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

Hiroaki Anada\textsuperscript{1}, Seiko Arita\textsuperscript{2}, and Kouichi Sakurai\textsuperscript{3,4}

\textsuperscript{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

\textsuperscript{2} Institute of Information Security
509, 2-14-1, Tsuruya-cho, Kanagawa-ku, Yokohama, 221-0835 JAPAN
arita@iisec.ac.jp

\textsuperscript{3} Department of Informatics, Kyushu University
W2-712, 744, Motooka, Nishi-ku, Fukuoka, 819-0395 JAPAN
sakurai@inf.kyushu-u.ac.jp

\textsuperscript{4} Institute of Systems, Information Technologies and Nanotechnologies
7F, Fukuoka SRP Center Bldg., 2-1-22, Momochihama, Sawara-ku, Fukuoka, 814-0001 JAPAN

August 31, 2016

Abstract. We propose a concrete procedure of a \(\Sigma\)-protocol proving knowledge that a set of witnesses satisfies a monotone predicate in witness-indistinguishable manner. Inspired by the high-level proposal by Cramer, Damgård and Schoenmakers at CRYPTO ’94, we construct the concrete procedure by extending the so-called OR-proof. Next, using as a witness a signature-bundle of the Fiat-Shamir signatures, we provide an attribute-based identification scheme (ABID). Then, applying the Fiat-Shamir transform to our ABID, we obtain an attribute-based signature scheme (ABS). These generic schemes are constructed from a given \(\Sigma\)-protocol, and the latter scheme has a feature of linkable signatures. 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. We provide two directions of instantiation. One is to use the Guillou-Quisquater and the Schnorr \(\Sigma\)-protocols, which produce ABID, ABS and ABTTS schemes with a loose security reduction in the random oracle model in pairing-free. The other is to use the Camenisch-Lysyanskaya \(\Sigma\)-protocols in the RSA setting and discrete-logarithm setting, which produce ABTTS schemes with a tighter security reduction in the standard model.

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

1 Introduction

A \(\Sigma\)-protocol formalized in the doctoral thesis of Cramer [13] is a protocol of a 3-move public-coin interactive proof system with completeness, special soundness and honest-verifier zero-knowledge. It is one of the simplest protocols of zero-knowledge interactive proof systems with an easy simulator. Also, it is one of the most typical

\textsuperscript{*} The first and the second authors are partially supported by \textit{kakenhi} Grant-in-Aid for Scientific Research (C) JP15K00029 from Japan Society for the Promotion of Science.
proof of knowledge systems [6]; witness-extraction property by the special soundness enables us to prove that an identification scheme by a \( \Sigma \)-protocol is secure against active and concurrent attacks via a reduction to a number-theoretic assumption [7]. Instantiations of the \( \Sigma \)-protocol have been known as the Schnorr protocol [46] and the Guillou-Quisquater protocol [28] of identification schemes. They can be converted into digital signature schemes by the Fiat-Shamir heuristic [20]. The signature scheme can be proved secure against chosen-message attacks in the random oracle model [43], based on the security of the identification scheme against passive attacks [1]. By virtue of these features, a \( \Sigma \)-protocol can be adopted into building blocks of various cryptographic primitives such as anonymous credential systems [12] and group signature schemes [11].

The OR-proof proposed by Cramer, Damgård and Schoenmakers at CRYPTO ’94 [14] is a \( \Sigma \)-protocol derived from an original \( \Sigma \)-protocol [15]. It is a witness-indistinguishable protocol [19] by which a prover can convince a verifier that a prover knows one of two (or both) witnesses while even an unbounded verifier cannot tell which witness is used. The OR-proof is essentially applied in, for example, the construction of a non-malleable proof of plaintext knowledge [35]. In the paper of Cramer et al. [14], a more general protocol was proposed\(^5\); suppose a prover and a verifier are given a monotone boolean predicate \( f \) over boolean variables. Here a monotone boolean predicate means a boolean predicate without negation; that is, boolean variables connected by AND-gates and OR-gates, but no NOT-gate is used. ‘1’ (TRUE) is substituted into every variable in \( f \) at which the prover knows the corresponding witness, and ‘0’ (FALSE) is substituted into every remaining variable. The protocol attains witness-indistinguishability in the sense that the prover knows a satisfying set of witnesses while even an unbounded verifier cannot tell which satisfying set is used. This protocol is an extension of the OR-proof to any monotone boolean predicate, and in [14] a high-level construction that employed a “semi-smooth” secret-sharing scheme was given. (As is explained in [14], to remove the restriction of the monotonicity of \( f \) looks hard.)

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

Then, we will try to apply the protocol \( \Sigma_f \) to construct an attribute-based identification scheme (ABID) and an attribute-based signature scheme (ABS). In ABID, an identification-session is associated with an access structure described as a boolean predicate over an attribute universe. A prover can make a verifier accept only when the prover’s set of attributes satisfies the access structure. ABS, in our strategy, is obtained by applying the Fiat-Shamir heuristic to ABID. The concept of ABS has been developed since 2008 [29, 47, 38, 37, 39, 34, 18, 41, 22, 33, 42, 30, 17, 16, 17, 23, 31, 45], however, almost all the constructions are via the approach similar to that of attribute-based encryption schemes (ABE, [44]), which uses bilinear maps (that is, pairings on elliptic curves). A few exception are generic constructions by Maji et al. [39] and Bellare et al. [5], and concrete constructions by Herranz in the RSA setting [30] and in the discrete logarithm setting [31]. In contrast to the approach by bilinear maps, we work through a different approach in the Fiat-Shamir paradigm [20], which shares a spirit with [30]. Note that, in this paper, we do not try to proceed the usual way to attain attribute-privacy [39, 41, 30] which means that signatures reveal nothing about the identity or attributes of the signer beyond what is explicitly revealed by the satisfied boolean predicate, but we will pursue the Fiat-Shamir approach. First, we construct a linkable attribute-based signature scheme. Then, after introducing a syntax of attribute-based two-tier signature scheme (ABTTS), we construct ABTTS to attain attribute-privacy paying expense of the secondary-key issuing [8].

### 1.1 Our Construction Idea

To provide a concrete procedure for the above protocol \( \Sigma_f \) with witness-indistinguishability, we look into the technique employed in the OR-proof [14] and expand it so that it can treat any monotone boolean predicate, as follows. First express the boolean predicate \( f \) as a binary tree \( T_f \). That is, we put leaves each of which corresponds to each position of a variable in \( f \). We connect two leaves by an \( \land \)-node or an \( \lor \)-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 \( \land \)-node or an \( \lor \)-node in the same way, until we reach to the root node (which is also an \( \land \)-node or an \( \lor \)-node).

A verification equation of the \( \Sigma \)-protocol is secure against active and concurrent attacks via a reduction to a number-theoretic assumption \([7]\). Instantiations of the \( \Sigma \)-protocol have been known as the Schnorr protocol \([46]\) and the Guillou-Quisquater protocol \([28]\) of identification schemes. They can be converted into digital signature schemes by the Fiat-Shamir heuristic \([20]\). The signature scheme can be proved secure against chosen-message attacks in the random oracle model \([43]\), based on the security of the identification scheme against passive attacks \([1]\). By virtue of these features, a \( \Sigma \)-protocol can be adopted into building blocks of various cryptographic primitives such as anonymous credential systems \([12]\) and group signature schemes \([11]\).

The OR-proof proposed by Cramer, Damgård and Schoenmakers at CRYPTO ’94 \([14]\) is a \( \Sigma \)-protocol derived from an original \( \Sigma \)-protocol \([15]\). It is a witness-indistinguishable protocol \([19]\) by which a prover can convince a verifier that a prover knows one of two (or both) witnesses while even an unbounded verifier cannot tell which witness is used. The OR-proof is essentially applied in, for example, the construction of a non-malleable proof of plaintext knowledge \([35]\). In the paper of Cramer et al. \([14]\), a more general protocol was proposed\(^5\); suppose a prover and a verifier are given a monotone boolean predicate \( f \) over boolean variables. Here a monotone boolean predicate means a boolean predicate without negation; that is, boolean variables connected by AND-gates and OR-gates, but no NOT-gate is used. ‘1’ (TRUE) is substituted into every variable in \( f \) at which the prover knows the corresponding witness, and ‘0’ (FALSE) is substituted into every remaining variable. The protocol attains witness-indistinguishability in the sense that the prover knows a satisfying set of witnesses while even an unbounded verifier cannot tell which satisfying set is used. This protocol is an extension of the OR-proof to any monotone boolean predicate, and in \([14]\) a high-level construction that employed a “semi-smooth” secret-sharing scheme was given. (As is explained in \([14]\), to remove the restriction of the monotonicity of \( f \) looks hard.)

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

Then, we will try to apply the protocol \( \Sigma_f \) to construct an attribute-based identification scheme (ABID) and an attribute-based signature scheme (ABS). In ABID, an identification-session is associated with an access structure described as a boolean predicate over an attribute universe. A prover can make a verifier accept only when the prover’s set of attributes satisfies the access structure. ABS, in our strategy, is obtained by applying the Fiat-Shamir heuristic to ABID. The concept of ABS has been developed since 2008 \([29, 47, 38, 37, 39, 34, 18, 41, 22, 33, 42, 30, 17, 16, 17, 23, 31, 45]\), however, almost all the constructions are via the approach similar to that of attribute-based encryption schemes (ABE, \([44]\)), which uses bilinear maps (that is, pairings on elliptic curves). A few exception are generic constructions by Maji et al. \([39]\) and Bellare et al. \([5]\), and concrete constructions by Herranz in the RSA setting \([30]\) and in the discrete logarithm setting \([31]\). In contrast to the approach by bilinear maps, we work through a different approach in the Fiat-Shamir paradigm \([20]\), which shares a spirit with \([30]\). Note that, in this paper, we do not try to proceed the usual way to attain attribute-privacy \([39, 41, 30]\) which means that signatures reveal nothing about the identity or attributes of the signer beyond what is explicitly revealed by the satisfied boolean predicate, but we will pursue the Fiat-Shamir approach. First, we construct a linkable attribute-based signature scheme. Then, after introducing a syntax of attribute-based two-tier signature scheme (ABTTS), we construct ABTTS to attain attribute-privacy paying expense of the secondary-key issuing \([8]\).

\(^5\) 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.
to the left child and the right child, respectively. Here $\oplus$ means a bitwise exclusive-OR operation. Then the prover continues to apply this rule at each height, step by step, until he reaches to every leaf. Basically, the OR-proof technique assures that we can either honestly execute the $\Sigma$-protocol $\Sigma$ or execute the simulator of $\Sigma$. Only when a set of witnesses satisfies the binary tree $T_f$, 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 $\Sigma$-protocol of [14], which is comparable with the original abstract protocol [14]. That is, given a $\Sigma$-protocol $\Sigma$ and a monotone boolean predicate $f$, we construct a concrete procedure $\Sigma_f$ in a recursive form that is suitable for implementation. Then we show that $\Sigma_f$ is certainly a $\Sigma$-protocol with witness-indistinguishability.

Our second contribution is to provide a concrete schemes in two directions. One is to use the Guillou-Quisquater [28] and the Schnorr [46] $\Sigma$-protocols, which produce ABID, ABS and ABTTS schemes with a loose security reduction in the random oracle model in pairing-free. The other is to use the Camenisch-Lysyanskaya $\Sigma$-protocols in the RSA setting [12] and discrete-logarithm setting [21, 40, 48] to exit the drawbacks of the loose reduction. For the purpose, we introduce a syntax of attribute-based two-tier signature scheme, and construct concrete ABTTS schemes with a tighter reduction in the standard model.

1.3 Related Work on ABS

At a high level, our ABS is obtained by the Fiat-Shamir transform of our $\Sigma_f$, where a set of witnesses is the Fiat-Shamir signature-bundle (credential bundle [39]). This construction can be compared with the generic construction of the ABS scheme by Maji et al. [39]. They started with a signature bundle (of Boneh-Boyen signatures [10], for instance), then they employed a non-interactive witness-indistinguishable proof of knowledge system (NIWIPoK) of Groth and Sahai [27] to prove the knowledge of a signature bundle which satisfies a given (monotone) access formula, in the standard model.

Okamoto and Takashima (OT11) [41] gave an ABS scheme with full-security; security against adaptive target in the standard model under a non-$q$-type assumption; it can treat any non-monotone access formula and multi-use of attributes, and possesses attribute privacy in the information-theoretic sense. The construction is based on their Dual Pairing Vector Space.

Herranz [30] 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 [30], the concrete procedure was described in detail for threshold-type access formulas. In contrast, our ABS is without pairings and provides a concrete procedure for any monotone access formulas without attribute privacy. Recently, Herranz [31] provided an ABS scheme without pairings in the discrete-logarithm setting, but it has a constraint that the number of private secret keys is bounded in the set-up phase.

Kaafarani et al. [16] proposed the functionality of “User-Controlled Linkability” (UCL) in the case of attribute-based signatures. UCL property in the work [16] can be captured as a kind of public linkability. In general, public linkability is achieved with the expense of losing attribute privacy in ABS, and hence the scheme [16] and our ABS scheme do not possess attribute privacy.

