Appendix

A Security of IO-Compatible Cryptographic Primitives

In this section, we formally define the security properties of the IO-compatible primitives from Section 2.3.

A.1 Security of Puncturable Pseudorandom Function

► **Selective Pseudorandomness at Punctured Points**: This property of a PPRF is defined through the following experiment between an adversary $\mathcal{B}$ and a challenger $\mathcal{C}$:

- $\mathcal{B}$ submits a challenge input $x^* \in \mathcal{X}_{\text{PPRF}}$ to $\mathcal{C}$.
- $\mathcal{C}$ chooses uniformly at random a PPRF key $K^* \xleftarrow{\$} \mathcal{K}_{\text{PPRF}}$ and a random bit $\hat{b} \xleftarrow{\$} \{0,1\}$. It computes the punctured key $K^*\{x^*\} \xleftarrow{\$} \mathcal{F}.\mathsf{Puncture}(K^*, x^*)$. If $\hat{b} = 0$, it sets $r^* = \mathcal{F}(K^*, x^*)$. Otherwise, it selects $r^* \xleftarrow{\$} \mathcal{Y}_{\text{PPRF}}$. It sends back $(K^*\{x^*\}, r^*)$ to $\mathcal{B}$.
- $\mathcal{B}$ outputs a guess bit $\hat{b}' \in \{0,1\}$.

The PPRF $\mathcal{F}$ is selectively pseudorandom at punctured points if for any PPT adversary $\mathcal{B}$, for any security parameter λ,

$$\mathsf{Adv}_{\mathcal{B}}^{\mathcal{F},\text{SEL-PR}}(\lambda) = |\mathsf{Pr}[\hat{b} = \hat{b}'] - 1/2| \leq \mathsf{negl}(\lambda)$$

for some negligible function negl.

A.2 Security of Somewhere Statistically Binding Hash Function

► **Index Hiding**: The index hiding property of an SSB hash is defined through the following experiment between an adversary $\mathcal{B}$ and a challenger $\mathcal{C}$:

- $\mathcal{B}$ chooses an integer $n_{\text{SSB-BLK}} \leq 2^\lambda$ together with a pair of indices $i_0^*, i_1^* \in [0, n_{\text{SSB-BLK}} - 1]$, and submits them to $\mathcal{C}$.
- $\mathcal{C}$ selects a random bit $\hat{b} \xleftarrow{\$} \{0,1\}$ and computes $\text{HK} \xleftarrow{\$} \mathsf{SSB.Gen}(1^\lambda, n_{\text{SSB-BLK}}, i_{\hat{b}}^*)$, and returns HK to $\mathcal{B}$.
- $\mathcal{B}$ eventually outputs a guess bit $\hat{b}' \in \{0,1\}$.

The SSB hash is said to be index hiding if for any PPT adversary $\mathcal{B}$, for any security parameter λ,

$$\mathsf{Adv}_{\mathcal{B}}^{\text{SSB,IH}}(\lambda) = |\mathsf{Pr}[\hat{b} = \hat{b}'] - 1/2| \leq \mathsf{negl}(\lambda)$$

for some negligible function negl.

► **Somewhere Statistically Binding**: An SSB hash key HK is said to be statistically binding for an index $i^* \in [0, n_{\text{SSB-BLK}} - 1]$ if there do not exist any $h \in \{0,1\}^{\ell_{\text{SSB-HASH}}}, u \neq u' \in \Sigma_{\text{SSB-BLK}}$, and $\pi_{\text{SSB}}, \pi'_{\text{SSB}} \in \Pi_{\text{SSB}}$ such that $\mathsf{SSB.Verify}(\text{HK}, h, i^*, u, \pi_{\text{SSB}}) = 1 = \mathsf{SSB.Verify}(\text{HK}, h, i^*, u', \pi'_{\text{SSB}})$.

The SSB hash is defined to be somewhere statistically binding if for any security parameter λ, integer $n_{\text{SSB-BLK}} \leq 2^\lambda$, index $i^* \in [0, n_{\text{SSB-BLK}} - 1]$, the hash key $\text{HK} \xleftarrow{\$} \mathsf{SSB.Gen}(1^\lambda, n_{\text{SSB-BLK}}, i^*)$ is statistically binding for i^*. Note that this is an information theoretic property.

A.3 Security of Positional Accumulator

► **Indistinguishability of Read Setup**: This property of a positional accumulator is defined through the following experiment between an adversary $\mathcal{B}$ and a challenger $\mathcal{C}$:

- $\mathcal{B}$ chooses a bound $n_{\text{ACC-BLK}} \leq 2^\lambda$ of the number of blocks, κ symbol-index pairs $((x_1, i_1), \ldots, (x_\kappa, i_\kappa)) \in (\Sigma_{\text{ACC-BLK}} \times [0, n_{\text{ACC-BLK}} - 1])^\kappa$, and an index $i^* \in [0, n_{\text{ACC-BLK}} - 1]$. It submits all of those to $\mathcal{C}$.
- $\mathcal{C}$ selects a random bit $\hat{b} \xleftarrow{\$} \{0,1\}$. If $\hat{b} = 0$, $\mathcal{C}$ generates $(\text{PP}_{\text{ACC}}, w_0, \text{STORE}_0) \xleftarrow{\$} \mathsf{ACC.Setup}(1^\lambda, n_{\text{ACC-BLK}})$. Otherwise, $\mathcal{C}$ generates $(\text{PP}_{\text{ACC}}, w_0, \text{STORE}_0) \xleftarrow{\$} \mathsf{ACC.Setup\text{-}Enforce\text{-}Read}(1^\lambda, n_{\text{ACC-BLK}}, ((x_1, i_1), \ldots, (x_\kappa, i_\kappa)), i^*)$. It returns $(\text{PP}_{\text{ACC}}, w_0, \text{STORE}_0)$ to $\mathcal{B}$.
- $\mathcal{B}$ outputs a guess bit $\hat{b}' \in \{0,1\}$.

The positional accumulator is said to satisfy indistinguishability of read setup if for any PPT adversary $\mathcal{B}$, for any security parameter λ, we have

$$\mathsf{Adv}_{\mathcal{B}}^{\text{ACC,IND-READ}}(\lambda) = |\mathsf{Pr}[\hat{b} = \hat{b}'] - 1/2| \leq \mathsf{negl}(\lambda)$$

for some negligible function negl.

► **Indistinguishability of Write Setup**: This property of a positional accumulator is defined through the following experiment between an adversary $\mathcal{B}$ and a challenger $\mathcal{C}$:

- $\mathcal{B}$ chooses a bound $n_{\text{ACC-BLK}} \leq 2^\lambda$ of the number of blocks, and κ symbol-index pairs $((x_1, i_1), \ldots, (x_\kappa, i_\kappa)) \in (\Sigma_{\text{ACC-BLK}} \times [0, n_{\text{ACC-BLK}} - 1])^\kappa$. It submits all of those to $\mathcal{C}$.
- $\mathcal{C}$ selects a random bit $\hat{b} \xleftarrow{\$} \{0,1\}$. If $\hat{b} = 0$, $\mathcal{C}$ generates $(\text{PP}_{\text{ACC}}, w_0, \text{STORE}_0) \xleftarrow{\$} \mathsf{ACC.Setup}(1^\lambda, n_{\text{ACC-BLK}})$. Otherwise, $\mathcal{C}$ generates $(\text{PP}_{\text{ACC}}, w_0, \text{STORE}_0) \xleftarrow{\$} \mathsf{ACC.Setup\text{-}Enforce\text{-}Write}(1^\lambda, n_{\text{ACC-BLK}}, ((x_1, i_1), \ldots, (x_\kappa, i_\kappa)))$. It returns $(\text{PP}_{\text{ACC}}, w_0, \text{STORE}_0)$ to $\mathcal{B}$.
- $\mathcal{B}$ outputs a guess bit $\hat{b}' \in \{0,1\}$.

A positional accumulator is said to satisfy indistinguishability of write setup if for any PPT adversary $\mathcal{B}$, for any security parameter λ, we have

$$\mathsf{Adv}_{\mathcal{B}}^{\text{ACC,IND-WRITE}}(\lambda) = |\mathsf{Pr}[\hat{b} = \hat{b}'] - 1/2| \leq \mathsf{negl}(\lambda)$$

for some negligible function negl.

► **Read Enforcing**: Consider any security parameter λ, $n_{\text{ACC-BLK}} \leq 2^\lambda$, $((x_1, i_1), \ldots, (x_\kappa, i_\kappa)) \in (\Sigma_{\text{ACC-BLK}} \times [0, n_{\text{ACC-BLK}} - 1])^\kappa$, and $i^* \in [0, n_{\text{ACC-BLK}} - 1]$. Let $(\text{PP}_{\text{ACC}}, w_0, \text{STORE}_0) \xleftarrow{\$} \mathsf{ACC.Setup\text{-}Enforce\text{-}Read}(1^\lambda, n_{\text{ACC-BLK}}, ((x_1, i_1), \ldots, (x_\kappa, i_\kappa)), i^*)$. For $j = 1, \ldots, \kappa$, iteratively define the following:

- $\text{STORE}_j = \mathsf{ACC.Write\text{-}Store}(\text{PP}_{\text{ACC}}, \text{STORE}_{j-1}, i_j, x_j)$
- $\text{AUX}_j = \mathsf{ACC.Prep\text{-}Write}(\text{PP}_{\text{ACC}}, \text{STORE}_{j-1}, i_j)$
- $w_j = \mathsf{ACC.Update}(\text{PP}_{\text{ACC}}, w_{j-1}, x_j, i_j, \text{AUX}_j)$

The positional accumulator is said to be read enforcing if $\mathsf{ACC.Verify\text{-}Read}(\text{PP}_{\text{ACC}}, w_\kappa, x_{\text{IN}}, i^*, \pi_{\text{ACC}}) = 1$ implies either $[i^* \notin \{i_1, \ldots, i_\kappa\}] \wedge [x_{\text{IN}} = \epsilon]$ or $x_{\text{IN}} = x_j$ for the largest $j \in [\kappa]$ such that $i_j = i^*$. Note that this is an information theoretic property.

► **Write Enforcing**: Consider any security parameter λ, $n_{\text{ACC-BLK}} \leq 2^\lambda$, and $((x_1, i_1), \ldots, (x_\kappa, i_\kappa)) \in (\Sigma_{\text{ACC-BLK}} \times [0, n_{\text{ACC-BLK}} - 1])^\kappa$. Let $(\text{PP}_{\text{ACC}}, w_0, \text{STORE}_0) \xleftarrow{\$} \mathsf{ACC.Setup\text{-}}$

$\mathsf{Enforce\text{-}Write}(1^\lambda, n_{\text{ACC-BLK}}, ((x_1, i_1), \ldots, (x_\kappa, i_\kappa)))$. For $j = 1, \ldots, \kappa$, iteratively define the following:

- $\text{STORE}_j = \mathsf{ACC.Write\text{-}Store}(\text{PP}_{\text{ACC}}, \text{STORE}_{j-1}, i_j, x_j)$
- $\text{AUX}_j = \mathsf{ACC.Prep\text{-}Write}(\text{PP}_{\text{ACC}}, \text{STORE}_{j-1}, i_j)$
- $w_j = \mathsf{ACC.Update}(\text{PP}_{\text{ACC}}, w_{j-1}, x_j, i_j, \text{AUX}_j)$

The positional accumulator is defined to be write enforcing if $\mathsf{ACC.Update}(\text{PP}_{\text{ACC}}, w_{\kappa-1}, x_\kappa, i_\kappa, \text{AUX}) = w_{\text{OUT}} \neq \perp$, for any AUX, implies $w_{\text{OUT}} = w_\kappa$. Observe that this is an information theoretic property as well.

A.4 Security of Iterator

► **Indistinguishability of Enforcing Setup**: This property of a cryptographic iterator is defined through the following experiment between an adversary $\mathcal{B}$ and a challenger $\mathcal{C}$:

- $\mathcal{B}$ chooses an integer bound $n_{\text{ITR}} \leq 2^\lambda$, along with a sequence of κ messages $(\mu_1, \ldots, \mu_\kappa) \in \mathcal{M}_{\text{ITR}}^\kappa$, and submits them to $\mathcal{C}$.
- $\mathcal{C}$ selects a random bit $\hat{b} \xleftarrow{\$} \{0,1\}$. If $\hat{b} = 0$, $\mathcal{C}$ generates $(\text{PP}_{\text{ITR}}, v_0) \xleftarrow{\$} \mathsf{ITR.Setup}(1^\lambda, n_{\text{ITR}})$. Else, $\mathcal{C}$ generates $(\text{PP}_{\text{ITR}}, v_0) \xleftarrow{\$} \mathsf{ITR.Setup\text{-}Enforce}(1^\lambda, n_{\text{ITR}}, (\mu_1, \ldots, \mu_\kappa))$. It sends back $(\text{PP}_{\text{ITR}}, v_0)$ to $\mathcal{B}$.
- $\mathcal{B}$ outputs a guess bit $\hat{b}' \in \{0,1\}$.

The cryptographic iterator is said to satisfy indistinguishability of enforcing setup if for any PPT adversary $\mathcal{B}$, for any security parameter λ,

$$\mathsf{Adv}_{\mathcal{B}}^{\text{ITR,IND-ENF}}(\lambda) = |\mathsf{Pr}[\hat{b} = \hat{b}'] - 1/2| \leq \mathsf{negl}(\lambda)$$

for some negligible function negl.

► **Enforcing**: Consider any security parameter λ, $n_{\text{ITR}} \leq 2^\lambda, \kappa \leq n_{\text{ITR}}$, and $(\mu_1, \ldots, \mu_\kappa) \in \mathcal{M}_{\text{ITR}}^\kappa$. Let $(\text{PP}_{\text{ITR}}, v_0) \xleftarrow{\$} \mathsf{ITR.Setup\text{-}Enforce}(1^\lambda, n_{\text{ITR}}, (\mu_1, \ldots, \mu_\kappa))$ and $v_j = \mathsf{ITR.Iterate}^j(\text{PP}_{\text{ITR}}, v_0, (\mu_1, \ldots, \mu_j))$ for all $j \in [\kappa]$. The cryptographic iterator is said to be enforcing if $v_k = \mathsf{ITR.Iterate}(\text{PP}_{\text{ITR}}, v', \mu')$ implies $(v', \mu') = (v_{\kappa-1}, \mu_\kappa)$. Note that this is an information theoretic property.

A.5 Security of Splittable Signature

► $\text{VK}_{\text{SPS-REJ}}$ **Indistinguishability**: This property of a splittable signature scheme is defined through the following experiment between an adversary $\mathcal{B}$ and a challenger $\mathcal{C}$:

- $\mathcal{C}$ generates $(\text{SK}_{\text{SPS}}, \text{VK}_{\text{SPS}}, \text{VK}_{\text{SPS-REJ}}) \xleftarrow{\$} \mathsf{SPS.Setup}(1^\lambda)$. Next it chooses a random bit $\hat{b} \xleftarrow{\$} \{0,1\}$. If $\hat{b} = 0$, it sends VK_{SPS} to $\mathcal{B}$. Otherwise, it sends $\text{VK}_{\text{SPS-REJ}}$ to $\mathcal{B}$.
- $\mathcal{B}$ outputs a guess bit $\hat{b}' \in \{0,1\}$.

The splittable signature scheme is said to be $\text{VK}_{\text{SPS-REJ}}$ indistinguishable if for any PPT adversary $\mathcal{B}$, for any security parameter λ,

$$\mathsf{Adv}_{\mathcal{B}}^{\text{SPS,IND-REJ}}(\lambda) = |\mathsf{Pr}[\hat{b} = \hat{b}'] - 1/2| \leq \mathsf{negl}(\lambda)$$

for some negligible function negl.

► $\text{VK}_{\text{SPS-ONE}}$ **Indistinguishability**: This feature of a splittable signature scheme is defined through the following experiment between an adversary $\mathcal{B}$ and a challenger $\mathcal{C}$:

- $\mathcal{B}$ submits a message $m^* \in \mathcal{M}_{\text{SPS}}$ to $\mathcal{C}$.
- $\mathcal{C}$ generates $(\text{SK}_{\text{SPS}}, \text{VK}_{\text{SPS}}, \text{VK}_{\text{SPS-REJ}}) \xleftarrow{\$} \mathsf{SPS.Setup}(1^\lambda)$. Next it computes $(\sigma_{\text{SPS-ONE},m^*}, \text{VK}_{\text{SPS-ONE}}, \text{SK}_{\text{SPS-ABO}}, \text{VK}_{\text{SPS-ABO}}) \xleftarrow{\$} \mathsf{SPS.Split}(\text{SK}_{\text{SPS}}, m^*)$. Then it chooses a random bit $\hat{b} \xleftarrow{\$} \{0,1\}$. If $\hat{b} = 0$, it returns $(\sigma_{\text{SPS-ONE},m^*}, \text{VK}_{\text{SPS-ONE}})$ to $\mathcal{B}$. Else, it returns $(\sigma_{\text{SPS-ONE},m^*}, \text{VK}_{\text{SPS}})$ to $\mathcal{B}$.
- $\mathcal{B}$ outputs a guess bit $\hat{b}' \in \{0,1\}$.

The splittable signature scheme is said to be $\text{VK}_{\text{SPS-ONE}}$ indistinguishable if for any PPT adversary $\mathcal{B}$, for any security parameter λ,

$$\mathsf{Adv}_{\mathcal{B}}^{\text{SPS,IND-ONE}}(\lambda) = |\mathsf{Pr}[\hat{b} = \hat{b}'] - 1/2| \leq \mathsf{negl}(\lambda)$$

for some negligible function negl.

► **$\text{VK}_{\text{SPS-ABO}}$ Indistinguishability**: This feature of a splittable signature scheme is defined through the following experiment between an adversary $\mathcal{B}$ and a challenger $\mathcal{C}$:

- $\mathcal{B}$ submits a message $m^* \in \mathcal{M}_{\text{SPS}}$ to $\mathcal{C}$.
- $\mathcal{C}$ generates $(\text{SK}_{\text{SPS}}, \text{VK}_{\text{SPS}}, \text{VK}_{\text{SPS-REJ}}) \xleftarrow{\$} \mathsf{SPS.Setup}(1^\lambda)$. Next it computes $(\sigma_{\text{SPS-ONE},m^*}, \text{VK}_{\text{SPS-ONE}}, \text{SK}_{\text{SPS-ABO}}, \text{VK}_{\text{SPS-ABO}}) \xleftarrow{\$} \mathsf{SPS.Split}(\text{SK}_{\text{SPS}}, m^*)$. Then it chooses a random bit $\hat{b} \xleftarrow{\$} \{0,1\}$. If $\hat{b} = 0$, it returns $(\text{SK}_{\text{SPS-ABO}}, \text{VK}_{\text{SPS-ABO}})$ to $\mathcal{B}$. Else, it returns $(\text{SK}_{\text{SPS-ABO}}, \text{VK}_{\text{SPS}})$ to $\mathcal{B}$.
- $\mathcal{B}$ outputs a guess bit $\hat{b}' \in \{0,1\}$.

The splittable signature scheme is said to be $\text{VK}_{\text{SPS-ABO}}$ indistinguishable if for any PPT adversary $\mathcal{B}$, for any security parameter λ,

$$\mathsf{Adv}_{\mathcal{B}}^{\text{SPS,IND-ABO}}(\lambda) = |\mathsf{Pr}[\hat{b} = \hat{b}'] - 1/2| \leq \mathsf{negl}(\lambda)$$

for some negligible function negl.

► **Splitting Indistinguishability**: This feature of a splittable signature scheme is defined through the following experiment between an adversary $\mathcal{B}$ and a challenger $\mathcal{C}$:

- $\mathcal{B}$ submits a message $m^* \in \mathcal{M}_{\text{SPS}}$ to $\mathcal{C}$.
- $\mathcal{C}$ forms $(\text{SK}_{\text{SPS}}, \text{VK}_{\text{SPS}}, \text{VK}_{\text{SPS-REJ}}) \xleftarrow{\$} \mathsf{SPS.Setup}(1^\lambda)$ and $(\text{SK}'_{\text{SPS}}, \text{VK}'_{\text{SPS}}, \text{VK}'_{\text{SPS-REJ}}) \xleftarrow{\$} \mathsf{SPS.Setup}(1^\lambda)$. Next it computes $(\sigma_{\text{SPS-ONE},m^*}, \text{VK}_{\text{SPS-ONE}}, \text{SK}_{\text{SPS-ABO}}, \text{VK}_{\text{SPS-ABO}}) \xleftarrow{\$} \mathsf{SPS.Split}(\text{SK}_{\text{SPS}}, m^*)$ as well as $(\sigma'_{\text{SPS-ONE},m^*}, \text{VK}'_{\text{SPS-ONE}}, \text{SK}'_{\text{SPS-ABO}}, \text{VK}'_{\text{SPS-ABO}}) \xleftarrow{\$} \mathsf{SPS.Split}(\text{SK}'_{\text{SPS}}, m^*)$. Then it chooses a random bit $\hat{b} \xleftarrow{\$} \{0,1\}$. If $\hat{b} = 0$, it returns $(\sigma_{\text{SPS-ONE},m^*}, \text{VK}_{\text{SPS-ONE}}, \text{SK}_{\text{SPS-ABO}}, \text{VK}_{\text{SPS-ABO}})$ to $\mathcal{B}$. Else, it returns $(\sigma_{\text{SPS-ONE},m^*}, \text{VK}_{\text{SPS-ONE}}, \text{SK}'_{\text{SPS-ABO}}, \text{VK}'_{\text{SPS-ABO}})$ to $\mathcal{B}$.
- $\mathcal{B}$ outputs a guess bit $\hat{b}' \in \{0,1\}$.

The splittable signature scheme is said to be splitting indistinguishable if for any PPT adversary $\mathcal{B}$, for any security parameter λ,

$$\mathsf{Adv}_{\mathcal{B}}^{\text{SPS,IND-SPL}}(\lambda) = |\mathsf{Pr}[\hat{b} = \hat{b}'] - 1/2| \leq \mathsf{negl}(\lambda)$$

for some negligible function negl.

B Lemmas for the Proof of Theorem 4.1

Lemma B.1 *Assuming $\mathcal{IO}$ is a secure indistinguishability obfuscator for $\mathsf{P/poly}$, $\mathcal{F}$ is a secure puncturable pseudorandom function, SSB is a somewhere statistically*

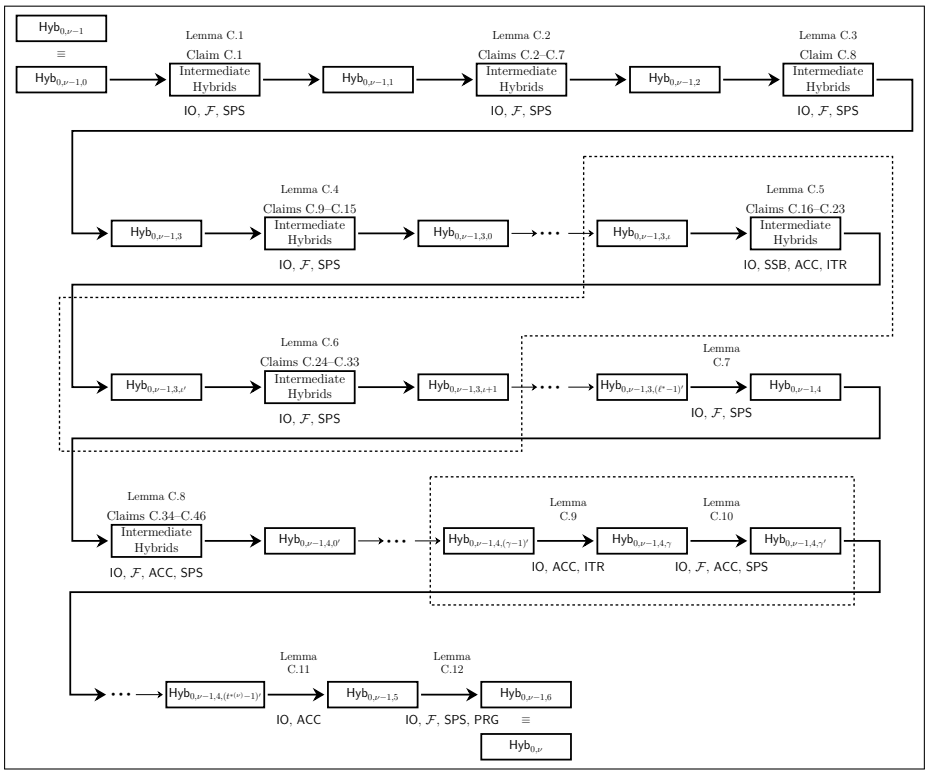

Fig. 9 Structure of the Hybrid Reduction Proving Lemma B.1

binding hash function, ACC *is a secure positional accumulator,* ITR *is a secure cryptographic iterator,* SPS *is a secure splittable signature scheme, and* PRG *is a secure injective pseudorandom generator, for any* PPT *adversary* $\mathcal{A}$*, for any security parameter* λ*,* $|Adv_{\mathcal{A}}^{(0,\nu-1)}(\lambda) - Adv_{\mathcal{A}}^{(0,\nu)}(\lambda)| \leq \mathsf{negl}(\lambda)$ *for some negligible function* negl*.*

Proof The proof of our Lemma B.1 extends the ideas involved in the security proof for the message-hiding encoding of [17]. Lemma 1 in the security proof of the CPRF construction of [7] also employs a similar technique. However, as mentioned earlier, we extend the technique of [7] to support adaptive signing key queries of the adversary as stipulated in our unforgeability experiment. We will first provide a complete description of the sequence of hybrid experiments involved in the proof of Lemma B.1 and then provide the analysis of those hybrid experiments providing the details for only those segments which are technically distinct from [7]. Please refer to Fig. **9** for an overview of the hybrid transitions and their analysis.

Let $t^{*(\nu)}$ denotes the running time of the TM $M^{(\nu)} \in \mathbb{M}_\lambda$, queried by the adversary $\mathcal{A}$, on input the challenge string x^* and $2^{\tau^{*(\nu)}}$ be the smallest power of two greater than $t^{*(\nu)}$. The sequence of intermediate hybrid experiments between $\mathsf{Hyb}_{0,\nu-1}$ and $\mathsf{Hyb}_{0,\nu}$ are described below:

Sequence of Intermediate Hybrids between $\mathsf{Hyb}_{0,\nu-1}$ and $\mathsf{Hyb}_{0,\nu}$

$\mathsf{Hyb}_{0,\nu-1,0}$: This experiment coincides with $\mathsf{Hyb}_{0,\nu-1}$.

$\mathsf{Hyb}_{0,\nu-1,1}$: This experiment if analogous to $\mathsf{Hyb}_{0,\nu-1,0}$ except that to answer the ν^{th} signing key query of $\mathcal{A}$ corresponding to TM $M^{(\nu)} \in \mathbb{M}_\lambda$ with $M^{(\nu)}(x^*) = 0$, $\mathcal{B}$ proceeds as follows:

1. It first picks PPRF keys $K_1^{(\nu)}, \ldots, K_\lambda^{(\nu)}, K_{\text{SPS},A}^{(\nu)}, \underline{K_{\text{SPS},B}^{(\nu)}}, K_{\text{SPS},E}^{(\nu)} \xleftarrow{\$} \mathcal{F}.\mathsf{Setup}(1^\lambda)$.
2. Next, it generates $(\text{PP}_{\text{ACC}}^{(\nu)}, w_0^{(\nu)}, \text{STORE}_0^{(\nu)}) \xleftarrow{\$} \mathsf{ACC.Setup}(1^\lambda, n_{\text{ACC-BLK}} = 2^\lambda)$ and $(\text{PP}_{\text{ITR}}^{(\nu)}, v_0^{(\nu)}) \xleftarrow{\$} \mathsf{ITR.Setup}(1^\lambda, n_{\text{ITR}} = 2^\lambda)$.
3. It provides $\mathcal{A}$ with the signing key
$$\text{SK}_{\text{ABS}}\{M^{(\nu)}\} = \left(\begin{array}{l} \text{HK}, \text{PP}_{\text{ACC}}^{(\nu)}, w_0^{(\nu)}, \text{STORE}_0^{(\nu)}, \text{PP}_{\text{ITR}}^{(\nu)}, v_0^{(\nu)}, \\ \mathcal{IO}(\mathsf{Init\text{-}SPS.Prog}[q_0^{(\nu)}, w_0^{(\nu)}, v_0^{(\nu)}, K_{\text{SPS},E}^{(\nu)}]), \\ \mathcal{IO}(\mathsf{Accumulate.Prog}[n_{\text{SSB-BLK}} = 2^\lambda, \text{HK}, \text{PP}_{\text{ACC}}^{(\nu)}, \text{PP}_{\text{ITR}}^{(\nu)}, K_{\text{SPS},E}^{(\nu)}]), \\ \mathcal{IO}(\mathsf{Change\text{-}SPS.Prog}[K_{\text{SPS},A}^{(\nu)}, K_{\text{SPS},E}^{(\nu)}]), \\ \mathcal{IO}(\underline{\mathsf{Constrained\text{-}Key.Prog}_{\text{ABS}}^{(1)}[M^{(\nu)}, T = 2^\lambda, t^{*(\nu)}, \text{PP}_{\text{ACC}}^{(\nu)}, \text{PP}_{\text{ITR}}^{(\nu)}, K,} \\ \quad \underline{K_1^{(\nu)}, \ldots, K_\lambda^{(\nu)}, K_{\text{SPS},A}^{(\nu)}, K_{\text{SPS},B}^{(\nu)}, h^*, \ell^*])} \end{array}\right),$$
where the program $\mathsf{Constrained\text{-}Key.Prog}_{\text{ABS}}^{(1)}$ is a modification of the program $\mathsf{Constrained\text{-}Key.Prog}'_{\text{ABS}}$ (Fig. **7**) and is depicted in Fig. **10**.

$\mathsf{Hyb}_{0,\nu-1,2}$: In this experiment, in response to the ν^{th} signing key query of $\mathcal{A}$ corresponding to TM $M^{(\nu)} \in \mathbb{M}_\lambda$ with $M^{(\nu)}(x^*) = 0$, $\mathcal{B}$ performs the following steps:

1. It chooses PPRF keys $K_1^{(\nu)}, \ldots, K_\lambda^{(\nu)}, K_{\text{SPS},A}^{(\nu)}, K_{\text{SPS},B}^{(\nu)}, K_{\text{SPS},E}^{(\nu)}, \underline{K_{\text{SPS},F}^{(\nu)}} \xleftarrow{\$} \mathcal{F}.\mathsf{Setup}(1^\lambda)$.
2. Next, it generates $(\text{PP}_{\text{ACC}}^{(\nu)}, w_0^{(\nu)}, \text{STORE}_0^{(\nu)}) \xleftarrow{\$} \mathsf{ACC.Setup}(1^\lambda, n_{\text{ACC-BLK}} = 2^\lambda)$ and $(\text{PP}_{\text{ITR}}^{(\nu)}, v_0^{(\nu)}) \xleftarrow{\$} \mathsf{ITR.Setup}(1^\lambda, n_{\text{ITR}} = 2^\lambda)$.
3. It provides $\mathcal{A}$ with the signing key
$$\text{SK}_{\text{ABS}}\{M^{(\nu)}\} = \left(\begin{array}{l} \text{HK}, \text{PP}_{\text{ACC}}^{(\nu)}, w_0^{(\nu)}, \text{STORE}_0^{(\nu)}, \text{PP}_{\text{ITR}}^{(\nu)}, v_0^{(\nu)}, \\ \mathcal{IO}(\mathsf{Init\text{-}SPS.Prog}[q_0^{(\nu)}, w_0^{(\nu)}, v_0^{(\nu)}, K_{\text{SPS},E}^{(\nu)}]), \\ \mathcal{IO}(\underline{\mathsf{Accumulate.Prog}^{(1)}[n_{\text{SSB-BLK}} = 2^\lambda, \text{HK}, \text{PP}_{\text{ACC}}^{(\nu)}, \text{PP}_{\text{ITR}}^{(\nu)}, K_{\text{SPS},E}^{(\nu)},} \\ \quad \underline{K_{\text{SPS},F}^{(\nu)}, h^*, \ell^*]}), \\ \mathcal{IO}(\underline{\mathsf{Change\text{-}SPS.Prog}^{(1)}[K_{\text{SPS},A}^{(\nu)}, K_{\text{SPS},B}^{(\nu)}, K_{\text{SPS},E}^{(\nu)}, K_{\text{SPS},F}^{(\nu)}, h^*, \ell^*]}), \\ \mathcal{IO}(\underline{\mathsf{Constrained\text{-}Key.Prog}_{\text{ABS}}^{(1)}[M^{(\nu)}, T = 2^\lambda, t^{*(\nu)}, \text{PP}_{\text{ACC}}^{(\nu)}, \text{PP}_{\text{ITR}}^{(\nu)}, K,} \\ \quad K_1^{(\nu)}, \ldots, K_\lambda^{(\nu)}, K_{\text{SPS},A}^{(\nu)}, K_{\text{SPS},B}^{(\nu)}, h^*, \ell^*]) \end{array}\right),$$
where the programs $\mathsf{Accumulate.Prog}^{(1)}$ and $\mathsf{Change\text{-}SPS.Prog}^{(1)}$ are modifications of the programs $\mathsf{Accumulate.Prog}$ and $\mathsf{Change\text{-}SPS.Prog}$ (Figs. **3** and **4**) and are depicted in Figs. **11** and **12** respectively.

The rest of the experiment proceeds in the same way as in $\mathsf{Hyb}_{0,\nu-1,1}$.

Constants: TM $M = \langle Q, \Sigma_{\text{INP}}, \Sigma_{\text{TAPE}}, \delta, q_0, q_{\text{AC}}, q_{\text{REJ}} \rangle$, Time bound $T = 2^\lambda$, Running time on challenge input t^*, Public parameters for positional accumulator PP_{ACC}, Public parameters for iterator PP_{ITR}, PPRF keys $K, K_1, \ldots, K_\lambda, K_{\text{SPS},A}, K_{\text{SPS},B}$, SSB hash value of challenge input h^*, Length of challenge input ℓ^*

Inputs: Time t, String SEED_{IN}, Header position POS_{IN}, Symbol SYM_{IN}, TM state ST_{IN}, Accumulator value w_{IN}, Accumulator proof π_{ACC}, Auxiliary value AUX, Iterator value v_{IN}, SSB hash value h, Length ℓ_{INP}, Signature $\sigma_{\text{SPS,IN}}$

Output: CPRF evaluation $\mathcal{F}(K, (h, \ell_{\text{INP}}))$, or (Header Position POS_{OUT}, Symbol SYM_{OUT}, TM state ST_{OUT}, Accumulator value w_{OUT}, Iterator value v_{OUT}, Signature $\sigma_{\text{SPS,OUT}}$, String SEED_{OUT}), or $\perp$

1. Identify an integer τ such that $2^\tau \leq t < 2^{\tau+1}$. If $[\mathsf{PRG}(\text{SEED}_{\text{IN}}) \neq \mathsf{PRG}(\mathcal{F}(K_\tau, (h, \ell_{\text{INP}})))] \wedge [t > 1]$, output $\perp$.
2. If $\mathsf{ACC.Verify\text{-}Read}(\text{PP}_{\text{ACC}}, w_{\text{IN}}, \text{SYM}_{\text{IN}}, \text{POS}_{\text{IN}}, \pi_{\text{ACC}}) = 0$, output $\perp$.
3. (a) Compute $r_{\text{SPS},A} = \mathcal{F}(K_{\text{SPS},A}, (h, \ell_{\text{INP}}, t-1)), (\text{SK}_{\text{SPS},A}, \text{VK}_{\text{SPS},A}, \text{VK}_{\text{SPS-REJ},A}) = \mathsf{SPS.Setup}(1^\lambda; r_{\text{SPS},A})$.
   (b) Compute $r_{\text{SPS},B} = \mathcal{F}(K_{\text{SPS},B}, (h, \ell_{\text{INP}}, t-1)), (\text{SK}_{\text{SPS},B}, \text{VK}_{\text{SPS},B}, \text{VK}_{\text{SPS-REJ},B}) = \mathsf{SPS.Setup}(1^\lambda; r_{\text{SPS},B})$.
   (c) Set $m_{\text{IN}} = (v_{\text{IN}}, \text{ST}_{\text{IN}}, w_{\text{IN}}, \text{POS}_{\text{IN}})$ and $\alpha =$'-'.
   (d) If $\mathsf{SPS.Verify}(\text{VK}_{\text{SPS},A}, m_{\text{IN}}, \sigma_{\text{SPS,IN}}) = 1$, set $\alpha =$'A'.
   (e) If $[\alpha =$'-'$] \wedge [(t > t^*) \vee (h \neq h^*) \vee (\ell_{\text{INP}} \neq \ell^*)]$, output $\perp$.
   Else if $[\alpha =$'-'$] \wedge [\mathsf{SPS.Verify}(\text{VK}_{\text{SPS},B}, m_{\text{IN}}, \sigma_{\text{SPS,IN}}) = 1]$, set $\alpha =$'B'.
   (f) If $\alpha =$'-', output $\perp$.
4. (a) Compute $(\text{ST}_{\text{OUT}}, \text{SYM}_{\text{OUT}}, \beta) = \delta(\text{ST}_{\text{IN}}, \text{SYM}_{\text{IN}})$ and $\text{POS}_{\text{OUT}} = \text{POS}_{\text{IN}} + \beta$.
   (b) If $\text{ST}_{\text{OUT}} = q_{\text{REJ}}$, output $\perp$.
   Else if $[\text{ST}_{\text{OUT}} = q_{\text{AC}}] \wedge [\alpha =$'$B$'$]$, output $\perp$.
   Else if $\text{ST}_{\text{OUT}} = q_{\text{AC}}$, perform the following:
   (I) Compute $r_{\text{SIG}} = \mathcal{F}(K, (h, \ell_{\text{INP}})), (\text{SK}_{\text{SIG}}, \text{VK}_{\text{SIG}}) = \mathsf{SIG.Setup}(1^\lambda; r_{\text{SIG}})$.
   (II) Output $(\text{SK}_{\text{SIG}}, \text{VK}_{\text{SIG}})$.
5. (a) Compute $w_{\text{OUT}} = \mathsf{ACC.Update}(\text{PP}_{\text{ACC}}, w_{\text{IN}}, \text{SYM}_{\text{OUT}}, \text{POS}_{\text{IN}}, \text{AUX})$. If $w_{\text{OUT}} = \perp$, output $\perp$.
   (b) Compute $v_{\text{OUT}} = \mathsf{ITR.Iterate}(\text{PP}_{\text{ITR}}, v_{\text{IN}}, (\text{ST}_{\text{IN}}, w_{\text{IN}}, \text{POS}_{\text{IN}}))$.
6. (a) Compute $r'_{\text{SPS},A} = \mathcal{F}(K_{\text{SPS},A}, (h, \ell_{\text{INP}}, t)), (\text{SK}'_{\text{SPS},A}, \text{VK}'_{\text{SPS},A}, \text{VK}'_{\text{SPS-REJ},A}) = \mathsf{SPS.Setup}(1^\lambda; r'_{\text{SPS},A})$.
   (b) Compute $r'_{\text{SPS},B} = \mathcal{F}(K_{\text{SPS},B}, (h, \ell_{\text{INP}}, t)), (\text{SK}'_{\text{SPS},B}, \text{VK}'_{\text{SPS},B}, \text{VK}'_{\text{SPS-REJ},B}) = \mathsf{SPS.Setup}(1^\lambda; r'_{\text{SPS},B})$.
   (c) Set $m_{\text{OUT}} = (v_{\text{OUT}}, \text{ST}_{\text{OUT}}, w_{\text{OUT}}, \text{POS}_{\text{OUT}})$.
   Compute $\sigma_{\text{SPS,OUT}} = \mathsf{SPS.Sign}(\text{SK}'_{\text{SPS},\alpha}, m_{\text{OUT}})$.
7. If $t + 1 = 2^{\tau'}$, for some integer τ', set $\text{SEED}_{\text{OUT}} = \mathcal{F}(K_{\tau'}, (h, \ell_{\text{INP}}))$.
   Else, set $\text{SEED}_{\text{OUT}} = \epsilon$.
8. Output $(\text{POS}_{\text{OUT}}, \text{SYM}_{\text{OUT}}, \text{ST}_{\text{OUT}}, w_{\text{OUT}}, v_{\text{OUT}}, \sigma_{\text{SPS,OUT}}, \text{SEED}_{\text{OUT}})$.

Fig. 10 $\mathsf{Constrained\text{-}Key.Prog}^{(1)}_{\text{ABS}}$

Constants: Maximum number of blocks for SSB hash $n_{\text{SSB-BLK}} = 2^\lambda$, SSB hash key HK, Public parameters for positional accumulator PP_{ACC}, Public parameters for iterator PP_{ITR}, PPRF keys $K_{\text{SPS},E}, K_{\text{SPS},F}$, SSB hash value of challenge input h^*, Length of challenge input ℓ^*

Inputs: Index i, Symbol SYM_{IN}, TM state ST, Accumulator value w_{IN}, Auxiliary value AUX, Iterator value v_{IN}, Signature $\sigma_{\text{SPS,IN}}$, SSB hash value h, SSB opening value π_{SSB}

Output: (Accumulator value w_{OUT}, Iterator value v_{OUT}, Signature $\sigma_{\text{SPS-OUT}}$), or $\perp$

1. (a) Compute $r_{\text{SPS},E} = \mathcal{F}(K_{\text{SPS},E}, (h, i)), (\text{SK}_{\text{SPS},E}, \text{VK}_{\text{SPS},E}, \text{VK}_{\text{SPS-REJ},E}) = \mathsf{SPS.Setup}(1^\lambda; r_{\text{SPS},E})$.
   (b) Compute $r_{\text{SPS},F} = \mathcal{F}(K_{\text{SPS},F}, (h, i)), (\text{SK}_{\text{SPS},F}, \text{VK}_{\text{SPS},F}, \text{VK}_{\text{SPS-REJ},F}) = \mathsf{SPS.Setup}(1^\lambda; r_{\text{SPS},F})$.
   (c) Set $m_{\text{IN}} = (v_{\text{IN}}, \text{ST}, w_{\text{IN}}, 0)$ and $\alpha =$'-'.
   (d) If $\mathsf{SPS.Verify}(\text{VK}_{\text{SPS},E}, m_{\text{IN}}, \sigma_{\text{SPS,IN}}) = 1$, set $\alpha =$'E'.
   (e) If $[\alpha =$'-'$] \wedge [(i \neq \ell^*) \vee (h \neq h^*)]$, output $\perp$.
   Else if $[\alpha =$'-'$] \wedge [\mathsf{SPS.Verify}(\text{VK}_{\text{SPS},F}, m_{\text{IN}}, \sigma_{\text{SPS,IN}}) = 1]$, set $\alpha =$'F'.
   (f) If $\alpha =$'-', output $\perp$.
2. If $\mathsf{SSB.Verify}(\text{HK}, h, i, \text{SYM}_{\text{IN}}, \pi_{\text{SSB}}) = 0$, output $\perp$.
3. (a) Compute $w_{\text{OUT}} = \mathsf{ACC.Update}(\text{PP}_{\text{ACC}}, w_{\text{IN}}, \text{SYM}_{\text{IN}}, i, \text{AUX})$. If $w_{\text{OUT}} = \perp$, output $\perp$.
   (b) Compute $v_{\text{OUT}} = \mathsf{ITR.Iterate}(\text{PP}_{\text{ITR}}, v_{\text{IN}}, (\text{ST}, w_{\text{IN}}, 0))$.
4. (a) Compute $r'_{\text{SPS},E} = \mathcal{F}(K_{\text{SPS},E}, (h, i+1)), (\text{SK}'_{\text{SPS},E}, \text{VK}'_{\text{SPS},E}, \text{VK}'_{\text{SPS-REJ},E}) = \mathsf{SPS.Setup}(1^\lambda; r'_{\text{SPS},E})$.
   (b) Set $m_{\text{OUT}} = (v_{\text{OUT}}, \text{ST}, w_{\text{OUT}}, 0)$. Compute $\sigma_{\text{SPS,OUT}} = \mathsf{SPS.Sign}(\text{SK}'_{\text{SPS},E}, m_{\text{OUT}})$.
5. Output $(w_{\text{OUT}}, v_{\text{OUT}}, \sigma_{\text{SPS,OUT}})$.

Fig. 11 $\mathsf{Accumulate.Prog}^{(1)}$

Constants: PPRF keys $K_{\text{SPS},A}$, $K_{\text{SPS},B}$, $K_{\text{SPS},E}$, $K_{\text{SPS},F}$, SSB hash value of challenge input h^*, Length of challenge input ℓ^*

Inputs: TM state ST, Accumulator value w, Iterator value v, SSB hash value h, Length ℓ_{INP}, Signature $\sigma_{\text{SPS,IN}}$

Output: Signature $\sigma_{\text{SPS,OUT}}$, or $\perp$

1. (a) Compute $r_{\text{SPS},E} = \mathcal{F}(K_{\text{SPS},E}, (h, \ell_{\text{INP}}))$, $(\text{SK}_{\text{SPS},E}, \text{VK}_{\text{SPS},E}, \text{VK}_{\text{SPS-REJ},E}) = \mathsf{SPS.Setup}(1^\lambda; r_{\text{SPS},E})$.
   (b) Compute $r_{\text{SPS},F} = \mathcal{F}(K_{\text{SPS},F}, (h, \ell_{\text{INP}}))$, $(\text{SK}_{\text{SPS},F}, \text{VK}_{\text{SPS},F}, \text{VK}_{\text{SPS-REJ},F}) = \mathsf{SPS.Setup}(1^\lambda; r_{\text{SPS},F})$.
   (c) Set $m = (v, \text{ST}, w, 0)$ and α='-'.
   (d) If $\mathsf{SPS.Verify}(\text{VK}_{\text{SPS},E}, m, \sigma_{\text{SPS,IN}}) = 1$, set α ='E'.
   (e) If $[\alpha$ ='-'$] \wedge [(\ell_{\text{INP}} \neq \ell^*) \vee (h \neq h^*)]$, output $\perp$.
   Else if $[\alpha$ ='-'$] \wedge [\mathsf{SPS.Verify}(\text{VK}_{\text{SPS},F}, m, \sigma_{\text{SPS,IN}}) = 1]$, set α ='F'.
   (f) If α ='-', output $\perp$.
2. (a) Compute $r_{\text{SPS},A} = \mathcal{F}(K_{\text{SPS},A}, (h, \ell_{\text{INP}}, 0)), (\text{SK}_{\text{SPS},A}, \text{VK}_{\text{SPS},A}, \text{VK}_{\text{SPS-REJ},A}) = \mathsf{SPS.Setup}(1^\lambda; r_{\text{SPS},A})$.
   (b) Compute $r_{\text{SPS},B} = \mathcal{F}(K_{\text{SPS},B}, (h, \ell_{\text{INP}}, 0)), (\text{SK}_{\text{SPS},B}, \text{VK}_{\text{SPS},B}, \text{VK}_{\text{SPS-REJ},B}) = \mathsf{SPS.Setup}(1^\lambda; r_{\text{DSPS},B})$.
   (c) If $[(h, \ell_{\text{INP}}) = (h^*, \ell^*)] \wedge [\alpha$ ='F'$]$, output $\sigma_{\text{SPS,OUT}} = \mathsf{SPS.Sign}(\text{SK}_{\text{SPS},B}, m)$.
   Else, output $\sigma_{\text{SPS,OUT}} = \mathsf{SPS.Sign}(\text{SK}_{\text{SPS},A}, m)$.

Fig. 12 $\mathsf{Change\text{-}SPS.Prog}^{(1)}$

$\mathsf{Hyb}_{0,\nu-1,3}$: In this experiment, to answer the ν^{th} signing key query of $\mathcal{A}$ corresponding to TM $M^{(\nu)} \in \mathbb{M}_\lambda$ with $M^{(\nu)}(x^*) = 0$, $\mathcal{B}$ generates all the PPRF keys as well as the public parameters for the positional accumulator and iterator just as in $\mathsf{Hyb}_{0,\nu-1,2}$, however, it returns the signing key

$$\text{SK}_{\text{ABS}}\{M^{(\nu)}\} = \left(\begin{array}{l} \text{HK}, \text{PP}_{\text{ACC}}^{(\nu)}, w_0^{(\nu)}, \text{STORE}_0^{(\nu)}, \text{PP}_{\text{ITR}}^{(\nu)}, v_0^{(\nu)}, \\ \mathcal{IO}(\mathsf{Init\text{-}SPS.Prog}[q_0^{(\nu)}, w_0^{(\nu)}, v_0^{(\nu)}, K_{\text{SPS},E}^{(\nu)}]), \\ \mathcal{IO}(\mathsf{Accumulate.Prog}^{(2)}[n_{\text{SSB-BLK}} = 2^\lambda, \text{HK}, \text{PP}_{\text{ACC}}^{(\nu)}, \text{PP}_{\text{ITR}}^{(\nu)}, K_{\text{SPS},E}^{(\nu)}, \\ \quad K_{\text{SPS},F}^{(\nu)}, h^*, \ell^*]), \\ \mathcal{IO}(\mathsf{Change\text{-}SPS.Prog}^{(2)}[K_{\text{SPS},A}^{(\nu)}, K_{\text{SPS},B}^{(\nu)}, K_{\text{SPS},E}^{(\nu)}, K_{\text{SPS},F}^{(\nu)}, h^*, \ell^*]), \\ \mathcal{IO}(\mathsf{Constrained\text{-}Key.Prog}_{\text{ABS}}^{(1)}[M^{(\nu)}, T = 2^\lambda, t^{*(\nu)}, \text{PP}_{\text{ACC}}^{(\nu)}, \text{PP}_{\text{ITR}}^{(\nu)}, K, \\ \quad K_1^{(\nu)}, \ldots, K_\lambda^{(\nu)}, K_{\text{SPS},A}^{(\nu)}, K_{\text{SPS},B}^{(\nu)}, h^*, \ell^*]) \end{array}\right),$$

where the programs $\mathsf{Accumulate.Prog}^{(2)}$ and $\mathsf{Change\text{-}SPS.Prog}^{(2)}$ are modifications of the programs $\mathsf{Accumulate.Prog}^{(1)}$ and $\mathsf{Change\text{-}SPS.Prog}^{(1)}$ (Figs. **11** and **12**) and are depicted in Figs. **13** and **14** respectively. The remaining part of the experiment is similar to $\mathsf{Hyb}_{0,\nu-1,2}$.

Constants: Maximum number of blocks for SSB hash $n_{\text{SSB-BLK}} = 2^\lambda$, SSB hash key HK, Public parameters for positional accumulator PP_{ACC}, Public parameters for iterator PP_{ITR}, PPRF keys $K_{\text{SPS},E}$, $K_{\text{SPS},F}$, SSB hash value of challenge input h^*, Length of challenge input ℓ^*

Inputs: Index i, Symbol SYM_{IN}, TM state ST, Accumulator value w_{IN}, Auxiliary value AUX, Iterator value v_{IN}, Signature $\sigma_{\text{SPS,IN}}$, SSB hash value h, SSB opening value π_{SSB}

Output: (Accumulator value w_{OUT}, Iterator value v_{OUT}, Signature $\sigma_{\text{SPS-OUT}}$), or $\perp$

1. (a) Compute $r_{\text{SPS},E} = \mathcal{F}(K_{\text{SPS},E}, (h, i))$, $(\text{SK}_{\text{SPS},E}, \text{VK}_{\text{SPS},E}, \text{VK}_{\text{SPS-REJ},E}) = \mathsf{SPS.Setup}(1^\lambda; r_{\text{SPS},E})$.
   (b) Compute $r_{\text{SPS},F} = \mathcal{F}(K_{\text{SPS},F}, (h, i))$, $(\text{SK}_{\text{SPS},F}, \text{VK}_{\text{SPS},F}, \text{VK}_{\text{SPS-REJ},F}) = \mathsf{SPS.Setup}(1^\lambda; r_{\text{SPS},F})$.
   (c) Set $m_{\text{IN}} = (v_{\text{IN}}, \text{ST}, w_{\text{IN}}, 0)$ and $\alpha =$'-'.
   (d) If $\mathsf{SPS.Verify}(\text{VK}_{\text{SPS},E}, m_{\text{IN}}, \sigma_{\text{SPS,IN}}) = 1$, set $\alpha =$'E'.
   (e) If $[\alpha =\text{'-'}] \wedge [\underline{(i > \ell^*) \vee (i = 0)} \vee (h \neq h^*)]$, output $\perp$.
   Else if $[\alpha =\text{'-'}] \wedge [\mathsf{SPS.Verify}(\text{VK}_{\text{SPS},F}, m_{\text{IN}}, \sigma_{\text{SPS,IN}}) = 1]$, set $\alpha =$'F'.
   (f) If $\alpha =$'-', output $\perp$.
2. If $\mathsf{SSB.Verify}(\text{HK}, h, i, \text{SYM}_{\text{IN}}, \pi_{\text{SSB}}) = 0$, output $\perp$.
3. (a) Compute $w_{\text{OUT}} = \mathsf{ACC.Update}(\text{PP}_{\text{ACC}}, w_{\text{IN}}, \text{SYM}_{\text{IN}}, i, \text{AUX})$. If $w_{\text{OUT}} = \perp$, output $\perp$.
   (b) Compute $v_{\text{OUT}} = \mathsf{ITR.Iterate}(\text{PP}_{\text{ITR}}, v_{\text{IN}}, (\text{ST}, w_{\text{IN}}, 0))$.
4. (a) Compute $r'_{\text{SPS},E} = \mathcal{F}(K_{\text{SPS},E}, (h, i+1))$, $(\text{SK}'_{\text{SPS},E}, \text{VK}'_{\text{SPS},E}, \text{VK}'_{\text{SPS-REJ},E}) = \mathsf{SPS.Setup}(1^\lambda; r'_{\text{SPS},E})$.
   (b) Compute $\underline{r'_{\text{SPS},F} = \mathcal{F}(K_{\text{SPS},F}, (h, i+1)), (\text{SK}'_{\text{SPS},F}, \text{VK}'_{\text{SPS},F}, \text{VK}'_{\text{SPS-REJ},F}) = \mathsf{SPS.Setup}(1^\lambda; r'_{\text{SPS},F})}$.
   (c) Set $m_{\text{OUT}} = (v_{\text{OUT}}, \text{ST}, w_{\text{OUT}}, 0)$.
   $\underline{\text{If } i < \ell^*, \text{ compute } \sigma_{\text{SPS,OUT}} = \mathsf{SPS.Sign}(\text{SK}'_{\text{SPS},\alpha}, m_{\text{OUT}}).}$
   $\underline{\text{Else, compute } \sigma_{\text{SPS,OUT}} = \mathsf{SPS.Sign}(\text{SK}'_{\text{SPS},E}, m_{\text{OUT}}).}$
5. Output $(w_{\text{OUT}}, v_{\text{OUT}}, \sigma_{\text{SPS,OUT}})$.

Fig. 13 $\mathsf{Accumulate.Prog}^{(2)}$

Constants: PPRF keys $K_{\text{SPS},A}$, $K_{\text{SPS},B}$, $K_{\text{SPS},E}$, $K_{\text{SPS},F}$, SSB hash value of challenge input h^*, Length of challenge input ℓ^*

Inputs: TM state ST, Accumulator value w, Iterator value v, SSB hash value h, Length ℓ_{INP}, Signature $\sigma_{\text{SPS,IN}}$

Output: Signature $\sigma_{\text{SPS,OUT}}$, or $\perp$

1. (a) Compute $r_{\text{SPS},E} = \mathcal{F}(K_{\text{SPS},E}, (h, \ell_{\text{INP}}))$, $(\text{SK}_{\text{SPS},E}, \text{VK}_{\text{SPS},E}, \text{VK}_{\text{SPS-REJ},E}) = \mathsf{SPS.Setup}(1^\lambda; r_{\text{SPS},E})$.
   (b) Compute $r_{\text{SPS},F} = \mathcal{F}(K_{\text{SPS},F}, (h, \ell_{\text{INP}}))$, $(\text{SK}_{\text{SPS},F}, \text{VK}_{\text{SPS},F}, \text{VK}_{\text{SPS-REJ},F}) = \mathsf{SPS.Setup}(1^\lambda; r_{\text{SPS},F})$.
   (c) Set $m = (v, \text{ST}, w, 0)$ and α='-'.
   (d) If $\mathsf{SPS.Verify}(\text{VK}_{\text{SPS},E}, m, \sigma_{\text{SPS,IN}}) = 1$, set $\alpha =$'E'.
   (e) If $[\alpha =\text{'-'}] \wedge [\underline{(\ell_{\text{INP}} > \ell^*)} \vee (h \neq h^*)]$, output $\perp$.
   Else if $[\alpha =\text{'-'}] \wedge [\mathsf{SPS.Verify}(\text{VK}_{\text{SPS},F}, m, \sigma_{\text{SPS,IN}}) = 1]$, set $\alpha =$'F'.
   (f) If $\alpha =$'-', output $\perp$.
2. (a) Compute $r_{\text{SPS},A} = \mathcal{F}(K_{\text{SPS},A}, (h, \ell_{\text{INP}}, 0))$, $(\text{SK}_{\text{SPS},A}, \text{VK}_{\text{SPS},A}, \text{VK}_{\text{SPS-REJ},A}) = \mathsf{SPS.Setup}(1^\lambda; r_{\text{SPS},A})$.
   (b) Compute $r_{\text{SPS},B} = \mathcal{F}(K_{\text{SPS},B}, (h, \ell_{\text{INP}}, 0))$, $(\text{SK}_{\text{SPS},B}, \text{VK}_{\text{SPS},B}, \text{VK}_{\text{SPS-REJ},B}) = \mathsf{SPS.Setup}(1^\lambda; r_{\text{DSPS},B})$.
   (c) If $[(h, \ell_{\text{INP}}) = (h^*, \ell^*)] \wedge [\alpha =$'$F$'$]$, output $\sigma_{\text{SPS,OUT}} = \mathsf{SPS.Sign}(\text{SK}_{\text{SPS},B}, m)$.
   Else, output $\sigma_{\text{SPS,OUT}} = \mathsf{SPS.Sign}(\text{SK}_{\text{SPS},A}, m)$.

Fig. 14 $\mathsf{Change\text{-}SPS.Prog}^{(2)}$

$\mathsf{Hyb}_{0,\nu-1,3,\iota}$ ($\iota = 0, \ldots, \ell^* - 1$): In this hybrid experiment, to answer the ν^{th} signing key query of $\mathcal{A}$ corresponding to TM $M^{(\nu)} \in \mathbb{M}_\lambda$ with $M^{(\nu)}(x^*) = 0$, $\mathcal{B}$ proceeds as follows:

1. It first generates all the PPRF keys as well as the public parameters for the positional accumulator and the iterator as in $\mathsf{Hyb}_{0,\nu-1,3}$.
2. $\underline{\text{Next, it sets } m^{(\nu)}_{0,0} = (v^{(\nu)}_0, q^{(\nu)}_0, w^{(\nu)}_0, 0). \text{ For } j = 1, \ldots, \iota, \text{ it iteratively computes the following:}}$
   – $\underline{\text{AUX}^{(\nu)}_j = \mathsf{ACC.Prep\text{-}Write}(\text{PP}^{(\nu)}_{\text{ACC}}, \text{STORE}^{(\nu)}_{j-1}, j-1)}$

– $w_j^{(\nu)} = \mathsf{ACC.Update}(\mathrm{PP}_{\mathrm{ACC}}^{(\nu)}, w_{j-1}^{(\nu)}, x_{j-1}^*, j-1, \mathrm{AUX}_j^{(\nu)})$

– $\mathrm{STORE}_j^{(\nu)} = \mathsf{ACC.Write\text{-}Store}(\mathrm{PP}_{\mathrm{ACC}}^{(\nu)}, \mathrm{STORE}_{j-1}^{(\nu)}, j-1, x_{j-1}^*)$

– $v_j^{(\nu)} = \mathsf{ITR.Iterate}(\mathrm{PP}_{\mathrm{ITR}}^{(\nu)}, v_{j-1}^{(\nu)}, (q_0^{(\nu)}, w_{j-1}^{(\nu)}, 0))$

It sets $m_{\iota,0}^{(\nu)} = (v_\iota^{(\nu)}, q_0^{(\nu)}, w_\iota^{(\nu)}, 0)$.

3. It gives $\mathcal{A}$ the signing key

$$\mathrm{SK}_{\mathrm{ABS}}\{M^{(\nu)}\} = \left(\begin{array}{l} \mathrm{HK}, \mathrm{PP}_{\mathrm{ACC}}^{(\nu)}, w_0^{(\nu)}, \mathrm{STORE}_0^{(\nu)}, \mathrm{PP}_{\mathrm{ITR}}^{(\nu)}, v_0^{(\nu)}, \\ \mathcal{IO}(\mathsf{Init\text{-}SPS.Prog}[q_0^{(\nu)}, w_0^{(\nu)}, v_0^{(\nu)}, K_{\mathrm{SPS},E}^{(\nu)}]), \\ \mathcal{IO}(\mathsf{Accumulate.Prog}^{(3,\iota)}[n_{\text{SSB-BLK}} = 2^\lambda, \mathrm{HK}, \mathrm{PP}_{\mathrm{ACC}}^{(\nu)}, \mathrm{PP}_{\mathrm{ITR}}^{(\nu)}, K_{\mathrm{SPS},E}^{(\nu)}, \\ \quad K_{\mathrm{SPS},F}^{(\nu)}, m_{\iota,0}^{(\nu)}, h^*, \ell^*]), \\ \mathcal{IO}(\mathsf{Change\text{-}SPS.Prog}^{(3,\iota)}[K_{\mathrm{SPS},A}^{(\nu)}, K_{\mathrm{SPS},B}^{(\nu)}, K_{\mathrm{SPS},E}^{(\nu)}, K_{\mathrm{SPS},F}^{(\nu)}, h^*, \ell^*]), \\ \mathcal{IO}(\mathsf{Constrained\text{-}Key.Prog}_{\mathrm{ABS}}^{(1)}[M^{(\nu)}, T = 2^\lambda, t^{*(\nu)}, \mathrm{PP}_{\mathrm{ACC}}^{(\nu)}, \mathrm{PP}_{\mathrm{ITR}}^{(\nu)}, K, \\ \quad K_1^{(\nu)}, \ldots, K_\lambda^{(\nu)}, K_{\mathrm{SPS},A}^{(\nu)}, K_{\mathrm{SPS},B}^{(\nu)}, h^*, \ell^*]) \end{array} \right),$$

where the programs $\mathsf{Accumulate.Prog}^{(3,\iota)}$ and $\mathsf{Change\text{-}SPS.Prog}^{(3,\iota)}$ are alterations of the programs $\mathsf{Accumulate.Prog}^{(2)}$ and $\mathsf{Change\text{-}SPS.Prog}^{(2)}$ (Figs. **13** and **14**) and are described in Figs. **15** and **16** respectively.

The remaining part of the experiment is analogous to $\mathsf{Hyb}_{0,\nu-1,3}$.

$\mathsf{Hyb}_{0,\nu-1,3,\iota'}$ ($\iota = 0, \ldots, \ell^* - 1$): This experiment is identical to $\mathsf{Hyb}_{0,\nu-1,3}$ except

Constants: Maximum number of blocks for SSB hash $n_{\text{SSB-BLK}} = 2^\lambda$, SSB hash key HK, Public parameters for positional accumulator $\mathrm{PP}_{\mathrm{ACC}}$, Public parameters for iterator $\mathrm{PP}_{\mathrm{ITR}}$, PPRF keys $K_{\mathrm{SPS},E}, K_{\mathrm{SPS},F}$, Message $m_{\iota,0}$, SSB hash value of challenge input h^*, Length of challenge input ℓ^*

Inputs: Index i, Symbol $\mathrm{SYM}_{\mathrm{IN}}$, TM state ST, Accumulator value w_{IN}, Auxiliary value AUX, Iterator value v_{IN}, Signature $\sigma_{\mathrm{SPS,IN}}$, SSB hash value h, SSB opening value π_{SSB}

Output: (Accumulator value w_{OUT}, Iterator value v_{OUT}, Signature $\sigma_{\text{SPS-OUT}}$), or $\perp$

1. (a) Compute $r_{\mathrm{SPS},E} = \mathcal{F}(K_{\mathrm{SPS},E}, (h,i))$, $(\mathrm{SK}_{\mathrm{SPS},E}, \mathrm{VK}_{\mathrm{SPS},E}, \mathrm{VK}_{\text{SPS-REJ},E}) = \mathsf{SPS.Setup}(1^\lambda; r_{\mathrm{SPS},E})$.
   (b) Compute $r_{\mathrm{SPS},F} = \mathcal{F}(K_{\mathrm{SPS},F}, (h,i))$, $(\mathrm{SK}_{\mathrm{SPS},F}, \mathrm{VK}_{\mathrm{SPS},F}, \mathrm{VK}_{\text{SPS-REJ},F}) = \mathsf{SPS.Setup}(1^\lambda; r_{\mathrm{SPS},F})$.
   (c) Set $m_{\mathrm{IN}} = (v_{\mathrm{IN}}, \mathrm{ST}, w_{\mathrm{IN}}, 0)$ and $\alpha =$'-'.
   (d) If $\mathsf{SPS.Verify}(\mathrm{VK}_{\mathrm{SPS},E}, m_{\mathrm{IN}}, \sigma_{\mathrm{SPS,IN}}) = 1$, set $\alpha =$'E'.
   (e) If $[\alpha =$'-'$] \wedge [(i > \ell^*) \vee (0 \le i \le \iota) \vee (h \ne h^*)]$, output $\perp$.
   Else if $[\alpha =$'-'$] \wedge [\mathsf{SPS.Verify}(\mathrm{VK}_{\mathrm{SPS},F}, m_{\mathrm{IN}}, \sigma_{\mathrm{SPS,IN}}) = 1]$, set $\alpha =$'F'.
   (f) If $\alpha =$'-', output $\perp$.
2. If $\mathsf{SSB.Verify}(\mathrm{HK}, h, i, \mathrm{SYM}_{\mathrm{IN}}, \pi_{\mathrm{SSB}}) = 0$, output $\perp$.
3. (a) Compute $w_{\mathrm{OUT}} = \mathsf{ACC.Update}(\mathrm{PP}_{\mathrm{ACC}}, w_{\mathrm{IN}}, \mathrm{SYM}_{\mathrm{IN}}, i, \mathrm{AUX})$. If $w_{\mathrm{OUT}} = \perp$, output $\perp$.
   (b) Compute $v_{\mathrm{OUT}} = \mathsf{ITR.Iterate}(\mathrm{PP}_{\mathrm{ITR}}, v_{\mathrm{IN}}, (\mathrm{ST}, w_{\mathrm{IN}}, 0))$.
4. (a) Compute $r'_{\mathrm{SPS},E} = \mathcal{F}(K_{\mathrm{SPS},E}, (h,i+1))$, $(\mathrm{SK}'_{\mathrm{SPS},E}, \mathrm{VK}'_{\mathrm{SPS},E}, \mathrm{VK}'_{\text{SPS-REJ},E}) = \mathsf{SPS.Setup}(1^\lambda; r'_{\mathrm{SPS},E})$.
   (b) Compute $r'_{\mathrm{SPS},F} = \mathcal{F}(K_{\mathrm{SPS},F}, (h,i+1))$, $(\mathrm{SK}'_{\mathrm{SPS},F}, \mathrm{VK}'_{\mathrm{SPS},F}, \mathrm{VK}'_{\text{SPS-REJ},F}) = \mathsf{SPS.Setup}(1^\lambda; r'_{\mathrm{SPS},F})$.
   (c) Set $m_{\mathrm{OUT}} = (v_{\mathrm{OUT}}, \mathrm{ST}, w_{\mathrm{OUT}}, 0)$. If $[(h,i) = (h^*, \iota)] \wedge [m_{\mathrm{IN}} = m_{\iota,0}]$, compute $\sigma_{\mathrm{SPS,OUT}} = \mathsf{SPS.Sign}(\mathrm{SK}'_{\mathrm{SPS},E}, m_{\mathrm{OUT}})$.
   Else if $[(h,i) = (h^*, \iota)] \wedge [m_{\mathrm{IN}} \ne m_{\iota,0}]$, compute $\sigma_{\mathrm{SPS,OUT}} = \mathsf{SPS.Sign}(\mathrm{SK}'_{\mathrm{SPS},F}, m_{\mathrm{OUT}})$.
   Else if $i < \ell^*$, compute $\sigma_{\mathrm{SPS,OUT}} = \mathsf{SPS.Sign}(\mathrm{SK}'_{\mathrm{SPS},\alpha}, m_{\mathrm{OUT}})$.
   Else, compute $\sigma_{\mathrm{SPS,OUT}} = \mathsf{SPS.Sign}(\mathrm{SK}'_{\mathrm{SPS},E}, m_{\mathrm{OUT}})$.
5. Output $(w_{\mathrm{OUT}}, v_{\mathrm{OUT}}, \sigma_{\mathrm{SPS,OUT}})$.

Fig. 15 $\mathsf{Accumulate.Prog}^{(3,\iota)}$

that in response to the ν^{th} signing key query of $\mathcal{A}$ corresponding to TM $M^{(\nu)} \in \mathbb{M}_\lambda$ with $M^{(\nu)}(x^*) = 0$, $\mathcal{B}$ executes the following steps:

Constants: PPRF keys $K_{\text{SPS},A}$, $K_{\text{SPS},B}$, $K_{\text{SPS},E}$, $K_{\text{SPS},F}$, SSB hash value of challenge input h^*, Length of challenge input ℓ^*

Inputs: TM state ST, Accumulator value w, Iterator value v, SSB hash value h, Length ℓ_{INP}, Signature $\sigma_{\text{SPS,IN}}$

Output: Signature $\sigma_{\text{SPS,OUT}}$, or $\perp$

1. (a) Compute $r_{\text{SPS},E} = \mathcal{F}(K_{\text{SPS},E}, (h, \ell_{\text{INP}}))$, $(\text{SK}_{\text{SPS},E}, \text{VK}_{\text{SPS},E}, \text{VK}_{\text{SPS-REJ},E}) = \mathsf{SPS.Setup}(1^\lambda; r_{\text{SPS},E})$.
   (b) Compute $r_{\text{SPS},F} = \mathcal{F}(K_{\text{SPS},F}, (h, \ell_{\text{INP}}))$, $(\text{SK}_{\text{SPS},F}, \text{VK}_{\text{SPS},F}, \text{VK}_{\text{SPS-REJ},F}) = \mathsf{SPS.Setup}(1^\lambda; r_{\text{SPS},F})$.
   (c) Set $m = (v, \text{ST}, w, 0)$ and α='-'.
   (d) If $\mathsf{SPS.Verify}(\text{VK}_{\text{SPS},E}, m, \sigma_{\text{SPS,IN}}) = 1$, set α ='E'.
   (e) If $[\alpha$ ='-'$] \wedge [(\ell_{\text{INP}} > \ell^*) \vee \underline{(0 < \ell_{\text{INP}} \leq \iota)} \vee (h \neq h^*)]$, output $\perp$.
   Else if $[\alpha$ ='-'$] \wedge [\mathsf{SPS.Verify}(\text{VK}_{\text{SPS},F}, m, \sigma_{\text{SPS,IN}}) = 1]$, set α ='F'.
   (f) If α ='-', output $\perp$.
2. (a) Compute $r_{\text{SPS},A} = \mathcal{F}(K_{\text{SPS},A}, (h, \ell_{\text{INP}}, 0))$, $(\text{SK}_{\text{SPS},A}, \text{VK}_{\text{SPS},A}, \text{VK}_{\text{SPS-REJ},A}) = \mathsf{SPS.Setup}(1^\lambda; r_{\text{SPS},A})$.
   (b) Compute $r_{\text{SPS},B} = \mathcal{F}(K_{\text{SPS},B}, (h, \ell_{\text{INP}}, 0))$, $(\text{SK}_{\text{SPS},B}, \text{VK}_{\text{SPS},B}, \text{VK}_{\text{SPS-REJ},B}) = \mathsf{SPS.Setup}(1^\lambda; r_{\text{DSPS},B})$.
   (c) If $[(h, \ell_{\text{INP}}) = (h^*, \ell^*)] \wedge [\alpha$ ='F'$]$, output $\sigma_{\text{SPS,OUT}} = \mathsf{SPS.Sign}(\text{SK}_{\text{SPS},B}, m)$.
   Else, output $\sigma_{\text{SPS,OUT}} = \mathsf{SPS.Sign}(\text{SK}_{\text{SPS},A}, m)$.

Fig. 16 $\mathsf{Change\text{-}SPS.Prog}^{(3,\iota)}$

1. It first generates all the PPRF keys as well as the public parameters for the positional accumulator and the iterator as in $\mathsf{Hyb}_{0,\nu-1,3}$.
2. Next, it sets $m_{0,0}^{(\nu)} = (v_0^{(\nu)}, q_0^{(\nu)}, w_0^{(\nu)}, 0)$. For $\underline{j = 1, \ldots, \iota+1}$, it iteratively computes the following:
   - $\text{AUX}_j^{(\nu)} = \mathsf{ACC.Prep\text{-}Write}(\text{PP}_{\text{ACC}}^{(\nu)}, \text{STORE}_{j-1}^{(\nu)}, j-1)$
   - $w_j^{(\nu)} = \mathsf{ACC.Update}(\text{PP}_{\text{ACC}}^{(\nu)}, w_{j-1}^{(\nu)}, x_{j-1}^*, j-1, \text{AUX}_j^{(\nu)})$
   - $\text{STORE}_j^{(\nu)} = \mathsf{ACC.Write\text{-}Store}(\text{PP}_{\text{ACC}}^{(\nu)}, \text{STORE}_{j-1}^{(\nu)}, j-1, x_{j-1}^*)$
   - $v_j^{(\nu)} = \mathsf{ITR.Iterate}(\text{PP}_{\text{ITR}}^{(\nu)}, v_{j-1}^{(\nu)}, (q_0^{(\nu)}, w_{j-1}^{(\nu)}, 0))$

   It sets $\underline{m_{\iota+1,0}^{(\nu)} = (v_{\iota+1}^{(\nu)}, q_0^{(\nu)}, w_{\iota+1}^{(\nu)}, 0)}$.
3. It gives $\mathcal{A}$ the signing key

$$\text{SK}_{\text{ABS}}\{M^{(\nu)}\} = \left(\begin{array}{l} \text{HK}, \text{PP}_{\text{ACC}}^{(\nu)}, w_0^{(\nu)}, \text{STORE}_0^{(\nu)}, \text{PP}_{\text{ITR}}^{(\nu)}, v_0^{(\nu)}, \\ \mathcal{IO}(\mathsf{Init\text{-}SPS.Prog}[q_0^{(\nu)}, w_0^{(\nu)}, v_0^{(\nu)}, K_{\text{SPS},E}^{(\nu)}]), \\ \mathcal{IO}(\mathsf{Accumulate.Prog}^{(3,\iota')}[n_{\text{SSB-BLK}} = 2^\lambda, \text{HK}, \text{PP}_{\text{ACC}}^{(\nu)}, \text{PP}_{\text{ITR}}^{(\nu)}, K_{\text{SPS},E}^{(\nu)}, \\ \quad \underline{K_{\text{SPS},F}^{(\nu)}, m_{\iota+1,0}^{(\nu)}, h^*, \ell^*}]), \\ \mathcal{IO}(\mathsf{Change\text{-}SPS.Prog}^{(3,\iota)}[K_{\text{SPS},A}^{(\nu)}, K_{\text{SPS},B}^{(\nu)}, K_{\text{SPS},E}^{(\nu)}, K_{\text{SPS},F}^{(\nu)}, h^*, \ell^*]), \\ \mathcal{IO}(\mathsf{Constrained\text{-}Key.Prog}_{\text{ABS}}^{(1)}[M^{(\nu)}, T = 2^\lambda, t^{*(\nu)}, \text{PP}_{\text{ACC}}^{(\nu)}, \text{PP}_{\text{ITR}}^{(\nu)}, K, \\ \quad K_1^{(\nu)}, \ldots, K_\lambda^{(\nu)}, K_{\text{SPS},A}^{(\nu)}, K_{\text{SPS},B}^{(\nu)}, h^*, \ell^*]) \end{array} \right),$$

   where the program $\mathsf{Accumulate.Prog}^{(3,\iota')}$ is an alteration of the program $\mathsf{Accumulate.Prog}^{(3,\iota)}$ (Fig. **15**) and is shown in Fig. **17**.

