Attribute-Based Signatures without Pairings by the Fiat-Shamir Transformation*

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

[1] Institute of Systems, Information Technologies and Nanotechnologies, Japan (ISIT),
anada@isit.or.jp
[2] Institute of Information Security, Japan (IISEC),
arita@iisec.ac.jp
[3] Department of Informatics, Graduate School of ISEE Faculty, Kyushu University,
sakurai@csce.kyushu-u.ac.jp

July 22, 2014

Abstract. We propose the first practical attribute-based signature (ABS) scheme with attribute privacy without pairings in the random oracle model. Our strategy is in the Fiat-Shamir paradigm; we first provide a concrete construction of a Σ-protocol of *boolean proof*, which is a generalization of the well-known Σ-protocol of OR-proof, so that it can treat any monotone boolean formula instead of a single OR-gate. Then, we apply the Fiat-Shamir transformation to our Σ-protocol of boolean proof and obtain a non-interactive witness-indistinguishable proof of knowledge system (NIWIPoK) which has a knowledge extractor in the random oracle model. Finally, by combining our NIWIPoK with a credential bundle scheme of the Fiat-Shamir signature, we obtain an attribute-based signature scheme (ABS) which possesses the property of attribute privacy. The series of constructions are obtained from a given Σ-protocol and can be attained without pairings.

Keywords: access control, attribute, signature, proof system, boolean formula, identification, Fiat-Shamir transformation

1 Introduction

Since the invention of digital signature scheme by Diffie and Hellman in 1976, there have been significant evolutions in the area and many functional variants have been proposed. A distinguished variant is *attribute-based signature* (ABS), which has been developed since 2008 [16, 18, 19]. In ABS scheme, a message is associated with an access policy that limits signers by their attributes which the signers possess. The access policy is described with a boolean formula over those attributes. A signer with a set of authorized attributes can make a legitimate signature on the message only when his attributes satisfy the access policy. Then, a verifier can check whether the pair of the message and the signature is valid or not concerning the access policy. If the access policy limits accepting attributes to a single identity, ABS coincides with identity-based signature (IBS) [23]. In that sense, ABS is a natural extension of IBS.

One remarkable property of ABS is *attribute privacy* for signers. In the case that a verifier knows nothing about prover's attributes except that the prover's attributes satisfy the access policy of the

* The preliminary version of this paper appeared in *Proceedings in the 2nd ACM ASIA Public-Key Cryptography Workshop - ASIAPKC 2014*, pp. 49-58, Keita Emura, Goichiro Hanaoka and Yunlei Zhao eds., under the title of *Attribute-Based Signatures without Pairings via the Fiat-Shamir Paradigm*. This is the full version and more than a half has been rewritten.

** The third author is partially supported by Japan Society for the Promotion of Science, Grants-in-Aid for Scientific Research; Research Project Number:25540004.

signed message, the ABS scheme is called to have attribute privacy. Some ABS scheme ([16]) did not care attribute privacy but it is considered desirable for ABS to possess it.

We will use in this paper the Fiat-Shamir transformation [13] to obtain such an ABS scheme. The Fiat-Shamir transformation is now an established technique in interactive proof theory and cryptography. It transforms a public-coin 3-move interactive proof system into a non-interactive zero-knowledge proof system (NIZK) by replacing the second message with the hash value of the first message. If the initial interactive proof system is an identification scheme, then the resulting NIZK is a digital signature scheme.

One of such interactive proof systems is the Σ-protocol [9, 11]. It allows a prover, in a 3-move, to convince a verifier that the prover knows a witness of a statement. That is, the Σ-protocol is a public-coin 3-move interactive proof of knowledge system. When we apply the Fiat-Shamir transformation to the Σ-protocol, we obtain a zero-knowledge proof of knowledge system (NIZKPoK) whose extractor is in the random oracle model. An extended notion of the original Σ-protocol is the OR-proof [10, 11]. It allows a prover to convince a verifier that the prover knows one (or both) of witnesses of two statement *hiding which witness he knows*.

In this paper, we will provide an attribute-based signature scheme via the Fiat-Shamir paradigm. First, we look into the protocol introduced by Cramer, Damgård and Schoenmakers [10][4] in CRYPTO '94. Given a boolean formula f, the protocol provides a witness hiding protocol [12]. That is, in f, a value '1' (TRUE) is substituted for boolean variables for which a prover has the corresponding witnesses; a value '0' (FALSE) is substituted for the remaining boolean variables. Then the prover in the protocol can show to a verifier that he knows a set of witnesses that satisfies the boolean formula f, but even a cheating verifier will not learn enough to be able to compute the prover's witnesses. We call the notion a *boolean proof*. In [10], the OR-proof is deduced as a special case of the boolean proof. For the boolean proof we will provide a concrete protocol, $\mathbf{\Sigma}_f$, from a given Σ-protocol $\mathbf{\Sigma}$ and provide a proof that $\mathbf{\Sigma}_f$ is also a Σ-protocol.

Then we will proceed into the Fiat-Shamir paradigm. We apply the Fiat-Shamir transformation to our Σ-protocol $\mathbf{\Sigma}_f$, which is a boolean proof, and obtain a NIZKPoK system and hence a non-interactive witness-indistinguishable proof of knowledge system (NIWIPoK) which has a knowledge extractor in the random oracle model. Finally, by combining our NIWIPoK with a credential bundle scheme of the Fiat-Shamir signature, we obtain an attribute-based signature scheme (ABS) which possesses the property of attribute privacy. The series of constructions are obtained from a given Σ-protocol. Hence we find that our ABS scheme can be constructed without pairing maps ([8]) (pairings, for short) with the expense that its security proof is in the random oracle model. As a result, our ABS scheme can be implemented with more efficiency than previous ABS schemes.

1.1 Our Construction Idea of a Concrete Boolean Proof

To construct the above concrete boolean proof from a given Σ-protocol and a boolean formula f, we will look into the technique employed in the OR-proof [10] and expand it so that it can treat any monotone (that is, no negation) boolean formula, as follows.

First express the boolean formula f as a binary tree $\mathcal{T}_f$. That is, we put leaf nodes each of which corresponds to each term in f. We connect two leaf nodes by an $\wedge$-node or an $\vee$-node according to an AND-gate or an OR-gate that lies between corresponding terms. Then we connect the resulting nodes by an $\wedge$-node or an $\vee$-node in the same way, until we reach to the only $\wedge$-node or $\vee$-node that is called the root node. A verification equation of the Σ-protocol $\mathbf{\Sigma}$ is assigned to each leaf node. Suppose that a challenge string CHA of $\mathbf{\Sigma}$ is given. Assign the string CHA to the root node. If the root node is an AND-gate, assign the same string CHA to two children. Else if the root node is an

[4] In the preliminary version [3] of this e-print the authors could not refer to this previous work.

OR-gate, divide $\textsc{Cha}$ into two random strings $\textsc{Cha}_{\mathrm{L}}$ and $\textsc{Cha}_{\mathrm{R}}$ which satisfy $\textsc{Cha} = \textsc{Cha}_{\mathrm{L}} \oplus \textsc{Cha}_{\mathrm{R}}$, and assign $\textsc{Cha}_{\mathrm{L}}$ and $\textsc{Cha}_{\mathrm{R}}$ to the left and right children, respectively. Here $\oplus$ means a bitwise exclusive-OR operation. Then continue to apply this rule at each height, step by step, until we reach to each leaf node. Then, basically, the OR-proof technique assures that we can either honestly execute the Σ-protocol $\mathbf{\Sigma}$ or execute $\mathbf{\Sigma}$ in a simulated way. Only when a set of witnesses each of which gives the value '1' to satisfies the binary tree the above procedure succeeds in satisfying verification equations for all leaf nodes.

1.2 Our Contributions

Our first contribution is to provide a concrete protocol of boolean proof, which is comparable with the original protocol [10]. That is, given a (monotone) boolean formula f and a Σ-protocol $\mathbf{\Sigma}$, we construct a protocol $\mathbf{\Sigma}_f$ in a recursive form that is suitable for implementation. Then we show that $\mathbf{\Sigma}_f$ is certainly a Σ-protocol. Especially we show that $\mathbf{\Sigma}_f$ is a protocol to prove knowledge of a witness set that satisfies f.

And our second contribution is to provide an attribute-based signature scheme (ABS) *without pairings*. More precisely, As a Σ-protocol $\mathbf{\Sigma}$ can be instantiated without pairings (for example, the case of Schnorr scheme [22, 6]), our construction can produce a practical boolean proof $\mathbf{\Sigma}_f$ that is a Σ-protocol, and hence, ABS without pairings. We note here that our ABS attains attribute privacy. We also note, as negative points, that security of our ABS can be proved in the random oracle model. Moreover, the reduction of advantages of adversaries is loose.

1.3 Related Work and Technical Comparison

The Σ-protocol is formalized in an abstract, present form by [9]. The Σ-protocol of OR-proof had been developed in [10] (see also [11]). The Σ-protocol of boolean proof had also been given in [10], but not been given as a concrete form.

At a high level, our ABS scheme is captured as a combination of a credential bundle scheme [19] with our NIWIPoK, whose extractor is in the random oracle model. This construction can be compared with the generic construction of the ABS scheme by Maji et al. [19]. They started with a credential bundle scheme. Then they employed a NIWIPoK in the standard model by Groth and Sahai [15] to prove the knowledge of a credential bundle which satisfies a given access formula. Hence the difference between our construction and their construction becomes apparent. In our ABS, a knowledge extractor is constructed in the random oracle model by using rewinding technique. In contrast, in their ABS, a knowledge extractor is constructed in the standard model.

From a practical point of view, it is notable that attribute-based cryptographic primitives originated in [21, 14] and developed so far employ linear secret sharing schemes (LSSS) ([4]) and pairings ([8]) in non-interactive setting (except lattice constructions which are less efficient). In contrast, our approach for ABS is via the interactive technique.

1.4 Efficiency Comparison

The most efficient, previously known ABS scheme is the one by Okamoto and Takashima (OT11) [20] that is in the standard model. We compare efficiency of our `ABS` with their scheme in the length of a signature as well as underlying assumption, in the discrete-logarithm setting as follows.

A prime of bit length λ (the security parameter) is denoted by p and we fix a cyclic group $\mathbb{G}_p$ of order p. We assume that an element of $\mathbb{G}_p$ is represented by 2λ bits. Let l denote the number of leaf nodes in the binary tree $\mathcal{T}_f$. DLIN and CR hash mean the Decisional Linear assumption for pairing group [20] and the collision resistance of an employed hash function, respectively.

Then the lengths of a signature of the scheme OT11 and our ABS are as in the Table 1. OT11 scheme has advantages in the security-proof model and boolean formula, whereas our ABS realizes shorter length of signature.

Table 1. Efficiency Comparison.

Scheme	OT11[20]	Our ABS
Len. of Sig. (bit)	$(2\lambda)(9l+11)$	$(2\lambda)l+(\lambda)(4l-1)$
Sec.-Proof Model	Standard	Random Oracle
Assumption	DLIN $\wedge$ CR hash	DLog $\wedge$ CR hash
Boolean Formula	Non-monotone	Monotone
Adaptive Target	Yes	Yes
Attribute Privacy	Yes	Yes

1.5 Organization of this Paper

In Section 2, we prepare for the required tools and notions. In Section 3, we summarize a definition of a boolean proof. In Section 4, we describe the concrete construction of our boolean proof, $\mathbf{\Sigma}_f$. In Section 5, we apply the Fiat-Shamir transformation to our $\mathbf{\Sigma}_f$ to obtain our NIWIPoK system In Section 6, by combining the NIWIPoK system FS($\mathbf{\Sigma}_f$) with a credential bundle scheme of the Fiat-Shamir signature FS($\mathbf{\Sigma}$), we obtain our ABS scheme. In Section 7, we conclude our work in this paper. In Appendix A, we show how concretely our ABS is instantiated in discrete-logarithm setting and RSA setting. In Appendix B, we summarize the notion and definition of attribute-based identification (ABID) scheme. In Appendix C, by combining our $\mathbf{\Sigma}_f$ with a credential bundle scheme of the Fiat-Shamir signature FS($\mathbf{\Sigma}$), we obtain our ABID scheme.

