Policy-based Signatures for Predicates

Fei Tang, Yousheng Zhou

(Corresponding authors: Yousheng Zhou)

College of Cyberspace Security and Law, Chongqing University of Posts and Telecommunications
No. 2 Chongwen Road, Nanan District, Chongqing 400065, China
(Email: zhouys@cqupt.edu.cn)

(Received Nov. 18, 2016; revised and accepted Feb. 20 & Mar. 11, 2017)

Abstract

Policy-based Signatures (PBS), which were introduced by Bellare and Fuchsbauer, enable signers to sign messages that conform to some policy, yet privacy of the policy is maintained. Bellare et al. defined the policy in any NP language. In PBS schemes for NP language, one should have a valid witness for the policy checking and signing algorithms. In this work, we consider the case of PBS for P language which is a special case of NP language. In PBS schemes for P language, one can directly run the policy checking and signing algorithms without witness. We set policies as some boolean predicates and define the notion of PBS for predicates and its security. Next, for an important class of policy predicates described as (1-dimensional) ranges (i.e., prefix predicate), we design a PBS scheme for such predicate based on tree-based signatures and analyze its application in some real-world scenarios. In addition, based on multilinear maps, we design three PBS schemes for more complex predicates, bit-fixing predicate, left/right predicate, and circuits predicate, respectively.

Keywords: Attribute-based Signatures; Digital Signatures; Group Signatures; Policy-based Signatures

1 Introduction

Digital signatures are one of the most fundamental and well studied cryptographic primitives for providing authentication. In standard signature schemes, a signer who has established a public key pk and a matching secret key sk can sign any message that it wants.

Policy-based Signatures (PBS). The notion of policy-based signatures was introduced by Bellare and Fuchsbauer [1]. In PBS schemes, signer's secret key sk_p is associated with a policy p that allows the signer to produce a valid signature σ of a message m if and only if the message satisfies the policy. PBS provides flexible and fine-grained privacy-respecting authentication which cannot be provided by the other signature variants. For example, group signatures [6, 14], ring signatures [22, 23, 25], and attribute-based signatures [20] also have private signing policy. In these signature variants, any verifier can be convinced that the message was signed by someone entitled to, but not who this person is. In addition, in mesh signature scheme [3], the policy itself is always public, as in the warrant, which specifies the policy in proxy signatures [7, 17, 21]. However, note that there has a big difference between policy-based signatures and the other signature variants that the policy-based signatures provide fine-grained control over what kind of messages can be signed by sk_p which associates a policy p.

Bellare et al. [1] defined the policy to be any NP language $\mathcal{L}$. In order to check that whether a message satisfies a policy or not, they defined a policy checker (i.e., an NP-relation) $\mathsf{PC} : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}$. The first input is a pair (p, m) representing a policy $p \in \{0,1\}^*$ and a message $m \in \{0,1\}^*$, while the second input is a witness $w \in \{0,1\}^*$. The associated language $\mathcal{L}(\mathsf{PC}) = \{(p, m) : \exists w \text{ s.t. } m \text{ satisfies } p\}$ is called the policy language associated to PC. Given a witness w, one can test in polynomial time whether a given policy p allows a given message m, where $(p, m) \in \mathcal{L}(\mathsf{PC})$, or $\mathsf{PC}((p, m), w) \stackrel{?}{=} 1$ for short.

Our Motivation. Bellare et al. [1] designed a generic construction of PBS scheme for any NP language based on Groth-Sahai proofs [11]. In the scheme of [1], policies can be expressed and enforced are restricted neither in form nor in type, the only condition being that, given a witness, one can test in polynomial time whether a policy allows a message or not. However, in real-world applications, we may only need some specific policies. In addition, it is preferable that check whether a policy allows a message without the help of any witness. In the case of PBS for NP, the witness is necessary for the policy checker and signing algorithm. Hence, in this work, we consider a special case that the policy language in P, meaning that one can directly (without any witness w) run the signing algorithm and test in polynomial time whether a given policy allows a given message, or $p(m) \stackrel{?}{=} 1$ for short.

Our Results. First of all, we define the notion of PBS for predicates, i.e., describe the policies as some boolean predicates, and its security. Then, we design several con-

crete PBS schemes for different predicate families.

- Prefix predicates are an important class of policy predicates. Based on tree-based signatures, we design a PBS scheme and prove its security. In addition, we will analyze the applications of PBS scheme for prefix predicates in some real-world scenarios.

- Furthermore, we provide another method to construct PBS for more complex predicates, bit-fixing predicate, left/right predicate, and circuits predicate. The main tool for these three constructions is multilinear map [4]. However, low efficiency which dues to the low efficiency of existing multilinear map candidates and selective unforgeability are two major shortcomings with respect to these mulitlinear-map-based constructions. Therefore, this part of work is tend to theoretical realization.

2 Preliminaries

2.1 Multilinear Maps

The notion of multilinear maps was introduced by Boneh and Silverberg [4]. Then, Garg et al. [9] gave the first approximate candidate of multilinear maps. Then, many subsequent schemes have been proposed, e.g., [8, 10]. Unfortunately, some of them have been breached, e.g., [12, 18]. However, even so, many cryptographic schemes based on multilinear maps have been proposed, for examples, aggregate signatures [13], attribute-based signatures for circuits [24], constrained PRFs [5] and so on.

Let $\overrightarrow{\mathbb{G}} = (\mathbb{G}_1, \ldots, \mathbb{G}_k)$ be a sequence of groups each of large prime order p, and g_i be a canonical generator of $\mathbb{G}_i$, where we set $g = g_1$. There exists bilinear maps $\{e_{i,j} : \mathbb{G}_i \times \mathbb{G}_j \to \mathbb{G}_{i+j} | i, j \geq 1 \wedge i + j \leq k\}$, which satisfy: $e_{i,j}(g_i^a, g_j^b) = g_{i+j}^{ab} : \forall a, b \in \mathbb{Z}_p$. When the context is obvious, we drop the subscripts i and j, such as $e(g_i^a, g_j^b) = g_{i+j}^{ab}$. It also will be convenient to abbreviate $e(h_1, h_2, \ldots, h_j) = e(h_1, e(h_2, \ldots, e(h_{j-1}, h_j)\ldots)) \in \mathbb{G}_i$ for $h_j \in \mathbb{G}_{i_j}$ and $i_1 + i_2 + \ldots + i_j \leq k$. We assume that $\mathcal{G}(1^\lambda, k)$ is a PPT group generator algorithm which takes as input a security parameter λ and a positive integer k to indicate the number of allowed pairing operations, then it outputs the multilinear parameters $mp = (\mathbb{G}_1, \ldots, \mathbb{G}_k, p, g = g_1, g_2, \ldots, g_k, e_{i,j})$ to satisfy the above properties.

The assumption of Multilinear Computational Diffie-Hellman (MCDH) can be viewed as an adaptation of Bilinear Computational Diffie-Hellman assumption in the setting of multilinear maps.

Definition 1. *For any PPT algorithm $\mathcal{B}$, any polynomial $p(\cdot)$, any integer k, and all sufficiently large $\lambda \in \mathbb{N}$,*

$$Pr\left[\begin{array}{l} mp \leftarrow \mathcal{G}(1^\lambda, k); \\ c_1, \ldots, c_k \xleftarrow{R} \mathbb{Z}_p; \\ T \leftarrow \mathcal{B}(mp, g^{c_1}, \ldots, g^{c_k}) \end{array} : T = g_{k-1}^{\prod_{i\in[k]} c_i}\right] < \frac{1}{p(\lambda)},$$

where $c_i \xleftarrow{R} \mathbb{Z}_p$ means that c_i is randomly and uniformly chosen from the set $\mathbb{Z}_p$, and $[k]$ is an abbreviation of the set $\{1, 2, \ldots, k\}$.

2.2 Example Predicate Families

We take the predicate families in [5] as examples to realize PBS schemes.

Prefix predicates. Let $v \in \{0,1\}^n$, where $n \in [k]$, be a bit string, the prefix predicate $p_v : \{0,1\}^k \to \{0,1\}$ is defined as: $p_v(m) = 1 \Leftrightarrow m$ has v as a prefix. The set of prefix predicates is $\mathcal{P}_{pre} = \{p_v : v \in \{0,1,\perp\}^n, n \in [k]\}$. Prefix predicate is a special case of bit-fixing predicate.

Bit-fixing predicates. Let $\mathbf{v} \in \{0,1,\perp\}^n$ be a vector, the bit-fixing predicate $p_{\mathbf{v}}^{(BF)} : \{0,1\}^n \to \{0,1\}$ is defined as: $p_{\mathbf{v}}^{(BF)}(m) = 1 \Leftrightarrow (\mathbf{v}_i = m_i$ or $\mathbf{v}_i = \perp)$ for all $i = 1, \ldots, n$. The set of bit-fixing predicates is $\mathcal{P}_{BF} = \{p_{\mathbf{v}}^{(BF)} : \mathbf{v} \in \{0,1,\perp\}^n\}$.

Left/right predicates. For a bit string $w \in \{0,1\}^{|m|/2}$, where $m \in \mathcal{M}$ and $|m| = 2 \cdot s$ denotes the size of the message, define two predicates $p_w^{(L)}, p_w^{(R)} : \{0,1\}^{|m|/2} \to \{0,1\}$ as: $p_w^{(L)}(m_1, m_2) = 1 \Leftrightarrow m_1 = w$ and $p_w^{(R)}(m_1, m_2) = 1 \Longleftrightarrow m_2 = w$.