1.4 Technical and Efficiency Comparison on ABS

We compare our scheme with the above previously proposed schemes from the view point of security, functionality and length of a signature. The comparison is summarized in Table 1 with notations as follows. A prime of bit length $\lambda$ (the security parameter) is denoted by $p$. Though a pairing map $e$ should be analysed for the asymmetric bilinear groups [26], we simply evaluate for the symmetric case in which both source groups are $G_p$ of order $p$. We assume that an element of $G_p$ is represented by $2\lambda$ 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 [9]. DLIN means the Decisional Linear assumption [41]. DDH means the Decisional Diffie-Hellman assumption [16]. DL means the Discrete-Logarithm assumption [16]. $q$-SRSA means the strong RSA assumption with $q$-type input [12, 30]. DDH in $QR(N)$ means the Decisional Diffie-Hellman assumption for quadratic residues modulo $N$ (the RSA modulus) [30]. In [30, 31], $\theta$ is the threshold value of a threshold-type access structure. In
Table 1. Technical and Efficiency Comparison on ABS: Security, Functionality and Length of Signature.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Maji et al. [39]</td>
<td>Mono. Std.</td>
<td>$q$-SDH $\land$ DDIN</td>
<td>✓</td>
<td>✓ (info.)</td>
<td>✓</td>
<td>-</td>
<td>$(2\lambda) \times$ -</td>
</tr>
<tr>
<td>[41]</td>
<td>Non-mono. Std.</td>
<td>$\land$ CR</td>
<td>✓</td>
<td>✓ (info.)</td>
<td>✓</td>
<td>-</td>
<td>$(2\lambda)(9\ell + 11)$</td>
</tr>
<tr>
<td>Herranz [30]</td>
<td>Mono. R.O.</td>
<td>$q$-SRSA $\land$ DHI $\land$ CR</td>
<td>✓</td>
<td>✓ (comp.)</td>
<td>✓</td>
<td>-</td>
<td>$\lambda_{rsa}(5 + \frac{\kappa}{\sqrt{\theta}}) + 1$</td>
</tr>
<tr>
<td>Herranz [31]</td>
<td>Mono. R.O.</td>
<td>$\land$ CR</td>
<td>✓</td>
<td>✓ (info.)</td>
<td>✓</td>
<td>-</td>
<td>$+\lambda_{rsa}3 - \kappa(\theta - 1)$</td>
</tr>
<tr>
<td>Kaafarani et al. [16]</td>
<td>Mono. R.O. $\land$ DL $\land$ CR</td>
<td>✓</td>
<td>✓ (info.)</td>
<td>✓</td>
<td>-</td>
<td>-</td>
<td>$+\lambda_{LS}3\ell + 3$</td>
</tr>
<tr>
<td><strong>Our ABS</strong></td>
<td>Mono. R.O.</td>
<td>$\land$ CR</td>
<td>✓</td>
<td>✓ (info.)</td>
<td>✓</td>
<td>-</td>
<td>$(2\lambda)(2\ell)$ -</td>
</tr>
<tr>
<td><strong>Our ABTTS (FS-sig.)</strong></td>
<td>Mono. R.O. $\land$ CR</td>
<td>✓</td>
<td>✓ (info.)</td>
<td>✓</td>
<td>✓ (two-tier)</td>
<td>+λ</td>
<td></td>
</tr>
<tr>
<td><strong>Our ABTTS' (CL-sig.)</strong></td>
<td>Mono. Std. $\land$ CR</td>
<td>✓</td>
<td>✓ (info.)</td>
<td>✓</td>
<td>-</td>
<td>✓ (two-tier)</td>
<td>$λ(3\ell - 1)$ -</td>
</tr>
</tbody>
</table>


The rigorous notion of ABS scheme was pioneered by the work of Maji et al. [39]. The ABS scheme by Okamoto and Takashima [41] has advantages in the security model, the assumption, the treatable access formulas. The scheme by Herranz [30] is the only ABS scheme with the pairing-free feature, and with collusion resistance and computational attribute privacy and in the RSA setting. Our procedure $\Sigma_f$ of the $\Sigma$-protocol in [14] for any monotone predicate serves as a building block of the $\Sigma$-protocol of [30]. Note that the security parameter $\lambda_{rsa}$ in the RSA setting ([30], our ABS, our ABTTS and our ABTTS' in RSA) is almost 9 times longer than $\lambda$ in the discrete logarithm setting. For example, $\lambda_{rsa} = 2048$ achieves almost equivalent security of $\lambda = 224$ [49].

Note that the ABS scheme by Herranz [31] 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 have to be issued for each signing session and the secondary keys is used only one time. Hence we believe that there is still an open problem to construct a pairing-free efficient ABS scheme in the discrete-logarithm setting.

The ABS scheme by Kaafarani et al. [16] has a feature of the user-controlled linkability. In contrast, our ABS has only the public linkability. It is notable that the ABS scheme [16] uses pairings and can be set up in the multi-authorities setting [42, 17, 23].

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 $\Sigma$-protocol $\Sigma_f$. In Section 4, by using a signature-bundle of the Fiat-Shamir signatures as a witness of our $\Sigma_f$, we obtain our ABID. In Section 5, by applying the Fiat-Shamir transform to our ABID, we obtain our ABS. In Section 6, we define the syntax of ABTTS. In Section 7, by applying the technique of two-tier signature to our ABID, we obtain our ABTTS. In Section 8, we conclude our work in this paper. In Appendix A, B, C, D and E, we put the definitions of needed cryptographic primitives. In Appendix F and G, we show concrete instantiations of our ABID, ABS and ABTTS in the RSA setting and the discrete-logarithm setting.

2 Preliminaries

The security parameter is denoted by $\lambda$. Bit length of a string $x$ is denoted by $|x|$. A uniform random sampling of an element $a$ from a set $S$ is denoted as $a \in_R S$. 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 \( z \leftarrow (A(a), B(b)) \). 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 \( A^{\mathcal{O}_1, \ldots, \mathcal{O}_n}. \) Here “concurrent” means that \( A \) accesses 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 \( a =? b \) returns a value 1 (TRUE) when \( a = b \) and 0 (FALSE) otherwise. A probability of an event \( E \) is denoted by \( \Pr[E] \). A probability of an event \( E \) on condition that events \( E_1, \ldots, E_m \) occur in this order is denoted as \( \Pr[E_1, \ldots, E_m : E] \).

2.1 Language, Proof of Knowledge and \( \Sigma \)-protocol \([6, 14, 15]\)

**Language** Let \( R = \{(x, w)\} \subseteq \{0, 1\}^* \times \{0, 1\}^* \) be a binary relation. We say that \( R \) is polynomially bounded if there exists a polynomial \( \text{poly} \) such that \( |w| \leq \text{poly}(|x|) \) for any \((x, w) \in R \). 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 of \((x, w) \in R \).

A *language* for a relation \( R \) is defined as:

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

\( L_R \) 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:

\[
R(\cdot, \cdot) : \{0, 1\}^* \times \{0, 1\}^* \rightarrow \{0, 1\},
\]

\[
(x, w) \mapsto 1 \text{ if } (x, w) \in R, \text{ 0 otherwise}.
\]

Denote the set of witnesses each of which is a witness of a statement \( x \) by \( w(x) = \{w \in \{0, 1\}^*; R(x, w) = 1\} \).

**Proof of Knowledge** Informally, an interactive proof system \([4, 24]\) is a proof of knowledge system if the knowledge being proved can be efficiently computed by using the prover as a subroutine.

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

Completeness. For any statement \( x \in L_R \) and for any witness \( w \) such that \((x, w) \in R \), \( P \) with the witness \( w \) makes \( V \) accept for the statement \( x \) with probability 1:

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

**Knowledge Soundness.** There are a PPT algorithm \( \mathcal{KE} \) called a *knowledge extractor*, a function \( \kappa : \{0, 1\}^* \rightarrow \{0, 1\} \) called a *knowledge error function* and a constant \( c > 0 \) that satisfy the following: If there exists a PPT algorithm \( A \) that satisfies \( p(x) := \Pr[1 \leftarrow \langle A(x), V(x) \rangle] > \kappa(x) \), then \( \mathcal{KE}(x) \) that has oracle-access to \( A(x) \) 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-Indistinguishable Proof of Knowledge** \([19, 14]\) Informally, an interactive proof system \([4, 24]\) is witness indistinguishable if the verifier cannot tell which witness \( w \in w(x) \) the prover is using.

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

**Witness-Indistinguishability.** For any unbounded algorithm \( A \), we have

\[
\Pr[(x, w_0, w_1) \leftarrow A^{(1^\lambda)}, 1 \leftarrow \langle P(x, w_0), A \rangle] = \Pr[(x, w_0, w_1) \leftarrow A^{(1^\lambda)}, 1 \leftarrow \langle P(x, w_1), A \rangle]
\]

where \((R(x, w_0) = 1 \land R(x, w_1) = 1) \lor (R(x, w_0) = 0 \land R(x, w_1) = 0) \) holds.
The OR-proof [15]. Let \( R \) be an NP relation. A \( \Sigma \)-protocol on a relation \( R \) is a public-coin 3-move protocol of a proof of knowledge system \( \Pi = (\mathcal{P}, \mathcal{V}) \). \( \mathcal{P} \) sends the first message called a commitment \( \text{Cmt} \) to \( \mathcal{V} \), then \( \mathcal{V} \) sends the second message that is a public random string called a challenge \( \text{Cha} \) to \( \mathcal{P} \), and then \( \mathcal{P} \) answers with the third message called a response \( \text{Res} \) to \( \mathcal{V} \). Then \( \mathcal{V} \) applies a decision test on \((x, \text{Cmt}, \text{Cha}, \text{Res})\) to return 1 (accept) or 0 (reject). If \( \mathcal{V} \) accepts, then the triple \((\text{Cmt}, \text{Cha}, \text{Res})\) is said to be an accepting conversation on \( x \). Here \( \text{Cha} \) is chosen uniformly at random from \( \text{CHASP}(1^\lambda) := \{1,0\}^{\ell(\lambda)} \) with \( \ell(\cdot) \) being a super-log function.

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

A protocol must possess the following three requirements.

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

Special Soundness. Any PPT algorithm \( \mathcal{P}^* \) without any witness \( w \in w(x) \) can respond to only one possible challenge \( \text{Cha} \). In other words, there is a PPT algorithm called a knowledge extractor, \( \Sigma^{\text{KE}} \), 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 \((\text{Cmt}, \text{Cha}, \text{Res})\) and \((\text{Cmt}, \text{Cha}', \text{Res}')\) with \( \text{Cha} \neq \text{Cha}' \).

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

As a zero-knowledge proof of knowledge system, we denote \( \Sigma \) as \text{ZKPoK} \([w : x] \), where \( w \) is a witness whose knowledge is to be proved by a prover \( \mathcal{P} \), and \( x \) is a statement for which the prover \( \mathcal{P} \) and the verifier \( \mathcal{V} \) have conversation. Any \( \Sigma \)-protocol is actually known to be a protocol of a proof of knowledge system ([15]).

We will need in this paper a property called the unique answer property [8] that for legitimately produced commitment \( \text{Cmt} \) and challenge \( \text{Cha} \), there exists one and only one response \( \text{Res} =: \bar{w} \) that is accepted by a verifier. Known \( \Sigma \)-protocols such as the Schnorr protocol and the Guillou-Quisquater protocol [46, 7] possess this property. For such a unique answer \( \bar{w} \) we consider a statement \( \bar{x} \) such that \((\bar{x}, \bar{w}) \in R \). Then, we further assume that both a prover and a verifier can compute, in polynomial-time, such an \( \bar{x} \) from \((x, \text{Cmt}, \text{Cha}) \). We denote the PPT algorithm as \( \Sigma^{\text{stmtgen}} \). That is:

\[
\Sigma^{\text{stmtgen}}(x, \text{Cmt}, \text{Cha}) :
\begin{align*}
&\text{Compute } \bar{x} \text{ s.t. } \\
&\exists! \bar{w} \text{ s.t. } ((\text{Cmt}, \text{Cha}, \text{Res}) \text{ is an accepting conversation on } x \land \text{Res} = \bar{w} \land (\bar{x}, \bar{w}) \in R) \\
&\text{Return } \bar{x}
\end{align*}
\]

Known \( \Sigma \)-protocols [46, 7] possess this statement generation property (see Section F).

**The OR-proof [15]** Consider the following relation for a boolean predicate \( f(X_1, X_2) = X_1 \lor X_2 \).

\[
R_{\text{OR}} = \{(x = (x_0, x_1), w = (w_0, w_1)) \in \{1,0\}^* \times \{1,0\}^*; \\
f(R(x_0, w_0), R(x_1, w_1)) = 1\}.
\]

We consider a \( \Sigma \)-protocol \( \Sigma_{\text{OR}} \) on the relation \( R_{\text{OR}} \). The corresponding language is

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

Suppose that a \( \Sigma \)-protocol \( \Sigma \) on a relation \( R \) is given. Then we can construct the protocol \( \Sigma_{\text{OR}} \) on a relation \( R_{\text{OR}} \) as follows. For instance, suppose \((x_0, w_0) \in R \) holds. \( \mathcal{P} \) computes \( \text{Cmt}_0 \leftarrow \Sigma^1(x_0, w_0), \text{Cha}_1 \leftarrow \Sigma^2(1^\lambda), \) \( \langle \text{Cmt}_1, \text{Res}_1 \rangle \leftarrow \Sigma^{\text{sim}}(x_1, \text{Cha}_1) \) and sends \((\text{Cmt}_0, \text{Cmt}_1)\) to \( \mathcal{V} \). Then \( \mathcal{V} \) sends \( \text{Cha} \leftarrow \Sigma^2(1^\lambda) \) to \( \mathcal{P} \). Then, \( \mathcal{P} \) computes \( \text{Cha}_0 := \text{Cha} \oplus \text{Cha}_1, \text{Res}_0 \leftarrow \Sigma^3(x_0, w_0, \text{Cmt}_0, \text{Cha}_0) \) answers to \( \mathcal{V} \) with \((\text{Cha}_0, \text{Cha}_1) \) and \((\text{Res}_0, \text{Res}_1) \). Here \( \oplus \) denotes a bitwise exclusive-OR operation. Then both \((\text{Cmt}_0, \text{Cha}_0, \text{Res}_0) \) and \((\text{Cmt}_1, \text{Cha}_1, \text{Res}_1) \) are accepting conversations on \( x \) and have the same distribution as real accepting conversations. This protocol \( \Sigma_{\text{OR}} \) can be proved to be a \( \Sigma \)-protocol. We often call \( \Sigma_{\text{OR}} \) the OR-proof. The OR-proof is known to be witness-indistinguishable [14].
The Fiat-Shamir Transform [1] Suppose that a cryptographic hash function with a hash key \( \mu \) and with collision resistance, \( \text{Hash}_\mu(\cdot):\{0,1\}^* \rightarrow \{0,1\}^{(\lambda)} \), is given. We fix the hash key \( \mu \) hereafter. A \( \Sigma \)-protocol \( \Sigma \) on a relation \( R \) can be transformed into non-interactive witness-indistinguishable proof of knowledge system (NIWIPoK for short) [1]. When a \( \Sigma \)-protocol \( \Sigma \) is an identification scheme, the resulting scheme is a digital signature scheme [1]. The transform is described as follows. (Here, in the case of a NIWIPoK, the message \( m \) is empty.) Given a message \( m \in \{0,1\}^* \), execute: \( \text{CMT} \leftarrow \Sigma^l(x,w), \text{CHA} \leftarrow \text{Hash}_\mu(\text{CMT} \parallel m) \), \( \text{RES} \leftarrow \Sigma^r(x,w,\text{CMT},\text{CHA}) \). Then \( \sigma := (\text{CMT}, \text{RES}) \) is a signature on \( m \). We denote the above signing algorithm as \( \text{FS}(\Sigma)^\text{sign}(x,w,m) \rightarrow (\text{CMT}, \text{RES}) =: \sigma \). The verification algorithm \( \text{FS}(\Sigma)^\text{vrfy}(x,m,\sigma) \) is given as: \( \text{CHA} \leftarrow \text{Hash}_\mu(\text{CMT} \parallel m) \), Return \( b \leftarrow \Sigma^\text{vrfy}(x,\text{CMT},\text{CHA},\text{RES}) \).

