Boolean Formula-Proof and Its Application to Attribute-Based Identifications and Signatures

Hiroaki Anada[1,2,a)] Seiko Arita[2,b)] Kouichi Sakurai[1,3,c)]

Abstract: We propose a notion of *boolean formula-proof system (f-proof)* in the form of a Σ-protocol. Our f-proof system is a generalization of the well-known Σ-protocol of OR-proof system; that is, it treats any boolean formula (which does not have a NOT-gate) instead of a single OR-gate. In addition, as is the case for usual Σ-protocol, our f-proof system is a proof of knowledge system; that is, it possesses a knowledge extractor that extracts a witness set that satisfies the boolean formula f. Next, we provide a construction of our f-proof system from a given Σ-protocol. Then, by combining our f-proof system with a credential bundle scheme, we obtain an attribute-based identification scheme of proof of knowledge type. It possesses a property of attribute-privacy and a feature that it can be constructed without pairings. Finally, by applying the Fiat-Shamir Transform, we obtain an attribute-based signature scheme, which is secure in the random oracle model.
我々はブール式証明システムの概念を Σ-プロトコルの形で提案する．我々のブール式証明システムはよく知られた OR-証明システムの一般化である．すなわち，単一の OR ゲートに代わって（NOT ゲートを持たない）任意のブール式を取り扱う．加えて，通常の Σ-プロトコルの場合と同じく，我々のブール式証明システムは知識の証明システムである；すなわち，ブール式を満足する証拠の集合を抽出する知識抽出機を有する．次いで，我々は，与えられた Σ-プロトコルから我々のブール式証明システムを構成する方法を与える．そして，我々のブール式証明システムを証明書バンドルスキームと組み合わせることで，我々は知識の証明タイプの属性ベース認証スキームを得る．それは属性プライバシーの性質及びペアリング無しで構成されうる特徴を有する．最後に，Fiat-Shamir 変換を適用することで，我々は属性ベース署名スキームを得る．それはランダムオラクルモデルで安全である．

1. Introduction

The Σ-protocol [7], [8] is a proof system that allows a prover, in a 3-move, to convince a verifier that the prover knows an answer (that is, a witness) of a hard problem instance. The knowledge is assured by the presence of a knowledge extractor, which extracts the witness by employing the prover as a subroutine. In that sense, the Σ-protocol is a proof of knowledge system. In addition, the Σ-protocol is known to be a zero-knowledge proof system against honest verifiers. That is, after a whole interaction, an honest verifier knows nothing but only the fact that the prover knows the witness. That Σ-protocol, which has those distinguished properties, is concretely realized as the Schnorr protocol [13] and the Guillou-Quisquater protocol (which are the origin of the Σ-protocol).

An extended notion, the Σ-protocol of OR-proof, has been developed [8] and applied so as to achieve, for example, man-in-the-middle security (Katz02). In the OR-proof, there are two instances and a prover can convince a verifier that the prover knows a witness of one instance, or another, or both.

In this paper, we propose and construct a Σ-protocol of *boolean formula-proof (f-proof)*, which is a generalization of the Σ-protocol of OR-proof. In our Σ-protocol of f-proof, f is a boolean formula written in boolean variables and boolean connectives (AND gates and OR gates). There is also a set of instances. Then a prover can convince a verifier that the prover knows a set of witness (witness set) that satisfies the formula f. Here each boolean variable appearing in f take 1 or 0 according to whether the prover knows the corresponding witness or not.

Our Σ-protocol of f-proof has interesting application to *attribute-based identification of proof of knowledge (ABID of PoK)* and attribute-based signatures (ABS).

Let us first remember the notion of ABID. Attribute-based identification (ABID) was introduced in [2]. In an ABID scheme, a prover has credentials called attributes. On the other hand, a verifier maintains an access policy written as a boolean formula over those attributes. Then, a verifier can identify that a prover possesses a set of attributes that satisfies the verifier's access policy.

Hence an ABID of PoK scheme is defined as an ABID scheme with a *knowledge extractor* that extracts a witness set that satisfies the verifier's access policy.

Is is notable that almost all attribute-based cryptographic primitives so far (originated in [10], [14]) are constructed basecally with linear secret sharing schemes (LSSS)([3]) and pairing maps ([6]). Our approach is different from that construction.

1.1 Our Idea and Contributions

To construct a Σ-protocol if f-proof, we will look into the OR-proof technique [8] and expand it so as to treat any boolean formula (without negations), as follows.

First express a boolean formula (an access formula) as a binary tree (an access tree); that is, with its inner nodes being AND gates and OR gates and its leaf nodes being terms that map to attributes. An verification equation of Σ is assigned to each leaf node. Sup-

1 Institute of Systems, Information Technologies and Nanotechnologies (ISIT), Fukuoka SRP Center Building 7F, Momochihama 2-1-22, Sawara-ku, Fukuoka, 814-0001, Japan
2 Graduate School of Information Security, Institute of Information Security, 2-14-1 Tsuruyacho, Kanagawa-ku, Yokohama, 221-0835, Japan
3 Graduate School and Faculty of Information Science and Electrical Engineering, Kyushu University, Room 712, West Bldg. No.2, Moto-oka 744, Nishi-ku, Fukuoka, 819-0395, Japan
a) anada@isit.or.jp
b) arita@iisec.ac.jp
c) sakurai@csce.kyushu-u.ac.jp

pose that a challenge string CHA of $\boldsymbol{\Sigma}$ is given, and then assign 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 OR gate, divide CHA into two random strings CHA_L and CHA_R which satisfy $\mathrm{CHA} = \mathrm{CHA}_\mathrm{L} \oplus \mathrm{CHA}_\mathrm{R}$, and assign CHA_L and CHA_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 we can either honestly execute Σ-protocol $\boldsymbol{\Sigma}$ or execute $\boldsymbol{\Sigma}$ in a simulated way. Only when a set of attributes satisfies the access tree the above procedure succeeds in satisfying verification equations for each leaf node. We call the above procedure *boolean formula-proof*. The boolean formula-proof is a natural, but non-trivial extension of the OR-proof Σ-protocol.

As our theoretical contribution, we provide a Σ-protocol of boolean formula-proof, $\boldsymbol{\Sigma}_f$. Given a boolean formula f without negation and a Σ-protocol $\boldsymbol{\Sigma}$, we generically construct $\boldsymbol{\Sigma}_f$ so that $\boldsymbol{\Sigma}_f$ is a Σ-protocol to prove knowledge of a witness set that satisfies f.

As our practical contribution, we provide ABID of PoK schemes and ABS schemes *without* pairings. More precisely, As a Σ-protocol $\boldsymbol{\Sigma}$ can be constructed without pairings (for example, the case of Schnorr scheme [5], [13]), our generic construction results in a concrete boolean formula-proof $\boldsymbol{\Sigma}_f$ without pairings, and hence, an ABID of PoK scheme and an ABS scheme, without pairings, whose security are in the random oracle model.

1.2 Related Work

The OR-proof is precisely explained in, for example, [8]. The OR-proofs have applications such as achieving security against man-in-the-middle adversary. Our boolean formula-proof can be seen as a natural extension of the OR-proof.

ABID is introduced by Anada et al. in [2]. Then they provided a challenge-and-response construction of ABID scheme using attribute-based key encapsulation mechanism. In contrast, we provide three move protocol with knowledge-extraction property.