Circuit predicates. Let $\mathcal{C}$ be the set of polynomial size circuits. Circuit predicate family is defined as $\mathcal{P}_{cir} = \{p : p \in \mathcal{C}\}$.

3 Policy-based Signatures for Predicates

3.1 Definition

A policy-based signature scheme, $\mathcal{PBS}$, consists of the following four PPT algorithms:

- $\mathsf{Setup}(1^\lambda)$: The setup algorithm takes as input a security parameter λ. It outputs public parameters pp and a master secret key msk, where the public parameters, pp, contain the descriptions of the message space $\mathcal{M}$, signature space $\mathcal{S}$, and policy (boolean predicate) space $\mathcal{P}$. The master secret key msk can sign all messages in $\mathcal{M}$.

- $\mathsf{KeyGen}(msk, p)$: The key generation algorithm takes as input the master secret key msk and a boolean predicate $p \in \mathcal{P}$. It outputs a signing key sk_p for the predicate p.

- $\mathsf{Sign}(sk_p, m)$: The signing algorithm takes as input a signing key sk_p and a message $m \in \mathcal{M}$. It outputs a signature $\sigma \in \mathcal{S}$ if $p(m) = 1$. Otherwise, it outputs $\perp$.

- $\mathsf{Verify}(pp, m, \sigma)$: The verification algorithm takes as input the public parameters pp and a purported signature σ for a message m. It outputs 1 or 0.

We require that for all security parameter λ, $(msk, pp) \leftarrow \mathsf{Setup}(1^\lambda)$, $p \in \mathcal{P}, sk_p \leftarrow \mathsf{KeyGen}(msk, p)$, and $m \in \mathcal{M}$, if $p(m) = 1$ and $\sigma \leftarrow \mathsf{Sign}(sk_p, m)$, then we have $\mathsf{Verify}(pp, m, \sigma) = 1$.

3.2 Security Models

The security of policy-based signature for predicates is defined by the following two notions: unforgeability and privacy (i.e., indistinguishability in [1]).

Unforgeability. This notion guarantees that one can sign some message m only if it has a signing key sk_p such that $p(m) = 1$.

- **Setup:** The challenger runs the setup algorithm to generate public parameters pp and a master secret key msk. It then gives pp to the adversary and keeps msk to itself.
- **Key Generation Oracle:** The adversary adaptively makes any polynomial number of signing key queries for boolean predicate $p \in \mathcal{P}$ of its choice. The challenger returns back $sk_p \leftarrow \mathsf{KeyGen}(msk, p)$.
- **Signing Oracle:** The adversary adaptively makes any polynomial number of signature queries on input a message $m \in \mathcal{M}$. The challenger chooses a $p \in \mathcal{P}$ such that $p(m) = 1$ and returns back $\sigma \leftarrow \mathsf{Sign}(sk_p, m)$, where sk_p is obtained from the key generation algorithm.
- **Forgery:** The adversary finally outputs a tuple (m^*, σ^*). It wins the game if (1) $\mathsf{Verify}(pp, m^*, \sigma^*) = 1$; (2) m^* was never queried to the signing oracle; and (3) $p(m^*) = 0$ for all p queried to the key generation oracle.

We denote the advantage of a PPT adversary $\mathcal{A}$ (taken over the random choices of the challenger and adversary) to win the game as $\mathbf{Adv}_{\mathcal{A}}^{Unf} = \Pr[\mathcal{A} \text{ wins}]$.

Definition 2. *A policy-based signature scheme is existentially unforgeable if any PPT adversary can win the above game with at most negligible advantage.*

Remark 1. In the case of PBS for NP, the above notion is unsatisfactory. This is because one cannot efficiently verify whether an adversary has won the game, as this needs it has a valid witness w to checking that whether $(p, m) \in \mathcal{L}(\mathsf{PC})$ for all p queried to the key generation oracle and m from the adversary's final output. However, in the case of PBS for P, one can always efficiently verify it without any witness.

We also define a weaker (selective) variant to the above definition where the adversary is required to commit to a challenge message, m^*, before the setup phase.

Definition 3. *A policy-based signature scheme is selectively unforgeable if any PPT adversary can win the selective game with at most negligible advantage.*

Perfect Privacy. This notion guarantees that a valid signature will reveal nothing about the signing policy.

- **Setup:** The challenger runs the setup algorithm to generate public parameters pp and a master secret key msk. It then gives pp and msk to the adversary.
- **Challenge:** The adversary submits a challenge message $m^* \in \mathcal{M}$ and two *different* boolean predicates $p_0, p_1 \in \mathcal{P}$ such that $p_0(m^*) = p_1(m^*) = 1$, to the challenger. The challenger flips a random coin $b \leftarrow \{0, 1\}$ and returns back $\sigma_b \leftarrow \mathsf{Sign}(sk_{p_b}, m^*)$, where sk_{p_b} is obtained from the key generation algorithm.
- **Guess:** Finally, the adversary outputs his guessing bit b' and wins the game if $b' = b$.

We denote the advantage of an unbounded adversary $\mathcal{A}$ (taken over the random choices of the challenger and adversary) to win the game as $\mathbf{Adv}_{\mathcal{A}}^{Pri} = |\Pr[b' = b] - \frac{1}{2}|$.

Definition 4. *A policy-based signature scheme is perfectly private if even an unbounded adversary wins the above game with at most negligible advantage.*

4 Policy-based Signatures Based on One-Way Functions

We now construct a policy-based signature scheme supporting the class of prefix predicate based on the tree-based signature scheme [19].

4.1 Tree-based Signature Scheme

Let $\mathcal{TTS} = (\mathsf{keygen}, \mathsf{sign}, \mathsf{verify})$ be a two-time signature scheme.[1] For a binary string m, let $m|_i = m_1 \cdots m_i$ denote the i-bit prefix of m (with $m|_0 := \varepsilon$, the empty string). More specifically, we imagine a binary tree of depth k where the root is labelled by ε (i.e., the empty string), and a node that is labelled with a binary string w, where $|w| < k$, has left-child labelled $w0$ and right-child labelled $w1$. For every node w, we associate a key pair (pk_w, sk_w) from the two-time signature scheme. The public key of the root, pk_ε, is the actual public key of the signer. To sign a message $m \in \{0, 1\}^k$, the signer does the following steps:

1) It first generates keys for all nodes on the path from the root to the leaf labelled m. Some of these public keys may generated in the process of signing previous

[1] The original tree-based signature scheme is based on any one-time signature (e.g., [16]). For ease of description, we define the tree-based signature scheme based on any $\mathcal{TTS}$ which can be easily realized from any one-time signature scheme.

messages, in such case the previous values are stored as part of the state.

2) It then "certifies" the path from the root to the leaf labelled m by computing a signature on pk_{w0} or pk_{w1} (which depends on whether $m_{|w|+1} = 0$ or 1), using secret key sk_w, for each string w that is a proper prefix of m.

3) Finally, the signer "certifies" m itself by computing a signature on m using the secret key sk_m.

Formally, the tree-based signature scheme $\mathcal{TBS} = (\mathsf{keygen}^*, \mathsf{sign}^*, \mathsf{verify}^*)$ is as follows:

keygen$^*(1^\lambda)$**:** The key generation algorithm runs $(pk_\varepsilon, sk_\varepsilon) \leftarrow \mathsf{keygen}(1^\lambda)$ and outputs the public key pk_ε. The secret key and initial state are sk_ε.

sign$^*(sk, m \in \{0,1\}^k)$**:** To sign a message $m \in \{0,1\}^k$ using the current state, the singing algorithm does the following:

1) For $i = 0, \ldots, k-1$: let $m|_i b$ be the $(i+1)$-bit prefix of m.
   - If $pk_{m|_i b}$ and $\sigma_{m|_i b}$ are not in the current state, compute them:
   $$(pk_{m|_i b}, sk_{m|_i b}) \leftarrow \mathsf{keygen}(1^\lambda);$$
   $$\sigma_{m|_i b} \leftarrow \mathsf{sign}(sk_{m|_i}, pk_{m|_i b}),$$
   and then store all these computed values as part of the state.
2) If σ_m is not yet included in the state, compute $\sigma_m \leftarrow \mathsf{sign}(sk_m, m)$ and store it as part of the state.
3) Output $\sigma = (\{(pk_{m|_i b}, \sigma_{m|_i b})\}_{i\in[0,k-1]}, \sigma_m)$ as the signature for message m.

verify$^*(pk, m, \sigma)$**:** Given a public key pk_ε, a message $m \in \{0,1\}^k$, and a signature $(\{(pk_{m|_i b}, \sigma_{m|_i b})\}_{i=0}^{k-1}, \sigma_m)$, output 1 if and only if the following two equations hold:

1) $\mathsf{verify}(pk_{m|_i}, pk_{m|_i b}, \sigma_{m|_i b}) = 1$ for all $i = 0, \ldots, k-1$.
2) $\mathsf{verify}(pk_m, m, \sigma_m) = 1$.

Theorem 1. *If $\mathcal{TTS}$ is a signature scheme that is existentially unforgeable against two-time chosen-message attack. Then the tree-based signature scheme $\mathcal{TBS}$ is existentially unforgeable against adaptive chosen-message attack.*

The proof this theorem is similar to that of the tree-based signature scheme based on any one-time signature scheme. We omit the proof.