2 Preliminaries

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

2.1 Language and Proof of Knowledge [5, 10, 11]

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

polynomial-time algorithm for deciding membership in R. Then a *language* for a NP relation R is defined as:

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

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

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

Proof of Knowledge A *proof of knowledge system* (PoK for short) $\Pi = (\mathcal{P}, \mathcal{V})$ for a language L_R is a protocol between interactive PPT algorithms $\mathcal{P}$[5] and $\mathcal{V}$ on initial input $(x,w) \in R$ for $\mathcal{P}$ and x for $\mathcal{V}$, where $\mathcal{V}$ outputs 1 (accept) or 0 (reject) after finite rounds of interaction. $\mathcal{P}$ is called a prover and $\mathcal{V}$ is called a verifier. Π must possess the following two properties.
Completeness. For any statement $x \in L_R$ and for any witness w such that $(x,w) \in R$, $\mathcal{P}$ with the witness w can make $\mathcal{V}$ accept for the statement x with probability 1:

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

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

$$\frac{|x|^c}{p(x) - \kappa(x)}.$$

Non-interactive Witness-Indistinguishable Proof of Knowledge [15] A *non-interactive witness-indistinguishable proof of knowledge system* (NIWIPoK, for short) $\Pi = (K, \mathcal{P}, \mathcal{V})$ for a language L_R is a protocol, where a PPT algorithm K, on input λ, outputs crs called a *common reference string*; a PPT algorithm $\mathcal{P}$, on input $(x,w) \in R$, outputs π called a *proof*; and a PPT algorithm V, on input (x,π), outputs 1 (accept) or 0 (reject). Π must possess the following three properties.
Completeness. For any statement $x \in L_R$ and for any witness w such that $(x,w) \in R$, $\mathcal{P}$ with the witness w can make $\mathcal{V}$ accept on the statement x with probability 1:

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

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

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

[5] In the general theory of interactive proof a prover is assumed to have unbounded computational power, but in this paper we assume that a prover is PPT.

and for any unbounded algorithm B, we have

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

where $\big((x, w_0) \in R \wedge (x, w_1) \in R\big) \vee \big((x, w_0) \notin R \wedge (x, w_1) \notin R\big)$ holds.

Σ-protocol [9, 11] A *Σ-protocol* on a relation R is a public coin 3-move protocol between interactive PPT algorithms $\mathcal{P}$ and $\mathcal{V}$ on initial input $(x, w) \in R$ for $\mathcal{P}$ and x for $\mathcal{V}$. $\mathcal{P}$ sends the first message called a commitment CMT, then $\mathcal{V}$ sends a random bit string called a challenge CHA, and $\mathcal{P}$ answers with a third message called a response RES. Then $\mathcal{V}$ applies a decision test on $(x, \mathrm{CMT}, \mathrm{CHA}, \mathrm{RES})$ to return accept (1) or reject (0). If $\mathcal{V}$ accepts, then the triple $(\mathrm{CMT}, \mathrm{CHA}, \mathrm{RES})$ is said to be an *accepting conversation*. CHA is chosen uniformly at random from $\mathrm{CHASP}(\lambda) := \{1, 0\}^{l(\lambda)}$ with $l(\cdot)$ being a super-log function.

This protocol is written by a PPT algorithm $\mathbf{\Sigma}$ as follows. $\mathrm{CMT} \leftarrow \mathbf{\Sigma}^1(x, w)$: the process of selecting the first message CMT according to the protocol $\mathbf{\Sigma}$ on input $(x, w) \in R$. Similarly we denote $\mathrm{CHA} \leftarrow \mathbf{\Sigma}^2(\lambda)$, $\mathrm{RES} \leftarrow \mathbf{\Sigma}^3(x, w, \mathrm{CMT}, \mathrm{CHA})$ and $b \leftarrow \mathbf{\Sigma}^{\mathrm{vrfy}}(x, \mathrm{CMT}, \mathrm{CHA}, \mathrm{RES})$.

Σ-protocol must possess the following three properties.

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

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

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

Any Σ-protocol is known to be a proof of knowledge system ([11]).

We will use in this paper a property called *unique answer property* [7] that for legitimately produced commitment CMT and challenge CHA, there is one and only one response w' that is accepted by a verifier. Known Σ-protocols such as the Schnorr protocol and the Guillou-Quisquater protocol [22, 6] possess the unique answer property. Further, we will assume that for such a unique response w' there is one and only one statement $x' \in L_R$ (that is, $(x', w') \in R$) and both a prover and a verifier can compute, in polynomial-time, such an x' from x, CMT and CHA. We denote the algorithm as $\mathbf{\Sigma}^{\mathrm{stmtRES}}$:

$$\begin{aligned}&x' \leftarrow \mathbf{\Sigma}^{\mathrm{stmtRES}}(x, \mathrm{CMT}, \mathrm{CHA})\\ &\text{s.t. } {}^{\exists 1}w', (\mathrm{CMT}, \mathrm{CHA}, w') \text{ is an accepting conversation, } (x', w') \in R.\end{aligned}$$

Known Σ-protocols [22, 6] also possess this *unique statement property* (see Appendix A).

The OR-proof [11] Suppose that a Σ-protocol $\mathbf{\Sigma}$ on a relation R is given. Consider the following relation for a boolean formula $f(X_1, X_2) = X_1 \vee X_2$.

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

Then we construct a new protocol, $\mathbf{\Sigma}_{\mathrm{OR}}$, on a relation R_{OR} as follows. For an explanation, suppose $(x_0, w_0) \in R$ holds. $\mathcal{P}$ computes $\mathrm{CMT}_0 \leftarrow \mathbf{\Sigma}^1(x_0, w), \mathrm{CHA}_1 \leftarrow \mathbf{\Sigma}^2(\lambda)$, $(\mathrm{CMT}_1, \mathrm{RES}_1) \leftarrow \mathbf{\Sigma}^{\mathrm{sim}}(x_1, \mathrm{CHA}_1)$ and sends $(\mathrm{CMT}_0, \mathrm{CMT}_1)$ to V. Then $\mathcal{V}$ sends $\mathrm{CHA} \leftarrow \mathbf{\Sigma}^2(\lambda)$ to P. Then, $\mathcal{P}$ computes $\mathrm{CHA}_0 := \mathrm{CHA} \oplus \mathrm{CHA}_1, \mathrm{RES}_0 \leftarrow \mathbf{\Sigma}^3(x_0, w_0, \mathrm{CMT}_0, \mathrm{CHA}_0)$ answers to $\mathcal{V}$ with $(\mathrm{CHA}_0, \mathrm{CHA}_1)$ and $(\mathrm{RES}_0, \mathrm{RES}_1)$. Here $\oplus$ denotes a bitwise exclusive-OR operation. Then both $(\mathrm{CMT}_0, \mathrm{CHA}_0, \mathrm{RES}_0)$ and $(\mathrm{CMT}_1, \mathrm{CHA}_1, \mathrm{RES}_1)$ are accepting conversations and have the same distribution as real accepting conversations. This protocol $\mathbf{\Sigma}_{\mathrm{OR}}$ can also be proved to be a Σ-protocol, and is called the *OR-proof*.

The Fiat-Shamir transformation [1] Employing a cryptographic hash function with collision resistance, $\mathit{Hash}_\mu(\cdot) : \{1,0\}^* \to \{1,0\}^{l(\lambda)}$, a Σ-protocol $\mathbf{\Sigma}$ can be transformed into a non-interactive zero-knowledge proof of knowledge system (NIZKPoK) and hence a non-interactive witness-indistinguishable proof of knowledge system (NIWIPoK), with its knowledge extractor is in the random oracle model. If a Σ-protocol $\mathbf{\Sigma}$ is an identification scheme, then the resulting scheme is a digital signature scheme. The transformation is described as follows. (Here, a message m is taken as an empty string in the case of a NIWIPoK.) Given a message $m \in \{1,0\}^*$, execute: $a \leftarrow \mathbf{\Sigma}^1(x, w)$, $c \leftarrow \mathit{Hash}_\mu(a \parallel m)$, $z \leftarrow \mathbf{\Sigma}^3(x, w, a, c)$. Then $\sigma := (a, z)$ is a signature on m. We denote the above signing algorithm as $\mathrm{FS}(\mathbf{\Sigma})^{\mathrm{sign}}(x, w, m) \to (a, z) =: \sigma$. The verification algorithm $\mathrm{FS}(\mathbf{\Sigma})^{\mathrm{vrfy}}(x, m, \sigma)$ is given as follows: $c \leftarrow \mathit{Hash}_\mu(a \parallel m)$, Return $b \leftarrow \mathbf{\Sigma}^{\mathrm{vrfy}}(x, a, c, z)$.

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

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

2.2 Access Structure [14]

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 unless we state the large case explicitly.

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

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

For $f(X_{i_1}, \ldots, X_{i_a})$, we denote the set of indices (that is, attributes) $\{i_1, \ldots, i_a\}$ by $\mathrm{Att}(f)$. Hereafter we use the symbol i_j to mean the index of a boolean variable that is the j-th argument of f. We note the arity of f as $\mathrm{arity}(f)$.

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

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

Under this definition, a boolean formula f can be seen as a map: $f : 2^{\mathcal{U}} \to \{1, 0\}$. We call a boolean formula f with this map an *access formula* over $\mathcal{U}$.

In this paper, we do *not* consider NOT-gate ($\neg$). In other words, we only consider *monotone* access formulas[6].

Access Tree We consider in this paper a finite binary tree $\mathcal{T}$, that is, a tree that has finite number of nodes and each non-leaf node has two branches. For a tree $\mathcal{T}$, let $\mathrm{Nd}(\mathcal{T})$, $\mathrm{rt}(\mathcal{T})$, $\mathrm{Lf}(\mathcal{T})$, $\mathrm{iNd}(\mathcal{T})$ and $\mathrm{tNd}(\mathcal{T})$ denote the set of all nodes, the root node, the set of all leaf nodes, the set of all inner nodes (that is, all nodes excluding leaf nodes) and the set of all tree-nodes (that is, all nodes excluding the root node) in $\mathcal{T}$, respectively. An access formula f can be represented by a finite binary tree $\mathcal{T}_f$. Each inner node represents an operator, $\wedge$-gate or $\vee$-gate, in f. Each leaf node corresponds to a term X_i (not a variable X_i) in f in one-to-one way. An attribute map $i(\cdot)$ is defined as:

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

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

2.3 Credential Bundle Scheme [19]

Credential bundle is an extended notion of digital signature. Suppose that we are given a digital signature scheme (KG, Sign, Vrfy). We construct a credential bundle scheme (CB.KG, CB.Sign, CB.Vrfy) as follows. CB.KG takes as input the security parameter λ. CB.KG outputs a verification key PK and a signing key SK. Then CB.Sign takes as input PK, SK, a string τ called a *tag* and a set of messages $(m_i)_{i=1,\ldots,n}$. The tag τ can be chosen as a publicly known string unique for each user such as an e-mail address [19]. CB.Sign executes Sign on each *tagged message* $(\tau \parallel m_i), i = 1, \ldots, n$ and outputs signatures $(\sigma_i)_{i=1,\ldots,n}$. CB.Vrfy takes as input PK, τ and $(m_i, \sigma_i)_{i=1,\ldots,n}$. CB.Vrfy executes Vrfy on each tagged message and signature $((\tau \parallel m_i), \sigma_i), i = 1, \ldots, n$. CB.Vrfy returns 1 (valid) if and only if Vrfy returns 1 for all i, $i = 1, \ldots, n$.

2.4 Pseudorandom Function Family [17]

A pseudorandom function family, $\{PRF_k\}_{k \in PRFkeysp(\lambda)}$, is a function family in which each function $PRF_k : \{1, 0\}^* \to \{1, 0\}^*$ is an efficiently-computable function that looks random to any polynomial-time distinguisher, where k is called a key and $PRFkeysp(\lambda)$ is called a key space. (See more details in [17].)

2.5 Attribute-Based Signature Scheme [20]

An attribute-based signature scheme, `ABS`, consists of four PPT algorithms: (Setup, KG, Sign, Vrfy).
Setup$(\lambda, \mathcal{U}) \to (\mathbf{PK}, \mathbf{MSK})$**. Setup** takes as input the security parameter λ and an attribute universe $\mathcal{U}$. It outputs a public key PK and a master secret key MSK.
KG$(\mathbf{PK}, \mathbf{MSK}, \tau, S) \to \mathbf{SK}_S$**.** A key-generation algorithm **KG** takes as input the public key PK, the master secret key MSK, a tag τ, and an attribute set $S \subset \mathcal{U}$. It outputs a signing key SK_S corresponding to S.

[6] This limitation can be removed by adding *negation attributes* to $\mathcal{U}$ for each attribute in the original $\mathcal{U}$ (but as a result, the size $|\mathcal{U}|$ doubles).

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

Vrfy$(\mathbf{PK}, (m, f), \sigma)$**.** A verification algorithm **Vrfy** takes as input a public key PK, a pair (m, f) of a message and an access formula, and a signature σ. It outputs a decision 1 or 0. When it is 1, we say that $((m, f), \sigma)$ is *valid*. When it is 0, we say that $((m, f), \sigma)$ is *invalid*. We demand correctness of ABS that for any λ, any τ, and if $f(S) = 1$, then $\Pr[(\mathrm{PK}, \mathrm{MSK}) \leftarrow \mathbf{Setup}(\lambda, \mathcal{U}), \mathrm{SK}_S \leftarrow \mathbf{KG}(\mathrm{PK}, \mathrm{MSK}, \tau, S), \sigma \leftarrow \mathbf{Sign}(\mathrm{PK}, \mathrm{SK}_S, (m, f)), b \leftarrow \mathbf{Vrfy}(\mathrm{PK}, (m, f), \sigma) : b = 1] = 1$.

