A New Constant-size Accountable Ring Signature Scheme Without Random Oracles

Sudhakar Kumawat[1] and Souradyuti Paul[2]

[1] Indian Institute of Technology Gandhinagar
[2] Indian Institute of Technology Bhilai
{sudhakar.bm07,souradyuti.paul}@gmail.com

Abstract. Accountable ring signature (ARS), introduced by Xu and Yung (CARDIS 2004), combines many useful properties of ring and group signatures. In particular, the signer in an ARS scheme has the flexibility of choosing an ad hoc group of users, and signing on their behalf (like a ring signature). Furthermore, the signer can designate an opener who may later reveal his identity, if required (like a group signature). In 2015, Bootle *et al.* (ESORICS 2015) formalized the notion and gave an efficient construction for ARS with signature-size logarithmic in the size of the ring. Their scheme is proven to be secure in the random oracle model. Recently, Russell *et al.* (ESORICS 2016) gave a construction with constant signature-size that is secure in the standard model. Their scheme is based on q-type assumptions (q-SDH).
In this paper, we give a new construction for ARS having the following properties: signature is constant-sized, secure in the standard model, and based on indistinguishability obfuscation ($i\mathcal{O}$) and one-way functions. To the best of our knowledge, this is the first $i\mathcal{O}$-based ARS scheme. Independent of this, our work can be viewed as a new application of *puncturable programming* and *hidden sparse trigger* techniques introduced by Sahai and Waters (STOC 2014) to design $i\mathcal{O}$-based deniable encryption.

Keywords: Accountable ring signatures, indistinguishability obfuscation, puncturable PRFs.

1 Introduction

The notion of group signature, introduced by Chaum and Van Heyst [9], allows a user to sign anonymously on behalf of a group of users without revealing his identity. Here, the membership of the group is controlled exclusively by the group manager, that is, he can include as well as expel users in the group. Moreover, the group manager can revoke the anonymity of the signer using a secret tracing key. Unlike a group signature, a ring signature, introduced by Rivest *et al.* [18], allows a signer to choose an ad hoc group of users (called the ring) and to sign on their behalf. Also, there is no group manager in a ring signature that controls the group and traces the signer. The notion of accountable ring signature (ARS) [20] was borne out of the above two notions. It provides the flexibility of choosing a group of users, and ensures accountability by allowing an opener – designated by the signer – to later reveal the signer's identity. An ARS scheme must be unforgeable, anonymous, traceable and traceably sound. Applications of accountable ring signature include anonymous e-cash schemes [**?**], anonymous forums and auction systems [4].

Related Work In 2004, Xu and Yung [20] introduced the notion of accountable ring signature (ARS). Their construction relies on the existence of a trapdoor permutation and uses a threshold decryption scheme to reveal the signer's identity. Their main trick lies in the use of tamper-resistant smart cards to retain some footprint of the identity of the signer in the signature. The size of the signature in their scheme grows linearly with that of the ring. In 2015, Bootle *et al.*[4] formalized the notion of ARS, and gave an efficient construction secure in the random oracle model by combining Camenisch's group signature scheme [8] with a generalized version of Groth and Kohlweiss's one-out-of-many proofs of knowledge [15]. Their construction relies on the hardness of Decision Diffie-Hellman (DDH) problem. The signature-size of their scheme grows logarithmically with the ring-size. Recently, Lai *et al.* [17] gave the first constant-size ARS scheme in standard model by combining the ring signature scheme of [5], with the structure-preserving encryption scheme of [7]. Their construction relies on q-type assumptions, in particular the q-SDH assumption.

et al.

Paper	**Signature size**	**Security model**	**Unforgeability**	**Anonymity**	**Hardness assumption**
[20]	$O(\|R\|)$	RO	Fully adaptive	Fully adaptive	Trapdoor permutations
[4]	$O(\log \|R\|)$	RO	Fully adaptive	Fully adaptive	DDH
[17]	$O(1)$	Standard	q-Adaptive	q-Adaptive	q-type
This paper	$O(1)$	Standard	Selective	Selective	$i\mathcal{O}$+OWF

Table 1: Comparison between several *accountable ring signature* schemes. The symbol R denotes the ring.

Our Contribution We propose a new constant-size accountable ring signature scheme. Our scheme relies on the security of indistinguishability obfuscation ($i\mathcal{O}$) and one-way functions. Our approach is inspired by the *puncturable programming* and *hidden sparse trigger* techniques introduced, and used by Sahai and Waters [19] to construct deniable encryption schemes. Two important features of our scheme are as follows: Unlike the schemes in [20] and [4], (1) our scheme is proven secure in the standard model, and (2) the signature-size of our scheme is constant, and does not increase with the ring-size. While the scheme in [17] achieves stronger security guarantees with respect to unforgeability and anonymity, it does not achieve full adaptiveness (i.e. q-adaptive). This is because, its security proof relies on q-type assumption (q-SDH) that restricts the number of queries made to the signing oracle to a fixed parameter q (determined in the setup phase). However, this does not work in the real world as an adversary can always get more than q valid signatures. Note that setting q to a very large number makes the assumption stronger and, also, drastically degrades the practical efficiency of the scheme. In contrast, our scheme is secure against arbitrary number (polynomial) of signing queries. One major disadvantage of our scheme is that it is selective secure. Thus, constructing a fully unforgeable and anonymous, and constant-size ARS scheme in the standard model still remains an interesting open problem. In Table. 1, we compare our scheme with previous ones.

In this paper, we stress that, due to the low efficiency of the existing $i\mathcal{O}$ candidates, our focus is mainly on the existence of a constant signature-size ARS scheme (secure without random oracles) based on $i\mathcal{O}$, but not on their practicability. However, note that, with the recent publication of results on efficient implementation of $i\mathcal{O}$-based cryptographic primitives [2], we hope that our scheme may be close to being practical.

Technical Overview First, a trusted authority runs the setup algorithm *once* that gives two obfuscated programs Sign and Verify. Two random keys K_1 and K_2 are hardwired into these programs, and are known only to the trusted authority. It is important for any ARS scheme that the footprint of the signer be retained in the signature so that the signer's identity could be revealed by a designated party, if required. To achieve this, we use the *hidden sparse trigger* technique introduced by Sahai and Waters [19], and briefly describe it in the Sign and Verify programs below.

– Sign: This program takes as input a message m, a signing-verification key pair (sk, vk), a ring R (a set of verification keys arranged in a specified order), a public key ek of an opener and some randomness r. Then, it checks if $vk = f(sk)$ and $vk \in R$. If so, it sets $\alpha = F_1(K_1, (m \| vk \| \mathsf{Hash}(R) \| ek \| \mathsf{PRG}(r)))$ and $\beta = F_2(K_2, \alpha) \oplus (m \| vk \| \mathsf{Hash}(R) \| ek \| \mathsf{PRG}(r))$; and outputs $\sigma = (\alpha, \beta)$ as a signature of (m, sk, R, ek, r). The security relies on the following assumptions: (1) $F_1(K_1, \cdot)$ is *injective pucturable PRF*; $F_2(K_2, \cdot)$ is pucturable PRF; $\mathsf{Hash}(\cdot)$ is collision-resistant; and $f(\cdot)$ is one-way. (These are discussed in detail in Sect. 5).

– Verify: This program takes as input a message m, signature $\sigma = (\alpha, \beta)$, a ring R and the encryption key ek of an opener. Then it checks if the claimed signature σ is a valid encoding of these inputs under the encoding scheme as described in the Sign program above. More concretely, it checks if $F_2(K_2, \alpha) \oplus \beta = (m' \| vk' \| h' \| ek' \| r')$, $m = m'$, $\mathsf{Hash}(R) = h'$, $ek = ek'$ and $f_1(\alpha) = f_1(F_1(K_1, (m' \| vk' \| h' \| ek' \| r')))$. If so, it outputs verification key vk' of the signer encrypted under the public key ek'. Otherwise it outputs $\perp$. Here $f_2(\cdot)$ is a one-way function.

In addition, we require two more functionalities, namely Open and Judge, to reveal and prove the signer's identity. Open works as follows: it invokes Verify to obtain the encrypted verification key. Next, it decrypts it. Finally, it produces a

NIZK proof of correct decryption. Judge verifies if the proof returned by Open is correct.

Notation We use $x \leftarrow S$ to denote that x is sampled uniformly from the set S; $y := \mathcal{A}(x)$ denotes that $\mathcal{A}$ is a deterministic algorithm whose output is assigned to y; when $\mathcal{A}$ is randomized, the process is denoted by $y \leftarrow \mathcal{A}(x)$ (or $y = A(x; r)$). "PPT" stands for probabilistic polynomial time.

Paper Organization The rest of the paper is organized as follows. In Sect. 2, we recall the definition and security model of ARS scheme from [**?**]. In Sect. 3, we recall the definitions of various cryptographic primitives used in our construction. In Sect. 4, we present our $i\mathcal{O}$-based ARS scheme. In Sect. 4, we give a proof of security of our construction. Finally, we conclude in Sect. 6 giving some open problems in the area.

2 Accountable Ring Signature Scheme

2.1 Syntax

Assume that each user in the system is uniquely identified by an index $i \in [n], n \in \mathbb{N}$. In addition, imagine that the PKI maintains a public registry reg of the registered users.

Definition 1 (Accountable Ring Signature (ARS)[4]). *An ARS scheme* ARS=(ARS.UKGen,ARS.OKGen,ARS.Sign,ARS.Verify,ARS.Open,ARS.Judge) *over a* PPT *setup* ARS.Setup *is a 6-tuple of algorithms.*

- *ARS.Setup*(1^λ) : *On input the security parameter* 1^λ*,* ARS.Setup *outputs a list of public parameters params consisting of a class* $\{\mathcal{SK}, \mathcal{VK}, \mathcal{EK}, \mathcal{DK}\}$ *– denoting key spaces for signing, verification, encryption and decryption – along with poly-time algorithms for sampling and deciding memberships.*
- ARS.UKGen$(params)$: *On input params, the* PPT *algorithm* ARS.UKGen *outputs a signing key sk and corresponding verification key vk for a user.*
- ARS.OKGen$(params)$: *On input params, the* PPT *algorithm* ARS.OKGen *outputs an encryption/ decryption key pair* (ek, dk) *for an opener.*
- ARS.Sign$(params, m, sk, R, ek)$: *On input params, message m, signer's secret key sk, a ring R – which is a set of verification keys – and an opener's public key ek, the* PPT *algorithm* ARS.Sign *outputs the ring signature* σ*.*
- ARS.Verify$(params, m, \sigma, R, ek)$: *On input params, message m, signature* σ*, a ring R and a public key ek,* ARS.Verify *outputs 1/0.*
- ARS.Open$(params, m, \sigma, R, dk)$: *On input params, message m, signature* σ*, a ring R and the opener's secret key dk,* ARS.Open *outputs the verification key vk of signer and a proof* ϕ *that the owner of vk generated* σ*.*
- ARS.Judge$(params, m, \sigma, R, ek, vk, \phi)$: *On input params, message m, signature* σ*, a ring R, the opener's public key ek, the verification key vk and the proof* ϕ*,* ARS.Judge *outputs 1 if the proof is correct and 0 otherwise.*

Correctness An ARS scheme is correct if for any PPT adversary $\mathcal{A}$:

$$\Pr\left[\begin{array}{c} params \leftarrow \mathsf{ARS.Setup}(1^\lambda);\ (vk, sk) \leftarrow \mathsf{ARS.UKGen}(params); \\ (m, R, ek) \leftarrow \mathcal{A}(params, sk);\ \sigma \leftarrow \mathsf{ARS.Sign}(m, sk, R, ek) \\ :\ vk \in R \wedge \mathsf{ARS.Verify}(m, \sigma, R, ek) = 1 \end{array}\right] \approx 1$$

2.2 Security Model

Unforgeability Unforgeability requires that an adversary cannot falsely accuse an honest user of creating a ring signature even if some of the members of the ring are corrupt and that the adversary controls the opener. More concretely, consider the following game between a PPT adversary $\mathcal{A}$ and a challenger $\mathcal{C}$.