1.3 Organization of this Paper

In Section 2, we prepair for the needed tools and notions. In Section 3, we give a definiton a Σ-protocol of boolean formula-proof. In Section 4, we describe the construction of our Σ-protocol of boolean formula-proof, $\boldsymbol{\Sigma}_f$. In Section 5, we modify our $\boldsymbol{\Sigma}_f$ into a ABID of PoK scheme. In Section 7, we show how concretely our ABIDofPoK is realized in discrete logarithm-setting and RSA setting. In Section 6, we apply the Fiat-Shamir Transform to our our ABIDofPoK scheme and obtain a ABS scheme. In Section 8, we conclude our work in this paper.

2. Preliminaries

The security parameter is denoted by λ. When an algorithm A with input a outputs z, we denote it as $z \leftarrow A(a)$, or, because of space limitation, $A(a) \to z$. 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 O, we denote it as A^O. When A has concurrent oracle-access to n oracles $O_1, \ldots, O_n$, we denote it as $A^{O_i|_{i=1}^n}$. Here "concurrent" means that A accesses to oracles in arbitrarily interleaved order of messages.

2.1 Access Formula [10]

Let $\mathcal{U} = \{\mathrm{att}_1, \ldots, \mathrm{att}_U\}$ be an attribute universe. We must distinguish two cases: the case that $\mathcal{U}$ is small (that is, $|\mathcal{U}| = U$ is bounded by some 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.

Suppose that we are given an access policy as a boolean formula f over variables $\{X_{\mathrm{att}}\}_{\mathrm{att}\in\mathcal{U}}$ (for example, $f = X_{\mathrm{att}_1} \wedge ((X_{\mathrm{att}_2} \wedge X_{\mathrm{att}_3}) \vee X_{\mathrm{att}_4})$). That is, each term X_{att} is a predicate which, on input $S \in 2^{\mathcal{U}}$, takes a boolean value according to whether $\mathrm{att} \in S$ or not. Then the boolean output of f at S is evaluated according to boolean connectives, that is, AND gates ($\wedge$-gates) or OR gates ($\vee$-gates). Hence f is a function: $f : 2^{\mathcal{U}} \to \{1, 0\}$. We call the boolean formula f an *access formula* over $\mathcal{U}$.

In this paper, we do not consider NOT gates ($\neg$), that is, we only consider monotone boolean formulas[*1].

2.1.1 Access Tree

We consider in this paper a finte 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 leaf node corresponds to a term X_{att} in f in one-to-one way. Each inner node represents an operator, $\wedge$-gate or $\vee$-gate, in f. An attribute map $\mathrm{att}(\cdot) : \mathrm{Lf}(\mathcal{T}) \to \mathcal{U}$ is defined, for $\mathrm{lf} \in \mathrm{Lf}(\mathcal{T})$, as $\mathrm{att}(\mathrm{lf}) :=$ (the attribute att of the term X_{att}).

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.

2.2 Proof of Knowledge System [4], [8]

Let R be a binary polynomial-time relation $R = \{(x, w)\} \subset \{0,1\}^* \times \{0,1\}^*$. x is called a *statement* and w is called a *witness* for the statement x. We assume that, given a statement x as an input, any PPT algorithm can only output a witness w satisfying $(x, w) \in R$ with a negligible probability. If this assumption holds, we say that R is a hard relation ([8]).

A *proof of knowledge system* (PoK for short) is a protocol between interactive PPT algorhtms P and V on initial input $(x, w) \in R$ for P and x for V.

Completeness. An honest prover P with a legitimate witness w can make V accept with probability 1.

Knowledge Soundness. There is a PPT algorithm called a *knowledge extractor*, which, given a statement x and employing P as a subroutine, can compute a witness w satisfying $(x, w) \in R$ with at most a negligible error probability.

2.3 Σ-protocol [7], [8]

Let R be a binary polynomial-time relation $R = \{(x, w)\} \subset \{0,1\}^* \times \{0,1\}^*$.

A Σ-*protocol* on a relation R is a 3-move protocol between interactive PPT algorhtms P and V on initial input $(x, w) \in R$ for P and x for V. P sends the first message called a commitment CMT, then V sends a random bit string called a challenge CHA, and P answers with a third message called a response RES. Then V applies a local decision test on $(x, \mathrm{CMT}, \mathrm{CHA}, \mathrm{RES})$ to return accept (1) or reject (0).

[*1] 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).

Cha is chosen at random from ChaSp := $\{1,0\}^{l(\lambda)}$ with $l(\cdot)$ being a super-log function.

This protocol is written by a PPT algorithm $\mathbf{\Sigma}$ as follows. 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 Cha $\leftarrow \mathbf{\Sigma}^2(\lambda)$, Res $\leftarrow \mathbf{\Sigma}^3(x,w,\text{Cmt},\text{Cha})$ and $b \leftarrow \mathbf{\Sigma}^{\text{vrfy}}(x,\text{Cmt},\text{Cha},\text{Res})$.

Σ-protocol satisfies three constraints:

Completeness. An honest prover P with a legitimate witness w can make V accept with probability 1.

Special Soundness. Any PPT algorithm 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*, $\mathbf{\Sigma}^{\text{ke}}$, which, given a statement x and using 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 (Cmt, Cha, Res) and (Cmt, Cha′, Res′) with Cha ≠ Cha′.

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

Σ-protocols are known to be proofs of knowledge ([8]).

We will use in this paper a property that not only a prover but also a verifier can compute a new statement y on input $(x,\text{Cmt},\text{Cha})$ in polynomial-time, which makes Res the corresponding witness. We denote the algorithm as $\mathbf{\Sigma}^{\text{stmtforRes}}$;

$$y \leftarrow \mathbf{\Sigma}^{\text{stmtforRes}}(x,\text{Cmt},\text{Cha}) \text{ s.t. } (y, {}^{\exists}\text{Res}) \in R,$$
$$\wedge\ (\text{Cmt},\text{Cha},\text{Res}) : \text{an accepting conversation.}$$

Known Σ-protocols such as the Schnorr protocol and the Guillou-Quisquater protocol [5], [13] possess this property.

2.3.1 The OR-proof [8]

Suppose that a Σ-protocol $\mathbf{\Sigma}$ on a relation R is given. Consider the following relation.

$$R_{\text{OR}} = \{(\{x_0,x_1\},\{w_0,w_1\}); (x_0,w_0) \in R \vee (x_1,w_1) \in R\}$$

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

2.3.2 Fiat-Shamir Transform [1]

Employing a cryptographic hash function with collision resistance, $H_\lambda(\cdot) : \{1,0\}^* \to \{1,0\}^{l(\lambda)}$, a Σ-protocol $\mathbf{\Sigma}$ can be transformed into a digital signature scheme as follows. Given a message m, execute: $a \leftarrow \mathbf{\Sigma}^1(x,w)$, $c \leftarrow H_\lambda(m \parallel a)$, $z \leftarrow \mathbf{\Sigma}^3(x,w,a,c)$. Then $\sigma := (a,z)$ is a signature on m. We denote the above signing algorithm as $\text{FS}^{\text{sign}}_{\mathbf{\Sigma}}(\lambda; x,w,m) \to (a,z) =: \sigma$. The verification algorithm is: $\text{FS}^{\text{vrfy}}_{\mathbf{\Sigma}}(x,m,\sigma) : c \leftarrow H_\lambda(m \parallel a)$, $b \leftarrow \mathbf{\Sigma}^{\text{vrfy}}(x,a,c,z)$, return b.