Chosen-Message Attack on ABS and Security An adversary $\mathcal{F}$'s objective is to make an *existential forgery*. $\mathcal{F}$ tries to make a forgery $((m^*, f^*), \sigma^*)$ of a message, a target access policy and a signature. The following experiment $\mathbf{Exprmt}^{\text{euf-cma}}_{\mathcal{F},\mathtt{ABS}}(\lambda, \mathcal{U})$ of a forger $\mathcal{F}$ defines the *game of existential unforgeability against chosen-message attack* on ABS.

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

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

The access formula f^* declared by $\mathcal{F}$ is called a *target access formula*. Here we consider the *adaptive* target in the sense that $\mathcal{F}$ is allowed to choose f^* after seeing PK and issuing some key-extraction queries and signing queries. Two restrictions are imposed on $\mathcal{F}$ concerning f^*. In key-extraction queries, S_i that satisfies $f^*(S_i) = 1$ was never queried. In signing queries, (m^*, f^*) was never queried. The number of key-extraction queries and the number of signing queries are at most q_k and q_s in total, respectively, which are bounded by a polynomial in λ.

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

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

ABS is called *secure in the game of existential unforgeability against chosen-message attacks* if, for any PPT $\mathcal{F}$, $\mathbf{Adv}^{\text{euf-cma}}_{\mathcal{F},\mathtt{ABS}}(\lambda)$ is negligible in λ.

Attribute Privacy of ABS Consider the following experiment $\mathbf{Exprmt}^{\text{att-prv}}_{\mathcal{A},\mathtt{ABS}}(\lambda,\mathcal{U})$.

$$
\begin{aligned}
&\mathbf{Exprmt}^{\text{att-prv}}_{\mathcal{A},\mathtt{ABS}}(\lambda,\mathcal{U}):\\
&(\mathrm{PK},\mathrm{MSK})\leftarrow\mathbf{Setup}(\lambda,\mathcal{U}),(\tau^*,S_0,S_1,f^*)\leftarrow\mathcal{A}(\mathrm{PK})\\
&\text{ s.t. } (f^*(S_0)=f^*(S_1)=1)\vee((f^*(S_0)=f^*(S_1)=0)\\
&\mathrm{SK}_{S_0}\leftarrow\mathbf{KG}(\mathrm{PK},\mathrm{MSK},\tau^*,S_0),\mathrm{SK}_{S_1}\leftarrow\mathbf{KG}(\mathrm{PK},\mathrm{MSK},\tau^*,S_1)\\
&b\leftarrow\{1,0\},\hat{b}\leftarrow\mathcal{A}^{\mathcal{SIGN}(\mathrm{PK},\mathrm{SK}_{S_b},\cdot,\cdot)}(\mathrm{PK},\mathrm{SK}_{S_0},\mathrm{SK}_{S_1})\\
&\text{If } b=\hat{b} \text{ Return } \textsc{Win} \text{ else Return } \textsc{Lose}
\end{aligned}
$$

We say that ABS has *attribute privacy* if, for any unbounded algorithm $\mathcal{A}$, the following advantage of $\mathcal{A}$ is 0.

$$\mathbf{Adv}^{\text{att-prv}}_{\mathcal{A},\mathtt{ABS}}(\lambda)\stackrel{\text{def}}{=}|\Pr[\mathbf{Exprmt}^{\text{att-prv}}_{\mathcal{A},\mathtt{ABS}}(\lambda,\mathcal{U})\text{ returns }\textsc{Win}]-1/2|.$$

3 Definition of Boolean Proof

In this section, we revisit the notion of a public coin interactive proof of knowledge system for the language L_{R_f} introduced by Cramer, Damgård and Schoenmakers [10], which we call a *boolean proof*. Then we restate the definitions for the sake of clarity.

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

Definition 1 ([10], Rewritten Form) *Given R and f, we define a new relation R_f by:*

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

where $R(\cdot,\cdot)$ denotes the relation-function associated with R, which returns 1 or 0.

Note that, if R is a NP relation, then R_f is a NP relation because we have assumed that a, the arity of f, is bounded by a polynomial in λ. Note also that R_f is a generalization of the relation R_{OR}.

Then the corresponding language for the NP relation R_f is given as follows.

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

Finally, we achieves the following definition.

Definition 2 *A boolean proof is a public coin interactive proof of knowledge system for the language L_{R_f}.*

We note that a knowledge extractor must be constructed.

4 Our Construction of Boolean Proof

In this section, we construct a proof system $\mathbf{\Sigma}_f$ from a given Σ-protocol $\mathbf{\Sigma}$ and a boolean formula f. Our $\mathbf{\Sigma}_f$ is proved to be a Σ-protocol. Also, our $\mathbf{\Sigma}_f$ is proved to be a boolean proof for L_{R_f}.

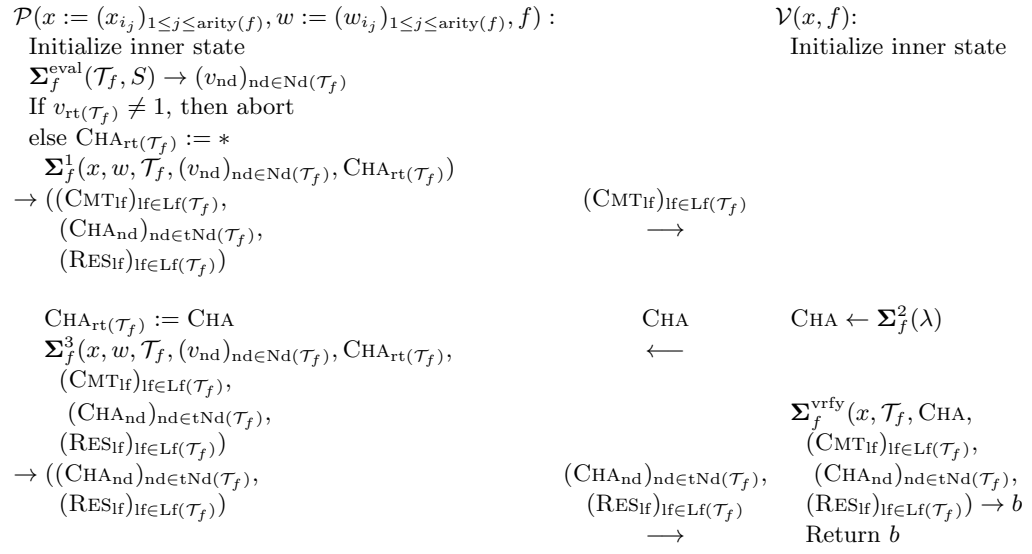

Fig. 1. Our boolean proof $\mathbf{\Sigma}_f$ for the language L_{R_f}.

Fig. 1 shows our protocol $\mathbf{\Sigma}_f$. Basically $\mathbf{\Sigma}_f$ is a 3-move protocol between interactive PPT algorithms $\mathcal{P}$ and $\mathcal{V}$ on initial input $(x := (x_{i_j})_{1\le j\le \mathrm{arity}(f)}, w := (w_{i_j})_{1\le j\le \mathrm{arity}(f)}) \in R_f$ for $\mathcal{P}$ and x for $\mathcal{V}$.

In our prover-algorithm $\mathcal{P}$, there are three PPT subroutines; $\mathbf{\Sigma}_f^{\mathrm{eval}}$, $\mathbf{\Sigma}_f^1$ and $\mathbf{\Sigma}_f^3$. On the other hand, in our verifier-algorithm $\mathcal{V}$ there are PPT subroutines; $\mathbf{\Sigma}_f^2$ and $\mathbf{\Sigma}_f^{\mathrm{vrfy}}$. $\mathbf{\Sigma}_f^{\mathrm{vrfy}}$ has two subroutines; **VrfyCha** and **VrfyRes**.

Evaluation of Satisfiability. The prover $\mathcal{P}$ begins with evaluation concerning whether and how S satisfies f. We label each node of $\mathcal{T}$ with a value $v = 1$ (TRUE) or 0 (FALSE). For each leaf node lf, we label lf with $v_{\mathrm{lf}} = 1$ if $i(\mathrm{lf}) \in S$ and $v_{\mathrm{lf}} = 0$ otherwise. For each inner node nd, we label nd with $v_{\mathrm{nd}} = v_{\mathrm{nd_L}} \wedge v_{\mathrm{nd_R}}$ or $v_{\mathrm{nd}} = v_{\mathrm{nd_L}} \vee v_{\mathrm{nd_L}}$ according to AND/OR evaluation of two labels of its two children $\mathrm{nd_L}, \mathrm{nd_R}$. The computation is executed for every node from the root node to each leaf node, recursively, in the following way.

$\mathbf{\Sigma}_f^{\mathrm{eval}}(\mathcal{T}, S)$:
$\mathcal{T}_{\mathrm{L}} := \mathrm{Lsub}(\mathcal{T}), \mathcal{T}_{\mathrm{R}} := \mathrm{Rsub}(\mathcal{T})$
If $\mathrm{rt}(\mathcal{T})$ is an $\wedge$-node,
then Return $v_{\mathrm{rt}(\mathcal{T})} := (\mathbf{\Sigma}_f^{\mathrm{eval}}(\mathcal{T}_{\mathrm{L}}, S) \wedge \mathbf{\Sigma}_f^{\mathrm{eval}}(\mathcal{T}_{\mathrm{R}}, S))$
else if $\mathrm{rt}(\mathcal{T})$ is an $\vee$-node,
then Return $v_{\mathrm{rt}(\mathcal{T})} := ((\mathbf{\Sigma}_f^{\mathrm{eval}}(\mathcal{T}_{\mathrm{L}}, S) \vee \mathbf{\Sigma}_f^{\mathrm{eval}}(\mathcal{T}_{\mathrm{R}}, S))$
else if $\mathrm{rt}(\mathcal{T})$ is a leaf node,
then Return $v_{\mathrm{rt}(\mathcal{T})} := (i(\mathrm{rt}(\mathcal{T})) \stackrel{?}{\in} S)$

Commitment. Prover's computation of a commitment value for each leaf node is described in Fig. 2. Basically, the algorithm $\mathbf{\Sigma}_f^1$ runs for every node from the root node to each leaf node, recursively. As a result, $\mathbf{\Sigma}_f^1$ generates for each leaf node lf a value $\mathrm{CMT}_{\mathrm{lf}}$; If $v_{\mathrm{lf}} = 1$, then $\mathrm{CMT}_{\mathrm{lf}}$ is computed honestly according to $\mathbf{\Sigma}^1$. Else if $v_{\mathrm{lf}} = 1$, then $\mathrm{CMT}_{\mathrm{lf}}$ is computed in the simulated way according to $\mathbf{\Sigma}^{\mathrm{sim}}$. Other values, $(\mathrm{CHA}_{\mathrm{nd}})_{\mathrm{nd}\in \mathrm{tNd}(\mathcal{T}_f)}$ and $(\mathrm{RES}_{\mathrm{lf}})_{\mathrm{lf}\in \mathrm{Lf}(\mathcal{T}_f)}$, are needed for the simulation. Note that a distinguished symbol '$*$' is used for those other values to indicate the honest computation.

