf{Supp}(\mathrm{PK}, \mathrm{SK}_S, f)$, **StmtGen** and the prover of our boolean proof system $\mathbf{\Sigma}_f$ 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.

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

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

$\mathbf{ABS.Sign}(\mathrm{PK}, \mathrm{SK}_S, (m, f))$:
- $\mathbf{Supp}(\mathrm{PK}, \mathrm{SK}_S, f) \rightarrow (a_{i_j}, w_{i_j})_j$
- $w := (w_{i_j})_j$
- $\mathbf{StmtGen}(\mathrm{PK}, \tau, (a_{i_j})_j) \rightarrow (x_{i_j})_j =: x$
- $\mathbf{\Sigma}_f^{\mathrm{eval}}(\mathcal{T}_f, S) \rightarrow (v_n)_n$
- If $v_{r(\mathcal{T}_f)} \neq 1$, then abort else $\textsc{Cha}_{r(\mathcal{T}_f)} := *$
- $\mathbf{\Sigma}_f^1(x, w, \mathcal{T}_f, (v_n)_n, \textsc{Cha}_{r(\mathcal{T}_f)}) \rightarrow ((\textsc{Cmt}_l)_l, (\textsc{Cha}_n)_n, (\textsc{Res}_l)_l)$
- $\textsc{Cha} \leftarrow \mathit{Hash}_\mu(x \parallel (\textsc{Cmt}_l)_l \parallel m)$
- $\textsc{Cha}_{r(\mathcal{T}_f)} := \textsc{Cha}$
- $\mathbf{\Sigma}_f^3(x, w, \mathcal{T}_f, (v_n)_n, (\textsc{Cmt}_l)_l, (\textsc{Cha}_n)_n, (\textsc{Res}_l)_l) \rightarrow ((\textsc{Cha}_n)_n, (\textsc{Res}_l)_l)$
- Return $\sigma := (\tau, (a_{i_j})_j, (\textsc{Cmt}_l)_l, (\textsc{Cha}_n)_n, (\textsc{Res}_l)_l)$

$\mathbf{ABS.Vrfy}(\mathrm{PK}, (m, f), \sigma := (\tau, (a_{i_j})_j, (\textsc{Cmt}_l)_l, (\textsc{Cha}_n)_n, (\textsc{Res}_l)_l))$:
- $\mathbf{StmtGen}(\mathrm{PK}, \tau, (a_{i_j})_j) \rightarrow (x_{i_j})_j =: x$
- $\textsc{Cha} \leftarrow \mathit{Hash}_\mu(x \parallel (\textsc{Cmt}_l)_l \parallel m)$
- $\mathbf{\Sigma}_f^{\mathrm{vrfy}}(x, \mathcal{T}_f, \textsc{Cha}, (\textsc{Cmt}_l)_l, (\textsc{Cha}_n)_n, (\textsc{Res}_l)_l) \rightarrow b$, Return b

Fig. 5. The scheme of our $\mathtt{ABS}$.

5.2 Security of Our ABS

Applying the standard technique in the work of Abdalla et al. [1] shows that the security of our $\mathtt{ABS}$ is equivalent to the security of an attribute-based identification scheme, $\mathtt{ABID}$, against passive attacks, where our $\mathtt{ABID}$ is obtained by combining our Σ-protocol $\mathbf{\Sigma}_f$ with the signature-bundle scheme of the Fiat-Shamir signature FS($\mathbf{\Sigma}$) in the same way as $\mathtt{ABS}$ (See Appendix 4 for our $\mathtt{ABID}$).

Theorem 3 (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).*

$$\mathbf{Adv}^{\textit{euf-cma}}_{\texttt{ABS},\mathcal{F}}(\lambda,\mathcal{U}) \leq q_H \mathbf{Adv}^{\textit{pa}}_{\texttt{ABID},\mathcal{B}}(\lambda,\mathcal{U}) + neg(\lambda). \tag{3}$$

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 $(a_{i_j})_{1\leq j\leq \mathrm{arity}(f)}$ 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. So $\texttt{ABS} = \mathrm{FS}(\texttt{ABID})$.

As is discussed in Section 2.1, we can reduce the advantage $\mathbf{Adv}^{\text{euf-cma}}_{\texttt{ABS},\mathcal{F}}(\lambda,\mathcal{U})$ to the advantage $\mathbf{Adv}^{\text{pa}}_{\texttt{ABID},\mathcal{B}}(\lambda,\mathcal{U})$ 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 (3) with the inequalities (1) and (2) in Section 2.5 and Section 4, we obtain the following security reduction of advantages.

