Ring Signatures with User-Controlled Linkability

Dario Fiore[1], Lydia Garms[1,2], Dimitris Kolonelos[1,3], Claudio Soriente[4], and Ida Tucker[5]

[1] IMDEA Software Institute, Madrid, Spain
[2] Keyless Technologies Limited
[3] Universidad Politecnica de Madrid, Spain
[4] NEC Laboratories Europe
[5] Zondax AG

Abstract. Anonymous authentication primitives, e.g., group or ring signatures, allow one to realize privacy-preserving data collection applications, as they strike a balance between authenticity of data being collected and privacy of data providers. At PKC 2021, Diaz and Lehmann defined group signatures with User-Controlled Linkability (UCL) and provided an instantiation based on BBS+ signatures. In a nutshell, a signer of a UCL group signature scheme can link any of her signatures: linking evidence can be produced at signature time, or after signatures have been output, by providing an explicit linking proof.

In this paper, we introduce Ring Signatures with User-Controlled Linkability (RS-UCL). Compared to group signatures with user-controlled linkability, RS-UCL require no group manager and can be instantiated in a completely decentralized manner. We also introduce a variation, User Controlled and Autonomous Linkability (RS-UCAL), which gives the user full control of the linkability of their signatures.

We provide a formal model for both RS-UCL and RS-UCAL and introduce a compiler that can upgrade any ring signature scheme to RS-UCAL. The compiler leverages a new primitive we call Anonymous Key Randomizable Signatures (AKRS) — a signature scheme where the verification key can be randomized — that can be of independent interest. We also provide different instantiations of AKRS based on Schnorr signatures and on lattices. Finally, we show that an AKRS scheme can additionally be used to construct an RS-UCL scheme.

1 Introduction

Group signatures [CvH91,BMW03,BSZ05,BCC+16] and ring signatures [RST01] allow users to sign messages, while providing anonymity of signers and unlinkability of signatures. Group signatures require a central entity that manages group membership, whereas ring signatures allow a signer to choose an arbitrary ring of public keys, and sign on behalf of that ring.

Many application scenarios do not require full anonymity/ unlinkability, and may actually ask for mechanisms to identify the signer or link signatures produced by the same party. Group signatures feature an opening authority that can de-anonymize signers and thereby test if two signatures have the same signer.

Recently, researchers have proposed more flexible linkability options for group signature schemes, where even less power is entrusted to the opener. In [HLC+11,HLC+13,SSU14,KTY04] group signatures with an authority who can test whether two signatures have the same signer but cannot open signatures were introduced. In [GL19,FGL21], group signatures are originally unlinkable, but later on can be converted to linkable signatures by an oblivious "converter". Group signatures with message-dependant opening [SEH+12,OSEH13,LJ14,LMN16] introduce an additional entity, the admitter, who can specify messages such that the corresponding signatures can be opened. Group signatures with certified limited opening [ZWC19] introduce a certifier, instead of an admitter, who can certify a particular opener to allow them to open signatures on messages within a particular context. Abe

et al., [ACHO13] use "public-key anonymous tag system" to build a traceable signature scheme. In all these lines of work, linking is performed by a trusted party, and signers have no say in which of their signatures can be linked together. Another line of work [BCC04,BFG+13,CDL16b,CDL16a] considers scenarios where a so-powerful authority is undesirable, and achieves linkability by including a one-way function of the signing key and a "scope" chosen by the signer — also known as "pseudonym" — in each signature, so that two signatures with the same pseudonym can be trivially linked.

Pseudonym-based linkability, however, require signers to decide at signature time whether their signatures should be ever linked: two signatures using different scopes – hence, with different pseudonyms – would be unlinkable by definition. Recently, Diaz and Lehmann [DL21] introduced group signatures with User-Controlled Linkability (UCL) that provide pseudonym-based linkability (labelled "implicit" linkability), but also allow a signer to link any set of her signatures generated with the same linking secret, even if those had different pseudonyms (labelled "explicit" linkability). This linking model turns useful in applications where authenticated data is collected in anonymous fashion but, later on, one may be interested to link specific data items. For example, in smart-metering applications, energy consumptions may be collected in a fully anonymous way, while, at a later time, a user may want to link her measurements to, e.g., receive tailored offers from the energy providers. Similarly, connected vehicles can anonymously report their mobility traces but, at a later time, a driver may want to link her reports so to obtain discounts from insurance companies.

Our contributions In this paper, we continue the study of UCL but focus on ring signatures. Compared to group signatures, a ring signature scheme enables fully decentralized applications as no group manager is needed. Previous linkable ring signatures [LWW04,SALY17] solely allow anyone to link two signatures by the same signer on the same ring.

Our first contribution is the formalization of UCL in ring signatures. We introduce the first formal model for ring signatures with UCL allowing for both implicit and explicit linkability, as in [DL21]. Next, we introduce ring signatures with *User Controlled Autonomous Linkability* (UCAL) to give signers full control over the linkability of their signatures. Different from UCL, UCAL does not use the signing key to create pseudonyms, but instead uses a "linking secret" which can be re-used or chosen afresh. A fresh linking secret ensures that signatures cannot be linked (i.e., via implicit linkability) even if they have the same scope. Of course, a signer can use the same linking secret on multiple signatures with the same scope so to provide implicit linkability. Ultimately, a signer can prove linkability of any set of her signatures generated with the same linking secret, even if those had different pseudonyms. Further, using different linking secrets ensures that past signatures cannot be linked even if the signer is corrupted, providing a form of forward anonymity.

As a second contribution we propose constructions of UCL and UCAL ring signatures. To do so, we introduce a new cryptographic primitive that we label *Anonymous Key Randomizable Signatures* (AKRS) that may be of independent interest. AKRS is essentially a signature scheme where public keys can be re-randomized, while maintaining the correspondence to the same secret key, so that a randomized public key cannot be linked to the original one. However, by using the secret, the signer can prove that multiple public keys are all randomized versions of the original one. The primitive is in a similar spirit to Signatures with Flexible Public Key [BHKS18], which however is not fully suitable for our requirements. We elaborate more on the differences in section 4.

We show that AKRS can be used to upgrade *any ring signature scheme* to UCAL. In particular, we use an AKRS public key that has been re-randomised with the scope as the pseudonym for the (ring) signature. By using a public key corresponding to the same AKRS secret key and scope, we

provide implicit linkability. Otherwise, the signer may use a different AKRS secret key to make two signatures unlinkable even on the same scope. At a later time, the signer can use her AKRS secret key to link the pseudonyms of a set of ring signatures, thereby proving that all such signatures are linked. Notably, our construction does not modify the original ring signature public keys and thus can be used to upgrade to UCAL an already up-and-running system using ring signatures.

We also show how to use AKRS, along with a NIZK, to build a UCL ring signature. The linking mechanism is obtained using the AKRS in a similar way to the UCAL construction. The difference is that, for UCL the AKRS secret key is used as the ring signature secret. Therefore, to sign a message with some scope, in addition to using the AKRS secret key to compute the pseudonym, we also add a non-interactive signature of knowledge [CS97] that the secret used to derive the pseudonym corresponds to one of the public keys in the ring.

Finally, we propose two instantiations of AKRS: one based on Schnorr's signatures in prime order groups, and one based on Lyubashevsky's signatures [Lyu12] on lattices. Compiling our AKRS with a ring signature scheme we get UCAL ring signatures with minimal overhead: one additional Schnorr or Lyubashevsky signature, respectively. In contrast to previous works (on group signatures) [DL21], the constructions are generic and can bootstrap any arbitrary ring signature scheme to a UCAL one. For instance we can achieve UCAL without pairings.

Related Work There is an extensive line of work studying and constructing efficient ring signatures from various settings and assumptions, with the today's state-of-the art achieving signatures of size logarithmic in the size of the ring [GK15,BCC$^+$15,LPQ18,LRR$^+$19,YSL$^+$20,YEL$^+$21].

Park and Sealfon [PS19] proposed the related notions of (un)Claimable and (un)Repudiable Ring Signatures. Claimability states that one can always claim a signature after she signed it, while Repudiability states the opposite, that one can always repudiate a signature she did not sign. Claimability is a notion relevant to user-controlled linkability, although it is weaker: a signer can claim a signature by linking it to a signature of a dummy message on-the-fly. The inverse, achieving user-controlled linkability from claimability, does not apply.

The idea of linking anonymous signatures of a user by using a Pseudorandom Function (PRF) has been used in the past. A user can additionally sign a random value together with its PRF output, where the seed is kept by the signer. Then, a NIZK proving knowledge of the (same) seed can be used to link signatures. However, this linking mechanism provides no succinctness: the linking proof typically grows with the number of linked signatures. AKRS generalizes this linking mechanism and guarantees succinctness; we note that an AKRS may also be instantiated with a PRF and a (succinct) NIZK.

2 Standard ring signatures

We now provide definitions for standard ring signatures following notation from [PS19].

Definition 2.1 (Ring signature). *A* ring signature scheme *is a triple of PPT algorithms* $(\mathsf{KGen}, \mathsf{Sig}, \mathsf{Vf})$*, satisfying* correctness *(Def. 2.2),* anonymity *(Def. 2.3) and* unforgeability *(Def. 2.4); and with the following syntax:*

$\mathsf{KGen}(1^\lambda)$ *on input a security parameter* 1^λ *outputs a signing key* sk *and a verification key* vk*.*
$\mathsf{Sig}(\mathsf{sk}, R, m)$ *on input a signing key* sk*, a message* m*, and a set of verification keys (called* the ring*)* $R = \{\mathsf{vk}_1, \ldots, \mathsf{vk}_n\}$*, the signing algorithm outputs a signature* σ*.*

$\mathsf{Vf}(\sigma, R, m)$ *on input a signature* σ*, a set of verification keys* $R = \{\mathsf{vk}_1, \ldots, \mathsf{vk}_n\}$*, and a message* m*, the verification algorithm returns 1 if* σ *is a valid signature on* m *w.r.t.* R*, and 0 otherwise.*

Definition 2.2 (Correctness). *A* ring signature scheme *satisfies* correctness *if for any* $n = \mathsf{poly}(\lambda)$*, any* $\{(\mathsf{vk}_i, \mathsf{sk}_i)\}_{i\in[n]} \leftarrow \mathsf{KGen}(1^\lambda)$*, any* $i \in [n]$*, and any message* m*, there exists a negligible function* ϵ *such that:*

$$\Pr[\mathsf{Vf}(\mathsf{Sig}(\mathsf{sk}_i, R, m), R, m) = 1] = 1 - \epsilon(\lambda),$$

where $R = \{\mathsf{vk}_1, \ldots, \mathsf{vk}_n\}$*. If* $\epsilon = 0$ *then* perfect correctness *is satisfied.*

2.1 Security notions for ring signatures

Before providing formal definitions for anonymity and unforgeability, we introduce a number of oracles that will be used in the security experiments.

Oracles. All oracles are parametrised by a list of keys pairs $\{(\mathsf{vk}_i, \mathsf{sk}_i)\}_{i\in[n]}$, where $n = \mathsf{poly}(\lambda)$.

Corruption oracle Corr**:** Parametrised by the list I of corrupted indices (queried to Corr). The corruption oracle Corr takes input an index $i \in [n]$, adds i to the list of corrupted indices $I \leftarrow I \cup \{i\}$, and outputs the randomness w_i used to compute key pair $(\mathsf{sk}_i, \mathsf{vk}_i)$.

Signing oracle OSign**:** Parametrised by the lists $\mathsf{SIG}[i]$ for $i \in [n]$ of signature tuples produced by OSign for user index i. The oracle OSign on input a set R, an index $i \in [n]$ and a message m computes $\sigma \leftarrow \mathsf{Sig}(\mathsf{sk}_i, R \cup \mathsf{vk}_i, m)$, adds (σ, m, R) to the list of signatures signed by user index i: $\mathsf{SIG}[i] \leftarrow \mathsf{SIG}[i] \cup \{(\sigma, m, R)\}$, and returns σ.

Challenge oracle $\mathsf{Ch} - \mathsf{Sign}_b$**:** The oracle $\mathsf{Ch} - \mathsf{Sign}_b$ on input challenge identities $\{i_0, i_1\}$, a set R and a message m computes $\sigma \leftarrow \mathsf{Sig}(\mathsf{sk}_{i_b}, R \cup \{\mathsf{vk}_{i_0}, \mathsf{vk}_{i_1}\}, m)$ and outputs σ.

Anonymity There are a series of notions of anonymity defined for ring signatures, of varying strength. The natural *adaptive anonymity against adversarially chosen keys*, defined by [BKM09], requires that an adversary who controls up to all but two parties in a ring, and who may produce its' own malformed key pairs as well as corrupt honest parties' keys, cannot do significantly better than randomly guess which of the honest parties produced a given signature. The stronger notion of *anonymity against full key exposure* requires that even if an adversary compromises every single party in a ring, the adversary cannot identify the signers of past signatures. We hence use the modular definition of $(\mathcal{O}, \alpha)$-anonymity adopted in [PS19], which allows to capture these varying degrees of anonymity.

The security notions are captured in Figure 1.

Definition 2.3 ($(\mathcal{O}, \alpha)$-anonymity). *Let* $\alpha \in \{0, 1, 2\}$*, and let* $\mathcal{O}$ *be a set of oracles. A ring signature scheme satisfies* $(\mathcal{O}, \alpha)$-anonymity *if for any PPT adversary* $\mathcal{A}$*, and any* $n = \mathsf{poly}(\lambda)$*, the following is negligible in* λ*:*

$$|\Pr[\mathsf{Exp}^{(\mathcal{O},\alpha)-\mathsf{anon}\text{-}1}_{RS,\mathcal{A}}(1^\lambda, n) = 1] - \Pr[\mathsf{Exp}^{(\mathcal{O},\alpha)-\mathsf{anon}\text{-}0}_{RS,\mathcal{A}}(1^\lambda, n) = 1]|.$$

Experiment $:\mathsf{Exp}^{(\mathcal{O},\alpha)-\mathsf{anon}\text{-}b}_{RS,\mathcal{A}}(1^\lambda, n)$

1: **for** $i = 1, \ldots, n$ **do**
2: $\quad(\mathsf{sk}_i, \mathsf{vk}_i) \leftarrow \mathsf{KGen}(1^\lambda)$ *// using random coins* w_i
3: $d \leftarrow \mathcal{A}^{\mathcal{O},\mathsf{Corr}}(\mathsf{vk}_1, \ldots, \mathsf{vk}_n)$
4: **if** $|\{i_0, i_1\} \cap I| \leq \alpha$ **then return** d *// where* I *is the set of queries to* Corr
5: **else return** $\perp$

Security notion	Definition	Remarks
Adaptive anonymity against adversarially chosen keys	$(\{\mathsf{OSign}, \mathsf{Ch}-\mathsf{Sign}_b\}, 0)$-anonymous	Strongest notion compatible with user-controlled linkability.
Anonymity against full key exposure	$(\{\mathsf{OSign}, \mathsf{Ch}-\mathsf{Sign}_b\}, 2)$-anonymous	Incompatible with user-controlled linkability. Equivalent to unrepudiability [PS19].

Fig. 1. Anonymity notions for ring signatures

Unforgeability We now present the unforgeability definition for ring signatures, also referred to as unforgeability w.r.t. insider corruption in [BKM09].

Definition 2.4 (unforgeability). *A ring signature scheme is* unforgeable *if for any PPT adversary $\mathcal{A}$, and any $n = \mathsf{poly}(\lambda)$, the following is negligible in λ:* $\Pr[\mathsf{Exp}^{uf}_{RS,\mathcal{A}}(1^\lambda, n) = 1]$.

Experiment $:\mathsf{Exp}^{uf}_{RS,\mathcal{A}}(1^\lambda, n)$

1: **for** $i = 1, \ldots, n$ **do**
2: $\quad(\mathsf{sk}_i, \mathsf{vk}_i) \leftarrow \mathsf{KGen}(1^\lambda)$ *// using random coins w_i*
3: $(\sigma^*, m^*, R^*) \leftarrow \mathcal{A}^{\mathsf{OSign},\mathsf{Corr}}(\mathsf{vk}_1, \ldots, \mathsf{vk}_n)$
4: **if** $\mathsf{Vf}(\sigma^*, R^*, m^*) = 0$ **then** **return** 0
5: **if** $R^* \subseteq \{\mathsf{vk}_i\}_{i\in[n]\setminus I}$ *// none of the keys in R^* were corrupted*
6: $\quad$**then if** $\forall i \in [n], (*, m^*, R^*) \notin \mathit{SIG}[i]$ *// no signature on m^* for ring R^* has been queried*
7: $\quad\quad$**then return** 1
8: **else return** 0

3 Ring signatures with user-controlled linkability

3.1 Standard linkability

As in [DL21], we consider two types of linkability:

Implicit linkability: Signatures are accompanied by a pseudonym, generated by the user for a particular scope. Re-using the same scope leads to the same pseudonym, making all signatures with the same scope linkable. Signatures with different scopes cannot be linked, except via explicit link proofs.

Explicit linkability: A user can prove that she created a set of previously generated signatures, i.e. link the signatures in the set.

Definition 3.1 (RS-UCL). *A* ring signature scheme with user controlled linkability *(RS-UCL) is a tuple of PPT algorithms* $(\mathsf{KGen}, \mathsf{Sig}, \mathsf{Vf}, \mathsf{Link}, \mathsf{VerifyLink})$*, satisfying* correctness *(Defs. 3.3 and 3.4),* anonymity *(Def. 3.5),* unforgeability *(Defs. 3.6 and 3.8) and* non-frameability *(Defs. 3.9 and 3.10); and with the following syntax:*

$\mathsf{KGen}(1^\lambda)$ *on input a security parameter, outputs a signing key* sk *and a verification key* vk*.*

$\mathsf{Sig}(\mathsf{sk}, R, m, \mathsf{scp})$ *signs a message m w.r.t. scope* scp *via secret signing key* sk *for set of verification keys (called* the ring*) $R = \{\mathsf{vk}_1, \ldots, \mathsf{vk}_n\}$. The output is a pseudonym* nym *and a ring signature σ.*

$\mathsf{Vf}(\Sigma)$ *on input a signature tuple* $\Sigma = (m, \mathsf{scp}, R, \sigma, \mathsf{nym})$ *for ring* $R = \{\mathsf{vk}_1, \ldots, \mathsf{vk}_n\}$*, returns 1 if* σ *and* nym *are valid for message* m *and scope* scp *w.r.t.* R*, and 0 otherwise.*

$\mathsf{Link}(\mathsf{sk}, \mathsf{lm}, \boldsymbol{\Sigma})$ *on input a set of signature tuples* $\Sigma = \{\Sigma_i\}_{i\in[n]}$*, a user secret key* sk*, and a linking message* lm *(can be used to ensure freshness of the proof), outputs a proof* π_l *that these signatures are linked, or the error symbol* $\perp$*.*

$\mathsf{VerifyLink}(\mathsf{lm}, \boldsymbol{\Sigma}, \pi_l)$ *returns 1 if* π_l *is a valid proof that* $\boldsymbol{\Sigma} = \{\Sigma_i\}_{i\in[n]}$ *were produced by the same signer for link message* lm*, and 0 otherwise.*

3.2 Autonomous linking

We also introduce the notion of ring signatures with user controlled, and *autonomous* linking (RS-UCAL). This variant allows users to chose the linking secret independently of the signing key. It hence gives users the liberty of choosing which of their signatures should be linkable in the future. We express this feature via an additional algorithm $\mathsf{GenLinkSec}$ which outputs a linking secret ls. Now both the signing algorithm and the linking algorithm should input both the signing secret sk and the linking secret ls.

Definition 3.2. *A* ring signature scheme with user controlled autonomous linking *(RS-UCAL) is a tuple of PPT algorithms* $(\mathsf{KGen}, \mathsf{GenLinkSec}, \mathsf{Sig}, \mathsf{Vf}, \mathsf{Link}, \mathsf{VerifyLink})$*, where* $\mathsf{KGen}, \mathsf{Vf}, \mathsf{VerifyLink}$ *have the same syntax as an RS-UCL, and:*

$\mathsf{GenLinkSec}(1^\lambda)$ *takes input a security parameter, and outputs a linking secret* ls*.*

$\mathsf{Sig}(\mathsf{sk}, \mathsf{ls}, R, m, \mathsf{scp})$ *as in a RS-UCL scheme, only with additional input a linking secret* ls*.*

$\mathsf{Link}(\mathsf{ls}, \mathsf{lm}, \boldsymbol{\Sigma})$ *as in an RS-UCL scheme, only with input a linking secret* ls *instead of the secret key.*

An RS-UCAL scheme satisfies correctness *(Defs. 3.3 and 3.4),* anonymity *(Def. 3.5),* unforgeability *(Defs. 3.7 and 3.8) and* non-frameability *(Defs. 3.9 and 3.10).*

Text which is highlighted in blue only occurs for a RS-UCAL scheme. Text which is highlighted in green only occurs for a RS-UCL scheme.

3.3 Correctness

An RS-UC(A)L scheme should satisfy both verification correctness, which is equivalent to correctness for standard ring signatures, and linking correctness which ensures the correctness of the explicit linking algorithm Link. We give the full definitions below.

Definition 3.3 (Verification correctness). *An RS-UCAL scheme satisfies* verification correctness *if for any* $n = \mathsf{poly}(\lambda)$*, any* $\{(\mathsf{vk}_i, \mathsf{sk}_i)\}_{i\in[n]} \leftarrow \mathsf{KGen}(1^\lambda)$*, any* $i \in [n]$*, any* $\mathsf{ls} \leftarrow \mathsf{GenLinkSec}(1^\lambda)$*, and any message* m *and scope* scp*, there exists a negligible function* ϵ *such that:*

$$\Pr[\mathsf{Vf}((m, \mathsf{scp}, R = \{\mathsf{vk}_1, \ldots, \mathsf{vk}_n\}, \mathsf{Sig}(\mathsf{sk}_i, \mathsf{ls}, R, m, \mathsf{scp}))) = 1] = 1 - \epsilon(\lambda).$$

If $\epsilon = 0$ *then* perfect verification correctness *is satisfied.*

Definition 3.4 (Linking correctness). *An RS-UCAL scheme satisfies* linking correctness *if for any* $n = \mathsf{poly}(\lambda)$*, any messages* $m_1, \ldots, m_n$*, any scopes* $\mathsf{scp}_1, \ldots, \mathsf{scp}_n$*, any* $\{(\mathsf{vk}_i, \mathsf{sk}_i)\}_{i\in[n]} \leftarrow \mathsf{KGen}(1^\lambda)$*, any rings* $R_1, \ldots, R_n \subset \{\mathsf{vk}_1, \ldots, \mathsf{vk}_n\}$*, any* $i \in [n]$ *and any* $\mathsf{ls} \leftarrow \mathsf{GenLinkSec}(1^\lambda)$*, there exists a negligible function* ϵ *such that, denoting* $\Sigma := \{(m_j, \mathsf{scp}_j, R_j, \mathsf{Sig}(\mathsf{sk}_i, \mathsf{ls}, R_j, m_j))\}_{j\in[n]}$*, it holds that, for any linking message* lm*:*

$$\Pr[\mathsf{VerifyLink}(\mathsf{lm}, \Sigma, \mathsf{Link}(\mathsf{sk}_i, \mathsf{ls}, \mathsf{lm}, \Sigma)) = 1] = 1 - \epsilon(\lambda).$$

If $\epsilon = 0$ *then* perfect linking correctness *is satisfied.*

3.4 Security model

For privacy, signatures must not reveal anything about the signer's identity beyond what was intended by her (**anonymity**). Security is expressed through both **unforgeability**, which ensures that an adversary cannot forge a signature on behalf of a ring they are not a member of or forge a link proof for signatures they did not generate, and **non-frameability**, which states that an honest user cannot be framed by the adversary, so that signatures of an honest user are (implicitly or explicitly) linkable to signatures that she has not generated.