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

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

3 Our Witness-Indistinguishable Proof of Knowledge on Monotone Predicates

In this section, we first construct a \( \Sigma \)-protocol \( \Sigma_f \) from a given \( \Sigma \)-protocol \( \Sigma \) and a monotone boolean predicate \( f \) so that \( \Sigma_f \) is a protocol of WIPoK on the relation \( R_f \).

3.1 Witness-Indistinguishable Proof of Knowledge on Monotone Predicates [14, 3]

We revisit the notion of a 3-move public-coin interactive proof of knowledge system introduced by Cramer, Damgård and Schoenmakers [14]. Then we restate the definitions for the sake of concreteness.

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

Definition 1 (Cramer, Damgård and Schoenmakers [14], Our Rewritten Form) A relation \( R_f \) is defined by:

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

\( R_f \) is a generalization of the relation \( R_{OR} \) for the OR-proof [14, 15], where \( f \) is a boolean predicate with the single boolean connective OR: \( X_1 \lor X_2 \). 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 \( \lambda \). The corresponding language for the relation \( R_f \) is given as follows.

\[
L_f \overset{\text{def}}{=} \{x \in \{0,1\}^*; \exists w, (x, w) \in R_f\}.
\]

3.2 Our Procedure \( \Sigma_f \) of WIPoK on Relation \( R_f \)

We will provide a concrete procedure \( \Sigma_f \) of a \( \Sigma \)-protocol of WIPoK on the relation \( R_f \). \( \Sigma_f \) 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 for \( \mathcal{V} \), where \( (x := (x_{i_1})_{1 \leq j \leq \text{arity}(f)}), w := (w_{i_1})_{1 \leq j \leq \text{arity}(f)} \in R_f \). In our prover algorithm \( \mathcal{P} \), there are three PPT subroutines \( \Sigma_f^\text{val}, \Sigma_f^1 \text{ and } \Sigma_f^3 \). On the other hand, in our verifier algorithm \( \mathcal{V} \), there are two PPT subroutines \( \Sigma_f^l \text{ and } \Sigma_f^r \). Moreover, \( \Sigma_f^r \) has two subroutines \( \text{Vrfy Cha} \) and \( \text{Vrfy Res} \). Fig. 1 shows the construction of our procedure \( \Sigma_f \). (For the tree expressions of a boolean predicate \( f \), see Appendix C.)

Evaluation of Satisfiability The prover \( \mathcal{P} \) begins with evaluation of whether and how \( S \) satisfies \( f \) by running the evaluation algorithm \( \Sigma_f^\text{val} \). It labels each node of \( T \) with a value \( v = 1 \) (TRUE) or \( 0 \) (FALSE). For each leaf \( l \), we label \( l \) with \( v_l = 1 \) if \( \rho(l) \in S \) and \( v_l = 0 \) otherwise. (For the definition of the function \( \rho \), see Appendix C.) For each inner node \( n \), we label \( n \) with \( v_n = v_{n_l} \land v_{n_R} \) or \( v_n = v_{n_l} \lor v_{n_R} \) according to AND/OR evaluation.
\[ P(x, w, f) : \]
\[ \Sigma_f^\text{eval}(T_r, S) \rightarrow (v_n)_n \]
If \( v_r(T_r) \neq 1 \), then abort
else \( \text{CHAv}_{r(T_r)} := * \)

\[ \Sigma_f^1(x, w, T_r, (v_n)_n, \text{CHAv}_{r(T_r)}) \rightarrow (((\text{CMTr})_l, (\text{CHAn})_n, (\text{Res}_l)_l) \rightarrow (\text{CMTr})_l) \rightarrow \]
\[ \text{CHAv}_{r(T_r)} := \text{CHA} \]
\[ \text{CHA} \leftarrow \Sigma_f^2(1^\lambda) \]

\[ \Sigma_f^2(x, w, T_r, (v_n)_n, \text{CMTr})_l, (\text{CHAn})_n, (\text{Res}_l)_l) \rightarrow (((\text{CHAn})_n, (\text{Res}_l)_l) \rightarrow (\text{CMTr})_l, (\text{CHAn})_n, (\text{Res}_l)_l) \rightarrow \rightarrow b, \text{Return } b \]

**Fig. 1.** Our WIPoK \( \Sigma_f \) on the relation \( R_f \).  

of two labels of its two children \( n_L, n_R \). The computation is executed for every node from the root to each leaf, recursively, in the following way:

\[ \Sigma_f^\text{eval}(T, S) : \]
\[ T_L := \text{Lsub}(T), T_R := \text{Rsub}(T) \]
If \( r(T) \) is an \( \land \)-node, then Return \( v_r(T) := (\Sigma_f^\text{eval}(T_L, S) \land \Sigma_f^\text{eval}(T_R, S)) \)
else if \( r(T) \) is an \( \lor \)-node, then Return \( v_r(T) := (\Sigma_f^\text{eval}(T_L, S) \lor \Sigma_f^\text{eval}(T_R, S)) \)
else if \( r(T) \) is a leaf, then Return \( v_r(T) := (\rho(r(T)) \in ? S) \)

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

\[ \Sigma_f^1(x, w, T_r, (v_n)_n, \text{CHAv}) : \]
\[ T_L := \text{Lsub}(T), T_R := \text{Rsub}(T) \]
If \( r(T) \) is an \( \land \)-node, then Return \( \text{CHAv}_{r(T_L)} := \text{CHA}, \text{CHAv}_{r(T_R)} := \text{CHA} \)
Return(\( \text{CHAv}_{r(T_L)}, \Sigma_f^1(x, w, T_r, (v_n)_n, \text{CHAv}_{r(T_L)}), \text{CHAv}_{r(T_R)}, \Sigma_f^1(x, w, T_r, (v_n)_n, \text{CHAv}_{r(T_R)}) \))
else if \( r(T) \) is \( \lor \)-node, then
If \( v_r(T_L) = 1 \land v_r(T_R) = 1 \), then \( \text{CHAv}_{r(T_L)} := * \), \( \text{CHAv}_{r(T_R)} := * \)
else if \( v_r(T_L) = 1 \land v_r(T_R) = 0 \), then \( \text{CHAv}_{r(T_L)} := * \), \( \text{CHAv}_{r(T_R)} := \Sigma^2(1^\lambda) \)
else if \( v_r(T_L) = 0 \land v_r(T_R) = 1 \), then \( \text{CHAv}_{r(T_L)} := \Sigma^2(1^\lambda), \text{CHAv}_{r(T_R)} := * \)
else if \( v_r(T_L) = 0 \land v_r(T_R) = 0 \), then \( \text{CHAv}_{r(T_L)} := \Sigma^2(1^\lambda), \text{CHAv}_{r(T_R)} := \Sigma^2(1^\lambda), \text{CHAv}_{r(T_L)} := \Sigma^2(1^\lambda), \text{CHAv}_{r(T_R)} := \Sigma^2(1^\lambda) \)
Return(\( \text{CHAv}_{r(T_L)}, \Sigma_f^1(x, w, T_r, (v_n)_n, \text{CHAv}_{r(T_L)}), \text{CHAv}_{r(T_R)}, \Sigma_f^1(x, w, T_r, (v_n)_n, \text{CHAv}_{r(T_R)}) \))
else if \( r(T) \) is a leaf, then
If \( v_r(T) = 1 \), then \( \text{CMTr}_r(T) := \Sigma^1(x_{\rho(r(T))}, w_{\rho(r(T))}), \text{Res}_r(T) := * \)
else if \( v_r(T) = 0 \), then \( (\text{CMTr}_r(T), \text{Res}_r(T)) := \Sigma_f^\text{sim}(x_{\rho(r(T))}, \text{CHA}) \)
Return(\( \text{CMTr}_r(T), \text{Res}_r(T) \))

**Fig. 2.** The subroutine \( \Sigma_f^1 \) of our \( \Sigma_f \).

**Challenge** \( V \) chooses a challenge value (that is, a public coin) by \( \Sigma^2 \).

\[ \Sigma_f^2(1^\lambda) : \text{CHA} \leftarrow \Sigma^2(1^\lambda), \text{Return}(\text{CHA}) \]

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

\[
\Sigma_f^{\text{f}}(x, w, T, (v_n)_n, (\text{CMT})_{l}), (\text{CHA})_{n}, (\text{RES})_{l}):
\]
\[
\begin{align*}
\mathcal{L}_l & := \text{Lsub}(T), \quad \mathcal{R}_l := \text{Rsub}(T) \\
\text{if } r(T) \text{ is } \wedge \text{-node, then } & \quad \text{CHA}_r(\mathcal{L}_l) := \text{CHA}_r(T), \quad \text{CHA}_r(\mathcal{R}_l) := \text{CHA}_r(T) \\
\text{Return}(\text{CHA}_r(\mathcal{L}_l), \Sigma_f^{\text{f}}(x, w, \mathcal{L}_l, (v_n)_n, (\text{CMT})_{l}), (\text{CHA})_{n}, (\text{RES})_{l})) \\
\text{else if } r(T) \text{ is } \vee \text{-node, then } & \\
\text{if } v_r(\mathcal{L}_l) = 1 \land v_r(\mathcal{R}_l) = 1, \text{ then } & \quad \text{CHA}_r(\mathcal{L}_l) := \Sigma^2(1^{\Lambda}), \quad \text{CHA}_r(\mathcal{R}_l) := \text{CHA}_r(T) \oplus \text{CHA}_r(\mathcal{L}_l) \\
\text{else if } v_r(\mathcal{L}_l) = 1 \land v_r(\mathcal{R}_l) = 0, \text{ then } & \quad \text{CHA}_r(\mathcal{L}_l) := \text{CHA} \oplus \text{CHA}_r(\mathcal{R}_l), \quad \text{CHA}_r(\mathcal{R}_l) := \text{CHA}_r(T) \\
\text{else if } v_r(\mathcal{L}_l) = 0 \land v_r(\mathcal{R}_l) = 1, \text{ then } & \quad \text{CHA}_r(\mathcal{L}_l) := \text{CHA}_r(T), \quad \text{CHA}_r(\mathcal{R}_l) := \text{CHA}_r(\mathcal{L}_l) \oplus \text{CHA}_r(T) \\
\text{else if } v_r(\mathcal{L}_l) = 0 \land v_r(\mathcal{R}_l) = 0, \text{ then } & \quad \text{CHA}_r(\mathcal{L}_l) := \text{CHA}_r(T), \quad \text{CHA}_r(\mathcal{R}_l) := \text{CHA}_r(\mathcal{R}_l) \oplus \text{CHA}_r(T) \\
\text{Return}(\text{CHA}_r(\mathcal{L}_l), \Sigma_f^{\text{f}}(x, w, \mathcal{L}_l, (v_n)_n, (\text{CMT})_{l}), (\text{CHA})_{n}, (\text{RES})_{l})) \\
\text{else if } r(T) \text{ is a leaf, then } & \\
\text{if } v_r(T) = 1, \text{ then } & \quad \text{RES}_r(T) := \Sigma^2(x_{\rho_r(T)}, w_{\rho_r(T)}, \text{CMT}_r(T), \text{CHA}_r(T)) \\
\text{else if } v_r(T) = 0, \text{ then } & \quad \text{RES}_r(T) := \text{RES}_r(T) \\
\text{Return}(\text{RES}_r(T)) \\
\end{align*}
\]

Fig. 3. The subroutine $\Sigma_f^{\text{f}}$ of our $\Sigma_f$.

Verification $V$ computes a decision by running from the root to each leaf, recursively, the following algorithm $\Sigma_f^{\text{f}}$.

\[
\Sigma_f^{\text{f}}(x, T, \text{CHA}, (\text{CMT})_{l}), (\text{CHA})_{n}, (\text{RES})_{l}):
\]
\[
\text{Return}(\text{VrfyCha}(T, \text{CHA}, (\text{CHA})_{n}) \land \text{VrfyRes}(x, T, (\text{CMT})_{l}, \text{CHA}, (\text{RES})_{l}))
\]

\[
\text{VrfyCha}(T, \text{CHA}, (\text{CHA})_{n}) :
\]
\[
\begin{align*}
\mathcal{L}_l & := \text{Lsub}(T), \quad \mathcal{R}_l := \text{Rsub}(T) \\
\text{if } r(T) \text{ is an } \wedge \text{-node, then } & \quad \text{Return } \left( (\text{CHA} =? \text{CHA}_r(\mathcal{L}_l)) \land (\text{CHA} =? \text{CHA}_r(\mathcal{R}_l)) \right. \\
\land & \quad \text{VrfyCha}(\mathcal{L}_l, \text{CHA}_r(\mathcal{L}_l), (\text{CHA})_{n}) \land \text{VrfyCha}(\mathcal{R}_l, \text{CHA}_r(\mathcal{R}_l), (\text{CHA})_{n}) \\
\text{else if } r(T) \text{ is an } \vee \text{-node, then } & \quad \text{Return } \left( (\text{CHA} =? \text{CHA}_r(\mathcal{L}_l) \oplus \text{CHA}_r(\mathcal{R}_l)) \right. \\
\land & \quad \text{VrfyCha}(\mathcal{L}_l, \text{CHA}_r(\mathcal{L}_l), (\text{CHA})_{n}) \land \text{VrfyCha}(\mathcal{R}_l, \text{CHA}_r(\mathcal{R}_l), (\text{CHA})_{n}) \\
\text{else if } r(T) \text{ is a leaf, then } & \\
\text{then Return } (\text{CHA} \in? \text{CHA}_{\text{Sp}}(1^{\Lambda})) \\
\end{align*}
\]

\[
\text{VrfyRes}(x, T, (\text{CMT})_{l}, \text{CHA}, (\text{RES})_{l}) :
\]
\[
\begin{align*}
\text{For } l \in \text{Leaf}(T) : & \quad \text{if } \Sigma^{\text{f}}(x_{\rho_l(T)}, \text{CMT}_l, \text{CHA}_l, \text{RES}_l) = 0, \text{ then Return } (0) \\
\text{Return } (1)
\end{align*}
\]

Now we have to check that $\Sigma_f$ is certainly a $\Sigma$-protocol on the relation $R_f$.

