DualDory: Logarithmic-verifier linkable ring signatures through preprocessing

Jonathan Bootle[1], Kaoutar Elkhiyaoui[1], Julia Hesse[1], and Yacov Manevich[2]

[1] IBM Research - Zurich
{jbt,kao,jhs}@zurich.ibm.com
[2] IBM Research - Haifa
yacovm@il.ibm.com

Abstract. A linkable ring signature allows a user to sign anonymously on behalf of a group while guaranteeing that multiple signatures from the same user can be detected. For applications such as privacy-preserving e-voting and e-cash, linkable ring signatures can significantly improve privacy and anonymity guarantees. To scale to systems involving large numbers of users, we need short signatures with fast verification. Concretely efficient ring signatures currently rely on a trusted authority maintaining a master secret, or follow an accumulator-based approach that requires a trusted group setup.

In this work, we construct the first linkable ring signature with both logarithmic signature size and verification that does not require any trusted mechanism. Our scheme, which relies on discrete-log type assumptions and a bilinear map, improves upon a recent concise ring signature called DualRing by integrating improved preprocessing arguments to reduce the verification time from linear to logarithmic in the size of the ring. Our ring signature allows signatures to be linked based on what message is signed, ranging from linking signatures on any message to only signatures on the same message.

We provide benchmarks for our scheme and prove its security under standard assumptions. The proposed linkable ring signature is particularly relevant to use cases that require privacy-preserving enforcement of threshold policies in a fully decentralized context and e-voting.

1 Introduction

Ring signatures allow a signer to spontaneously choose a set of public keys called ring and sign messages anonymously on behalf of the ring. Linkable ring signatures, on the other hand, prevent a signer from signing twice without this being detected.

Linkable ring signatures are particularly useful in e-voting applications, in which a signer (i.e., voter) must cast at most one vote. Multiple votes cast by the same voter are easily detected and only the first vote is counted. Compared to linkable group signatures, linkable ring signatures allow voters to bring their own public keys without any additional setup, which fits neatly into the emerging paradigm of self-sovereign identity.

Linkable ring signatures can also be used to implement *threshold ring signatures* in a straightforward fashion. Threshold ring signatures allow signers to generate t signatures on behalf of the ring while assuring the verifier that the signatures were produced

by t distinct signers. An application of threshold ring signatures is decentralized threshold policy enforcement for privacy-preserving Defi applications. For example, regulation could mandate that a financial transaction is only valid if it has been endorsed by t out of n authorities. If the authorities are independent, threshold signatures cannot be used directly. Multi-signatures, on the other hand, reveal the identities of the signers, leaking potentially information about the origin of the transaction. Threshold ring signatures, however, allow the authorities to sign anonymously guaranteeing zero leakage.

If a linkable ring signature is to be used in e-voting or threshold ring signatures, it must be efficiently verifiable. In these settings, the verifier is required to check the validity of multiple signatures at once. Sublinear-verifier (linkable) ring signatures require a trusted setup, where the system parameters are computed using a secret trapdoor and security relies, for example, on q-SDH, subgroup or RSA assumptions (cf. Table 1). (Linkable) ring signatures that do without such a setup in favor of a *transparent* one see their verification cost linearly increase with the size of the ring. A transparent setup is one where the system parameters are generated without a secret trapdoor, yielding itself easily to decentralization.

In this paper, we introduce DualDory, a linkable ring signature with a logarithmic verifier and transparent setup. We start from the observation that with preprocessing we can bring the linear verification cost of DualRing [26] down to logarithmic. DualRing is a ring signature that incorporates discrete-logarithm-based interactive arguments building on [11,13] to obtain logarithmic-size ring signatures, albeit with a linear verifier. Our idea is to replace these with the pairing-based interactive arguments from Dory [20], which, thanks to an offline preprocessing phase, have logarithmic verifier time. To avoid repeated preprocessing for signatures with respect to one-time rings, we recommend our scheme for rings that are either *static*, which are relevant to threshold policy enforcement, or *updatable* with a subset of signers joining or leaving, which are well-suited for e-voting.

We further enhance DualRing with linkability through *deterministic* tags. More precisely, we combine Pedersen commitments and signatures of knowledge to show that the tag is computed using one of the secret keys in the ring. As a positive side effect, we are able to precompute the linear work of signing, leaving only a constant number of operations to be performed when messages are known.

Contributions. The contributions of this paper are three-fold:

- DualDory: a linkable ring signature with a *logarithmic* verifier and a *transparent* setup. We leverage an argument of knowledge of bilinear pairing products [20], which thanks to an *one-time offline preprocessing* phase gives us a logarithmic verifier. Furthermore, while signature generation is linear in the size of the ring, most of the work can be precomputed before knowing the message.
- A performance evaluation that demonstrates the practicality of DualDory.
- A security analysis that proves that DualDory is a secure linkable ring signature under the SXDH assumption.

Related Work. Signatures that allow for anonymous signing on behalf of a group have been extensively studied since the early ninetees [16,4,22,21,10,17,5,14,19,18,26], with

ongoing attempts to reduce signature sizes and computational complexity. We give an overview in Table 1. If a group manager is trusted with the issuance of keys, so-called group signatures with constant signature size and computations are known [10]. The trust model can be reduced to an authority issuing only an RSA group setup, which anybody can use to generate their own keys and form ad-hoc groups referred to as *rings* [22]. Schemes with constant size and computational complexity are known in this setting [25,5]. Recently, research has focused on improving efficiency also for ring signatures based on discrete-log-type assumptions. These schemes do not rely on any authority and can be deployed in elliptic curve groups which are several orders of magnitude smaller than RSA groups. Unfortunately, it has proven to be difficult to achieve optimal asymptotics for discrete-log-based ring signatures (see Table 1 for references). Chandran et al. [14] were the first to achieve sublinear signature sizes, namely $O(\sqrt{n})$. Subsequently, Groth and Kohlweiss [19] achieved logarithmic signature sizes through concise one-out-of-many proofs, inspiring subsequent works such as DualRing [26], which we describe in more detail in Section 2.1. However, all the aforementioned schemes take linear time to verify a signature.

	Sign	Verify	Sig. size	Assumptions and model		KGen Authority	Malicious pk	Linkable
Ateniese et al. [4]	$O(1)$	$O(1)$	$O(1)$	strong RSA, DDH	RO	●	❍	(●)
Rivest et al. [22]	$O(n)$	$O(n)$	$O(n)$	TD-OWP	RO	❍	❍	❍
Liu et al. [21]	$O(n)$	$O(n)$	$O(n)$	DDH	RO	❍	❍	●
BBS Signatures [10]	$O(1)$	$O(1)$	$O(1)$	q-SDH, DLin	RO	●	❍	(●)
Dodis et al. [17]	$O(1)$	$O(1)$	$O(1)$	strong RSA	RO	❍	❍	❍
Au et al. [5]	$O(1)$	$O(1)$	$O(1)$	strong RSA,DDH,LD-RSA	RO	❍	❍	●
Chandran et al. [14]	$O(n)$	$O(n)$	$O(\sqrt{n})$	strong DDH, SUB	CRS	❍	●	❍
Groth et al. [19]	$O(n\log n)$	$O(n)$	$O(\log n)$	DLOG	RO	❍	●	❍
CLSAG [18]	$O(n)$	$O(n)$	$O(n)$	OM-LC-DLOG,DDH	RO	❍	●	●
DualRing-EC [26]	$O(n)$	$O(n)$	$O(\log n)$	DLOG	RO	❍	❍	❍
DualDory, **this work**	$O(n)$	$O(n)+O(\log n)$	$O(\log n)$	SXDH	RO	❍	❍	●

Table 1: Development of the asymptotic efficiency of practical RSA- and DLOG-based signature schemes that allow signing on behalf of a group with n members. If applicable, linking costs are negligible. Costs depict exponentiations in the group for Sign and Verify, and number of group elements for Signature size. In DualDory, verification time is split into preprocessing effort per group, plus verification effort per signature. ● means applicable/required, ❍ means not applicable/required. (●) means linkable only by the key generation authority.

Paper Organization. The paper is organized as follows. Section 2 provides an overview of DualDory and the techniques used to achieve linkability and logarithmic verification. Section 3 introduces the cryptographic assumptions and building blocks, whereas Section 4 formalizes the security of linkable ring signatures. Section 5 describes DualDory and analyzes its security. Finally, Section 6 evaluates the performances of DualDoray and Section 7 concludes the paper.

2 Technical overview

In this section, we explain our new construction of a linkable ring signature. We obtain our construction by using the basic DualRing ring signature of [26] as a starting point, and modifying it in two ways.

1. We make the scheme prefix linkable according to Definition 14 by applying a tagging technique and using the tag proofs in Section 5.1.
2. We simultaneously improve the proof size and online signature verification time of the basic signature scheme to logarithmic in the number of users using the pairing-based inner-product arguments of Lee [20].

2.1 DualRing

Given public keys $pk_1, \ldots, pk_n$, a DualRing signature [26] has the signer prove that they know a single public key sk_j corresponding to one of the public keys, without leaking sk_j or the index j. The construction is built in two parts; a basic signature scheme which builds on the classic construction of ring signatures from [3], and has signatures of size $O(n)$, and a "sum argument" which compresses basic signatures to size $O(\log n)$.

In more detail, basic signatures consist of elements $(X, c_1, ..., c_n, y)$ satisfying the following equations:

$$H(pk_1, ..., pk_n, X, m) = \sum_{i=1}^{n} c_i \tag{1}$$

$$P^y / X = \prod_{i=1}^{n} pk_i^{c_i} \tag{2}$$

Checking Equation 1 and Equation 2 involves calculations on n user public keys $pk_1, \ldots, pk_n$ and n challenges $c_1, \ldots, c_n$, leading to signature sizes and verification time $O(n)$. In [26], the authors observed that P^y/X is a Pedersen commitment to $c_1, \ldots, c_n$ under commitment key $(pk_1, \ldots, pk_n)$, and used a *sum argument* to prove that Equation 1 was satisfied using the committed values.

The sum argument is based on split-and-fold zero-knowledge arguments such as [11,13], which can prove that Equation 1 is satisfied with proof sizes of $O(\log n)$, but still require the verifier to perform a multi-exponentiation in $pk_1, \ldots, pk_n$, which costs at best $O(n/\log n)$ operations using Pippenger's algorithm. Further, the construction does not have the linkability property.

2.2 Adding linkability

First, we explain how to make the construction linkable. We want a linking algorithm to be able to identify signatures made by the same user with the same prefix. To this end, we ask the signer to compute a tag $H'(\mathsf{prfx})^{sk}$, using a similar strategy to prior work [21]. Since the tag is uniquely determined by the user's secret key and the prefix prfx,

this allows an efficient linking algorithm. To ensure that the tag is computed correctly using the same secret key as the rest of the signature, we have the signer produce a Pedersen commitment $\text{com} = P^{sk}Q^r$ to their secret key, and use a "tag proof" based on standard sigma protocols to show that both com and the tag use the same secret key. Note that we cannot perform this consistency check on the user's public key, since this would leak the identity of the user.

This leaves us with a further problem. A signer can use DualRing to prove that they know a secret key sk_i corresponding to public key pk_i from a list $pk_1, \ldots, pk_n$, but this proof is not connected with the tag tag or the commitment com. To solve this problem, we use an idea from [19]. Since the signer knows an opening, $\text{com} = P^{sk}Q^r$, they know how to open exactly one of the commitments $\text{com}/pk_1, \ldots, \text{com}/pk_n$ to zero i.e. they know a discrete logarithm r to base Q of com/pk_i.

Applying DualRing to $\text{com}/pk_1, \ldots, \text{com}/pk_n$ and adding the tag proof produces a linkable ring signature where the verifier checks

$$H(\text{com}/pk_1, ..., \text{com}/pk_n, X, m) = \sum_{i=1}^{n} c_i \tag{3}$$

$$P^y/X = \prod_{i=1}^{n} (\text{com}/pk_i)^{c_i} \tag{4}$$

The size of this signature can be reduced to $O(\log n)$ using the same sum argument as [26]. However, the verification time is still $O(n)$.

2.3 Reducing signature size and verification time simultaneously

We explain how to achieve signature verification time of $O(\log n)$ by replacing the sum argument with an improved succinct argument.

The sum argument is based on split-and-fold zero-knowledge arguments such as [11,13]. Lee [20] extends [11,13] to the setting of bilinear pairings. This setting uses a pairing-based commitment scheme which commits to a message $\vec{\Omega} \in \mathbb{G}_1^n$ with commitment key $\vec{\Gamma} \in \mathbb{G}_2^n$ using the commitment $\mathbf{A} = \prod_{i=1}^{n} e(\vec{\Omega}_i, \vec{\Gamma}_i)$ (and similarly for messages $\vec{\Omega} \in \mathbb{G}_2^n$ and keys in $\vec{\Gamma} \in \mathbb{G}_1^n$). Lee's argument, called Dory, allows the prover to prove knowledge of $\vec{\Omega} \in \mathbb{G}_1^n$ and $\vec{\Omega} \in \mathbb{G}_2^n$ satisfying $\mathbf{A} = e(\vec{\Omega}, \vec{\Gamma})$, $\mathbf{B} = e(\vec{\Gamma}, \vec{\Omega})$ and $\mathbf{C} = e(\vec{\Omega}, \vec{\Omega})$, for publicly known commitment keys $\vec{\Gamma} \in \mathbb{G}_1^n$, $\vec{\Gamma} \in \mathbb{G}_2^n$ and target values $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C} \in \mathbb{G}_T$.