Oracles and state All oracles are parametrised by a list of keys pairs $\{(\mathsf{vk}_i, \mathsf{sk}_i)\}_{i\in[n]}$ and linking secrets $\{\mathsf{ls}_i\}_{i\in[n]}$, where $n = \mathsf{poly}(\lambda)$, $k = \mathsf{poly}(\lambda)$. In Figure 2 we describe the global state variables that all oracles can access.

Corruption oracle Corr takes as input an index $i \in [n]$, adds i to the list of corrupted indices $I \leftarrow I \cup \{i\}$, and outputs the randomness w_i used to compute key pair $(\mathsf{vk}_i, \mathsf{sk}_i)$.

Linking secret oracle OLS takes as input an index $i \in [k]$, adds i to the list of corrupted indices $J \leftarrow J \cup \{i\}$, and outputs the randomness w'_i used to compute linking secret ls_i.

Signing oracle $\mathsf{OSign}^{\mathbf{ucl\text{-}r}}$**:** takes as input a set R, an index $i \in [N]$, an index $j \in [k]$, a message m and a scope scp computes $(\mathsf{nym}, \sigma) \leftarrow \mathsf{Sig}(\mathsf{sk}_i, \mathsf{ls}_j, R \cup \mathsf{vk}_i, m, \mathsf{scp})$, adds $\Sigma := (m, \mathsf{scp}, R, \sigma, \mathsf{nym})$ to the list of signatures signed by user index i with linking secret index j: $\mathsf{SIG}[i, j] \leftarrow \mathsf{SIG}[i, j] \cup \{\Sigma\}$, and returns (nym, σ).

Linking oracle OLink**:** Allows the adversary to obtain link proofs for signatures of its choice. On input an index $i \in [n]$, an index $i \in [k]$, a linking message lm and a set of tuples $\mathbf{\Sigma} = \{\Sigma = (m, \mathsf{scp}, R, \sigma, \mathsf{nym})\}$, this oracle adds $(\mathsf{lm}, \mathbf{\Sigma})$ to the list of link proofs produced for index i: $\mathsf{LNK}[i] \leftarrow \mathsf{LNK}[i] \cup \{(\mathsf{lm}, \mathbf{\Sigma})\}$, and returns $\pi_l \leftarrow \mathsf{Link}(\mathsf{sk}_i, \mathsf{ls}_i, \mathsf{lm}, \mathbf{\Sigma})$.

Challenge signing oracle $\mathsf{Ch-Sign}_b$**:** Allows $\mathcal{A}$ to get signatures for challenge user index i_b with linking secret index j_b. On input a ring R, a message m, and a scope scp, $\mathsf{Ch-Sign}_b$ computes $(\mathsf{nym}, \sigma) \leftarrow \mathsf{Sig}(\mathsf{sk}_{i_b}, \mathsf{ls}_{j_b}, R \cup \{\mathsf{vk}_{i_0}, \mathsf{vk}_{i_1}\}, m, \mathsf{scp})$, sets $\Sigma := (m, \mathsf{scp}, R, \sigma, \mathsf{nym})$, adds Σ to the list of queried challenge signatures $\mathsf{CSIG} \leftarrow \mathsf{CSIG} \cup \{\Sigma\}$, and returns (nym, σ).

Challenge linking oracle $\mathsf{Ch-Link}_b$**:** Allows $\mathcal{A}$ to get link proofs for a challenge index i_b/ j_b. Precisely, on input a linking message lm and a set of tuples $\mathbf{\Sigma} = \{\Sigma = (m, \mathsf{scp}, R, \sigma, \mathsf{nym})\}$, it adds $(\mathsf{lm}, \mathbf{\Sigma})$ to the list of challenge link proofs $\mathsf{CLNK} \leftarrow \mathsf{CLNK} \cup \{(\mathsf{lm}, \mathbf{\Sigma})\}$, and returns $\pi_l \leftarrow \mathsf{Link}(\mathsf{sk}_{i_b}, \mathsf{ls}_{j_b}, \mathsf{lm}, \mathbf{\Sigma})$.

Variable	Content
I	List of corrupted indices (queried to Corr)
J	List of corrupted indices (queried to OLS)
i_b	Challenge user index in $\mathsf{anon\text{-}b}$. Ignored in the other games
j_b	Challenge linking secret index in $\mathsf{anon\text{-}b}$. Ignored in the other games
$\mathsf{SIG}[i]$	Signature tuples produced by OSign for user index i
CSIG	Signature tuples produced by $\mathsf{Ch-Sign}_b$ for challenge user index i_b
$\mathsf{LNK}[i]$	Link queries sent to OLink for user index i
CLNK	Link queries made to $\mathsf{Ch-Link}_b$

Fig. 2. Global state variables and their contents.

Helper Algorithm As in [DL21], we introduce the helper algorithm $\mathsf{Identify}$. In this case we do not need to require the existence of $\mathsf{Identify}$, because there is no trusted issuer in the ring signature setting. Therefore, user secret keys do not need to be extracted from join protocols as in the group signature setting. $\mathsf{Identify}$ here is just defined for notational simplicity, and can be achieved from the user controlled linkability functionality.

$\mathsf{Identify}(\mathsf{sk}, \mathsf{vk}, \mathsf{ls}, \Sigma = (m, \mathsf{scp}, R, \sigma, \mathsf{nym}))$

$(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{KGen}(1^\lambda), (\mathsf{nym}', \sigma') \leftarrow \mathsf{Sig}(\mathsf{sk}, \mathsf{ls}, \{vk\}, m, \mathsf{scp})$

if $\mathsf{nym}' = \mathsf{nym}$ **return** 1 **else return** 0

Anonymity Anonymity ensures that an adversary cannot figure out which of two (honest) challenge users generated a given signature. In formalizing this notion, one must be careful to exclude trivial wins leveraging user-controlled linkability (see the comments in the security experiment).

Definition 3.5 (Anonymity). *An RS-UCAL scheme satisfies adaptive anonymity against adversarially chosen keys if, for any PPT adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$*, and any* $n = \mathsf{poly}(\lambda)$*,* $k = \mathsf{poly}(\lambda)$*, it holds that* $|\Pr[\mathsf{Exp}^{\textit{anon-1}}_{\mathsf{UCL},\mathcal{A}}(1^\lambda, n, k) = 1] - \Pr[\mathsf{Exp}^{\textit{anon-0}}_{\mathsf{UCL},\mathcal{A}}(1^\lambda, n, k) = 1]|$ *is negligible in* λ*.*

Experiment $:\mathsf{Exp}^{\textit{anon-b}}_{\mathsf{UCL},\mathcal{A}}(1^\lambda, n, k)$

1: **for** $i = 1, \ldots, n$ $(\mathsf{sk}_i, \mathsf{vk}_i) \leftarrow \mathsf{KGen}(1^\lambda)$
2: **for** $i = 1, \ldots, k$ $\mathsf{ls}_i \leftarrow \mathsf{GenLinkSec}(1^\lambda)$
3: $(i_0, i_1, j_0, j_1, \mathsf{state}) \leftarrow \mathcal{A}_1^{\mathsf{OSign}^{\textit{ucl-r}}, \mathsf{OLink}, \mathsf{Corr}, \mathsf{OLS}}(\mathsf{choose}, \mathsf{vk}_1, \ldots, \mathsf{vk}_n)$
4: $d \leftarrow \mathcal{A}_2^{\mathsf{OSign}^{\textit{ucl-r}}, \mathsf{OLink}, \mathsf{Ch}-\mathsf{Sign}_b, \mathsf{Ch}-\mathsf{Link}_b, \mathsf{Corr}, \mathsf{OLS}}(\mathsf{guess}, \mathsf{state})$
5: *// Exclude trivial wins via implicit linking: a signature for scope* scp *was queried to* $\mathsf{Ch}-\mathsf{Sign}_b$ *and to* OSign *for index* i_0 *or* i_1
6: **if** $\exists \mathsf{scp}$ *s.t.* $(*, \mathsf{scp}, *, *, *) \in \mathit{CSIG} \wedge (*, \mathsf{scp}, *, *, *) \in \mathit{SIG}[i_0, j_0] \cup \mathit{SIG}[i_1, j_1]$
7: **then return** $\perp$
8: *// Exclude trivial wins via explicit linking: both challenge and non challenge signatures were queried to* $\mathsf{Ch}-\mathsf{Link}_b$ *or to* OLink
9: **if** $\exists \boldsymbol{\Sigma}$ *s.t.* $(\boldsymbol{\Sigma} \cap \mathit{CSIG} \neq \emptyset \wedge (*, \boldsymbol{\Sigma}) \in \mathit{LNK}[*])$
10: $\vee\, (\boldsymbol{\Sigma} \cap (\mathit{SIG}[i_0, j_0] \cup \mathit{SIG}[i_1, j_1]) \neq \emptyset \wedge (*, \boldsymbol{\Sigma}) \in \mathit{CLNK})$ **then return** $\perp$
11: **if** $\{i_0, i_1\} \cap I \neq \emptyset$ **then return** $\perp$
12: **if** $\{j_0, j_1\} \cap J \neq \emptyset$ **then return** $\perp$
13: **else return** d

Unforgeability For both RS-UCL and RS-UCAL we must ensure that only ring members can sign. However, for RS-UCL, we must also prevent signers from outputting multiple unlinkable signatures on the same scope. For RS-UCL one captures both by defining an attack as the generation of more signatures with different pseudonyms than corrupted users in the ring, using the same scope. This is meaningless for RS-UCAL where a single corrupted user can generate many linking secrets.

Definition 3.6 (Signature unforgeability). *An RS-UCL scheme satisfies signature unforgeability if for any PPT adversary* $\mathcal{A}$*, and any* $n = \mathsf{poly}(\lambda)$*, it holds that* $\Pr[\mathsf{Exp}^{\textit{sig-uf}}_{\mathsf{UCL},\mathcal{A}}(1^\lambda, n) = 1]$ *is negligible in* λ*.*

Experiment $:\mathsf{Exp}^{\textit{sig-uf}}_{\mathsf{UCL},\mathcal{A}}(1^\lambda, n)$

$(\mathsf{sk}_i, \mathsf{vk}_i) \leftarrow \mathsf{KGen}(1^\lambda)$ **for** $i = 1, \ldots, n$

$(\{\Sigma_1^*, \cdots, \Sigma_m^*\}, R^*) \leftarrow \mathcal{A}^{\mathsf{OSign}^{\textit{ucl-r}}, \mathsf{OLink}, \mathsf{Corr}}(\mathsf{vk}_1, \ldots, \mathsf{vk}_n)$

Parse $\Sigma_j^* = (m_j^*, \mathsf{scp}_j^*, R_j, \sigma_j^*, \mathsf{nym}_j^*)$ *for* $j \in [m]$

if $\exists j \in [m]$ *s.t.* $R_j \neq R^*$ **return** 0

$m^* \leftarrow |R^* \backslash \{\mathsf{vk}_i\}_{i \in [n] \backslash I}|$

if *the following conditions all hold*

$1. m > m^*$ // *more signatures are output than the corrupted users in the ring*

$2. \forall j_1, j_2 \in [m]$ $\mathsf{scp}_{j_1}^* = \mathsf{scp}_{j_2}^*$ *and* $\mathsf{nym}_{j_1}^* \neq \mathsf{nym}_{j_2}^*$ // *all signatures are unlinked*

$3. \forall j \in [m]$ $\mathsf{Vf}(\Sigma_j^*) = 1$

$4. \forall i \in [n], j \in [m]$ $(m_j^*, \mathsf{scp}_j^*, R^*, *, *) \notin \textit{SIG}[i]$

then return 1

else return 0

As mentioned earlier, Definition 3.6 is too strong for RS-UCAL. Indeed, the autonomous linking property allows parties to change the linking secret as frequently as desired. Hence an adversary which wants to output many unlinkable secrets could simply keep changing the linking secret. For such schemes we adopt a similar notion of (signature) unforgeability as that of standard ring signatures, with the difference that the adversary also has access to a linking oracle. We give the full experiment below.

Experiment $:\mathsf{Exp}^{\mathsf{sig\text{-}uf\text{-}al}}_{\mathsf{UCAL},\mathcal{A}}(1^\lambda, n, k)$

for $i = 1, \ldots, n$

$(\mathsf{sk}_i, \mathsf{vk}_i) \leftarrow \mathsf{KGen}(1^\lambda)$

for $i = 1, \ldots, k$ $\mathsf{ls}_i \leftarrow \mathsf{GenLinkSec}(1^\lambda)$

$\Sigma^* := (m^*, \mathsf{scp}^*, R, \sigma^*, \mathsf{nym}^*) \leftarrow \mathcal{A}^{\mathsf{OSign}^{\textit{ucl-r}}, \mathsf{OLink}, \mathsf{Corr}, \mathsf{OLS}}(\mathsf{vk}_1, \ldots, \mathsf{vk}_n)$

if $\mathsf{Vf}(\Sigma^*) = 0$ **then return** 0

if $R^* \subseteq \{\mathsf{vk}_i\}_{i \in [n] \backslash I}$ // none of the keys in R^* were corrupted

then if $\forall i \in [n], (m^*, \mathsf{scp}^*, R^*, *, *) \notin \mathsf{SIG}[i, \cdot]$

then return 1

else return 0

Definition 3.7 (Signature unforgeability with autonomous linking). *An RS-UCAL scheme satisfies signature unforgeability if, for any PPT adversary* $\mathcal{A}$*, and any* $n = \mathsf{poly}(\lambda)$*,* $k = \mathsf{poly}(\lambda)$*, it holds that* $\Pr[\mathsf{Exp}^{\textit{sig-uf-al}}_{\mathsf{UCAL},\mathcal{A}}(1^\lambda, n, k) = 1]$ *is negligible in* λ*.*

We additionally define link unforgeability for both the RS-UCL and RS-UCAL schemes, which ensures that the linking proof is unforgeable.

Definition 3.8 (Link unforgeability). *An RS-UCAL scheme satisfies link unforgeability if, for any PPT adversary* $\mathcal{A}$*, and any* $n = \mathsf{poly}(\lambda)$*,* $k = \mathsf{poly}(\lambda)$*, the following is negligible in* λ*:* $\Pr[\mathsf{Exp}^{\textit{link-uf}}_{\mathsf{UCL},\mathcal{A}}(1^\lambda, n, k) = 1]$.

Experiment $:\mathsf{Exp}^{\textit{link-uf}}_{\mathsf{UCL},\mathcal{A}}(1^\lambda, n, k)$

$(\mathsf{sk}_i, \mathsf{vk}_i) \leftarrow \mathsf{KGen}(1^\lambda)$ **for** $i = 1, \ldots, n$

for $i = 1, \ldots, k$ $\mathsf{ls}_i \leftarrow \mathsf{GenLinkSec}(1^\lambda)$

$(\mathsf{lm}, \mathbf{\Sigma}, \pi_l) \leftarrow \mathcal{A}^{\mathsf{OSign}^{\textit{ucl-r}}, \mathsf{OLink}, \mathsf{Corr}, \mathsf{OLS}}(\mathsf{vk}_1, \ldots, \mathsf{vk}_n)$

$\mathsf{VerifyLink}(\mathsf{lm}, \mathbf{\Sigma}, \pi_l) = 0$ *or* $\exists i \in [n][k] : (\mathsf{lm}, \mathbf{\Sigma}) \in \textit{LNK}[i]$ **return** 0

if $\exists i \in [n] : \forall \Sigma \in \mathbf{\Sigma} : \Sigma \in \textit{SIG}[i]$ *and* $i \notin I$ **return** 1

if $\exists i \in [k] : \forall \Sigma \in \mathbf{\Sigma} : \Sigma \in \textit{SIG}[\cdot, i]$ *and* $i \notin J$ **return** 1

else return 0

Non-frameability Non-frameability guarantees that an honest user cannot be framed by the adversary, such that signatures of an honest user are linkable to signatures that she has not generated. We capture this in the implicit/ explicit setting with signature/ link non-frameability respectively. Roughly, signature non–frameability ensures that an adversary cannot output a valid signature that links to another signature generated by an uncorrupted user via the signing oracle. Link non-frameability ensures that an adversary cannot explicitly link two signatures that were either from different users or such that one of the two signatures is not from a user in the experiment.

Definition 3.9 (Signature non-frameability). *An RS-UCAL scheme satisfies signature non-frameability if, for any PPT adversary $\mathcal{A}$, and any $n = \mathsf{poly}(\lambda)$, $k = \mathsf{poly}(\lambda)$, the following is negligible in λ:* $\Pr[\mathsf{Exp}^{\textit{sign-frame}}_{\mathsf{UCL},\mathcal{A}}(1^\lambda, n, k) = 1]$.

Definition 3.10 (Link non-frameability). *An RS-UCAL scheme satisfies link non-frameability if, for any PPT adversary $\mathcal{A}$, and any $n = \mathsf{poly}(\lambda)$, $k = \mathsf{poly}(\lambda)$, the following is negligible in λ:* $\Pr[\mathsf{Exp}^{\textit{link-frame}}_{\mathsf{UCL},\mathcal{A}}(1^\lambda, n, k) = 1]$.

Experiment $:\mathsf{Exp}^{\mathsf{sign\text{-}frame}}_{\mathsf{UCL},\mathcal{A}}(1^\lambda, n, k)$

$(\mathsf{sk}_i, \mathsf{vk}_i) \leftarrow \mathsf{KGen}(1^\lambda)$ **for** $i = 1, \ldots, n$

$\mathsf{ls}_i \leftarrow \mathsf{GenLinkSec}(1^\lambda)$ **for** $i = 1, \ldots, k$

$\Sigma := (m, \mathsf{scp}, R, \sigma, \mathsf{nym}) \leftarrow \mathcal{A}^{\mathsf{OSign}^{\mathbf{ucl\text{-}r}}, \mathsf{OLink}, \mathsf{Corr}, \mathsf{OLS}}(\mathsf{vk}_1, \ldots, \mathsf{vk}_n)$

return 1 **if** :

$\mathsf{Vf}(\Sigma) = 1$ and

$\exists i \in [n] : (m, \mathsf{scp}, R, \mathsf{nym}, \cdot) \notin \mathsf{SIG}[i] \wedge i \notin I \wedge (\cdot, \mathsf{scp}, \cdot, \cdot, \mathsf{nym}) \in \mathsf{SIG}[i]$

$\exists i \in [k] : \Sigma \notin \mathsf{SIG}[\cdot, i] \wedge i \notin J \wedge (\cdot, \mathsf{scp}, \cdot, \cdot, \mathsf{nym}) \in \mathsf{SIG}[\cdot, i]$

else return 0

Experiment $:\mathsf{Exp}^{\mathsf{link\text{-}frame}}_{\mathsf{UCL},\mathcal{A}}(1^\lambda, n, k)$

$(\mathsf{sk}_i, \mathsf{vk}_i) \leftarrow \mathsf{KGen}(1^\lambda)$ **for** $i = 1, \ldots, n$

$\mathsf{ls}_i \leftarrow \mathsf{GenLinkSec}(1^\lambda)$ **for** $i = 1, \ldots, k$

$(\mathsf{lm}, \mathbf{\Sigma}, \pi_l) \leftarrow \mathcal{A}^{\mathsf{OSign}^{\mathbf{ucl\text{-}r}}, \mathsf{OLink}, \mathsf{Corr}, \mathsf{OLS}}(\mathsf{vk}_1, \ldots, \mathsf{vk}_N)$

if $\mathsf{VerifyLink}(\mathsf{lm}, \mathbf{\Sigma}, \pi_l) = 0$ **return** 0

Parse $\mathbf{\Sigma} = \{\Sigma_1, \cdots \Sigma_m\}$

$\forall j \in [m]$

if $\exists i \in [n] : \Sigma_j \in \mathsf{SIG}[i]$ **then** $i_j \leftarrow i$

if $\exists i \in [k] : \Sigma_j \in \mathsf{SIG}[\cdot, i]$ **then** $i_j \leftarrow i$

elseif $\exists i \in [n] : \mathsf{Identify}(\mathsf{sk}_i, \mathsf{vk}_i, \Sigma_j) = 1$ **then** $i_j \leftarrow i$

elseif $\exists i \in [k] : \mathsf{Identify}(\mathsf{ls}_i, \Sigma_j) = 1$ **then** $i_j \leftarrow i$

else $i_j \leftarrow n + 1$

if $\exists j_1, j_2 \in [m] \quad i_{j_1} \neq i_{j_2}$ **return** 1

else return 0

4 Anonymous Key Randomisable Signatures

We will now give the syntax and security requirements for a new primitive we call Anonymous Key Randomisable Signatures (AKRS), which is closely related to Signatures with Flexible Public Key (SFPK) [BHKS18].

Intuition An SFPK scheme is a standard signature scheme that allows for public and secret keys to be re-randomised. These re-randomised keys are considered to be in the same equivalence class as the original key pair. Such re-randomised public keys, and signatures which verify for them, should not be linkable to the original key pair of their equivalence class. This is formalised by a class hiding requirement. Furthermore, during key generation, a trapdoor can be generated allowing to efficiently decide if public keys are in the same equivalence class.

At first sight, an SFPK scheme seems to allow transforming ring signatures into ring signatures with user controlled and autonomous linking. A user's key pair would be that of a standard ring signature, while their linking secret key would be an SFPK key pair, with the corresponding trapdoor. During signing, the user's pseudonym could be a re-randomisation of the SFPK public key with the randomness set to be the scope, so that the resulting public key is within the same equivalence class as the original keypair in their linking key. The signature should include a standard ring signature to ensure that the signature was output by a ring member and a SFPK signature valid under the public key in the pseudonym to prevent framing attacks.

However, in the SFPK class hiding requirement, which is necessary to ensure anonymity of the ring signature, the adversary does not get to see the randomness used to re-randomise the public key. In their game, the knowledge of this randomness allows to trivially win, as an adversary can re-run the randomisation algorithm itself. For the application to ring signatures described in the previous paragraph, this will not do, since the randomness is a public value: the scope. On the other hand, in this application, the SFPK's secret key remains hidden: it is part of the linking secret, which is unknown to the anonymity adversary. Therefore, we modify SFPK to ensure that the secret key is necessary to generate public keys in the same equivalence class, thereby avoiding the aforementioned trivial attack, while allowing for the adversary to know the randomness.

A second issue in the above construction, is that signatures can only be explicitly linked by revealing the trapdoor, which would allow *all signatures* under the same linking key to be linked. One way around this is to add an additional functionality, proving, in zero-knowledge, that the pseudonyms in signatures are in the same equivalence class, using the trapdoor in the linking secret key. For efficiency, however, we chose a different approach: we allow for several key pairs in the same equivalence class to be accumulated into one key pair. Then the accumulated secret key could be used to sign a signature valid under the accumulated public key. A verifier can check that this signature is valid under an accumulated public key, resulting from the pseudonyms of all signatures being linked. This means the linking proof is only the size of an SFPK signature. However, explicitly linking signatures requires keeping track of each re-randomised secret key used in the linking key. We overcome this by re-randomising only public keys, but not secret keys, i.e., the secret key remains the same for all public keys in a given equivalence class. An accumulated public key will then correspond to the same secret key as the public keys it was built from, as long as the latter lie in the same equivalence class. Finally, observe that the secret key essentially fulfils the role of the trapdoor, as it allows to link public keys in the same equivalence class. We thus remove the trapdoor from our primitive's syntax.

We now formally define Anonymous Key Randomisable Signatures. The security properties required are given in Sections 4.1 to 4.3.