4.2 PBS for Prefix Predicates based on $\mathcal{TBS}$

We now construct a policy-based signature scheme for the prefix predicates based on the tree-based signature scheme. The idea of our construction is as follows:[2] a singing key sk_{p_v} corresponding to a predicate $p_v \in \mathcal{P}_{pre}$ will be the partial certification of the $\mathcal{TBS}$ tree, at level $|v|$. Given this partial certification, a signer will be able to compute the completion for any message m which has v as a prefix. However, as we will argue, the computation of all other messages will remain unknown to the signer. Formally, our PBS scheme (Setup, KeyGen, Sign, Verify) for prefix predicates is as follows:

Setup(1^λ)**:** This algorithm runs $(pk_\varepsilon, sk_\varepsilon) \leftarrow \mathsf{keygen}^*(1^\lambda)$ and sets $msk := sk_\varepsilon, pp := pk_\varepsilon$.

KeyGen(msk, p_v)**:** To compute a signing key sk_{p_v} for a prefix predicate $p_v \in \mathcal{P}_{pre}$, where $|v| \le k$. This key generation algorithm does the following:

1) For $i = 0, \ldots, |v|-1$: let $v|_i b$ be the $(i+1)$-bit prefix of the vector v.
   - If $pk_{v|_i b}$ and $\sigma_{v|_i b}$ are not in the current state, compute them:
   $$(pk_{v|_i b}, sk_{v|_i b}) \leftarrow \mathsf{keygen}(1^\lambda);$$
   $$\sigma_{v|_i b} \leftarrow \mathsf{sign}(sk_{v|_i}, pk_{v|_i b}),$$
   and then store all these computed values as part of the state.
2) Output $sk_{p_v} = (\{(pk_{v|_i b}, \sigma_{v|_i b})\}_{i\in[0,|v|-1]}, sk_v)$ as the signing key for p_v.

Sign$(sk_{p_v}, m \in \{0,1\}^k)$**:** To sign a message $m \in \{0,1\}^k$ using the current state, the signing algorithm does the following:

1) If $p_v(m) = 0$, i.e., $\exists i \in [|v|]$ s.t. $m_i \neq v_i$, then abort.
2) For $i = |v|, \ldots, k-1$: let $m|_i b$ be the $i+1$-bit prefix of m.
   - If $pk_{m|_i b}$ and $\sigma_{m|_i b}$ are not in the current state, compute them:
   $$(pk_{m|_i b}, sk_{m|_i b}) \leftarrow \mathsf{keygen}(1^\lambda);$$
   $$\sigma_{m|_i} \leftarrow \mathsf{sign}(sk_{m|_i}, pk_{m|_i b}),$$
   and then store all these computed values as part of the state.
3) If σ_m is not yet included in the state, compute $\sigma_m \leftarrow \mathsf{sign}(sk_m, m)$ and store it as part of the state.
4) Output $\sigma = (\{(pk_{m|_i b}, \sigma_{m|_i b})\}_{i\in[0,k-1]}, \sigma_m)$ as the signature for message m.

Verify(pp, m, σ)**:** It is same the verify^* algorithm in the $\mathcal{TBS}$ scheme.

[2] A similar idea has been used to construct constrained PRFs [2].

Unforgeability. Our PBS scheme $\mathcal{PBS}$ is similar to the tree-based signature scheme $\mathcal{TBS}$. The only difference is that in $\mathcal{PBS}$ the signer obtains a signing key sk_{p_v}, rather than a full-featured key $msk := sk_\varepsilon$ as in $\mathcal{TBS}$, which enables him to signing a subset of the domain $\{0,1\}^k$. For completeness, we give the following proof.

Theorem 2. *If $\mathcal{TTS}$ is a signature scheme that is existentially unforgeable under a two-time chosen-message attack. Then the policy-based signature scheme $\mathcal{PBS}$ for message space $\{0,1\}^k$ is existentially unforgeable under an adaptive chosen-message attack.*

Proof. For length of messages k, we prove security based on two-time signature scheme $\mathcal{TTS}$. We show that if there exists a PPT adversary $\mathcal{A}$ on our PBS scheme then we can construct an efficient algorithm $\mathcal{B}$ to break the security of scheme $\mathcal{TTS}$. We describe how $\mathcal{B}$ interacts with $\mathcal{A}$. The algorithm $\mathcal{B}$ first receives a challenge key pk from the challenger of scheme $\mathcal{TTS}$.

Let q_k, q_s be the upper bounds on the number of key generation queries and signing queries, respectively, made by $\mathcal{A}$, and set $\ell = k(q_k+q_s)+1$. Note that ℓ upper-bounds the number of public keys from $\mathcal{TTS}$ that are needed to generate q_k keys and q_s signatures using $\mathcal{PBS}$ (in the worst case), and there is one additional key form $\mathcal{TTS}$ that is used as the actual public key pk_ε.

Setup: Initially, $\mathcal{B}$ chooses a random $i^* \leftarrow [\ell]$, we assume that i^* is put on the node (or leaf) w^*. Construct a list $pk^1, \ldots, pk^\ell$ of keys as follows:

- Set $pk^{i^*} := pk$.
- For $i \neq i^*$, run $(pk^i, sk^i) \leftarrow \mathsf{keygen}(1^\lambda)$.

Then $\mathcal{B}$ runs $\mathcal{A}$ on input the public key $pk_\varepsilon := pk^1$. Note that $\mathcal{B}$ knows all secret key sk^i, for $i \in [\ell]$, except that the challenge one $sk^{i^*} := sk$. With respect to the challenge key sk, $\mathcal{B}$ can make at most two signing queries to the $\mathcal{TTS}$ challenger according to the security of the $\mathcal{TTS}$ scheme.

Key Generation Oracle: $\mathcal{A}$ will query for a signing key for a prefix predicate $p_v \in \mathcal{P}_{pre}$. $\mathcal{B}$ creates it for the adversary as follows:

- If w^* is not on the path from the root to the node v, then $\mathcal{B}$ can create signing key sk_{p_v} honestly since it knows all secret keys with respect to the nodes on the path from the root to the node v.
- If w^* is on the path from the root to the node v, to certify w^*'s child w^*b, $\mathcal{B}$ first makes a signing query to the $\mathcal{TTS}$ challenger on input pk^{i^*+1}, and then it will receive a signature $\sigma_{w^*b} \leftarrow \mathsf{sign}(sk, pk^{i^*+1})$. Finally, $\mathcal{B}$ can create sk_{p_v}.

Signing Oracle: The adversary $\mathcal{A}$ will query for a signature for a message $m \in \{0,1\}^k$. $\mathcal{B}$ creates signatures for the adversary as follows:

- If w^* is not on the path from the root to the leaf m, then $\mathcal{B}$ can create signature σ honestly since it knows all secret keys with respect to the nodes on the path from the root to the leaf m.
- If w^* is on the path from the root to the leaf m, and if w^* is a internal node, to certify the w^*'s child w^*b, $\mathcal{B}$ first makes a signing query to the $\mathcal{TTS}$ challenger on input pk^{i^*+1} and then it will receive a signature $\sigma_{w^*b} \leftarrow \mathsf{sign}(sk, pk^{i^*+1})$; if w^* is the leaf m, then $\mathcal{B}$ makes singing query to the $\mathcal{TTS}$ challenger on input m and then it will receive a signature $\sigma_m \leftarrow \mathsf{sign}(sk, m)$. Finally $\mathcal{B}$ can compute the complete signature σ.

Forgery: Eventually, $\mathcal{A}$ outputs

$$\sigma^* = (\{(pk'_{m^*|_i b}, \sigma'_{m^*|_i b})\}_{i=0}^{k-1}, \sigma'_{m^*})$$

for message m^*. If it is valid, then:

Case 1: There exists a $j \in [0, k-1]$ for which $pk'_{m^*|_j b} \neq pk_{m^*|_j b}$, this means that $\mathcal{A}$ creates a new key $pk'_{m^*|_j b}$ but not initially defined by $\mathcal{B}$. If $j = i^*$, $\mathcal{B}$ outputs $(pk'_{m^*|_j b}, \sigma'_{m^*|_j b})$.

Case 2: If Case 1 does not hold, then $pk'_{m^*} = pk_{m^*}$. Let j be such that $pk^j = pk_m$. If $j = i^*$, $\mathcal{B}$ outputs (m^*, σ'_{m^*}).

Note that i^* was chosen uniformly at random and is independent of the view of $\mathcal{A}$, and the list $pk^1, \ldots, pk^\ell$ generated by $\mathsf{keygen}(1^\lambda)$ is distributed identically to the view of $\mathcal{A}$ in the real unforgeability game. Thus if $\mathcal{A}$ outputs a valid forgery (regardless of which of the above cases occurs) with probability ϵ, $\mathcal{B}$ can outputs a forgery with probability ϵ/ℓ, where $1/\ell$ means the probability of $j = i^*$ in which of the above cases occurs. By the assumed security of $\mathcal{TTS}$ and the fact that ℓ is polynomial, we conclude that ϵ must negligible. □

Perfect Privacy. Given a valid signature (m^*, σ^*), we show that any key sk_{p_v} such that $p_v(m^*) = 1$ could possibly have created it. The proof is straightforward.