1. **Setup Phase:** $\mathcal{C}$ runs the algorithms $\mathsf{ARS.Setup}$ to generate public parameters $params$. Then, it chooses a set $S \subset [n]$ of signers, runs $\mathsf{ARS.UKGen}$ to generate key-pairs $\{(sk_i, vk_i)\}_{i \in S}$ and registers them with the PKI, and sends $params, \{vk_i\}_{i \in S}$ to $\mathcal{A}$.
2. **Query Phase:** $\mathcal{A}$ can make the following three types of queries to $\mathcal{C}$. $\mathcal{C}$ answers these queries via oracles $\mathcal{O}_{Reg}$, $\mathcal{O}_{Cor}$ and $\mathcal{O}_{Sig}$.
   - **Registration query:** $\mathcal{A}$ runs the algorithm $\mathsf{ARS.UKGen}$ to generate a signing-verification key pair $(sk_i, vk_i), i \notin [n] \setminus S$ and interacts with the oracle $\mathcal{O}_{Reg}$ to register vk_i with the PKI. Let $\mathcal{Q}_{Reg}$ be the set of verification keys registered by $\mathcal{A}$.
   - **Corruption query:** $\mathcal{A}$ queries a verification key $vk_i, i \in S$ to the oracle $\mathcal{O}_{Cor}$. The oracle returns the corresponding signing key sk_i. Let $\mathcal{Q}_{Cor}$ be the set of verification keys vk_i for which the corresponding signing keys has been revealed.
   - **Signing query:** $\mathcal{A}$ queries (m, vk_i, R, ek, r) to the oracle $\mathcal{O}_{Sign}$. The oracle returns a signature $\sigma_i = \mathsf{Sign}(m, vk_i, R, ek, r)$ if $vk_i \notin \mathcal{Q}_{Reg} \cup \mathcal{Q}_{Cor}$. Let $\mathcal{Q}_{Sign}$ be the set of queries and their responses $(m, vk, R, ek, r, \sigma)$.
3. **Forgery Phase:** $\mathcal{A}$ outputs a signature σ^* w.r.t some $(m^*, vk^*, R^*, ek^*, r^*)$.

$\mathcal{A}$ wins the above game if $\mathsf{ARS.Verify}$ outputs 1 on input $(m^*, \sigma^*, R^*, ek^*)$.

Definition 2. *We say that the ARS scheme is unforgeable, if for any* PPT *adversary $\mathcal{A}$, its advantage in the game above is negligible i.e.* $\boldsymbol{Adv}_{\mathcal{A}}^{Unforge} = \Pr[\mathcal{A}\,\text{wins}] \leq \mathsf{negl}(\lambda)$.

Remark 1. In our scheme we consider a selective variant of the above definition where adversary $\mathcal{A}$ is required to commit to a forgery input $(m^*, vk^*, R^*, ek^*, r^*)$ in the setup phase. Then $\mathcal{A}$ cannot make signing query $(m^*, vk^*, R^*, ek^*, r^*)$ to the oracle $\mathcal{O}_{Sign}$.

Anonymity Anonymity requires that the signature keeps the identity of signer (who is a ring member) anonymous unless the opener explicitly want to open the signature and reveal the signer's identity. Our definition also captures unlinkability (i.e. anonymity breaks if an adversary can link signatures from same signer) and anonymity against full key exposure attacks (i.e. the signatures remain anonymous even if the secret signing keys were revealed). More concretely, consider the following game between a PPT adversary $\mathcal{A}$ and a challenger $\mathcal{C}$.

1. **Setup Phase:** $\mathcal{C}$ runs the algorithm $\mathsf{ARS.Setup}$ to generate public parameters *params*. Then, $\mathcal{C}$ runs the algorithm $\mathsf{ARS.KGen}$ to generate signing-verification key pairs $\{(sk_i, vk_i)\}_{i\in[n]}$ for the users and registers them with the PKI. In addition, it also generates a private-public key pair (dk^*, ek^*) of an opener. Then, it sends $params, ek^*, \{(sk_i, vk_i)\}_{i\in[n]}$ to $\mathcal{A}$.
2. **Challenge Phase:** $\mathcal{A}$ submits a message m^*, a ring $R^* \subseteq \{vk_i\}_{i\in[n]}$ and two secret signing keys $sk^*_{i_0}, sk^*_{i_1} \in \{sk_i\}_{i\in[n]}, i_0 \neq i_1$, such that $vk^*_{i_0}, vk^*_{i_1} \in R^*$. Next, $\mathcal{C}$ chooses $b \leftarrow \{0,1\}$ and produces an accountable ring signature $\sigma^*_{i_b} = \mathsf{ARS.Sign}(m^*, sk^*_{i_b}, R^*, ek^*; r)$ and returns $\sigma^*_{i_b}$ to $\mathcal{A}$.
3. **Guess Phase:** $\mathcal{A}$ guesses b and outputs $b' \in \{0,1\}$.

Definition 3. *We say that the ARS scheme is anonymous, if for any* PPT *adversary $\mathcal{A}$, its advantage in the game above is negligible i.e.* $\boldsymbol{Adv}_{\mathcal{A}}^{Anony} = |\Pr[b = b'] - \frac{1}{2}| \leq \mathsf{negl}(\lambda)$.

Remark 2. In our scheme we consider a selective variant of the above definition where adversary $\mathcal{A}$ is required to commit to a target input $(m^*, sk^*_{i_0}, sk^*_{i_1}, R^*, ek^*)$ in the setup phase.

Traceability Traceability requires that the specified opener is always able to identify the signer in the ring and produce a valid proof that the signer actually produced that particular signature; i.e. for any PPT adversary $\mathcal{A}$:

$$\Pr\left[\begin{array}{c} params \leftarrow \mathsf{ARS.Setup}(1^\lambda);\ (m,\sigma,R,ek,dk) \leftarrow \mathcal{A}(params); \\ (vk,\phi) \leftarrow \mathsf{ARS.Open}(m,\sigma,R,dk) \\ : \mathsf{ARS.Verify}(m,\sigma,R,ek) = 1 \wedge \mathsf{ARS.Judge}(m,\sigma,R,ek,vk,\phi) = 0 \end{array}\right] \leq \mathsf{negl}(\lambda)$$

Tracing Soundness Tracing soundness requires that a signature can be traced to only one user; even when all users as well as the opener are corrupt; i.e. for any PPT adversary $\mathcal{A}$:

$$\Pr\left[\begin{array}{c} params \leftarrow \mathsf{ARS.Setup}(1^\lambda);\ (m,\sigma,R,ek,vk_1,vk_2,\phi_1,\phi_2) \leftarrow \\ \mathcal{A}(params) : \forall i \in \{1,2\}, \mathsf{ARS.Judge}(m,\sigma,R,ek,vk_i,\phi_i) = 1 \\ \wedge vk_1 \neq vk_2 \end{array}\right] \leq \mathsf{negl}(\lambda)$$

3 Preliminaries

3.1 Indistinguishability Obfuscation

Definition 4 (Indistinguishability Obfuscator ($i\mathcal{O}$)[1]). *A* PPT *algorithm $i\mathcal{O}$ is said to be an indistinguishability obfuscator for a collection of circuits $\{\mathcal{C}_\lambda\}$ (λ is the security parameter), if it satisfies the following two conditions:*

1. ***Functionality:*** $\forall \lambda \in \mathbb{N}$ *and* $\forall C \in \mathcal{C}_\lambda$, $\Pr[\forall x : i\mathcal{O}(1^\lambda, C)(x) = C(x)] = 1$

2. ***Indistinguishability:*** *For all* PPT *distinguisher* $\mathcal{D}$*, there exists a negligible function* negl*, such that for all* $\lambda \in \mathbb{N}$ *and for all pairs of circuits* $C_1, C_2 \in \mathcal{C}_\lambda$*, if* $\Pr[\forall x : C_1(x) = C_2(x)] = 1$*, then we have* $\boldsymbol{Adv}_{\mathcal{D}}^{Obf} \leq \mathsf{negl}(\lambda)$ *i.e.*

$$|\Pr[\mathcal{D}(i\mathcal{O}(1^\lambda, C_1)) = 1] - \Pr[\mathcal{D}(i\mathcal{O}(1^\lambda, C_2)) = 1]| \leq \mathsf{negl}(\lambda)$$

Here, we will consider polynomial-size circuits only. In 2013, Garg *et al.* [11] proposed the first candidate construction of an efficient *indistinguishability obfuscator* for any general purpose boolean circuit. Their construction is based on the multilinear map candidates of [11] and [10].

3.2 Puncturable Pseudorandom Functions

Definition 5 (Puncturable Pseudorandom Functions (PPRFs)[19]). *Let* $\ell(\cdot)$ *and* $m(\cdot)$ *be two polynomially bounded length functions. Let* $\mathcal{F} = \{\mathsf{F}_K : \{0,1\}^{\ell(\lambda)} \rightarrow \{0,1\}^{m(\lambda)} | K \leftarrow \{0,1\}^\lambda, \lambda \in \mathbb{N}\}$. $\mathcal{F}$ *is called a family of puncturable PRFs if it is associated with two turing machines* $\mathsf{K}_\mathcal{F}$ *and* $\mathsf{P}_\mathcal{F}$*, such that* $\mathsf{P}_\mathcal{F}$ *takes as input a key* $K \leftarrow \{0,1\}^\lambda$ *and a point* $u^* \in \{0,1\}^{\ell(\lambda)}$*, and outputs a punctured key* K_{u^*}*, so that the following two conditions are satisfied:*

1. ***Functionality preserved under puncturing:*** *For every* $u^* \in \{0,1\}^{\ell(\lambda)}$,

$$\Pr\left[K \leftarrow K_\mathcal{F}(1^\lambda);\ K_{u^*} = \mathsf{P}_\mathcal{F}(K, u^*) : \forall u \neq u^*, \mathsf{F}_K(u) = \mathsf{F}_{K_{u^*}}(u)\right] = 1$$

2. ***Indistinguishability at punctured points:*** $\forall$ PPT *distinguisher* $\mathcal{D}$*, below ensembles are computationally indistinguishable i.e.* $\boldsymbol{Adv}_{\mathcal{D}}^{PPRF} \leq \mathsf{negl}(\lambda)$*:*
   - $\{u^*, K_{u^*}, \mathsf{F}_K(u^*) : K \leftarrow K_\mathcal{F}(1^\lambda); K_{u^*} = \mathsf{P}_\mathcal{F}(K, u^*)\}$
   - $\{u^*, K_{u^*}, x : K \leftarrow K_\mathcal{F}(1^\lambda); K_{u^*} = \mathsf{P}_\mathcal{F}(K, u^*); x \leftarrow \{0,1\}^{\ell(\lambda)}\}$

Recently, [3, 16, 6] observed that puncturable PRFs can easily be constructed from GGM's PRFs [13] which are based on one-way functions. We will use the following lemma on statistical injective PPRF in our construction.