When proving statements of this form, the verifier must read the commitment keys in their entirety, leading to $O(n)$ verification costs. However, unlike [11,13], in which calculations on keys must be done online, Dory allows the verifier to preprocess the commitment keys once and for all in an offline phase. Thereafter, the verifier need only use succinct commitments to these keys and incurs $O(\log n)$ costs.

Now, we explain how to apply Dory to Equation 3 and Equation 4. First, we map Equation 3 and Equation 4 to equations over bilinear groups. Imagine that the DualRing scheme has been executed over $\mathbb{G}_1$ of the bilinear group . Consider group element

$e(P,\tilde{P})$ (where $P \in \mathbb{G}_1$ and $\tilde{P} \in \mathbb{G}_2$). Exponentiate using the left and right hand sides of Equation 3, using the bilinearity of e, to get

$$e(P^{H(\mathrm{com}/pk_1,\ldots,\mathrm{com}/pk_n,X,m)},\tilde{P}) = \prod_{i=1}^{n} e(P,\tilde{P}^{c_i}) \tag{5}$$

In a similar way, Equation 6 can be paired with group element $\tilde{P}$ and rearranged to obtain

$$e(P^y/X,\tilde{P}) = \prod_{i=1}^{n} e(\mathrm{com}/pk_i,\tilde{P}^{c_i}) \tag{6}$$

Since the exponentiation and pairing maps are injective, Equation 5 and Equation 6 imply Equation 3 and Equation 4.

Thus, given commitments to $(P,\ldots,P) \in \mathbb{G}_1^n$, $(\tilde{P}^{c_1},\ldots,\tilde{P}^{c_n}) \in \mathbb{G}_2^n$, and $(\mathrm{com}/pk_i)_{i=1}^n \in \mathbb{G}_1^n$, the signer can apply Dory to prove that Equation 5 and Equation 6 hold with the left hand side of each equation as target values. Note that the target value from Equation 5 involves n values, so to avoid $O(n)$ verifier costs here, we replace these values with the commitment to $(\mathrm{com}/pk_1,\ldots,\mathrm{com}/pk_n)$. This is still sufficient for security of DualRing as long as the commitment scheme is binding.

Dory relies on the SXDH assumption for security, which on particular, implies the hardness of the discrete logarithm assumption over $\mathbb{G}_1$, and therefore, the security of DualRing over $\mathbb{G}_1$.

Fast online verification time relies on the verifier being able to compute a commitment to $(\mathrm{com}/pk_1,\ldots,\mathrm{com}/pk_n) \in \mathbb{G}_1^n$ using $O(n)$ offline operations and $O(\log n)$ online operations. Since com depends on randomness r, it is different for every signature, so it is impossible for the verifier to compute this commitment once and for all independently of any signatures. Instead, the verifier computes a commitment $\mathbf{A}_0 := \prod_{i=1}^n e(pk_i,\vec{\tilde{\Gamma}}_i)$ to $(pk_1,\ldots,pk_n)$, and $\tilde{\Gamma} := \prod_{i=1}^n \tilde{\Gamma}_i$ offline, which costs $O(n)$ operations. Unlike in [26], this does not depend on any part of the signature and can be computed once "offline" for each ring and then reused in "online" signature verification. When verifying a signature, the verifier can compute a commitment to $(\mathrm{com}/pk_1,\ldots,\mathrm{com}/pk_n)$ as $e(\mathrm{com},\tilde{\Gamma})/\mathbf{A}_0$.

On logarithmic verification time The construction described above achieves logarithmic online verification time when the list of user public keys in the ring signature is read just once and used to compute a succinct commitment. Logarithmic verification time is achieved in an amortised sense, when verifying many signatures with respect to the same set of users. This is the best one can hope for, as verifying signatures with respect to many different sets of users either requires the verifier to read the set of users each time, or a succinct representation of it (such as a Merkle tree, accumulator, or commitment), which must be computed by somebody who has read the entire set.

Further, since the commitment to the set of user public keys is of the form $\prod_{i=1}^n e(pk_i,\vec{\tilde{\Gamma}}_i)$, given a commitment to a large group of users, it is also easy to update the commitment to include new users by multiplying the commitment by $e(pk_{n+1},\vec{\tilde{\Gamma}}_{n+1})$, or remove existing users by dividing by a suitable value. This means that logarithmic verifier complexity can be maintained up to small changes in the set of users.

Comparison with accumulator-based approaches In our construction, the commitment to the set of public keys related to a given signature acts like an accumulator for those public keys. However, prior accumulator-based approaches rely on either q-type assumptions over bilinear groups, or RSA groups. Both approaches have trapdoors. This means that the public parameters for group and ring signature schemes based on these approaches must be generated by a trusted party or via a secure multiparty computation protocol. By contrast, the public parameters for our scheme can be generated without a trusted setup.

3 Preliminaries

On input the security parameter 1^λ, a *group generator* $\mathsf{G.Gen}(1^\lambda)$ produces public parameters $\mathsf{Gpp} = (p, \mathbb{G}, P)$, where p is a prime of bitlength λ, and $\mathbb{G}$ is a cyclic group of order p with generator P. Similarly, a *bilinear group generator* $\mathsf{BG.Gen}(1^\lambda)$ produces public parameters $\mathsf{BGpp} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, P, \tilde{P})$ where $\mathbb{G}_1 = \langle P \rangle$, $\mathbb{G}_2 = \langle \tilde{P} \rangle$, $\mathbb{G}_T$ are groups of order p. The map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is *bilinear* (for all $u, v \in \mathbb{Z}_p$, $e(P^u, \tilde{P}^v) = e(P, \tilde{P})^{uv}$ and non-degenerate (for all generators P of $\mathbb{G}_1$, $\tilde{P}$ of $\mathbb{G}_2$, $\mathbb{G}_T = \langle e(P, \tilde{P}) \rangle$). If $\vec{P} \in \mathbb{G}_1^n$ and $\vec{\tilde{P}} \in \mathbb{G}_2^n$, then let $e(\vec{P}, \vec{\tilde{P}}) := \prod_{i=1}^n e(P_i, \tilde{P}_i)$.

Notations. We refer to group elements with upper-case letters. Elements in $\mathbb{Z}_p$ are referred to using lower-case letters. We use $\tilde{\star}$ to denote elements in $\mathbb{G}_2$ and bold font to denote elements in $\mathbb{G}_T$.

Definition 1 (DDH assumption). *Let $(p, \mathbb{G}, P) \leftarrow \mathsf{G.Gen}(1^\lambda)$ be a group generator. The DDH assumption holds for* $\mathsf{G.Gen}$ *if the following distributions are indistinguishable:*

$$\{P, U = P^u, V = P^v, W = P^{uv} : u, v \leftarrow \mathbb{Z}_p\} \text{ , and}$$
$$\{P, U = P^u, V = P^v, W = P^w : u, v, w \leftarrow \mathbb{Z}_p\} \ .$$

Definition 2 (DLOG assumption). *Let $(p, \mathbb{G}, P) \leftarrow \mathsf{G.Gen}(1^\lambda)$ be a group generator. The DLOG assumption holds for* $\mathsf{G.Gen}$ *if for all p.p.t. adversaries $\mathcal{A}$, we have*

$$\Pr\left[\mathcal{A}(p, \mathbb{G}, P, U) = u \,\middle|\, \begin{matrix}(p, \mathbb{G}, P) \leftarrow \mathsf{G.Gen}(1^\lambda) \\ u \leftarrow \mathbb{Z}_p \\ U = P^u\end{matrix}\right] \approx 0 \ .$$

Definition 3 (SXDH assumption). *Let $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, P, \tilde{P}) \leftarrow \mathsf{BG.Gen}(1^\lambda)$ be a bilinear group generator. The SXDH assumption holds for* $\mathsf{BG.Gen}$ *if the DDH assumption holds for $\mathbb{G}_1$ and $\mathbb{G}_2$ (replacing* $\mathsf{G.Gen}$ *and tuple $(p, \mathbb{G}, P)$ with* $\mathsf{BG.Gen}$ *and tuple $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, P, \tilde{P})$).*

Definition 4 (DPair assumption). *Let $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, P, \tilde{P}) \leftarrow \mathsf{BG.Gen}(1^\lambda)$ be a bilinear group generator, and let $n = poly(\lambda)$. The double-pairing (DPair) assumption holds for* $\mathsf{BG.Gen}$ *if for all probabilistic polynomial time adversaries $\mathcal{A}$, for $\vec{P} \leftarrow \mathbb{G}_1^n$, the probability that $\mathcal{A}$ can produce $\vec{\tilde{P}} \in \mathbb{G}_2^n$ such that $e(\vec{P}, \vec{\tilde{P}}) = 0$ is negligible.*

3.1 Arguments of knowledge

Definition 5. *A relation $\mathcal{R}$ is a set of tuples $(\mathsf{pp}, \mathbb{x}, \mathbb{w})$ where pp is called the public parameters, $\mathbb{x}$ is called the instance and $\mathbb{w}$ is called the witness. The language $\mathcal{L}_{\mathcal{R}}$ corresponding to $\mathcal{R}$ is the set of triples $(\mathsf{pp}, \mathbb{x})$ such that there exists a witness $\mathbb{w}$ with $(\mathsf{pp}, \mathbb{x}, \mathbb{w}) \in \mathcal{R}$.*

Definition 6. *An interactive argument is a tuple of three algorithms $(\mathsf{G}, \mathsf{P}, \mathsf{V})$ with the following syntax.*

- $\mathsf{G}(1^\lambda, n) \to \mathsf{pp}$. *The generator G is a p.p.t. algorithm which takes the security parameter λ and instance size n as input and outputting public parameters pp.*
- *The prover P and verifier V are p.p.t. interactive algorithms. The prover takes pp, $\mathbb{x}$ and $\mathbb{w}$ as inputs. The verifier takes pp and $\mathbb{x}$ as inputs. An interaction between P and V on inputs s and t, producing transcript tr is denoted by $\mathsf{tr} \leftarrow \langle \mathsf{P}(s), \mathsf{V}(t) \rangle$. The output of V at the end of an interaction is denoted by $\langle \mathsf{P}(s), \mathsf{V}(t) \rangle = b$. If $b = 1$, we say that the transcript is* accepted *by the verifier and if $b = 0$ it is* rejected.

We say that $(\mathsf{G}, \mathsf{P}, \mathsf{V})$ is an argument of knowledge for a relation $\mathcal{R}$ if it satisfies the following completeness and knowledge soundness definitions.

- **Completeness.** For all $\lambda, n \in \mathbb{N}$ and all adversaries $\mathcal{A}$,

$$\Pr\left[\begin{array}{c}(\mathsf{pp}, \mathbb{x}, \mathbb{w}) \in \mathcal{R} \\ \wedge \\ \langle \mathsf{P}(\mathsf{pp}, \mathbb{x}, \mathbb{w}), \mathsf{V}(\mathsf{pp}, \mathbb{x}) \rangle = 1\end{array}\middle|\begin{array}{c}\mathsf{pp} \leftarrow \mathsf{G}(1^\lambda, n) \\ (\mathbb{x}, \mathbb{w}) \leftarrow \mathcal{A}(\mathsf{pp})\end{array}\right] = 1 \ .$$

- **Knowledge soundness.** For all $\lambda, n \in \mathbb{N}$, there exists an expected polynomial time emulator E such that for all efficient adversaries $\mathcal{A}$, we have

$$\begin{aligned}&\Pr\left[\mathcal{A}(\mathsf{st}, \mathsf{tr}) = 1 \middle| \begin{array}{r}\mathsf{pp} \leftarrow \mathsf{G}(1^\lambda, n) \\ (\mathbb{x}, \mathsf{st}) \leftarrow \mathcal{A}(\mathsf{pp}) \\ \mathsf{tr} \leftarrow \langle \mathcal{A}(\mathsf{st}), \mathsf{V}(\mathsf{pp}, \mathbb{x}) \rangle\end{array}\right] \\ \approx &\Pr\left[\begin{array}{c}\mathcal{A}(\mathsf{st}, \mathsf{tr}) = 1 \quad \wedge \\ (\mathsf{tr} \text{ is accepting} \to (\mathsf{pp}, \mathbb{x}, \mathbb{w}) \in \mathcal{R})\end{array}\middle|\begin{array}{r}\mathsf{pp} \leftarrow \mathsf{G}(1^\lambda, n) \\ (\mathbb{x}, \mathsf{st}) \leftarrow \mathcal{A}(\mathsf{pp}) \\ (\mathsf{tr}, \mathbb{w}) \leftarrow \mathsf{E}^{\mathcal{A}(\mathsf{st})}(\mathsf{pp}, \mathbb{x})\end{array}\right] \ .\end{aligned}$$

Argument of knowledge for pairing products

Definition 7. *Define the relation $\mathcal{R}^n_{\text{PProd}}$ as the set of tuples $(\mathsf{pp}, \mathbb{x}, \mathbb{w})$ such that*

- $\mathsf{pp} = ((p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, P, \tilde{P}), (\vec{\Gamma}, \vec{\tilde{\Gamma}}))$ *where* $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, P, \tilde{P}) \leftarrow \mathsf{BG.Gen}(1^\lambda)$, $\vec{\Gamma} \in \mathbb{G}_1^n$ *and* $\vec{\tilde{\Gamma}} \in \mathbb{G}_2^n$*;*
- $\mathbb{x} = (\mathbf{A}, \mathbf{B}, \mathbf{C}) \in \mathbb{G}_T^3$*; and*
- $\mathbb{w} = (\vec{\Omega}, \vec{\tilde{\Omega}})$ *where* $\vec{\Omega} \in \mathbb{G}_1^n$, $\vec{\tilde{\Omega}} \in \mathbb{G}_2^n$*;*

satisfying $\mathbf{A} = e(\vec{\Omega}, \vec{\tilde{\Gamma}})$, $\mathbf{B} = e(\vec{\Gamma}, \vec{\tilde{\Omega}})$ *and* $\mathbf{C} = e(\vec{\Omega}, \vec{\tilde{\Omega}})$.