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

Attribute Privacy Our ABS does not have attribute privacy defined in Section 2.6 because of its linkability; that is, the constant components $\tau, (a_{i_j})_j$ make two signatures linkable (especially, τ is a component of a private secret key SK_S). Hence, our ABS merely has attribute privacy *as a one-time signature.*

6 Attribute-Based Two-Tier Signature: Syntax

In this section, we define a syntax of attribute-based two-tier signature scheme (ABTTS). Then, we define a chosen-message attack (CMA) on ABTTS by which an adversary makes an existential forgery, and define the existential unforgeability (EUF) security against 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 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 $f(S) = 1$, $\Pr[(\mathrm{MSK}, \mathrm{PK}) \leftarrow \mathbf{ABTTS}.\mathbf{Setup}(1^\lambda, \mathcal{U}),\ \mathrm{SK}_S \leftarrow \mathbf{ABTTS}.\mathbf{PKG}(\mathrm{MSK}, \mathrm{PK}, S),\ (\mathrm{SSK}_{S,f}, \mathrm{SPK}_f) \leftarrow \mathbf{ABTTS}.\mathbf{SKG}(\mathrm{MSK}, \mathrm{PK}, \mathrm{SK}_S, f),\ \sigma \leftarrow \mathbf{ABTTS}.\mathbf{Sign}(\mathrm{SK}_S, \mathrm{PK}, \mathrm{SSK}_{S,f}, \mathrm{SPK}_f, (m, f)),$
$b \leftarrow \mathbf{ABS}.\mathbf{Vrfy}(\mathrm{PK}, \mathrm{SPK}_f, (m, f), \sigma) : b = 1] = 1$.

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 $\mathbf{Expr}^{\text{euf-cma}}_{\mathtt{ABTTS},\mathcal{F}}(1^\lambda, \mathcal{U})$ of a forger $\mathcal{F}$ defines the *chosen-message attack on ABTTS making an existential forgery*.

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

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 when $f(S_j) = 1$. 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 $\mathcal{SIGN}$ [7].

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 $\mathcal{F}$: 1) $f^*(S_i) = 0$ for all S_i in key-extraction queries; 2) (m^*, f^*) was never queried in signing queries. 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 $\mathbf{Adv}^{\text{euf-cma}}_{\mathtt{ABTTS},\mathcal{F}}(\lambda, \mathcal{U}) \stackrel{\text{def}}{=} \Pr[\mathbf{Expr}^{\text{euf-cma}}_{\mathtt{ABTTS},\mathcal{F}}(1^\lambda, \mathcal{U}) \text{ returns } \textsc{Win}]$.

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

Then we define *attribute privacy* of ABTTS.

Definition 5 (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 and S_2,
for all primary secret keys $SK_{S_1} \leftarrow \mathbf{ABTTS}.\mathbf{PKG}(PK, MSK, S_1)$ and
$SK_{S_2} \leftarrow \mathbf{ABTTS}.\mathbf{PKG}(PK, MSK, S_2)$,
for all secondary secret keys $(SSK_{S_1,f}, SPK_f) \leftarrow \mathbf{ABTTS}.\mathbf{SKG}(MSK, PK, SK_S, f)$ and
$(SSK_{S_2,f}, SPK_f) \leftarrow \mathbf{ABTTS}.\mathbf{SKG}(MSK, PK, SK_S, f)$
and for all access formula f such that $[f(S_1) = 1 \wedge f(S_2) = 1] \vee [f(S_1) \neq 1 \wedge f(S_2) \neq 1]$, 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) by applying the two-tier framework in Section 6 to our `ABID` in Section 4.1. Our ABTTS is collusion resistant against collecting private secret keys, and EUF-CMA secure in the standard model. We note that our ABTTS has attribute privacy.

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 EUF-CMA, collusion resistance and attribute privacy, in the standard model. The critical point is that the secondary key generator **ABTTS.SKG** can issue a legitimate statement x for the boolean proof system $\boldsymbol{\Sigma}_f$. Hence our ABTTS can avoid collusion attacks on secret keys.

Fig. 6 shows our construction: $\mathtt{ABTTS} = (\mathbf{ABTTS.Setup}, \mathbf{ABTTS.PKG}, \mathbf{ABTTS.SKG}, \mathbf{ABTTS.Sign}, \mathbf{ABTTS.Vrfy})$.

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 $\boldsymbol{\Sigma}_f$ to generate the first message as

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

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 [7].

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 $\textsc{Cha}$ by hashing the string $(\textsc{Cmt}_l)_l \parallel m$. Then, it runs the prover $\mathcal{P}$ according to $\boldsymbol{\Sigma}_f$ as

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

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 $\textsc{Cha}$ by hashing the string $(\textsc{Cmt}_l)_l \parallel m$. Then, it runs the verifier $\mathcal{V}$ according to $\boldsymbol{\Sigma}_f$ as

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

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 [7].