**Proposition 1 (Completeness)** Completeness holds for our $\Sigma_f$.

**Proof.** Suppose that $v_r(T_l) = 1$. We show that, for every node in $\text{Node}(T_l)$, either $v_n = 1$ or $\text{CHA}_n \neq *$ holds after executing $\Sigma_l$. The proof is by induction on the height of $T_l$. The case of height 0 follows from $v_r(T_l) = 1$ and the completeness of $\Sigma$. Suppose that the case of height $k$ holds and consider the case of height $k + 1$. The construction of $\Sigma_f$ assures the case of height $k + 1$.

**Proposition 2 (Special Soundness)** Special soundness holds for our $\Sigma_f$.

9
We can construct a knowledge extractor $\Sigma_f^{KE}$ from a knowledge extractor $\Sigma^{KE}$ of the underlying $\Sigma$-protocol $\Sigma$ as follows.

$$
\Sigma_f^{KE}(x, (CMT, CHA, RES)_l, (CMT, CHA', RES')_l) :
$$
For $1 \leq j \leq \text{arity}(f)$ : $w_j^* := *

For $l \in \text{Leaf}(T_f)$

If $\text{CHA} \neq \text{CHA'}$, then $w_{\rho(l)}^* \leftarrow \Sigma^{KE}(x_{\rho(l)}, (CMT, CHA, RES)_l, (CMT, CHA', RES')_l)$

else If $w_{\rho(l)}^* = *$, then $w_{\rho(l)}^* \leftarrow \{1, 0\}^*$

Return $(w^* := (w_j^*)_{1 \leq j \leq \text{arity}(f)})$

Then Lemma 1 assures the proposition.

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

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

Suppose that all nodes in $\text{inode}(T_f)$ are $\wedge$-nodes. Then the above claim follows immediately because $\text{CHA} \neq \text{CHA'}$ 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 $\text{CHA}_{n^*} \neq \text{CHA'}_{n^*}$ because all nodes with height less than $h^*$ are $\wedge$-nodes. So at least one of children of $n^*$, say $n_{L^*}^*$ satisfies $\text{CHA}_{n_{L^*}^*} \neq \text{CHA'}_{n_{L^*}^*}$. Divide the tree $T_f$ 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 $\Sigma_f$.

**Proof.** This is the immediate consequence of honest-verifier zero-knowledge property of $\Sigma$. That is, we can construct a polynomial-time simulator $\Sigma_f^{im}$ which, on input $(\text{PK}, \text{CHA})$, outputs commitment and response message of $\Sigma_f$.

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

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

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

**Proof.** For a fixed statement $x$ and two witnesses $w_1$ and $w_2$ satisfying $R(x, w_1) = R(x, w_2) = 1$ or $R(x, w_1) = R(x, w_2) = 0$, $\mathcal{P}(x, w)$ and $\mathcal{V}(x)$ of $\Sigma_f$ generate transcripts $((\text{CMT})_l, \text{CHA}, (\text{CHA}_n)_n, (\text{RES})_l)$ that has the same distribution.

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

The Fiat-Shamir transform $\text{FS}(\cdot)$ can be applied to any $\Sigma$-protocol $\Sigma$ ([20, 1]). Therefore, the non-interactive version of our procedure $\Sigma_f$ is obtained.

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

### 3.4 Discussion

As is mentioned in [14], the $\Sigma$-protocol $\Sigma_f$ can be considered as aproto-type of an attribute-based identification scheme. Also, the non-interactive version $\text{FS}(\Sigma_f)$ can be considered a prototypical of an attribute-based signature scheme. That is, $\Sigma_f$ and $\text{FS}(\Sigma_f)$ are ABID and ABS without collusion resistance on secret keys, respectively.


4 Our Attribute-Based Identification Scheme

In this section, we provide an attribute-based identification scheme (ABID) by combining our $\Sigma$-protocol $\Sigma_f$ in Section 3 with a signature bundle of the Fiat-Shamir signatures. Our ABID is verifier-policy scheme [2].

4.1 Our ABID

By using a signature-bundle (see Appendix A) as a witness of our WIPOK system $\Sigma_f$ in Section 3, we obtain a verifier-policy attribute-based identification scheme, ABID [2]. Our ABID is collision resistant against collecting private secret keys. Fig. 4 shows our construction: ABID = \(\text{ABID.Setup, ABID.KG, } \mathcal{P}, \mathcal{V}\).

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

\[
\text{ABID.Setup}(1^\lambda, \mathcal{U}) : \\
(x_{\text{mst}}, w_{\text{mst}}) \leftarrow \text{Instance}_R(1^\lambda), \mu \in_R \text{Hashkeysp}(\lambda) \\
PK := (x_{\text{mst}}, \mathcal{U}, \mu), \text{MSK} := (w_{\text{mst}}) \\
\text{Return}(PK, \text{MSK})
\]

**ABID.KG** takes as input \(PK, \text{MSK}, S\). It chooses a PRF key \(k\) from the key space \(\text{PRFkeysp}(\lambda)\) at random and a random string \(\tau\) from \(\{0, 1\}^\lambda\) at random. Then it applies the signature-bundle technique [39] for each message \(m_i := (\tau \parallel i), i \in S\). Here we employ the Fiat-Shamir signing algorithm \(\text{FS(}\Sigma\text{)}\) (see 2.1). It returns \(\text{SK}_S\).

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

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

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

**Supp** runs for \(j, 1 \leq j \leq \text{arity}(f)\), and generates simulated keys \((a_{ij}, w_{ij})\) for \(i_j \notin S\).

\[
\text{Supp}(PK, \text{SK}_S, f) : \\
\text{For } j = 1 \text{ to arity}(f) : \\
\text{If } i_j \notin S, \text{ then } \\
m_{ij} := (\tau \parallel i_j), c_{ij} \leftarrow \text{PRF}_k(m_{ij} || 0) \\
(a_{ij}, w_{ij}) \leftarrow \Sigma^\text{sim}(x_{\text{mst}}, c_{ij}, \text{PRF}_k(m_{ij} || 1)) \\
\text{Return } (a_{ij}, w_{ij})_{1 \leq j \leq \text{arity}(f)}
\]

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

\[
\text{StmtGen}(PK, \tau, (a_{ij})_{1 \leq j \leq \text{arity}(f)}) : \\
\text{For } j = 1 \text{ to arity}(f) : \\
m_{ij} := (\tau \parallel i_j), c_{ij} \leftarrow \text{Hash}_\mu(a_{ij} \parallel m_{ij}) \\
x_{ij} \leftarrow \Sigma^\text{stmtgen}(x_{\text{mst}}, a_{ij}, c_{ij}) \\
\text{Return } (x_{ij})_{1 \leq j \leq \text{arity}(f)}
\]

Note that \((x_i, w_i) \in R\) for \(i \in S\) but \(\text{Pr}[(x_i, w_i) \in R] = \text{neg}(\lambda)\) for \(i \notin S\).
The above procedures are needed to input a pair of statement and witness, \((x = (x_{ij})_{1 \leq j \leq \text{arity}(f)}, w = (w_{ij})_{1 \leq j \leq \text{arity}(f)})\), to \(\Sigma_f\), into the prover of our procedure \(\Sigma_f\). Note here that \((x_{ij}, w_{ij}) \in R\) for any \(i_j \in S\). On the other hand, \((x_{ij}, w_{ij}) \notin R\) for any \(i_j \notin S\), without a negligible probability, \(\text{neg}(\lambda)\). Note also that \(\mathcal{P}\) has to send a string \(\tau\) and elements \((a_{ij})_{1 \leq j \leq \text{arity}(f)}\) to the verifier \(\mathcal{V}\).

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

Finally, \(\mathcal{P}\) and \(\mathcal{V}\) of our \(\text{ABID}\) execute the prover and the verifier of our procedure \(\Sigma_f\), respectively. \(\mathcal{V}\) returns 1 or 0 according to the return of the verifier of \(\Sigma_f\).

\[
\begin{align*}
\mathbf{ABID\ Setup}(1^\lambda, \mathcal{U}): & \quad (x_{\text{mst}}, w_{\text{mst}}) \leftarrow \text{Instance}_R(1^\lambda) \\
& \quad \mu \leftarrow \text{Hashkeysp}(\lambda) \\
& \quad \text{PK} := (x_{\text{mst}}, \mathcal{U}, \mu), \text{MSK} := (w_{\text{mst}}) \\
& \quad \text{Return}(\text{PK}, \text{MSK})
\end{align*}
\]

\[
\begin{align*}
\mathbf{ABID\ KG}(\text{PK}, \text{MSK}, S): & \quad k \leftarrow \text{PRFkeysp}(\lambda), \tau \leftarrow \{1, 0\}^\lambda \\
& \quad \text{For } i \in S \\
& \quad \quad m_i := (\tau \parallel i), a_i \leftarrow \Sigma_{\tau}(x_{\text{mst}}, w_{\text{mst}}) \\
& \quad \quad c_i \leftarrow \text{Hash}_\mu(a_i \parallel m_i), w_i \leftarrow \Sigma_{\tau}(x_{\text{mst}}, w_{\text{mst}}, a_i, c_i) \\
& \quad \text{SK}_S := (k, \tau, (a_i, w_i)_{i \in S}) \\
& \quad \text{Return SK}_S
\end{align*}
\]

\[
\begin{align*}
\mathcal{P}(\text{PK}, \text{SK}_S, f): & \quad \text{Supp}(\text{PK}, \text{SK}_S, f) \rightarrow (a_{ij}, w_{ij}) \textrm{ for any } i \in S \\
& \quad w := (w_{ij})_i \\
& \quad \text{StmtGen}(\text{PK}, \tau, (a_{ij})_j) \\
& \quad \rightarrow (x_{ij})_j =: x \\
& \quad \Sigma_{\tau}^f(\tau_f, S) \rightarrow (v_n)_n \\
& \quad \text{If } v_i(\tau_f) \neq 1, \text{ then abort} \\
& \quad \text{else } \text{Cha}_{\tau}(\tau_f) := * \\
& \quad \quad \Sigma_{\tau}^1(x, w, \tau_f, (v_n)_n, \text{Cha}_{\tau}(\tau_f)) \\
& \quad \quad \rightarrow ((\text{Cmt}_i)_i, (\text{Cha}_n)_n, (\text{Res}_i)_i) \\
& \quad \quad \rightarrow \tau, (a_{ij})_j, (\text{Cmt}_i)_i \\
& \quad \quad \rightarrow \text{StmtGen}(\text{PK}, \tau, (a_{ij})_j) \\
& \quad \quad \rightarrow (x_{ij})_j =: x \\
& \quad \quad \text{Cha}_{\tau}(\tau_f) := \text{Cha} \\
& \quad \quad \text{Cha} \leftarrow \Sigma_{\tau}^2(1^\lambda) \\
& \quad \quad \Sigma_{\tau}^2(x, w, \tau_f, (v_n)_n, (\text{Cmt}_i)_i, (\text{Cha}_n)_n, (\text{Res}_i)_i) \\
& \quad \quad \rightarrow ((\text{Cha}_n)_n, (\text{Res}_i)_i) \\
& \quad \quad \rightarrow (\text{Cha}_n)_n, (\text{Res}_i)_i \\
& \quad \quad \rightarrow b, \text{ Return } b
\end{align*}
\]

Fig. 4. The scheme of our \(\text{ABID}\).

4.2 Security of Our \(\text{ABID}\)

Theorem 4 (Concurrent Security) If the employed signature scheme \(FS(\Sigma)\) is existentially unforgeable against chosen-message attacks, then our \(\text{ABID}\) 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 (\(\text{neg}(\cdot)\) means a negligible function).

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

Note that \(FS(\Sigma)\) 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 \(FS(\Sigma)\) as follows. \(\mathcal{F}\) can answer to \(\mathcal{A}\’\)s key-extraction queries for a secret key \(\text{SK}_S\) because \(\mathcal{F}\) can query his signing oracle about \((m_i := \tau \parallel i; i \in S)\), where \(\mathcal{F}\) chooses \(\tau\) at random. \(\mathcal{F}\) can simulate any concurrent prover with \(\text{SK}_S\) which \(\mathcal{A}\) invokes because \(\mathcal{F}\) can generate \(\text{SK}_S\) 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 \(\Sigma_{\mathcal{F}}^{\text{KE}}\), \(\mathcal{F}\) obtains a witness \(w^*\), a set of attributes \(S^*\)
and a target access formula $f^*$ with $f^*(S^*) = 1$, Finally, $\mathcal{F}$ succeeds in making at least one valid signature $(a_i, w_i)$ for $i \in S^*$ due to $f^*(S^*) = 1$ and the special soundness. By the Reset Lemma [7], the advantage $\text{Adv}_{\text{ABID}, A}^{ca}(\lambda, \mathcal{U})$ is reduced to $\text{Adv}_{\text{FS}(\Sigma), \mathcal{F}}^{\text{euf-cma}}(\lambda)$ with a loss of exponent by 1/2.