Lemma 1 ([19]). *If one-way functions exist, then for all efficiently computable functions* $\ell(\lambda)$*,* $m(\lambda)$*, and* $e(\lambda)$ *such that* $m(\lambda) \geq 2\ell(\lambda) + e(\lambda)$*, there exists a statistically injective* PPRF *family with failure probability* $2^{-e(\lambda)}$ *that maps* $\ell(\lambda)$ *bits to* $m(\lambda)$ *bits.*

3.3 IND-CPA Secure Public Key Encryption Scheme

Definition 6 (Public Key Encryption Scheme (PKE)). *A public key encryption scheme* PKE=($\mathsf{PKE.KGen}$,$\mathsf{PKE.Encrypt}$,$\mathsf{PKE.Decrypt}$) *over a* PPT *setup* $\mathsf{PKE.Setup}$ *is a 3-tuple of algorithms.*

- $\mathsf{PKE.Setup}(1^\lambda)$: *On input the security parameter* 1^λ, $\mathsf{PKE.Setup}$ *outputs a list of public parameters params as follows: key spaces* $(\mathcal{EK}, \mathcal{DK})$*; plaintext and ciphertext spaces* $\mathcal{P}$ *and* $\mathcal{C}$.

– $\mathsf{PKE.KGen}(params)$: *On input params, the* PPT *algorithm* $\mathsf{PKE.KGen}$ *outputs a pair of public and private keys* (ek, dk)*.*
– $\mathsf{PKE.Encrypt}(ek, m \in \mathcal{M})$: *On input public key ek and message m, the* PPT *algorithm* $\mathsf{PKE.Encrypt}$ *outputs a ciphertext* $c \in \mathcal{C}$*.*
– $\mathsf{PKE.Decrypt}(dk, c)$ *: On input secret key dk and ciphertext c,* $\mathsf{PKE.Decrypt}$ *outputs a message* $m \in \mathcal{M}$*.*

Correctness For all key pairs $(ek, dk) \leftarrow \mathsf{PKE.KGen}(params)$, and for all messages $m \in \mathcal{M}$: $\Pr[\mathsf{PKE.Decrypt}(dk, \mathsf{PKE.Encrypt}(ek, m)) = m] = 1$.

Security. For all PPT adversaries $\mathcal{A}$:

$$\Pr\begin{bmatrix} params \leftarrow \mathsf{PKE.Setup}(1^\lambda);\ (ek, dk) \leftarrow \mathsf{PKE.KGen}(params); \\ (m_0, m_1) \leftarrow \mathcal{A}(ek, params);\ b \leftarrow \{0,1\}; \\ c \leftarrow \mathsf{PKE.Encrypt}(ek, m_b) : \mathcal{A}(c) = m_b \end{bmatrix} \leq \frac{1}{2} + \mathsf{negl}(\lambda)$$

3.4 One-way Function

Definition 7 (One-way Function (OWF)[12]). *A function* $f : \mathcal{X} \to \mathcal{Y}$ *(over* PPT *setup* $\mathsf{OWF.Setup}$*, which defines* $f, \mathcal{X}, \mathcal{Y}$ *is one-way if f is polynomial-time computable and is hard to invert, i.e. for all* PPT *adversaries* $\mathcal{A}$*,* $\boldsymbol{Adv}_{\mathcal{A}}^{OWF} \leq \mathsf{negl}(\lambda)$ *i.e.* $\Pr\left[x \leftarrow \mathcal{X};\ y := f(x) : \mathcal{A}(y) = x\right] \leq \mathsf{negl}(\lambda)$

3.5 Pseudorandom Generator

Definition 8 (Pseudorandom Generator (PRG)[14]). *A function* $\mathsf{PRG} : \{0,1\}^\lambda \to \{0,1\}^{p(\lambda)}$ *(over a setup* $\mathsf{PRG.Setup}$*, which defines the function* PRG*, its domain and range) is PRG if it is polynomial-time computable, length expanding and is pseudorandom, i.e. for all* PPT *adversaries* $\mathcal{A}$*,* $\boldsymbol{Adv}_{\mathcal{A}}^{PRG} \leq \mathsf{negl}(\lambda)$ *i.e.*

$$\left|\Pr[\mathcal{A}(\mathsf{PRG}(s)) = 1 : s \leftarrow \{0,1\}^\lambda] - \Pr[\mathcal{A}(r) = 1 : r \leftarrow \{0,1\}^{p(\lambda)}]\right| \leq \mathsf{negl}(\lambda)$$

3.6 Collision Resistant Hash Function

Definition 9 (Collision Resistant Hash Function (CRHF)). *A family* $\mathcal{H}$ *of collection of functions* $\mathsf{Hash} : \{0,1\}^* \to \{0,1\}^{\ell_h}$ *(over* PPT *setup* $\mathsf{CRHF.Setup}$*, which defines the function* Hash*, its domain and range) is called a family of collision resistant hash functions if* Hash *is polynomial-time computable, length compressing and it is hard to find collisions in* Hash*, i.e. for all* PPT *adversaries* $\mathcal{A}$*,* $\boldsymbol{Adv}_{\mathcal{A}}^{CRHF} \leq \mathsf{negl}(\lambda)$ *i.e.*

$$\Pr_{\mathsf{Hash} \leftarrow \mathcal{H}}\begin{bmatrix} params \leftarrow \mathsf{CRHF.Setup}(1^\lambda);\ (x_0, x_1) \leftarrow \mathcal{A}(params, \mathsf{Hash}) \\ : x_0 \neq x_1 \wedge \mathsf{Hash}(x_0) = \mathsf{Hash}(x_1) \end{bmatrix} \leq \mathsf{negl}(\lambda)$$

4 A New Accountable Ring Signature Scheme

We now present the details of our construction. We assume that each user in the system is uniquely identified by an index $i \in [n], n \in \mathbb{N}$. Let $\mathcal{M} = \{0,1\}^{\ell_m}$ be our message space. Let $\mathcal{SK} = \{0,1\}^{\ell_s}$, $\mathcal{VK} = \{0,1\}^{\ell_v}$, $\mathcal{EK} = \{0,1\}^{\ell_e}$ and $\mathcal{DK} = \{0,1\}^{\ell_d}$ respectively be our signing, verification, encryption and decryption key spaces. Here $\ell_m, \ell_s, \ell_v, \ell_e$ and ℓ_d are polynomials in security parameter λ. We assume that the set of verification keys in the ring R are always arranged in *increasing order* of their indexes, in order to enable the hash function to compute on R. We will use following primitives in our construction:

- Pseudo-random generator $\mathsf{PRG} : \{0,1\}^{\lambda} \to \{0,1\}^{\ell_p}$ with $\ell_p \geq 2\lambda$.
- One-way function $f : \{0,1\}^{\ell_s} \to \{0,1\}^{\ell_v}$.
- IND-CPA secure PKE scheme as defined in Sect. 3.3.
- Collision resistant hash function $\mathsf{Hash} : \{0,1\}^{n \cdot \ell_v} \to \{0,1\}^{\ell_h}$. Here n is size of the ring and $n \cdot \ell_v > \ell_h$.
- A statistically injective PPRF $F_1(K_1, \cdot)$ that accepts inputs of length $\ell_m + \ell_s + \ell_h + \ell_e + \ell_p$, and outputs strings of length ℓ_1. Note that this type of PPRF exists, and is easy to construct from Lemma. 1.
- A PPRF $F_2(K_2, \cdot)$ that accepts inputs of length ℓ_1, and outputs strings of length ℓ_2 such that $\ell_2 = \ell_m + \ell_s + \ell_h + \ell_e + \ell_p$.
- One-way function $f_1 : \{0,1\}^{\ell_1} \to \{0,1\}^{\ell_3}$.
- A PPRF $F_3(K_3, \cdot)$ that accepts inputs of length $\ell_e + \ell_c + \ell_v$, and outputs strings of length ℓ_ϕ. Here ℓ_c is the length of ciphertext as output by the PKE scheme and ℓ_ϕ is the length of NIZK proof.
- One-way function $f_2 : \{0,1\}^{\ell_\phi} \to \{0,1\}^{\ell_4}$.
- An indistinguishability obfuscator $i\mathcal{O}$ as defined in Sect. 3.1.

Construction of our ARS scheme from $i\mathcal{O}$.

- $\mathsf{ARS.Setup}(1^\lambda)$: A trusted authority runs this algorithm for once. It takes security parameter 1^λ as input and generates key spaces as defined above. Next, it chooses keys K_1, K_2, K_3 for PPRFs F_1, F_2, F_3 respectively, and creates obfuscated programs $\mathsf{Sign} = i\mathcal{O}(\mathcal{P}_S), \mathsf{Verify} = i\mathcal{O}(\mathcal{P}_V), \mathsf{NIZKprove} = i\mathcal{O}(\mathcal{P}_{NP})$ and $\mathsf{NIZKverify} = i\mathcal{O}(\mathcal{P}_{NV})$ (shown in Figs. 1,2,3,4). Finally, it outputs public parameters $params = (\mathcal{SK}, \mathcal{VK}, \mathcal{DK}, \mathcal{EK}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{NIZKprove}, \mathsf{NIZKverify})$.
- $\mathsf{ARS.UKGen}(params)$: This algorithm takes $params$ as input, chooses a secret signing key $sk \leftarrow \mathcal{SK}$ and computes verification key as $vk = f(sk) \in \mathcal{VK}$.
- $\mathsf{ARS.OKGen}(params)$: This algorithm takes $params$ as input and outputs a key pair (ek, dk) for the PKE scheme. This algorithm is same as $\mathsf{PKE.KGen}$.
- $\mathsf{ARS.Sign}(params, m, sk, vk, R, ek, r)$: This algorithm takes $params$, message m, a key pair (sk, vk), a ring R, an opener's public key ek and randomness r as input, runs Sign on these inputs and returns ring signature σ.

```
$\mathcal{P}_S$
Hardwired: Keys $K_1, K_2$
Input: Message $m$, key pair $(sk, vk)$ , ring $R$, key $ek$ and randomness $r$
1 if $(f(sk) = vk \wedge vk \in R)$ then
2     Set $\alpha := F_1(K_1, (m \| vk \| \mathsf{Hash}(R) \| ek \| \mathsf{PRG}(r)))$
3     Set $\beta := F_2(K_2, \alpha) \oplus (m \| vk \| \mathsf{Hash}(R) \| ek \| \mathsf{PRG}(r))$
4     return $(\sigma := (\alpha, \beta))$
5 else
6     return $\perp$
```