$\boldsymbol{\Sigma}_f^1(x, w, \mathcal{T}, (v_{\mathrm{nd}})_{\mathrm{nd}\in\mathrm{Nd}(\mathcal{T})}, \mathrm{CHA}):$
$\mathcal{T}_{\mathrm{L}} := \mathrm{Lsub}(\mathcal{T}), \mathcal{T}_{\mathrm{R}} := \mathrm{Rsub}(\mathcal{T})$
If $\mathrm{rt}(\mathcal{T})$ is $\wedge$-node, then $\mathrm{CHA}_{\mathrm{rt}(\mathcal{T}_{\mathrm{L}})} := \mathrm{CHA}, \mathrm{CHA}_{\mathrm{rt}(\mathcal{T}_{\mathrm{R}})} := \mathrm{CHA}$
Return$(\mathrm{CHA}_{\mathrm{rt}(\mathcal{T}_{\mathrm{L}})}, \boldsymbol{\Sigma}_f^1(x, w, \mathcal{T}_{\mathrm{L}}, (v_{\mathrm{nd}})_{\mathrm{nd}\in\mathrm{Nd}(\mathcal{T}_{\mathrm{L}})}, \mathrm{CHA}_{\mathrm{rt}(\mathcal{T}_{\mathrm{L}})}),$
$\mathrm{CHA}_{\mathrm{rt}(\mathcal{T}_{\mathrm{R}})}, \boldsymbol{\Sigma}_f^1(x, w, \mathcal{T}_{\mathrm{R}}, (v_{\mathrm{nd}})_{\mathrm{nd}\in\mathrm{Nd}(\mathcal{T}_{\mathrm{R}})}, \mathrm{CHA}_{\mathrm{rt}(\mathcal{T}_{\mathrm{R}})}))$
else if $\mathrm{rt}(\mathcal{T})$ is $\vee$-node, then
If $v_{\mathrm{rt}(\mathcal{T}_{\mathrm{L}})} = 1 \wedge v_{\mathrm{rt}(\mathcal{T}_{\mathrm{R}})} = 1$, then $\mathrm{CHA}_{\mathrm{rt}(\mathcal{T}_{\mathrm{L}})} := *, \quad \mathrm{CHA}_{\mathrm{rt}(\mathcal{T}_{\mathrm{R}})} := *$
else if $v_{\mathrm{rt}(\mathcal{T}_{\mathrm{L}})} = 1 \wedge v_{\mathrm{rt}(\mathcal{T}_{\mathrm{R}})} = 0$, then $\mathrm{CHA}_{\mathrm{rt}(\mathcal{T}_{\mathrm{L}})} := *, \quad \mathrm{CHA}_{\mathrm{rt}(\mathcal{T}_{\mathrm{R}})} \leftarrow \boldsymbol{\Sigma}^2(\lambda)$
else if $v_{\mathrm{rt}(\mathcal{T}_{\mathrm{L}})} = 0 \wedge v_{\mathrm{rt}(\mathcal{T}_{\mathrm{R}})} = 1$, then $\mathrm{CHA}_{\mathrm{rt}(\mathcal{T}_{\mathrm{L}})} \leftarrow \boldsymbol{\Sigma}^2(\lambda), \mathrm{CHA}_{\mathrm{rt}(\mathcal{T}_{\mathrm{R}})} := *$
else if $v_{\mathrm{rt}(\mathcal{T}_{\mathrm{L}})} = 0 \wedge v_{\mathrm{rt}(\mathcal{T}_{\mathrm{R}})} = 0$, then $\mathrm{CHA}_{\mathrm{rt}(\mathcal{T}_{\mathrm{L}})} \leftarrow \boldsymbol{\Sigma}^2(\lambda), \mathrm{CHA}_{\mathrm{rt}(\mathcal{T}_{\mathrm{R}})} := \mathrm{CHA} \oplus \mathrm{CHA}_{\mathrm{rt}(\mathcal{T}_{\mathrm{L}})}$
Return$(\mathrm{CHA}_{\mathrm{rt}(\mathcal{T}_{\mathrm{L}})}, \boldsymbol{\Sigma}_f^1(x, w, \mathcal{T}_{\mathrm{L}}, (v_{\mathrm{nd}})_{\mathrm{nd}\in\mathrm{Nd}(\mathcal{T}_{\mathrm{L}})}, \mathrm{CHA}_{\mathrm{rt}(\mathcal{T}_{\mathrm{L}})}),$
$\mathrm{CHA}_{\mathrm{rt}(\mathcal{T}_{\mathrm{R}})}, \boldsymbol{\Sigma}_f^1(x, w, \mathcal{T}_{\mathrm{R}}, (v_{\mathrm{nd}})_{\mathrm{nd}\in\mathrm{Nd}(\mathcal{T}_{\mathrm{R}})}, \mathrm{CHA}_{\mathrm{rt}(\mathcal{T}_{\mathrm{R}})}))$
else if $\mathrm{rt}(\mathcal{T})$ is a leaf-node, then
If $v_{\mathrm{rt}(\mathcal{T})} = 1$, then $\mathrm{CMT}_{\mathrm{rt}(\mathcal{T})} \leftarrow \boldsymbol{\Sigma}^1(x_{i(\mathrm{rt}(\mathcal{T}))}, w_{i(\mathrm{rt}(\mathcal{T}))}), \mathrm{RES}_{\mathrm{rt}(\mathcal{T})} := *$
else if $v_{\mathrm{rt}(\mathcal{T})} = 0$, then $(\mathrm{CMT}_{\mathrm{rt}(\mathcal{T})}, \mathrm{RES}_{\mathrm{rt}(\mathcal{T})}) \leftarrow \boldsymbol{\Sigma}^{\mathrm{sim}}(x_{i(\mathrm{rt}(\mathcal{T}))}, \mathrm{CHA})$
Return$(\mathrm{CMT}_{\mathrm{rt}(\mathcal{T})}, \mathrm{RES}_{\mathrm{rt}(\mathcal{T})})$

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

Challenge. Verifier picks up a challenge value by using $\boldsymbol{\Sigma}^2$.

$\boldsymbol{\Sigma}_f^2(\lambda):$
$\mathrm{CHA} \leftarrow \boldsymbol{\Sigma}^2(\lambda)$
Return(CHA)

Response. Prover's computation of a response value for each leaf node is described in Fig. 3. Basically, the algorithm $\boldsymbol{\Sigma}_f^3$ runs for every node from the root node to each leaf node, recursively. As a result, $\boldsymbol{\Sigma}_f^3$ generates values, $(\mathrm{CHA}_{\mathrm{nd}})_{\mathrm{nd}\in\mathrm{tNd}(\mathcal{T}_f)}$ and $(\mathrm{RES}_{\mathrm{lf}})_{\mathrm{lf}\in\mathrm{Lf}(\mathcal{T}_f)}$. Note that all challenge values $(\mathrm{CHA}_{\mathrm{nd}})_{\mathrm{nd}\in\mathrm{tNd}(\mathcal{T}_f)}$ are completed according to the "division rule" described in Section 1.1.

$\boldsymbol{\Sigma}_f^3(x, w, \mathcal{T}, (v_{\mathrm{nd}})_{\mathrm{nd}\in\mathrm{Nd}(\mathcal{T})}, \mathrm{CHA}, (\mathrm{CMT}_{\mathrm{lf}})_{\mathrm{lf}\in\mathrm{Lf}(\mathcal{T})}, (\mathrm{CHA}_{\mathrm{nd}})_{\mathrm{nd}\in\mathrm{tNd}(\mathcal{T})}, (\mathrm{RES}_{\mathrm{lf}})_{\mathrm{lf}\in\mathrm{Lf}(\mathcal{T})}):$
$\mathcal{T}_{\mathrm{L}} := \mathrm{Lsub}(\mathcal{T}), \mathcal{T}_{\mathrm{R}} := \mathrm{Rsub}(\mathcal{T})$
If $\mathrm{rt}(\mathcal{T})$ is $\wedge$-node, then $\mathrm{CHA}_{\mathrm{rt}(\mathcal{T}_{\mathrm{L}})} := \mathrm{CHA}, \mathrm{CHA}_{\mathrm{rt}(\mathcal{T}_{\mathrm{R}})} := \mathrm{CHA}$
Return$(\mathrm{CHA}_{\mathrm{rt}(\mathcal{T}_{\mathrm{L}})}, \boldsymbol{\Sigma}_f^3(x, w, \mathcal{T}_{\mathrm{L}}, (v_{\mathrm{nd}})_{\mathrm{nd}\in\mathrm{Nd}(\mathcal{T}_{\mathrm{L}})}, (\mathrm{CMT}_{\mathrm{lf}})_{\mathrm{lf}\in\mathrm{Lf}(\mathcal{T}_{\mathrm{L}})}, \mathrm{CHA}_{\mathrm{rt}(\mathcal{T}_{\mathrm{L}})}, (\mathrm{CHA}_{\mathrm{nd}})_{\mathrm{nd}\in\mathrm{tNd}(\mathcal{T}_{\mathrm{L}})}, (\mathrm{RES}_{\mathrm{lf}})_{\mathrm{lf}\in\mathrm{Lf}(\mathcal{T}_{\mathrm{L}})}),$
$\mathrm{CHA}_{\mathrm{rt}(\mathcal{T}_{\mathrm{R}})}, \boldsymbol{\Sigma}_f^3(x, w, \mathcal{T}_{\mathrm{R}}, (v_{\mathrm{nd}})_{\mathrm{nd}\in\mathrm{Nd}(\mathcal{T}_{\mathrm{R}})}, (\mathrm{CMT}_{\mathrm{lf}})_{\mathrm{lf}\in\mathrm{Lf}(\mathcal{T}_{\mathrm{R}})}, \mathrm{CHA}_{\mathrm{rt}(\mathcal{T}_{\mathrm{R}})}, (\mathrm{CHA}_{\mathrm{nd}})_{\mathrm{nd}\in\mathrm{tNd}(\mathcal{T}_{\mathrm{R}})}, (\mathrm{RES}_{\mathrm{lf}})_{\mathrm{lf}\in\mathrm{Lf}(\mathcal{T}_{\mathrm{R}})}))$
else if $\mathrm{rt}(\mathcal{T})$ is $\vee$-node, then
If $v_{\mathrm{rt}(\mathcal{T}_{\mathrm{L}})} = 1 \wedge v_{\mathrm{rt}(\mathcal{T}_{\mathrm{R}})} = 1$, then $\mathrm{CHA}_{\mathrm{rt}(\mathcal{T}_{\mathrm{L}})} \leftarrow \boldsymbol{\Sigma}^2(\lambda), \quad \mathrm{CHA}_{\mathrm{rt}(\mathcal{T}_{\mathrm{R}})} := \mathrm{CHA} \oplus \mathrm{CHA}_{\mathrm{rt}(\mathcal{T}_{\mathrm{L}})}$
else if $v_{\mathrm{rt}(\mathcal{T}_{\mathrm{L}})} = 1 \wedge v_{\mathrm{rt}(\mathcal{T}_{\mathrm{R}})} = 0$, then $\mathrm{CHA}_{\mathrm{rt}(\mathcal{T}_{\mathrm{L}})} := \mathrm{CHA} \oplus \mathrm{CHA}_{\mathrm{rt}(\mathcal{T}_{\mathrm{R}})}, \mathrm{CHA}_{\mathrm{rt}(\mathcal{T}_{\mathrm{R}})} := \mathrm{CHA}_{\mathrm{rt}(\mathcal{T}_{\mathrm{R}})}$
else if $v_{\mathrm{rt}(\mathcal{T}_{\mathrm{L}})} = 0 \wedge v_{\mathrm{rt}(\mathcal{T}_{\mathrm{R}})} = 1$, then $\mathrm{CHA}_{\mathrm{rt}(\mathcal{T}_{\mathrm{L}})} := \mathrm{CHA}_{\mathrm{rt}(\mathcal{T}_{\mathrm{L}})}, \quad \mathrm{CHA}_{\mathrm{rt}(\mathcal{T}_{\mathrm{R}})} := \mathrm{CHA} \oplus \mathrm{CHA}_{\mathrm{rt}(\mathcal{T}_{\mathrm{L}})}$
else if $v_{\mathrm{rt}(\mathcal{T}_{\mathrm{L}})} = 0 \wedge v_{\mathrm{rt}(\mathcal{T}_{\mathrm{R}})} = 0$, then $\mathrm{CHA}_{\mathrm{rt}(\mathcal{T}_{\mathrm{L}})} := \mathrm{CHA}_{\mathrm{rt}(\mathcal{T}_{\mathrm{L}})}, \quad \mathrm{CHA}_{\mathrm{rt}(\mathcal{T}_{\mathrm{R}})} := \mathrm{CHA}_{\mathrm{rt}(\mathcal{T}_{\mathrm{R}})}$
Return$(\mathrm{CHA}_{\mathrm{rt}(\mathcal{T}_{\mathrm{L}})}, \boldsymbol{\Sigma}_f^3(x, w, \mathcal{T}_{\mathrm{L}}, (v_{\mathrm{nd}})_{\mathrm{nd}\in\mathrm{Nd}(\mathcal{T}_{\mathrm{L}})}, (\mathrm{CMT}_{\mathrm{lf}})_{\mathrm{lf}\in\mathrm{Lf}(\mathcal{T}_{\mathrm{L}})}, \mathrm{CHA}_{\mathrm{rt}(\mathcal{T}_{\mathrm{L}})}, (\mathrm{CHA}_{\mathrm{nd}})_{\mathrm{nd}\in\mathrm{tNd}(\mathcal{T}_{\mathrm{L}})}, (\mathrm{RES}_{\mathrm{lf}})_{\mathrm{lf}\in\mathrm{Lf}(\mathcal{T}_{\mathrm{L}})}),$
$\mathrm{CHA}_{\mathrm{rt}(\mathcal{T}_{\mathrm{R}})}, \boldsymbol{\Sigma}_f^3(x, w, \mathcal{T}_{\mathrm{R}}, (v_{\mathrm{nd}})_{\mathrm{nd}\in\mathrm{Nd}(\mathcal{T}_{\mathrm{R}})}, (\mathrm{CMT}_{\mathrm{lf}})_{\mathrm{lf}\in\mathrm{Lf}(\mathcal{T}_{\mathrm{R}})}, \mathrm{CHA}_{\mathrm{rt}(\mathcal{T}_{\mathrm{R}})}, (\mathrm{CHA}_{\mathrm{nd}})_{\mathrm{nd}\in\mathrm{tNd}(\mathcal{T}_{\mathrm{R}})}, (\mathrm{RES}_{\mathrm{lf}})_{\mathrm{lf}\in\mathrm{Lf}(\mathcal{T}_{\mathrm{R}})}))$
else if $\mathrm{rt}(\mathcal{T})$ is a leaf-node, then
If $v_{\mathrm{rt}(\mathcal{T})} = 1$, then $\mathrm{RES}_{\mathrm{rt}(\mathcal{T})} \leftarrow \boldsymbol{\Sigma}^3(x_{i(\mathrm{rt}(\mathcal{T}))}, w_{i(\mathrm{rt}(\mathcal{T}))}, \mathrm{CMT}_{\mathrm{rt}(\mathcal{T})}, \mathrm{CHA})$
else if $v_{\mathrm{rt}(\mathcal{T})} = 0$, then do nothing
Return$(\mathrm{RES}_{\mathrm{rt}(\mathcal{T})})$