Definition 4.1 (Anonymous Key Randomisable Signature (AKRS)). *An* anonymous key randomisable signature scheme *is a tuple of PPT algorithms* $(\mathsf{KGen}, \mathsf{ChgPK}, \mathsf{Sig}, \mathsf{Vf}, \mathsf{Accum})$*, satisfying* correctness *(Def. 4.2),* class hiding *(Def. 4.3),* existential unforgeability under chosen message attacks *(Def. 4.4) and* accumulation soundness *(Def. 4.5); and with the following syntax:*

$\mathsf{KGen}(1^\lambda)$ *on input a security parameter* 1^λ*, outputs a signing key* sk *and a public key* pk*.*

$\mathsf{ChgPK}(\mathsf{sk}, t)$ *on input a secret key* sk *and a tag* t*, outputs a new public key* pk' *in the same equivalence class.*

$\mathsf{Sig}(\mathsf{sk}, \mathsf{pk}, m)$ *on input a signing key* sk*, a public key* pk *and a message* m*, outputs a signature* σ*.*

$\mathsf{Vf}(\mathsf{pk}, m, \sigma)$ *on input a public key* pk*, a message* m *and a signature* σ*, the verification algorithm returns 1 if* σ *is a valid signature on* m *w.r.t.* pk*, and 0 otherwise.*

$\mathsf{Accum}((t_1, \mathsf{pk}_1), \cdots, (t_k, \mathsf{pk}_k))$ *on input* k *public keys* $\mathsf{pk}_1, \cdots, \mathsf{pk}_k$ *with respect to tags* $t_1, \cdots, t_k$*, outputs an accumulated public key* $\tilde{\mathsf{pk}}$*.*

Definition 4.2 (Correctness). *Consider any positive integer* k*; any tags* $t_1, \cdots, t_k \in \{0,1\}^*$*; let* $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KGen}(1^\lambda)$*, for all* $i \in [k]$ $\mathsf{pk}'_i \leftarrow \mathsf{ChgPK}(\mathsf{sk}, t_i)$*, and* $\tilde{\mathsf{pk}} \leftarrow \mathsf{Accum}((t_1, \mathsf{pk}'_1), \cdots (t_k, \mathsf{pk}'_k))$*. An AKRS scheme is* correct *if for any message* m*, there exists a negligible function* ϵ *such that:*

$$\Pr\left[\begin{array}{c|c} \mathsf{Vf}(\mathsf{pk}, m, \sigma) = 1 & \sigma \leftarrow \mathsf{Sig}(\mathsf{sk}, \mathsf{pk}, m), \\ \wedge \forall i \in [k],\ \mathsf{Vf}(\mathsf{pk}'_i, m, \sigma_i) = 1 & \forall i \in [k],\ \sigma_i \leftarrow \mathsf{Sig}(\mathsf{sk}, \mathsf{pk}'_i, m), \\ \wedge \mathsf{Vf}(\tilde{\mathsf{pk}}, m, \tilde{\sigma}) = 1 & \tilde{\sigma} \leftarrow \mathsf{Sig}(\mathsf{sk}, \tilde{\mathsf{pk}}, m) \end{array}\right] = 1 - \epsilon(\lambda).$$

If $\epsilon = 0$ *then* perfect correctness *is satisfied.*

4.1 Class Hiding

We ensure that an adversary cannot guess which of two public keys are re-randomised with a tag *chosen by the adversary* (through oracle OChgPK), even while they can obtain signatures on these public keys and the re-randomised keys (via the OSign oracle).

Definition 4.3 (Class Hiding). *An* anonymous key randomisable signature scheme *satisfies class hiding if for any PPT adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$*,* $\Pr[\mathsf{Exp}^{\textit{class-hiding}}_{\mathsf{AKRS},\mathcal{A}}(1^\lambda) = 1]$ *is negligible in* λ*:*

Experiment $:\mathsf{Exp}^{\mathsf{class\text{-}hiding}}_{\mathsf{AKRS},\mathcal{A}}(1^\lambda)$

$b^* \leftarrow\!\!\$\ \{0,1\}$

for $b \in \{0,1\}$ $\quad (\mathsf{sk}_b, \mathsf{pk}_b) \leftarrow \mathsf{KGen}(1^\lambda)$

$\mathsf{sk}_2 \leftarrow \mathsf{sk}_{b^*}, L_1 := \{\}, L_2 := \{\}$

$d \leftarrow \mathcal{A}^{\mathsf{OChgPK}_{L_1,L_2}, \mathsf{OSign}_{L_1,L_2}}(\mathsf{choose}, \mathsf{pk}_0, \mathsf{pk}_1)$

return $((d = b^*) \wedge (L_1 \cap L_2 = \emptyset))$

$\mathsf{OChgPK}_{L_1,L_2}(a, t)$

1 : **if** $\mathsf{sk}_a = \perp$ **then return** $\perp$
2 : **if** $a \in \{0,1\}$ **then** $L_1 \leftarrow L_1 \cup \{t\}$
3 : **if** $a = 2$ **then** $L_2 \leftarrow L_2 \cup \{t\}$
4 : $\mathsf{pk}' \leftarrow \mathsf{ChgPK}(\mathsf{sk}_a, t)$
5 : **return** pk'

$\mathsf{OSign}_{L_1,L_2}(a, m, \mathsf{pk}', \{t_1, \cdots, t_m\})$

1 : **if** $\mathsf{sk}_a = \perp$ **then return** $\perp$
2 : **if** $\{t_1, \cdots, t_m\} \neq *$ **then**
3 : **if** $a \in \{0,1\}$ **then** $L_1 \leftarrow L_1 \cup \{t_1, \cdots, t_m\}$
4 : **if** $a = 2$ **then** $L_2 \leftarrow L_2 \cup \{t_1, \cdots, t_m\}$
5 : **if** $\mathsf{pk}' \neq \mathsf{Accum}((t_1, \mathsf{ChgPK}(\mathsf{sk}_a, t_1)), \cdots, (t_m, \mathsf{ChgPK}(\mathsf{sk}_a, t_m)))$ **then return** $\perp$
6 : $\sigma \leftarrow \mathsf{Sig}(\mathsf{sk}_a, \mathsf{pk}', m)$
7 : **if** $t = *$ **then if** $a = 2$ **then return** $\perp$
8 : **else** $\sigma \leftarrow \mathsf{Sig}(\mathsf{sk}_a, \mathsf{pk}_a, m)$
9 : **return** σ

4.2 Existential Unforgeability under Chosen Message Attacks

We ensure that signatures cannot be forged for a public key in the equivalence class of an honest user, even when they can obtain multiple public keys in that equivalence class on tags of their choice. Below we provide the full experiment for our AKRS existential unforgeability under chosen message attacks requirement.

Experiment :$\mathsf{Exp}^{\mathsf{euf-cma}}_{\mathsf{AKRS},\mathcal{A}}(1^\lambda)$

$Q \leftarrow \{\}, (\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KGen}(1^\lambda)$
$(\mathsf{pk}', \{t_1, \cdots, t_m\}, m, \sigma) \leftarrow \mathcal{A}^{\mathsf{OChgPK},\mathsf{OSign}}(\mathsf{pk})$
if $\mathsf{Vf}(\mathsf{pk}', m, \sigma) = 1 \wedge (m, \{t_1, \cdots, t_m\},) \notin Q$
$\wedge\ \mathsf{pk}' = \mathsf{Accum}((t_1, \mathsf{ChgPK}(\mathsf{sk}, t_1)), \cdots, (t_m, \mathsf{ChgPK}(\mathsf{sk}, t_m)))$
  then return 1
else return 0

$\mathsf{OChgPK}(t)$

$\mathsf{pk}' \leftarrow \mathsf{ChgPK}(\mathsf{sk}, t)$
return pk'

$\mathsf{OSign}_Q(m, \mathsf{pk}', \{t_1, \cdots, t_m\})$

if $t \neq *$ **then**
  if $\mathsf{pk}' \neq \mathsf{Accum}((t_1, \mathsf{ChgPK}(\mathsf{sk}, t_1)), \cdots, (t_m, \mathsf{ChgPK}(\mathsf{sk}, t_m)))$
    then return $\perp$
  $\sigma \leftarrow \mathsf{Sig}(\mathsf{sk}, \mathsf{pk}', m)$
if $t = *$ **then** $\sigma \leftarrow \mathsf{Sig}(\mathsf{sk}, \mathsf{pk}, m)$
$Q \leftarrow Q \cup \{(m, t)\}$
return σ

Definition 4.4 (Existential Unforgeability under Chosen Message Attacks). *An* anonymous key randomisable signature scheme *satisfies existential unforgeability under chosen message attacks if for any PPT adversary* $\mathcal{A}$, $\Pr[\mathsf{Exp}^{\mathsf{euf-cma}}_{\mathsf{AKRS},\mathcal{A}}(1^\lambda) = 1]$ *is negligible in* λ.

4.3 Accumulation Soundness

This is a new requirement necessary due to the accumulation functionality. It must not be possible to produce a signature which verifies for an accumulated public key, if the public keys input to the accumulation algorithm do not belong to the same equivalence class.

Definition 4.5 (Accumulation Soundness). *An* anonymous key randomisable signature scheme *satisfies accumulation soundness if for any PPT adversary* $\mathcal{A}$, $\Pr[\mathsf{Exp}^{\mathsf{acc-sound}}_{\mathsf{AKRS},\mathcal{A}}(1^\lambda) = 1]$ *is negligible in* λ.

Experiment :$\mathsf{Exp}^{\mathsf{acc-sound}}_{\mathsf{AKRS},\mathcal{A}}(1^\lambda)$

for $i \in [k]$ $(\mathsf{sk}_i, \mathsf{pk}_i) \leftarrow \mathsf{KGen}(1^\lambda)$
$(\{(\hat{t}_i, \hat{pk}_i, \hat{m}_i, \hat{\sigma}_i)\}_{i\in[k+1,k+l]}, \mathcal{I}, \{t_i\}_{i\in\mathcal{I}}, m, \sigma) \leftarrow \mathcal{A}((\mathsf{sk}_1, \mathsf{pk}_1), \cdots, (\mathsf{sk}_k, \mathsf{pk}_k))$
if $\exists (i,j) \in \mathcal{I} \cup [k+1, k+l]$ *such that* $i \neq j$ *and* $t_i = t_j$ **return** 0
if $\mathcal{I} = \emptyset$ **return** 0
if $\exists i \in [k+1, k+l]$ *s.t* $\mathsf{Vf}(\hat{pk}_i, \hat{m}_i, \hat{\sigma}_i) = 0$ **return** 0
for $i \in \mathcal{I}$ $\mathsf{pk}'_i \leftarrow \mathsf{ChgPK}(\mathsf{sk}_i, t_i)$
$\tilde{\mathsf{pk}} \leftarrow \mathsf{Accum}(\{(t_i, \mathsf{pk}'_i)\}_{i\in\mathcal{I}}, \{(\hat{t}_i, \hat{\mathsf{pk}}_i)\}_{i\in[k+1,k+l]})$
if $\mathsf{Vf}(\tilde{\mathsf{pk}}, m, \sigma) = 0$ **return** 0
if $\exists (i,j) \in \mathcal{I}$ *s.t.* $\mathsf{sk}_i \neq \mathsf{sk}_j \vee \exists i \in [k+1, k+l]$ *s.t.* $\forall j \in \mathcal{I}, \mathsf{ChgPK}(sk_j, \hat{t}_i) \neq \hat{pk}_i$
  then return 1
else return 0

$\mathsf{KGen}(1^\lambda)$

$(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{RS.KGen}(1^\lambda)$
return $(\mathsf{sk}, \mathsf{vk})$

$\mathsf{GenLinkSec}(1^\lambda)$

$(\mathsf{ls}, \cdot) \leftarrow \mathsf{AKRS.KGen}(1^\lambda)$ **return** ls

$\mathsf{Sig}(\mathsf{sk}, \mathsf{ls}, R, m, \mathsf{scp})$

Parse $(\mathsf{vk}_1, \ldots, \mathsf{vk}_n) \leftarrow R$
$\mathsf{nym} \leftarrow \mathsf{AKRS.ChgPK}(\mathsf{ls}, \mathsf{scp})$
$\Omega \leftarrow \mathsf{RS.Sig}(\mathsf{sk}, R, m||\mathsf{scp}||\mathsf{nym})$
$\Psi \leftarrow \mathsf{AKRS.Sig}(\mathsf{ls}, \mathsf{nym}, m||\mathsf{scp}||R||\Omega)$
$\sigma \leftarrow (\Omega, \Psi)$
return (nym, σ)

$\mathsf{Vf}(m, \mathsf{scp}, R, (\Omega, \Psi), \mathsf{nym})$

Parse $(\mathsf{vk}_1, \ldots, \mathsf{vk}_n) \leftarrow R$
if $\mathsf{RS.Vf}(\Omega, R, m||\mathsf{scp}||\mathsf{nym}) = 0$
  then return 0
if $\mathsf{AKRS.Vf}(\mathsf{nym}, m||\mathsf{scp}||R||\Omega, \Psi) = 0$
  then return 0
return 1

$\mathsf{Link}(\mathsf{ls}, \mathsf{lm}, \mathbf{\Sigma})$

if Parse $\{(m_i, \mathsf{scp}_i, R_i, \sigma_i, \mathsf{nym}_i)\}_{i\in[k]} \leftarrow \mathbf{\Sigma}$
if $\exists i, j \in [k], (i \neq j) \wedge (\mathsf{scp}_i = \mathsf{scp}_j)$
  then return $\perp$
if $\exists i \in [k], \mathsf{nym}_i \neq \mathsf{AKRS.ChgPK}(\mathsf{ls}, \mathsf{scp}_i)$
  then return $\perp$
if $\exists i \in [k], \mathsf{Vf}(m_i, \mathsf{scp}_i, R_i, \sigma_i, \mathsf{nym}_i) = 0$
  then return $\perp$
$\tilde{\mathsf{nym}} \leftarrow \mathsf{Accum}(\mathsf{nym}_1, \ldots, \mathsf{nym}_k)$
$\pi_l \leftarrow \mathsf{AKRS.Sig}(\mathsf{ls}, \tilde{\mathsf{nym}}, \mathsf{lm}||\mathbf{\Sigma})$
return π_l

$\mathsf{VerifyLink}(\mathsf{lm}, \mathbf{\Sigma}, \pi_l)$

Parse $\{(m_i, \mathsf{scp}_i, R_i, \sigma_i, \mathsf{nym}_i)\}_{i\in[k]} \leftarrow \mathbf{\Sigma}$
if $\exists i, j \in [k], (i \neq j) \wedge (\mathsf{scp}_i = \mathsf{scp}_j)$
  then return 0
if $\exists i \in [k], \mathsf{Vf}(m_i, \mathsf{scp}_i, R_i, \sigma_i, \mathsf{nym}_i) = 0$
  then return 0
$\tilde{\mathsf{nym}} \leftarrow \mathsf{Accum}(\mathsf{nym}_1, \ldots, \mathsf{nym}_k)$
return $\mathsf{AKRS.Vf}(\tilde{\mathsf{nym}}, \mathsf{lm}||\mathbf{\Sigma}, \pi_l)$

Fig. 3. RS-UCAL from standard ring signatures and AKRS.

5 Constructions for RS-UCL and RS-UCAL

5.1 A RS-UCAL Construction

We provide a generic construction, upgrading a standard ring signature scheme to a scheme with user controlled autonomous linking. We build upon a standard ring signature scheme $\mathsf{RS} := (\mathsf{KGen}, \mathsf{Sig}, \mathsf{Vf})$ as described formally in Appendix 2 and an AKRS $\mathsf{AKRS} = (\mathsf{KGen}, \mathsf{ChgPK}, \mathsf{Sig}, \mathsf{Vf}, \mathsf{Accum})$, as defined in Section 4.

Our RS-UCAL scheme UCAL, given in Figure 3, works as follows. Key generation is simply the key generation for a standard ring signature, whereas the linking secret is the secret key for an AKRS. When signing, the pseudonym is the public key corresponding to the linking secret, randomised with respect to the scope, a one way function of the linking secret and scope as desired. We then included a standard ring signature to ensure that a non-member cannot sign on behalf of the ring, i.e. signature unforgeability. In order to prevent a signature non-frameability attack, where an adversary uses the pseudonym of an honest user's signature in their own signature, we also include an AKRS signature with respect to the pseudonym as the public key. In order to explicitly link signatures we make use of the accumulation functionality from AKRS. A set of signatures that were all generated with the same linking secret contain pseudonyms that can be accumulated. This accumulated public key can be used to sign an AKRS signature, which will be the link proof. Due to the accumulation soundness property and unforgeability of AKRS, link non-frameability and link unforgeability are ensured, respectively. Anonymity follows from the anonymity of standard ring signatures, and the class hiding property of the AKRS which ensures AKRS public keys and signatures cannot be linked based on equivalence class, and so UCAL signatures cannot be linked based on the linking secret.

Theorem 5.1. *If* AKRS *is an anonymous key re-randomisable signature scheme and* RS *is a (standard) ring signature scheme, then* UCAL *is a ring signature scheme with user controlled autonomous linking.*

Proof

Verification correctness: Follows from the correctness of AKRS, and the correctness of RS.

Linking correctness: Follows from the correctness of AKRS.

Anonymity: We show that if AKRS is class hiding and unforgeable, and RS satisfies adaptive anonymity against adversarially chosen keys, then so does UCAL.

We proceed through a game hop that we show is indistinguishable to the adversary assuming that the RS satisfies adaptive anonymity against adversarially chosen keys. In the final game, assuming there exists an algorithm $\mathcal{A}$ that wins with probability ϵ_{AKRS}, we show how to construct an algorithm $\mathcal{B}_{\mathsf{AKRS}}$, breaking the class hiding property of AKRS.

We define Game 0 as the experiment $\mathsf{Exp}^{\mathsf{anon\text{-}b}}_{\mathsf{UCL},\mathcal{A}}(1^\lambda, n)$. Let S_i be the event that adversary $\mathcal{A}$ wins in this experiment.

Game 1 is identical to Game 0, except that in the $\mathsf{Ch-Sign}_b$ oracle sk_{i_0} is always used to sign Ω instead of sk_{i_b}.

We show that Game 0 and Game 1 are indistinguishable assuming the RS satisfies adaptive anonymity against adversarially chosen keys. We give a distinguishing algorithm $\mathcal{B}_{\mathsf{RS}}$ below that aims to break the anonymity of the RS.

Algorithm $\mathcal{B}_{\mathsf{RS}}$ gets as input a set of public verification keys $\mathsf{vk}_1, \ldots, \mathsf{vk}_n$. It samples $(\cdot, \mathsf{ls}_i) \leftarrow \mathsf{AKRS.KGen}(1^\lambda)$ for $i \in [k]$, and initialises empty lists LNK and CLNK. It calls upon $\mathcal{A}_1(\mathsf{choose}, \mathsf{vk_1}, \ldots, \mathsf{vk_n})$, and answers $\mathcal{A}_1$'s oracle queries as follows. $\mathcal{B}_{\mathsf{RS}}$ knows all the linking secrets so can perform OLink and OLS as normal.

$\mathsf{OSign}^{\mathbf{ucl\text{-}r}}(i, R, m, \mathsf{scp})$**:** $\mathcal{B}_{\mathsf{RS}}$ parses $(\mathsf{vk}_1, \ldots, \mathsf{vk}_n) \leftarrow R$, and computes $\mathsf{nym} \leftarrow \mathsf{AKRS.ChgPK}(\mathsf{ls}_i, \mathsf{scp})$. It then calls upon its own signing oracle, with input $(i, R, m||\mathsf{nym})$, which returns Ω. Finally, it computes $\Psi \leftarrow \mathsf{AKRS.Sig}(\mathsf{ls}_i, \mathsf{nym}, m||\mathsf{scp}||R||\Omega)$ and sends $(\mathsf{nym}, (\Omega, \Psi))$ to $\mathcal{A}_1$.

$\mathsf{Corr}(i)$**:** $\mathcal{B}_{\mathsf{RS}}$ queries i to its own corruption oracle, which returns sk_i, and sends $(\mathsf{sk}_i, \mathsf{ls}_i)$ to $\mathcal{A}_1$.

After a polynomial number of queries, $\mathcal{A}_1$ outputs $(i_0, i_1), (j_0, j_1), \mathsf{state}$. $\mathcal{B}$ samples a random bit β.

Algorithm $\mathcal{B}_{\mathsf{RS}}$ now calls upon $\mathcal{A}_2(\mathsf{guess}, \mathsf{state})$, and answers $\mathcal{A}_2$'s oracle queries as follows.

$\mathsf{OSign}^{\mathbf{ucl\text{-}r}}, \mathsf{OLink}, \mathsf{Corr}, \mathsf{OLS}$**:** $\mathcal{B}_{\mathsf{RS}}$ behaves as it did with $\mathcal{A}_1$.

$\mathsf{Ch-Sign}_b(R, m, \mathsf{scp})$**:** $\mathcal{B}_{\mathsf{RS}}$ first computes $\mathsf{nym} \leftarrow \mathsf{AKRS.ChgPK}(\mathsf{ls}_{j_\beta}, \mathsf{scp})$. It then calls upon its own challenge signing oracle, with input $(\{i_\beta, i_0\}, R, m||\mathsf{nym})$, which returns Ω. Finally, $\mathcal{B}_{\mathsf{RS}}$ computes $\Psi \leftarrow \mathsf{AKRS.Sig}(\mathsf{ls}_{j_\beta}, \mathsf{nym}, m||\mathsf{scp}||R||\Omega)$ and sends $(\mathsf{nym}, (\Omega, \Psi))$ to $\mathcal{A}_2$.

$\mathsf{Ch-Link}_b(\mathsf{lm}, \mathbf{\Sigma})$**:** $\mathcal{B}_{\mathsf{RS}}$ performs the operations of Link with linking secret ls_{j_β} to obtain π_l – note that it needn't know sk_{i_β} for this. Then $\mathcal{B}_{\mathsf{RS}}$ adds $(\mathsf{lm}, \mathbf{\Sigma})$ to CLNK: $\mathsf{CLNK} \leftarrow \mathsf{CLNK} \cup \{(\mathsf{lm}, \mathbf{\Sigma})\}$. Finally it sends π_l to $\mathcal{A}_2$.

After a polynomial number of queries, $\mathcal{A}_2$ outputs a bit d. $\mathcal{B}_{\mathsf{RS}}$ returns 1 to its own challenger if $d = \beta$, and 0 otherwise.

Analysis: Let the bit chosen by the challenger in the game for $\mathcal{B}_{\mathsf{RS}}$ be $\tilde{\beta}$. If $\tilde{\beta} = 0$, then inputs to $\mathcal{A}$ are identically distributed to in Game 0, and if $\tilde{\beta} = 1$, then inputs to $\mathcal{A}$ are identically distributed to in Game 1.

Therefore $|Pr[S_1] - Pr[S_0]| \leq \epsilon_{\mathsf{RS}}$, where ϵ_{RS} is the advantage in the anonymity game for RS.

Precisely, assuming there exists an algorithm $\mathcal{A}$ which wins in Game 1, we show how to construct an algorithm $\mathcal{B}_{\mathsf{AKRS}}$, breaking the class hiding requirement of AKRS.

Algorithm $\mathcal{B}_{\mathsf{AKRS}}$ gets as input AKRS public keys $\mathsf{pk}_0, \mathsf{pk}_1$. It randomly samples j_0^* and j_1^* from $[n]$ (such that $j_0^* \neq j_1^*$). It samples $(\cdot, \mathsf{ls}_j) \leftarrow \mathsf{AKRS.KGen}(1^\lambda)$ for $j \in [k] \backslash \{j_0^*, j_1^*\}$. For $i \in [n]$, it samples $(\mathsf{sk}_i, \mathsf{vk}_i) \leftarrow \mathsf{RS.KGen}(1^\lambda)$, and initialises empty lists LNK and CLNK. It calls upon $\mathcal{A}_1(\mathsf{choose}, \mathsf{vk}_1, \ldots, \mathsf{vk}_n)$, and answers $\mathcal{A}_1$'s oracle queries as follows. $\mathcal{B}_{\mathsf{AKRS}}$ knows all the secret keys so can perform Corr as normal.