The signature scheme $\text{FS}_{\mathbf{\Sigma}} = (R, \text{FS}^{\text{sign}}_{\mathbf{\Sigma}}, \text{FS}^{\text{vrfy}}_{\mathbf{\Sigma}})$ is known to be *extentially unforgeable against chosen-message attacks* if and only if underlying Σ-protocol $\mathbf{\Sigma}$ is secure against *passive attacks* [1].

2.4 Credential-Bundle [12]

Credential-bundle is an extended notion of digital signature. Suppose that we are given a digital signature scheme (KG, Sign, Vrfy). To construct a credential-bundle scheme, first choose a string τ, a *tag*. τ can be chosen as a random string or a publicly known string such as an e-mail address. Then, for a set of messages $m_1,\ldots,m_n$, execute Sign on each *tagged message* $(\tau \parallel m_i), i = 1,\ldots,n$. Verify is applied in a obvious way.

2.5 Attribute-Based Identification Scheme [2]

In this paper, we will describe *verifier-policy* attribute-based identification schemes [2].

2.5.1 Scheme

An attribute-based identification scheme, ABID, consists of four PPT algorithms: (Setup, KeyGen, P, V).

Setup$(\lambda,\mathcal{U}) \to$ **(PK, 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.

KeyGen$(\mathbf{PK},\mathbf{MSK},S) \to \mathbf{SK}_S$**.** A key-generation algorithm KeyGen takes as input the public key PK, the master secret key MSK and an attribute set $S \subset \mathcal{U}$. It outputs a secret key SK_S corresponding to S.

P$(\mathbf{PK},\mathbf{SK}_S)$ **and V**$(\mathbf{PK},f)$**.** P and V are interactive algorithms called a *prover* and a *verifier*, respectively. P takes as input the public key PK and the secret key SK_S. Here the secret key SK_S is given to P by an authority that runs KeyGen(PK,MSK,S). V takes as input the public key PK and an access formula f. P is provided V's access formula f by the first round. P and V interact with each other for some, at most constant rounds. Then, V finally returns its decision 1 or 0. 1 means that V *accepts* P in the sense P has a secret key SK_S such that $f(S) = 1$. 0 means that V *rejects* P. We demand correctness of ABID that for any λ and $\mathcal{U}$, and if $f(S) = 1$, then $\Pr[(\text{PK},\text{MSK}) \leftarrow \text{Setup}(\lambda,\mathcal{U}); \text{SK}_S \leftarrow \text{KeyGen}(\text{PK},\text{MSK},S); b \leftarrow \langle \text{P}(\text{PK},\text{SK}_S), \text{V}(\text{PK},f)\rangle : b = 1] = 1$.

2.5.2 Concurrent Attack on ABID and Security

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

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

In the experiment, $\mathcal{A}$ issues key-extraction queries to the key-generation oracle $\mathcal{KG}$. Giving an attribute set S_i, $\mathcal{A}$ queries $\mathcal{KG}(\text{PK},\text{MSK},\cdot)$ for the secret key SK_{S_i}. We do not require any two input, S_{i_1} and S_{i_2}, to be distinct. On the other hand, the adversary $\mathcal{A}$ invokes provers $\text{P}_j(\text{PK},\text{SK}_\cdot)$, $j = 1,\ldots,q'_\text{p},\ldots,q_\text{p}$, by giving an attribute set S_j of $\mathcal{A}$'s choice. Acting as a verifier with an access formula f_j, $\mathcal{A}$ interacts with each P_j.

The access formula f^* declared by $\mathcal{A}$ is called a *target access formula*. Here we consider an adaptive target access formula f^* in the sense that $\mathcal{A}$'s declaration of f^* is not limited before seeing the public key PK. Two restrictions are imposed on $\mathcal{A}$ concerning f^*. In key-extraction queries, each attribute set S_i must satisfy $S_i \notin f^*$.

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 $\mathcal{A}$ and for any attribute universe $\mathcal{U}$, $\mathbf{Adv}^{\mathrm{ca}}_{\mathcal{A},\mathtt{ABID}}(\lambda)$ is negligible in λ.

2.5.3 Anonymity

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

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

We say that ABID have *anonymity* if, for any PPT $\mathcal{A}$ and for any $\mathcal{U}$, the following advantage of $\mathcal{A}$ is negligible in λ.

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

2.5.4 Attribute-Based Identification Scheme of Proof of Knowledge

Let f be an access formula. An attribute-based identification scheme of proof of knowledge is an attribute-based identification scheme which satisfies the following knowledge soundness.

Knowledge Soundness. There is a PPT algorithm called a *knowledge extractor*, which, given a public key PK and employing P as a subroutine, can compute a secret key SK_S for some $S \subset \mathcal{U}$ satisfying $f(S) = 1$, with at most a negligible error probability.

3. Definition: Σ-protocol of Boolean Formula-Proof

In this section, we define a Σ-protocol of a boolean formula-proof, which is a generalization of the Σ-protocol of OR-proof [8].

Let R be a binary polynomial-time relation $R = \{(x,w)\} \subset \{1,0\}^* \times \{1,0\}^*$. Then we create a new relation R_f:

$$R_f \stackrel{\mathrm{def}}{=} \{(X := \{x_{\mathrm{att}}\}, W := \{w_{\mathrm{att}}\}) \in \{1,0\}^* \times \{1,0\}^*; f(S := \{\mathrm{att}; (x_{\mathrm{att}}, w_{\mathrm{att}}) \in R\}) = 1\}. \quad (1)$$

Then a Σ-*protocol of* f-*proof* on the relation R_f is defined as a 3-move protocol between interactive PPT algorhtms P and V on initial input $(X := \{x_{\mathrm{att}}\}, W := \{w_{\mathrm{att}}\}) \in R_f$ for P, and X for V. P sends the first message called a commitment Cmt, then V sends a random bit string called a challenge Cha, and P answers with a third message called a response Res. Then V applies a local decision test on $(X, \mathrm{Cmt}, \mathrm{Cha}, \mathrm{Res})$ to return accept (1) or reject (0). Here Cha is chosen at random from $\mathrm{ChaSp} := \{1,0\}^{l(\lambda)}$ with $l(\cdot)$ being a super-log function.

This protocol is written by a PPT algorithm $\mathbf{\Sigma}_f$ as follows. $\mathrm{Cmt} \leftarrow \mathbf{\Sigma}^1_f(X,W)$: the process of selecting the first message Cmt according to the protocol $\mathbf{\Sigma}_f$ on input $(X,W) \in R_f$. Similarly we denote $\mathrm{Cha} \leftarrow \mathbf{\Sigma}^2_f(\lambda)$, $\mathrm{Res} \leftarrow \mathbf{\Sigma}^3_f(X,W,\mathrm{Cmt},\mathrm{Cha})$ and $b \leftarrow \mathbf{\Sigma}^{\mathrm{vrfy}}_f(X,\mathrm{Cmt},\mathrm{Cha},\mathrm{Res})$.