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

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

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

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

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

$\mathrm{SSK}_{S,f} := (w, (\textsc{Cmt}_l)_l \parallel st)$
$\mathrm{SPK}_f := (x, (\textsc{Cmt}_l)_l)$
Return$(\mathrm{SSK}_{S,f}, \mathrm{SPK}_f)$

ABTTS.**Sign**$(\mathrm{PK}, \mathrm{SK}_S, \mathrm{SSK}_{S,f}, \mathrm{SPK}_f, (m, f))$:
$\textsc{Cha} \leftarrow \mathit{Hash}_\mu((\textsc{Cmt}_l)_l \parallel m)$
$\textsc{Cha}_{r(\mathcal{T}_f)} := \textsc{Cha}$

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

ABTTS.**Vrfy**$(\mathrm{PK}, \mathrm{SPK}_f, (m, f),$
$\sigma := ((\textsc{Cha}_n)_n, (\textsc{Res}_l)_l))$:

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

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

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

Theorem 4 (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).*

$$\mathbf{Adv}^{euf\text{-}cma}_{ABTTS,\mathcal{F}}(\lambda,\mathcal{U}) \leq q_H \mathbf{Adv}^{ca}_{ABID,\mathcal{B}}(\lambda,\mathcal{U}) + neg(\lambda). \tag{4}$$

Proof. We just note that the same argument in [7] is applied to our ABTTS. □

Theorem 5 (Attribute Privacy) *Our attribute-based two-tier signature scheme* ABTTS *has attribute privacy. More precisely,*

Proof. A valid signature of ABTTS, $\sigma := ((\text{CHA}_n)_n, (\text{RES}_l)_l)$, is a valid boolean proof of $\mathbf{\Sigma}_f$. Therefore, $(\text{CMT}_l, \text{CHA}_l, \text{RES}_l)$ is a valid transcript of $\mathbf{\Sigma}$ for all leaves $l \in \text{Leaf}(\mathcal{T}_f)$. Then the definition 5 is satisfied. □

8 Instantiation Using Fiat-Shamir Signature-Bundle as Witness

In this section, we provide instantiations of our boolean proof system $\mathbf{\Sigma}_f$ and ABID using the Fiat-Shamir signatures [19] as a witness. We give two instantiations in the RSA setting and the discrete-logarithm setting.

8.1 Our ABID in RSA Using FS Signature-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 $\text{Instance}_R(1^\lambda)$ chooses an element $(\beta, \alpha) \in R_\lambda$ at random. **ABID.Setup** returns a public key and a master secret key: $\text{PK} = ((N, e, \beta), \mathcal{U}, \mu), \text{MSK} = \alpha$.
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 $k \leftarrow Hash_\mu(\alpha \parallel \tau)$, put $m_i = \tau \parallel i$, and $r_i \in \mathbb{Z}_N$ is chosen at random according to a random tape: $PRF_k(m_i)$, and c_i is obtained by $c_i \leftarrow Hash_\mu(a_i \parallel m_i)$. $\mathbf{\Sigma}^{\text{stmtgen}}(\beta, a_i, c_i)$ 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.

$$\text{CMT}_l = {r_l}^e, \text{RES}_l = r_l(w_{\rho(l)})^{\text{CHA}_l},$$
$$\text{Verification Equation} : {\text{RES}_l}^e \stackrel{?}{=} \text{CMT}_l\ (x_{\rho(l)})^{\text{CHA}_l}.$$

8.2 Our ABID in Discrete Log Using FS Signature-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 $k \leftarrow Hash_\mu(\alpha \parallel \tau)$, put $m_i = \tau \parallel i$, and $r_i \in \mathbb{Z}_p$ is chosen at random according to a random tape: $PRF_k(m_i)$, and c_i is obtained by $c_i \leftarrow Hash_\mu(a_i \parallel m_i)$. $\mathbf{\Sigma}^{\text{stmtgen}}(\beta, a_i, c_i)$ 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.

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

9 Conclusions

We provided a concrete procedure $\mathbf{\Sigma}_f$ of a Σ-protocol of the high-level construction of the boolean proof system [13]. 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 [13]) without collusion resistance on 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`. Pairing-free instantiations are provided in both the RSA setting and the discrete-logarithm setting by employing the Schnorr identification scheme and the GQ identification scheme [44, 6] as $\mathbf{\Sigma}$, respectively. 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 [7].

The scheme by Herranz [28] is the only ABS scheme with collusion resistance, (computational) attribute privacy and pairing-free property, in the RSA setting. Our procedure $\mathbf{\Sigma}_f$ of the boolean proof [13] for any monotone predicate serves as a building block of (the Σ-protocol of) the ABS scheme [28]. We believe that there is still an open problem to construct a pairing-free efficient ABS scheme in the discrete-logarithm setting.