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

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

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

**More on Reduction of Concurrent Security** We mean by "a number theoretic problem" the discrete-logarithm problem or the RSA-inverse problem ([7]). There exists the following (very loose) security reduction of both attacks in Section D.1. Here we denote $H_1$ as the maximum number of hash queries issued by forger $\mathcal{F}$ on $\text{FS}(\Sigma)$ in the random oracle model. This is because (as is in Section 2.1) we can reduce the advantage $\text{Adv}_{\text{FS}(\Sigma), \mathcal{F}}^{\text{euf-cma}}(\lambda)$ to the advantage $\text{Adv}_{\Sigma, B}^{pa}(\lambda)$ of passive security of the underlying $\Sigma$-protocol $\Sigma$, in the random oracle model, with a loss factor $q_H$. Applying the Reset Lemma [7], we can reduce $\text{Adv}_{\Sigma, B}^{pa}(\lambda)$ to the advantage $\text{Adv}_{\text{Grp}, S}^{\text{num.prob.}}(\lambda)$ of a PPT solver $S$ of a number theoretic problem, with a loss of exponent by 1/2.

## 5 Our Attribute-Based Signature Scheme

In this section, we provide an attribute-based signature scheme (ABS) by applying the Fiat-Shamir transform (Section 2.1) to our ABID in Section 4. Our ABS is collusion resistant against collecting private secret keys, and EUF-CMA secure in the random oracle model. We note that our ABS has attribute privacy only as one-time signature because of its linkability.

### 5.1 Our ABS

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

$\text{ABS.Setup}$ and $\text{ABS.KG}$ are the same as $\text{ABID.Setup}$ and $\text{ABID.KG}$, respectively.

$\text{ABS.Sign}$ takes as input $\text{PK, SK}_S$ and $(m, f)$. It runs $\text{Supp}((\text{PK, SK}_S, f))$, $\text{StmtGen}$ and the prover of our procedure $\Sigma_f$ with a challenge string $\text{CHA}$ obtained by hashing the string $(x \parallel (\text{CMT}_i))_l$/$m$. It returns a signature

$$\sigma = (t, (a_i)_j, (\text{CMT}_i)_l, (\text{CHA})_n, (\text{RES})_l).$$

$\text{ABS.Vrfy}$ takes as input $\text{PK}$, $(m, f)$ and $\sigma$. It utilizes $\text{StmtGen}$ and $\Sigma_f^{vrfy}$ to check validity of the pair $(m, f)$ and the signature $\sigma$ under the public key PK.

### 5.2 Security of Our ABS

Applying the standard technique in the work of Abdalla et al. [1] shows that the security of our ABS is equivalent to the security of an attribute-based identification scheme, ABID, against passive attacks, where our ABID is obtained by combining our $\Sigma$-protocol $\Sigma_f$ with the signature-bundle scheme of the Fiat-Shamir signature FS($\Sigma$).

**Theorem 5 (Unforgeability)** Our attribute-based signature scheme $\text{ABS}$ 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).

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

(2)
\textbf{ABS.Setup}(1^\lambda, U):
\begin{align*}
(x_{\text{inst}}, w_{\text{inst}}) &\leftarrow \text{Instance}_R(1^\lambda) \\
\mu &\leftarrow \text{Hashkeys}(\lambda) \\
PK &:= (x_{\text{inst}}, \mathcal{U}, \mu), \text{MSK} := (w_{\text{inst}}) \\
\text{Return}(PK, \text{MSK})
\end{align*}

\textbf{ABS.KG}(PK, MSK, S):
\begin{align*}
k &\leftarrow \text{PRFkeys}(\lambda), \tau \leftarrow \{0, 1\}^\lambda \\
\text{For} \ i \in S \\
m_i &:= (\tau \parallel i), a_i \leftarrow \Sigma^1(x_{\text{inst}}, w_{\text{inst}}) \\
c_i &\leftarrow \text{Hash}_\mu(a_i \parallel m_i), w_i \leftarrow \Sigma^3(x_{\text{inst}}, w_{\text{inst}}, a_i, c_i) \\
SK_S &:= (k, \tau, (a_i, w_i)_{i \in S}) \\
\text{Return} \ SK_S
\end{align*}

\textbf{ABS.Sign}(PK, SK_S, (m, f)):
\textbf{Supp}(PK, SK_S, f) \rightarrow (a_{ij}, w_{ij})_{j}
\begin{align*}
\textbf{StmtGen}(PK, \tau, (a_{ij})_{j}) \\
\rightarrow (x_{ij}) =: x
\end{align*}
\begin{align*}
\Sigma^\text{val}_j(T_f, S) &\rightarrow (v_n)_n \\
\text{If} \ \tau_{\tau(T_f)} \neq 1, \text{then abort} \\
\text{else} \ CHA_{\tau(T_f)} := *
\end{align*}
\begin{align*}
\Sigma^\downarrow_j(x, w, T_f, (v_n)_n, CHA_{\tau(T_f)}) &\rightarrow ((\text{Cmt}^i)_i, (\text{Chan}^n)_n, (\text{Res}^l)_l) \\
CHA &\leftarrow \text{Hash}_\mu(x \parallel (\text{Cmt}^i)_i \parallel m) \\
CHA_{\tau(T_f)} &:= \text{CHA}
\end{align*}
\begin{align*}
\Sigma^\uparrow_j(x, w, T_f, (v_n)_n, (\text{Cmt}^i)_i, (\text{Chan}^n)_n, (\text{Res}^l)_l) &\rightarrow ((\text{Chan}^n)_n, (\text{Res}^l)_l) \\
\text{Return} \ \sigma := (\tau, (a_{ij})_{j}, (\text{Cmt}^i)_i, (\text{Chan}^n)_n, (\text{Res}^l)_l)
\end{align*}

\textbf{ABS.Vrfy}(PK, (m, f), \sigma := (\tau, (a_{ij})_{j}, (\text{Cmt}^i)_i, (\text{Chan}^n)_n, (\text{Res}^l)_l)) :
\begin{align*}
\textbf{StmtGen}(PK, \tau, (a_{ij})_{j}) \\
\rightarrow (x_{ij}) =: x
\end{align*}
\begin{align*}
\Sigma^\text{val}_j(x, T_f, CHA, (\text{Cmt}^i)_i, (\text{Chan}^n)_n, (\text{Res}^l)_l) &\rightarrow b, \text{Return} \ b
\end{align*}
\begin{align*}
\text{CHA} &\leftarrow \text{Hash}_\mu(x \parallel (\text{Cmt}^i)_i \parallel m) \\
\text{CHA}_{\tau(T_f)} &:= \text{CHA}
\end{align*}
\begin{align*}
\Sigma^\downarrow_j(x, w, T_f, (v_n)_n, (\text{Cmt}^i)_i, (\text{Chan}^n)_n, (\text{Res}^l)_l) &\rightarrow ((\text{Chan}^n)_n, (\text{Res}^l)_l) \\
\text{Return} \ \sigma := (\tau, (a_{ij})_{j}, (\text{Cmt}^i)_i, (\text{Chan}^n)_n, (\text{Res}^l)_l)
\end{align*}

\textbf{Fig. 5.} The scheme of our ABS.
6.1 Chosen-Message Attack on ABTTS and Security Definition

Let $q_H$ denote the maximum number of hash queries issued by a forger $F$ on ABS and a forger $F'$ on FS(Σ). Combining the inequality (2) with the inequalities (4) and (1) in Section D and Section 4, we obtain the following (very loose) security reduction of advantages.

$$
\text{Adv}_{\text{ABS},F}^{\text{euf-cma}}(\lambda, U) \leq \frac{3}{2}q_H \left( \text{Adv}_{\text{Grp,SK}}^{\text{num,prob}}(\lambda) \right)^{1/4} + \text{neg}(\lambda).
$$

Attribute Privacy Our ABS does not have attribute privacy defined in Section E.2 because of its linkability; that is, the constant components $\tau, (a_i)_j$ make two signatures linkable. 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) according to the syntax of the two-tier signature scheme [8]. Then, we define a chosen-message attack on ABTTS by which an adversary makes an existential forgery, and define the existential unforgeability security against chosen-message attacks (EUF-CMA).

An attribute-based two-tier signature scheme, ABTTS, consists of five PPT algorithms: ABTTS = (ABTTS.Setup, ABTTS_PKG, ABTTS_SKG, ABTTS_Sign, ABTTS.Vrfy).

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

ABTTS_PKG(MSK, PK, S) → SSK. This PPT algorithm for primary-key generation takes as input the master secret key MSK, the public key PK and an attribute set $S \subset U$. It returns a secret key SSK that corresponds to $S$.

ABTTS_SKG(MSK, PK, SSK, f) → (SSK, SPK). This PPT algorithm for secondary-key generation takes as input the master secret key MSK, the public key PK, a secret key SSK and an access formula $f$. It returns a pair (SSK, SPK) of a secondary secret key and a secondary public key.

ABTTS_Sign(PK, SSK, SPK, (m, f)) → σ. This PPT algorithm for signing takes as input the public key PK, a secret key SSK, a secondary secret key SPK and a message $(m, f)$ of a message $m \in \{0, 1\}^*$ and an access formula $f$. It returns a signature $\sigma$.

ABTTS_Vrfy(PK, SPK, (m, f), σ) → 1/0. This deterministic polynomial-time algorithm for verification takes as input the public key PK, the secondary public key SPK and a signature $\sigma$. It returns a decision 1 or 0. When it is 1, we say that $(m, f)$ is valid. When it is 0, we say that $(m, f)$ is invalid.

We demand correctness of ABTTS that, for any $\lambda$, any $U$, any $S \subset U$ and any $(m, f)$ such that $f(S) = 1$, $\Pr[(\text{MSK, PK}) \leftarrow \text{ABTTS.Setup}(1^\lambda, U), (\text{SKS, SPK}) \leftarrow \text{ABTTS_PKG}(\text{MSK, PK, S}), (\text{SSK}, \text{SPK}) \leftarrow \text{ABTTS_SKG}(\text{MSK, PK, SSK}), (\text{PK}, (m, f)), b \leftarrow \text{ABTTS_Vrfy}(( \text{PK, SPK}, (m, f), \sigma) : b = 1)] = 1$.

6.1 Chosen-Message Attack on ABTTS and Security Definition

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

\[
\text{Exp}_{\text{ABTTS},F}^{\text{euf-cma}}(\lambda, U) :
\]
\[(PK, MSK) \leftarrow \text{ABTTS.Setup}(1^\lambda, U)
\]
\[
((m^*, f^*), \sigma^*) \leftarrow F^{PKG(MSK, PK, \cdot), SPKG(\cdot), SIGN(PK, SK, SSK_{S,f}, SPK_j(\cdot))}(PK)
\]
\[\text{If } \text{ABTTS.Vrfy}(PK, SPK_f, (m^*, f^*), \sigma^*) = 1
\]
\[\text{then Return WIN else Return LOSE}
\]

In the experiment, \( F \) issues key-extraction queries to its oracle \( PKG \), secondary public key queries to its oracle \( SPXG \) and signing queries to its oracle \( SIGN \). Giving an attribute set \( S_i \), \( F \) queries \( PKG(MSK, PK, \cdot) \) for a secret key \( SK_{S_i} \). Giving an attribute set \( S \) and an access formula \( f \), \( F \) queries \( SPKG(\cdot, \cdot) \) 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, \( F \) queries \( SIGN(PK, SK, SSK_{\cdot}, SPK_{\cdot}, (\cdot, \cdot)) \) for a valid signature \( \sigma \) 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 \( SIGN \) \[8\].

\( f^* \) is called a target access formula of \( F \). Here we consider the adaptive target case in the sense that \( F \) is allowed to choose \( f^* \) after seeing \( PK \) and issuing three queries. Two restrictions are imposed on \( F \) concerning \( f^* \). For all key-extraction queries, \( f^*(S_i) = 0 \). For all signing queries, \((m^*, f^*)\) was never queried and \( f^*(S_j) = 0 \). 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 \( \lambda \). The advantage of \( F \) over ABTTS is defined as

\[
\text{Adv}_{\text{ABTTS},F}^{\text{euf-cma}}(\lambda, U) \overset{\text{def}}{=} \Pr[\text{Exp}_{\text{ABTTS},F}^{\text{euf-cma}}(1^\lambda, U) \text{ returns WIN}].
\]

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

Then we define attribute privacy of ABTTS.

Definition 3 (Attribute Privacy of ABTTS) ABTTS is called to have attribute privacy if, for all \((PK, MSK) \leftarrow \text{ABTTS.Setup}(1^\lambda, U)\), for all message \( m \), for all attribute sets \( S_1 \) and \( S_2 \), for all primary secret keys \( SK_{S_1} \leftarrow \text{ABTTS.PKG}(PK, MSK, S_1) \) and \( SK_{S_2} \leftarrow \text{ABTTS.PKG}(PK, MSK, S_2) \), for all secondary secret keys \( (SSK_{S_1,f}, SPK_f) \leftarrow \text{ABTTS.SKG}(MSK, PK, SK, f) \) and \( (SSK_{S_2,f}, SPK_f) \leftarrow \text{ABTTS.SKG}(MSK, PK, SK, f) \) and for all access formula \( f \) such that \( \{f(S_1) = 1 \land f(S_2) = 1\} \lor \{f(S_1) \neq 1 \land f(S_2) \neq 1\} \), two distributions \( \sigma_1 \leftarrow \text{ABTTS.Sign}(PK, SK_{S_1}, SSK_{S_1,f}, SPK_f, (m,f)) \) and \( \sigma_2 \leftarrow \text{ABTTS.Sign}(PK, SK_{S_2}, SSK_{S_2,f}, SPK_f, (m,f)) \) 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. Collusion resistance against collecting private secret keys is assured by the issuer of second secret / public keys. Attribute privacy is assured by the witness-indistinguishability of the underlying procedure \( \Sigma_f \).