To be a Σ-protocol, $\mathbf{\Sigma}_f$ must satisfy three constraints:

Completeness An honest prover P with a legitimate witness set W can make V accept with probability 1.

Special Soundness. Any PPT algorithm P^* without any witness set, a cheating prover, can only respond for one possible challenge Cha. In other words, there is a PPT algorithm called a *knowledge extractor*, $\mathbf{\Sigma}^{\mathrm{ke}}_f$, which, given a statement X and using P^* as a subroutine, can compute a witness set W satisfying $(X,W) \in R_f$ 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 set X and a *random* challenge $\mathrm{Cha} \in \mathrm{ChaSp}$, we can produce (in polynomial-time) an accepting conversation $(\mathrm{Cmt},\mathrm{Cha},\mathrm{Res})$, with the same distribution of real accepting conversations, without knowing the witness set W. In other words, there is a PPT algorithm called a *simulator*, $\mathbf{\Sigma}^{\mathrm{sim}}_f$, such that $(\mathrm{Cmt},\mathrm{Res}) \leftarrow \mathbf{\Sigma}^{\mathrm{sim}}_f(X,\mathrm{Cha})$. Hohest-verifier zero-knowledge implies zero-knowledge with respect to the honest verifier.

A Σ-protocol of f-proof can be proved to be a proof of knowledge system.

4. Construction of a Σ-protocol of Boolean Formula-Proof

In this section, from a given Σ-protocol $\mathbf{\Sigma}$ and an access formula f, we construct a Σ-protocol $\mathbf{\Sigma}_f$ of f-proof.

The outline of our protocol $\mathbf{\Sigma}_f$ is described in Fig. 1. Basically $\mathbf{\Sigma}_f$ is a 3-move protocol between interactive PPT algorhtms P and V on initial input $(X := \{x_{\mathrm{att}}\}, W := \{w_{\mathrm{att}}\}) \in R_f$ for P and X for V.

Satisfiability Evaluation. A prover begins with satisfiability evaluation concerning $f(S)$; we label each node of $\mathcal{T}$ with a value v (True(1) or False(0)) according to whether or not $S \in \mathrm{att}(f)$ holds at each leaf node and AND or OR holds at each inner node. This computation is described as follows.

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

Commitment. Prover's computation of a commitment value for each leaf node is described in Fig. 2. Basically, the algorithm $\mathbf{\Sigma}^1_f$ is a executed recursively.

Challenge. Verifier picks up a challenge value as follows.

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

Response. Prover's computation of a response value for each leaf node is described in Fig. 3. Basically, the algorithm $\mathbf{\Sigma}^3_f$ is executed recursively.

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

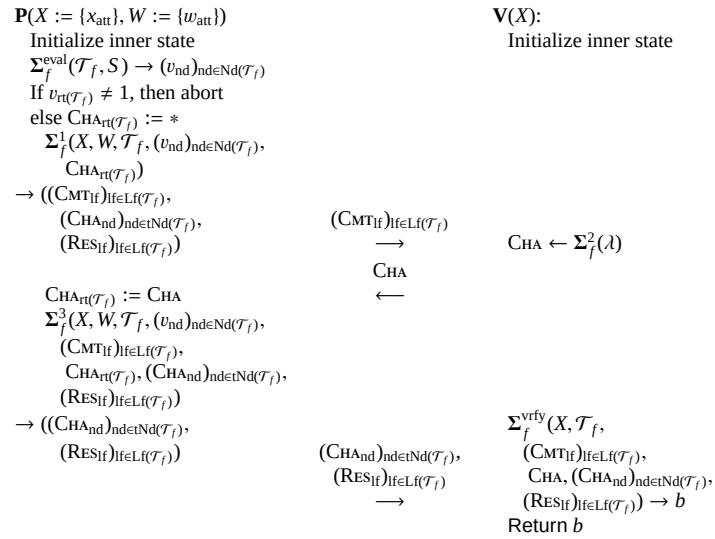

Fig. 1 Our Σ-protocol $\mathbf{\Sigma}_f$ of boolean formula-proof on a relation $R_f = \{(X, W)\}$.

$\mathbf{\Sigma}_f^1(X, W, \mathcal{T}, (v_{\mathrm{nd}})_{\mathrm{nd}\in\mathrm{Nd}(\mathcal{T})}, \mathrm{CHA})$:
$\mathcal{T}_{\mathrm{L}} := \mathrm{Lsub}(\mathcal{T}), \mathcal{T}_{\mathrm{R}} := \mathrm{Rsub}(\mathcal{T})$
If rt($\mathcal{T}$) is an $\wedge$-node, then $\mathrm{CHA}_{\mathrm{L}} := \mathrm{CHA}, \mathrm{CHA}_{\mathrm{R}} := \mathrm{CHA}$
Return($\mathrm{CHA}_{\mathrm{L}}, \mathrm{CHA}_{\mathrm{R}}, \mathbf{\Sigma}_f^1(X, W, \mathcal{T}_{\mathrm{L}}, (v_{\mathrm{nd}})_{\mathrm{nd}\in\mathrm{Nd}(\mathcal{T}_{\mathrm{L}})}, \mathrm{CHA}_{\mathrm{L}}), \mathbf{\Sigma}_f^1(X, W, \mathcal{T}_{\mathrm{R}}, (v_{\mathrm{nd}})_{\mathrm{nd}\in\mathrm{Nd}(\mathcal{T}_{\mathrm{R}})}, \mathrm{CHA}_{\mathrm{R}}))$
else if rt($\mathcal{T}$) is an $\vee$-node, then
If $v_{\mathrm{rt}(\mathcal{T}_{\mathrm{L}})} = 1 \wedge v_{\mathrm{rt}(\mathcal{T}_{\mathrm{R}})} = 1$, then $\mathrm{CHA}_{\mathrm{L}} := *$, $\mathrm{CHA}_{\mathrm{R}} := *$
else if $v_{\mathrm{rt}(\mathcal{T}_{\mathrm{L}})} = 1 \wedge v_{\mathrm{rt}(\mathcal{T}_{\mathrm{R}})} = 0$, then $\mathrm{CHA}_{\mathrm{L}} := *$, $\mathrm{CHA}_{\mathrm{R}} \leftarrow \mathrm{CHASP}$
else if $v_{\mathrm{rt}(\mathcal{T}_{\mathrm{L}})} = 0 \wedge v_{\mathrm{rt}(\mathcal{T}_{\mathrm{R}})} = 1$, then $\mathrm{CHA}_{\mathrm{L}} \leftarrow \mathrm{CHASP}, \mathrm{CHA}_{\mathrm{R}} := *$
else if $v_{\mathrm{rt}(\mathcal{T}_{\mathrm{L}})} = 0 \wedge v_{\mathrm{rt}(\mathcal{T}_{\mathrm{R}})} = 0$, then $\mathrm{CHA}_{\mathrm{L}} \leftarrow \mathrm{CHASP}, \mathrm{CHA}_{\mathrm{R}} := \mathrm{CHA} \oplus \mathrm{CHA}_{\mathrm{L}}$
Return($\mathrm{CHA}_{\mathrm{L}}, \mathbf{\Sigma}_f^1(X, W, \mathcal{T}_{\mathrm{L}}, (v_{\mathrm{nd}})_{\mathrm{nd}\in\mathrm{Nd}(\mathcal{T}_{\mathrm{L}})}, \mathrm{CHA}_{\mathrm{L}}), \mathrm{CHA}_{\mathrm{R}}, \mathbf{\Sigma}_f^1(X, W, \mathcal{T}_{\mathrm{R}}, (v_{\mathrm{nd}})_{\mathrm{nd}\in\mathrm{Nd}(\mathcal{T}_{\mathrm{R}})}, \mathrm{CHA}_{\mathrm{R}}))$
else if rt($\mathcal{T}$) is a leaf-node, then
If $\mathrm{CHA} = *$, then $\mathrm{CMT} \leftarrow \mathbf{\Sigma}^1(x_{\mathrm{att}(\mathrm{rt}(\mathcal{T}))}), \mathrm{RES} := *$
else if $\mathrm{CHA} \neq *$, then $(\mathrm{CMT}, \mathrm{RES}) \leftarrow \mathbf{\Sigma}^{\mathrm{sim}}(x_{\mathrm{att}(\mathrm{rt}(\mathcal{T}))}, \mathrm{CHA})$
Return($\mathrm{CMT}, \mathrm{RES}$)