$\mathsf{Hyb}_{0,\nu-1,4}$: This experiment is identical to $\mathsf{Hyb}_{0,\nu-1,3,(\ell^*-1)'}$ with the exception that now in response to the ν^{th} signing key query of $\mathcal{A}$ corresponding to TM $M^{(\nu)} \in \mathbb{M}_\lambda$ with $M^{(\nu)}(x^*) = 0$, $\mathcal{B}$ $\underline{\text{does not generate the PPRF key } K_{\text{SPS},F}^{(\nu)}}$ and gives $\mathcal{A}$ the signing

Constants: Maximum number of blocks for SSB hash $n_{\text{SSB-BLK}} = 2^\lambda$, SSB hash key HK, Public parameters for positional accumulator PP_{ACC}, Public parameters for iterator PP_{ITR}, PPRF keys $K_{\text{SPS},E}$, $K_{\text{SPS},F}$, Message $m_{\iota+1,0}$, SSB hash value of challenge input h^*, Length of challenge input ℓ^*

Inputs: Index i, Symbol SYM_{IN}, TM state ST, Accumulator value w_{IN}, Auxiliary value AUX, Iterator value v_{IN}, Signature $\sigma_{\text{SPS,IN}}$, SSB hash value h, SSB opening value π_{SSB}

Output: (Accumulator value w_{OUT}, Iterator value v_{OUT}, Signature $\sigma_{\text{SPS-OUT}}$), or $\perp$

1. (a) Compute $r_{\text{SPS},E} = \mathcal{F}(K_{\text{SPS},E}, (h,i))$, $(\text{SK}_{\text{SPS},E}, \text{VK}_{\text{SPS},E}, \text{VK}_{\text{SPS-REJ},E}) = \mathsf{SPS.Setup}(1^\lambda; r_{\text{SPS},E})$.
   (b) Compute $r_{\text{SPS},F} = \mathcal{F}(K_{\text{SPS},F}, (h,i))$, $(\text{SK}_{\text{SPS},F}, \text{VK}_{\text{SPS},F}, \text{VK}_{\text{SPS-REJ},F}) = \mathsf{SPS.Setup}(1^\lambda; r_{\text{SPS},F})$.
   (c) Set $m_{\text{IN}} = (v_{\text{IN}}, \text{ST}, w_{\text{IN}}, 0)$ and $\alpha =$'-'.
   (d) If $\mathsf{SPS.Verify}(\text{VK}_{\text{SPS},E}, m_{\text{IN}}, \sigma_{\text{SPS,IN}}) = 1$, set $\alpha =$'E'.
   (e) If $[\alpha =\text{'-'}] \ \wedge \ [(i > \ell^*) \ \vee \ (0 \leq i \leq \iota) \ \vee \ (h \neq h^*)]$, output $\perp$.
   Else if $[\alpha =\text{'-'}] \ \wedge \ [\mathsf{SPS.Verify}(\text{VK}_{\text{SPS},F}, m_{\text{IN}}, \sigma_{\text{SPS,IN}}) = 1]$, set $\alpha =$'F'.
   (f) If $\alpha =$'-', output $\perp$.
2. If $\mathsf{SSB.Verify}(\text{HK}, h, i, \text{SYM}_{\text{IN}}, \pi_{\text{SSB}}) = 0$, output $\perp$.
3. (a) Compute $w_{\text{OUT}} = \mathsf{ACC.Update}(\text{PP}_{\text{ACC}}, w_{\text{IN}}, \text{SYM}_{\text{IN}}, i, \text{AUX})$. If $w_{\text{OUT}} = \perp$, output $\perp$.
   (b) Compute $v_{\text{OUT}} = \mathsf{ITR.Iterate}(\text{PP}_{\text{ITR}}, v_{\text{IN}}, (\text{ST}, w_{\text{IN}}, 0))$.
4. (a) Compute $r'_{\text{SPS},E} = \mathcal{F}(K_{\text{SPS},E}, (h,i+1))$, $(\text{SK}'_{\text{SPS},E}, \text{VK}'_{\text{SPS},E}, \text{VK}'_{\text{SPS-REJ},E}) = \mathsf{SPS.Setup}(1^\lambda; r'_{\text{SPS},E})$.
   (b) Compute $r'_{\text{SPS},F} = \mathcal{F}(K_{\text{SPS},F}, (h,i+1))$, $(\text{SK}'_{\text{SPS},F}, \text{VK}'_{\text{SPS},F}, \text{VK}'_{\text{SPS-REJ},F}) = \mathsf{SPS.Setup}(1^\lambda; r'_{\text{SPS},F})$.
   (c) Set $m_{\text{OUT}} = (v_{\text{OUT}}, \text{ST}, w_{\text{OUT}}, 0)$. If $[(h,i) = (h^*, \iota)] \ \wedge \ \underline{[m_{\text{OUT}} = m_{\iota+1,0}]}$, compute $\sigma_{\text{SPS,OUT}} = \mathsf{SPS.Sign}(\text{SK}'_{\text{SPS},E}, m_{\text{OUT}})$.
   Else if $[(h,i) = (h^*, \iota)] \ \wedge \ [m_{\text{OUT}} \neq m_{\iota+1,0}]$, compute $\sigma_{\text{SPS,OUT}} = \mathsf{SPS.Sign}(\text{SK}'_{\text{SPS},F}, m_{\text{OUT}})$.
   Else if $i < \ell^*$, compute $\sigma_{\text{SPS,OUT}} = \mathsf{SPS.Sign}(\text{SK}'_{\text{SPS},\alpha}, m_{\text{OUT}})$.
   Else, compute $\sigma_{\text{SPS,OUT}} = \mathsf{SPS.Sign}(\text{SK}'_{\text{SPS},E} m_{\text{OUT}})$.
5. Output $(w_{\text{OUT}}, v_{\text{OUT}}, \sigma_{\text{SPS,OUT}})$.

Fig. 17 $\mathsf{Accumulate.Prog}^{(3,\iota')}$

key

$$\text{SK}_{\text{ABS}}\{M^{(\nu)}\} = \left(\begin{array}{l} \text{HK}, \text{PP}_{\text{ACC}}^{(\nu)}, w_0^{(\nu)}, \text{STORE}_0^{(\nu)}, \text{PP}_{\text{ITR}}^{(\nu)}, v_0^{(\nu)}, \\ \mathcal{IO}(\mathsf{Init\text{-}SPS.Prog}[q_0^{(\nu)}, w_0^{(\nu)}, v_0^{(\nu)}, K_{\text{SPS},E}^{(\nu)}]), \\ \mathcal{IO}(\mathsf{Accumulate.Prog}[n_{\text{SSB-BLK}} = 2^\lambda, \text{HK}, \text{PP}_{\text{ACC}}^{(\nu)}, \text{PP}_{\text{ITR}}^{(\nu)}, K_{\text{SPS},E}^{(\nu)}]), \\ \mathcal{IO}(\overline{\mathsf{Change\text{-}SPS.Prog}^{(4)}[K_{\text{SPS},A}^{(\nu)}, K_{\text{SPS},B}^{(\nu)}, K_{\text{SPS},E}^{(\nu)}, m_{\ell^*,0}^{(\nu)}, h^*, \ell^*]}), \\ \mathcal{IO}(\overline{\mathsf{Constrained\text{-}Key.Prog}_{\text{ABS}}^{(1)}[M^{(\nu)}, T = 2^\lambda, t^{*(\nu)}, \text{PP}_{\text{ACC}}^{(\nu)}, \text{PP}_{\text{ITR}}^{(\nu)}, K,} \\ \qquad K_1^{(\nu)}, \ldots, K_\lambda^{(\nu)}, K_{\text{SPS},A}^{(\nu)}, K_{\text{SPS},B}^{(\nu)}, h^*, \ell^*]) \end{array} \right),$$

where the program $\mathsf{Accumulate.Prog}$ is shown in Fig. **3** while the program $\mathsf{Change\text{-}SPS.Prog}^{(4)}$, which is a modification of the program $\mathsf{Change\text{-}SPS.Prog}^{(3,\iota)}$ (Fig. **16**), is depicted in Fig. **18**.

Constants: PPRF keys $K_{\text{SPS},A}, K_{\text{SPS},B}, K_{\text{SPS},E}$, Message $m_{\ell^*,0}$, SSB hash value of challenge input h^*, Length of challenge input ℓ^*

Inputs: TM state ST, Accumulator value w, Iterator value v, SSB hash value h, Length ℓ_{INP}, Signature $\sigma_{\text{SPS,IN}}$

Output: Signature $\sigma_{\text{SPS,OUT}}$, or $\perp$

1. (a) Compute $r_{\text{SPS},E} = \mathcal{F}(K_{\text{SPS},E}, (h, \ell_{\text{INP}}), (\text{SK}_{\text{SPS},E}, \text{VK}_{\text{SPS},E}, \text{VK}_{\text{SPS-REJ},E}) = \mathsf{SPS.Setup}(1^\lambda; r_{\text{SPS},E})$.
   (b) Set $m = (v, \text{ST}, w, 0)$.
   (c) If $\mathsf{SPS.Verify}(\text{VK}_{\text{SPS},E}, m, \sigma_{\text{SPS,IN}}) = 0$, output $\perp$.
2. (a) Compute $r_{\text{SPS},A} = \mathcal{F}(K_{\text{SPS},A}, (h, \ell_{\text{INP}}, 0)), (\text{SK}_{\text{SPS},A}, \text{VK}_{\text{SPS},A}, \text{VK}_{\text{SPS-REJ},A}) = \mathsf{SPS.Setup}(1^\lambda; r_{\text{SPS},A})$.
   (b) Compute $r_{\text{SPS},B} = \mathcal{F}(K_{\text{SPS},B}, (h, \ell_{\text{INP}}, 0)), (\text{SK}_{\text{SPS},B}, \text{VK}_{\text{SPS},B}, \text{VK}_{\text{SPS-REJ},B}) = \mathsf{SPS.Setup}(1^\lambda; r_{\text{SPS},B})$.
   (c) If $[(h, \ell_{\text{INP}}) = (h^*, \ell^*)] \wedge [m \neq m_{\ell^*,0}]$, output $\sigma_{\text{SPS,OUT}} = \mathsf{SPS.Sign}(\text{SK}_{\text{SPS},B}, m)$.
   Else, output $\sigma_{\text{SPS,OUT}} = \mathsf{SPS.Sign}(\text{SK}_{\text{SPS},A}, m)$.

Fig. 18 $\mathsf{Change\text{-}SPS.Prog}^{(4)}$

$\mathsf{Hyb}_{0,\nu-1,4,\gamma}$ ($\gamma = 1, \ldots, t^{*(\nu)} - 1$): This experiment is analogous to $\mathsf{Hyb}_{0,\nu-1,4}$ except that in response to the ν^{th} signing key query of $\mathcal{A}$ corresponding to TM $M^{(\nu)} \in \mathbb{M}_\lambda$ with $M^{(\nu)}(x^*) = 0$, $\mathcal{B}$ proceeds as follows:

1. It first generates all the PPRF keys as well as the public parameters for the positional accumulator and the iterator as in $\mathsf{Hyb}_{0,\nu-1,4}$.
2. Next, it sets $m_{0,0}^{(\nu)} = (v_0^{(\nu)}, q_0^{(\nu)}, w_0^{(\nu)}, 0)$. For $j = 1, \ldots, \ell^*$, it iteratively computes the following:
   - $\text{AUX}_j^{(\nu)} = \mathsf{ACC.Prep\text{-}Write}(\text{PP}_{\text{ACC}}^{(\nu)}, \text{STORE}_{j-1}^{(\nu)}, j-1)$
   - $w_j^{(\nu)} = \mathsf{ACC.Update}(\text{PP}_{\text{ACC}}^{(\nu)}, w_{j-1}^{(\nu)}, x_{j-1}^*, j-1, \text{AUX}_j^{(\nu)})$
   - $\text{STORE}_j^{(\nu)} = \mathsf{ACC.Write\text{-}Store}(\text{PP}_{\text{ACC}}^{(\nu)}, \text{STORE}_{j-1}^{(\nu)}, j-1, x_{j-1}^*)$
   - $v_j^{(\nu)} = \mathsf{ITR.Iterate}(\text{PP}_{\text{ITR}}^{(\nu)}, v_{j-1}^{(\nu)}, (q_0^{(\nu)}, w_{j-1}^{(\nu)}, 0))$
3. Then, $\mathcal{B}$ sets $\text{ST}_{M^{(\nu)},0} = q_0^{(\nu)}$, $\text{POS}_{M^{(\nu)},0} = 0$, and for $t = 1, \ldots, \gamma$, computes the following:
   - $(\text{SYM}_{M^{(\nu)},t-1}, \pi_{\text{ACC},t-1}^{(\nu)}) = \mathsf{ACC.Prep\text{-}Read}(\text{PP}_{\text{ACC}}^{(\nu)}, \text{STORE}_{\ell^*+t-1}^{(\nu)}, \text{POS}_{M^{(\nu)},t-1})$
   - $\text{AUX}_{\ell^*+t}^{(\nu)} = \mathsf{ACC.Prep\text{-}Write}(\text{PP}_{\text{ACC}}^{(\nu)}, \text{STORE}_{\ell^*+t-1}^{(\nu)}, \text{POS}_{M^{(\nu)},t-1})$
   - $(\text{ST}_{M^{(\nu)},t}, \text{SYM}_{M^{(\nu)},t}^{(\text{WRITE})}, \beta) = \delta^{(\nu)}(\text{ST}_{M^{(\nu)},t-1}, \text{SYM}_{M^{(\nu)},t-1})$
   - $w_{\ell^*+t}^{(\nu)} = \mathsf{ACC.Update}(\text{PP}_{\text{ACC}}^{(\nu)}, w_{\ell^*+t-1}^{(\nu)}, \text{SYM}_{M^{(\nu)},t}^{(\text{WRITE})}, \text{POS}_{M^{(\nu)},t-1}, \text{AUX}_{\ell^*+t}^{(\nu)})$
   - $v_{\ell^*+t}^{(\nu)} = \mathsf{ITR.Iterate}(\text{PP}_{\text{ITR}}^{(\nu)}, v_{\ell^*+t-1}^{(\nu)}, (\text{ST}_{M^{(\nu)},t-1}, w_{\ell^*+t-1}^{(\nu)}, \text{POS}_{M^{(\nu)},t-1}))$
   - $\text{STORE}_{\ell^*+t}^{(\nu)} = \mathsf{ACC.Write\text{-}Store}(\text{PP}_{\text{ACC}}^{(\nu)}, \text{STORE}_{\ell^*+t-1}^{(\nu)}, \text{POS}_{M^{(\nu)},t-1}, \text{SYM}_{M^{(\nu)},t}^{(\text{WRITE})})$
   - $\text{POS}_{M^{(\nu)},t} = \text{POS}_{M^{(\nu)},t-1} + \beta$

   $\mathcal{B}$ sets $m_{\ell^*,\gamma}^{(\nu)} = (v_{\ell^*+\gamma}^{(\nu)}, \text{ST}_{M^{(\nu)},\gamma}, w_{\ell^*+\gamma}^{(\nu)}, \text{POS}_{M^{(\nu)},\gamma})$.

4. $\mathcal{B}$ provides $\mathcal{A}$ with the signing key
$\mathrm{SK}_{\mathrm{ABS}}\{M^{(\nu)}\} =$
$$\left(\begin{array}{l}\mathrm{HK}, \mathrm{PP}_{\mathrm{ACC}}^{(\nu)}, w_0^{(\nu)}, \mathrm{STORE}_0^{(\nu)}, \mathrm{PP}_{\mathrm{ITR}}^{(\nu)}, v_0^{(\nu)}, \\ \mathcal{IO}(\mathsf{Init\text{-}SPS.Prog}[q_0^{(\nu)}, w_0^{(\nu)}, v_0^{(\nu)}, K_{\mathrm{SPS},E}^{(\nu)}]), \\ \mathcal{IO}(\mathsf{Accumulate.Prog}[n_{\text{SSB-BLK}} = 2^\lambda, \mathrm{HK}, \mathrm{PP}_{\mathrm{ACC}}^{(\nu)}, \mathrm{PP}_{\mathrm{ITR}}^{(\nu)}, K_{\mathrm{SPS},E}^{(\nu)}]), \\ \mathcal{IO}(\overline{\mathsf{Change\text{-}SPS.Prog}[K_{\mathrm{SPS},A}^{(\nu)}, K_{\mathrm{SPS},E}^{(\nu)}]}), \\ \mathcal{IO}(\overline{\mathsf{Constrained\text{-}Key.Prog}_{\mathrm{ABS}}^{(2,\gamma)}[M^{(\nu)}, T = 2^\lambda, t^{*(\nu)}, \mathrm{PP}_{\mathrm{ACC}}^{(\nu)}, \mathrm{PP}_{\mathrm{ITR}}^{(\nu)}, K, \\ \quad K_1^{(\nu)}, \ldots, K_\lambda^{(\nu)}, K_{\mathrm{SPS},A}^{(\nu)}, K_{\mathrm{SPS},B}^{(\nu)}, m_{\ell^*,\gamma}^{(\nu)}, h^*, \ell^*]})\end{array}\right),$$
where the program $\mathsf{Change\text{-}SPS.Prog}$ is described in Fig. **4** and the program $\mathsf{Constrained\text{-}Key.Prog}_{\mathrm{ABS}}^{(2,\gamma)}$, a modification of program $\mathsf{Constrained\text{-}Key.Prog}_{\mathrm{ABS}}^{(1)}$ (Fig. **10**), is described in Fig. **19**.

$\mathsf{Hyb}_{0,\nu-1,4,\gamma'}$ ($\gamma = 0, \ldots, t^{*(\nu)} - 1$): This experiment is identical to $\mathsf{Hyb}_{0,\nu-1,4}$ except that in response to the ν^{th} signing key query of $\mathcal{A}$ corresponding to TM $M^{(\nu)} \in \mathbb{M}_\lambda$ with $M^{(\nu)}(x^*) = 0$, $\mathcal{B}$ proceeds as follows:

1. It first generates all the PPRF keys as well as the public parameters for the positional accumulator and the iterator as in $\mathsf{Hyb}_{0,\nu-1,4}$.
2. Next, it sets $m_{0,0}^{(\nu)} = (v_0^{(\nu)}, q_0^{(\nu)}, w_0^{(\nu)}, 0)$. For $j = 1, \ldots, \ell^*$, it iteratively computes the following:
   - $\mathrm{AUX}_j^{(\nu)} = \mathsf{ACC.Prep\text{-}Write}(\mathrm{PP}_{\mathrm{ACC}}^{(\nu)}, \mathrm{STORE}_{j-1}^{(\nu)}, j-1)$
   - $w_j^{(\nu)} = \mathsf{ACC.Update}(\mathrm{PP}_{\mathrm{ACC}}^{(\nu)}, w_{j-1}^{(\nu)}, x_{j-1}^*, j-1, \mathrm{AUX}_j^{(\nu)})$
   - $\mathrm{STORE}_j^{(\nu)} = \mathsf{ACC.Write\text{-}Store}(\mathrm{PP}_{\mathrm{ACC}}^{(\nu)}, \mathrm{STORE}_{j-1}^{(\nu)}, j-1, x_{j-1}^*)$
   - $v_j^{(\nu)} = \mathsf{ITR.Iterate}(\mathrm{PP}_{\mathrm{ITR}}^{(\nu)}, v_{j-1}^{(\nu)}, (q_0^{(\nu)}, w_{j-1}^{(\nu)}, 0))$
3. Then, $\mathcal{B}$ sets $\mathrm{ST}_{M^{(\nu)},0} = q_0^{(\nu)}, \mathrm{POS}_{M^{(\nu)},0} = 0$, and for $t = 1, \ldots, \gamma$, computes the following:
   - $(\mathrm{SYM}_{M^{(\nu)},t-1}, \pi_{\mathrm{ACC},t-1}^{(\nu)}) = \mathsf{ACC.Prep\text{-}Read}(\mathrm{PP}_{\mathrm{ACC}}^{(\nu)}, \mathrm{STORE}_{\ell^*+t-1}^{(\nu)}, \mathrm{POS}_{M^{(\nu)},t-1})$
   - $\mathrm{AUX}_{\ell^*+t}^{(\nu)} = \mathsf{ACC.Prep\text{-}Write}(\mathrm{PP}_{\mathrm{ACC}}^{(\nu)}, \mathrm{STORE}_{\ell^*+t-1}^{(\nu)}, \mathrm{POS}_{M^{(\nu)},t-1})$
   - $(\mathrm{ST}_{M^{(\nu)},t}, \mathrm{SYM}_{M^{(\nu)},t}^{(\mathrm{WRITE})}, \beta) = \delta^{(\nu)}(\mathrm{ST}_{M^{(\nu)},t-1}, \mathrm{SYM}_{M^{(\nu)},t-1})$
   - $w_{\ell^*+t}^{(\nu)} = \mathsf{ACC.Update}(\mathrm{PP}_{\mathrm{ACC}}^{(\nu)}, w_{\ell^*+t-1}^{(\nu)}, \mathrm{SYM}_{M^{(\nu)},t}^{(\mathrm{WRITE})}, \mathrm{POS}_{M^{(\nu)},t-1}, \mathrm{AUX}_{\ell^*+t}^{(\nu)})$
   - $v_{\ell^*+t}^{(\nu)} = \mathsf{ITR.Iterate}(\mathrm{PP}_{\mathrm{ITR}}^{(\nu)}, v_{\ell^*+t-1}^{(\nu)}, (\mathrm{ST}_{M^{(\nu)},t-1}, w_{\ell^*+t-1}^{(\nu)}, \mathrm{POS}_{M^{(\nu)},t-1}))$
   - $\mathrm{STORE}_{\ell^*+t}^{(\nu)} = \mathsf{ACC.Write\text{-}Store}(\mathrm{PP}_{\mathrm{ACC}}^{(\nu)}, \mathrm{STORE}_{\ell^*+t-1}^{(\nu)}, \mathrm{POS}_{M^{(\nu)},t-1}, \mathrm{SYM}_{M^{(\nu)},t}^{(\mathrm{WRITE})})$
   - $\mathrm{POS}_{M^{(\nu)},t} = \mathrm{POS}_{M^{(\nu)},t-1} + \beta$

   $\mathcal{B}$ sets $m_{\ell^*,\gamma}^{(\nu)} = (v_{\ell^*+\gamma}^{(\nu)}, \mathrm{ST}_{M^{(\nu)},\gamma}, w_{\ell^*+\gamma}^{(\nu)}, \mathrm{POS}_{M^{(\nu)},\gamma})$.

Constants: TM $M = \langle Q, \Sigma_{\text{INP}}, \Sigma_{\text{TAPE}}, \delta, q_0, q_{\text{AC}}, q_{\text{REJ}} \rangle$, Time bound $T = 2^\lambda$, Running time on challenge input t^*, Public parameters for positional accumulator PP_{ACC}, Public parameters for iterator PP_{ITR}, PPRF keys $K, K_1, \ldots, K_\lambda, K_{\text{SPS},A}, K_{\text{SPS},B}$, Message $m_{\ell^*,\gamma}$, SSB hash value of challenge input h^*, Length of challenge input ℓ^*

Inputs: Time t, String SEED_{IN}, Header position POS_{IN}, Symbol SYM_{IN}, TM state ST_{IN}, Accumulator value w_{IN}, Accumulator proof π_{ACC}, Auxiliary value AUX, Iterator value v_{IN}, SSB hash value h, Length ℓ_{INP}, Signature $\sigma_{\text{SPS,IN}}$

Output: CPRF evaluation $\mathcal{F}(K, (h, \ell_{\text{INP}}))$, or (Header Position POS_{OUT}, Symbol SYM_{OUT}, TM state ST_{OUT}, Accumulator value w_{OUT}, Iterator value v_{OUT}, Signature $\sigma_{\text{SPS,OUT}}$, String SEED_{OUT}), or $\perp$

1. Identify an integer τ such that $2^\tau \leq t < 2^{\tau+1}$. If $[\mathsf{PRG}(\text{SEED}_{\text{IN}}) \neq \mathsf{PRG}(\mathcal{F}(K_\tau, (h, \ell_{\text{INP}})))] \wedge [t > 1]$, output $\perp$.
2. If $\mathsf{ACC.Verify\text{-}Read}(\text{PP}_{\text{ACC}}, w_{\text{IN}}, \text{SYM}_{\text{IN}}, \text{POS}_{\text{IN}}, \pi_{\text{ACC}}) = 0$, output $\perp$.
3. (a) Compute $r_{\text{SPS},A} = \mathcal{F}(K_{\text{SPS},A}, (h, \ell_{\text{INP}}, t-1)), (\text{SK}_{\text{SPS},A}, \text{VK}_{\text{SPS},A}, \text{VK}_{\text{SPS-REJ},A}) = \mathsf{SPS.Setup}(1^\lambda; r_{\text{SPS},A})$.
   (b) Compute $r_{\text{SPS},B} = \mathcal{F}(K_{\text{SPS},B}, (h, \ell_{\text{INP}}, t-1)), (\text{SK}_{\text{SPS},B}, \text{VK}_{\text{SPS},B}, \text{VK}_{\text{SPS-REJ},B}) = \mathsf{SPS.Setup}(1^\lambda; r_{\text{SPS},B})$.
   (c) Set $m_{\text{IN}} = (v_{\text{IN}}, \text{ST}_{\text{IN}}, w_{\text{IN}}, \text{POS}_{\text{IN}})$ and $\alpha =$'-'.
   (d) If $\mathsf{SPS.Verify}(\text{VK}_{\text{SPS},A}, m_{\text{IN}}, \sigma_{\text{SPS,IN}}) = 1$, set $\alpha =$'A'.
   (e) If $[\alpha =$'-'$] \wedge [(t > t^*) \vee (t \leq \gamma) \vee (h \neq h^*) \vee (\ell_{\text{INP}} \neq \ell^*)]$, output $\perp$.
   Else if $[\alpha =$'-'$] \wedge [\mathsf{SPS.Verify}(\text{VK}_{\text{SPS},B}, m_{\text{IN}}, \sigma_{\text{SPS,IN}}) = 1]$, set $\alpha =$'B'.
   (f) If $\alpha =$'-', output $\perp$.
4. (a) Compute $(\text{ST}_{\text{OUT}}, \text{SYM}_{\text{OUT}}, \beta) = \delta(\text{ST}_{\text{IN}}, \text{SYM}_{\text{IN}})$ and $\text{POS}_{\text{OUT}} = \text{POS}_{\text{IN}} + \beta$.
   (b) If $\text{ST}_{\text{OUT}} = q_{\text{REJ}}$, output $\perp$.
   Else if $[\text{ST}_{\text{OUT}} = q_{\text{AC}}] \wedge [\alpha =$'$B$'$]$, output $\perp$.
   Else if $[\text{ST}_{\text{OUT}} = q_{\text{AC}}] \wedge [\alpha =$'$A$'$] \wedge [(h, \ell_{\text{INP}}) = (h^*, \ell^*)] \wedge [t \leq \gamma]$, output $\perp$.
   Else if $\text{ST}_{\text{OUT}} = q_{\text{AC}}$, perform the following:
   (I) Compute $r_{\text{SIG}} = \mathcal{F}(K, (h, \ell_{\text{INP}})), (\text{SK}_{\text{SIG}}, \text{VK}_{\text{SIG}}) = \mathsf{SIG.Setup}(1^\lambda; r_{\text{SIG}})$.
   (II) Output $(\text{SK}_{\text{SIG}}, \text{VK}_{\text{SIG}})$.
5. (a) Compute $w_{\text{OUT}} = \mathsf{ACC.Update}(\text{PP}_{\text{ACC}}, w_{\text{IN}}, \text{SYM}_{\text{OUT}}, \text{POS}_{\text{IN}}, \text{AUX})$. If $w_{\text{OUT}} = \perp$, output $\perp$.
   (b) Compute $v_{\text{OUT}} = \mathsf{ITR.Iterate}(\text{PP}_{\text{ITR}}, v_{\text{IN}}, (\text{ST}_{\text{IN}}, w_{\text{IN}}, \text{POS}_{\text{IN}}))$.
6. (a) Compute $r'_{\text{SPS},A} = \mathcal{F}(K_{\text{SPS},A}, (h, \ell_{\text{INP}}, t)), (\text{SK}'_{\text{SPS},A}, \text{VK}'_{\text{SPS},A}, \text{VK}'_{\text{SPS-REJ},A}) = \mathsf{SPS.Setup}(1^\lambda; r'_{\text{SPS},A})$.
   (b) Compute $r'_{\text{SPS},B} = \mathcal{F}(K_{\text{SPS},B}, (h, \ell_{\text{INP}}, t)), (\text{SK}'_{\text{SPS},B}, \text{VK}'_{\text{SPS},B}, \text{VK}'_{\text{SPS-REJ},B}) = \mathsf{SPS.Setup}(1^\lambda; r'_{\text{SPS},B})$.
   (c) Set $m_{\text{OUT}} = (v_{\text{OUT}}, \text{ST}_{\text{OUT}}, w_{\text{OUT}}, \text{POS}_{\text{OUT}})$.
   If $[(h, \ell_{\text{INP}}, t) = (h^*, \ell^*, \gamma)] \wedge [m_{\text{OUT}} = m_{\ell^*,\gamma}]$, compute $\sigma_{\text{SPS,OUT}} = \mathsf{SPS.Sign}(\text{SK}'_{\text{SPS},A}, m_{\text{OUT}})$.
   Else if $[(h, \ell_{\text{INP}}, t) = (h^*, \ell^*, \gamma)] \wedge [m_{\text{OUT}} \neq m_{\ell^*,\gamma}]$, compute $\sigma_{\text{SPS,OUT}} = \mathsf{SPS.Sign}(\text{SK}'_{\text{SPS},B}, m_{\text{OUT}})$.
   Else, compute $\sigma_{\text{SPS,OUT}} = \mathsf{SPS.Sign}(\text{SK}'_{\text{SPS},\alpha}, m_{\text{OUT}})$.
7. If $t + 1 = 2^{\tau'}$, for some integer τ', set $\text{SEED}_{\text{OUT}} = \mathcal{F}(K_{\tau'}, (h, \ell_{\text{INP}}))$.
   Else, set $\text{SEED}_{\text{OUT}} = \epsilon$.
8. Output $(\text{POS}_{\text{OUT}}, \text{SYM}_{\text{OUT}}, \text{ST}_{\text{OUT}}, w_{\text{OUT}}, v_{\text{OUT}}, \sigma_{\text{SPS,OUT}}, \text{SEED}_{\text{OUT}})$.

Fig. 19 $\mathsf{Constrained\text{-}Key.Prog}^{(2,\gamma)}_{\text{ABS}}$

4. $\mathcal{B}$ provides $\mathcal{A}$ with the signing key

$$\text{SK}_{\text{ABS}}\{M^{(\nu)}\} = \begin{pmatrix} \text{HK}, \text{PP}^{(\nu)}_{\text{ACC}}, w^{(\nu)}_0, \text{STORE}^{(\nu)}_0, \text{PP}^{(\nu)}_{\text{ITR}}, v^{(\nu)}_0, \\ \mathcal{IO}(\mathsf{Init\text{-}SPS.Prog}[q^{(\nu)}_0, w^{(\nu)}_0, v^{(\nu)}_0, K^{(\nu)}_{\text{SPS},E}]), \\ \mathcal{IO}(\mathsf{Accumulate.Prog}[n_{\text{SSB-BLK}} = 2^\lambda, \text{HK}, \text{PP}^{(\nu)}_{\text{ACC}}, \text{PP}^{(\nu)}_{\text{ITR}}, K^{(\nu)}_{\text{SPS},E}]), \\ \mathcal{IO}(\mathsf{Change\text{-}SPS.Prog}[K^{(\nu)}_{\text{SPS},A}, K^{(\nu)}_{\text{SPS},E}]), \\ \mathcal{IO}(\underline{\mathsf{Constrained\text{-}Key.Prog}^{(2,\gamma')}_{\text{ABS}}[M^{(\nu)}, T = 2^\lambda, t^{*(\nu)}, \text{PP}^{(\nu)}_{\text{ACC}}, \text{PP}^{(\nu)}_{\text{ITR}}, K,} \\ \underline{K^{(\nu)}_1, \ldots, K^{(\nu)}_\lambda, K^{(\nu)}_{\text{SPS},A}, K^{(\nu)}_{\text{SPS},B}, m^{(\nu)}_{\ell^*,\gamma}, h^*, \ell^*]}) \end{pmatrix},$$

where the program $\mathsf{Constrained\text{-}Key.Prog}^{(2,\gamma')}_{\text{ABS}}$ is an alteration of the program $\mathsf{Constrained\text{-}Key.Prog}^{(2,\gamma)}_{\text{ABS}}$ (Fig. **19**) and is described in Fig. **20**.

Constants: TM $M = \langle Q, \Sigma_{\text{INP}}, \Sigma_{\text{TAPE}}, \delta, q_0, q_{\text{AC}}, q_{\text{REJ}} \rangle$, Time bound $T = 2^\lambda$, Running time on challenge input t^*, Public parameters for positional accumulator PP_{ACC}, Public parameters for iterator PP_{ITR}, PPRF keys $K, K_1, \ldots, K_\lambda, K_{\text{SPS},A}, K_{\text{SPS},B}$, Message $m_{\ell^*,\gamma}$, SSB hash value of challenge input h^*, Length of challenge input ℓ^*

Inputs: Time t, String SEED_{IN}, Header position POS_{IN}, Symbol SYM_{IN}, TM state ST_{IN}, Accumulator value w_{IN}, Accumulator proof π_{ACC}, Auxiliary value AUX, Iterator value v_{IN}, SSB hash value h, Length ℓ_{INP}, Signature $\sigma_{\text{SPS,IN}}$

Output: CPRF evaluation $\mathcal{F}(K, (h, \ell_{\text{INP}}))$, or (Header Position POS_{OUT}, Symbol SYM_{OUT}, TM state ST_{OUT}, Accumulator value w_{OUT}, Iterator value v_{OUT}, Signature $\sigma_{\text{SPS,OUT}}$, String SEED_{OUT}), or $\perp$

1. Identify an integer τ such that $2^\tau \leq t < 2^{\tau+1}$. If $[\mathsf{PRG}(\text{SEED}_{\text{IN}}) \neq \mathsf{PRG}(\mathcal{F}(K_\tau, (h, \ell_{\text{INP}})))] \wedge [t > 1]$, output $\perp$.
2. If $\mathsf{ACC.Verify\text{-}Read}(\text{PP}_{\text{ACC}}, w_{\text{IN}}, \text{SYM}_{\text{IN}}, \text{POS}_{\text{IN}}, \pi_{\text{ACC}}) = 0$, output $\perp$.
3. (a) Compute $r_{\text{SPS},A} = \mathcal{F}(K_{\text{SPS},A}, (h, \ell_{\text{INP}}, t-1)), (\text{SK}_{\text{SPS},A}, \text{VK}_{\text{SPS},A}, \text{VK}_{\text{SPS-REJ},A}) = \mathsf{SPS.Setup}(1^\lambda; r_{\text{SPS},A})$.
   (b) Compute $r_{\text{SPS},B} = \mathcal{F}(K_{\text{SPS},B}, (h, \ell_{\text{INP}}, t-1)), (\text{SK}_{\text{SPS},B}, \text{VK}_{\text{SPS},B}, \text{VK}_{\text{SPS-REJ},B}) = \mathsf{SPS.Setup}(1^\lambda; r_{\text{SPS},B})$.
   (c) Set $m_{\text{IN}} = (v_{\text{IN}}, \text{ST}_{\text{IN}}, w_{\text{IN}}, \text{POS}_{\text{IN}})$ and $\alpha =$'-'.
   (d) If $\mathsf{SPS.Verify}(\text{VK}_{\text{SPS},A}, m_{\text{IN}}, \sigma_{\text{SPS,IN}}) = 1$, set $\alpha =$'A'.
   (e) If $[\alpha =$'-'$] \wedge [(t > t^*) \vee \underline{(t \leq \gamma + 1)} \vee (h \neq h^*) \vee (\ell_{\text{INP}} \neq \ell^*)]$, output $\perp$.
   Else if $[\alpha =$'-'$] \wedge [\mathsf{SPS.Verify}(\text{VK}_{\text{SPS},B}, m_{\text{IN}}, \sigma_{\text{SPS,IN}}) = 1]$, set $\alpha =$'B'.
   (f) If $\alpha =$'-', output $\perp$.
4. (a) Compute $(\text{ST}_{\text{OUT}}, \text{SYM}_{\text{OUT}}, \beta) = \delta(\text{ST}_{\text{IN}}, \text{SYM}_{\text{IN}})$ and $\text{POS}_{\text{OUT}} = \text{POS}_{\text{IN}} + \beta$.
   (b) If $\text{ST}_{\text{OUT}} = q_{\text{REJ}}$, output $\perp$.
   Else if $[\text{ST}_{\text{OUT}} = q_{\text{AC}}] \wedge [\alpha =$'$B$'$]$, output $\perp$.
   Else if $[\text{ST}_{\text{OUT}} = q_{\text{AC}}] \wedge [\alpha =$'$A$'$] \wedge [(h, \ell_{\text{INP}}) = (h^*, \ell^*)] \wedge \underline{[t \leq \gamma + 1]}$, output $\perp$.
   Else if $\text{ST}_{\text{OUT}} = q_{\text{AC}}$, perform the following:
   (I) Compute $r_{\text{SIG}} = \mathcal{F}(K, (h, \ell_{\text{INP}}))$, $(\text{SK}_{\text{SIG}}, \text{VK}_{\text{SIG}}) = \mathsf{SIG.Setup}(1^\lambda; r_{\text{SIG}})$.
   (II) Output $(\text{SK}_{\text{SIG}}, \text{VK}_{\text{SIG}})$.
5. (a) Compute $w_{\text{OUT}} = \mathsf{ACC.Update}(\text{PP}_{\text{ACC}}, w_{\text{IN}}, \text{SYM}_{\text{OUT}}, \text{POS}_{\text{IN}}, \text{AUX})$. If $w_{\text{OUT}} = \perp$, output $\perp$.
   (b) Compute $v_{\text{OUT}} = \mathsf{ITR.Iterate}(\text{PP}_{\text{ITR}}, v_{\text{IN}}, (\text{ST}_{\text{IN}}, w_{\text{IN}}, \text{POS}_{\text{IN}}))$.
6. (a) Compute $r'_{\text{SPS},A} = \mathcal{F}(K_{\text{SPS},A}, (h, \ell_{\text{INP}}, t)), (\text{SK}'_{\text{SPS},A}, \text{VK}'_{\text{SPS},A}, \text{VK}'_{\text{SPS-REJ},A}) = \mathsf{SPS.Setup}(1^\lambda; r'_{\text{SPS},A})$.
   (b) Compute $r'_{\text{SPS},B} = \mathcal{F}(K_{\text{SPS},B}, (h, \ell_{\text{INP}}, t)), (\text{SK}'_{\text{SPS},B}, \text{VK}'_{\text{SPS},B}, \text{VK}'_{\text{SPS-REJ},B}) = \mathsf{SPS.Setup}(1^\lambda; r'_{\text{SPS},B})$.
   (c) Set $m_{\text{OUT}} = (v_{\text{OUT}}, \text{ST}_{\text{OUT}}, w_{\text{OUT}}, \text{POS}_{\text{OUT}})$.
   If $\underline{[(h, \ell_{\text{INP}}, t) = (h^*, \ell^*, \gamma + 1)] \wedge [m_{\text{IN}} = m_{\ell^*,\gamma}]}$, compute $\underline{\sigma_{\text{SPS,OUT}} = \mathsf{SPS.Sign}(\text{SK}'_{\text{SPS},A}, m_{\text{OUT}})}$.
   $\underline{\text{Else if } [(h, \ell_{\text{INP}}, t) = (h^*, \ell^*, \gamma + 1)] \wedge [m_{\text{IN}} \neq m_{\ell^*,\gamma}]}$, compute $\underline{\sigma_{\text{SPS,OUT}} = \mathsf{SPS.Sign}(\text{SK}'_{\text{SPS},B}, m_{\text{OUT}})}$.
   $\underline{\text{Else, compute } \sigma_{\text{SPS,OUT}} = \mathsf{SPS.Sign}(\text{SK}'_{\text{SPS},\alpha}, m_{\text{OUT}})}$.
7. If $t + 1 = 2^{\tau'}$, for some integer τ', set $\text{SEED}_{\text{OUT}} = \mathcal{F}(K_{\tau'}, (h, \ell_{\text{INP}}))$.
   Else, set $\text{SEED}_{\text{OUT}} = \epsilon$.
8. Output $(\text{POS}_{\text{OUT}}, \text{SYM}_{\text{OUT}}, \text{ST}_{\text{OUT}}, w_{\text{OUT}}, v_{\text{OUT}}, \sigma_{\text{SPS,OUT}}, \text{SEED}_{\text{OUT}})$.

Fig. 20 $\mathsf{Constrained\text{-}Key.Prog}^{(2,\gamma')}_{\text{ABS}}$

$\mathsf{Hyb}_{0,\nu-1,5}$: This experiment is similar to $\mathsf{Hyb}_{0,\nu-1,4,(t^{*(\nu)}-1)'}$ with the exception that in responding to the ν^{th} signing key query of $\mathcal{A}$ corresponding to TM $M^{(\nu)} \in \mathbb{M}_\lambda$ with $M^{(\nu)}(x^*) = 0$, $\mathcal{B}$ gives $\mathcal{A}$ the signing key

$$\mathrm{SK}_{\mathrm{ABS}}\{M^{(\nu)}\} = \left(\begin{array}{l} \mathrm{HK}, \mathrm{PP}_{\mathrm{ACC}}^{(\nu)}, w_0^{(\nu)}, \mathrm{STORE}_0^{(\nu)}, \mathrm{PP}_{\mathrm{ITR}}^{(\nu)}, v_0^{(\nu)}, \\ \mathcal{IO}(\mathsf{Init\text{-}SPS.Prog}[q_0^{(\nu)}, w_0^{(\nu)}, v_0^{(\nu)}, K_{\mathrm{SPS},E}^{(\nu)}]), \\ \mathcal{IO}(\mathsf{Accumulate.Prog}[n_{\text{SSB-BLK}} = 2^\lambda, \mathrm{HK}, \mathrm{PP}_{\mathrm{ACC}}^{(\nu)}, \mathrm{PP}_{\mathrm{ITR}}^{(\nu)}, K_{\mathrm{SPS},E}^{(\nu)}]), \\ \mathcal{IO}(\mathsf{Change\text{-}SPS.Prog}[K_{\mathrm{SPS},A}^{(\nu)}, K_{\mathrm{SPS},E}^{(\nu)}]), \\ \mathcal{IO}(\underline{\mathsf{Constrained\text{-}Key.Prog}_{\mathrm{ABS}}^{(3)}[M^{(\nu)}, T = 2^\lambda, t^{*(\nu)}, \mathrm{PP}_{\mathrm{ACC}}^{(\nu)}, \mathrm{PP}_{\mathrm{ITR}}^{(\nu)}, K,} \\ \quad \underline{K_1^{(\nu)}, \ldots, K_\lambda^{(\nu)}, K_{\mathrm{SPS},A}^{(\nu)}, K_{\mathrm{SPS},B}^{(\nu)}, h^*, \ell^*])} \end{array}\right),$$

where the program $\mathsf{Constrained\text{-}Key.Prog}_{\mathrm{ABS}}^{(3)}$ is a modification of the program $\mathsf{Constrained\text{-}Key.Prog}_{\mathrm{ABS}}^{(2,\gamma')}$ (Fig. **20**) and is described in Fig. **21**.

Constants: TM $M = \langle Q, \Sigma_{\mathrm{INP}}, \Sigma_{\mathrm{TAPE}}, \delta, q_0, q_{\mathrm{AC}}, q_{\mathrm{REJ}} \rangle$, Time bound $T = 2^\lambda$, Running time on challenge input t^*, Public parameters for positional accumulator $\mathrm{PP}_{\mathrm{ACC}}$, Public parameters for iterator $\mathrm{PP}_{\mathrm{ITR}}$, PPRF keys $K, K_1, \ldots, K_\lambda, K_{\mathrm{SPS},A}, K_{\mathrm{SPS},B}$, SSB hash value of challenge input h^*, Length of challenge input ℓ^*

Inputs: Time t, String $\mathrm{SEED}_{\mathrm{IN}}$, Header position $\mathrm{POS}_{\mathrm{IN}}$, Symbol $\mathrm{SYM}_{\mathrm{IN}}$, TM state $\mathrm{ST}_{\mathrm{IN}}$, Accumulator value w_{IN}, Accumulator proof π_{ACC}, Auxiliary value AUX, Iterator value v_{IN}, SSB hash value h, Length ℓ_{INP}, Signature $\sigma_{\mathrm{SPS,IN}}$

Output: CPRF evaluation $\mathcal{F}(K, (h, \ell_{\mathrm{INP}}))$, or (Header Position ($\mathrm{POS}_{\mathrm{OUT}}$, Symbol $\mathrm{SYM}_{\mathrm{OUT}}$, TM state $\mathrm{ST}_{\mathrm{OUT}}$, Accumulator value w_{OUT}, Iterator value v_{OUT}, Signature $\sigma_{\mathrm{SPS,OUT}}$, String $\mathrm{SEED}_{\mathrm{OUT}}$), or $\perp$

1. Identify an integer τ such that $2^\tau \leq t < 2^{\tau+1}$. If $[\mathsf{PRG}(\mathrm{SEED}_{\mathrm{IN}}) \neq \mathsf{PRG}(\mathcal{F}(K_\tau, (h, \ell_{\mathrm{INP}})))] \wedge [t > 1]$, output $\perp$.
2. If $\mathsf{ACC.Verify\text{-}Read}(\mathrm{PP}_{\mathrm{ACC}}, w_{\mathrm{IN}}, \mathrm{SYM}_{\mathrm{IN}}, \mathrm{POS}_{\mathrm{IN}}, \pi_{\mathrm{ACC}}) = 0$, output $\perp$.
3. (a) Compute $r_{\mathrm{SPS},A} = \mathcal{F}(K_{\mathrm{SPS},A}, (h, \ell_{\mathrm{INP}}, t-1)), (\mathrm{SK}_{\mathrm{SPS},A}, \mathrm{VK}_{\mathrm{SPS},A}, \mathrm{VK}_{\text{SPS-REJ},A}) = \mathsf{SPS.Setup}(1^\lambda; r_{\mathrm{SPS},A})$.
   (b) Set $m_{\mathrm{IN}} = (v_{\mathrm{IN}}, \mathrm{ST}_{\mathrm{IN}}, w_{\mathrm{IN}}, \mathrm{POS}_{\mathrm{IN}})$. <u>If $\mathsf{SPS.Verify}(\mathrm{VK}_{\mathrm{SPS},A}, m_{\mathrm{IN}}, \sigma_{\mathrm{SPS,IN}}) = 0$, output $\perp$.</u>
4. (a) Compute $(\mathrm{ST}_{\mathrm{OUT}}, \mathrm{SYM}_{\mathrm{OUT}}, \beta) = \delta(\mathrm{ST}_{\mathrm{IN}}, \mathrm{SYM}_{\mathrm{IN}})$ and $\mathrm{POS}_{\mathrm{OUT}} = \mathrm{POS}_{\mathrm{IN}} + \beta$.
   (b) If $\mathrm{ST}_{\mathrm{OUT}} = q_{\mathrm{REJ}}$, output $\perp$.
   Else if $[\mathrm{ST}_{\mathrm{OUT}} = q_{\mathrm{AC}}] \wedge [(h, \ell_{\mathrm{INP}}) = (h^*, \ell^*)] \wedge \underline{[t \leq t^*]}$, output $\perp$.
   Else if $\mathrm{ST}_{\mathrm{OUT}} = q_{\mathrm{AC}}$, perform the following:
   (I) Compute $r_{\mathrm{SIG}} = \mathcal{F}(K, (h, \ell_{\mathrm{INP}}))$, $(\mathrm{SK}_{\mathrm{SIG}}, \mathrm{VK}_{\mathrm{SIG}}) = \mathsf{SIG.Setup}(1^\lambda; r_{\mathrm{SIG}})$.
   (II) Output $(\mathrm{SK}_{\mathrm{SIG}}, \mathrm{VK}_{\mathrm{SIG}})$.
5. (a) Compute $w_{\mathrm{OUT}} = \mathsf{ACC.Update}(\mathrm{PP}_{\mathrm{ACC}}, w_{\mathrm{IN}}, \mathrm{SYM}_{\mathrm{OUT}}, \mathrm{POS}_{\mathrm{IN}}, \mathrm{AUX})$. If $w_{\mathrm{OUT}} = \perp$, output $\perp$.
   (b) Compute $v_{\mathrm{OUT}} = \mathsf{ITR.Iterate}(\mathrm{PP}_{\mathrm{ITR}}, v_{\mathrm{IN}}, (\mathrm{ST}_{\mathrm{IN}}, w_{\mathrm{IN}}, \mathrm{POS}_{\mathrm{IN}}))$.
6. (a) Compute $r'_{\mathrm{SPS},A} = \mathcal{F}(K_{\mathrm{SPS},A}, (h, \ell_{\mathrm{INP}}, t)), (\mathrm{SK}'_{\mathrm{SPS},A}, \mathrm{VK}'_{\mathrm{SPS},A}, \mathrm{VK}'_{\text{SPS-REJ},A}) = \mathsf{SPS.Setup}(1^\lambda; r'_{\mathrm{SPS},A})$.
   (b) Compute $r'_{\mathrm{SPS},B} = \mathcal{F}(K_{\mathrm{SPS},B}, (h, \ell_{\mathrm{INP}}, t)), (\mathrm{SK}'_{\mathrm{SPS},B}, \mathrm{VK}'_{\mathrm{SPS},B}, \mathrm{VK}'_{\text{SPS-REJ},B}) = \mathsf{SPS.Setup}(1^\lambda; r'_{\mathrm{SPS},B})$.
   (c) Set $m_{\mathrm{OUT}} = (v_{\mathrm{OUT}}, \mathrm{ST}_{\mathrm{OUT}}, w_{\mathrm{OUT}}, \mathrm{POS}_{\mathrm{OUT}})$.
   <u>If $(h, \ell_{\mathrm{INP}}, t) = (h^*, \ell^*, t^*)$, compute $\sigma_{\mathrm{SPS,OUT}} = \mathsf{SPS.Sign}(\mathrm{SK}'_{\mathrm{SPS},B}, m_{\mathrm{OUT}})$.</u>
   <u>Else, compute $\sigma_{\mathrm{SPS,OUT}} = \mathsf{SPS.Sign}(\mathrm{SK}'_{\mathrm{SPS},A}, m_{\mathrm{OUT}})$.</u>
7. If $t + 1 = 2^{\tau'}$, for some integer τ', set $\mathrm{SEED}_{\mathrm{OUT}} = \mathcal{F}(K_{\tau'}, (h, \ell_{\mathrm{INP}}))$.
   Else, set $\mathrm{SEED}_{\mathrm{OUT}} = \epsilon$.
8. Output $(\mathrm{POS}_{\mathrm{OUT}}, \mathrm{SYM}_{\mathrm{OUT}}, \mathrm{ST}_{\mathrm{OUT}}, w_{\mathrm{OUT}}, v_{\mathrm{OUT}}, \sigma_{\mathrm{SPS,OUT}}, \mathrm{SEED}_{\mathrm{OUT}})$.

Fig. 21 $\mathsf{Constrained\text{-}Key.Prog}_{\mathrm{ABS}}^{(3)}$

$\mathsf{Hyb}_{0,\nu-1,6}$: This experiment corresponds to $\mathsf{Hyb}_{0,\nu}$.

Analysis

Let $\mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,0)}(\lambda), \mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,1)}(\lambda), \mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,2)}(\lambda), \mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,3)}(\lambda), \mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,3,\iota)}(\lambda)$ $(\iota = 0, \ldots, \ell^* - 1), \mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,3,\iota')}(\lambda) (\iota = 0, \ldots, \ell^* - 1), \mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,4)}(\lambda), \mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,4,\gamma)}(\lambda)$ $(\gamma = 1, \ldots, t^{*(\nu)} - 1), \mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,4,\gamma')}(\lambda)$ $(\gamma = 0, \ldots, t^{*(\nu)} - 1)$, $\mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,5)}(\lambda)$, and $\mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,6)}(\lambda)$ represent respectively the advantage of the adversary $\mathcal{A}$, i.e., the absolute difference between $1/2$ and $\mathcal{A}$'s probability of correctly guessing the random bit selected by the challenger $\mathcal{B}$, in the hybrid experiment Hyb_{Υ} with Υ as indicated in the superscript of the advantage notation. By the description of the hybrid experiments it follows that $\mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1)}(\lambda) \equiv \mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,0)}(\lambda)$ and $\mathsf{Adv}_{\mathcal{A}}^{(0,\nu)}(\lambda) \equiv \mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,6)}(\lambda)$. Thus we have,

$$
\begin{aligned}
&|\mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1)}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{(0,\nu)}(\lambda)| \leq \\
&|\mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,0)}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,1)}(\lambda)| + |\mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,1)}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,2)}(\lambda)| + \\
&|\mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,2)}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,3)}(\lambda)| + |\mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,3)}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,3,0)}(\lambda)| + \\
&\sum_{\iota=0}^{\ell^*-1} |\mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,3,\iota)}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,3,\iota')}(\lambda)| + \\
&\sum_{\iota=0}^{\ell^*-2} |\mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,3,\iota')}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,3,\iota+1)}(\lambda)| + \\
&|\mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,3,(\ell^*-1)')}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,4)}(\lambda)| + |\mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,4)}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,4,0')}(\lambda)| + \\
&\sum_{\gamma=1}^{t^{*(\nu)}-1} |\mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,4,(\gamma-1)')}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,4,\gamma)}(\lambda)| + \\
&\sum_{\gamma=1}^{t^{*(\nu)}-1} |\mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,4,\gamma)}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,4,\gamma')}(\lambda)| + \\
&|\mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,4,(t^{*(\nu)}-1)')}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,5)}(\lambda)| + |\mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,5)}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,6)}(\lambda)|.
\end{aligned}
\tag{B.1}
$$

Lemmas C.1–C.12 provided in Appendix C will prove that the RHS of Eq. (B.1) is negligible and hence Lemma B.1 follows. □

Lemma B.2 *Assuming $\mathcal{IO}$ is a secure indistinguishability obfuscator for* P/poly *and $\mathcal{F}$ satisfies the correctness under puncturing property, for any* PPT *adversary $\mathcal{A}$, for any security parameter λ, $|\mathsf{Adv}_{\mathcal{A}}^{(1)}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{(2)}(\lambda)| \leq \mathsf{negl}(\lambda)$ for some negligible function* negl.

Proof The two differences between Hyb_1 and Hyb_2 are the following:

(I) In Hyb_1, $\mathcal{B}$ includes $\mathcal{IO}(V_0)$ within the public parameters $\mathrm{PP}_{\mathrm{ABS}}$ provided to $\mathcal{A}$, whereas, in Hyb_2, $\mathcal{B}$ includes the program $\mathcal{IO}(V_1)$ within $\mathrm{PP}_{\mathrm{ABS}}$, where
  - $(V_0) = \mathsf{Verify.Prog}_{\mathrm{ABS}}[K]$ (Fig. **1**),
  - $(V_1) = \mathsf{Verify.Prog}'_{\mathrm{ABS}}[K\{(h^*, \ell^*)\}, \widehat{\mathrm{VK}}^*_{\mathrm{ABS}}, h^*, \ell^*]$ (Fig. **8**).

(II) For $\eta = 1, \ldots, \hat{q}_{\mathrm{KEY}}$, the signing key $\mathrm{SK}_{\mathrm{ABS}}(M^{(\eta)})$ returned by $\mathcal{B}$ to $\mathcal{A}$ corresponding to signing policy TM $M^{(\eta)} \in \mathbb{M}_\lambda$ with $M^{(\eta)}(x^*) = 0$, includes the program $\mathcal{IO}(P_0^{(\eta)})$ in the experiment Hyb_1, while $\mathrm{SK}_{\mathrm{ABS}}(M^{(\eta)})$ includes the program $\mathcal{IO}(P_1^{(\eta)})$ in Hyb_2, where

– $P_0^{(\eta)} = \mathsf{Constrained\text{-}Key.Prog}'_{\mathrm{ABS}}[M^{(\eta)}, T = 2^\lambda, \mathrm{PP}_{\mathrm{ACC}}^{(\eta)}, \mathrm{PP}_{\mathrm{ITR}}^{(\eta)}, K, K_1^{(\eta)}, \ldots, K_\lambda^{(\eta)}, K_{\mathrm{SPS},A}^{(\eta)}, h^*, \ell^*]$,
– $P_1^{(\eta)} = \mathsf{Constrained\text{-}Key.Prog}'_{\mathrm{ABS}}[M^{(\eta)}, T = 2^\lambda, \mathrm{PP}_{\mathrm{ACC}}^{(\eta)}, \mathrm{PP}_{\mathrm{ITR}}^{(\eta)}, K\{(h^*, \ell^*)\}, K_1^{(\eta)}, \ldots, K_\lambda^{(\eta)}, K_{\mathrm{SPS},A}^{(\eta)}, h^*, \ell^*]$,

the program $\mathsf{Constrained\text{-}Key.Prog}'_{\mathrm{ABS}}$ being described in Fig. **7**.

Now, observe that on input $(h, \ell_{\mathrm{INP}}) \neq (h^*, \ell^*)$, both the programs V_0 and V_1 operates in the same manner only that the latter one uses the punctured PPRF key $K\{(h^*, \ell^*)\}$ for computing the string $\hat{r}_{\mathrm{SIG}}$ instead of the full PPRF key K used by the former program. Therefore, by the correctness under puncturing property of PPRF $\mathcal{F}$, it follows that for all inputs $(h, \ell_{\mathrm{INP}}) \neq (h^*, \ell^*)$, both the programs have identical output. Moreover, on input (h^*, ℓ^*), V_1 outputs the hardwired SIG verification key $\widehat{\mathrm{VK}}_{\mathrm{SIG}}^*$ which is computed as $(\widehat{\mathrm{SK}}_{\mathrm{SIG}}^*, \widehat{\mathrm{VK}}_{\mathrm{SIG}}^*) = \mathsf{SIG.Setup}(1^\lambda;\ \hat{r}_{\mathrm{SIG}}^*)$, where $\hat{r}_{\mathrm{SIG}}^* = \mathcal{F}(K, (h^*, \ell^*))$. Notice that these values are exactly the same as those outputted V_0 on input (h^*, ℓ^*). Thus, the two programs are functionally equivalent.

Further, note that the program $\mathsf{Constrained\text{-}Key.Prog}'_{\mathrm{ABS}}$ computes $\mathcal{F}(K, (h, \ell_{\mathrm{INP}}))$ if and only if $(h, \ell_{\mathrm{INP}}) \neq (h^*, \ell^*)$. Thus, again by the correctness under puncturing property of PPRF $\mathcal{F}$, the programs $P_0^{(\eta)}$ and $P_1^{(\eta)}$ are functionally equivalent as well for all $\eta \in [\hat{q}_{\mathrm{KEY}}]$.

Thus the security of $\mathcal{IO}$, Lemma B.2 follows. Observe that to prove this lemma we would actually have to proceed through a sequence of intermediate hybrid experiments where in each hybrid experiment we switch the programs one at a time. □

Lemma B.3 *Assuming $\mathcal{F}$ is a secure puncturable pseudorandom function, for any* PPT *adversary $\mathcal{A}$, for any security parameter λ, $|\mathsf{Adv}_{\mathcal{A}}^{(2)}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{(3)}(\lambda)| \leq \mathsf{negl}(\lambda)$ for some negligible function* negl.

Proof Suppose there exists a PPT adversary $\mathcal{A}$ for which $|\mathsf{Adv}_{\mathcal{A}}^{(2)}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{(3)}(\lambda)|$ is non-negligible. Below we construct a PPT adversary $\mathcal{B}$ that breaks the selective pseudorandomness of the PPRF $\mathcal{F}$ using $\mathcal{A}$ as a sub-routine.

- $\mathcal{B}$ initializes $\mathcal{A}$ on input 1^λ and receives a challenge attribute string $x^* = x_0^* \ldots x_{\ell^*-1}^* \in \mathcal{U}_{\mathrm{ABS}}$ with $|x^*| = \ell^*$ from $\mathcal{A}$.
- After receiving x^*, $\mathcal{B}$ proceeds as follows:
  1. $\mathcal{B}$ first generates $\mathrm{HK} \xleftarrow{\$} \mathsf{SSB.Gen}(1^\lambda, n_{\text{SSB-BLK}} = 2^\lambda, i^* = 0)$ and computes $h^* = \mathcal{H}_{\mathrm{HK}}(x^*)$.
  2. $\mathcal{B}$ sends (h^*, ℓ^*) as the challenge input to its PPRF selective pseudorandomness challenger $\mathcal{C}$ and receives back a punctured PPRF key $K^*\{(h^*, \ell^*)\}$ along with a challenge value $r^* \in \mathcal{Y}_{\mathrm{PPRF}}$, where either $r^* = \mathcal{F}(K^*, (h^*, \ell^*)\}$ or $r^* \xleftarrow{\$} \mathcal{Y}_{\mathrm{PPRF}}$. $\mathcal{B}$ *implicitly* views the key K^* as the key K.
  3. Then $\mathcal{B}$ creates $(\widehat{\mathrm{SK}}_{\mathrm{SIG}}^*, \widehat{\mathrm{VK}}_{\mathrm{SIG}}^*) = \mathsf{SIG.Setup}(1^\lambda;\ r^*)$.
  4. Next, $\mathcal{B}$ sets the public parameters $\mathrm{PP}_{\mathrm{ABS}} = (\mathrm{HK}, \mathcal{IO}(\mathsf{Verify.Prog}'_{\mathrm{ABS}}[K^*\{(h^*, \ell^*)\}, \widehat{\mathrm{VK}}_{\mathrm{SIG}}^*, h^*, \ell^*]))$ and gives it to $\mathcal{A}$.
- For $\eta = 1, \ldots, \hat{q}_{\mathrm{KEY}}$, to answer the η^{th} signing key query of $\mathcal{A}$ corresponding to signing policy TM $M^{(\eta)} \in \mathbb{M}_\lambda$ with $M^{(\eta)}(x^*) = 0$, $\mathcal{B}$ executes the following steps:
  1. At first, $\mathcal{B}$ chooses PPRF keys $K_1^{(\eta)}, \ldots, K_\lambda^{(\eta)}, K_{\mathrm{SPS},A}^{(\eta)}, K_{\mathrm{SPS},E}^{(\eta)} \xleftarrow{\$} \mathcal{F}.\mathsf{Setup}(1^\lambda)$.
  2. Next, $\mathcal{B}$ generates $(\mathrm{PP}_{\mathrm{ACC}}^{(\eta)}, w_0^{(\eta)}, \mathrm{STORE}_0^{(\eta)}) \xleftarrow{\$} \mathsf{ACC.Setup}(1^\lambda, n_{\text{ACC-BLK}} = 2^\lambda)$ and $(\mathrm{PP}_{\mathrm{ITR}}^{(\eta)}, v_0^{(\eta)}) \xleftarrow{\$} \mathsf{ITR.Setup}(1^\lambda, n_{\mathrm{ITR}} = 2^\lambda)$.

3. $\mathcal{B}$ returns $\mathcal{A}$ the signing key

$$\mathrm{SK}_{\mathrm{ABS}}(M^{(\eta)}) = \left(\begin{array}{l} \mathrm{HK}, \mathrm{PP}_{\mathrm{ACC}}^{(\eta)}, w_0^{(\eta)}, \mathrm{STORE}_0^{(\eta)}, \mathrm{PP}_{\mathrm{ITR}}^{(\eta)}, v_0^{(\eta)}, \\ \mathcal{IO}(\mathsf{Init\text{-}SPS.Prog}[q_0^{(\eta)}, w_0^{(\eta)}, v_0^{(\eta)}, K_{\mathrm{SPS},E}^{(\eta)}]) \\ \mathcal{IO}(\mathsf{Accumulate.Prog}[n_{\text{SSB-BLK}} = 2^\lambda, \mathrm{HK}, \mathrm{PP}_{\mathrm{ACC}}^{(\eta)}, \mathrm{PP}_{\mathrm{ITR}}^{(\eta)}, K_{\mathrm{SPS},E}^{(\eta)}]) \\ \mathcal{IO}(\mathsf{Change\text{-}SPS.Prog}[K_{\mathrm{SPS},A}^{(\eta)}, K_{\mathrm{SPS},E}^{(\eta)}]) \\ \mathcal{IO}(\mathsf{Constrained\text{-}Key.Prog}'_{\mathrm{ABS}}[M^{(\eta)}, T = 2^\lambda, \mathrm{PP}_{\mathrm{ACC}}^{(\eta)}, \mathrm{PP}_{\mathrm{ITR}}^{(\eta)}, \\ \qquad K^*\{(h^*, \ell^*)\}, K_1^{(\eta)}, \ldots, K_\lambda^{(\eta)}, K_{\mathrm{SPS},A}^{(\eta)}, h^*, \ell^*]) \end{array} \right).$$

- For $\theta = 1, \ldots, \hat{q}_{\mathrm{SIGN}}$, in response to the θ^{th} signature query of $\mathcal{A}$ on message $\mathsf{msg}^{(\theta)} \in \mathcal{M}_{\mathrm{ABS}}$ under attribute string x^*, $\mathcal{B}$ computes $\sigma_{\mathrm{SIG}}^{(\theta)} \xleftarrow{\$} \mathsf{SIG.Sign}(\widehat{\mathrm{SK}}_{\mathrm{SIG}}^*, \mathsf{msg}^{(\theta)})$ and provides $\mathcal{A}$ with $\sigma_{\mathrm{ABS}}^{(\theta)} = (\widehat{\mathrm{VK}}_{\mathrm{SIG}}^*, \sigma_{\mathrm{SIG}}^{(\theta)})$.
- Finally, $\mathcal{A}$ output a signature σ_{ABS}^* on some message msg^* under attribute string x^*. $\mathcal{B}$ outputs $\hat{b}' = 1$ as its guess bit in its PPRF selective pseudorandomness experiment if $\mathcal{A}$ wins, i.e., if $\mathsf{ABS.Verify}(\mathrm{PP}_{\mathrm{ABS}}, x^*, \mathsf{msg}^*, \sigma_{\mathrm{ABS}}^*) = 1$ and $\mathsf{msg}^* \neq \mathsf{msg}^{(\theta)}$ for any $\theta \in [\hat{q}_{\mathrm{SIGN}}]$. Otherwise, $\mathcal{B}$ outputs $\hat{b}' = 0$ in its PPRF selective pseudorandomness experiment.

Notice that if $r^* = \mathcal{F}(K^*, (h^*, \ell^*))$, then $\mathcal{B}$ perfectly simulates Hyb_2. On the other hand, if $r^* \xleftarrow{\$} \mathcal{Y}_{\mathrm{PPRF}}$, then $\mathcal{B}$ perfectly simulates Hyb_3. This completes the proof of Lemma B.3. □

Lemma B.4 *Assuming* SIG *is existentially unforgeable against* CMA*, for any* PPT *adversary* $\mathcal{A}$*, for any security parameter* λ*,* $\mathsf{Adv}_{\mathcal{A}}^{(3)}(\lambda) \leq \mathsf{negl}(\lambda)$ *for some negligible function* negl*.*

Proof Suppose that there exists a PPT adversary $\mathcal{A}$ for which $\mathsf{Adv}_{\mathcal{A}}^{(3)}(\lambda)$ is non-negligible. We construct a PPT adversary $\mathcal{B}$ that breaks the existential unforgeability of SIG using $\mathcal{A}$ as a sub-routine. The description $\mathcal{B}$ is as follows:

- $\mathcal{B}$ receives a SIG verification key $\mathrm{VK}_{\mathrm{SIG}}^*$ from its SIG existential unforgeability challenger $\mathcal{C}$. Then, $\mathcal{B}$ runs $\mathcal{A}$ on input 1^λ and receives a challenge attribute string $x^* = x_0^* \ldots x_{\ell^*-1}^* \in \mathcal{U}_{\mathrm{ABS}}$ with $|x^*| = \ell^*$ from $\mathcal{A}$.
- After receiving x^*, $\mathcal{B}$ proceeds as follows:
  1. $\mathcal{B}$ first generates $\mathrm{HK} \xleftarrow{\$} \mathsf{SSB.Gen}(1^\lambda, n_{\text{SSB-BLK}} = 2^\lambda, i^* = 0)$ and computes $h^* = \mathcal{H}_{\mathrm{HK}}(x^*)$.
  2. Next, it selects a PPRF key $K \xleftarrow{\$} \mathcal{F}.\mathsf{Setup}(1^\lambda)$ and creates the punctured PPRF key $K\{(h^*, \ell^*)\} \xleftarrow{\$} \mathcal{F}.\mathsf{Puncture}(K, (h^*, \ell^*))$.
  3. Next, $\mathcal{B}$ sets the public parameters $\mathrm{PP}_{\mathrm{ABS}} = (\mathrm{HK}, \mathcal{IO}(\mathsf{Verify.Prog}'_{\mathrm{ABS}}[K\{(h^*, \ell^*)\}, \mathrm{VK}_{\mathrm{SIG}}^*, h^*, \ell^*]))$ and gives it to $\mathcal{A}$.
- For $\eta = 1, \ldots, \hat{q}_{\mathrm{KEY}}$, to answer the η^{th} signing key query of $\mathcal{A}$ corresponding to signing policy TM $M^{(\eta)} \in \mathbb{M}_\lambda$ with $M^{(\eta)}(x^*) = 0$, $\mathcal{B}$ executes the following steps:
  1. At first, $\mathcal{B}$ chooses PPRF keys $K_1^{(\eta)}, \ldots, K_\lambda^{(\eta)}, K_{\mathrm{SPS},A}^{(\eta)}, K_{\mathrm{SPS},E}^{(\eta)} \xleftarrow{\$} \mathcal{F}.\mathsf{Setup}(1^\lambda)$.
  2. Next, $\mathcal{B}$ generates $(\mathrm{PP}_{\mathrm{ACC}}^{(\eta)}, w_0^{(\eta)}, \mathrm{STORE}_0^{(\eta)}) \xleftarrow{\$} \mathsf{ACC.Setup}(1^\lambda, n_{\text{ACC-BLK}} = 2^\lambda)$ and $(\mathrm{PP}_{\mathrm{ITR}}^{(\eta)}, v_0^{(\eta)}) \xleftarrow{\$} \mathsf{ITR.Setup}(1^\lambda, n_{\mathrm{ITR}} = 2^\lambda)$.

3. $\mathcal{B}$ returns $\mathcal{A}$ the signing key

$$\mathrm{SK}_{\mathrm{ABS}}(M^{(\eta)}) = \left(\begin{array}{l} \mathrm{HK}, \mathrm{PP}_{\mathrm{ACC}}^{(\eta)}, w_0^{(\eta)}, \mathrm{STORE}_0^{(\eta)}, \mathrm{PP}_{\mathrm{ITR}}^{(\eta)}, v_0^{(\eta)}, \\ \mathcal{IO}(\mathsf{Init\text{-}SPS.Prog}[q_0^{(\eta)}, w_0^{(\eta)}, v_0^{(\eta)}, K_{\mathrm{SPS},E}^{(\eta)}]) \\ \mathcal{IO}(\mathsf{Accumulate.Prog}[n_{\text{SSB-BLK}} = 2^\lambda, \mathrm{HK}, \mathrm{PP}_{\mathrm{ACC}}^{(\eta)}, \mathrm{PP}_{\mathrm{ITR}}^{(\eta)}, K_{\mathrm{SPS},E}^{(\eta)}]) \\ \mathcal{IO}(\mathsf{Change\text{-}SPS.Prog}[K_{\mathrm{SPS},A}^{(\eta)}, K_{\mathrm{SPS},E}^{(\eta)}]) \\ \mathcal{IO}(\mathsf{Constrained\text{-}Key.Prog}'_{\mathrm{ABS}}[M^{(\eta)}, T = 2^\lambda, \mathrm{PP}_{\mathrm{ACC}}^{(\eta)}, \mathrm{PP}_{\mathrm{ITR}}^{(\eta)}, \\ \quad K\{(h^*, \ell^*)\}, K_1^{(\eta)}, \ldots, K_\lambda^{(\eta)}, K_{\mathrm{SPS},A}^{(\eta)}, h^*, \ell^*]) \end{array} \right).$$

- For $\theta = 1, \ldots, \hat{q}_{\mathrm{SIGN}}$, in response to the θ^{th} signature query of $\mathcal{A}$ on message $\mathsf{msg}^{(\theta)} \in \mathcal{M}_{\mathrm{ABS}}$ under attribute string x^*, $\mathcal{B}$ forwards the message $\mathsf{msg}^{(\theta)}$ to $\mathcal{C}$ and receives back a signature $\sigma_{\mathrm{SIG}}^{(\theta)}$ on $\mathsf{msg}^{(\theta)}$ from $\mathcal{C}$. $\mathcal{B}$ provides, $\sigma_{\mathrm{ABS}}^{(\theta)} = (\mathrm{VK}_{\mathrm{SIG}}^*, \sigma_{\mathrm{SIG}}^{(\theta)})$ to $\mathcal{A}$.
- At the end of interaction, $\mathcal{A}$ outputs a signature $\sigma_{\mathrm{ABS}}^* = (\widehat{\mathrm{VK}}_{\mathrm{SIG}}^*, \sigma_{\mathrm{SIG}}^*)$ on some message msg^* under attribute string x^*. $\mathcal{B}$ outputs $(\mathsf{msg}^*, \sigma_{\mathrm{SIG}}^*)$ as a forgery in its existential unforgeability experiment against SIG.

Observe that the simulation of the experiment Hyb_3 by $\mathcal{B}$ is perfect. Now, if $\mathcal{A}$ wins in the above simulated experiment, then the following must hold simultaneously:

(I) $\mathsf{ABS.Verify}(\mathrm{PP}_{\mathrm{ABS}}, x^*, \mathsf{msg}^*, \sigma_{\mathrm{ABS}}^*) = 1$.
(II) $\mathsf{msg}^* \neq \mathsf{msg}^{(\theta)}$ for any $\theta \in [\hat{q}_{\mathrm{SIGN}}]$.

Note that $\mathsf{ABS.Verify}(\mathrm{PP}_{\mathrm{ABS}}, x^*, \mathsf{msg}^*, \sigma_{\mathrm{ABS}}^*) = 1$ implies $[\widehat{\mathrm{VK}}_{\mathrm{SIG}}^* = \mathrm{VK}_{\mathrm{SIG}}^*] \wedge [\mathsf{SIG.Verify}(\widehat{\mathrm{VK}}_{\mathrm{SIG}}^*, \mathsf{msg}^*, \sigma_{\mathrm{SIG}}^*) = 1]$, i.e., $\mathsf{SIG.Verify}(\mathrm{VK}_{\mathrm{SIG}}^*, \mathsf{msg}^*, \sigma_{\mathrm{SIG}}^*) = 1$. Further, notice that $\mathsf{msg}^{(\theta)}$, for $\theta \in [\hat{q}_{\mathrm{SIGN}}]$, are the only messages that $\mathcal{B}$ queried a signature on to $\mathcal{C}$. Thus, $(\mathsf{msg}^*, \sigma_{\mathrm{SIG}}^*)$ is indeed a valid forgery in the existential unforgeability experiment against SIG. □

C Lemmas for the proof of Lemma B.1

Lemma C.1 *Assuming $\mathcal{IO}$ is a secure indistinguishability obfuscator for $\mathsf{P/poly}$, $\mathcal{F}$ is a secure puncturable pseudorandom function, and SPS is a splittable signature scheme satisfying '$\mathrm{VK}_{\text{SPS-REJ}}$ indistinguishability', for any PPT adversary $\mathcal{A}$, for any security parameter λ, $|\mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,0)}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,1)}(\lambda)| \leq \mathsf{negl}(\lambda)$ for some negligible function negl.*

Proof To establish Lemma C.1, we introduce $t^{*(\nu)} + 1$ intermediate hybrid experiments between $\mathsf{Hyb}_{0,\nu-1,0}$ and $\mathsf{Hyb}_{0,\nu-1,1}$, namely, $\mathsf{Hyb}_{0,\nu-1,0,\gamma}$, for $\gamma \in [0, t^{*(\nu)}]$ such that $\mathsf{Hyb}_{0,\nu-1,0,t^{*(\nu)}}$ coincides with $\mathsf{Hyb}_{0,\nu-1,0}$ and $\mathsf{Hyb}_{0,\nu-1,0,0}$ coincides with $\mathsf{Hyb}_{0,\nu-1,1}$.

Sequence of Intermediate Hybrids Between $\mathsf{Hyb}_{0,\nu-1,0}$ and $\mathsf{Hyb}_{0,\nu-1,1}$

$\mathsf{Hyb}_{0,\nu-1,0,\gamma}$ ($\gamma = 0, \ldots, t^{*(\nu)}$): In this experiment in response to the ν^{th} signing key query of $\mathcal{A}$ corresponding to TM $M^{(\nu)} \in \mathbb{M}_\lambda$ with $M^{(\nu)}(x^*) = 0$, $\mathcal{B}$ proceeds as follows:

1. It first picks PPRF keys $K_1^{(\nu)}, \ldots, K_\lambda^{(\nu)}$, $K_{\mathrm{SPS},A}^{(\nu)}, \underline{K_{\mathrm{SPS},B}^{(\nu)}}, K_{\mathrm{SPS},E}^{(\nu)} \xleftarrow{\$} \mathcal{F}.\mathsf{Setup}(1^\lambda)$.
2. Next, it generates $(\mathrm{PP}_{\mathrm{ACC}}^{(\nu)}, w_0^{(\nu)}, \mathrm{STORE}_0^{(\nu)}) \xleftarrow{\$} \mathsf{ACC.Setup}(1^\lambda, n_{\text{ACC-BLK}} = 2^\lambda)$ and $(\mathrm{PP}_{\mathrm{ITR}}^{(\nu)}, v_0^{(\nu)}) \xleftarrow{\$} \mathsf{ITR.Setup}(1^\lambda, n_{\mathrm{ITR}} = 2^\lambda)$.