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

Verification. Verifier's computation is executed for each leaf node as follows.

$$
\begin{aligned}
&\boldsymbol{\Sigma}_f^{\mathrm{vrfy}}(x, \mathcal{T}, \mathrm{CHA}, \\
&\quad (\mathrm{CMT}_{\mathrm{lf}})_{\mathrm{lf}\in\mathrm{Lf}(\mathcal{T})}, (\mathrm{CHA}_{\mathrm{nd}})_{\mathrm{nd}\in\mathrm{tNd}(\mathcal{T})}, (\mathrm{RES}_{\mathrm{lf}})_{\mathrm{lf}\in\mathrm{Lf}(\mathcal{T})}) : \\
&\quad \mathrm{Return}(\mathbf{VrfyCha}(\mathcal{T}, \mathrm{CHA}, (\mathrm{CHA}_{\mathrm{nd}})_{\mathrm{nd}\in\mathrm{tNd}(\mathcal{T})}) \\
&\qquad \wedge \mathbf{VrfyRes}(x, \mathcal{T}, (\mathrm{CMT}, \mathrm{CHA}, \mathrm{RES})_{\mathrm{lf}\in\mathrm{Lf}(\mathcal{T})}))
\end{aligned}
$$

$$
\begin{aligned}
&\mathbf{VrfyCha}(\mathcal{T}, \mathrm{CHA}, (\mathrm{CHA}_{\mathrm{nd}})_{\mathrm{nd}\in\mathrm{tNd}(\mathcal{T})}) : \\
&\quad \mathcal{T}_{\mathrm{L}} := \mathrm{Lsub}(\mathcal{T}), \mathcal{T}_{\mathrm{R}} := \mathrm{Rsub}(\mathcal{T}) \\
&\quad \text{If } \mathrm{rt}(\mathcal{T}) \text{ is an } \wedge\text{-node}, \\
&\qquad \text{then Return } ((\mathrm{CHA} \stackrel{?}{=} \mathrm{CHA}_{\mathrm{rt}(\mathcal{T}_{\mathrm{L}})}) \wedge (\mathrm{CHA} \stackrel{?}{=} \mathrm{CHA}_{\mathrm{rt}(\mathcal{T}_{\mathrm{R}})}) \\
&\qquad\quad \wedge \mathbf{VrfyCha}(\mathcal{T}_{\mathrm{L}}, \mathrm{CHA}_{\mathrm{rt}(\mathcal{T}_{\mathrm{L}})}, (\mathrm{CHA}_{\mathrm{nd}})_{\mathrm{nd}\in\mathrm{tNd}(\mathcal{T}_{\mathrm{L}})}) \\
&\qquad\quad \wedge \mathbf{VrfyCha}(\mathcal{T}_{\mathrm{R}}, \mathrm{CHA}_{\mathrm{rt}(\mathcal{T}_{\mathrm{R}})}, (\mathrm{CHA}_{\mathrm{nd}})_{\mathrm{nd}\in\mathrm{tNd}(\mathcal{T}_{\mathrm{R}})})) \\
&\quad \text{else if } \mathrm{rt}(\mathcal{T}) \text{ is an } \vee\text{-node}, \\
&\qquad \text{then Return } ((\mathrm{CHA} \stackrel{?}{=} \mathrm{CHA}_{\mathrm{rt}(\mathcal{T}_{\mathrm{L}})} \oplus \mathrm{CHA}_{\mathrm{rt}(\mathcal{T}_{\mathrm{R}})}) \\
&\qquad\quad \wedge \mathbf{VrfyCha}(\mathcal{T}_{\mathrm{L}}, \mathrm{CHA}_{\mathrm{rt}(\mathcal{T}_{\mathrm{L}})}, (\mathrm{CHA}_{\mathrm{nd}})_{\mathrm{nd}\in\mathrm{tNd}(\mathcal{T}_{\mathrm{L}})}) \\
&\qquad\quad \wedge \mathbf{VrfyCha}(\mathcal{T}_{\mathrm{R}}, \mathrm{CHA}_{\mathrm{rt}(\mathcal{T}_{\mathrm{R}})}, (\mathrm{CHA}_{\mathrm{nd}})_{\mathrm{nd}\in\mathrm{tNd}(\mathcal{T}_{\mathrm{R}})})) \\
&\quad \text{else if } \mathrm{rt}(\mathcal{T}) \text{ is a leaf node}, \\
&\qquad \text{then Return } (\mathrm{CHA} \stackrel{?}{\in} \mathrm{CHASP}(\lambda))
\end{aligned}
$$

$$
\begin{aligned}
&\mathbf{VrfyRes}(x, \mathcal{T}, (\mathrm{CMT}, \mathrm{CHA}, \mathrm{RES})_{\mathrm{lf}\in\mathrm{Lf}(\mathcal{T})}) : \\
&\quad \text{For lf} \in \mathrm{Lf}(\mathcal{T}) \\
&\qquad \text{If } \boldsymbol{\Sigma}^{\mathrm{vrfy}}(x_{i(\mathrm{lf})}, \mathrm{CMT}_{\mathrm{lf}}, \mathrm{CHA}_{\mathrm{lf}}, \mathrm{RES}_{\mathrm{lf}}) = 0, \text{then Return } (0) \\
&\quad \text{Return } (1)
\end{aligned}
$$

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

Proposition 1 (Completeness) *Completeness holds for our $\boldsymbol{\Sigma}_f$. More precisely, Suppose that $v_{rt(\mathcal{T}_f)} = 1$. Then, for every node in $Nd(\mathcal{T}_f)$, either $v_{nd} = 1$ or $\mathrm{CHA}_{nd} \neq *$ holds after executing $\boldsymbol{\Sigma}_f^1$.*

Proof. Induction on the height of $\mathcal{T}_f$. The case of height 0 follows from $v_{\mathrm{rt}(\mathcal{T}_f)} = 1$ and the completeness of $\boldsymbol{\Sigma}$. Suppose that the case of height k holds and consider the case of height $k+1$. The construction of $\boldsymbol{\Sigma}_f^1$ assures the case of height $k+1$. □

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

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

$\mathbf{\Sigma}_f^{\mathrm{KE}}(x, (\mathrm{CMT}_{\mathrm{lf}}, \mathrm{CHA}_{\mathrm{lf}}, \mathrm{RES}_{\mathrm{lf}})_{\mathrm{lf}\in \mathrm{Lf}(\mathcal{T}_f)}, (\mathrm{CMT}_{\mathrm{lf}}, \mathrm{CHA}'_{\mathrm{lf}}, \mathrm{RES}'_{\mathrm{lf}})_{\mathrm{lf}\in \mathrm{Lf}(\mathcal{T}_f)})$:

For $1 \leq j \leq \mathrm{arity}(f) : w_{i_j}^* := *$

For $\mathrm{lf} \in \mathrm{Lf}(\mathcal{T}_f)$

If $\mathrm{CHA}_{\mathrm{lf}} \neq \mathrm{CHA}'_{\mathrm{lf}}$, then $w_{i(\mathrm{lf})}^* \leftarrow \mathbf{\Sigma}^{\mathrm{KE}}(x_{i(\mathrm{lf})}, (\mathrm{CMT}_{\mathrm{lf}}, \mathrm{CHA}_{\mathrm{lf}}, \mathrm{RES}_{\mathrm{lf}}), (\mathrm{CMT}_{\mathrm{lf}}, \mathrm{CHA}'_{\mathrm{lf}}, \mathrm{RES}'_{\mathrm{lf}}))$

else If $w_{i(\mathrm{lf})}^* = *$, then $w_{i(\mathrm{lf})}^* \leftarrow \{1,0\}^*$

Return $(w^* := (w_{i_j}^*)_{1\leq j\leq \mathrm{arity}(f)})$

Then Lemma 1 assures the proposition.

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

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

Suppose that all nodes in $\mathrm{iNd}(T_f)$ are $\wedge$-nodes. Then the above claim follows immediately because $\mathrm{CHA}_{\mathrm{lf}} \neq \mathrm{CHA}'_{\mathrm{lf}}$ holds for all leaf nodes.

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 nd^*, respectively. Then $\mathrm{CHA}_{\mathrm{nd}^*} \neq \mathrm{CHA}'_{\mathrm{nd}^*}$ because all nodes with height less than h^* are $\wedge$-nodes. So at least one of children of nd^*, say nd_L^*, satisfies $\mathrm{CHA}_{\mathrm{nd}_L^*} \neq \mathrm{CHA}'_{\mathrm{nd}_L^*}$. Divide the tree $\mathcal{T}_f$ into two subtrees by cutting the branch right above nd^*, and the induction hypothesis assures the claim. □

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

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

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

Theorem 1 *Our protocol $\mathbf{\Sigma}_f$ obtained from a Σ-protocol $\mathbf{\Sigma}$ for the language L_R and a boolean formula f is a Σ-protocol for the language L_{R_f}.*

Corollary 1 *Our protocol $\mathbf{\Sigma}_f$ is a boolean proof for the language L_{R_f}.*

5 Our Non-interactive Witness-Indistinguishable Proof of Knowledge System

In this section, we apply the Fiat-Shamir transformation $\mathrm{FS}(\cdot)$ to our Σ-protocol $\mathbf{\Sigma}_f$. The resulting system $\mathrm{FS}(\mathbf{\Sigma}_f)$ is a non-interactive zero-knowledge proof of knowledge (NIZKPoK) system and hence a non-interactive witness-indistinguishable proof of knowledge (NIWIPoK) system.

The Fiat-Shamir transformation $\mathrm{FS}(\cdot)$ can be applied to any Σ-protocol $\mathbf{\Sigma}$ ([13, 1], see Section 2.1) to obtain a NIZKPoK system. As a NIZKPoK system is a NIWIPoK system, we obtain a NIWIPoK system. Here the generator of common reference strings is becomes as follows.

$K(\lambda)$:

$\mu \leftarrow \mathit{Hashkeysp}(\lambda), \mathrm{crs} := \mu$

Return crs

Hence we obtain the following theorem.

Theorem 2 *FS($\mathbf{\Sigma}_f$) is a non-interactive witness-indistinguishable proof of knowledge system for the language L_{R_f}. A knowledge extractor can be constructed in the random oracle model.*

6 Our Attribute-Based Signature Scheme

In this section, we propose an attribute-based signature scheme. By combining the NIWIPoK system FS($\mathbf{\Sigma}_f$) with a credential bundle scheme of the Fiat-Shamir signature FS($\mathbf{\Sigma}$), we obtain an attribute-based signature scheme ABS, which has collusion resistance against collecting private secret keys. Our ABS is secure in the random oracle model, possesses attribute privacy, and has a feature that it can be constructed without pairings.

6.1 Our ABS

Our ABS = (**Setup**, **KG**, **Sign**, **Vrfy**) is described as follows. Fig. 4 shows our construction of ABS scheme, ABS = (**Setup**, **KG**, **Sign**, **Vrfy**).

Setup computes a public key PK and a master secret key MSK by running Instance$_R(\lambda)$ which chooses a pair $(x_{\mathrm{mst}}, w_{\mathrm{mst}})$ at random from $R = \{(x, w)\}$, where $|x|$ and $|w|$ are bounded by a polynomial in λ.