7.1 Our ABTTS

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

Fig. 6 shows our construction: \( \text{ABTTS} = (\text{ABTTS.Setup}, \text{ABTTS.PKG}, \text{ABTTS.SKG}, \text{ABTTS.Sign}, \text{ABTTS.Vrfy}) \). \( \text{ABTTS.Setup} \) and \( \text{ABTTS.PKG} \) are the same as \( \text{ABID.Setup} \) and \( \text{ABID.KG} \) in Section 4, respectively. \( \text{ABTTS.SKG}(MSK, PK, SK, f) \) takes as input \( MSK, PK, SK \) and \( f \). It uses a supplementary algorithm \( \text{Supp} \) and a statement-generator algorithm \( \text{StmtGen} \) 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 \( \Sigma_f \) to generate the first message as

\[
((\text{Cmt}_l)_t, st) \leftarrow \Sigma_f^1(x, w, T_f, (v_n)_t, \text{Cha}_r(T_f)).
\]

Then it puts \( \text{SSK}_{S,f} := (w, (\text{Cmt}_l)_t) \parallel st \) and \( \text{SPK}_f := (x, (\text{Cmt}_l)_t) \). Here \( st \) denotes the inner state of \( \mathcal{P} \). It returns \( \text{SSK}_{S,f} \) and \( \text{SPK}_f \). Note that the secondary public key \( \text{SPK}_f \) should be issued by a key-issuing center [8].

\[\text{ABTTS.Sign}(\text{PK}, \text{SK}_S, \text{SSK}_{S,f}, \text{SPK}_f, (m, f)) \rightarrow \sigma.\]

Given \( \text{PK} \), \( \text{SK}_S \), the secondary secret key \( \text{SSK}_{S,f} \), the secondary public key \( \text{SPK}_f \), and a pair \((m, f)\) of a message and an access formula \( f \), it computes a challenge \( \text{CHA} \) by hashing the string \( (\text{Cmt}_l)_t \parallel m \). Then, it runs the prover \( \mathcal{P} \) according to \( \Sigma_f \) as

\[
((\text{CHA}_n)_t, (\text{Res}_l)_t) \leftarrow \Sigma_f^2(x, w, T_f, (v_n)_t, (\text{Cmt}_l)_t, (\text{CHA}_n)_t, (\text{Res}_l)_t; st)
\]

Finally, it returns a signature

\[
\sigma := ((\text{CHA}_n)_t, (\text{Res}_l)_t).
\]

\[\text{ABTTS.Vrfy}(\text{PK}, \text{SPK}_f, (m, f), \sigma) \rightarrow 1/0.\]

Given \( \text{PK} \), the secondary public key \( \text{SPK}_f \), a pair \((m, f)\) and a signature \( \sigma \), it computes a challenge \( \text{CHA} \) by hashing the string \( (\text{Cmt}_l)_t \parallel m \). Then, it runs the verifier \( \mathcal{V} \) according to \( \Sigma_f \) as

\[
1 \text{ or } 0 \leftarrow \Sigma_f^\text{vrfy}(x, T_f, \text{CHA}, (\text{Cmt}_l)_t, (\text{CHA}_n)_t, (\text{Res}_l)_t).
\]

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 [8].

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

\[
\text{Adv}_{\text{ABTTS}, \mathcal{F}}^{\text{euf-cma}}(\lambda, \mathcal{U}) \leq q_H \text{Adv}_{\text{ABID}, \mathcal{B}}^{\text{cma}}(\lambda, \mathcal{U}) + \text{neg}(\lambda).
\]

**Proof.** We just note that the same argument in [8] is applied to our ABTTS. \( \square \)

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

**Proof.** A valid signature of ABTTS, \( \sigma := ((\text{CHA}_n)_t, (\text{Res}_l)_t) \), is a part of a valid proof of \( \Sigma_f \). According to the witness-indistinguishability of \( \Sigma_f \), the attribute privacy holds. \( \square \)

### 8 Conclusions

We provided a concrete procedure \( \Sigma_f \) of a \( \Sigma \)-protocol of the WIPoK system on monotone predicates. Our \( \Sigma_f \) can be considered as a proto-type of an attribute-based identification scheme, and also, FS(\( \Sigma_f \)) can be considered a proto-type of an attribute-based signature scheme [14], 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. 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 [8].

The scheme by Herranz [30] is the only ABS scheme with collusion resistance, computational attribute privacy and pairing-free property, in the RSA setting. Our procedure \( \Sigma_f \) of WIPoK on any monotone predicate serves as a building block of the \( \Sigma \)-protocol of the ABS scheme [30]. We believe that there is still an open problem to construct a pairing-free efficient ABS scheme in the discrete-logarithm setting.
ABTTS.Setup(1^λ, U):

(\xi_{\text{inst}}, w_{\text{inst}}) \leftarrow \text{Instance}_R(1^\lambda)

\mu \leftarrow \text{Hashkeys}(\lambda)

PK := (\xi_{\text{inst}}, U, \mu), \text{MSK} := (w_{\text{inst}})

\text{Return}(PK, \text{MSK})

ABTTS_PKG(PK, MSK, S):

k \leftarrow \text{PRFkeys}(\lambda), \tau \leftarrow \{1, 0\}^\lambda

\text{For } i \in S

m_i := (\tau \parallel i), a_i \leftarrow \Sigma^1(\xi_{\text{inst}}, w_{\text{inst}})

\chi_i \leftarrow \text{Hash}_{\mu}(a_i \parallel m_i), w_i \leftarrow \Sigma^3(\xi_{\text{inst}}, w_{\text{inst}}, a_i, \chi_i)

SK_S := (k, \tau, (a_i, w_i)_{i \in S})

\text{Return} SK_S

ABTTS.SKG(MSK, PK, SK_S, f) \rightarrow (SSK_S, f, SPK_f):

\textbf{Supp}(PK, SK_S, f) \rightarrow (a_{ij}, w_{ij})_{ij}

w := (w_{ij})_{ij}

\textbf{StmtGen}(PK, \tau, (a_{ij})_{ij})

\rightarrow (x_{ij})_{ij} := x

\Sigma^1_{\text{val}}(T_f, S) \rightarrow (\nu_n)_n

\text{If } \nu_r(T_f) \neq 1, \text{ then abort}

\text{else } \text{Cha}_{r(T_f)} := *

\Sigma^1_f(x, w, T_f, (\nu_n)_n, \text{Cha}_{r(T_f)})

\rightarrow ((\text{Cmt}_l)_l, (\text{Ch}_a)_n, (\text{Res}_l)_l; \text{st})

\text{SSK}_{S, f} := ((\text{Cmt}_l)_l \parallel \text{st})

\text{SPK}_f := ((x, (\text{Cmt}_l)_l))

\text{Return}(SSK_{S, f}, SPK_f)

ABTTS.Sign(PK, SK_S, SSK_{S, f}, SPK_f, (m, f)):

\textbf{Cha} \leftarrow \text{Hash}_{\mu}((\text{Cmt}_l)_l \parallel m)

\text{Cha}_{r(T_f)} := \text{Cha}

\Sigma^1_f(x, w, T_f, (\nu_n)_n, (\text{Cmt}_l)_l, (\text{Ch}_a)_n, (\text{Res}_l)_l; \text{st})

\rightarrow ((\text{Ch}_a)_n, (\text{Res}_l)_l)

\text{Return } \sigma := ((\text{Ch}_a)_n, (\text{Res}_l)_l)

ABTTS.Vrfy(PK, SPK_f, (m, f), \sigma)

\sigma := ((\text{Ch}_a)_n, (\text{Res}_l)_l)

\textbf{Cha} \leftarrow \text{Hash}_{\mu}((\text{Cmt}_l)_l \parallel m)

\Sigma^1_{\text{vrfy}}(x, T_f, \text{Cha}, (\text{Cmt}_l)_l, (\text{Ch}_a)_n, (\text{Res}_l)_l)

\rightarrow b, \text{Return } b

\textbf{Fig. 6. The scheme of our ABTTS.}
Acknowledgements We appreciate sincere comments from Javier Herranz in Universitat Politècnica de Catalunya, Barcelona, Spain via e-mail communication [32] 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 $\Sigma$-protocol on monotone predicates.

References

A Signature Bundle Scheme [39]

A signature bundle (a credential bundle in [39]) scheme SB is an extended notion of a digital signature scheme. It consists of three PPTs: SB = (SB.KG, SB.Sign, SB.Vrfy).

SB.KG(1^λ) → (PK, SK). Given 1^λ as input, it returns a public key PK and a secret key SK.

SB.Sign(PK, SK, (m_i)_i=1^n) → (τ, (σ_i)_i=1^n). Given PK, SK and messages (m_i)_i=1^n, it returns a tag τ and signatures (σ_i)_i=1^n, n is bounded by a polynomial in λ.

SB.Vrfy(PK, (m_i)_i=1^n, (τ, (σ_i)_i=1^n)) → 1/0. Given PK, messages (m_i)_i=1^n, a tag τ and signatures (σ_i)_i=1^n, it returns 1 or 0.

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

\[
\mathsf{Exprnt}_{\mathsf{SB},\mathcal{F}}^{\text{euf-cma}}(1^λ) = (PK, SK) \leftarrow \mathsf{SB.KG}(1^λ), ((m_i)_i=1^n, (τ, (σ_i)_i=1^n)) \leftarrow \mathcal{F}^{\mathsf{SBSign}}(PK)\\\text{If } \mathsf{SB.Vrfy}(PK, (m_i)_i=1^n, (τ, (σ_i)_i=1^n)) = 1 \\
\text{then Return WIN else Return LOSE}
\]

Giving a vector of messages (m_i)_i=1^n, \( \mathcal{F} \) queries \( \mathsf{SBSign}(PK, SK, \cdot) \) for a valid signature bundle \( (τ, (σ_i)_i=1^n) \). \( τ \) should be different from any queried tag \( τ \), or, whenever \( τ \) is equal to a queried tag \( τ \), it should hold that \( \{m_i\}_i=1^n \not\subseteq \{m_i\}_i=1^n \) for any queried \( (m_i)_i=1^n \). The advantage of \( \mathcal{F} \) over SB in the experiment of existential forgery by chosen-message attack is defined as \( \text{Adv}_{\mathsf{SB},\mathcal{F}}^{\text{euf-cma}}(λ) \triangleq \text{Pr}[\mathsf{Exprnt}_{\mathsf{SB},\mathcal{F}}^{\text{euf-cma}}(1^λ) \text{ returns WIN}] \).

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

B Pseudorandom Function Family [36]

A pseudorandom function family, \{PRF_k\}_k\in\mathsf{PRFkeys}(λ), is a function family in which each function \( \text{PRF}_k : \{0, 1\}^* \rightarrow \{0, 1\}^* \) is an efficiently-computable function that looks random to any polynomial-time distinguisher, where \( k \) is called a key and \( \text{PRFkeys}(λ) \) is called a key space. (See more details in, for example, the book [36].)

C Access Structure [25]

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

\[
\psi : U \rightarrow \mathcal{U}, \ \psi(X_i) \triangleq i.
\]
C.1 Access Tree

A monotone access formula \( f \) can be represented by a finite binary tree \( T_f \). Each inner node represents a boolean connective, \&-gate or \lor-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 \( T \), 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 Node(\( T \)), \( r(T) \), Leaf(\( T \)), iNode(\( T \)) and tNode(\( T \)), respectively. Then an attribute map \( \rho(\cdot) \) is defined as:

\[
\rho : \text{Leaf}(T) \rightarrow \mathcal{U}, \quad \rho(l) \overset{\text{def}}{=} (\text{the attribute } i \text{ that corresponds to } l \text{ through } \psi).
\]

If \( \rho \) is not injective, then we call the case multi-use of attributes.

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

D Attribute-Based Identification Scheme [2]

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

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

\textbf{ABID.KG}(\text{PK, MSK, } S) \rightarrow \text{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 \( \text{SK}_S \) corresponding to \( S \).

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

We demand correctness of \textbf{ABID} that, for any \( \lambda \) and if \( f(S) = 1 \), \( \text{Pr}[(\text{PK, MSK}) \leftarrow \textbf{ABID.Setup}(1^\lambda, \mathcal{U})] = 1 \).

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

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

\( \footnote{This limitation can be removed by adding negation attributes to } \mathcal{U} \text{ for each attribute in the original } \mathcal{U} \text{ though the size of the attribute universe } |\mathcal{U}| \text{ doubles.} \)
The following experiment $\text{Exprnt}_{\text{ABID},\mathcal{A}}^{\text{pa}}(1^\lambda,\mathcal{U})$ of an adversary $\mathcal{A}$ defines the game of passive attack on ABID.

\[
\text{Exprnt}_{\text{ABID},\mathcal{A}}^{\text{pa}}(1^\lambda,\mathcal{U}) : \\
(PK, MSK) \leftarrow \text{ABID.Setup}(1^\lambda,\mathcal{U}) \\
(f^*, st) \leftarrow \mathcal{A}^{KG(PK,MSK,\cdot),Transc(\mathcal{P}(PK,SK,\cdot),\mathcal{V}(PK,\cdot))}(PK,\mathcal{U}) \\
b \leftarrow \langle \mathcal{A}(st), \mathcal{V}(PK, f^*) \rangle \\
\text{If } b = 1 \text{ then Return Win else Return Lose}
\]

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

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

\[
\text{Adv}_{\text{ABID,}\mathcal{A}}^{\text{pa}}(\lambda,\mathcal{U}) \overset{\Delta}{=} \Pr[\text{Exprnt}_{\text{ABID,}\mathcal{A}}^{\text{pa}}(1^\lambda,\mathcal{U}) \text{ returns Win}].
\]

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

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

\[
\text{Exprnt}_{\text{ABID,}\mathcal{A}}^{\text{ca}}(1^\lambda,\mathcal{U}) : \\
(PK, MSK) \leftarrow \text{ABID.Setup}(1^\lambda,\mathcal{U}) \\
(f^*, st) \leftarrow \mathcal{A}^{KG(PK,MSK,\cdot),\mathcal{P}(PK,SK,\cdot)}(PK,\mathcal{U}) \\
b \leftarrow \langle \mathcal{A}(st), \mathcal{V}(PK, f^*) \rangle \\
\text{If } b = 1 \text{ then Return Win else Return Lose}
\]

In the experiment, $\mathcal{A}$ issues key-extraction queries to its key-generation oracle $KG$. Giving an attribute set $S_i$, $\mathcal{A}$ queries $KG(PK,MSK,\cdot)$ for the secret key $SK_{S_i}$. In addition, $\mathcal{A}$ invokes provers $\mathcal{P}(PK,SK,\cdot)$, $j = 1,\ldots,q_p$, 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 $\mathcal{P}(PK,SK_{S_j}, f_j)$ 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 provers. Two restrictions are imposed on $\mathcal{A}$ concerning $f^*$. For all key-extraction queries, $f^*(S_i) = 0$. For all interactions with each prover, $f^*(S_j) = 0$. The number of key-extraction queries and the number of invoked provers are at most $q_p$ and $q_p$ in total, respectively, which are bounded by a polynomial in $\lambda$.