Theorem 1 ([20]). *Assuming that SXDH holds for* $\mathsf{BG.Gen}$*, then there is a preprocessing argument of knowledge* $(\mathsf{G}_{\mathrm{PProd}}, \mathsf{P}_{\mathrm{PProd}}, \mathsf{V}_{\mathrm{PProd}})$ *for* $\mathcal{R}^n_{\mathrm{PProd}}$*, for every* $n \in \mathbb{N}$*, with the following performance parameters:*

- *prover time dominated by* $O(n)$ *pairing operations and* $\mathbb{G}_1$*,* $\mathbb{G}_2$ *and* $\mathbb{G}_T$ *operations;*
- *offline verifier time dominated by* $O(n)$ *pairing operations and* $\mathbb{G}_T$ *operations in a one-time preprocessing phase; and*
- *online verifier time dominated by* $O(\log n)$ *pairing operations and* $\mathbb{G}_T$ *operations.*

3.2 Signatures of Knowledge

We describe here signatures of knowledge following the definitions in [15]. In a nutshell, a signature of knowledge generalizes the concept of public key signatures to NP statements. I.e., such a signature proves that "a person holding a witness $\mathbb{w}$ to a statement $\mathbb{x}$ has signed a message m".

Definition 8. *A signature of knowledge (SoK) is a tuple of three algorithms* $(\mathsf{G}, \mathsf{S}, \mathsf{V})$ *with the following syntax.*

- $\mathsf{G}(1^\lambda) \to \mathsf{pp}$*. The generator* G *is a p.p.t. algorithm which takes the security parameter* λ *as input and produces public parameters* pp *as output.*
- *The signer* S *is a p.p.t. algorithm that takes* pp*,* $\mathbb{x}$*,* $\mathbb{w}$ *and a message* m *as inputs and produces a signature* σ*.*
- *The verifier* V *is a p.p.t. that takes as input* pp*,* $\mathbb{x}$*, message* m *and signature* σ*, and outputs a bit* b*. If* $b = 1$*, we say that* σ *is a valid signature on message* m *relative to* pp *and* $\mathbb{x}$*.*

We say that $(\mathsf{G}, \mathsf{S}, \mathsf{V})$ is a signature of knowledge for a relation $\mathcal{R}$ if it satisfies the following properties.

- **Completeness.** For all $\lambda, n \in \mathbb{N}$, $m \in \{0,1\}^*$ and all adversaries $\mathcal{A}$

$$\Pr\left[\begin{array}{c}(\mathsf{pp}, \mathbb{x}, \mathbb{w}) \in \mathcal{R} \wedge \\ \sigma \leftarrow \mathsf{S}(\mathsf{pp}, \mathbb{x}, \mathbb{w}, m) \wedge \\ \mathsf{V}(\mathsf{pp}, \mathbb{x}, m, \sigma) = 1\end{array} \middle| \begin{array}{c}\mathsf{pp} \leftarrow \mathsf{G}(1^\lambda, n) \\ (\mathbb{x}, \mathbb{w}) \leftarrow \mathcal{A}(\mathsf{pp})\end{array}\right] = 1 \ .$$

- **Simulatability.** There exists a polynomial-time simulator Sim that runs two algorithms
  - $\mathsf{SimG}(1^\lambda) \to (\mathsf{pp}, \tau)$: The generator SimG is a p.p.t. algorithm which takes the security parameter λ as input and produces public parameters pp and trapdoor τ as output.
  - SimS is a p.p.t. algorithm that takes pp, trapdoor τ, $\mathbb{x}$ and a message m as inputs and produces a simulated signature σ.

  such that Sim receives values $(\mathbb{x}, \mathbb{w}, m)$ as inputs, checks whether $\mathbb{w}$ is valid and outputs $\mathsf{SimS}(\mathsf{pp}, \tau, \mathbb{x}, m)$, and for all p.p.t. adversaries $\mathcal{A}$ with oracle access to simulator Sim and SoK signer S

$$\Pr\left[\, 1 \leftarrow \mathcal{A}^{\mathsf{Sim}}(\mathsf{pp}) \middle| (\mathsf{pp}, \tau) \leftarrow \mathsf{SimG}(1^\lambda, n) \,\right] \approx \Pr\left[\, 1 \leftarrow \mathcal{A}^{\mathsf{S}}(\mathsf{pp}) \middle| \mathsf{pp} \leftarrow \mathsf{G}(1^\lambda, n) \,\right] \ .$$

– **Simulation Extractability.** These exists a polynomial time extractor $\mathsf{Extractor}$ such that for all p.p.t adversaries $\mathcal{A}$

$$\Pr\left[\begin{array}{c|c} (\mathsf{pp},\mathbb{x},\mathbb{w}) \in \mathcal{R} \quad \vee & (\mathsf{pp},\tau) \leftarrow \mathsf{SimG}(1^\lambda) \\ (\mathbb{x},\mathbb{w},m) \in \mathsf{Queries} \quad \vee & (\mathbb{x},m,\sigma) \leftarrow \mathcal{A}^{\mathsf{Sim}}(\mathsf{pp}) \\ \mathsf{V}(\mathsf{pp},\mathbb{x},\mathbb{w},m) = 0 & \mathbb{w} \leftarrow \mathsf{Extractor}(\mathsf{pp},\tau,\mathbb{x},m,\sigma) \end{array}\right] \approx 1 \ .$$

where $\mathsf{Queries}$ denotes all queries $(\mathbb{x},\mathbb{w},m)$ that Sim receives from $\mathcal{A}$.

4 Prefix-Linkable Ring Signature Schemes

We now give a definition of prefix-linkable ring signatures. We follow [5] and [6], and modify their definitions by splitting into message and prefix, and link only with respect to the prefix (but not the message).

Definition 9 (Prefix-linkable ring signature scheme). *A* prefix-linkable ring signature *(PLRS) scheme is a tuple of algorithms* $\mathsf{RS} = (\mathsf{Gen},\mathsf{KeyGen},\mathsf{Sign},\mathsf{Verify},\mathsf{Link})$ *with message space* $\mathcal{M}$ *and prefix space* $\mathcal{P}$ *where*

– $\mathsf{pp} \leftarrow \mathsf{RS.Gen}(1^\lambda)$ *produces public parameters* pp*, which we assume to be available to all the algorithms below.*
– $(sk,pk) \leftarrow \mathsf{RS.KeyGen}(\mathsf{pp})$ *produces a key pair.*
– $\sigma \leftarrow \mathsf{RS.Sign}(\vec{pk},sk,m,\mathsf{prfx})$*, where* m *is a message,* prfx *is a prefix, and* $\vec{pk}$ *is a vector of public keys produced by* $\mathsf{RS.KeyGen}$ *that includes the public key* pk *corresponding to secret key* sk*.*
– $0/1 \leftarrow \mathsf{RS.Verify}(\vec{pk},m,\mathsf{prfx},\sigma)$
– $0/1 \leftarrow \mathsf{RS.Link}(\vec{pk},\sigma,m,\sigma',m',\mathsf{prfx})$.

We require that the scheme is correct*; that is, for any* $m,m' \in \mathcal{M}$*, any* $\mathsf{prfx} \in \mathcal{P}$*, any* $(sk,pk),(sk',pk')$ *produced by* $\mathsf{RS.Gen}(1^\lambda)$ *with* $pk \neq pk'$ *and any* $\vec{pk}$ *of public keys produced by* $\mathsf{RS.KeyGen}$ *that includes* pk *and* pk'*,*

– $\mathsf{RS.Verify}(\vec{pk},m,\mathsf{prfx},\mathsf{RS.Sign}(\vec{pk},sk,m,\mathsf{prfx})) = 1$.
– $\mathsf{RS.Link}(\vec{pk},\mathsf{RS.Sign}(\vec{pk},sk,m,\mathsf{prfx}),m,\mathsf{RS.Sign}(\vec{pk},sk,m',\mathsf{prfx}),m',\mathsf{prfx}) = 1$.
– $\mathsf{RS.Link}(\vec{pk},m,\mathsf{RS.Sign}(\vec{pk},sk,m,\mathsf{prfx}),m',\mathsf{RS.Sign}(\vec{pk},sk',m',\mathsf{prfx}),\mathsf{prfx}) = 0$ *except with negligible probability.*

We now define various security properties. First, unforgeability demands that an adversary cannot produce a valid signature for any message-prefix pair, for a ring for which the adversary does not know any secret key, even when equipped with a signing oracle for that ring. Forgeries need to verify w.r.t the ring generated by the experiment. All our notions below are in the "honest ring with insider corruption" setting [9], i.e., the games all sample a ring

$$\begin{gathered} \mathsf{pp} \leftarrow \mathsf{RS.Gen}(1^\lambda), \\ (sk_i,pk_i) \leftarrow \mathsf{RS.KeyGen}(\mathsf{pp}),\ i \in [n], \end{gathered}$$

and set $\vec{pk} := (pk_1,\ldots,pk_n)$.

Definition 10 (Corruption oracle). *Given a well-formed public key pk produced by* $\mathsf{RS.KeyGen}$*, the* corruption oracle CO *returns the corresponding sk.*

Definition 11 (Signing oracle). *Given a well-formed set of public keys* $\vec{pk}$*, on input* $pk \in \vec{pk}$*, a message m, and a prefix* prfx*, the* signing oracle $\mathsf{SO}_{\vec{pk}}$ *returns a signature* σ *whose distribution is computationally indistinguishable from the output of* $\mathsf{RS.Sign}(\mathsf{pp}, \vec{pk}, sk, m, \mathsf{prfx})$*, where sk corresponds to pk.*

Definition 12 (Unforgeability). *A PLRS is* unforgeable *if for all efficient adversaries* $\mathcal{A}^{\mathsf{SO}_{\vec{pk}}}(\mathsf{pp}, \vec{pk})$ *outputting* $(m, \mathsf{prfx}, \sigma)$*, the probability that* σ *was not produced by* $\mathsf{SO}_{\vec{pk}}$ *on any input* (m, prfx) *and* $\mathsf{RS.Verify}(\mathsf{pp}, \vec{pk}, m, \mathsf{prfx}, \sigma) = 1$ *is negligible.*

We next define anonymity, which demands that an adversary cannot tell which of a ring's secret keys was used to produce a signature. A notable difference to anonymity of ring signature schemes with "standard", i.e., full-message linkability is that we can grant the adversary access to a signing oracle even for the two challenge public keys. In standard linkable ring signature schemes, such an oracle would let the adversary win trivially, by producing a signature of another message under these public keys, and test to which one the challenge signature links. In case of prefix linkable schemes, such trivial wins are only possible if the adversary can use the same prefix in the signing oracle, and hence we can formulate a strong anonymity game by allowing access to a signing oracle wrt. prefixes that differ from the challenge prefix.

Definition 13 (Anonymity). *A PLRS is* anonymous *if for all efficient stateful adversaries* $\mathcal{A}$*, the probability of winning the following game is negligibly close to* $1/2$*.*

- $(m, \mathsf{prfx}, pk_1, pk_2) \leftarrow \mathcal{A}^{\mathsf{CO},\mathsf{SO}_{\vec{pk}}}(\mathsf{pp}, \vec{pk})$
- $b \leftarrow \{0,1\}$*;*
- $\sigma \leftarrow \mathsf{RS.Sign}(\mathsf{pp}, \vec{pk}, sk_b, m, \mathsf{prfx})$*;*
- $b' \leftarrow \mathcal{A}(\sigma)$.