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

$\mathbf{\Sigma}_f^{\mathrm{vrfy}}(X, \mathcal{T}, (\mathrm{CMT}_{\mathrm{lf}})_{\mathrm{lf}\in\mathrm{Lf}(\mathcal{T})}, \mathrm{CHA}, (\mathrm{CHA}_{\mathrm{nd}})_{\mathrm{nd}\in\mathrm{tNd}(\mathcal{T})}, (\mathrm{RES}_{\mathrm{lf}})_{\mathrm{lf}\in\mathrm{Lf}(\mathcal{T})})$:
If ($\mathbf{VrfyCha}(\mathcal{T}, \mathrm{CHA}, (\mathrm{CHA}_{\mathrm{nd}})_{\mathrm{nd}\in\mathrm{tNd}(\mathcal{T})}) = 1$
$\wedge\ \mathbf{VrfyRes}(X, \mathcal{T}, (\mathrm{CMT}, \mathrm{CHA}, \mathrm{RES})_{\mathrm{lf}\in\mathrm{Lf}(\mathcal{T})}) = 1$),
then Return 1 else Return 0

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

$\mathbf{VrfyRes}(X, \mathcal{T}, (\mathrm{CMT}, \mathrm{CHA}, \mathrm{RES})_{\mathrm{lf}\in\mathrm{Lf}(\mathcal{T})})$:
For lf $\in \mathrm{Lf}(\mathcal{T})$
If $\mathbf{\Sigma}^{\mathrm{vrfy}}(x_{\mathrm{att(lf)}}, \mathrm{CMT}_{\mathrm{lf}}, \mathrm{CHA}_{\mathrm{lf}}, \mathrm{RES}_{\mathrm{lf}}) = 0$,
then Return (0)
Return (1)

Now we have to check that $\mathbf{\Sigma}_f$ is certainly a Σ-protocol of f-proof.

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

Proof (sketch). 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 $\mathbf{\Sigma}$. Suppose that the case of height k holds and consider the case of height $k+1$. The construction of $\mathbf{\Sigma}_f^1$ assures the case of height $k+1$. □

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

Proof (sketch). 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.

$\mathbf{\Sigma}_f^{\mathrm{ke}}((\mathrm{CMT}_{\mathrm{lf}}, \mathrm{CHA}_{\mathrm{lf}}, \mathrm{RES}_{\mathrm{lf}})_{\mathrm{lf}\in\mathrm{Lf}(\mathcal{T}_f)}, (\mathrm{CMT}_{\mathrm{lf}}, \mathrm{CHA}'_{\mathrm{lf}}, \mathrm{RES}'_{\mathrm{lf}})_{\mathrm{lf}\in\mathrm{Lf}(\mathcal{T}_f)})$:
$S^* := \phi$
For lf $\in \mathrm{Lf}(\mathcal{T}_f)$
If $\mathrm{CHA}_{\mathrm{lf}} \neq \mathrm{CHA}'_{\mathrm{lf}}$, then $S^* := S^* \cup \{\mathrm{att(lf)}\}$
$w^*_{\mathrm{att(lf)}} \leftarrow \mathbf{\Sigma}^{\mathrm{ke}}((\mathrm{CMT}_{\mathrm{lf}}, \mathrm{CHA}_{\mathrm{lf}}, \mathrm{RES}_{\mathrm{lf}}), (\mathrm{CMT}_{\mathrm{lf}}, \mathrm{CHA}'_{\mathrm{lf}}, \mathrm{RES}'_{\mathrm{lf}}))$
Return ($W^* := \{w^*_{\mathrm{att}}\}_{\mathrm{att}\in S^*}, S^*$)

Then Lemma 1 assures the proposition.

Lemma 1 (Witness Set Extraction) The set W^* output by Σ_f^{ke} satisfies $(X, W^*) \in R_f$.

Proof (sketch). 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, and hence $S^* = \mathrm{Lf}(\mathcal{T}_f)$.

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 Σ_f.

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

Theorem 1 Σ_f is certainly a Σ-protocol of f-proof on the relation R_f.

5. Our Attribute-Based Identification Scheme of Proof of Knowledge

Employing the credential-bundle technique, we modify Σ_f into `ABIDofPoK` that has collusion resistance. The obtained scheme is an ABID of PoK scheme.

5.1 Scheme

The outline of our scheme `ABIDofPoK` is described as follows. We describe below two additional algorithms which are not in Σ_f.

Supplement. This is a generator of each supplementary element a_{att} for $\mathrm{att} \in \mathrm{att}(f) \backslash S$, which is described as follows.

$$\begin{aligned}&\textbf{Supplement}(\mathrm{PK} = x_{\mathrm{M}}, (a_{\mathrm{att}})_{\mathrm{att}\in S}, \mathrm{att}(f)):\\ &\quad \text{For } \mathrm{att} \in \mathrm{att}(f), \text{If } \mathrm{att} \notin S, \text{then}\\ &\qquad c_{\mathrm{att}} \leftarrow \mathrm{CHASP}, (a_{\mathrm{att}}, z_{\mathrm{att}}) \leftarrow \Sigma^{\mathrm{sim}}(x_{\mathrm{M}}, c_{\mathrm{att}})\\ &\quad \text{Return } (a_{\mathrm{att}})_{\mathrm{att}\in\mathrm{att}(f)}\end{aligned}$$

StmtGen. This is a generator of each statement x_{att} for $\mathrm{att} \in \mathrm{att}(f)$, which is described as follows.

$$\begin{aligned}&\textbf{StmtGen}(\mathrm{PK} = x_{\mathrm{M}}, \tau, (a_{\mathrm{att}})_{\mathrm{att}\in\mathrm{att}(f)})\\ &\quad \text{For } \mathrm{att} \in \mathrm{att}(f)\\ &\qquad m_{\mathrm{att}} := (\tau \parallel \mathrm{att}), c_{\mathrm{att}} \leftarrow H_\lambda(m_{\mathrm{att}} \parallel a_{\mathrm{att}})\\ &\qquad x_{\mathrm{att}} \leftarrow \Sigma^{\mathrm{stmtforRES}}(x_{\mathrm{M}}, a_{\mathrm{att}}, c_{\mathrm{att}})\\ &\quad \text{Return } \{x_{\mathrm{att}}\}_{\mathrm{att}\in\mathrm{att}(f)}\end{aligned}$$