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

\[
\text{Adv}_{\text{ABID,}\mathcal{A}}^{\text{ca}}(\lambda,\mathcal{U}) \overset{\Delta}{=} \Pr[\text{Exprnt}_{\text{ABID,}\mathcal{A}}^{\text{ca}}(1^\lambda,\mathcal{U}) \text{ returns Win}].
\]

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

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

\[
\text{Adv}_{\text{ABID,}\mathcal{A}}^{\text{pa}}(\lambda,\mathcal{U}) \leq \text{Adv}_{\text{ABID,}\mathcal{B}}^{\text{ca}}(\lambda,\mathcal{U}).
\]

### E Attribute-Based Signature Scheme [39, 41]

An attribute-based signature scheme, $\text{ABS}$, consists of four PPT algorithms [41]: $\text{ABS} = (\text{ABS.Setup, ABS.KG, ABS.Sign, ABS.Vrfy})$.

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

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

23
Informally speaking, an adversary $E.$ Chosen-Message Attack on ABS and Security Definition

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

E.2 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 5 (Attribute Privacy (Perfect Privacy [39, 41]))) ABS is called to have attribute privacy if, for all $(PK, MSK) \leftarrow \text{ABS.Setup}(1^\lambda, U)$, for all message $m$, for all attribute sets $S_1$ and $S_2$, for all signing keys $\text{SK}_{S_1} \leftarrow \text{ABS.KG}(PK, MSK, S_1)$ and $\text{SK}_{S_2} \leftarrow \text{ABS.KG}(PK, MSK, S_2)$ 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 $\text{ABS.Sign}(PK, \text{SK}_{S_1}, (m, f))$ and $\text{ABS.Sign}(PK, \text{SK}_{S_2}, (m, f))$ are identical.

F Instantiation Using Fiat-Shamir Signature-Bundle as Witness

In this section, we provide instantiations of our procedure $\Sigma_f$ and $\text{ABID}$ using the Fiat-Shamir signatures [20] as a witness. We give two instantiations in the RSA setting and the discrete-logarithm setting.
F.1 Our ABID in RSA Using FS Signature-Bundle as Witness

An RSA modulus of bit length $\lambda$ is denoted by $N$. An RSA exponent of odd prime is denoted by $e$.

**ABID Setup** takes as input $(1^\lambda, \mathcal{U})$. Let $R_s := \{(\beta, \alpha) \in \mathbb{Z}_N \times \mathbb{Z}_N; \beta = \alpha^e\}$. Then 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: $PK = ((N, e, \beta, \mathcal{U}, \mu), MSK = \alpha$.

**ABID KG** returns SK$_S$ with signatures, for $i \in S$, $s_i = (a_i = r_i^e, w_i = r_i \alpha^{e_i})$. Here we use a key $k$ obtained by $k \leftarrow Hash_a(\mu || \tau)$, put $m_i = \tau || i$, 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 $c_i \leftarrow Hash_a(\mu || m_i)$. $\Sigma^{sttgen}(\beta, a_i, c_i)$ is an algorithm that computes $x_i := a_i \beta^{e_i} \in \mathbb{Z}_N$.

The rest of protocol is executed according to $\Sigma_f$ on input $(x, w)$ and with the following setting.

\[
Cmt_l = r_t^e, Res_l = r_l (w_{\rho(l)})^{CHAI},
\]

Verification Equation : $Res_l^e = Cmt_l (x_{\rho(l)})^{CHAI}$.

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

A prime of bit length $\lambda$ is denoted by $p$. A multiplicative cyclic group of order $p$ is denoted by $\mathbb{G}_p$. We fix a base $g \in \mathbb{G}_p$, $(g) = \mathbb{G}_p$. 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 $R_s := \{(\beta, \alpha) \in \mathbb{G}_p \times \mathbb{Z}_p; \beta = g^\alpha\}$. Then 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: $PK = ((g, \beta, \mathcal{U}, \mu), MSK = \alpha$.

**ABID KG** returns SK$_S$ with signatures, for $i \in S$, $s_i = (a_i = g^{r_i}, w_i = r_i + c_i \alpha)$. Here we use a key $k$ obtained by $k \leftarrow Hash_a(\alpha || \tau)$, put $m_i = \tau || i$, 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 $c_i \leftarrow Hash_a(\alpha || m_i)$. $\Sigma^{sttgen}(\beta, a_i, c_i)$ is an algorithm that computes $x_i := a_i \beta^{c_i} \in \mathbb{G}_p$.

The rest of protocol is executed according to $\Sigma_f$ on input $(x, w)$ and with the following setting.

\[
Cmt_l = g^{r_t}, Res_l = r_l + CHAI w_{\rho(l)},
\]

Verification Equation : $g^{Res_l^e} = Cmt_l (x_{\rho(l)})^{CHAI}$.

G Instantiations Using Camenisch-Lysyanskaya Signature-Bundle as Witness

In this section, we provide another type of instantiations of our procedure $\Sigma_f$, ABID and ABTTS using the Camenisch-Lysyanskaya Signatures as a witness. We give two instantiations in the RSA setting [12] and the discrete-logarithm setting [48, 21, 40].

G.1 Our $\Sigma$-protocol $\Sigma_f$ in the Case of CL Signature-Bundle

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

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

(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 $\Sigma_f$ in the following way.

$\mathcal{P}(x, w, f)$ to prove the knowledge of those representations $(e_{\rho(l)}, s_{\rho(l)})$, $\mathcal{P}$ computes the first message, a commitment $(Cmt_l)_l$, as follows. Let $\bar{Z}$ be the exponent domain for the above expression. To do the computation honestly at a leaf $l$, $\mathcal{P}$ chooses $\eta_{e,l}, \eta_{s,l} \in R \bar{Z}$, and puts $Cmt_l := Z_{\eta_{e,l}, 1}^{e_{\rho(l)}} Z_{\eta_{s,l}, 2}^{s_{\rho(l)}}$. To simulate the honest computation at a leaf $l$, $\mathcal{P}$ chooses $\eta_{e,l}, \theta_{s,l} \in R \bar{Z}$, and in addition, the divided challenge strings $(\text{CHAI}_n)_n, \text{CHAI}_n \in \bar{Z}$, which are in accordance with our procedure $\Sigma_f$. Then $\mathcal{P}$ puts, for each leaf $l$, $\theta_{e,l} := \eta_{e,l} + \text{CHAI}_{e_{\rho(l)}}$, and $Cmt_l := Z_{\text{CHAI}_n, 1}^{e_{\rho(l)}} Z_{\text{CHAI}_n, 2}^{s_{\rho(l)}}$. $\mathcal{P}$ sends $(Cmt_l)_l$ to a verifier $\mathcal{V}$.

$\mathcal{V}(x, f)$, Receiving $(Cmt_l)_l$, $\mathcal{V}(x, f)$ chooses the second message: a challenge $\text{CHAI} \in R \bar{Z}$, uniformly at random, and sends $\text{CHAI}$ to $\mathcal{P}$.
\( \mathcal{P}(x, w, f) \). Receiving \( \mathcal{C}_A \), \( \mathcal{P} \) completes to compute the third message; that is, \( \mathcal{P} \) completes the division \((\mathcal{C}_A)_{\lambda} \) such that \( \mathcal{C}_A(T_f) = \mathcal{C}_A \), and a response \((\text{Res}_I) := (\theta_{e,i}, \theta_{s,i}) \) with \( \theta_{e,i} := \eta_{e,i} + \mathcal{C}_A e_{(l,i)} \), \( \theta_{s,i} := \eta_{s,i} + \mathcal{C}_A s_{(l,i)} \). \( \mathcal{P} \) sends \((\mathcal{C}_A)_{I} \) and \((\text{Res}_I) \) to \( V \). Receiving \((\mathcal{C}_A)_{I} \) and \((\text{Res}_I) \), \( V \) checks the integrity of the division \((\mathcal{C}_A)_{I} \). Then \( V \) verifies:

\[
\text{CMT}_I = \gamma Z_{\rho(l)}^{\mathcal{C}_A} Z_{\rho(l),1}^{\theta_{e,i}} Z_{\rho(l),2}^{\theta_{s,i}}, \quad l \in \text{Leaf}(T_f). \tag{6}
\]

According to the division rule of our procedure \( \Sigma_f \), the integrity of \((\mathcal{C}_A)_{I} \) can be checked as follows: From the leaves to the root, and at every inner node \( n \in \text{iNode}(T_f) \) and its two children \( chd_{1}, chd_{2} \):

- If \( n \) is an AND node (\( \wedge \)), then verify \( \mathcal{C}_A_{chd_{1}} = \gamma \mathcal{C}_A_{chd_{2}} \). If so, put \( \mathcal{C}_A_{n} := \mathcal{C}_A_{chd_{1}} \).
- Else if \( n \) is an OR node (\( \vee \)), then just put \( \mathcal{C}_A_{n} := \mathcal{C}_A_{chd_{1}} + \mathcal{C}_A_{chd_{2}} \).
- If \( n \) is the root node, then verify \( \mathcal{C}_A_{n} = \gamma \mathcal{C}_A \).
- Repeat until all \( n \in \text{iNode}(T_f) \) are verified.

The above procedure, \( \Sigma_f \), can be shown to possess the three requirements of \( \Sigma \)-protocol: completeness, special soundness and honest-verifier zero-knowledge.

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

#### Strong RSA Assumption [12]

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 \( 1^\lambda \) as \( \text{RSAmod} \). 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 \( a \equiv x^2 \mod N \) for some \( x \in \mathbb{Z}_{N}^* \). The strong RSA assumption [12] states that for any PPT \( A \), the following advantage is negligible in \( \lambda \):

\[
\text{Adv}_{\text{RSAmod}, g}(\lambda) := \Pr[\mathcal{N} \leftarrow \text{RSAmod}(1^\lambda), g \in R QR_N, (V, e) \leftarrow A(\mathcal{N}, g) : e > 1 \land Ve \equiv g \mod N].
\]

### CL Signature-Bundle in RSA

Our signature-bundle scheme \( \mathbb{SB} = (\mathbb{SB}_KG, \mathbb{SB}_Sign, \mathbb{SB}_Vrfy) \) is described as follows. Let \( l_M \) be a parameter. The message space \( \mathcal{M} \) consists of all binary strings of length \( l_M \). Let \( n = n(\lambda) \) denote the maximum number of messages made into a bundle, which is a polynomial in \( \lambda \).

#### \( \mathbb{SB}_KG(1^\lambda) \rightarrow (\text{PK}, \text{SK}) \)

Given \( 1^\lambda \), it chooses a special RSA modulus \( N = pq \) of length \( l_N = \lambda \), where \( p = 2p' + 1 \) and \( q = 2q' + 1 \) are safe primes. For \( i = 1 \) to \( n \), it chooses \( g_{i,0}, g_{i,1}, g_{i,2} \in R QR_N \). It puts \( \text{PK} := (N, (g_{i,0}, g_{i,1}, g_{i,2})_{i=1}^{n}) \) and \( \text{SK} \), and returns \((\text{PK}, \text{SK})\).

#### \( \mathbb{SB}_Sign(\text{PK}, \text{SK}, (m_i)_{i=1}^{n}) \rightarrow (\tau, \sigma_i)_{i=1}^{n} \)

Given \( \text{PK}, \text{SK} \) and messages \((m_i)_{i=1}^{n} \) each of which is of length \( l_M \), it chooses a prime \( e \) of length \( l_e = l_M + 2 \) at random. For \( i = 1 \) to \( n \), it chooses an integer \( s_i \) of length \( l_s = l_N + l_M + l \) at random, where \( l \) is a security parameter, and it computes the value \( A_i \):

\[
A_i := (g_{i,0} g_{i,1} g_{i,2})^{s_i}. \tag{7}
\]

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

#### \( \mathbb{SB}_Vrfy(\text{PK}, (m_i)_{i=1}^{n}, (\tau, (\sigma_i)_{i=1}^{n})) \rightarrow 1/0 \)

Given \( \text{PK}, (m_i)_{i=1}^{n} \) and a signature bundle \((\tau, (\sigma_i)_{i=1}^{n}) \), it verifies whether the following holds:

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

#### Theorem 8 (Unforgeability of Our \( \mathbb{SB} \))

Our signature-bundle scheme \( \mathbb{SB} \) 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 [12]. The difference only arises in the case that the simulation of the signature-bundle oracle needs precomputation.

Let \( F \) be a given PPT forger on our \( \mathbb{SB} \). We construct a PPT solver \( S \) of any instance \((N, g)\) of the Strong RSA problem. To describe three cases of \( F \)’s behavior, suppose that \( F \) issues at most \( q \) signature-bundle queries. Suppose that the signature-bundle oracle \( \text{SIG} \) replies the tags (that is, exponents) \( e_1, \ldots, e_q \) according to \( F \)’s queries, which are primes of length \( l_e \). Suppose that \( F \)’s forgery is \((m_i^*)_{i=1}^{n}, \tau^* = e^*, (\sigma_i^*)_{i=1}^{n} = (s_i^*, A_i^*)_{i=1}^{n} \). Let us distinguish three types of forgeries.
1. $e^*$ is relatively prime to any of $\{e_j\}$.
2. $e^*$ is not relatively prime to some of $\{e_j\}$, and $g_i^{m_i} g_{i,2}^{s_{i,j}} = g_i^{m_j} g_{i,2}^{s_{j,i}}$ 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 $\{e_j\}$, and $g_i^{m_i} g_{i,2}^{s_{i,j}} \neq g_i^{m_j} g_{i,2}^{s_{j,i}}$ for any $j$ s.t. $\gcd(e^*, e_j) \neq 1$ and any $i$.