Theorem 3. *The above PBS scheme based on OWF is perfectly private.*

Proof. Any unbounded adversary $\mathcal{A}$ submits a challenge tuple (p_v, p_w, m^*) such that $p_v(m^*) = p_w(m^*) = 1$. The distribution of the signatures for m^* generated by the signing key sk_{p_v} is: $(\{(pk_{m^*|_i b}, \sigma_{m^*|_i b})\}_{i\in[0,k-1]}, \sigma_{m^*})$, where each key pair (pk_i, sk_i) is generated by the key generation algorithm randomly and independently. Similarly, The distribution of the signatures for m^* generated by the signing key sk_{p_w} is: $(\{(pk'_{m^*|_i b}, \sigma'_{m^*|_i b})\}_{i\in[0,k-1]}, \sigma'_{m^*})$,

where each key pair (pk'_i, sk'_i) also is generated by the key generation algorithm randomly and independently. Therefore, these two distributions are identical. The perfect privacy follows easily from this observation. □

4.3 Application of PBS for Prefix Predicates

We consider the application scenario which was raised by Bellare et al. [1]. A company implements a scheme where each employee gets a signing key with a policy and there is only one public key which is used by outsiders to verify signatures in the name of the company. Company stipulates that employee in different department has different policy. For example, the policy for sales department states the prices of the products, the policy for technology department states the functionalities of the products, and so on.

PBS for prefix predicates is a useful tool in addressing this problem. For an employee in sales department, company sets policy p_s which states the prices of the products, then distributes a signing key sk_{p_s} to this employee. Finally, the employee can sign messages on behalf of the company, where these messages may contain the statement "`product prices||after-sale service terms||`$\cdots$". In such a scenario, the employee can decide the after-sale service terms and some other regulations, however, the product prices which are stipulated in the policy p_s cannot be changed. If the PBS scheme for prefix predicates is secure, then any outsider can be convinced that, from a valid PBS signature, the regulations (i.e., message) was agreed by someone entitled to, but not who this person is.

Related Works. Append-Only Signatures (AOS) [15] are a similar notion to the PBS for prefix predicates. In AOS schemes, any party is given an AOS signature σ_{m_1} for message m_1 can compute $\sigma_{m_1||m_2}$ for message $m_1||m_2$, where the message m_2 is chosen by the party. In AOS, *anyone* can append and verify signatures. However, in PBS, the signing key sk_p cannot be opened, and hence only the holder of the signing key can sign messages. Kiltz et al. [15] showed that AOS is equivalent to Hierarchical Identity-Based signatures (HIBS), and it can be used to the Border Gateway Protocol (BGP). PBS for prefix predicates and AOS are two different signature variants because: (1) there has no obvious evidence shows that the PBS for prefix predicates has the properties (i.e., connection to HIBS and application to BGP) provided by the AOS; and (2) AOS apparently does not apply to the above application scenario because, in the above application scenario, the signer should to be some authorized employee rather than anyone. Although, there may be have some potential connections between these two signature variants, e.g., realize one from the other one by some transformation, which beyond the reach of this work.

5 Policy-based Signatures Based on Multilinear Maps

In this section, we take advantage of the multilinear maps to realize three PBS constructions for bit-fixing predicates, left/right predicates, and circuit predicates, respectively. The main technique of our multilinear-map-based PBS schemes follows Boneh and Waters' [5] work which constructs constrained pseudorandom functions. Their technique has been used to construct different cryptographic primitives, such as attribute-based signatures for circuits [24] and so on. In this work, we also follow Boneh et al.'s [5] technique, however, to construct a different cryptographic primitive, policy-based signatures.

5.1 PBS for Bit-Fixing Predicates

Setup$(1^\lambda, k)$**:** The setup algorithm takes as input a security parameter λ and an integer k. The algorithm then runs $\mathcal{G}(1^\lambda, k)$ that produces groups $\vec{\mathbb{G}} = (\mathbb{G}_1, \ldots, \mathbb{G}_k)$ of prime order p, with canonical generators $g_1, \ldots, g_k$, where we let $g = g_1$. Next it chooses random $\alpha \in \mathbb{Z}_p$ and $(a_{1,0}, a_{1,1}), \ldots, (a_{k-1,0}, a_{k-1,1}) \in \mathbb{Z}_p^2$ and computes $A_{i,\beta} = g^{a_{i,\beta}}$ for $i \in [k-1]$ and $\beta \in \{0,1\}$.

The master secret key is $msk = \alpha$. The public parameters, pp, consist of the group sequence description plus group elements $g^\alpha, \{A_{i,\beta} | i \in [k-1], \beta \in \{0,1\}\}$, message space $\mathcal{M} = \{0,1\}^{k-1}$, signature space $\mathcal{S} = \mathbb{G}_{k-1}$, and predicate space $\mathcal{P} = \{0,1,\perp\}^{k-1}$.

KeyGen$(msk, \mathbf{v} \in \{0,1,\perp\}^{k-1})$**:** For a vector $\mathbf{v} \in \{0,1,\perp\}^{k-1}$, let V be the set of indices $i \in [k-1]$ such that $\mathbf{v}_i \neq \perp$. That is the indices for which the bit is fixed to 0 or 1. The signing key for the predicate $p_{\mathbf{v}}$ is:

$$sk_{\mathbf{v}} = (g_{|V|}^{\prod_{i \in V} a_{i,\mathbf{v}_i}})^\alpha \in \mathbb{G}_{|V|}.$$

Sign$(sk_{\mathbf{v}}, m \in \{0,1\}^k)$**:** Given a message, m, of length $k-1$, let $m_1, \ldots, m_{k-1}$ be the bits of this message. If the message m does not satisfy the predicate, i.e., $p_{\mathbf{v}}(m) = 0$, then abort. Otherwise, $p_{\mathbf{v}}(m) = 1$. That is $\mathbf{v}_i = m_i$ for all $i \in V$, then the signing algorithm compute a signature:

$$\sigma = e(sk_{\mathbf{v}}, g_{k-|V|-1}^{\prod_{i \in [k-1] \setminus V} a_{i,m_i}}) = g_{k-1}^{\alpha \cdot \prod_{i \in [k-1]} a_{i,m_i}} \in \mathbb{G}_{k-1},$$

where $g_{k-|V|-1}^{\prod_{i \in [k-1] \setminus V} a_{i,m_i}}$ can be computed by the multilinear maps from the parameters A_{i,m_i} for $i \in [k-1] \setminus V$.

Verify(pp, m, σ)**:** Given a purported signature σ on a message m, verify the following equation:

$$e(\sigma, g) = e(g^\alpha, A_{1,m_1}, \ldots, A_{k-1,m_{k-1}}).$$

Output 1 if it holds, else 0.

Correctness. The verification of the final signatures is justified by the following two equations:

$$\begin{aligned} e(\sigma, g) &= e((g_{k-1}^{\prod_{i\in[k-1]} a_{i,m_i}})^{\alpha}, g) \\ &= g_k^{\alpha\cdot\prod_{i\in[k-1]} a_{i,m_i}}. \end{aligned}$$

and

$$e(g^{\alpha}, A_{1,m_1}, \ldots, A_{k-1,m_{k-1}}) = g_k^{\alpha\cdot\prod_{i\in[k-1]} a_{i,m_i}}.$$

Theorem 4. *If the k-MCDH assumption is hold in the multilinear groups, then the above PBS construction for bit-fixing predicates and for messages of length $k-1$ is selectively unforgeable and perfectly private.*

Selective Unforgeability. We prove selective unforgeability, where the key access structures are bit-fixting predicates. For length of messages $k-1$, we prove security under the k-MCDH assumption. We show that if there exists a PPT adversary $\mathcal{A}$ on our PBS scheme for messages of length $k-1$ in the selective security game then we can construct an efficient algorithm $\mathcal{B}$ on the k-MCDH assumption. We describe how $\mathcal{B}$ interacts with $\mathcal{A}$.

The algorithm $\mathcal{B}$ first receives a k-MCDH challenge instance consisting of the group sequence description $\overrightarrow{\mathbb{G}}$ and $g = g_1, g^{c_1}, \ldots, g^{c_k}$. It also receives challenge message $m^* = m_1^* \ldots m_{k-1}^* \in \{0,1\}^{k-1}$ from the adversary.

Setup: Initially, $\mathcal{B}$ chooses random $u_1, \ldots, u_{k-1} \in \mathbb{Z}_p$ and sets

$$A_{i,\beta} = \begin{cases} g^{c_i}, & \text{if } m_i^* = \beta \\ g^{u_i}, & \text{if } m_i^* \neq \beta \end{cases}$$

for $i \in [k-1], \beta \in \{0,1\}$. This corresponds to setting $a_{i,\beta} = c_i$ if $m_i^* = \beta$ and u_i otherwise. We observe these are distributed identically to the real scheme. In addition, it will internally view $\alpha = c_k$.

Key Generation Oracle: The adversary $\mathcal{A}$ will query for a signing key for a bit-fixing predicate $p_{\mathbf{v}}$, where $\mathbf{v} \in \{0,1,\perp\}^{k-1}$. We let V be the set of indices $i \in [k-1]$ such that $\mathbf{v}_i \neq \perp$.

If $p_{\mathbf{v}}(m^*) = 1$, that is $m_i^* = \mathbf{v}_i$ for all $i \in V$. Then $\mathcal{B}$ aborts the game.