$\mathsf{OSign}^{\mathbf{ucl\text{-}r}}(i, j, R, m, \mathsf{scp})$**:** If $j = j_b^*$ then $\mathcal{B}_{\mathsf{AKRS}}$ queries their oracle $\mathsf{OChgPK}(b, \mathsf{scp})$ to obtain nym. They compute $\Omega \leftarrow \mathsf{RS.Sig}(\mathsf{sk}_i, R, m||\mathsf{scp}||\mathsf{nym})$. They query their oracle $\mathsf{OSign}(b, m||\mathsf{scp}||R||\Omega, \mathsf{nym}, \mathsf{scp})$ to obtain Ψ. They update the lists as normal and return $(\mathsf{nym}, \sigma = (\Omega, \Psi))$. Otherwise, $\mathcal{B}$ performs the oracle as in the original experiment.

$\mathsf{OLink}(j, \mathsf{lm}, \mathbf{\Sigma})$**:** Parse $\mathbf{\Sigma}$ as $\{(m_i, \mathsf{scp}_i, R_i, \sigma_i, \mathsf{nym}_i)\}_{i \in [m]}$. If $j = j_b^*$ then $\mathcal{B}_{\mathsf{AKRS}}$ first checks if $\exists (i_0', i_1') \in [m]$ such that $\mathsf{scp}_{i_0'} = \mathsf{scp}_{i_1'}$ and $i_0' \neq i_1'$ and if so returns $\perp$. Then for $i' \in [m]$ $\mathcal{B}_{\mathsf{AKRS}}$ checks that the output of its oracle $\mathsf{OChgPK}(b, \mathsf{scp}_{i'})$ is the same as $\mathsf{nym}_{i'}$, returning $\perp$ otherwise. $\mathcal{B}_{\mathsf{AKRS}}$ then computes $\tilde{\mathsf{nym}} \leftarrow \mathsf{Accum}(\mathsf{nym}_1, \ldots, \mathsf{nym}_k)$. Finally, $\mathcal{B}_{\mathsf{AKRS}}$ queries its oracle $\mathsf{OSign}(b, (\mathsf{lm}, \mathbf{\Sigma}), \tilde{\mathsf{nym}}, \{scp_1, \cdots, \mathsf{scp}_m\})$ to obtain π_l. If $j \notin \{j_0^*, j_1^*\}$, then $\mathcal{B}_{\mathsf{AKRS}}$ computes $\pi_l \leftarrow \mathsf{Link}(\mathsf{ls}_j, \mathsf{lm}, \mathbf{\Sigma})$ to obtain π_l. $\mathcal{B}$ adds $(\mathsf{lm}, \mathbf{\Sigma})$ to LNK for user index j: $\mathsf{LNK}[j] \leftarrow \mathsf{LNK}[j] \cup \{(\mathsf{lm}, \mathbf{\Sigma})\}$. It sends π_l to $\mathcal{A}_1$.

$\mathsf{OLS}(j)$**:** If $j \notin \{j_0^*, j_1^*\}$ then return ls_j. Otherwise $\mathcal{B}_{\mathsf{AKRS}}$ aborts.

After a polynomial number of queries, $\mathcal{A}_1$ outputs $(i_0, i_1), (j_0, j_1), \mathsf{state}$. If $j_0 \neq j_0^*$ or $j_1 \neq j_1^*$ then $\mathcal{B}_{\mathsf{AKRS}}$ aborts.

Algorithm $\mathcal{B}_{\mathsf{AKRS}}$ now calls upon $\mathcal{A}_2(\mathsf{guess}, \mathsf{state})$, and answers $\mathcal{A}_2$'s oracle queries as follows.

$\mathsf{OSign}^{\mathbf{ucl\text{-}r}}, \mathsf{OLink}, \mathsf{OLS}, \mathsf{Corr}$**:** $\mathcal{B}$ behaves as it did with $\mathcal{A}_1$.

$\mathsf{Ch} - \mathsf{Sign}_b(R, m, \mathsf{scp})$**:** $\mathcal{B}_{\mathsf{AKRS}}$ queries their oracle $\mathsf{OChgPK}(2, \mathsf{scp})$ to obtain nym. They compute $\Omega \leftarrow \mathsf{RS.Sig}(\mathsf{sk}_{i_0}, R, m||\mathsf{scp}||\mathsf{nym})$. They query their oracle $\mathsf{OSign}(2, m||\mathsf{scp}||R||\Omega, \mathsf{nym}, \mathsf{scp})$ to obtain Ψ. They update the lists as normal and return $(\mathsf{nym}, \sigma = (\Omega, \Psi))$.

$\mathsf{Ch} - \mathsf{Link}_b(\mathsf{lm}, \mathbf{\Sigma})$**:** Parse $\mathbf{\Sigma}$ as $\{(m_i, \mathsf{scp}_i, R_i, \sigma_i, \mathsf{nym}_i)\}_{i \in [m]}$. $\mathcal{B}_{\mathsf{AKRS}}$ first checks if $\exists (i_0', i_1') \in [m]$ such that $\mathsf{scp}_{i_0'} = \mathsf{scp}_{i_1'}$ and $i_0' \neq i_1'$ and if so returns $\perp$. Then for $i' \in [m]$ $\mathcal{B}_{\mathsf{AKRS}}$ checks that the output of its oracle $\mathsf{OChgPK}(2, \mathsf{scp}_{i'})$ is the same as $\mathsf{nym}_{i'}$, returning $\perp$ otherwise. $\mathcal{B}_{\mathsf{AKRS}}$ then computes $\tilde{\mathsf{nym}} \leftarrow \mathsf{Accum}(\mathsf{nym}_1, \ldots, \mathsf{nym}_k)$. Finally, $\mathcal{B}_{\mathsf{AKRS}}$ queries its oracle $\mathsf{OSign}(2, (\mathsf{lm}, \mathbf{\Sigma}), \tilde{\mathsf{nym}}, \{scp_1, \cdots, \mathsf{scp}_k\})$ to obtain π_l. $\mathcal{B}_{\mathsf{AKRS}}$ adds $(\mathsf{lm}, \mathbf{\Sigma})$ to CLNK: $\mathsf{CLNK} \leftarrow \mathsf{CLNK} \cup \{(\mathsf{lm}, \mathbf{\Sigma})\}$. It sends π_l to $\mathcal{A}_1$.

After a polynomial number of queries, $\mathcal{A}_2$ outputs a bit d. Let us denote $\tilde{\beta}$ the random bit chosen by $\mathcal{B}_{\mathsf{AKRS}}$'s challenger. We then check the lists CLNK, $\mathsf{LNK}[j_0^*]$ and $\mathsf{LNK}[j_1^*]$, for any signatures that were not output by one of the signing oracles. If there exists $(m, \mathsf{scp}, R, \sigma, \mathsf{nym}) \in \mathbf{\Sigma}$, $b' \in \{0, 1\}$ such that $\mathbf{\Sigma} \in \mathsf{LNK}[j_{b'}^*] \cup \mathsf{CLNK}$, and $(m, \mathsf{scp}, R, \sigma, \mathsf{nym}) \notin \mathsf{CSIG} \cup \mathsf{SIG}[\cdot, j_{b'}^*]$, then $\mathcal{B}_{\mathsf{AKRS}}$ aborts. Otherwise, $\mathcal{B}_{\mathsf{AKRS}}$ forwards d to its own challenger, as the answer to its' own challenge.

Analysis: We now consider the probability that $\mathcal{B}_{\mathsf{AKRS}}$ aborts early returning 0. $\mathcal{A}$ chooses $(j_0, j_1) \neq (j_0^*, j_1^*)$ with probability $1 - \frac{2}{n(n-1)}$. Assuming that $\mathcal{A}$ is successful, then $\mathcal{A}$ will not query j_0^* or j_1^* to the Corr oracle.

If $\mathcal{B}_{\mathsf{AKRS}}$ aborts because there is a signature submitted to the challenge linking oracle or the linking oracle for j_0^* or j_1^*, that does not originate from the challenge signature oracle or the signing oracle for j_0^* or j_1^*, then this would clearly break the unforgeability of the AKRS.

Assuming that $\mathcal{B}_{\mathsf{AKRS}}$ never aborts early, we now show that they win, provided that $\mathcal{A}$ is successful. It is easy to see that $\mathcal{B}$ perfectly simulates Game 1 for $b = \tilde{\beta}$, where b is the bit chosen in Game 1. Therefore, if $\mathcal{A}$ guesses b correctly, then $\mathcal{B}$ guesses $\tilde{\beta}$ correctly. To win in the class hiding experiment we also need that scopes queried to the OChgPK and OSign oracles for $a \in \{0,1\}$ were not also queried to those oracles for $a = 2$. For this we need that scopes that were queried by $\mathcal{A}$ to the oracles $\mathsf{OSign}^{\mathsf{ucl\text{-}r}}$ or OLink for j_0^*, j_1^* were not queried to $\mathsf{Ch-Sign}_b$ or $\mathsf{Ch-Link}_b$. Due to line 9 of the Anonymity requirement, clearly the same scopes were not queried to both $\mathsf{OSign}^{\mathsf{ucl\text{-}r}}$ for j_0^*, j_1^* and $\mathsf{Ch-Sign}_b$. As $\mathcal{B}_{\mathsf{AKRS}}$ has not aborted, all signatures in CLNK must originate from $\mathsf{CSIG} \cup \mathsf{SIG}[\cdot, j_{b'}^*]$ for $b' \in \{0,1\}$. Therefore, all signatures input to CLNK with scopes that have been input to $\mathsf{OSign}^{\mathsf{ucl\text{-}r}}$, must have originally come from $\mathsf{OSign}^{\mathsf{ucl\text{-}r}}$. However, due to line 13 in the anonymity experiment, signatures output by $\mathsf{OSign}^{\mathsf{ucl\text{-}r}}$ cannot be input to CLNK. Therefore, no scopes were input to both $\mathsf{OSign}^{\mathsf{ucl\text{-}r}}$ and CLNK. As $\mathcal{B}_{\mathsf{AKRS}}$ has not aborted, all signatures in $\mathsf{LNK}[j_0^*]$ or $\mathsf{LNK}[j_1^*]$ must originate from $\mathsf{CSIG} \cup \mathsf{SIG}[j_{b'}^*]$ for $b' \in \{0,1\}$. Therefore, all signatures input to OLink with scopes that have been input to $\mathsf{Ch-Sign}_b$, must have originally come from $\mathsf{Ch-Sign}_b$. However, due to line 13 in the anonymity experiment, signatures output by $\mathsf{Ch-Sign}_b$ cannot be input to OLink. Therefore, no scopes were input to both $\mathsf{Ch-Sign}_b$ and OLink. Finally, we consider scopes input to both OLink for j_0^*, j_1^* and $\mathsf{Ch-Link}_b$. As discussed, all scopes input to $\mathsf{Ch-Link}_b$ must originate from $\mathsf{Ch-Sign}_b$ and all scopes input to OLink must originate from $\mathsf{OSign}^{\mathsf{ucl\text{-}r}}$. Therefore, again due to line 9, no scope will be input to both oracles.

We therefore have that provided $\mathcal{A}$ is successful, the probability of $\mathcal{B}_{\mathsf{AKRS}}$ aborting early without winning is negligible, and otherwise they will be successful. Therefore, our construction satisfies anonymity.

Signature unforgeability: We show that if RS is unforgeable, then UCAL satisfies signature unforgeability.

Precisely, assuming there exists an algorithm $\mathcal{A}$ which breaks the signature unforgeability of UCAL with probability ϵ, we show how to construct an algorithm $\mathcal{B}$, breaking the unforgeability of RS.

Algorithm $\mathcal{B}$ gets as input public ring signature keys $\mathsf{vk}_1, \ldots, \mathsf{vk}_n$ from its challenger for RS. For $i \in [k]$, it sets $\mathsf{ls}_i \leftarrow \mathsf{UCAL.GenLinkSec}(\lambda)$. Finally, $\mathcal{B}$ forwards $\mathsf{vk}_1, \ldots, \mathsf{vk}_n$ to $\mathcal{A}$, and answers the latter's oracle queries as follows. $\mathcal{B}$ knows all the linking secrets so can perform OLink and OLS as normal.

$\mathsf{OSign}^{\mathsf{ucl\text{-}r}}(i, j, R, m, \mathsf{scp})$**:** $\mathcal{B}$ computes $\mathsf{nym} \leftarrow \mathsf{ChgPK}(\mathsf{ls}_j, \mathsf{scp})$. Next, $\mathcal{B}$ calls upon oracle OSign for RS, with input $(i, R, m||scp||\mathsf{nym})$, from which it gets output Ω, and finally it computes $\Psi \leftarrow \mathsf{AKRS.Sig}(\mathsf{ls}_j, \mathsf{nym}, m||\mathsf{scp}||R||\Omega)$. $\mathcal{B}$ then sends $(\mathsf{nym}, (\Omega, \Psi))$ to $\mathcal{A}$.

$\mathsf{Corr}(i)$**:** $\mathcal{B}$ queries i to its own corruption oracle for RS, and forwards the received sk_i to $\mathcal{A}$.

Now with probability ϵ, $\mathcal{A}$ outputs a forged signature $\Sigma^* := (m^*, \mathsf{scp}^*, R^*, (\Omega^*, \Psi^*), \mathsf{nym}^*)$, and $\mathcal{B}$ forwards $(\Omega^*, R^*, m^*||scp^*||R^*||\mathsf{nym}^*)$ as its forgery for RS.

Analysis: For $\mathcal{A}$'s output to be a forgery, the following must hold.

1. None of the keys in R^* were corrupted.

2. $\mathsf{Vf}(\Sigma^*) = 1$, and hence $\mathsf{RS.Vf}(\Omega^*, R^*, m^*||scp^*||\mathsf{nym}^*) = 1$.
3. $\forall i \in [n]$, no signature on m^* for scope scp^*, ring R^*, and user i has been queried to $\mathcal{B}$.

Now observe that, from item 3, it holds that $\mathcal{B}$ has not queried any instance of oracle OSign for RS with input $(i, R^*, m^*||scp^*||\mathsf{nym}^*)$. Combined with items 1 and 2, it is clear that $\mathcal{A}$'s forgery for UCAL is successful, then $\mathcal{B}$'s forgery for RS is also successful.

Link unforgeability We show that if the AKRS is unforgeable, then UCAL satisfies link unforgeability.

Precisely, assuming there exists an algorithm $\mathcal{A}$ which breaks the link unforgeability of UCAL with probability ϵ_{UCAL}, we show how to construct an algorithm $\mathcal{B}$, breaking the unforgeability of AKRS.

Algorithm $\mathcal{B}$ gets as input an AKRS public key pk from its challenger for AKRS. $\mathcal{B}$ randomly chooses $i^* \leftarrow\$ [k]$ and sets $\mathsf{vk}_{i^*} \leftarrow \mathsf{pk}$. For $i \in [n]$, it sets $(\mathsf{sk}_i, \mathsf{vk}_i) \leftarrow \mathsf{UCAL.KGen}(\lambda)$. For $i \in [k]\backslash\{i^*\}$, it sets $\mathsf{ls}_i \leftarrow \mathsf{UCAL.GenLinkSec}(\lambda)$. Finally, $\mathcal{B}$ forwards $\mathsf{vk}_1, \ldots, \mathsf{vk}_n$ to $\mathcal{A}$, and answers the latter's oracle queries as follows. $\mathcal{B}$ knows all the secret keys so can perform Corr as normal.

$\mathsf{OSign}^{\mathbf{ucl\text{-}r}}(i, j, R, m, \mathsf{scp})$**:** If $j \neq i^*$ proceed as normal. Otherwise, $\mathcal{B}$ calls upon oracle ChgPK for AKRS, with input scp, from which it gets output nym. Next, $\mathcal{B}$ computes $\Omega \leftarrow \mathsf{RS.Sig}(\mathsf{sk}_i, R, m||scp||\mathsf{nym})$. Finally $\mathcal{B}$ calls upon oracle OSign for AKRS, with input $(m||\mathsf{scp}||R||\Omega, \mathsf{nym}, scp)$, from which it gets output Ψ. $\mathcal{B}$ then sends $(\mathsf{nym}, (\Omega, \Psi))$ to $\mathcal{A}$.

$\mathsf{OLink}(i, \mathsf{lm}, \mathbf{\Sigma})$**:** If $i \neq i^*$ proceed as normal. Otherwise, letting $\mathbf{\Sigma} = \{(m_i, \mathsf{scp}_i, R_i, \sigma_i, \mathsf{nym}_i)\}_{i\in[m]}$, $\mathcal{B}$ computes $\mathsf{n\tilde{y}m} \leftarrow \mathsf{Accum}(\mathsf{nym}_1, \ldots, \mathsf{nym}_m)$. $\mathcal{B}$ calls upon oracle OSign for AKRS, with input $lm||\mathbf{\Sigma}, \mathsf{n\tilde{y}m}, \{\mathsf{scp}_i\}_{i\in[m]}$, from which it gets output π_l, and adds $(\mathsf{lm}, \mathbf{\Sigma})$ to LNK for user index i: $\mathsf{LNK}[i] \leftarrow \mathsf{LNK}[i] \cup \{(\mathsf{lm}, \mathbf{\Sigma})\}$. It sends π_l to $\mathcal{A}$.

$\mathsf{OLS}(i)$**:** If $i \neq i^*$ proceed as normal. Otherwise, $\mathcal{B}$ aborts.

Now with probability ϵ_{UCAL}, $\mathcal{A}$ outputs a forged link proof $(\mathsf{lm}, \mathbf{\Sigma}, \pi_l)$, let $\mathbf{\Sigma} = \{(m_i, \mathsf{scp}_i, R_i, \sigma_i, \mathsf{nym}_i)\}_{i\in[m]}$. $\mathcal{B}$ computes $\mathsf{n\tilde{y}m} \leftarrow \mathsf{Accum}(\mathsf{nym}_1, \ldots, \mathsf{nym}_m)$. $\mathcal{B}$ forwards $(\mathsf{n\tilde{y}m}, \{\mathsf{scp}_1, \cdots, \mathsf{scp}_m\}, \mathsf{lm}||\mathbf{\Sigma}, \pi_l)$ as its forgery for AKRS.

Analysis: For $\mathcal{A}$'s output to be a forgery, the following must hold.

1. $\mathsf{lm}, \mathbf{\Sigma}$ were not queried to OLink.
2. $\mathsf{Vf}(\mathsf{lm}, \mathbf{\Sigma}, \pi_l) = 1$, and hence $\mathsf{AKRS.Vf}(\mathsf{n\tilde{y}m}, \mathsf{lm}||\mathbf{\Sigma}, \pi_l) = 1$.
3. $\exists \tilde{i} \in [k]$ such that $\tilde{i} \notin J$ and for all $j \in [m]$ $\Sigma_j \in \mathsf{SIG}[\tilde{i}]$.

Assume that $i^* = \tilde{i}$, which occurs with probability $1/k$. In this case $\mathcal{B}$ will not abort because i^* was not queried to the OLS oracle. We also have that for all $j \in [m]$, nym_j is a correctly formed public key for the scope scp_j because this was honestly generated in the OSign oracle. This means that $n\tilde{y}m = \mathsf{Accum}(\mathsf{ChgPK}(\mathsf{sk}, \mathsf{scp}_1), \cdots, \mathsf{ChgPK}(\mathsf{sk}, \mathsf{scp}_m))$ where sk is the secret key corresponding to pk (unknown to $\mathcal{B}$). Now observe that, from item 1, it holds that the signature output is a valid AKRS signature. $\mathcal{B}$ has not queried any instance of oracle OSign for AKRS with input $(\mathsf{lm}||\mathbf{\Sigma}, \cdot)$. It is clear that if $\mathcal{A}$'s forgery for UCAL is successful, then $\mathcal{B}$'s forgery for AKRS is also successful. Therefore, $\mathcal{B}$ wins with probability $\epsilon_{\mathsf{UCAL}}/k$

Signature non-frameability We show that if the AKRS is unforgeable, then UCAL satisfies signature non-frameability.

Precisely, assuming there exists an algorithm $\mathcal{A}$ which breaks the signature non-frameability of UCAL with probability ϵ_{UCAL}, we show how to construct an algorithm $\mathcal{B}$, breaking the unforgeability of AKRS.

Algorithm $\mathcal{B}$ gets as input an AKRS public key pk from its challenger for AKRS. $\mathcal{B}$ randomly chooses $i^* \leftarrow\$ [k]$. For $i \in [n]$, it sets $(\mathsf{sk}_i, \mathsf{vk}_i) \leftarrow \mathsf{UCAL.KGen}(\lambda)$. For $i \in [k] \backslash \{i^*\}$, it sets $\mathsf{ls}_i \leftarrow \mathsf{UCAL.GenLinkSec}(\lambda)$. Finally, $\mathcal{B}$ forwards $\mathsf{vk}_1, \ldots, \mathsf{vk}_n$ to $\mathcal{A}$, and answers the latter's oracle queries as follows. $\mathcal{B}$ knows all the secret keys so can perform Corr as normal.

$\mathsf{OSign}^{\mathbf{ucl\text{-}r}}(i, j, R, m, \mathsf{scp})$**:** If $j \neq i^*$ proceed as normal. Otherwise, $\mathcal{B}$ calls upon oracle ChgPK for AKRS, with input scp, from which it gets output nym. Next, $\mathcal{B}$ computes $\Omega \leftarrow \mathsf{RS.Sig}(\mathsf{sk}_i, R, m||scp||\mathsf{nym})$. Finally $\mathcal{B}$ calls upon oracle OSign for AKRS, with input $(m||\mathsf{scp}||R||\Omega, \mathsf{nym}, scp)$, from which it gets output Ψ. $\mathcal{B}$ then sends $(\mathsf{nym}, (\Omega, \Psi))$ to $\mathcal{A}$.

$\mathsf{OLink}(i, \mathsf{lm}, \mathbf{\Sigma})$**:** If $i \neq i^*$ proceed as normal. Otherwise, letting $\mathbf{\Sigma} = \{(m_i, \mathsf{scp}_i, R_i, \sigma_i, \mathsf{nym}_i)\}_{i \in [m]}$, $\mathcal{B}$ computes $\tilde{\mathsf{nym}} \leftarrow \mathsf{Accum}(\mathsf{nym}_1, \ldots, \mathsf{nym}_m)$. $\mathcal{B}$ calls upon oracle OSign for AKRS, with input $lm||\mathbf{\Sigma}, \tilde{\mathsf{nym}}, \{\mathsf{scp}_i\}_{i \in [m]}$, from which it gets output π_l, and adds $(\mathsf{lm}, \mathbf{\Sigma})$ to LNK for user index i: $\mathsf{LNK}[i] \leftarrow \mathsf{LNK}[i] \cup \{(\mathsf{lm}, \mathbf{\Sigma})\}$. It sends π_l to $\mathcal{A}$.

$\mathsf{OLS}(i)$**:** If $i \neq i^*$ proceed as normal. Otherwise, $\mathcal{B}$ aborts.

Now with probability ϵ_{UCAL}, $\mathcal{A}$ outputs a forged signature $\Sigma = (m, \mathsf{scp}, R, \sigma, \mathsf{nym})$, where $\sigma = (\Omega, \Psi)$. $\mathcal{B}$ forwards $(\mathsf{nym}, \{\mathsf{scp}\}, m||scp||R||\Omega, \Psi)$ as its forgery for AKRS.

Analysis: For $\mathcal{A}$'s output to be a forgery, there must exists $\tilde{i} \in [k]$ such that the following holds.