Then, our ABID of PoK scheme, `ABIDofPoK` = (**Setup**, **KeyGen**, **P**, **V**), is described as follows.

$$\begin{aligned}&\textbf{Setup}(\lambda, \mathcal{U}):\\ &\quad (x_{\mathrm{M}}, w_{\mathrm{M}}) \leftarrow \mathrm{Instance}_R(\lambda)\\ &\quad \mathrm{PK} := x_{\mathrm{M}}, \mathrm{MSK} := w_{\mathrm{M}}\\ &\quad \mathrm{Return}(\mathrm{PK}, \mathrm{MSK})\end{aligned}$$

$$\begin{aligned}&\textbf{KeyGen}(\mathrm{PK}, \mathrm{MSK}, S):\\ &\quad \tau \leftarrow \{0,1\}^\lambda\\ &\quad \text{For } \mathrm{att} \in S\\ &\qquad m_{\mathrm{att}} := (\tau \parallel \mathrm{att})\\ &\qquad \mathrm{FS}^{\mathrm{sign}}_{\Sigma}(\lambda; \mathrm{PK}, \mathrm{MSK}, m_{\mathrm{att}})\\ &\qquad \rightarrow (a_{\mathrm{att}}, w_{\mathrm{att}}) =: \sigma_{\mathrm{att}}\\ &\quad \mathrm{SK}_S := (\tau, (\sigma_{\mathrm{att}})_{\mathrm{att}\in S})\\ &\quad \mathrm{Return}\ \mathrm{SK}_S.\end{aligned}$$

Note that the credential-bundle technique [12] is employed.

Then, to obtain a whole scheme of ABID of PoK, add the following two procedures (1) and (2) in $\mathbf{P}(\mathrm{PK}, \mathrm{SK}_S; f)$ and $\mathbf{V}(\mathrm{PK}, f)$, respectively.

(1) Just after setting $\mathrm{CHA}_{\mathrm{rt}(\mathcal{T}_f)} := *$, do the following so as to be able to run $\Sigma_f^1(X, W, \mathcal{T}_f, (v_{\mathrm{nd}})_{\mathrm{nd}\in\mathrm{Nd}(\mathcal{T}_f)}, \mathrm{CHA}_{\mathrm{rt}(\mathcal{T}_f)})$:

$$\textbf{Supplement}(\mathrm{PK}, (a_{\mathrm{att}})_{\mathrm{att}\in S}, \mathrm{att}(f)) \rightarrow (a_{\mathrm{att}})_{\mathrm{att}\in\mathrm{att}(f)}$$

$$\textbf{StmtGen}(\mathrm{PK}, \tau, (a_{\mathrm{att}})_{\mathrm{att}\in\mathrm{att}(f)}) \rightarrow \{x_{\mathrm{att}}\}_{\mathrm{att}\in\mathrm{att}(f)} =: X, W := \{w_{\mathrm{att}}\}_{\mathrm{att}\in S}.$$

In the first move, P sends to the verifier V additional elements τ and $(a_{\mathrm{att}})_{\mathrm{att}\in\mathrm{att}(f)}$.

(2) Just after running $\mathrm{CHA} \leftarrow \Sigma_f^2(\lambda)$, do:

$$\textbf{StmtGen}(\mathrm{PK}, \tau, (a_{\mathrm{att}})_{\mathrm{att}\in\mathrm{att}(f)}) \rightarrow \{x_{\mathrm{att}}\}_{\mathrm{att}\in\mathrm{att}(f)} =: X.$$

Theorem 2 (Our ABID of PoK is a Σ-protocol) Suppose that a given Σ-protocol Σ possesses a polynomial-time algorithm $\Sigma^{\mathrm{stmtforRES}}$. Then our `ABIDofPoK` is a Σ-protocol on the relation $R_f := \{(X, W)\}$.

Proof (sketch). Completeness follows from the Completeness of Σ_f. For special soundness, just convert the output of Σ^{ke} into SK_S. For honest verifier zero-knowledge, convert $\Sigma_f^{\mathrm{sim}}(\mathrm{PK}, \mathrm{CHA})$ so as to generate additional elements (a random τ and $(a_{\mathrm{att}})_{\mathrm{att}\in\mathrm{att}(f)}$).

5.2 Security of Our ABID of PoK and its Proof

Theorem 3 (Security against Concurrent Attacks) If the employed signature scheme FS_Σ is existentially unforgeable against chosen-plaintext attacks, then our `ABIDofPoK` is secure against concurrent attacks.

Proof (sketch). Employing any given adversary $\mathcal{A}$ as subroutine, we construct a signature forger $\mathcal{F}$

$\mathcal{F}$ can answer to $\mathcal{A}$'s key extraction queries for a secret key SK_S because $\mathcal{F}$ can query his signing oracle about $(\tau \parallel \mathrm{att}; \mathrm{att} \in S)$, 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^*, S^*) by running Σ_f^{ke}. Finally $\mathcal{F}$ converts (W^*, S^*) into SK_{S^*}. □

5.3 Anonymity

For the case of honest verifier, anonymity described in Section 2.5.3 follows from the honest verifier zero-knowledge property. As for the case of dishonest verifier, it is not obvious that the anonymity holds.

6. Application: Attribute-Based Signature Scheme via the Fiat-Shamir Transform

In this section, we briefly describe an attribute-based signature

scheme obtained by applying the Fiat-Shamir Transform [1].

The Fiat-Shamir Transform ([1], [9]), briefly described in Section 2.3.2, can be directly applied to our ABIDofPoK because our ABIDofPoK is a Σ-protocol.

Discussions in [1] can be applied, so the security of the obtained attribute-based signature scheme is equivalent to the security of our ABIDofPoK against passive attacks.

Corollary 1 (Our Attribute-Based Signature Scheme) Our attribute-based signature scheme obtained by applying the Fiat-Shamir Transform to our ABIDofPoK is secure based on the passive security of ABIDofPoK.

According the Reset Lemma [5], the security of our ABIDofPoK against passive attacks reduces to the security of the employed signature scheme, which is obtained, in our construction, by the Fiat-Shamir Transform of a Σ-protocol $\mathbf{\Sigma}$ in our KeyGen algorithm. Again, the security of the employed signature scheme is equivalent to the security of a Σ-protocol against passive attacks. And finally, according the Reset Lemma, the security of a Σ-protocol against passive attacks reduces to some number theoretic assumptions (such as the discrete logarithm assumption and the RSA assumtion). We remark that security reduction is loose.

7. Concrete Constructions

In this section, we explain our ABID of PoK scheme in concrete forms, that is, in discrete logarithm-setting and RSA setting.