$\mathcal{A}$ *wins if all of the following hold:*

- $b = b'$*;*
- $pk_1, pk_2 \in \vec{pk}$ *and* $pk_1 \neq pk_2$*;*
- pk_1, pk_2 *were never queried to* CO*;*
- (pk_1, prfx) *and* (pk_2, prfx) *were never used in any* $\mathsf{SO}_{\vec{pk}}$ *query.*

We next demand that it must be hard to bypass the linking property of the signature scheme. For standard linkable ring signatures, i.e., ones that link with respect to any message, this property simply demands that it is hard to create two signatures from the same secret key that do not link with each other. Here, we additionally require the adversary to create such non-linking signatures w.r.t the same prefix, as otherwise the game would be trivial to win. As in [6], the adversary can use all secret keys of the ring to achieve this goal. Our definition differs from [6] in that we fix the ring $\vec{pk}$, i.e., we do not allow the adversary to introduce adversarially-generated public keys into the ring, or drop some of the public keys from it.

Definition 14 (Prefix linkability). *A PLRS is* prefix-linkable *if for all efficient adversaries* $\mathcal{A}$*, the probability of winning the following game is negligible.*

- $\mathsf{pp} \leftarrow \mathsf{RS.Gen}(1^\lambda)$;
- $(sk_i, pk_i) \leftarrow \mathsf{RS.KeyGen}(\mathsf{pp})$, $i \in [n]$;
- $\vec{pk} := (pk_1, \ldots, pk_n)$;
- $(m_i, \mathsf{prfx}, \sigma_i)_{i \in [n+1]} \leftarrow \mathcal{A}^{\mathsf{CO},\mathsf{SO}_{\vec{pk}}}(\mathsf{pp}, \vec{pk})$

$\mathcal{A}$ wins if all of the following hold:
- $\mathsf{RS.Verify}(\mathsf{pp}, \vec{pk}, m_i, \mathsf{prfx}, \sigma_i) = 1$ *for all* $i \in [n+1]$;
- $\mathsf{RS.Link}(\mathsf{pp}, \vec{pk}, \sigma_i, m_i, \sigma_j, m_j, \mathsf{prfx}) = 0$ *for all* $i, j \in [n+1], i \neq j$.

Finally, we demand that it must be hard to create a signature that links to one of an honest signer. We follow the 2-staged definition of [6] and grant the adversary access to all secret keys of the ring only after producing the "slandering" signature σ'.

Definition 15 (Non-slanderability). *A PLRS is* non-slanderable *if for all efficient adversaries $\mathcal{A}$, the probability of winning the following game is negligible.*

- $(m', \mathsf{prfx}', \sigma') \leftarrow \mathcal{A}^{\mathsf{SO}_{\vec{pk}}}(\mathsf{pp}, \vec{pk})$
- $(m, \sigma) \leftarrow \mathcal{A}^{\mathsf{CO},\mathsf{SO}_{\vec{pk}}}$

$\mathcal{A}$ wins if all of the following hold:
- $\mathsf{RS.Verify}(\mathsf{pp}, \vec{pk}, m, \mathsf{prfx}', \sigma) = 1$
- $\mathsf{RS.Verify}(\mathsf{pp}, \vec{pk}, m', \mathsf{prfx}', \sigma') = 1$;
- $\mathsf{RS.Link}(\vec{pk}, \sigma, m, \sigma;, m', \mathsf{prfx}') = 1$;
- σ' *was not received from* $\mathsf{SO}_{\vec{pk}}$.

Discussion. In case of prfx being the empty string, the definitions in this section define a linkable RS with message space $\mathcal{M}$. Dropping prfx from them leads to, e.g., the anonymity game (Definition 13) to forbid usage of signing oracles w.r.t pk_1, pk_2 completely. The definitions then become equivalent to, e.g., the definition of Au et al. for linkable RS [5]. In case of m being the empty string, the definitions in this section define a "same-message" linkable RS with message space $\mathcal{P}$, where linking of signatures is only possible if a signer signs the same message more than once. Hence, our notion of prefix-linkable ring signatures is a generalization of linkable ring signatures that allows to fine-tune linkability.

5 Our Construction

We are now ready to present our new ring signature scheme which allows to link signatures via message prefixes, and where verification time for re-used rings can be compressed to logarithmic at the cost of one-time linear-time preprocessing of rings.

5.1 Tag Proof

We start by giving a "tag proof" scheme, which allows to prove knowledge of a secret key that was used to create both a linking tag *and* a commitment. For our purposes, it will be convenient to link this proof with a message m, making it a signature of knowledge.

Definition 16. *The relation $\mathcal{R}_{\text{Tag}}$ consists of tuples*

$$(\mathsf{pp}_{\text{Tag}}, \mathbb{x}, \mathbb{w}) = \big((p, \mathbb{G}, P, Q, H, H'), (\mathsf{prfx}, \mathsf{tag}, \mathsf{com}), (sk, r)\big)$$

such that $\mathbb{G}$ is a group of prime order p with generators P and Q, $H\colon \{0,1\}^* \to \mathbb{Z}_p$ and $H'\colon \{0,1\}^* \to \mathbb{G}$ are two hash functions, $\mathrm{com}, \mathrm{tag} \in \mathbb{G}$, $\mathsf{prfx} \in \{0,1\}^*$, and $sk, r \in \mathbb{Z}_p$, satisfy $\mathrm{com} = P^{sk}Q^r$ and $\mathrm{tag} = H'(\mathsf{prfx})^{sk}$.

Construction 1 *Let $\lambda \in \mathbb{N}$ denote a security parameter and let $\mathsf{G.Gen}$ be a group generator. Then our tag proof scheme is a tuple of algorithms $(\mathsf{G}_{\mathrm{Tag}}, \mathsf{S}_{\mathrm{Tag}}, \mathsf{V}_{\mathrm{Tag}})$ defined as follows.*

$\underline{\mathsf{pp}_{\mathrm{Tag}} \leftarrow \mathsf{G}_{\mathrm{Tag}}(1^\lambda)\text{:}}$
$(p, \mathbb{G}, P) \leftarrow \mathsf{G.Gen}(1^\lambda)$
$Q \leftarrow_{\$} \mathbb{G}$
define hash functions
$H\colon \{0,1\}^* \to \mathbb{Z}_p$ *and*
$H'\colon \{0,1\}^* \to \mathbb{G}$
output $\mathsf{pp}_{\mathrm{Tag}} = (p, \mathbb{G}, P, Q, H, H')$.

$\underline{\sigma_{\mathrm{Tag}} \leftarrow \mathsf{S}_{\mathrm{Tag}}(\mathsf{pp}_{\mathrm{Tag}}, \mathbb{x}, \mathbb{w}, m)\text{:}}$
parse $(\mathsf{prfx}, \mathrm{tag}, \mathrm{com}) := \mathbb{x}$ *and* $(sk, r) := \mathbb{w}$
$a, b \leftarrow_{\$} \mathbb{Z}_p$
$A := H'(\mathsf{prfx})^a \in \mathbb{G}$, $B := P^a Q^b \in \mathbb{G}$
$c = H(\mathsf{prfx}, \mathrm{com}, \mathrm{tag}, A, B, m) \in \mathbb{Z}_p$
$\bar{a} := a + c \cdot sk \in \mathbb{Z}_p$, $\bar{b} := b + c \cdot r \in \mathbb{Z}_p$
output $\sigma_{\mathrm{Tag}} := (A, B, \bar{a}, \bar{b}) \in \mathbb{G}^2 \times \mathbb{Z}_p^2$.

$\underline{b \leftarrow \mathsf{V}_{\mathrm{Tag}}(\mathsf{pp}_{\mathrm{Tag}}, \mathbb{x}, \sigma_{\mathrm{Tag}}, m)\text{:}}$
parse $(\mathsf{prfx}, \mathrm{tag}, \mathrm{com}) := \mathbb{x}$, $(A, B, \bar{a}, \bar{b}) \in \mathbb{G}^2 \times \mathbb{Z}_p^2 := \sigma_{\mathrm{Tag}}$
$c = H(\mathsf{prfx}, \mathrm{com}, \mathrm{tag}, A, B, m) \in \mathbb{Z}_p$
if $H'(\mathsf{prfx})^{\bar{a}} = A \cdot \mathrm{tag}^c$ *and* $P^{\bar{a}} Q^{\bar{b}} = B \cdot \mathrm{com}^c$ *then output* 1, *else output* 0

Theorem 2. *Tag proof (Construction 1) is a signature of knowledge for $\mathcal{R}_{\mathrm{Tag}}$.*

Proof (Sketch). Tag proof is a non-interactive zero-knowledge proof of knowledge for $\mathcal{R}_{\mathrm{Tag}}$ made up of two simple zero-knowledge proofs:

- one Schnorr proof-of-knowledge of the discrete logarithm sk of A to base $H'(\mathsf{prfx})$ (see e.g. [24, Fig. 4.3]); and
- one Okamoto proof of knowledge of exponents sk, r such that $\mathrm{com} = P^{sk}Q^r$ for $P, Q, \mathrm{com} \in \mathbb{G}$ (see e.g. [24, Fig. 4.5]).

The proofs are combined using an EQ transformation in which parts of the Schnorr proof (such as $a, \bar{a}$) are reused in the Okamoto proof (see [24, Section 5.2.2]), before making the resulting proof non-interactive using the Fiat-Shamir heuristic (i.e., using hash H to generate the challenge). We now argue that if H takes also message m as input, this results in a signature of knowledge for $(\mathbb{x}, \mathbb{w}) = ((\mathsf{prfx}, \mathrm{tag}, \mathrm{com}), (sk, r))$. First, given that Schnorr and Okamoto proofs are simulatable [23], the corresponding signature is also simulatable in the random oracle model [8]. Second, applying the forking lemma to H, one can successfully extract a valid witness from a valid forgery [8].

5.2 DualDory

In this section, we give our main ring signature construction.

Construction 2 *Let $\lambda \in \mathbb{N}$ denote a security parameter and $n \in \mathbb{N}$ an upper bound on the ring size. Let $\mathsf{BG.Gen}$ a bilinear group generator, $(\mathsf{G}_{\mathrm{Tag}}, \mathsf{S}_{\mathrm{Tag}}, \mathsf{V}_{\mathrm{Tag}})$ the tag proof scheme of Construction 1, $\mathcal{R}^n_{\mathrm{PProd}}$ the relation w.r.t $\mathsf{BG.Gen}$ as in Definition 7, and $(\mathsf{G}_{\mathrm{PProd}}, \mathsf{P}_{\mathrm{PProd}}, \mathsf{V}_{\mathrm{PProd}})$ a preprocessing argument of knowledge for $\mathcal{R}^n_{\mathrm{PProd}}$.*

Then DualDory is defined by the set of procedures in Figure 1.

Setup $\mathsf{RS.Gen}(1^\lambda, n)$:
$\mathsf{BGpp} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, P, \tilde{P}) \leftarrow \mathsf{BG.Gen}(1^\lambda)$
$Q \leftarrow \mathbb{G}_1$, $\vec{\Gamma} \leftarrow \mathbb{G}_1^n$, $\vec{\tilde{\Gamma}} \leftarrow \mathbb{G}_2^n$
define hash functions
$H' \colon \{0,1\}^* \to \mathbb{G}_1$ and $H \colon \{0,1\}^* \to \mathbb{Z}_p$
$\mathsf{pp}_{\mathrm{Tag}} := (\mathbb{G}_1, p, P, Q, H, H')$, $\mathsf{pp}_{\mathrm{PProd}} := (\mathsf{BGpp}, (\vec{\Gamma}, \vec{\tilde{\Gamma}}))$
output $\mathsf{pp} := (\mathsf{pp}_{\mathrm{PProd}}, \mathsf{pp}_{\mathrm{Tag}})$

Key Generation $\mathsf{RS.KeyGen}(\mathsf{pp})$:
$sk \leftarrow_\$ \mathbb{Z}_p$
output $(sk, pk := P^{sk})$

Pre-processing per ring:
$\mathbf{A}_0 \leftarrow e(\vec{pk}, \vec{\tilde{\Gamma}})$, $\mathbf{D} \leftarrow e(\vec{P}, \vec{\tilde{\Gamma}})$
$\tilde{\Gamma} \leftarrow \prod_{i=1}^n \tilde{\Gamma}_i$
$\mathsf{aux} := (\mathbf{A}_0, \mathbf{D}, \tilde{\Gamma})$

Signing $\mathsf{RS.Sign}(\mathsf{pp}, \mathsf{aux}, \vec{pk}, sk_j, m, \mathsf{prfx})$:
$\vec{pk} := (pk_i)_{i=1}^n \in \mathbb{G}_1^n$, parse m in $\{0,1\}^*$
$r \leftarrow \mathbb{Z}_p$, $\mathrm{com} \leftarrow P^{sk_j} Q^r$
$\vec{pk}' := (pk'_1, ..., pk'_n)$
with $pk'_i \leftarrow \mathrm{com}/pk_i$ (i.e., $pk'_j = Q^r$)