3. It provides $\mathcal{A}$ with the signing key

$$\mathrm{SK}_{\mathrm{ABS}}\{M^{(\nu)}\} = \left(\begin{array}{l} \mathrm{HK}, \mathrm{PP}_{\mathrm{ACC}}^{(\nu)}, w_0^{(\nu)}, \mathrm{STORE}_0^{(\nu)}, \mathrm{PP}_{\mathrm{ITR}}^{(\nu)}, v_0^{(\nu)}, \\ \mathcal{IO}(\mathsf{Init\text{-}SPS.Prog}[q_0^{(\nu)}, w_0^{(\nu)}, v_0^{(\nu)}, K_{\mathrm{SPS},E}^{(\nu)}]), \\ \mathcal{IO}(\mathsf{Accumulate.Prog}[n_{\text{SSB-BLK}} = 2^\lambda, \mathrm{HK}, \mathrm{PP}_{\mathrm{ACC}}^{(\nu)}, \mathrm{PP}_{\mathrm{ITR}}^{(\nu)}, K_{\mathrm{SPS},E}^{(\nu)}]), \\ \mathcal{IO}(\mathsf{Change\text{-}SPS.Prog}[K_{\mathrm{SPS},A}^{(\nu)}, K_{\mathrm{SPS},E}^{(\nu)}]), \\ \mathcal{IO}(\underline{\mathsf{Constrained\text{-}Key.Prog}_{\mathrm{ABS}}^{(0,\gamma)}[M^{(\nu)}, T = 2^\lambda, t^{*(\nu)}, \mathrm{PP}_{\mathrm{ACC}}^{(\nu)}, \mathrm{PP}_{\mathrm{ITR}}^{(\nu)}, K,} \\ \quad \underline{K_1^{(\nu)}, \ldots, K_\lambda^{(\nu)}, K_{\mathrm{SPS},A}^{(\nu)}, K_{\mathrm{SPS},B}^{(\nu)}, h^*, \ell^*])} \end{array}\right),$$

where the program $\mathsf{Constrained\text{-}Key.Prog}_{\mathrm{ABS}}^{(0,\gamma)}$, depicted in Fig. **22**, is a modification of the program $\mathsf{Constrained\text{-}Key.Prog}_{\mathrm{ABS}}^{(1)}$, shown in Fig. **10**.

The rest of the experiment is identical to $\mathsf{Hyb}_{0,\nu-1,0}$.

Constants: TM $M = \langle Q, \Sigma_{\mathrm{INP}}, \Sigma_{\mathrm{TAPE}}, \delta, q_0, q_{\mathrm{AC}}, q_{\mathrm{REJ}} \rangle$, Time bound $T = 2^\lambda$, Running time on challenge input t^*, Public parameters for positional accumulator $\mathrm{PP}_{\mathrm{ACC}}$, Public parameters for iterator $\mathrm{PP}_{\mathrm{ITR}}$, PPRF keys $K, K_1, \ldots, K_\lambda, K_{\mathrm{SPS},A}, K_{\mathrm{SPS},B}$, SSB hash value of challenge input h^*, Length of challenge input ℓ^*

Inputs: Time t, String $\mathrm{SEED}_{\mathrm{IN}}$, Header position $\mathrm{POS}_{\mathrm{IN}}$, Symbol $\mathrm{SYM}_{\mathrm{IN}}$, TM state $\mathrm{ST}_{\mathrm{IN}}$, Accumulator value w_{IN}, Accumulator proof π_{ACC}, Auxiliary value AUX, Iterator value v_{IN}, SSB hash value h, Length ℓ_{INP}, Signature $\sigma_{\mathrm{SPS,IN}}$

Output: CPRF evaluation $\mathcal{F}(K, (h, \ell_{\mathrm{INP}}))$, or (Header Position $\mathrm{POS}_{\mathrm{OUT}}$, Symbol $\mathrm{SYM}_{\mathrm{OUT}}$, TM state $\mathrm{ST}_{\mathrm{OUT}}$, Accumulator value w_{OUT}, Iterator value v_{OUT}, Signature $\sigma_{\mathrm{SPS,OUT}}$, String $\mathrm{SEED}_{\mathrm{OUT}}$), or $\perp$

1. Identify an integer τ such that $2^\tau \leq t < 2^{\tau+1}$. If $[\mathsf{PRG}(\mathrm{SEED}_{\mathrm{IN}}) \neq \mathsf{PRG}(\mathcal{F}(K_\tau, (h, \ell_{\mathrm{INP}})))] \wedge [t > 1]$, output $\perp$.
2. If $\mathsf{ACC.Verify\text{-}Read}(\mathrm{PP}_{\mathrm{ACC}}, w_{\mathrm{IN}}, \mathrm{SYM}_{\mathrm{IN}}, \mathrm{POS}_{\mathrm{IN}}, \pi_{\mathrm{ACC}}) = 0$, output $\perp$.
3. (a) Compute $r_{\mathrm{SPS},A} = \mathcal{F}(K_{\mathrm{SPS},A}, (h, \ell_{\mathrm{INP}}, t-1)), (\mathrm{SK}_{\mathrm{SPS},A}, \mathrm{VK}_{\mathrm{SPS},A}, \mathrm{VK}_{\text{SPS-REJ},A}) = \mathsf{SPS.Setup}(1^\lambda; r_{\mathrm{SPS},A})$.
   (b) Compute $r_{\mathrm{SPS},B} = \mathcal{F}(K_{\mathrm{SPS},B}, (h, \ell_{\mathrm{INP}}, t-1)), (\mathrm{SK}_{\mathrm{SPS},B}, \mathrm{VK}_{\mathrm{SPS},B}, \mathrm{VK}_{\text{SPS-REJ},B}) = \mathsf{SPS.Setup}(1^\lambda; r_{\mathrm{SPS},B})$.
   (c) Set $m_{\mathrm{IN}} = (v_{\mathrm{IN}}, \mathrm{ST}_{\mathrm{IN}}, w_{\mathrm{IN}}, \mathrm{POS}_{\mathrm{IN}})$ and $\alpha =$'-'.
   (d) If $\mathsf{SPS.Verify}(\mathrm{VK}_{\mathrm{SPS},A}, m_{\mathrm{IN}}, \sigma_{\mathrm{SPS,IN}}) = 1$, set $\alpha =$'A'.
   (e) If $[\alpha =$'-'$] \wedge [(t > t^*) \vee (t \leq \gamma) \vee (h \neq h^*) \vee (\ell_{\mathrm{INP}} \neq \ell^*)]$, output $\perp$.
   Else if $[\alpha =$'-'$] \wedge [\mathsf{SPS.Verify}(\mathrm{VK}_{\mathrm{SPS},B}, m_{\mathrm{IN}}, \sigma_{\mathrm{SPS,IN}}) = 1]$, set $\alpha =$'B'.
   (f) If $\alpha =$'-', output $\perp$.
4. (a) Compute $(\mathrm{ST}_{\mathrm{OUT}}, \mathrm{SYM}_{\mathrm{OUT}}, \beta) = \delta(\mathrm{ST}_{\mathrm{IN}}, \mathrm{SYM}_{\mathrm{IN}})$ and $\mathrm{POS}_{\mathrm{OUT}} = \mathrm{POS}_{\mathrm{IN}} + \beta$.
   (b) If $\mathrm{ST}_{\mathrm{OUT}} = q_{\mathrm{REJ}}$, output $\perp$.
   Else if $[\mathrm{ST}_{\mathrm{OUT}} = q_{\mathrm{AC}}] \wedge [\alpha =$'$B$'$]$, output $\perp$.
   Else if $\mathrm{ST}_{\mathrm{OUT}} = q_{\mathrm{AC}}$, perform the following:
   (I) Compute $r_{\mathrm{SIG}} = \mathcal{F}(K, (h, \ell_{\mathrm{INP}}))$, $(\mathrm{SK}_{\mathrm{SIG}}, \mathrm{VK}_{\mathrm{SIG}}) = \mathsf{SIG.Setup}(1^\lambda; r_{\mathrm{SIG}})$.
   (II) Output $(\mathrm{SK}_{\mathrm{SIG}}, \mathrm{VK}_{\mathrm{SIG}})$.
5. (a) Compute $w_{\mathrm{OUT}} = \mathsf{ACC.Update}(\mathrm{PP}_{\mathrm{ACC}}, w_{\mathrm{IN}}, \mathrm{SYM}_{\mathrm{OUT}}, \mathrm{POS}_{\mathrm{IN}}, \mathrm{AUX})$. If $w_{\mathrm{OUT}} = \perp$, output $\perp$.
   (b) Compute $v_{\mathrm{OUT}} = \mathsf{ITR.Iterate}(\mathrm{PP}_{\mathrm{ITR}}, v_{\mathrm{IN}}, (\mathrm{ST}_{\mathrm{IN}}, w_{\mathrm{IN}}, \mathrm{POS}_{\mathrm{IN}}))$.
6. (a) Compute $r'_{\mathrm{SPS},A} = \mathcal{F}(K_{\mathrm{SPS},A}, (h, \ell_{\mathrm{INP}}, t)), (\mathrm{SK}'_{\mathrm{SPS},A}, \mathrm{VK}'_{\mathrm{SPS},A}, \mathrm{VK}'_{\text{SPS-REJ},A}) = \mathsf{SPS.Setup}(1^\lambda; r'_{\mathrm{SPS},A})$.
   (b) Compute $r'_{\mathrm{SPS},B} = \mathcal{F}(K_{\mathrm{SPS},B}, (h, \ell_{\mathrm{INP}}, t)), (\mathrm{SK}'_{\mathrm{SPS},B}, \mathrm{VK}'_{\mathrm{SPS},B}, \mathrm{VK}'_{\text{SPS-REJ},B}) = \mathsf{SPS.Setup}(1^\lambda; r'_{\mathrm{SPS},B})$.
   (c) Set $m_{\mathrm{OUT}} = (v_{\mathrm{OUT}}, \mathrm{ST}_{\mathrm{OUT}}, w_{\mathrm{OUT}}, \mathrm{POS}_{\mathrm{OUT}})$.
   Compute $\sigma_{\mathrm{SPS,OUT}} = \mathsf{SPS.Sign}(\mathrm{SK}'_{\mathrm{SPS},\alpha}, m_{\mathrm{OUT}})$.
7. If $t + 1 = 2^{\tau'}$, for some integer τ', set $\mathrm{SEED}_{\mathrm{OUT}} = \mathcal{F}(K_{\tau'}, (h, \ell_{\mathrm{INP}}))$.
   Else, set $\mathrm{SEED}_{\mathrm{OUT}} = \epsilon$.
8. Output $(\mathrm{POS}_{\mathrm{OUT}}, \mathrm{SYM}_{\mathrm{OUT}}, \mathrm{ST}_{\mathrm{OUT}}, w_{\mathrm{OUT}}, v_{\mathrm{OUT}}, \sigma_{\mathrm{SPS,OUT}}, \mathrm{SEED}_{\mathrm{OUT}})$.

Fig. 22 $\mathsf{Constrained\text{-}Key.Prog}_{\mathrm{ABS}}^{(0,\gamma)}$

Analysis

Let us denote by $\mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,0,\gamma)}(\lambda)$ the advantage of $\mathcal{A}$, i.e., the absolute difference between $1/2$ and $\mathcal{A}$'s probability of correctly guessing the random bit selected by the challenger $\mathcal{B}$, in the hybrid experiment $\mathsf{Hyb}_{0,\nu-1,0,\gamma}$, for $\gamma \in [0, t^{*(\nu)}]$. Clearly, $\mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,0)}(\lambda) \equiv \mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,0,t^{*(\nu)})}(\lambda)$ and $\mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,1)}(\lambda) \equiv \mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,0,0)}(\lambda)$. Hence, we have

$$|\mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,0)}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,1)}(\lambda)| \leq \sum_{\gamma=1}^{t^{*(\nu)}} |\mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,0,\gamma)}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,0,\gamma-1)}(\lambda)|. \tag{C.1}$$

Claim C.1 below justifies that the RHS of Eq. (C.1) is negligible and consequently Lemma C.1 follows.

Claim C.1 *Assuming $\mathcal{IO}$ is a secure indistinguishability obfuscator for* P/poly*, $\mathcal{F}$ is a secure puncturable pseudorandom function, and* SPS *is a splittable signature scheme satisfying '$\mathrm{VK}_{\text{SPS-REJ}}$ indistinguishability', for any* PPT *adversary $\mathcal{A}$, for any security parameter λ, $|\mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,0,\gamma)}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,0,\gamma-1)}(\lambda)| \leq \mathsf{negl}(\lambda)$ for some negligible function* negl*.*

Proof The proof of Claim C.1 is similar to that of Claim B.1 of [7]. □

□

Lemma C.2 *Assuming $\mathcal{IO}$ is a secure indistinguishability obfuscator for* P/poly*, $\mathcal{F}$ is a secure puncturable pseudorandom function, and* SPS *is a splittable signature scheme satisfying '$\mathrm{VK}_{\text{SPS-REJ}}$ indistinguishability', for any* PPT *adversary $\mathcal{A}$, for any security parameter λ, $|\mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,1)}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,2)}(\lambda)| \leq \mathsf{negl}(\lambda)$ for some negligible function* negl*.*

Proof To prove Lemma C.2, we consider the following sequence of intermediate hybrid experiments between $\mathsf{Hyb}_{0,\nu-1,1}$ and $\mathsf{Hyb}_{0,\nu-1,2}$:

Sequence of Intermediate Hybrids between $\mathsf{Hyb}_{0,\nu-1,1}$ and $\mathsf{Hyb}_{0,\nu-1,2}$

$\mathsf{Hyb}_{0,\nu-1,1,0}$: This experiment coincides with $\mathsf{Hyb}_{0,\nu-1,1}$.

$\mathsf{Hyb}_{0,\nu-1,1,1}$: In this experiment, to answer the ν^{th} signing key query of $\mathcal{A}$ corresponding to TM $M^{(\nu)} \in \mathbb{M}_\lambda$ with $M^{(\nu)}(x^*) = 0$, $\mathcal{B}$ selects an additional PPRF key $K_{\text{SPS},F}^{(\nu)} \xleftarrow{\$} \mathcal{F}.\mathsf{Setup}(1^\lambda)$ along with all the other PPRF keys as well as the public parameters for positional accumulator and iterator as generated in $\mathsf{Hyb}_{0,\nu-1,1,0}$, providing $\mathcal{A}$ with the signing key

$$\mathrm{SK}_{\text{ABS}}\{M^{(\nu)}\} = \left(\begin{array}{l} \mathrm{HK}, \mathrm{PP}_{\text{ACC}}^{(\nu)}, w_0^{(\nu)}, \mathrm{STORE}_0^{(\nu)}, \mathrm{PP}_{\text{ITR}}^{(\nu)}, v_0^{(\nu)}, \\ \mathcal{IO}(\mathsf{Init\text{-}SPS.Prog}[q_0^{(\nu)}, w_0^{(\nu)}, v_0^{(\nu)}, K_{\text{SPS},E}^{(\nu)}]), \\ \mathcal{IO}(\mathsf{Accumulate.Prog}^{(0,1)}[n_{\text{SSB-BLK}} = 2^\lambda, \mathrm{HK}, \mathrm{PP}_{\text{ACC}}^{(\nu)}, \mathrm{PP}_{\text{ITR}}^{(\nu)}, K_{\text{SPS},E}^{(\nu)}, \\ \quad K_{\text{SPS},F}^{(\nu)}, h^*, \ell^*]), \\ \mathcal{IO}(\mathsf{Change\text{-}SPS.Prog}^{(0,1)}[K_{\text{SPS},A}^{(\nu)}, K_{\text{SPS},B}^{(\nu)}, K_{\text{SPS},E}^{(\nu)}, K_{\text{SPS},F}^{(\nu)}, h^*, \ell^*]), \\ \mathcal{IO}(\mathsf{Constrained\text{-}Key.Prog}_{\text{ABS}}^{(1)}[M^{(\nu)}, T = 2^\lambda, t^{*(\nu)}, \mathrm{PP}_{\text{ACC}}^{(\nu)}, \mathrm{PP}_{\text{ITR}}^{(\nu)}, K, \\ \quad K_1^{(\nu)}, \ldots, K_\lambda^{(\nu)}, K_{\text{SPS},A}^{(\nu)}, K_{\text{SPS},B}^{(\nu)}, h^*, \ell^*]) \end{array}\right),$$

where the programs $\mathsf{Accumulate.Prog}^{(0,1)}$ and $\mathsf{Change\text{-}SPS.Prog}^{(0,1)}$ are the alterations of the programs $\mathsf{Accumulate.Prog}^{(1)}$ and $\mathsf{Change\text{-}SPS.Prog}^{(1)}$ (Figs. **11** and **12**) and are depicted in Figs. **23** and **24** respectively. The rest of the experiment proceeds in the same way as in $\mathsf{Hyb}_{0,\nu-1,1,0}$.

Constants: Maximum number of blocks for SSB hash $n_{\text{SSB-BLK}} = 2^\lambda$, SSB hash key HK, Public parameters for positional accumulator PP_{ACC}, Public parameters for iterator PP_{ITR}, PPRF keys $K_{\text{SPS},E}$, $K_{\text{SPS},F}$, SSB hash value of challenge input h^*, Length of challenge input ℓ^*

Inputs: Index i, Symbol SYM_{IN}, TM state ST, Accumulator value w_{IN}, Auxiliary value AUX, Iterator value v_{IN}, Signature $\sigma_{\text{SPS,IN}}$, SSB hash value h, SSB opening value π_{SSB}

Output: (Accumulator value w_{OUT}, Iterator value v_{OUT}, Signature $\sigma_{\text{SPS-OUT}}$), or $\perp$

1. (a) Compute $r_{\text{SPS},E} = \mathcal{F}(K_{\text{SPS},E}, (h,i))$, $(\text{SK}_{\text{SPS},E}, \text{VK}_{\text{SPS},E}, \text{VK}_{\text{SPS-REJ},E}) = \mathsf{SPS.Setup}(1^\lambda; r_{\text{SPS},E})$.
   (b) Compute $r_{\text{SPS},F} = \mathcal{F}(K_{\text{SPS},F}, (h,i))$, $(\text{SK}_{\text{SPS},F}, \text{VK}_{\text{SPS},F}, \text{VK}_{\text{SPS-REJ},F}) = \mathsf{SPS.Setup}(1^\lambda; r_{\text{SPS},F})$.
   (c) If $(h,i) = (h^*, \ell^*)$, set $\text{VK} = \text{VK}_{\text{SPS-REJ},F}$.
   (d) Set $m_{\text{IN}} = (v_{\text{IN}}, \text{ST}, w_{\text{IN}}, 0)$ and $\alpha =$'-'.
   (e) If $\mathsf{SPS.Verify}(\text{VK}_{\text{SPS},E}, m_{\text{IN}}, \sigma_{\text{SPS,IN}}) = 1$, set $\alpha =$'E'.
   (f) If $[\alpha =$'-'$] \wedge [(i \neq \ell^*) \vee (h \neq h^*)]$, output $\perp$.
   Else if $[\alpha =$'-'$] \wedge [\mathsf{SPS.Verify}(\text{VK}, m_{\text{IN}}, \sigma_{\text{SPS,IN}}) = 0]$, output $\perp$.
   Else if $[\alpha =$'-'$] \wedge [\mathsf{SPS.Verify}(\text{VK}, m_{\text{IN}}, \sigma_{\text{SPS,IN}}) = 1]$, set $\alpha =$'F'.
   (g) If $\alpha =$'-', output $\perp$.
2. If $\mathsf{SSB.Verify}(\text{HK}, h, i, \text{SYM}_{\text{IN}}, \pi_{\text{SSB}}) = 0$, output $\perp$.
3. (a) Compute $w_{\text{OUT}} = \mathsf{ACC.Update}(\text{PP}_{\text{ACC}}, w_{\text{IN}}, \text{SYM}_{\text{IN}}, i, \text{AUX})$. If $w_{\text{OUT}} = \perp$, output $\perp$.
   (b) Compute $v_{\text{OUT}} = \mathsf{ITR.Iterate}(\text{PP}_{\text{ITR}}, v_{\text{IN}}, (\text{ST}, w_{\text{IN}}, 0))$.
4. (a) Compute $r'_{\text{SPS},E} = \mathcal{F}(K_{\text{SPS},E}, (h, i+1))$, $(\text{SK}'_{\text{SPS},E}, \text{VK}'_{\text{SPS},E}, \text{VK}'_{\text{SPS-REJ},E}) = \mathsf{SPS.Setup}(1^\lambda; r'_{\text{SPS},E})$.
   (b) Set $m_{\text{OUT}} = (v_{\text{OUT}}, \text{ST}, w_{\text{OUT}}, 0)$. Compute $\sigma_{\text{SPS,OUT}} = \mathsf{SPS.Sign}(\text{SK}'_{\text{SPS},E}, m_{\text{OUT}})$.
5. Output $(w_{\text{OUT}}, v_{\text{OUT}}, \sigma_{\text{SPS,OUT}})$.

Fig. 23 $\mathsf{Accumulate.Prog}^{(0,1)}$

Constants: PPRF keys $K_{\text{SPS},A}, K_{\text{SPS},B}, K_{\text{SPS},E}, K_{\text{SPS},F}$, SSB hash value of challenge input h^*, Length of challenge input ℓ^*

Inputs: TM state ST, Accumulator value w, Iterator value v, SSB hash value h, Length ℓ_{INP}, Signature $\sigma_{\text{SPS,IN}}$

Output: Signature $\sigma_{\text{SPS,OUT}}$, or $\perp$

1. (a) Compute $r_{\text{SPS},E} = \mathcal{F}(K_{\text{SPS},E}, (h, \ell_{\text{INP}}))$, $(\text{SK}_{\text{SPS},E}, \text{VK}_{\text{SPS},E}, \text{VK}_{\text{SPS-REJ},E}) = \mathsf{SPS.Setup}(1^\lambda; r_{\text{SPS},E})$.
   (b) Compute $r_{\text{SPS},F} = \mathcal{F}(K_{\text{SPS},F}, (h, \ell_{\text{INP}}))$, $(\text{SK}_{\text{SPS},F}, \text{VK}_{\text{SPS},F}, \text{VK}_{\text{SPS-REJ},F}) = \mathsf{SPS.Setup}(1^\lambda; r_{\text{SPS},F})$.
   (c) If $(h, \ell_{\text{INP}}) = (h^*, \ell^*)$, set $\text{VK} = \text{VK}_{\text{SPS-REJ},F}$.
   (d) Set $m = (v, \text{ST}, w, 0)$ and $\alpha =$'-'.
   (e) If $\mathsf{SPS.Verify}(\text{VK}_{\text{SPS},E}, m, \sigma_{\text{SPS,IN}}) = 1$, set $\alpha =$'E'.
   (f) If $[\alpha =$'-'$] \wedge [\ell_{\text{INP}} \neq \ell^*) \vee (h \neq h^*)]$, output $\perp$.
   Else if $[\alpha =$'-'$] \wedge [\mathsf{SPS.Verify}(\text{VK}, m, \sigma_{\text{SPS,IN}}) = 0]$, output $\perp$.
   Else if $[\alpha =$'-'$] \wedge [\mathsf{SPS.Verify}(\text{VK}, m, \sigma_{\text{SPS,IN}}) = 1]$, set $\alpha =$'F'.
   (g) If $\alpha =$'-', output $\perp$.
2. (a) Compute $r_{\text{SPS},A} = \mathcal{F}(K_{\text{SPS},A}, (h, \ell_{\text{INP}}, 0))$, $(\text{SK}_{\text{SPS},A}, \text{VK}_{\text{SPS},A}, \text{VK}_{\text{SPS-REJ},A}) = \mathsf{SPS.Setup}(1^\lambda; r_{\text{SPS},A})$.
   (b) Compute $r_{\text{SPS},B} = \mathcal{F}(K_{\text{SPS},B}, (h, \ell_{\text{INP}}, 0))$, $(\text{SK}_{\text{SPS},B}, \text{VK}_{\text{SPS},B}, \text{VK}_{\text{SPS-REJ},B}) = \mathsf{SPS.Setup}(1^\lambda; r_{\text{DSPS},B})$.
   (c) If $[(h, \ell_{\text{INP}}) = (h^*, \ell^*)] \wedge [\alpha =$'$F$'$]$, output $\sigma_{\text{SPS,OUT}} = \mathsf{SPS.Sign}(\text{SK}_{\text{SPS},B}, m)$.
   Else, output $\sigma_{\text{SPS,OUT}} = \mathsf{SPS.Sign}(\text{SK}_{\text{SPS},A}, m)$.

Fig. 24 $\mathsf{Change\text{-}SPS.Prog}^{(0,1)}$

$\mathsf{Hyb}_{0,\nu-1,1,2}$: In this experiment, in response to the ν^{th} signing key query of $\mathcal{A}$ corresponding to TM $M^{(\nu)} \in \mathbb{M}_\lambda$ with $M^{(\nu)}(x^*) = 0$, $\mathcal{B}$ proceeds as follows:

1. It first generates all the PPRF keys as well as the public parameters for the positional accumulator and the iterator just as in $\mathsf{Hyb}_{0,\nu-1,1,1}$.

2. Next it creates the punctured PPRF key $K^{(\nu)}_{\text{SPS},F}\{(h^*,\ell^*)\} \xleftarrow{\$} \mathcal{F}.\mathsf{Puncture}(K^{(\nu)}_{\text{SPS},F},(h^*,\ell^*))$ as well as computes $r^{(\nu,\ell^*)}_{\text{SPS},H} = \mathcal{F}(K^{(\nu)}_{\text{SPS},F},(h^*,\ell^*))$ and $(\text{SK}^{(\nu,\ell^*)}_{\text{SPS},H}, \text{VK}^{(\nu,\ell^*)}_{\text{SPS},H}, \text{VK}^{(\nu,\ell^*)}_{\text{SPS-REJ},H}) = \mathsf{SPS.Setup}(1^\lambda; r^{(\nu,\ell^*)}_{\text{SPS},H})$.
3. It hands $\mathcal{A}$ the signing key

$$\text{SK}_{\text{ABS}}\{M^{(\nu)}\} = \left(\begin{array}{l} \text{HK}, \text{PP}^{(\nu)}_{\text{ACC}}, w^{(\nu)}_0, \text{STORE}^{(\nu)}_0, \text{PP}^{(\nu)}_{\text{ITR}}, v^{(\nu)}_0, \\ \mathcal{IO}(\mathsf{Init\text{-}SPS.Prog}[q^{(\nu)}_0, w^{(\nu)}_0, v^{(\nu)}_0, K^{(\nu)}_{\text{SPS},E}]), \\ \mathcal{IO}(\mathsf{Accumulate.Prog}^{(0,2)}[n_{\text{SSB-BLK}} = 2^\lambda, \text{HK}, \text{PP}^{(\nu)}_{\text{ACC}}, \text{PP}^{(\nu)}_{\text{ITR}}, K^{(\nu)}_{\text{SPS},E}, \\ \quad \underline{K^{(\nu)}_{\text{SPS},F}\{(h^*,\ell^*)\}, \text{VK}^{(\nu,\ell^*)}_{\text{SPS-REJ},H}}, h^*, \ell^*]), \\ \mathcal{IO}(\mathsf{Change\text{-}SPS.Prog}^{(0,2)}[K^{(\nu)}_{\text{SPS},A}, K^{(\nu)}_{\text{SPS},B}, K^{(\nu)}_{\text{SPS},E}, \underline{K^{(\nu)}_{\text{SPS},F}\{(h^*,\ell^*)\}}, \\ \quad \underline{\text{VK}^{(\nu,\ell^*)}_{\text{SPS-REJ},H}}, h^*, \ell^*]), \\ \mathcal{IO}(\mathsf{Constrained\text{-}Key.Prog}^{(1)}_{\text{ABS}}[M^{(\nu)}, T = 2^\lambda, t^{*(\nu)}, \text{PP}^{(\nu)}_{\text{ACC}}, \text{PP}^{(\nu)}_{\text{ITR}}, K, \\ \quad K^{(\nu)}_1, \ldots, K^{(\nu)}_\lambda, K^{(\nu)}_{\text{SPS},A}, K^{(\nu)}_{\text{SPS},B}, h^*, \ell^*]) \end{array}\right),$$

where the programs $\mathsf{Accumulate.Prog}^{(0,2)}$ and $\mathsf{Change\text{-}SPS.Prog}^{(0,2)}$ are the modifications of the programs $\mathsf{Accumulate.Prog}^{(0,1)}$ and $\mathsf{Change\text{-}SPS.Prog}^{(0,1)}$ (Figs. **23** and **24**) and are described in Figs. **25** and **26** respectively.

The remaining part of the experiment is analogous to $\mathsf{Hyb}_{0,\nu-1,1,1}$.

Constants: Maximum number of blocks for SSB hash $n_{\text{SSB-BLK}} = 2^\lambda$, SSB hash key HK, Public parameters for positional accumulator PP_{ACC}, Public parameters for iterator PP_{ITR}, PPRF key $K_{\text{SPS},E}$, Punctured PPRF key $K_{\text{SPS},F}\{(h^*,\ell^*)\}$, Verification key VK_H, SSB hash value of challenge input h^*, Length of challenge input ℓ^*

Inputs: Index i, Symbol SYM_{IN}, TM state ST, Accumulator value w_{IN}, Auxiliary value AUX, Iterator value v_{IN}, Signature $\sigma_{\text{SPS,IN}}$, SSB hash value h, SSB opening value π_{SSB}

Output: (Accumulator value w_{OUT}, Iterator value v_{OUT}, Signature $\sigma_{\text{SPS-OUT}}$), or $\perp$

1. (a) Compute $r_{\text{SPS},E} = \mathcal{F}(K_{\text{SPS},E},(h,i))$, $(\text{SK}_{\text{SPS},E}, \text{VK}_{\text{SPS},E}, \text{VK}_{\text{SPS-REJ},E}) = \mathsf{SPS.Setup}(1^\lambda; r_{\text{SPS},E})$.
   (b) Compute $r_{\text{SPS},F} = \mathcal{F}(K_{\text{SPS},F}\{(h^*,\ell^*)\},(h,i))$, $(\text{SK}_{\text{SPS},F}, \text{VK}_{\text{SPS},F}, \text{VK}_{\text{SPS-REJ},F}) = \mathsf{SPS.Setup}(1^\lambda; r_{\text{SPS},F})$.
   (c) Set $m_{\text{IN}} = (v_{\text{IN}}, \text{ST}, w_{\text{IN}}, 0)$ and $\alpha =$'-'.
   (d) If $\mathsf{SPS.Verify}(\text{VK}_{\text{SPS},E}, m_{\text{IN}}, \sigma_{\text{SPS,IN}}) = 1$, set $\alpha =$'E'.
   (e) If $[\alpha =\text{'-'}] \wedge [(i \neq \ell^*) \vee (h \neq h^*)]$, output $\perp$.
   Else if $[\alpha=\text{'-'}] \wedge \underline{[\mathsf{SPS.Verify}(\text{VK}_H, m_{\text{IN}}, \sigma_{\text{SPS,IN}}) = 0]}$, output $\perp$.
   Else if $[\alpha =\text{'-'}] \wedge \underline{[\mathsf{SPS.Verify}(\text{VK}_H, m_{\text{IN}}, \sigma_{\text{SPS,IN}}) = 1]}$, set $\alpha =$'F'.
   (f) If $\alpha =$'-', output $\perp$.
2. If $\mathsf{SSB.Verify}(\text{HK}, h, i, \text{SYM}_{\text{IN}}, \pi_{\text{SSB}}) = 0$, output $\perp$.
3. (a) Compute $w_{\text{OUT}} = \mathsf{ACC.Update}(\text{PP}_{\text{ACC}}, w_{\text{IN}}, \text{SYM}_{\text{IN}}, i, \text{AUX})$. If $w_{\text{OUT}} = \perp$, output $\perp$.
   (b) Compute $v_{\text{OUT}} = \mathsf{ITR.Iterate}(\text{PP}_{\text{ITR}}, v_{\text{IN}}, (\text{ST}, w_{\text{IN}}, 0))$.
4. (a) Compute $r'_{\text{SPS},E} = \mathcal{F}(K_{\text{SPS},E},(h,i+1))$, $(\text{SK}'_{\text{SPS},E}, \text{VK}'_{\text{SPS},E}, \text{VK}'_{\text{SPS-REJ},E}) = \mathsf{SPS.Setup}(1^\lambda; r'_{\text{SPS},E})$.
   (b) Set $m_{\text{OUT}} = (v_{\text{OUT}}, \text{ST}, w_{\text{OUT}}, 0)$. Compute $\sigma_{\text{SPS,OUT}} = \mathsf{SPS.Sign}(\text{SK}'_{\text{SPS},E}, m_{\text{OUT}})$.
5. Output $(w_{\text{OUT}}, v_{\text{OUT}}, \sigma_{\text{SPS,OUT}})$.

Fig. 25 $\mathsf{Accumulate.Prog}^{(0,2)}$

$\mathsf{Hyb}_{0,\nu-1,1,3}$: This experiment is identical to $\mathsf{Hyb}_{0,\nu-1,1,2}$ except that while creating the ν^{th} signing key queried by $\mathcal{A}$ corresponding to TM $M^{(\nu)} \in \mathbb{M}_\lambda$ with $M^{(\nu)}(x^*) = 0$, $\mathcal{B}$ selects $\underline{r^{(\nu,\ell^*)}_{\text{SPS},H} \xleftarrow{\$} \mathcal{Y}_{\text{PPRF}}}$, i.e., in other words, $\mathcal{B}$ generates $\underline{(\text{SK}^{(\nu,\ell^*)}_{\text{SPS},H},}$

Constants: PPRF keys $K_{\text{SPS},A}, K_{\text{SPS},B}, K_{\text{SPS},E}$, Punctured PPRF key $K_{\text{SPS},F}\{(h^*,\ell^*)\}$, Verification key VK_H, SSB hash value of challenge input h^*, Length of challenge input ℓ^*

Inputs: TM state ST, Accumulator value w, Iterator value v, SSB hash value h, Length ℓ_{INP}, Signature $\sigma_{\text{SPS,IN}}$

Output: Signature $\sigma_{\text{SPS,OUT}}$, or $\perp$

1. (a) Compute $r_{\text{SPS},E} = \mathcal{F}(K_{\text{SPS},E}, (h, \ell_{\text{INP}}))$, $(\text{SK}_{\text{SPS},E}, \text{VK}_{\text{SPS},E}, \text{VK}_{\text{SPS-REJ},E}) = \mathsf{SPS.Setup}(1^\lambda; r_{\text{SPS},E})$.
   (b) Compute $r_{\text{SPS},F} = \mathcal{F}(K_{\text{SPS},F}\{(h^*,\ell^*)\}, (h, \ell_{\text{INP}}))$, $(\text{SK}_{\text{SPS},F}, \text{VK}_{\text{SPS},F}, \text{VK}_{\text{SPS-REJ},F}) = \mathsf{SPS.Setup}(1^\lambda; r_{\text{SPS},F})$.
   (c) Set $m = (v, \text{ST}, w, 0)$ and α='-'.
   (d) If $\mathsf{SPS.Verify}(\text{VK}_{\text{SPS},E}, m, \sigma_{\text{SPS,IN}}) = 1$, set α ='E'.
   (e) If $[\alpha$ ='-'$] \wedge [(\ell_{\text{INP}} \neq \ell^*) \vee (h \neq h^*)]$, output $\perp$.
   Else if $[\alpha$ ='-'$] \wedge \underline{[\mathsf{SPS.Verify}(\text{VK}_H, m, \sigma_{\text{SPS,IN}}) = 0]}$, output $\perp$.
   Else if $[\alpha$ ='-'$] \wedge \underline{[\mathsf{SPS.Verify}(\text{VK}_H, m, \sigma_{\text{SPS,IN}}) = 1]}$, set α ='F'.
   (f) If α ='-', output $\perp$.
2. (a) Compute $r_{\text{SPS},A} = \mathcal{F}(K_{\text{SPS},A}, (h, \ell_{\text{INP}}, 0))$, $(\text{SK}_{\text{SPS},A}, \text{VK}_{\text{SPS},A}, \text{VK}_{\text{SPS-REJ},A}) = \mathsf{SPS.Setup}(1^\lambda; r_{\text{SPS},A})$.
   (b) Compute $r_{\text{SPS},B} = \mathcal{F}(K_{\text{SPS},B}, (h, \ell_{\text{INP}}, 0))$, $(\text{SK}_{\text{SPS},B}, \text{VK}_{\text{SPS},B}, \text{VK}_{\text{SPS-REJ},B}) = \mathsf{SPS.Setup}(1^\lambda; r_{\text{DSPS},B})$.
   (c) If $[(h, \ell_{\text{INP}}) = (h^*, \ell^*)] \wedge [\alpha$ ='F'$]$, output $\sigma_{\text{SPS,OUT}} = \mathsf{SPS.Sign}(\text{SK}_{\text{SPS},B}, m)$.
   Else, output $\sigma_{\text{SPS,OUT}} = \mathsf{SPS.Sign}(\text{SK}_{\text{SPS},A}, m)$.

Fig. 26 $\mathsf{Change\text{-}SPS.Prog}^{(0,2)}$

$\underbrace{\text{VK}^{(\nu,\ell^*)}_{\text{SPS},H}, \text{VK}^{(\nu,\ell^*)}_{\text{SPS-REJ},H}) \xleftarrow{\$} \mathsf{SPS.Setup}(1^\lambda)}$, and gives $\mathcal{A}$ the signing key

$$\text{SK}_{\text{ABS}}\{M^{(\nu)}\} = \left(\begin{array}{l} \text{HK}, \text{PP}^{(\nu)}_{\text{ACC}}, w^{(\nu)}_0, \text{STORE}^{(\nu)}_0, \text{PP}^{(\nu)}_{\text{ITR}}, v^{(\nu)}_0, \\ \mathcal{IO}(\mathsf{Init\text{-}SPS.Prog}[q^{(\nu)}_0, w^{(\nu)}_0, v^{(\nu)}_0, K^{(\nu)}_{\text{SPS},E}]), \\ \mathcal{IO}(\mathsf{Accumulate.Prog}^{(0,2)}[n_{\text{SSB-BLK}} = 2^\lambda, \text{HK}, \text{PP}^{(\nu)}_{\text{ACC}}, \text{PP}^{(\nu)}_{\text{ITR}}, K^{(\nu)}_{\text{SPS},E}, \\ \quad K^{(\nu)}_{\text{SPS},F}\{(h^*,\ell^*)\}, \text{VK}^{(\nu,\ell^*)}_{\text{SPS-REJ},H}, h^*, \ell^*]), \\ \mathcal{IO}(\mathsf{Change\text{-}SPS.Prog}^{(0,2)}[K^{(\nu)}_{\text{SPS},A}, K^{(\nu)}_{\text{SPS},B}, K^{(\nu)}_{\text{SPS},E}, K^{(\nu)}_{\text{SPS},F}\{(h^*,\ell^*)\}, \\ \quad \text{VK}^{(\nu,\ell^*)}_{\text{SPS-REJ},H}, h^*, \ell^*]), \\ \mathcal{IO}(\mathsf{Constrained\text{-}Key.Prog}^{(1)}_{\text{ABS}}[M^{(\nu)}, T = 2^\lambda, t^{*(\nu)}, \text{PP}^{(\nu)}_{\text{ACC}}, \text{PP}^{(\nu)}_{\text{ITR}}, K, \\ \quad K^{(\nu)}_1, \ldots, K^{(\nu)}_\lambda, K^{(\nu)}_{\text{SPS},A}, K^{(\nu)}_{\text{SPS},B}, h^*, \ell^*]) \end{array}\right).$$

$\mathsf{Hyb}_{0,\nu-1,1,4}$: In this experiment, to answer the ν^{th} signing key query of $\mathcal{A}$ corresponding to TM $M^{(\nu)} \in \mathbb{M}_\lambda$ with $M^{(\nu)}(x^*) = 0$, $\mathcal{B}$ creates all the components as in $\mathsf{Hyb}_{0,\nu-1,1,3}$, however, it provides $\mathcal{A}$ with the signing key

$$\text{SK}_{\text{ABS}}\{M^{(\nu)}\} = \left(\begin{array}{l} \text{HK}, \text{PP}^{(\nu)}_{\text{ACC}}, w^{(\nu)}_0, \text{STORE}^{(\nu)}_0, \text{PP}^{(\nu)}_{\text{ITR}}, v^{(\nu)}_0, \\ \mathcal{IO}(\mathsf{Init\text{-}SPS.Prog}[q^{(\nu)}_0, w^{(\nu)}_0, v^{(\nu)}_0, K^{(\nu)}_{\text{SPS},E}]), \\ \mathcal{IO}(\mathsf{Accumulate.Prog}^{(0,2)}[n_{\text{SSB-BLK}} = 2^\lambda, \text{HK}, \text{PP}^{(\nu)}_{\text{ACC}}, \text{PP}^{(\nu)}_{\text{ITR}}, K^{(\nu)}_{\text{SPS},E}, \\ \quad K^{(\nu)}_{\text{SPS},F}\{(h^*,\ell^*)\}, \text{VK}^{(\nu,\ell^*)}_{\text{SPS},H}, h^*, \ell^*]), \\ \mathcal{IO}(\mathsf{Change\text{-}SPS.Prog}^{(0,2)}[K^{(\nu)}_{\text{SPS},A}, K^{(\nu)}_{\text{SPS},B}, K^{(\nu)}_{\text{SPS},E}, K^{(\nu)}_{\text{SPS},F}\{(h^*,\ell^*)\}, \\ \quad \underline{\text{VK}^{(\nu,\ell^*)}_{\text{SPS},H}}, h^*, \ell^*]), \\ \mathcal{IO}(\underline{\mathsf{Constrained\text{-}Key.Prog}}^{(1)}_{\text{ABS}}[M^{(\nu)}, T = 2^\lambda, t^{*(\nu)}, \text{PP}^{(\nu)}_{\text{ACC}}, \text{PP}^{(\nu)}_{\text{ITR}}, K, \\ \quad K^{(\nu)}_1, \ldots, K^{(\nu)}_\lambda, K^{(\nu)}_{\text{SPS},A}, K^{(\nu)}_{\text{SPS},B}, h^*, \ell^*]) \end{array}\right).$$

The rest of the experiment is the same as $\mathsf{Hyb}_{0,\nu-1,1,3}$.

$\mathsf{Hyb}_{0,\nu-1,1,5}$: In this experiment, in response to the ν^{th} signing key query of $\mathcal{A}$ corresponding to TM $M^{(\nu)} \in \mathbb{M}_\lambda$ with $M^{(\nu)}(x^*) = 0$, $\mathcal{B}$ forms all the components as in $\mathsf{Hyb}_{0,\nu-1,1,4}$ except that it computes $r^{(\nu,\ell^*)}_{\text{SPS},H} = \mathcal{F}(K^{(\nu)}_{\text{SPS},F}, (h^*, \ell^*))$, $(\text{SK}^{(\nu,\ell^*)}_{\text{SPS},H}, \text{VK}^{(\nu,\ell^*)}_{\text{SPS},H}, \text{VK}^{(\nu,\ell^*)}_{\text{SPS-REJ},H}) = \mathsf{SPS.Setup}(1^\lambda;\ r^{(\nu,\ell^*)}_{\text{SPS},H})$, and hands $\mathcal{A}$ the signing key

$$\text{SK}_{\text{ABS}}\{M^{(\nu)}\} = \left(\begin{array}{l} \text{HK}, \text{PP}^{(\nu)}_{\text{ACC}}, w^{(\nu)}_0, \text{STORE}^{(\nu)}_0, \text{PP}^{(\nu)}_{\text{ITR}}, v^{(\nu)}_0, \\ \mathcal{IO}(\mathsf{Init\text{-}SPS.Prog}[q^{(\nu)}_0, w^{(\nu)}_0, v^{(\nu)}_0, K^{(\nu)}_{\text{SPS},E}]), \\ \mathcal{IO}(\mathsf{Accumulate.Prog}^{(0,2)}[n_{\text{SSB-BLK}} = 2^\lambda, \text{HK}, \text{PP}^{(\nu)}_{\text{ACC}}, \text{PP}^{(\nu)}_{\text{ITR}}, K^{(\nu)}_{\text{SPS},E}, \\ \qquad K^{(\nu)}_{\text{SPS},F}\{(h^*,\ell^*)\}, \text{VK}^{(\nu,\ell^*)}_{\text{SPS},H}, h^*, \ell^*]), \\ \mathcal{IO}(\mathsf{Change\text{-}SPS.Prog}^{(0,2)}[K^{(\nu)}_{\text{SPS},A}, K^{(\nu)}_{\text{SPS},B}, K^{(\nu)}_{\text{SPS},E}, K^{(\nu)}_{\text{SPS},F}\{(h^*,\ell^*)\}, \\ \qquad \text{VK}^{(\nu,\ell^*)}_{\text{SPS},H}, h^*, \ell^*]), \\ \mathcal{IO}(\mathsf{Constrained\text{-}Key.Prog}^{(1)}_{\text{ABS}}[M^{(\nu)}, T = 2^\lambda, t^{*(\nu)}, \text{PP}^{(\nu)}_{\text{ACC}}, \text{PP}^{(\nu)}_{\text{ITR}}, K, \\ \qquad K^{(\nu)}_1, \ldots, K^{(\nu)}_\lambda, K^{(\nu)}_{\text{SPS},A}, K^{(\nu)}_{\text{SPS},B}, h^*, \ell^*]) \end{array}\right).$$

The rest of the experiment is analogous to $\mathsf{Hyb}_{0,\nu-1,1,4}$.

$\mathsf{Hyb}_{0,\nu-1,1,6}$: This experiment corresponds to $\mathsf{Hyb}_{0,\nu-1,2}$.

Analysis

Let $\mathsf{Adv}^{(0,\nu-1,1,\vartheta)}_{\mathcal{A}}(\lambda)$ represents the advantage of $\mathcal{A}$, i.e., the absolute difference between $1/2$ and $\mathcal{A}$'s probability of correctly guessing the random bit selected by the challenger $\mathcal{B}$, in $\mathsf{Hyb}_{0,\nu-1,1,\vartheta}$, for $\vartheta \in [0,6]$. By definition, $\mathsf{Adv}^{(0,\nu-1,1)}_{\mathcal{A}}(\lambda) \equiv \mathsf{Adv}^{(0,\nu-1,1,0)}_{\mathcal{A}}(\lambda)$ and $\mathsf{Adv}^{(0,\nu-1,2)}_{\mathcal{A}}(\lambda) \equiv \mathsf{Adv}^{(0,\nu-1,1,6)}_{\mathcal{A}}(\lambda)$. Then, we have

$$|\mathsf{Adv}^{(0,\nu-1,1)}_{\mathcal{A}}(\lambda) - \mathsf{Adv}^{(0,\nu-1,2)}_{\mathcal{A}}(\lambda)| \leq \sum_{\vartheta=1}^{6} |\mathsf{Adv}^{(0,\nu-1,1,\vartheta-1)}_{\mathcal{A}}(\lambda) - \mathsf{Adv}^{(0,\nu-1,1,\vartheta)}_{\mathcal{A}}(\lambda)|. \tag{C.2}$$

Claims C.2–C.7 below will demonstrate that the RHS of Eq. (C.2) is negligible and thus Lemma C.2 follows.

Claim C.2 *Assuming $\mathcal{IO}$ is a secure indistinguishability obfuscator for $\mathsf{P/poly}$, for any PPT adversary $\mathcal{A}$, for any security parameter λ, $|\mathsf{Adv}^{(0,\nu-1,1,0)}_{\mathcal{A}}(\lambda) - \mathsf{Adv}^{(0,\nu-1,1,1)}_{\mathcal{A}}(\lambda)| \leq \mathsf{negl}(\lambda)$ for some negligible function negl.*

Proof The difference between $\mathsf{Hyb}_{0,\nu-1,1,0}$ and $\mathsf{Hyb}_{0,\nu-1,1,1}$ is the following: In $\mathsf{Hyb}_{0,\nu-1,1,0}$, $\mathcal{B}$ includes the programs $\mathcal{IO}(P_0)$ and $\mathcal{IO}(P'_0)$ within the ν^{th} signing key returned to $\mathcal{A}$, while in $\mathsf{Hyb}_{0,\nu-1,1,1}$, $\mathcal{B}$ includes the programs $\mathcal{IO}(P_1)$ and $\mathcal{IO}(P'_1)$ instead, where

- $P_0 = \mathsf{Accumulate.Prog}[n_{\text{SSB-BLK}} = 2^\lambda, \text{HK}, \text{PP}^{(\nu)}_{\text{ACC}}, \text{PP}^{(\nu)}_{\text{ITR}}, K^{(\nu)}_{\text{SPS},E}]$ (Fig. **3**),
- $P'_0 = \mathsf{Change\text{-}SPS.Prog}[K^{(\nu)}_{\text{SPS},A}, K^{(\nu)}_{\text{SPS},E}]$ (Fig. **4**),
- $P_1 = \mathsf{Accumulate.Prog}^{(0,1)}[n_{\text{SSB-BLK}} = 2^\lambda, \text{HK}, \text{PP}^{(\nu)}_{\text{ACC}}, \text{PP}^{(\nu)}_{\text{ITR}}, K^{(\nu)}_{\text{SPS},E}, K^{(\nu)}_{\text{SPS},F}, h^*, \ell^*]$ (Fig. **23**),
- $P'_1 = \mathsf{Change\text{-}SPS.Prog}^{(0,1)}[K^{(\nu)}_{\text{SPS},A}, K^{(\nu)}_{\text{SPS},B}, K^{(\nu)}_{\text{SPS},E}, K^{(\nu)}_{\text{SPS},F}, h^*, \ell^*]$ (Fig. **24**).

Now, observe that the programs P_0 and P_1 clearly have identical outputs for inputs corresponding to $(h, i) \neq (h^*, \ell^*)$. Also, by the correctness [Property (vii)] of splittable signature scheme SPS, both the programs output $\perp$ in case $\mathsf{SPS.Verify}$ $(\mathrm{VK}_{\mathrm{SPS},E}, m_{\mathrm{IN}}, \sigma_{\mathrm{SPS,IN}}) = 0$ for inputs corresponding to (h^*, ℓ^*). Thus the programs P_0 and P_1 are functionally equivalent. A similar argument justifies the functional equivalence of the programs P'_0 and P'_1.

Thus, by the security of $\mathcal{IO}$, Claim C.2 follows. Of course, we need to consider a sequence of hybrid experiments to arrive at the result where in each hybrid experiment we change the programs one at a time. □

Claim C.3 *Assuming $\mathcal{IO}$ is a secure indistinguishability obfuscator for* $\mathsf{P/poly}$ *and $\mathcal{F}$ satisfy the correctness under puncturing property, for any* PPT *adversary $\mathcal{A}$, for any security parameter λ, $|\mathsf{Adv}^{(0,\nu-1,1,1)}_{\mathcal{A}}(\lambda) - \mathsf{Adv}^{(0,\nu-1,1,2)}_{\mathcal{A}}(\lambda)| \leq \mathsf{negl}(\lambda)$ for some negligible function* negl.

Proof The difference between $\mathsf{Hyb}_{0,\nu-1,1,1}$ and $\mathsf{Hyb}_{0,\nu-1,1,2}$ is the following: In $\mathsf{Hyb}_{0,\nu-1,1,1}$, $\mathcal{B}$ includes the programs $\mathcal{IO}(P_0)$ and $\mathcal{IO}(P'_0)$ within the ν^{th} signing key returned to $\mathcal{A}$, while in $\mathsf{Hyb}_{0,\nu-1,1,2}$, $\mathcal{B}$ includes the programs $\mathcal{IO}(P_1)$ and $\mathcal{IO}(P'_1)$ instead, where

- $P_0 = \mathsf{Accumulate.Prog}^{(0,1)}[n_{\text{SSB-BLK}} = 2^\lambda, \mathrm{HK}, \mathrm{PP}^{(\nu)}_{\mathrm{ACC}}, \mathrm{PP}^{(\nu)}_{\mathrm{ITR}}, K^{(\nu)}_{\mathrm{SPS},E}, K^{(\nu)}_{\mathrm{SPS},F}, h^*, \ell^*]$ (Fig. **23**),
- $P'_0 = \mathsf{Change\text{-}SPS.Prog}^{(0,1)}[K^{(\nu)}_{\mathrm{SPS},A}, K^{(\nu)}_{\mathrm{SPS},B}, K^{(\nu)}_{\mathrm{SPS},E}, K^{(\nu)}_{\mathrm{SPS},F}, h^*, \ell^*]$ (Fig. **24**),
- $P_1 = \mathsf{Accumulate.Prog}^{(0,2)}[n_{\text{SSB-BLK}} = 2^\lambda, \mathrm{HK}, \mathrm{PP}^{(\nu)}_{\mathrm{ACC}}, \mathrm{PP}^{(\nu)}_{\mathrm{ITR}}, K^{(\nu)}_{\mathrm{SPS},E}, K^{(\nu)}_{\mathrm{SPS},F}\{(h^*, \ell^*)\}, \mathrm{VK}^{(\nu,\ell^*)}_{\text{SPS-REJ},H}, h^*, \ell^*]$ (Fig. **25**),
- $P'_1 = \mathsf{Change\text{-}SPS.Prog}^{(0,2)}[K^{(\nu)}_{\mathrm{SPS},A}, K^{(\nu)}_{\mathrm{SPS},B}, K^{(\nu)}_{\mathrm{SPS},E}, K^{(\nu)}_{\mathrm{SPS},F}\{(h^*, \ell^*)\}, \mathrm{VK}^{(\nu,\ell^*)}_{\text{SPS-REJ},H}, h^*, \ell^*]$ (Fig. **26**).

Now, by the correctness under puncturing property of the PPRF $\mathcal{F}$, both the programs P_0 and P_1 have identical outputs on inputs corresponding to $(h, i) \neq (h^*, \ell^*)$. For inputs corresponding to (h^*, ℓ^*), P_1 uses the hardwired verification key $\mathrm{VK}^{(\nu,\ell^*)}_{\text{SPS-REJ},H}$, where in $\mathsf{Hyb}_{0,\nu-1,1,2}$, $\mathrm{VK}^{(\nu,\ell^*)}_{\text{SPS-REJ},H}$ is computed as $(\mathrm{SK}^{(\nu,\ell^*)}_{\mathrm{SPS},H}, \mathrm{VK}^{(\nu,\ell^*)}_{\mathrm{SPS},H}, \mathrm{VK}^{(\nu,\ell^*)}_{\text{SPS-REJ},H}) = \mathsf{SPS.Setup}(1^\lambda; r^{(\nu,\ell^*)}_{\mathrm{SPS},H})$ and $r^{(\nu,\ell^*)}_{\mathrm{SPS},H} = \mathcal{F}(K^{(\nu)}_{\mathrm{SPS},F}, (h^*, \ell^*))$. Observe that these values are exactly the same as those used by the program P_0 for inputs corresponding to (h^*, ℓ^*). Thus, both programs have identical outputs for inputs corresponding to (h^*, ℓ^*) as well. Hence, the two programs are functionally equivalent. A similar argument will justify that the programs P'_0 and P'_1 are functionally equivalent.

Therefore, by the security of $\mathcal{IO}$, Claim C.3 follows, considering a sequence of hybrid experiments where in each hybrid experiment we change the programs one at a time. □

Claim C.4 *Assuming $\mathcal{F}$ is a secure puncturable pseudorandom function, for any* PPT *adversary $\mathcal{A}$, for any security parameter λ, $|\mathsf{Adv}^{(0,\nu-1,1,2)}_{\mathcal{A}}(\lambda) - \mathsf{Adv}^{(0,\nu-1,1,3)}_{\mathcal{A}}(\lambda)| \leq \mathsf{negl}(\lambda)$ for some negligible function* negl.

Proof Suppose there exists a PPT adversary $\mathcal{A}$ for which $|\mathsf{Adv}^{(0,\nu-1,1,2)}_{\mathcal{A}}(\lambda) - \mathsf{Adv}^{(0,\nu-1,1,3)}_{\mathcal{A}}(\lambda)|$ is non-negligible. We construct a PPT adversary $\mathcal{B}$ that breaks the selective pseudorandomness of the PPRF $\mathcal{F}$ using $\mathcal{A}$ as a sub-routine. The description of $\mathcal{B}$ follows:

- $\mathcal{B}$ initializes $\mathcal{A}$ on input 1^λ and receives a challenge signing attribute string $x^* = x_0^* \ldots x_{\ell^*-1}^* \in \mathcal{U}_{\text{ABS}}$ with $|x^*| = \ell^*$ from $\mathcal{A}$.
- Upon receiving x^*, $\mathcal{B}$ proceeds as follows:
  1. $\mathcal{B}$ generates $\text{HK} \xleftarrow{\$} \mathsf{SSB.Gen}(1^\lambda, n_{\text{SSB-BLK}} = 2^\lambda, i^* = 0)$ and computes $h^* = \mathcal{H}_{\text{HK}}(x^*)$.
  2. Then, $\mathcal{B}$ selects a PPRF key $K \xleftarrow{\$} \mathcal{F}.\mathsf{Setup}(1^\lambda)$.
  3. After that, $\mathcal{B}$ computes $r^*_{\text{SIG}} = \mathcal{F}(K, (h^*, \ell^*))$ followed by $(\widehat{\text{SK}}^*_{\text{SIG}}, \widehat{\text{VK}}^*_{\text{SIG}}) = \mathsf{SIG.Setup}(1^\lambda;\ r^*_{\text{SIG}})$.
  4. $\mathcal{B}$ returns the public parameters $\text{PP}_{\text{ABS}} = (\text{HK}, \mathcal{IO}(\mathsf{Verify.Prog}_{\text{ABS}}[K]))$ to $\mathcal{A}$.
- For $\eta \in [\hat{q}_{\text{KEY}}]$, in response to the η^{th} signing key query of $\mathcal{A}$ corresponding to TM $M^{(\eta)} \in \mathbb{M}_\lambda$ with $M^{(\eta)}(x^*) = 0$, if $\eta \neq \nu$, then $\mathcal{B}$ proceeds exactly as in $\mathsf{Hyb}_{0,\nu-1,1,2}$, while if $\eta = \nu$, then $\mathcal{B}$ proceeds as follows:
  1. $\mathcal{B}$ first selects PPRF keys $K_1^{(\nu)}, \ldots, K_\lambda^{(\nu)}$, $K_{\text{SPS},A}^{(\nu)}$, $K_{\text{SPS},B}^{(\nu)}$, $K_{\text{SPS},E}^{(\nu)} \xleftarrow{\$} \mathcal{F}.\mathsf{Setup}(1^\lambda)$.
  2. Next, it generates $(\text{PP}_{\text{ACC}}^{(\nu)}, w_0^{(\nu)}, \text{STORE}_0^{(\nu)}) \xleftarrow{\$} \mathsf{ACC.Setup}(1^\lambda, n_{\text{ACC-BLK}} = 2^\lambda)$ and $(\text{PP}_{\text{ITR}}^{(\nu)}, v_0^{(\nu)}) \xleftarrow{\$} \mathsf{ITR.Setup}(1^\lambda, n_{\text{ITR}} = 2^\lambda)$.
  3. Then, $\mathcal{B}$ sends (h^*, ℓ^*) as the challenge input to its PPRF selective pseudorandomness challenger $\mathcal{C}$ and receives back a punctured PPRF key $K^*\{(h^*, \ell^*)\}$ and a value $r^* \in \mathcal{Y}_{\text{PPRF}}$, where either $r^* = \mathcal{F}(K^*, (h^*, \ell^*))$ or $r^* \xleftarrow{\$} \mathcal{Y}_{\text{PPRF}}$. $\mathcal{B}$ will *implicitly* view the key K^* as the key $K_{\text{SPS},F}^{(\nu)}$.
  4. $\mathcal{B}$ generates $(\text{SK}_{\text{SPS},H}^{(\nu,\ell^*)}, \text{VK}_{\text{SPS},H}^{(\nu,\ell^*)}, \text{VK}_{\text{SPS-REJ},H}^{(\nu,\ell^*)}) = \mathsf{SPS.Setup}(1^\lambda;\ r^*)$.
  5. $\mathcal{B}$ gives $\mathcal{A}$ the signing key
$$\text{SK}_{\text{ABS}}\{M^{(\nu)}\} = \left(\begin{array}{l} \text{HK}, \text{PP}_{\text{ACC}}^{(\nu)}, w_0^{(\nu)}, \text{STORE}_0^{(\nu)}, \text{PP}_{\text{ITR}}^{(\nu)}, v_0^{(\nu)}, \\ \mathcal{IO}(\mathsf{Init\text{-}SPS.Prog}[q_0^{(\nu)}, w_0^{(\nu)}, v_0^{(\nu)}, K_{\text{SPS},E}^{(\nu)}]), \\ \mathcal{IO}(\mathsf{Accumulate.Prog}^{(0,2)}[n_{\text{SSB-BLK}} = 2^\lambda, \text{HK}, \text{PP}_{\text{ACC}}^{(\nu)}, \text{PP}_{\text{ITR}}^{(\nu)}, K_{\text{SPS},E}^{(\nu)}, \\ \qquad K^*\{(h^*, \ell^*)\}, \text{VK}_{\text{SPS-REJ},H}^{(\nu,\ell^*)}, h^*, \ell^*]), \\ \mathcal{IO}(\mathsf{Change\text{-}SPS.Prog}^{(0,2)}[K_{\text{SPS},A}^{(\nu)}, K_{\text{SPS},B}^{(\nu)}, K_{\text{SPS},E}^{(\nu)}, K^*\{(h^*, \ell^*)\}, \\ \qquad \text{VK}_{\text{SPS-REJ},H}^{(\nu,\ell^*)}, h^*, \ell^*]), \\ \mathcal{IO}(\mathsf{Constrained\text{-}Key.Prog}_{\text{ABS}}^{(1)}[M^{(\nu)}, T = 2^\lambda, t^{*(\nu)}, \text{PP}_{\text{ACC}}^{(\nu)}, \text{PP}_{\text{ITR}}^{(\nu)}, K, \\ \qquad K_1^{(\nu)}, \ldots, K_\lambda^{(\nu)}, K_{\text{SPS},A}^{(\nu)}, K_{\text{SPS},B}^{(\nu)}, h^*, \ell^*]) \end{array} \right).$$
- For $\theta \in [\hat{q}_{\text{SIGN}}]$, in reply to the θ^{th} signature query of $\mathcal{A}$ on message $\mathsf{msg}^{(\theta)} \in \mathcal{M}_{\text{ABS}}$ under attribute string x^*, $\mathcal{B}$ computes $\sigma_{\text{SIG}}^{(\theta)} \xleftarrow{\$} \mathsf{SIG.Sign}(\widehat{\text{SK}}^*_{\text{SIG}}, \mathsf{msg}^{(\theta)})$ and provides $\mathcal{A}$ with $\sigma_{\text{ABS}}^{(\theta)} = (\widehat{\text{VK}}^*_{\text{SIG}}, \sigma_{\text{SIG}}^{(\theta)})$.
- Finally, $\mathcal{A}$ output a signature σ^*_{ABS} on some message msg^* under attribute string x^*. $\mathcal{B}$ outputs $\hat{b}' = 1$ as its guess bit in its PPRF selective pseudorandomness experiment if $\mathcal{A}$ wins, i.e., if $\mathsf{ABS.Verify}(\text{PP}_{\text{ABS}}, x^*, \mathsf{msg}^*, \sigma^*_{\text{ABS}}) = 1$ and $\mathsf{msg}^* \neq \mathsf{msg}^{(\theta)}$ for any $\theta \in [\hat{q}_{\text{SIGN}}]$. Otherwise, $\mathcal{B}$ outputs $\hat{b}' = 0$ in its PPRF selective pseudorandomness experiment.

Note that if $r^* = \mathcal{F}(K^*, (h^*, \ell^*))$, then $\mathcal{B}$ perfectly simulates $\mathsf{Hyb}_{0,\nu-1,1,2}$. On the other hand, if $r^* \xleftarrow{\$} \mathcal{Y}_{\text{PPRF}}$, then $\mathcal{B}$ perfectly simulates $\mathsf{Hyb}_{0,\nu-1,1,3}$. This completes the proof of Claim C.4. □

Claim C.5 *Assuming* SPS *is a splittable signature scheme satisfying* '$\text{VK}_{\text{SPS-REJ}}$ *indistinguishability', for any* PPT *adversary* $\mathcal{A}$*, for any security parameter* λ*,* $|\mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,1,3)}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,1,4)}(\lambda)| \leq \mathsf{negl}(\lambda)$ *for some negligible function* negl.

Proof Suppose there exists a PPT adversary $\mathcal{A}$ for which $|\mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,1,3)}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,1,4)}(\lambda)|$ is non-negligible. Below we construct a PPT adversary $\mathcal{B}$ that breaks the $\text{VK}_{\text{SPS-REJ}}$ indistinguishability of SPS using $\mathcal{A}$ as a sub-routine.

- $\mathcal{B}$ receives a verification key VK of the splittable signature scheme SPS from its $\text{VK}_{\text{SPS-REJ}}$ indistinguishability challenger $\mathcal{C}$, where VK is either a proper verification key VK_{SPS} or a reject verification key $\text{VK}_{\text{SPS-REJ}}$. Then, $\mathcal{B}$ initializes $\mathcal{A}$ on input 1^λ and receives a challenge signing attribute string $x^* = x_0^* \ldots x_{\ell^*-1}^* \in \mathcal{U}_{\text{ABS}}$ with $|x^*| = \ell^*$ from $\mathcal{A}$.
- Upon receiving x^*, $\mathcal{B}$ proceeds as follows:
  1. $\mathcal{B}$ generates $\text{HK} \xleftarrow{\$} \mathsf{SSB.Gen}(1^\lambda, n_{\text{SSB-BLK}} = 2^\lambda, i^* = 0)$ and computes $h^* = \mathcal{H}_{\text{HK}}(x^*)$.
  2. Then, $\mathcal{B}$ selects a PPRF key $K \xleftarrow{\$} \mathcal{F}.\mathsf{Setup}(1^\lambda)$.
  3. After that, $\mathcal{B}$ computes $r_{\text{SIG}}^* = \mathcal{F}(K, (h^*, \ell^*))$ followed by $(\widehat{\text{SK}}_{\text{SIG}}^*, \widehat{\text{VK}}_{\text{SIG}}^*) = \mathsf{SIG.Setup}(1^\lambda;\ r_{\text{SIG}}^*)$.
  4. $\mathcal{B}$ returns the public parameters $\text{PP}_{\text{ABS}} = (\text{HK}, \mathcal{IO}(\mathsf{Verify.Prog}_{\text{ABS}}[K]))$ to $\mathcal{A}$.
- For $\eta \in [\hat{q}_{\text{KEY}}]$, in response to the η^{th} signing key query of $\mathcal{A}$ corresponding to TM $M^{(\eta)} \in \mathbb{M}_\lambda$ with $M^{(\eta)}(x^*) = 0$, if $\eta \neq \nu$, then $\mathcal{B}$ proceeds exactly as in $\mathsf{Hyb}_{0,\nu-1,1,3}$, while if $\eta = \nu$, then $\mathcal{B}$ proceeds as follows:
  1. $\mathcal{B}$ first selects PPRF keys $K_1^{(\nu)}, \ldots, K_\lambda^{(\nu)}, K_{\text{SPS},A}^{(\nu)}, K_{\text{SPS},B}^{(\nu)}, K_{\text{SPS},E}^{(\nu)}, K_{\text{SPS},F}^{(\nu)} \xleftarrow{\$} \mathcal{F}.\mathsf{Setup}(1^\lambda)$.
  2. Next, it generates $(\text{PP}_{\text{ACC}}^{(\nu)}, w_0^{(\nu)}, \text{STORE}_0^{(\nu)}) \xleftarrow{\$} \mathsf{ACC.Setup}(1^\lambda, n_{\text{ACC-BLK}} = 2^\lambda)$ and $(\text{PP}_{\text{ITR}}^{(\nu)}, v_0^{(\nu)}) \xleftarrow{\$} \mathsf{ITR.Setup}(1^\lambda, n_{\text{ITR}} = 2^\lambda)$.
  3. $\mathcal{B}$ creates the punctured PPRF key $K_{\text{SPS},F}^{(\nu)}\{(h^*, \ell^*)\} \xleftarrow{\$} \mathcal{F}.\mathsf{Puncture}(K_{\text{SPS},F}, (h^*, \ell^*))$.
  4. $\mathcal{B}$ gives $\mathcal{A}$ the signing key
$$\text{SK}_{\text{ABS}}\{M^{(\nu)}\} = \left(\begin{array}{l} \text{HK}, \text{PP}_{\text{ACC}}^{(\nu)}, w_0^{(\nu)}, \text{STORE}_0^{(\nu)}, \text{PP}_{\text{ITR}}^{(\nu)}, v_0^{(\nu)}, \\ \mathcal{IO}(\mathsf{Init\text{-}SPS.Prog}[q_0^{(\nu)}, w_0^{(\nu)}, v_0^{(\nu)}, K_{\text{SPS},E}^{(\nu)}]), \\ \mathcal{IO}(\mathsf{Accumulate.Prog}^{(0,2)}[n_{\text{SSB-BLK}} = 2^\lambda, \text{HK}, \text{PP}_{\text{ACC}}^{(\nu)}, \text{PP}_{\text{ITR}}^{(\nu)}, K_{\text{SPS},E}^{(\nu)}, \\ \qquad K_{\text{SPS},F}^{(\nu)}\{(h^*, \ell^*)\}, \text{VK}, h^*, \ell^*]), \\ \mathcal{IO}(\mathsf{Change\text{-}SPS.Prog}^{(0,2)}[K_{\text{SPS},A}^{(\nu)}, K_{\text{SPS},B}^{(\nu)}, K_{\text{SPS},E}^{(\nu)}, K_{\text{SPS},F}^{(\nu)}\{(h^*, \ell^*)\}, \\ \qquad \text{VK}, h^*, \ell^*]), \\ \mathcal{IO}(\mathsf{Constrained\text{-}Key.Prog}_{\text{ABS}}^{(1)}[M^{(\nu)}, T = 2^\lambda, t^{*(\nu)}, \text{PP}_{\text{ACC}}^{(\nu)}, \text{PP}_{\text{ITR}}^{(\nu)}, K, \\ \qquad K_1^{(\nu)}, \ldots, K_\lambda^{(\nu)}, K_{\text{SPS},A}^{(\nu)}, K_{\text{SPS},B}^{(\nu)}, h^*, \ell^*]) \end{array}\right).$$
- For $\theta \in [\hat{q}_{\text{SIGN}}]$, in reply to the θ^{th} signature query of $\mathcal{A}$ on message $\mathsf{msg}^{(\theta)} \in \mathcal{M}_{\text{ABS}}$ under attribute string x^*, $\mathcal{B}$ computes $\sigma_{\text{SIG}}^{(\theta)} \xleftarrow{\$} \mathsf{SIG.Sign}(\widehat{\text{SK}}_{\text{SIG}}^*, \mathsf{msg}^{(\theta)})$ and provides $\mathcal{A}$ with $\sigma_{\text{ABS}}^{(\theta)} = (\widehat{\text{VK}}_{\text{SIG}}^*, \sigma_{\text{SIG}}^{(\theta)})$.
- Finally, $\mathcal{A}$ output a signature σ_{ABS}^* on some message msg^* under attribute string x^*. $\mathcal{B}$ outputs $\hat{b}' = 1$ as its guess bit in its $\text{VK}_{\text{SPS-REJ}}$ indistinguishability experiment if $\mathcal{A}$ wins, i.e., if $\mathsf{ABS.Verify}(\text{PP}_{\text{ABS}}, x^*, \mathsf{msg}^*, \sigma_{\text{ABS}}^*) = 1$ and $\mathsf{msg}^* \neq \mathsf{msg}^{(\theta)}$ for

any $\theta \in [\hat{q}_{\text{SIGN}}]$. Otherwise, $\mathcal{B}$ outputs $\hat{b}' = 0$ in its $\text{VK}_{\text{SPS-REJ}}$ indistinguishability experiment.

Notice that if $\text{VK} = \text{VK}_{\text{SPS-REJ}}$, then $\mathcal{B}$ perfectly simulates $\mathsf{Hyb}_{0,\nu-1,1,3}$. On the other hand, if $\text{VK} = \text{VK}_{\text{SPS}}$, then $\mathcal{B}$ perfectly simulates $\mathsf{Hyb}_{0,\nu-1,1,4}$. This completes the proof of Claim C.5. □

Claim C.6 *Assuming $\mathcal{F}$ is a secure puncturable pseudorandom function, for any* PPT *adversary $\mathcal{A}$, for any security parameter λ, $|\mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,1,4)}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,1,5)}(\lambda)| \leq \mathsf{negl}(\lambda)$ for some negligible function* negl.

Proof The proof of Claim C.6 is similar to that of Claim C.4 with some appropriate changes which can be readily identified. □

Claim C.7 *Assuming $\mathcal{IO}$ is a secure indistinguishability obfuscator for* P/poly *and $\mathcal{F}$satisfies the correctness under puncturing property, for any* PPT *adversary $\mathcal{A}$, for any security parameter λ, $|\mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,1,5)}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,1,6)}(\lambda)| \leq \mathsf{negl}(\lambda)$ for some negligible function* negl.

Proof The proof of Claim C.7 is analogous to that of Claim C.3 with some appropriate changes that are easy to determine. □

□

Lemma C.3 *Assuming $\mathcal{IO}$ is a secure indistinguishability obfuscator for* P/poly*, $\mathcal{F}$ is a secure puncturable pseudorandom function, and* SPS *is a splittable signature scheme satisfying '$\text{VK}_{\text{SPS-REJ}}$ indistinguishability', for any* PPT *adversary $\mathcal{A}$, for any security parameter λ, $|\mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,2)}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,3)}(\lambda)| \leq \mathsf{negl}(\lambda)$ for some negligible function* negl.

Proof To prove Lemma C.3, we consider the following sequence of ℓ^* intermediate hybrid experiments between $\mathsf{Hyb}_{0,\nu-1,2}$ and $\mathsf{Hyb}_{0,\nu-1,3}$:

Sequence of Intermediate Hybrids between $\mathsf{Hyb}_{0,\nu-1,2}$ and $\mathsf{Hyb}_{0,\nu-1,3}$

$\mathsf{Hyb}_{0,\nu-1,2,\iota}$ ($\iota = 0, \ldots, \ell^* - 1$): In this experiment in response to the ν^{th} signing key query of $\mathcal{A}$ corresponding to TM $M^{(\nu)} \in \mathbb{M}_\lambda$ with $M^{(\nu)}(x^*) = 0$, $\mathcal{B}$ proceeds as follows:

1. It first chooses PPRF keys $K_1^{(\nu)}, \ldots, K_\lambda^{(\nu)}, K_{\text{SPS},A}^{(\nu)}, K_{\text{SPS},B}^{(\nu)}, K_{\text{SPS},E}^{(\nu)}, K_{\text{SPS},F}^{(\nu)} \xleftarrow{\$} \mathcal{F}.\mathsf{Setup}(1^\lambda)$.
2. Next, it generates $(\text{PP}_{\text{ACC}}^{(\nu)}, w_0^{(\nu)}, \text{STORE}_0^{(\nu)}) \xleftarrow{\$} \mathsf{ACC.Setup}(1^\lambda, n_{\text{ACC-BLK}} = 2^\lambda)$ and $(\text{PP}_{\text{ITR}}^{(\nu)}, v_0^{(\nu)}) \xleftarrow{\$} \mathsf{ITR.Setup}(1^\lambda, n_{\text{ITR}} = 2^\lambda)$.
3. It provides $\mathcal{A}$ with the signing key

$$\text{SK}_{\text{ABS}}\{M^{(\nu)}\} = \left(\begin{array}{l} \text{HK}, \text{PP}_{\text{ACC}}^{(\nu)}, w_0^{(\nu)}, \text{STORE}_0^{(\nu)}, \text{PP}_{\text{ITR}}^{(\nu)}, v_0^{(\nu)}, \\ \mathcal{IO}(\mathsf{Init\text{-}SPS.Prog}[q_0^{(\nu)}, w_0^{(\nu)}, v_0^{(\nu)}, K_{\text{SPS},E}^{(\nu)}]), \\ \mathcal{IO}(\underline{\mathsf{Accumulate.Prog}^{(1,\iota)}[n_{\text{SSB-BLK}} = 2^\lambda, \text{HK}, \text{PP}_{\text{ACC}}^{(\nu)}, \text{PP}_{\text{ITR}}^{(\nu)}, K_{\text{SPS},E}^{(\nu)}, K_{\text{SPS},F}^{(\nu)}, h^*, \ell^*]}), \\ \mathcal{IO}(\underline{\mathsf{Change\text{-}SPS.Prog}^{(1,\iota)}[K_{\text{SPS},A}^{(\nu)}, K_{\text{SPS},B}^{(\nu)}, K_{\text{SPS},E}^{(\nu)}, K_{\text{SPS},F}^{(\nu)}, h^*, \ell^*]}), \\ \mathcal{IO}(\underline{\mathsf{Constrained\text{-}Key.Prog}_{\text{ABS}}^{(1)}[M^{(\nu)}, T = 2^\lambda, t^{*(\nu)} \text{PP}_{\text{ACC}}^{(\nu)}, \text{PP}_{\text{ITR}}^{(\nu)}, K, K_1^{(\nu)}, \ldots, K_\lambda^{(\nu)}, K_{\text{SPS},A}^{(\nu)}, K_{\text{SPS},B}^{(\nu)}, h^*, \ell^*]}) \end{array} \right),$$

where the programs $\mathsf{Accumulate.Prog}^{(1,\iota)}$ and $\mathsf{Change\text{-}SPS.Prog}^{(1,\iota)}$ are the modifications of the programs $\mathsf{Accumulate.Prog}^{(2)}$ and $\mathsf{Change\text{-}SPS.Prog}^{(2)}$ (Figs. **13** and **14**) and are depicted in Figs. **27** and **28** respectively.