7.1 Discrete Logarithm-Setting

Suppose that $\mathbb{G}_p$ is a cyclic group of prime order p, $|p| = \lambda$ and $\beta = g^{\alpha} \in \mathbb{G}_p$. Setup outputs the following master secret key and public key.

$$\mathrm{MSK} = \alpha, \mathrm{PK} = (g, \beta).$$

KeyGen outputs SK_S with signatures $\sigma = (a_{\mathrm{att}} = g^{r_{\mathrm{att}}}, w_{\mathrm{att}} = r_{\mathrm{att}} + c_{\mathrm{att}}\alpha)$, where $r_{\mathrm{att}} \in \mathbb{Z}_p$ is chosen at random and $m_{\mathrm{att}} := \tau \parallel \mathrm{att}, c_{\mathrm{att}} \leftarrow H_\lambda(m_{\mathrm{att}} \parallel a_{\mathrm{att}})$. $\mathbf{\Sigma}^{\mathrm{stmtforRes}}$ is realized by computing: $x_{\mathrm{att}} := a_{\mathrm{att}}\beta^{c_{\mathrm{att}}}$.

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

$$\mathrm{Cmt}_{\mathrm{lf}} = g^{r_{\mathrm{Cmt}_{\mathrm{lf}}}}, \mathrm{Cha} \leftarrow \mathrm{ChaSp},$$
$$\mathrm{Res}_{\mathrm{lf}} = r_{\mathrm{Cmt}_{\mathrm{lf}}} + \mathrm{Cha}_{\mathrm{lf}}\, w_{\mathrm{att(lf)}},$$
$$\text{Verification Equation} : g^{\mathrm{Res}_{\mathrm{lf}}} \stackrel{?}{=} \mathrm{Cmt}_{\mathrm{lf}}\, (x_{\mathrm{att(lf)}})^{\mathrm{Cha}_{\mathrm{lf}}}.$$

7.2 RSA Setting

Suppose that N is an RSA modulus with $|N| = \lambda$, e is an RSA exponent of odd prime and $\beta = \alpha^e \mod N$. Setup outputs the following master secret key and public key.

$$\mathrm{MSK} = \alpha, \mathrm{PK} = (N, e, \beta).$$

KeyGen outputs SK_S with signatures $\sigma = (a_{\mathrm{att}} = r_{\mathrm{att}}^e, w_{\mathrm{att}} = r_{\mathrm{att}}\alpha^{c_{\mathrm{att}}})$, where $r_{\mathrm{att}} \in \mathbb{Z} \mod N$ is chosen at random and $m_{\mathrm{att}} := \tau \parallel \mathrm{att}, c_{\mathrm{att}} \leftarrow H_\lambda(m_{\mathrm{att}} \parallel a_{\mathrm{att}})$. $\mathbf{\Sigma}^{\mathrm{stmtforRes}}$ is realized by computing: $x_{\mathrm{att}} := a_{\mathrm{att}}\beta^{c_{\mathrm{att}}}$.

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

$$\mathrm{Cmt}_{\mathrm{lf}} = {r_{\mathrm{Cmt}_{\mathrm{lf}}}}^e, \mathrm{Cha} \leftarrow \mathrm{ChaSp}, \mathrm{Res}_{\mathrm{lf}} = r_{\mathrm{Cmt}_{\mathrm{lf}}}(w_{\mathrm{att(lf)}})^{\mathrm{Cha}_{\mathrm{lf}}},$$
$$\text{Verification Equation} : {\mathrm{Res}_{\mathrm{lf}}}^e \stackrel{?}{=} \mathrm{Cmt}_{\mathrm{lf}}\, (x_{\mathrm{att(lf)}})^{\mathrm{Cha}_{\mathrm{lf}}}.$$

8. Conclusions and Future Work

We first construct a Σ-protocol of boolean formula-proof from a given Σ-protocol. There, a knowledge extractor is provided to extract a witness set. Then, using the credential bundle technique, we construct an attribute-based identification scheme of proof of knowledge, ABIDofPoK. Applying the Fiat-Shamir Transform to our ABIDofPoK results in an attribute based signature scheme.

References

[1] Abdalla, M., An, J. H., Bellare, M., Namprempre, C., "From Identification to Signatures via the Fiat-Shamir Transform: Minimizing Assumptions for Security and Forward-Security". Proc. EUROCRYPT 2002, LNCS, vol. 2332, pp. 418-433, Springer-Verlag, Heidelberg (1992).

[2] Anada, H., Arita, S., Handa, S., Iwabuchi, Y., "Attribute-Based Identification: Definitions and Efficient Constructions". Proc. ACISP 2013, LNCS, vol. 7959, pp. 168-186, Springer-Verlag, Heidelberg. (2013).

[3] Beimel, A., "Secure schemes for secret sharing and key distribution'., Ph.D. thesis, Israel Institute of Technology, Technion, Haifa, Israel, 1996.

[4] Bellare, M., Goldreich, O., "On Defining Proofs of Knowledge". Proc. CRYPTO '92, LNCS, vol. 740, pp. 390-420, Springer-Verlag, Heidelberg (1992).

[5] Bellare, M., Palacio, A., "GQ and Schnorr Identification Schemes: Proofs of Security against Impersonation under Active and Concurrent Attacks". Proc. CRYPTO 2002, LNCS, vol. 2442, pp. 162-177, Springer-Verlag, Heidelberg (2002).

[6] Boneh, D., Boyen, X., "Efficient Selective-ID Secure Identity-Based Encryption Without Random Oracles". Proc. EUROCRYPT 2004, LNCS, vol. 3027, pp. 223-238, Springer-Verlag, Heidelberg (2004).

[7] Cramer, R., Damgård, I., Nielsen, J. B., "Multiparty Computation from Threshold Homomorphic Encryption". Proc. EUROCRYPT 2001, LNCS, vol. 2045, pp. 280-300, Springer-Verlag, Heidelberg (2001).

[8] Damgård, I., "On Σ-protocols. In Course Notes, http://www.daimi.au.dk/ ivan/Sigma.ps, 2004.

[9] Fiat, A., Shamir, A., "How to prove yourself: Practical solutions to identification and signature problems". Proc. CRYPTO '86, LNCS, vol. 263, pp. 186194, Springer-Verlag, Heidelberg (1987).

[10] Goyal, V., Pandey, O., Sahai, A., Waters, B., "Attribute-Based Encryption for Fine-Grained Access Control of Encrypted Data". Proc. ACM Conference on Computer and Communications Security, pp.89-98, 2006.

[11] Katz, J., "Efficient Cryptographic Protocols Preventing "Man-in-the-Middle" Attacks". Doctor of Philosophy Dissertation, Columbia University, New York (2002).

[12] Maji, H. K., Prabhakaran, M., Rosulek, M., "Attribute-based Signatures". Proc. CT-RSA 2011, LNCS, vol. 6558, pp. 376-392, Springer-Verlag, Heidelberg (2011). Full version available at Cryptology ePrint Archive, 2010/595, http://eprint.iacr.org/

[13] Schnorr, C. P., "Efficient Identification and Signatures for Smart Cards". Proc. CRYPTO '89, LNCS, vol. 435, pp. 239-252, Springer-Verlag, Heidelberg (1990).

[14] Sahai, A., Waters, B., "Fuzzy Identity-Based Encryption". Proc. EUROCRYPT 2005, LNCS., vol. 3494, pp. 457-473, Springer-Verlag, Heidelberg (2005).