If $p_{\mathbf{v}}(m^*) = 0$, that is $\exists j \in [k-1]\backslash V$, s.t. $m_j^* \neq \mathbf{v}_j$. Then $\mathcal{B}$ will be able to create signing keys for the adversary, because his query will differ from the challenge message at least one bit. More specifically, $\mathcal{B}$ produces the signing key as $sk_{\mathbf{v}} = e(g^{c_k}, g_{|V|-1}^{\prod_{i\neq j\in V} a_{i,x_i}})^{u_j}$.

Signing Oracle: The adversary $\mathcal{A}$ will query for a signature for a message $m \neq m^*$, and we let $m_j \neq m_j^*, j \in [k-1]$. Then $\mathcal{B}$ will be able to produce a valid signature:

$$\sigma = e(g^{c_k}, g_{k-2}^{\prod_{i\neq j\in[k-1]} a_{i,m_i}})^{u_j}.$$

Forgery: Eventually, $\mathcal{A}$ outputs a signature σ^* on message m^*. Then $\mathcal{B}$ outputs σ^* as the solution of the given instance of the k-MCDH assumption.

According to the public parameters built in the setup phase and the assumption that σ^* is valid, we know that $\sigma^* = g_{k-1}^{\prod_{i\in[k]} c_i}$, implies that σ^* is a solution for the given instance of the k-MCDH problem, and thus $\mathcal{B}$ breaks the k-MCDH assumption. It is clear that the view of $\mathcal{A}$ simulated by $\mathcal{B}$ in the above game is distributed statistically exponentially closely to that in the real unforgeability game, hence $\mathcal{B}$ succeeds whenever $\mathcal{A}$ does. □

Perfect Privacy. Given a valid signature (m^*, σ^*), we show that any signing key $sk_{\mathbf{v}}$ such that $p_{\mathbf{v}}(m^*) = 1$ could possibly have created it. The proof is straightforward.

According to the setup of the signing algorithm, for any tuple $(\mathbf{v}[0], \mathbf{v}[1], m^*)$ such that $p_{\mathbf{v}[0]}(m^*) = p_{\mathbf{v}[1]}(m^*) = 1$, which was chosen by an unbounded adversary $\mathcal{A}$, both of the signatures created by the signing key $sk_{\mathbf{v}[0]}$ and $sk_{\mathbf{v}[1]}$ are $g_{k-1}^{\alpha\cdot\prod_{i\in[k-1]} a_{i,m_i^*}}$. Therefore, any signing key $sk_{\mathbf{v}}$ such that $p_{\mathbf{v}}(m^*) = 1$ can compute a same signature on a given message m^*. The perfect privacy follows easily from this observation. □

5.2 PBS for Left/Right Predicates

Setup$(1^{\lambda}, k = 2\cdot s + 1)$**:** The setup algorithm takes as input a security parameter λ and an odd number $k = 2\cdot s + 1$. The algorithm then runs $\mathcal{G}(1^{\lambda}, k)$ that produces groups $\overrightarrow{\mathbb{G}} = (\mathbb{G}_1, \ldots, \mathbb{G}_k)$ of prime order p, with canonical generators $g_1, \ldots, g_k$, where we let $g = g_1$. Next it chooses random $\alpha \in \mathbb{Z}_p, (a_{1,0}, a_{1,1}), \ldots, (a_{s,0}, a_{s,1})$, $(b_{1,0}, b_{1,1}), \ldots, (b_{s,0}, b_{s,1}) \in \mathbb{Z}_p^2$ and computes $A_{i,\beta} = g^{a_{i,\beta}}$, $B_{i,\beta} = g^{b_{i,\beta}}$ for $i \in [s]$ and $\beta \in \{0,1\}$.

The master secret key is $msk = \alpha$. The public parameters, pp, consist of the group sequence description plus group elements $g^{\alpha}, \{A_{i,\beta}, B_{i,\beta} | i \in [s], \beta \in \{0,1\}\}$, message space $\mathcal{M} = \{0,1\}^{k-1}$, signature space $\mathcal{S} = \mathbb{G}_{k-1}$, and predicate space $\mathcal{P} = \{0,1\}^s$.

KeyGen$(msk, p = (p_x^{(L)}, p_y^{(R)}))$**:** For $(x,y) \in \{0,1\}^s$, the signing keys for the predicates $p_x^{(L)}, p_y^{(R)}$ are $sk_{p_x^{(L)}} = (g_s^{\prod_{i\in[s]} a_{i,x_i}})^{\alpha} \in \mathbb{G}_s$ and $sk_{p_y^{(R)}} = (g_s^{\prod_{i\in[s]} b_{i,y_i}})^{\alpha} \in \mathbb{G}_s$, respectively.

Sign(sk_p, m)**:** Given a message, m, of length $k-1$, let $m_1, \ldots, m_s, m_{s+1}, \ldots, m_{k-1}$ be the bits of this message. If the message m satisfies the left predicate $p_x^{(L)}$, i.e., $m_i = x_i$ for $i \in [s]$, then the signing algorithm can compute a signature:

$$\sigma = e(sk_{p_x^{(L)}}, B_{1,m_{s+1}}, \ldots, B_{s,m_{k-1}}) = g_{k-1}^{\alpha\cdot\prod_{i\in[s]} a_{i,m_i}\cdot\prod_{i\in[s]} b_{i,m_{s+i}}} \in \mathbb{G}_{k-1}.$$

If the message m satisfies the right predicate $p_y^{(R)}$, i.e., $m_{s+i} = y_i$ for $i \in [s]$, then the signing algorithm can compute a signature:

$$\sigma = e(sk_{p_y^{(R)}}, A_{1,m_1}, \ldots, A_{s,m_1}) = g_{k-1}^{\alpha \cdot \prod_{i\in[s]} a_{i,m_i} \cdot \prod_{i\in[s]} b_{i,m_{s+i}}} \in \mathbb{G}_{k-1}.$$

Verify(pp, m, σ)**:** Given a purported signature σ on a message m, verify the following equation:

$$e(\sigma, g) = e(g^{\alpha}, A_{1,m_1}, \ldots, A_{s,m_s}, B_{1,m_{s+1}}, \ldots, B_{s,m_{k-1}}).$$

Output 1 if it holds, else 0.

Correctness. The verification of the final signatures is justified by the following two equations:

$$\begin{aligned} e(\sigma, g) &= e((g_{k-1}^{\prod_{i\in[s]} a_{i,m_i} \cdot \prod_{i\in[s]} b_{i,m_{s+i}}})^{\alpha}, g) \\ &= g_k^{\alpha \cdot \prod_{i\in[s]} a_{i,m_i} \cdot \prod_{i\in[s]} b_{i,m_{s+i}}}. \end{aligned}$$

and

$$e(g^{\alpha}, A_{1,m_1}, \ldots, A_{s,m_s}, B_{1,m_{s+1}}, \ldots, B_{s,m_{k-1}}) = g_k^{\alpha \cdot \prod_{i\in[s]} a_{i,m_i} \cdot \prod_{i\in[s]} b_{i,m_{s+i}}}.$$

Theorem 5. *If the $k = (2s+1)$-MCDH assumption is hold in the multilinear groups, then the above PBS construction for left/right predicates and for messages of length s is selectively unforgeable and perfectly private.*

Selective Unforgeability. For length of messages $k - 1 = 2 \cdot s$, we prove security under the k-MCDH assumption. We show that if there exists a PPT adversary $\mathcal{A}$ on our PBS scheme for messages of length $k-1$ in the selective security game then we can construct an efficient algorithm $\mathcal{B}$ on the k-MCDH assumption. We describe how $\mathcal{B}$ interacts with $\mathcal{A}$.

The algorithm $\mathcal{B}$ first receives a $k = (2 \cdot s + 1)$-MCDH challenge consisting of the group sequence description $\overrightarrow{\mathbb{G}}$ and $g = g_1, g^{c_1}, \ldots, g^{c_k}$. It also receives challenge message $m^* = m_1^* \ldots m_s^* m_{s+1}^* \ldots m_{k-1} \in \{0,1\}^{k-1}$ from the adversary.

Setup: Initially, $\mathcal{B}$ chooses random $u_1, \ldots, u_s \in \mathbb{Z}_p$ and sets

$$A_{i,\beta} = \begin{cases} g^{c_i}, & \text{if } m_i^* = \beta \\ g^{u_i}, & \text{if } m_i^* \neq \beta \end{cases}$$

for $i \in [s], \beta \in \{0,1\}$. This corresponds to setting $a_{i,\beta} = c_i$ if $m_i^* = \beta$ and u_i otherwise.

It also chooses random $v_1, \ldots, v_s \in \mathbb{Z}_p$ and sets

$$B_{i,\beta} = \begin{cases} g^{c_{s+i}}, & \text{if } m_{s+i}^* = \beta \\ g^{v_i}, & \text{if } m_{s+i}^* \neq \beta \end{cases}$$

for $i \in [s], \beta \in \{0,1\}$. This corresponds to setting $b_{i,\beta} = c_{s+i}$ if $m_{s+i}^* = \beta$ and v_i otherwise. We observe these are distributed identically to the real scheme. In addition, it will internally view $\alpha = c_k$.

Key Generation Oracle: The adversary $\mathcal{A}$ will query for a secret key for a left predicate $p_x^{(L)}$ or a right predicate $p_y^{(R)}$.