The rest of the experiment is similar to $\mathsf{Hyb}_{0,\nu-1,2}$. Observe that $\mathsf{Hyb}_{0,\nu-1,2,\ell^*-1}$ coincides with $\mathsf{Hyb}_{0,\nu-1,2}$ and $\mathsf{Hyb}_{0,\nu-1,2,0}$ corresponds to $\mathsf{Hyb}_{0,\nu-1,3}$.

Constants: Maximum number of blocks for SSB hash $n_{\text{SSB-BLK}} = 2^\lambda$, SSB hash key HK, Public parameters for positional accumulator PP_{ACC}, Public parameters for iterator PP_{ITR}, PPRF keys $K_{\text{SPS},E}$, $K_{\text{SPS},F}$, SSB hash value of challenge input h^*, Length of challenge input ℓ^*

Inputs: Index i, Symbol SYM_{IN}, TM state ST, Accumulator value w_{IN}, Auxiliary value AUX, Iterator value v_{IN}, Signature $\sigma_{\text{SPS,IN}}$, SSB hash value h, SSB opening value π_{SSB}

Output: (Accumulator value w_{OUT}, Iterator value v_{OUT}, Signature $\sigma_{\text{SPS-OUT}}$), or $\perp$

1. (a) Compute $r_{\text{SPS},E} = \mathcal{F}(K_{\text{SPS},E}, (h,i))$, $(\text{SK}_{\text{SPS},E}, \text{VK}_{\text{SPS},E}, \text{VK}_{\text{SPS-REJ},E}) = \mathsf{SPS.Setup}(1^\lambda; r_{\text{SPS},E})$.
   (b) Compute $r_{\text{SPS},F} = \mathcal{F}(K_{\text{SPS},F}, (h,i))$, $(\text{SK}_{\text{SPS},F}, \text{VK}_{\text{SPS},F}, \text{VK}_{\text{SPS-REJ},F}) = \mathsf{SPS.Setup}(1^\lambda; r_{\text{SPS},F})$.
   (c) Set $m_{\text{IN}} = (v_{\text{IN}}, \text{ST}, w_{\text{IN}}, 0)$ and $\alpha =$'-'.
   (d) If $\mathsf{SPS.Verify}(\text{VK}_{\text{SPS},E}, m_{\text{IN}}, \sigma_{\text{SPS,IN}}) = 1$, set $\alpha =$'E'.
   (e) If $[\alpha =$'-'$] \wedge [(i > \ell^*) \vee \underline{(i \leq \iota)} \vee (h \neq h^*)]$, output $\perp$.
   Else if $[\alpha =$'-'$] \wedge [\mathsf{SPS.Verify}(\text{VK}_{\text{SPS},F}, m_{\text{IN}}, \sigma_{\text{SPS,IN}}) = 1]$, set $\alpha =$'F'.
   (f) If $\alpha =$'-', output $\perp$.
2. If $\mathsf{SSB.Verify}(\text{HK}, h, i, \text{SYM}_{\text{IN}}, \pi_{\text{SSB}}) = 0$, output $\perp$.
3. (a) Compute $w_{\text{OUT}} = \mathsf{ACC.Update}(\text{PP}_{\text{ACC}}, w_{\text{IN}}, \text{SYM}_{\text{IN}}, i, \text{AUX})$. If $w_{\text{OUT}} = \perp$, output $\perp$.
   (b) Compute $v_{\text{OUT}} = \mathsf{ITR.Iterate}(\text{PP}_{\text{ITR}}, v_{\text{IN}}, (\text{ST}, w_{\text{IN}}, 0))$.
4. (a) Compute $r'_{\text{SPS},E} = \mathcal{F}(K_{\text{SPS},E}, (h, i+1))$, $(\text{SK}'_{\text{SPS},E}, \text{VK}'_{\text{SPS},E}, \text{VK}'_{\text{SPS-REJ},E}) = \mathsf{SPS.Setup}(1^\lambda; r'_{\text{SPS},E})$.
   (b) Compute $r'_{\text{SPS},F} = \mathcal{F}(K_{\text{SPS},F}, (h, i+1))$, $(\text{SK}'_{\text{SPS},F}, \text{VK}'_{\text{SPS},F}, \text{VK}'_{\text{SPS-REJ},F}) = \mathsf{SPS.Setup}(1^\lambda; r'_{\text{SPS},F})$.
   (c) Set $m_{\text{OUT}} = (v_{\text{OUT}}, \text{ST}, w_{\text{OUT}}, 0)$. If $i < \ell^*$, compute $\sigma_{\text{SPS,OUT}} = \mathsf{SPS.Sign}(\text{SK}'_{\text{SPS},\alpha}, m_{\text{OUT}})$.
   Else, compute $\sigma_{\text{SPS,OUT}} = \mathsf{SPS.Sign}(\text{SK}'_{\text{SPS},E}, m_{\text{OUT}})$.
5. Output $(w_{\text{OUT}}, v_{\text{OUT}}, \sigma_{\text{SPS,OUT}})$.

Fig. 27 $\mathsf{Accumulate.Prog}^{(1,\iota)}$

Constants: PPRF keys $K_{\text{SPS},A}$, $K_{\text{SPS},B}$, $K_{\text{SPS},E}$, $K_{\text{SPS},F}$, SSB hash value of challenge input h^*, Length of challenge input ℓ^*

Inputs: TM state ST, Accumulator value w, Iterator value v, SSB hash value h, Length ℓ_{INP}, Signature $\sigma_{\text{SPS,IN}}$

Output: Signature $\sigma_{\text{SPS,OUT}}$, or $\perp$

1. (a) Compute $r_{\text{SPS},E} = \mathcal{F}(K_{\text{SPS},E}, (h, \ell_{\text{INP}}))$, $(\text{SK}_{\text{SPS},E}, \text{VK}_{\text{SPS},E}, \text{VK}_{\text{SPS-REJ},E}) = \mathsf{SPS.Setup}(1^\lambda; r_{\text{SPS},E})$.
   (b) Compute $r_{\text{SPS},F} = \mathcal{F}(K_{\text{SPS},F}, (h, \ell_{\text{INP}}))$, $(\text{SK}_{\text{SPS},F}, \text{VK}_{\text{SPS},F}, \text{VK}_{\text{SPS-REJ},F}) = \mathsf{SPS.Setup}(1^\lambda; r_{\text{SPS},F})$.
   (c) Set $m = (v, \text{ST}, w, 0)$ and $\alpha =$'-'.
   (d) If $\mathsf{SPS.Verify}(\text{VK}_{\text{SPS},E}, m, \sigma_{\text{SPS,IN}}) = 1$, set $\alpha =$'E'.
   (e) If $[\alpha =$'-'$] \wedge [(\ell_{\text{INP}} > \ell^*) \vee \underline{(\ell_{\text{INP}} \leq \iota)} \vee (h \neq h^*)]$, output $\perp$.
   Else if $[\alpha =$'-'$] \wedge [\mathsf{SPS.Verify}(\text{VK}_{\text{SPS},F}, m, \sigma_{\text{SPS,IN}}) = 1]$, set $\alpha =$'F'.
   (f) If $\alpha =$'-', output $\perp$.
2. (a) Compute $r_{\text{SPS},A} = \mathcal{F}(K_{\text{SPS},A}, (h, \ell_{\text{INP}}, 0))$, $(\text{SK}_{\text{SPS},A}, \text{VK}_{\text{SPS},A}, \text{VK}_{\text{SPS-REJ},A}) = \mathsf{SPS.Setup}(1^\lambda; r_{\text{SPS},A})$.
   (b) Compute $r_{\text{SPS},B} = \mathcal{F}(K_{\text{SPS},B}, (h, \ell_{\text{INP}}, 0))$, $(\text{SK}_{\text{SPS},B}, \text{VK}_{\text{SPS},B}, \text{VK}_{\text{SPS-REJ},B}) = \mathsf{SPS.Setup}(1^\lambda; r_{\text{SPS},B})$.
   (c) If $[(h, \ell_{\text{INP}}) = (h^*, \ell^*)] \wedge [\alpha =$'$F$'$]$, output $\sigma_{\text{SPS,OUT}} = \mathsf{SPS.Sign}(\text{SK}_{\text{SPS},B}, m)$.
   Else, output $\sigma_{\text{SPS,OUT}} = \mathsf{SPS.Sign}(\text{SK}_{\text{SPS},A}, m)$.

Fig. 28 $\mathsf{Change\text{-}SPS.Prog}^{(1,\iota)}$

Analysis

Let us denote by $\mathsf{Adv}^{(0,\nu-1,2,\iota)}_{\mathcal{A}}(\lambda)$ the advantage of $\mathcal{A}$, i.e., the absolute difference between $1/2$ and $\mathcal{A}$'s probability of correctly guessing the random bit selected by

the challenger $\mathcal{B}$, in the hybrid experiment $\mathsf{Hyb}_{0,\nu-1,2,\iota}$, for $\iota \in [0, \ell^* - 1]$. Clearly, $\mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,2)}(\lambda) \equiv \mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,2,\ell^*-1)}(\lambda)$ and $\mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,3)}(\lambda) \equiv \mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,2,0)}(\lambda)$. Hence we have,

$$|\mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,2)}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,3)}(\lambda)| \leq \sum_{\iota=1}^{\ell^*-1} |\mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,2,\iota)}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,2,\iota-1)}(\lambda)|. \tag{C.3}$$

Claim C.8 below justifies that the RHS of Eq. (C.3) is negligible and consequently Lemma C.3 follows.

Claim C.8 *Assuming $\mathcal{IO}$ is a secure indistinguishability obfuscator for $\mathsf{P/poly}$, $\mathcal{F}$ is a secure puncturable pseudorandom function, and SPS is a splittable signature scheme satisfying '$\mathrm{VK}_{\text{SPS-REJ}}$ indistinguishability', for any PPT adversary $\mathcal{A}$, for any security parameter λ, $|\mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,2,\iota)}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,2,\iota-1)}(\lambda)| \leq \mathsf{negl}(\lambda)$ for some negligible function negl.*

Proof The proof of Claim C.8 is similar to that of Lemma C.2 with some appropriate modifications which are easy to find out. □

□

Lemma C.4 *Assuming $\mathcal{IO}$ is a secure indistinguishability obfuscator for $\mathsf{P/poly}$, $\mathcal{F}$ is a secure puncturable pseudorandom function, and SPS is a splittable signature scheme satisfying '$\mathrm{VK}_{\text{SPS-ONE}}$ indistinguishability', for any PPT adversary $\mathcal{A}$, for any security parameter λ, $|\mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,3)}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,3,0)}(\lambda)| \leq \mathsf{negl}(\lambda)$ for some negligible function negl.*

Proof In order to prove Lemma C.4, we consider the following sequence of intermediate hybrid experiments between $\mathsf{Hyb}_{0,\nu-1,3}$ and $\mathsf{Hyb}_{0,\nu-1,3,0}$.

Sequence of Intermediate Hybrids between $\mathsf{Hyb}_{0,\nu-1,3}$ and $\mathsf{Hyb}_{0,\nu-1,3,0}$

$\mathsf{Hyb}_{0,\nu-1,3\text{-I}}$: This experiment coincides with $\mathsf{Hyb}_{0,\nu-1,3}$.

$\mathsf{Hyb}_{0,\nu-1,3\text{-II}}$: In this experiment, to answer the ν^{th} signing key query of $\mathcal{A}$ corresponding to TM $M^{(\nu)} \in \mathbb{M}_\lambda$ with $M^{(\nu)}(x^*) = 0$, $\mathcal{B}$ proceeds as follows:

1. It first generates all the PPRF keys together with the public parameters for the positional accumulator and the iterator just as in $\mathsf{Hyb}_{0,\nu-1,3\text{-I}}$.
2. Next, it creates the punctured PPRF key $K_{\text{SPS},E}^{(\nu)}\{(h^*, 0)\} \xleftarrow{\$} \mathcal{F}.\mathsf{Puncture}(K_{\text{SPS},E}^{(\nu)}, (h^*, 0))$ as well as computes $r_{\text{SPS},G}^{(\nu,0)} = \mathcal{F}(K_{\text{SPS},E}^{(\nu)}, (h^*, 0))$ and $(\mathrm{SK}_{\text{SPS},G}^{(\nu,0)}, \mathrm{VK}_{\text{SPS},G}^{(\nu,0)}, \mathrm{VK}_{\text{SPS-REJ},G}^{(\nu,0)} = \mathsf{SPS.Setup}(1^\lambda; r_{\text{SPS},G}^{(\nu,0)})$.
3. Then, it sets $m_{0,0}^{(\nu)} = (v_0^{(\nu)}, q_0^{(\nu)}, w_0^{(\nu)}, 0)$ and computes $\sigma_{\text{SPS},G}^{(\nu,0)} = \mathsf{SPS.Sign}(\mathrm{SK}_{\text{SPS},G}^{(\nu,0)}, m_{0,0}^{(\nu)})$.

4. $\mathcal{B}$ gives $\mathcal{A}$ the signing key
$\mathrm{SK}_{\mathrm{ABS}}\{M^{(\nu)}\} =$

$$\left(\begin{array}{l}\mathrm{HK}, \mathrm{PP}_{\mathrm{ACC}}^{(\nu)}, w_0^{(\nu)}, \mathrm{STORE}_0^{(\nu)}, \mathrm{PP}_{\mathrm{ITR}}^{(\nu)}, v_0^{(\nu)}, \\ \mathcal{IO}(\mathsf{Init\text{-}SPS.Prog}^{(1)}[q_0^{(\nu)}, w_0^{(\nu)}, v_0^{(\nu)}, K_{\mathrm{SPS},E}^{(\nu)}\{(h^*,0)\}, \sigma_{\mathrm{SPS},G}^{(\nu,0)}, h^*]), \\ \mathcal{IO}(\mathsf{Accumulate.Prog}^{(2,1)}[n_{\text{SSB-BLK}} = 2^\lambda, \mathrm{HK}, \mathrm{PP}_{\mathrm{ACC}}^{(\nu)}, \mathrm{PP}_{\mathrm{ITR}}^{(\nu)}, K_{\mathrm{SPS},E}^{(\nu)}\{(h^*,0)\}, \\ \quad K_{\mathrm{SPS},F}^{(\nu)}, \mathrm{VK}_{\mathrm{SPS},G}^{(\nu,0)}, h^*, \ell^*]), \\ \mathcal{IO}(\mathsf{Change\text{-}SPS.Prog}^{(2)}[K_{\mathrm{SPS},A}^{(\nu)}, K_{\mathrm{SPS},B}^{(\nu)}, K_{\mathrm{SPS},E}^{(\nu)}\{(h^*,0)\}, K_{\mathrm{SPS},F}^{(\nu)}, h^*, \ell^*]), \\ \mathcal{IO}(\mathsf{Constrained\text{-}Key.Prog}_{\mathrm{ABS}}^{(1)}[M^{(\nu)}, T = 2^\lambda, t^{*(\nu)}, \mathrm{PP}_{\mathrm{ACC}}^{(\nu)}, \mathrm{PP}_{\mathrm{ITR}}^{(\nu)}, K, \\ \quad K_1^{(\nu)}, \ldots, K_\lambda^{(\nu)}, K_{\mathrm{SPS},A}^{(\nu)}, K_{\mathrm{SPS},B}^{(\nu)}, h^*, \ell^*])\end{array}\right),$$

where the programs $\mathsf{Init\text{-}SPS.Prog}^{(1)}$ and $\mathsf{Accumulate.Prog}^{(2,1)}$ respectively are the alterations of the programs $\mathsf{Init\text{-}SPS.Prog}$ and $\mathsf{Accumulate.Prog}^{(2)}$ (Figs. **2** and **13**) and are depicted in Figs. **29** and **30**.

The remaining part of the experiment is similar to $\mathsf{Hyb}_{0,\nu-1,3\text{-I}}$.

Constants:	Initial TM state q_0, Accumulator value w_0, Iterator value v_0, Punctured PPRF key $K_{\mathrm{SPS},E}\{(h^*,0)\}$, Signature σ_G, SSB hash value of challenge input h^*
Input:	SSB hash value h
Output:	Signature $\sigma_{\mathrm{SPS,OUT}}$

1. If $h = h^*$, output σ_G.
Else, compute $r_{\mathrm{SPS},E} = \mathcal{F}(K_{\mathrm{SPS},E}\{(h^*,0)\}, (h,0)), (\mathrm{SK}_{\mathrm{SPS},E}, \mathrm{VK}_{\mathrm{SPS},E}, \mathrm{VK}_{\text{SPS-REJ},E}) =$ $\mathsf{SPS.Setup}(1^\lambda; r_{\mathrm{SPS},E})$.
2. Output $\sigma_{\mathrm{SPS,OUT}} = \mathsf{SPS.Sign}(\mathrm{SK}_{\mathrm{SPS},E}, (v_0, q_0, w_0, 0))$.

Fig. 29 $\mathsf{Init\text{-}SPS.Prog}^{(1)}$

$\mathsf{Hyb}_{0,\nu-1,3\text{-III}}$: This experiment is analogous to $\mathsf{Hyb}_{0,\nu-1,3\text{-II}}$ with the only exception that while constructing the ν^{th} signing key queried by $\mathcal{A}$ corresponding to TM $M^{(\nu)} \in \mathbb{M}_\lambda$ with $M^{(\nu)}(x^*) = 0$, $\mathcal{B}$ selects $r_{\mathrm{SPS},G}^{(\nu,0)} \xleftarrow{\$} \mathcal{Y}_{\mathrm{PPRF}}$. More formally, to answer the ν^{th} signing key query of $\mathcal{A}$, $\mathcal{B}$ creates all the components as in $\mathsf{Hyb}_{0,\nu-1,3\text{-II}}$ except that it generates $(\mathrm{SK}_{\mathrm{SPS},G}^{(\nu,0)}, \mathrm{VK}_{\mathrm{SPS},G}^{(\nu,0)}, \mathrm{VK}_{\text{SPS-REJ},G}^{(\nu,0)}) \xleftarrow{\$} \mathsf{SPS.Setup}(1^\lambda)$, sets $m_{0,0}^{(\nu)} = (v_0^{(\nu)}, q_0^{(\nu)}, w_0^{(\nu)}, 0)$, computes $\sigma_{\mathrm{SPS},G}^{(\nu,0)} = \mathsf{SPS.Sign}(\mathrm{SK}_{\mathrm{SPS},G}^{(\nu,0)}, m_{0,0}^{(\nu)})$ and provides $\mathcal{A}$ with the signing key
$\mathrm{SK}_{\mathrm{ABS}}\{M^{(\nu)}\} =$

$$\left(\begin{array}{l}\mathrm{HK}, \mathrm{PP}_{\mathrm{ACC}}^{(\nu)}, w_0^{(\nu)}, \mathrm{STORE}_0^{(\nu)}, \mathrm{PP}_{\mathrm{ITR}}^{(\nu)}, v_0^{(\nu)}, \\ \mathcal{IO}(\mathsf{Init\text{-}SPS.Prog}^{(1)}[q_0^{(\nu)}, w_0^{(\nu)}, v_0^{(\nu)}, K_{\mathrm{SPS},E}^{(\nu)}\{(h^*,0)\}, \sigma_{\mathrm{SPS},G}^{(\nu,0)}, h^*]), \\ \mathcal{IO}(\mathsf{Accumulate.Prog}^{(2,1)}[n_{\text{SSB-BLK}} = 2^\lambda, \mathrm{HK}, \mathrm{PP}_{\mathrm{ACC}}^{(\nu)}, \mathrm{PP}_{\mathrm{ITR}}^{(\nu)}, K_{\mathrm{SPS},E}^{(\nu)}\{(h^*,0)\}, \\ \quad K_{\mathrm{SPS},F}^{(\nu)}, \mathrm{VK}_{\mathrm{SPS},G}^{(\nu,0)}, h^*, \ell^*]), \\ \mathcal{IO}(\mathsf{Change\text{-}SPS.Prog}^{(2)}[K_{\mathrm{SPS},A}^{(\nu)}, K_{\mathrm{SPS},B}^{(\nu)}, K_{\mathrm{SPS},E}^{(\nu)}\{(h^*,0)\}, K_{\mathrm{SPS},F}^{(\nu)}, h^*, \ell^*]), \\ \mathcal{IO}(\mathsf{Constrained\text{-}Key.Prog}_{\mathrm{ABS}}^{(1)}[M^{(\nu)}, T = 2^\lambda, t^{*(\nu)}, \mathrm{PP}_{\mathrm{ACC}}^{(\nu)}, \mathrm{PP}_{\mathrm{ITR}}^{(\nu)}, K, \\ \quad K_1^{(\nu)}, \ldots, K_\lambda^{(\nu)}, K_{\mathrm{SPS},A}^{(\nu)}, K_{\mathrm{SPS},B}^{(\nu)}, h^*, \ell^*])\end{array}\right).$$

$\mathsf{Hyb}_{0,\nu-1,3\text{-IV}}$: This experiment is the same as $\mathsf{Hyb}_{0,\nu-1,3\text{-III}}$ with the exception that in response to the ν^{th} signing key query of $\mathcal{A}$ corresponding to TM $M^{(\nu)} \in \mathbb{M}_\lambda$ with $M^{(\nu)}(x^*) = 0$, $\mathcal{B}$ proceeds as follows:

Constants: Maximum number of blocks for SSB hash $n_{\text{SSB-BLK}} = 2^\lambda$, SSB hash key HK, Public parameters for positional accumulator PP_{ACC}, Public parameters for iterator PP_{ITR}, Punctured PPRF key $K_{\text{SPS},E}\{(h^*,0)\}$, PPRF key $K_{\text{SPS},F}$, Verification key VK_G, SSB hash value of challenge input h^*, Length of challenge input ℓ^*

Inputs: Index i, Symbol SYM_{IN}, TM state ST, Accumulator value w_{IN}, Auxiliary value AUX, Iterator value v_{IN}, Signature $\sigma_{\text{SPS,IN}}$, SSB hash value h, SSB opening value π_{SSB}

Output: (Accumulator value w_{OUT}, Iterator value v_{OUT}, Signature $\sigma_{\text{SPS-OUT}}$), or $\perp$

1. (a) If $(h,i) \neq (h^*,0)$, compute $r_{\text{SPS},E} = \mathcal{F}(K_{\text{SPS},E}\{(h^*,0)\},(h,i))$, $(\text{SK}_{\text{SPS},E}, \text{VK}_{\text{SPS},E}, \text{VK}_{\text{SPS-REJ},E}) = \mathsf{SPS.Setup}(1^\lambda; r_{\text{SPS},E})$.
   Else, set $\text{VK}_{\text{SPS},E} = \text{VK}_G$.
   (b) Compute $r_{\text{SPS},F} = \mathcal{F}(K_{\text{SPS},F},(h,i))$, $(\text{SK}_{\text{SPS},F}, \text{VK}_{\text{SPS},F}, \text{VK}_{\text{SPS-REJ},F}) = \mathsf{SPS.Setup}(1^\lambda; r_{\text{SPS},F})$.
   (c) Set $m_{\text{IN}} = (v_{\text{IN}}, \text{ST}, w_{\text{IN}}, 0)$ and $\alpha =$'-'.
   (d) If $\mathsf{SPS.Verify}(\text{VK}_{\text{SPS},E}, m_{\text{IN}}, \sigma_{\text{SPS,IN}}) = 1$, set $\alpha =$'E'.
   (e) If $[\alpha =\text{'-'}] \wedge [(i > \ell^*) \vee (i = 0) \vee (h \neq h^*)]$, output $\perp$.
   Else if $[\alpha =\text{'-'}] \wedge [\mathsf{SPS.Verify}(\text{VK}_{\text{SPS},F}, m_{\text{IN}}, \sigma_{\text{SPS,IN}}) = 1]$, set $\alpha =$'F'.
   (f) If $\alpha =$'-', output $\perp$.
2. If $\mathsf{SSB.Verify}(\text{HK}, h, i, \text{SYM}_{\text{IN}}, \pi_{\text{SSB}}) = 0$, output $\perp$.
3. (a) Compute $w_{\text{OUT}} = \mathsf{ACC.Update}(\text{PP}_{\text{ACC}}, w_{\text{IN}}, \text{SYM}_{\text{IN}}, i, \text{AUX})$. If $w_{\text{OUT}} = \perp$, output $\perp$.
   (b) Compute $v_{\text{OUT}} = \mathsf{ITR.Iterate}(\text{PP}_{\text{ITR}}, v_{\text{IN}}, (\text{ST}, w_{\text{IN}}, 0))$.
4. (a) Compute $r'_{\text{SPS},E} = \mathcal{F}(K_{\text{SPS},E}\{(h^*,0)\},(h,i+1))$, $(\text{SK}'_{\text{SPS},E}, \text{VK}'_{\text{SPS},E}, \text{VK}'_{\text{SPS-REJ},E}) = \mathsf{SPS.Setup}(1^\lambda; r'_{\text{SPS},E})$.
   (b) Compute $r'_{\text{SPS},F} = \mathcal{F}(K_{\text{SPS},F},(h,i+1))$, $(\text{SK}'_{\text{SPS},F}, \text{VK}'_{\text{SPS},F}, \text{VK}'_{\text{SPS-REJ},F}) = \mathsf{SPS.Setup}(1^\lambda; r'_{\text{SPS},F})$.
   (c) Set $m_{\text{OUT}} = (v_{\text{OUT}}, \text{ST}, w_{\text{OUT}}, 0)$. If $i < \ell^*$, compute $\sigma_{\text{SPS,OUT}} = \mathsf{SPS.Sign}(\text{SK}'_{\text{SPS},\alpha}, m_{\text{OUT}})$.
   Else, compute $\sigma_{\text{SPS,OUT}} = \mathsf{SPS.Sign}(\text{SK}'_{\text{SPS},E}, m_{\text{OUT}})$.
5. Output $(w_{\text{OUT}}, v_{\text{OUT}}, \sigma_{\text{SPS,OUT}})$.

Fig. 30 $\mathsf{Accumulate.Prog}^{(2,1)}$

1. It first generates the full and punctured PPRF keys together with the public parameters for the positional accumulator and the iterator just as in $\mathsf{Hyb}_{0,\nu-1,3\text{-III}}$.
2. Next, it creates $(\text{SK}^{(\nu,0)}_{\text{SPS},G}, \text{VK}^{(\nu,0)}_{\text{SPS},G}, \text{VK}^{(\nu,0)}_{\text{SPS-REJ},G}) \xleftarrow{\$} \mathsf{SPS.Setup}(1^\lambda)$, sets $m^{(\nu)}_{0,0} = (v^{(\nu)}_0, q^{(\nu)}_0, w^{(\nu)}_0, 0)$, and forms $(\sigma^{(\nu,0)}_{\text{SPS-ONE},m^{(\nu)}_{0,0},G}, \text{VK}^{(\nu,0)}_{\text{SPS-ONE},G}, \text{SK}^{(\nu,0)}_{\text{SPS-ABO},G}, \text{VK}^{(\nu,0)}_{\text{SPS-ABO},G}) \xleftarrow{\$} \mathsf{SPS.Split}(\text{SK}^{(\nu,0)}_{\text{SPS},G}, m^{(\nu)}_{0,0})$.
3. $\mathcal{B}$ hands $\mathcal{A}$ the signing key $\text{SK}_{\text{ABS}}\{M^{(\nu)}\} =$

$$\left(\begin{array}{l}
\text{HK}, \text{PP}^{(\nu)}_{\text{ACC}}, w^{(\nu)}_0, \text{STORE}^{(\nu)}_0, \text{PP}^{(\nu)}_{\text{ITR}}, v^{(\nu)}_0, \\
\mathcal{IO}(\mathsf{Init\text{-}SPS.Prog}^{(1)}[q^{(\nu)}_0, w^{(\nu)}_0, v^{(\nu)}_0, K^{(\nu)}_{\text{SPS},E}\{(h^*,0)\}, \sigma^{(\nu,0)}_{\text{SPS-ONE},m^{(\nu)}_{0,0},G}, h^*]), \\
\mathcal{IO}(\mathsf{Accumulate.Prog}^{(2,1)}[n_{\text{SSB-BLK}} = 2^\lambda, \text{HK}, \text{PP}^{(\nu)}_{\text{ACC}}, \text{PP}^{(\nu)}_{\text{ITR}}, K^{(\nu)}_{\text{SPS},E}\{(h^*,0)\}, \\
\quad K^{(\nu)}_{\text{SPS},F}, \text{VK}^{(\nu,0)}_{\text{SPS-ONE},G}, h^*, \ell^*]), \\
\mathcal{IO}(\mathsf{Change\text{-}SPS.Prog}^{(2)}[K^{(\nu)}_{\text{SPS},A}, K^{(\nu)}_{\text{SPS},B}, K^{(\nu)}_{\text{SPS},E}\{(h^*,0)\}, K^{(\nu)}_{\text{SPS},F}, h^*, \ell^*]), \\
\mathcal{IO}(\mathsf{Constrained\text{-}Key.Prog}^{(1)}_{\text{ABS}}[M^{(\nu)}, T = 2^\lambda, t^{*(\nu)}, \text{PP}^{(\nu)}_{\text{ACC}}, \text{PP}^{(\nu)}_{\text{ITR}}, K, \\
\quad K^{(\nu)}_1, \ldots, K^{(\nu)}_\lambda, K^{(\nu)}_{\text{SPS},A}, K^{(\nu)}_{\text{SPS},B}, h^*, \ell^*])
\end{array}\right).$$

$\mathsf{Hyb}_{\mathbf{0,\nu-1,3\text{-}V}}$: In this experiment, in reply to the ν^{th} signing key query of $\mathcal{A}$ corresponding to TM $M^{(\nu)} \in \mathbb{M}_\lambda$ with $M^{(\nu)}(x^*) = 0$, $\mathcal{B}$ generates all the components just

as in $\mathsf{Hyb}_{0,\nu-1,3\text{-IV}}$ and gives $\mathcal{A}$ the signing key

$$\mathrm{SK}_{\mathrm{ABS}}\{M^{(\nu)}\} = \left(\begin{array}{l} \mathrm{HK}, \mathrm{PP}_{\mathrm{ACC}}^{(\nu)}, w_0^{(\nu)}, \mathrm{STORE}_0^{(\nu)}, \mathrm{PP}_{\mathrm{ITR}}^{(\nu)}, v_0^{(\nu)}, \\ \mathcal{IO}(\mathsf{Init\text{-}SPS.Prog}^{(1)}[q_0^{(\nu)}, w_0^{(\nu)}, v_0^{(\nu)}, K_{\mathrm{SPS},E}^{(\nu)}\{(h^*,0)\}, \sigma^{(\nu,0)}_{\text{SPS-ONE},m_{0,0}^{(\nu)},G}, h^*]), \\ \mathcal{IO}(\mathsf{Accumulate.Prog}^{(2,2)}[n_{\text{SSB-BLK}} = 2^\lambda, \mathrm{HK}, \mathrm{PP}_{\mathrm{ACC}}^{(\nu)}, \mathrm{PP}_{\mathrm{ITR}}^{(\nu)}, K_{\mathrm{SPS},E}^{(\nu)}\{(h^*,0)\}, \\ \quad \underline{K_{\mathrm{SPS},F}^{(\nu)}, \mathrm{VK}^{(\nu,0)}_{\text{SPS-ONE},G}, m_{0,0}^{(\nu)}, h^*, \ell^*}]), \\ \mathcal{IO}(\mathsf{Change\text{-}SPS.Prog}^{(2)}[K_{\mathrm{SPS},A}^{(\nu)}, K_{\mathrm{SPS},B}^{(\nu)}, K_{\mathrm{SPS},E}^{(\nu)}\{(h^*,0)\}, K_{\mathrm{SPS},F}^{(\nu)}, h^*, \ell^*]), \\ \mathcal{IO}(\mathsf{Constrained\text{-}Key.Prog}^{(1)}_{\mathrm{ABS}}[M^{(\nu)}, T = 2^\lambda, t^{*(\nu)}, \mathrm{PP}_{\mathrm{ACC}}^{(\nu)}, \mathrm{PP}_{\mathrm{ITR}}^{(\nu)}, K, \\ \quad K_1^{(\nu)}, \ldots, K_\lambda^{(\nu)}, K_{\mathrm{SPS},A}^{(\nu)}, K_{\mathrm{SPS},B}^{(\nu)}, h^*, \ell^*]) \end{array}\right),$$

where the program $\mathsf{Accumulate.Prog}^{(2,2)}$, described in Fig. **31**, is an alteration of the program $\mathsf{Accumulate.Prog}^{(2,1)}$ (Fig. **30**). The rest of the experiment is similar to $\mathsf{Hyb}_{0,\nu-1,3\text{-IV}}$.

Constants: Maximum number of blocks for SSB hash $n_{\text{SSB-BLK}} = 2^\lambda$, SSB hash key HK, Public parameters for positional accumulator $\mathrm{PP}_{\mathrm{ACC}}$, Public parameters for iterator $\mathrm{PP}_{\mathrm{ITR}}$, Punctured PPRF key $K_{\mathrm{SPS},E}\{(h^*,0)\}$, PPRF key $K_{\mathrm{SPS},F}$, Verification key VK_G, Message $m_{0,0}$, SSB hash value of challenge input h^*, Length of challenge input ℓ^*

Inputs: Index i, Symbol $\mathrm{SYM}_{\mathrm{IN}}$, TM state ST, Accumulator value w_{IN}, Auxiliary value AUX, Iterator value v_{IN}, Signature $\sigma_{\mathrm{SPS,IN}}$, SSB hash value h, SSB opening value π_{SSB}

Output: (Accumulator value w_{OUT}, Iterator value v_{OUT}, Signature $\sigma_{\text{SPS-OUT}}$), or $\perp$

1. (a) If $(h,i) \neq (h^*,0)$, compute $r_{\mathrm{SPS},E} = \mathcal{F}(K_{\mathrm{SPS},E}\{(h^*,0)\},(h,i))$, $(\mathrm{SK}_{\mathrm{SPS},E}, \mathrm{VK}_{\mathrm{SPS},E}, \mathrm{VK}_{\text{SPS-REJ},E}) = \mathsf{SPS.Setup}(1^\lambda; r_{\mathrm{SPS},E})$.
   Else, set $\mathrm{VK}_{\mathrm{SPS},E} = \mathrm{VK}_G$.
   (b) Compute $r_{\mathrm{SPS},F} = \mathcal{F}(K_{\mathrm{SPS},F},(h,i))$, $(\mathrm{SK}_{\mathrm{SPS},F}, \mathrm{VK}_{\mathrm{SPS},F}, \mathrm{VK}_{\text{SPS-REJ},F}) = \mathsf{SPS.Setup}(1^\lambda; r_{\mathrm{SPS},F})$.
   (c) Set $m_{\mathrm{IN}} = (v_{\mathrm{IN}}, \mathrm{ST}, w_{\mathrm{IN}}, 0)$ and $\alpha =$'-'.
   (d) If $\mathsf{SPS.Verify}(\mathrm{VK}_{\mathrm{SPS},E}, m_{\mathrm{IN}}, \sigma_{\mathrm{SPS,IN}}) = 1$, set $\alpha =$'E'.
   (e) If $[\alpha =\text{'-'}] \wedge [(i > \ell^*) \vee (i = 0) \vee (h \neq h^*)]$, output $\perp$.
   Else if $[\alpha =\text{'-'}] \wedge [\mathsf{SPS.Verify}(\mathrm{VK}_{\mathrm{SPS},F}, m_{\mathrm{IN}}, \sigma_{\mathrm{SPS,IN}}) = 1]$, set $\alpha =$'F'.
   (f) If $\alpha =$'-', output $\perp$.
2. If $\mathsf{SSB.Verify}(\mathrm{HK}, h, i, \mathrm{SYM}_{\mathrm{IN}}, \pi_{\mathrm{SSB}}) = 0$, output $\perp$.
3. (a) Compute $w_{\mathrm{OUT}} = \mathsf{ACC.Update}(\mathrm{PP}_{\mathrm{ACC}}, w_{\mathrm{IN}}, \mathrm{SYM}_{\mathrm{IN}}, i, \mathrm{AUX})$. If $w_{\mathrm{OUT}} = \perp$, output $\perp$.
   (b) Compute $v_{\mathrm{OUT}} = \mathsf{ITR.Iterate}(\mathrm{PP}_{\mathrm{ITR}}, v_{\mathrm{IN}}, (\mathrm{ST}, w_{\mathrm{IN}}, 0))$.
4. (a) Compute $r'_{\mathrm{SPS},E} = \mathcal{F}(K_{\mathrm{SPS},E}\{(h^*,0)\},(h,i+1))$, $(\mathrm{SK}'_{\mathrm{SPS},E}, \mathrm{VK}'_{\mathrm{SPS},E}, \mathrm{VK}'_{\text{SPS-REJ},E}) = \mathsf{SPS.Setup}(1^\lambda; r'_{\mathrm{SPS},E})$.
   (b) Compute $r'_{\mathrm{SPS},F} = \mathcal{F}(K_{\mathrm{SPS},F},(h,i+1))$, $(\mathrm{SK}'_{\mathrm{SPS},F}, \mathrm{VK}'_{\mathrm{SPS},F}, \mathrm{VK}'_{\text{SPS-REJ},F}) = \mathsf{SPS.Setup}(1^\lambda; r'_{\mathrm{SPS},F})$.
   (c) Set $m_{\mathrm{OUT}} = (v_{\mathrm{OUT}}, \mathrm{ST}, w_{\mathrm{OUT}}, 0)$. If $[(h,i) = (h^*,0)] \wedge [m_{\mathrm{IN}} = m_{0,0}]$, compute $\sigma_{\mathrm{SPS,OUT}} = \mathsf{SPS.Sign}(\mathrm{SK}'_{\mathrm{SPS},E}, m_{\mathrm{OUT}})$.
   Else if $[(h,i) = (h^*,0)] \wedge [m_{\mathrm{IN}} \neq m_{0,0}]$, compute $\sigma_{\mathrm{SPS,OUT}} = \mathsf{SPS.Sign}(\mathrm{SK}'_{\mathrm{SPS},F}, m_{\mathrm{OUT}})$.
   Else if $i < \ell^*$, compute $\sigma_{\mathrm{SPS,OUT}} = \mathsf{SPS.Sign}(\mathrm{SK}'_{\mathrm{SPS},\alpha}, m_{\mathrm{OUT}})$.
   Else, compute $\sigma_{\mathrm{SPS,OUT}} = \mathsf{SPS.Sign}(\mathrm{SK}'_{\mathrm{SPS},E} m_{\mathrm{OUT}})$.
5. Output $(w_{\mathrm{OUT}}, v_{\mathrm{OUT}}, \sigma_{\mathrm{SPS,OUT}})$.

Fig. 31 $\mathsf{Accumulate.Prog}^{(2,2)}$

$\mathsf{Hyb}_{0,\nu-1,3\text{-VI}}$: In this experiment, in response to the ν^{th} signing key query of $\mathcal{A}$ corresponding to TM $M^{(\nu)} \in \mathbb{M}_\lambda$ with $M^{(\nu)}(x^*) = 0$, $\mathcal{B}$ proceeds as follows:

1. It first generates all the full and punctured PPRF keys as well as the public parameters for the positional accumulator and its iterator as in $\mathsf{Hyb}_{0,\nu-1,3\text{-V}}$.
2. Next, it forms $(\mathrm{SK}^{(\nu,0)}_{\mathrm{SPS},G}, \mathrm{VK}^{(\nu,0)}_{\mathrm{SPS},G}, \mathrm{VK}^{(\nu,0)}_{\text{SPS-REJ},G}) \xleftarrow{\$} \mathsf{SPS.Setup}(1^\lambda)$, sets $m_{0,0}^{(\nu)} = (v_0^{(\nu)}, q_0^{(\nu)}, w_0^{(\nu)}, 0)$, and computes $\sigma^{(\nu,0)}_{\mathrm{SPS},G} = \mathsf{SPS.Sign}(\mathrm{SK}^{(\nu,0)}_{\mathrm{SPS},G}, m^{(\nu)}_{0,0})$.

3. It provides $\mathcal{A}$ with the signing key
$\mathrm{SK}_{\mathrm{ABS}}\{M^{(\nu)}\} =$
$$\left(\begin{array}{l}\mathrm{HK}, \mathrm{PP}_{\mathrm{ACC}}^{(\nu)}, w_0^{(\nu)}, \mathrm{STORE}_0^{(\nu)}, \mathrm{PP}_{\mathrm{ITR}}^{(\nu)}, v_0^{(\nu)}, \\ \mathcal{IO}(\mathsf{Init\text{-}SPS.Prog}^{(1)}[q_0^{(\nu)}, w_0^{(\nu)}, v_0^{(\nu)}, K_{\mathrm{SPS},E}^{(\nu)}\{(h^*,0)\}, \sigma_{\mathrm{SPS},G}^{(\nu,0)}, h^*]), \\ \mathcal{IO}(\mathsf{Accumulate.Prog}^{(2,2)}[n_{\text{SSB-BLK}} = 2^\lambda, \mathrm{HK}, \mathrm{PP}_{\mathrm{ACC}}^{(\nu)}, \mathrm{PP}_{\mathrm{ITR}}^{(\nu)}, K_{\mathrm{SPS},E}^{(\nu)}\{(h^*,0)\}, \\ \qquad K_{\mathrm{SPS},F}^{(\nu)}, \mathrm{VK}_{\mathrm{SPS},G}^{(\nu,0)}, m_{0,0}^{(\nu)}, h^*, \ell^*]), \\ \mathcal{IO}(\mathsf{Change\text{-}SPS.Prog}^{(2)}[K_{\mathrm{SPS},A}^{(\nu)}, K_{\mathrm{SPS},B}^{(\nu)}, K_{\mathrm{SPS},E}^{(\nu)}\{(h^*,0)\}, K_{\mathrm{SPS},F}^{(\nu)}, h^*, \ell^*]), \\ \mathcal{IO}(\mathsf{Constrained\text{-}Key.Prog}_{\mathrm{ABS}}^{(1)}[M^{(\nu)}, T = 2^\lambda, t^{*(\nu)}, \mathrm{PP}_{\mathrm{ACC}}^{(\nu)}, \mathrm{PP}_{\mathrm{ITR}}^{(\nu)}, K, \\ \qquad K_1^{(\nu)}, \ldots, K_\lambda^{(\nu)}, K_{\mathrm{SPS},A}^{(\nu)}, K_{\mathrm{SPS},B}^{(\nu)}, h^*, \ell^*])\end{array}\right).$$

The remaining portion of the experiment is identical to $\mathsf{Hyb}_{0,\nu-1,3\text{-V}}$.

$\mathsf{Hyb}_{0,\nu-1,3\text{-VII}}$: In this experiment, while constructing the ν^{th} signing key queried by $\mathcal{A}$ corresponding to TM $M^{(\nu)} \in \mathbb{M}_\lambda$ with $M^{(\nu)}(x^*) = 0$, $\mathcal{B}$ generates everything just as in $\mathsf{Hyb}_{0,\nu-1,3\text{-VI}}$ except that it computes $r_{\mathrm{SPS},G}^{(\nu,0)} = \mathcal{F}(K_{\mathrm{SPS},E}^{(\nu)}, (h^*,0))$, forms $(\mathrm{SK}_{\mathrm{SPS},G}^{(\nu,0)}, \mathrm{VK}_{\mathrm{SPS},G}^{(\nu,0)}, \mathrm{VK}_{\mathrm{SPS\text{-}REJ},G}^{(\nu,0)}) = \mathsf{SPS.Setup}(1^\lambda;\ r_{\mathrm{SPS},G}^{(\nu,0)})$, and provides $\mathcal{A}$ with the signing key
$\mathrm{SK}_{\mathrm{ABS}}\{M^{(\nu)}\} =$
$$\left(\begin{array}{l}\mathrm{HK}, \mathrm{PP}_{\mathrm{ACC}}^{(\nu)}, w_0^{(\nu)}, \mathrm{STORE}_0^{(\nu)}, \mathrm{PP}_{\mathrm{ITR}}^{(\nu)}, v_0^{(\nu)}, \\ \mathcal{IO}(\mathsf{Init\text{-}SPS.Prog}^{(1)}[q_0^{(\nu)}, w_0^{(\nu)}, v_0^{(\nu)}, K_{\mathrm{SPS},E}^{(\nu)}\{(h^*,0)\}, \sigma_{\mathrm{SPS},G}^{(\nu,0)}, h^*]), \\ \mathcal{IO}(\mathsf{Accumulate.Prog}^{(2,2)}[n_{\text{SSB-BLK}} = 2^\lambda, \mathrm{HK}, \mathrm{PP}_{\mathrm{ACC}}^{(\nu)}, \mathrm{PP}_{\mathrm{ITR}}^{(\nu)}, K_{\mathrm{SPS},E}^{(\nu)}\{(h^*,0)\}, \\ \qquad K_{\mathrm{SPS},F}^{(\nu)}, \mathrm{VK}_{\mathrm{SPS},G}^{(\nu,0)}, m_{0,0}^{(\nu)}, h^*, \ell^*]), \\ \mathcal{IO}(\mathsf{Change\text{-}SPS.Prog}^{(2)}[K_{\mathrm{SPS},A}^{(\nu)}, K_{\mathrm{SPS},B}^{(\nu)}, K_{\mathrm{SPS},E}^{(\nu)}\{(h^*,0)\}, K_{\mathrm{SPS},F}^{(\nu)}, h^*, \ell^*]), \\ \mathcal{IO}(\mathsf{Constrained\text{-}Key.Prog}_{\mathrm{ABS}}^{(1)}[M^{(\nu)}, T = 2^\lambda, t^{*(\nu)}, \mathrm{PP}_{\mathrm{ACC}}^{(\nu)}, \mathrm{PP}_{\mathrm{ITR}}^{(\nu)}, K, \\ \qquad K_1^{(\nu)}, \ldots, K_\lambda^{(\nu)}, K_{\mathrm{SPS},A}^{(\nu)}, K_{\mathrm{SPS},B}^{(\nu)}, h^*, \ell^*])\end{array}\right).$$
The rest of the experiment is analogous to $\mathsf{Hyb}_{0,\nu-1,3\text{-VI}}$.

$\mathsf{Hyb}_{0,\nu-1,3\text{-VIII}}$: This experiment corresponds to $\mathsf{Hyb}_{0,\nu-1,3,0}$.

Analysis

Let $\mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,3\text{-}\vartheta)}(\lambda)$ represents the advantage of the adversary $\mathcal{A}$, i.e., the absolute difference between $1/2$ and $\mathcal{A}$'s probability of correctly guessing the random bit selected by the challenger $\mathcal{B}$, in $\mathsf{Hyb}_{0,\nu-1,3\text{-}\vartheta}$, for $\vartheta \in \{\mathrm{I}, \ldots, \mathrm{VIII}\}$. Clearly, $\mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,3)}(\lambda) \equiv \mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,3\text{-I})}(\lambda)$ and $\mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,3,0)}(\lambda) \equiv \mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,3\text{-VIII})}(\lambda)$. Therefore, we have
$$|\mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,3)}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,3,0)}(\lambda)| \leq \sum_{\vartheta=\mathrm{II}}^{\mathrm{VIII}} |\mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,3\text{-}(\vartheta-\mathrm{I}))}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,3\text{-}\vartheta)}(\lambda)|. \tag{C.4}$$
Claims C.9–C.15 below will justify that the RHS of Eq. (C.4) is negligible and hence Lemma C.4 follows.

Claim C.9 *Assuming $\mathcal{IO}$ is a secure indistinguishability obfuscator for $\mathsf{P/poly}$ and $\mathcal{F}$ satisfies the correctness under puncturing property, for any PPT adversary $\mathcal{A}$, for*

any security parameter λ, $|\mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,3\text{-}I)}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,3\text{-}II)}(\lambda)| \leq \mathsf{negl}(\lambda)$ *for some negligible function* negl.

Proof The difference between $\mathsf{Hyb}_{0,\nu-1,3\text{-I}}$ and $\mathsf{Hyb}_{0,\nu-1,3\text{-II}}$ is the following: In $\mathsf{Hyb}_{0,\nu-1,3\text{-I}}$, $\mathcal{B}$ includes the programs $\mathcal{IO}(P_0)$ and $\mathcal{IO}(P_0')$ within the ν^{th} signing key returned to $\mathcal{A}$, while in $\mathsf{Hyb}_{0,\nu-1,3\text{-II}}$, $\mathcal{B}$ includes the programs $\mathcal{IO}(P_1)$ and $\mathcal{IO}(p_1')$ instead, where

- $P_0 = \mathsf{Init\text{-}SPS.Prog}[q_0^{(\nu)}, w_0^{(\nu)}, v_0^{(\nu)}, K_{\text{SPS},E}^{(\nu)}]$ (Fig. **2**),
- $P_0' = \mathsf{Accumulate.Prog}^{(2)}[n_{\text{SSB-BLK}} = 2^\lambda, \text{HK}, \text{PP}_{\text{ACC}}^{(\nu)}, \text{PP}_{\text{ITR}}^{(\nu)}, K_{\text{SPS},E}^{(\nu)}, K_{\text{SPS},F}^{(\nu)}, h^*, \ell^*]$ (Fig. **13**),
- $P_1 = \mathsf{Init\text{-}SPS.Prog}^{(1)}[q_0^{(\nu)}, w_0^{(\nu)}, v_0^{(\nu)}, K_{\text{SPS},E}^{(\nu)}\{(h^*,0)\}, \sigma_{\text{SPS},G}^{(\nu,0)}, h^*]$ (Fig. **29**),
- $P_1' = \mathsf{Accumulate.Prog}^{(2,1)}[n_{\text{SSB-BLK}} = 2^\lambda, \text{HK}, \text{PP}_{\text{ACC}}^{(\nu)}, \text{PP}_{\text{ITR}}^{(\nu)}, K_{\text{SPS},E}^{(\nu)}\{(h^*,0)\}, K_{\text{SPS},F}^{(\nu)}, \text{VK}_{\text{SPS},G}^{(\nu,0)}, h^*, \ell^*]$ (Fig. **30**).

Now observe that the programs P_0 and P_1 are functionally equivalent since by the correctness under puncturing property of the PPRF $\mathcal{F}$, the PPRF output remains the same at all non-punctured points and at the point of puncturing, i.e., $(h^*, 0)$, the correct signature is hardwired in the program P_1. Similarly, P_0' and P_1' are also functionally equivalent by the correctness under puncturing property of $\mathcal{F}$ and the fact that at the point of puncturing i.e., $(h^*, 0)$ the correct verification key is hardwired into the program P_1'.

Therefore, by the security of $\mathcal{IO}$, Claim C.9 follows. □

Claim C.10 *Assuming* $\mathcal{F}$ *is a secure puncturable pseudorandom function, for any* PPT *adversary* $\mathcal{A}$*, for any security parameter* λ, $|\mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,3\text{-}II)}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,3\text{-}III)}(\lambda)| \leq \mathsf{negl}(\lambda)$ *for some negligible function* negl.

Proof Suppose there exists a PPT adversary $\mathcal{A}$ for which $|\mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,3\text{-II})}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,3\text{-III})}(\lambda)|$ is non-negligible. We construct a PPT adversary $\mathcal{B}$ that breaks the selective pseudorandomness of the PPRF $\mathcal{F}$ using $\mathcal{A}$ as a sub-routine. The description of $\mathcal{B}$ follows:

- $\mathcal{B}$ initializes $\mathcal{A}$ on input 1^λ and receives a challenge signing attribute string $x^* = x_0^* \ldots x_{\ell^*-1}^* \in \mathcal{U}_{\text{ABS}}$ with $|x^*| = \ell^*$ from $\mathcal{A}$.
- Upon receiving x^*, $\mathcal{B}$ proceeds as follows:
  1. $\mathcal{B}$ generates $\text{HK} \xleftarrow{\$} \mathsf{SSB.Gen}(1^\lambda, n_{\text{SSB-BLK}} = 2^\lambda, i^* = 0)$ and computes $h^* = \mathcal{H}_{\text{HK}}(x^*)$.
  2. Then, $\mathcal{B}$ selects a PPRF key $K \xleftarrow{\$} \mathcal{F}.\mathsf{Setup}(1^\lambda)$.
  3. After that, $\mathcal{B}$ computes $r_{\text{SIG}}^* = \mathcal{F}(K, (h^*, \ell^*))$ followed by $(\widehat{\text{SK}}_{\text{SIG}}^*, \widehat{\text{VK}}_{\text{SIG}}^*) = \mathsf{SIG.Setup}(1^\lambda; r_{\text{SIG}}^*)$.
  4. $\mathcal{B}$ returns the public parameters $\text{PP}_{\text{ABS}} = (\text{HK}, \mathcal{IO}(\mathsf{Verify.Prog}_{\text{ABS}}[K]))$ to $\mathcal{A}$.
- For $\eta \in [\hat{q}_{\text{KEY}}]$, in response to the η^{th} signing key query of $\mathcal{A}$ corresponding to TM $M^{(\eta)} \in \mathbb{M}_\lambda$ with $M^{(\eta)}(x^*) = 0$, if $\eta \neq \nu$, then $\mathcal{B}$ proceeds exactly as in $\mathsf{Hyb}_{0,\nu-1,3\text{-II}}$, while if $\eta = \nu$, then $\mathcal{B}$ proceeds as follows:
  1. $\mathcal{B}$ first selects PPRF keys $K_1^{(\nu)}, \ldots, K_\lambda^{(\nu)}, K_{\text{SPS},A}^{(\nu)}, K_{\text{SPS},B}^{(\nu)}, K_{\text{SPS},F}^{(\nu)} \xleftarrow{\$} \mathcal{F}.\mathsf{Setup}(1^\lambda)$.

2. Next, it generates $(\mathrm{PP}_{\mathrm{ACC}}^{(\nu)}, w_0^{(\nu)}, \mathrm{STORE}_0^{(\nu)}) \xleftarrow{\$} \mathsf{ACC.Setup}(1^\lambda, n_{\text{ACC-BLK}} = 2^\lambda)$ and $(\mathrm{PP}_{\mathrm{ITR}}^{(\nu)}, v_0^{(\nu)}) \xleftarrow{\$} \mathsf{ITR.Setup}(1^\lambda, n_{\mathrm{ITR}} = 2^\lambda)$.
3. Then, $\mathcal{B}$ sends $(h^*, 0)$ as the challenge input to its PPRF selective pseudorandomness challenger $\mathcal{C}$ and receives back a punctured PPRF key $K^*\{(h^*,0)\}$ and a value $r^* \in \mathcal{Y}_{\mathrm{PPRF}}$, where either $r^* = \mathcal{F}(K^*, (h^*, 0))$ or $r^* \xleftarrow{\$} \mathcal{Y}_{\mathrm{PPRF}}$. $\mathcal{B}$ *implicitly* views the key K^* as the key $K_{\mathrm{SPS},E}^{(\nu)}$.
4. $\mathcal{B}$ generates $(\mathrm{SK}_{\mathrm{SPS},G}^{(\nu,0)}, \mathrm{VK}_{\mathrm{SPS},G}^{(\nu,0)}, \mathrm{VK}_{\text{SPS-REJ},G}^{(\nu,0)}) = \mathsf{SPS.Setup}(1^\lambda;\ r^*)$.
5. Then, $\mathcal{B}$ sets $m_{0,0}^{(\nu)} = (v_0^{(\nu)}, q_0^{(\nu)}, w_0^{(\nu)}, 0)$ and computes $\sigma_{\mathrm{SPS},G}^{(\nu,0)} = \mathsf{SPS.Sign}(\mathrm{SK}_{\mathrm{SPS},G}^{(\nu,0)}, m_{0,0}^{(\nu)})$.
6. $\mathcal{B}$ gives $\mathcal{A}$ the signing key $\mathrm{SK}_{\mathrm{ABS}}\{M^{(\nu)}\} =$

$$\left(\begin{array}{l} \mathrm{HK}, \mathrm{PP}_{\mathrm{ACC}}^{(\nu)}, w_0^{(\nu)}, \mathrm{STORE}_0^{(\nu)}, \mathrm{PP}_{\mathrm{ITR}}^{(\nu)}, v_0^{(\nu)}, \\ \mathcal{IO}(\mathsf{Init\text{-}SPS.Prog}^{(1)}[q_0^{(\nu)}, w_0^{(\nu)}, v_0^{(\nu)}, K^*\{(h^*,0)\}, \sigma_{\mathrm{SPS},G}^{(\nu,0)}, h^*]), \\ \mathcal{IO}(\mathsf{Accumulate.Prog}^{(2,1)}[n_{\text{SSB-BLK}} = 2^\lambda, \mathrm{HK}, \mathrm{PP}_{\mathrm{ACC}}^{(\nu)}, \mathrm{PP}_{\mathrm{ITR}}^{(\nu)}, K^*\{(h^*,0)\}, \\ \qquad K_{\mathrm{SPS},F}^{(\nu)}, \mathrm{VK}_{\mathrm{SPS},G}^{(\nu,0)}, h^*, \ell^*]), \\ \mathcal{IO}(\mathsf{Change\text{-}SPS.Prog}^{(2)}[K_{\mathrm{SPS},A}^{(\nu)}, K_{\mathrm{SPS},B}^{(\nu)}, K^*\{(h^*,0)\}, K_{\mathrm{SPS},F}^{(\nu)}, h^*, \ell^*]), \\ \mathcal{IO}(\mathsf{Constrained\text{-}Key.Prog}_{\mathrm{ABS}}^{(1)}[M^{(\nu)}, T = 2^\lambda, t^{*(\nu)}, \mathrm{PP}_{\mathrm{ACC}}^{(\nu)}, \mathrm{PP}_{\mathrm{ITR}}^{(\nu)}, K, \\ \qquad K_1^{(\nu)}, \ldots, K_\lambda^{(\nu)}, K_{\mathrm{SPS},A}^{(\nu)}, K_{\mathrm{SPS},B}^{(\nu)}, h^*, \ell^*]) \end{array}\right).$$

- For $\theta \in [\hat{q}_{\mathrm{SIGN}}]$, in reply to the θ^{th} signature query of $\mathcal{A}$ on message $\mathsf{msg}^{(\theta)} \in \mathcal{M}_{\mathrm{ABS}}$ under attribute string x^*, $\mathcal{B}$ computes $\sigma_{\mathrm{SIG}}^{(\theta)} \xleftarrow{\$} \mathsf{SIG.Sign}(\widehat{\mathrm{SK}}_{\mathrm{SIG}}^*, \mathsf{msg}^{(\theta)})$ and provides $\mathcal{A}$ with $\sigma_{\mathrm{ABS}}^{(\theta)} = (\widehat{\mathrm{VK}}_{\mathrm{SIG}}^*, \sigma_{\mathrm{SIG}}^{(\theta)})$.
- Finally, $\mathcal{A}$ output a signature σ_{ABS}^* on some message msg^* under attribute string x^*. $\mathcal{B}$ outputs $\hat{b}' = 1$ as its guess bit in its PPRF selective pseudorandomness experiment if $\mathcal{A}$ wins, i.e., if $\mathsf{ABS.Verify}(\mathrm{PP}_{\mathrm{ABS}}, x^*, \mathsf{msg}^*, \sigma_{\mathrm{ABS}}^*) = 1$ and $\mathsf{msg}^* \neq \mathsf{msg}^{(\theta)}$ for any $\theta \in [\hat{q}_{\mathrm{SIGN}}]$. Otherwise, $\mathcal{B}$ outputs $\hat{b}' = 0$ in its PPRF selective pseudorandomness experiment.

Note that if $r^* = \mathcal{F}(K^*, (h^*, 0))$, then $\mathcal{B}$ perfectly simulates $\mathsf{Hyb}_{0,\nu-1,3\text{-II}}$. On the other hand, if $r^* \xleftarrow{\$} \mathcal{Y}_{\mathrm{PPRF}}$, the $\mathcal{B}$ perfectly simulates $\mathsf{Hyb}_{0,\nu-1,3\text{-III}}$. This completes the proof of Claim C.10. □

Claim C.11 *Assuming* SPS *is a splittable signature scheme satisfying* '$\mathrm{VK}_{\text{SPS-ONE}}$' *indistinguishability, for any* PPT *adversary* $\mathcal{A}$*, for any security parameter* λ*,* $|\mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,3\text{-}III)}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,3\text{-}IV)}(\lambda)| \leq \mathsf{negl}(\lambda)$ *for some negligible function* negl.

Proof Suppose there exists a PPT adversary $\mathcal{A}$ for which $|\mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,3\text{-III})}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,3\text{-IV})}(\lambda)|$ is non-negligible. Below we construct a PPT adversary $\mathcal{B}$ that breaks the $\mathrm{VK}_{\text{SPS-ONE}}$ indistinguishability of SPS using $\mathcal{A}$ as a sub-routine.

- $\mathcal{B}$ initializes $\mathcal{A}$ on input 1^λ and receives a challenge signing attribute string $x^* = x_0^* \ldots x_{\ell^*-1}^* \in \mathcal{U}_{\mathrm{ABS}}$ with $|x^*| = \ell^*$ from $\mathcal{A}$.
- Upon receiving x^*, $\mathcal{B}$ proceeds as follows:
  1. $\mathcal{B}$ generates $\mathrm{HK} \xleftarrow{\$} \mathsf{SSB.Gen}(1^\lambda, n_{\text{SSB-BLK}} = 2^\lambda, i^* = 0)$ and computes $h^* = \mathcal{H}_{\mathrm{HK}}(x^*)$.
  2. Then, $\mathcal{B}$ selects a PPRF key $K \xleftarrow{\$} \mathcal{F}.\mathsf{Setup}(1^\lambda)$.

3. After that, $\mathcal{B}$ computes $r^*_{\text{SIG}} = \mathcal{F}(K, (h^*, \ell^*))$ followed by $(\widehat{\text{SK}}^*_{\text{SIG}}, \widehat{\text{VK}}^*_{\text{SIG}}) = \mathsf{SIG.Setup}(1^\lambda;\ r^*_{\text{SIG}})$.
4. $\mathcal{B}$ returns the public parameters $\text{PP}_{\text{ABS}} = (\text{HK}, \mathcal{IO}(\mathsf{Verify.Prog}_{\text{ABS}}[K]))$ to $\mathcal{A}$.

- For $\eta \in [\hat{q}_{\text{KEY}}]$, in response to the η^{th} signing key query of $\mathcal{A}$ corresponding to $\mathsf{TM}\ M^{(\eta)} \in \mathbb{M}_\lambda$ with $M^{(\eta)}(x^*) = 0$, if $\eta \neq \nu$, then $\mathcal{B}$ proceeds exactly as in $\mathsf{Hyb}_{0,\nu-1,3\text{-III}}$, while if $\eta = \nu$, then $\mathcal{B}$ proceeds as follows:
  1. $\mathcal{B}$ first selects PPRF keys $K^{(\nu)}_1, \ldots, K^{(\nu)}_\lambda, K^{(\nu)}_{\text{SPS},A}, K^{(\nu)}_{\text{SPS},B}, K^{(\nu)}_{\text{SPS},E}, K^{(\nu)}_{\text{SPS},F} \xleftarrow{\$} \mathcal{F}.\mathsf{Setup}(1^\lambda)$.
  2. Next, it generates $(\text{PP}^{(\nu)}_{\text{ACC}}, w^{(\nu)}_0, \text{STORE}^{(\nu)}_0) \xleftarrow{\$} \mathsf{ACC.Setup}(1^\lambda, n_{\text{ACC-BLK}} = 2^\lambda)$ and $(\text{PP}^{(\nu)}_{\text{ITR}}, v^{(\nu)}_0) \xleftarrow{\$} \mathsf{ITR.Setup}(1^\lambda, n_{\text{ITR}} = 2^\lambda)$.
  3. Then, $\mathcal{B}$ creates the punctured PPRF key $K^{(\nu)}_{\text{SPS},E}\{(h^*, 0)\} \xleftarrow{\$} \mathcal{F}.\mathsf{Puncture}(K^{(\nu)}_{\text{SPS},E}, (h^*, 0))$.
  4. After that, $\mathcal{B}$ sends $m^{(\nu)}_{0,0} = (v^{(\nu)}_0, q^{(\nu)}_0, w^{(\nu)}_0, 0)$ as the challenge message to its SPS $\text{VK}_{\text{SPS-ONE}}$ indistinguishability challenger $\mathcal{C}$ and receives back a signature-verification key pair $(\sigma_{\text{SPS-ONE},m^{(\nu)}_{0,0}}, \text{VK})$, where VK is either a normal verification key VK_{SPS} or a one verification key $\text{VK}_{\text{SPS-ONE}}$ for the message $m^{(\nu)}_{0,0}$.
  5. $\mathcal{B}$ gives $\mathcal{A}$ the signing key
  $$\text{SK}_{\text{ABS}}\{M^{(\nu)}\} = \left(\begin{array}{l} \text{HK}, \text{PP}^{(\nu)}_{\text{ACC}}, w^{(\nu)}_0, \text{STORE}^{(\nu)}_0, \text{PP}^{(\nu)}_{\text{ITR}}, v^{(\nu)}_0, \\ \mathcal{IO}(\mathsf{Init\text{-}SPS.Prog}^{(1)}[q^{(\nu)}_0, w^{(\nu)}_0, v^{(\nu)}_0, K^{(\nu)}_{\text{SPS},E}\{(h^*,0)\}, \sigma_{\text{SPS-ONE},m^{(\nu)}_{0,0}}, h^*]), \\ \mathcal{IO}(\mathsf{Accumulate.Prog}^{(2,1)}[n_{\text{SSB-BLK}} = 2^\lambda, \text{HK}, \text{PP}^{(\nu)}_{\text{ACC}}, \text{PP}^{(\nu)}_{\text{ITR}}, \\ \qquad K^{(\nu)}_{\text{SPS},E}\{(h^*,0)\}, K^{(\nu)}_{\text{SPS},F}, \text{VK}, h^*, \ell^*]), \\ \mathcal{IO}(\mathsf{Change\text{-}SPS.Prog}^{(2)}[K^{(\nu)}_{\text{SPS},A}, K^{(\nu)}_{\text{SPS},B}, K^{(\nu)}_{\text{SPS},E}\{(h^*,0)\}, K^{(\nu)}_{\text{SPS},F}, \\ \qquad h^*, \ell^*]), \\ \mathcal{IO}(\mathsf{Constrained\text{-}Key.Prog}^{(1)}_{\text{ABS}}[M^{(\nu)}, T = 2^\lambda, t^{*(\nu)}, \text{PP}^{(\nu)}_{\text{ACC}}, \text{PP}^{(\nu)}_{\text{ITR}}, K, \\ \qquad K^{(\nu)}_1, \ldots, K^{(\nu)}_\lambda, K^{(\nu)}_{\text{SPS},A}, K^{(\nu)}_{\text{SPS},B}, h^*, \ell^*]) \end{array}\right).$$

- For $\theta \in [\hat{q}_{\text{SIGN}}]$, in reply to the θ^{th} signature query of $\mathcal{A}$ on message $\mathsf{msg}^{(\theta)} \in \mathcal{M}_{\text{ABS}}$ under attribute string x^*, $\mathcal{B}$ computes $\sigma^{(\theta)}_{\text{SIG}} \xleftarrow{\$} \mathsf{SIG.Sign}(\widehat{\text{SK}}^*_{\text{SIG}}, \mathsf{msg}^{(\theta)})$ and provides $\mathcal{A}$ with $\sigma^{(\theta)}_{\text{ABS}} = (\widehat{\text{VK}}^*_{\text{SIG}}, \sigma^{(\theta)}_{\text{SIG}})$.
- Finally, $\mathcal{A}$ output a signature σ^*_{ABS} on some message msg^* under attribute string x^*. $\mathcal{B}$ outputs $\hat{b}' = 1$ as its guess bit in its $\text{VK}_{\text{SPS-ONE}}$ indistinguishability experiment if $\mathcal{A}$ wins, i.e., if $\mathsf{ABS.Verify}(\text{PP}_{\text{ABS}}, x^*, \mathsf{msg}^*, \sigma^*_{\text{ABS}}) = 1$ and $\mathsf{msg}^* \neq \mathsf{msg}^{(\theta)}$ for any $\theta \in [\hat{q}_{\text{SIGN}}]$. Otherwise, $\mathcal{B}$ outputs $\hat{b}' = 0$ in its $\text{VK}_{\text{SPS-ONE}}$ indistinguishability experiment.

Notice that if $\text{VK} = \text{VK}_{\text{SPS}}$, then $\mathcal{B}$ perfectly simulates $\mathsf{Hyb}_{0,\nu-1,3\text{-III}}$. On the other hand, if $\text{VK} = \text{VK}_{\text{SPS-ONE}}$, then $\mathcal{B}$ perfectly simulates $\mathsf{Hyb}_{0,\nu-1,3\text{-IV}}$. This completes the proof of Claim C.11. □

Claim C.12 *Assuming $\mathcal{IO}$ is a secure indistinguishability obfuscator for $\mathsf{P/poly}$, for any PPT adversary $\mathcal{A}$, for any security parameter λ, $|\mathsf{Adv}^{(0,\nu-1,3\text{-}IV)}_{\mathcal{A}}(\lambda) - \mathsf{Adv}^{(0,\nu-1,3\text{-}V)}_{\mathcal{A}}(\lambda)| \leq \mathsf{negl}(\lambda)$ for some negligible function negl.*

Proof The difference between $\mathsf{Hyb}_{0,\nu-1,3\text{-IV}}$ and $\mathsf{Hyb}_{0,\nu-1,3\text{-V}}$ is the following: In $\mathsf{Hyb}_{0,\nu-1,3\text{-IV}}$, $\mathcal{B}$ includes the program $\mathcal{IO}(P_0)$ within the ν^{th} signing key returned to $\mathcal{A}$, while in $\mathsf{Hyb}_{0,\nu-1,3\text{-V}}$, $\mathcal{B}$ includes the program $\mathcal{IO}(P_1)$ instead, where

- $P_0 = \mathsf{Accumulate.Prog}^{(2,1)}[n_{\text{SSB-BLK}} = 2^\lambda, \text{HK}, \text{PP}^{(\nu)}_{\text{ACC}}, \text{PP}^{(\nu)}_{\text{ITR}}, K^{(\nu)}_{\text{SPS},E}\{(h^*,0)\}, K^{(\nu)}_{\text{SPS},F}, \text{VK}^{(\nu,0)}_{\text{SPS-ONE},G}, h^*, \ell^*]$ (Fig. **30**),
- $P_1 = \mathsf{Accumulate.Prog}^{(2,2)}[n_{\text{SSB-BLK}} = 2^\lambda, \text{HK}, \text{PP}^{(\nu)}_{\text{ACC}}, \text{PP}^{(\nu)}_{\text{ITR}}, K^{(\nu)}_{\text{SPS},E}\{(h^*,0)\}, K^{(\nu)}_{\text{SPS},F}, \text{VK}^{(\nu,0)}_{\text{SPS-ONE},G}, m^{(\nu)}_{0,0}, h^*, \ell^*]$ (Fig. **31**).

Observe that the only inputs for which the programs P_0 and P_1 can possibly differ are those corresponding to $(h,i) = (h^*,0)$. However, the verification key hardwired in both the programs is $\text{VK}^{(\nu,0)}_{\text{SPS-ONE},G}$ which only accepts signature for $m_{\text{IN}} = m^{(\nu)}_{0,0}$ by the correctness [Properties (i), (iii) and (v)]. This ensures that for inputs corresponding to $(h^*,0)$, if $m_{\text{IN}} = m^{(\nu)}_{0,0}$ both the programs output an 'E' type signature, else, both output $\perp$. Thus, P_0 and P_1 are functionally equivalent.

Therefore, by the security of $\mathcal{IO}$, Claim C.12 follows. □

Claim C.13 *Assuming* SPS *is a splittable signature scheme satisfying '*$\text{VK}_{\text{SPS-ONE}}$ *indistinguishability', for any* PPT *adversary* $\mathcal{A}$*, for any security parameter* λ*,* $|\mathsf{Adv}^{(0,\nu-1,3\text{-}V)}_{\mathcal{A}}(\lambda) - \mathsf{Adv}^{(0,\nu-1,3\text{-}VI)}_{\mathcal{A}}(\lambda)| \leq \mathsf{negl}(\lambda)$ *for some negligible function* negl.

Proof The proof of Claim C.13 is similar to that of Claim C.11 with some readily identifiable modifications. □

Claim C.14 *Assuming* $\mathcal{F}$ *is a secure puncturable pseudorandom function, for any* PPT *adversary* $\mathcal{A}$*, for any security parameter* λ*,* $|\mathsf{Adv}^{(0,\nu-1,3\text{-}VI)}_{\mathcal{A}}(\lambda) - \mathsf{Adv}^{(0,\nu-1,3\text{-}VII)}_{\mathcal{A}}(\lambda)| \leq \mathsf{negl}(\lambda)$ *for some negligible function* negl.

Proof The proof of Claim C.14 is analogous to that of Claim C.10 with some appropriate changes that are easy to find out. □

Claim C.15 *Assuming* $\mathcal{IO}$ *is a secure indistinguishability obfuscator for* P/poly *and* $\mathcal{F}$ *satisfies the correctness under puncturing property, for any* PPT *adversary* $\mathcal{A}$*, for any security parameter* λ*,* $|\mathsf{Adv}^{(0,\nu-1,3\text{-}VII)}_{\mathcal{A}}(\lambda) - \mathsf{Adv}^{(0,\nu-1,3\text{-}VIII)}_{\mathcal{A}}(\lambda)| \leq \mathsf{negl}(\lambda)$ *for some negligible function* negl.

Proof The proof of Claim C.15 is analogous to that of Claim C.9. □

□

Lemma C.5 *Assuming* $\mathcal{IO}$ *is a secure indistinguishability obfuscator for* P/poly*,* SSB *is a somewhere statistically binding hash function,* ACC *is a positional accumulator satisfying 'indistinguishability of write setup' and 'write enforcing' properties, as well as* ITR *is a secure cryptographic iterator, for any* PPT *adversary* $\mathcal{A}$*, for any security parameter* λ*,* $|\mathsf{Adv}^{(0,\nu-1,3,\iota)}_{\mathcal{A}}(\lambda) - \mathsf{Adv}^{(0,\nu-1,3,\iota')}_{\mathcal{A}}(\lambda)| \leq \mathsf{negl}(\lambda)$ *for some negligible function* negl.

Proof To prove Lemma C.5, we introduce the following sequence of intermediate hybrid experiments between $\mathsf{Hyb}_{0,\nu-1,3,\iota}$ and $\mathsf{Hyb}_{0,\nu-1,3,\iota'}$:

Sequence of Intermediate Hybrids between $\mathsf{Hyb}_{0,\nu-1,3,\iota}$ and $\mathsf{Hyb}_{0,\nu-1,3,\iota'}$

$\mathbf{Hyb_{0,\nu-1,3,\iota,0}}$: This experiment coincides with $\mathsf{Hyb}_{0,\nu-1,3,\iota}$.

$\mathbf{Hyb_{0,\nu-1,3,\iota,1}}$: In this experiment the challenger $\mathcal{B}$ generates the SSB hash key $\text{HK} \xleftarrow{\$} \mathsf{SSB.Gen}(1^\lambda, n_{\text{SSB-BLK}} = 2^\lambda, i^* = \iota)$. The rest of the experiment proceeds in an analogous fashion to $\mathsf{Hyb}_{0,\nu-1,3,\iota,0}$.

$\mathbf{Hyb_{0,\nu-1,3,\iota,2}}$: In this experiment, to answer the ν^{th} signing key query of $\mathcal{A}$ corresponding to TM $M^{(\nu)} \in \mathbb{M}_\lambda$ with $M^{(\nu)}(x^*) = 0$, $\mathcal{B}$ proceeds as follows:

1. It first generates all the PPRF keys and the public parameters for the iterator just as in $\mathsf{Hyb}_{0,\nu-1,3,\iota,1}$.
2. After that, it generates $(\text{PP}^{(\nu)}_{\text{ACC}}, w_0^{(\nu)}, \text{STORE}_0^{(\nu)} \xleftarrow{\$} \mathsf{ACC.Setup\text{-}Enforce\text{-}Write}(1^\lambda, n_{\text{ACC-BLK}} = 2^\lambda, ((x_0^*, 0), \ldots, (x_\iota^*, \iota)))$.
3. Next, it sets $m_{0,0}^{(\nu)} = (v_0^{(\nu)}, q_0^{(\nu)}, w_0^{(\nu)}, 0)$. For $j = 1, \ldots, \iota$, it iteratively computes the following:
   - $\text{AUX}_j^{(\nu)} = \mathsf{ACC.Prep\text{-}Write}(\text{PP}_{\text{ACC}}^{(\nu)}, \text{STORE}_{j-1}^{(\nu)}, j-1)$
   - $w_j^{(\nu)} = \mathsf{ACC.Update}(\text{PP}_{\text{ACC}}^{(\nu)}, w_{j-1}^{(\nu)}, x_{j-1}^*, j-1, \text{AUX}_j^{(\nu)})$
   - $\text{STORE}_j^{(\nu)} = \mathsf{ACC.Write\text{-}Store}(\text{PP}_{\text{ACC}}^{(\nu)}, \text{STORE}_{j-1}^{(\nu)}, j-1, x_{j-1}^*)$
   - $v_j^{(\nu)} = \mathsf{ITR.Iterate}(\text{PP}_{\text{ITR}}^{(\nu)}, v_{j-1}^{(\nu)}, (q_0^{(\nu)}, w_{j-1}^{(\nu)}, 0))$

   It sets $m_{\iota,0}^{(\nu)} = (v_\iota^{(\nu)}, q_0^{(\nu)}, w_\iota^{(\nu)}, 0)$.
4. $\mathcal{B}$ gives $\mathcal{A}$ the signing key

$$\text{SK}_{\text{ABS}}\{M^{(\nu)}\} = \left(\begin{array}{l} \text{HK}, \text{PP}_{\text{ACC}}^{(\nu)}, w_0^{(\nu)}, \text{STORE}_0^{(\nu)}, \text{PP}_{\text{ITR}}^{(\nu)}, v_0^{(\nu)}, \\ \mathcal{IO}(\mathsf{Init\text{-}SPS.Prog}[q_0^{(\nu)}, w_0^{(\nu)}, v_0^{(\nu)}, K_{\text{SPS},E}^{(\nu)}]), \\ \mathcal{IO}(\mathsf{Accumulate.Prog}^{(3,\iota)}[n_{\text{SSB-BLK}} = 2^\lambda, \text{HK}, \text{PP}_{\text{ACC}}^{(\nu)}, \text{PP}_{\text{ITR}}^{(\nu)}, K_{\text{SPS},E}^{(\nu)}, \\ \qquad K_{\text{SPS},F}^{(\nu)}, m_{\iota,0}^{(\nu)}, h^*, \ell^*]), \\ \mathcal{IO}(\mathsf{Change\text{-}SPS.Prog}^{(3,\iota)}[K_{\text{SPS},A}^{(\nu)}, K_{\text{SPS},B}^{(\nu)}, K_{\text{SPS},E}^{(\nu)}, K_{\text{SPS},F}^{(\nu)}, h^*, \ell^*]), \\ \mathcal{IO}(\mathsf{Constrained\text{-}Key.Prog}_{\text{ABS}}^{(1)}[M^{(\nu)}, T = 2^\lambda, t^{*(\nu)}, \text{PP}_{\text{ACC}}^{(\nu)}, \text{PP}_{\text{ITR}}^{(\nu)}, K, \\ \qquad K_1^{(\nu)}, \ldots, K_\lambda^{(\nu)}, K_{\text{SPS},A}^{(\nu)}, K_{\text{SPS},B}^{(\nu)}, h^*, \ell^*]) \end{array}\right).$$

The rest of the experiment is similar to $\mathsf{Hyb}_{0,\nu-1,3,\iota,1}$.