Acknowledgements We appreciate sincere comments from Javier Herranz in Universitat Politècnica de Catalunya, Barcelona, Spain via e-mail communication [30] on the topic in this paper. We would like to thank to Keita Emura in National Institute of Information and Communications Technology (NICT), Tokyo, Japan and Takahiro Matsuda in National Institute of Advanced Industrial Science and Technology (AIST), Tokyo, Japan for their sincere comments and encouragements on the construction of attribute-based signature schemes. We would like to thank to Shingo Hasegawa in Tohoku University, Sendai, Japan and Masayuki Fukumitsu in Hokkaido Information University, Sapporo, Japan for their sincere comments on the construction of the Σ-protocol on monotone predicates.

References

1. M. Abdalla, J. H. An, M. Bellare, and C. Namprempre. From identification to signatures via the fiat-shamir transform: Minimizing assumptions for security and forward-security. In *EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 418–433. Springer, 2002.
2. H. Anada, S. Arita, S. Handa, and Y. Iwabuchi. Attribute-based identification: Definitions and efficient constructions. In *ACISP 2013*, volume 7959 of *LNCS*, pages 168–186. Springer, 2013.
3. H. Anada, S. Arita, and K. Sakurai. Attribute-based signatures without pairings via the fiat-shamir paradigm. In *ASIAPKC2014*, volume 2 of *ACM-ASIAPKC*, pages 49–58. ACM, 2014.
4. M. Bellare and G. 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.
5. M. Bellare and O. Goldreich. On defining proofs of knowledge. In *CRYPTO '92*, volume 740 of *LNCS*, pages 390–420. Springer, 1992.
6. M. Bellare and A. Palacio. Gq and schnorr identification schemes: Proofs of security against impersonation under active and concurrent attacks. In *CRYPTO 2002*, volume 2442 of *LNCS*, pages 162–177. Springer, 2002.
7. M. Bellare and S. 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.
8. D. Boneh and X. Boyen. Efficient selective-id secure identity-based encryption without random oracles. In *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 223–238. Springer, 2004.

9. D. Boneh and X. 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.
10. D. Boneh, X. Boyen, and H. 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.
11. J. Camenisch and A. 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.
12. R. Cramer. *Modular Designs of Secure, yet Practical Cyptographic Protocols.* PhD thesis, University of Amsterdam, Amsterdam, the Netherlands, 1996.
13. R. Cramer, I. Damgård, and B. Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *CRYPTO '94*, pages 174–187. Springer-Verlag, 1994.
14. I. Damgård. On σ-protocols. In Course Notes, http://www.cs.au.dk/ ivan/Sigma.pdf, 2011.
15. A. El Kaafarani, L. Chen, E. Ghadafi, and J. 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.
16. A. El Kaafarani, E. Ghadafi, and D. 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.
17. A. Escala, J. Herranz, and P. 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.
18. U. Feige and A. Shamir. Witness indistinguishable and witness hiding protocols. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing*, STOC '90, pages 416–426, New York, NY, USA, 1990. ACM.
19. A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *CRYPTO '86*, volume 263 of *LNCS*, pages 186–194. Springer, 1987.
20. J. Furukawa and H. 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.
21. M. Gagné, S. Narayan, and R. 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.
22. E. 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.
23. V. Goyal, O. Pandey, A. Sahai, and B. Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *ACM-CCS '06*, volume 263, pages 89–98. ACM, 2006.
24. R. Granger, T. Kleinjung, and J. 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.
25. J. Groth and A. 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.
26. L. C. Guillou and J. 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.
27. S. Guo and Y. Zeng. Attribute-based signature scheme. In *ISA '08*, pages 509–511. IEEE, 2008.
28. J. Herranz. Attribute-based signatures from rsa. *Theoretical Computer Science*, 527:73–82, 2014.
29. J. Herranz. Attribute-based versions of schnorr and elgamal. *Appl. Algebra Eng. Commun. Comput.*, 27(1):17–57, 2016.
30. J. Herranz. Private communication via e-mail, dept. matemàtica aplicada iv, universitat politècnica de catalunya, July 2014, Sept 2015 and May 2016.
31. J. Herranz, F. Laguillaumie, B. Libert, and C. 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.
32. J. Herranz, F. Laguillaumie, and C. 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.
33. J. 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.