(1) Commitment to $\vec{pk}'$:
$\mathbf{A} \leftarrow e(\mathrm{com}, \tilde{\Gamma})/\mathbf{A}_0$

(2) DualRing, applied to $\exists\ j \in [n], r \in \mathbb{Z}_p : pk'_j = Q^r$:
$x, c_1, ..., c_{j-1}, c_{j+1}, ..., c_n \leftarrow \mathbb{Z}_p$
$X = Q^x \prod_{i \in [n] \setminus \{j\}} {pk'_i}^{-c_i} \in \mathbb{G}_1$
$c_j = H(\mathbf{A}, X) - \sum_{i \in [n] \setminus j} c_i \in \mathbb{Z}_p$
$y \leftarrow x + c_j r \in \mathbb{Z}_p$

(3) Arg. of knowledge of pairing products:
$\vec{c} := (c_1, \ldots, c_n) \in \mathbb{Z}_p^n$
$\tilde{P}^{\vec{c}} := (\tilde{P}^{c_1}, \ldots, \tilde{P}^{c_n}) \in \mathbb{G}_2^n$
$\vec{P} := (P, \ldots, P) \in \mathbb{G}_1^n$
$\mathbf{B} \leftarrow e(\vec{\Gamma}, \tilde{P}^{\vec{c}})$, $\mathbf{C} \leftarrow e(Q^y/X, \tilde{P})$,
$\mathbf{E} \leftarrow e(P^{H(\mathbf{A},X)}, \tilde{P})$
$\pi_1 \leftarrow \mathsf{P}_{\mathrm{PProd}}(\mathsf{pp}_{\mathrm{PProd}}, (\mathbf{A}, \mathbf{B}, \mathbf{C}), (\vec{pk}', \tilde{P}^{\vec{c}}))$
$\pi_2 \leftarrow \mathsf{P}_{\mathrm{PProd}}(\mathsf{pp}_{\mathrm{PProd}}, (\mathbf{D}, \mathbf{B}, \mathbf{E}), (\vec{P}, \tilde{P}^{\vec{c}}))$

(4) Signature of knowledge/tag proof:
$\mathrm{tag} = H'(\mathsf{prfx})^{sk}$
$\mathbb{x} := (\mathsf{prfx}, \mathrm{tag}, \mathrm{com})$
$\sigma_{\mathrm{Tag}} \leftarrow \mathsf{S}_{\mathrm{Tag}}(\mathsf{pp}_{\mathrm{Tag}}, \mathbb{x}, (sk, r), m || \pi_1 || \pi_2)$
output $\sigma := (X, y, \mathbf{B}, \pi_1, \pi_2, \sigma_{\mathrm{Tag}}, \mathrm{tag}, \mathrm{com})$

Verification $\mathsf{RS.Verify}(\mathsf{pp}, \mathsf{aux}, \vec{pk}, m, \mathsf{prfx}, \sigma)$:
$\mathsf{aux} := (\mathbf{A}_0, \mathbf{D}, \tilde{\Gamma})$, parse m in $\{0,1\}^*$
$(X, y, \mathbf{B}, \pi_1, \pi_2, \sigma_{\mathrm{Tag}}, \mathrm{tag}, \mathrm{com}) := \sigma$
$\mathbf{A} \leftarrow e(\mathrm{com}, \tilde{\Gamma})/\mathbf{A}_0$, $\mathbf{C} \leftarrow e(Q^y/X, \tilde{P})$, $\mathbf{E} \leftarrow e(P^{H(\mathbf{A},X)}, \tilde{P})$
run $\mathsf{V}_{\mathrm{PProd}}(\mathsf{pp}_{\mathrm{PProd}}, (\mathbf{A}, \mathbf{B}, \mathbf{C}), \pi_1)$
run $\mathsf{V}_{\mathrm{PProd}}(\mathsf{pp}_{\mathrm{PProd}}, (\mathbf{D}, \mathbf{B}, \mathbf{E}), \pi_2)$
output $\mathsf{V}_{\mathrm{Tag}}(\mathsf{pp}_{\mathrm{Tag}}, (\mathsf{prfx}, \mathrm{tag}, \mathrm{com}), \sigma_{\mathrm{Tag}}, m || \pi_1 || \pi_2)$

Linking $\mathsf{RS.Link}(\mathsf{pp}, \vec{pk}, \sigma, m, \sigma', m', \mathsf{prfx})$:
$(X, y, \mathbf{B}, \pi_1, \pi_2, \sigma_{\mathrm{Tag}}, \mathrm{tag}, \mathrm{com}) := \sigma$
$(X', y', \mathbf{B}', \pi'_1, \pi'_2, \sigma'_{\mathrm{Tag}}, \mathrm{tag}', \mathrm{com}') := \sigma'$
output 1 if $\mathrm{tag} = \mathrm{tag}'$ and 0 otherwise.

Fig. 1: The DualDory linkable ring signature scheme. Note that the parameter generation $\mathsf{G}_{\mathrm{Tag}}$ is not used, and instead tag proof is run on the pairing source group $\mathbb{G}_1$ produced by $\mathsf{BG.Gen}$.

Theorem 3. *DualDory (Construction 2) is an unforgeable, anonymous, prefix-linkable and non-slanderable PLRS scheme with the following complexity parameters:*

- *public parameter size $O(n)$ elements of $\mathbb{G}_1$ and $\mathbb{G}_2$;*
- *signature size $O(\log n)$ $\mathbb{G}_T$-elements, $O(1)$ $\mathbb{G}_1$-elements and $O(1)$ $\mathbb{Z}_p$-elements;*

- *signing complexity dominated by $O(n)$ pairing operations;*
- *online verification complexity dominated by $O(\log n)$ pairing operations;*
- *offline verification complexity dominated by $O(n)$ pairing operations;*

Proof. The proof follows from a straightforward inspection of the complexity of the procedures, and from Theorems 4-8.

Theorem 4 (Correctness). *DualDory satisfies correctness (Definition 9).*

Proof. We show each of the correctness requirements separately.

$\mathsf{RS.Verify}(\vec{pk}, m, \mathsf{prfx}, \mathsf{RS.Sign}(\vec{pk}, sk, m, \mathsf{prfx})) = 1$*:* Let $\vec{pk}' = (\mathrm{com}/pk_1, \ldots, \mathrm{com}/pk_n)$ and recall that $\mathbf{A} = e(\mathrm{com}, \tilde{\Gamma})/\mathbf{A}_0 = e(\vec{pk}', \vec{\tilde{\Gamma}})$, $\mathbf{B} = e(\vec{\Gamma}, \tilde{P}^{\vec{c}})$, $\mathbf{C} = e(Q^y/X, \tilde{P})$, $\mathbf{D} = e(\vec{P}, \vec{\tilde{\Gamma}})$ and $\mathbf{E} = e(P^{H(\mathbf{A},X)}, \tilde{P})$. Parse the last input element $\mathsf{RS.Sign}(sk, \vec{pk}, m, \mathsf{prfx})$ as $(X, y, \mathbf{B}, \pi_1, \pi_2, \sigma_{\mathrm{Tag}}, \mathrm{tag}, \mathrm{com})$. Following DualRing: $Q^y/X = \prod_{i=1}^n pk'^{c_i}_i$ and $\sum_{i=1}^n c_i = H(\mathbf{A}, X)$. Therefore, $\mathbf{C} = e(\vec{pk}', \tilde{P}^{\vec{c}})$ and $\mathbf{E} = e(\vec{P}, \tilde{P}^{\vec{c}})$.

$\mathsf{V}_{\mathrm{PProd}}(\mathsf{pp}_{\mathrm{PProd}}, (\mathbf{A}, \mathbf{B}, \mathbf{C}), \pi_1) = 1$ because $\pi_1 \leftarrow \mathsf{P}_{\mathrm{PProd}}(\mathsf{pp}_{\mathrm{PProd}}, (\mathbf{A}, \mathbf{B}, \mathbf{C}), (\vec{pk}', \tilde{P}^{\vec{c}}))$. Similarly, $\pi_2 \leftarrow \mathsf{P}_{\mathrm{PProd}}(\mathsf{pp}_{\mathrm{PProd}}, (\mathbf{D}, \mathbf{B}, \mathbf{E}), (\vec{P}, \tilde{P}^{\vec{c}}))$ and $\mathsf{V}_{\mathrm{PProd}}(\mathsf{pp}_{\mathrm{PProd}}, (\mathbf{D}, \mathbf{B}, \mathbf{E}), \pi_2) = 1$. Finally, $\sigma_{\mathrm{Tag}} \leftarrow \mathsf{S}_{\mathrm{Tag}}(\mathsf{pp}_{\mathrm{Tag}}, (\mathsf{prfx}, \mathrm{tag}, \mathrm{com}), (sk, r), m||\pi_1||\pi_2)$, which means that $\mathsf{V}_{\mathrm{Tag}}(\mathsf{pp}_{\mathrm{Tag}}, (\mathsf{prfx}, \mathrm{tag}, \mathrm{com}), \sigma_{\mathrm{Tag}}, m||\pi_1||\pi_2) = 1$.

$\mathsf{RS.Link}(\vec{pk}, \mathsf{RS.Sign}(\vec{pk}, sk, m, \mathsf{prfx}), m, \mathsf{RS.Sign}(\vec{pk}, sk, m', \mathsf{prfx}), m', \mathsf{prfx}) = 1$*:* Two signatures generated using the same prefix prfx and secret key sk will always yield the same tag $H'(\mathsf{prfx})^{sk}$, leading RS.Link to output 1.

$\mathsf{RS.Link}(\vec{pk}, m, \mathsf{RS.Sign}(\vec{pk}, sk, m, \mathsf{prfx}), m', \mathsf{RS.Sign}(\vec{pk}, sk', m', \mathsf{prfx}), \mathsf{prfx}) = 0$*:* Two signatures generated using the same prefix prfx and two different secret keys sk and sk' will always yield two distinct tags $H'(\mathsf{prfx})^{sk}$ and $H'(\mathsf{prfx})^{sk'}$, leading RS.Link to output 0.

Due to space limitations, Theorems **??**-8 and their proofs are deferred to **??**.

6 Evaluation

We implement our linkable ring signature in $\sim$ 1,500 lines of Go, in which we instantiate the argument of knowledge of bilinear pairing products with Dory [20]. Our implementation is publicly available [1] and open source. We use the BN254 [7] elliptic curve implementation of gnark-crypto [12]. We evaluate the performance by running 100 independent trials for each measured operation on an Ubuntu 22.04 AWS c5a.xlarge machine equipped with 4 2.8Ghz CPUs with 8GB RAM.

Fig 2 depicts the time it takes to produce and verify a DualDory signature and the cost of the offline preprocessing in relation to the size of the ring. We draw three conclusions from our empirical results. (1) As expected, the verification speed of our linkable ring signature is logarithmic in the size of the ring, and scales well even for large rings. (2) The time it takes to generate a ring signature is linear in the size of the ring. For a

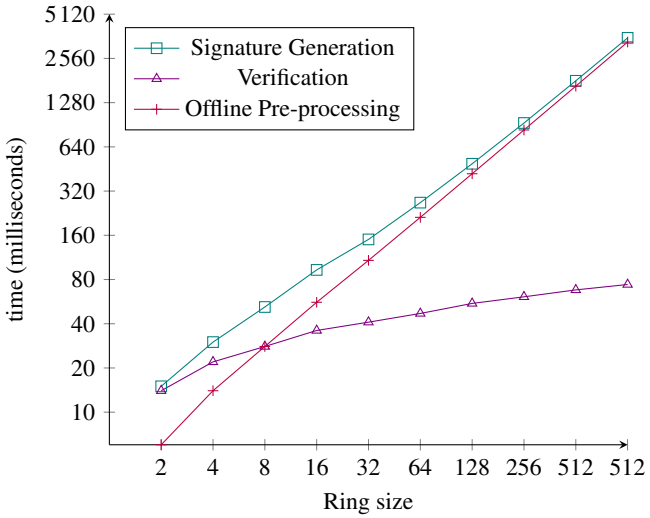

Fig. 2: Performance measurements of DualDory

ring of size 1024, signing takes 3.53s. Fortunately, most of the signature work, which corresponds to the prover computation of DualRing and the argument of knowledge of bilinear pairing products, can be precomputed by the signer before knowing either the message or the prefix. This leaves only the tag proof to be generated online (i.e., when the message and the prefix are known). Our benchmarks show that the time it takes to compute the tag proof is less than 1ms for a ring of size 1024. The cost of tag proof can be made constant by precomputing the hash of $\pi_1||\pi_2$ and including the result in the tag proof instead of $\pi_1||\pi_2$. This would slightly increase the cost of verification (one additional hash). (3) The offline preprocessing, which is performed once per ring, grows linearly with the size of the ring. Furthermore, for small-sized rings ≤ 8, the preprocessing is cheaper than the verification, which probably indicates that there is no tangible gain yet and that DualDory is ill-suited for rings ≤ 8. When the size of the ring increases ≥ 16, the preprocessing overtakes signature verification. In particular, for a ring of size 1024, verification takes 74ms, whereas preprocessing takes 3.31s. Although the offline preprocessing is not too expensive, it is desirable to amortize its cost over multiple verifications. Thus, we recommend DualDory for settings with static or incrementally-updatable rings.