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

KG first, on input $\mathrm{PK}, \mathrm{MSK}, \tau, S$, obtains a hash function value k on input $w_{\mathrm{mst}} \parallel \tau$, which is used as a pseudorandom function key. Then KG applies the credential bundle technique [19] for each message $m_i := (\tau \parallel i), i \in S$. Here we employ the Fiat-Shamir signing algorithm FS($\mathbf{\Sigma}$)$^{\mathrm{sign}}$ with a random tape $\mathit{PRF}_k(m_i \parallel 0)$ for $\mathbf{\Sigma}$ and a hash key μ for *Hash* (see 2.1).

$$
\begin{aligned}
&\mathbf{KG}(\mathrm{PK}, \mathrm{MSK}, \tau, S) : \\
&\ k \leftarrow \mathit{Hash}_\mu(w_{\mathrm{mst}} \parallel \tau) \\
&\ \mathrm{For}\ i \in S : \\
&\ \ m_i := (\tau \parallel i), (a_i, w_i) \leftarrow \mathrm{FS}(\mathbf{\Sigma})^{\mathrm{sign}}(x_{\mathrm{mst}}, w_{\mathrm{mst}}, m_i; \mathit{PRF}_k(m_i \parallel 0)) \\
&\ \mathrm{SK}_S := (\tau, k, ((a_i, w_i))_{i \in S}), \mathrm{Return\ SK}_S.
\end{aligned}
$$

Sign uses a supplementary algorithm **Supp** and a statement-generator algorithm **StmtGen**. **Supp** runs for $j,\ 1 \le j \in \mathrm{arity}(f)$ and generates simulated keys (a_{i_j}, w_{i_j}) for $i_j \notin S$.

$$
\begin{aligned}
&\mathbf{Supp}(\mathrm{PK}, \mathrm{SK}_S, f) : \\
&\ \mathrm{For}\ j = 1\ \mathrm{to}\ \mathrm{arity}(f) : \\
&\ \ \mathrm{If}\ i_j \notin S,\ \mathrm{then} \\
&\ \ \ m_{i_j} := (\tau \parallel i_j), c_{i_j} \leftarrow \mathit{PRF}_k(m_{i_j} \parallel 1), (a_{i_j}, w_{i_j}) \leftarrow \mathbf{\Sigma}^{\mathrm{sim}}(x_{\mathrm{mst}}, c_{i_j}; \mathit{PRF}_k(m_{i_j} \parallel 2)) \\
&\ \mathrm{Return}\ ((a_{i_j}, w_{i_j}))_{1 \le j \le \mathrm{arity}(f)}
\end{aligned}
$$

Note here that:

$$\Pr[c_{i_j} = \mathit{Hash}_\mu(a_{i_j} \parallel m_{i_j})] = \mathrm{neg}(\lambda) \text{ for } i_j \notin S.$$

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

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

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

Then **Sign** is obtained by adding the following procedures.

$$
\begin{aligned}
&\mathbf{Supp}(\mathrm{PK}, \mathrm{SK}_S, f) \to ((a_{i_j}, w_{i_j}))_{1\le j\le \mathrm{arity}(f)} \\
&\mathbf{StmtGen}(\mathrm{PK}, \tau, (a_{i_j})_{1\le j\le \mathrm{arity}(f)}) \\
&\quad \to (x_{i_j})_{1\le j\le \mathrm{arity}(f)} =: x \\
&\quad w := (w_{i_j})_{1\le j\le \mathrm{arity}(f)}.
\end{aligned}
$$

The above procedures are needed to give a pair of statement and witness, (x, w), to $\mathbf{\Sigma}^1_f$. Note here that $(x = (x_{i_j})_{1\le j\le \mathrm{arity}(f)}, w = (w_{i_j})_{1\le j\le \mathrm{arity}(f)})$ and $(x_{i_j}, w_{i_j}) \in R$ for any $i_j \in S$. On the other hand, $(x_{i_j}, w_{i_j}) \notin R$ for any $i_j \notin S$, without a negligible probability, $\mathrm{neg}(\lambda)$.

Hence the signature σ that should be output by **Sign** has to include the tag τ and elements $(a_i)_{i\in \mathrm{Att}(f)}$ for the verifying algorithm **Vrfy** to be able to produce the same statement x by using τ and $(a_i)_{i\in \mathrm{Att}(f)}$ as well as x_{mst} in PK.

Finally, **Vrfy** utilizes **StmtGen** and $\mathbf{\Sigma}^{\mathrm{vrfy}}_f$ to check validity of the message m and the signature σ under the public key PK and the access formula f.

6.2 Security of Our ABS

Applying the standard technique in [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 Σ-protocol $\mathbf{\Sigma}_f$ with the credential bundle scheme of the Fiat-Shamir signature $\mathrm{FS}(\mathbf{\Sigma})$ in the same way as ABS (See Appendix C for our ABID).

Theorem 3 *Our attribute-based signature scheme ABS is secure in the game of existential unforgeability 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 ($\mathit{neg}(\cdot)$ means a negligible function).*

$$
\mathbf{Adv}^{\mathit{euf\text{-}cma}}_{\mathcal{F},\mathit{ABS}}(\lambda) \le q_H \mathbf{Adv}^{\mathit{pa}}_{\mathcal{B},\mathit{ABID}}(\lambda) + \mathit{neg}(\lambda). \tag{1}
$$

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

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

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

$\mathbf{KG}(\mathrm{PK}, \mathrm{MSK}, \tau, S)$:
$k \leftarrow \mathit{Hash}_\mu(w_{\mathrm{mst}} \parallel \tau)$
For $i \in S$
$m_i := (\tau \parallel i)$
$\mathrm{FS}(\mathbf{\Sigma})^{\mathrm{sign}}(x_{\mathrm{mst}}, w_{\mathrm{mst}}, m_i; \mathit{PRF}_k(m_i \parallel 0))$
$\rightarrow (a_i, w_i)$
$\mathrm{SK}_S := (\tau, k, ((a_i, w_i))_{i \in S})$
Return SK_S

$\mathbf{Sign}(\mathrm{PK}, \mathrm{SK}_S, (m, f))$:
Initialize inner state
$\mathbf{\Sigma}_f^{\mathrm{eval}}(\mathcal{T}_f, S) \rightarrow (v_{\mathrm{nd}})_{\mathrm{nd} \in \mathrm{Nd}(\mathcal{T}_f)}$
If $v_{\mathrm{rt}(\mathcal{T}_f)} \neq 1$, then abort
else $\mathrm{CHA}_{\mathrm{rt}(\mathcal{T}_f)} := *$

$\mathbf{Supp}(\mathrm{PK}, \mathrm{SK}_S, f) \rightarrow ((a_{i_j}, w_{i_j}))_{1 \leq j \leq \mathrm{arity}(f)}$
$\mathbf{StmtGen}(\mathrm{PK}, \tau, (a_{i_j})_{1 \leq j \leq \mathrm{arity}(f)})$
$\rightarrow (x_{i_j})_{1 \leq j \leq \mathrm{arity}(f)} =: x$
$w := (w_{i_j})_{1 \leq j \leq \mathrm{arity}(f)}$

$\mathbf{\Sigma}_f^1(x, w, \mathcal{T}_f, (v_{\mathrm{nd}})_{\mathrm{nd} \in \mathrm{Nd}(\mathcal{T}_f)}, \mathrm{CHA}_{\mathrm{rt}(\mathcal{T}_f)})$
$\rightarrow ((\mathrm{CMT}_{\mathrm{lf}})_{\mathrm{lf} \in \mathrm{Lf}(\mathcal{T}_f)},$
$(\mathrm{CHA}_{\mathrm{nd}})_{\mathrm{nd} \in \mathrm{tNd}(\mathcal{T}_f)},$
$(\mathrm{RES}_{\mathrm{lf}})_{\mathrm{lf} \in \mathrm{Lf}(\mathcal{T}_f)})$

$\mathrm{CHA} \leftarrow \mathit{Hash}_\mu((\mathrm{CMT}_{\mathrm{lf}})_{\mathrm{lf} \in \mathrm{Lf}(\mathcal{T}_f)} \parallel m)$
$\mathrm{CHA}_{\mathrm{rt}(\mathcal{T}_f)} := \mathrm{CHA}$

$\mathbf{\Sigma}_f^3(x, w, \mathcal{T}_f, (v_{\mathrm{nd}})_{\mathrm{nd} \in \mathrm{Nd}(\mathcal{T}_f)}, \mathrm{CHA}_{\mathrm{rt}(\mathcal{T}_f)},$
$(\mathrm{CMT}_{\mathrm{lf}})_{\mathrm{lf} \in \mathrm{Lf}(\mathcal{T}_f)},$
$(\mathrm{CHA}_{\mathrm{nd}})_{\mathrm{nd} \in \mathrm{tNd}(\mathcal{T}_f)},$
$(\mathrm{RES}_{\mathrm{lf}})_{\mathrm{lf} \in \mathrm{Lf}(\mathcal{T}_f)})$
$\rightarrow ((\mathrm{CHA}_{\mathrm{nd}})_{\mathrm{nd} \in \mathrm{tNd}(\mathcal{T}_f)},$
$(\mathrm{RES}_{\mathrm{lf}})_{\mathrm{lf} \in \mathrm{Lf}(\mathcal{T}_f)})$

Return $\sigma := (\tau, (a_{i_j})_{1 \leq j \leq \mathrm{arity}(f)},$
$(\mathrm{CMT}_{\mathrm{lf}})_{\mathrm{lf} \in \mathrm{Lf}(\mathcal{T}_f)},$
$(\mathrm{CHA}_{\mathrm{nd}})_{\mathrm{nd} \in \mathrm{tNd}(\mathcal{T}_f)},$
$(\mathrm{RES}_{\mathrm{lf}})_{\mathrm{lf} \in \mathrm{Lf}(\mathcal{T}_f)}))$

$\mathbf{Vrfy}(\mathrm{PK}, (m, f), \sigma := (\tau, (a_{i_j})_{1 \leq j \leq \mathrm{arity}(f)},$
$(\mathrm{CMT}_{\mathrm{lf}})_{\mathrm{lf} \in \mathrm{Lf}(\mathcal{T}_f)},$
$(\mathrm{CHA}_{\mathrm{nd}})_{\mathrm{nd} \in \mathrm{tNd}(\mathcal{T}_f)},$
$(\mathrm{RES}_{\mathrm{lf}})_{\mathrm{lf} \in \mathrm{Lf}(\mathcal{T}_f)}))$:
Initialize inner state
$\mathrm{CHA} \leftarrow \mathit{Hash}_\mu((\mathrm{CMT}_{\mathrm{lf}})_{\mathrm{lf} \in \mathrm{Lf}(\mathcal{T}_f)} \parallel m)$

$\mathbf{StmtGen}(\mathrm{PK}, \tau, (a_{i_j})_{1 \leq j \leq \mathrm{arity}(f)})$
$\rightarrow (x_{i_j})_{1 \leq j \leq \mathrm{arity}(f)} =: x$

$\mathbf{\Sigma}_f^{\mathrm{vrfy}}(x, \mathcal{T}_f,$
$(\mathrm{CMT}_{\mathrm{lf}})_{\mathrm{lf} \in \mathrm{Lf}(\mathcal{T}_f)},$
$\mathrm{CHA}, (\mathrm{CHA}_{\mathrm{nd}})_{\mathrm{nd} \in \mathrm{tNd}(\mathcal{T}_f)},$
$(\mathrm{RES}_{\mathrm{lf}})_{\mathrm{lf} \in \mathrm{Lf}(\mathcal{T}_f)})$
$\rightarrow b$
Return b

Fig. 4. Our ABS scheme.

6.3 Discussion

Attribute Privacy of our ABS. As opposed to the case of our ABID, our ABS has attribute privacy defined in Section 2.5. Actually, more strongly, attribute privacy holds for $\mathcal{A}$ that has unbounded computational ability because, for a fixed access policy f, the distribution of messages and signatures (m, σ) does not depend on secret keys SK_S where $f(S) = 1$.

Security Reduction. Let q_H denote the maximum number of hash queries issued by a forger $\mathcal{F}$ on ABS and a forger $\mathcal{F}'$ on $\mathrm{FS}(\mathbf{\Sigma})$. Combining the inequality (1) with the inequalities (2) and (3) in Appendix B and C, we obtain the following security reduction of advantages.

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

7 Conclusions

We proposed the first practical attribute-based signature scheme with attribute privacy without pairings in the random oracle model. We first provide a concrete construction of a Σ-protocol of boolean proof. Then, we apply the Fiat-Shamir transformation to our Σ-protocol of boolean proof and obtain a non-interactive witness-indistinguishable proof of knowledge system (NIWIPoK). Finally, by combining our NIWIPoK with a credential bundle scheme of the Fiat-Shamir signature, we obtain an attribute-based signature scheme which possesses attribute privacy. The series of constructions are obtained from a given Σ-protocol and can be constructed without pairings.

Our concrete construction of boolean proof is limited to monotone formulas and to extend the construction to any formula that includes NOT-gate is an open problem [10].

8 Acknowledgements

We appreciate sincere comments of Dr. Keita Emura and Dr. Shingo Hasegawa at SCIS2014 as well as anonymous reviewers of ASIAPKC2014. We also thank to Prof. Tsuyoshi Takagi and Dr. Kirill Morozov for encouraging discussion at Kyushu University.