34. J. Katz and Y. Lindell. *Introduction to Modern Cryptography.* Chapman and Hall/CRC, 2008.
35. S. Kumar, S. Agrawal, S. Balaraman, and C. P. 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.
36. J. Li, M. H. Au, W. Susilo, D. Xie, and R. K. Attribute-based signature and its applications. In *ASIA-CCS '10*, volume 5, pages 60–69. ACM, 2010.
37. H. K. Maji, M. Prabhakaran, and M. Rosulek. Attribute-based signatures. In *CT-RSA 2011*, volume 6558 of *LNCS*, pages 376–392. Springer, 2011. Full version available at IACR Cryptology ePrint Archive, 2010/595, http://eprint.iacr.org/.
38. T. 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.
39. T. Okamoto and K. 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.
40. T. Okamoto and K. 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.
41. D. Pointcheval and J. 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.
42. A. Sahai and B. Waters. Fuzzy identity based encryption. Cryptology ePrint Archive, Report 2004/086, 2004. http://eprint.iacr.org/.
43. Y. Sakai, N. Attrapadung, and G. 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.
44. C. P. Schnorr. Efficient identification and signatures for smart cards. In *CRYPTO '89*, volume 435 of *LNCS*, pages 239–252. Springer, 1990.
45. S. F. Shahandashti and R. 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.
46. I. Teranishi and J. Furukawa. Anonymous credential with attributes certification after registration. *IEICE Transactions*, 95-A(1):125–137, 2012.
47. M. Yasuda, T. Shimoyama, J. Kogure, and T. 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 Non-interactive Witness-Indistinguishable Proof of Knowledge [25]

In this appendix, we summarize the notion of *non-interactive witness-indistinguishable proof of knowledge system* (NIWIPoK, for short).

A NIWIPoK $\Pi = (K, \mathcal{P}, \mathcal{V})$ for a language L_R is a protocol, where a PPT algorithm K, on input 1^λ, outputs crs called a *common reference string*; a PPT algorithm $\mathcal{P}$, on input $(x, w) \in R$ and crs, outputs π called a *proof*; and a PPT algorithm V, on input (x, π) and crs, outputs 1 (accept) or 0 (reject).

Π must possess the following three properties.

Completeness. For any statement $x \in L_R$ and for any witness w such that $(x, w) \in R$, $\mathcal{P}$ with the witness w can make $\mathcal{V}$ accept on the statement x with probability 1:

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

Knowledge Soundness. There are an algorithm $\mathcal{KE}$ called a *knowledge extractor*, a function $\kappa : \{1,0\}^* \to [1, 0]$ called a *knowledge error function* and a constant $c > 0$ that satisfy the following:
If there exists a PPT algorithm $\mathcal{A}$ that satisfies $p(x) := \Pr[\mathrm{crs} \leftarrow K(1^\lambda), \pi \leftarrow \mathcal{A}(\mathrm{crs}) : \mathcal{V}(x, \pi) = 1] > \kappa(x)$, then $\mathcal{KE}(x)$, employing $\mathcal{A}(x)$ as a subroutine that allows to be rewinded, outputs a witness w which satisfies $(x, w) \in R$ within an expected number of steps bounded by: $|x|^c/(p(x) - \kappa(x))$.

Witness-Indistinguishability. There is a polynomial-time algorithm S called a *simulator*, such that for any non-uniform polynomial-time algorithm $\mathcal{A}$ we have

$$\Pr[\mathrm{crs} \leftarrow K(1^\lambda) : \mathcal{A}(\mathrm{crs}) = 1] \approx \Pr[\mathrm{crs} \leftarrow S(1^\lambda) : \mathcal{A}(\mathrm{crs}) = 1]$$
$$\text{(computationally indistinguishable)}$$

and for any unbounded algorithm $\mathcal{A}$, we have

$$\begin{aligned} &\Pr[\mathrm{crs} \leftarrow S(1^\lambda), (x, w_0, w_1) \leftarrow \mathcal{A}(\mathrm{crs}), \pi \leftarrow \mathcal{P}(\mathrm{crs}, x, w_0) : \mathcal{A}(\pi) = 1] \\ =&\Pr[\mathrm{crs} \leftarrow S(1^\lambda), (x, w_0, w_1) \leftarrow \mathcal{A}(\mathrm{crs}), \pi \leftarrow \mathcal{P}(\mathrm{crs}, x, w_1) : \mathcal{A}(\pi) = 1] \end{aligned}$$

where $\big(R(x, w_0) = 1 \wedge R(x, w_1) = 1\big) \vee \big(R(x, w_0) = 0 \wedge R(x, w_1) = 0\big)$ holds.

B Our Non-interactive Witness-Indistinguishable Proof of Knowledge on Monotone Predicates

In this appendix, we provide our NIWIPoK by applying the Fiat-Shamir transform (Section 2.1) to our boolean proof system $\mathbf{\Sigma}_f$ in Section 3.

The Fiat-Shamir transform FS$(\cdot)$ can be applied to any Σ-protocol $\mathbf{\Sigma}$ ([19, 1], see Section 2.1) to obtain a NIZKPoK system.

The generator K of common reference strings is becomes as follows.

$$K(1^\lambda): \ \mu \leftarrow \mathit{Hashkeysp}(\lambda), \mathrm{crs} := \mu, \ \text{Return crs}$$

Hence we obtain the following theorem.