7 Conclusion

We introduced DualDory, a prefix-linkable ring signature with logarithmic verification and transparent setup. We proved its security under SXDH assumption and benchmarked its performances. The benchmarks, as expected, show that the verifier performs logarithmic work at the expense of a *linear* offline pre-processing operation. Assuming *static* rings or ones that are updated *incrementally*, the cost of pre-processing can be amortized over an unlimited number of verifications. This makes DualDory a suitable candidate for threshold ring signatures or e-voting applications.

References

1. Dual-dory implementation. `https://github.com/yacovm/DualDory`.
2. Masayuki Abe, Georg Fuchsbauer, Jens Groth, Kristiyan Haralambiev, and Miyako Ohkubo. Structure-preserving signatures and commitments to group elements. In *CRYPTO*, pages 209–236, 2010.
3. Masayuki Abe, Miyako Ohkubo, and Koutarou Suzuki. 1-out-of-n signatures from a variety of keys. pages 131–140, 2004.
4. Giuseppe Ateniese, Jan Camenisch, Marc Joye, and Gene Tsudik. A practical and provably secure coalition-resistant group signature scheme. In Mihir Bellare, editor, *CRYPTO*, volume 1880 of *LNCS*, pages 255–270, 2000.
5. Man Ho Au, Sherman S. M. Chow, Willy Susilo, and Patrick P. Tsang. Short linkable ring signatures revisited. In *EuroPKI*, volume 4043 of *LNCS*, pages 101–115, 2006.
6. Michael Backes, Nico Döttling, Lucjan Hanzlik, Kamil Kluczniak, and Jonas Schneider. Ring signatures: Logarithmic-size, no setup—from standard assumptions. 11478:281–311, 2019.
7. Paulo S. L. M. Barreto and Michael Naehrig. Pairing-friendly elliptic curves of prime order. In Bart Preneel and Stafford Tavares, editors, *Selected Areas in Cryptography*, pages 319–331, Berlin, Heidelberg, 2006.
8. Mihir Bellare and Gregory Neven. New multi-signature schemes and a general forking lemma, 2005.
9. Adam Bender, Jonathan Katz, and Ruggero Morselli. Ring signatures: Stronger definitions, and constructions without random oracles. In Shai Halevi and Tal Rabin, editors, *TCC*, volume 3876 of *LNCS*, pages 60–79, 2006.
10. Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In Matthew K. Franklin, editor, *CRYPTO*, volume 3152 of *LNCS*, pages 41–55, 2004.
11. Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, Jens Groth, and Christophe Petit. Efficient zero-knowledge arguments for arithmetic circuits in the discrete log setting. EUROCRYPT, pages 327–357, 2016.
12. Gautam Botrel, Thomas Piellard, Youssef El Housni, Arya Tabaie, and Ivo Kubjas. Consensys/gnark-crypto: v0.6.1, February 2022.
13. Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell. Bulletproofs: Short proofs for confidential transactions and more. IEEE Security & Privacy, pages 315–334, 2018.
14. Nishanth Chandran, Jens Groth, and Amit Sahai. Ring signatures of sub-linear size without random oracles. In Lars Arge, Christian Cachin, Tomasz Jurdzinski, and Andrzej Tarlecki, editors, *ICALP*, volume 4596 of *LNCS*, pages 423–434, 2007.
15. Melissa Chase and Anna Lysyanskaya. On signatures of knowledge. In *Annual International Cryptology Conference*, pages 78–96. Springer, 2006.
16. David Chaum and Eugène van Heyst. Group signatures. In Donald W. Davies, editor, *EUROCRYPT*, volume 547 of *LNCS*, pages 257–265, 1991.
17. Yevgeniy Dodis, Aggelos Kiayias, Antonio Nicolosi, and Victor Shoup. Anonymous identification in ad hoc groups. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT*, volume 3027 of *LNCS*, pages 609–626, 2004.
18. Brandon Goodell, Sarang Noether, and RandomRun. Concise linkable ring signatures and forgery against adversarial keys. IACR Cryptol. ePrint Arch., 2019. `https://ia.cr/2019/654`.
19. Jens Groth and Markulf Kohlweiss. One-out-of-many proofs: Or how to leak a secret and spend a coin. In *EUROCRYPT*, volume 9057 of *LNCS*, pages 253–280, 2015.

20. Jonathan Lee. Dory: Efficient, transparent arguments for generalised inner products and polynomial commitments. In *TCC*, volume 13043 of *LNCS*, pages 1–34, 2021.
21. Joseph K. Liu, Victor K. Wei, and Duncan S. Wong. Linkable spontaneous anonymous group signature for ad hoc groups (extended abstract). In *ACISP*, volume 3108 of *LNCS*, pages 325–335, 2004.
22. Ronald L. Rivest, Adi Shamir, and Yael Tauman. How to leak a secret. In Colin Boyd, editor, *ASIACRYPT*, volume 2248 of *LNCS*, pages 552–565, 2001.
23. Lior Rotem and Gil Segev. Tighter security for schnorr identification and signatures: A high-moment forking lemma for sigma protocols. In *CRYPTO*, pages 222–250, 2021.
24. Berry Schoenmakers. Lecture notes cryptographic protocols. `https://www.win.tue.nl/~berry/2WC13/LectureNotes.pdf`, 2021.
25. Patrick P. Tsang and Victor K. Wei. Short linkable ring signatures for e-voting, e-cash and attestation. In *ISPEC*, volume 3439 of *LNCS*, pages 48–60, 2005.
26. Tsz Hon Yuen, Muhammed F. Esgin, Joseph K. Liu, Man Ho Au, and Zhimin Ding. Dualring: Generic construction of ring signatures with efficient instantiations. In *CRYPTO*, LNCS, pages 251–281, 2021.

A DualRing Construction

Definition 17. *The relation* $\mathcal{R}_{\mathsf{DR}}$ *consists of tuples*

$$(pp_{\mathsf{DR}}, \mathbb{x}, \mathbb{w}) = \left((p, \mathbb{G}, P), \vec{pk}, (sk_j, j)\right)$$

such that $\mathbb{G}$ *is a group of prime order* p *with generators* P, $\vec{pk} \in \mathbb{G}^n$, $sk_j \in \mathbb{Z}_p$ *and* $pk_j = P^{sk_j}$.

Theorem 5. *Construction 3 is a secure ring signature scheme, with the following complexity parameters:*

- *signature size* 1 *element of* $\mathbb{G}$ *and* $n+1$ *elements of* $\mathbb{Z}_p$*;*
- *signing complexity* 1 *multi-exponentiation of length* n *in* $\mathbb{G}$ *and* $n+1$ *operations in* $\mathbb{Z}_p$*;*
- *verification complexity* 1 *multi-exponentiation of length* $n+1$ *in* $\mathbb{G}$ *and* $n-1$ *operations in* $\mathbb{Z}_p$.

Moreover, the signing algorithm $\mathsf{DR.Sign}$ *is a signature of knowledge for relation* $\mathcal{R}_{\mathsf{DR}}$.

Construction 3 *We describe the ring signature construction of [26].*

- $pp_{\mathsf{DR}} \leftarrow \mathsf{DR.Gen}(1^\lambda)$*: runs* $\mathsf{G.Gen}(1^\lambda)$ *to obtain* $\mathsf{Gpp} = (p, \mathbb{G}, P)$ *and outputs* $pp_{\mathsf{DR}} := \mathsf{Gpp}$.
- $(sk, pk) \leftarrow \mathsf{DR.KeyGen}(pp_{\mathsf{DR}})$*: sample* $sk \leftarrow \mathbb{Z}_p$ *and output* $(sk, pk := P^{sk})$.
- $\sigma_{\mathsf{DR}} \leftarrow \mathsf{DR.Sign}(pp_{\mathsf{DR}}, sk_j, \vec{pk}, m)$*: Parse* $\vec{pk}$ *as* $(pk_i)_{i=1}^n \in \mathbb{G}^n$ *and* $m \in \mathbb{Z}_p$.
  - *sample* $c_1, ..., c_{j-1}, c_{j+1}, ..., c_n, x \leftarrow \mathbb{Z}_p$*;*
  - *set* $X := P^x \prod_{i \in [n] \setminus j} pk_i^{-c_i}$
  - *set* $c_j := H(pk_1, ..., pk_n, X, m) - \sum_{i \in [n] \setminus j} c_i$

- *set* $y := x + c_j sk_j$
- *output* $\sigma_{\mathsf{DR}} := (X, c_1, ..., c_n, y)$

– $\mathsf{DR.Verify}(pp_{\mathsf{DR}}, \vec{pk}, m, \sigma_{\mathsf{DR}})$*: Parse* $\vec{pk}$ *as* $(pk_j)_{j=1}^n \in \mathbb{G}^n$ *and* $m \in \mathbb{Z}_p$.

$$H(pk_1, ..., pk_n, X, m) = \sum_{i=1}^{n} c_i \tag{7}$$

$$P^y / X = \prod_{i=1}^{n} pk_i^{c_i} \tag{8}$$

If Equation 7 *and Equation* 8 *above both hold then output 1, else output 0.*

B Anonymity

Theorem 6. *DualDory is anonymous (Definition 13) in the random oracle model under the DDH assumption.*

Proof. Suppose that there is an adversary $\mathcal{A}$ which wins the anonymity game (Definition 13) with non-negligible advantage. We show that there is a distinguisher $\mathcal{D}$ which leverages $\mathcal{A}$ to break the DDH assumption in $\mathbb{G}_1$ in the random oracle model.

Let $\mathsf{BGpp} := (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, P, \tilde{P}) \leftarrow \mathsf{BG.Gen}(1^\lambda)$, and let $(U, V, W) \in \mathbb{G}_1^3$ be sampled either as a DDH tuple or uniformly at random, as in Definition 1. The distinguisher $\mathcal{D}$ receives BGpp and (U, V, W) as input. The distinguisher $\mathcal{D}$ simulates the anonymity game for $\mathcal{A}$ as follows. First, $\mathcal{D}$ samples public parameters exactly as $\mathsf{RS.Gen}$ would. $\mathcal{D}$ produces group elements $Q \leftarrow \mathbb{G}_1, \vec{\Gamma} \leftarrow \mathbb{G}_1^n$ and $\vec{\tilde{\Gamma}} \leftarrow \mathbb{G}_2^n$, sets $\mathsf{pp}_{\mathrm{Tag}} := (\mathbb{G}_1, p, P, Q, H, H')$ and $\mathsf{pp}_{\mathrm{PProd}} := (\mathsf{BGpp}, (\vec{\Gamma}, \vec{\tilde{\Gamma}}))$, and outputs $\mathsf{pp} := (\mathsf{pp}_{\mathrm{PProd}}, \mathsf{pp}_{\mathrm{Tag}})$. Next, $\mathcal{D}$ simulates $\mathsf{RS.KeyGen}$. $\mathcal{D}$ samples $j \leftarrow [n]$. Then, for each $i \in [n] \setminus \{j\}$, $\mathcal{D}$ samples secret key $sk_i \leftarrow \mathbb{Z}_p$ and computes corresponding public key $pk_i := P^{sk_i}$. Then, $\mathcal{D}$ sets $pk_j := U$.