By $F_1, F_2$ and $F_3$ let us denote the forger who runs $F$ 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, $S$ first guesses one of the three types at random (hence the advantage of $S$ reduces by the factor of $1/3$ here).

When $F$ is of Type 1 or Type 2, simulations of $F$’s signature-bundle oracle $S_{SSS\Sigma N}$ and the extraction of an answer of an instance $(N, g)$ go in the same way as the Camenisch-Lysyanskaya signature scheme [12].

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

For $j \neq j^*$, the simulation of $S_{SSS\Sigma N}$ for a query $(m_{j^*})$, issued by $F$ goes in the same way as in [12].

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

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

When $F$ returns a forgery $(m_{j^*})_{i=1}^n, (e^* \equiv e_{j^*}(s_{i,j})_{i=1}^n)$, the extraction of an answer of an instance goes in the same way as in [12]. Note that $e^* \equiv e_{j^*}$ holds with at least a non-negligible probability $1/q$. □

Our ABID in RSA Using CL-SB as Witness

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

**ABID KG**$(MSK, PK, S) \rightarrow SK_S$. Given PK, MSK and an attribute subset $S$, it chooses a prime $e$ of length $l_e$. For $i \in S$, it computes $a_i := \langle Hash_{\mu}(i) \rangle, s_i \in \mathbb{Z}$ of length $l_e$, $A_i := (g_{i,0} g_{i,1}^{s_i})^{1/2}$. It puts $SK_S := (e, (s_i, A_i)_{i \in S})$. $\mathcal{P}(SK_S, PK, f)$ and $\mathcal{V}(PK, f)$ execute $\Sigma_f$ with the following precomputation. For $i \in Att(f)$, $P$ chooses $r_i \in \mathbb{Z}$ of length $l_e$. If $i \in S$ then $s_i' := s_i + r_i$, $A_i' := A_i g_{i,2}^{r_i}$. Otherwise $s_i' \in \mathbb{Z}$ of length $l_e$, $A_i' \in \mathbb{Z}_N$. $P$ puts

$$Z_i := g_{i,0} g_{i,1}^{s_i'}, Z_{i,1} := A_i', Z_{i,2} := g_{i,2}.$$  

Then the statement for $\Sigma_f$ is $x := (x_i := (Z_i, Z_{i,1}, Z_{i,2})), and the witness is $w := (\tau := e, (w_i := s_{i,j}'))_{i \in Att(f)}$, where $i \in Att(f)$ for $x$ and $w$. $P$ sends the re-randomized values $(A_i')_{i \in Att(f)}$ to $\mathcal{V}$ for $\mathcal{V}$ to be able to compute the statement $x$.

After the above precomputation, $P$ and $V$ can execute $\Sigma_f$ on the relation $R_f$. In other words, $P$ and $V$ execute $ZKPoK[(\epsilon_{\rho(l)}, s_{\rho(l)})_l, l \in Leaf(T_f)] : equations$, for the language $L_f$, where the equations are:

$$Z_{\rho(l)} = Z_{\rho(l),1} Z_{\rho(l),2}^2, l \in Leaf(T_f).$$

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

$$CMT_l = Z_{\rho(l)}^{ChAI} Z_{\rho(l),1}^{\rho(l),1} Z_{\rho(l),2}^{\rho(l),2}, l \in Leaf(T_f).$$

$\mathcal{V}$ 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 $\Sigma$-protocol $\Sigma_f$ is a common single value.

**Proof.** All the answers of the oracles to queries of a PPT adversary $A$ on ABID can be perfectly simulated by the oracles of SB. As for the extraction of a signature bundle, we can do it under the condition that the extracted value $e$ is a common single value.

Note that Claim 1 is needed only as an intermediate result. That is, the assumption that the extracted value $e$ is a common single value is assured by the two-tier key-issuer, ABTTS.SK, 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 G.2, respectively. **ABTTS.SKG**, **ABTTS.Sign** and **ABTTS.Vrfy** are obtained along the design principle of two-tier signature schemes for the canonical identification schemes [8]. That is, on input MSK, PK, a primary secret key SK, and an access formula f, **ABTTS.SKG** first computes a statement x and a corresponding witness w. Then, on input (x, w), the prover P is executed in **ABTTS.SKG** to obtain the commitment (CMT), and the inner state st of P with the commitment is included in the secondary secret key; SSK := (w; (CMT) || st), SPK := (x, (CMT) || st). **ABTTS.Sign** and **ABTTS.Vrfy** run the remaining protocol of our **ABID** in the two-tier framework [8] as in Section 7. The signature is:

\[ \sigma := ((\text{CH}_A)_n, (\text{RES})_l). \]

**Security of Our ABTTS in RSA Using CL-SB**

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

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

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

**Proof.** The witness-indistinguishability of \( \Sigma_f \) assures the attribute privacy. □

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

**Strong Diffie-Hellman Assumption** [9] Let p denote a prime of bit length \( \lambda \). 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 \( g_1 \), \( \mathbb{G}_2 \) is generated by \( g_2 \) and \( \mathbb{G}_T \) is generated by \( e(g_1, g_2) \neq 1_{\mathbb{G}_T} \). We denote the probabilistic algorithm that generates such parameters params := \((p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)\) on input 1\(^\lambda\) as BlGrp. Let \( q \) denote a number that is less than a fixed polynomial in \( \lambda \). The strong Diffie-Hellman assumption [9] states that for any PPT \( A \), the following advantage is negligible in \( \lambda \): 

\[ \text{Adv}_{\text{BlGrp}, S}^{\text{sdh}}(\lambda) := \text{Pr}[\text{params} \leftarrow \text{BlGrp}(1\^{\lambda}), (u, e) \leftarrow A(\text{params}, (g, g^\alpha, \ldots, g^\alpha, h, h^\alpha)) : u^\alpha e = g]. \]

**CL Signature-Bundle in DL**

We propose a signature-bundle scheme in the discrete-logarithm setting by modifying the pairing-based CL signature scheme [48, 21, 40]. Our pairing-based signature-bundle scheme, \( SB = (SB.KG, SB.Sign, SB.Vrfy) \), is described as follows.

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

**SB.Sign(PK, 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 \( l_M \), it chooses \( e \in R \mathbb{Z}_p \). For \( i = 1 \) to \( n \), it chooses \( s_i \in R \mathbb{Z}_p \), and it computes the value \( A_i \): 

\[ A_i := (g_{i,0}g_{i,1}^{m_i}g_{i,2}^{s_i})^{1/e}. \]

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

**SB.Vrfy(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:

\[ e(A_i, h_0^\alpha) = e(g_{i,0}g_{i,1}^{m_i}g_{i,2}^{s_i}, h_0), \quad i = 1, \ldots, n. \]

**Theorem 11 (Unforgeability of Our SB)** Our signature-bundle scheme \( SB \) is existentially unforgeable against chosen-message attack under the Strong Diffie-Hellman assumption.
Proof. Everything can be done as in [40] except the following slight enhancement. S chooses \( q \) elements \( e_j \in \mathbb{Z}_p, j = 1, \ldots, q \), at random. Then \( S \) 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 \( n \), \( S \) chooses \( r_i, t_i, u_i, \alpha_i \leftarrow \mathbb{Z}_p \) and implicitly puts \( \alpha_i := \bar{\alpha}_i \alpha \), and puts \( g_{i,0} := g_{f,0}^{(\alpha_i + \epsilon^*)_{t_i} - u_i} \), \( g_{i,1} := g_{f,1}^{(\alpha_i + \epsilon^*)_{t_i - u_i}} = g_{f,1}^{(\alpha_i)_{t_i} - g_{-1}^{u_i} f_{j^*}(\alpha_i)}, s_{i^*} := u_i - r_{i^*}, A_i^* := g_{i,2}^{t_i}. \)

\[
A_{j^*}^{\alpha_i + \epsilon^*} = g_{2}^{(\alpha_i + \epsilon^*)_{t_i} - u_i + u_i} = g_{h_0}^{u_i} = g_{h_0}^{r_{i^*} m_{i^*} + s_{i^*}} = g_{h_0}^{m_{i^*} s_{i^*}}.
\]

This completes the simulation of the signature-bundle oracle \( \text{SBSIGN} \).

The extraction of the answer to an instance of the Strong Diffie-Hellman assumption can be done in the same way as [40] with division by \( \bar{\alpha}_i \). \( \square \)

Our ABID in DL Using CL-SB as Witness

**ABID.Setup**(\( \lambda, U \)) \( \rightarrow \) (MSK, PK). Given the security parameter \( \lambda \) and an attribute universe \( U \), it executes a group generator \( \text{BGrp}(\lambda) \) to get \( \{p, G_1, G_2, G_r, \{\cdot, \cdot\}\} \). For \( i \in U \), it chooses \( g_{i,0}, g_{i,1}, g_{i,2} \in G_1, h_0 \in G_2, \alpha \in \mathbb{Z}_p, h_1 := h_0^\alpha \) and a hash key \( \mu \in G_2 \) of a hash function \( \text{Hash}_2^{\mu} \) with the value in \( \mathbb{Z}_p \). It puts \( PK := ((g_{i,0}, g_{i,1}, g_{i,2})_{i \in U}, h_0, h_1, \mu, U) \) and \( MSK := \alpha \). It returns PK and MSK.

**ABID.KG** (MSK, PK, S) \( \rightarrow \) SKS. Given PK, MSK and an attribute subset \( S \), it chooses \( e \in \mathbb{Z}_p \). For \( i \in S \), it computes \( \alpha_i := \text{Hash}_2^{\mu}(i), s_i \in \mathbb{Z}_p, A_i := (g_{i,0} g_{i,1}^{s_i^{e_i}}) g_{i,2}^{s_i} \) and \( h_0 \in G_1 \). It puts \( SKS := (e, (s_i, A_i)_{i \in S}) \) and \( \mathcal{P}(SKS, PK, f) \) and \( \mathcal{V}(PK, f) \) execute \( \Sigma_f \) with the following precomputation. For \( \alpha \in \text{Att}(f) \), \( \mathcal{P} \) chooses \( r_i \in \mathbb{Z}_p \).

If \( i \in S \) then \( s_i := s_i + e r_i, A_i' := A_i g_{i,2}^{e r_i} \in G_1 \). Otherwise \( A_i' \in \mathbb{Z}_p, A_i' \in G_1 \). \( \mathcal{P} \) puts:

\[
Z_i := e(g_{i,0} g_{i,1}^{s_i}, h_0) e(A_i', h_1)^{-1}, Z_{i,1} := e(A_i', h_0), Z_{i,2} := e(g_{i,2}, h_0), Z_{i,3} := e(g_{i,2}, h_1).
\]

Then the statement for \( \Sigma_f \) is \( x := (x_i := (Z_{i,1}, Z_{i,2}, Z_{i,3}))_{i \in \text{Att}(f)} \) and the witness is \( w := (\tau := e, (w_i := s_i')_{i \in \text{Att}(f)}) \), where \( i \in \text{Att}(f) \) 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 \( \Sigma_f \) on the relation \( R_f \). In other words, \( \mathcal{P} \) and \( \mathcal{V} \) execute \( \text{ZKPoK}[(e_{\rho(l)}, s_{\rho(l)})_{l \in \text{Leaf}(T_f)} : \text{equations}] \), for the language \( L_f \), where the equations are:

\[
Z_{\rho(l)} = Z_{\rho(l),1} Z_{\rho(l),2}^{s_{\rho(l),3}} Z_{\rho(l),3}^{s_{\rho(l),3}} \text{, } l \in \text{Leaf}(T_f).
\]

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

\[
\text{CMT}_l = Z_{\rho(l)}^{\text{CMT}_l} Z_{\rho(l),1}^{\theta_{\rho(l),1}} Z_{\rho(l),2}^{\theta_{\rho(l),2}} Z_{\rho(l),3}^{\theta_{\rho(l),3}}, l \in \text{Leaf}(T_f).
\]

\( \mathcal{V} \) 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 \( \Sigma \)-protocol \( \Sigma_f \) is a common single value.

*Proof.* All the answers of the oracles to queries of a PPT adversary \( A \) on ABID can be perfectly simulated by using the oracles of SB. As for the extraction of a signature bundle, we can do it under the condition the extracted value \( e \) is a common single value.

Note that Claim 2 is needed only as an intermediate result. That is, the assumption that the extracted value \( e \) is a common single value is assured by the two-tier key-issuer, ABTT.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 G.2, respectively. ABTTS.SKG, ABTTS.Sign and ABTTS.Vrfy are obtained along the design principle of two-tier signature schemes for the canonical identification schemes [8]. 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 $P$ is executed in ABTTS.SKG to obtain the commitment $(CMT_I)_i$, and the inner state $st$ of $P$ with the commitment is included in the secondary secret key; SSK$_{S,f} := (w, (CMT_I)_i \parallel st)$, SPK$_f := (x, (CMT_I)_i)$. ABTTS.Sign and ABTTS.Vrfy run the remaining protocol of our ABID in the two-tier framework [8] as in Section 7. The signature is:

$$\sigma := ((:\text{Cha}_n)_i, (:\text{Res}_i)_i).$$

Security of Our ABTTS in DL Using CL-SB

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

Proof. According to the same discussion in Bellare et al. [8] as well as Theorem 11 and Claim 2, we deduce the claim. □

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

Proof. The witness-indistinguishability of $\Sigma_f$ assures the attribute privacy. □
Using a boolean formula on attributes of ratees, signatures with predications on ratees are generated, which we call expressive ratings. Public linkability of ABS is effectively used to prohibit double ratings. Also, employing an ABS scheme of the Fiat-Shamir type, we construct a concrete efficient expressive rating scheme. Keywords. Reputation Rating Attribute Predication Digital signature. This is a preview of subscription content, log in to check access. Notes. Acknowledgements. Anada, H., Arita, S., Sakurai, K.: Proof of knowledge on monotone predicates and its application to attribute-based identifications and signatures. Cryptology ePrint Archive, Report 2016/483 (2016). http://eprint.iacr.org/2016/483. 3.