Theorem 6 (FS($\mathbf{\Sigma}_f$) is NIWIPoK) *FS($\mathbf{\Sigma}_f$) is a non-interactive witness-indistinguishable proof of knowledge system for the language L_f. A knowledge extractor is constructed in the random oracle model.*

C Instantiations Using Camenisch-Lysyanskaya Signature-Bundle as Witness

In this section, we provide another type of instantiations of our boolean proof system $\mathbf{\Sigma}_f$, ABID and ABTTS using the Camenisch-Lysyanskaya Signatures as a witness. We give two instantiations in the RSA setting [11] and the discrete-logarithm setting [46, 20, 38].

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

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

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

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 $\eta_{e,l}, \eta_{s,n} \xleftarrow{\$} \bar{\mathbb{Z}}$, and puts $\textsc{Cmt}_l := Z_{\rho(l),1}^{\eta_{e,l}} Z_{\rho(l),2}^{\eta_{s,n}}$. To simulate the honest computation at a leaf l, $\mathcal{P}$ chooses $\eta_{e,l}, \theta_{s,l} \xleftarrow{\$} \bar{\mathbb{Z}}$, and in addition, the divided challenge strings $(\textsc{Cha}_n)_n, \textsc{Cha}_n \in \bar{\mathbb{Z}}$, which are in accordance with the boolean proof system $\mathbf{\Sigma}_f$. 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 $\textsc{Cha} \xleftarrow{\$} \bar{\mathbb{Z}}$, 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 $\theta_{e,l} := \eta_{e,l} + \textsc{Cha}_l e_{\rho(l)}$, $\theta_{s,l} := \eta_{s,l} + \textsc{Cha}_l v_l$. $\mathcal{P}$ 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:

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

According to the division rule of the boolean proof system $\mathbf{\Sigma}_f$, the integrity of $(\textsc{Cha}_l = \textsc{Cha}_l)_l$ can be checked as follows: From the leaves to the root, and at every inner node $n \in \mathrm{iNode}(\mathcal{T}_f)$ as well as its two children chd_1, chd_2;

- If n is an AND node $(\wedge)$, then verify $\textsc{Cha}_{chd_1} \stackrel{?}{=} \textsc{Cha}_{chd_2}$. 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 $\textsc{Cha}_n \stackrel{?}{=} \textsc{Cha}$.
- Repeat until all $n \in \mathrm{iNode}(\mathcal{T}_f)$ 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.

C.2 Our ABID and ABTTS in RSA Using CL Signature-Bundle as Witness

Strong RSA Assumption [11] 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 `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 [11] states that for any PPT $\mathcal{A}$, the following advantage is negligible in λ: $\mathbf{Adv}^{\mathrm{srsa}}_{\mathtt{RSAmod},\mathcal{S}}(\lambda, \mathcal{U}) := \Pr[N \leftarrow \mathtt{RSAmod}(1^\lambda), g \xleftarrow{\$} QR_N, (V,e) \leftarrow \mathcal{A}(N,g) : e > 1 \wedge V^e \equiv g \bmod N]$.

CL Signature-Bundle in RSA

Our signature-bundle scheme $\mathtt{SB} = (\mathbf{SB.KG}, \mathbf{SB.Sign}, \mathbf{SB.Vrfy})$ 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.

$\mathbf{SB.KG}(1^\lambda) \to (\mathrm{PK}, \mathrm{SK})$. 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 $g_{i,0}, g_{i,1}, g_{i,2} \xleftarrow{\$} QR_N$. It puts $\mathrm{PK} := (N, (g_{i,0}, g_{i,1}, g_{i,2})_{i=1}^n)$ and $\mathrm{SK} = p$, and returns $(\mathrm{PK}, \mathrm{SK})$.

$\mathbf{SB.Sign}(\mathrm{PK}, \mathrm{SK}, (m_i)_{i=1}^n) \to (\tau, (\sigma_i)_{i=1}^n)$. 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 $(\tau, (\sigma_i)_{i=1}^n)$.

$\mathbf{SB.Vrfy}(\mathrm{PK}, (m_i)_{i=1}^n, (\tau, (\sigma_i)_{i=1}^n)) \to 1/0$. Given PK, $(m_i)_{i=1}^n$ and a signature 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 7 (Unforgeability of Our SB) *Our signature-bundle scheme* SB *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 [11]. The difference only arises in the case that the simulation of the signature-bundle oracle needs precomputation.

Let $\mathcal{F}$ be a given PPT forger on our SB. 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 signature-bundle queries. Suppose that the signature-bundle oracle $\mathcal{SBSIGN}$ replies the tags (that is, exponents) $e_1, \dots, e_q$ according 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 signature-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 [11].

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 $j^* \in \{1, \dots, q\}$ 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 $\gcd(\bar{\alpha}_i, e_{j^*}) = 1$, and 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 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 [11].