$\mathbf{\Sigma}_f^3(X, W, \mathcal{T}, (v_{\mathrm{nd}})_{\mathrm{nd}\in\mathrm{Nd}(\mathcal{T})}, (\mathrm{CMT}_{\mathrm{lf}})_{\mathrm{lf}\in\mathrm{Lf}(\mathcal{T})}, \mathrm{CHA}, (\mathrm{CHA}_{\mathrm{nd}})_{\mathrm{nd}\in\mathrm{tNd}(\mathcal{T})}, (\mathrm{RES}_{\mathrm{lf}})_{\mathrm{lf}\in\mathrm{Lf}(\mathcal{T})})$:
  $\mathcal{T}_{\mathrm{L}} := \mathrm{Lsub}(\mathcal{T}), \mathcal{T}_{\mathrm{R}} := \mathrm{Rsub}(\mathcal{T})$
  If rt($\mathcal{T}$) is an $\wedge$-node, then $\mathrm{CHA}_{\mathrm{L}} := \mathrm{CHA}, \mathrm{CHA}_{\mathrm{R}} := \mathrm{CHA}$
    Return($\mathrm{CHA}_{\mathrm{L}}, \mathbf{\Sigma}_f^3(X, W, \mathcal{T}_{\mathrm{L}}, (v_{\mathrm{nd}})_{\mathrm{nd}\in\mathrm{Nd}(\mathcal{T}_{\mathrm{L}})}, (\mathrm{CMT}_{\mathrm{lf}})_{\mathrm{lf}\in\mathrm{Lf}(\mathcal{T}_{\mathrm{L}})}, \mathrm{CHA}_{\mathrm{L}}, (\mathrm{CHA}_{\mathrm{nd}})_{\mathrm{nd}\in\mathrm{tNd}(\mathcal{T}_{\mathrm{L}})}, (\mathrm{RES}_{\mathrm{lf}})_{\mathrm{lf}\in\mathrm{Lf}(\mathcal{T}_{\mathrm{L}})})$
      $\mathrm{CHA}_{\mathrm{R}}, \mathbf{\Sigma}_f^3(X, W, \mathcal{T}_{\mathrm{R}}, (v_{\mathrm{nd}})_{\mathrm{nd}\in\mathrm{Nd}(\mathcal{T}_{\mathrm{R}})}, (\mathrm{CMT}_{\mathrm{lf}})_{\mathrm{lf}\in\mathrm{Lf}(\mathcal{T}_{\mathrm{R}})}, \mathrm{CHA}_{\mathrm{R}}, (\mathrm{CHA}_{\mathrm{nd}})_{\mathrm{nd}\in\mathrm{tNd}(\mathcal{T}_{\mathrm{R}})}, (\mathrm{RES}_{\mathrm{lf}})_{\mathrm{lf}\in\mathrm{Lf}(\mathcal{T}_{\mathrm{R}})}))$
  else if rt($\mathcal{T}$) is an $\vee$-node, then
    If $v_{\mathrm{rt}(\mathcal{T}_{\mathrm{L}})} = 1 \wedge v_{\mathrm{rt}(\mathcal{T}_{\mathrm{R}})} = 1$, then $\mathrm{CHA}_{\mathrm{L}} \leftarrow \mathrm{CHASP}$, $\mathrm{CHA}_{\mathrm{R}} := \mathrm{CHA} \oplus \mathrm{CHA}_{\mathrm{L}}$
    else if $v_{\mathrm{rt}(\mathcal{T}_{\mathrm{L}})} = 1 \wedge v_{\mathrm{rt}(\mathcal{T}_{\mathrm{R}})} = 0$, then $\mathrm{CHA}_{\mathrm{L}} := \mathrm{CHA} \oplus \mathrm{CHA}_{\mathrm{rt}(\mathcal{T}_{\mathrm{R}})}, \mathrm{CHA}_{\mathrm{R}} := \mathrm{CHA}_{\mathrm{rt}(\mathcal{T}_{\mathrm{R}})}$
    else if $v_{\mathrm{rt}(\mathcal{T}_{\mathrm{L}})} = 0 \wedge v_{\mathrm{rt}(\mathcal{T}_{\mathrm{R}})} = 1$, then $\mathrm{CHA}_{\mathrm{L}} := \mathrm{CHA}_{\mathrm{rt}(\mathcal{T}_{\mathrm{L}})}$, $\mathrm{CHA}_{\mathrm{R}} := \mathrm{CHA} \oplus \mathrm{CHA}_{\mathrm{rt}(\mathcal{T}_{\mathrm{L}})}$
    else if $v_{\mathrm{rt}(\mathcal{T}_{\mathrm{L}})} = 0 \wedge v_{\mathrm{rt}(\mathcal{T}_{\mathrm{R}})} = 0$, then $\mathrm{CHA}_{\mathrm{L}} := \mathrm{CHA}_{\mathrm{rt}(\mathcal{T}_{\mathrm{L}})}$, $\mathrm{CHA}_{\mathrm{R}} := \mathrm{CHA}_{\mathrm{rt}(\mathcal{T}_{\mathrm{R}})}$
    Return($\mathrm{CHA}_{\mathrm{L}}, \mathbf{\Sigma}_f^3(X, W, \mathcal{T}_{\mathrm{L}}, (v_{\mathrm{nd}})_{\mathrm{nd}\in\mathrm{Nd}(\mathcal{T}_{\mathrm{L}})}, (\mathrm{CMT}_{\mathrm{lf}})_{\mathrm{lf}\in\mathrm{Lf}(\mathcal{T}_{\mathrm{L}})}, \mathrm{CHA}_{\mathrm{L}}, (\mathrm{CHA}_{\mathrm{nd}})_{\mathrm{nd}\in\mathrm{tNd}(\mathcal{T}_{\mathrm{L}})}, (\mathrm{RES}_{\mathrm{lf}})_{\mathrm{lf}\in\mathrm{Lf}(\mathcal{T}_{\mathrm{L}})})$
      $\mathrm{CHA}_{\mathrm{R}}, \mathbf{\Sigma}_f^3(X, W, \mathcal{T}_{\mathrm{R}}, (v_{\mathrm{nd}})_{\mathrm{nd}\in\mathrm{Nd}(\mathcal{T}_{\mathrm{R}})}, (\mathrm{CMT}_{\mathrm{lf}})_{\mathrm{lf}\in\mathrm{Lf}(\mathcal{T}_{\mathrm{R}})}, \mathrm{CHA}_{\mathrm{R}}, (\mathrm{CHA}_{\mathrm{nd}})_{\mathrm{nd}\in\mathrm{tNd}(\mathcal{T}_{\mathrm{R}})}, (\mathrm{RES}_{\mathrm{lf}})_{\mathrm{lf}\in\mathrm{Lf}(\mathcal{T}_{\mathrm{R}})}))$
  else if rt($\mathcal{T}$) is a leaf-node, then
    If $\mathrm{RES}_{\mathrm{rt}(\mathcal{T})} = *$, then $\mathrm{RES} \leftarrow \mathbf{\Sigma}^3(x_{\mathrm{att}(\mathrm{rt}(\mathcal{T}))}, w_{\mathrm{att}(\mathrm{rt}(\mathcal{T}))}, \mathrm{CMT}_{\mathrm{rt}(\mathcal{T})}, \mathrm{CHA})$
    else if $\mathrm{RES}_{\mathrm{rt}(\mathcal{T})} \neq *$, then $\mathrm{RES} \leftarrow \mathrm{RES}_{\mathrm{rt}(\mathcal{T})}$
    Return(RES)

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