Figure 1: The circuit Sign

– $\mathsf{ARS.Verify}(params, m, \sigma, R, ek)$: This algorithm takes $params$, message m, signature σ, a ring R and an opener's public key ek as input, runs Verify on these inputs and returns a verification key of the signer encrypted under the public key ek of the opener if the signature is accepted, else it returns $\perp$.

```
$\mathcal{P}_V$
Hardwired: Keys $K_1, K_2$
Input: Message $m$, signature $\sigma = (\alpha, \beta)$, ring $R$ and opener's public key $ek$
1 Compute $F_2(K_2, \alpha) \oplus \beta = (m' \| vk' \| h' \| ek' \| r')$
2 if $\big(m = m' \wedge \mathsf{Hash}(R) = h' \wedge ek = ek' \wedge f_1(\alpha) =$
    $f_1(F_1(K_1, (m' \| vk' \| h' \| ek' \| r')))\big)$ then
3     return $(c := \mathsf{PKE.Encrypt}(ek, vk'; r'))$
4 else
5     return $\perp$
```

Figure 2: The circuit Verify

– $\mathsf{ARS.Open}(params, m, \sigma, R, ek, dk)$: This algorithm takes $params$, message m, signature σ, a ring R and an opener's PKE key pair (ek, dk) as input, runs Verify on these inputs to retrieve c. Next, it decrypts c using secret key dk to output verification key vk of the signer. In addition, it also outputs a proof ϕ of correct decryption using the program NIZKprove.

```
$\mathcal{P}_{NP}$
Hardwired: Keys $K_3$
Input: Statement $x = (ek\|c\|vk)$ and witness $dk$
1 if $\Big((ek = \mathsf{PKE.KGen}(params, dk) \in \mathcal{EK}) \wedge (vk = \mathsf{PKE.Decrypt}(dk, c)) \wedge (vk \in \mathcal{VK})\Big) = 1$ then
2     return $\phi := F_3(K_3, x)$
3 else
4     return $\perp$
```

Figure 3: The circuit $\mathsf{NIZKprove}$

– $\mathsf{ARS.Judge}(params, m, \sigma, R, ek, vk, \phi)$: This algorithm takes $params$, message m, signature σ, a ring R and an opener's public key ek, verification key vk, proof ϕ as input, runs Verify on these inputs to retrieve c and outputs 1 if $\mathsf{NIZKverify}((ek\|c\|vk), \phi) = 1$ else 0.

```
$\mathcal{P}_{NV}$
Hardwired: Keys $K_3$
Input: Statement $x = (ek\|c\|vk)$ and proof $\phi$
1 if $f_2(\phi) = f_2(F_3(K_3, x))$ then
2     return 1
3 else
4     return $\perp$
```

Figure 4: The circuit $\mathsf{NIZKverify}$

5 Proof of Security

We will use the following lemma in our security proofs.

Lemma 2. *For any fixed input (m, sk, R, ek, r) to the* Sign *program, there can exist at most one valid signature string $\sigma = (\alpha, \beta)$. This happens with high probability if K_1 is chosen such that the $F_1(K_1, \cdot)$ is statistically injective and* Hash *is a collision resistant hash function.*

Proof. Since $F_1(K_1, \cdot)$ is injective over the choice of K_1 and Hash is collision resistant therefore, there can exist at most one string α (which implies unique (α, β)) when $F_1(K_1, \cdot)$ is applied to a unique string $(m, vk, \mathsf{Hash}(R), \mathsf{PRG}(r))$. Furthermore, if $\sigma = (\alpha, \beta)$ is choses at random then there can exists at most $2^{\ell_m + \ell_v + \ell_h + \ell_e + \ell_p}$ valid values of σ. (This lemma is adapted from a similar lemma of [**?**] to include an additional hash function.)

Theorem 1. *The* ARS *scheme of Sect. 4 is selectively unforgeable if the primitives of Sect. 3 satisfy their respective security properties.*

Proof. We prove the security by a sequence of hybrid experiments and show that if there exists a PPT adversary $\mathcal{A}$ that can break the selective unforgeability of our ARS scheme, then we can construct a challenger $\mathcal{B}$ who can break the security of one-way function.

Hybrid 0. This hybrid is same as the unforgeability game of Sect. 2.

1. **Setup Phase:**
   (a) $\mathcal{C}$ chooses a set $S \subset [n]$ and runs the algorithm $\mathsf{ARS.UKGen}$ to generate signing-verification key pairs $\{(sk_i, vk_i)\}_{i \in S}$ for the users and registers them with the PKI. Finally, it sends $\{vk_i\}_{i \in S}$ to the adversary $\mathcal{A}$.
   (b) $\mathcal{A}$ sends a message m^*, a ring $R^* \subseteq \{vk_i\}_{i \in S}$, a verification key $vk^* \in R^*$, an opener's public key ek^* and randomness r^* to challenger $\mathcal{C}$, claiming it can forge an accountable ring signature σ^* w.r.t $(m^*, vk^*, R^*, ek^*, r^*)$.
   (c) The challenger $\mathcal{C}$ first chooses keys K_1, K_2 for PPRFs F_1, F_2 respectively, and produces two obfuscated programs $\mathsf{Sign} = i\mathcal{O}(\mathcal{P}_S)$ and $\mathsf{Verify} = i\mathcal{O}(\mathcal{P}_V)$ (Figs. 1,2). Then, it sends these programs to the adversary $\mathcal{A}$.
2. **Query Phase:** $\mathcal{A}$ can make the following three types of queries to $\mathcal{C}$. $\mathcal{C}$ answers these queries via oracles $\mathcal{O}_{Reg}$, $\mathcal{O}_{Cor}$ and $\mathcal{O}_{Sig}$.
   – **Registration query:** $\mathcal{A}$ runs the algorithm $\mathsf{ARS.UKGen}$ to generate a signing-verification key pair $(sk_i, vk_i), i \notin [n] \setminus S$ and interacts with the oracle $\mathcal{O}_{Reg}$ to register vk_i with the PKI. Let $\mathcal{Q}_{Reg}$ be the set of verification keys registered by $\mathcal{A}$.
   – **Corruption query:** $\mathcal{A}$ queries a verification key $vk_i, i \in S$ such that $vk_i \neq vk^*$ to the oracle $\mathcal{O}_{Cor}$. The oracle returns the corresponding signing key sk_i. Let $\mathcal{Q}_{Cor}$ be the set of verification keys vk_i for which the corresponding signing keys has been revealed.
   – **Signing query:** $\mathcal{A}$ queries (m, vk_i, R, ek, r) to the oracle $\mathcal{O}_{Sign}$. The oracle returns a signature $\sigma_i = \mathsf{Sign}(m, vk_i, R, ek, r)$ if $vk_i \neq vk^*$ and $vk_i \notin \mathcal{Q}_{Reg} \cup \mathcal{Q}_{Cor}$. Let $\mathcal{Q}_{Sign}$ be the set of queries and their responses $(m, vk, R, ek, r, \sigma)$.
3. **Forgery Phase:** $\mathcal{A}$ outputs a signature σ^* w.r.t $(m^*, vk^*, R^*, ek^*, r^*)$.

$\mathcal{A}$ wins the above game if $\mathsf{ARS.Verify}(m, \sigma, R, ek) = 1$. Let $\boldsymbol{Adv}_{\mathcal{A}}^{Hyb\,0}$ denote the advantage of $\mathcal{A}$ in this hybrid.

Hybrid 1. This hybrid is identical to *Hybrid 0* with the exception that $\mathcal{C}$ gives out the two obfuscated programs as $\mathsf{Sign} = i\mathcal{O}(\mathcal{P}_S^*)$ and $\mathsf{Verify} = i\mathcal{O}(\mathcal{P}_V^*)$ (Figs. 5,6). The details of the modifications are as follows: $\mathcal{C}$ punctures the key K_2 at the point $\alpha^* = F_1(K_1, (m^* \| vk^* \| \mathsf{Hash}(R^*) \| ek^* \| \mathsf{PRG}(r^*))$ and hardwires this punctured key $K_2(\{\alpha^*\})$ along with message m^*, verification key vk^*, ring R^*, public key ek^* and randomness r^* in programs $\mathcal{P}_S^*$ and $\mathcal{P}_V^*$. In addition, it hardwires value $\beta^* = F_2(K_2, \alpha^*) \oplus (m^* \| vk^* \| \mathsf{Hash}(R^*) \| ek^* \| \mathsf{PRG}(r^*))$ in program $\mathcal{P}_S^*$. Furthermore, a "if" condition is added at line 1 in program $\mathcal{P}_S^*$ to check $(m, vk, R, ek, r) = (m^*, vk^*, R^*, ek^*, r^*)$. If so, output the signature as $\sigma^* = (\alpha^*, \beta^*)$. Similarly, $\mathcal{P}_V^*$ is modified to output c^* if $(m, \sigma, R, ek) = (m^*, \sigma^*, R^*, ek^*)$. Let $\boldsymbol{Adv}_{\mathcal{A}}^{Hyb\,1}$ denote the advantage of $\mathcal{A}$ in this hybrid.

```
                              $\mathcal{P}_S^*$
Hardwired: Keys $K_1, K_2(\{\alpha^*\})$ and values $m^*, vk^*, R^*, ek^*, r^*, \beta^*$
Input: Message $m$, key pair $(sk, vk)$, ring $R$, key $ek$ and randomness $r$
1 if $(m = m^* \wedge f(sk) = vk^* \wedge f(sk) \in R^* \wedge R = R^* \wedge ek = ek^* \wedge r = r^*)$
    then
2       Set $\alpha^* := F_1(K_1, (m^* \| vk^* \| \mathsf{Hash}(R^*) \| ek^* \| \mathsf{PRG}(r^*)))$
3       return $(\sigma^* := (\alpha^*, \beta^*))$
4 else if $(f(sk) = vk \wedge f(sk) \in R)$ then
5       Set $\alpha := F_1(K_1, (m \| vk \| \mathsf{Hash}(R) \| ek \| \mathsf{PRG}(r)))$
6       Set $\beta := F_2(K_2, \alpha) \oplus (m \| vk \| \mathsf{Hash}(R) \| ek \| \mathsf{PRG}(r))$
7       return $(\sigma := (\alpha, \beta))$
8 else
9       return $\perp$
```

Figure 5: The circuit Sign

```
                              $\mathcal{P}_V^*$
Hardwired: Keys $K_1, K_2(\{\alpha^*\})$ and values $m^*, vk^*, R^*, ek^*, r^*$
Input: Message $m$, signature $\sigma = (\alpha, \beta)$, ring $R$ and opener's public key $ek$
1 if $\big(m = m^* \wedge R = R^* \wedge ek = ek^* \wedge f_1(\alpha) =$
    $f_1(F_1(K_1, (m^* \| vk^* \| \mathsf{Hash}(R^*) \| ek^* \| \mathsf{PRG}(r^*))))\big)$ then
2       return $(c^* := \mathsf{PKE.Encrypt}(ek, vk^*; r^*))$
3 else if $\big(F_2(K_2, \alpha) \oplus \beta = (m' \| vk' \| h' \| ek' \| r') \wedge m = m' \wedge \mathsf{Hash}(R) =$
    $h' \wedge ek = ek' \wedge f_1(\alpha) = f_1(F_1(K_1, (m' \| vk' \| h' \| ek' \| r')))\big)$ then
4       return $(c := \mathsf{PKE.Encrypt}(ek, vk'; r'))$
5 return $\perp$
```

Figure 6: The circuit Verify

Hybrid 2. This hybrid is identical to *Hybrid 1* with the exception that β^* is chosen uniformly at random from the set $\{0,1\}^{\ell_2}$. Let $\boldsymbol{Adv}_{\mathcal{A}}^{Hyb\,2}$ denote the advantage of $\mathcal{A}$ in this hybrid.