1. $(m, \mathsf{scp}, R, \mathsf{nym}, \cdot) \notin \mathsf{SIG}[\tilde{i}]$.
2. $\mathsf{Vf}(\Sigma) = 1$, and hence $\mathsf{AKRS.Vf}(\mathsf{nym}, m||\mathsf{scp}||R||\Omega, \psi) = 1$.
3. $\tilde{i} \notin J$.
4. $(\cdot, \mathsf{scp}, \cdot, \cdot, \mathsf{nym}) \in \mathsf{SIG}[\cdot, \tilde{i}]$.

Assume that $i^* = \tilde{i}$, which occurs with probability $1/k$. In this case $\mathcal{B}$ will not abort because i^* was not queried to the OLS oracle. We also have that $(\cdot, \mathsf{scp}, \cdot, \cdot, \mathsf{nym}) \in \mathsf{SIG}[\cdot, i^*]$, and so $\mathsf{nym} = \mathsf{ChgPK}(\mathsf{sk}, \mathsf{scp})$ where sk is the secret key corresponding to pk (unknown to $\mathcal{B}$), because this was honestly generated in the OSign oracle. Now observe that, from item 2, it holds that the signature output is a valid AKRS signature. $\mathcal{B}$ has not queried any instance of oracle OSign for AKRS with input $(m||\mathsf{scp}||R||\Omega, \mathsf{scp})$. It is clear that if $\mathcal{A}$'s forgery for UCAL is successful, then $\mathcal{B}$'s forgery for AKRS is also successful. Therefore, $\mathcal{B}$ wins with probability $\epsilon_{\mathsf{UCAL}}/k$

Link non-frameability We show that if the AKRS is accumulation sound, then UCAL satisfies link non-frameability.

Precisely, assuming there exists an algorithm $\mathcal{A}$ which breaks the link non-frameability of UCAL with probability ϵ_{UCAL}, we show how to construct an algorithm $\mathcal{B}$, breaking the accumulation soundness of AKRS.

Algorithm $\mathcal{B}$ gets as input AKRS key pairs $((\mathsf{ls}_1, \mathsf{pk}_1), \cdots, (\mathsf{ls}_k, \mathsf{pk}_k))$ from its challenger for AKRS. For $i \in [n]$, it samples $(\mathsf{sk}_i, \mathsf{vk}_i) \leftarrow \mathsf{UCAL.KGen}(\lambda)$. Finally, $\mathcal{B}$ forwards $\mathsf{vk}_1, \ldots, \mathsf{vk}_n$ to $\mathcal{A}$, and answers the latter's oracle queries as normal, which is possible because $\mathcal{B}$ knows all the secret keys and linking secrets.

Now with probability ϵ_{UCAL}, $\mathcal{A}$ outputs a forged link proof $(\mathsf{lm}, \mathbf{\Sigma}, \pi_l)$, where $\mathbf{\Sigma} = \{(m_i, \mathsf{scp}_i, R_i, \sigma_i = (\Omega_i, \Psi_i), \mathsf{nym}_i)\}_{i \in [m]}$.

Let $A, B = \emptyset$. For all $j \in [m]$, if $\exists i \in [k] : \Sigma_j \in \mathsf{SIG}[\cdot, i]$ or $\mathsf{Identify}(\mathsf{ls}_i, \Sigma_j) = 1$ let $i_j = i$ and $\mathcal{I} \leftarrow \mathcal{I} \cup [i]$, $A \leftarrow A \cup \mathsf{scp}_j$. Otherwise $i_j = k + 1$, $B \leftarrow B \cup (\mathsf{scp}_j, \mathsf{nym}_j, m_j || scp_j || R_j || \Omega_j, \Psi_j,)$.

$\mathcal{B}$ computes $\tilde{\mathsf{nym}} \leftarrow \mathsf{Accum}(\mathsf{nym}_1, \ldots, \mathsf{nym}_m)$. $\mathcal{B}$ forwards $(B, \mathcal{I}, A, \mathsf{lm} || \boldsymbol{\Sigma}, \pi_l)$ as its forgery for AKRS.

Analysis: For $\mathcal{A}$'s output to be a forged link proof, it must satisfy $\mathsf{VerifyLink}(\mathsf{lm}, \boldsymbol{\Sigma}, \pi_l) = 1$; and the signatures in $\boldsymbol{\Sigma}$ were *not all* signed for (or identified as signed for) the same linking secret ls, i.e $\exists (j_1, j_2) \in [m]$ such that $i_{j_1} \neq i_{j_2}$.

For $\mathcal{B}$ to be successful in the accumulation soundness game, we need 4 conditions to be satisfied.

Firstly, we need that $\mathcal{I} \neq \emptyset$. If $\mathcal{I} = \emptyset$, for all $j \in [m]$, $i_j = k + 1$, in which case $\mathcal{A}$ would not be successful in the link non-frameability game.

We also need that signatures in the set B are valid. For $\mathsf{VerifyLink}(\tilde{\mathsf{nym}}, \mathsf{lm} || \boldsymbol{\Sigma}, \pi_l) = 1$, we must have that each signature in $\boldsymbol{\Sigma}$ is a valid ring signature. This ensures that for all $j \in [m]$, Ψ_j is a valid AKRS signature.

We additionally need that π_l is a valid AKRS signature on message $\mathsf{lm} || \boldsymbol{\Sigma}$ and public key $\tilde{\mathsf{nym}}$. This is again satisfied because $\mathsf{VerifyLink}(\tilde{\mathsf{nym}}, \mathsf{lm} || \boldsymbol{\Sigma}, \pi_l) = 1$.

Finally we need that either $\exists (i, j) \in \mathcal{I}$ s.t. $\mathsf{ls}_i \neq \mathsf{ls}_j$ or $\exists (\mathsf{scp}^*, \mathsf{nym}^*, m^* || scp^* || R^* || \Omega^*, \Psi^*) \in B$ such that $\forall j \in \mathcal{I}, \mathsf{ChgPK}(ls_j, scp^*) \neq \mathsf{nym}^*$. Say this does not hold, then for all $(j_1, j_2) \in I$, $i_{j_1} = i_{j_2}$ and B is the emptyset. Therefore, $\mathcal{A}$ would not be successful in the link non–frameability game.

We therefore have the output of $\mathcal{B}$ is a successful forgery in the accumulation soundness game.

5.2 Construction for Ring Signatures with User Controlled Linkability

We provide a generic construction for a ring signature scheme with user controlled linking. We build upon a NIZK for the language $\mathcal{L} = \{(\mathsf{nym}, \mathsf{scp}, (\mathsf{vk}_i)_{i \in [n]}; \mathsf{sk}) : \mathsf{nym} = \mathsf{AKRS.ChgPK}(\mathsf{sk}, \mathsf{scp}) \wedge \bigvee_{i \in [n]} (\mathsf{vk}_i, \mathsf{sk}) = \mathsf{AKRS.KGen}(1^\lambda)\}$ and an AKRS $\mathsf{AKRS} = (\mathsf{KGen}, \mathsf{ChgPK}, \mathsf{Sig}, \mathsf{Vf}, \mathsf{Accum})$, as defined in Section 4.

Our RS-UCL scheme UCL, given in Figure 4, works as follows. Key generation (instead of the linking secret generation) is now identical to key generation for an AKRS. When signing, the pseudonym, similarly to the UCAL construction, is the AKRS public key randomised with the scope and now with the ring signature secret key corresponding to the AKRS secret key. For the RS-UCL model, we additionally need to prove that the pseudonym was generated with a secret key corresponding to one of the public keys in the ring to ensure signature unforgeability. This ensures that if an adversary holds n keys in the ring, they can only generate n different pseudonyms and so output n unlinked signatures. We therefore attach a non-interactive zero-knowledge proof of knowledge that attests to this, which also fulfills the role of the ring signature in the UCAL construction. The AKRS signature from the UCAL construction is now also not needed to ensure signature non-frameability, because it is necessary to know the secret key corresponding to a pseudonym in order to generate the NIZK. Explicit linking can be done in exactly the same way as the UCAL construction with link non-frameability and link unforgeability satisfied in the same way. Anonymity similarly follows from the class hiding property of the AKRS, and now also from the zero knowledge property of the NIZK.

Theorem 5.2. *If* AKRS *is an anonymous key re-randomisable signature scheme and* NIZK *is a non-interactive zero knowledge proof of knowledge satisfying simulation sound extractability, then* UCL *is a UCL ring signature.*

$\mathsf{KGen}(1^\lambda)$

$(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{AKRS.KGen}(1^\lambda)$
return $(\mathsf{sk}, \mathsf{vk})$

$\mathsf{Sig}(\mathsf{sk}, R, m, \mathsf{scp})$

Parse $(\mathsf{vk}_1, \ldots, \mathsf{vk}_n) \leftarrow R$
$\mathsf{nym} \leftarrow \mathsf{AKRS.ChgPK}(\mathsf{sk}, \mathsf{scp})$
$\sigma \leftarrow \mathsf{NIZK}\{\mathsf{sk} : \mathsf{nym} = \mathsf{AKRS.ChgPK}(\mathsf{sk}, \mathsf{scp})$
$\wedge \bigvee_{i \in [n]} (\mathsf{vk}_i, \mathsf{sk}) \in \mathsf{AKRS.KGen}(1^\lambda)\}(m)$
return (nym, σ)

$\mathsf{Vf}(m, \mathsf{scp}, R, \sigma, \mathsf{nym})$

Parse $(\mathsf{vk}_1, \ldots, \mathsf{vk}_n) \leftarrow R$
Verify NIZK σ wrt statement $(\mathsf{nym}, \mathsf{scp}, m,$
$(\mathsf{vk}_1, \ldots, \mathsf{vk}_n))$
// Note m was auxilary input to the NIZK σ

$\mathsf{Link}(\mathsf{sk}, \mathsf{lm}, \mathbf{\Sigma})$

Parse $\{(m_i, \mathsf{scp}_i, R_i, \sigma_i, \mathsf{nym}_i)\}_{i \in [k]} \leftarrow \mathbf{\Sigma}$
if $\exists i, j \in [k], (i \neq j) \wedge (\mathsf{scp}_i = \mathsf{scp}_j)$
  then return $\perp$
if $\exists i \in [k], \mathsf{nym}_i \neq \mathsf{AKRS.ChgPK}(\mathsf{sk}, \mathsf{scp}_i)$
  then return $\perp$
if $\exists i \in [k], \mathsf{Vf}(m_i, \mathsf{scp}_i, R_i, \sigma_i, \mathsf{nym}_i) = 0$
  then return $\perp$
$\tilde{\mathsf{nym}} \leftarrow \mathsf{Accum}(\mathsf{nym}_1, \ldots, \mathsf{nym}_k)$
$\pi_l \leftarrow \mathsf{AKRS.Sig}(\mathsf{sk}, \tilde{\mathsf{nym}}, \mathsf{lm} || \mathbf{\Sigma})$
return π_l

$\mathsf{VerifyLink}(\mathsf{lm}, \mathbf{\Sigma}, \pi_l)$

Parse $\{(m_i, \mathsf{scp}_i, R_i, \sigma_i, \mathsf{nym}_i)\}_{i \in [k]} \leftarrow \mathbf{\Sigma}$
if $\exists i, j \in [k], (i \neq j) \wedge (\mathsf{scp}_i = \mathsf{scp}_j)$
  then return 0
if $\exists i \in [k], \mathsf{Vf}(m_i, \mathsf{scp}_i, R_i, \sigma_i, \mathsf{nym}_i) = 0$
  then return 0
$\tilde{\mathsf{nym}} \leftarrow \mathsf{Accum}(\mathsf{nym}_1, \ldots, \mathsf{nym}_k)$
return $\mathsf{AKRS.Vf}(\tilde{\mathsf{nym}}, \mathsf{lm} || \mathbf{\Sigma}, \pi_l)$

Fig. 4. RS-UCL from AKRS and non-interactive zero knowledge proofs of knowledge.

Proof

Verification correctness: Follows from the correctness of AKRS, and the correctness of NIZK.

Linking correctness: Follows from the correctness of AKRS.

Anonymity: We show that if AKRS is class hiding and accumulation sound and the NIZK is zero knowledge, then UCL satisfies adaptive anonymity against adversarially chosen keys.

Precisely, assuming there exists an algorithm $\mathcal{A}$ which breaks the anonymity of UCL with probability ϵ, we show how to construct an algorithm $\mathcal{B}$, breaking the class hiding requirement of AKRS with probability $\epsilon + \mathsf{negl}(\lambda)$.

Algorithm $\mathcal{B}$ gets as input AKRS public keys $\mathsf{pk}_0, \mathsf{pk}_1$. It randomly samples i_0^* and i_1^* from $[n]$ (such that $i_0^* \neq i_1^*$) and sets $\mathsf{vk}_{i_0^*} \leftarrow \mathsf{pk}_0$ and $\mathsf{vk}_{i_1^*} \leftarrow \mathsf{pk}_1$. For $i \in [n] \backslash \{i_0^*, i_1^*\}$, it samples $(\mathsf{sk}_i, \mathsf{vk}_i) \leftarrow \mathsf{AKRS.KGen}(1^\lambda)$, and initialises empty lists LNK and CLNK. It calls upon $\mathcal{A}_1(\mathsf{choose}, \mathsf{vk_1}, \ldots, \mathsf{vk_n})$, and answers $\mathcal{A}_1$'s oracle queries as follows.

$\mathsf{OSign}^{\mathsf{ucl\text{-}r}}(i, R, m, \mathsf{scp})$**:** If $i = i_b^*$ then $\mathcal{B}$ queries their oracle $\mathsf{OChgPK}(b, \mathsf{scp})$ to obtain nym. They then simulate σ making use of the zero knowledge property of the NIZK. They update the lists as normal and return (nym, σ). Otherwise, $\mathcal{B}$ performs the oracle as in the original experiment.

$\mathsf{OLink}(i, \mathsf{lm}, \mathbf{\Sigma})$**:** Parse $\mathbf{\Sigma}$ as $\{(m_i, \mathsf{scp}_i, R_i, \sigma_i, \mathsf{nym}_i)\}_{i \in [k]}$. If $i = i_b^*$ then $\mathcal{B}$ first checks if $\exists (i_0', i_1') \in [k]$ such that $\mathsf{scp}_{i_0'} = \mathsf{scp}_{i_1'}$ and $i_0' \neq i_1'$ and if so returns $\perp$. Then for $i' \in [k]$ $\mathcal{B}$ checks that the output of its oracle $\mathsf{OChgPK}(b, \mathsf{scp}_{i'})$ is the same as $\mathsf{nym}_{i'}$, returning $\perp$ otherwise. $\mathcal{B}$ then computes $\tilde{\mathsf{nym}} \leftarrow \mathsf{Accum}(\mathsf{nym}_1, \ldots, \mathsf{nym}_k)$. Finally, $\mathcal{B}$ queries its oracle $\mathsf{OSign}(b, (\mathsf{lm}, \mathbf{\Sigma}), \tilde{\mathsf{nym}}, \{scp_1, \cdots, \mathsf{scp}_k\})$ to obtain π_l. If $i \notin \{i_0^*, i_1^*\}$, then $\mathcal{B}$ computes $\pi_l \leftarrow \mathsf{Link}(\mathsf{sk}_i, \mathsf{lm}, \mathbf{\Sigma})$ to obtain π_l. $\mathcal{B}$ adds $(\mathsf{lm}, \mathbf{\Sigma})$ to LNK for user index i: $\mathsf{LNK}[i] \leftarrow \mathsf{LNK}[i] \cup \{(\mathsf{lm}, \mathbf{\Sigma})\}$. It sends π_l to $\mathcal{A}_1$.

$\mathsf{Corr}(i)$**:** If $i \notin \{i_0^*, i_1^*\}$ then return sk_i. Otherwise $\mathcal{B}$ aborts.

After a polynomial number of queries, $\mathcal{A}_1$ outputs i_0, i_1, state. If $i_0 \neq i_0^*$ or $i_1 \neq i_1^*$ then $\mathcal{B}$ aborts.

Algorithm $\mathcal{B}$ now calls upon $\mathcal{A}_2(\mathsf{guess}, \mathsf{state})$, and answers $\mathcal{A}_2$'s oracle queries as follows.

$\mathsf{OSign}^{\mathsf{ucl\text{-}r}}, \mathsf{OLink}, \mathsf{Corr}$**:** $\mathcal{B}$ behaves as it did with $\mathcal{A}_1$.

$\mathsf{Ch} - \mathsf{Sign}_b(R, m, \mathsf{scp})$**:** $\mathcal{B}$ queries their oracle $\mathsf{OChgPK}(2, \mathsf{scp})$ to obtain nym. They then simulate σ making use of the zero knowledge property of the NIZK. They update the lists as normal and return (nym, σ).

$\mathsf{Ch} - \mathsf{Link}_b(\mathsf{lm}, \mathbf{\Sigma})$**:** Parse $\mathbf{\Sigma}$ as $\{(m_i, \mathsf{scp}_i, R_i, \sigma_i, \mathsf{nym}_i)\}_{i \in [k]}$. $\mathcal{B}$ first checks if $\exists (i'_0, i'_1) \in [k]$ such that $\mathsf{scp}_{i'_0} = \mathsf{scp}_{i'_1}$ and $i'_0 \neq i'_1$ and if so returns $\perp$. Then for $i' \in [k]$ $\mathcal{B}$ checks that the output of its oracle $\mathsf{OChgPK}(2, \mathsf{scp}_{i'})$ is the same as $\mathsf{nym}_{i'}$, returning $\perp$ otherwise. $\mathcal{B}$ then computes $\tilde{\mathsf{nym}} \leftarrow \mathsf{Accum}(\mathsf{nym}_1, \ldots, \mathsf{nym}_k)$. Finally, $\mathcal{B}$ queries its oracle $\mathsf{OSign}(2, (\mathsf{lm}, \mathbf{\Sigma}), \tilde{\mathsf{nym}}, \{scp_1, \cdots, \mathsf{scp}_k\})$ to obtain π_l. $\mathcal{B}$ adds $(\mathsf{lm}, \mathbf{\Sigma})$ to CLNK: $\mathsf{CLNK} \leftarrow \mathsf{CLNK} \cup \{(\mathsf{lm}, \mathbf{\Sigma})\}$. It sends π_l to $\mathcal{A}_1$.

After a polynomial number of queries, $\mathcal{A}_2$ outputs a bit d. Let us denote $\tilde{\beta}$ the random bit chosen by $\mathcal{B}$'s challenger. We then check the lists CLNK, $\mathsf{LNK}[i_0^*]$ and $\mathsf{LNK}[i_1^*]$, for any signatures that were not output by one of the signing oracles. For $(m, \mathsf{scp}, R, \sigma, \mathsf{nym}) \in \mathbf{\Sigma}$, $b' \in \{0, 1\}$ such that $\mathbf{\Sigma} \in \mathsf{LNK}[i_{b'}^*] \cup \mathsf{CLNK}$, and $(m, \mathsf{scp}, R, \sigma, \mathsf{nym}) \notin \mathsf{CSIG} \cup \mathsf{SIG}[i_{b'}^*]$, we make use of the simulation sound extractability of the NIZK to extract from σ a secret key sk such that $\mathsf{nym} = \mathsf{AKRS.ChgPK}(\mathsf{sk}, \mathsf{scp})$. If sk was not the secret key corresponding to $\mathsf{pk}_{b'}$ if $\mathbf{\Sigma} \in \mathsf{LNK}[i_{b'}^*]$ or $\mathsf{pk}_{\tilde{\beta}}$ if $\mathbf{\Sigma} \in \mathsf{CLNK}$, then the signing oracle would have failed, due to the accumulation soundness of AKRS. Otherwise clearly $\mathcal{B}$ can win in the class hiding game, and so aborts.

Otherwise, $\mathcal{B}$ forwards d to its own challenger, as the answer to its' own challenge.

Analysis: We now consider the probability that $\mathcal{B}$ aborts early returning 0. $\mathcal{A}$ chooses $(i_0, i_1) \neq (i_0^*, i_1^*)$ with probability $1 - \frac{2}{n(n-1)}$. Assuming that $\mathcal{A}$ is successful, then $\mathcal{A}$ will not query i_0^* or i_1^* to the Corr oracle.

Assuming that $\mathcal{B}$ never aborts early, we now show that they win, provided that $\mathcal{A}$ is successful. It is easy to see that $\mathcal{B}$ perfectly simulates $\mathcal{A}$'s challenger for $b = \tilde{\beta}$, where b is the bit chosen by the challenger for $\mathcal{A}$. Therefore, if $\mathcal{A}$ guesses b correctly, then $\mathcal{B}$ guesses $\tilde{\beta}$ correctly. To win in the class hiding experiment we also need that scopes queried to the OChgPK and OSign oracles for $a \in \{0, 1\}$ were not also queried to those oracles for $a = 2$. For this we need that scopes that were queried by $\mathcal{A}$ to the oracles $\mathsf{OSign}^{\mathsf{ucl\text{-}r}}$ or OLink for i_0^*, i_1^* were not queried to $\mathsf{Ch} - \mathsf{Sign}_b$ or $\mathsf{Ch} - \mathsf{Link}_b$. Due to line 9 of the Anonymity requirement, clearly the same scopes were not queried to both $\mathsf{OSign}^{\mathsf{ucl\text{-}r}}$ for i_0^*, i_1^* and $\mathsf{Ch} - \mathsf{Sign}_b$. As $\mathcal{B}$ has not aborted, all signatures in CLNK must originate from $\mathsf{CSIG} \cup \mathsf{SIG}[i_{b'}^*]$ for $b' \in \{0, 1\}$. Therefore, all signatures input to CLNK with scopes that have been input to $\mathsf{OSign}^{\mathsf{ucl\text{-}r}}$, must have originally come from $\mathsf{OSign}^{\mathsf{ucl\text{-}r}}$. However, due to line 13 in the anonymity experiment, signatures output by $\mathsf{OSign}^{\mathsf{ucl\text{-}r}}$ cannot be input to CLNK. Therefore, no scopes were input to both $\mathsf{OSign}^{\mathsf{ucl\text{-}r}}$ and CLNK. As $\mathcal{B}$ has not aborted, all signatures in $\mathsf{LNK}[i_0^*]$ or $\mathsf{LNK}[i_1^*]$ must originate from $\mathsf{CSIG} \cup \mathsf{SIG}[i_{b'}^*]$ for $b' \in \{0, 1\}$. Therefore, all signatures input to OLink with scopes that have been input to $\mathsf{Ch} - \mathsf{Sign}_b$, must have originally come from $\mathsf{Ch} - \mathsf{Sign}_b$. However, due to line 13 in the anonymity experiment, signatures output by $\mathsf{Ch} - \mathsf{Sign}_b$ cannot be input to OLink. Therefore, no scopes were input to both $\mathsf{Ch} - \mathsf{Sign}_b$ and OLink. Finally, we consider scopes input to both OLink for i_0^*, i_1^* and $\mathsf{Ch} - \mathsf{Link}_b$. As discussed, all scopes input to $\mathsf{Ch} - \mathsf{Link}_b$ must originate from $\mathsf{Ch} - \mathsf{Sign}_b$ and all scopes input to OLink must originate from $\mathsf{OSign}^{\mathsf{ucl\text{-}r}}$. Therefore, again due to line 9, no scope will be input to both oracles.

We therefore have that provided $\mathcal{A}$ is successful, the probability of $\mathcal{B}$ aborting early without winning is negligible, and otherwise they will be successful. Therefore, our construction satisfies anonymity.

Signature unforgeability: We show that if the AKRS is class hiding and the NIZK is simulation sound extractable, then UCL satisfies signature unforgeability.