References

1. M. Abdalla, J. H. An, M. Bellare, and C. Namprempre. From Identification to Signatures via the Fiat-Shamir Transform: Minimizing Assumptions for Security and Forward-Security. In *EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 418–433. Springer.
2. H. Anada, S. Arita, S. Handa, and Y. Iwabuchi. Attribute-Based Identification: Definitions and Efficient Constructions. In *ACISP 2013*, volume 7959 of *LNCS*, pages 168–186. Springer.
3. H. Anada, S. Arita, and K. Sakurai. Attribute-Based Signatures without Pairings via the Fiat-Shamir Paradigm. In *ASIAPKC2014*, volume 2 of *ACM-ASIAPKC*, pages 49–58.
4. A. Beimel. *Secure Schemes for Secret Sharing and Key Distribution.* PhD thesis, Israel Institute of Technology, Technion, Haifa, Israel, 1996.
5. M. Bellare and O. Goldreich. On Defining Proofs of Knowledge. In *CRYPTO '92*, volume 740 of *LNCS*, pages 390–420. Springer.
6. M. Bellare and A. Palacio. GQ and Schnorr Identification Schemes: Proofs of Security against Impersonation under Active and Concurrent Attacks. In *CRYPTO 2002*, volume 2442 of *LNCS*, pages 162–177. Springer.
7. M. Bellare and S. Shoup. Two-tier Signatures, Strongly Unforgeable Signatures, and Fiat-Shamir without Random Oracles. In *PKC 2007*, pages 201–216. Springer-Verlag, 2007.
8. D. Boneh and X. Boyen. Efficient Selective-ID Secure Identity-Based Encryption without Random Oracles. In *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 223–238. Springer.
9. R. Cramer. *Modular Designs of Secure, yet Practical Cyptographic Protocols.* PhD thesis, University of Amsterdam, Amsterdam, the Netherlands, 1996.

10. R. Cramer, I. Damgård, and B. Schoenmakers. Proofs of Partial Knowledge and Simplified Design of Witness Hiding Protocols. In *CRYPTO '94*, pages 174–187. Springer-Verlag, 1994.
11. I. Damgård. On σ-protocols. In Course Notes, https://services.brics.dk/java/courseadmin/CPT/documents, 2011.
12. U. Feige and A. Shamir. Witness Indistinguishable and Witness Hiding Protocols. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing*, STOC '90, pages 416–426, New York, NY, USA, 1990. ACM.
13. A. Fiat and A. Shamir. How to Prove Yourself: Practical Solutions to Identification and Signature Problems. In *CRYPTO '86*, volume 263 of *LNCS*, pages 186–194. Springer.
14. V. Goyal, O. Pandey, A. Sahai, and B. Waters. Attribute-Based Encryption for Fine-Grained Access Control of Encrypted Data. In *ACM-CCS '06*, volume 263, pages 89–98. ACM.
15. J. Groth and A. Sahai. Efficient Non-interactive Proof Systems for Bilinear Groups. In *Proceedings of the Theory and Applications of Cryptographic Techniques 27th Annual International Conference on Advances in Cryptology*, EUROCRYPT'08, pages 415–432, Berlin, Heidelberg, 2008. Springer-Verlag.
16. S. Guo and Y. Zeng. Attribute-Based Signature Scheme. In *ISA '08*, pages 509–511. IEEE.
17. J. Katz and Y. Lindell. *Introduction to Modern Cryptography*. Chapman and Hall/CRC, 2008.
18. J. Li, M. H. Au, W. Susilo, D. Xie, and R. K. Attribute-Based Signature and its Applications. In *ASIA-CCS '10*, volume 5, pages 60–69. ACM.
19. H. K. Maji, M. Prabhakaran, and M. Rosulek. Attribute-Based Signatures. In *CT-RSA 2011*, volume 6558 of *LNCS*, pages 376–392. Springer.
20. T. Okamoto and K. Takashima. Efficient Attribute-Based Signatures for Non-monotone Predicates in the Standard Model. In *PKC 2011*, volume 6571 of *LNCS*, pages 35–52. Springer.
21. A. Sahai and B. Waters. Fuzzy Identity-Based Encryption. In *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 457–473. Springer.
22. C. P. Schnorr. Efficient Identification and Signatures for Smart Cards. In *CRYPTO '89*, volume 435 of *LNCS*, pages 239–252. Springer.
23. A. Shamir. Identity-Based Cryptosystems and Signature Schemes. In *CRYPTO '84*, volume 196 of *LNCS*, pages 47–53. Springer.

A Instantiations

In this section, we provide two instantiations of our $\mathbf{\Sigma}_f$ and `ABS` (and our `ABID`); that is, in the discrete-logarithm setting and the RSA setting.

A.1 Discrete-Logarithm Setting

A prime of bit length λ is denoted by p. A multiplicative cyclic group of order p is denoted by $\mathbb{G}_p$. We fix a base $g \in \mathbb{G}_p, \langle g \rangle = \mathbb{G}_p$. The ring of the exponent domain of $\mathbb{G}_p$, which consists of integers from 0 to $p-1$ with modulo p operation, is denoted by $\mathbb{Z}_p$.

Setup takes as input $(\lambda, \mathcal{U})$. Let $R_\lambda := \{(\beta, \alpha) \in \mathbb{G}_p \times \mathbb{Z}_p; \beta = g^\alpha\}$. Then $\mathrm{Instance}_R(\lambda)$ chooses an element $(\beta, \alpha) \in R_\lambda$ at random. **Setup** outputs a public key and a master secret key: $\mathrm{PK} = ((g, \beta), \mathcal{U}, \mu), \mathrm{MSK} = \alpha$.

KG outputs SK_S with signatures, for $i \in S$, $\sigma_i = (a_i = g^{r_i}, w_i = r_i + c_i\alpha)$. Here we use a key k obtained by $k \leftarrow \mathit{Hash}_\mu(\alpha \parallel \tau)$, put $m_i = \tau \parallel i$, and $r_i \in \mathbb{Z}_p$ is chosen at random according to a random tape: $\mathit{PRF}_k(m_i)$, and c_i is obtained by $c_i \leftarrow \mathit{Hash}_\mu(a_i \parallel m_i)$. $\mathbf{\Sigma}^{\mathrm{stmtRES}}(\beta, a_i, c_i)$ is an algorithm that computes $x_i := a_i \beta^{c_i} \in \mathbb{G}_p$.

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

$$\mathrm{CMT}_{\mathrm{lf}} = g^{r_{\mathrm{lf}}}, \mathrm{RES}_{\mathrm{lf}} = r_{\mathrm{lf}} + \mathrm{CHA}_{\mathrm{lf}}\ w_{i(\mathrm{lf})},$$
$$\text{Verification Equation} : g^{\mathrm{RES}_{\mathrm{lf}}} \stackrel{?}{=} \mathrm{CMT}_{\mathrm{lf}}\ (x_{i(\mathrm{lf})})^{\mathrm{CHA}_{\mathrm{lf}}}.$$

A.2 RSA Setting

An RSA modulus of bit length λ is denoted by N. An RSA exponent of odd prime is denoted by e.

Setup takes as input $(\lambda, \mathcal{U})$. Let $R_\lambda := \{(\beta, \alpha) \in \mathbb{Z}_N \times \mathbb{Z}_N; \beta = \alpha^e\}$. Then $\mathrm{Instance}_R(\lambda)$ chooses an element $(\beta, \alpha) \in R_\lambda$ at random. **Setup** outputs a public key and a master secret key: $\mathrm{PK} = ((N, e, \beta), \mathcal{U}, \mu), \mathrm{MSK} = \alpha$.

KG outputs SK_S with signatures, for $i \in S$, $\sigma = (a_i = r_i^e,\ w_i = r_i \alpha^{c_i})$. Here we use a key k obtained by $k \leftarrow \mathit{Hash}_\mu(\alpha \parallel \tau)$, put $m_i = \tau \parallel i$, and $r_i \in \mathbb{Z}_N$ is chosen at random according to a random tape: $\mathit{PRF}_k(m_i)$, and c_i is obtained by $c_i \leftarrow \mathit{Hash}_\mu(a_i \parallel m_i)$. $\mathbf{\Sigma}^{\mathrm{stmtRes}}(\beta, a_i, c_i)$ is an algorithm that computes $x_i := a_i \beta^{c_i} \in \mathbb{Z}_N$.

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

$$\begin{aligned}
&\textsc{Cmt}_{\mathrm{lf}} = {r_{\mathrm{lf}}}^e, \textsc{Res}_{\mathrm{lf}} = r_{\mathrm{lf}} (w_{i(\mathrm{lf})})^{\textsc{Cha}_{\mathrm{lf}}}, \\
&\text{Verification Equation} : {\textsc{Res}_{\mathrm{lf}}}^e \stackrel{?}{=} \textsc{Cmt}_{\mathrm{lf}}\ (x_{i(\mathrm{lf})})^{\textsc{Cha}_{\mathrm{lf}}}.
\end{aligned}$$

B Attribute-Based Identification Scheme [2]

An attribute-based identification scheme consists of four PPT algorithms: $(\mathbf{Setup}, \mathbf{KG}, \mathcal{P}, \mathcal{V})$ [2].
$\mathbf{Setup}(\lambda, \mathcal{U}) \to (\mathbf{PK}, \mathbf{MSK})$. Setup takes as input the security parameter λ and an attribute universe $\mathcal{U}$. It outputs a public key PK and a master secret key MSK.
$\mathbf{KG}(\mathbf{PK}, \mathbf{MSK}, \tau, S) \to \mathbf{SK}_S$. A key-generation algorithm **KG** takes as input the public key PK, the master secret key MSK, a tag τ[7], and an attribute set $S \subset \mathcal{U}$. It outputs a signing key SK_S corresponding to S.
$\mathcal{P}(\mathbf{PK}, \mathbf{SK}_S)$ and $\mathcal{V}(\mathbf{PK}, f)$. $\mathcal{P}$ and $\mathcal{V}$ are interactive algorithms called a *prover* and a *verifier*, respectively. $\mathcal{P}$ takes as input the public key PK and the secret key SK_S. Here the secret key SK_S is given to $\mathcal{P}$ by an authority that runs $\mathbf{KG}(\mathrm{PK}, \mathrm{MSK}, \tau, S)$. $\mathcal{V}$ takes as input the public key PK and an access formula f. $\mathcal{P}$ is provided V's access formula f by the first move. $\mathcal{P}$ and $\mathcal{V}$ interact with each other for at most constant rounds. Then, $\mathcal{V}$ returns its decision 1 or 0. When it is 1, we say that $\mathcal{V}$ *accepts* $\mathcal{P}$ for f. When it is 0, we say that $\mathcal{V}$ *rejects* $\mathcal{P}$ for f. We demand correctness of ABID that for any λ, and if $f(S) = 1$, then $\Pr[(\mathrm{PK}, \mathrm{MSK}) \leftarrow \mathbf{Setup}(\lambda, \mathcal{U}), \mathrm{SK}_S \leftarrow \mathbf{KG}(\mathrm{PK}, \mathrm{MSK}, S), b \leftarrow \langle \mathcal{P}(\mathrm{PK}, \mathrm{SK}_S), \mathcal{V}(\mathrm{PK}, f) \rangle : b = 1] = 1$.

Attacks on ABID and Security An adversary $\mathcal{A}$'s objective is impersonation. $\mathcal{A}$ tries to make a verifier $\mathcal{V}$ accept with an access formula f^*. The following experiment $\mathbf{Exprmt}^{\mathrm{ca}}_{\mathcal{A},\mathtt{ABID}}(\lambda, \mathcal{U})$ of an adversary $\mathcal{A}$ defines the game of *concurrent attack* on ABID.

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

In the experiment, $\mathcal{A}$ issues key-extraction queries to its key-generation oracle $\mathcal{KG}$. Giving an attribute set S_i, $\mathcal{A}$ queries $\mathcal{KG}(\mathrm{PK}, \mathrm{MSK}, \cdot)$ for the secret key SK_{S_i}. In addition, $\mathcal{A}$ invokes provers

[7] Our key-generation algorithm **KG** is a tag-version compared with the one in [2].

$\mathrm{P}_j(\mathrm{PK}, \mathrm{SK}_\cdot)$, $j = 1, \ldots, q'_\mathrm{p}, \ldots, q_\mathrm{p}$, by giving a pair (S_j, f_j) of an attribute set and an access formula. Acting as a verifier with an access formula f_j, $\mathcal{A}$ interacts with each $\mathrm{P}_j(\mathrm{PK}, \mathrm{SK}_{S_j})$ concurrently.

The access formula f^* declared by $\mathcal{A}$ is called a *target access formula*. Here we consider the *adaptive* target in the sense that $\mathcal{A}$ is allowed to choose f^* after seeing PK, issuing key-extraction queries and interacting with of provers. Two restrictions are imposed on $\mathcal{A}$ concerning f^*. In key-extraction queries, each attribute set S_i must satisfy $f^*(S_i) = 0$. In interactions with each prover, $f^*(S_j) = 0$. The number of key-extraction queries and the number of invoked provers are at most q_k and q_p in total, respectively, which are bounded by a polynomial in λ.

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

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

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

The game of *passive attack* on ABID is obtained by replacing concurrent provers $\mathcal{P}_j(\mathrm{PK}, \mathrm{SK}_\cdot)|_{j=1}^{q_\mathrm{p}}$ with transcript oracle Transc in $\mathbf{Exprmt}^{\mathrm{ca}}_{\mathcal{A},\mathtt{ABID}}(\lambda,\mathcal{U})$. The following experiment $\mathbf{Exprmt}^{\mathrm{pa}}_{\mathcal{A},\mathtt{ABID}}(\lambda,\mathcal{U})$ of an adversary $\mathcal{A}$ defines the game of *passive attack* on ABID.