If $p_x^{(L)}(m^*) = 1$ or $p_y^{(R)}(m^*) = 1$, then $\mathcal{B}$ aborts the game.

Otherwise, $p_x^{(L)}(m^*) = p_y^{(R)}(m^*) = 0$, then $\mathcal{B}$ will be able to create signing keys for the adversary, because his query will differ from the challenge message at least one bit. More specifically, for the left predicate $p_x^{(L)}$, we let $x_j \neq m_j^*, j \in [s]$, then $\mathcal{B}$ produces the delegation key as $sk_{p_x^{(L)}} = e(g^{c_k}, g_{s-1}^{\prod_{i\neq j\in[s]} a_{i,x_i}})^{u_j}$; for the right predicate $p_y^{(R)}$, we let $y_j \neq m_{s+j}^*, j \in [s]$, then $\mathcal{B}$ produces the signing key as $sk_{p_y^{(R)}} = e(g^{c_k}, g_{s-1}^{\prod_{i\neq j\in[s]} b_{i,y_i}})^{v_j}$.

Signing Oracle: The adversary $\mathcal{A}$ will query for a signature for a message $m \neq m^*$, and we let $m_j \neq m_j^*, j \in [k-1]$. Conceptually, $\mathcal{B}$ will be able to create signature for the adversary, because his query will differ from the challenge message in at least one bit. More specifically, $\mathcal{B}$ proceeds to make the signature according to the following two cases.

Case 1: If $j \in [s]$, $\mathcal{B}$ produces the signature as:

$$\sigma = e(g^{c_k}, g_{k-2}^{\prod_{i\neq j\in[s]} a_{i,m_i} \cdot \prod_{i\in[s]} b_{i,m_{s+i}}})^{u_j}.$$

Case 2: If $j \in \{s+1, \ldots, k-1\}$, $\mathcal{B}$ produces the signature as:

$$\sigma = e(g^{c_k}, g_{k-2}^{\prod_{i\in[s]} a_{i,m_i} \cdot \prod_{i\neq j\in[s]} b_{i,m_{s+i}}})^{v_j}.$$

Forgery: Eventually, $\mathcal{A}$ outputs a signature σ^* on message m^*. Then $\mathcal{B}$ outputs σ^* as the solution of the given instance of the k-MCDH assumption.

According to the public parameters built in the setup phase and the assumption that σ^* is valid, we know that $\sigma^* = g_{k-1}^{\prod_{i\in[k]} c_i}$, implies that σ^* is a solution for the given instance of the k-MCDH problem, and thus $\mathcal{B}$ breaks the k-MCDH assumption. It is clear that the view of $\mathcal{A}$ simulated by $\mathcal{B}$ in the above game is distributed statistically exponentially closely to that in the real unforgeability game, hence $\mathcal{B}$ succeeds whenever $\mathcal{A}$ does. □

Perfect Privacy: Given a valid signature (m^*, σ^*), we show that any signing key sk_p such that $p(m^*) = 1$ could possibly have created it. The proof is straightforward.

According to the setup of the signing algorithm, for any tuple (p_0, p_1, m^*) such that $p_0(m^*) = p_1(m^*) = 1$, which was chosen by any adversary $\mathcal{A}$, both of the signatures created by the signing key sk_{p_0} and sk_{p_1} are $(g_{k-1}^{\prod_{i\in[s]} a_{i,m_i^*} \cdot \prod_{i\in[s]} b_{i,m_{s+i}^*}})^{\alpha}$. Therefore, any signing key sk_p such that $p(m^*) = 1$ can compute a same signature on a given message m^*. The perfect privacy follows easily from this observation. □

5.3 PBS for Circuit Predicates

We now construct a policy-based signature scheme for boolean circuit predicates. Our circuit notion is from [5], please refer to [5] for details.

Setup$(1^\lambda, k = \ell + n + 1)$**:** The setup algorithm takes as input a security parameter λ, the maximum depth ℓ of a circuit and the length of the message n (it also is the number of boolean inputs).

The algorithm then runs $\mathcal{G}(1^\lambda, k = n + \ell + 1)$ that produces groups $\overrightarrow{\mathbb{G}} = (\mathbb{G}_1, \ldots, \mathbb{G}_k)$ of prime order p, with canonical generators $g_1, \ldots, g_k$, where we let $g = g_1$. Next it chooses random $\alpha \in \mathbb{Z}_p$ and $(a_{1,0}, a_{1,1}), \ldots, (a_{n,0}, a_{n,1}) \in \mathbb{Z}_p^2$ and computes $A_{i,\beta} = g^{a_{i,\beta}}$ for $i \in [n], \beta \in \{0,1\}$.

The master secret key is $msk = \alpha$. The public parameters, pk, consist of the group sequence description plus group elements $g_{\ell+1}^{\alpha}, \{A_{i,\beta} | i \in [n], \beta \in \{0,1\}\}$ and message space $\mathcal{M} = \{0,1\}^n$, signature space $\mathcal{S} = \mathbb{G}_{k-1}$, and predicate space $\mathcal{P} = \mathcal{C}$ that is the set of polynomial size circuit predicates.

KeyGen$(sk, p = (n, q, A, B, \texttt{GateType}))$**:** The key generation algorithm takes as input the master secret key msk and a description p of a circuit. The circuit has $n + q$ wires with n input wires, q gates and the $(n + q)$-th wire designated as the output wire.

The key generation algorithm chooses random integers $r_1, \ldots, r_{n+q-1} \in \mathbb{Z}_p$, where we think of the random value r_w as being associated with wire w. It sets $r_{n+q} = \alpha$.

Next, the algorithm generates key components for every wire w. The structure of the key components depends upon whether w is an input wire, an OR gate, or an AND gate. We describe how it generates components for each case.

- *Input wire.*
  By our convention if $w \in [n]$ then it corresponds to the w-th input. The key component is:
  $$K_w = g_2^{r_w a_{w,1}}.$$

- *OR gate.*
  Suppose that wire $w \in$ Gates and that $\texttt{GateType}(w) = \text{OR}$. In addition, let $j = \texttt{depth}(w)$ be the depth of the wire. The algorithm will choose random $a_w, b_w \in \mathbb{Z}_p$. Then the algorithm creates key components as:
  $$K_{w,1} = g^{a_w}, K_{w,2} = g^{b_w}, K_{w,3} = g_j^{r_w - a_w \cdot r_{A(w)}}, K_{w,4} = g_j^{r_w - b_w \cdot r_{B(w)}}.$$

- *AND gate.*
  Suppose that wire $w \in$ Gates and that $\texttt{GateType}(w) = \text{AND}$. In addition, let $j = \texttt{depth}(w)$ be the depth of wire w. The algorithm chooses random $a_w, b_w \in \mathbb{Z}_p$ and creates the key components as:
  $$K_{w,1} = g^{a_w}, K_{w,2} = g^{b_w}, K_{w,3} = g_j^{r_w - a_w \cdot r_{A(w)} - b_w \cdot r_{B(w)}}.$$

The signing key sk_p consists of the description of p along with these $n + q$ key components.

Sign$(sk_p, m \in \{0,1\}^n)$**:** The signing algorithm takes as input a signing key sk_p for a circuit predicate $p = (n, q, A, B, \texttt{GateType})$ and a message $m = m_1 \ldots m_n$. The algorithm first checks that $p(m) = 1$; if not it aborts.

The goal of the algorithm is to compute the signature $\sigma = g_{n+\ell}^{\alpha \cdot \prod_{i \in [n]} a_{i,m_i}} \in \mathbb{G}_{n+\ell}$. We will compute the circuit from the bottom up.

- *Input wire.*
  By our convention if $w \in [n]$ then it corresponds to the w-th input. Suppose that $m_w = p_w(m) = 1$. The algorithm computes $E_w = g_{n+1}^{r_w \cdot \prod_{i \neq w} a_{i,m_i}}$. Using the multilinear operation from A_{i,m_i} for $i \in [n] \neq w$. It then computes:
  $$E_w = e(K_w, g_{n-1}^{\prod_{i \neq w} a_{i,m_i}}) = e(g_2^{r_w a_{w,1}}, g_{n-1}^{\prod_{i \neq w} a_{i,m_i}}) = g_{n+1}^{r_w \prod_{i \in [n]} a_{i,m_i}}.$$

- *OR gate.*
  Consider a wire $w \in$ Gates and that $\texttt{GateType}(w) = \text{OR}$. In addition, let $j = \texttt{depth}(w)$ be the depth of the wire. For exposition we define $D(m) = g_n^{\prod_{i \in [n]} a_{i,m_i}}$. This is computable via the multilinear operation from A_{i,m_i} for $i \in [n]$. The computation is performed if $p_w(m) = 1$. If $p_{A(w)}(m) = 1$ (i.e., the first input evaluated to 1) then it computes:
  $$\begin{aligned} E_w &= e(E_{A(w)}, K_{w,1}) \cdot e(K_{w,3}, D(m)) \\ &= e(g_{j+n-1}^{r_{A(w)} \prod_{i \in [n]} a_{i,m_i}}, g^{a_w}) \\ &\quad \cdot e(g_j^{r_w - a_w \cdot r_{A(w)}}, g_n^{\prod_{i \in [n]} a_{i,m_i}}) \\ &= g_{j+n}^{r_w \cdot \prod_{i \in [n]} a_{i,m_i}}. \end{aligned}$$