Hybrid 3. This hybrid is identical to *Hybrid 2* with the exception that $\mathcal{C}$ gives out the two obfuscated programs as $\mathsf{Sign} = i\mathcal{O}(\mathcal{P}_S^{**})$ and $\mathsf{Verify} = i\mathcal{O}(\mathcal{P}_V^{**})$ (Figs. 7,8). The details of the modifications are as follows: $\mathcal{C}$ punctures the key K_1 at the point $(m^* \| vk^* \| \mathsf{Hash}(R) \| ek^* \| \mathsf{PRG}(r^*))$ and hardwires this punctured key $K_1(\{(m^* \| vk^* \| \mathsf{Hash}(R) \| ek^* \| \mathsf{PRG}(r^*))\})$ in programs $\mathcal{P}_S^{**}$ and $\mathcal{P}_V^{**}$. Finally, it hardwires values $\alpha^* = F_1(K_1, (m^* \| vk^* \| \mathsf{Hash}(R) \| ek^* \| \mathsf{PRG}(r^*)))$ and $u^* = f_1(\alpha^*)$ in programs $\mathcal{P}_S^{**}$ and $\mathcal{P}_V^{**}$ respectively. Let $\boldsymbol{Adv}_{\mathcal{A}}^{Hyb\,3}$ denote the advantage of $\mathcal{A}$ in this hybrid.

```
                         $\mathcal{P}_S^{**}$
Hardwired: Keys $K_1(\{m^*\|vk^*\|\mathsf{Hash}(R^*)\|ek^*\|\mathsf{PRG}(r^*)\}), K_2(\{\alpha^*\})$ and
           values $m^*, vk^*, R^*, ek^*, r^*, \alpha^*, \beta^*$
Input: Message $m$, key pair $(sk, vk)$ , ring $R$, key $ek$ and randomness $r$
1 if $(m = m^* \wedge f(sk) = vk^* \wedge f(sk) \in R^* \wedge R = R^* \wedge ek = ek^* \wedge r = r^*)$
   then
2     return $(\sigma^* := (\alpha^*, \beta^*))$
3 else if $(f(sk) = vk \wedge f(sk) \in R)$ then
4     Set $\alpha := F_1(K_1, (m\|vk\|\mathsf{Hash}(R)\|ek\|\mathsf{PRG}(r)))$
5     Set $\beta := F_2(K_2, \alpha) \oplus (m\|vk\|\mathsf{Hash}(R)\|ek\|\mathsf{PRG}(r))$
6     return $(\sigma := (\alpha, \beta))$
7 else
8     return $\perp$
```

Figure 7: The circuit Sign

```
                         $\mathcal{P}_V^{**}$
Hardwired: Keys $K_1, K_2(\{\alpha^*\})$ and values $m^*, vk^*, R^*, ek^*, r^*, u^*$
Input: Message $m$, signature $\sigma = (\alpha, \beta)$, ring $R$ and opener's public key $ek$
1 if $\big(m = m^* \wedge R = R^* \wedge ek = ek^* \wedge f_1(\alpha) = u^*\big)$ then
2     return $(c^* := \mathsf{PKE.Encrypt}(ek, vk^*; r^*))$
3 else if $\big(F_2(K_2, \alpha) \oplus \beta = (m'\|vk'\|h'\|ek'\|r') \wedge m = m' \wedge \mathsf{Hash}(R) =$
   $h' \wedge ek = ek' \wedge f_1(\alpha) = f_1(F_1(K_1, (m'\|vk'\|h'\|ek'\|r')))\big)$ then
4     return $(c := \mathsf{PKE.Encrypt}(ek, vk'; r'))$
5 return $\perp$
```

Figure 8: The circuit Verify

Hybrid 4. This hybrid is identical to *Hybrid 3* with the exception that α^* is chosen uniformly at random from the set $\{0,1\}^{\ell_1}$. Let $\boldsymbol{Adv}_{\mathcal{A}}^{Hyb\,4}$ denote the advantage of $\mathcal{A}$ in this hybrid.

The indistinguishability of successive hybrid games (discussed above) is shown in Lemmas. 3, 4, 5, 6, 7.

Lemma 3. *If $\mathcal{A}$ can distinguish* Hybrid 0 *from* Hybrid 1 *in the* unforgeability game*, i.e.* $\boldsymbol{Adv}_{\mathcal{A}}^{Hyb\,0} - \boldsymbol{Adv}_{\mathcal{A}}^{Hyb\,1}$ *is non-negligible then there exists an adversary $\mathcal{B}$ who can break the security of $i\mathcal{O}$.*

Proof. We argue that if there is a non-negligible difference in advantages of $\mathcal{A}$ in hybrids *Hybrid 0* and *Hybrid 1* then we can construct an adversary $\mathcal{B}$ that breaks the security of $i\mathcal{O}$. Firstly, observe that the programs $\mathcal{P}_S$ and $\mathcal{P}_S^*$ as well as the programs $\mathcal{P}_V$ and $\mathcal{P}_V^*$, are functionally equivalent. Now, consider a scenario where the attacker $\mathcal{B}$ interacts with an $i\mathcal{O}$ challenger while acting as a challenger for $\mathcal{A}$. $\mathcal{B}$ submits the program pairs $P_0 = (\mathcal{P}_S, \mathcal{P}_V)$ and $P_1 = (\mathcal{P}_S^*, \mathcal{P}_V^*)$ to the $i\mathcal{O}$ challenger. The $i\mathcal{O}$ challenger chooses $b \leftarrow \{0,1\}$ at random, obfuscates the programs in P_b , and returns the output to $\mathcal{B}$. $\mathcal{B}$ plugs in these obfuscated programs in *Hybrid 0* and plays the rest of the game. Note that if the $i\mathcal{O}$ challenger chooses P_0 then we are in *Hybrid 0*. If it chooses P_1 then we are in *Hybrid 1*. $\mathcal{B}$ will output 1 if $\mathcal{A}$ successfully forges. Therefore, if $\mathcal{A}$ have different advantages in hybrids *Hybrid 0* and *Hybrid 1* then $\mathcal{B}$ can attack $i\mathcal{O}$ security with high probability i.e. $\boldsymbol{Adv}_{\mathcal{A}}^{Hyb\,0} - \boldsymbol{Adv}_{\mathcal{A}}^{Hyb\,1} = \boldsymbol{Adv}_{\mathcal{B}}^{i\mathcal{O}} \leq \mathsf{negl}(\lambda)$ (Sect. 3.1). $\square$

Lemma 4. *If $\mathcal{A}$ can distinguish* Hybrid 1 *from* Hybrid 2 *in the* unforgeability game*, i.e.* $\boldsymbol{Adv}_{\mathcal{A}}^{Hyb\,1} - \boldsymbol{Adv}_{\mathcal{A}}^{Hyb\,2}$ *is non-negligible then there exists an adversary $\mathcal{B}$ who can break the security of* PPRF F_2.

Proof. We argue that if there is a non-negligible difference in advantages of $\mathcal{A}$ in hybrids *Hybrid 1* and *Hybrid 2* then we can construct an adversary $\mathcal{B}$ who can break the security of PPRF F_2. Consider a scenario where the attacker $\mathcal{B}$ interacts with a PPRF challenger while acting as a challenger for $\mathcal{A}$. First, $\mathcal{B}$ receives $(m^*, vk^*, R^*, ek^*, r^*)$ from $\mathcal{A}$. Then, it chooses $b \leftarrow \{0,1\}, K_1 \leftarrow \{0,1\}^\lambda$, computes $\alpha^* = F_1(K_1, (m^* \| vk^* \| \mathsf{Hash}(R^*) \| ek^* \| \mathsf{PRG}(r^*)))$, and submits α^* to the PPRF challenger. Next, the PPRF challenger choses $K_2 \leftarrow \{0,1\}^\lambda$, punctures K_2 at point α^*, and sends this punctured PRF key $K_2(\{\alpha^*\})$ along with a challenge $w^* \in \{0,1\}^{\ell_2}$ to $\mathcal{B}$. Then, $\mathcal{B}$ continues to run the game in *Hybrid 1* except that it plugs in β^* as $\beta^* = w^* \oplus (m^* \| vk^* \| \mathsf{Hash}(R^*) \| ek^* \| \mathsf{PRG}(r^*))$. Note that if the PPRF challenger outputs w^* as $w^* = F_2(K_2, \alpha^*)$ then we are in *Hybrid 1*. If it outputs w^* as $w^* \leftarrow \{0,1\}^{\ell_2}$ then we are in *Hybrid 2*. $\mathcal{B}$ will output 1 if $\mathcal{A}$ successfully forges. Therefore, if $\mathcal{A}$ have different advantages in hybrids *Hybrid 1* and *Hybrid 3* then $\mathcal{B}$ can attack the PPRF security with high probability i.e. $\boldsymbol{Adv}_{\mathcal{A}}^{Hyb\,1} - \boldsymbol{Adv}_{\mathcal{A}}^{Hyb\,2} = \boldsymbol{Adv}_{\mathcal{B}}^{PPRF} \leq \mathsf{negl}(\lambda)$ (Sect. 3.2). $\square$

Lemma 5. *If $\mathcal{A}$ can distinguish* Hybrid 2 *from* Hybrid 3 *in the* unforgeability game*, i.e.* $\boldsymbol{Adv}_{\mathcal{A}}^{Hyb\,2} - \boldsymbol{Adv}_{\mathcal{A}}^{Hyb\,3}$ *is non-negligible then there exists an adversary $\mathcal{B}$ who can break the security of $i\mathcal{O}$.*

Proof. We argue that if there is a non-negligible difference in advantages of $\mathcal{A}$ in hybrids *Hybrid 2* and *Hybrid 3* then we can construct an adversary $\mathcal{B}$ that breaks the security of $i\mathcal{O}$. Firstly, observe that the programs $\mathcal{P}_S^*$ and $\mathcal{P}_S^{**}$ as well as the programs $\mathcal{P}_V^*$ and $\mathcal{P}_V^{**}$, are functionally equivalent. Now, consider a scenario where the attacker $\mathcal{B}$ interacts with an $i\mathcal{O}$ challenger while acting as a challenger for $\mathcal{A}$. $\mathcal{B}$ submits the program pairs $P_0 = (\mathcal{P}_S^*, \mathcal{P}_V^*)$ and $P_1 = (\mathcal{P}_S^{**}, \mathcal{P}_V^{**})$ to the $i\mathcal{O}$ challenger. The $i\mathcal{O}$ challenger chooses $b \leftarrow \{0,1\}$ at random, obfuscates the programs in P_b , and returns the output to $\mathcal{B}$. $\mathcal{B}$ plugs in these obfuscated programs in *Hybrid 2* and plays the rest of the game. Note that if the $i\mathcal{O}$ challenger chooses P_0 then we are in *Hybrid 2*. If it chooses P_1 then we are in *Hybrid 3*. $\mathcal{B}$ will output 1 if $\mathcal{A}$ successfully forges. Therefore, if $\mathcal{A}$ have different advantages in hybrids *Hybrid 2* and *Hybrid 3* then $\mathcal{B}$ can attack $i\mathcal{O}$ security with high probability i.e. $\boldsymbol{Adv}_{\mathcal{A}}^{Hyb\,2} - \boldsymbol{Adv}_{\mathcal{A}}^{Hyb\,3} = \boldsymbol{Adv}_{\mathcal{B}}^{i\mathcal{O}} \leq \mathsf{negl}(\lambda)$ (Sect. 3.1). $\square$

Lemma 6. *If $\mathcal{A}$ can distinguish* Hybrid 3 *from* Hybrid 4 *in the* unforgeability game*, i.e.* $\boldsymbol{Adv}_{\mathcal{A}}^{Hyb\,3} - \boldsymbol{Adv}_{\mathcal{A}}^{Hyb\,4}$ *is non-negligible then there exists an adversary $\mathcal{B}$ who can break the security of* PPRF F_1.