Next, $\mathcal{D}$ simulates the anonymity game for $\mathcal{A}$. When $\mathcal{A}$ makes a corruption query pk_i, $\mathcal{D}$ checks whether $i = j$, and if so, aborts. If not, $\mathcal{D}$ outputs secret key sk_i. For simplicity purposes, we assume that $\mathcal{A}$ issues $n-2$ corruption queries for distinct indices. Consequently, the probability that $\mathcal{D}$ aborts is equal to $2/n$. On receiving signing query (m, prfx, pk_i), $\mathcal{D}$ computes the response by simulating random oracles $H' \colon \{0,1\}^* \to \mathbb{G}$ and $H \colon \{0,1\}^* \to \mathbb{Z}_p$ according to Algorithm 1 and Algorithm 2 respectively. More precisely:

– If $i \neq j$, then $\mathcal{D}$ outputs a signature according to $\mathsf{RS.Sign}$ from Construction 2, making calls to H and H' as needed.
– If $i = j$, then $\mathcal{D}$ computes commitment $\mathrm{com} = UQ^r$ and calls DualRing randomness r, index j and the vector $\vec{pk'} = (pk'_1, ..., pk'_n) = (\mathrm{com}/pk_1, ..., \mathrm{com}/pk_n)$. This results in tuple $(X, \vec{c}, y)$. Next, given $\vec{pk'}$, X, y and $\tilde{P}^{\vec{c}}$, $\mathcal{D}$ calls $\mathsf{P}_{\mathrm{PProd}}$ to produce proofs π_1 and π_2.
  $\mathcal{D}$ calls H' to compute $H'(\mathsf{prfx}) = V^{r'}$. $\mathcal{D}$ afterwards retrieves randomness r' from table H' and computes tag as $W^{r'}$. It then randomly picks $(\bar{a}, \bar{b}) \in \mathbb{Z}_p^2$ and invokes

H with tuple $\langle \mathsf{prfx}, \mathrm{com}, H'(\mathsf{prfx}), \mathrm{tag}, m||\pi_1||\pi_2, r, pk_j, \bar{a}, \bar{b}\rangle$. H outputs hash value c which $\mathcal{D}$ uses to compute pair

$$(A,B) = (H'(\mathsf{prfx})^{\bar{a}}\mathrm{tag}^{-c}, P^{\bar{a}}pk_j^{-c}Q^{\bar{b}-rc})$$

Using notations from Construction 2, $(A,B,\bar{a},\bar{b})$ corresponds to tag proof σ_{Tag}. $\mathcal{D}$ finally outputs signature $\sigma = (X, y, \mathbf{B}, \pi_1, \pi_2, \sigma_{\mathrm{Tag}}, \mathrm{tag}, \mathrm{com})$, where $\mathbf{B} = e(\vec{\Gamma}, \tilde{P}^{\vec{c}})$. Observe that if (U,V,W) is a DDH tuple, signature σ is statistically indistinguishable from a signature output by $\mathsf{RS.Sign}$. Otherwise, tag is not computed correctly. Yet, under the DDH assumption, $\mathcal{A}$ cannot tell the difference, given that $\mathsf{RS.Verify}(\vec{pk}, m, \mathsf{prfx}, \sigma) = 1$.

At some point of the experiment, $\mathcal{A}$ outputs a pair of public keys (pk_0^*, pk_1^*) and pair (m, prfx). Since $\mathcal{D}$ has not aborted, $pk_j \in \{pk_0^*, pk_1^*\}$. Without loss of generality, we assume that $pk_j = pk_0^*$. $\mathcal{D}$ simulates a signature for public key pk_j as described previously and returns the result. $\mathcal{A}$ then outputs her guess b. To break DDH, $\mathcal{D}$ outputs bit $1-b$, with 1 indicating that (U,V,W) is a DDH tuple, and 0 indicating otherwise. If (U,V,W) is a DDH tuple, then $\mathcal{A}$ will have a non-negligible advantage ε in outputting the correct guess $b = 0$. Namely, if $\mathcal{D}$ does not abort the experiment, then it will be able to to break DDH with the non-negligible advantage of $2\varepsilon/n$, $2/n$ is the probability that $\mathcal{D}$ aborts. Else if (U,V,W) is not a DDH tuple, then $\mathcal{A}$ will not perform better than a random guess, and so will $\mathcal{D}$. Actually, tuple $(\mathrm{com}, \mathrm{tag}, \sigma_{\mathrm{Tag}})$ in the signature leaks no information whatsoever about the underlying secret key – com is perfectly hiding, $\mathrm{tag} = W^r$ is a random group element and σ_{Tag} is computed without using any secret keys.

Algorithm 1 Simulating $H\colon \{0,1\}^* \to \mathbb{Z}_p$.

if $H[\mathsf{Q}]$ is undefined **then**
  $c \leftarrow \mathbb{Z}_p$
  if $\mathsf{Q} = \langle \mathsf{prfx}, \mathrm{com}, H'(\mathsf{prfx}), \mathrm{tag}, m, r, pk, \bar{a}, \bar{b}\rangle$ **then**
    $A \leftarrow H'(\mathsf{prfx})^{\bar{a}}\mathrm{tag}^{-c}$
    $B \leftarrow P^{\bar{a}}pk^{-c}Q^{\bar{b}-rc}$
    $H[\mathsf{prfx}, \mathrm{com}, \mathrm{tag}, A, B, m] \leftarrow c$
    Output c
  else
    $H[\mathsf{Q}] := c$
    Output c
  end if
else
  Output $H[\mathsf{Q}]$
end if

Algorithm 2 Simulating $H'\colon \{0,1\}^* \to \mathbb{G}_1$.

if $H'[\mathsf{Q}]$ is undefined **then**
  $r' \leftarrow \mathbb{Z}_p$
  $H'[\mathsf{Q}] \leftarrow \langle r', V^{r'}\rangle$
  Output $V^{r'}$
else
  $\langle r', \mathrm{hash}\rangle := H'[\mathsf{Q}]$
  Output hash
end if

C Prefix Linkability

Theorem 7. *DualDory is prefix linkable (Definition 14) in the random oracle model under the SXDH assumption.*

Proof. Assume there is an adversary $\mathcal{A}$ which breaks the prefix linkability of DualDory. We construct an adversary $\mathcal{B}$ which uses $\mathcal{A}$ to break the DPair assumption with two generators. Let $\mathsf{BGpp} := (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, P, \tilde{P}) \leftarrow \mathsf{BG.Gen}(1^\lambda)$, and let (P_1, P_2) be two additional generators of $\mathbb{G}_1$. Adversary $\mathcal{B}$ receives BGpp and (P_1, P_2) as input. $\mathcal{B}$'s goal is to output two generators $(\tilde{P}_1, \tilde{P}_2) \in \mathbb{G}_2^2$ such that $e(P_1, \tilde{P}_1)e(P_2, \tilde{P}_2) = 1$. Note that the SXDH assumption implies the DPair assumption with two generators [2], as well as the knowledge soundness of the tag proof, Dory, and DualRing. To simulate the prefix linkability experiment, $\mathcal{B}$ first produces group elements $\vec{\Gamma} \leftarrow \mathbb{G}_1^n$ and $\vec{\tilde{\Gamma}} \leftarrow \mathbb{G}_2^n$, sets $\mathsf{pp}_{\mathrm{Tag}} := (\mathbb{G}_1, p, P_1, P_2, H, H')$ and $\mathsf{pp}_{\mathrm{PProd}} := (\mathsf{BGpp}, (\vec{\Gamma}, \vec{\tilde{\Gamma}}))$, and outputs $\mathsf{pp} := (\mathsf{pp}_{\mathrm{PProd}}, \mathsf{pp}_{\mathrm{Tag}})$. $\mathcal{B}$ then uses $\mathsf{RS.KeyGen}(\mathsf{pp})$ to generate n key-pairs $\{(sk_i, pk_i)\}_{i=1}^n$ with $pk_i := P_1^{sk_i}$. During the experiment, $\mathcal{B}$ returns honest answers to $\mathcal{A}$'s queries to signing oracle $\mathsf{SO}_{\vec{pk}}$ and corruption oracle CO. At the end of the experiment $\mathcal{A}$ outputs $n+1$ signatures $(m_i, \mathsf{prfx}, \sigma_i)$ for $i \in [n+1]$. We parse σ_i as $(X_i, y_i, \mathbf{B}_i, \pi_{1,i}, \pi_{2,i}, \sigma_{\mathrm{Tag},i}, \mathrm{tag}_i, \mathrm{com}_i)$. Note that if $\mathcal{A}$ is able to break prefix linkability then $\mathsf{RS.Verify}(\mathsf{pp}, \vec{pk}, m_i, \mathsf{prfx}, \sigma_i) = 1$ for all $i \in [n+1]$, and $\mathsf{RS.Link}(\mathsf{pp}, \vec{pk}, \sigma_i, m_i, \sigma_j, m_j, \mathsf{prfx}) = 0$ for all $i \neq j \in [n+1]$. This implies that $\mathrm{tag}_i \neq \mathrm{tag}_j$ for all $i \neq j \in [n+1]$. Thanks to the extractability property of tag proof (Theorem 2), we can efficiently extract witnesses $sk_i', r_i' \in \mathbb{Z}_p$ such that

$$\forall i \in [n+1], \mathrm{com}_i = P_1^{sk_i'} P_2^{r_i'} \ \wedge \ \mathrm{tag}_i = H'(\mathsf{prfx})^{sk_i'} \ . \tag{9}$$

Since all of the tags tag_i are pairwise distinct, and each value sk_i' is uniquely determined by tag_i according to Equation 9, we conclude that the $n+1$ commitments com_i open to $n+1$ distinct values $sk_i' \in \mathbb{Z}_p$. Without loss of generality, we assume that $sk_{n+1}' \notin \{sk_1, ..., sk_n\}$. The knowledge soundness property of arguments of knowledge of bilinear pairing products (Theorem 1) allows us to extract $\vec{\tilde{\Omega}}$ such that $e((\frac{\mathrm{com}_{n+1}}{pk_i})_{i=1}^n, \vec{\tilde{\Omega}}) = e(\frac{P_2^{y_{n+1}}}{X_{n+1}}, \tilde{P})$ and $e(\vec{P}, \vec{\tilde{\Omega}}) = e(P, \tilde{P}^c)$ with $c = H(\mathbf{A}, X_{n+1})$. Now to break the DPair assumption, we use the forking lemma on hash $H(\mathbf{A}, X_{n+1})$. This yields another forgery

$$\sigma_{n+1}' = (X_{n+1}, y_{n+1}', \mathbf{B}_{n+1}', \pi_{n+1,1}', \pi_{n+1,2}', \pi_{\mathrm{Tag},n+1}', \mathrm{tag}_{n+1}, \mathrm{com}_{n+1})$$

Using the knowledge soundness property of arguments of knowledge of bilinear pairing products on the new forgery enables us to extract $\vec{\tilde{\Omega}}'$ such that $e((\frac{\mathrm{com}_{n+1}}{pk_i})_{i=1}^n, \vec{\tilde{\Omega}}') = e(\frac{P_2^{y_{n+1}'}}{X_{n+1}}, \tilde{P})$ and $e(\vec{P}, \vec{\tilde{\Omega}}') = e(P, \tilde{P}^{c'})$ with $c' = H(\mathbf{A}, X_{n+1})$. We have therefore:

$$e((\frac{\mathrm{com}_{n+1}}{pk_i})_{i=1}^n, \frac{\vec{\tilde{\Omega}}}{\vec{\tilde{\Omega}}'}) = \prod_{i=1}^n e(\frac{\mathrm{com}_{n+1}}{pk_i}, \frac{\tilde{\Omega}_i}{\tilde{\Omega}_i'}) = e(\frac{P_2^{y_{n+1}}}{P_2^{y_{n+1}'}}, \tilde{P})$$

Replacing com_{n+1} with $P_1^{sk_{n+1}'} P_2^{r_{n+1}}$ and pk_i with P^{sk_i}, and using the bilinearity of e, we get: $e(P_1, \prod_{i=1}^n (\frac{\tilde{\Omega}_i}{\tilde{\Omega}_i'})^{(sk_{n+1}' - sk_i)}) e(P_2, \prod_{i=1}^n (\frac{\tilde{\Omega}_i}{\tilde{\Omega}_i'})^{r_{n+1}}) = e(P_2, \frac{\tilde{P}^{y_{n+1}}}{\tilde{P}^{y_{n+1}'}})$. It follows that:

$$e(P_1, \prod_{i=1}^n (\frac{\tilde{\Omega}_i}{\tilde{\Omega}_i'})^{(sk_{n+1}' - sk_i)}) e(P_2, \frac{\tilde{P}^{y_{n+1}'}}{\tilde{P}^{y_{n+1}}} \prod_{i=1}^n (\frac{\tilde{\Omega}_i}{\tilde{\Omega}_i'})^{r_{n+1}}) = 1 \tag{10}$$

$\mathcal{B}$ breaks the DPair assumption by outputting

$$\tilde{P}_1 = \prod_{i=1}^{n}(\frac{\tilde{\Omega}_i}{\tilde{\Omega}'_i})^{(sk'_{n+1}-sk_i)} \; ; \; \tilde{P}_2 = \frac{\tilde{P}^{y'_{n+1}}}{\tilde{P}^{y_{n+1}}}\prod_{i=1}^{n}(\frac{\tilde{\Omega}_i}{\tilde{\Omega}'_i})^{r_{n+1}} = \frac{\tilde{P}^{y'_{n+1}}}{\tilde{P}^{y_{n+1}}}\tilde{P}^{(c-c')r_{n+1}}.$$

What remains now is to show that $\tilde{P}_1 \neq 1$ and $\tilde{P}_2 \neq 1$. Let ω_i denote $\log_{\tilde{P}}(\tilde{\Omega}_i)$ and ω'_i denote $\log_{\tilde{P}}(\tilde{\Omega}'_i)$. Accordingly, $\log_{\tilde{P}}(\tilde{P}_1) = \sum_{i=1}^{n}(\omega_i - \omega'_i)(sk_i - sk_{n+1})$. Let $\delta(x_1, x_2, ..., x_n) = \sum_{i=1}^{n}(\omega_i - \omega'_i)(x_i - sk_{n+1})$. Note that since $c = \sum_{i=1}^{n}\omega_i \neq c' = \sum_{i=1}^{n}\omega'_i$, there exists $i \in [n]$ such that $\omega_i \neq \omega'_i$. This means that polynomial δ is not a zero polynomial. Applying Schwartz-Zippel lemma to $\delta(sk_1, ..., sk_n)$, we have $\Pr[\log_{\tilde{P}}(\tilde{P}_1) = 0] \leq 1/p$. Recall that sk_i is sampled randomly in $\mathbb{Z}_p$. Given Equation 10, if $\log_{\tilde{P}}(\tilde{P}_1) \neq 0$, then so is $\log_{\tilde{P}}(\tilde{P}_2)$. Therefore, $Pr[\log_{\tilde{P}}(\tilde{P}_1) \neq 0 \wedge \log_{\tilde{P}}(\tilde{P}_2) \neq 0] \geq 1 - 1/p$. Consequently, $\tilde{P}_1$ and $\tilde{P}_2$ are two generators of $\mathbb{G}_2$ with probability $1 - 1/p$.