  Otherwise, if $p_{A(w)}(m) = 0$ but $p_{B(w)}(m) = 1$, then it computes:
  $$\begin{aligned} E_w &= e(E_{B(w)}, K_{w,2}) \cdot e(K_{w,4}, D(m)) \\ &= e(g_{j+n-1}^{r_{B(w)} \prod_{i \in [n]} a_{i,m_i}}, g^{b_w}) \\ &\quad \cdot e(g_j^{r_w - b_w \cdot r_{B(w)}}, g_n^{\prod_{i \in [n]} a_{i,m_i}}) \\ &= g_{j+n}^{r_w \cdot \prod_{i \in [n]} a_{i,m_i}}. \end{aligned}$$

- *AND gate.*
  Consider a wire $w \in$ Gates and that $\texttt{GateType}(w) = \text{AND}$. In addition, let $j = \texttt{depth}(w)$ be the depth of the wire. The computation is performed if $p_w(m) = 1$ (i.e.,

$p_{A(w)}(m) = p_{B(w)}(m) = 1$) then it computes:

$$\begin{aligned} E_w &= e(E_{A(w)}, K_{w,1}) \cdot e(E_{B(w)}, K_{w,2}) \\ &\quad \cdot e(K_{w,3}, D(m)) \\ &= e(g_{j+n-1}^{r_{A(w)} \prod_i a_{i,m_i}}, g^{a_w}) \cdot e(g_{j+n-1}^{r_{B(w)} \prod_i a_{i,m_i}}, g^{b_w}) \\ &\quad \cdot e(g_j^{r_w - a_w \cdot r_{A(w)} - b_w \cdot r_{B(w)}}, g_n^{\prod_i a_{i,m_i}}) \\ &= g_{j+n}^{r_w \cdot \prod_i a_{i,m_i}}. \end{aligned}$$

The above procedures are evaluated in order for all w for which $p_w(m) = 1$. The final output of these procedures gives a signature:

$$\sigma = g_{n+\ell}^{r_{n+q} \prod_{i \in [n]} a_{i,m_i}} = g_{n+\ell}^{\alpha \cdot \prod_{i \in [n]} a_{i,m_i}} \in \mathbb{G}_{n+\ell}.$$

Verify$(pk, m \in \{0,1\}^n, \sigma \in \mathbb{G}_{k-1})$**:** Given a purported signature σ on a message m, verify the following equation:

$$e(\sigma, g) = e(g_{\ell+1}^{\alpha}, A_{1,m_1}, \ldots, A_{n,m_n}).$$

Output 1 if it holds, else 0.

Correctness. The verification of the signature is justified by the following two equations:

$$\begin{aligned} e(\sigma, g) &= e(g_{n+\ell}^{\alpha \cdot \prod_{i \in [n]} a_{i,m_i}}, g) \\ &= g_{n+\ell+1}^{\alpha \cdot \prod_{i \in [n]} a_{i,m_i}}. \end{aligned}$$

and

$$e(g_{\ell+1}^{\alpha}, A_{1,m_1}, \ldots, A_{n,m_n}) = g_{n+\ell+1}^{\alpha \cdot \prod_{i \in [n]} a_{i,m_i}}.$$

Theorem 6. *If the $k = (n+\ell+1)$-MCDH assumption is hold in the multilinear groups, then the above PBS construction for arbitrary circuits of depth ℓ and input length n, and messages of length n is selectively unforgeable and perfectly private.*

Selective Unforgeability. For length of messages n and a circuit of max depth ℓ and input length n, we prove security under the $k = (n + \ell + 1)$-Multilinear Computational Diffie-Hellman assumption.

We show that if there exists a PPT adversary $\mathcal{A}$ on our PBS scheme for messages of length s and circuits of depth ℓ and inputs of length n in the selective security game then we can construct an efficient algorithm $\mathcal{B}$ on the $(n + \ell + 1)$-MCDH assumption. We describe how $\mathcal{B}$ interacts with $\mathcal{A}$.

The algorithm $\mathcal{B}$ first receives a $k = (n+\ell+1)$-MCDH challenge consisting of the group sequence description $\overrightarrow{\mathbb{G}}$ and $g = g_1, g^{c_1}, \ldots, g^{c_k}$. It also receives challenge attribute message $m^* \in \{0,1\}^n$ from the adversary $\mathcal{A}$.

Setup: Initially, $\mathcal{B}$ chooses random $u_1, \ldots, u_n \in \mathbb{Z}_p$ and sets

$$A_{i,\beta} = \begin{cases} g^{c_i}, & \text{if } m_i^* = \beta \\ g^{u_i}, & \text{if } m_i^* \neq \beta \end{cases}$$

for $i \in [n], \beta \in \{0,1\}$. This corresponds to setting $a_{i,\beta} = c_i$ if $m_i^* = \beta$ and u_i otherwise. In addition, it will internally view $\alpha = c_{n+1} \cdot c_{n+2} \cdots c_{n+\ell+1}$.

Key Generation Oracle: The adversary $\mathcal{A}$ will query for a signing key for a circuit $p = (n, q, A, B, \texttt{GateType})$, where $p(m^*) = 0$. $\mathcal{B}$ proceeds to make the key. The idea for this oracle is same as in [5]. We will think have some invariant properties for each gate. Consider a gate w at depth j and the simulators viewpoint (symbolically) of r_w. If $p_w(x^*) = 0$, then the simulator will view r_w as the term $c_{n+1} \cdot c_{n+2} \cdots c_{n+j+1}$ plus some additional known randomization terms. If $p_w(x^*) = 1$, then the simulator will view r_w as the 0 plus some additional known randomization terms. If we can keep this property intact for simulating the keys up the circuit, the simulator will view r_{n+q} as $c_{n+1} \cdot c_{n+2} \cdots c_{n+\ell}$.

We describe how to create the key components for each wire w. Again, we organize key component creation into input wires, OR gates, and AND gates.

- *Input wire.*
  Suppose $w \in [n]$ and is therefore by convention an input wire.
  - If $(m^*)_w = 1$ then we choose random $r_w \leftarrow \mathbb{Z}_p$ (as is done honestly). The key component is:
    $$K_w = g_2^{r_w a_{w,1}}.$$
  - If $(m^*)_w = 0$ then we let $r_w = c_{n+1}c_{n+2} + \eta_w$ where $\eta_w \in \mathbb{Z}_p$ is a randomly chosen value. The key component is:
    $$K_w = (e(g^{c_{n+1}}, g^{c_{n+2}}) \cdot g_2^{\eta_w})^{u_w} = g_2^{r_w a_{w,1}}.$$
- *OR gate.*
  Suppose that wire $w \in$ Gates and that $\texttt{GateType}(w) = \text{OR}$. In addition, let $j = \texttt{depth}(w)$ be the depth of the wire.
  - If $p_w(x^*) = 1$, then algorithm will choose random $a_w, b_w, r_w \in \mathbb{Z}_p$. Then the algorithm creates key components as:
    $$K_{w,1} = g^{a_w}, K_{w,2} = g^{b_w}, K_{w,3} = g_j^{r_w - a_w \cdot r_{A(w)}}, K_{w,4} = g_j^{r_w - b_w \cdot r_{B(w)}}.$$
  - If $p_w(x^*) = 0$, then we set $a_w = c_{n+j+1} + \psi_w, b_w = c_{n+j+1} + \phi_w$, and $r_w = c_{n+1} \cdot c_{n+2} \cdots c_{n+j+1} + \eta_w$, where ψ_w, ϕ_w, η_w are chosen randomly. Then the algorithm creates key components as:
    $$K_{w,1} = g^{c_{n+j+1} + \psi_w}, K_{w,2} = g^{c_{n+j+1} + \phi_w},$$
    $$K_{w,3} = g_j^{\eta_w - c_{c+j+1}\eta_{A(w)} - \psi_w(c_{n+1} \cdots c_{n+j} + \eta_{A(w)})},$$
    $$K_{w,4} = g_j^{\eta_w - c_{c+j+1}\eta_{B(w)} - \psi_w(c_{n+1} \cdots c_{n+j} + \eta_{B(w)})}.$$
    $\mathcal{B}$ can create the last two key components due to a cancellation. Since both the $A(w)$ and $B(w)$ gates evaluated to 0, we have $r_{A(w)} = c_{n+1} \cdots c_{n+j} + \eta_{A(w)}$ and similarly for $r_{B(w)}$. Note that $g_j^{c_{n+1} \cdots c_{n+j}}$ is always using the multilinear maps.

- *AND gate.*
  Suppose that wire $w \in$ Gates and that $\texttt{GateType}(w) =$ AND. In addition, let $j = \texttt{depth}(w)$ be the depth of wire w.
  - If $p_w(x^*) = 1$, then the algorithm chooses random $a_w, b_w, r_w \in \mathbb{Z}_p$ and creates the key components as:
    $$K_{w,1} = g^{a_w}, K_{w,2} = g^{b_w},$$
    $$K_{w,3} = g_j^{r_w - a_w \cdot r_{A(w)} - b_w \cdot r_{B(w)}}.$$
  - If $p_w(x^*) = 0$ and $p_{A(w)}(x^*) = 0$, then we let $a_w = c_{n+j+1} + \psi_w, b_w = \phi_w$, and $r_w = c_{n+1} \cdot c_{n+2} \cdots c_{n+j+1} + \eta_w$, where ψ_w, ϕ_w, η_w are chosen randomly. Then the algorithm creates key components as:
    $$K_{w,1} = g^{c_{n+j+1}+\psi_w}, K_{w,2} = g^{\phi_w},$$
    $$K_{w,3} = g_j^{\eta_w - \psi_w c_{n+1} \cdots c_{n+j} - (c_{n+j+1}+\psi_w)\eta_{A(w)} - \phi_w r_{B(w)}}.$$
    $\mathcal{B}$ can create the last key component due to a cancellation. Since the $A(w)$ gate evaluated to 0, we have $r_{A(w)} = c_{n+1} \cdots c_{n+j} + \eta_{A(w)}$. Note that $g_j^{r_{B(w)}}$ always computable regardless of whether $p_{A(w)}(x^*)$ evaluated to 0 or 1, since $g_j^{c_{n+1} \cdots c_{n+j}}$ is always using the multilinear maps.