Proof. We argue that if there is a non-negligible difference in advantages of $\mathcal{A}$ in hybrids *Hybrid 1* and *Hybrid 2* then we can construct an adversary $\mathcal{B}$ who can break the security of PPRF F_1. Consider a scenario where the attacker $\mathcal{B}$ interacts with a PPRF challenger while acting as a challenger for $\mathcal{A}$. First, $\mathcal{B}$ receives $(m^*, vk^*, R^*, ek^*, r^*)$ from $\mathcal{A}$ and submits it to the PPRF challenger. Next,the PPRF challenger chooses a PPRF key $K_1 \leftarrow \{0,1\}^\lambda$ and punctures K_1 at point $(m^*\|vk^*\|\mathsf{Hash}(R^*)\|ek^*\|\mathsf{PRG}(r^*))$ and sends this punctured key $K_1(\{(m^*\|vk^*\|\mathsf{Hash}(R^*)\|ek^*\|\mathsf{PRG}(r^*))\})$ and a challenge z^* to $\mathcal{B}$. Then, $\mathcal{B}$ continues to run the game in *Hybrid 3* except it plugs in $\alpha^* = z^*$. Note that if the PPRF challenger outputs z^* as $z^* = F_1(K_1, (m^*\|vk^*\|\mathsf{Hash}(R^*)\|ek^*\|\mathsf{PRG}(r^*)))$ then we are in *Hybrid 3*. If it outputs z^* as $z^* \leftarrow \{0,1\}^{\ell_1}$ then we are in *Hybrid 2*. $\mathcal{B}$ will output 1 if $\mathcal{A}$ successfully forges. Therefore, if $\mathcal{A}$ have different advantages in hybrids *Hybrid 3* and *Hybrid 4* then $\mathcal{B}$ can attack the PPRF security with high probability i.e. $\boldsymbol{Adv}_{\mathcal{A}}^{Hyb\,3} - \boldsymbol{Adv}_{\mathcal{A}}^{Hyb\,4} = \boldsymbol{Adv}_{\mathcal{B}}^{PPRF} \leq \mathsf{negl}(\lambda)$ (Sect. 3.2). □

Lemma 7. *If $\mathcal{A}$'s advantage $\boldsymbol{Adv}_{\mathcal{A}}^{Hyb\,4}$ in* Hybrid 4 *is non-negligible, then there exists an adversary $\mathcal{B}$ who can break the security of one-way function f_1.*

Proof. We argue that if $\mathcal{A}$ successfully forges an ARS signature in *Hybrid 4* then we can construct an adversary $\mathcal{B}$ who can break the security of OWF f_1. Consider a scenario where the attacker $\mathcal{B}$ interacts with a OWF challenger while acting as a challenger for $\mathcal{A}$. First, $\mathcal{B}$ receives $(m^*, vk^*, R^*, ek^*, r^*)$ from $\mathcal{A}$ and submits it to the OWF challenger. Next, $\mathcal{B}$ receives a OWF challenge y^* and sets u^* as $u^* = y^*$ and continues to run the game in *Hybrid 4*. Now, if $\mathcal{A}$ successfully forges a signature on $(m^*\|vk^*\|\mathsf{Hash}(R^*)\|ek^*\|\mathsf{PRG}(r^*))$, then by definition it has computed a σ^* such that $f_1(\sigma^*) = u^*$. Therefore, if f_1 is secure, then $\mathcal{A}$ cannot forge with non-negligible advantage i.e. $\boldsymbol{Adv}_{\mathcal{A}}^{Hyb\,4} = \boldsymbol{Adv}_{\mathcal{B}}^{OWF} \leq \mathsf{negl}(\lambda)$ (Sect. 3.4). □

Hence, $\boldsymbol{Adv}_{\mathcal{A}}^{Unforge} = \boldsymbol{Adv}_{\mathcal{A}}^{Hyb\,0} - \boldsymbol{Adv}_{\mathcal{A}}^{Hyb\,4} \leq \mathsf{negl}(\lambda)$.

Theorem 2. *The ARS scheme of Sect. 4 is anonymous if the primitives of Sect. 3 satisfy their respective security properties.*

Proof. Our proof proceeds through a sequence of hybrids. The hybrids are chosen such that each successive hybrid game is indistinguishable from each other.

Hybrid 0. This hybrid is the *selective* variant of the *anonymity game* of Sect. 2.

1. **Setup Phase:** $\mathcal{C}$ runs the algorithm $\mathsf{ARS.UKGen}$ to generate signing- verification key pairs $\{(sk_i, vk_i)\}_{i\in[n]}$ for the users and registers them with the PKI. In addition, it also generates a private-public key pair (dk^*, ek^*) of an opener. Then, it sends $ek^*, \{(sk_i, vk_i)\}_{i\in[n]}$ to $\mathcal{A}$.
2. **Challenge Phase:** $\mathcal{A}$ submits a message m^*, a ring $R^* \subseteq \{vk_i\}_{i\in[n]}$ and two secret signing keys $sk_{i_0}^*, sk_{i_1}^* \in \{sk_i\}_{i\in[n]}, i_0 \neq i_1$, such that $vk_{i_0}^*, vk_{i_1}^* \in R^*$. Next, $\mathcal{C}$ chooses $b \leftarrow \{0,1\}$ and produces an ARS signature σ_{i_b} as follows:
   (a) Choose $r^* \leftarrow \{0,1\}^\lambda$ and let $t^* := \mathsf{PRG}(r^*)$ where $t^* \in \{0,1\}^{\ell_p}$.

(b) Set $\alpha_{i_b} := F_1(K_1, (m^* \| vk_{i_b} \| \mathsf{Hash}(R^*) \| ek^* \| t^*))$

(c) Set $\beta_{i_b} := F_2(K_2, \alpha_{i_b}) \oplus (m^* \| vk_{i_b} \| \mathsf{Hash}(R^*) \| ek^* \| t^*)$

(d) Let $\sigma_{i_b} := (\alpha_{i_b}, \beta_{i_b})$

Finally, $\mathcal{C}$ chooses keys K_1, K_2 for PPRFs F_1, F_2 respectively, and produces two obfuscated programs $\mathsf{Sign} = i\mathcal{O}(\mathcal{P}_S)$ and $\mathsf{Verify} = i\mathcal{O}(\mathcal{P}_V)$ (Figs. 1,2). Then, it sends these programs with the signature $\sigma_{i_b} = (\alpha_{i_b}, \beta_{i_b})$ to $\mathcal{A}$.

3. **Guess Phase:** $\mathcal{A}$ guesses b and outputs $b' \in \{0,1\}$. Let $\boldsymbol{Adv}_{\mathcal{A}}^{Hyb\,0}$ denote the advantage of $\mathcal{A}$ in this hybrid.

Hybrid 1. This hybrid is identical to *Hybrid 0* with the exception that in the *challenge phase* t^* is chosen at random from $\{0,1\}^{\ell_p}$. Let $\boldsymbol{Adv}_{\mathcal{A}}^{Hyb\,1}$ denote the advantage of $\mathcal{A}$ in this hybrid.

Hybrid 2. This hybrid is identical to *Hybrid 1* with the exception that, in the *challenge phase*, $\mathcal{C}$ gives out the two obfuscated programs as $\mathsf{Sign} = i\mathcal{O}(\mathcal{P}_S^\dagger)$ and $\mathsf{Verify} = i\mathcal{O}(\mathcal{P}_V^\dagger)$ (Figs. 9,10). The details of the modifications are as follows: $\mathcal{C}$ punctures the key K_2 at the point $\alpha_{i_b}^* = F_1(K_1, (m^* \| vk_{i_b}^* \| R^* \| ek^* \| t^*))$, and hardwires this punctured key $K_2(\{\alpha_{i_b}^*\})$ in $\mathcal{P}_S^\dagger$ and $\mathcal{P}_V^\dagger$. In addition, it also hardwires message m^*, verification key $vk_{i_b}^*$, ring R^*, public key ek^* and randomness t^* in $\mathcal{P}_V^\dagger$. Furthermore, $\mathcal{P}_V^\dagger$ is modified to output c_b^* if $(m, \sigma, R, ek) = (m^*, \sigma_b^*, R^*, ek^*)$. Let $\boldsymbol{Adv}_{\mathcal{A}}^{Hyb\,1}$ denote the advantage of $\mathcal{A}$ in this hybrid.

```
$\mathcal{P}_S^\dagger$
Hardwired: Keys $K_1, K_2(\{\alpha_{i_b}^*\})$
Input: Message $m$, key pair $(sk, vk)$, ring $R$, key $ek$ and randomness $r$
if $(f(sk) = vk \wedge f(sk) \in R)$ then
    Set $\alpha := F_1(K_1, (m \| vk \| \mathsf{Hash}(R) \| ek \| \mathsf{PRG}(r)))$
    Set $\beta := F_2(K_2, \alpha) \oplus (m \| vk \| \mathsf{Hash}(R) \| ek \| \mathsf{PRG}(r))$
    return $(\sigma := (\alpha, \beta))$
else
    return $\perp$
```

Figure 9: The circuit Sign

```
$\mathcal{P}_V^{\dagger}$
Hardwired: Keys $K_1, K_2(\{\alpha_{i_b}^*\})$ and values $m^*, vk_{i_b}^*, R^*, ek^*, t^*$
Input: Message $m$, signature $\sigma = (\alpha, \beta)$, ring $R$ and opener's public key $ek$
1 if $\Big(m = m^* \wedge R = R^* \wedge ek = ek^* \wedge f_1(\alpha) = f_1(F_1(K_1, (m^* \| vk_{i_b}^* \| \mathit{Hash}(R^*) \| ek^* \| t^*)))\Big)$ then
2     return $(c_b^* := \mathsf{PKE.Encrypt}(ek, vk_{i_b}^*; t^*))$
3 else if $\big(F_2(K_2, \alpha) \oplus \beta = (m' \| vk' \| h' \| ek' \| r') \wedge m = m' \wedge \mathsf{Hash}(R) = h' \wedge ek = ek' \wedge f_1(\alpha) = f_1(F_1(K_1, (m' \| vk' \| h' \| ek' \| r')))\big)$ then
4     return $(c := \mathsf{PKE.Encrypt}(ek, vk'; r'))$
5 return $\perp$
```

Figure 10: The circuit Verify

Hybrid 3. This hybrid is identical to *Hybrid 2* with the exception that in the *challenge phase* $\beta_{i_b}^*$ is chosen uniformly at random from the set $\{0,1\}^{\ell_2}$. Let $\boldsymbol{Adv}_{\mathcal{A}}^{Hyb\,3}$ denote the advantage of $\mathcal{A}$ in this hybrid.