Precisely, assuming there exists an algorithm $\mathcal{A}$ which breaks the signature unforgeability of UCL with probability ϵ, we show how to construct an algorithm $\mathcal{B}$, breaking the class hiding requirement of AKRS.

Algorithm $\mathcal{B}$ gets as input AKRS public keys $\mathsf{pk}_0, \mathsf{pk}_1$. It randomly samples i^* from $[n]$ and sets $\mathsf{vk}_{i^*} \leftarrow \mathsf{pk}_0$. For $i \in [n] \backslash \{i^*\}$, it samples $(\mathsf{sk}_i, \mathsf{vk}_i) \leftarrow \mathsf{AKRS.KGen}(1^\lambda)$. Finally, $\mathcal{B}$ forwards $\mathsf{vk}_1, \ldots, \mathsf{vk}_n$ to $\mathcal{A}$, and answers the latter's oracle queries as follows.

$\mathsf{OSign}^{\mathbf{ucl\text{-}r}}(i, R, m, \mathsf{scp})$**:** If $i = i^*$ then $\mathcal{B}$ queries their oracle $\mathsf{OChgPK}(0, \mathsf{scp})$ to obtain nym. They then simulate σ making use of the zero knowledge property of the NIZK. They update the lists as normal and return (nym, σ). Otherwise, $\mathcal{B}$ performs the oracle as in the original experiment.

$\mathsf{OLink}(i, \mathsf{lm}, \mathbf{\Sigma})$**:** Parse $\mathbf{\Sigma}$ as $\{(m_i, \mathsf{scp}_i, R_i, \sigma_i, \mathsf{nym}_i)\}_{i \in [k]}$. If $i = i^*$ then $\mathcal{B}$ first checks if $\exists (i'_0, i'_1) \in [k]$ such that $\mathsf{scp}_{i'_0} = \mathsf{scp}_{i'_1}$ and $i'_0 \neq i'_1$ and if so returns $\perp$. Then for $i' \in [k]$ $\mathcal{B}$ checks that the output of its oracle $\mathsf{OChgPK}(0, \mathsf{scp}_{i'})$ is the same as $\mathsf{nym}_{i'}$, returning $\perp$ otherwise. $\mathcal{B}$ then computes $\tilde{\mathsf{nym}} \leftarrow \mathsf{Accum}(\mathsf{nym}_1, \ldots, \mathsf{nym}_k)$. Finally, $\mathcal{B}$ queries its oracle $\mathsf{OSign}(0, (\mathsf{lm}, \mathbf{\Sigma}), \tilde{\mathsf{nym}}, \{scp_1, \cdots, \mathsf{scp}_k\})$ to obtain π_l. If $i \neq i^*$, then $\mathcal{B}$ computes $\pi_l \leftarrow \mathsf{Link}(\mathsf{sk}_i, \mathsf{lm}, \mathbf{\Sigma})$ to obtain π_l. $\mathcal{B}$ adds $(\mathsf{lm}, \mathbf{\Sigma})$ to LNK for user index i: $\mathsf{LNK}[i] \leftarrow \mathsf{LNK}[i] \cup \{(\mathsf{lm}, \mathbf{\Sigma})\}$. It sends π_l to $\mathcal{A}_1$.

$\mathsf{Corr}(i)$**:** If $i \neq i^*$ then return sk_i. Otherwise $\mathcal{B}$ aborts.

Now with probability ϵ, $\mathcal{A}$ outputs $(\{\Sigma_1^*, \cdots, \Sigma_m^*\}, R^*)$. For $i \in [m]$ parse $\Sigma_i^* = (m_i^*, \mathsf{scp}_i^*, R^*, \sigma_i^*, \mathsf{nym}_i^*)$. For all $i \in [m]$, $\mathcal{B}$ extracts a secret key sk_i^* from the NIZK which is possible due to the simulation sound extractability. $\mathcal{B}$ checks if any of these secret keys correspond to the public key pk_0. If so, they can easily use this to win in the class hiding experiment. Otherwise they abort.

Analysis: Assume that $\mathcal{A}$ is successful. Firstly, consider that $\mathcal{A}$ has used the corruption oracle for all $i \in R^*$. In that case to win $\mathcal{A}$ must output $n+1$ signatures. As all pseudonyms are different and all scopes must be the same this means that all secret keys extracted must be different. Each secret key must also correspond to a public key in R^*. This is clearly a contradiction.

Therefore, the adversary must not output at least one user in R^* that has not been corrupted. Let the number of honest users in R^* be γ. Assume with probability at most γ/n that one of these users was i^*. Then $\mathcal{A}$ must output at least $n - \gamma + 1$ signatures. Again, as all pseudonyms are different and all scopes must be the same this means that all $n - \gamma + 1$ secret keys extracted must be different. Each secret key must also correspond to a public key in R^*. Therefore, a secret key corresponding to one of the uncorrupted users must be extracted. Assume with probability $1/\gamma$ that this user was i^*. Then $\mathcal{B}$ does not abort and successfully wins in the class hiding experiment with probability $1/n$.

Link unforgeability We show that if the AKRS is unforgeable, then UCL satisfies link unforgeability.

Precisely, assuming there exists an algorithm $\mathcal{A}$ which breaks the link unforgeability of UCL with probability ϵ_{UCL}, we show how to construct an algorithm $\mathcal{B}$, breaking the unforgeability of AKRS.

Algorithm $\mathcal{B}$ gets as input an AKRS public key pk from its challenger for AKRS. $\mathcal{B}$ randomly chooses $i^* \leftarrow\!\!\$\ [k]$ and sets $\mathsf{vk}_{i^*} \leftarrow \mathsf{pk}$.. For $i \in [n]$, it sets $(\mathsf{sk}_i, \mathsf{vk}_i) \leftarrow \mathsf{AKRS.KGen}(\lambda)$. Finally, $\mathcal{B}$ forwards $\mathsf{vk}_1, \ldots, \mathsf{vk}_n$ to $\mathcal{A}$, and answers the latter's oracle queries as follows.

$\mathsf{OSign}^{\mathbf{ucl\text{-}r}}(i, R, m, \mathsf{scp})$**:** If $i \neq i^*$ proceed as normal. Otherwise, $\mathcal{B}$ calls upon oracle ChgPK for AKRS, with input *scp*, from which it gets output nym. They then simulate σ making use of the zero knowledge property of the NIZK. $\mathcal{B}$ then sends (nym, σ) to $\mathcal{A}$.

$\mathsf{OLink}(i, \mathsf{lm}, \mathbf{\Sigma})$**:** If $i \neq i^*$ proceed as normal. Otherwise, letting $\mathbf{\Sigma} = \{(m_i, \mathsf{scp}_i, R_i, \sigma_i, \mathsf{nym}_i)\}_{i\in[m]}$, then $\mathcal{B}$ first checks if $\exists(i'_0, i'_1) \in [m]$ such that $\mathsf{scp}_{i'_0} = \mathsf{scp}_{i'_1}$ and $i'_0 \neq i'_1$ and if so returns $\perp$. Then for $i' \in [k]$ $\mathcal{B}$ checks that the output of its oracle $\mathsf{OChgPK}(\mathsf{scp}_{i'})$ is the same as $\mathsf{nym}_{i'}$, returning $\perp$ otherwise. $\mathcal{B}$ then computes $\mathsf{n\tilde{y}m} \leftarrow \mathsf{Accum}(\mathsf{nym}_1, \ldots, \mathsf{nym}_k)$. $\mathcal{B}$ calls upon oracle OSign for AKRS, with input $lm||\mathbf{\Sigma}, \mathsf{n\tilde{y}m}, \{\mathsf{scp}_i\}_{i\in[m]}$, from which it gets output π_l, and adds $(\mathsf{lm}, \mathbf{\Sigma})$ to LNK for user index i: $\mathsf{LNK}[i] \leftarrow \mathsf{LNK}[i] \cup \{(\mathsf{lm}, \mathbf{\Sigma})\}$. It sends π_l to $\mathcal{A}$.

$\mathsf{Corr}(i)$**:** If $i \neq i^*$ proceed as normal. Otherwise, $\mathcal{B}$ aborts.

Now with probability ϵ_{UCL}, $\mathcal{A}$ outputs a forged link proof $(\mathsf{lm}, \mathbf{\Sigma}, \pi_l)$, let $\mathbf{\Sigma} = \{(m_i, \mathsf{scp}_i, R_i, \sigma_i, \mathsf{nym}_i)\}_{i\in[m]}$. $\mathcal{B}$ computes $\mathsf{n\tilde{y}m} \leftarrow \mathsf{Accum}(\mathsf{nym}_1, \ldots, \mathsf{nym}_m)$. $\mathcal{B}$ forwards $(\mathsf{n\tilde{y}m}, \{\mathsf{scp}_1, \cdots, \mathsf{scp}_m\}, \mathsf{lm}||\mathbf{\Sigma}, \pi_l)$ as its forgery for AKRS.

Analysis: For $\mathcal{A}$'s output to be a forgery, the following must hold.

1. $\mathsf{lm}, \mathbf{\Sigma}$ were not queried to OLink.
2. $\mathsf{VerifyLink}(\mathsf{lm}, \mathbf{\Sigma}, \pi_l) = 1$, and hence $\mathsf{AKRS.Vf}(\mathsf{n\tilde{y}m}, \mathsf{lm}||\mathbf{\Sigma}, \pi_l) = 1$.
3. $\exists\tilde{i} \in [n]$ such that $\tilde{i} \notin I$ and for all $j \in [m]$ $\Sigma_j \in \mathsf{SIG}[\tilde{i}]$.

Assume that $i^* = \tilde{i}$, which occurs with probability $1/n$. In this case $\mathcal{B}$ will not abort because i^* was not queried to the Corr oracle. We also have that for all $j \in [m]$, nym_j is a correctly formed public key for the scope scp_j because this was honestly generated in the OSign oracle. This means that $n\tilde{y}m = \mathsf{Accum}(\mathsf{ChgPK}(\mathsf{sk}, \mathsf{scp}_1), \cdots, \mathsf{ChgPK}(\mathsf{sk}, \mathsf{scp}_m))$ where sk is the secret key corresponding to pk (unknown to $\mathcal{B}$). Now observe that, from item 1, it holds that the signature output is a valid AKRS signature. $\mathcal{B}$ has not queried any instance of oracle OSign for AKRS with input $(\mathsf{lm}||\mathbf{\Sigma}, \cdot)$. It is clear that if $\mathcal{A}$'s forgery for UCL is successful, then $\mathcal{B}$'s forgery for AKRS is also successful. Therefore, $\mathcal{B}$ wins with probability $\epsilon_{\mathsf{UCL}}/n$.

Signature non-frameability We show that if the AKRS is unforgeable and the NIZK is simulation sound extractable, then UCL satisfies signature non-frameability.

Precisely, assuming there exists an algorithm $\mathcal{A}$ which breaks the signature non-frameability of UCL with probability ϵ_{UCL}, we show how to construct an algorithm $\mathcal{B}$, breaking the unforgeability of AKRS.

Algorithm $\mathcal{B}$ gets as input an AKRS public key pk from its challenger for AKRS. It randomly samples i^* from $[n]$ and sets $\mathsf{vk}_{i^*} \leftarrow \mathsf{pk}$. For $i \in [n]\backslash\{i^*\}$, it samples $(\mathsf{sk}_i, \mathsf{vk}_i) \leftarrow \mathsf{AKRS.KGen}(1^\lambda)$. Finally, $\mathcal{B}$ forwards $\mathsf{vk}_1, \ldots, \mathsf{vk}_n$ to $\mathcal{A}$, and answers the latter's oracle queries as follows.

$\mathsf{OSign}^{\mathbf{ucl\text{-}r}}(i, R, m, \mathsf{scp})$**:** If $i = i^*$ then $\mathcal{B}$ queries their oracle $\mathsf{OChgPK}(\mathsf{scp})$ to obtain nym. They then simulate σ making use of the zero knowledge property of the NIZK. They update the lists as normal and return (nym, σ). Otherwise, $\mathcal{B}$ performs the oracle as in the original experiment.

$\mathsf{OLink}(i, \mathsf{lm}, \mathbf{\Sigma})$**:** Parse $\mathbf{\Sigma}$ as $\{(m_i, \mathsf{scp}_i, R_i, \sigma_i, \mathsf{nym}_i)\}_{i\in[k]}$. If $i = i^*$ then $\mathcal{B}$ first checks if $\exists(i'_0, i'_1) \in [k]$ such that $\mathsf{scp}_{i'_0} = \mathsf{scp}_{i'_1}$ and $i'_0 \neq i'_1$ and if so returns $\perp$. Then for $i' \in [k]$ $\mathcal{B}$ checks that the output of its oracle $\mathsf{OChgPK}(\mathsf{scp}_{i'})$ is the same as $\mathsf{nym}_{i'}$, returning $\perp$ otherwise. $\mathcal{B}$ then computes $\mathsf{n\tilde{y}m} \leftarrow \mathsf{Accum}(\mathsf{nym}_1, \ldots, \mathsf{nym}_k)$. Finally, $\mathcal{B}$ queries its oracle $\mathsf{OSign}((\mathsf{lm}, \mathbf{\Sigma}), \mathsf{n\tilde{y}m}, \{scp_1, \cdots, \mathsf{scp}_k\})$ to obtain π_l. If $i \neq i^*$, then $\mathcal{B}$ computes $\pi_l \leftarrow \mathsf{Link}(\mathsf{sk}_i, \mathsf{lm}, \mathbf{\Sigma})$ to obtain π_l. $\mathcal{B}$ adds $(\mathsf{lm}, \mathbf{\Sigma})$ to LNK for user index i: $\mathsf{LNK}[i] \leftarrow \mathsf{LNK}[i] \cup \{(\mathsf{lm}, \mathbf{\Sigma})\}$. It sends π_l to $\mathcal{A}_1$.

$\mathsf{Corr}(i)$**:** If $i \neq i^*$ proceed as normal. Otherwise, $\mathcal{B}$ aborts.

Now with probability ϵ_{UCL}, $\mathcal{A}$ outputs a forged signature $\Sigma = (m, \mathsf{scp}, R, \sigma, \mathsf{nym})$. $\mathcal{B}$ extracts a secret key sk^* from the NIZK, σ, which is possible due to the simulation sound extractability. $\mathcal{B}$ chooses any m^* that hasn't ever been queried to OSign and computes $\sigma^* \leftarrow \mathsf{AKRS.Sig}(\mathsf{sk}^*, \mathsf{nym}, m^*)$. $\mathcal{B}$ forwards $(\mathsf{nym}, \mathsf{scp}, m^*, \sigma^*)$ as its forgery for AKRS.

Analysis: For $\mathcal{A}$'s output to be a forgery, there must exists $\tilde{i} \in [k]$ such that the following holds.

1. $(m, \mathsf{scp}, R, \mathsf{nym}, \cdot) \notin \mathsf{SIG}[\tilde{i}]$.
2. $\mathsf{Vf}(\Sigma) = 1$, and hence $\mathsf{nym} = \mathsf{AKRS.ChgPK}(\mathsf{sk}^*, \mathsf{scp})$ (if extraction successful).
3. $\tilde{i} \notin I$.
4. $(\cdot, \mathsf{scp}, \cdot, \cdot, \mathsf{nym}) \in \mathsf{SIG}[\tilde{i}]$.

Assume that $i^* = \tilde{i}$, which occurs with probability $1/n$. In this case $\mathcal{B}$ will not abort because i^* was not queried to the Corr oracle. Now observe that, from items 1 and item 2, it holds that the signature output is a valid AKRS signature that has not originated from the signing oracle for i^*, therefore extraction will be successful. We therefore have that $\mathsf{nym} = \mathsf{ChgPK}(\mathsf{sk}^*, \mathsf{scp})$ and sk^* corresponds to a public key in R. Therefore, clearly σ^* is a valid signature due to correctness. As $(\cdot, \mathsf{scp}, \cdot, \cdot, \mathsf{nym}) \in \mathsf{SIG}[\tilde{i}]$ then nym was output previously on input $\mathsf{OChgPK}(\mathsf{scp})$. Therefore, letting sk be the key corresponding to pk, $\mathsf{nym} = \mathsf{ChgPK}(\mathsf{sk}, \mathsf{scp})$. $\mathcal{B}$ therefore wins the unforgeability game with probability $\epsilon_{\mathsf{UCL}}/n$

Link non-frameability We show that if the AKRS is accumulation sound, then UCL satisfies link non-frameability.

Precisely, assuming there exists an algorithm $\mathcal{A}$ which breaks the link non-frameability of UCL with probability ϵ_{UCL}, we show how to construct an algorithm $\mathcal{B}$, breaking the accumulation soundness of AKRS.

Algorithm $\mathcal{B}$ gets as input AKRS key pairs $((\mathsf{sk}_1, \mathsf{vk}_1), \cdots, (\mathsf{sk}_n, \mathsf{vk}_n))$ from its challenger for AKRS. Finally, $\mathcal{B}$ forwards $\mathsf{vk}_1, \ldots, \mathsf{vk}_n$ to $\mathcal{A}$, and answers the latter's oracle queries as normal, which is possible because $\mathcal{B}$ knows all the secret keys.

Now with probability ϵ_{UCL}, $\mathcal{A}$ outputs a forged link proof $(\mathsf{lm}, \boldsymbol{\Sigma}, \pi_l)$, where $\boldsymbol{\Sigma} = \{(m_i, \mathsf{scp}_i, R_i, \sigma_i, \mathsf{nym}_i)\}_{i \in [m]}$. $\mathcal{B}$ computes $\tilde{\mathsf{nym}} \leftarrow \mathsf{Accum}(\mathsf{nym}_1, \ldots, \mathsf{nym}_m)$.

Let $A, B = \emptyset$. For all $j \in [m]$, if $\exists i \in [n] : \Sigma_j \in \mathsf{SIG}[i]$ or $\mathsf{Identify}(\mathsf{sk}_i, \Sigma_j) = 1$ let $i_j = i$ and $\mathcal{I} \leftarrow \mathcal{I} \cup [i]$, $A \leftarrow A \cup \mathsf{scp}_j$. Otherwise $i_j = n+1$, and we can extract sk'_j from σ_j due to the extractability of the NIZK. We can then compute $\Psi \leftarrow \mathsf{AKRS}(\mathsf{sk}'_j, \mathsf{nym}_j, 0)$ and $B \leftarrow B \cup (\mathsf{scp}_j, \mathsf{nym}_j, 0, \Psi_j,)$.

$\mathcal{B}$ forwards $(B, \mathcal{I}, A, \mathsf{lm} || \boldsymbol{\Sigma}, \pi_l)$ as its forgery for AKRS.

Analysis: For $\mathcal{A}$'s output to be a forged link proof, it must satisfy $\mathsf{VerifyLink}(\mathsf{lm}, \boldsymbol{\Sigma}, \pi_l) = 1$; and the signatures in $\boldsymbol{\Sigma}$ were *not all* signed for (or identified as signed for) the same secret key sk, i.e $\exists (j_1, j_2) \in [m]$ such that $i_{j_1} \neq i_{j_2}$.

For $\mathcal{B}$ to be successful in the accumulation soundness game, we need 4 conditions to be satisfied.

Firstly, we need that $\mathcal{I} \neq \emptyset$. If $\mathcal{I} = \emptyset$, for all $j \in [m]$, $i_j = n+1$, in which case $\mathcal{A}$ would not be successful in the link non-frameability game.

We also need that signatures in the set B are valid. For $\mathsf{VerifyLink}(\tilde{\mathsf{nym}}, \mathsf{lm} || \boldsymbol{\Sigma}, \pi_l) = 1$, we must have that each signature in $\boldsymbol{\Sigma}$ is a valid ring signature. Due to the extractability, we have that, for all $j \in [m]$ such that $i_j = n+1$, $\mathsf{nym}_j = \mathsf{AKRS.ChgPK}(\mathsf{sk}'_j, \mathsf{scp}_j)$ and that sk'_j corresponds to

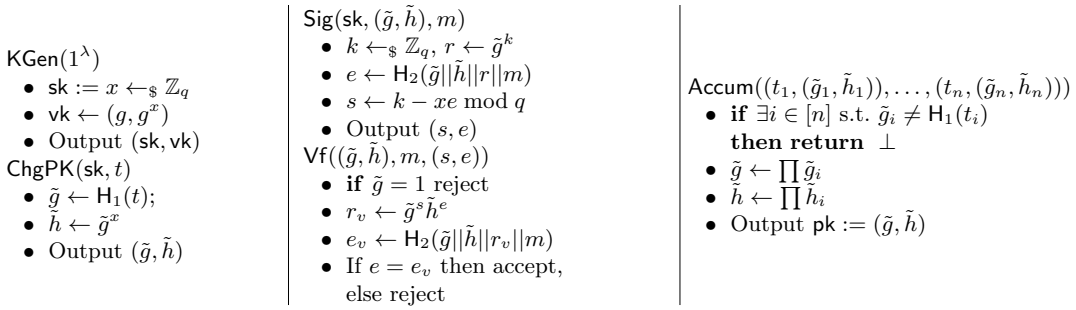

Fig. 5. Schnorr AKRS

a public key in the ring R_j. Therefore, due to the correctness of the AKRS each $\Psi_j \in B$ is a valid signature.

We additionally need that π_l is a valid AKRS signature on message $\mathsf{lm}||\boldsymbol{\Sigma}$ and public key $\tilde{\mathsf{nym}}$. This is again satisfied because $\mathsf{VerifyLink}(\tilde{\mathsf{nym}}, \mathsf{lm}||\boldsymbol{\Sigma}, \pi_l) = 1$.

Finally we need that either $\exists (i, j) \in \mathcal{I}$ s.t. $\mathsf{sk}_i \neq \mathsf{sk}_j$ or $\exists (\mathsf{scp}^*, \mathsf{nym}^*, 0, \Psi^*) \in B$ such that $\forall j \in \mathcal{I}, \mathsf{ChgPK}(\mathsf{sk}_j, scp^*) \neq \mathsf{nym}^*$. Say this does not hold, then for all $(j_1, j_2) \in \mathcal{I}$, $i_{j_1} = i_{j_2}$ and B is the emptyset. Therefore, $\mathcal{A}$ would not be successful in the link non–frameability game.

We therefore have the output of $\mathcal{B}$ is a successful forgery in the accumulation soundness game.

6 AKRS instantiations

6.1 Construction from Schnorr signatures

Consider a group $\mathbb{G}$, of prime order q, generated by g; and hash functions $\mathsf{H}_1 : \{0,1\}^* \to \mathbb{G}$, and $\mathsf{H}_2 : \{0,1\}^* \to \mathbb{Z}_q$. In figure 5, we recall the standard algorithms of the standard Schnorr signature scheme. We also introduce new algorithms ChgPK and Accum which augment the scheme to an AKRS.

Security argument. We now provide some intuition as to why the augmented Schnorr signature given in Figure 5 satisfies our requirements for an AKRS. Unforgeability is, modulo minor technical details, inherent from the unforgeability property of Schnorr Signatures. Class hiding follows from the fact that two public keys in the same equivalence class are of the form $(\mathsf{H}_1(t), \mathsf{H}_1(t)^x)$ and $(\mathsf{H}_1(t'), \mathsf{H}_1(t')^x)$, which is a DDH tuple. Therefore, linking public keys by equivalence class can be reduced to distinguishing DDH tuples. In the proof, signatures can be simulated without the secret key, assuming that H_2 is a random oracle. Accumulation soundness is the property that involves more novel techniques, in the design of our construction and its security analysis. The accumulation algorithm can be seen as a way to batch ℓ Schnorr statements with the same witness into a *succinct proof* that, to the best of our knowledge, is new. Roughly, accumulation soundness follows from the fact that signing with respect to an accumulated public key $(\prod \tilde{g}_i, \prod \tilde{h}_i)$ requires knowledge of the discrete logarithm of $\prod \tilde{h}_i$ base $\prod \tilde{g}_i$. Letting $\prod \tilde{g}_i = \prod \mathsf{H}_1(t_i) = g^{\sum \tilde{t}_i}$, where $\mathsf{H}_1(t_i) = g^{\tilde{t}_i}$ the adversary must know $\frac{\sum \mathsf{sk}_i \tilde{t}_i}{\sum \tilde{t}_i}$, which entails knowledge of the $\tilde{t}_i$, i.e. breaking the discrete logarithm. We prove this formally below.