  The case where $p_{B(w)}(x^*) = 0$ and $p_{A(w)}(x^*) = 1$ is performed in a symmetric to what is above, with the roles of a_w and b_w reversed.

Signing Oracle: The adversary $\mathcal{A}$ will query for a signature for a message $m \neq m^*$, and we let $m_j \neq m_j^*$. $\mathcal{B}$ can produce a valid signature $\sigma = (g_{n+\ell}^{\prod_{i \neq j \in [k]} c_i})^{u_j}$ by knowing the exponent u_j.

Forgery: Eventually, $\mathcal{A}$ outputs an attribute signature σ^* on message m^*. Then $\mathcal{B}$ outputs σ^* as the solution of the given instance of the $k = (n+\ell+1)$-MCDH assumption. According to the public key built in the setup phase and the assumption that σ^* is valid, we know that $\sigma^* = g_{k-1}^{\prod_{i \in [k]} c_i}$, implies that σ^* is a solution for the given instance of the k-MCDH problem, and thus $\mathcal{B}$ breaks the k-MCDH assumption.

It is clear that the view of $\mathcal{A}$ simulated by $\mathcal{B}$ in the above game is distributed statistically exponentially closely to that in the real unforgeability game, hence $\mathcal{B}$ succeeds whenever $\mathcal{A}$ does. □

Perfect Privacy. Given a valid signature $(m^* \in \{0,1\}^n, \sigma^* \in \mathbb{G}_{k-1})$, we show that any signing key sk_p such that $p(m^*) = 1$ could possibly have created it. The proof is straight-forward.

According to the setup of the signing algorithm, for any tuple (p_0, p_1, m^*) such that $p_0(m^*) = p_1(m^*) = 1$, which was chosen by an unbounded adversary $\mathcal{A}$, both of the signatures created by the signing key sk_{p_0} and sk_{p_1} are $g_{n+\ell}^{\alpha \cdot \prod_{i \in [n]} a_{i,m_i^*}}$. Therefore, any signing key sk_p such that $p(m^*) = 1$ can compute a same signature on a given message m^*. The perfect privacy follows easily from this observation. □

6 Conclusion

In this work, we introduce the notion of policy-based signature for predicates. In such a signature scheme, signers can sign messages that conform to some predicate, yet privacy of the predicate is maintained. Then, we construct a policy-based signature scheme for prefix predicate based on tree-based signature scheme. Furthermore, we also construct several policy-based signature schemes for bit-fixing predicate, left/right predicate, and circuits predicate, respectively, based on multilinear maps.

Acknowledgments

This study was supported by the Science and technology research project of Chongqing Municipal Education Commission (No. KJ1600445). The authors gratefully acknowledge the anonymous reviewers for their valuable comments.

References

[1] M. Bellare and G. Fuchsbauer, "Policy-based signatures", *The 17th IACR International Conference on Practice and Theory of Public-Key Cryptography (PKC'14)*, pp. 520–537, Buenos Aires, Argentina, 2014.

[2] E. Boyle, S. Goldwasser, and I. Ivan, "Functional signatures and pseudorandom functions", *The 17th IACR International Conference on Practice and Theory of Public-Key Cryptography (PKC'14)*, pp. 501–519, Buenos Aires, Argentina, 2014.

[3] X. Boyen, "Mesh signatures", *The 26th Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT'07)*, pp. 210–227, Barcelona, Spain, 2007.

[4] D. Boneh and A. Silverberg, "Applications of multilinear forms to cryptography", *Contemporary Mathematics*, vol. 324, no. 1, pp. 71–90, 2003.

[5] D. Boneh and B. Waters, "Constrained pseudorandom functions and their applications", *The 19th International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT'13)*, pp. 280–300, Bengaluru, India, 2013.

[6] D. Chaum and E. Van Heyst, "Group signatures", *Workshop on the Theory and Application of Cryptographic Techniques (EUROCRYPT'91)*, pp. 257–265, Brighton, UK, 1991.

[7] M. L. Chande, C. C. Lee, and C. T. Li, "Message recovery via an efficient multi-proxy signature with self-certified keys", *International Journal of Network Security*, vol. 19, no. 3, pp. 340–346, 2017.

[8] J. S. Coron, T. Lepoint, and M. Tibouchi, "Practical multilinear maps over the integers", *The 33rd Annual Cryptology Conference (CRYPTO'13)*, pp. 476–493, Santa Barbara, CA, USA, 2013.

[9] S. Garg, C. Gentry, and S. Halevi, "Candidate multilinear maps from ideal lattices", *The 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT'13)*, pp. 1–17, Athens, Greece, 2013.

[10] C. Gentry, S. Gorbunov, and S. Halevi, "Graph-induced multilinear maps from lattices", *The 12th IACR Theory of Cryptography Conference (TCC'15)*, pp. 498–527, Warsaw, Poland, 2015.

[11] J. Groth and A. Sahai, "Efficient non-interactive proof systems for bilinear groups", *The 27th Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT'08)*, pp. 415–432, Istanbul, Turkey, 2008.

[12] Y. Hu and H. Jia, "Cryptanalysis of GGH map", *The 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT'16)*, pp. 537–565, Vienna, Austria, 2016.

[13] S. Hohenberger, A. Sahai, and B. Waters, "Full domain hash from (leveled) multilinear maps and identity-based aggregate signatures", *The 33rd Annual Cryptology Conference (CRYPTO'13)*, pp. 494–512, Santa Barbara, CA, USA, 2013.

[14] M. Ibrahim, "Resisting traitors in linkable democratic group signatures", *International Journal of Network Security*, vol. 9, no. 1, pp. 51–60, 2009.

[15] E. Kiltz, A. Mityagin, S. Panjwani, and B. Raghavan, "Append-only signatures", *The 32nd International Colloquium on Automata, Languages and Programming (ICALP'05)*, pp. 434–445, Lisbon, Portugal, 2005.

[16] L. Lamport, "Constructing digital signatures from a one-way function", *Technical Report SRI-CSL-98*, SRI Intl. Computer Science Laboratory, Oct. 1979.

[17] C. C. Lee, T. C. Lin, S. F. Tzeng, M. S. Hwang, Generalization of proxy signature based on factorization", *International Journal of Innovative Computing, Information and Control*, vol. 7, no. 3, pp. 1039–1054, 2011.

[18] H. T. Lee and J. H. Seo, "Security analysis of multilinear maps over the integers", *The 34th Annual Cryptology Conference (CRYPTO'14)*, pp. 224–240, Santa Barbara, CA, USA, 2014.

[19] R. C. Merkle, "A certified digital signature (that antique paper from 1979)", *Annual Cryptology Conference (CRYPTO'89)*, pp. 218–238, Santa Barbara, CA, USA, 1989.

[20] H. K. Maji, M. Prabhakaran, and M. Rosulek, "Attribute-based signatures", *The Cryptographers' Track at the RSA Conference (CT-RSA'11)*, pp. 376–392, San Francisco, CA, USA, 2011.

[21] M. Mambo, K. Usuda, and E. Okamoto, "Proxy signatures for delegating signing operation", *The 3rd ACM conference on computer and communications security (CCS'96)*, pp. 48–57, New Delhi, India, 1996.

[22] Z. Qin, H. Xiong, and F. Li, "A Provably Secure Certificate Based Ring Signature Without Pairing", *International Journal of Network Security*, vol. 16, no. 4, pp. 278–285, 2014.

[23] R. L. Rivest, A. Shamir, and Y. Tauman, "How to leak a secret", *The 7th International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT'01)*, pp. 552–565, Gold Coast, Australia, 2001.

[24] F. Tang, H. Li, and B. Liang, "Attribute-based signature for circuits from multilinear maps", *The 17th Information Security Conference (ISC'14)*, pp. 54–71, Hongkong, China, 2014.

[25] S. Zeng, Y. Huang, and X. Liu, "Privacy-preserving communication for VANETs with conditionally anonymous ring signature", *International Journal of Network Security*, vol. 17, no. 2, pp. 135–141, 2015.

Biography

Fei Tang received his Ph.D. from the Institute of Information Engineering of Chinese Academy of Sciences in 2015. He is currently a lecturer of the College of Cyberspace Security and Law, Chongqing University of Posts and Telecommunications. His research interest is public key cryptography.

Yousheng Zhou is currently an associate professor of the College of Cyberspace Security and Law, Chongqing University of Posts and Telecommunications. He received his Ph.D. from Beijing University of Posts and Telecommunications in 2011. He completed one-year postdoctorate work at Dublin City University, Ireland in 2016. His research interests include network security and cloud security.