Hybrid 4. This hybrid is identical to *Hybrid 3* with the exception that in the *challenge phase*, $\mathcal{C}$ gives out the two obfuscated programs as $\mathsf{Sign} = i\mathcal{O}(\mathcal{P}_S^{\ddagger})$ and $\mathsf{Verify} = i\mathcal{O}(\mathcal{P}_V^{\ddagger})$ (Figs. 11,12). The details of the modifications are as follows: $\mathcal{C}$ punctures the key K_1 at point $(m^* \| vk_{i_b}^* \| \mathsf{Hash}(R) \| ek^* \| t^*)$, and hardwires this punctured key $K_1(\{(m^* \| vk_{i_b}^* \| \mathsf{Hash}(R) \| ek^* \| t^*)\})$ in programs $\mathcal{P}_S^{\ddagger}$ and $\mathcal{P}_V^{\ddagger}$. In addition, it also hardwires message m^*, ring R^*, public key ek^*, and value $\alpha_{i_b}^* = F_1(K_1, (m^* \| vk_{i_b}^* \| \mathsf{Hash}(R) \| ek^* \| t^*))$ in program $\mathcal{P}_V^{\ddagger}$. Let $\boldsymbol{Adv}_{\mathcal{A}}^{Hyb\,4}$ denote the advantage of $\mathcal{A}$ in this hybrid.

```
$\mathcal{P}_S^{\ddagger}$
Hardwired: Keys $K_1(\{(m^* \| vk_b^* \| \mathsf{Hash}(R^*) \| ek^* \| t^*)\}), K_2(\{\alpha_{i_b}^*\})$
Input: Message $m$, key pair $(sk, vk)$, ring $R$, key $ek$ and randomness $r$
1 if $(f(sk) = vk \wedge f(sk) \in R)$ then
2     Set $\alpha := F_1(K_1, (m \| vk \| \mathsf{Hash}(R) \| ek \| \mathsf{PRG}(r)))$
3     Set $\beta := F_2(K_2, \alpha) \oplus (m \| vk \| \mathsf{Hash}(R) \| ek \| \mathsf{PRG}(r))$
4     return $(\sigma := (\alpha, \beta))$
5 else
6     return $\perp$
```

Figure 11: The circuit Sign

```
                        $\mathcal{P}_V^{\ddagger}$
Hardwired: Keys $K_1(\{(m^*\|vk_b^*\|\mathsf{Hash}(R^*)\|ek^*\|t^*)\}), K_2(\{\alpha_{i_b}^*\})$ and values
           $m^*, R^*, ek^*, t^*, \alpha_{i_b}^*$
Input: Message $m$, signature $\sigma = (\alpha, \beta)$, ring $R$ and opener's public key $ek$
1 if $\left(m = m^* \wedge R = R^* \wedge ek = ek^* \wedge f_1(\alpha) = f_1(\alpha_{i_b}^*)\right)$ then
2 |   return $(c_b^* := \mathsf{PKE.Encrypt}(ek, vk_{i_b}^*; t^*))$
3 else if $\big(F_2(K_2, \alpha) \oplus \beta = (m'\|vk'\|h'\|ek'\|r') \wedge m = m' \wedge \mathsf{Hash}(R) =$
      $h' \wedge ek = ek' \wedge f_1(\alpha) = f_1(F_1(K_1, (m'\|vk'\|h'\|ek'\|r')))\big)$ then
4 |   return $(c := \mathsf{PKE.Encrypt}(ek, vk'; r'))$
5 return $\perp$
```

Figure 12: The circuit Verify

Hybrid 5. This hybrid is identical to *Hybrid 4* with the exception that in the *challenge phase*, $\alpha_{i_b}^*$ is chosen at random from $\{0,1\}^{\ell_1}$ instead of computing them using PPRF. Let $\boldsymbol{Adv}_{\mathcal{A}}^{Hyb\,5}$ denote the advantage of $\mathcal{A}$ in this hybrid.

The indistinguishability of successive hybrid games (discussed above) is shown in Lemmas. 8, 9, 10, 11, 12, 13.

Lemma 8. *If $\mathcal{A}$ can distinguish* Hybrid 0 *from* Hybrid 1 *in the* anonymity game*, i.e. $\boldsymbol{Adv}_{\mathcal{A}}^{Hyb\,0} - \boldsymbol{Adv}_{\mathcal{A}}^{Hyb\,1}$ is non-negligible then there exists an adversary $\mathcal{B}$ who can break the security of pseudo-random generator* PRG*.*

Proof. We argue that if there is a non-negligible difference in advantages of $\mathcal{A}$ in hybrids *Hybrid 0* and *Hybrid 1* then we can construct an adversary $\mathcal{B}$ that breaks the security of PRG. Now, consider a scenario where the attacker $\mathcal{B}$ interacts with PRG challenger while acting as a challenger for $\mathcal{A}$. $\mathcal{B}$ receives a challenge y^* from the PRG challenger, plugs in $t^* = y^*$ in *Hybrid 0* and plays the rest of the game. Note that if the PRG challenger computes $t^* = \mathsf{PRG}(r^*), r^* \leftarrow \{0,1\}^{\lambda}$ then we are in *Hybrid 0*. If it chooses $t^* \leftarrow \{0,1\}^{\ell_p}$ then we are in *Hybrid 1*. $\mathcal{B}$ will output 1 if $\mathcal{A}$ successfully guesses. Therefore, if $\mathcal{A}$ have different advantages in hybrids *Hybrid 0* and *Hybrid 1* then $\mathcal{B}$ can attack $i\mathcal{O}$ security with high probability i.e. $\boldsymbol{Adv}_{\mathcal{A}}^{Hyb\,0} - \boldsymbol{Adv}_{\mathcal{A}}^{Hyb\,1} = \boldsymbol{Adv}_{\mathcal{B}}^{PRG} \leq \mathsf{negl}(\lambda)$ (Sect. 3.5). □

Lemma 9. *If $\mathcal{A}$ can distinguish* Hybrid 1 *from* Hybrid 2 *in the* anonymity game*, i.e. $\boldsymbol{Adv}_{\mathcal{A}}^{Hyb\,1} - \boldsymbol{Adv}_{\mathcal{A}}^{Hyb\,2}$ is non-negligible then there exists an adversary $\mathcal{B}$ who can break the security of $i\mathcal{O}$.*

Proof. We argue that if there is a non-negligible difference in advantages of $\mathcal{A}$ in hybrids *Hybrid 1* and *Hybrid 2* then we can construct an adversary $\mathcal{B}$ that breaks the security of $i\mathcal{O}$. Firstly, observe that the programs $\mathcal{P}_S$ and $\mathcal{P}_S^{\dagger}$ as well as the programs $\mathcal{P}_V$ and $\mathcal{P}_V^{\dagger}$, are functionally equivalent. This is because when t^* is chosen at random, with probability $1 - \frac{1}{2^{\lambda}}$, t^* is not in the image of the PRG. Therefore neither program will evaluate. Thus, puncturing excising t^* out from the key will not make a difference in input/ output behavior. Now, consider a scenario where the attacker $\mathcal{B}$ interacts with an $i\mathcal{O}$ challenger while acting as a challenger for $\mathcal{A}$. $\mathcal{B}$ submits the program pairs $P_0 = (\mathcal{P}_S, \mathcal{P}_V)$ and $P_1 = (\mathcal{P}_S^{\dagger}, \mathcal{P}_V^{\dagger})$ to the $i\mathcal{O}$ challenger. The $i\mathcal{O}$ challenger chooses $b \leftarrow \{0,1\}$ at random, obfuscates the programs in P_b , and returns the output to $\mathcal{B}$. $\mathcal{B}$ plugs in these obfuscated programs in *Hybrid 0* and plays the rest of the game. Note that if the $i\mathcal{O}$ challenger chooses P_0 then we are in *Hybrid 1*. If it chooses P_1 then we are in *Hybrid 2*. $\mathcal{B}$ will output 1 if $\mathcal{A}$ successfully guesses. Hence, if $\mathcal{A}$ have different advantages in hybrids *Hybrid 1* and *Hybrid 2* then $\mathcal{B}$ can attack $i\mathcal{O}$ security with high probability i.e. $\boldsymbol{Adv}_{\mathcal{A}}^{Hyb\,1} - \boldsymbol{Adv}_{\mathcal{A}}^{Hyb\,2} = \boldsymbol{Adv}_{\mathcal{B}}^{i\mathcal{O}} \leq \mathsf{negl}(\lambda)$ (Sect. 3.1). □

Lemma 10. *If $\mathcal{A}$ can distinguish* Hybrid 2 *from* Hybrid 3 *in the* anonymity game*, i.e.* $\boldsymbol{Adv}_{\mathcal{A}}^{Hyb\,2} - \boldsymbol{Adv}_{\mathcal{A}}^{Hyb\,3}$ *is non-negligible then there exists an adversary $\mathcal{B}$ who can break the security of* PPRF F_2.

Proof. We argue that if there is a non-negligible difference in advantages of $\mathcal{A}$ in hybrids *Hybrid 2* and *Hybrid 3* then we can construct an adversary $\mathcal{B}$ who can break the security of PPRF F_2. Consider a scenario where the attacker $\mathcal{B}$ interacts with a PPRF challenger while acting as a challenger for $\mathcal{A}$. First, $\mathcal{B}$ receives $(m^*, R^*, sk_{i_0}^*, sk_{i_1}^*)$ from $\mathcal{A}$. Then, it chooses $b \leftarrow \{0,1\}, K_1 \leftarrow \{0,1\}^{\lambda}, t^* \leftarrow \{0,1\}^{\ell_p}$, computes $\alpha_{i_b}^* = F_1(K_1, (m^* \| vk_{i_b}^* \| R^* \| ek^* \| t^*))$, and submits $\alpha_{i_b}^*$ to the PPRF challenger. Next, the PPRF challenger choses $K_2 \leftarrow \{0,1\}^{\lambda}$, punctures K_2 at point $\alpha_{i_b}^*$, and sends this punctured PRF key $K_2(\{\alpha_{i_b}^*\})$ along with a challenge $w^* \in \{0,1\}^{\ell_2}$ to $\mathcal{B}$. Then, $\mathcal{B}$ continues to run the game in *Hybrid 2* except that it plugs in $\beta_{i_b}^*$ as $\beta_{i_b}^* = z_{i_b}^* \oplus (m^* \| vk_{i_b}^* \| R^* \| ek^* \| t^*)$. Note that if the PPRF challenger outputs $z_{i_b}^*$ as $z_{i_b}^* = F_2(K_2, \alpha_{i_b}^*)$ then we are in *Hybrid 2*. If it outputs z^* as $z_{i_b}^* \leftarrow \{0,1\}^{\ell_2}$ then we are in *Hybrid 3*. $\mathcal{B}$ will output 1 if $\mathcal{A}$ successfully guesses. Therefore, if $\mathcal{A}$ have different advantages in hybrids *Hybrid 2* and *Hybrid 3* then $\mathcal{B}$ can attack the PPRF security with high probability i.e. $\boldsymbol{Adv}_{\mathcal{A}}^{Hyb\,2} - \boldsymbol{Adv}_{\mathcal{A}}^{Hyb\,3} = \boldsymbol{Adv}_{\mathcal{B}}^{PPRF} \leq \mathsf{negl}(\lambda)$ (Sect. 3.2). □