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

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

The *advantage* $\mathbf{Adv}^{\mathrm{pa}}_{\mathcal{A},\mathtt{ABID}}(\lambda)$ and security are defined in the same way as the concurrent case. Concurrent security means passive security; for any PPT $\mathcal{A}$, there exists a PPT $\mathcal{B}$ that satisfies the following inequality.

$$\mathbf{Adv}^{\mathrm{pa}}_{\mathcal{A},\mathtt{ABID}}(\lambda) \leq \mathbf{Adv}^{\mathrm{ca}}_{\mathcal{B},\mathtt{ABID}}(\lambda). \tag{2}$$

Attribute Privacy of ABID Consider the following experiment $\mathbf{Exprmt}^{\text{att-prv}}_{\mathcal{A},\mathtt{ABID}}(\lambda,\mathcal{U})$. (In the experiment, an adversary $\mathcal{A}$ interacts with $\mathcal{P}(\mathrm{PK}, \mathrm{SK}_{S_b})$ as a verifier with f^*.)

$\mathbf{Exprmt}^{\text{att-prv}}_{\mathcal{A},\mathtt{ABID}}(\lambda,\mathcal{U})$:
$(\mathrm{PK}, \mathrm{MSK}) \leftarrow \mathbf{Setup}(\lambda,\mathcal{U}), (S_0, S_1, f^*) \leftarrow \mathcal{A}(\mathrm{PK})$
s.t. $(f^*(S_0) = f^*(S_1) = 1) \vee (f^*(S_0) = f^*(S_1) = 0)$
$\mathrm{SK}_{S_0} \leftarrow \mathbf{KG}(\mathrm{PK}, \mathrm{MSK}, S_0), \mathrm{SK}_{S_1} \leftarrow \mathbf{KG}(\mathrm{PK}, \mathrm{MSK}, S_1)$
$b \leftarrow \{1, 0\}, \hat{b} \leftarrow \mathcal{A}^{\mathcal{P}(\mathrm{PK},\mathrm{SK}_{S_b})}(\mathrm{PK}, \mathrm{SK}_{S_0}, \mathrm{SK}_{S_1})$
If $b = \hat{b}$ Return Win else Return Lose

We say that ABID has *attribute privacy* if, for any PPT $\mathcal{A}$, the following advantage is negligible in λ.

$$\mathbf{Adv}^{\text{att-prv}}_{\mathcal{A},\mathtt{ABID}}(\lambda) \stackrel{\mathrm{def}}{=} |\Pr[\mathbf{Exprmt}^{\text{att-prv}}_{\mathcal{A},\mathtt{ABID}}(\lambda,\mathcal{U}) \text{ returns } \textsc{Win}] - 1/2|.$$

C Our Attribute-Based Identification Scheme

In this section, we will describe (verifier-policy) attribute-based identification schemes.

C.1 Our ABID

By combining our Σ-protocol of boolean proof, $\mathbf{\Sigma}_f$, with a credential bundle scheme of the Fiat-Shamir signature $\mathrm{FS}(\mathbf{\Sigma})$, we obtain an attribute-based identification scheme $\mathtt{ABID}$ of proof of knowledge, which has collusion resistance against collecting private secret keys. $\mathtt{ABID}$ has a feature that it can be constructed without pairings.

Fig. 5 shows our construction of ABID scheme, $\mathtt{ABID} = (\mathbf{Setup}, \mathbf{KG}, \mathcal{P}, \mathcal{V})$.

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

$\mathbf{KG}(\mathrm{PK}, \mathrm{MSK}, \tau, S)$:
$k \leftarrow \mathit{Hash}_\mu(w_i \parallel \tau)$
For $i \in S$
$m_i := (\tau \parallel i)$
$\mathrm{FS}(\mathbf{\Sigma})^{\mathrm{sign}}(x_{\mathrm{mst}}, w_{\mathrm{mst}}, m_i; \mathit{PRF}_k(m_i \parallel 0))$
$\rightarrow (a_i, w_i)$
$\mathrm{SK}_S := (\tau, k, ((a_i, w_i))_{i \in S})$
Return SK_S

$\mathcal{P}(\mathrm{PK}, \mathrm{SK}_S)$:
Initialize inner state

$\mathcal{V}(\mathrm{PK}, f)$:
Initialize inner state

$\xleftarrow{f}$

$\mathbf{\Sigma}_f^{\mathrm{eval}}(\mathcal{T}_f, S) \rightarrow (v_{\mathrm{nd}})_{\mathrm{nd} \in \mathrm{Nd}(\mathcal{T}_f)}$
If $v_{\mathrm{rt}(\mathcal{T}_f)} \neq 1$, then abort
else $\mathrm{CHA}_{\mathrm{rt}(\mathcal{T}_f)} := *$
$\mathbf{Supp}(\mathrm{PK}, \mathrm{SK}_S, f) \rightarrow ((a_{i_j}, w_{i_j}))_{1 \leq j \leq \mathrm{arity}(f)}$
$\mathbf{StmtGen}(\mathrm{PK}, \tau, (a_{i_j})_{1 \leq j \leq \mathrm{arity}(f)})$
$\rightarrow (x_{i_j})_{1 \leq j \leq \mathrm{arity}(f)} =: x$
$w := (w_{i_j})_{1 \leq j \leq \mathrm{arity}(f)}$
$\mathbf{\Sigma}_f^1(x, w, \mathcal{T}_f, (v_{\mathrm{nd}})_{\mathrm{nd} \in \mathrm{Nd}(\mathcal{T}_f)}, \mathrm{CHA}_{\mathrm{rt}(\mathcal{T}_f)})$
$\rightarrow ((\mathrm{CMT}_{\mathrm{lf}})_{\mathrm{lf} \in \mathrm{Lf}(\mathcal{T}_f)}, (\mathrm{CHA}_{\mathrm{nd}})_{\mathrm{nd} \in \mathrm{tNd}(\mathcal{T}_f)}, (\mathrm{RES}_{\mathrm{lf}})_{\mathrm{lf} \in \mathrm{Lf}(\mathcal{T}_f)})$

$\tau, (a_{i_j})_{1 \leq j \leq \mathrm{arity}(f)}$
$(\mathrm{CMT}_{\mathrm{lf}})_{\mathrm{lf} \in \mathrm{Lf}(\mathcal{T}_f)}$
$\longrightarrow$

$\mathrm{CHA}_{\mathrm{rt}(\mathcal{T}_f)} := \mathrm{CHA}$
$\mathbf{\Sigma}_f^3(x, w, \mathcal{T}_f, (v_{\mathrm{nd}})_{\mathrm{nd} \in \mathrm{Nd}(\mathcal{T}_f)}, (\mathrm{CMT}_{\mathrm{lf}})_{\mathrm{lf} \in \mathrm{Lf}(\mathcal{T}_f)}, \mathrm{CHA}_{\mathrm{rt}(\mathcal{T}_f)}, (\mathrm{CHA}_{\mathrm{nd}})_{\mathrm{nd} \in \mathrm{tNd}(\mathcal{T}_f)}, (\mathrm{RES}_{\mathrm{lf}})_{\mathrm{lf} \in \mathrm{Lf}(\mathcal{T}_f)})$
$\rightarrow ((\mathrm{CHA}_{\mathrm{nd}})_{\mathrm{nd} \in \mathrm{tNd}(\mathcal{T}_f)}, (\mathrm{RES}_{\mathrm{lf}})_{\mathrm{lf} \in \mathrm{Lf}(\mathcal{T}_f)})$

CHA
$\longleftarrow$
$(\mathrm{CHA}_{\mathrm{nd}})_{\mathrm{nd} \in \mathrm{tNd}(\mathcal{T}_f)}, (\mathrm{RES}_{\mathrm{lf}})_{\mathrm{lf} \in \mathrm{Lf}(\mathcal{T}_f)}$
$\longrightarrow$

$\mathrm{CHA} \leftarrow \mathbf{\Sigma}_f^2(\lambda)$
$\mathbf{StmtGen}(\mathrm{PK}, \tau, (a_{i_j})_{1 \leq j \leq \mathrm{arity}(f)})$
$\rightarrow (x_{i_j})_{1 \leq j \leq \mathrm{arity}(f)} =: x$
$\mathbf{\Sigma}_f^{\mathrm{vrfy}}(x, \mathcal{T}_f, (\mathrm{CMT}_{\mathrm{lf}})_{\mathrm{lf} \in \mathrm{Lf}(\mathcal{T}_f)}, \mathrm{CHA}, (\mathrm{CHA}_{\mathrm{nd}})_{\mathrm{nd} \in \mathrm{tNd}(\mathcal{T}_f)}, (\mathrm{RES}_{\mathrm{lf}})_{\mathrm{lf} \in \mathrm{Lf}(\mathcal{T}_f)}) \rightarrow b$
Return b

Fig. 5. Our $\mathtt{ABID}$ scheme.

C.2 Security of Our ABID

Theorem 4 *If the employed signature scheme FS($\mathbf{\Sigma}$) is secure in the game of existential unforgeability against chosen-message attacks, then our $\mathtt{ABID}$ is secure against concurrent attacks. More*

precisely, for any PPT algorithm $\mathcal{A}$, there exists a PPT algorithm $\mathcal{F}$ which satisfies the following inequality ($neg(\cdot)$ means a negligible function).

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

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

Proof. Employing any given adversary $\mathcal{A}$ as subroutine, we construct a signature forger $\mathcal{F}$ as follows. $\mathcal{F}$ can answer to $\mathcal{A}$'s key-extraction queries for a secret key SK_S because $\mathcal{F}$ can query his signing oracle about $(\tau \parallel i; i \in S)$, where $\mathcal{F}$ choose τ at random. $\mathcal{F}$ can simulate any concurrent prover with SK_S which $\mathcal{A}$ invokes because $\mathcal{F}$ can generate SK_S in the above way. After the learning phase above, $\mathcal{F}$ simulates a verifier with which $\mathcal{A}$ begins to interact as a prover. There $\mathcal{F}$ rewinds $\mathcal{A}$ once and $\mathcal{F}$ can obtain a witness set w^* by running $\mathbf{\Sigma}^{\mathrm{KE}}_f$, as well as the set of attributes S^* where the rewinding works. We use here the Reset Lemma [6], which reduces the advantage $\mathbf{Adv}^{\mathrm{ca}}_{\mathcal{A},\mathtt{ABID}}(\lambda)$ to $\mathbf{Adv}^{\text{euf-cma}}_{\mathcal{F},\mathrm{FS}(\mathbf{\Sigma})}(\lambda)$ with a loss of exponent by $1/2$. Finally $\mathcal{F}$ converts (w^*, S^*) into SK_{S^*}. □

C.3 Discussion

Attribute-Based Proof of Knowledge Our ABID is a proof of knowledge system. That is, for a fixed access policy f, a PPT knowledge extractor can be constructed, which extracts a secret key SK_{S^*} for some attribute set S^* with $f(S^*) = 1$.

Attribute Privacy of our ABID. For the case of an honest verifier, that is, if an adversary $\mathcal{A}$ chooses a challenge Cha uniformly at random from $\textsc{ChaSp}(\lambda)$ in the attribute privacy game in Section B , attribute privacy follows from the honest verifier zero-knowledge property. However, in general, attribute privacy is not obvious. It seems an open problem to the best of authors' knowledge.

Security Reduction. We mean a number theoretic problem here as the discrete-logarithm problem or the RSA-inverse problem ([6]).

There exists the following security reduction to a number theoretic problem.

$$\mathbf{Adv}^{\mathrm{ca}}_{\mathcal{A},\mathtt{ABID}}(\lambda) \leq q_H^{1/2}(\mathbf{Adv}^{\mathrm{num.prob.}}_{\mathcal{S},\mathbf{Grp}}(\lambda))^{1/4} + \mathrm{neg}(\lambda). \tag{3}$$

Here we denote q_H as the maximum number of hash queries issued by forger $\mathcal{F}$ on $\mathrm{FS}(\mathbf{\Sigma})$ in the random oracle model.

Proof. As is discussed in Section 2.1, we can reduce the advantage $\mathbf{Adv}^{\text{euf-cma}}_{\mathcal{F},\mathrm{FS}(\mathbf{\Sigma})}(\lambda)$ to the advantage $\mathbf{Adv}^{\mathrm{pa}}_{\mathcal{B},\mathbf{\Sigma}}(\lambda)$ of passive security of the underlying Σ-protocol, in the random oracle model, with a loss factor q_H. Applying the Reset Lemma [6], we can reduce $\mathbf{Adv}^{\mathrm{pa}}_{\mathcal{B},\mathbf{\Sigma}}(\lambda)$ to the advantage $\mathbf{Adv}^{\mathrm{num.prob.}}_{\mathcal{S},\mathbf{Grp}}(\lambda)$ of a PPT solver $\mathcal{S}$ of a number theoretic problem, with a loss of exponent by $1/2$. □