$\mathbf{Hyb_{0,\nu-1,3,\iota,3}}$: In this experiment, in response to the ν^{th} signing key query of $\mathcal{A}$ corresponding to TM $M^{(\nu)} \in \mathbb{M}_\lambda$ with $M^{(\nu)}(x^*) = 0$, $\mathcal{B}$ proceeds as follows:

1. It first generates all the PPRF keys as well as the public parameters for the positional accumulator and the iterator as in $\mathsf{Hyb}_{0,\nu-1,3,\iota,2}$.
2. Next, it sets $m_{0,0}^{(\nu)} = (v_0^{(\nu)}, q_0^{(\nu)}, w_0^{(\nu)}, 0)$. For $j = 1, \ldots, \iota+1$, it iteratively computes the following:
   - $\text{AUX}_j^{(\nu)} = \mathsf{ACC.Prep\text{-}Write}(\text{PP}_{\text{ACC}}^{(\nu)}, \text{STORE}_{j-1}^{(\nu)}, j-1)$
   - $w_j^{(\nu)} = \mathsf{ACC.Update}(\text{PP}_{\text{ACC}}^{(\nu)}, w_{j-1}^{(\nu)}, x_{j-1}^*, j-1, \text{AUX}_j^{(\nu)})$
   - $\text{STORE}_j^{(\nu)} = \mathsf{ACC.Write\text{-}Store}(\text{PP}_{\text{ACC}}^{(\nu)}, \text{STORE}_{j-1}^{(\nu)}, j-1, x_{j-1}^*)$
   - $v_j^{(\nu)} = \mathsf{ITR.Iterate}(\text{PP}_{\text{ITR}}^{(\nu)}, v_{j-1}^{(\nu)}, (q_0^{(\nu)}, w_{j-1}^{(\nu)}, 0))$

It sets $m_{\iota,0}^{(\nu)} = (v_\iota^{(\nu)}, q_0^{(\nu)}, w_\iota^{(\nu)}, 0)$ and $\underline{m_{\iota+1,0}^{(\nu)} = (v_{\iota+1}^{(\nu)}, q_0^{(\nu)}, w_{\iota+1}^{(\nu)}, 0)}$.

3. It gives $\mathcal{A}$ the signing key

$$\mathrm{SK}_{\mathrm{ABS}}\{M^{(\nu)}\} = \left(\begin{array}{l} \mathrm{HK}, \mathrm{PP}_{\mathrm{ACC}}^{(\nu)}, w_0^{(\nu)}, \mathrm{STORE}_0^{(\nu)}, \mathrm{PP}_{\mathrm{ITR}}^{(\nu)}, v_0^{(\nu)}, \\ \mathcal{IO}(\mathsf{Init\text{-}SPS.Prog}[q_0^{(\nu)}, w_0^{(\nu)}, v_0^{(\nu)}, K_{\mathrm{SPS},E}^{(\nu)}]), \\ \mathcal{IO}(\underline{\mathsf{Accumulate.Prog}^{(3,\iota,1)}[n_{\text{SSB-BLK}} = 2^\lambda, \mathrm{HK}, \mathrm{PP}_{\mathrm{ACC}}^{(\nu)}, \mathrm{PP}_{\mathrm{ITR}}^{(\nu)}, K_{\mathrm{SPS},E}^{(\nu)}, \\ \quad K_{\mathrm{SPS},F}^{(\nu)}, m_{\iota,0}^{(\nu)}, m_{\iota+1,0}^{(\nu)}, h^*, \ell^*]}), \\ \mathcal{IO}(\mathsf{Change\text{-}SPS.Prog}^{(3,\iota)}[K_{\mathrm{SPS},A}^{(\nu)}, K_{\mathrm{SPS},B}^{(\nu)}, K_{\mathrm{SPS},E}^{(\nu)}, K_{\mathrm{SPS},F}^{(\nu)}, h^*, \ell^*]), \\ \mathcal{IO}(\mathsf{Constrained\text{-}Key.Prog}_{\mathrm{ABS}}^{(1)}[M^{(\nu)}, T = 2^\lambda, t^{*(\nu)}, \mathrm{PP}_{\mathrm{ACC}}^{(\nu)}, \mathrm{PP}_{\mathrm{ITR}}^{(\nu)}, K, \\ \quad K_1^{(\nu)}, \ldots, K_\lambda^{(\nu)}, K_{\mathrm{SPS},A}^{(\nu)}, K_{\mathrm{SPS},B}^{(\nu)}, h^*, \ell^*]) \end{array}\right),$$

where the program $\mathsf{Accumulate.Prog}^{(3,\iota,1)}$ is a modification of the program $\mathsf{Accumulate.Prog}^{(3,\iota)}$ (Fig. **15**) and is described in Fig. **32**.

The rest of the experiment if analogous to $\mathsf{Hyb}_{0,\nu-1,3,\iota,2}$.

Constants: Maximum number of blocks for SSB hash $n_{\text{SSB-BLK}} = 2^\lambda$, SSB hash key HK, Public parameters for positional accumulator $\mathrm{PP}_{\mathrm{ACC}}$, Public parameters for iterator $\mathrm{PP}_{\mathrm{ITR}}$, PPRF keys $K_{\mathrm{SPS},E}, K_{\mathrm{SPS},F}$, Messages $m_{\iota,0}$, $\underline{m_{\iota+1,0}}$, SSB hash value of challenge input h^*, Length of challenge input ℓ^*

Inputs: Index i, Symbol $\mathrm{SYM}_{\mathrm{IN}}$, TM state ST, Accumulator value w_{IN}, Auxiliary value AUX, Iterator value v_{IN}, Signature $\sigma_{\mathrm{SPS,IN}}$, SSB hash value h, SSB opening value π_{SSB}

Output: (Accumulator value w_{OUT}, Iterator value v_{OUT}, Signature $\sigma_{\text{SPS-OUT}}$), or $\perp$

1. (a) Compute $r_{\mathrm{SPS},E} = \mathcal{F}(K_{\mathrm{SPS},E}, (h,i))$, $(\mathrm{SK}_{\mathrm{SPS},E}, \mathrm{VK}_{\mathrm{SPS},E}, \mathrm{VK}_{\text{SPS-REJ},E}) = \mathsf{SPS.Setup}(1^\lambda; r_{\mathrm{SPS},E})$.
   (b) Compute $r_{\mathrm{SPS},F} = \mathcal{F}(K_{\mathrm{SPS},F}, (h,i))$, $(\mathrm{SK}_{\mathrm{SPS},F}, \mathrm{VK}_{\mathrm{SPS},F}, \mathrm{VK}_{\text{SPS-REJ},F}) = \mathsf{SPS.Setup}(1^\lambda; r_{\mathrm{SPS},F})$.
   (c) Set $m_{\mathrm{IN}} = (v_{\mathrm{IN}}, \mathrm{ST}, w_{\mathrm{IN}}, 0)$ and $\alpha =$'-'.
   (d) If $\mathsf{SPS.Verify}(\mathrm{VK}_{\mathrm{SPS},E}, m_{\mathrm{IN}}, \sigma_{\mathrm{SPS,IN}}) = 1$, set $\alpha =$'E'.
   (e) If $[\alpha =\text{'-'}] \wedge [(i > \ell^*) \vee (0 \le i \le \iota) \vee (h \ne h^*)]$, output $\perp$.
   Else if $[\alpha =\text{'-'}] \wedge [\mathsf{SPS.Verify}(\mathrm{VK}_{\mathrm{SPS},F}, m_{\mathrm{IN}}, \sigma_{\mathrm{SPS,IN}}) = 1]$, set $\alpha =$'F'.
   (f) If $\alpha =$'-', output $\perp$.
2. If $\mathsf{SSB.Verify}(\mathrm{HK}, h, i, \mathrm{SYM}_{\mathrm{IN}}, \pi_{\mathrm{SSB}}) = 0$, output $\perp$.
3. (a) Compute $w_{\mathrm{OUT}} = \mathsf{ACC.Update}(\mathrm{PP}_{\mathrm{ACC}}, w_{\mathrm{IN}}, \mathrm{SYM}_{\mathrm{IN}}, i, \mathrm{AUX})$. If $w_{\mathrm{OUT}} = \perp$, output $\perp$.
   (b) Compute $v_{\mathrm{OUT}} = \mathsf{ITR.Iterate}(\mathrm{PP}_{\mathrm{ITR}}, v_{\mathrm{IN}}, (\mathrm{ST}, w_{\mathrm{IN}}, 0))$.
4. (a) Compute $r'_{\mathrm{SPS},E} = \mathcal{F}(K_{\mathrm{SPS},E}, (h,i+1))$, $(\mathrm{SK}'_{\mathrm{SPS},E}, \mathrm{VK}'_{\mathrm{SPS},E}, \mathrm{VK}'_{\text{SPS-REJ},E}) = \mathsf{SPS.Setup}(1^\lambda; r'_{\mathrm{SPS},E})$.
   (b) Compute $r'_{\mathrm{SPS},F} = \mathcal{F}(K_{\mathrm{SPS},F}, (h,i+1))$, $(\mathrm{SK}'_{\mathrm{SPS},F}, \mathrm{VK}'_{\mathrm{SPS},F}, \mathrm{VK}'_{\text{SPS-REJ},F}) = \mathsf{SPS.Setup}(1^\lambda; r'_{\mathrm{SPS},F})$.
   (c) Set $m_{\mathrm{OUT}} = (v_{\mathrm{OUT}}, \mathrm{ST}, w_{\mathrm{OUT}}, 0)$. If $[(h,i) = (h^*, \iota)] \wedge [(m_{\mathrm{IN}} = m_{\iota,0}) \underline{\wedge (m_{\mathrm{OUT}} = m_{\iota+1,0})}]$, compute $\sigma_{\mathrm{SPS,OUT}} = \mathsf{SPS.Sign}(\mathrm{SK}'_{\mathrm{SPS},E}, m_{\mathrm{OUT}})$.
   Else if $[(h,i) = (h^*, \iota)] \wedge [(m_{\mathrm{IN}} \ne m_{\iota,0}) \underline{\vee (m_{\mathrm{OUT}} \ne m_{\iota+1,0})}]$, compute $\sigma_{\mathrm{SPS,OUT}} = \mathsf{SPS.Sign}(\mathrm{SK}'_{\mathrm{SPS},F}, m_{\mathrm{OUT}})$.
   Else if $i < \ell^*$, compute $\sigma_{\mathrm{SPS,OUT}} = \mathsf{SPS.Sign}(\mathrm{SK}'_{\mathrm{SPS},\alpha}, m_{\mathrm{OUT}})$.
   Else, compute $\sigma_{\mathrm{SPS,OUT}} = \mathsf{SPS.Sign}(\mathrm{SK}'_{\mathrm{SPS},E}, m_{\mathrm{OUT}})$.
5. Output $(w_{\mathrm{OUT}}, v_{\mathrm{OUT}}, \sigma_{\mathrm{SPS,OUT}})$.

Fig. 32 $\mathsf{Accumulate.Prog}^{(3,\iota,1)}$

$\mathsf{Hyb}_{0,\nu-1,3,\iota,4}$: This experiment is identical to $\mathsf{Hyb}_{0,\nu-1,3,\iota,3}$ with the only exception that while constructing the ν^{th} signing key queried by $\mathcal{A}$, $\mathcal{B}$ generates $\underline{(\mathrm{PP}_{\mathrm{ACC}}^{(\nu)}, w_0^{(\nu)}, \mathrm{STORE}_0^{(\nu)}) \xleftarrow{\$} \mathsf{ACC.Setup}(1^\lambda, n_{\text{ACC-BLK}} = 2^\lambda)}$.

$\mathsf{Hyb}_{0,\nu-1,3,\iota,5}$: This experiment is identical to $\mathsf{Hyb}_{0,\nu-1,3,\iota,4}$ with the only exception that $\mathcal{B}$ generates $\underline{\mathrm{HK} \xleftarrow{\$} \mathsf{SSB.Gen}(1^\lambda, n_{\text{SSB-BLK}} = 2^\lambda, i^* = 0)}$.

$\mathsf{Hyb}_{0,\nu-1,3,\iota,6}$: In this experiment, in response to the ν^{th} signing key query of $\mathcal{A}$ corresponding to TM $M^{(\nu)} \in \mathbb{M}_\lambda$ with $M^{(\nu)}(x^*) = 0$, $\mathcal{B}$ proceeds as follows:

1. It first generates all the PPRF keys as well as the public parameters for the positional accumulator as in $\mathsf{Hyb}_{0,\nu-1,3,\iota,5}$.
2. For $j = 1, \ldots, \iota+1$, it iteratively computes the following:
   - $\text{AUX}_j^{(\nu)} = \mathsf{ACC.Prep\text{-}Write}(\text{PP}_{\text{ACC}}^{(\nu)}, \text{STORE}_{j-1}^{(\nu)}, j-1)$
   - $w_j^{(\nu)} = \mathsf{ACC.Update}(\text{PP}_{\text{ACC}}^{(\nu)}, w_{j-1}^{(\nu)}, x_{j-1}^*, j-1, \text{AUX}_j^{(\nu)})$
   - $\text{STORE}_j^{(\nu)} = \mathsf{ACC.Write\text{-}Store}(\text{PP}_{\text{ACC}}^{(\nu)}, \text{STORE}_{j-1}^{(\nu)}, j-1, x_{j-1}^*)$
3. Then, it generates $\underline{(\text{PP}_{\text{ITR}}^{(\nu)}, v_0^{(\nu)}) \xleftarrow{\$} \mathsf{ITR.Setup\text{-}Enforce}(1^\lambda, n_{\text{ITR}} = 2^\lambda, ((q_0^{(\nu)}, w_0^{(\nu)}, 0), \ldots, (q_0^{(\nu)}, w_\iota^{(\nu)}, 0)))}$.
4. After that, for $j = 1, \ldots, \iota+1$, it iteratively computes $v_j^{(\nu)} = \mathsf{ITR.Iterate}(\text{PP}_{\text{ITR}}^{(\nu)}, v_{j-1}^{(\nu)}, (q_0^{(\nu)}, w_{j-1}^{(\nu)}, 0))$.
5. It sets $m_{\iota,0}^{(\nu)} = (v_\iota^{(\nu)}, q_0^{(\nu)}, w_\iota^{(\nu)}, 0)$ and $m_{\iota+1,0}^{(\nu)} = (v_{\iota+1}^{(\nu)}, q_0^{(\nu)}, w_{\iota+1}^{(\nu)}, 0)$.
6. It gives $\mathcal{A}$ the signing key

$$\text{SK}_{\text{ABS}}\{M^{(\nu)}\} = \left(\begin{array}{l} \text{HK}, \text{PP}_{\text{ACC}}^{(\nu)}, w_0^{(\nu)}, \text{STORE}_0^{(\nu)}, \text{PP}_{\text{ITR}}^{(\nu)}, v_0^{(\nu)}, \\ \mathcal{IO}(\mathsf{Init\text{-}SPS.Prog}[q_0^{(\nu)}, w_0^{(\nu)}, v_0^{(\nu)}, K_{\text{SPS},E}^{(\nu)}]), \\ \mathcal{IO}(\mathsf{Accumulate.Prog}^{(3,\iota,1)}[n_{\text{SSB-BLK}} = 2^\lambda, \text{HK}, \text{PP}_{\text{ACC}}^{(\nu)}, \text{PP}_{\text{ITR}}^{(\nu)}, K_{\text{SPS},E}^{(\nu)}, \\ \qquad K_{\text{SPS},F}^{(\nu)}, m_{\iota,0}^{(\nu)}, m_{\iota+1,0}^{(\nu)}, h^*, \ell^*]), \\ \mathcal{IO}(\mathsf{Change\text{-}SPS.Prog}^{(3,\iota)}[K_{\text{SPS},A}^{(\nu)}, K_{\text{SPS},B}^{(\nu)}, K_{\text{SPS},E}^{(\nu)}, K_{\text{SPS},F}^{(\nu)}, h^*, \ell^*]), \\ \mathcal{IO}(\mathsf{Constrained\text{-}Key.Prog}_{\text{ABS}}^{(1)}[M^{(\nu)}, T = 2^\lambda, t^{*(\nu)}, \text{PP}_{\text{ACC}}^{(\nu)}, \text{PP}_{\text{ITR}}^{(\nu)}, K, \\ \qquad K_1^{(\nu)}, \ldots, K_\lambda^{(\nu)}, K_{\text{SPS},A}^{(\nu)}, K_{\text{SPS},B}^{(\nu)}, h^*, \ell^*]) \end{array}\right).$$

The rest of the experiment if analogous to $\mathsf{Hyb}_{0,\nu-1,3,\iota,5}$.

$\mathsf{Hyb}_{0,\nu-1,3,\iota,7}$: In this experiment, to answer the ν^{th} signing key query of $\mathcal{A}$ corresponding to TM $M^{(\nu)} \in \mathbb{M}_\lambda$ with $M^{(\nu)}(x^*) = 0$, $\mathcal{B}$ generates everything as in $\mathsf{Hyb}_{0,\nu-1,3,\iota,6}$, however, it hands $\mathcal{A}$ the signing key

$$\text{SK}_{\text{ABS}}\{M^{(\nu)}\} = \left(\begin{array}{l} \text{HK}, \text{PP}_{\text{ACC}}^{(\nu)}, w_0^{(\nu)}, \text{STORE}_0^{(\nu)}, \text{PP}_{\text{ITR}}^{(\nu)}, v_0^{(\nu)}, \\ \mathcal{IO}(\mathsf{Init\text{-}SPS.Prog}[q_0^{(\nu)}, w_0^{(\nu)}, v_0^{(\nu)}, K_{\text{SPS},E}^{(\nu)}]), \\ \mathcal{IO}(\mathsf{Accumulate.Prog}^{(3,\iota')}[n_{\text{SSB-BLK}} = 2^\lambda, \text{HK}, \text{PP}_{\text{ACC}}^{(\nu)}, \text{PP}_{\text{ITR}}^{(\nu)}, K_{\text{SPS},E}^{(\nu)}, \\ \qquad \underline{K_{\text{SPS},F}^{(\nu)}, m_{\iota+1,0}^{(\nu)}, h^*, \ell^*]}), \\ \underline{\mathcal{IO}(\mathsf{Change\text{-}SPS.Prog}^{(3,\iota)}[K_{\text{SPS},A}^{(\nu)}, K_{\text{SPS},B}^{(\nu)}, K_{\text{SPS},E}^{(\nu)}, K_{\text{SPS},F}^{(\nu)}, h^*, \ell^*])}, \\ \mathcal{IO}(\mathsf{Constrained\text{-}Key.Prog}_{\text{ABS}}^{(1)}[M^{(\nu)}, T = 2^\lambda, t^{*(\nu)}, \text{PP}_{\text{ACC}}^{(\nu)}, \text{PP}_{\text{ITR}}^{(\nu)}, K, \\ \qquad K_1^{(\nu)}, \ldots, K_\lambda^{(\nu)}, K_{\text{SPS},A}^{(\nu)}, K_{\text{SPS},B}^{(\nu)}, h^*, \ell^*]) \end{array}\right),$$

where the program $\mathsf{Accumulate.Prog}^{(3,\iota')}$ is depicted in Fig. **17**. The rest of the experiment is similar to $\mathsf{Hyb}_{0,\nu-1,3,\iota,6}$.

$\mathsf{Hyb}_{0,\nu-1,3,\iota,8}$: This experiment is analogous to $\mathsf{Hyb}_{0,\nu-1,3,\iota,7}$ with the only exception that while constructing the ν^{th} signing key queried by $\mathcal{A}$, $\mathcal{B}$ generates

$(\mathrm{PP}_{\mathrm{ITR}}^{(\nu)}, v_0^{(\nu)}) \xleftarrow{\$} \mathsf{ITR.Setup}(1^\lambda, n_{\mathrm{ITR}} = 2^\lambda)$. Notice that this experiment coincides with $\mathsf{Hyb}_{0,\nu-1,3,\iota'}$.

Analysis

Let $\mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,3,\iota,\vartheta)}(\lambda)$ represents the advantage of the adversary $\mathcal{A}$, i.e., the absolute difference between 1/2 and $\mathcal{A}$'s probability of correctly guessing the random bit selected by the challenger $\mathcal{B}$, in $\mathsf{Hyb}_{0,\nu-1,3,\iota,\vartheta}$, for $\vartheta \in [0,8]$. From the description of the hybrid experiments it follows that $\mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,3,\iota)}(\lambda) \equiv \mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,3,\iota,0)}(\lambda)$ and $\mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,3,\iota')}(\lambda) \equiv \mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,3,\iota,8)}(\lambda)$. Hence, we have

$$\begin{aligned} &|\mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,3,\iota)}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,3,\iota')}(\lambda)| \\ &\qquad \leq \sum_{\vartheta=1}^{8} |\mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,3,\iota,\vartheta-1)}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,3,\iota,\vartheta)}(\lambda)|. \end{aligned} \tag{C.5}$$

Claims C.16–C.23 below will show that the RHS of Eq. (C.5) is negligible and thus Lemma C.5 follows.

Claim C.16 *Assuming* SSB *satisfies the 'index hiding' property, for any* PPT *adversary* $\mathcal{A}$*, for any security parameter* λ*,* $|\mathit{Adv}_{\mathcal{A}}^{(0,\nu-1,3,\iota,0)}(\lambda) - \mathit{Adv}_{\mathcal{A}}^{(0,\nu-1,3,\iota,1)}(\lambda)| \leq \mathsf{negl}(\lambda)$ *for some negligible function* negl.

Proof Suppose there exists a PPT adversary $\mathcal{A}$ for which $|\mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,3,\iota,0)}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,3,\iota,1)}(\lambda)|$ is non-negligible. We construct a PPT adversary $\mathcal{B}$ that breaks the index hiding property of SSB using $\mathcal{A}$ as a sub-routine. The description of $\mathcal{B}$ follows:

- $\mathcal{B}$ initializes $\mathcal{A}$ on input 1^λ and receives a challenge signing attribute string $x^* = x_0^* \ldots x_{\ell^*-1}^* \in \mathcal{U}_{\mathrm{ABS}}$ with $|x^*| = \ell^*$ from $\mathcal{A}$.
- Upon receiving x^*, $\mathcal{B}$ proceeds as follows:
  1. $\mathcal{B}$ submits $n_{\text{SSB-BLK}} = 2^\lambda$ and the pair of indices $(i_0^* = 0, i_1^* = \iota)$ to its SSB index hiding challenger $\mathcal{C}$ and receives back a hash key HK, where either $\mathrm{HK} \xleftarrow{\$} \mathsf{SSB.Gen}(1^\lambda, n_{\text{SSB-BLK}} = 2^\lambda, i_0^* = 0)$ or $\mathrm{HK} \xleftarrow{\$} \mathsf{SSB.Gen}(1^\lambda, n_{\text{SSB-BLK}} = 2^\lambda, i_1^* = \iota)$.
  2. Next, $\mathcal{B}$ computes $h^* = \mathcal{H}_{\mathrm{HK}}(x^*)$.
  3. Then, $\mathcal{B}$ selects a PPRF key $K \xleftarrow{\$} \mathcal{F}.\mathsf{Setup}(1^\lambda)$.
  4. After that, $\mathcal{B}$ computes $r_{\mathrm{SIG}}^* = \mathcal{F}(K, (h^*, \ell^*))$ followed by $(\widehat{\mathrm{SK}}_{\mathrm{SIG}}^*, \widehat{\mathrm{VK}}_{\mathrm{SIG}}^*) = \mathsf{SIG.Setup}(1^\lambda;\ r_{\mathrm{SIG}}^*)$.
  5. $\mathcal{B}$ returns the public parameters $\mathrm{PP}_{\mathrm{ABS}} = (\mathrm{HK}, \mathcal{IO}(\mathsf{Verify.Prog}_{\mathrm{ABS}}[K]))$ to $\mathcal{A}$.
- For $\eta \in [\hat{q}_{\mathrm{KEY}}]$, in response to the η^{th} signing key query of $\mathcal{A}$ corresponding to TM $M^{(\eta)} \in \mathbb{M}_\lambda$ with $M^{(\eta)}(x^*) = 0$, $\mathcal{B}$ proceeds exactly as in $\mathsf{Hyb}_{0,\nu-1,3,\iota,0}$.
- For $\theta \in [\hat{q}_{\mathrm{SIGN}}]$, in reply to the θ^{th} signature query of $\mathcal{A}$ on message $\mathsf{msg}^{(\theta)} \in \mathcal{M}_{\mathrm{ABS}}$ under attribute string x^*, $\mathcal{B}$ computes $\sigma_{\mathrm{SIG}}^{(\theta)} \xleftarrow{\$} \mathsf{SIG.Sign}(\widehat{\mathrm{SK}}_{\mathrm{SIG}}^*, \mathsf{msg}^{(\theta)})$ and provides $\mathcal{A}$ with $\sigma_{\mathrm{ABS}}^{(\theta)} = (\widehat{\mathrm{VK}}_{\mathrm{SIG}}^*, \sigma_{\mathrm{SIG}}^{(\theta)})$.
- Finally, $\mathcal{A}$ output a signature σ_{ABS}^* on some message msg^* under attribute string x^*. $\mathcal{B}$ outputs $\hat{b}' = 1$ as its guess bit in its SSB index hiding experiment if $\mathcal{A}$ wins, i.e., if $\mathsf{ABS.Verify}(\mathrm{PP}_{\mathrm{ABS}}, x^*, \mathsf{msg}^*, \sigma_{\mathrm{ABS}}^*) = 1$ and $\mathsf{msg}^* \neq \mathsf{msg}^{(\theta)}$ for any $\theta \in [\hat{q}_{\mathrm{SIGN}}]$. Otherwise, $\mathcal{B}$ outputs $\hat{b}' = 0$ in its SSB index experiment.

Note that if $\text{HK} \xleftarrow{\$} \mathsf{SSB.Gen}(1^\lambda, n_{\text{SSB-BLK}} = 2^\lambda, i_0^* = 0)$, then $\mathcal{B}$ perfectly simulates $\mathsf{Hyb}_{0,\nu-1,3,\iota,0}$. On the other hand, if $\text{HK} \xleftarrow{\$} \mathsf{SSB.Gen}(1^\lambda, n_{\text{SSB-BLK}} = 2^\lambda, i_1^* = \iota)$, then $\mathcal{B}$ perfectly simulates $\mathsf{Hyb}_{0,\nu-1,3,\iota,1}$. This completes the proof of Claim C.16. □

Claim C.17 *Assuming* ACC *is a positional accumulator satisfying the 'indistinguishability of write setup' property, for any* PPT *adversary* $\mathcal{A}$*, for any security parameter* λ*,* $|\mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,3,\iota,1)}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,3,\iota,2)}(\lambda)| \leq \mathsf{negl}(\lambda)$ *for some negligible function* negl*.*

Proof Suppose there exists a PPT adversary $\mathcal{A}$ for which $|\mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,3,\iota,1)}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,3,\iota,2)}(\lambda)|$ is non-negligible. We construct a PPT adversary $\mathcal{B}$ that breaks the indistinguishability of write setup property of the positional accumulator ACC using $\mathcal{A}$ as a sub-routine. The description of $\mathcal{B}$ follows:

- $\mathcal{B}$ initializes $\mathcal{A}$ on input 1^λ and receives a challenge signing attribute string $x^* = x_0^* \ldots x_{\ell^*-1}^* \in \mathcal{X}_{\text{CPRF}}$ with $|x^*| = \ell^*$ from $\mathcal{A}$.
- Upon receiving x^*, $\mathcal{B}$ proceeds as follows:
  1. $\mathcal{B}$ first generates $\text{HK} \xleftarrow{\$} \mathsf{SSB.Gen}(1^\lambda, n_{\text{SSB-BLK}} = 2^\lambda, i^* = \iota)$ and computes $h^* = \mathcal{H}_{\text{HK}}(x^*)$.
  2. Then, $\mathcal{B}$ selects a PPRF key $K \xleftarrow{\$} \mathcal{F}.\mathsf{Setup}(1^\lambda)$.
  3. After that, $\mathcal{B}$ computes $r_{\text{SIG}}^* = \mathcal{F}(K, (h^*, \ell^*))$ followed by $(\widehat{\text{SK}}_{\text{SIG}}^*, \widehat{\text{VK}}_{\text{SIG}}^*) = \mathsf{SIG.Setup}(1^\lambda;\ r_{\text{SIG}}^*)$.
  4. $\mathcal{B}$ returns the public parameters $\text{PP}_{\text{ABS}} = (\text{HK}, \mathcal{IO}(\mathsf{Verify.Prog}_{\text{ABS}}[K]))$ to $\mathcal{A}$.
- For $\eta \in [\hat{q}_{\text{KEY}}]$, in response to the η^{th} signing key query of $\mathcal{A}$ corresponding to TM $M^{(\eta)} \in \mathbb{M}_\lambda$ with $M^{(\nu)}(x^*) = 0$, if $\eta \neq \nu$, then $\mathcal{B}$ proceeds exactly as in $\mathsf{Hyb}_{0,\nu-1,3,\iota,1}$, while if $\eta = \nu$, then $\mathcal{B}$ proceeds as follows:
  1. $\mathcal{B}$ first selects PPRF keys $K_1^{(\nu)}, \ldots, K_\lambda^{(\nu)}, K_{\text{SPS},A}^{(\nu)}, K_{\text{SPS},B}^{(\nu)}, K_{\text{SPS},E}^{(\nu)}, K_{\text{SPS},F}^{(\nu)} \xleftarrow{\$} \mathcal{F}.\mathsf{Setup}(1^\lambda)$.
  2. Next, $\mathcal{B}$ sends $n_{\text{ACC-BLK}} = 2^\lambda$ and the sequence of symbol-index pairs $((x_0^*, 0), \ldots, (x_\iota^*, \iota))$ to its ACC write setup indistinguishability challenger $\mathcal{C}$ and receives back $(\text{PP}_{\text{ACC}}, w_0, \text{STORE}_0)$, where either $(\text{PP}_{\text{ACC}}, w_0, \text{STORE}_0) \xleftarrow{\$} \mathsf{ACC.Setup}(1^\lambda, n_{\text{ACC-BLK}} = 2^\lambda)$ or $(\text{PP}_{\text{ACC}}, w_0, \text{STORE}_0) \xleftarrow{\$} \mathsf{ACC.Setup\text{-}Enfoce\text{-}Write}(1^\lambda, n_{\text{ACC-BLK}} = 2^\lambda, ((x_0^*, 0), \ldots, (x_\iota^*, \iota)))$.
  3. Next, it generates $(\text{PP}_{\text{ITR}}^{(\nu)}, v_0^{(\nu)}) \xleftarrow{\$} \mathsf{ITR.Setup}(1^\lambda, n_{\text{ITR}} = 2^\lambda)$.
  4. Then, it sets $m_{0,0}^{(\nu)} = (v_0^{(\nu)}, q_0^{(\nu)}, w_0, 0)$. For $j = 1, \ldots, \iota$, it iteratively computes the following:
     - $\text{AUX}_j^{(\nu)} = \mathsf{ACC.Prep\text{-}Write}(\text{PP}_{\text{ACC}}, \text{STORE}_{j-1}, j-1)$
     - $w_j = \mathsf{ACC.Update}(\text{PP}_{\text{ACC}}, w_{j-1}, x_{j-1}^*, j-1, \text{AUX}_j^{(\nu)})$
     - $\text{STORE}_j = \mathsf{ACC.Write\text{-}Store}(\text{PP}_{\text{ACC}}\text{STORE}_{j-1}, j-1, x_{j-1}^*)$
     - $v_j^{(\nu)} = \mathsf{ITR.Iterate}(\text{PP}_{\text{ITR}}^{(\nu)}, v_{j-1}^{(\nu)}, (q_0^{(\nu)}, w_{j-1}, 0))$

     It sets $m_{\iota,0}^{(\nu)} = (v_\iota^{(\nu)}, q_0^{(\nu)}, w_\iota, 0)$.

5. It gives $\mathcal{A}$ the signing key
$\mathrm{SK}_{\mathrm{ABS}}\{M^{(\nu)}\} =$

$$\left(\begin{array}{l}\mathrm{HK}, \mathrm{PP}_{\mathrm{ACC}}, w_0, \mathrm{STORE}_0, \mathrm{PP}_{\mathrm{ITR}}^{(\nu)}, v_0^{(\nu)}, \\ \mathcal{IO}(\mathsf{Init\text{-}SPS.Prog}[q_0^{(\nu)}, w_0, v_0^{(\nu)}, K_{\mathrm{SPS},E}^{(\nu)}]), \\ \mathcal{IO}(\mathsf{Accumulate.Prog}^{(3,\iota)}[n_{\text{SSB-BLK}} = 2^\lambda, \mathrm{HK}, \mathrm{PP}_{\mathrm{ACC}}, \mathrm{PP}_{\mathrm{ITR}}^{(\nu)}, K_{\mathrm{SPS},E}^{(\nu)}, \\ \qquad K_{\mathrm{SPS},F}^{(\nu)}, m_{\iota,0}^{(\nu)}, h^*, \ell^*]), \\ \mathcal{IO}(\mathsf{Change\text{-}SPS.Prog}^{(3,\iota)}[K_{\mathrm{SPS},A}^{(\nu)}, K_{\mathrm{SPS},B}^{(\nu)}, K_{\mathrm{SPS},E}^{(\nu)}, K_{\mathrm{SPS},F}^{(\nu)}, h^*, \ell^*]), \\ \mathcal{IO}(\mathsf{Constrained\text{-}Key.Prog}_{\mathrm{ABS}}^{(1)}[M^{(\nu)}, T = 2^\lambda, t^{*(\nu)}, \mathrm{PP}_{\mathrm{ACC}}, \mathrm{PP}_{\mathrm{ITR}}^{(\nu)}, K, \\ \qquad K_1^{(\nu)}, \ldots, K_\lambda^{(\nu)}, K_{\mathrm{SPS},A}^{(\nu)}, K_{\mathrm{SPS},B}^{(\nu)}, h^*, \ell^*])\end{array}\right).$$

- For $\theta \in [\hat{q}_{\mathrm{SIGN}}]$, in reply to the θ^{th} signature query of $\mathcal{A}$ on message $\mathsf{msg}^{(\theta)} \in \mathcal{M}_{\mathrm{ABS}}$ under attribute string x^*, $\mathcal{B}$ computes $\sigma_{\mathrm{SIG}}^{(\theta)} \xleftarrow{\$} \mathsf{SIG.Sign}(\widehat{\mathrm{SK}}_{\mathrm{SIG}}^*, \mathsf{msg}^{(\theta)})$ and provides $\mathcal{A}$ with $\sigma_{\mathrm{ABS}}^{(\theta)} = (\widehat{\mathrm{VK}}_{\mathrm{SIG}}^*, \sigma_{\mathrm{SIG}}^{(\theta)})$.
- Finally, $\mathcal{A}$ output a signature σ_{ABS}^* on some message msg^* under attribute string x^*. $\mathcal{B}$ outputs $\hat{b}' = 1$ as its guess bit in its ACC write setup indistinguishability experiment if $\mathcal{A}$ wins, i.e., if $\mathsf{ABS.Verify}(\mathrm{PP}_{\mathrm{ABS}}, x^*, \mathsf{msg}^*, \sigma_{\mathrm{ABS}}^*) = 1$ and $\mathsf{msg}^* \neq \mathsf{msg}^{(\theta)}$ for any $\theta \in [\hat{q}_{\mathrm{SIGN}}]$. Otherwise, $\mathcal{B}$ outputs $\hat{b}' = 0$ in its ACC write setup indistinguishability experiment.

Note that if $(\mathrm{PP}_{\mathrm{ACC}}, w_0, \mathrm{STORE}_0) \xleftarrow{\$} \mathsf{ACC.Setup}(1^\lambda, n_{\text{ACC-BLK}} = 2^\lambda)$, then $\mathcal{B}$ perfectly simulates $\mathsf{Hyb}_{0,\nu-1,3,\iota,1}$. On the other hand, if $(\mathrm{PP}_{\mathrm{ACC}}, w_0, \mathrm{STORE}_0) \xleftarrow{\$} \mathsf{ACC.Setup\text{-}Enforce\text{-}Write}(1^\lambda, n_{\text{ACC-BLK}} = 2^\lambda, ((x_0^*, 0), \ldots, (x_\iota^*, \iota)))$, then $\mathcal{B}$ perfectly simulates $\mathsf{Hyb}_{0,\nu-1,3,\iota,2}$. This completes the proof of Claim C.17. □

Claim C.18 *Assuming $\mathcal{IO}$ is a secure indistinguishability obfuscator for* P/poly*, SSB possesses the 'somewhere statistically binding' property, and ACC is a positional accumulator having the 'write enforcing' property, for any* PPT *adversary $\mathcal{A}$, for any security parameter λ, $|\mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,3,\iota,2)}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,3,\iota,3)}(\lambda)| \leq \mathsf{negl}(\lambda)$ for some negligible function* negl.

Proof The difference between $\mathsf{Hyb}_{0,\nu-1,3,\iota,2}$ and $\mathsf{Hyb}_{0,\nu-1,3,\iota,3}$ is the following: In $\mathsf{Hyb}_{0,\nu-1,3,\iota,2}$, $\mathcal{B}$ includes the program $\mathcal{IO}(P_0)$ within the ν^{th} signing key provided to $\mathcal{A}$, whereas, in $\mathsf{Hyb}_{0,\nu-1,3,\iota,3}$, $\mathcal{B}$ includes the program $\mathcal{IO}(P_1)$ instead, where

- $P_0 = \mathsf{Accumulate.Prog}^{(3,\iota)}[n_{\text{SSB-BLK}} = 2^\lambda, \mathrm{HK}, \mathrm{PP}_{\mathrm{ACC}}^{(\nu)}, \mathrm{PP}_{\mathrm{ITR}}^{(\nu)}, K_{\mathrm{SPS},E}^{(\nu)}, K_{\mathrm{SPS},F}^{(\nu)}, m_{\iota,0}^{(\nu)}, h^*, \ell^*]$ (Fig. **15**),
- $P_1 = \mathsf{Accumulate.Prog}^{(3,\iota,1)}[n_{\text{SSB-BLK}} = 2^\lambda, \mathrm{HK}, \mathrm{PP}_{\mathrm{ACC}}^{(\nu)}, \mathrm{PP}_{\mathrm{ITR}}^{(\nu)}, K_{\mathrm{SPS},E}^{(\nu)}, K_{\mathrm{SPS},F}^{(\nu)}, m_{\iota,0}^{(\nu)}, m_{\iota+1,0}^{(\nu)}, h^*, \ell^*]$ (Fig. **32**).

We will argue that the programs P_0 and P_1 are functionally equivalent, so that, by the security of $\mathcal{IO}$ Claim C.18 follows. The inputs on which the outputs of the two programs can possibly differ are those corresponding to $(h, i) = (h^*, \iota)$. For inputs corresponding to (h^*, ι), the program P_1 performs the additional check '$m_{\mathrm{OUT}} = m_{\iota+1,0}^{(\nu)}$' to determine the type of the outputted signature. We show that this check is redundant by demonstrating that for inputs corresponding to (h^*, ι), if $m_{\mathrm{IN}} = m_{\iota,0}^{(\nu)}$, then either both the programs output $\perp$ or it must hold that $m_{\mathrm{OUT}} = m_{\iota+1,0}^{(\nu)}$ and, therefore, both the programs output signatures of the same

type. Notice that $m_{\text{IN}} = m_{\iota,0}^{(\nu)}$ means $v_{\text{IN}} = v_\iota^{(\nu)}, \text{ST} = q_0^{(\nu)}$, and $w_{\text{IN}} = w_\iota^{(\nu)}$. Thus, $v_{\text{OUT}} = \mathsf{ITR.Iterate}(\text{PP}_{\text{ITR}}^{(\nu)}, v_{\text{IN}}, (\text{ST}, w_{\text{IN}}, 0)) = \mathsf{ITR.Iterate}(\text{PP}_{\text{ITR}}^{(\nu)}, v_\iota^{(\nu)}, (q_0^{(\nu)}, w_\iota^{(\nu)}, 0)) = v_{\iota+1}^{(\nu)}$. Now, recall that in both experiments $\text{HK} \xleftarrow{\$} \mathsf{SSB.Gen}(1^\lambda, n_{\text{SSB-BLK}} = 2^\lambda, i^* = \iota)$. Therefore, by the somewhere statistically binding property of SSB it follows that $\mathsf{SSB.Verify}(\text{HK}, h^* = \mathcal{H}_{\text{HK}}(x^*), \iota, \text{SYM}_{\text{IN}}, \pi_{\text{SSB}}) = 1$ if and only if $\text{SYM}_{\text{IN}} = x_\iota^*$. Thus, for inputs corresponding to (h^*, ι), both programs will output $\perp$ in case $\text{SYM}_{\text{IN}} \neq x_\iota^*$. Further, in both the experiments, $(\text{PP}_{\text{ACC}}^{(\nu)}, w_0^{(\nu)}, \text{STORE}_0^{(\nu)}) \xleftarrow{\$} \mathsf{ACC.Setup\text{-}Enforce\text{-}Write}(1^\lambda, n_{\text{ACC-BLK}} = 2^\lambda, ((x_0^*, 0), \ldots, (x_\iota^*, \iota)))$. Therefore, by the write enforcing property of ACC it follows that if $w_{\text{IN}} = w_\iota^{(\nu)}$ and $\text{SYM}_{\text{IN}} = x_\iota^*$, then $w_{\text{OUT}} = \mathsf{ACC.Update}(\text{PP}_{\text{ACC}}^{(\nu)}, w_{\text{IN}}, \text{SYM}_{\text{IN}}, \iota, \text{AUX})$ results in $w_{\text{OUT}} = w_{\iota+1}^{(\nu)}$ or $w_{\text{OUT}} = \perp$. In case $w_{\text{OUT}} = \perp$, then clearly both the programs output $\perp$. On the other hand, $w_{\text{OUT}} = w_{\iota+1}^{(\nu)}$ implies $m_{\text{OUT}} = (v_{\text{OUT}}, \text{ST}, w_{\text{OUT}}, 0) = (v_{\iota+1}^{(\nu)}, q_0^{(\nu)}, w_{\iota+1}^{(\nu)}, 0) = m_{\iota+1,0}^{(\nu)}$ and the two programs have identical outputs in this case as well.

□

Claim C.19 *Assuming* ACC *is a positional accumulator satisfying the 'indistinguishability of write setup' property, for any* PPT *adversary* $\mathcal{A}$*, for any security parameter* λ*,* $|\mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,3,\iota,3)}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,3,\iota,4)}(\lambda)| \leq \mathsf{negl}(\lambda)$ *for some negligible function* negl*.*

Proof The proof of Claim C.19 is similar to that of Claim C.17 with some appropriate modifications which can be readily figured out. □

Claim C.20 *Assuming* SSB *satisfies the 'index hiding' property, for any* PPT *adversary* $\mathcal{A}$*, for any security parameter* λ*,* $|\mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,3,\iota,4)}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,3,\iota,5)}(\lambda)| \leq \mathsf{negl}(\lambda)$ *for some negligible function* negl*.*

Proof The proof of Claim C.20 is analogous to that of Claim C.16 with certain approximate changes which are easy to determine. □

Claim C.21 *Assuming* ITR *satisfies the 'indistinguishability of enforcing setup' property, for any* PPT *adversary* $\mathcal{A}$*, for any security parameter* λ*,* $|\mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,3,\iota,5)}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,3,\iota,6)}(\lambda)| \leq \mathsf{negl}(\lambda)$ *for some negligible function* negl*.*

Proof Suppose there exists a PPT adversary $\mathcal{A}$ for which $|\mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,3,\iota,5)}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,3,\iota,6)}(\lambda)|$ is non-negligible. Below, we construct a PPT adversary $\mathcal{B}$ that breaks the indistinguishability of enforcing setup property of the iterator ITR using $\mathcal{A}$ as a sub-routine.

- $\mathcal{B}$ initializes $\mathcal{A}$ on input 1^λ and receives a challenge input $x^* = x_0^* \ldots x_{\ell^*-1}^* \in \mathcal{U}_{\text{ABS}}$ with $|x^*| = \ell^*$ from $\mathcal{A}$.
- Upon receiving x^*, $\mathcal{B}$ proceeds as follows:
  1. $\mathcal{B}$ first generates $\text{HK} \xleftarrow{\$} \mathsf{SSB.Gen}(1^\lambda, n_{\text{SSB-BLK}} = 2^\lambda, i^* = 0)$ and computes $h^* = \mathcal{H}_{\text{HK}}(x^*)$.
  2. Then, $\mathcal{B}$ selects a PPRF key $K \xleftarrow{\$} \mathcal{F}.\mathsf{Setup}(1^\lambda)$.
  3. After that, $\mathcal{B}$ computes $r_{\text{SIG}}^* = \mathcal{F}(K, (h^*, \ell^*))$ followed by $(\widehat{\text{SK}}_{\text{SIG}}^*, \widehat{\text{VK}}_{\text{SIG}}^*) = \mathsf{SIG.Setup}(1^\lambda;\ r_{\text{SIG}}^*)$.
  4. $\mathcal{B}$ returns the public parameters $\text{PP}_{\text{ABS}} = (\text{HK}, \mathcal{IO}(\mathsf{Verify.Prog}_{\text{ABS}}[K]))$ to $\mathcal{A}$.

- For $\eta \in [\hat{q}_{\text{KEY}}]$, in response to the η^{th} signing key query of $\mathcal{A}$ corresponding to TM $M^{(\eta)} \in \mathbb{M}_\lambda$ with $M^{(\eta)}(x^*) = 0$, if $\eta \neq \nu$, then $\mathcal{B}$ proceeds exactly as in $\mathsf{Hyb}_{0,\nu-1,3,\iota,5}$, while if $\eta = \nu$, then $\mathcal{B}$ proceeds as follows:
  1. $\mathcal{B}$ first selects PPRF keys $K_1^{(\nu)}, \ldots, K_\lambda^{(\nu)}, K_{\text{SPS},A}^{(\nu)}, K_{\text{SPS},B}^{(\nu)}, K_{\text{SPS},E}^{(\nu)}, K_{\text{SPS},F}^{(\nu)} \xleftarrow{\$} \mathcal{F}.\mathsf{Setup}(1^\lambda)$.
  2. Next, it generates $(\text{PP}_{\text{ACC}}^{(\nu)}, w_0^{(\nu)}, \text{STORE}_0^{(\nu)}) \xleftarrow{\$} \mathsf{ACC.Setup}(1^\lambda, n_{\text{ACC-BLK}} = 2^\lambda)$.
  3. For $j = 1, \ldots, \iota + 1$, it iteratively computes the following:
     - $\text{AUX}_j^{(\nu)} = \mathsf{ACC.Prep\text{-}Write}(\text{PP}_{\text{ACC}}^{(\nu)}, \text{STORE}_{j-1}^{(\nu)}, j-1)$
     - $w_j^{(\nu)} = \mathsf{ACC.Update}(\text{PP}_{\text{ACC}}^{(\nu)}, w_{j-1}^{(\nu)}, x_{j-1}^*, j-1, \text{AUX}_j^{(\nu)})$
     - $\text{STORE}_j^{(\nu)} = \mathsf{ACC.Write\text{-}Store}(\text{PP}_{\text{ACC}}^{(\nu)}, \text{STORE}_{j-1}^{(\nu)}, j-1, x_{j-1}^*)$
  4. Then, $\mathcal{B}$ sends $n_{\text{ITR}} = 2^\lambda$ along with the sequence of messages $((q_0^{(\nu)}, w_0^{(\nu)}, 0), \ldots, (q_0^{(\nu)}, w_\iota^{(\nu)}, 0))$ to its ITR enforcing setup indistinguishability challenger $\mathcal{C}$ and receives back $(\text{PP}_{\text{ITR}}, v_0)$, where either $(\text{PP}_{\text{ITR}}, v_0) \xleftarrow{\$} \mathsf{ITR.Setup}(1^\lambda, n_{\text{ITR}} = 2^\lambda)$ or $(\text{PP}_{\text{ITR}}, v_0) \xleftarrow{\$} \mathsf{ITR.Setup\text{-}Enforce}(1^\lambda, n_{\text{ITR}} = 2^\lambda, ((q_0^{(\nu)}, w_0^{(\nu)}, 0), \ldots, (q_0^{(\nu)}, w_\iota^{(\nu)}, 0)))$.
  5. For $j = 1, \ldots, \iota + 1$, $\mathcal{B}$ iteratively computes $v_j = \mathsf{ITR.Iterate}(\text{PP}_{\text{ITR}}, v_{j-1}, (q_0^{(\nu)}, w_{j-1}^{(\nu)}, 0))$. It sets $m_{\iota,0}^{(\nu)} = (v_\iota, q_0^{(\nu)}, w_\iota^{(\nu)}, 0)$ and $m_{\iota+1,0}^{(\nu)} = (v_{\iota+1}, q_0^{(\nu)}, w_{\iota+1}^{(\nu)}, 0)$.
  6. It gives $\mathcal{A}$ the signing key
$$\text{SK}_{\text{ABS}}\{M^{(\nu)}\} = \left(\begin{array}{l} \text{HK}, \text{PP}_{\text{ACC}}^{(\nu)}, w_0^{(\nu)}, \text{STORE}_0^{(\nu)}, \text{PP}_{\text{ITR}}, v_0, \\ \mathcal{IO}(\mathsf{Init\text{-}SPS.Prog}[q_0^{(\nu)}, w_0^{(\nu)}, v_0, K_{\text{SPS},E}^{(\nu)}]), \\ \mathcal{IO}(\mathsf{Accumulate.Prog}^{(3,\iota,1)}[n_{\text{SSB-BLK}} = 2^\lambda, \text{HK}, \text{PP}_{\text{ACC}}^{(\nu)}, \text{PP}_{\text{ITR}}, K_{\text{SPS},E}^{(\nu)}, \\ \qquad K_{\text{SPS},F}^{(\nu)}, m_{\iota,0}^{(\nu)}, m_{\iota+1,0}^{(\nu)}, h^*, \ell^*]), \\ \mathcal{IO}(\mathsf{Change\text{-}SPS.Prog}^{(3,\iota)}[K_{\text{SPS},A}^{(\nu)}, K_{\text{SPS},B}^{(\nu)}, K_{\text{SPS},E}^{(\nu)}, K_{\text{SPS},F}^{(\nu)}, h^*, \ell^*]), \\ \mathcal{IO}(\mathsf{Constrained\text{-}Key.Prog}_{\text{ABS}}^{(1)}[M^{(\nu)}, T = 2^\lambda, t^{*(\nu)}, \text{PP}_{\text{ACC}}^{(\nu)}, \text{PP}_{\text{ITR}}, K, \\ \qquad K_1^{(\nu)}, \ldots, K_\lambda^{(\nu)}, K_{\text{SPS},A}^{(\nu)}, K_{\text{SPS},B}^{(\nu)}, h^*, \ell^*]) \end{array} \right).$$
- For $\theta \in [\hat{q}_{\text{SIGN}}]$, in reply to the θ^{th} signature query of $\mathcal{A}$ on message $\mathsf{msg}^{(\theta)} \in \mathcal{M}_{\text{ABS}}$ under attribute string x^*, $\mathcal{B}$ computes $\sigma_{\text{SIG}}^{(\theta)} \xleftarrow{\$} \mathsf{SIG.Sign}(\widehat{\text{SK}}_{\text{SIG}}^*, \mathsf{msg}^{(\theta)})$ and provides $\mathcal{A}$ with $\sigma_{\text{ABS}}^{(\theta)} = (\widehat{\text{VK}}_{\text{SIG}}^*, \sigma_{\text{SIG}}^{(\theta)})$.
- Finally, $\mathcal{A}$ output a signature σ_{ABS}^* on some message msg^* under attribute string x^*. $\mathcal{B}$ outputs $\hat{b}' = 1$ as its guess bit in its ITR enforcing setup indistinguishability experiment if $\mathcal{A}$ wins, i.e., if $\mathsf{ABS.Verify}(\text{PP}_{\text{ABS}}, x^*, \mathsf{msg}^*, \sigma_{\text{ABS}}^*) = 1$ and $\mathsf{msg}^* \neq \mathsf{msg}^{(\theta)}$ for any $\theta \in [\hat{q}_{\text{SIGN}}]$. Otherwise, $\mathcal{B}$ outputs $\hat{b}' = 0$ in its ITR enforcing setup indistinguishability experiment.

Note that if $(\text{PP}_{\text{ITR}}, v_0) \xleftarrow{\$} \mathsf{ITR.Setup}(1^\lambda, n_{\text{ITR}} = 2^\lambda)$, then $\mathcal{B}$ perfectly simulates $\mathsf{Hyb}_{0,\nu-1,3,\iota,5}$. On the other hand, if $(\text{PP}_{\text{ITR}}, v_0) \xleftarrow{\$} \mathsf{ITR.Setup\text{-}Enforce}(1^\lambda, n_{\text{ITR}} = 2^\lambda, ((q_0^{(\nu)}, w_0^{(\nu)}, 0), \ldots, (q_0^{(\nu)}, w_\iota^{(\nu)}, 0)))$, then $\mathcal{B}$ perfectly simulates $\mathsf{Hyb}_{0,\nu-1,3,\iota,6}$. This completes the proof of Claim C.21. □

Claim C.22 *Assuming $\mathcal{IO}$ is a secure indistinguishability obfuscator for* P/poly *and ITR has the 'enforcing' property, for any* PPT *adversary $\mathcal{A}$, for any security*

parameter λ, $|\mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,3,\iota,6)}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,3,\iota,7)}(\lambda)| \leq \mathsf{negl}(\lambda)$ *for some negligible function* negl.

Proof The difference between $\mathsf{Hyb}_{0,\nu-1,3,\iota,6}$ and $\mathsf{Hyb}_{0,\nu-1,3,\iota,7}$ is the following: In $\mathsf{Hyb}_{0,\nu-1,3,\iota,6}$, $\mathcal{B}$ includes the program $\mathcal{IO}(P_0)$ within the ν^{th} signing key provided to $\mathcal{A}$, whereas, in $\mathsf{Hyb}_{0,\nu-1,3,\iota,7}$, $\mathcal{B}$ includes the program $\mathcal{IO}(P_1)$ instead, where

- $P_0 = \mathsf{Accumulate.Prog}^{(3,\iota,1)}[n_{\text{SSB-BLK}} = 2^\lambda, \text{HK}, \text{PP}_{\text{ACC}}^{(\nu)}, \text{PP}_{\text{ITR}}^{(\nu)}, K_{\text{SPS},E}^{(\nu)}, K_{\text{SPS},F}^{(\nu)}, m_{\iota,0}^{(\nu)}, m_{\iota+1,0}^{(\nu)}, h^*, \ell^*]$ (Fig. **32**),
- $P_1 = \mathsf{Accumulate.Prog}^{(3,\iota')}[n_{\text{SSB-BLK}} = 2^\lambda, \text{HK}, \text{PP}_{\text{ACC}}^{(\nu)}, \text{PP}_{\text{ITR}}^{(\nu)}, K_{\text{SPS},E}^{(\nu)}, K_{\text{SPS},F}^{(\nu)}, m_{\iota+1,0}^{(\nu)}, h^*, \ell^*]$ (Fig. **17**).

We will argue that the programs P_0 and P_1 are functionally identical, so that, by the security of $\mathcal{IO}$ Claim C.22 follows. The only inputs on which the outputs of the two programs can possibly differ are those corresponding to $(h, i) = (h^*, \iota)$. For inputs corresponding to (h^*, ι), the program P_0 checks whether '$m_{\text{IN}} = m_{\iota,0}^{(\nu)}$' and '$m_{\text{OUT}} = m_{\iota+1,0}^{(\nu)}$' to determine the type of the outputted signature, while the program P_1 only checks whether '$m_{\text{OUT}} = m_{\iota+1,0}^{(\nu)}$'. Thus, the two programs will be functionally equivalent if we can show that for inputs corresponding to (h^*, ι), $m_{\text{OUT}} = m_{\iota+1,0}^{(\nu)}$ implies $m_{\text{IN}} = m_{\iota,0}^{(\nu)}$. Recall that in both experiment $(\text{PP}_{\text{ITR}}^{(\nu)}, v_0^{(\nu)} \xleftarrow{\$} \mathsf{ITR.Setup\text{-}Enforce}(1^\lambda, n_{\text{ITR}} = 2^\lambda, ((q_0^{(\nu)}, w_0^{(\nu)}, 0), \ldots, (q_0^{(\nu)}, w_\iota^{(\nu)}, 0)))$. Now, $m_{\text{OUT}} = m_{\iota+1,0}^{(\nu)}$ implies $v_{\text{OUT}} = v_{\iota+1}^{(\nu)}$. Therefore, by the enforcing property of ITR it follows that $v_{\text{IN}} = v_\iota^{(\nu)}$ and $(\text{ST}, w_{\text{IN}}, 0) = (q_0^{(\nu)}, w_\iota^{(\nu)}, 0)$, which in turn implies that $m_{\text{IN}} = (v_{\text{IN}}, \text{ST}, w_{\text{IN}}, 0) = (v_\iota^{(\nu)}, q_0^{(\nu)}, w_\iota^{(\nu)}, 0) = m_{\iota,0}^{(\nu)}$. □

Claim C.23 *Assuming* ITR *satisfies the 'indistinguishability of enforcing setup' property, for any* PPT *adversary* $\mathcal{A}$*, for any security parameter* λ, $|\mathit{Adv}_{\mathcal{A}}^{(0,\nu-1,3,\iota,7)}(\lambda) - \mathit{Adv}_{\mathcal{A}}^{(0,\nu-1,3,\iota,8)}(\lambda)| \leq \mathsf{negl}(\lambda)$ *for some negligible function* negl.

Proof The proof of Claim C.23 is analogous to that of Claim C.21 with some appropriate modifications which are easy to determine. □

□

Lemma C.6 *Assuming* $\mathcal{IO}$ *is a secure indistinguishability obfuscator for* $\mathsf{P/poly}$*,* $\mathcal{F}$ *is a secure puncturable pseudorandom function, and* SPS *is a splittable signature scheme satisfying '*$\text{VK}_{\text{SPS-ONE}}$ *indistinguishability', '*$\text{VK}_{\text{SPS-ABO}}$ *indistinguishability', as well as 'splitting indistinguishability', for any* PPT *adversary* $\mathcal{A}$*, for any security parameter* λ, $|\mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,3,\iota')}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,3,\iota+1)}(\lambda)| \leq \mathsf{negl}(\lambda)$ *for some negligible function* negl.

Proof In order to establish Lemma C.6, we consider the following sequence of intermediate hybrid experiments between $\mathsf{Hyb}_{0,\nu-1,3,\iota'}$ and $\mathsf{Hyb}_{0,\nu-1,3,\iota+1}$:

Sequence of Intermediate Hybrids between $\mathsf{Hyb}_{0,\nu-1,3,\iota'}$ and $\mathsf{Hyb}_{0,\nu-1,3,\iota+1}$

$\mathsf{Hyb}_{0,\nu-1,3,\iota',0}$: This experiment coincides with $\mathsf{Hyb}_{0,\nu-1,3,\iota'}$.

$\mathsf{Hyb}_{0,\nu-1,3,\iota',1}$: This experiment is identical to $\mathsf{Hyb}_{0,\nu-1,3,\iota',0}$ except that in response to the ν^{th} signing key query of $\mathcal{A}$ corresponding to TM $M^{(\nu)} \in \mathbb{M}_\lambda$ with $M^{(\nu)}(x^*) = 0$, $\mathcal{B}$ executes the following steps:

1. It first generates all the PPRF keys as well as the public parameters for the positional accumulator and the iterator as in $\mathsf{Hyb}_{0,\nu-1,3,\iota',0}$.
2. Then, it creates the punctured PPRF keys $K^{(\nu)}_{\text{SPS},E}\ \{\ (h^*,\ \iota+1)\ \} \xleftarrow{\$} \mathcal{F}.\mathsf{Puncture}(K^{(\nu)}_{\text{SPS},E},\ (h^*,\ \iota+1))$ and $K^{(\nu)}_{\text{SPS},F}\{(h^*,\ \iota+1)\} \xleftarrow{\$} \mathcal{F}.\mathsf{Puncture}(K^{(\nu)}_{\text{SPS},F}, (h^*,\iota+1))$.
3. After that it computes $r^{(\nu,\iota+1)}_{\text{SPS},G} = \mathcal{F}(K^{(\nu)}_{\text{SPS},E},\ (h^*,\iota+1))$, $r^{(\nu,\iota+1)}_{\text{SPS},H} = \mathcal{F}(K^{(\nu)}_{\text{SPS},F}, (h^*,\ \iota+1))$, and forms $(\text{SK}^{(\nu,\iota+1)}_{\text{SPS},G},\ \text{VK}^{(\nu,\iota+1)}_{\text{SPS},G},\ \text{VK}^{(\nu,\iota+1)}_{\text{SPS-REJ},G}) = \mathsf{SPS.Setup}(1^\lambda; r^{(\nu,\iota+1)}_{\text{SPS},G})$, $(\text{SK}^{(\nu,\iota+1)}_{\text{SPS},H},\ \text{VK}^{(\nu,\iota+1)}_{\text{SPS},H},\ \text{VK}^{(\nu,\iota+1)}_{\text{SPS-REJ},H}) = \mathsf{SPS.Setup1}^\lambda;\ r^{(\nu,\iota+1)}_{\text{SPS},H})$.
4. Next, it sets $m^{(\nu)}_{0,0} = (v^{(\nu)}_0, q^{(\nu)}_0, w^{(\nu)}_0, 0)$. For $j = 1,\ldots,\iota+1$, it iteratively computes the following:
   - $\text{AUX}^{(\nu)}_j = \mathsf{ACC.Prep\text{-}Write}(\text{PP}^{(\nu)}_{\text{ACC}}, \text{STORE}^{(\nu)}_{j-1}, j-1)$
   - $w^{(\nu)}_j = \mathsf{ACC.Update}(\text{PP}^{(\nu)}_{\text{ACC}}, w^{(\nu)}_{j-1}, x^*_{j-1}, j-1, \text{AUX}^{(\nu)}_j)$
   - $\text{STORE}^{(\nu)}_j = \mathsf{ACC.Write\text{-}Store}(\text{PP}^{(\nu)}_{\text{ACC}}, \text{STORE}^{(\nu)}_{j-1}, j-1, x^*_{j-1})$
   - $v^{(\nu)}_j = \mathsf{ITR.Iterate}(\text{PP}^{(\nu)}_{\text{ITR}}, v^{(\nu)}_{j-1}, (q^{(\nu)}_0, w^{(\nu)}_{j-1}, 0))$

   It sets $m^{(\nu)}_{\iota+1,0} = (v^{(\nu)}_{\iota+1}, q^{(\nu)}_0, w^{(\nu)}_{\iota+1}, 0)$.
5. It gives $\mathcal{A}$ the signing key
$$\text{SK}_{\text{ABS}}\{M^{(\nu)}\} = \left(\begin{array}{l} \text{HK}, \text{PP}^{(\nu)}_{\text{ACC}}, w^{(\nu)}_0, \text{STORE}^{(\nu)}_0, \text{PP}^{(\nu)}_{\text{ITR}}, v^{(\nu)}_0, \\ \mathcal{IO}(\mathsf{Init\text{-}SPS.Prog}[q^{(\nu)}_0, w^{(\nu)}_0, v^{(\nu)}_0, K^{(\nu)}_{\text{SPS},E}\{(h^*,\iota+1)\}]), \\ \mathcal{IO}(\mathsf{Accumulate.Prog}^{(3,\iota',1)}[n_{\text{SSB-BLK}} = 2^\lambda, \text{HK}, \text{PP}^{(\nu)}_{\text{ACC}}, \text{PP}^{(\nu)}_{\text{ITR}}, \\ \quad K^{(\nu)}_{\text{SPS},E}\{(h^*,\iota+1)\}, K^{(\nu)}_{\text{SPS},F}\{(h^*,\iota+1)\}, \text{SK}^{(\nu,\iota+1)}_{\text{SPS},G}, \text{SK}^{(\nu,\iota+1)}_{\text{SPS},H}, \\ \quad \text{VK}^{(\nu,\iota+1)}_{\text{SPS},G}, \text{VK}^{(\nu,\iota+1)}_{\text{SPS},H}, m^{(\nu)}_{\iota+1,0}, h^*, \ell^*]), \\ \mathcal{IO}(\mathsf{Change\text{-}SPS.Prog}^{(3,\iota,1)}[K^{(\nu)}_{\text{SPS},A}, K^{(\nu)}_{\text{SPS},B}, K^{(\nu)}_{\text{SPS},E}\{(h^*,\iota+1)\}, \\ \quad K^{(\nu)}_{\text{SPS},F}\{(h^*,\iota+1)\}, \text{VK}^{(\nu,\iota+1)}_{\text{SPS},G}, \text{VK}^{(\nu,\iota+1)}_{\text{SPS},H}, h^*, \ell^*]), \\ \mathcal{IO}(\mathsf{Constrained\text{-}Key.Prog}^{(1)}_{\text{ABS}}[M^{(\nu)}, T = 2^\lambda, t^{*(\nu)}, \text{PP}^{(\nu)}_{\text{ACC}}, \text{PP}^{(\nu)}_{\text{ITR}}, K, \\ \quad K^{(\nu)}_1, \ldots, K^{(\nu)}_\lambda, K^{(\nu)}_{\text{SPS},A}, K^{(\nu)}_{\text{SPS},B}, h^*, \ell^*]) \end{array}\right),$$
   where the programs $\mathsf{Accumulate.Prog}^{(3,\iota',1)}$ and $\mathsf{Change\text{-}SPS.Prog}^{(3,\iota,1)}$ respectively are the modifications of the programs $\mathsf{Accumulate.Prog}^{(3,\iota')}$ and $\mathsf{Change\text{-}SPS.Prog}^{(3,\iota)}$ (Figs. **17** and **16**) and are shown in Figs. **33** and **34**.

$\mathsf{Hyb}_{0,\nu-1,3,\iota',2}$: This experiment is analogous to $\mathsf{Hyb}_{0,\nu-1,3,\iota',1}$ with the only exception that while constructing the ν^{th} signing key queried by $\mathcal{A}$, $\mathcal{B}$ selects $r^{(\nu,\iota+1)}_{\text{SPS},G}$, $r^{(\nu,\iota+1)}_{\text{SPS},H} \xleftarrow{\$} \mathcal{Y}_{\text{PPRF}}$, i.e., in other words, $\mathcal{B}$ generates $(\text{SK}^{(\nu,\iota+1)}_{\text{SPS},G}, \text{VK}^{(\nu,\iota+1)}_{\text{SPS},G}, \text{VK}^{(\nu,\iota+1)}_{\text{SPS-REJ},G})$, $(\text{SK}^{(\nu,\iota+1)}_{\text{SPS},H}, \text{VK}^{(\nu,\iota+1)}_{\text{SPS},H}, \text{VK}^{(\nu,\iota+1)}_{\text{SPS-REJ},H}) \xleftarrow{\$} \mathsf{SPS.Setup}(1^\lambda)$.

$\mathsf{Hyb}_{0,\nu-1,3,\iota',3}$: This experiment is identical to $\mathsf{Hyb}_{0,\nu-1,3,\iota',2}$ except that in response to the ν^{th} signing key query of $\mathcal{A}$ corresponding to TM $M^{(\nu)} \in \mathbb{M}_\lambda$ with $M^{(\nu)}(x^*) = 0$, $\mathcal{B}$ executes the following steps:

1. It first generates all the PPRF keys as well as the public parameters for the positional accumulator and the iterator as in $\mathsf{Hyb}_{0,\nu-1,3,\iota',2}$.

Constants: Maximum number of blocks for SSB hash $n_{\text{SSB-BLK}} = 2^\lambda$, SSB hash key HK, Public parameters for positional accumulator PP_{ACC}, Public parameters for iterator PP_{ITR}, Punctured PPRF keys $K_{\text{SPS},E}\{(h^*, \iota+1)\}, K_{\text{SPS},F}\{(h^*, \iota+1)\}$, Signing keys SK_G, SK_H, Verification keys VK_G, VK_H, Message $m_{\iota+1,0}$, SSB hash value of challenge input h^*, Length of challenge input ℓ^*

Inputs: Index i, Symbol SYM_{IN}, TM state ST, Accumulator value w_{IN}, Auxiliary value AUX, Iterator value v_{IN}, Signature $\sigma_{\text{SPS,IN}}$, SSB hash value h, SSB opening value π_{SSB}

Output: (Accumulator value w_{OUT}, Iterator value v_{OUT}, Signature $\sigma_{\text{SPS-OUT}}$), or $\perp$

1. (a) If $(h,i) \neq (h^*, \iota+1)$, compute $r_{\text{SPS},E} = \mathcal{F}(K_{\text{SPS},E}\{(h^*, \iota+1)\}, (h,i))$, $(\text{SK}_{\text{SPS},E}, \text{VK}_{\text{SPS},E}, \text{VK}_{\text{SPS-REJ},E}) = \mathsf{SPS.Setup}(1^\lambda; r_{\text{SPS},E})$.
   Else, set $\text{VK}_{\text{SPS},E} = \text{VK}_G$.
   (b) If $(h,i) \neq (h^*, \iota+1)$, compute $r_{\text{SPS},F} = \mathcal{F}(K_{\text{SPS},F}\{(h^*, \iota+1)\}, (h,i))$, $(\text{SK}_{\text{SPS},F}, \text{VK}_{\text{SPS},F}, \text{VK}_{\text{SPS-REJ},F}) = \mathsf{SPS.Setup}(1^\lambda; r_{\text{SPS},F})$.
   Else, set $\text{VK}_{\text{SPS},F} = \text{VK}_H$.
   (c) Set $m_{\text{IN}} = (v_{\text{IN}}, \text{ST}, w_{\text{IN}}, 0)$ and $\alpha =$'-'.
   (d) If $\mathsf{SPS.Verify}(\text{VK}_{\text{SPS},E}, m_{\text{IN}}, \sigma_{\text{SPS,IN}}) = 1$, set $\alpha =$'E'.
   (e) If $[\alpha =\text{'-'}] \wedge [(i > \ell^*) \vee (0 \leq i \leq \iota) \vee (h \neq h^*)]$, output $\perp$.
   Else if $[\alpha =\text{'-'}] \wedge [\mathsf{SPS.Verify}(\text{VK}_{\text{SPS},F}, m_{\text{IN}}, \sigma_{\text{SPS,IN}}) = 1]$, set $\alpha =$'F'.
   (f) If $\alpha =$'-', output $\perp$.
2. If $\mathsf{SSB.Verify}(\text{HK}, h, i, \text{SYM}_{\text{IN}}, \pi_{\text{SSB}}) = 0$, output $\perp$.
3. (a) Compute $w_{\text{OUT}} = \mathsf{ACC.Update}(\text{PP}_{\text{ACC}}, w_{\text{IN}}, \text{SYM}_{\text{IN}}, i, \text{AUX})$. If $w_{\text{OUT}} = \perp$, output $\perp$.
   (b) Compute $v_{\text{OUT}} = \mathsf{ITR.Iterate}(\text{PP}_{\text{ITR}}, v_{\text{IN}}, (\text{ST}, w_{\text{IN}}, 0))$.
4. (a) If $(h,i) \neq (h^*, \iota)$, compute $r'_{\text{SPS},E} = \mathcal{F}(K_{\text{SPS},E}\{(h^*, \iota+1)\}, (h, i+1))$, $(\text{SK}'_{\text{SPS},E}, \text{VK}'_{\text{SPS},E}, \text{VK}'_{\text{SPS-REJ},E}) = \mathsf{SPS.Setup}(1^\lambda; r'_{\text{SPS},E})$.
   Else, set $\text{SK}'_{\text{SPS},E} = \text{SK}_G$.
   (b) If $(h,i) \neq (h^*, \iota)$, compute $r'_{\text{SPS},F} = \mathcal{F}(K_{\text{SPS},F}\{(h^*, \iota+1)\}, (h, i+1))$, $(\text{SK}'_{\text{SPS},F}, \text{VK}'_{\text{SPS},F}, \text{VK}'_{\text{SPS-REJ},F}) = \mathsf{SPS.Setup}(1^\lambda; r'_{\text{SPS},F})$.
   Else, set $\text{SK}'_{\text{SPS},F} = \text{SK}_H$.
   (c) Set $m_{\text{OUT}} = (v_{\text{OUT}}, \text{ST}, w_{\text{OUT}}, 0)$. If $[(h,i) = (h^*, \iota)] \wedge [m_{\text{OUT}} = m_{\iota+1,0}]$, compute $\sigma_{\text{SPS,OUT}} = \mathsf{SPS.Sign}(\text{SK}'_{\text{SPS},E}, m_{\text{OUT}})$.
   Else if $[(h,i) = (h^*, \iota)] \wedge [m_{\text{OUT}} \neq m_{\iota+1,0}]$, compute $\sigma_{\text{SPS,OUT}} = \mathsf{SPS.Sign}(\text{SK}'_{\text{SPS},F}, m_{\text{OUT}})$.
   Else if $i < \ell^*$, compute $\sigma_{\text{SPS,OUT}} = \mathsf{SPS.Sign}(\text{SK}'_{\text{SPS},\alpha}, m_{\text{OUT}})$.
   Else, compute $\sigma_{\text{SPS,OUT}} = \mathsf{SPS.Sign}(\text{SK}'_{\text{SPS},E}, m_{\text{OUT}})$.
5. Output $(w_{\text{OUT}}, v_{\text{OUT}}, \sigma_{\text{SPS,OUT}})$.

Fig. 33 $\mathsf{Accumulate.Prog}^{(3,\iota',1)}$

2. Then, it creates the punctured PPRF keys $K^{(\nu)}_{\text{SPS},E}\{(h^*, \iota+1)\} \xleftarrow{\$} \mathcal{F}.\mathsf{Puncture}(K^{(\nu)}_{\text{SPS},E}, (h^*, \iota+1))$ and $K^{(\nu)}_{\text{SPS},F}\{(h^*, \iota+1)\} \xleftarrow{\$} \mathcal{F}.\mathsf{Puncture}(K^{(\nu)}_{\text{SPS},F}, (h^*, \iota+1))$.
3. After that it forms $(\text{SK}^{(\nu,\iota+1)}_{\text{SPS},G}, \text{VK}^{(\nu,\iota+1)}_{\text{SPS},G}, \text{VK}^{(\nu,\iota+1)}_{\text{SPS-REJ},G}), (\text{SK}^{(\nu,\iota+1)}_{\text{SPS},H}, \text{VK}^{(\nu,\iota+1)}_{\text{SPS},H}, \text{VK}^{(\nu,\iota+1)}_{\text{SPS-REJ},H}) \xleftarrow{\$} \mathsf{SPS.Setup}(1^\lambda)$.
4. Next, it computes $m^{(\nu)}_{\iota+1,0} = (v^{(\nu)}_{\iota+1}, q^{(\nu)}_0, w^{(\nu)}_{\iota+1}, 0)$ just as in $\mathsf{Hyb}_{0,\nu-1,3,\iota',2}$.
5. After that, it creates $(\sigma^{(\nu,\iota+1)}_{\text{SPS-ONE},m^{(\nu)}_{\iota+1,0},G}, \text{VK}^{(\nu,\iota+1)}_{\text{SPS-ONE},G}, \text{SK}^{(\nu,\iota+1)}_{\text{SPS-ABO},G}, \text{VK}^{(\nu,\iota+1)}_{\text{SPS-ABO},G}) \xleftarrow{\$} \mathsf{SPS.Split}(\text{SK}^{(\nu,\iota+1)}_{\text{SPS},G}, m^{(\nu)}_{\iota+1,0})$ and $(\sigma^{(\nu,\iota+1)}_{\text{SPS-ONE},m^{(\nu)}_{\iota+1,0},H}, \text{VK}^{(\nu,\iota+1)}_{\text{SPS-ONE},H}, \text{SK}^{(\nu,\iota+1)}_{\text{SPS-ABO},H}, \text{VK}^{(\nu,\iota+1)}_{\text{SPS-ABO},H}) \xleftarrow{\$} \mathsf{SPS.Split}(\text{SK}^{(\nu,\iota+1)}_{\text{SPS},H}, m^{(\nu)}_{\iota+1,0})$.