Correctness w.r.t. standard public keys: Consider a security parameter $\lambda \in \mathbb{N}$, and $(x, (g, g^x)) \leftarrow \mathsf{KGen}(1^\lambda)$. Let $h = g^x$. Then for any $(s, e) \leftarrow \mathsf{Sig}(x, (g, h), m)$, there exists $k \in \mathbb{Z}_q$ such that $s \equiv k - xe \bmod q$ and $e = \mathsf{H}_2(g||g^x||g^k||m)$. Hence, $g^s h^e = g^{(s+xe)} = g^k$, and so $e = \mathsf{H}_2(g||h||g^s h^e||m)$. This allows us to conclude that $\mathsf{Vf}((g, h), m, (s, e)) = 1$.

Correctness w.r.t. randomised public keys: Consider a security parameter $\lambda \in \mathbb{N}$, and $(x, g^x) \leftarrow \mathsf{KGen}(1^\lambda)$. For any $t \in \{0,1\}^*$, let $(\tilde{g} = \mathsf{H}_1(t), \tilde{h} = \mathsf{H}_1(t)^x) \leftarrow \mathsf{ChgPK}(x, t)$, then for any $(s, e) \leftarrow \mathsf{Sig}(x, (\tilde{g}, \tilde{h}), m)$, there exists $k \in \mathbb{Z}_q$ such that $s \equiv k - xe \bmod q$ and $e = \mathsf{H}_2(\tilde{g}||\tilde{h}||\tilde{g}^k||m)$. Hence $\tilde{g}^s\tilde{h}^e = \tilde{g}^{(s+xe)} = \tilde{g}^k$, and so $e = \mathsf{H}_2(\tilde{g}||\tilde{h}||\tilde{g}^s\tilde{h}^e||m)$. This allows us to conclude that $\mathsf{Vf}((\tilde{g}, \tilde{h}), m, (s, e)) = 1$.

Correctness w.r.t. accumulated public keys: Consider a security parameter $\lambda \in \mathbb{N}$, and $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{KGen}(1^\lambda)$. For $i \in [n]$, consider any $t_i \in \{0,1\}^*$ and let $(\tilde{g}_i, \tilde{h}_i) \leftarrow \mathsf{ChgPK}(\mathsf{sk}, t_i)$. Then $\mathsf{Accum}\left((\tilde{g}_1, \tilde{h}_1), \ldots, (\tilde{g}_n, \tilde{h}_n)\right) \rightarrow (\tilde{g}, \tilde{h})$, where $\tilde{g} = \prod \tilde{g}_i$ and $\tilde{h} = \prod \tilde{h}_i = \prod \tilde{g}_i^{\mathsf{sk}} = \tilde{g}^{\mathsf{sk}}$. Therefore, for any $(s, e) \leftarrow \mathsf{Sig}(\mathsf{sk}, (\tilde{g}, \tilde{h}), m)$, we also have that $\mathsf{Vf}((\tilde{g}, \tilde{h}), m, (s, e)) = 1$.

Class Hiding: Let $\mathcal{A}$ be an adversary in the class hiding experiment for the Schnorr AKRS, such that $\mathsf{H}_1 : \{0,1\}^* \rightarrow \mathbb{G}, \mathsf{H}_2 : \{0,1\}^* \rightarrow \mathbb{Z}_q^*$ are modelled as random oracles, that makes at most r queries of t to either H_1, OChgPK, or OSign. More specifically the adversary $\mathcal{A}$ is provided with the H_1 and H_2 oracles which keep track of lists RO_1 and RO_2 respectively. If $\mathcal{A}$ queries $t \in \mathbb{Z}_q$ to H_1 then the oracle checks if there exist $t, \tilde{g}$ such that $(t, \tilde{g}) \in \mathsf{RO}_1$. If so, it returns $\tilde{g}$. Else, it samples $\tilde{g} \leftarrow_\$ \mathbb{G}$ uniformly at random, adds $(t, \tilde{g})$ to RO_1, and outputs $\tilde{g}$. If $\mathcal{A}$ queries $\alpha \in \{0,1\}^*$ to H_2, then the oracle checks if there exist β such that $(\alpha, \beta) \in \mathsf{RO}_2$. If so, it returns β. Else, it samples $\beta \leftarrow_\$ \mathbb{Z}_q$ uniformly at random, adds (α, β) to RO_2, and returns β to $\mathcal{A}$.

We proceed through a series of game hops that we show are indistinguishable to the adversary. In the final game, the view of the adversary will be identical for $b^* = 0$ and $b^* = 1$ and the adversary will have a negligible chance of guessing b^*. We define Game 0 as the experiment $\mathsf{Exp}^{\mathsf{class\text{-}hiding}}_{\mathsf{AKRS},\mathcal{A}}(1^\lambda)$. Let S_i be the event that adversary $\mathcal{A}$ wins in the class hiding experiment. We model H_1, H_2 as random oracles.

Game 1 is identical to Game 0, except that in the OSign oracle, instead of generating the signatures output via $\sigma \leftarrow \mathsf{Sig}(\mathsf{sk}_a, \mathsf{pk}', m)$, we instead sample $e, s \leftarrow_\$ \mathbb{Z}_q$ uniformly at random. Letting $pk' = (\tilde{g}, \tilde{h})$, if $(\tilde{g}||\tilde{h}||\tilde{g}^s\tilde{h}^e||m, \cdot) \in \mathsf{RO}_2$ then Game 1 aborts (as e, s were chosen randomly this occurs with negligible probability), otherwise $((\tilde{g}||\tilde{h}||\tilde{g}^s\tilde{h}^e||m, e)$ is added to RO_2. The OSign oracle outputs (s, e). As (s, e) satisfies $\mathsf{H}_2((\tilde{g}||\tilde{h}||\tilde{g}^s\tilde{h}^e||m) = e$, this does not affect the distribution of inputs to the adversary. Therefore, $Pr[S_0] = Pr[S_1]$.

Game 2, is identical to Game 1, except for a change to the OChgPK and OSign oracles when $a = 2$ is input. Instead of using sk_{b^*} to generate pk' via ChgPK in both oracles, we will instead obtain $\tilde{g} = \mathsf{H}_1(t)$ as usual, but randomly sample $\tilde{h} \leftarrow_\$ \mathbb{G}$. We give the OChgPK and OSign oracles used in Game 2 in Figure 6.

This game hop will make use of a hybrid argument. We split the reduction into r steps. We define Game $1.i$ to behave as in Game 2 only for the first i distinct queries submitted to the H_1 oracle (including via the OChgPK and OSign oracles) and otherwise behave as in Game 1. Clearly, Game $1.r$ will be identical to Game 2 and Game 1.0 will be identical to Game 1.

We show that Game $1.i$ and Game $1.(i+1)$ are indistinguishable assuming the DDH problem is hard. We give a distinguishing algorithm $\mathcal{D}_1$ in Figure 7. $\mathcal{D}_1$ is input (y_1, y_2, y_3, y_4) and aims to distinguish if this is a DDH tuple or (y_1, y_2, y_3, y_4) were chosen randomly and independently.

We show that, when a DDH tuple is input, inputs to $\mathcal{A}$ are identical to Game $1.i$ and otherwise, inputs to $\mathcal{A}$ are identical to Game $1.(i+1)$. Note that sk_{1-b^*}, pk_{1-b^*} are generated identically to in both games $1.i$ and $1.(i+1)$. Furthermore, g, pk_{b^*} are identically distributed to both games $1.i$ and $1.(i+1)$. The H_1 oracle is distributed identically to both games $1.i$ and $1.(i+1)$, because $\tilde{t}$ is

$\mathsf{OChgPK}_{L_1,L_2}(a,t)$

.**if** $\mathsf{sk}_a = \perp$ **then return** $\perp$
if $a \in \{0,1\}$ **then** $L_1 \leftarrow L_1 \cup \{t\}$
if $a = 2$ **then** $L_2 \leftarrow L_2 \cup \{t\}$
if $a = 2$ **then** $\tilde{g} \leftarrow \mathsf{H}_1(t), \tilde{h} \leftarrow_\$ \mathbb{G}, \mathsf{pk}' \leftarrow (\tilde{g}, \tilde{h})$
else $\mathsf{pk}' \leftarrow \mathsf{ChgPK}(\mathsf{sk}_a, t)$
return pk'

$\mathsf{OSign}_{L_1,L_2}(a, m, \mathsf{pk}', \{t_1, \cdots, t_m\})$

if $\mathsf{sk}_a = \perp$ **then return** $\perp$
if $\{t_1, \cdots, t_m\} \neq *$ **then**
  if $a \in \{0,1\}$ **then** $L_1 \leftarrow L_1 \cup \{t_1, \cdots, t_m\}$
  if $a = 2$ **then** $L_2 \leftarrow L_2 \cup \{t_1, \cdots, t_m\}$
  if $\mathsf{pk}' \neq \mathsf{Accum}((t_1, \mathsf{OChgPK}(a, t_1)), \cdots, (t_m, \mathsf{OChgPK}(a, t_m)))$ **then return** $\perp$
  Let $\mathsf{pk}' = (\tilde{g}, \tilde{h}), e, s \leftarrow_\$ \mathbb{Z}_q$, add $((\tilde{g}||\tilde{h}||\tilde{g}^s\tilde{h}^e||m, e)$ to RO_2
if $t = *$ **then if** $a = 2$ **then return** $\perp$
  else Let $\mathsf{pk}_a = (g, h_a), e, s \leftarrow_\$ \mathbb{Z}_q$, add $((g||h_a||g^s h_a^e||m, e)$ to RO_2
return (s, e)

Fig. 6. OChgPK and OSign oracles in Game 2.

chosen randomly and so a random group element is output. The H_2 oracle is unchanged. For the first distinct i values of t both the OSign and OChgPK oracles are identical to both games $1.i$ and $1.(i+1)$, except for when $a = b^*$. In this case, $\tilde{h}$ is distributed identically because letting $y_2 = y_1^{x_1}$ then $y_2^{\tilde{t}} = \tilde{g}^{x_1}$. For the $i+1$st query, $\mathcal{A}$ can only query either $a \in \{0,1\}$ or $a = 2$. If they query $a \in \{0,1\}$, then $\mathcal{D}$ will abort with probability $1/2$ if $\tilde{b} = 0$. If they do not abort, then $\tilde{g} = y_1^{\tilde{t}}$ and so, when $a = b^*$, $\tilde{h} = y_2^{\tilde{t}}$ is correctly distributed. If they query $a = 2$, then $\mathcal{D}$ will abort with probability $1/2$ if $\tilde{b} = 1$. If they do not abort, then $\tilde{g} = y_3^{\tilde{t}}$ and so, $\tilde{h} = y_4^{\tilde{t}}$ is correctly distributed. For the final queries, both the OSign and OChgPK oracles are identical to both games $1.i$ and $1.(i+1)$, except for when $a \in \{b^*, 2\}$. In this case, $\tilde{h}$ is distributed identically because $y_2^{\tilde{t}} = \tilde{g}^{x_1}$. The signatures output by OSign are generated the same way as in both games.

Therefore $|Pr[S_{1.i}] - Pr[S_{1.(i+1)}]| \leq 2\epsilon_{\mathsf{DDH}}$, where ϵ_{DDH} is the advantage in distinguishing DDH tuples. By a standard argument, $|Pr[S_1] - Pr[S_2]| \leq 2r\epsilon_{\mathsf{DDH}}$.

In Game 2, all inputs to the adversary are now independent of b^* and so $Pr[S_2] = 1/2$.
We now have that

$$|Pr[S_0] - 1/2| \leq 2r\epsilon_{\mathsf{DDH}}$$

and conclude that the advantage of the adversary in Game 0 is negligible as required.

Unforgeability: Assume $\mathsf{H}_1 : \{0,1\}^* \rightarrow \mathbb{G}$ and $\mathsf{H}_2 : \{0,1\}^* \rightarrow \mathbb{Z}_q^*$ are modelled as random oracles, and there exists a PPT algorithm $\mathcal{A}$ that has non-negligible advantage in outputting valid forgeries for the Schnorr AKRS. We here show how to build a PPT algorithm $\mathcal{B}$, which, using $\mathcal{A}$ as a subroutine breaks the discrete logarithm assumption. Hence proving that in the random oracle model, and

$\mathsf{H}_{1\mathsf{RO}_1}(t)$

if $\exists(t, \tilde{t}, \tilde{g}, r') \in \mathsf{RO}_1$
  return $\tilde{g}$
else $\tilde{r} \leftarrow \tilde{r} + 1$
  if $\tilde{r} = i + 1$ and $\tilde{b} = 0$ **then** $\tilde{t} \leftarrow_\$ \mathbb{Z}_q, \tilde{g} := y_3^{\tilde{t}}$
  else $\tilde{t} \leftarrow_\$ \mathbb{Z}_q, \tilde{g} := g^{\tilde{t}}$
  add $(t, \tilde{t}, \tilde{g}, \tilde{r})$ to RO_1
  return $\tilde{g}$

$\mathsf{H}_{2\mathsf{RO}_2}(\alpha)$

if $\exists(\alpha, \beta) \in \mathsf{RO}_1$ **return** β
else $\beta \leftarrow_\$ \mathbb{Z}_q$
  add (α, β) to RO_2
  return β

$\mathsf{OChgPK}_{L_1,L_2}(a, t)$

.**if** $a \notin \{0, 1, 2, \}$ **then return** $\perp$
if $a \in \{0, 1\}$ **then** $L_1 \leftarrow L_1 \cup \{t\}$
if $a = 2$ **then** $L_2 \leftarrow L_2 \cup \{t\}$
if $(t, \cdot, \cdot, \cdot) \notin \mathsf{RO}_1$ **then** $\tilde{r} \leftarrow \tilde{r} + 1$
    if $\tilde{r} = i + 1$ and $\tilde{b} = 0$ $\tilde{t} \leftarrow_\$ \mathbb{Z}_q, \tilde{g} := y_3^{\tilde{t}}$
    else $\tilde{t} \leftarrow_\$ \mathbb{Z}_q, \tilde{g} := g^{\tilde{t}}$
    add $(t, \tilde{t}, \tilde{g}, \tilde{r})$ to RO_1
For $(t, \tilde{t}, \tilde{g}, r') \in \mathsf{RO}_1$
  if $r' > i + 1$
    if $a \in \{b^*, 2\}$ **then** $\tilde{h} := y_2^{\tilde{t}}, \mathsf{pk}' \leftarrow (\tilde{g}, \tilde{h})$
    else $\mathsf{pk}' \leftarrow \mathsf{ChgPK}(\mathsf{sk}_a, t)$
  if $r' = i + 1$
    if $a \in \{0, 1\}$ **then if** $\tilde{b} = 0$ **then** $\mathcal{D}_1$ aborts
      else if $a = b^*$ **then** $\tilde{h} := y_2^{\tilde{t}}, \mathsf{pk}' \leftarrow (\tilde{g}, \tilde{h})$
        else $\mathsf{pk}' \leftarrow \mathsf{ChgPK}(\mathsf{sk}_a, t)$
    if $a = 2$ **then if** $\tilde{b} = 1$ **then** $\mathcal{D}_1$ aborts
      else $\tilde{h} := y_4^{\tilde{t}}, \mathsf{pk}' \leftarrow (\tilde{g}, \tilde{h})$
  if $r' \leq i$
    if $a = 2$ **then** $\tilde{h} \leftarrow_\$ \mathbb{G}, \mathsf{pk}' \leftarrow (\tilde{g}, \tilde{h})$
    if $a = b^*$ **then** $\tilde{h} := y_2^{\tilde{t}}, \mathsf{pk}' \leftarrow (\tilde{g}, \tilde{h})$
    else $\mathsf{pk}' \leftarrow \mathsf{ChgPK}(\mathsf{sk}_a, t)$
return pk'

$\mathsf{OSign}_{L_1,L_2}(a, m, \mathsf{pk}', \{t_1, \cdots, t_m\})$

if $a \notin \{0, 1, 2\}$ **then return** $\perp$
if $t \neq *$ **then**
  if $\mathsf{pk}' \neq \mathsf{Accum}((t_1, \mathsf{OChgPK}(a, t_1)), \cdots, (t_m, \mathsf{OChgPK}(a, t_m)))$
    then return $\perp$
  Let $\mathsf{pk}' = (\tilde{g}, \tilde{h}), e, s \leftarrow_\$ \mathbb{Z}_q$
  add $(\tilde{g}||\tilde{h}||\tilde{g}^s\tilde{h}^e||m, e)$ to RO_2
if $t = *$ **then if** $a = 2$ **then return** $\perp$
  else Let $\mathsf{pk}_a = (g, h_a), e, s \leftarrow_\$ \mathbb{Z}_q$
  add $(g||h_a||g^s h_a^e||m, e)$ to RO_2
return (s, e)

$\mathcal{D}_1(y_1, y_2, y_3, y_4)$

$\tilde{r} \leftarrow 0, \tilde{b} \leftarrow_\$ \{0, 1\}, \mathsf{RO}_1, \mathsf{RO}_2 \leftarrow \emptyset$
$b^* \leftarrow_\$ \{0, 1\}, L_1 := \{\}, L_2 := \{\}$
$g \leftarrow y_1, pk_{b^*} \leftarrow (g, y_2), (\mathsf{sk}_{1-b^*}, \mathsf{pk}_{1-b^*}) \leftarrow \mathsf{KGen}(1^\lambda)$
$d \leftarrow \mathcal{A}^{\mathsf{OChgPK}_{L_1,L_2}, \mathsf{OSign}_{L_1,L_2}}(\mathsf{choose}, \mathsf{g}, \mathsf{pk}_0, \mathsf{pk}_1)$
return $((d = b^*) \wedge (L_1 \cap L_2 = \emptyset))$

Fig. 7. $\mathcal{D}_1$ that distinguishes between Game 1.i and Game 1.i+1.

under the assumption computing discrete logarithms is hard, the AKRS of Figure 5 is unforgeable. The reduction is described hereafter.

Algorithm $\mathcal{B}$ receives y from the discrete logarithm challenger, sets the verification key $\mathsf{vk} = y$, and initializes empty lists $\mathsf{RO}_1, \mathsf{RO}_2$. Let x be the discrete logarithm of $y = g^x$. This value is unknown to $\mathcal{B}$, and is to the secret key corresponding to vk. Then $\mathcal{B}$ sends vk to $\mathcal{A}$, and simulates the random oracles H_1, H_2 as well as oracles $\mathcal{O}^{\mathsf{ChgPK}}$ and $\mathcal{O}^{\mathsf{Sig}}$ as follows:

- If $\mathcal{A}$ queries $t \in \mathbb{Z}_q$ to H_1, then $\mathcal{B}$ checks if there exist $\tilde{t}, \tilde{g}, \tilde{h}$ and $\mathsf{type} \in \{\mathsf{ro}, \mathsf{key}\}$ such that $(t, \tilde{t}, \tilde{g}, \tilde{h}, \mathsf{type}) \in \mathsf{RO}_1$. If so, it returns $\tilde{g}$. Else, it samples $\tilde{t} \leftarrow_\$ \mathbb{Z}_q$ uniformly at random, sets $\tilde{g} := g^{\tilde{t}}$, $\tilde{h} := \mathsf{vk}^{\tilde{t}}$, adds $(t, \tilde{t}, \tilde{g}, \tilde{h}, \mathsf{ro})$ to RO_1, and returns $\tilde{g}$ to $\mathcal{A}$.
- If $\mathcal{A}$ queries $\alpha \in \{0,1\}^*$ to H_2, then $\mathcal{B}$ checks if there exist β such that $(\alpha, \beta) \in \mathsf{RO}_2$. If so, it returns β. Else, it samples $\beta \leftarrow_\$ \mathbb{Z}_q$ uniformly at random, adds (α, β) to RO_2, and returns β to $\mathcal{A}$.
- If $\mathcal{A}$ queries $t \in \mathbb{Z}_q$ to $\mathcal{O}^{\mathsf{ChgPK}}$, then $\mathcal{B}$ checks if there exist $\tilde{t}, \tilde{g}, \tilde{h}$ and $\mathsf{type} \in \{\mathsf{ro}, \mathsf{key}\}$ such that $(t, \tilde{t}, \tilde{g}, \tilde{h}, \mathsf{type}) \in \mathsf{RO}_1$. If so, it returns $(\tilde{g}, \tilde{h})$, and overwrites type to be key. Else, it samples $\tilde{t} \leftarrow_\$ \mathbb{Z}_q$ uniformly at random, sets $\tilde{g} := g^{\tilde{t}}$, $\tilde{h} := \mathsf{vk}^{\tilde{t}}$, adds $(t, \tilde{t}, \tilde{g}, \tilde{h}, \mathsf{key})$ to RO_1, and returns $(\tilde{g}, \tilde{h})$ to $\mathcal{A}$.
- If $\mathcal{A}$ queries a signature for m w.r.t. $\mathsf{pk}' = (\tilde{g}, \tilde{h})$ and set $\{t_1, \cdots, t_m\}$ to $\mathcal{O}^{\mathsf{Sig}}$ then for $i \in [m]$ $\mathcal{B}$ proceeds as in $\mathcal{O}^{\mathsf{ChgPK}}$ with input t_i to obtain pk_i. They then check if $\mathsf{pk}' = \mathsf{Accum}(\mathsf{pk}_1, \cdots, \mathsf{pk}_m)$.
  - If not, then $\mathcal{B}$ ignores the query.
  - If so, $\mathcal{B}$ samples $e, s \leftarrow_\$ \mathbb{Z}_q$ uniformly at random. If $(\tilde{g}||\tilde{h}||\tilde{g}^s\tilde{h}^e||m, \cdot) \in \mathsf{RO}_2$ then $\mathcal{B}$ aborts (as e, s were chosen randomly this occurs with negligible probability), otherwise $\mathcal{B}$ adds $(\tilde{g}||\tilde{h}||\tilde{g}^s\tilde{h}^e||m, e)$ to RO_2. It sends (s, e) to $\mathcal{A}$.
    Note that (s, e) satisfies $\mathsf{H}_2(\tilde{g}||\tilde{h}||\tilde{g}^s\tilde{h}^e||m) = e$.

Now $\mathcal{A}$ outputs $((\tilde{g}, \tilde{h}), \{t_1, \cdots, t_m\}, m^*, (s^*, e^*))$. With significant probability ϵ, $\mathcal{A}$'s output is a successful forgery, i.e., denoting $r^* := \tilde{g}^{s^*}\tilde{h}^{e^*}$, it holds that $(\tilde{g}||\tilde{h}||r^*||m^*, e^*) \in \mathsf{RO}_2$, for all $i \in [m]$ there exists $(t_i, \tilde{t}_i, \tilde{g}_i, \tilde{h}_i, \cdot) \in \mathsf{RO}_1$, and letting $\tilde{t} = \sum_{i \in [m]} \tilde{t}_i$, then $\tilde{g} = g^{\tilde{t}}$, $\tilde{h} = \mathsf{vk}^{\tilde{t}}$. Via the forking lemma we can rewind to the point that $\mathcal{A}$ queried $(\tilde{g}||\tilde{h}||r^*||m^*)$ to the H_2 random oracle (if $(\tilde{g}||\tilde{h}||r^*||m^*, e^*)$ was added to RO_2 by the signature oracle, this would not be a forgery), and sample a different value $\tilde{e}$, continuing as normal. Then, with non negligible probability $\mathcal{A}$ outputs $(\tilde{g}, \tilde{h})$, $\{t'_1, \cdots, t'_m\}$, m^*, (s', e') which also satisfy $\tilde{g}^{s'}\tilde{h}^{e'} = r^*$. This implies $s^* + xe^* = s' + xe'$, and so x can be recovered breaking the discrete logarithm problem.