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 of an instance goes in the same way as in [11]. Note that $e^* = e_{j^*}$ holds with at least a non-negligible probability $1/q$. □

Our ABID in RSA Using CL-SB as Witness

ABID.Setup$(1^\lambda, \mathcal{U}) \to$ (MSK, 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 $g_{i,0}, g_{i,1}, g_{i,2} \xleftarrow{\$} QR_N$ and a hash key $\mu \xleftarrow{\$} \mathit{Hashkeysp}(\lambda)$ of a hash function Hash_μ with the value in $\mathbb{Z}_{\phi(N)}$. It puts PK $:= (N, (g_{i,0}, g_{i,1}, g_{i,2})_{i \in \mathcal{U}}, \mu, \mathcal{U})$ and MSK $:= p$. It returns PK and MSK.

ABID.KG(MSK, PK, S) $\to$ 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 $a_i \leftarrow \mathit{Hash}_\mu(i)$, $s_i \xleftarrow{\$} \mathbb{Z}$ of length l_e, $A_i := (g_{i,0} g_{i,1}^{a_i} g_{i,2}^{-s_i})^{\frac{1}{e}}$. It puts 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 $r_i \xleftarrow{\$} \mathbb{Z}$ of length l_e. If $i \in S$ then $s'_i := s_i + er_i, A'_i := A_i g_{i,2}^{-r_i}$. Else $s'_i \xleftarrow{\$} \mathbb{Z}$ of length l_e, $A'_i \xleftarrow{\$} \mathbb{Z}_N^*$. $\mathcal{P}$ 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 $\mathbf{\Sigma}_f$ for the language L_f. In other words, $\mathcal{P}$ and $\mathcal{V}$ execute $\mathbf{ZKPoK}[(e_{\rho(l)}, s'_{\rho(l)})_l, l \in \mathrm{Leaf}(\mathcal{T}_f) : \textit{equations}]$, 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:

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

$\mathcal{V}$ 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 signature-bundle scheme* `SB` *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 in `SB`. As for the extraction of a signature bundle, we can do it under the condition that the same e is answered. □

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-SB as Witness

ABTTS.Setup and **ABTTS.PKG** are the same as **ABID.Setup** and **ABID.KG** in Section C.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 [7]. 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 [7] as in Section 7. The signature is:

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

Security of Our ABTTS in RSA Using CL-SB

Theorem 8 (Unforgeability) *Our attribute-based two-tier signature scheme* $\mathtt{ABTTS'}$ *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. [7] as well as Theorem 7 and Claim 1, we deduce the claim. □

Theorem 9 (Attribute Privacy) *Our attribute-based two-tier signature scheme* $\mathtt{ABTTS'}$ *has attribute privacy.*

Proof. The witness-hiding property assures the attribute privacy. □

C.3 Our ABID and ABTTS in Discrete Log Using CL Signature-Bundle as Witness

Strong Diffie-Hellman Assumption [8] 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 $\mathtt{BlGrp}$. Let q denote a number that is less than a fixed polynomial in λ. The strong Diffie-Hellman assumption [8] states that for any PPT $\mathcal{A}$, the following advantage is negligible in λ: $\mathbf{Adv}^{\mathrm{sdh}}_{\mathtt{BlGrp},\mathcal{S}}(\lambda, \mathcal{U}) := \Pr[\mathrm{params} \leftarrow \mathtt{BlGrp}(1^\lambda), \alpha \xleftarrow{\$} \mathbb{Z}_p,\ (u, e) \leftarrow \mathcal{A}(\mathrm{params}, (g, g^\alpha, g^{\alpha^2}, \ldots, g^{\alpha^q}, h, h^\alpha)) : u^{\alpha+e} = g]$.

CL Signature-Bundle in DL

We propose a signature-bundle scheme in the discrete-logarithm setting by modifying the pairing-based CL signature scheme [46, 20, 38]. Our pairing-based signature-bundle scheme, $\mathtt{SB} = (\mathbf{SB.KG}, \mathbf{SB.Sign}, \mathbf{SB.Vrfy})$, is described as follows.

$\mathbf{SB.KG}(1^\lambda) \to (\mathrm{PK}, \mathrm{SK})$. Given 1^λ as input, it runs a group generator $\mathtt{BlGrp}(1^\lambda)$ to get $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e(\cdot, \cdot))$. For $i = 1$ to n, it chooses $g_{i,0}, g_{i,1}, g_{i,2} \xleftarrow{\$} \mathbb{G}_1, h_0 \xleftarrow{\$} \mathbb{G}_2, \alpha \xleftarrow{\$} \mathbb{Z}_p$ 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 $(\mathrm{PK}, \mathrm{SK})$.