Constants: PPRF keys $K_{\text{SPS},A}$, $K_{\text{SPS},B}$, Punctured PPRF keys $K_{\text{SPS},E}\{(h^*,\iota+1)\}$, $K_{\text{SPS},F}\{(h^*,\iota+1)\}$, Verification keys VK_G, VK_H, SSB hash value of challenge input h^*, Length of challenge input ℓ^*

Inputs: TM state ST, Accumulator value w, Iterator value v, SSB hash value h, Length ℓ_{INP}, Signature $\sigma_{\text{SPS,IN}}$

Output: Signature $\sigma_{\text{SPS,OUT}}$, or $\perp$

1. (a) If $(h,\ell_{\text{INP}}) \neq (h^*,\iota+1)$, compute $r_{\text{SPS},E} = \mathcal{F}(K_{\text{SPS},E}\{(h^*,\iota+1)\},(h,\ell_{\text{INP}}))$, $(\text{SK}_{\text{SPS},E},\text{VK}_{\text{SPS},E},\text{VK}_{\text{SPS-REJ},E}) = \mathsf{SPS.Setup}(1^\lambda; r_{\text{SPS},E})$.
   Else, set $\text{VK}_{\text{SPS},E} = \text{VK}_G$.
   (b) If $(h,\ell_{\text{INP}}) \neq (h^*,\iota+1)$, compute $r_{\text{SPS},F} = \mathcal{F}(K_{\text{SPS},F}\{(h^*,\iota+1)\},(h,\ell_{\text{INP}}))$, $(\text{SK}_{\text{SPS},F},\text{VK}_{\text{SPS},F},\text{VK}_{\text{SPS-REJ},F}) = \mathsf{SPS.Setup}(1^\lambda; r_{\text{SPS},F})$.
   Else, set $\text{VK}_{\text{SPS},F} = \text{VK}_H$.
   (c) Set $m = (v,\text{ST},w,0)$ and α='-'.
   (d) If $\mathsf{SPS.Verify}(\text{VK}_{\text{SPS},E},m,\sigma_{\text{SPS,IN}}) = 1$, set α ='E'.
   (e) If $[\alpha$ ='-'$] \wedge [(\ell_{\text{INP}} > \ell^*) \vee (0 < \ell_{\text{INP}} \leq \iota) \vee (h \neq h^*)]$, output $\perp$.
   Else if $[\alpha$ ='-'$] \wedge [\mathsf{SPS.Verify}(\text{VK}_{\text{SPS},F},m,\sigma_{\text{SPS,IN}}) = 1]$, set α ='F'.
   (f) If α ='-', output $\perp$.
2. (a) Compute $r_{\text{SPS},A} = \mathcal{F}(K_{\text{SPS},A},(h,\ell_{\text{INP}},0)),(\text{SK}_{\text{SPS},A},\text{VK}_{\text{SPS},A},\text{VK}_{\text{SPS-REJ},A}) = \mathsf{SPS.Setup}(1^\lambda; r_{\text{SPS},A})$.
   (b) Compute $r_{\text{SPS},B} = \mathcal{F}(K_{\text{SPS},B},(h,\ell_{\text{INP}},0)),(\text{SK}_{\text{SPS},B},\text{VK}_{\text{SPS},B},\text{VK}_{\text{SPS-REJ},B}) = \mathsf{SPS.Setup}(1^\lambda; r_{\text{SPS},B})$.
   (c) If $[(h,\ell_{\text{INP}}) = (h^*,\ell^*)] \wedge [\alpha$ ='F'$]$, output $\sigma_{\text{SPS,OUT}} = \mathsf{SPS.Sign}(\text{SK}_{\text{SPS},B},m)$.
   Else, output $\sigma_{\text{SPS,OUT}} = \mathsf{SPS.Sign}(\text{SK}_{\text{SPS},A},m)$.

Fig. 34 $\mathsf{Change\text{-}SPS.Prog}^{(3,\iota,1)}$

6. It gives $\mathcal{A}$ the signing key

$$\text{SK}_{\text{ABS}}\{M^{(\nu)}\} = \left(\begin{array}{l} \text{HK}, \text{PP}_{\text{ACC}}^{(\nu)}, w_0^{(\nu)}, \text{STORE}_0^{(\nu)}, \text{PP}_{\text{ITR}}^{(\nu)}, v_0^{(\nu)}, \\ \mathcal{IO}(\mathsf{Init\text{-}SPS.Prog}[q_0^{(\nu)}, w_0^{(\nu)}, v_0^{(\nu)}, K_{\text{SPS},E}^{(\nu)}\{(h^*,\iota+1)\}]), \\ \mathcal{IO}(\mathsf{Accumulate.Prog}^{(3,\iota',2)}[n_{\text{SSB-BLK}} = 2^\lambda, \text{HK}, \text{PP}_{\text{ACC}}^{(\nu)}, \text{PP}_{\text{ITR}}^{(\nu)}, \\ \quad K_{\text{SPS},E}^{(\nu)}\{(h^*,\iota+1)\}, K_{\text{SPS},F}^{(\nu)}\{(h^*,\iota+1)\}, \sigma_{\text{SPS-ONE},m_{\iota+1,0}^{(\nu)},G}^{(\nu,\iota+1)}, \\ \quad \text{SK}_{\text{SPS-ABO},H}^{(\nu,\iota+1)}, \text{VK}_{\text{SPS},G}^{(\nu,\iota+1)}, \text{VK}_{\text{SPS},H}^{(\nu,\iota+1)}, m_{\iota+1,0}^{(\nu)}, h^*, \ell^*]), \\ \mathcal{IO}(\mathsf{Change\text{-}SPS.Prog}^{(3,\iota,1)}[K_{\text{SPS},A}^{(\nu)}, K_{\text{SPS},B}^{(\nu)}, K_{\text{SPS},E}^{(\nu)}\{(h^*,\iota+1)\}, \\ \quad K_{\text{SPS},F}^{(\nu)}\{(h^*,\iota+1)\}, \text{VK}_{\text{SPS},G}^{(\nu,\iota+1)}, \text{VK}_{\text{SPS},H}^{(\nu,\iota+1)}, h^*, \ell^*]), \\ \mathcal{IO}(\mathsf{Constrained\text{-}Key.Prog}_{\text{ABS}}^{(1)}[M^{(\nu)}, T = 2^\lambda, t^{*(\nu)}, \text{PP}_{\text{ACC}}^{(\nu)}, \text{PP}_{\text{ITR}}^{(\nu)}, K, \\ \quad K_1^{(\nu)}, \ldots, K_\lambda^{(\nu)}, K_{\text{SPS},A}^{(\nu)}, K_{\text{SPS},B}^{(\nu)}, h^*, \ell^*]) \end{array}\right),$$

where the program $\mathsf{Accumulate.Prog}^{(3,\iota',2)}$ is an alteration of the program $\mathsf{Accumulate.Prog}^{(3,\iota',1)}$ (Fig. **33**) and is shown in Fig. **35**.

$\mathsf{Hyb}_{\mathbf{0},\nu-\mathbf{1},\mathbf{3},\iota',\mathbf{4}}$: In this experiment, in response to the ν^{th} signing key query of $\mathcal{A}$ corresponding to TM $M^{(\nu)} \in \mathbb{M}_\lambda$ with $M^{(\nu)}(x^*) = 0$, $\mathcal{B}$ generates everything as in

Constants: Maximum number of blocks for SSB hash $n_{\text{SSB-BLK}} = 2^\lambda$, SSB hash key HK, Public parameters for positional accumulator PP_{ACC}, Public parameters for iterator PP_{ITR}, Punctured PPRF keys $K_{\text{SPS},E}\{(h^*, \iota{+}1)\}$, $K_{\text{SPS},F}\{(h^*, \iota{+}1)\}$, Signature $\underline{\sigma_G}$, Signing key SK_H, Verification keys VK_G, VK_H, Message $m_{\iota+1,0}$, SSB hash value of challenge input h^*, Length of challenge input ℓ^*

Inputs: Index i, Symbol SYM_{IN}, TM state ST, Accumulator value w_{IN}, Auxiliary value AUX, Iterator value v_{IN}, Signature $\sigma_{\text{SPS,IN}}$, SSB hash value h, SSB opening value π_{SSB}

Output: (Accumulator value w_{OUT}, Iterator value v_{OUT}, Signature $\sigma_{\text{SPS-OUT}}$), or $\perp$

1. (a) If $(h,i) \neq (h^*, \iota+1)$, compute $r_{\text{SPS},E} = \mathcal{F}(K_{\text{SPS},E}\{(h^*, \iota+1)\}, (h,i))$, $(\text{SK}_{\text{SPS},E}, \text{VK}_{\text{SPS},E}, \text{VK}_{\text{SPS-REJ},E}) = \mathsf{SPS.Setup}(1^\lambda; r_{\text{SPS},E})$.
   Else, set $\text{VK}_{\text{SPS},E} = \text{VK}_G$.
   (b) If $(h,i) \neq (h^*, \iota+1)$, compute $r_{\text{SPS},F} = \mathcal{F}(K_{\text{SPS},F}\{(h^*, \iota+1)\}, (h,i))$, $(\text{SK}_{\text{SPS},F}, \text{VK}_{\text{SPS},F}, \text{VK}_{\text{SPS-REJ},F}) = \mathsf{SPS.Setup}(1^\lambda; r_{\text{SPS},F})$.
   Else, set $\text{VK}_{\text{SPS},F} = \text{VK}_H$.
   (c) Set $m_{\text{IN}} = (v_{\text{IN}}, \text{ST}, w_{\text{IN}}, 0)$ and $\alpha =$'-'.
   (d) If $\mathsf{SPS.Verify}(\text{VK}_{\text{SPS},E}, m_{\text{IN}}, \sigma_{\text{SPS,IN}}) = 1$, set $\alpha =$'E'.
   (e) If $[\alpha =\text{'-'}] \wedge [(i > \ell^*) \vee (0 \leq i \leq \iota) \vee (h \neq h^*)]$, output $\perp$.
   Else if $[\alpha =\text{'-'}] \wedge [\mathsf{SPS.Verify}(\text{VK}_{\text{SPS},F}, m_{\text{IN}}, \sigma_{\text{SPS,IN}}) = 1]$, set $\alpha =$'F'.
   (f) If $\alpha =$'-', output $\perp$.
2. If $\mathsf{SSB.Verify}(\text{HK}, h, i, \text{SYM}_{\text{IN}}, \pi_{\text{SSB}}) = 0$, output $\perp$.
3. (a) Compute $w_{\text{OUT}} = \mathsf{ACC.Update}(\text{PP}_{\text{ACC}}, w_{\text{IN}}, \text{SYM}_{\text{IN}}, i, \text{AUX})$. If $w_{\text{OUT}} = \perp$, output $\perp$.
   (b) Compute $v_{\text{OUT}} = \mathsf{ITR.Iterate}(\text{PP}_{\text{ITR}}, v_{\text{IN}}, (\text{ST}, w_{\text{IN}}, 0))$.
4. (a) If $(h,i) \neq (h^*, \iota)$, compute $r'_{\text{SPS},E} = \mathcal{F}(K_{\text{SPS},E}\{(h^*, \iota+1)\}, (h, i+1))$, $(\text{SK}'_{\text{SPS},E}, \text{VK}'_{\text{SPS},E}, \text{VK}'_{\text{SPS-REJ},E}) = \mathsf{SPS.Setup}(1^\lambda; r'_{\text{SPS},E})$.
   (b) If $(h,i) \neq (h^*, \iota)$, compute $r'_{\text{SPS},F} = \mathcal{F}(K_{\text{SPS},F}\{(h^*, \iota+1)\}, (h, i+1))$, $(\text{SK}'_{\text{SPS},F}, \text{VK}'_{\text{SPS},F}, \text{VK}'_{\text{SPS-REJ},F}) = \mathsf{SPS.Setup}(1^\lambda; r'_{\text{SPS},F})$.
   Else, set $\text{SK}'_{\text{SPS},F} = \text{SK}_H$.
   (c) Set $m_{\text{OUT}} = (v_{\text{OUT}}, \text{ST}, w_{\text{OUT}}, 0)$. If $[(h,i) = (h^*, \iota)] \wedge [m_{\text{OUT}} = m_{\iota+1,0}]$, set $\underline{\sigma_{\text{SPS,OUT}} = \sigma_G}$.
   Else if $[(h,i) = (h^*, \iota)] \wedge [m_{\text{OUT}} \neq m_{\iota+1,0}]$, compute $\sigma_{\text{SPS,OUT}} = \mathsf{SPS.Sign}(\text{SK}'_{\text{SPS},F}, m_{\text{OUT}})$.
   Else if $i < \ell^*$, compute $\sigma_{\text{SPS,OUT}} = \mathsf{SPS.Sign}(\text{SK}'_{\text{SPS},\alpha}, m_{\text{OUT}})$.
   Else, compute $\sigma_{\text{SPS,OUT}} = \mathsf{SPS.Sign}(\text{SK}'_{\text{SPS},E}, m_{\text{OUT}})$.
5. Output $(w_{\text{OUT}}, v_{\text{OUT}}, \sigma_{\text{SPS,OUT}})$.

Fig. 35 $\mathsf{Accumulate.Prog}^{(3,\iota',2)}$

$\mathsf{Hyb}_{0,\nu-1,3,\iota',3}$, however, it hands $\mathcal{A}$ the signing key

$$\text{SK}_{\text{ABS}}\{M^{(\nu)}\} = \left(\begin{array}{l} \text{HK}, \text{PP}_{\text{ACC}}^{(\nu)}, w_0^{(\nu)}, \text{STORE}_0^{(\nu)}, \text{PP}_{\text{ITR}}^{(\nu)}, v_0^{(\nu)}, \\ \mathcal{IO}(\mathsf{Init\text{-}SPS.Prog}[q_0^{(\nu)}, w_0^{(\nu)}, v_0^{(\nu)}, K_{\text{SPS},E}^{(\nu)}\{(h^*, \iota+1)\}]), \\ \mathcal{IO}(\mathsf{Accumulate.Prog}^{(3,\iota',2)}[n_{\text{SSB-BLK}} = 2^\lambda, \text{HK}, \text{PP}_{\text{ACC}}^{(\nu)}, \text{PP}_{\text{ITR}}^{(\nu)}, \\ \quad K_{\text{SPS},E}^{(\nu)}\{(h^*, \iota+1)\}, K_{\text{SPS},F}^{(\nu)}\{(h^*, \iota+1)\}, \sigma^{(\nu,\iota+1)}_{\text{SPS-ONE},m_{\iota+1,0}^{(\nu)},G}, \text{SK}_{\text{SPS-ABO},H}^{(\nu,\iota+1)}, \\ \quad \underline{\text{VK}_{\text{SPS-ONE},G}^{(\nu,\iota+1)}}, \text{VK}_{\text{SPS},H}^{(\nu,\iota+1)}, m_{\iota+1,0}^{(\nu)}, h^*, \ell^*]), \\ \mathcal{IO}(\mathsf{Change\text{-}SPS.Prog}^{(3,\iota,1)}[K_{\text{SPS},A}^{(\nu)}, K_{\text{SPS},B}^{(\nu)}, K_{\text{SPS},E}^{(\nu)}\{(h^*, \iota+1)\}, \\ \quad K_{\text{SPS},F}^{(\nu)}\{(h^*, \iota+1)\}, \underline{\text{VK}_{\text{SPS-ONE},G}^{(\nu,\iota+1)}}, \text{VK}_{\text{SPS},H}^{(\nu,\iota+1)}, h^*, \ell^*]), \\ \mathcal{IO}(\mathsf{Constrained\text{-}Key.Prog}^{(1)}_{\text{ABS}}[M^{(\nu)}, T = 2^\lambda, t^{*(\nu)}, \text{PP}_{\text{ACC}}^{(\nu)}, \text{PP}_{\text{ITR}}^{(\nu)}, K, \\ \quad K_1^{(\nu)}, \ldots, K_\lambda^{(\nu)}, K_{\text{SPS},A}^{(\nu)}, K_{\text{SPS},B}^{(\nu)}, h^*, \ell^*]) \end{array}\right).$$

The rest of the experiment is analogous to $\mathsf{Hyb}_{0,\nu-1,3,\iota',3}$.

$\mathsf{Hyb}_{0,\nu-1,3,\iota',5}$: In this experiment, in response to the ν^{th} signing key query of $\mathcal{A}$ corresponding to TM $M^{(\nu)} \in \mathbb{M}_\lambda$ with $M^{(\nu)}(x^*) = 0$, $\mathcal{B}$ forms all the components

just as in $\mathsf{Hyb}_{0,\nu-1,3,\iota',4}$, however, it gives $\mathcal{A}$ the signing key

$$\mathrm{SK}_{\mathrm{ABS}}\{M^{(\nu)}\} = \left(\begin{array}{l} \mathrm{HK}, \mathrm{PP}_{\mathrm{ACC}}^{(\nu)}, w_0^{(\nu)}, \mathrm{STORE}_0^{(\nu)}, \mathrm{PP}_{\mathrm{ITR}}^{(\nu)}, v_0^{(\nu)}, \\ \mathcal{IO}(\mathsf{Init\text{-}SPS.Prog}[q_0^{(\nu)}, w_0^{(\nu)}, v_0^{(\nu)}, K_{\mathrm{SPS},E}^{(\nu)}\{(h^*, \iota+1)\}]), \\ \mathcal{IO}(\mathsf{Accumulate.Prog}^{(3,\iota',2)}[n_{\text{SSB-BLK}} = 2^\lambda, \mathrm{HK}, \mathrm{PP}_{\mathrm{ACC}}^{(\nu)}, \mathrm{PP}_{\mathrm{ITR}}^{(\nu)}, \\ \quad K_{\mathrm{SPS},E}^{(\nu)}\{(h^*, \iota+1)\}, K_{\mathrm{SPS},F}^{(\nu)}\{(h^*, \iota+1)\}, \sigma^{(\nu,\iota+1)}_{\text{SPS-ONE},m_{\iota+1,0}^{(\nu)},G}, \mathrm{SK}_{\text{SPS-ABO},H}^{(\nu,\iota+1)}, \\ \quad \mathrm{VK}_{\text{SPS-ONE},G}^{(\nu,\iota+1)}, \underline{\mathrm{VK}_{\text{SPS-ABO},H}^{(\nu,\iota+1)}}, m_{\iota+1,0}^{(\nu)}, h^*, \ell^*]), \\ \mathcal{IO}(\mathsf{Change\text{-}SPS.Prog}^{(3,\iota,1)}[K_{\mathrm{SPS},A}^{(\nu)}, K_{\mathrm{SPS},B}^{(\nu)}, K_{\mathrm{SPS},E}^{(\nu)}\{(h^*, \iota+1)\}, \\ \quad K_{\mathrm{SPS},F}^{(\nu)}\{(h^*, \iota+1)\}, \mathrm{VK}_{\text{SPS-ONE},G}^{(\nu,\iota+1)}, \underline{\mathrm{VK}_{\text{SPS-ABO},H}^{(\nu,\iota+1)}}, h^*, \ell^*]), \\ \mathcal{IO}(\mathsf{Constrained\text{-}Key.Prog}_{\mathrm{ABS}}^{(1)}[M^{(\nu)}, T = 2^\lambda, t^{*(\nu)}, \mathrm{PP}_{\mathrm{ACC}}^{(\nu)}, \mathrm{PP}_{\mathrm{ITR}}^{(\nu)}, K, \\ \quad K_1^{(\nu)}, \ldots, K_\lambda^{(\nu)}, K_{\mathrm{SPS},A}^{(\nu)}, K_{\mathrm{SPS},B}^{(\nu)}, h^*, \ell^*]) \end{array}\right).$$

The rest of the experiment is analogous to $\mathsf{Hyb}_{0,\nu-1,3,\iota',4}$.

$\mathsf{Hyb}_{0,\nu-1,3,\iota',6}$: In this experiment, to answer the ν^{th} signing key query of $\mathcal{A}$ corresponding to TM $M^{(\nu)} \in \mathbb{M}_\lambda$ with $M^{(\nu)}(x^*) = 0$, $\mathcal{B}$ creates all the components as in $\mathsf{Hyb}_{0,\nu-1,3,\iota',5}$, however, it returns the signing key

$$\mathrm{SK}_{\mathrm{ABS}}\{M^{(\nu)}\} = \left(\begin{array}{l} \mathrm{HK}, \mathrm{PP}_{\mathrm{ACC}}^{(\nu)}, w_0^{(\nu)}, \mathrm{STORE}_0^{(\nu)}, \mathrm{PP}_{\mathrm{ITR}}^{(\nu)}, v_0^{(\nu)}, \\ \mathcal{IO}(\mathsf{Init\text{-}SPS.Prog}[q_0^{(\nu)}, w_0^{(\nu)}, v_0^{(\nu)}, K_{\mathrm{SPS},E}^{(\nu)}\{(h^*, \iota+1)\}]), \\ \mathcal{IO}(\underline{\mathsf{Accumulate.Prog}^{(3,\iota',3)}[n_{\text{SSB-BLK}} = 2^\lambda, \mathrm{HK}, \mathrm{PP}_{\mathrm{ACC}}^{(\nu)}, \mathrm{PP}_{\mathrm{ITR}}^{(\nu)},} \\ \quad \underline{K_{\mathrm{SPS},E}^{(\nu)}\{(h^*, \iota+1)\}, K_{\mathrm{SPS},F}^{(\nu)}\{(h^*, \iota+1)\}, \sigma^{(\nu,\iota+1)}_{\text{SPS-ONE},m_{\iota+1,0}^{(\nu)},G}, \mathrm{SK}_{\text{SPS-ABO},H}^{(\nu,\iota+1)},} \\ \quad \underline{\mathrm{VK}_{\text{SPS-ONE},G}^{(\nu,\iota+1)}, \mathrm{VK}_{\text{SPS-ABO},H}^{(\nu,\iota+1)}, m_{\iota+1,0}^{(\nu)}, h^*, \ell^*]}), \\ \mathcal{IO}(\mathsf{Change\text{-}SPS.Prog}^{(3,\iota,1)}[K_{\mathrm{SPS},A}^{(\nu)}, K_{\mathrm{SPS},B}^{(\nu)}, K_{\mathrm{SPS},E}^{(\nu)}\{(h^*, \iota+1)\}, \\ \quad K_{\mathrm{SPS},F}^{(\nu)}\{(h^*, \iota+1)\}, \mathrm{VK}_{\text{SPS-ONE},G}^{(\nu,\iota+1)}, \mathrm{VK}_{\text{SPS-ABO},H}^{(\nu,\iota+1)}, h^*, \ell^*]), \\ \mathcal{IO}(\mathsf{Constrained\text{-}Key.Prog}_{\mathrm{ABS}}^{(1)}[M^{(\nu)}, T = 2^\lambda, t^{*(\nu)}, \mathrm{PP}_{\mathrm{ACC}}^{(\nu)}, \mathrm{PP}_{\mathrm{ITR}}^{(\nu)}, K, \\ \quad K_1^{(\nu)}, \ldots, K_\lambda^{(\nu)}, K_{\mathrm{SPS},A}^{(\nu)}, K_{\mathrm{SPS},B}^{(\nu)}, h^*, \ell^*]) \end{array}\right)$$

to $\mathcal{A}$, where the program $\mathsf{Accumulate.Prog}^{(3,\iota',3)}$ is a modification of the program $\mathsf{Accumulate.Prog}^{(3,\iota',2)}$ (Fig. **35**) and is depicted in Fig. **36**. The remaining part of the experiment is identical to $\mathsf{Hyb}_{0,\nu-1,3,\iota',5}$.

$\mathsf{Hyb}_{0,\nu-1,3,\iota',7}$: In this experiment, to answer the ν^{th} signing key query of $\mathcal{A}$ corresponding to TM $M^{(\nu)} \in \mathbb{M}_\lambda$ with $M^{(\nu)}(x^*) = 0$, $\mathcal{B}$ generates all the components exactly as in $\mathsf{Hyb}_{0,\nu-1,3,\iota',6}$ except that it does not generate $(\sigma^{(\nu,\iota+1)}_{\text{SPS-ONE},m_{\iota+1,0}^{(\nu)},H}, \mathrm{VK}_{\text{SPS-ONE},H}^{(\nu,\iota+1)}, \mathrm{SK}_{\text{SPS-ABO},H}^{(\nu,\iota+1)}, \mathrm{VK}_{\text{SPS-ABO},H}^{(\nu,\iota+1)}) \xleftarrow{\$} \mathsf{SPS.Split}(\mathrm{SK}_{\mathrm{SPS},H}^{(\nu,\iota+1)}, m_{\iota+1,0}^{(\nu)})$ and provides $\mathcal{A}$

Constants: Maximum number of blocks for SSB hash $n_{\text{SSB-BLK}} = 2^\lambda$, SSB hash key HK, Public parameters for positional accumulator PP_{ACC}, Public parameters for iterator PP_{ITR}, Punctured PPRF keys $K_{\text{SPS},E}\{(h^*, \iota{+}1)\}$, $K_{\text{SPS},F}\{(h^*, \iota{+}1)\}$, Signature σ_G, Signing key SK_H, Verification keys VK_G, VK_H, Message $m_{\iota+1,0}$, SSB hash value of challenge input h^*, Length of challenge input ℓ^*

Inputs: Index i, Symbol SYM_{IN}, TM state ST, Accumulator value w_{IN}, Auxiliary value AUX, Iterator value v_{IN}, Signature $\sigma_{\text{SPS,IN}}$, SSB hash value h, SSB opening value π_{SSB}

Output: (Accumulator value w_{OUT}, Iterator value v_{OUT}, Signature $\sigma_{\text{SPS-OUT}}$), or $\perp$

1. (a) If $(h,i) \neq (h^*, \iota+1)$, compute $r_{\text{SPS},E} = \mathcal{F}(K_{\text{SPS},E}\{(h^*, \iota+1)\}, (h,i))$, $(\text{SK}_{\text{SPS},E}, \text{VK}_{\text{SPS},E}, \text{VK}_{\text{SPS-REJ},E}) = \mathsf{SPS.Setup}(1^\lambda; r_{\text{SPS},E})$.
   Else, set $\text{VK}_{\text{SPS},E} = \text{VK}_G$.
   (b) If $(h,i) \neq (h^*, \iota+1)$, compute $r_{\text{SPS},F} = \mathcal{F}(K_{\text{SPS},F}\{(h^*, \iota+1)\}, (h,i))$, $(\text{SK}_{\text{SPS},F}, \text{VK}_{\text{SPS},F}, \text{VK}_{\text{SPS-REJ},F}) = \mathsf{SPS.Setup}(1^\lambda; r_{\text{SPS},F})$.
   Else, set $\text{VK}_{\text{SPS},F} = \text{VK}_H$.
   (c) Set $m_{\text{IN}} = (v_{\text{IN}}, \text{ST}, w_{\text{IN}}, 0)$ and $\alpha =$'-'.
   (d) If $\mathsf{SPS.Verify}(\text{VK}_{\text{SPS},E}, m_{\text{IN}}, \sigma_{\text{SPS,IN}}) = 1$, set $\alpha =$'E'.
   (e) If $[\alpha =\text{'-'}] \wedge [(i > \ell^*) \vee (0 \leq i \leq \iota) \vee (h \neq h^*)]$, output $\perp$.
   Else if $[\alpha =\text{'-'}] \wedge [\mathsf{SPS.Verify}(\text{VK}_{\text{SPS},F}, m_{\text{IN}}, \sigma_{\text{SPS,IN}}) = 1]$, set $\alpha =$'F'.
   (f) If $\alpha =$'-', output $\perp$.
2. If $\mathsf{SSB.Verify}(\text{HK}, h, i, \text{SYM}_{\text{IN}}, \pi_{\text{SSB}}) = 0$, output $\perp$.
3. (a) Compute $w_{\text{OUT}} = \mathsf{ACC.Update}(\text{PP}_{\text{ACC}}, w_{\text{IN}}, \text{SYM}_{\text{IN}}, i, \text{AUX})$. If $w_{\text{OUT}} = \perp$, output $\perp$.
   (b) Compute $v_{\text{OUT}} = \mathsf{ITR.Iterate}(\text{PP}_{\text{ITR}}, v_{\text{IN}}, (\text{ST}, w_{\text{IN}}, 0))$.
4. (a) If $(h,i) \neq (h^*, \iota)$, compute $r'_{\text{SPS},E} = \mathcal{F}(K_{\text{SPS},E}\{(h^*, \iota+1)\}, (h, i+1))$, $(\text{SK}'_{\text{SPS},E}, \text{VK}'_{\text{SPS},E}, \text{VK}'_{\text{SPS-REJ},E}) = \mathsf{SPS.Setup}(1^\lambda; r'_{\text{SPS},E})$.
   (b) If $(h,i) \neq (h^*, \iota)$, compute $r'_{\text{SPS},F} = \mathcal{F}(K_{\text{SPS},F}\{(h^*, \iota+1)\}, (h, i+1))$, $(\text{SK}'_{\text{SPS},F}, \text{VK}'_{\text{SPS},F}, \text{VK}'_{\text{SPS-REJ},F}) = \mathsf{SPS.Setup}(1^\lambda; r'_{\text{SPS},F})$.
   Else, set $\text{SK}'_{\text{SPS},F} = \text{SK}_H$.
   (c) Set $m_{\text{OUT}} = (v_{\text{OUT}}, \text{ST}, w_{\text{OUT}}, 0)$. If $[(h,i) = (h^*, \iota)] \wedge [m_{\text{OUT}} = m_{\iota+1,0}]$, set $\sigma_{\text{SPS,OUT}} = \sigma_G$.
   Else if $[(h,i) = (h^*, \iota)] \wedge [m_{\text{OUT}} \neq m_{\iota+1,0}]$, compute $\sigma_{\text{SPS,OUT}} = \mathsf{SPS.Sign}(\text{SK}'_{\text{SPS},F}, m_{\text{OUT}})$.
   Else if $[(h,i) = (h^*, \iota+1)] \wedge [m_{\text{IN}} = m_{\iota+1,0}]$, compute $\sigma_{\text{SPS,OUT}} = \mathsf{SPS.Sign}(\text{SK}'_{\text{SPS},E}, m_{\text{OUT}})$.
   Else if $[(h,i) = (h^*, \iota+1)] \wedge [m_{\text{IN}} \neq m_{\iota+1,0}]$, compute $\sigma_{\text{SPS,OUT}} = \mathsf{SPS.Sign}(\text{SK}'_{\text{SPS},F}, m_{\text{OUT}})$.
   Else if $i < \ell^*$, compute $\sigma_{\text{SPS,OUT}} = \mathsf{SPS.Sign}(\text{SK}'_{\text{SPS},\alpha}, m_{\text{OUT}})$.
   Else, compute $\sigma_{\text{SPS,OUT}} = \mathsf{SPS.Sign}(\text{SK}'_{\text{SPS},E}, m_{\text{OUT}})$.
5. Output $(w_{\text{OUT}}, v_{\text{OUT}}, \sigma_{\text{SPS,OUT}})$.

Fig. 36 $\mathsf{Accumulate.Prog}^{(3,\iota',3)}$

with the signing key

$$\text{SK}_{\text{ABS}}\{M^{(\nu)}\} = \left(\begin{array}{l} \text{HK}, \text{PP}_{\text{ACC}}^{(\nu)}, w_0^{(\nu)}, \text{STORE}_0^{(\nu)}, \text{PP}_{\text{ITR}}^{(\nu)}, v_0^{(\nu)}, \\ \mathcal{IO}(\mathsf{Init\text{-}SPS.Prog}[q_0^{(\nu)}, w_0^{(\nu)}, v_0^{(\nu)}, K_{\text{SPS},E}^{(\nu)}\{(h^*, \iota+1)\}]), \\ \mathcal{IO}(\mathsf{Accumulate.Prog}^{(3,\iota',3)}[n_{\text{SSB-BLK}} = 2^\lambda, \text{HK}, \text{PP}_{\text{ACC}}^{(\nu)}, \text{PP}_{\text{ITR}}^{(\nu)}, \\ \quad K_{\text{SPS},E}^{(\nu)}\{(h^*, \iota+1)\}, K_{\text{SPS},F}^{(\nu)}\{(h^*, \iota+1)\}, \sigma^{(\nu,\iota+1)}_{\text{SPS-ONE},m_{\iota+1,0}^{(\nu)},G}, \underline{\text{SK}^{(\nu,\iota+1)}_{\text{SPS-ABO},G}}, \\ \quad \text{VK}^{(\nu,\iota+1)}_{\text{SPS-ONE},G}, \underline{\text{VK}^{(\nu,\iota+1)}_{\text{SPS-ABO},G}}, m_{\iota+1,0}^{(\nu)}, h^*, \ell^*]), \\ \mathcal{IO}(\mathsf{Change\text{-}SPS.Prog}^{(3,\iota,1)}[K_{\text{SPS},A}^{(\nu)}, K_{\text{SPS},B}^{(\nu)}, K_{\text{SPS},E}^{(\nu)}\{(h^*, \iota+1)\}, \\ \quad K_{\text{SPS},F}^{(\nu)}\{(h^*, \iota+1)\}, \text{VK}^{(\nu,\iota+1)}_{\text{SPS-ONE},G}, \underline{\text{VK}^{(\nu,\iota+1)}_{\text{SPS-ABO},G}}, h^*, \ell^*]), \\ \mathcal{IO}(\mathsf{Constrained\text{-}Key.Prog}^{(1)}_{\text{ABS}}[M^{(\nu)}, T = 2^\lambda, t^{*(\nu)}, \text{PP}_{\text{ACC}}^{(\nu)}, \text{PP}_{\text{ITR}}^{(\nu)}, K, \\ \quad K_1^{(\nu)}, \ldots, K_\lambda^{(\nu)}, K_{\text{SPS},A}^{(\nu)}, K_{\text{SPS},B}^{(\nu)}, h^*, \ell^*]) \end{array}\right).$$

The rest of the experiment is the same as $\mathsf{Hyb}_{0,\nu-1,3,\iota',6}$.

$\mathbf{Hyb_{0,\nu-1,3,\iota',8}}$: In this experiment, in response to the ν^{th} signing key query of $\mathcal{A}$ corresponding to TM $M^{(\nu)} \in \mathbb{M}_\lambda$ with $M^{(\nu)}(x^*) = 0$, $\mathcal{B}$ generates all the components

as in $\mathsf{Hyb}_{0,\nu-1,3,\iota',7}$, however, it returns the signing key

$$\mathrm{SK}_{\mathrm{ABS}}\{M^{(\nu)}\} =$$

$$\left(\begin{array}{l}
\mathrm{HK}, \mathrm{PP}_{\mathrm{ACC}}^{(\nu)}, w_0^{(\nu)}, \mathrm{STORE}_0^{(\nu)}, \mathrm{PP}_{\mathrm{ITR}}^{(\nu)}, v_0^{(\nu)}, \\
\mathcal{IO}(\mathsf{Init\text{-}SPS.Prog}[q_0^{(\nu)}, w_0^{(\nu)}, v_0^{(\nu)}, K_{\mathrm{SPS},E}^{(\nu)}\{(h^*, \iota+1)\}]), \\
\mathcal{IO}(\underline{\mathsf{Accumulate.Prog}^{(3,\iota',4)}[n_{\text{SSB-BLK}} = 2^\lambda, \mathrm{HK}, \mathrm{PP}_{\mathrm{ACC}}^{(\nu)}, \mathrm{PP}_{\mathrm{ITR}}^{(\nu)},} \\
\quad \underline{K_{\mathrm{SPS},E}^{(\nu)}\{(h^*, \iota+1)\}, K_{\mathrm{SPS},F}^{(\nu)}\{(h^*, \iota+1)\}, \mathrm{SK}_{\mathrm{SPS},G}^{(\nu,\iota+1)}, \mathrm{VK}_{\mathrm{SPS},G}^{(\nu,\iota+1)},} \\
\quad \underline{m_{\iota+1,0}^{(\nu)}, h^*, \ell^*]}), \\
\mathcal{IO}(\underline{\mathsf{Change\text{-}SPS.Prog}^{(3,\iota,2)}[K_{\mathrm{SPS},A}^{(\nu)}, K_{\mathrm{SPS},B}^{(\nu)}, K_{\mathrm{SPS},E}^{(\nu)}\{(h^*, \iota+1)\},} \\
\quad \underline{K_{\mathrm{SPS},F}^{(\nu)}\{(h^*, \iota+1)\}, \mathrm{VK}_{\mathrm{SPS},G}^{(\nu,\iota+1)}, h^*, \ell^*]}), \\
\mathcal{IO}(\mathsf{Constrained\text{-}Key.Prog}_{\mathrm{ABS}}^{(1)}[M^{(\nu)}, T = 2^\lambda, t^{*(\nu)}, \mathrm{PP}_{\mathrm{ACC}}^{(\nu)}, \mathrm{PP}_{\mathrm{ITR}}^{(\nu)}, K, \\
\quad K_1^{(\nu)}, \ldots, K_\lambda^{(\nu)}, K_{\mathrm{SPS},A}^{(\nu)}, K_{\mathrm{SPS},B}^{(\nu)}, h^*, \ell^*])
\end{array}\right)$$

to $\mathcal{A}$, where the programs $\mathsf{Accumulate.Prog}^{(3,\iota',4)}$ and $\mathsf{Change\text{-}SPS.Prog}^{(3,\iota,2)}$ respectively are the modifications of the programs $\mathsf{Accumulate.Prog}^{(3,\iota',3)}$ and $\mathsf{Change\text{-}SPS.Prog}^{(3,\iota,1)}$ (Figs. **36** and **34**) and are shown in Figs. **37** and **38**. The rest of the experiment if identical to $\mathsf{Hyb}_{0,\nu-1,3,\iota',7}$.

Constants: Maximum number of blocks for SSB hash $n_{\text{SSB-BLK}} = 2^\lambda$, SSB hash key HK, Public parameters for positional accumulator $\mathrm{PP}_{\mathrm{ACC}}$, Public parameters for iterator $\mathrm{PP}_{\mathrm{ITR}}$, Punctured PPRF keys $K_{\mathrm{SPS},E}\{(h^*, \iota+1)\}, K_{\mathrm{SPS},F}\{(h^*, \iota+1)\}$, Signing key SK_G, Verification key VK_G, Message $m_{\iota+1,0}$, SSB hash value of challenge input h^*, Length of challenge input ℓ^*

Inputs: Index i, Symbol $\mathrm{SYM}_{\mathrm{IN}}$, TM state ST, Accumulator value w_{IN}, Auxiliary value AUX, Iterator value v_{IN}, Signature $\sigma_{\mathrm{SPS,IN}}$, SSB hash value h, SSB opening value π_{SSB}

Output: (Accumulator value w_{OUT}, Iterator value v_{OUT}, Signature $\sigma_{\text{SPS-OUT}}$), or $\perp$

1. (a) If $(h,i) \neq (h^*, \iota+1)$, compute $r_{\mathrm{SPS},E} = \mathcal{F}(K_{\mathrm{SPS},E}\{(h^*, \iota+1)\}, (h,i))$, $(\mathrm{SK}_{\mathrm{SPS},E}, \mathrm{VK}_{\mathrm{SPS},E}, \mathrm{VK}_{\text{SPS-REJ},E}) = \mathsf{SPS.Setup}(1^\lambda; r_{\mathrm{SPS},E})$.
   Else, set $\mathrm{VK}_{\mathrm{SPS},E} = \mathrm{VK}_G$.
   (b) If $(h,i) \neq (h^*, \iota+1)$, compute $r_{\mathrm{SPS},F} = \mathcal{F}(K_{\mathrm{SPS},F}\{(h^*, \iota+1)\}, (h,i))$, $(\mathrm{SK}_{\mathrm{SPS},F}, \mathrm{VK}_{\mathrm{SPS},F}, \mathrm{VK}_{\text{SPS-REJ},F}) = \mathsf{SPS.Setup}(1^\lambda; r_{\mathrm{SPS},F})$.
   (c) Set $m_{\mathrm{IN}} = (v_{\mathrm{IN}}, \mathrm{ST}, w_{\mathrm{IN}}, 0)$ and $\alpha =$'-'.
   (d) If $\mathsf{SPS.Verify}(\mathrm{VK}_{\mathrm{SPS},E}, m_{\mathrm{IN}}, \sigma_{\mathrm{SPS,IN}}) = 1$, set $\alpha =$'E'.
   (e) If $[\alpha =$'-'$] \wedge [(i > \ell^*) \vee \underline{(0 \leq i \leq \iota+1)} \vee (h \neq h^*)]$, output $\perp$.
   Else if $[\alpha =$'-'$] \wedge [\underline{\mathsf{SPS.Verify}(\mathrm{VK}_{\mathrm{SPS},F}, m_{\mathrm{IN}}, \sigma_{\mathrm{SPS,IN}}) = 1}]$, set $\alpha =$'F'.
   (f) If $\alpha =$'-', output $\perp$.
2. If $\mathsf{SSB.Verify}(\mathrm{HK}, h, i, \mathrm{SYM}_{\mathrm{IN}}, \pi_{\mathrm{SSB}}) = 0$, output $\perp$.
3. (a) Compute $w_{\mathrm{OUT}} = \mathsf{ACC.Update}(\mathrm{PP}_{\mathrm{ACC}}, w_{\mathrm{IN}}, \mathrm{SYM}_{\mathrm{IN}}, i, \mathrm{AUX})$. If $w_{\mathrm{OUT}} = \perp$, output $\perp$.
   (b) Compute $v_{\mathrm{OUT}} = \mathsf{ITR.Iterate}(\mathrm{PP}_{\mathrm{ITR}}, v_{\mathrm{IN}}, (\mathrm{ST}, w_{\mathrm{IN}}, 0))$.
4. (a) If $(h,i) \neq (h^*, \iota)$, compute $r'_{\mathrm{SPS},E} = \mathcal{F}(K_{\mathrm{SPS},E}\{(h^*, \iota+1)\}, (h, i+1))$, $(\mathrm{SK}'_{\mathrm{SPS},E}, \mathrm{VK}'_{\mathrm{SPS},E}, \mathrm{VK}'_{\text{SPS-REJ},E}) = \mathsf{SPS.Setup}(1^\lambda; r'_{\mathrm{SPS},E})$.
   (b) If $(h,i) \neq (h^*, \iota)$, compute $r'_{\mathrm{SPS},F} = \mathcal{F}(K_{\mathrm{SPS},F}\{(h^*, \iota+1)\}, (h, i+1))$, $(\mathrm{SK}'_{\mathrm{SPS},F}, \mathrm{VK}'_{\mathrm{SPS},F}, \mathrm{VK}'_{\text{SPS-REJ},F}) = \mathsf{SPS.Setup}(1^\lambda; r'_{\mathrm{SPS},F})$.
   (c) Set $m_{\mathrm{OUT}} = (v_{\mathrm{OUT}}, \mathrm{ST}, w_{\mathrm{OUT}}, 0)$. $\underline{\text{If } (h,i) = (h^*, \iota), \text{ compute } \sigma_{\mathrm{SPS,OUT}} = \mathsf{SPS.Sign}(\mathrm{SK}_G, m_{\mathrm{OUT}}).}$
   Else if $[(h,i) = (h^*, \iota+1)] \wedge [m_{\mathrm{IN}} = m_{\iota+1,0}]$, compute $\sigma_{\mathrm{SPS,OUT}} = \mathsf{SPS.Sign}(\mathrm{SK}'_{\mathrm{SPS},E}, m_{\mathrm{OUT}})$.
   Else if $[(h,i) = (h^*, \iota+1)] \wedge [m_{\mathrm{IN}} \neq m_{\iota+1,0}]$, compute $\sigma_{\mathrm{SPS,OUT}} = \mathsf{SPS.Sign}(\mathrm{SK}'_{\mathrm{SPS},F}, m_{\mathrm{OUT}})$.
   Else if $i < \ell^*$, compute $\sigma_{\mathrm{SPS,OUT}} = \mathsf{SPS.Sign}(\mathrm{SK}'_{\mathrm{SPS},\alpha}, m_{\mathrm{OUT}})$.
   Else, compute $\sigma_{\mathrm{SPS,OUT}} = \mathsf{SPS.Sign}(\mathrm{SK}'_{\mathrm{SPS},E}, m_{\mathrm{OUT}})$.
5. Output $(w_{\mathrm{OUT}}, v_{\mathrm{OUT}}, \sigma_{\mathrm{SPS,OUT}})$.

Fig. 37 $\mathsf{Accumulate.Prog}^{(3,\iota',4)}$

Constants: PPRF keys $K_{\text{SPS},A}$, $K_{\text{SPS},B}$, Punctured PPRF keys $K_{\text{SPS},E}\{(h^*, \iota+1)\}$, $K_{\text{SPS},F}\{(h^*, \iota+1)\}$, Verification key VK_G, SSB hash value of challenge input h^*, Length of challenge input ℓ^*

Inputs: TM state ST, Accumulator value w, Iterator value v, SSB hash value h, Length ℓ_{INP}, Signature $\sigma_{\text{SPS,IN}}$

Output: Signature $\sigma_{\text{SPS,OUT}}$, or $\perp$

1. (a) If $(h, \ell_{\text{INP}}) \neq (h^*, \iota+1)$, compute $r_{\text{SPS},E} = \mathcal{F}(K_{\text{SPS},E}\{(h^*, \iota+1)\}, (h, \ell_{\text{INP}}))$, $(\text{SK}_{\text{SPS},E}, \text{VK}_{\text{SPS},E}, \text{VK}_{\text{SPS-REJ},E}) = \mathsf{SPS.Setup}(1^\lambda; r_{\text{SPS},E})$.
   Else, set $\text{VK}_{\text{SPS},E} = \text{VK}_G$.
   (b) If $(h, \ell_{\text{INP}}) \neq (h^*, \iota+1)$, compute $r_{\text{SPS},F} = \mathcal{F}(K_{\text{SPS},F}\{(h^*, \iota+1)\}, (h, \ell_{\text{INP}}))$, $(\text{SK}_{\text{SPS},F}, \text{VK}_{\text{SPS},F}, \text{VK}_{\text{SPS-REJ},F}) = \mathsf{SPS.Setup}(1^\lambda; r_{\text{SPS},F})$.
   (c) Set $m = (v, \text{ST}, w, 0)$ and α='-'.
   (d) If $\mathsf{SPS.Verify}(\text{VK}_{\text{SPS},E}, m, \sigma_{\text{SPS,IN}}) = 1$, set α ='E'.
   (e) If $[\alpha$ ='-'$] \wedge [(\ell_{\text{INP}} > \ell^*) \vee (0 < \ell_{\text{INP}} \leq \iota+1) \vee (h \neq h^*)]$, output $\perp$.
   Else if $[\alpha$ ='-'$] \wedge [\mathsf{SPS.Verify}(\text{VK}_{\text{SPS},F}, m, \sigma_{\text{SPS,IN}}) = 1]$, set α ='F'.
   (f) If α ='-', output $\perp$.
2. (a) Compute $r_{\text{SPS},A} = \mathcal{F}(K_{\text{SPS},A}, (h, \ell_{\text{INP}}, 0))$, $(\text{SK}_{\text{SPS},A}, \text{VK}_{\text{SPS},A}, \text{VK}_{\text{SPS-REJ},A}) = \mathsf{SPS.Setup}(1^\lambda; r_{\text{SPS},A})$.
   (b) Compute $r_{\text{SPS},B} = \mathcal{F}(K_{\text{SPS},B}, (h, \ell_{\text{INP}}, 0))$, $(\text{SK}_{\text{SPS},B}, \text{VK}_{\text{SPS},B}, \text{VK}_{\text{SPS-REJ},B}) = \mathsf{SPS.Setup}(1^\lambda; r_{\text{DSPS},B})$.
   (c) If $[(h, \ell_{\text{INP}}) = (h^*, \ell^*)] \wedge [\alpha$ ='F'$]$, output $\sigma_{\text{SPS,OUT}} = \mathsf{SPS.Sign}(\text{SK}_{\text{SPS},B}, m)$.
   Else, output $\sigma_{\text{SPS,OUT}} = \mathsf{SPS.Sign}(\text{SK}_{\text{SPS},A}, m)$.

Fig. 38 $\mathsf{Change\text{-}SPS.Prog}^{(3,\iota,2)}$

$\mathsf{Hyb}_{0,\nu-1,3,\iota',9}$: This experiments analogous to $\mathsf{Hyb}_{0,\nu-1,3,\iota',8}$ with the only exception that while constructing the ν^{th} signing key queried by $\mathcal{A}$ corresponding to TM $M^{(\nu)} \in \mathbb{M}_\lambda$ with $M^{(\nu)}(x^*) = 0$, $\mathcal{B}$ generates $(\text{SK}_{\text{SPS},G}^{(\nu,\iota+1)}, \text{VK}_{\text{SPS},G}^{(\nu,\iota+1)}, \text{VK}_{\text{SPS-REJ},G}^{(\nu,\iota+1)}) = \mathsf{SPS.Setup}(1^\lambda;\ r_{\text{SPS},G}^{(\nu,\iota+1)} = \mathcal{F}(K_{\text{SPS},E}^{(\nu)}, (h^*, \iota+1)))$.

$\mathsf{Hyb}_{0,\nu-1,3,\iota',10}$: This experiment corresponds to $\mathsf{Hyb}_{0,\nu-1,3,\iota+1}$.

Analysis

Let $\mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,3,\iota',\vartheta)}(\lambda)$ represents the advantage of the adversary $\mathcal{A}$, i.e., the absolute difference between $1/2$ and $\mathcal{A}$'s probability of correctly guessing the random bit selected by the challenger $\mathcal{B}$, in $\mathsf{Hyb}_{0,\nu-1,3,\iota',\vartheta}$, for $\vartheta \in [0, 10]$. From the description of the hybrid experiments it follows that $\mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,3,\iota')}(\lambda) \equiv \mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,3,\iota',0)}(\lambda)$ and $\mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,3,\iota+1)}(\lambda) \equiv \mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,3,\iota',10)}(\lambda)$. Hence, we have

$$|\mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,3,\iota')}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,3,\iota+1)}(\lambda)| \leq \sum_{\vartheta=1}^{10} |\mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,3,\iota',\vartheta-1)}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,3,\iota',\vartheta)}(\lambda)|. \tag{C.6}$$

Claims C.24–C.33 below will show that the RHS of Eq. (C.6) is negligible and thus Lemma C.6 follows.

Claim C.24 *Assuming $\mathcal{IO}$ is a secure indistinguishability obfuscator for* P/poly *and $\mathcal{F}$ satisfies the correctness under puncturing property, for any* PPT *adversary $\mathcal{A}$, for any security parameter λ, $|\mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,3,\iota',0)}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,3,\iota',1)}(\lambda)| \leq \mathsf{negl}(\lambda)$ for some negligible function* negl.

Proof The only difference between $\mathsf{Hyb}_{0,\nu-1,3,\iota',0}$ and $\mathsf{Hyb}_{0,\nu-1,3,\iota',1}$ is the following: In $\mathsf{Hyb}_{0,\nu-1,3,\iota',0}$, $\mathcal{B}$ includes the programs $\mathcal{IO}(P_0)$ and $\mathcal{IO}(P_0')$ within the ν^{th} signing key provided to $\mathcal{A}$, while in $\mathsf{Hyb}_{0,\nu-1,3,\iota',1}$, it includes the programs $\mathcal{IO}(P_1)$ and $\mathcal{IO}(P_1')$ instead, where

- $P_0 = \mathsf{Accumulate.Prog}^{(3,\iota')}[n_{\text{SSB-BLK}} = 2^\lambda, \text{HK}, \text{PP}_{\text{ACC}}^{(\nu)}, \text{PP}_{\text{ITR}}^{(\nu)}, K_{\text{SPS},E}^{(\nu)}, K_{\text{SPS},F}^{(\nu)}, m_{\iota+1,0}^{(\nu)}, h^*, \ell^*]$ (Fig. **17**),
- $P_0' = \mathsf{Change\text{-}SPSProg}^{(3,\iota)}[K_{\text{SPS},A}^{(\nu)}, K_{\text{SPS},B}^{(\nu)}, K_{\text{SPS},E}^{(\nu)}, K_{\text{SPS},F}^{(\nu)}, h^*, \ell^*]$ (Fig. **16**),
- $P_1 = \mathsf{Accumulate.Prog}^{(3,\iota',1)}[n_{\text{SSB-BLK}} = 2^\lambda, \text{HK}, \text{PP}_{\text{ACC}}^{(\nu)}, \text{PP}_{\text{ITR}}^{(\nu)}, K_{\text{SPS},E}^{(\nu)}\{(h^*, \iota+1)\}, K_{\text{SPS},F}^{(\nu)}\{(h^*, \iota+1)\}, \text{SK}_{\text{SPS},G}^{(\nu,\iota+1)}, \text{SK}_{\text{SPS},H}^{(\nu,\iota+1)}, \text{VK}_{\text{SPS},G}^{(\nu,\iota+1)}, \text{VK}_{\text{SPS},H}^{(\nu,\iota+1)}, m_{\iota+1,0}^{(\nu)}, h^*, \ell^*]$ (Fig. **33**),
- $P_1' = \mathsf{Change\text{-}SPS.Prog}^{(3,\iota,1)}[K_{\text{SPS},A}^{(\nu)}, K_{\text{SPS},B}^{(\nu)}, K_{\text{SPS},E}^{(\nu)}\{(h^*, \iota+1)\}, K_{\text{SPS},F}^{(\nu)}\{(h^*, \iota+1)\}, \text{VK}_{\text{SPS},G}^{(\nu,\iota+1)}, \text{VK}_{\text{SPS},H}^{(\nu,\iota+1)}, h^*, \ell^*]$ (Fig. **34**).

Observe that by the correctness under puncturing property of the PPRF $\mathcal{F}$, the programs P_0 and P_1 are functionally identical for all inputs corresponding to $(h,i) \neq (h^*,\iota)$ and $(h,i) \neq (h^*,\iota+1)$. For inputs corresponding to (h^*,ι),the program P_1 uses the hardwired signing keys which are exactly same as those computed by the program P_0. The same is true for the hardwired verification keys used by P_1 for inputs corresponding to $(h^*,\iota+1)$. Thus, the programs P_0 and P_1 are functionally equivalent. A similar argument shows that the same is correct for programs P_0' and P_1'. Therefore, by the security of $\mathcal{IO}$ Claim C.24 follows. Of course, we need to consider a sequence of intermediate hybrid experiments to switch the programs one at a time. □

Claim C.25 *Assuming $\mathcal{F}$ is a secure puncturable pseudorandom function, for any* PPT *adversary $\mathcal{A}$, for any security parameter λ, $|\mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,3,\iota',1)}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,3,\iota',2)}(\lambda)| \leq \mathsf{negl}(\lambda)$ for some negligible function* negl.

Proof Suppose there exists a PPT adversary $\mathcal{A}$ for which $|\mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,3,\iota',1)}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,3,\iota',2)}(\lambda)|$ is non-negligible. We construct a PPT adversary $\mathcal{B}$ that breaks the selective pseudorandomness of the PPRF $\mathcal{F}$ using $\mathcal{A}$ as a sub-routine. The description of $\mathcal{B}$ is given below. We note that in the following we work in a model of selective pseudorandomness for PPRF involving two independent punctured keys and two challenge values for a challenge input, one under each key. However, this model is clearly equivalent to the original single punctured key and single challenge value model described in Section 2.3.1 through a hybrid argument.

- $\mathcal{B}$ initializes $\mathcal{A}$ on input 1^λ and receives a challenge signing attribute string $x^* = x_0^* \ldots x_{\ell^*-1}^* \in \mathcal{U}_{\text{ABS}}$ with $|x^*| = \ell^*$ from $\mathcal{A}$.
- Upon receiving x^*, $\mathcal{B}$ proceeds as follows:
  1. $\mathcal{B}$ first generates $\text{HK} \xleftarrow{\$} \mathsf{SSB.Gen}(1^\lambda, n_{\text{SSB-BLK}} = 2^\lambda, i^* = 0)$ and computes $h^* = \mathcal{H}_{\text{HK}}(x^*)$.
  2. Then, $\mathcal{B}$ selects a PPRF key $K \xleftarrow{\$} \mathcal{F}.\mathsf{Setup}(1^\lambda)$.
  3. After that, $\mathcal{B}$ computes $r_{\text{SIG}}^* = \mathcal{F}(K, (h^*, \ell^*))$ followed by $(\widehat{\text{SK}}_{\text{SIG}}^*, \widehat{\text{VK}}_{\text{SIG}}^*) = \mathsf{SIG.Setup}(1^\lambda;\ r_{\text{SIG}}^*)$.
  4. $\mathcal{B}$ returns the public parameters $\text{PP}_{\text{ABS}} = (\text{HK}, \mathcal{IO}(\mathsf{Verify.Prog}_{\text{ABS}}[K]))$ to $\mathcal{A}$.

- For $\eta \in [\hat{q}_{\text{KEY}}]$, in response to the η^{th} signing key query of $\mathcal{A}$ corresponding to TM $M^{(\eta)} \in \mathbb{M}_\lambda$ with $M^{(\eta)}(x^*) = 0$, if $\eta \neq \nu$, then $\mathcal{B}$ proceeds exactly as in $\mathsf{Hyb}_{0,\nu-1,3,\iota',1}$, while if $\eta = \nu$, then $\mathcal{B}$ proceeds as follows:
  1. $\mathcal{B}$ first selects PPRF keys $K_1^{(\nu)}, \ldots, K_\lambda^{(\nu)}, K_{\text{SPS},A}^{(\nu)}, K_{\text{SPS},B}^{(\nu)} \xleftarrow{\$} \mathcal{F}.\mathsf{Setup}(1^\lambda)$.
  2. Next, it generates $(\text{PP}_{\text{ACC}}^{(\nu)}, w_0^{(\nu)}, \text{STORE}_0^{(\nu)}) \xleftarrow{\$} \mathsf{ACC.Setup}(1^\lambda, n_{\text{ACC-BLK}} = 2^\lambda)$ and $(\text{PP}_{\text{ITR}}^{(\nu)}, v_0^{(\nu)}) \xleftarrow{\$} \mathsf{ITR.Setup}(1^\lambda, n_{\text{ITR}} = 2^\lambda)$.
  3. $\mathcal{B}$ sends $(h^*, \iota+1)$ as the challenge input to its PPRF selective pseudorandomness challenger $\mathcal{C}$ and receives back two punctured PPRF keys $K_1^*\{(h^*, \iota+1)\}, K_2^*\{(h^*, \iota+1)\}$ and two values $r_1^*, r_2^* \in \mathcal{Y}_{\text{PPRF}}$, where either $r_1^* = \mathcal{F}(K_1^*, (h^*, \iota+1)), r_2^* = \mathcal{F}(K_2^*, (h^*, \iota+1))$ or $r_1^*, r_2^* \xleftarrow{\$} \mathcal{Y}_{\text{PPRF}}$. $\mathcal{B}$ *implicitly* views the keys K_1^* and K_2^* as the keys $K_{\text{SPS},E}^{(\nu)}$ and $K_{\text{SPS},F}^{(\nu)}$ respectively.
  4. $\mathcal{B}$ generates $(\text{SK}_{\text{SPS},G}^{(\nu,\iota+1)}, \text{VK}_{\text{SPS},G}^{(\nu,\iota+1)}, \text{VK}_{\text{SPS-REJ},G}^{(\nu,\iota+1)}) = \mathsf{SPS.Setup}(1^\lambda;\ r_1^*)$ and $(\text{SK}_{\text{SPS},H}^{(\nu,\iota+1)}, \text{VK}_{\text{SPS},H}^{(\nu,\iota+1)}, \text{VK}_{\text{SPS-REJ},H}^{(\nu,\iota+1)}) = \mathsf{SPS.Setup}(1^\lambda;\ r_2^*)$.
  5. Next, it sets $m_{0,0}^{(\nu)} = (v_0^{(\nu)}, q_0^{(\nu)}, w_0^{(\nu)}, 0)$. For $j = 1, \ldots, \iota+1$, it iteratively computes the following:
     - $\text{AUX}_j^{(\nu)} = \mathsf{ACC.Prep\text{-}Write}(\text{PP}_{\text{ACC}}^{(\nu)}, \text{STORE}_{j-1}^{(\nu)}, j-1)$
     - $w_j^{(\nu)} = \mathsf{ACC.Update}(\text{PP}_{\text{ACC}}^{(\nu)}, w_{j-1}^{(\nu)}, x_{j-1}^*, j-1, \text{AUX}_j^{(\nu)})$
     - $\text{STORE}_j^{(\nu)} = \mathsf{ACC.Write\text{-}Store}(\text{PP}_{\text{ACC}}^{(\nu)}, \text{STORE}_{j-1}^{(\nu)}, j-1, x_{j-1}^*)$
     - $v_j^{(\nu)} = \mathsf{ITR.Iterate}(\text{PP}_{\text{ITR}}^{(\nu)}, v_{j-1}^{(\nu)}, (q_0^{(\nu)}, w_{j-1}^{(\nu)}, 0))$

     It sets $m_{\iota+1,0}^{(\nu)} = (v_{\iota+1}^{(\nu)}, q_0^{(\nu)}, w_{\iota+1}^{(\nu)}, 0)$.
  6. It gives $\mathcal{A}$ the signing key

$$\text{SK}_{\text{ABS}}\{M^{(\nu)}\} = \left(\begin{array}{l} \text{HK}, \text{PP}_{\text{ACC}}^{(\nu)}, w_0^{(\nu)}, \text{STORE}_0^{(\nu)}, \text{PP}_{\text{ITR}}^{(\nu)}, v_0^{(\nu)}, \\ \mathcal{IO}(\mathsf{Init\text{-}SPS.Prog}[q_0^{(\nu)}, w_0^{(\nu)}, v_0^{(\nu)}, K_1^*\{(h^*, \iota+1)\}]), \\ \mathcal{IO}(\mathsf{Accumulate.Prog}^{(3,\iota',1)}[n_{\text{SSB-BLK}} = 2^\lambda, \text{HK}, \text{PP}_{\text{ACC}}^{(\nu)}, \text{PP}_{\text{ITR}}^{(\nu)}, \\ \qquad K_1^*\{(h^*, \iota+1)\}, K_2^*\{(h^*, \iota+1)\}, \text{SK}_{\text{SPS},G}^{(\nu,\iota+1)}, \text{SK}_{\text{SPS},H}^{(\nu,\iota+1)}, \\ \qquad \text{VK}_{\text{SPS},G}^{(\nu,\iota+1)}, \text{VK}_{\text{SPS},H}^{(\nu,\iota+1)}, m_{\iota+1,0}^{(\nu)}, h^*, \ell^*]), \\ \mathcal{IO}(\mathsf{Change\text{-}SPS.Prog}^{(3,\iota,1)}[K_{\text{SPS},A}^{(\nu)}, K_{\text{SPS},B}^{(\nu)}, K_1^*\{(h^*, \iota+1)\}, \\ \qquad K_2^*\{(h^*, \iota+1)\}, \text{VK}_{\text{SPS},G}^{(\nu,\iota+1)}, \text{VK}_{\text{SPS},H}^{(\nu,\iota+1)}, h^*, \ell^*]), \\ \mathcal{IO}(\mathsf{Constrained\text{-}Key.Prog}_{\text{ABS}}^{(1)}[M^{(\nu)}, T = 2^\lambda, t^{*(\nu)}, \text{PP}_{\text{ACC}}^{(\nu)}, \text{PP}_{\text{ITR}}^{(\nu)}, K, \\ \qquad K_1^{(\nu)}, \ldots, K_\lambda^{(\nu)}, K_{\text{SPS},A}^{(\nu)}, K_{\text{SPS},B}^{(\nu)}, h^*, \ell^*]) \end{array}\right).$$

- For $\theta \in [\hat{q}_{\text{SIGN}}]$, in reply to the θ^{th} signature query of $\mathcal{A}$ on message $\mathsf{msg}^{(\theta)} \in \mathcal{M}_{\text{ABS}}$ under attribute string x^*, $\mathcal{B}$ computes $\sigma_{\text{SIG}}^{(\theta)} \xleftarrow{\$} \mathsf{SIG.Sign}(\widehat{\text{SK}}_{\text{SIG}}^*, \mathsf{msg}^{(\theta)})$ and provides $\mathcal{A}$ with $\sigma_{\text{ABS}}^{(\theta)} = (\widehat{\text{VK}}_{\text{SIG}}^*, \sigma_{\text{SIG}}^{(\theta)})$.
- Finally, $\mathcal{A}$ output a signature σ_{ABS}^* on some message msg^* under attribute string x^*. $\mathcal{B}$ outputs $\hat{b}' = 1$ as its guess bit in its PPRF selective pseudorandomness experiment if $\mathcal{A}$ wins, i.e., if $\mathsf{ABS.Verify}(\text{PP}_{\text{ABS}}, x^*, \mathsf{msg}^*, \sigma_{\text{ABS}}^*) = 1$ and $\mathsf{msg}^* \neq \mathsf{msg}^{(\theta)}$ for any $\theta \in [\hat{q}_{\text{SIGN}}]$. Otherwise, $\mathcal{B}$ outputs $\hat{b}' = 0$ in its PPRF selective pseudorandomness experiment.

Note that if $r_1^* = \mathcal{F}(K_1^*, (h^*, \iota+1)), r_2^* = \mathcal{F}(K_2^*, (h^*, \iota+1))$, then $\mathcal{B}$ perfectly simulates $\mathsf{Hyb}_{0,\nu-1,3,\iota',1}$. On the other hand, if $r_1^*, r_2^* \xleftarrow{\$} \mathcal{Y}_{\text{PPRF}}$, the $\mathcal{B}$ perfectly simulates $\mathsf{Hyb}_{0,\nu-1,3,\iota',2}$. This completes the proof of Claim C.25. □

Claim C.26 *Assuming $\mathcal{IO}$ is a secure indistinguishability obfuscator for* $\mathsf{P/poly}$*, for any* PPT *adversary* $\mathcal{A}$*, for any security parameter* λ*,* $|\mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,3,\iota',2)}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,3,\iota',3)}(\lambda)| \leq \mathsf{negl}(\lambda)$ *for some negligible function* negl.

Proof The only difference between $\mathsf{Hyb}_{0,\nu-1,3,\iota',2}$ and $\mathsf{Hyb}_{0,\nu-1,3,\iota',3}$ is the following: In $\mathsf{Hyb}_{0,\nu-1,3,\iota',2}$, $\mathcal{B}$ includes the program $\mathcal{IO}(P_0)$ within the ν^{th} signing key provided to $\mathcal{A}$, while in $\mathsf{Hyb}_{0,\nu-1,3,\iota',3}$, it includes the program $\mathcal{IO}(P_1)$ instead, where

- $P_0 = \mathsf{Accumulate.Prog}^{(3,\iota',1)}[n_{\text{SSB-BLK}} = 2^\lambda, \text{HK}, \text{PP}_{\text{ACC}}^{(\nu)}, \text{PP}_{\text{ITR}}^{(\nu)}, K_{\text{SPS},E}^{(\nu)}\{(h^*, \iota+1)\}, K_{\text{SPS},F}^{(\nu)}\{(h^*, \iota+1)\}, \text{SK}_{\text{SPS},G}^{(\nu,\iota+1)}, \text{SK}_{\text{SPS},H}^{(\nu,\iota+1)}, \text{VK}_{\text{SPS},G}^{(\nu,\iota+1)}, \text{VK}_{\text{SPS},H}^{(\nu,\iota+1)}, m_{\iota+1,0}^{(\nu)}, h^*, \ell^*]$ (Fig. **33**),
- $P_1 = \mathsf{Accumulate.Prog}^{(3,\iota',2)}[n_{\text{SSB-BLK}} = 2^\lambda, \text{HK}, \text{PP}_{\text{ACC}}^{(\nu)}, \text{PP}_{\text{ITR}}^{(\nu)}, K_{\text{SPS},E}^{(\nu)}\{(h^*, \iota+1)\}, K_{\text{SPS},F}^{(\nu)}\{(h^*, \iota+1)\}, \sigma_{\text{SPS-ONE}, m_{\iota+1,0}^{(\nu)}, G}^{(\nu,\iota+1)}, \text{SK}_{\text{SPS-ABO},H}^{(\nu,\iota+1)}, \text{VK}_{\text{SPS},G}^{(\nu,\iota+1)}, \text{VK}_{\text{SPS},H}^{(\nu,\iota+1)}, m_{\iota+1,0}^{(\nu)}, h^*, \ell^*]$ (Fig. **35**).

Now, the only inputs on which the outputs of the two programs can possibly differ are those corresponding to $(h, i) = (h^*, \iota)$. However, observe that for inputs corresponding to (h^*, ι), if $m_{\text{OUT}} = m_{\iota+1,0}^{(\nu)}$, then both programs clearly output the same signature, where P_0 computes the signature explicitly and P_1 has the signature hardwired into it. On the other hand, by the correctness [Property (ii)] of the splittable signature SPS defined in Section 2.3.5 it follows that the programs P_0 and P_1 output same signatures even when $m_{\text{OUT}} \neq m_{\iota+1,0}^{(\nu)}$ for inputs corresponding to (h^*, ι). Hence, the two programs are functionally equivalent. Therefore, Claim C.26 follows by the security of $\mathcal{IO}$. □

Claim C.27 *Assuming* SPS *is a splittable signature scheme satisfying '*$\text{VK}_{\text{SPS-ONE}}$ *indistinguishability', for any* PPT *adversary* $\mathcal{A}$*, for any security parameter* λ*,* $|\mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,3,\iota',3)}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,3,\iota',4)}(\lambda)| \leq \mathsf{negl}(\lambda)$ *for some negligible function* negl.

Proof Suppose there exists a PPT adversary $\mathcal{A}$ for which $|\mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,3,\iota',3)}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,3,\iota',4)}(\lambda)|$ is non-negligible. Below we construct a PPT adversary $\mathcal{B}$ that breaks the $\text{VK}_{\text{SPS-ONE}}$ indistinguishability of SPS using $\mathcal{A}$ as a sub-routine.

- $\mathcal{B}$ initializes $\mathcal{A}$ on input 1^λ and receives a challenge signing attribute string $x^* = x_0^* \ldots x_{\ell^*-1}^* \in \mathcal{U}_{\text{ABS}}$ with $|x^*| = \ell^*$ from $\mathcal{A}$.
- Upon receiving x^*, $\mathcal{B}$ proceeds as follows:
  1. $\mathcal{B}$ first generates $\text{HK} \xleftarrow{\$} \mathsf{SSB.Gen}(1^\lambda, n_{\text{SSB-BLK}} = 2^\lambda, i^* = 0)$ and computes $h^* = \mathcal{H}_{\text{HK}}(x^*)$.
  2. Then, $\mathcal{B}$ selects a PPRF key $K \xleftarrow{\$} \mathcal{F}.\mathsf{Setup}(1^\lambda)$.
  3. After that, $\mathcal{B}$ computes $r_{\text{SIG}}^* = \mathcal{F}(K, (h^*, \ell^*))$ followed by $(\widehat{\text{SK}}_{\text{SIG}}^*, \widehat{\text{VK}}_{\text{SIG}}^*) = \mathsf{SIG.Setup}(1^\lambda; r_{\text{SIG}}^*)$.
  4. $\mathcal{B}$ returns the public parameters $\text{PP}_{\text{ABS}} = (\text{HK}, \mathcal{IO}(\mathsf{Verify.Prog}_{\text{ABS}}[K]))$ to $\mathcal{A}$.
- For $\eta \in [\hat{q}_{\text{KEY}}]$, in response to the η^{th} signing key query of $\mathcal{A}$ corresponding to TM $M^{(\eta)} \in \mathbb{M}_\lambda$ with $M^{(\eta)}(x^*) = 0$, if $\eta \neq \nu$, then $\mathcal{B}$ proceeds exactly as in $\mathsf{Hyb}_{0,\nu-1,3,\iota',3}$, while if $\eta = \nu$, then $\mathcal{B}$ proceeds as follows:
  1. $\mathcal{B}$ first selects PPRF keys $K_1^{(\nu)}, \ldots, K_\lambda^{(\nu)}, K_{\text{SPS},A}^{(\nu)}, K_{\text{SPS},B}^{(\nu)}, K_{\text{SPS},E}^{(\nu)}, K_{\text{SPS},F}^{(\nu)} \xleftarrow{\$} \mathcal{F}.\mathsf{Setup}(1^\lambda)$.

2. Next, it generates $(\mathrm{PP}_{\mathrm{ACC}}^{(\nu)}, w_0^{(\nu)}, \mathrm{STORE}_0^{(\nu)}) \xleftarrow{\$} \mathsf{ACC.Setup}(1^\lambda, n_{\text{ACC-BLK}} = 2^\lambda)$ and $(\mathrm{PP}_{\mathrm{ITR}}^{(\nu)}, v_0^{(\nu)}) \xleftarrow{\$} \mathsf{ITR.Setup}(1^\lambda, n_{\mathrm{ITR}} = 2^\lambda)$.
3. Then, $\mathcal{B}$ creates the punctured PPRF keys $K_{\mathrm{SPS},E}^{(\nu)}\{(h^*, \iota+1)\} \xleftarrow{\$} \mathcal{F}.\mathsf{Puncture}(K_{\mathrm{SPS},E}^{(\nu)}, (h^*, \iota+1))$ and $K_{\mathrm{SPS},F}^{(\nu)}\{(h^*, \iota+1)\} \xleftarrow{\$} \mathcal{F}.\mathsf{Puncture}(K_{\mathrm{SPS},F}^{(\nu)}, (h^*, \iota+1))$.
4. Next, it sets $m_{0,0}^{(\nu)} = (v_0^{(\nu)}, q_0^{(\nu)}, w_0^{(\nu)}, 0)$. For $j = 1, \ldots, \iota+1$, it iteratively computes the following:
   - $\mathrm{AUX}_j^{(\nu)} = \mathsf{ACC.Prep\text{-}Write}(\mathrm{PP}_{\mathrm{ACC}}^{(\nu)}, \mathrm{STORE}_{j-1}^{(\nu)}, j-1)$
   - $w_j^{(\nu)} = \mathsf{ACC.Update}(\mathrm{PP}_{\mathrm{ACC}}^{(\nu)}, w_{j-1}^{(\nu)}, x_{j-1}^*, j-1, \mathrm{AUX}_j^{(\nu)})$
   - $\mathrm{STORE}_j^{(\nu)} = \mathsf{ACC.Write\text{-}Store}(\mathrm{PP}_{\mathrm{ACC}}^{(\nu)}, \mathrm{STORE}_{j-1}^{(\nu)}, j-1, x_{j-1}^*)$
   - $v_j^{(\nu)} = \mathsf{ITR.Iterate}(\mathrm{PP}_{\mathrm{ITR}}^{(\nu)}, v_{j-1}^{(\nu)}, (q_0^{(\nu)}, w_{j-1}^{(\nu)}, 0))$

   It sets $m_{\iota+1,0}^{(\nu)} = (v_{\iota+1}^{(\nu)}, q_0^{(\nu)}, w_{\iota+1}^{(\nu)}, 0)$.
5. After that, $\mathcal{B}$ sends $m_{\iota+1,0}^{(\nu)}$ as the challenge message to its SPS $\mathrm{VK}_{\text{SPS-ONE}}$ indistinguishability challenger $\mathcal{C}$ and receives back a signature-verification key pair $(\sigma_{\text{SPS-ONE},m_{\iota+1,0}^{(\nu)}}, \mathrm{VK})$, where VK is either a normal verification key $\mathrm{VK}_{\mathrm{SPS}}$ or a one verification key $\mathrm{VK}_{\text{SPS-ONE}}$ for the message $m_{\iota+1,0}^{(\nu)}$.
6. $\mathcal{B}$ generates $(\mathrm{SK}_{\mathrm{SPS},H}^{(\nu,\iota+1)}, \mathrm{VK}_{\mathrm{SPS},H}^{(\nu,\iota+1)}, \mathrm{VK}_{\text{SPS-REJ},H}^{(\nu,\iota+1)}) \xleftarrow{\$} \mathsf{SPS.Setup}(1^\lambda)$ and forms $(\sigma_{\text{SPS-ONE},m_{\iota+1,0}^{(\nu)},H}^{(\nu,\iota+1)}, \mathrm{VK}_{\text{SPS-ONE},H}^{(\nu,\iota+1)}, \mathrm{SK}_{\text{SPS-ABO},H}^{(\nu,\iota+1)}, \mathrm{VK}_{\text{SPS-ABO},H}^{(\nu,\iota+1)}) \xleftarrow{\$} \mathsf{SPS.Split}(\mathrm{SK}_{\mathrm{SPS},H}^{(\nu,\iota+1)}, m_{\iota+1,0}^{(\nu)})$.
7. $\mathcal{B}$ gives $\mathcal{A}$ the signing key

$$\mathrm{SK}_{\mathrm{ABS}}\{M^{(\nu)}\} = \left(\begin{array}{l} \mathrm{HK}, \mathrm{PP}_{\mathrm{ACC}}^{(\nu)}, w_0^{(\nu)}, \mathrm{STORE}_0^{(\nu)}, \mathrm{PP}_{\mathrm{ITR}}^{(\nu)}, v_0^{(\nu)}, \\ \mathcal{IO}(\mathsf{Init\text{-}SPS.Prog}[q_0^{(\nu)}, w_0^{(\nu)}, v_0^{(\nu)}, K_{\mathrm{SPS},E}^{(\nu)}\{(h^*, \iota+1)\}]), \\ \mathcal{IO}(\mathsf{Accumulate.Prog}^{(3,\iota',2)}[n_{\text{SSB-BLK}} = 2^\lambda, \mathrm{HK}, \mathrm{PP}_{\mathrm{ACC}}^{(\nu)}, \mathrm{PP}_{\mathrm{ITR}}^{(\nu)}, \\ \qquad K_{\mathrm{SPS},E}^{(\nu)}\{(h^*, \iota+1)\}, K_{\mathrm{SPS},F}^{(\nu)}\{(h^*, \iota+1)\}, \sigma_{\text{SPS-ONE},m_{\iota+1,0}^{(\nu)}}, \\ \qquad \mathrm{SK}_{\text{SPS-ABO},H}^{(\nu,\iota+1)}, \mathrm{VK}, \mathrm{VK}_{\mathrm{SPS},H}^{(\nu,\iota+1)}, m_{\iota+1,0}^{(\nu)}, h^*, \ell^*]), \\ \mathcal{IO}(\mathsf{Change\text{-}SPS.Prog}^{(3,\iota,1)}[K_{\mathrm{SPS},A}^{(\nu)}, K_{\mathrm{SPS},B}^{(\nu)}, K_{\mathrm{SPS},E}^{(\nu)}\{(h^*, \iota+1)\}, \\ \qquad K_{\mathrm{SPS},F}^{(\nu)}\{(h^*, \iota+1)\}, \mathrm{VK}, \mathrm{VK}_{\mathrm{SPS},H}^{(\nu,\iota+1)}, h^*, \ell^*]), \\ \mathcal{IO}(\mathsf{Constrained\text{-}Key.Prog}_{\mathrm{ABS}}^{(1)}[M^{(\nu)}, T = 2^\lambda, t^{*(\nu)}, \mathrm{PP}_{\mathrm{ACC}}^{(\nu)}, \mathrm{PP}_{\mathrm{ITR}}^{(\nu)}, K, \\ \qquad K_1^{(\nu)}, \ldots, K_\lambda^{(\nu)}, K_{\mathrm{SPS},A}^{(\nu)}, K_{\mathrm{SPS},B}^{(\nu)}, h^*, \ell^*]) \end{array}\right).$$

- For $\theta \in [\hat{q}_{\mathrm{SIGN}}]$, in reply to the θ^{th} signature query of $\mathcal{A}$ on message $\mathsf{msg}^{(\theta)} \in \mathcal{M}_{\mathrm{ABS}}$ under attribute string x^*, $\mathcal{B}$ computes $\sigma_{\mathrm{SIG}}^{(\theta)} \xleftarrow{\$} \mathsf{SIG.Sign}(\widehat{\mathrm{SK}}_{\mathrm{SIG}}^*, \mathsf{msg}^{(\theta)})$ and provides $\mathcal{A}$ with $\sigma_{\mathrm{ABS}}^{(\theta)} = (\widehat{\mathrm{VK}}_{\mathrm{SIG}}^*, \sigma_{\mathrm{SIG}}^{(\theta)})$.
- Finally, $\mathcal{A}$ output a signature σ_{ABS}^* on some message msg^* under attribute string x^*. $\mathcal{B}$ outputs $\hat{b}' = 1$ as its guess bit in its $\mathrm{VK}_{\text{SPS-ONE}}$ indistinguishability experiment if $\mathcal{A}$ wins, i.e., if $\mathsf{ABS.Verify}(\mathrm{PP}_{\mathrm{ABS}}, x^*, \mathsf{msg}^*, \sigma_{\mathrm{ABS}}^*) = 1$ and $\mathsf{msg}^* \neq \mathsf{msg}^{(\theta)}$ for any $\theta \in [\hat{q}_{\mathrm{SIGN}}]$. Otherwise, $\mathcal{B}$ outputs $\hat{b}' = 0$ in its $\mathrm{VK}_{\text{SPS-ONE}}$ indistinguishability experiment.