Accumulation. Assume $\mathsf{H}_1 : \{0,1\}^* \to \mathbb{G}, \mathsf{H}_2 : \{0,1\}^* \to \mathbb{Z}_q^*$ are modelled as random oracles. Consider any PT adversary $\mathcal{A}$ for the accumulation security of the Schnorr AKRS with $s = \mathsf{poly}(\lambda)$ queries to the random oracle for H_1. We show that either $\mathcal{A}$'s success probability is upper bounded by $1/q$, or there exists a PT adversary $\mathcal{B}$ which can use $\mathcal{A}$ to break the discrete logarithm problem.

The reduction works as follows:
Algorithm $\mathcal{B}$ receives Y from its challenger, where x such that $Y = g^x$ is unknown to $\mathcal{B}$. Next $\mathcal{B}$ initializes empty lists $\mathsf{RO}_1, \mathsf{RO}_2$; samples $\mathsf{sk}_1, \cdots \mathsf{sk}_n \leftarrow_\$ \mathbb{Z}_q$; samples $a^* \leftarrow_\$ [s]$; sends $\mathsf{sk}_1, \cdots, \mathsf{sk}_n$ to $\mathcal{A}$; and simulates the random oracles H_1, H_2 as follows:

- If $\mathcal{A}$ queries $t \in \mathbb{Z}_q$ to H_1, then $\mathcal{B}$ checks if there exist $\tilde{t}, \tilde{g}$ such that $(t, \tilde{t}, \tilde{g}) \in \mathsf{RO}_1$. If so, it returns $\tilde{g}$. Else, if this is the a^*th distinct query to this oracle, then $\mathcal{B}$ sets $\tilde{g} := Y$, adds $(t, \cdot, \tilde{g})$ to RO_1, and returns $\tilde{g}$ to $\mathcal{A}$. Otherwise, it samples $\tilde{t} \leftarrow_\$ \mathbb{Z}_q$ uniformly at random, sets $\tilde{g} := g^t$, adds $(t, \tilde{t}, \tilde{g})$ to RO_1, and returns $\tilde{g}$ to $\mathcal{A}$.

– If $\mathcal{A}$ queries $\alpha \in \{0,1\}^*$ to H_2, then $\mathcal{B}$ checks if there exist β such that $(\alpha, \beta) \in \mathsf{RO}_2$. If so, it returns β. Else, it samples $\beta \leftarrow_\$ \mathbb{Z}_q$ uniformly at random, adds (α, β) to RO_2, and returns β to $\mathcal{A}$.

Now $\mathcal{A}$ outputs $(\{(t_i, \mathsf{pk}_i, \hat{m}_i, \hat{\sigma}_i)\}_{i\in[k+1,k+l]}, (t_1, \cdots, t_k), (id_1, \cdots, id_k), m^*, (s^*, e^*)$. For all $i \in [1, k+l]$, let $pk_i = (\mathsf{H}_1(t_i), h'_i)$ (for $i \in [k]$ we have that $h'_i = \mathsf{H}_1(t_i)^{sk_{id_i}}$). Let $\tilde{g} = \Pi_{i\in[k+l]}\mathsf{H}_1(t_i)$ and $\tilde{h} = \Pi_{i\in[1,k+l]}h'_i$. Let sk^* be the discrete logarithm of $\tilde{h}$ with respect to $\tilde{g}$. With probability $\epsilon_{\mathcal{A}}$, $\mathcal{A}$'s output is a valid Schnorr AKRS, i.e. it holds that $(\tilde{g}||\tilde{h}||R^*||m^*, e^*) \in \mathsf{RO}_2$, $\tilde{g}^{s^*}\tilde{h}^{e^*} = R^*$. Via the generalized forking lemma given in [BCJ08], we can rewind to the point that $\mathcal{B}$ was input $((\tilde{g}||\tilde{h}||R^*||m)$ to the H_2 random oracle, and sample a different value $\tilde{e}$, continuing as normal. Then, $\mathcal{A}$ outputs $(\cdot, \cdot, \cdot, m^*, (\tilde{s}, \tilde{e}))$ such that $\tilde{g}^{\tilde{s}}\tilde{h}^{\tilde{e}} = R^*$. This implies $\tilde{s} + \tilde{e} \cdot sk^* = s^* + e^* \cdot sk^*$ and so sk^* can be recovered.

With probability $\epsilon_{\mathcal{A}}$, for all $i \in [k+1, k+l]$ the $\hat{\sigma}_i = (\hat{s}_i, \hat{e}_i)$ output by $\mathcal{A}$ are each a valid Schnorr AKRS, i.e. it holds that $(\mathsf{H}_1(t_i)||h'_i|\hat{R}_i||\hat{m}_i) \in \mathsf{RO}_2$, $\mathsf{H}_1(t_i)^{\hat{s}_i}h_i'^{\hat{e}_i} = \hat{R}_i$. Then for all $i \in [k+1, k+l]$, letting $\hat{\sigma}_i = (\hat{s}_i, \hat{e}_i)$, we can then rewind to the point that $\mathcal{B}$ was input $(\mathsf{H}_1(t_i)||h'_i|\hat{R}_i||\hat{m}_i)$ to the H_2 random oracle, and sample a different value e'_i, continuing as normal. Then, $\mathcal{A}$ outputs $(\{\cdots, (t_i, \mathsf{pk}_i, \hat{m}_i, (s'_i, e'_i)), \cdots\}, \cdot, \cdot, \cdot, \cdot)$ such that $\mathsf{H}_1(t_i)^{s'_i}h_i'^{e'_i} = \hat{R}_i$. Letting $h'_i = \mathsf{H}_1(t_i)^{sk_{id_i}}$ for $i \in [k+1, k+l]$, this implies $\hat{s}_i + \hat{e}_i \cdot sk_{id_i} = s'_i + e'_i \cdot sk_{id_i}$. Therefore, for all $i \in [k+1, k+l]$ sk_{id_i} can be recovered.

If for all $i \in [k+l]$, $sk_{id_i} = sk^*$ then for all $i, j \in [k]$, $id_i = id_j$ and $\forall i \in [k+1, k+l]$ there exists $j \in [k]$ such that $\mathsf{ChgPK}(sk_{id_j}, t_i) = pk_i$, so $\mathcal{A}$ is not successful. Therefore, $\exists i^* \in [k+l]$ such that $sk_{id_{i^*}} \neq sk^*$. If $\mathsf{H}_1(t_{i^*}) \neq Y$ then $\mathcal{B}$ aborts, which occurs with probability $(a-1)/a$. For all $i \in [k+l]\backslash\{i^*\}$, let $(t_i, \tilde{t}_i, \cdot) \in RO_1$. Then,

$$sk^* = \frac{\Sigma_{i\in[k+l]\backslash\{i^*\}} sk_{id_i} \cdot \tilde{t}_i + x \cdot sk_{id_{i^*}}}{\Sigma_{i\in[k+l]\backslash\{i^*\}} \tilde{t}_i + x}$$

(as $\tilde{g} \neq 1, \Sigma_{i\in[k+l]\backslash\{i^*\}}\tilde{t}_i + x \neq 0$). $\mathcal{B}$ outputs

$$\frac{\Sigma_{i\in[k+l]\backslash\{i^*\}}(sk_{id_i} - sk^*) \cdot \tilde{t}_i}{sk^* - sk_{id_{i^*}}}.$$

Therefore, the probability of success for $\mathcal{B}$ is $\epsilon_{\mathcal{A}}/a$, which is non-negligible.

This concludes the proof that, if the discrete logarithm problem is hard, then no PT algorithm $\mathcal{A}$ can break accumulation soundness with significant probability.

Compiling to UCAL and UCL Combined with any Ring Signature scheme the above AKRS gives a UCAL-Ring Signature with a very small overhead: for the signature size it is 1 additional Schnorr Signature, while all the extra computational costs are insignificant. Then, linking ℓ signatures requires a group multiplication of ℓ elements and a Schnorr Signature. Verifying the linking of ℓ signatures requires ℓ group multiplications and a Schnorr-Signature verification. For UCL the main efficiency overhead comes from the NIZK. Our Schnorr-based AKRS allows for the k-out-of-n NIZK by Attema et. al. [ACF21] to be used (setting $k = 1$), which gives similar asymptotic performance to the state-of-the-art on Ring Signatures [GK15,BCC+15,LPQ18,LRR+19,YSL+20,YEL+21].

6.2 Lattice Construction

Our Lattice-based AKRS is based on the Fiat-Shamir signature scheme by Lyubashevsky [Lyu12], which can be seen as the Lattice analogue of Schnorr signatures. We show how to bootstrap this

$\mathsf{KGen}(1^\lambda)$
- $\boldsymbol{S} \leftarrow\$ [-d,d]^{\mu\times k}$
  $\mathsf{sk} := \boldsymbol{S}$
- $\boldsymbol{A} \leftarrow\$ \mathbb{Z}_q^{n\times\mu}$, $T = \boldsymbol{AS}$
  $\mathsf{vk} := (\boldsymbol{A}, \boldsymbol{T})$
- Output $(\mathsf{sk}, \mathsf{vk})$

$\mathsf{ChgPK}(\mathsf{sk}, t)$
- $\tilde{\boldsymbol{A}} \leftarrow \mathsf{H}_1(t)$;
- $\tilde{\boldsymbol{T}} \leftarrow \tilde{\boldsymbol{A}}\boldsymbol{S}$
- Output $(\tilde{\boldsymbol{A}}, \tilde{\boldsymbol{T}})$

$\mathsf{Sig}(\mathsf{sk}, (\tilde{\boldsymbol{A}}, \tilde{\boldsymbol{T}}), m)$
- $\boldsymbol{k} \leftarrow\$ D_\sigma^\mu$, $\boldsymbol{r} \leftarrow \tilde{\boldsymbol{A}}\boldsymbol{k}$
- $\boldsymbol{e} \leftarrow \mathsf{H}_2(\tilde{\boldsymbol{A}}||\tilde{\boldsymbol{T}}||\boldsymbol{r}||m)$
- $\boldsymbol{s} \leftarrow \boldsymbol{Se} + \boldsymbol{k}$
- Output $(\boldsymbol{s}, \boldsymbol{e})$ with probability $\Pr = \min\left\{\frac{D_\sigma^\mu(\boldsymbol{z})}{MD_{\boldsymbol{Se},\sigma}^\mu(\boldsymbol{s})}, 1\right\}$
- Otherwise repeat.

$\mathsf{Vf}((\tilde{\boldsymbol{A}}, \tilde{\boldsymbol{T}}), m, (\boldsymbol{s}, \boldsymbol{e}))$
- Accept if:
  1. $\boldsymbol{e} = \mathsf{H}_2(\tilde{\boldsymbol{A}}||\tilde{\boldsymbol{T}}||\tilde{\boldsymbol{A}}\boldsymbol{s} - \tilde{\boldsymbol{T}}\boldsymbol{e}||m)$
  2. $\|\boldsymbol{s}\| \leq \eta\sigma\sqrt{\mu}$
- else reject

$\mathsf{Accum}(\left(t_i, (\tilde{\boldsymbol{A}}_i, \tilde{\boldsymbol{T}}_i)\right)_{i=1}^{\ell})$
- **if** $\tilde{\boldsymbol{A}}_i \neq \mathsf{H}_1(t_i)$ **then return** $\perp$
- $\tilde{\boldsymbol{A}} \leftarrow \sum_i \tilde{\boldsymbol{A}}_i$
- $\tilde{\boldsymbol{T}} \leftarrow \prod_i \tilde{\boldsymbol{T}}_i$
- Output $\mathsf{pk} := (\tilde{\boldsymbol{A}}, \tilde{\boldsymbol{T}})$

Fig. 8. Lattice-based AKRS

signature scheme to an AKRS. For the sake of simplicity we describe the scheme w.r.t. integer lattices (based on SIS). However, it extends normally to ideal lattices (based on ring-SIS). The construction is in fig. 8. where in the above $D_{\boldsymbol{v},\sigma}^\mu(\cdot)$ is the discrete normal distribution over $\mathbb{Z}^\mu$ centered around $\boldsymbol{v} \in \mathbb{Z}^\mu$ ($D_\sigma^\mu(\cdot)$ centered around $\boldsymbol{v} = 0$ resp.) with standard deviation σ and $n, \mu, k, \sigma, M, \eta, d$ are parameters. We refer to [Lyu12] for details.

Security and parameters For Class hiding to hold it is sufficient to show that $(\tilde{\boldsymbol{A}}_1, \tilde{\boldsymbol{A}}_1\boldsymbol{S}) \approx (\tilde{\boldsymbol{A}}_2, \tilde{\boldsymbol{A}}_2\boldsymbol{S}) \approx \ldots \approx (\tilde{\boldsymbol{A}}_\ell, \tilde{\boldsymbol{A}}_\ell\boldsymbol{S})$, where $\tilde{\boldsymbol{A}}_i \leftarrow\$ \mathbb{Z}_q^{n\times\mu}, \boldsymbol{S} \leftarrow\$ [-d,d]^{\mu\times k}$. If we apply the Leftover hash lemma [HILL99] to the hash function:

$$f(\boldsymbol{S}) = \begin{pmatrix} \tilde{\boldsymbol{A}}_1 & 0 & \ldots & 0 \\ 0 & \tilde{\boldsymbol{A}}_2 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & \tilde{\boldsymbol{A}}_\ell \end{pmatrix} \cdot \begin{pmatrix} \boldsymbol{S} \\ \boldsymbol{S} \\ \vdots \\ \boldsymbol{S} \end{pmatrix} = \begin{pmatrix} \tilde{\boldsymbol{A}}_1\boldsymbol{S} \\ \tilde{\boldsymbol{A}}_2\boldsymbol{S} \\ \vdots \\ \tilde{\boldsymbol{A}}_\ell\boldsymbol{S} \end{pmatrix}$$

then the statistical distance of $f(\boldsymbol{S})$ from the uniform over $\mathbb{Z}_q^{n\times k}$ is negligible ($2^{-\lambda}$) if $\mu\log(2d+1) \geq k\ell n + 2\lambda$. So if we set ℓ_{max} to be the maximum number of re-randomizations of the public key and $\mu \geq (k\ell_{\mathsf{max}}n + 2\lambda)/\log(2d+1)$ then we get class-hiding. The rest of the lattice parameters are set according to [Lyu12].

Unforgeability then comes directly from the unforgeability of [Lyu12] signatures and Accumulation Soundness is analogous to the Schnorr Signatures AKRS construction.

Compiling to UCL and UCAL As in the Schnorr signatures case, the overhead of bootstrapping a ring signature scheme to a UCAL one with the above AKRS is minimal. We further note that concrete costs of our AKRS (and thus the compilation to UCAL) can be optimized using follow-up optimizations on the Lyubashevsky signatures [DDLL13]. For UCL any general purpose NIZK for lattice relations can be used; our AKRS language is a basic lattice one.

7 Conclusions

In this paper, we have introduced Ring Signatures (RS) with User-Controlled Linkability (UCL) and User-Controlled Autonomous Linkability (UCAL). RS-UCL allows for both implicit and explicit linkability of signatures. Thus, signers can decide to make their signatures linkable either when

issuing the signatures (by using the same scope in all signatures to be linked) or at a later time (by providing an explicit linking proof). We note that UCL was recently defined for group signatures. However, we argue that ring signatures are better suited for distributed applications, as no group manager is necessary. As such RS-UCL finds direct applicability in smart metering or smart mobility applications as argued in section 1. Also, RS-UCL may be used in e-voting protocols where each election could use a different scope so that (i) double-voting in the same election round would be detected by implicit linkability, and (ii) voters can use the same (registered) key across election rounds.

RS-UCAL gives even more power to signers as they now can ensure unlinkability of their signatures, even if signatures use the same scope. Still, at a later time signers can prove linkability with an explicit linking proof.

We show how to upgrade *any* RS to RS-UCAL by means of a new cryptographic primitive that we have introduced in this paper and that we have labelled Anonymous Key Randomisable Signatures (AKRS). We have also shown how AKRS can be used to instantiate RS-UCL. We note that AKRS may be of independent interest and we have introduced two AKRS instantiations, one in prime-order groups and one based on lattices.

Acknowledgements

This work has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program under projects PICOCRYPT (grant agreement No. 101001283), and TERMINET (grant agreement No. 957406), by the Spanish Government under projects SCUM (ref. RTI2018-102043-B-I00), and RED2018-102321-T, by the Madrid Regional Government under project BLOQUES (ref. S2018/TCS-4339). This work is also supported by a grant from Nomadic Labs and the Tezos foundation.

References

[ACF21] T. Attema, R. Cramer, and S. Fehr. Compressing proofs of k-out-of-n partial knowledge. In *Annual International Cryptology Conference*, pages 65–91. Springer, 2021.

[ACHO13] Ma. Abe, S. S. M. Chow, K. Haralambiev, and M. Ohkubo. Double-trapdoor anonymous tags for traceable signatures. *International journal of information security*, 12(1):19–31, 2013.

[BCC04] E.F. Brickell, J. Camenisch, and L. Chen. Direct anonymous attestation. In *ACM CCS*, 2004.

[BCC+15] J. Bootle, A. Cerulli, P. Chaidos, E. Ghadafi, J. Groth, and C. Petit. Short accountable ring signatures based on ddh. In *European Symposium on Research in Computer Security*, pages 243–265. Springer, 2015.

[BCC+16] J. Bootle, A. Cerulli, P. Chaidos, E. Ghadafi, and J. Groth. Foundations of fully dynamic group signatures. In *ACNS*, 2016.

[BCJ08] A. Bagherzandi, J.-H. Cheon, and S. Jarecki. Multisignatures secure under the discrete logarithm assumption and a generalized forking lemma. In *CCS*, 2008.

[BFG+13] D. Bernhard, G. Fuchsbauer, E. Ghadafi, N. P. Smart, and B. Warinschi. Anonymous attestation with user-controlled linkability. *International Journal of Information Security*, 12(3), 2013.

[BHKS18] M. Backes, L. Hanzlik, K. Kluczniak, and J. Schneider. Signatures with flexible public key: Introducing equivalence classes for public keys. In *Asiacrypt*, 2018.

[BKM09] Adam Bender, Jonathan Katz, and Ruggero Morselli. Ring signatures: Stronger definitions, and constructions without random oracles. *Journal of Cryptology*, 22(1):114–138, 2009.

[BMW03] M. Bellare, D. Micciancio, and B. Warinschi. Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. In *Eurocrypt*, 2003.

[BSZ05] M. Bellare, H. Shi, and C. Zhang. Foundations of group signatures: The case of dynamic groups. In *CT-RSA*, 2005.

[CDL16a] J. Camenisch, M. Drijvers, and A. Lehmann. Anonymous attestation using the strong diffie hellman assumption revisited. In *TRUST*, 2016.

[CDL16b] J. Camenisch, M. Drijvers, and A. Lehmann. Universally composable direct anony- mous attestation. In *PKC*, 2016.

[CS97] J. Camenisch and M. Stadler. Efficient group signature schemes for large groups. In *Annual International Cryptology Conference*, pages 410–424. Springer, 1997.

[CvH91] D. Chaum and E. van Heyst. Group signatures. In *Eurocrypt*, 1991.

[DDLL13] L. Ducas, A. Durmus, T. Lepoint, and V. Lyubashevsky. Lattice signatures and bimodal gaussians. In *Annual Cryptology Conference*, pages 40–56. Springer, 2013.

[DL21] J. Diaz and A. Lehmann. Group signatures with user-controlled and sequential linkability. In *PKC*, 2021.

[FGL21] A. Fraser, L. Garms, and A. Lehmann. Selectively linkable group signatures - stronger security and preserved verifiability. In *CANS*, 2021.

[GK15] J. Groth and M. Kohlweiss. 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*, pages 253–280. Springer, 2015.

[GL19] L. Garms and A. Lehmann. Group signatures with selective linkability. In *PKC*, 2019.

[HILL99] J. Håstad, R. Impagliazzo, L. A. Levin, and M. Luby. A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 28(4):1364–1396, 1999.

[HLC+11] J. Y. Hwang, S. Lee, B.-H. Chung, H. S. Cho, and D. Nyang. Short group signatures with controllable linkability. In *Workshop on Lightweight Security & Privacy: (LightSec)*, 2011.

[HLC+13] J. Y. Hwang, S. Lee, B.-H. Chung, H. S. Cho, and D. Nyang. Group signatures with controllable linkability for dynamic membership. *Information Sciences*, 222, 2013.

[KTY04] A. Kiayias, Y. Tsiounis, and M. Yung. Traceable signatures. In *Eurocrypt*, 2004.

[LJ14] B. Libert and M. Joye. Group signatures with message-dependent opening in the standard model. In *Topics in Cryptology - CT-RSA*, 2014.

[LMN16] B. Libert, F. Mouhartem, and K. Nguyen. A lattice-based group signature scheme with message-dependent opening. In *ACNS*, 2016.

[LPQ18] B. Libert, T. Peters, and C. Qian. Logarithmic-size ring signatures with tight security from the ddh assumption. In *European Symposium on Research in Computer Security*, pages 288–308. Springer, 2018.

[LRR+19] R. W. F. Lai, V. Ronge, T. Ruffing, D. Schröder, S. A. K. Thyagarajan, and J. Wang. Omniring: Scaling private payments without trusted setup. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 31–48, 2019.

[LWW04] J. K. Liu, V. K. Wei, and D. S. Wong. Linkable spontaneous anonymous group signature for ad hoc groups. In *ACISP*, 2004.

[Lyu12] V. Lyubashevsky. Lattice signatures without trapdoors. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 738–755. Springer, 2012.

[OSEH13] K. Ohara, Y. Sakai, K. Emura, and G. Hanaoka. A group signature scheme with unbounded message-dependent opening. In *ASIA-CCS*, 2013.

[PS19] S. Park and A. Sealfon. It wasn't me! In *Annual International Cryptology Conference*, pages 159–190. Springer, 2019.

[RST01] R. L. Rivest, A. Shamir, and Y. Tauman. How to leak a secret. In *Asiacrypt*, 2001.

[SALY17] S.-F. Sun, M. H. Au, J. K. Liu, and T. H. Yuen. Ringct 2.0: A compact accumulator-based (linkable ring signature) protocol for blockchain cryptocurrency monero. In *ESORICS*, 2017.

[SEH+12] Y. Sakai, K. Emura, G. Hanaoka, Y. Kawai, T. Matsuda, and K. Omote. Group signatures with message-dependent opening. In *PAIRING*, 2012.

[SSU14] D. Slamanig, R. Spreitzer, and T. Unterluggauer. Adding controllable linkability to pairing-based group signatures for free. In *International Conference on Information Security*, 2014.

[YEL+21] T. H. Yuen, M. F Esgin, J. K. Liu, M. H. Au, and Z. Ding. Dualring: generic construction of ring signatures with efficient instantiations. In *Annual International Cryptology Conference*, pages 251–281. Springer, 2021.

[YSL+20] T. H. Yuen, S.-F. Sun, J. K. Liu, M. H. Au, M. F Esgin, Q. Zhang, and D. Gu. Ringct 3.0 for blockchain confidential transaction: Shorter size and stronger security. In *International Conference on Financial Cryptography and Data Security*, pages 464–483. Springer, 2020.

[ZWC19] T. Zhang, H. Wu, and S. S. M. Chow. Structure-preserving certificateless encryption and its application. In *Topics in Cryptology - CT-RSA*, 2019.