Lemma 11. *If $\mathcal{A}$ can distinguish* Hybrid 3 *from* Hybrid 4 *in the* anonymity game*, i.e.* $\boldsymbol{Adv}_{\mathcal{A}}^{Hyb\,3} - \boldsymbol{Adv}_{\mathcal{A}}^{Hyb\,4}$ *is non-negligible then there exists an adversary $\mathcal{B}$ who can break the security of $i\mathcal{O}$.*

Proof. We argue that if there is a non-negligible difference in advantages of $\mathcal{A}$ in hybrids *Hybrid 3* and *Hybrid 4* then we can construct an adversary $\mathcal{B}$ that breaks the security of $i\mathcal{O}$. Firstly, observe that the programs $\mathcal{P}_S^{\dagger}$ and $\mathcal{P}_S^{\ddagger}$ as well as the programs $\mathcal{P}_V^{\dagger}$ and $\mathcal{P}_V^{\ddagger}$, are functionally equivalent. Now, consider a scenario where the attacker $\mathcal{B}$ interacts with an $i\mathcal{O}$ challenger while acting as a challenger for $\mathcal{A}$. $\mathcal{B}$ submits the program pairs $P_0 = (\mathcal{P}_S^{\dagger}, \mathcal{P}_V^{\dagger})$ and $P_1 = (\mathcal{P}_S^{\ddagger}, \mathcal{P}_V^{\ddagger})$ to the $i\mathcal{O}$ challenger. The $i\mathcal{O}$ challenger chooses $b \leftarrow \{0,1\}$ at random, obfuscates the programs in P_b , and returns the output to $\mathcal{B}$. $\mathcal{B}$ plugs in these obfuscated programs in *Hybrid 3* and plays the rest of the game. Note that if the $i\mathcal{O}$ challenger chooses P_0 then we are in *Hybrid 3*. If it chooses P_1 then we are in *Hybrid 4*. $\mathcal{B}$ will output 1 if $\mathcal{A}$ successfully guesses. Therefore, if $\mathcal{A}$ have different advantages in hybrids *Hybrid 3* and *Hybrid 4* then $\mathcal{B}$ can attack $i\mathcal{O}$ security with high probability i.e. $\boldsymbol{Adv}_{\mathcal{A}}^{Hyb\,3} - \boldsymbol{Adv}_{\mathcal{A}}^{Hyb\,4} = \boldsymbol{Adv}_{\mathcal{B}}^{i\mathcal{O}} \leq \mathsf{negl}(\lambda)$ (Sect. 3.1). □

Lemma 12. *If $\mathcal{A}$ can distinguish* Hybrid 4 *from* Hybrid 5 *in the* anonymity game*, i.e.* $\boldsymbol{Adv}_{\mathcal{A}}^{Hyb\,4} - \boldsymbol{Adv}_{\mathcal{A}}^{Hyb\,5}$ *is non-negligible then there exists an adversary $\mathcal{B}$ who can break the security of* PPRF F_1*.*

Proof. We argue that if there is a non-negligible difference in advantages of $\mathcal{A}$ in hybrids *Hybrid 4* and *Hybrid 5* then we can construct an adversary $\mathcal{B}$ who can break the security of PPRF F_1. Consider a scenario where the attacker $\mathcal{B}$ interacts with a PPRF challenger while acting as a challenger for $\mathcal{A}$. First, $\mathcal{B}$ receives $(m^*, R^*, sk_{i_0}^*, sk_{i_1}^*)$ from $\mathcal{A}$. Then, it chooses $b \leftarrow \{0,1\}, t \leftarrow \{0,1\}^{\ell_p}$ and submits $(m^*, R^*, vk_{i_b}^*, t^*)$ to the PPRF challenger. The PPRF challenger chooses a PPRF key K_1 at random and punctures K_1 at point $(m^*\|vk_{i_b}^*\|R^*\|ek^*\|t^*)$ and sends this punctured key $K_1(\{(m^*\|vk_{i_b}^*\|R^*\|ek^*\|t^*)\})$ and a challenge z^* to $\mathcal{B}$. Then, $\mathcal{B}$ continues to run the game in *Hybrid 4* except it plugs in $\alpha_{i_b}^* = z^*$. Note that if the PPRF challenger outputs z^* as $z^* = F_1(K_1, (m^*\|vk_{i_b}^*\|R^*\|ek^*\|t^*))$ then we are in *Hybrid 4*. If it outputs z^* as $z^* \leftarrow \{0,1\}^{\ell_1}$ then we are in *Hybrid 5*. $\mathcal{B}$ will output 1 if $\mathcal{A}$ successfully guesses. Therefore, if $\mathcal{A}$ has different advantages in hybrids *Hybrid 4* and *Hybrid 5* then $\mathcal{B}$ can attack the PPRF security with high probability i.e. $\boldsymbol{Adv}_{\mathcal{A}}^{Hyb\,4} - \boldsymbol{Adv}_{\mathcal{A}}^{Hyb\,5} = \boldsymbol{Adv}_{\mathcal{B}}^{PPRF} \leq \mathsf{negl}(\lambda)$ (Sect. 3.2). □

Lemma 13. *$\mathcal{A}$'s advantage in* Hybrid 5 *is zero.*

Proof. We observe that the variables $\alpha_{i_b}^*$ and $\beta_{i_b}^*$ in *Hybrid 5* are both independently and uniformly chosen random strings. Thus, the distributions output by this hybrid for $b = 0$ and $b = 1$ are identical, and therefore even an unbounded adversary could have no advantage in distinguishing them i.e. $\boldsymbol{Adv}_{\mathcal{A}}^{Hyb\,5} = 0$. □

Hence, $\boldsymbol{Adv}_{\mathcal{A}}^{Anony} = \boldsymbol{Adv}_{\mathcal{A}}^{Hyb\,0} - \boldsymbol{Adv}_{\mathcal{A}}^{Hyb\,5} \leq \mathsf{negl}(\lambda)$. □

Theorem 3. *The ARS scheme of Sect. 4 is traceable and traceably sound if the primitives of Sect. 3 satisfy their respective security properties.*

Proof. In order to prove this theorem it is enough to prove that the proof system between the opener (prover) and the judge (verifier) is a *non-interactive zero knowledge* (NIZK) proof system where *traceability* and *tracing soundness* respectively correspond to *completeness* and *soundness* properties of NIZK.

Lemma 14 ([19]). *The* NIZK *proof system between* $\mathsf{ARS.Open}$ *and* $\mathsf{ARS.Judge}$ *with common reference string* $\mathsf{crs} = (\mathsf{NIZKprove}, \mathsf{NIZKverify})$ *is secure if $i\mathcal{O}$ is secure, F_3 is a secure* PPRF*, and f_2 is a secure one-way function.*

6 Conclusion

In this paper we proposed a new construction for accountable ring signature (ARS). This is the first indistinguishability obfuscation-based ARS scheme. Also, the signature-size of our scheme is constant, and does not vary with the size of the ring. We have provided proof of security in the standard model. Our construction can be viewed as a new application of *puncturable programming* and *hidden sparse trigger* techniques introduced by Sahai and Waters. We leave the issue of constructing fully unforgeable, fully anonymous, constant size ARS scheme in standard model as an open problem.

Acknowledgement First author is supported by Tata Consultancy Services (TCS) research fellowship. We thank anonymous reviewers for their constructive comments.

References

1. Barak, B., Goldreich, O., Impagliazzo, R., Rudich, S., Sahai, A., Vadhan, S., Yang, K.: On the (im) possibility of obfuscating programs. In: Annual International Cryptology Conference. pp. 1–18. Springer (2001)
2. Boneh, D., Ishai, Y., Sahai, A., Wu, D.J.: Lattice-based snargs and their application to more efficient obfuscation. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 247–277. Springer (2017)
3. Boneh, D., Waters, B.: Constrained pseudorandom functions and their applications. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 280–300. Springer (2013)
4. Bootle, J., Cerulli, A., Chaidos, P., Ghadafi, E., Groth, J., Petit, C.: Short accountable ring signatures based on ddh. In: European Symposium on Research in Computer Security. pp. 243–265. Springer (2015)
5. Bose, P., Das, D., Rangan, C.P.: Constant size ring signature without random oracle. In: Australasian Conference on Information Security and Privacy. pp. 230–247. Springer (2015)
6. Boyle, E., Goldwasser, S., Ivan, I.: Functional signatures and pseudorandom functions. In: Proceedings of the 17th International Conference on Public-Key Cryptography—PKC 2014-Volume 8383. pp. 501–519. Springer-Verlag New York, Inc. (2014)
7. Camenisch, J., Haralambiev, K., Kohlweiss, M., Lapon, J., Naessens, V.: Structure preserving cca secure encryption and applications. In: Asiacrypt. vol. 7073, pp. 89–106. Springer (2011)
8. Camenisch, J., et al.: Efficient and generalized group signatures. In: Eurocrypt. vol. 97, pp. 465–479. Springer (1997)
9. Chaum, D., Van Heyst, E.: Group signatures. In: Proceedings of the 10th Annual International Conference on Theory and Application of Cryptographic Techniques. pp. 257–265. Springer-Verlag (1991)
10. Coron, J.S., Lepoint, T., Tibouchi, M.: Practical multilinear maps over the integers. In: Advances in Cryptology–CRYPTO 2013, pp. 476–493. Springer (2013)
11. Garg, S., Gentry, C., Halevi, S., Raykova, M., Sahai, A., Waters, B.: Candidate indistinguishability obfuscation and functional encryption for all circuits. In: Foundations of Computer Science (FOCS), 2013 IEEE 54th Annual Symposium on. pp. 40–49. IEEE (2013)
12. Goldreich, O.: Foundations of cryptography: volume 2, basic applications. Cambridge university press (2009)

13. Goldreich, O., Goldwasser, S., Micali, S.: How to construct random functions. Journal of the ACM (JACM) 33(4), 792–807 (1986)
14. Goldwasser, S., Micali, S.: Probabilistic encryption & how to play mental poker keeping secret all partial information. In: Proceedings of the fourteenth annual ACM symposium on Theory of computing. pp. 365–377. ACM (1982)
15. Groth, J., Kohlweiss, M.: One-out-of-many proofs: Or how to leak a secret and spend a coin. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 253–280. Springer (2015)
16. Kiayias, A., Papadopoulos, S., Triandopoulos, N., Zacharias, T.: Delegatable pseudorandom functions and applications. In: Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security. pp. 669–684. ACM (2013)
17. Lai, R.W., Zhang, T., Chow, S.S., Schröder, D.: Efficient sanitizable signatures without random oracles. In: European Symposium on Research in Computer Security. pp. 363–380. Springer (2016)
18. Rivest, R.L., Shamir, A., Tauman, Y.: How to leak a secret. In: Proceedings of the 7th International Conference on the Theory and Application of Cryptology and Information Security: Advances in Cryptology. pp. 552–565. Springer-Verlag (2001)
19. Sahai, A., Waters, B.: How to use indistinguishability obfuscation: deniable encryption, and more. In: Proceedings of the forty-sixth annual ACM symposium on Theory of computing. pp. 475–484. ACM (2014)
20. Xu, S., Yung, M.: Accountable ring signatures: A smart card approach. Smart Card Research and Advanced Applications VI pp. 271–286 (2004)