Notice that if $\mathrm{VK} = \mathrm{VK}_{\mathrm{SPS}}$, then $\mathcal{B}$ perfectly simulates $\mathsf{Hyb}_{0,\nu-1,3,\iota',3}$. On the other hand, if $\mathrm{VK} = \mathrm{VK}_{\text{SPS-ONE}}$, then $\mathcal{B}$ perfectly simulates $\mathsf{Hyb}_{0,\nu-1,3,\iota',4}$. This completes the proof of Claim C.27. □

Claim C.28 *Assuming* SPS *is a splittable signature scheme satisfying '*$\mathrm{VK}_{\text{SPS-ABO}}$ *indistinguishability', for any* PPT *adversary* $\mathcal{A}$*, for any security parameter* λ*,* $|\mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,3,\iota',4)}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,3,\iota',5)}(\lambda)| \leq \mathsf{negl}(\lambda)$ *for some negligible function* negl.

Proof Suppose there exists a PPT adversary $\mathcal{A}$ for which $|\mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,3,\iota',4)}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,3,\iota',5)}(\lambda)|$ is non-negligible. Below we construct a PPT adversary $\mathcal{B}$ that breaks the $\mathrm{VK}_{\text{SPS-ABO}}$ indistinguishability of SPS using $\mathcal{A}$ as a sub-routine.

- $\mathcal{B}$ initializes $\mathcal{A}$ on input 1^λ and receives a challenge signing attribute string $x^* = x_0^* \ldots x_{\ell^*-1}^* \in \mathcal{X}_{\mathrm{CPRF}}$ with $|x^*| = \ell^*$ from $\mathcal{A}$.
- Upon receiving x^*, $\mathcal{B}$ proceeds as follows:
  1. $\mathcal{B}$ first generates $\mathrm{HK} \xleftarrow{\$} \mathsf{SSB.Gen}(1^\lambda, n_{\text{SSB-BLK}} = 2^\lambda, i^* = 0)$ and computes $h^* = \mathcal{H}_{\mathrm{HK}}(x^*)$.
  2. Then, $\mathcal{B}$ selects a PPRF key $K \xleftarrow{\$} \mathcal{F}.\mathsf{Setup}(1^\lambda)$.
  3. After that, $\mathcal{B}$ computes $r_{\mathrm{SIG}}^* = \mathcal{F}(K, (h^*, \ell^*))$ followed by $(\widehat{\mathrm{SK}}_{\mathrm{SIG}}^*, \widehat{\mathrm{VK}}_{\mathrm{SIG}}^*) = \mathsf{SIG.Setup}(1^\lambda;\ r_{\mathrm{SIG}}^*)$.
  4. $\mathcal{B}$ returns the public parameters $\mathrm{PP}_{\mathrm{ABS}} = (\mathrm{HK}, \mathcal{IO}(\mathsf{Verify.Prog}_{\mathrm{ABS}}[K]))$ to $\mathcal{A}$.
- For $\eta \in [\hat{q}_{\mathrm{KEY}}]$, in response to the η^{th} signing key query of $\mathcal{A}$ corresponding to TM $M^{(\eta)} \in \mathbb{M}_\lambda$ with $M^{(\eta)}(x^*) = 0$, if $\eta \neq \nu$, then $\mathcal{B}$ proceeds exactly as in $\mathsf{Hyb}_{0,\nu-1,3,\iota',4}$, while if $\eta = \nu$, then $\mathcal{B}$ proceeds as follows:
  1. $\mathcal{B}$ first selects PPRF keys $K_1^{(\nu)}, \ldots, K_\lambda^{(\nu)}, K_{\mathrm{SPS},A}^{(\nu)}, K_{\mathrm{SPS},B}^{(\nu)}, K_{\mathrm{SPS},E}^{(\nu)}, K_{\mathrm{SPS},F}^{(\nu)} \xleftarrow{\$} \mathcal{F}.\mathsf{Setup}(1^\lambda)$.
  2. Next, it generates $(\mathrm{PP}_{\mathrm{ACC}}^{(\nu)}, w_0^{(\nu)}, \mathrm{STORE}_0^{(\nu)}) \xleftarrow{\$} \mathsf{ACC.Setup}(1^\lambda, n_{\text{ACC-BLK}} = 2^\lambda)$ and $(\mathrm{PP}_{\mathrm{ITR}}^{(\nu)}, v_0^{(\nu)}) \xleftarrow{\$} \mathsf{ITR.Setup}(1^\lambda, n_{\mathrm{ITR}} = 2^\lambda)$.
  3. Then, $\mathcal{B}$ creates the punctured PPRF keys $K_{\mathrm{SPS},E}^{(\nu)}\{(h^*, \iota+1)\} \xleftarrow{\$} \mathcal{F}.\mathsf{Puncture}(K_{\mathrm{SPS},E}^{(\nu)}, (h^*, \iota+1))$ and $K_{\mathrm{SPS},F}^{(\nu)}\{(h^*, \iota+1)\} \xleftarrow{\$} \mathcal{F}.\mathsf{Puncture}(K_{\mathrm{SPS},F}^{(\nu)}, (h^*, \iota+1))$.
  4. Next, it sets $m_{0,0}^{(\nu)} = (v_0^{(\nu)}, q_0^{(\nu)}, w_0^{(\nu)}, 0)$. For $j = 1, \ldots, \iota+1$, it iteratively computes the following:
     - $\mathrm{AUX}_j^{(\nu)} = \mathsf{ACC.Prep\text{-}Write}(\mathrm{PP}_{\mathrm{ACC}}^{(\nu)}, \mathrm{STORE}_{j-1}^{(\nu)}, j-1)$
     - $w_j^{(\nu)} = \mathsf{ACC.Update}(\mathrm{PP}_{\mathrm{ACC}}^{(\nu)}, w_{j-1}^{(\nu)}, x_{j-1}^*, j-1, \mathrm{AUX}_j^{(\nu)})$
     - $\mathrm{STORE}_j^{(\nu)} = \mathsf{ACC.Write\text{-}Store}(\mathrm{PP}_{\mathrm{ACC}}^{(\nu)}, \mathrm{STORE}_{j-1}^{(\nu)}, j-1, x_{j-1}^*)$
     - $v_j^{(\nu)} = \mathsf{ITR.Iterate}(\mathrm{PP}_{\mathrm{ITR}}^{(\nu)}, v_{j-1}^{(\nu)}, (q_0^{(\nu)}, w_{j-1}^{(\nu)}, 0))$

     It sets $m_{\iota+1,0}^{(\nu)} = (v_{\iota+1}^{(\nu)}, q_0^{(\nu)}, w_{\iota+1}^{(\nu)}, 0)$.
  5. After that, $\mathcal{B}$ sends $m_{\iota+1,0}^{(\nu)}$ as the challenge message to its SPS $\mathrm{VK}_{\text{SPS-ABO}}$ indistinguishability challenger $\mathcal{C}$ and receives back an all-but-one signing key-verification key pair $(\mathrm{SK}_{\text{SPS-ABO}}, \mathrm{VK})$, where VK is either a normal verification key $\mathrm{VK}_{\mathrm{SPS}}$ or an all-but-one verification key $\mathrm{VK}_{\text{SPS-ABO}}$.

6. $\mathcal{B}$ generates $(\mathrm{SK}_{\mathrm{SPS},G}^{(\nu,\iota+1)}, \mathrm{VK}_{\mathrm{SPS},G}^{(\nu,\iota+1)}, \mathrm{VK}_{\mathrm{SPS\text{-}REJ},G}^{(\nu,\iota+1)}) \xleftarrow{\$} \mathsf{SPS.Setup}(1^\lambda)$ and forms $(\sigma_{\mathrm{SPS\text{-}ONE},m_{\iota+1,0}^{(\nu)},G}^{(\nu,\iota+1)}, \mathrm{VK}_{\mathrm{SPS\text{-}ONE},G}^{(\nu,\iota+1)}, \mathrm{SK}_{\mathrm{SPS\text{-}ABO},G}^{(\nu,\iota+1)}, \mathrm{VK}_{\mathrm{SPS\text{-}ABO},G}^{(\nu,\iota+1)}) \xleftarrow{\$} \mathsf{SPS.Split}(\mathrm{SK}_{\mathrm{SPS},G}^{(\nu,\iota+1)}, m_{\iota+1,0}^{(\nu)})$.
7. $\mathcal{B}$ gives $\mathcal{A}$ the signing key

$$\mathrm{SK}_{\mathrm{ABS}}\{M^{(\nu)}\} = \left(\begin{array}{l} \mathrm{HK}, \mathrm{PP}_{\mathrm{ACC}}^{(\nu)}, w_0^{(\nu)}, \mathrm{STORE}_0^{(\nu)}, \mathrm{PP}_{\mathrm{ITR}}^{(\nu)}, v_0^{(\nu)}, \\ \mathcal{IO}(\mathsf{Init\text{-}SPS.Prog}[q_0^{(\nu)}, w_0^{(\nu)}, v_0^{(\nu)}, K_{\mathrm{SPS},E}^{(\nu)}\{(h^*,\iota+1)\}]), \\ \mathcal{IO}(\mathsf{Accumulate.Prog}^{(3,\iota',2)}[n_{\mathrm{SSB\text{-}BLK}} = 2^\lambda, \mathrm{HK}, \mathrm{PP}_{\mathrm{ACC}}^{(\nu)}, \mathrm{PP}_{\mathrm{ITR}}^{(\nu)}, \\ \quad K_{\mathrm{SPS},E}^{(\nu)}\{(h^*,\iota+1)\}, K_{\mathrm{SPS},F}^{(\nu)}\{(h^*,\iota+1)\}, \sigma_{\mathrm{SPS\text{-}ONE},m_{\iota+1,0}^{(\nu)},G}^{(\nu,\iota+1)}, \\ \quad \mathrm{SK}_{\mathrm{SPS\text{-}ABO}}, \mathrm{VK}_{\mathrm{SPS\text{-}ONE},G}^{(\nu,\iota+1)}, \mathrm{VK}, m_{\iota+1,0}^{(\nu)}, h^*, \ell^*]), \\ \mathcal{IO}(\mathsf{Change\text{-}SPS.Prog}^{(3,\iota,1)}[K_{\mathrm{SPS},A}^{(\nu)}, K_{\mathrm{SPS},B}^{(\nu)}, K_{\mathrm{SPS},E}^{(\nu)}\{(h^*,\iota+1)\}, \\ \quad K_{\mathrm{SPS},F}^{(\nu)}\{(h^*,\iota+1)\}, \mathrm{VK}_{\mathrm{SPS\text{-}ONE},G}^{(\nu,\iota+1)}, \mathrm{VK}, h^*, \ell^*]), \\ \mathcal{IO}(\mathsf{Constrained\text{-}Key.Prog}_{\mathrm{ABS}}^{(1)}[M^{(\nu)}, T = 2^\lambda, t^{*(\nu)}, \mathrm{PP}_{\mathrm{ACC}}^{(\nu)}, \mathrm{PP}_{\mathrm{ITR}}^{(\nu)}, K, \\ \quad K_1^{(\nu)}, \ldots, K_\lambda^{(\nu)}, K_{\mathrm{SPS},A}^{(\nu)}, K_{\mathrm{SPS},B}^{(\nu)}, h^*, \ell^*]) \end{array}\right).$$

- For $\theta \in [\hat{q}_{\mathrm{SIGN}}]$, in reply to the θ^{th} signature query of $\mathcal{A}$ on message $\mathsf{msg}^{(\theta)} \in \mathcal{M}_{\mathrm{ABS}}$ under attribute string x^*, $\mathcal{B}$ computes $\sigma_{\mathrm{SIG}}^{(\theta)} \xleftarrow{\$} \mathsf{SIG.Sign}(\widehat{\mathrm{SK}}_{\mathrm{SIG}}^*, \mathsf{msg}^{(\theta)})$ and provides $\mathcal{A}$ with $\sigma_{\mathrm{ABS}}^{(\theta)} = (\widehat{\mathrm{VK}}_{\mathrm{SIG}}^*, \sigma_{\mathrm{SIG}}^{(\theta)})$.
- Finally, $\mathcal{A}$ output a signature σ_{ABS}^* on some message msg^* under attribute string x^*. $\mathcal{B}$ outputs $\hat{b}' = 1$ as its guess bit in its $\mathrm{VK}_{\mathrm{SPS\text{-}ABO}}$ indistinguishability experiment if $\mathcal{A}$ wins, i.e., if $\mathsf{ABS.Verify}(\mathrm{PP}_{\mathrm{ABS}}, x^*, \mathsf{msg}^*, \sigma_{\mathrm{ABS}}^*) = 1$ and $\mathsf{msg}^* \neq \mathsf{msg}^{(\theta)}$ for any $\theta \in [\hat{q}_{\mathrm{SIGN}}]$. Otherwise, $\mathcal{B}$ outputs $\hat{b}' = 0$ in its $\mathrm{VK}_{\mathrm{SPS\text{-}ABO}}$ indistinguishability experiment.

Notice that if $\mathrm{VK} = \mathrm{VK}_{\mathrm{SPS}}$, then $\mathcal{B}$ perfectly simulates $\mathsf{Hyb}_{0,\nu-1,3,\iota',4}$. On the other hand, if $\mathrm{VK} = \mathrm{VK}_{\mathrm{SPS\text{-}ABO}}$, then $\mathcal{B}$ perfectly simulates $\mathsf{Hyb}_{0,\nu-1,3,\iota',5}$. This completes the proof of Claim C.28. □

Claim C.29 *Assuming $\mathcal{IO}$ is a secure indistinguishability obfuscator for* $\mathsf{P/poly}$*, for any* PPT *adversary $\mathcal{A}$, for any security parameter λ, $|\mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,3,\iota',5)}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,3,\iota',6)}(\lambda)| \leq \mathsf{negl}(\lambda)$ for some negligible function* negl.

Proof The only difference between $\mathsf{Hyb}_{0,\nu-1,3,\iota',5}$ and $\mathsf{Hyb}_{0,\nu-1,3,\iota',6}$ is the following: In $\mathsf{Hyb}_{0,\nu-1,3,\iota',5}$, $\mathcal{B}$ includes the program $\mathcal{IO}(P_0)$ within the ν^{th} signing key returned to $\mathcal{A}$, while in $\mathsf{Hyb}_{0,\nu-1,3,\iota',6}$, it includes the program $\mathcal{IO}(P_1)$ instead, where

- $P_0 = \mathsf{Accumulate.Prog}^{(3,\iota',2)}[n_{\mathrm{SSB\text{-}BLK}} = 2^\lambda, \mathrm{HK}, \mathrm{PP}_{\mathrm{ACC}}^{(\nu)}, \mathrm{PP}_{\mathrm{ITR}}^{(\nu)}, K_{\mathrm{SPS},E}^{(\nu)}\{(h^*,\iota+1)\}, K_{\mathrm{SPS},F}^{(\nu)}\{(h^*,\iota+1)\}, \sigma_{\mathrm{SPS\text{-}ONE},\ m_{\iota+1,0}^{(\nu)},G}^{(\nu,\iota+1)}, \mathrm{SK}_{\mathrm{SPS\text{-}ABO},H}^{(\nu,\iota+1)}, \mathrm{VK}_{\mathrm{SPS\text{-}ONE},G}^{(\nu,\iota+1)}, \mathrm{VK}_{\mathrm{SPS\text{-}ABO},H}^{(\nu,\iota+1)}, m_{\iota+1,0}^{(\nu)}, h^*, \ell^*]$ (Fig. **35**),
- $P_1 = \mathsf{Accumulate.Prog}^{(3,\iota',3)}[n_{\mathrm{SSB\text{-}BLK}} = 2^\lambda, \mathrm{HK}, \mathrm{PP}_{\mathrm{ACC}}^{(\nu)}, \mathrm{PP}_{\mathrm{ITR}}^{(\nu)}, K_{\mathrm{SPS},E}^{(\nu)}\{(h^*,\iota+1)\}, K_{\mathrm{SPS},F}^{(\nu)}\{(h^*,\iota+1)\}, \sigma_{\mathrm{SPS\text{-}ONE},\ m_{\iota+1,0}^{(\nu)},G}^{(\nu,\iota+1)}, \mathrm{SK}_{\mathrm{SPS\text{-}ABO},H}^{(\nu,\iota+1)}, \mathrm{VK}_{\mathrm{SPS\text{-}ONE},G}^{(\nu,\iota+1)}, \mathrm{VK}_{\mathrm{SPS\text{-}ABO},H}^{(\nu,\iota+1)}, m_{\iota+1,0}^{(\nu)}, h^*, \ell^*]$ (Fig. **36**).

We will argue that the programs P_0 and P_1 are functionally equivalent, so that, by the security of $\mathcal{IO}$ Claim C.29 holds. First of all observe that the constants hardwired

in both the programs are identically generated. Clearly, the inputs on which the outputs of the programs P_0 and P_1 can possibly differ are those corresponding to $(h, i) = (h^*, \iota{+}1)$. For inputs corresponding to $(h^*, \iota{+}1)$, let us consider the following two cases:

(I) $(m_{\text{IN}} = m^{(\nu)}_{\iota+1,0})$: In this case, using the correctness [Properties (i), (iii) and (vi)] of the splittable signature SPS described in Section 2.3.5 it follows that for both programs either $\alpha =$'-' or $\alpha =$'E'. Now, if $\alpha =$'-', then both programs output $\perp$. On the other hand, if $\alpha =$'E', then P_0 outputs the signature $\sigma_{\text{SPS,OUT}} = \mathsf{SPS.Sign}(\text{SK}'_{\text{SPS},\alpha}, m_{\text{OUT}}) = \mathsf{SPS.Sign}(\text{SK}'_{\text{SPS},E}, m_{\text{OUT}})$, which is the same signature that P_1 is programmed to output in this case. Thus, both programs have identical outputs in this case.

(II) $(m_{\text{IN}} \neq m^{(\nu)}_{\iota+1,0})$: In this case, we use the correctness [Property (v)] of SPS described in Section 2.3.5 to conclude that $\alpha \neq$'E' and correctness [Properties (i) and (iv)] of SPS confirms that either $\alpha =$'-' or $\alpha =$'F'. Now, if $\alpha =$'-', then both programs output $\perp$ as earlier. Otherwise, if $\alpha =$'F', then P_0 outputs $\sigma_{\text{SPS,OUT}} = \mathsf{SPS.Sign}(\text{SK}'_{\text{SPS},\alpha}, m_{\text{OUT}}) = \mathsf{SPS.Sign}(\text{SK}'_{\text{SPS},F}, m_{\text{OUT}})$, which P_1 is programmed to output in this case. Therefore, both programs are functionally equivalent in this case as well.

□

Claim C.30 *Assuming* SPS *is a splittable signature scheme satisfying 'splitting indistinguishability', for any* PPT *adversary* $\mathcal{A}$*, for any security parameter* λ*,* $|\mathsf{Adv}^{(0,\nu-1,3,\iota',6)}_{\mathcal{A}}(\lambda) - \mathsf{Adv}^{(0,\nu-1,3,\iota',7)}_{\mathcal{A}}(\lambda)| \leq \mathsf{negl}(\lambda)$ *for some negligible function* negl*.*

Proof Suppose there exists a PPT adversary $\mathcal{A}$ for which $|\mathsf{Adv}^{(0,\nu-1,3,\iota',6)}_{\mathcal{A}}(\lambda) - \mathsf{Adv}^{(0,\nu-1,3,\iota',7)}_{\mathcal{A}}(\lambda)|$ is non-negligible. Below we construct a PPT adversary $\mathcal{B}$ that breaks the splitting indistinguishability of SPS using $\mathcal{A}$ as a sub-routine.

- $\mathcal{B}$ initializes $\mathcal{A}$ on input 1^λ and receives a challenge signing attribute string $x^* = x^*_0 \ldots x^*_{\ell^*-1} \in \mathcal{X}_{\text{CPRF}}$ with $|x^*| = \ell^*$ from $\mathcal{A}$.
- Upon receiving x^*, $\mathcal{B}$ proceeds as follows:
  1. $\mathcal{B}$ first generates $\text{HK} \xleftarrow{\$} \mathsf{SSB.Gen}(1^\lambda, n_{\text{SSB-BLK}} = 2^\lambda, i^* = 0)$ and computes $h^* = \mathcal{H}_{\text{HK}}(x^*)$.
  2. Then, $\mathcal{B}$ selects a PPRF key $K \xleftarrow{\$} \mathcal{F}.\mathsf{Setup}(1^\lambda)$.
  3. After that, $\mathcal{B}$ computes $r^*_{\text{SIG}} = \mathcal{F}(K, (h^*, \ell^*))$ followed by $(\widehat{\text{SK}}^*_{\text{SIG}}, \widehat{\text{VK}}^*_{\text{SIG}}) = \mathsf{SIG.Setup}(1^\lambda;\ r^*_{\text{SIG}})$.
  4. $\mathcal{B}$ returns the public parameters $\text{PP}_{\text{ABS}} = (\text{HK}, \mathcal{IO}(\mathsf{Verify.Prog}_{\text{ABS}}[K]))$ to $\mathcal{A}$.
- For $\eta \in [\hat{q}_{\text{KEY}}]$, in response to the η^{th} signing key query of $\mathcal{A}$ corresponding to TM $M^{(\eta)} \in \mathbb{M}_\lambda$ with $M^{(\eta)}(x^*) = 0$, if $\eta \neq \nu$, then $\mathcal{B}$ proceeds exactly as in $\mathsf{Hyb}_{0,\nu-1,3,\iota',6}$, while if $\eta = \nu$, then $\mathcal{B}$ proceeds as follows:
  1. $\mathcal{B}$ first selects PPRF keys $K^{(\nu)}_1, \ldots, K^{(\nu)}_\lambda, K^{(\nu)}_{\text{SPS},A}, K^{(\nu)}_{\text{SPS},B}, K^{(\nu)}_{\text{SPS},E}, K^{(\nu)}_{\text{SPS},F} \xleftarrow{\$} \mathcal{F}.\mathsf{Setup}(1^\lambda)$.
  2. Next, it generates $(\text{PP}^{(\nu)}_{\text{ACC}}, w^{(\nu)}_0, \text{STORE}^{(\nu)}_0) \xleftarrow{\$} \mathsf{ACC.Setup}(1^\lambda, n_{\text{ACC-BLK}} = 2^\lambda)$ and $(\text{PP}^{(\nu)}_{\text{ITR}}, v^{(\nu)}_0) \xleftarrow{\$} \mathsf{ITR.Setup}(1^\lambda, n_{\text{ITR}} = 2^\lambda)$.
  3. Then, $\mathcal{B}$ creates the punctured PPRF keys $K^{(\nu)}_{\text{SPS},E}\{(h^*, \iota+1)\} \xleftarrow{\$} \mathcal{F}.\mathsf{Puncture}(K^{(\nu)}_{\text{SPS},E}, (h^*, \iota+1))$ and $K^{(\nu)}_{\text{SPS},F}\{(h^*, \iota+1)\} \xleftarrow{\$} \mathcal{F}.\mathsf{Puncture}(K^{(\nu)}_{\text{SPS},F}, (h^*, \iota+1))$.

4. Next, it sets $m_{0,0}^{(\nu)} = (v_0^{(\nu)}, q_0^{(\nu)}, w_0^{(\nu)}, 0)$. For $j = 1, \ldots, \iota + 1$, it iteratively computes the following:
   - $\text{AUX}_j^{(\nu)} = \mathsf{ACC.Prep\text{-}Write}(\text{PP}_{\text{ACC}}^{(\nu)}, \text{STORE}_{j-1}^{(\nu)}, j-1)$
   - $w_j^{(\nu)} = \mathsf{ACC.Update}(\text{PP}_{\text{ACC}}^{(\nu)}, w_{j-1}^{(\nu)}, x_{j-1}^*, j-1, \text{AUX}_j^{(\nu)})$
   - $\text{STORE}_j^{(\nu)} = \mathsf{ACC.Write\text{-}Store}(\text{PP}_{\text{ACC}}^{(\nu)}, \text{STORE}_{j-1}^{(\nu)}, j-1, x_{j-1}^*)$
   - $v_j^{(\nu)} = \mathsf{ITR.Iterate}(\text{PP}_{\text{ITR}}^{(\nu)}, v_{j-1}^{(\nu)}, (q_0^{(\nu)}, w_{j-1}^{(\nu)}, 0))$

   It sets $m_{\iota+1,0}^{(\nu)} = (v_{\iota+1}^{(\nu)}, q_0^{(\nu)}, w_{\iota+1}^{(\nu)}, 0)$.
5. After that, $\mathcal{B}$ sends $m_{\iota+1,0}^{(\nu)}$ as the challenge message to its SPS splitting indistinguishability challenger $\mathcal{C}$ and receives back a tuple $(\sigma^*_{\text{SPS-ONE},m_{\iota+1,0}^{(\nu)}}, \text{VK}^*_{\text{SPS-ONE}}, \text{SK}^*_{\text{SPS-ABO}}, \text{VK}^*_{\text{SPS-ABO}})$, where
   - either $(\sigma^*_{\text{SPS-ONE},m_{\iota+1,0}^{(\nu)}}, \text{VK}^*_{\text{SPS-ONE}}, \text{SK}^*_{\text{SPS-ABO}}, \text{VK}^*_{\text{SPS-ABO}}) = (\sigma_{\text{SPS-ONE},m_{\iota+1,0}^{(\nu)}}, \text{VK}_{\text{SPS-ONE}}, \text{SK}_{\text{SPS-ABO}}, \text{VK}_{\text{SPS-ABO}})$
   - or $(\sigma^*_{\text{SPS-ONE},m_{\iota+1,0}^{(\nu)}}, \text{VK}^*_{\text{SPS-ONE}}, \text{SK}^*_{\text{SPS-ABO}}, \text{VK}^*_{\text{SPS-ABO}}) = (\sigma_{\text{SPS-ONE},m_{\iota+1,0}^{(\nu)}}, \text{VK}_{\text{SPS-ONE}}, \text{SK}'_{\text{SPS-ABO}}, \text{VK}'_{\text{SPS-ABO}})$

   such that $(\sigma_{\text{SPS-ONE},m_{\iota+1,0}^{(\nu)}}, \text{VK}_{\text{SPS-ONE}}, \text{SK}_{\text{SPS-ABO}}, \text{VK}_{\text{SPS-ABO}}) \xleftarrow{\$}$ $\mathsf{SPS.Split}(\text{SK}_{\text{SPS}}, m_{\iota+1,0}^{(\nu)}), (\sigma'_{\text{SPS-ONE},m_{\iota+1,0}^{(\nu)}}, \text{VK}'_{\text{SPS-ONE}}, \text{SK}'_{\text{SPS-ABO}}, \text{VK}'_{\text{SPS-ABO}}) \xleftarrow{\$}$ $\mathsf{SPS.Split}(\text{SK}'_{\text{SPS}}, m_{\iota+1,0}^{(\nu)})$, SK_{SPS} and SK'_{SPS} being two independently generated signing keys for SPS.
6. $\mathcal{B}$ gives $\mathcal{A}$ the signing key

$$\text{SK}_{\text{ABS}}\{M^{(\nu)}\} = \left(\begin{array}{l} \text{HK}, \text{PP}_{\text{ACC}}^{(\nu)}, w_0^{(\nu)}, \text{STORE}_0^{(\nu)}, \text{PP}_{\text{ITR}}^{(\nu)}, v_0^{(\nu)}, \\ \mathcal{IO}(\mathsf{Init\text{-}SPS.Prog}[q_0^{(\nu)}, w_0^{(\nu)}, v_0^{(\nu)}, K_{\text{SPS},E}^{(\nu)}\{(h^*, \iota+1)\}]), \\ \mathcal{IO}(\mathsf{Accumulate.Prog}^{(3,\iota',3)}[n_{\text{SSB-BLK}} = 2^\lambda, \text{HK}, \text{PP}_{\text{ACC}}^{(\nu)}, \text{PP}_{\text{ITR}}^{(\nu)}, \\ \quad K_{\text{SPS},E}^{(\nu)}\{(h^*, \iota+1)\}, K_{\text{SPS},F}^{(\nu)}\{(h^*, \iota+1)\}, \sigma^*_{\text{SPS-ONE},m_{\iota+1,0}^{(\nu)}}, \\ \quad \text{SK}^*_{\text{SPS-ABO}}, \text{VK}^*_{\text{SPS-ONE},}, \text{VK}^*_{\text{SPS-ABO}}, m_{\iota+1,0}^{(\nu)}, h^*, \ell^*]), \\ \mathcal{IO}(\mathsf{Change\text{-}SPS.Prog}^{(3,\iota,1)}[K_{\text{SPS},A}^{(\nu)}, K_{\text{SPS},B}^{(\nu)}, K_{\text{SPS},E}^{(\nu)}\{(h^*, \iota+1)\}, \\ \quad K_{\text{SPS},F}^{(\nu)}\{(h^*, \iota+1)\}, \text{VK}^*_{\text{SPS-ONE}}, \text{VK}^*_{\text{SPS-ABO}}, h^*, \ell^*]), \\ \mathcal{IO}(\mathsf{Constrained\text{-}Key.Prog}^{(1)}_{\text{ABS}}[M^{(\nu)}, T = 2^\lambda, t^{*(\nu)}, \text{PP}_{\text{ACC}}^{(\nu)}, \text{PP}_{\text{ITR}}^{(\nu)}, K, \\ \quad K_1^{(\nu)}, \ldots, K_\lambda^{(\nu)}, K_{\text{SPS},A}^{(\nu)}, K_{\text{SPS},B}^{(\nu)}, h^*, \ell^*]) \end{array}\right).$$

- For $\theta \in [\hat{q}_{\text{SIGN}}]$, in reply to the θ^{th} signature query of $\mathcal{A}$ on message $\mathsf{msg}^{(\theta)} \in \mathcal{M}_{\text{ABS}}$ under attribute string x^*, $\mathcal{B}$ computes $\sigma_{\text{SIG}}^{(\theta)} \xleftarrow{\$} \mathsf{SIG.Sign}(\widehat{\text{SK}}^*_{\text{SIG}}, \mathsf{msg}^{(\theta)})$ and provides $\mathcal{A}$ with $\sigma_{\text{ABS}}^{(\theta)} = (\widehat{\text{VK}}^*_{\text{SIG}}, \sigma_{\text{SIG}}^{(\theta)})$.
- Finally, $\mathcal{A}$ output a signature σ_{ABS}^* on some message msg^* under attribute string x^*. $\mathcal{B}$ outputs $\hat{b}' = 1$ as its guess bit in its SPS splitting indistinguishability experiment if $\mathcal{A}$ wins, i.e., if $\mathsf{ABS.Verify}(\text{PP}_{\text{ABS}}, x^*, \mathsf{msg}^*, \sigma_{\text{ABS}}^*) = 1$ and $\mathsf{msg}^* \neq \mathsf{msg}^{(\theta)}$ for any $\theta \in [\hat{q}_{\text{SIGN}}]$. Otherwise, $\mathcal{B}$ outputs $\hat{b}' = 0$ in its SPS splitting indistinguishability experiment.

Notice that if $(\sigma^*_{\text{SPS-ONE},m_{\iota+1,0}^{(\nu)}}, \text{VK}^*_{\text{SPS-ONE}}, \text{SK}^*_{\text{SPS-ABO}}, \text{VK}^*_{\text{SPS-ABO}}) = (\sigma_{\text{SPS-ONE},m_{\iota+1,0}^{(\nu)}}, \text{VK}_{\text{SPS-ONE}}, \text{SK}'_{\text{SPS-ABO}}, \text{VK}'_{\text{SPS-ABO}})$, then $\mathcal{B}$ perfectly simulates $\mathsf{Hyb}_{0,\nu-1,3,\iota',6}$. On the other

hand, if $(\sigma^*_{\text{SPS-ONE},m^{(\nu)}_{\iota+1,0}}, \text{VK}^*_{\text{SPS-ONE}}, \text{SK}^*_{\text{SPS-ABO}}, \text{VK}^*_{\text{SPS-ABO}}) = (\sigma_{\text{SPS-ONE},m^{(\nu)}_{\iota+1,0}}, \text{VK}_{\text{SPS-ONE}}, \text{SK}_{\text{SPS-ABO}}, \text{VK}_{\text{SPS-ABO}})$, then $\mathcal{B}$ perfectly simulates $\mathsf{Hyb}_{0,\nu-1,3,\iota',7}$. This completes the proof of Claim C.30. □

Claim C.31 *Assuming $\mathcal{IO}$ is a secure indistinguishability obfuscator for* $\mathsf{P/poly}$*, for any* PPT *adversary $\mathcal{A}$, for any security parameter λ, $|\mathsf{Adv}^{(0,\nu-1,3,\iota',7)}_{\mathcal{A}}(\lambda) - \mathsf{Adv}^{(0,\nu-1,3,\iota',8)}_{\mathcal{A}}(\lambda)| \leq \mathsf{negl}(\lambda)$ for some negligible function* negl.

Proof The only difference between $\mathsf{Hyb}_{0,\nu-1,3,\iota',7}$ and $\mathsf{Hyb}_{0,\nu-1,3,\iota',8}$ is the following: In $\mathsf{Hyb}_{0,\nu-1,3,\iota',7}$, $\mathcal{B}$ includes the programs $\mathcal{IO}(P_0)$ and $\mathcal{IO}(P'_0)$ within the ν^{th} signing key returned to $\mathcal{A}$, while in $\mathsf{Hyb}_{0,\nu-1,3,\iota',8}$, it includes the programs $\mathcal{IO}(P_1)$ and $\mathcal{IO}(P'_1)$ instead, where

- $P_0 = \mathsf{Accumulate.Prog}^{(3,\iota',3)}[n_{\text{SSB-BLK}} = 2^\lambda, \text{HK}, \text{PP}^{(\nu)}_{\text{ACC}}, \text{PP}^{(\nu)}_{\text{ITR}}, K^{(\nu)}_{\text{SPS},E}\{(h^*, \iota+1)\}, K^{(\nu)}_{\text{SPS},F}\{(h^*, \iota+1)\}, \sigma^{(\nu,\iota+1)}_{\text{SPS-ONE},\ m^{(\nu)}_{\iota+1,0},G}, \text{SK}^{(\nu,\iota+1)}_{\text{SPS-ABO},G}, \text{VK}^{(\nu,\iota+1)}_{\text{SPS-ONE},G}, \text{VK}^{(\nu,\iota+1)}_{\text{SPS-ABO},G}, m^{(\nu)}_{\iota+1,0}, h^*, \ell^*]$ (Fig. **36**),
- $P'_0 = \mathsf{Change\text{-}SPS.Prog}^{(3,\iota,1)}[K^{(\nu)}_{\text{SPS},A}, K^{(\nu)}_{\text{SPS},B}, K^{(\nu)}_{\text{SPS},E}\{(h^*, \iota+1)\}, K^{(\nu)}_{\text{SPS},F}\{(h^*, \iota+1)\}, \text{VK}^{(\nu,\iota=1)}_{\text{SPS-ONE},G}, \text{VK}^{(\nu,\iota+1)}_{\text{SPS-ABO},G}, h^*, \ell^*]$ (Fig. **34**),
- $P_1 = \mathsf{Accumulate.Prog}^{(3,\iota',4)}[n_{\text{SSB-BLK}} = 2^\lambda, \text{HK}, \text{PP}^{(\nu)}_{\text{ACC}}, \text{PP}^{(\nu)}_{\text{ITR}}, K^{(\nu)}_{\text{SPS},E}\{(h^*, \iota+1)\}, K^{(\nu)}_{\text{SPS},F}\{(h^*, \iota+1)\}, \text{SK}^{(\nu,\iota+1)}_{\text{SPS},G}, \text{VK}^{(\nu,\iota+1)}_{\text{SPS},G}, m^{(\nu)}_{\iota+1,0}, h^*, \ell^*]$ (Fig. **37**),
- $P'_1 = \mathsf{Change\text{-}SPS.Prog}^{(3,\iota,2)}[K^{(\nu)}_{\text{SPS},A}, K^{(\nu)}_{\text{SPS},B}, K^{(\nu)}_{\text{SPS},E}\{(h^*, \iota+1)\}, K^{(\nu)}_{\text{SPS},F}\{(h^*, \iota+1)\}, \text{VK}^{(\nu,\iota=1)}_{\text{SPS},G}, h^*, \ell^*]$ (Fig. **38**).

We will argue that the programs P_0 and P_1, as well as , the programs P'_0 and P'_1 are functionally equivalent, so that, by the security of $\mathcal{IO}$ Claim C.31 follows. First consider the programs P_0 and P_1. Clearly the only inputs on which the outputs of the two programs can possibly differ are those corresponding to $(h, i) = (h^*, \iota)$ and $(h, i) = (h^*, \iota+1)$. Now, for inputs corresponding to (h^*, ι), the outputs of the two programs are identical due to the correctness [Property (ii)] of the splittable signature SPS described in Section 2.3.5 and the fact that the hardwired signature $\sigma^{(\nu,\iota+1)}_{\text{SPS-ONE},m^{(\nu)}_{\iota+1,0},G}$ and the all-but-one signing key $\text{SK}^{(\nu,\iota+1)}_{\text{SPS-ABO},G}$ used by the program P_0 for inputs corresponding to (h^*, ι) are obtained by running $\mathsf{SPS.Split}(\text{SK}^{(\nu,\iota+1)}_{\text{SPS},G}, m^{(\nu)}_{\iota+1,0})$, while the hardwired signing key used by P_1 in this case is $\text{SK}^{(\nu,\iota+1)}_{\text{SPS},G}$. Similarly, for inputs corresponding to $(h^*, \iota+1)$, the outputs of the two programs are also identical because of the correctness [Properties (i), (iii), (v), (iv) and (vi)] of SPS and the fact that the hardwired one and all-but-one verification keys used by the program P_0 for inputs corresponding to $(h^*, \iota+1)$ are generated by running $\mathsf{SPS.Split}(\text{SK}^{(\nu,\iota+1)}_{\text{SPS},G}, m^{(\nu)}_{\iota+1,0})$, while the hardwired verification key used by the program P_1 in this case is $\text{VK}^{(\nu,\iota+1)}_{\text{SPS},G}$, which is the matching verification key of $\text{SK}^{(\nu,\iota+1)}_{\text{SPS},G}$. Hence, the two programs are functionally equivalent. The same type of argument holds for the programs P'_0 and P'_1. □

Claim C.32 *Assuming $\mathcal{F}$ is a secure puncturable pseudorandom function, for any* PPT *adversary $\mathcal{A}$, for any security parameter λ, $|\mathsf{Adv}^{(0,\nu-1,3,\iota',8)}_{\mathcal{A}}(\lambda) - \mathsf{Adv}^{(0,\nu-1,3,\iota',9)}_{\mathcal{A}}(\lambda)| \leq \mathsf{negl}(\lambda)$ for some negligible function* negl.

Proof The proof of Claim C.32 is analogous to that of Claim C.25 with some appropriate modifications that are readily identifiable. □

Claim C.33 *Assuming $\mathcal{IO}$ is a secure indistinguishability obfuscator for* $\mathsf{P/poly}$ *and $\mathcal{F}$ satisfies the correctness under puncturing property, for any* PPT *adversary $\mathcal{A}$, for any security parameter λ, $|\mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,3,\iota',9)}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,3,\iota',10)}(\lambda)| \leq \mathsf{negl}(\lambda)$ for some negligible function* negl.

Proof The proof of Claim C.33 is similar to that of Claim C.24 with some appropriate modifications which are easy to find out. □

□

Lemma C.7 *Assuming $\mathcal{IO}$ is a secure indistinguishability obfuscator for* $\mathsf{P/poly}$*, $\mathcal{F}$ is a secure puncturable pseudorandom function, and* SPS *is a secure splittable signature scheme satisfying '$\mathrm{VK}_{\text{SPS-ONE}}$ indistinguishability', '$\mathrm{VK}_{\text{SPS-ABO}}$ indistinguishability', as well as 'splitting indistinguishability', for any* PPT *adversary $\mathcal{A}$, for any security parameter λ, $|\mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,3,(\ell^*-1)')}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,4)}(\lambda)| \leq \mathsf{negl}(\lambda)$ for some negligible function* negl.

Proof The proof of Lemma C.7 is similar to that of Lemma C.6 with certain appropriate changes which can be readily determined. □

Lemma C.8 *Assuming $\mathcal{IO}$ is a secure indistinguishability obfuscator for* $\mathsf{P/poly}$*, $\mathcal{F}$ is a secure puncturable pseudorandom function, ACC is a secure positional accumulator possessing the 'indistinguishability of read setup' as well as 'read enforcing', and* SPS *is a secure splittable signature scheme satisfying '$\mathrm{VK}_{\text{SPS-ONE}}$ indistinguishability', '$\mathrm{VK}_{\text{SPS-ABO}}$ indistinguishability', as well as 'splitting indistinguishability', for any* PPT *adversary $\mathcal{A}$, for any security parameter λ, we have $|\mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,4)}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,4,0')}(\lambda)| \leq \mathsf{negl}(\lambda)$ for some negligible function* negl.

Proof In order to establish Lemma C.8, we consider the following sequence of intermediate hybrid experiments between $\mathsf{Hyb}_{0,\nu-1,4}$ and $\mathsf{Hyb}_{0,\nu-1,4,0'}$:

Sequence of Intermediate Hybrids between $\mathsf{Hyb}_{0,\nu-1,4}$ and $\mathsf{Hyb}_{0,\nu-1,4,0'}$

$\mathsf{Hyb}_{0,\nu-1,4\text{-I}}$: This experiment coincides with $\mathsf{Hyb}_{0,\nu-1,4}$.

$\mathsf{Hyb}_{0,\nu-1,4\text{-II}}$: This experiment is identical to $\mathsf{Hyb}_{0,\nu-1,4\text{-I}}$ except that in response to the ν^{th} signing key query of $\mathcal{A}$ corresponding to TM $M^{(\nu)} \in \mathbb{M}_\lambda$ with $M^{(\nu)}(x^*) = 0$, $\mathcal{B}$ executes the following steps:

1. It first generates all the PPRF keys as well as the public parameters for the positional accumulator and the iterator as in $\mathsf{Hyb}_{0,\nu-1,4\text{-I}}$.
2. Then, it creates the punctured PPRF keys $K_{\text{SPS},A}^{(\nu)}\{(h^*,\ell^*,0)\} \xleftarrow{\$} \mathcal{F}.\mathsf{Puncture}(K_{\text{SPS},A}^{(\nu)}, (h^*,\ell^*,0))$ and $K_{\text{SPS},B}^{(\nu)}\{(h^*,\ell^*,0)\} \xleftarrow{\$} \mathcal{F}.\mathsf{Puncture}(K_{\text{SPS},B}^{(\nu)}, (h^*,\ell^*,0))$.
3. After that it computes $r_{\text{SPS},C}^{(\nu,0)} = \mathcal{F}(K_{\text{SPS},A}^{(\nu)}, (h^*,\ell^*,0))$, $r_{\text{SPS},D}^{(\nu,0)} = \mathcal{F}(K_{\text{SPS},B}^{(\nu)}, (h^*,\ell^*,0))$, and forms $(\mathrm{SK}_{\text{SPS},C}^{(\nu,0)}, \mathrm{VK}_{\text{SPS},C}^{(\nu,0)}, \mathrm{VK}_{\text{SPS-REJ},C}^{(\nu,0)}) = \mathsf{SPS.Setup}(1^\lambda; r_{\text{SPS},C}^{(\nu,0)})$, $(\mathrm{SK}_{\text{SPS},D}^{(\nu,0)}, \mathrm{VK}_{\text{SPS},D}^{(\nu,0)}, \mathrm{VK}_{\text{SPS-REJ},D}^{(\nu,0)}) = \mathsf{SPS.Setup}(1^\lambda; r_{\text{SPS},D}^{(\nu,0)})$.

4. Next, it sets $m_{0,0}^{(\nu)} = (v_0^{(\nu)}, q_0^{(\nu)}, w_0^{(\nu)}, 0)$. For $j = 1, \ldots, \ell^*$, it iteratively computes the following:
   - $\text{AUX}_j^{(\nu)} = \mathsf{ACC.Prep\text{-}Write}(\text{PP}_{\text{ACC}}^{(\nu)}, \text{STORE}_{j-1}^{(\nu)}, j-1)$
   - $w_j^{(\nu)} = \mathsf{ACC.Update}(\text{PP}_{\text{ACC}}^{(\nu)}, w_{j-1}^{(\nu)}, x_{j-1}^*, j-1, \text{AUX}_j^{(\nu)})$
   - $\text{STORE}_j^{(\nu)} = \mathsf{ACC.Write\text{-}Store}(\text{PP}_{\text{ACC}}^{(\nu)}, \text{STORE}_{j-1}^{(\nu)}, j-1, x_{j-1}^*)$
   - $v_j^{(\nu)} = \mathsf{ITR.Iterate}(\text{PP}_{\text{ITR}}^{(\nu)}, v_{j-1}^{(\nu)}, (q_0^{(\nu)}, w_{j-1}^{(\nu)}, 0))$

   It sets $m_{\ell^*,0}^{(\nu)} = (v_{\ell^*}^{(\nu)}, q_0^{(\nu)}, w_{\ell^*}^{(\nu)}, 0)$.
5. It gives $\mathcal{A}$ the signing key

$$\text{SK}_{\text{ABS}}\{M^{(\nu)}\} = \left(\begin{array}{l} \text{HK}, \text{PP}_{\text{ACC}}^{(\nu)}, w_0^{(\nu)}, \text{STORE}_0^{(\nu)}, \text{PP}_{\text{ITR}}^{(\nu)}, v_0^{(\nu)}, \\ \mathcal{IO}(\mathsf{Init\text{-}SPS.Prog}[q_0^{(\nu)}, w_0^{(\nu)}, v_0^{(\nu)}, K_{\text{SPS},E}^{(\nu)}]), \\ \mathcal{IO}(\mathsf{Accumulate.Prog}[n_{\text{SSB-BLK}} = 2^\lambda, \text{HK}, \text{PP}_{\text{ACC}}^{(\nu)}, \text{PP}_{\text{ITR}}^{(\nu)}, K_{\text{SPS},E}^{(\nu)}]), \\ \mathcal{IO}(\underline{\mathsf{Change\text{-}SPS.Prog}^{(4,1)}[K_{\text{SPS},A}^{(\nu)}\{(h^*, \ell^*, 0)\}, K_{\text{SPS},B}^{(\nu)}\{(h^*, \ell^*, 0)\},} \\ \quad \underline{K_{\text{SPS},E}^{(\nu)}, \text{SK}_{\text{SPS},C}^{(\nu,0)}, \text{SK}_{\text{SPS},D}^{(\nu,0)}, m_{\ell^*,0}^{(\nu)}, h^*, \ell^*]}), \\ \mathcal{IO}(\underline{\mathsf{Constrained\text{-}Key.Prog}_{\text{ABS}}^{(1,0,1)}[M^{(\nu)}, T = 2^\lambda, t^{*(\nu)}, \text{PP}_{\text{ACC}}^{(\nu)}, \text{PP}_{\text{ITR}}^{(\nu)}, K,} \\ \quad \underline{K_1^{(\nu)}, \ldots, K_\lambda^{(\nu)}, K_{\text{SPS},A}^{(\nu)}\{(h^*, \ell^*, 0)\}, K_{\text{SPS},B}^{(\nu)}\{(h^*, \ell^*, 0)\}, \text{VK}_{\text{SPS},C}^{(\nu,0)},} \\ \quad \underline{\text{VK}_{\text{SPS},D}^{(\nu,0)}, h^*, \ell^*]}) \end{array}\right),$$

where the programs $\mathsf{Change\text{-}SPS.Prog}^{(4,1)}$ and $\mathsf{Constrained\text{-}Key.Prog}_{\text{ABS}}^{(1,0,1)}$ respectively are the modifications of the programs $\mathsf{Change\text{-}SPS.Prog}^{(4)}$ and $\mathsf{Constrained\text{-}Key.Prog}_{\text{ABS}}^{(1)}$ (Figs. **18** and **10**) and are depicted in Figs. **39** and **40**.

Constants:	Punctured PPRF keys $K_{\text{SPS},A}\{(h^*, \ell^*, 0)\}, K_{\text{SPS},B}\{(h^*, \ell^*, 0)\}$, PPRF key $K_{\text{SPS},E}$, Signing keys SK_C, SK_D, Message $m_{\ell^*,0}$, SSB hash value of challenge input h^*, Length of challenge input ℓ^*
Inputs:	TM state ST, Accumulator value w, Iterator value v, SSB hash value h, Length ℓ_{INP}, Signature $\sigma_{\text{SPS,IN}}$
Output:	Signature $\sigma_{\text{SPS,OUT}}$, or $\perp$

1. (a) Compute $r_{\text{SPS},E} = \mathcal{F}(K_{\text{SPS},E}, (h, \ell_{\text{INP}}), (\text{SK}_{\text{SPS},E}, \text{VK}_{\text{SPS},E}, \text{VK}_{\text{SPS-REJ},E}) = \mathsf{SPS.Setup}(1^\lambda; r_{\text{SPS},E})$.
   (b) Set $m = (v, \text{ST}, w, 0)$.
   (c) If $\mathsf{SPS.Verify}(\text{VK}_{\text{SPS},E}, m, \sigma_{\text{SPS,IN}}) = 0$, output $\perp$.
2. (a) If $(h, \ell_{\text{INP}}) \neq (h^*, \ell^*)$, compute $r_{\text{SPS},A} = \mathcal{F}(K_{\text{SPS},A}\{(h^*, \ell^*, 0)\}, (h, \ell_{\text{INP}}, 0))$, $(\text{SK}_{\text{SPS},A}, \text{VK}_{\text{SPS},A}, \text{VK}_{\text{SPS-REJ},A}) = \mathsf{SPS.Setup}(1^\lambda; r_{\text{SPS},A})$. Else, set $\text{SK}_{\text{SPS},A} = \text{SK}_C$.
   (b) If $(h, \ell_{\text{INP}}) \neq (h^*, \ell^*)$, compute $r_{\text{SPS},B} = \mathcal{F}(K_{\text{SPS},B}\{(h^*, \ell^*, 0)\}, (h, \ell_{\text{INP}}, 0))$, $(\text{SK}_{\text{SPS},B}, \text{VK}_{\text{SPS},B}, \text{VK}_{\text{SPS-REJ},B}) = \mathsf{SPS.Setup}(1^\lambda; r_{\text{SPS},B})$. Else, set $\text{SK}_{\text{SPS},B} = \text{SK}_D$.
   (c) If $[(h, \ell_{\text{INP}}) = (h^*, \ell^*)] \wedge [m \neq m_{\ell^*,0}]$, output $\sigma_{\text{SPS,OUT}} = \mathsf{SPS.Sign}(\text{SK}_{\text{SPS},B}, m)$. Else, output $\sigma_{\text{SPS,OUT}} = \mathsf{SPS.Sign}(\text{SK}_{\text{SPS},A}, m)$.

Fig. 39 $\mathsf{Change\text{-}SPS.Prog}^{(4,1)}$

$\mathsf{Hyb}_{0,\nu-1,4\text{-III}}$: This experiment is analogous to $\mathsf{Hyb}_{0,\nu-1,4\text{-II}}$ with the only exception that while constructing the ν^{th} signing key queried by $\mathcal{A}$, $\mathcal{B}$ selects $r_{\text{SPS},C}^{(\nu,0)}, r_{\text{SPS},D}^{(\nu,0)} \xleftarrow{\$} \mathcal{Y}_{\text{PPRF}}$, i.e., in other words, $\mathcal{B}$ generates $(\text{SK}_{\text{SPS},C}^{(\nu,0)}, \text{VK}_{\text{SPS},C}^{(\nu,0)}, \text{VK}_{\text{SPS-REJ},C}^{(\nu,0)}), (\text{SK}_{\text{SPS},D}^{(\nu,0)}, \text{VK}_{\text{SPS},D}^{(\nu,0)}, \text{VK}_{\text{SPS-REJ},D}^{(\nu,0)}) \xleftarrow{\$} \mathsf{SPS.Setup}(1^\lambda)$.

Constants: TM $M = \langle Q, \Sigma_{\text{INP}}, \Sigma_{\text{TAPE}}, \delta, q_0, q_{\text{AC}}, q_{\text{REJ}} \rangle$, Time bound $T = 2^\lambda$, Running time on challenge input t^*, Public parameters for positional accumulator PP_{ACC}, Public parameters for iterator PP_{ITR}, PPRF keys $K, K_1, \ldots, K_\lambda$, Punctured PPRF keys $K_{\text{SPS},A}\{(h^*, \ell^*, 0)\}, K_{\text{SPS},B}\{(h^*, \ell^*, 0)\}$, Verification keys VK_C, VK_D, SSB hash value of challenge input h^*, Length of challenge input ℓ^*

Inputs: Time t, String SEED_{IN}, Header position POS_{IN}, Symbol SYM_{IN}, TM state ST_{IN}, Accumulator value w_{IN}, Accumulator proof π_{ACC}, Auxiliary value AUX, Iterator value v_{IN}, SSB hash value h, Length ℓ_{INP}, Signature $\sigma_{\text{SPS,IN}}$

Output: CPRF evaluation $\mathcal{F}(K, (h, \ell_{\text{INP}}))$, or (Header Position POS_{OUT}, Symbol SYM_{OUT}, TM state ST_{OUT}, Accumulator value w_{OUT}, Iterator value v_{OUT}, Signature $\sigma_{\text{SPS,OUT}}$, String SEED_{OUT}), or $\perp$

1. Identify an integer τ such that $2^\tau \leq t < 2^{\tau+1}$. If $[\mathsf{PRG}(\text{SEED}_{\text{IN}}) \neq \mathsf{PRG}(\mathcal{F}(K_\tau, (h, \ell_{\text{INP}})))] \wedge [t > 1]$, output $\perp$.
2. If $\mathsf{ACC.Verify\text{-}Read}(\text{PP}_{\text{ACC}}, w_{\text{IN}}, \text{SYM}_{\text{IN}}, \text{POS}_{\text{IN}}, \pi_{\text{ACC}}) = 0$, output $\perp$.
3. (a) If $(h, \ell_{\text{INP}}, t) \neq (h^*, \ell^*, 1)$, compute $r_{\text{SPS},A} = \mathcal{F}(K_{\text{SPS},A}\{(h^*, \ell^*, 0)\}, (h, \ell_{\text{INP}}, t-1))$, $(\text{SK}_{\text{SPS},A}, \text{VK}_{\text{SPS},A}, \text{VK}_{\text{SPS-REJ},A}) = \mathsf{SPS.Setup}(1^\lambda; r_{\text{SPS},A})$.
Else, set $\text{VK}_{\text{SPS},A} = \text{VK}_C$.
(b) If $(h, \ell_{\text{INP}}, t) \neq (h^*, \ell^*, 1)$, compute $r_{\text{SPS},B} = \mathcal{F}(K_{\text{SPS},B}\{(h^*, \ell^*, 0)\}, (h, \ell_{\text{INP}}, t-1))$, $(\text{SK}_{\text{SPS},B}, \text{VK}_{\text{SPS},B}, \text{VK}_{\text{SPS-REJ},B}) = \mathsf{SPS.Setup}(1^\lambda; r_{\text{SPS},B})$.
Else, set $\text{VK}_{\text{SPS},B} = \text{VK}_D$.
(c) Set $m_{\text{IN}} = (v_{\text{IN}}, \text{ST}_{\text{IN}}, w_{\text{IN}}, \text{POS}_{\text{IN}})$ and $\alpha =$'-'.
(d) If $\mathsf{SPS.Verify}(\text{VK}_{\text{SPS},A}, m_{\text{IN}}, \sigma_{\text{SPS,IN}}) = 1$, set $\alpha =$'A'.
(e) If $[\alpha =$'-'$] \wedge [(t > t^*) \vee (h \neq h^*) \vee (\ell_{\text{INP}} \neq \ell^*)]$, output $\perp$.
Else if $[\alpha =$'-'$] \wedge [\mathsf{SPS.Verify}(\text{VK}_{\text{SPS},B}, m_{\text{IN}}, \sigma_{\text{SPS,IN}}) = 1]$, set $\alpha =$'B'.
(f) If $\alpha =$'-', output $\perp$.
4. (a) Compute $(\text{ST}_{\text{OUT}}, \text{SYM}_{\text{OUT}}, \beta) = \delta(\text{ST}_{\text{IN}}, \text{SYM}_{\text{IN}})$ and $\text{POS}_{\text{OUT}} = \text{POS}_{\text{IN}} + \beta$.
(b) If $\text{ST}_{\text{OUT}} = q_{\text{REJ}}$, output $\perp$.
Else if $[\text{ST}_{\text{OUT}} = q_{\text{AC}}] \wedge [\alpha =$'$B$'$]$, output $\perp$.
Else if $\text{ST}_{\text{OUT}} = q_{\text{AC}}$, perform the following:
(I) Compute $r_{\text{SIG}} = \mathcal{F}(K, (h, \ell_{\text{INP}}))$, $(\text{SK}_{\text{SIG}}, \text{VK}_{\text{SIG}}) = \mathsf{SIG.Setup}(1^\lambda; r_{\text{SIG}})$.
(II) Output $(\text{SK}_{\text{SIG}}, \text{VK}_{\text{SIG}})$.
5. (a) Compute $w_{\text{OUT}} = \mathsf{ACC.Update}(\text{PP}_{\text{ACC}}, w_{\text{IN}}, \text{SYM}_{\text{OUT}}, \text{POS}_{\text{IN}}, \text{AUX})$. If $w_{\text{OUT}} = \perp$, output $\perp$.
(b) Compute $v_{\text{OUT}} = \mathsf{ITR.Iterate}(\text{PP}_{\text{ITR}}, v_{\text{IN}}, (\text{ST}_{\text{IN}}, w_{\text{IN}}, \text{POS}_{\text{IN}}))$.
6. (a) Compute $r'_{\text{SPS},A} = \mathcal{F}(K_{\text{SPS},A}\{(h^*, \ell^*, 0)\}, (h, \ell_{\text{INP}}, t)), (\text{SK}'_{\text{SPS},A}, \text{VK}'_{\text{SPS},A}, \text{VK}'_{\text{SPS-REJ},A}) = \mathsf{SPS.Setup}(1^\lambda; r'_{\text{SPS},A})$.
(b) Compute $r'_{\text{SPS},B} = \mathcal{F}(K_{\text{SPS},B}\{(h^*, \ell^*, 0)\}, (h, \ell_{\text{INP}}, t)), (\text{SK}'_{\text{SPS},B}, \text{VK}'_{\text{SPS},B}, \text{VK}'_{\text{SPS-REJ},B}) = \mathsf{SPS.Setup}(1^\lambda; r'_{\text{SPS},B})$.
(c) Set $m_{\text{OUT}} = (v_{\text{OUT}}, \text{ST}_{\text{OUT}}, w_{\text{OUT}}, \text{POS}_{\text{OUT}})$.
Compute $\sigma_{\text{SPS,OUT}} = \mathsf{SPS.Sign}(\text{SK}'_{\text{SPS},\alpha}, m_{\text{OUT}})$.
7. If $t + 1 = 2^{\tau'}$, for some integer τ', set $\text{SEED}_{\text{OUT}} = \mathcal{F}(K_{\tau'}, (h, \ell_{\text{INP}}))$.
Else, set $\text{SEED}_{\text{OUT}} = \epsilon$.
8. Output $(\text{POS}_{\text{OUT}}, \text{SYM}_{\text{OUT}}, \text{ST}_{\text{OUT}}, w_{\text{OUT}}, v_{\text{OUT}}, \sigma_{\text{SPS,OUT}}, \text{SEED}_{\text{OUT}})$.

Fig. 40 $\mathsf{Constrained\text{-}Key.Prog}^{(1,0,1)}_{\text{ABS}}$

$\mathsf{Hyb}_{0,\nu-1,4\text{-IV}}$: This experiment is similar to $\mathsf{Hyb}_{0,\nu-1,4\text{-III}}$ except that in response to the ν^{th} signing key query of $\mathcal{A}$ corresponding to TM $M^{(\nu)} \in \mathbb{M}_\lambda$ with $M^{(\nu)}(x^*) = 0$, $\mathcal{B}$ executes the following steps:

1. It first generates all the PPRF keys as well as the public parameters for the positional accumulator and the iterator as in $\mathsf{Hyb}_{0,\nu-1,4\text{-III}}$.
2. Then, it creates the punctured PPRF keys $K^{(\nu)}_{\text{SPS},A}\{(h^*, \ell^*, 0)\} \xleftarrow{\$} \mathcal{F}.\mathsf{Puncture}(K^{(\nu)}_{\text{SPS},A}, (h^*, \ell^*, 0))$ and $K^{(\nu)}_{\text{SPS},B}\{(h^*, \ell^*, 0)\} \xleftarrow{\$} \mathcal{F}.\mathsf{Puncture}(K^{(\nu)}_{\text{SPS},B}, (h^*, \ell^*, 0))$.
3. After that it forms $(\text{SK}^{(\nu,0)}_{\text{SPS},C}, \text{VK}^{(\nu,0)}_{\text{SPS},C}, \text{VK}^{(\nu,0)}_{\text{SPS-REJ},C}), (\text{SK}^{(\nu,0)}_{\text{SPS},D}, \text{VK}^{(\nu,0)}_{\text{SPS},D}, \text{VK}^{(\nu,0)}_{\text{SPS-REJ},D}) \xleftarrow{\$} \mathsf{SPS.Setup}(1^\lambda)$.
4. Next, it computes $m^{(\nu)}_{\ell^*,0} = (v^{(\nu)}_{\ell^*}, q^{(\nu)}_0, w^{(\nu)}_{\ell^*}, 0)$ just as in $\mathsf{Hyb}_{0,\nu-1,4\text{-III}}$.

5. After that, it creates $(\sigma^{(\nu,0)}_{\text{SPS-ONE},m^{(\nu)}_{\ell^*,0},C},\ \text{VK}^{(\nu,0)}_{\text{SPS-ONE},C},\ \text{SK}^{(\nu,0)}_{\text{SPS-ABO},C},\ \text{VK}^{(\nu,0)}_{\text{SPS-ABO},C}) \xleftarrow{\$} \mathsf{SPS.Split}(\text{SK}^{(\nu,0)}_{\text{SPS},C},\ m^{(\nu)}_{\ell^*,0})$ and $(\sigma^{(\nu,0)}_{\text{SPS-ONE},m^{(\nu)}_{\ell^*,0},D},\ \text{VK}^{(\nu,0)}_{\text{SPS-ONE},D},\ \text{SK}^{(\nu,0)}_{\text{SPS-ABO},D},\ \text{VK}^{(\nu,0)}_{\text{SPS-ABO},D}) \xleftarrow{\$} \mathsf{SPS.Split}(\text{SK}^{(\nu,0)}_{\text{SPS},D}, m^{(\nu)}_{\ell^*,0})$.
6. It gives $\mathcal{A}$ the signing key

$$\text{SK}_{\text{ABS}}\{M^{(\nu)}\} = \left(\begin{array}{l} \text{HK}, \text{PP}^{(\nu)}_{\text{ACC}}, w^{(\nu)}_0, \text{STORE}^{(\nu)}_0, \text{PP}^{(\nu)}_{\text{ITR}}, v^{(\nu)}_0, \\ \mathcal{IO}(\mathsf{Init\text{-}SPS.Prog}[q^{(\nu)}_0, w^{(\nu)}_0, v^{(\nu)}_0, K^{(\nu)}_{\text{SPS},E}]), \\ \mathcal{IO}(\mathsf{Accumulate.Prog}[n_{\text{SSB-BLK}} = 2^\lambda, \text{HK}, \text{PP}^{(\nu)}_{\text{ACC}}, \text{PP}^{(\nu)}_{\text{ITR}}, K^{(\nu)}_{\text{SPS},E}]), \\ \mathcal{IO}(\mathsf{Change\text{-}SPS.Prog}^{(4,2)}[K^{(\nu)}_{\text{SPS},A}\{(h^*,\ell^*,0)\}, K^{(\nu)}_{\text{SPS},B}\{(h^*,\ell^*,0)\}, \\ \quad K^{(\nu)}_{\text{SPS},E}, \sigma^{(\nu,0)}_{\text{SPS-ONE},m^{(\nu)}_{\ell^*,0},C}, \text{SK}^{(\nu,0)}_{\text{SPS-ABO},D}, m^{(\nu)}_{\ell^*,0}, h^*, \ell^*]), \\ \mathcal{IO}(\mathsf{Constrained\text{-}Key.Prog}^{(1,0,1)}_{\text{ABS}}[M^{(\nu)}, T = 2^\lambda, t^{*(\nu)}, \text{PP}^{(\nu)}_{\text{ACC}}, \text{PP}^{(\nu)}_{\text{ITR}}, K, \\ \quad K^{(\nu)}_1, \ldots, K^{(\nu)}_\lambda, K^{(\nu)}_{\text{SPS},A}\{(h^*,\ell^*,0)\}, K^{(\nu)}_{\text{SPS},B}\{(h^*,\ell^*,0)\}, \text{VK}^{(\nu,0)}_{\text{SPS},C}, \\ \quad \text{VK}^{(\nu,0)}_{\text{SPS},D}, h^*, \ell^*]) \end{array}\right),$$

where the program $\mathsf{Change\text{-}SPS.Prog}^{(4,2)}$ is an alteration of the program $\mathsf{Change\text{-}SPS.Prog}^{(4,1)}$ (Fig. **39**) and is shown in Fig. **41**.

Constants: Punctured PPRF keys $K_{\text{SPS},A}\{(h^*,\ell^*,0)\}$, $K_{\text{SPS},B}\{(h^*,\ell^*,0)\}$, PPRF key $K_{\text{SPS},E}$, Signature σ_C, Signing key SK_D, Message $m_{\ell^*,0}$, SSB hash value of challenge input h^*, Length of challenge input ℓ^*

Inputs: TM state ST, Accumulator value w, Iterator value v, SSB hash value h, Length ℓ_{INP}, Signature $\sigma_{\text{SPS,IN}}$

Output: Signature $\sigma_{\text{SPS,OUT}}$, or $\perp$

1. (a) Compute $r_{\text{SPS},E} = \mathcal{F}(K_{\text{SPS},E}, (h, \ell_{\text{INP}}), (\text{SK}_{\text{SPS},E}, \text{VK}_{\text{SPS},E}, \text{VK}_{\text{SPS-REJ},E}) = \mathsf{SPS.Setup}(1^\lambda; r_{\text{SPS},E})$.
   (b) Set $m = (v, \text{ST}, w, 0)$.
   (c) If $\mathsf{SPS.Verify}(\text{VK}_{\text{SPS},E}, m, \sigma_{\text{SPS,IN}}) = 0$, output $\perp$.
2. (a) If $(h, \ell_{\text{INP}}) \neq (h^*, \ell^*)$, compute $r_{\text{SPS},A} = \mathcal{F}(K_{\text{SPS},A}\{(h^*,\ell^*,0)\}, (h, \ell_{\text{INP}}, 0))$, $(\text{SK}_{\text{SPS},A}, \text{VK}_{\text{SPS},A}, \text{VK}_{\text{SPS-REJ},A}) = \mathsf{SPS.Setup}(1^\lambda; r_{\text{SPS},A})$.
   (b) If $(h, \ell_{\text{INP}}) \neq (h^*, \ell^*)$, compute $r_{\text{SPS},B} = \mathcal{F}(K_{\text{SPS},B}\{(h^*,\ell^*,0)\}, (h, \ell_{\text{INP}}, 0))$, $(\text{SK}_{\text{SPS},B}, \text{VK}_{\text{SPS},B}, \text{VK}_{\text{SPS-REJ},B}) = \mathsf{SPS.Setup}(1^\lambda; r_{\text{SPS},B})$.
   Else, set $\text{SK}_{\text{SPS},B} = \text{SK}_D$.
   (c) If $[(h, \ell_{\text{INP}}) = (h^*, \ell^*)] \wedge [m \neq m_{\ell^*,0}]$, output $\sigma_{\text{SPS,OUT}} = \mathsf{SPS.Sign}(\text{SK}_{\text{SPS},B}, m)$.
   Else if $[(h, \ell_{\text{INP}} = (h^*, \ell^*)] \wedge [m = m_{\ell^*,0}]$, output $\sigma_{\text{SPS,OUT}} = \sigma_C$.
   Else, output $\sigma_{\text{SPS,OUT}} = \mathsf{SPS.Sign}(\text{SK}_{\text{SPS},A}, m)$.

Fig. 41 $\mathsf{Change\text{-}SPS.Prog}^{(4,2)}$

$\mathsf{Hyb}_{\mathbf{0},\nu-\mathbf{1},\mathbf{4\text{-}V}}$: In this experiment, in response to the ν^{th} signing key query of $\mathcal{A}$ corresponding to TM $M^{(\nu)} \in \mathbb{M}_\lambda$ with $M^{(\nu)}(x^*) = 0$, $\mathcal{B}$ generates all the components

as in $\mathsf{Hyb}_{0,\nu-1,4\text{-IV}}$, however, it hands $\mathcal{A}$ the signing key

$$\mathrm{SK}_{\mathrm{ABS}}\{M^{(\nu)}\} = \left(\begin{array}{l} \mathrm{HK}, \mathrm{PP}_{\mathrm{ACC}}^{(\nu)}, w_0^{(\nu)}, \mathrm{STORE}_0^{(\nu)}, \mathrm{PP}_{\mathrm{ITR}}^{(\nu)}, v_0^{(\nu)}, \\ \mathcal{IO}(\mathsf{Init\text{-}SPS.Prog}[q_0^{(\nu)}, w_0^{(\nu)}, v_0^{(\nu)}, K_{\mathrm{SPS},E}^{(\nu)}]), \\ \mathcal{IO}(\mathsf{Accumulate.Prog}[n_{\text{SSB-BLK}} = 2^\lambda, \mathrm{HK}, \mathrm{PP}_{\mathrm{ACC}}^{(\nu)}, \mathrm{PP}_{\mathrm{ITR}}^{(\nu)}, K_{\mathrm{SPS},E}^{(\nu)}]), \\ \mathcal{IO}(\mathsf{Change\text{-}SPS.Prog}^{(4,2)}[K_{\mathrm{SPS},A}^{(\nu)}\{(h^*,\ell^*,0)\}, K_{\mathrm{SPS},B}^{(\nu)}\{(h^*,\ell^*,0)\}, \\ \quad K_{\mathrm{SPS},E}^{(\nu)}, \sigma^{(\nu,0)}_{\text{SPS-ONE},m_{\ell^*,0}^{(\nu)},C}, \mathrm{SK}_{\text{SPS-ABO},D}^{(\nu,0)}, m_{\ell^*,0}^{(\nu)}, h^*, \ell^*]), \\ \mathcal{IO}(\mathsf{Constrained\text{-}Key.Prog}_{\mathrm{ABS}}^{(1,0,1)}[M^{(\nu)}, T = 2^\lambda, t^{*(\nu)}, \mathrm{PP}_{\mathrm{ACC}}^{(\nu)}, \mathrm{PP}_{\mathrm{ITR}}^{(\nu)}, K, \\ \quad K_1^{(\nu)}, \ldots, K_\lambda^{(\nu)}, K_{\mathrm{SPS},A}^{(\nu)}\{(h^*,\ell^*,0)\}, K_{\mathrm{SPS},B}^{(\nu)}\{(h^*,\ell^*,0)\}, \underline{\mathrm{VK}_{\text{SPS-ONE},C}^{(\nu,0)}}, \\ \quad \mathrm{VK}_{\mathrm{SPS},D}^{(\nu,0)}, h^*, \ell^*]) \end{array}\right),$$

The rest of the experiment is analogous to $\mathsf{Hyb}_{0,\nu-1,4\text{-IV}}$.

$\mathsf{Hyb}_{0,\nu-1,4\text{-VI}}$: In this experiment, in response to the ν^{th} signing key query of $\mathcal{A}$ corresponding to TM $M^{(\nu)} \in \mathbb{M}_\lambda$ with $M^{(\nu)}(x^*) = 0$, $\mathcal{B}$ creates all the components as in $\mathsf{Hyb}_{0,\nu-1,4\text{-V}}$, however, it hands $\mathcal{A}$ the signing key

$$\mathrm{SK}_{\mathrm{CPRF}}\{M^{(\nu)}\} = \left(\begin{array}{l} \mathrm{HK}, \mathrm{PP}_{\mathrm{ACC}}^{(\nu)}, w_0^{(\nu)}, \mathrm{STORE}_0^{(\nu)}, \mathrm{PP}_{\mathrm{ITR}}^{(\nu)}, v_0^{(\nu)}, \\ \mathcal{IO}(\mathsf{Init\text{-}SPS.Prog}[q_0^{(\nu)}, w_0^{(\nu)}, v_0^{(\nu)}, K_{\mathrm{SPS},E}^{(\nu)}]), \\ \mathcal{IO}(\mathsf{Accumulate.Prog}[n_{\text{SSB-BLK}} = 2^\lambda, \mathrm{HK}, \mathrm{PP}_{\mathrm{ACC}}^{(\nu)}, \mathrm{PP}_{\mathrm{ITR}}^{(\nu)}, K_{\mathrm{SPS},E}^{(\nu)}]), \\ \mathcal{IO}(\mathsf{Change\text{-}SPS.Prog}^{(4,2)}[K_{\mathrm{SPS},A}^{(\nu)}\{(h^*,\ell^*,0)\}, K_{\mathrm{SPS},B}^{(\nu)}\{(h^*,\ell^*,0)\}, \\ \quad K_{\mathrm{SPS},E}^{(\nu)}, \sigma^{(\nu,0)}_{\text{SPS-ONE},m_{\ell^*,0}^{(\nu)},C}, \mathrm{SK}_{\text{SPS-ABO},D}^{(\nu,0)}, m_{\ell^*,0}^{(\nu)}, h^*, \ell^*]), \\ \mathcal{IO}(\mathsf{Constrained\text{-}Key.Prog}_{\mathrm{ABS}}^{(1,0,1)}[M^{(\nu)}, T = 2^\lambda, t^{*(\nu)}, \mathrm{PP}_{\mathrm{ACC}}^{(\nu)}, \mathrm{PP}_{\mathrm{ITR}}^{(\nu)}, K, \\ \quad K_1^{(\nu)}, \ldots, K_\lambda^{(\nu)}, K_{\mathrm{SPS},A}^{(\nu)}\{(h^*,\ell^*,0)\}, K_{\mathrm{SPS},B}^{(\nu)}\{(h^*,\ell^*,0)\}, \mathrm{VK}_{\text{SPS-ONE},C}^{(\nu,0)}, \\ \quad \underline{\mathrm{VK}_{\text{SPS-ABO},D}^{(\nu,0)}}, h^*, \ell^*]) \end{array}\right),$$

The rest of the experiment is similar to $\mathsf{Hyb}_{0,\nu-1,4\text{-V}}$.