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

$\mathbf{SB.Vrfy}(\mathrm{PK}, (m_i)_{i=1}^n, (\tau, (\sigma_i)_{i=1}^n)) \to 1/0$. 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 10 (Unforgeability of Our SB) *Our signature-bundle scheme* $\mathtt{SB}$ *is existentially unforgeable against chosen-message attack under the Strong Diffie-Hellman assumption.*

Proof. Everything can be done as in [38] 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 $r_i, t_i, u_i, \bar{\alpha}_i \leftarrow \mathbb{Z}_p$ and implicitly puts $\alpha_i := \bar{\alpha}_i\alpha$, and puts $g_{i,2} := g^{f_{j^*}(\alpha_i)}, g_{i,1} := g_{i,2}^{r_i}, g_{i,0} := g_{i,2}^{(\alpha_i+e_{i^*})t_i-u_i} = (g^{f_{j^*}(\alpha_i)})^{(\alpha_i+e_{j^*})t_i-u_i} = g^{f(\alpha_i)t_i}g^{-u_if_{j^*}(\alpha_i)}$, $s_{i^*} := u_i - rm_{i^*}, A_{i^*} := g_{i,2}^{t_i}$. Then,

$$A_{j^*}^{\alpha_i+e_{j^*}} = (g_{i,2}^{t_i})^{\alpha_i+e_{j^*}} = g_2^{(\alpha_i+e_{j^*})t_i-u_i+u_i} = g_{i0}g_{i_2}^{u_i} = g_{i0}g_{i_2}^{r_im_{j^*}+s_{j^*}} = g_{i0}g_{i_1}^{m_{j^*}}g_2^{s_{j^*}}.$$

This completes the simulation of the signature-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 [38] with division by $\bar{\alpha}_i$. □

Our ABID in DL Using CL-SB 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 executes a group generator $\mathtt{BlGrp}(1^\lambda)$ to get $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e(\cdot,\cdot))$. For $i \in \mathcal{U}$, it chooses $g_{i,0}, g_{i,1}, g_{i,2} \xleftarrow{\$} \mathbb{G}_1, h_0 \xleftarrow{\$} \mathbb{G}_2, \alpha \xleftarrow{\$} \mathbb{Z}_p, h_1 := h_0^\alpha$ and a hash key $\mu \xleftarrow{\$} \mathit{Hashkeysp}(\lambda)$ 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.

ABID.**KG**$(\mathrm{MSK}, \mathrm{PK}, S) \to \mathrm{SK}_S$. Given PK, MSK and an attribute subset S, it chooses $e \xleftarrow{\$} \mathbb{Z}_p$. For $i \in S$, it computes $a_i \leftarrow \mathit{Hash}_\mu(i)$, $s_i \xleftarrow{\$} \mathbb{Z}_p$, $A_i := (g_{i,0}g_{i,1}^{a_i}g_{i,2}^{-s_i})^{\frac{1}{\alpha+e}} \in \mathbb{G}_1$. 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 $r_i \xleftarrow{\$} \mathbb{Z}_p$. If $i \in S$ then $s'_i := s_i + er_i, A'_i := A_ig_{i,2}^{-r_i} \in \mathbb{G}_1$. Else $s'_i \xleftarrow{\$} \mathbb{Z}_p$, $A'_i \xleftarrow{\$} \mathbb{G}_1$. $\mathcal{P}$ 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 $x := (x_i := (Z_i, Z_{i,1}, Z_{i,2}))_i$ 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 $\mathbf{\Sigma}_f$ for the language L_f. In other words, $\mathcal{P}$ and $\mathcal{V}$ execute $\mathbf{ZKPoK}[(e_{\rho(l)}, s'_{\rho(l)})_l, l \in \mathrm{Leaf}(\mathcal{T}_f) : \mathit{equations}]$, 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:

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

$\mathcal{V}$ 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 signature-bundle scheme* `SB` *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 for `SB`. As for the extraction of a signature bundle, we can do it under the condition that the same e is answered. □

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-SB as Witness

ABTTS.Setup and **ABTTS.PKG** are the same as **ABID.Setup** and **ABID.KG** in Section C.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 [7]. 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 $(\textsc{Cmt}_l)_l$, and the inner state st of $\mathcal{P}$ with the commitment is included in the secondary secret key; $\mathrm{SSK}_{S,f} := (w, (\textsc{Cmt}_l)_l \parallel st)$, $\mathrm{SPK}_f := (x, (\textsc{Cmt}_l)_l)$. **ABTTS.Sign** and **ABTTS.Vrfy** run the remaining protocol of our `ABID` in the two-tier framework [7] as in Section 7. The signature is:

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

Security of Our ABTTS in DL Using CL-SB

Theorem 11 (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. [7] as well as Theorem 10 and Claim 2, we deduce the claim. □

Theorem 12 (Attribute Privacy) *Our attribute-based two-tier signature scheme* `ABTTS'` *has attribute privacy.*

Proof. The witness-hiding property assures the attribute privacy. □