$\mathsf{Hyb}_{0,\nu-1,4\text{-VII}}$: In this experiment, to answer the ν^{th} signing key query of $\mathcal{A}$ corresponding to TM $M^{(\nu)} \in \mathbb{M}_\lambda$ with $M^{(\nu)}(x^*) = 0$, $\mathcal{B}$ generates all the components just as in $\mathsf{Hyb}_{0,\nu-1,4\text{-VI}}$, but it provides $\mathcal{A}$ with the signing key

$$\mathrm{SK}_{\mathrm{ABS}}\{M^{(\nu)}\} = \left(\begin{array}{l} \mathrm{HK}, \mathrm{PP}_{\mathrm{ACC}}^{(\nu)}, w_0^{(\nu)}, \mathrm{STORE}_0^{(\nu)}, \mathrm{PP}_{\mathrm{ITR}}^{(\nu)}, v_0^{(\nu)}, \\ \mathcal{IO}(\mathsf{Init\text{-}SPS.Prog}[q_0^{(\nu)}, w_0^{(\nu)}, v_0^{(\nu)}, K_{\mathrm{SPS},E}^{(\nu)}]), \\ \mathcal{IO}(\mathsf{Accumulate.Prog}[n_{\text{SSB-BLK}} = 2^\lambda, \mathrm{HK}, \mathrm{PP}_{\mathrm{ACC}}^{(\nu)}, \mathrm{PP}_{\mathrm{ITR}}^{(\nu)}, K_{\mathrm{SPS},E}^{(\nu)}]), \\ \mathcal{IO}(\mathsf{Change\text{-}SPS.Prog}^{(4,2)}[K_{\mathrm{SPS},A}^{(\nu)}\{(h^*,\ell^*,0)\}, K_{\mathrm{SPS},B}^{(\nu)}\{(h^*,\ell^*,0)\}, \\ \quad K_{\mathrm{SPS},E}^{(\nu)}, \sigma^{(\nu,0)}_{\text{SPS-ONE},m_{\ell^*,0}^{(\nu)},C}, \mathrm{SK}_{\text{SPS-ABO},D}^{(\nu,0)}, m_{\ell^*,0}^{(\nu)}, h^*, \ell^*]), \\ \mathcal{IO}(\underline{\mathsf{Constrained\text{-}Key.Prog}_{\mathrm{ABS}}^{(1,0,2)}[M^{(\nu)}, T = 2^\lambda, t^{*(\nu)}, \mathrm{PP}_{\mathrm{ACC}}^{(\nu)}, \mathrm{PP}_{\mathrm{ITR}}^{(\nu)}, K,} \\ \quad \underline{K_1^{(\nu)}, \ldots, K_\lambda^{(\nu)}, K_{\mathrm{SPS},A}^{(\nu)}\{(h^*,\ell^*,0)\}, K_{\mathrm{SPS},B}^{(\nu)}\{(h^*,\ell^*,0)\}, \mathrm{VK}_{\text{SPS-ONE},C}^{(\nu,0)},} \\ \quad \underline{\mathrm{VK}_{\text{SPS-ABO},D}^{(\nu,0)}, m_{\ell^*,0}^{(\nu)}, h^*, \ell^*]}) \end{array}\right),$$

where the program $\mathsf{Constrained\text{-}Key.Prog}_{\mathrm{ABS}}^{(1,0,2)}$ is a modification of the program $\mathsf{Constrained\text{-}Key.Prog}_{\mathrm{ABS}}^{(1,0,1)}$ (Fig. **40**) and is shown in Fig. **42**. The rest of the experiment is analogous to $\mathsf{Hyb}_{0,\nu-1,4\text{-VI}}$.

Constants: TM $M = \langle Q, \Sigma_{\mathrm{INP}}, \Sigma_{\mathrm{TAPE}}, \delta, q_0, q_{\mathrm{AC}}, q_{\mathrm{REJ}} \rangle$, Time bound $T = 2^\lambda$, Running time on challenge input t^*, Public parameters for positional accumulator $\mathrm{PP}_{\mathrm{ACC}}$, Public parameters for iterator $\mathrm{PP}_{\mathrm{ITR}}$, PPRF keys $K, K_1, \ldots, K_\lambda$, Punctured PPRF keys $K_{\mathrm{SPS},A}\{(h^*, \ell^*, 0)\}$, $K_{\mathrm{SPS},B}\{(h^*, \ell^*, 0)\}$, Verification keys VK_C, VK_D, Message $m_{\ell^*,0}$, SSB hash value of challenge input h^*, Length of challenge input ℓ^*

Inputs: Time t, String $\mathrm{SEED}_{\mathrm{IN}}$, Header position $\mathrm{POS}_{\mathrm{IN}}$, Symbol $\mathrm{SYM}_{\mathrm{IN}}$, TM state $\mathrm{ST}_{\mathrm{IN}}$, Accumulator value w_{IN}, Accumulator proof π_{ACC}, Auxiliary value AUX, Iterator value v_{IN}, SSB hash value h, Length ℓ_{INP}, Signature $\sigma_{\mathrm{SPS,IN}}$

Output: CPRF evaluation $\mathcal{F}(K, (h, \ell_{\mathrm{INP}}))$, or (Header Position $\mathrm{POS}_{\mathrm{OUT}}$, Symbol $\mathrm{SYM}_{\mathrm{OUT}}$, TM state $\mathrm{ST}_{\mathrm{OUT}}$, Accumulator value w_{OUT}, Iterator value v_{OUT}, Signature $\sigma_{\mathrm{SPS,OUT}}$, String $\mathrm{SEED}_{\mathrm{OUT}}$), or $\perp$

1. Identify an integer τ such that $2^\tau \leq t < 2^{\tau+1}$. If $[\mathsf{PRG}(\mathrm{SEED}_{\mathrm{IN}}) \neq \mathsf{PRG}(\mathcal{F}(K_\tau, (h, \ell_{\mathrm{INP}})))] \wedge [t > 1]$, output $\perp$.
2. If $\mathsf{ACC.Verify\text{-}Read}(\mathrm{PP}_{\mathrm{ACC}}, w_{\mathrm{IN}}, \mathrm{SYM}_{\mathrm{IN}}, \mathrm{POS}_{\mathrm{IN}}, \pi_{\mathrm{ACC}}) = 0$, output $\perp$.
3. (a) If $(h, \ell_{\mathrm{INP}}, t) \neq (h^*, \ell^*, 1)$, compute $r_{\mathrm{SPS},A} = \mathcal{F}(K_{\mathrm{SPS},A}\{(h^*, \ell^*, 0)\}, (h, \ell_{\mathrm{INP}}, t-1))$, $(\mathrm{SK}_{\mathrm{SPS},A}, \mathrm{VK}_{\mathrm{SPS},A}, \mathrm{VK}_{\mathrm{SPS\text{-}REJ},A}) = \mathsf{SPS.Setup}(1^\lambda; r_{\mathrm{SPS},A})$.
   Else, set $\mathrm{VK}_{\mathrm{SPS},A} = \mathrm{VK}_C$.
   (b) If $(h, \ell_{\mathrm{INP}}, t) \neq (h^*, \ell^*, 1)$, compute $r_{\mathrm{SPS},B} = \mathcal{F}(K_{\mathrm{SPS},B}\{(h^*, \ell^*, 0)\}, (h, \ell_{\mathrm{INP}}, t-1))$, $(\mathrm{SK}_{\mathrm{SPS},B}, \mathrm{VK}_{\mathrm{SPS},B}, \mathrm{VK}_{\mathrm{SPS\text{-}REJ},B}) = \mathsf{SPS.Setup}(1^\lambda; r_{\mathrm{SPS},B})$.
   Else, set $\mathrm{VK}_{\mathrm{SPS},B} = \mathrm{VK}_D$.
   (c) Set $m_{\mathrm{IN}} = (v_{\mathrm{IN}}, \mathrm{ST}_{\mathrm{IN}}, w_{\mathrm{IN}}, \mathrm{POS}_{\mathrm{IN}})$ and $\alpha =$'-'.
   (d) If $\mathsf{SPS.Verify}(\mathrm{VK}_{\mathrm{SPS},A}, m_{\mathrm{IN}}, \sigma_{\mathrm{SPS,IN}}) = 1$, set $\alpha =$'A'.
   (e) If $[\alpha =$'-'$] \wedge [(t > t^*) \vee (h \neq h^*) \vee (\ell_{\mathrm{INP}} \neq \ell^*)]$, output $\perp$.
   Else if $[\alpha =$'-'$] \wedge [\mathsf{SPS.Verify}(\mathrm{VK}_{\mathrm{SPS},B}, m_{\mathrm{IN}}, \sigma_{\mathrm{SPS,IN}}) = 1]$, set $\alpha =$'B'.
   (f) If $\alpha =$'-', output $\perp$.
4. (a) Compute $(\mathrm{ST}_{\mathrm{OUT}}, \mathrm{SYM}_{\mathrm{OUT}}, \beta) = \delta(\mathrm{ST}_{\mathrm{IN}}, \mathrm{SYM}_{\mathrm{IN}})$ and $\mathrm{POS}_{\mathrm{OUT}} = \mathrm{POS}_{\mathrm{IN}} + \beta$.
   (b) If $\mathrm{ST}_{\mathrm{OUT}} = q_{\mathrm{REJ}}$, output $\perp$.
   Else if $[\mathrm{ST}_{\mathrm{OUT}} = q_{\mathrm{AC}}] \wedge [\alpha =$'$B$'$]$, output $\perp$.
   Else if $\mathrm{ST}_{\mathrm{OUT}} = q_{\mathrm{AC}}$, perform the following:
   (I) Compute $r_{\mathrm{SIG}} = \mathcal{F}(K, (h, \ell_{\mathrm{INP}}))$, $(\mathrm{SK}_{\mathrm{SIG}}, \mathrm{VK}_{\mathrm{SIG}}) = \mathsf{SIG.Setup}(1^\lambda; r_{\mathrm{SIG}})$.
   (II) Output $(\mathrm{SK}_{\mathrm{SIG}}, \mathrm{VK}_{\mathrm{SIG}})$.
5. (a) Compute $w_{\mathrm{OUT}} = \mathsf{ACC.Update}(\mathrm{PP}_{\mathrm{ACC}}, w_{\mathrm{IN}}, \mathrm{SYM}_{\mathrm{OUT}}, \mathrm{POS}_{\mathrm{IN}}, \mathrm{AUX})$. If $w_{\mathrm{OUT}} = \perp$, output $\perp$.
   (b) Compute $v_{\mathrm{OUT}} = \mathsf{ITR.Iterate}(\mathrm{PP}_{\mathrm{ITR}}, v_{\mathrm{IN}}, (\mathrm{ST}_{\mathrm{IN}}, w_{\mathrm{IN}}, \mathrm{POS}_{\mathrm{IN}}))$.
6. (a) Compute $r'_{\mathrm{SPS},A} = \mathcal{F}(K_{\mathrm{SPS},A}\{(h^*, \ell^*, 0)\}, (h, \ell_{\mathrm{INP}}, t)), (\mathrm{SK}'_{\mathrm{SPS},A}, \mathrm{VK}'_{\mathrm{SPS},A}, \mathrm{VK}'_{\mathrm{SPS\text{-}REJ},A}) = \mathsf{SPS.Setup}(1^\lambda; r'_{\mathrm{SPS},A})$.
   (b) Compute $r'_{\mathrm{SPS},B} = \mathcal{F}(K_{\mathrm{SPS},B}\{(h^*, \ell^*, 0)\}, (h, \ell_{\mathrm{INP}}, t)), (\mathrm{SK}'_{\mathrm{SPS},B}, \mathrm{VK}'_{\mathrm{SPS},B}, \mathrm{VK}'_{\mathrm{SPS\text{-}REJ},B}) = \mathsf{SPS.Setup}(1^\lambda; r'_{\mathrm{SPS},B})$.
   (c) Set $m_{\mathrm{OUT}} = (v_{\mathrm{OUT}}, \mathrm{ST}_{\mathrm{OUT}}, w_{\mathrm{OUT}}, \mathrm{POS}_{\mathrm{OUT}})$.
   If $[(h, \ell_{\mathrm{INP}}, t) = (h^*, \ell^*, 1)] \wedge [m_{\mathrm{IN}} = m_{\ell^*,0}]$, compute $\sigma_{\mathrm{SPS,OUT}} = \mathsf{SPS.Sign}(\mathrm{SK}'_{\mathrm{SPS},A}, m_{\mathrm{OUT}})$.
   Else if $[(h, \ell_{\mathrm{INP}}, t) = (h^*, \ell^*, 1)] \wedge [m_{\mathrm{IN}} \neq m_{\ell^*,0}]$, compute $\sigma_{\mathrm{SPS,OUT}} = \mathsf{SPS.Sign}(\mathrm{SK}'_{\mathrm{SPS},B}, m_{\mathrm{OUT}})$.
   Else, compute $\sigma_{\mathrm{SPS,OUT}} = \mathsf{SPS.Sign}(\mathrm{SK}'_{\mathrm{SPS},\alpha}, m_{\mathrm{OUT}})$.
7. If $t + 1 = 2^{\tau'}$, for some integer τ', set $\mathrm{SEED}_{\mathrm{OUT}} = \mathcal{F}(K_{\tau'}, (h, \ell_{\mathrm{INP}}))$.
   Else, set $\mathrm{SEED}_{\mathrm{OUT}} = \epsilon$.
8. Output $(\mathrm{POS}_{\mathrm{OUT}}, \mathrm{SYM}_{\mathrm{OUT}}, \mathrm{ST}_{\mathrm{OUT}}, w_{\mathrm{OUT}}, v_{\mathrm{OUT}}, \sigma_{\mathrm{SPS,OUT}}, \mathrm{SEED}_{\mathrm{OUT}})$.

Fig. 42 $\mathsf{Constrained\text{-}Key.Prog}_{\mathrm{ABS}}^{(1,0,2)}$

$\mathsf{Hyb}_{0,\nu-1,4\text{-VIII}}$: In This experiment is the same as $\mathsf{Hyb}_{0,\nu-1,4\text{-VII}}$ with the only exception that while creating the ν^{th} signing key queried by $\mathcal{A}$, $\mathcal{B}$ generates $(\mathrm{PP}_{\mathrm{ACC}}^{(\nu)}, w_0^{(\nu)}, \mathrm{STORE}_0^{(\nu)}) \xleftarrow{\$} \mathsf{ACC.Setup\text{-}Enforce\text{-}Read}(1^\lambda, n_{\text{ACC-BLK}} = 2^\lambda, ((x_0^*, 0), \ldots, (x_{\ell^*-1}^*, \ell^* - 1)), i^* = 0)$.

$\mathsf{Hyb}_{0,\nu-1,4\text{-IX}}$: In this experiment, to answer the ν^{th} constrained key query of $\mathcal{A}$ corresponding to TM $M^{(\nu)} \in \mathbb{M}_\lambda$ with $M^{(\nu)}(x^*) = 0$, $\mathcal{B}$ generates all the components just as in $\mathsf{Hyb}_{0,\nu-1,4\text{-VIII}}$, however, it gives $\mathcal{A}$ the signing key

$$\mathrm{SK}_{\mathrm{ABS}}\{M^{(\nu)}\} = \left(\begin{array}{l} \mathrm{HK}, \mathrm{PP}_{\mathrm{ACC}}^{(\nu)}, w_0^{(\nu)}, \mathrm{STORE}_0^{(\nu)}, \mathrm{PP}_{\mathrm{ITR}}^{(\nu)}, v_0^{(\nu)}, \\ \mathcal{IO}(\mathsf{Init\text{-}SPS.Prog}[q_0^{(\nu)}, w_0^{(\nu)}, v_0^{(\nu)}, K_{\mathrm{SPS},E}^{(\nu)}]), \\ \mathcal{IO}(\mathsf{Accumulate.Prog}[n_{\text{SSB-BLK}} = 2^\lambda, \mathrm{HK}, \mathrm{PP}_{\mathrm{ACC}}^{(\nu)}, \mathrm{PP}_{\mathrm{ITR}}^{(\nu)}, K_{\mathrm{SPS},E}^{(\nu)}]), \\ \mathcal{IO}(\mathsf{Change\text{-}SPS.Prog}^{(4,2)}[K_{\mathrm{SPS},A}^{(\nu)}\{(h^*, \ell^*, 0)\}, K_{\mathrm{SPS},B}^{(\nu)}\{(h^*, \ell^*, 0)\}, \\ \quad K_{\mathrm{SPS},E}^{(\nu)}, \sigma^{(\nu,0)}_{\text{SPS-ONE},m_{\ell^*,0}^{(\nu)},C}, \mathrm{SK}_{\text{SPS-ABO},D}^{(\nu,0)}, m_{\ell^*,0}^{(\nu)}, h^*, \ell^*]), \\ \mathcal{IO}(\underline{\mathsf{Constrained\text{-}Key.Prog}_{\mathrm{ABS}}^{(1,0,3)}[M^{(\nu)}, T = 2^\lambda, t^{*(\nu)}, \mathrm{PP}_{\mathrm{ACC}}^{(\nu)}, \mathrm{PP}_{\mathrm{ITR}}^{(\nu)}, K,} \\ \quad \underline{K_1^{(\nu)}, \ldots, K_\lambda^{(\nu)}, K_{\mathrm{SPS},A}^{(\nu)}\{(h^*, \ell^*, 0)\}, K_{\mathrm{SPS},B}^{(\nu)}\{(h^*, \ell^*, 0)\}, \mathrm{VK}_{\text{SPS-ONE},C}^{(\nu,0)}}, \\ \quad \underline{\mathrm{VK}_{\text{SPS-ABO},D}^{(\nu,0)}, m_{\ell^*,0}^{(\nu)}, h^*, \ell^*]}) \end{array}\right),$$

where the program $\mathsf{Constrained\text{-}Key.Prog}_{\mathrm{ABS}}^{(1,0,3)}$ is a modification of the program $\mathsf{Constrained\text{-}Key.Prog}_{\mathrm{ABS}}^{(1,0,2)}$ (Fig. **42**) and is shown in Fig. **43**. The rest of the experiment is analogous to $\mathsf{Hyb}_{0,\nu-1,4\text{-VIII}}$.

$\mathsf{Hyb}_{0,\nu-1,4\text{-X}}$: This experiment is identical to $\mathsf{Hyb}_{0,\nu-1,4\text{-IX}}$ with the only exception that while constructing the ν^{th} signing key queried by $\mathcal{A}$, $\mathcal{B}$ forms $\underline{(\mathrm{PP}_{\mathrm{ACC}}^{(\nu)}, w_0^{(\nu)}, \mathrm{STORE}_0^{(\nu)}) \xleftarrow{\$} \mathsf{ACC.Setup}(1^\lambda, n_{\text{ACC-BLK}} = 2^\lambda)}$.

$\mathsf{Hyb}_{0,\nu-1,4\text{-XI}}$: In this experiment, in response to the ν^{th} signing key query of $\mathcal{A}$ corresponding to TM $M^{(\nu)} \in \mathbb{M}_\lambda$ with $M^{(\nu)}(x^*) = 0$, $\mathcal{B}$ creates all the components as in $\mathsf{Hyb}_{0,\nu-1,4\text{-X}}$ except that it $\underline{\text{does not generate } (\sigma^{(\nu,0)}_{\text{SPS-ONE},m_{\ell^*,0}^{(\nu)},D}, \mathrm{VK}_{\text{SPS-ONE},D}^{(\nu,0)}, \mathrm{SK}_{\text{SPS-ABO},D}^{\nu,0)}, \mathrm{VK}_{\text{SPS-ABO},D}^{(\nu,0)}) \xleftarrow{\$} \mathsf{SPS.Split}(\mathrm{SK}_{\mathrm{SPS},D}^{(\nu,0)}, m_{\ell^*,0}^{(\nu)})}$ and hands $\mathcal{A}$ the signing key

$$\mathrm{SK}_{\mathrm{ABS}}\{M^{(\nu)}\} = \left(\begin{array}{l} \mathrm{HK}, \mathrm{PP}_{\mathrm{ACC}}^{(\nu)}, w_0^{(\nu)}, \mathrm{STORE}_0^{(\nu)}, \mathrm{PP}_{\mathrm{ITR}}^{(\nu)}, v_0^{(\nu)}, \\ \mathcal{IO}(\mathsf{Init\text{-}SPS.Prog}[q_0^{(\nu)}, w_0^{(\nu)}, v_0^{(\nu)}, K_{\mathrm{SPS},E}^{(\nu)}]), \\ \mathcal{IO}(\mathsf{Accumulate.Prog}[n_{\text{SSB-BLK}} = 2^\lambda, \mathrm{HK}, \mathrm{PP}_{\mathrm{ACC}}^{(\nu)}, \mathrm{PP}_{\mathrm{ITR}}^{(\nu)}, K_{\mathrm{SPS},E}^{(\nu)}]), \\ \mathcal{IO}(\mathsf{Change\text{-}SPS.Prog}^{(4,2)}[K_{\mathrm{SPS},A}^{(\nu)}\{(h^*, \ell^*, 0)\}, K_{\mathrm{SPS},B}^{(\nu)}\{(h^*, \ell^*, 0)\}, \\ \quad K_{\mathrm{SPS},E}^{(\nu)}, \sigma^{(\nu,0)}_{\text{SPS-ONE},m_{\ell^*,0}^{(\nu)},C}, \underline{\mathrm{SK}_{\text{SPS-ABO},C}^{(\nu,0)}}, m_{\ell^*,0}^{(\nu)}, h^*, \ell^*]), \\ \mathcal{IO}(\mathsf{Constrained\text{-}Key.Prog}_{\mathrm{ABS}}^{(1,0,3)}[M^{(\nu)}, T = 2^\lambda, t^{*(\nu)}, \mathrm{PP}_{\mathrm{ACC}}^{(\nu)}, \mathrm{PP}_{\mathrm{ITR}}^{(\nu)}, K, \\ \quad K_1^{(\nu)}, \ldots, K_\lambda^{(\nu)}, K_{\mathrm{SPS},A}^{(\nu)}\{(h^*, \ell^*, 0)\}, K_{\mathrm{SPS},B}^{(\nu)}\{(h^*, \ell^*, 0)\}, \mathrm{VK}_{\text{SPS-ONE},C}^{(\nu,0)}, \\ \quad \underline{\mathrm{VK}_{\text{SPS-ABO},C}^{(\nu,0)}}, m_{\ell^*,0}^{(\nu)}, h^*, \ell^*]) \end{array}\right).$$

The rest of the experiment is analogous to $\mathsf{Hyb}_{0,\nu-1,4\text{-X}}$.

$\mathsf{Hyb}_{0,\nu-1,4\text{-XII}}$: In this experiment, to answer the ν^{th} signing key query of $\mathcal{A}$ corresponding to TM $M^{(\nu)} \in \mathbb{M}_\lambda$ with $M^{(\nu)}(x^*) = 0$, $\mathcal{B}$ creates all the components as in

Constants: TM $M = \langle Q, \Sigma_{\text{INP}}, \Sigma_{\text{TAPE}}, \delta, q_0, q_{\text{AC}}, q_{\text{REJ}} \rangle$, Time bound $T = 2^\lambda$, Running time on challenge input t^*, Public parameters for positional accumulator PP_{ACC}, Public parameters for iterator PP_{ITR}, PPRF keys $K, K_1, \ldots, K_\lambda$, Punctured PPRF keys $K_{\text{SPS},A}\{(h^*, \ell^*, 0)\}, K_{\text{SPS},B}\{(h^*, \ell^*, 0)\}$, Verification keys VK_C, VK_D, Message $m_{\ell^*,0}$, SSB hash value of challenge input h^*, Length of challenge input ℓ^*

Inputs: Time t, String SEED_{IN}, Header position POS_{IN}, Symbol SYM_{IN}, TM state ST_{IN}, Accumulator value w_{IN}, Accumulator proof π_{ACC}, Auxiliary value AUX, Iterator value v_{IN}, SSB hash value h, Length ℓ_{INP}, Signature $\sigma_{\text{SPS,IN}}$

Output: CPRF evaluation $\mathcal{F}(K, (h, \ell_{\text{INP}}))$, or (Header Position POS_{OUT}, Symbol SYM_{OUT}, TM state ST_{OUT}, Accumulator value w_{OUT}, Iterator value v_{OUT}, Signature $\sigma_{\text{SPS,OUT}}$, String SEED_{OUT}), or $\perp$

1. Identify an integer τ such that $2^\tau \leq t < 2^{\tau+1}$. If $[\mathsf{PRG}(\text{SEED}_{\text{IN}}) \neq \mathsf{PRG}(\mathcal{F}(K_\tau, (h, \ell_{\text{INP}})))] \wedge [t > 1]$, output $\perp$.
2. If $\mathsf{ACC.Verify\text{-}Read}(\text{PP}_{\text{ACC}}, w_{\text{IN}}, \text{SYM}_{\text{IN}}, \text{POS}_{\text{IN}}, \pi_{\text{ACC}}) = 0$, output $\perp$.
3. (a) If $(h, \ell_{\text{INP}}, t) \neq (h^*, \ell^*, 1)$, compute $r_{\text{SPS},A} = \mathcal{F}(K_{\text{SPS},A}\{(h^*, \ell^*, 0)\}, (h, \ell_{\text{INP}}, t-1))$, $(\text{SK}_{\text{SPS},A}, \text{VK}_{\text{SPS},A}, \text{VK}_{\text{SPS-REJ},A}) = \mathsf{SPS.Setup}(1^\lambda; r_{\text{SPS},A})$.
Else, set $\text{VK}_{\text{SPS},A} = \text{VK}_C$.
(b) If $(h, \ell_{\text{INP}}, t) \neq (h^*, \ell^*, 1)$, compute $r_{\text{SPS},B} = \mathcal{F}(K_{\text{SPS},B}\{(h^*, \ell^*, 0)\}, (h, \ell_{\text{INP}}, t-1))$, $(\text{SK}_{\text{SPS},B}, \text{VK}_{\text{SPS},B}, \text{VK}_{\text{SPS-REJ},B}) = \mathsf{SPS.Setup}(1^\lambda; r_{\text{SPS},B})$.
Else, set $\text{VK}_{\text{SPS},B} = \text{VK}_D$.
(c) Set $m_{\text{IN}} = (v_{\text{IN}}, \text{ST}_{\text{IN}}, w_{\text{IN}}, \text{POS}_{\text{IN}})$ and $\alpha =$'-'.
(d) If $\mathsf{SPS.Verify}(\text{VK}_{\text{SPS},A}, m_{\text{IN}}, \sigma_{\text{SPS,IN}}) = 1$, set $\alpha =$'A'.
(e) If $[\alpha =$'-'$] \wedge [(t > t^*) \vee (h \neq h^*) \vee (\ell_{\text{INP}} \neq \ell^*)]$, output $\perp$.
Else if $[\alpha =$'-'$] \wedge [\mathsf{SPS.Verify}(\text{VK}_{\text{SPS},B}, m_{\text{IN}}, \sigma_{\text{SPS,IN}}) = 1]$, set $\alpha =$'B'.
(f) If $\alpha =$'-', output $\perp$.
4. (a) Compute $(\text{ST}_{\text{OUT}}, \text{SYM}_{\text{OUT}}, \beta) = \delta(\text{ST}_{\text{IN}}, \text{SYM}_{\text{IN}})$ and $\text{POS}_{\text{OUT}} = \text{POS}_{\text{IN}} + \beta$.
(b) If $\text{ST}_{\text{OUT}} = q_{\text{REJ}}$, output $\perp$.
Else if $[\text{ST}_{\text{OUT}} = q_{\text{AC}}] \wedge [\alpha =$'$B$'$]$, output $\perp$.
Else if $[\text{ST}_{\text{OUT}} = q_{\text{AC}}] \wedge [\alpha =$'$A$'$] \wedge [(h, \ell_{\text{INP}}) = (h^*, \ell^*)] \wedge [t \leq 1]$, output $\perp$.
Else if $\text{ST}_{\text{OUT}} = q_{\text{AC}}$, perform the following:
(I) Compute $r_{\text{SIG}} = \mathcal{F}(K, (h, \ell_{\text{INP}}))$, $(\text{SK}_{\text{SIG}}, \text{VK}_{\text{SIG}}) = \mathsf{SIG.Setup}(1^\lambda;\ r_{\text{SIG}})$.
(II) Output $(\text{SK}_{\text{SIG}}, \text{VK}_{\text{SIG}})$.
5. (a) Compute $w_{\text{OUT}} = \mathsf{ACC.Update}(\text{PP}_{\text{ACC}}, w_{\text{IN}}, \text{SYM}_{\text{OUT}}, \text{POS}_{\text{IN}}, \text{AUX})$. If $w_{\text{OUT}} = \perp$, output $\perp$.
(b) Compute $v_{\text{OUT}} = \mathsf{ITR.Iterate}(\text{PP}_{\text{ITR}}, v_{\text{IN}}, (\text{ST}_{\text{IN}}, w_{\text{IN}}, \text{POS}_{\text{IN}}))$.
6. (a) Compute $r'_{\text{SPS},A} = \mathcal{F}(K_{\text{SPS},A}\{(h^*, \ell^*, 0)\}, (h, \ell_{\text{INP}}, t)), (\text{SK}'_{\text{SPS},A}, \text{VK}'_{\text{SPS},A}, \text{VK}'_{\text{SPS-REJ},A}) = \mathsf{SPS.Setup}(1^\lambda; r'_{\text{SPS},A})$.
(b) Compute $r'_{\text{SPS},B} = \mathcal{F}(K_{\text{SPS},B}\{(h^*, \ell^*, 0)\}, (h, \ell_{\text{INP}}, t)), (\text{SK}'_{\text{SPS},B}, \text{VK}'_{\text{SPS},B}, \text{VK}'_{\text{SPS-REJ},B}) = \mathsf{SPS.Setup}(1^\lambda; r'_{\text{SPS},B})$.
(c) Set $m_{\text{OUT}} = (v_{\text{OUT}}, \text{ST}_{\text{OUT}}, w_{\text{OUT}}, \text{POS}_{\text{OUT}})$.
If $[(h, \ell_{\text{INP}}, t) = (h^*, \ell^*, 1)] \wedge [m_{\text{IN}} = m_{\ell^*,0}]$, compute $\sigma_{\text{SPS,OUT}} = \mathsf{SPS.Sign}(\text{SK}'_{\text{SPS},A}, m_{\text{OUT}})$.
Else if $[(h, \ell_{\text{INP}}, t) = (h^*, \ell^*, 1)] \wedge [m_{\text{IN}} \neq m_{\ell^*,0}]$, compute $\sigma_{\text{SPS,OUT}} = \mathsf{SPS.Sign}(\text{SK}'_{\text{SPS},B}, m_{\text{OUT}})$.
Else, compute $\sigma_{\text{SPS,OUT}} = \mathsf{SPS.Sign}(\text{SK}'_{\text{SPS},\alpha}, m_{\text{OUT}})$.
7. If $t + 1 = 2^{\tau'}$, for some integer τ', set $\text{SEED}_{\text{OUT}} = \mathcal{F}(K_{\tau'}, (h, \ell_{\text{INP}}))$.
Else, set $\text{SEED}_{\text{OUT}} = \epsilon$.
8. Output $(\text{POS}_{\text{OUT}}, \text{SYM}_{\text{OUT}}, \text{ST}_{\text{OUT}}, w_{\text{OUT}}, v_{\text{OUT}}, \sigma_{\text{SPS,OUT}}, \text{SEED}_{\text{OUT}})$.

Fig. 43 $\mathsf{Constrained\text{-}Key.Prog}^{(1,0,3)}_{\text{ABS}}$

$\mathsf{Hyb}_{0,\nu-1,4\text{-XI}}$, but it provides $\mathcal{A}$ with the signing key

$$\text{SK}_{\text{ABS}}\{M^{(\nu)}\} = \left(\begin{array}{l} \text{HK}, \text{PP}^{(\nu)}_{\text{ACC}}, w^{(\nu)}_0, \text{STORE}^{(\nu)}_0, \text{PP}^{(\nu)}_{\text{ITR}}, v^{(\nu)}_0, \\ \mathcal{IO}(\mathsf{Init\text{-}SPS.Prog}[q^{(\nu)}_0, w^{(\nu)}_0, v^{(\nu)}_0, K^{(\nu)}_{\text{SPS},E}]), \\ \mathcal{IO}(\mathsf{Accumulate.Prog}[n_{\text{SSB-BLK}} = 2^\lambda, \text{HK}, \text{PP}^{(\nu)}_{\text{ACC}}, \text{PP}^{(\nu)}_{\text{ITR}}, K^{(\nu)}_{\text{SPS},E}]), \\ \mathcal{IO}(\underline{\mathsf{Change\text{-}SPS.Prog}^{(4,3)}[K^{(\nu)}_{\text{SPS},A}\{(h^*, \ell^*, 0)\}, K^{(\nu)}_{\text{SPS},E}, \text{SK}^{(\nu,0)}_{\text{SPS},C}, h^*, \ell^*]}), \\ \mathcal{IO}(\underline{\mathsf{Constrained\text{-}Key.Prog}^{(1,0,4)}_{\text{ABS}}[M^{(\nu)}, T = 2^\lambda, t^{*(\nu)}, \text{PP}^{(\nu)}_{\text{ACC}}, \text{PP}^{(\nu)}_{\text{ITR}}, K,} \\ \quad \underline{K^{(\nu)}_1, \ldots, K^{(\nu)}_\lambda, K^{(\nu)}_{\text{SPS},A}\{(h^*, \ell^*, 0)\}, K^{(\nu)}_{\text{SPS},B}\{(h^*, \ell^*, 0)\},} \\ \quad \underline{\text{VK}^{(\nu,0)}_{\text{SPS},C}, m^{(\nu)}_{\ell^*,0}, h^*, \ell^*]}) \end{array}\right),$$

where the programs $\mathsf{Change\text{-}SPS.Prog}^{(4,3)}$ and $\mathsf{Constrained\text{-}Key.Prog}^{(1,0,4)}_{\mathrm{ABS}}$ are the alterations of the programs $\mathsf{Change\text{-}SPS.Prog}^{(4,2)}$ and $\mathsf{Constrained\text{-}Key.Prog}^{(1,0,3)}_{\mathrm{ABS}}$ (Figs. **41** and **43**) and are shown in Figs. **44** and **45** respectively. The rest of the experiment is analogous to $\mathsf{Hyb}_{0,\nu-1,4\text{-XI}}$.

Constants:	Punctured PPRF key $K_{\mathrm{SPS},A}\{(h^*,\ell^*,0)\}$, PPRF key $K_{\mathrm{SPS},E}$, Signing key SK_C, SSB hash value of challenge input h^*, Length of challenge input ℓ^*
Inputs:	TM state ST, Accumulator value w, Iterator value v, SSB hash value h, Length ℓ_{INP}, Signature $\sigma_{\mathrm{SPS,IN}}$
Output:	Signature $\sigma_{\mathrm{SPS,OUT}}$, or $\perp$
1. (a)	Compute $r_{\mathrm{SPS},E} = \mathcal{F}(K_{\mathrm{SPS},E},(h,\ell_{\mathrm{INP}}),(\mathrm{SK}_{\mathrm{SPS},E},\mathrm{VK}_{\mathrm{SPS},E},\mathrm{VK}_{\mathrm{SPS\text{-}REJ},E}) = \mathsf{SPS.Setup}(1^\lambda; r_{\mathrm{SPS},E})$.
(b)	Set $m = (v, \mathrm{ST}, w, 0)$.
(c)	If $\mathsf{SPS.Verify}(\mathrm{VK}_{\mathrm{SPS},E}, m, \sigma_{\mathrm{SPS,IN}}) = 0$, output $\perp$.
2. (a)	If $(h,\ell_{\mathrm{INP}}) \neq (h^*,\ell^*)$, compute $r_{\mathrm{SPS},A} = \mathcal{F}(K_{\mathrm{SPS},A}\{(h^*,\ell^*,0)\},(h,\ell_{\mathrm{INP}},0))$, $(\mathrm{SK}_{\mathrm{SPS},A},\mathrm{VK}_{\mathrm{SPS},A},\mathrm{VK}_{\mathrm{SPS\text{-}REJ},A}) = \mathsf{SPS.Setup}(1^\lambda; r_{\mathrm{SPS},A})$.
(b)	If $(h,\ell_{\mathrm{INP}}) = (h^*,\ell^*)$, output $\sigma_{\mathrm{SPS,OUT}} = \mathsf{SPS.Sign}(\mathrm{SK}_C, m)$. Else, output $\sigma_{\mathrm{SPS,OUT}} = \mathsf{SPS.Sign}(\mathrm{SK}_{\mathrm{SPS},A}, m)$.

Fig. 44 $\mathsf{Change\text{-}SPS.Prog}^{(4,3)}$

$\mathsf{Hyb}_{0,\nu-1,4\text{-XIII}}$: This experiment is analogous to $\mathsf{Hyb}_{0,\nu-1,4\text{-XII}}$ except that while creating the ν^{th} signing key queried by $\mathcal{A}$, $\mathcal{B}$ and forms $(\mathrm{SK}^{(\nu,0)}_{\mathrm{SPS},C}, \mathrm{VK}^{(\nu,0)}_{\mathrm{SPS},C}, \mathrm{VK}^{(\nu,0)}_{\mathrm{SPS\text{-}REJ},C}) = \mathsf{SPS.Setup}(1^\lambda;\ r^{(\nu,0)}_{\mathrm{SPS},C} = \mathcal{F}(K^{(\nu)}_{\mathrm{SPS},A},(h^*,\ell^*,0)))$.

$\mathsf{Hyb}_{0,\nu-1,4\text{-XIV}}$: This experiment corresponds to $\mathsf{Hyb}_{0,\nu-1,4,0'}$.

Analysis

Let $\mathsf{Adv}^{(0,\nu-1,4\text{-}\vartheta)}_{\mathcal{A}}(\lambda)$ represents the advantage of the adversary $\mathcal{A}$, i.e., the absolute difference between $1/2$ and $\mathcal{A}$'s probability of correctly guessing the random bit selected by the challenger $\mathcal{B}$, in $\mathsf{Hyb}_{0,\nu-1,4\text{-}\vartheta}$, for $\vartheta \in \{\mathrm{I},\ldots,\mathrm{XIV}\}$. From the description of the hybrid experiments it follows that $\mathsf{Adv}^{(0,\nu-1,4)}_{\mathcal{A}}(\lambda) \equiv \mathsf{Adv}^{(0,\nu-1,4\text{-I})}_{\mathcal{A}}(\lambda)$ and $\mathsf{Adv}^{(0,\nu-1,4,0')}_{\mathcal{A}}(\lambda) \equiv \mathsf{Adv}^{(0,\nu-1,4\text{-XIV})}_{\mathcal{A}}(\lambda)$. Hence, we have

$$|\mathsf{Adv}^{(0,\nu-1,4)}_{\mathcal{A}}(\lambda) - \mathsf{Adv}^{(0,\nu-1,4,0')}_{\mathcal{A}}(\lambda)| \leq \sum_{\vartheta=\mathrm{II}}^{\mathrm{XIV}} |\mathsf{Adv}^{(0,\nu-1,4\text{-}(\vartheta-\mathrm{I}))}_{\mathcal{A}}(\lambda) - \mathsf{Adv}^{(0,\nu-1,4\text{-}\vartheta)}_{\mathcal{A}}(\lambda)|. \tag{C.7}$$

Claims C.34–C.46 below will show that the RHS of Eq. (C.7) is negligible and thus Lemma C.8 follows.

Claim C.34 *Assuming $\mathcal{IO}$ is a secure indistinguishability obfuscator for* $\mathsf{P/poly}$ *and $\mathcal{F}$ satisfies the correctness under puncturing property, for any* PPT *adversary $\mathcal{A}$, for any security parameter λ, $|\mathsf{Adv}^{(0,\nu-1,4\text{-}I)}_{\mathcal{A}}(\lambda) - \mathsf{Adv}^{(0,\nu-1,4\text{-}II)}_{\mathcal{A}}(\lambda)| \leq \mathsf{negl}(\lambda)$ for some negligible function* negl.

Proof The proof of Claim C.34 uses a similar kind of logic as that employed in the proof of Claim C.24. We omit the details here. □

Claim C.35 *Assuming $\mathcal{F}$ is a secure puncturable pseudorandom function, for any* PPT *adversary $\mathcal{A}$, for any security parameter λ, $|\mathsf{Adv}^{(0,\nu-1,4\text{-}II)}_{\mathcal{A}}(\lambda) - \mathsf{Adv}^{(0,\nu-1,4\text{-}III)}_{\mathcal{A}}(\lambda)| \leq \mathsf{negl}(\lambda)$ for some negligible function* negl.

Constants: TM $M = \langle Q, \Sigma_{\text{INP}}, \Sigma_{\text{TAPE}}, \delta, q_0, q_{\text{AC}}, q_{\text{REJ}} \rangle$, Time bound $T = 2^\lambda$, Running time on challenge input t^*, Public parameters for positional accumulator PP_{ACC}, Public parameters for iterator PP_{ITR}, PPRF keys $K, K_1, \ldots, K_\lambda$, Punctured PPRF keys $K_{\text{SPS},A}\{(h^*, \ell^*, 0)\}, K_{\text{SPS},B}\{(h^*, \ell^*, 0)\}$, Verification key VK_C, Message $m_{\ell^*,0}$, SSB hash value of challenge input h^*, Length of challenge input ℓ^*

Inputs: Time t, String SEED_{IN}, Header position POS_{IN}, Symbol SYM_{IN}, TM state ST_{IN}, Accumulator value w_{IN}, Accumulator proof π_{ACC}, Auxiliary value AUX, Iterator value v_{IN}, SSB hash value h, Length ℓ_{INP}, Signature $\sigma_{\text{SPS,IN}}$

Output: CPRF evaluation $\mathcal{F}(K, (h, \ell_{\text{INP}}))$, or (Header Position POS_{OUT}, Symbol SYM_{OUT}, TM state ST_{OUT}, Accumulator value w_{OUT}, Iterator value v_{OUT}, Signature $\sigma_{\text{SPS,OUT}}$, String SEED_{OUT}), or $\perp$

1. Identify an integer τ such that $2^\tau \leq t < 2^{\tau+1}$. If $[\mathsf{PRG}(\text{SEED}_{\text{IN}}) \neq \mathsf{PRG}(\mathcal{F}(K_\tau, (h, \ell_{\text{INP}})))] \wedge [t > 1]$, output $\perp$.
2. If $\mathsf{ACC.Verify\text{-}Read}(\text{PP}_{\text{ACC}}, w_{\text{IN}}, \text{SYM}_{\text{IN}}, \text{POS}_{\text{IN}}, \pi_{\text{ACC}}) = 0$, output $\perp$.
3. (a) If $(h, \ell_{\text{INP}}, t) \neq (h^*, \ell^*, 1)$, compute $r_{\text{SPS},A} = \mathcal{F}(K_{\text{SPS},A}\{(h^*, \ell^*, 0)\}, (h, \ell_{\text{INP}}, t-1))$, $(\text{SK}_{\text{SPS},A}, \text{VK}_{\text{SPS},A}, \text{VK}_{\text{SPS-REJ},A}) = \mathsf{SPS.Setup}(1^\lambda; r_{\text{SPS},A})$.
   Else, set $\text{VK}_{\text{SPS},A} = \text{VK}_C$.
   (b) If $(h, \ell_{\text{INP}}, t) \neq (h^*, \ell^*, 1)$, compute $r_{\text{SPS},B} = \mathcal{F}(K_{\text{SPS},B}\{(h^*, \ell^*, 0)\}, (h, \ell_{\text{INP}}, t-1))$, $(\text{SK}_{\text{SPS},B}, \text{VK}_{\text{SPS},B}, \text{VK}_{\text{SPS-REJ},B}) = \mathsf{SPS.Setup}(1^\lambda; r_{\text{SPS},B})$.
   (c) Set $m_{\text{IN}} = (v_{\text{IN}}, \text{ST}_{\text{IN}}, w_{\text{IN}}, \text{POS}_{\text{IN}})$ and $\alpha =$'-'.
   (d) If $\mathsf{SPS.Verify}(\text{VK}_{\text{SPS},A}, m_{\text{IN}}, \sigma_{\text{SPS,IN}}) = 1$, set $\alpha =$'A'.
   (e) If $[\alpha =$'-'$] \wedge [(t > t^*) \vee (t \leq 1) \vee (h \neq h^*) \vee (\ell_{\text{INP}} \neq \ell^*)]$, output $\perp$.
   Else if $[\alpha =$'-'$] \wedge [\mathsf{SPS.Verify}(\text{VK}_{\text{SPS},B}, m_{\text{IN}}, \sigma_{\text{SPS,IN}}) = 1]$, set $\alpha =$'B'.
   (f) If $\alpha =$'-', output $\perp$.
4. (a) Compute $(\text{ST}_{\text{OUT}}, \text{SYM}_{\text{OUT}}, \beta) = \delta(\text{ST}_{\text{IN}}, \text{SYM}_{\text{IN}})$ and $\text{POS}_{\text{OUT}} = \text{POS}_{\text{IN}} + \beta$.
   (b) If $\text{ST}_{\text{OUT}} = q_{\text{REJ}}$, output $\perp$.
   Else if $[\text{ST}_{\text{OUT}} = q_{\text{AC}}] \wedge [\alpha =$'$B$'$]$, output $\perp$.
   Else if $[\text{ST}_{\text{OUT}} = q_{\text{AC}}] \wedge [\alpha =$'$A$'$] \wedge [(h, \ell_{\text{INP}}) = (h^*, \ell^*)] \wedge [t \leq 1]$, output $\perp$.
   Else if $\text{ST}_{\text{OUT}} = q_{\text{AC}}$, perform the following:
   (I) Compute $r_{\text{SIG}} = \mathcal{F}(K, (h, \ell_{\text{INP}}))$, $(\text{SK}_{\text{SIG}}, \text{VK}_{\text{SIG}}) = \mathsf{SIG.Setup}(1^\lambda; r_{\text{SIG}})$.
   (II) Output $(\text{SK}_{\text{SIG}}, \text{VK}_{\text{SIG}})$.
5. (a) Compute $w_{\text{OUT}} = \mathsf{ACC.Update}(\text{PP}_{\text{ACC}}, w_{\text{IN}}, \text{SYM}_{\text{OUT}}, \text{POS}_{\text{IN}}, \text{AUX})$. If $w_{\text{OUT}} = \perp$, output $\perp$.
   (b) Compute $v_{\text{OUT}} = \mathsf{ITR.Iterate}(\text{PP}_{\text{ITR}}, v_{\text{IN}}, (\text{ST}_{\text{IN}}, w_{\text{IN}}, \text{POS}_{\text{IN}}))$.
6. (a) Compute $r'_{\text{SPS},A} = \mathcal{F}(K_{\text{SPS},A}\{(h^*, \ell^*, 0)\}, (h, \ell_{\text{INP}}, t)), (\text{SK}'_{\text{SPS},A}, \text{VK}'_{\text{SPS},A}, \text{VK}'_{\text{SPS-REJ},A}) = \mathsf{SPS.Setup}(1^\lambda; r'_{\text{SPS},A})$.
   (b) Compute $r'_{\text{SPS},B} = \mathcal{F}(K_{\text{SPS},B}\{(h^*, \ell^*, 0)\}, (h, \ell_{\text{INP}}, t)), (\text{SK}'_{\text{SPS},B}, \text{VK}'_{\text{SPS},B}, \text{VK}'_{\text{SPS-REJ},B}) = \mathsf{SPS.Setup}(1^\lambda; r'_{\text{SPS},B})$.
   (c) Set $m_{\text{OUT}} = (v_{\text{OUT}}, \text{ST}_{\text{OUT}}, w_{\text{OUT}}, \text{POS}_{\text{OUT}})$.
   If $[(h, \ell_{\text{INP}}, t) = (h^*, \ell^*, 1)] \wedge [m_{\text{IN}} = m_{\ell^*,0}]$, compute $\sigma_{\text{SPS,OUT}} = \mathsf{SPS.Sign}(\text{SK}'_{\text{SPS},A}, m_{\text{OUT}})$.
   Else if $[(h, \ell_{\text{INP}}, t) = (h^*, \ell^*, 1)] \wedge [m_{\text{IN}} \neq m_{\ell^*,0}]$, compute $\sigma_{\text{SPS,OUT}} = \mathsf{SPS.Sign}(\text{SK}'_{\text{SPS},B}, m_{\text{OUT}})$.
   Else, compute $\sigma_{\text{SPS,OUT}} = \mathsf{SPS.Sign}(\text{SK}'_{\text{SPS},\alpha}, m_{\text{OUT}})$.
7. If $t + 1 = 2^{\tau'}$, for some integer τ', set $\text{SEED}_{\text{OUT}} = \mathcal{F}(K_{\tau'}, (h, \ell_{\text{INP}}))$.
   Else, set $\text{SEED}_{\text{OUT}} = \epsilon$.
8. Output $(\text{POS}_{\text{OUT}}, \text{SYM}_{\text{OUT}}, \text{ST}_{\text{OUT}}, w_{\text{OUT}}, v_{\text{OUT}}, \sigma_{\text{SPS,OUT}}, \text{SEED}_{\text{OUT}})$.

Fig. 45 $\mathsf{Constrained\text{-}Key.Prog}^{(1,0,4)}_{\text{ABS}}$

Proof The proof of Claim C.35 resembles that of Claim C.25 with some suitable changes. The details are omitted. □

Claim C.36 *Assuming $\mathcal{IO}$ is a secure indistinguishability obfuscator for* $\mathsf{P/poly}$*, for any* PPT *adversary $\mathcal{A}$, for any security parameter λ,* $|\mathsf{Adv}^{(0,\nu-1,4\text{-}III)}_{\mathcal{A}}(\lambda) - \mathsf{Adv}^{(0,\nu-1,4\text{-}IV)}_{\mathcal{A}}(\lambda)| \leq \mathsf{negl}(\lambda)$ *for some negligible function* negl.

Proof Claim C.36 can be proven using an analogous logic as that used in the proof of Claim C.26. We omit the details here. □

Claim C.37 *Assuming* SPS *is a splittable signature scheme satisfying '*$\text{VK}_{\text{SPS-ONE}}$ *indistinguishability', for any* PPT *adversary $\mathcal{A}$, for any security parameter λ,* $|\mathsf{Adv}^{(0,\nu-1,4\text{-}IV)}_{\mathcal{A}}(\lambda) - \mathsf{Adv}^{(0,\nu-1,4\text{-}V)}_{\mathcal{A}}(\lambda)| \leq \mathsf{negl}(\lambda)$ *for some negligible function* negl.

Proof The proof of Claim C.37 is similar to that of Claim C.27 and, therefore, we do not provide the details here. □

Claim C.38 *Assuming* SPS *is a splittable signature scheme satisfying '*$\text{VK}_{\text{SPS-ABO}}$ *indistinguishability', for any* PPT *adversary* $\mathcal{A}$*, for any security parameter* λ*,* $|\mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,4\text{-}V)}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,4\text{-}VI)}(\lambda)| \leq \mathsf{negl}(\lambda)$ *for some negligible function* negl*.*

Proof The proof of Claim C.38 proceeds along a similar path to that of Claim C.28. We omit the details here. □

Claim C.39 *Assuming* $\mathcal{IO}$ *is a secure indistinguishability obfuscator for* P/poly*, for any* PPT *adversary* $\mathcal{A}$*, for any security parameter* λ*,* $|\mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,4\text{-}VI)}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,4\text{-}VII)}(\lambda)| \leq \mathsf{negl}(\lambda)$ *for some negligible function* negl*.*

Proof The proof of Claim C.39 employs the same type of logic as that applied in Claim C.29 and hence we do not provide the details in this case as well. □

Claim C.40 *Assuming* ACC *is a positional accumulator satisfying the 'indistinguishability of read setup' property, for any* PPT *adversary* $\mathcal{A}$*, for any security parameter* λ*,* $|\mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,4\text{-}VII)}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,4\text{-}VIII)}(\lambda)| \leq \mathsf{negl}(\lambda)$ *for some negligible function* negl*.*

Proof Suppose there exists a PPT adversary $\mathcal{A}$ for which $|\mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,4\text{-VII})}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,4\text{-VIII})}(\lambda)|$ is non-negligible. We construct a PPT adversary $\mathcal{B}$ that breaks the indistinguishability of read setup property of the positional accumulator ACC using $\mathcal{A}$ as a sub-routine. The description of $\mathcal{B}$ follows:

- $\mathcal{B}$ initializes $\mathcal{A}$ on input 1^λ and receives a challenge signing attribute string $x^* = x_0^* \ldots x_{\ell^*-1}^* \in \mathcal{U}_{\text{ABS}}$ with $|x^*| = \ell^*$ from $\mathcal{A}$.
- Upon receiving x^*, $\mathcal{B}$ proceeds as follows:
  1. $\mathcal{B}$ first generates $\text{HK} \xleftarrow{\$} \mathsf{SSB.Gen}(1^\lambda, n_{\text{SSB-BLK}} = 2^\lambda, i^* = 0)$ and computes $h^* = \mathcal{H}_{\text{HK}}(x^*)$.
  2. Then, $\mathcal{B}$ selects a PPRF key $K \xleftarrow{\$} \mathcal{F}.\mathsf{Setup}(1^\lambda)$.
  3. After that, $\mathcal{B}$ computes $r_{\text{SIG}}^* = \mathcal{F}(K, (h^*, \ell^*))$ followed by $(\widehat{\text{SK}}_{\text{SIG}}^*, \widehat{\text{VK}}_{\text{SIG}}^*) = \mathsf{SIG.Setup}(1^\lambda;\ r_{\text{SIG}}^*)$.
  4. $\mathcal{B}$ returns the public parameters $\text{PP}_{\text{ABS}} = (\text{HK}, \mathcal{IO}(\mathsf{Verify.Prog}_{\text{ABS}}[K]))$ to $\mathcal{A}$.
- For $\eta \in [\hat{q}_{\text{KEY}}]$, in response to the η^{th} signing key query of $\mathcal{A}$ corresponding to TM $M^{(\eta)} \in \mathbb{M}_\lambda$ with $M^{(\eta)}(x^*) = 0$, if $\eta \neq \nu$, then $\mathcal{B}$ proceeds exactly as in $\mathsf{Hyb}_{0,\nu-1,4\text{-VII}}$, while if $\eta = \nu$, then $\mathcal{B}$ proceeds as follows:
  1. $\mathcal{B}$ selects PPRF keys $K_1^{(\nu)}, \ldots, K_\lambda^{(\nu)}, K_{\text{SPS},A}^{(\nu)}, K_{\text{SPS},B}^{(\nu)}, K_{\text{SPS},E}^{(\nu)} \xleftarrow{\$} \mathcal{F}.\mathsf{Setup}(1^\lambda)$.
  2. Then, it creates the punctured PPRF keys $K_{\text{SPS},A}^{(\nu)}\{(h^*, \ell^*, 0)\} \xleftarrow{\$} \mathcal{F}.\mathsf{Puncture}(K_{\text{SPS},A}^{(\nu)}, (h^*, \ell^*, 0))$ and $K_{\text{SPS},B}^{(\nu)}\{(h^*, \ell^*, 0)\} \xleftarrow{\$} \mathcal{F}.\mathsf{Puncture}(K_{\text{SPS},B}^{(\nu)}, (h^*, \ell^*, 0))$.
  3. Next, $\mathcal{B}$ sends $n_{\text{ACC-BLK}} = 2^\lambda$, the sequence of symbol-index pairs $((x_0^*, 0), \ldots, (x_{\ell^*-1}^*, \ell^* - 1))$, and the index $i^* = 0$ to its ACC read setup indistinguishability challenger $\mathcal{C}$ and receives back $(\text{PP}_{\text{ACC}}, w_0, \text{STORE}_0)$, where either $(\text{PP}_{\text{ACC}}, w_0, \text{STORE}_0) \xleftarrow{\$} \mathsf{ACC.Setup}(1^\lambda, n_{\text{ACC-BLK}} = 2^\lambda)$ or $(\text{PP}_{\text{ACC}}, w_0, \text{STORE}_0) \xleftarrow{\$} \mathsf{ACC.Setup\text{-}Enforce\text{-}Read}(1^\lambda, n_{\text{ACC-BLK}} = 2^\lambda, ((x_0^*, 0), \ldots, (x_{\ell^*-1}^*, \ell^* - 1)), i^* = 0)$.

4. Next, it generates $(\mathrm{PP}_{\mathrm{ITR}}^{(\nu)}, v_0^{(\nu)}) \xleftarrow{\$} \mathsf{ITR.Setup}(1^\lambda, n_{\mathrm{ITR}} = 2^\lambda)$.
5. Then, it sets $m_{0,0}^{(\nu)} = (v_0^{(\nu)}, q_0^{(\nu)}, w_0, 0)$. For $j = 1, \ldots, \ell^*$, it iteratively computes the following:
   - $\mathrm{AUX}_j^{(\nu)} = \mathsf{ACC.Prep\text{-}Write}(\mathrm{PP}_{\mathrm{ACC}}, \mathrm{STORE}_{j-1}, j-1)$
   - $w_j = \mathsf{ACC.Update}(\mathrm{PP}_{\mathrm{ACC}}, w_{j-1}, x_{j-1}^*, j-1, \mathrm{AUX}_j^{(\nu)})$
   - $\mathrm{STORE}_j = \mathsf{ACC.Write\text{-}Store}(\mathrm{PP}_{\mathrm{ACC}}, \mathrm{STORE}_{j-1}, j-1, x_{j-1}^*)$
   - $v_j^{(\nu)} = \mathsf{ITR.Iterate}(\mathrm{PP}_{\mathrm{ITR}}^{(\nu)}, v_{j-1}^{(\nu)}, (q_0^{(\nu)}, w_{j-1}, 0))$

   It sets $m_{\ell^*,0}^{(\nu)} = (v_{\ell^*}^{(\nu)}, q_0^{(\nu)}, w_{\ell^*}, 0)$.
6. After that, it generates $(\mathrm{SK}_{\mathrm{SPS},C}^{(\nu,0)}, \mathrm{VK}_{\mathrm{SPS},C}^{(\nu,0)}, \mathrm{VK}_{\mathrm{SPS\text{-}REJ},C}^{(\nu,0)})$, $(\mathrm{SK}_{\mathrm{SPS},D}^{(\nu,0)}, \mathrm{VK}_{\mathrm{SPS},D}^{(\nu,0)}, \mathrm{VK}_{\mathrm{SPS\text{-}REJ},D}^{(\nu,0)}) \xleftarrow{\$} \mathsf{SPS.Setup}(1^\lambda)$ and forms $(\sigma_{\mathrm{SPS\text{-}ONE},m_{\ell^*,0}^{(\nu)},C}^{(\nu,0)}, \mathrm{VK}_{\mathrm{SPS\text{-}ONE},C}^{(\nu,0)}, \mathrm{SK}_{\mathrm{SPS\text{-}ABO},C}^{(\nu,0)}, \mathrm{VK}_{\mathrm{SPS\text{-}ABO},C}^{(\nu,0)}) \xleftarrow{\$} \mathsf{SPS.Split}(\mathrm{SK}_{\mathrm{SPS},C}^{(\nu,0)}, m_{\ell^*,0}^{(\nu)})$, $(\sigma_{\mathrm{SPS\text{-}ONE},m_{\ell^*,0}^{(\nu)},D}^{(\nu,0)}, \mathrm{VK}_{\mathrm{SPS\text{-}ONE},D}^{(\nu,0)}, \mathrm{SK}_{\mathrm{SPS\text{-}ABO},D}^{(\nu,0)}, \mathrm{VK}_{\mathrm{SPS\text{-}ABO},D}^{(\nu,0)}) \xleftarrow{\$} \mathsf{SPS.Split}(\mathrm{SK}_{\mathrm{SPS},D}^{(\nu,0)}, m_{\ell^*,0}^{(\nu)})$.
7. It gives $\mathcal{A}$ the signing key $\mathrm{SK}_{\mathrm{ABS}}\{M^{(\nu)}\} =$

$$\left(\begin{array}{l}
\mathrm{HK}, \mathrm{PP}_{\mathrm{ACC}}, w_0, \mathrm{STORE}_0, \mathrm{PP}_{\mathrm{ITR}}^{(\nu)}, v_0^{(\nu)}, \\
\mathcal{IO}(\mathsf{Init\text{-}SPS.Prog}[q_0^{(\nu)}, w_0, v_0^{(\nu)}, K_{\mathrm{SPS},E}^{(\nu)}]), \\
\mathcal{IO}(\mathsf{Accumulate.Prog}[n_{\mathrm{SSB\text{-}BLK}} = 2^\lambda, \mathrm{HK}, \mathrm{PP}_{\mathrm{ACC}}, \mathrm{PP}_{\mathrm{ITR}}^{(\nu)}, K_{\mathrm{SPS},E}^{(\nu)}]), \\
\mathcal{IO}(\mathsf{Change\text{-}SPS.Prog}^{(4,2)}[K_{\mathrm{SPS},A}^{(\nu)}\{(h^*, \ell^*, 0)\}, K_{\mathrm{SPS},B}^{(\nu)}\{(h^*, \ell^*, 0)\}, \\
\qquad K_{\mathrm{SPS},E}^{(\nu)}, \sigma_{\mathrm{SPS\text{-}ONE},m_{\ell^*,0}^{(\nu)},C}^{(\nu,0)}, \mathrm{SK}_{\mathrm{SPS\text{-}ABO},D}^{(\nu,0)}, m_{\ell^*,0}^{(\nu)}, h^*, \ell^*]), \\
\mathcal{IO}(\mathsf{Constrained\text{-}Key.Prog}_{\mathrm{ABS}}^{(1,0,2)}[M^{(\nu)}, T = 2^\lambda, t^{*(\nu)}, \mathrm{PP}_{\mathrm{ACC}}, \mathrm{PP}_{\mathrm{ITR}}^{(\nu)}, K, \\
\qquad K_1^{(\nu)}, \ldots, K_\lambda^{(\nu)}, K_{\mathrm{SPS},A}^{(\nu)}\{(h^*, \ell^*, 0)\}, K_{\mathrm{SPS},B}^{(\nu)}\{(h^*, \ell^*, 0)\}, \\
\qquad \mathrm{VK}_{\mathrm{SPS\text{-}ONE},C}^{(\nu,0)}, \mathrm{VK}_{\mathrm{SPS\text{-}ABO},D}^{(\nu,0)}, m_{\ell^*,0}^{(\nu)}, h^*, \ell^*])
\end{array}\right).$$

- For $\theta \in [\hat{q}_{\mathrm{SIGN}}]$, in reply to the θ^{th} signature query of $\mathcal{A}$ on message $\mathsf{msg}^{(\theta)} \in \mathcal{M}_{\mathrm{ABS}}$ under attribute string x^*, $\mathcal{B}$ computes $\sigma_{\mathrm{SIG}}^{(\theta)} \xleftarrow{\$} \mathsf{SIG.Sign}(\widehat{\mathrm{SK}}_{\mathrm{SIG}}^*, \mathsf{msg}^{(\theta)})$ and provides $\mathcal{A}$ with $\sigma_{\mathrm{ABS}}^{(\theta)} = (\widehat{\mathrm{VK}}_{\mathrm{SIG}}^*, \sigma_{\mathrm{SIG}}^{(\theta)})$.
- Finally, $\mathcal{A}$ output a signature σ_{ABS}^* on some message msg^* under attribute string x^*. $\mathcal{B}$ outputs $\hat{b}' = 1$ as its guess bit in its ACC read setup indistinguishability experiment if $\mathcal{A}$ wins, i.e., if $\mathsf{ABS.Verify}(\mathrm{PP}_{\mathrm{ABS}}, x^*, \mathsf{msg}^*, \sigma_{\mathrm{ABS}}^*) = 1$ and $\mathsf{msg}^* \neq \mathsf{msg}^{(\theta)}$ for any $\theta \in [\hat{q}_{\mathrm{SIGN}}]$. Otherwise, $\mathcal{B}$ outputs $\hat{b}' = 0$ in its ACC read setup indistinguishability experiment.

Note that if $(\mathrm{PP}_{\mathrm{ACC}}, w_0, \mathrm{STORE}_0) \xleftarrow{\$} \mathsf{ACC.Setup}(1^\lambda, n_{\mathrm{ACC\text{-}BLK}} = 2^\lambda)$, then $\mathcal{B}$ perfectly simulates $\mathsf{Hyb}_{0,\nu-1,4\text{-VII}}$. On the other hand, if $(\mathrm{PP}_{\mathrm{ACC}}, w_0, \mathrm{STORE}_0) \xleftarrow{\$} \mathsf{ACC.Setup\text{-}Enforce\text{-}Read}(1^\lambda, n_{\mathrm{ACC\text{-}BLK}} = 2^\lambda, ((x_0^*, 0), \ldots, (x_{\ell^*-1}^*, \ell^* - 1)), i^* = 0)$, then $\mathcal{B}$ perfectly simulates $\mathsf{Hyb}_{0,\nu-1,4\text{-VIII}}$. This completes the proof of Claim C.40. □

Claim C.41 *Assuming $\mathcal{IO}$ is a secure indistinguishability obfuscator for* $\mathsf{P/poly}$ *and* ACC *is a positional accumulator satisfying the 'read enforcing' property, for any* PPT *adversary $\mathcal{A}$, for any security parameter λ, $|\mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,4\text{-}VIII)}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,4\text{-}IX)}(\lambda)| \leq \mathsf{negl}(\lambda)$ for some negligible function* negl.

Proof The only difference between $\mathsf{Hyb}_{0,\nu-1,4\text{-VIII}}$ and $\mathsf{Hyb}_{0,\nu-1,4\text{-IX}}$ is the following: In $\mathsf{Hyb}_{0,\nu-1,4\text{-VIII}}$, $\mathcal{B}$ includes the program $\mathcal{IO}(P_0)$ within the ν^{th} signing key provided to $\mathcal{A}$, while in $\mathsf{Hyb}_{0,\nu-1,4\text{-IX}}$ it includes the program $\mathcal{IO}(P_1)$ instead, where

- $P_0 = \mathsf{Constrained\text{-}Key.Prog}^{(1,0,2)}_{\text{ABS}}[M^{(\nu)}, T = 2^\lambda, t^{*(\nu)}, \text{PP}^{(\nu)}_{\text{ACC}}, \text{PP}^{(\nu)}_{\text{ITR}}, K, K^{(\nu)}_1, \ldots, K^{(\nu)}_\lambda, K^{(\nu)}_{\text{SPS},A}\{(h^*, \ell^*, 0)\}, K^{(\nu)}_{\text{SPS},B}\{(h^*, \ell^*, 0)\}, \text{VK}^{(\nu,0)}_{\text{SPS-ONE},C}, \text{VK}^{(\nu,0)}_{\text{SPS-ABO},D}, m^{(\nu)}_{\ell^*,0}, h^*, \ell^*]$ (Fig. **42**),
- $P_1 = \mathsf{Constrained\text{-}Key.Prog}^{(1,0,3)}_{\text{ABS}}[M^{(\nu)}, T = 2^\lambda, t^{*(\nu)}, \text{PP}^{(\nu)}_{\text{ACC}}, \text{PP}^{(\nu)}_{\text{ITR}}, K, K^{(\nu)}_1, \ldots, K^{(\nu)}_\lambda, K^{(\nu)}_{\text{SPS},A}\{(h^*, \ell^*, 0)\}, K^{(\nu)}_{\text{SPS},B}\{(h^*, \ell^*, 0)\}, \text{VK}^{(\nu,0)}_{\text{SPS-ONE},C}, \text{VK}^{(\nu,0)}_{\text{SPS-ABO},D}, m^{(\nu)}_{\ell^*,0}, h^*, \ell^*]$ (Fig. **43**).

We will argue that the programs P_0 and P_1 are functionally equivalent, so that, by the security of $\mathcal{IO}$ Claim C.41 follows. Clearly, the only inputs for which the outputs of the two programs might differ are those corresponding to $(h, \ell_{\text{INP}}, t) = (h^*, \ell^*, 1)$. For inputs corresponding to $(h^*, \ell^*, 1)$, P_1 is programmed to output $\perp$ in case $\text{ST}_{\text{OUT}} = q_{\text{AC}}$ but $\alpha =$'A', whereas, P_0 has no such condition in its programming. Now, observe that for inputs corresponding to $(h^*, \ell^*, 1)$, both the programs will assign the value 'A' to α if and only if $\mathsf{SPS.Verify}(\text{VK}^{(\nu,0)}_{\text{SPS-ONE},C}, m_{\text{IN}}, \sigma_{\text{SPS,IN}}) = 1$, where $\text{VK}^{(\nu,0)}_{\text{SPS-ONE},C}$ is generated by running $\mathsf{SPS.Split}(\text{SK}^{(\nu,0)}_{\text{SPS},C}, m^{(\nu)}_{\ell^*,0})$. Hence, by the correctness [Properties (i), (iii) and (v)] of the splittable signature scheme SPS, described in Section 2.3.5, it is immediate that for inputs corresponding to $(h^*, \ell^*, 1)$, both programs will set $\alpha =$'A' only if $m_{\text{IN}} = m^{(\nu)}_{\ell^*,0}$. Now, $m_{\text{IN}} = m^{(\nu)}_{\ell^*,0}$ means $\text{ST}_{\text{IN}} = q^{(\nu)}_0, w_{\text{IN}} = w^{(\nu)}_{\ell^*}$, and $\text{POS}_{\text{IN}} = 0$. Further, recall that in both the hybrid experiments $\mathsf{Hyb}_{0,\nu-1,4\text{-VIII}}$ and $\mathsf{Hyb}_{0,\nu-1,4\text{-IX}}$, $(\text{PP}^{(\nu)}_{\text{ACC}}, w^{(\nu)}_0, \text{STORE}^{(\nu)}_0) \xleftarrow{\$} \mathsf{ACC.Setup\text{-}Enforce\text{-}Read}(1^\lambda, n_{\text{ACC-BLK}} = 2^\lambda, ((x^*_0, 0), \ldots, (x^*_{\ell^*-1}, \ell^* - 1)), i^* = 0)$. Therefore, by the read enforcing property of ACC it follows that if $w_{\text{IN}} = w^{(\nu)}_{\ell^*}$ and $\text{POS}_{\text{IN}} = 0$, then $\mathsf{ACC.Verify\text{-}Read}(\text{PP}^{(\nu)}_{\text{ACC}}, w_{\text{IN}}, \text{SYM}_{\text{IN}}, \text{POS}_{\text{IN}}, \pi_{\text{ACC}}) = 1$ implies $\text{SYM}_{\text{IN}} = x^*_0$. Hence, for both the programs, for inputs corresponding to $(h^*, \ell^*, 1)$, $\alpha =$'A' implies $\text{ST}_{\text{IN}} = q^{(\nu)}_0$ and $\text{SYM}_{\text{IN}} = x^*_0$, which in turn implies $\text{ST}_{\text{OUT}} \neq q_{\text{AC}}$. Hence, the two programs have identical outputs for inputs corresponding to $(h^*, \ell^*, 1)$ as well. Thus, the two programs are functionally equivalent. □

Claim C.42 *Assuming* ACC *is a positional accumulator satisfying the 'indistinguishability of read setup' property, for any* PPT *adversary* $\mathcal{A}$*, for any security parameter* λ*,* $|\mathsf{Adv}^{(0,\nu-1,4\text{-}IX)}_{\mathcal{A}}(\lambda) - \mathsf{Adv}^{(0,\nu-1,4\text{-}X)}_{\mathcal{A}}(\lambda)| \leq \mathsf{negl}(\lambda)$ *for some negligible function* negl.

Proof The proof of Claim C.42 is similar to that of Claim C.40 with some appropriate modifications that are easy to figure out. □

Claim C.43 *Assuming* SPS *is a splittable signature scheme satisfying 'splitting indistinguishability', for any* PPT *adversary* $\mathcal{A}$*, for any security parameter* λ*,* $|\mathsf{Adv}^{(0,\nu-1,4\text{-}X)}_{\mathcal{A}}(\lambda) - \mathsf{Adv}^{(0,\nu-1,4\text{-}XI)}_{\mathcal{A}}(\lambda)| \leq \mathsf{negl}(\lambda)$ *for some negligible function* negl.

Proof The proof of Claim C.43 proceeds along a similar path as that of the proof of Claim C.30. We omit the details here. □

Claim C.44 *Assuming $\mathcal{IO}$ is a secure indistinguishability obfuscator for* $\mathsf{P/poly}$*, for any* PPT *adversary* $\mathcal{A}$*, for any security parameter* λ*,* $|\mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,4\text{-}XI)}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,4\text{-}XII)}(\lambda)| \leq \mathsf{negl}(\lambda)$ *for some negligible function* negl*.*

Proof The proof of Claim C.44 employs a similar type of logic as that utilized in the proof of Claim C.31. We omit the details in this case as well. □

Claim C.45 *Assuming $\mathcal{F}$ is a secure puncturable pseudorandom function, for any* PPT *adversary* $\mathcal{A}$*, for any security parameter* λ*,* $|\mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,4\text{-}XII)}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,4\text{-}XIII)}(\lambda)| \leq \mathsf{negl}(\lambda)$ *for some negligible function* negl*.*

Proof The proof of Claim C.45 takes an analogous path as that taken by the proof of Claim C.25. The details are omitted. □

Claim C.46 *Assuming $\mathcal{IO}$ is a secure indistinguishability obfuscator for* $\mathsf{P/poly}$ *and $\mathcal{F}$ satisfies the correctness under puncturing property, for any* PPT *adversary* $\mathcal{A}$*, for any security parameter* λ*,* $|\mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,4\text{-}XIII)}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,4\text{-}XIV)}(\lambda)| \leq \mathsf{negl}(\lambda)$ *for some negligible function* negl*.*

Proof The proof of Claim C.46 applies the same kind of logic as that employed in the proof of Claim C.24. The details are again omitted. □

□

Lemma C.9 *Assuming $\mathcal{IO}$ is a secure indistinguishability obfuscator for* $\mathsf{P/poly}$*, ACC is a secure positional accumulator, and ITR is a secure cryptographic iterator, for any* PPT *adversary* $\mathcal{A}$*, for any security parameter* λ*,* $|\mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,4,(\gamma-1)')}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,4,\gamma)}(\lambda)| \leq \mathsf{negl}(\lambda)$ *for some negligible function* negl*.*

Proof The proof of Lemma C.9 follows an analogous path to that of Lemma B.4 of [7] with certain appropriate modifications that are easy to identify. □

Lemma C.10 *Assuming $\mathcal{IO}$ is a secure indistinguishability obfuscator for* $\mathsf{P/poly}$*, $\mathcal{F}$ is a secure puncturable pseudorandom function, ACC is a positional accumulator possessing the 'indistinguishability of read setup' as well as 'read enforcing' properties, and* SPS *is a splittable signature scheme satisfying '*$\mathrm{VK}_{\text{SPS-ONE}}$ *indistinguishability', '*$\mathrm{VK}_{\text{SPS-ABO}}$ *indistinguishability', as well as 'splitting indistinguishability' properties, for any* PPT *adversary* $\mathcal{A}$*, for any security parameter* λ*,* $|\mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,4,\gamma)}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,4,\gamma')}(\lambda)| \leq \mathsf{negl}(\lambda)$ *for some negligible function* negl*.*

Proof The proof of Lemma C.10 is similar to that of Lemma B.3 of [7] with some appropriate readily identifiable changes. □

Lemma C.11 *Assuming $\mathcal{IO}$ is a secure indistinguishability obfuscator for* $\mathsf{P/poly}$ *and ACC is a positional accumulator having 'indistinguishability of read setup' and 'read enforcing' properties, for any* PPT *adversary* $\mathcal{A}$*, for any security parameter* λ*,* $|\mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,4,(t^{*(\nu)}-1)')}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,5)}(\lambda)| \leq \mathsf{negl}(\lambda)$ *for some negligible function* negl*.*

Proof The proof of Lemma C.11 is similar to that of Lemma B.5 of [7] with some appropriate changes that are easy to determine. □

Lemma C.12 *Assuming $\mathcal{IO}$ is a secure indistinguishability obfuscator for* P/poly*, $\mathcal{F}$ is a secure puncturable pseudorandom function, SPS is a splittable signature scheme possessing the '$\mathrm{VK}_{\mathrm{SPS\text{-}REJ}}$ indistinguishability' property, and PRG is a secure injective pseudorandom generator, for any* PPT *adversary $\mathcal{A}$, for any security parameter λ, $|\mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,5)}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{(0,\nu-1,6)}(\lambda)| \leq \mathsf{negl}(\lambda)$ for some negligible function* negl.

Proof The proof of Lemma C.12 is similar to that of Lemma B.6 of [7]. □