D Non-slanderability

Theorem 8. *DualDory is prefix non-slanderable (Definition 15) in the random oracle model under the SXDH assumption.*

Proof. Suppose there is an adversary $\mathcal{A}$ that breaks the prefix non-slanderability of DualDory. We construct, in the random oracle model, an adversary $\mathcal{B}$ which uses $\mathcal{A}$ to break either CDH in $\mathbb{G}_1$ or the discrete logarithm in $\mathbb{G}_1$, both of which are implied by the SXDH assumption. Recall that in the non-slanderability experiment, $\mathcal{A}$ outputs two tuples $(m', \mathsf{prfx}', \sigma')$ and $(m, \mathsf{prfx}', \sigma)$, and the first tuple is produced before any call to the corruption oracle CO. We distinguish between two cases depending on whether prfx' was queried to the signing oracle $\mathsf{SO}_{\vec{pk}}$ before outputting σ' or not.

prfx' *was queried before.* In this case, we show that $\mathcal{B}$ is able to break the discrete logarithm in $\mathbb{G}_1$ under the simulation extractability of tag proof. We assume that $\mathcal{B}$ would like to compute $u = \log_P(U)$. $\mathcal{B}$ simulates the prefix non-slanderability experiment as follows. First, $\mathcal{B}$ computes $\mathsf{pp} \leftarrow \mathsf{RS.Gen}(1^\lambda, n)$. Next, $\mathcal{B}$ samples $j \in [n]$ and sets $pk_j = U$. $\mathcal{B}$ then randomly selects $sk_i \in \mathbb{Z}_p^*$ and defines $pk_i = P^{sk_i}$ for $i \in [n], i \neq j$. On a corruption query pk_i, $\mathcal{B}$ returns sk_i if $i \neq j$; otherwise, $\mathcal{B}$ aborts. On a signing query (pk_i, m, prfx), $\mathcal{B}$ responds with a signature $\sigma_i \leftarrow \mathsf{RS.Sign}(\mathsf{pp}, \vec{pk}, sk_i, m, \mathsf{prfx})$, if $i \neq j$. Else, $\mathcal{B}$ computes tuple $(\mathrm{com}_i, X_i, \vec{c}_i, y_i)$ and the argument of knowledge of bilinear pairing products correctly. Then calls random oracle H' (Algorithm 3) to get $H'(\mathsf{prfx}) = P^{r'}$. With r', $\mathcal{B}$ computes $\mathrm{tag} = pk_j^{r'}$. $\mathcal{B}$ then simulates the corresponding tag proof by leveraging random oracle H (Algorithm 1). Notice that this signature verifies correctly.

Before any corruption query, $\mathcal{A}$ outputs forgery $(m', \mathsf{prfx}', \sigma')$, $\mathcal{B}$ retrieves tag' from σ' and checks whether $\mathrm{tag}' = H'(\mathsf{prfx}')^{sk_i}$ for some $i \neq j$. If that's the case, then $\mathcal{B}$ aborts. If $\mathcal{B}$ does not abort, then $\mathrm{tag}' = H'(\mathsf{prfx}')^{sk_j}$ where sk_j is defined by $pk_j = U = P^{sk_j}$. In fact, given the soundness of DualRing and argument of knowledge of bilinear pairing products: $\exists\, i \in [n] : \mathrm{com}' = P^{sk_i}Q^r \wedge pk_i = P^{sk_i}$, and by the soundness of tag proof, $\mathrm{tag}' = H'(\mathsf{prfx}')^{sk_i} \wedge \mathrm{com}' = P^{sk_i}Q^r$, whereas the binding property of Pedersen commitment

ensures that $\text{tag}' = H'(\mathsf{prfx}')^{sk_i}$ with $i \in [n]$. Hence, if $\text{tag}' \neq H'(\mathsf{prfx}')^{sk_i} \forall i \in [n] : i \neq j$, then $\text{tag}' = H'(\mathsf{prfx}')^{sk_j}$.

Moreover, σ' contains a signature of knowledge σ_{Tag} that is valid with respect to statement $\exists(sk_j, r) : \text{com}' = P^{sk_j} Q^r \wedge \text{tag}' = H'(\mathsf{prfx}')^{sk_j}$. Thanks to the extractability of tag proof, $\mathcal{B}$ extracts witness (sk_j, r). By outputting $sk_j = \log_P(U)$, $\mathcal{B}$ breaks the discrete logarithm assumption in $\mathbb{G}_1$. Notice that $\mathcal{B}$ could stop the game before any corruption query and still breaks the discrete logarithm assumption.

Algorithm 3 Simulating $H' \colon \{0,1\}^* \to \mathbb{G}_1$.

if $H'[\mathsf{Q}]$ is undefined **then**
  $r' \leftarrow \mathbb{Z}_p$
  $H'[\mathsf{Q}] \leftarrow \langle r', P^{r'} \rangle$
  Output $P^{r'}$
else
  $\langle r', \text{hash} \rangle := H'[\mathsf{Q}]$
  Output hash
end if

Algorithm 4 Simulating $H' \colon \{0,1\}^* \to \mathbb{G}_1$.

Let q_H denote an upper bound of the number of hash queries.
if $H'[\mathsf{Q}]$ is undefined **then**
  $r' \leftarrow \mathbb{Z}_p$
  hash $:= P^{r'}$ with probability $1 - 1/q_H$
  hash $:= V^{r'}$ with probability $1/q_H$
  $H'[\mathsf{Q}] := \langle r', \text{hash} \rangle$
  Output hash
else
  $\langle r', \text{hash} \rangle := H'[\mathsf{Q}]$
  Output hash
end if

prfx' *was not queried before.* In this case, we show that $\mathcal{B}$ is able to break the CDH assumption. The adversary $\mathcal{B}$ is given pair $(U, V) = (P^u, P^v)$ for $u, v \leftarrow \mathbb{Z}_p$, and must output $W = P^{uv}$. $\mathcal{B}$ simulates the prefix non-slanderability experiment as follows. First, $\mathcal{B}$ computes $\mathsf{pp} \leftarrow \mathsf{RS.Gen}(1^\lambda, n)$. Next, $\mathcal{B}$ samples $j \in [n]$ and sets $pk_j = U$ (implying that $sk_j = u$). $\mathcal{B}$ then randomly selects $sk_i \in \mathbb{Z}_p^*$ and defines $pk_i = P^{sk_i}$ for $i \in [n], i \neq j$. On a corruption query pk_i, B returns the matching sk_i if $i \neq j$; otherwise, $\mathcal{B}$ aborts. On a signing query (pk_i, m, prfx) for $i \neq j$, $\mathcal{B}$ responds with a signature $\sigma \leftarrow \mathsf{RS.Sign}(\mathsf{pp}, \vec{pk}, sk_i, m, \mathsf{prfx})$. On receiving a signing query (pk_j, m, prfx), $\mathcal{B}$ calls random oracle H' (cf. Algorithm 4) to compute $\text{hash} = H'(\mathsf{prfx})$. $\mathcal{B}$ then retrieves from H' table randomness r' and checks if $\text{hash} = V^{r'}$. If so, then $\mathcal{B}$ aborts. Note that the probability of this event is $1/q_H$ with q_H being the total number of hash queries. If $\text{hash} \neq V^{r'}$, then $\text{hash} = P^{r'}$. $\mathcal{B}$ accordingly, computes $\text{tag} = U^{r'} = pk_j^{r'}$, which corresponds to $H'(\mathsf{prfx})^{sk_j}$, and computes the signature by leveraging random oracle H as described in Algorithm 1. Note that this signature verifies correctly.

Before any corruption query, $\mathcal{A}$ outputs $(m', \mathsf{prfx}', \sigma')$. $\mathcal{B}$ checks whether $H'(\mathsf{prfx}') = V^{r'}$ for some $r' \in \mathbb{Z}_p$. If not, then $\mathcal{B}$ aborts. The probability that $\mathcal{B}$ aborts here is $1 - 1/q_H$. Else, $\mathcal{B}$ retrieves tag' from σ' and checks whether $\text{tag}' = H'(\mathsf{prfx}')^{sk_i}$ for some $i \neq j \in [n]$. If so, then $\mathcal{B}$ aborts. If not, $\mathcal{B}$ checks if $r' \neq 0$ and outputs $W = \text{tag}'^{1/r'}$. The probability that $r' = 0$ is $1/p$. Note that by the soundness properties of DualRing, argument of knowledge of bilinear pairing products, and tag proof, we have $\text{tag}'^{1/r'} = (H'(\mathsf{prfx}')^{sk_j})^{1/r'} = (P^{uv^r})^{1/r}$. It should be noted that $\mathcal{B}$ could stop the game before any corruption query and still breaks CDH. .

E Unforgeability

Theorem 9. *If a prefix-linkable ring signature* RS *is prefix-linkable (Definition 14) and non-slanderable (Definition 15), then it is also unforgeable (Definition 12).*

Proof. Assume there is an adversary $\mathcal{A}$ which is breaks the unforgeability of RS. We construct an adversary $\mathcal{B}_0$ ($\mathcal{B}_1$ resp.), which uses $\mathcal{A}$ to break prefix linkability (non-slanderability resp.).

$\mathcal{B}_0$ breaks prefix-linkability. To break prefix-linkability of RS, $\mathcal{B}_0$ first simulates the unforgeability experiment for $\mathcal{A}$. This is achieved by forwarding $\mathcal{A}$'s signing queries to oracle $\mathsf{SO}_{\vec{pk}}$ in the prefix linkability experiment. At the end of the simulated experiment, $\mathcal{A}$ outputs a forgery $(m_{n+1}, \mathsf{prfx}, \sigma_{n+1})$. On seeing this forgery, $\mathcal{B}_0$ continues its prefix-linkability experiment by calling SO with n signing queries $(m_i, \mathsf{prfx}, pk_i)$ for $1 \leq i \leq n$. Let σ_i denote $\mathsf{SO}_{\vec{pk}}$'s response to query $(m_i, \mathsf{prfx}, pk_i)$. By construction $\mathsf{RS.Link}(\vec{pk}, m_i, \sigma_i, m_j, \sigma_j, \mathsf{prfx}) = 0$ for all $i \neq j \in [n]$. Now $\mathcal{B}_0$ checks if there exists $1 \leq k \leq n$ such that

$$\mathsf{RS.Link}(\vec{pk}, m_k, \sigma_k, m_{n+1}, \sigma_{n+1}, \mathsf{prfx}) = 1$$

In that case, $\mathcal{B}_0$ aborts. Assuming *non-slanderability* of RS, the probability of such an event is negligible. Therefore, for all $i \neq j \in [n+1]$, $\mathsf{RS.Link}(\vec{pk}, m_i, \sigma_i, m_j, \sigma_j, \mathsf{prfx}) = 0$. This breaks prefix linkability.

$\mathcal{B}_1$ breaks non-slanderability. To break non-slanderability of RS, $\mathcal{B}_1$ simulates the unforgeability experiment for $\mathcal{A}$ by forwarding $\mathcal{A}$'s signing queries to oracle $\mathsf{SO}_{\vec{pk}}$ in the non-slanderability experiment. At the end of the simulated experiment, $\mathcal{A}$ returns a forgery $(m_{n+1}, \mathsf{prfx}, \sigma_{n+1})$. On seeing this forgery, $\mathcal{B}_1$ continues its non-slanderability experiment by calling $\mathsf{SO}_{\vec{pk}}$ with n signing queries $(m_i, \mathsf{prfx}, pk_i)$ for $1 \leq i \leq n$. Let σ_i denote SO's response to query $(m_i, \mathsf{prfx}, pk_i)$. $\mathcal{B}_1$ then checks if for all $i \neq j \in [n+1]$, $\mathsf{RS.Link}(\vec{pk}, m_i, \sigma_i, m_j, \sigma_j) = 0$. If so, then $\mathcal{B}_1$ aborts. Assuming prefix-linkability of RS, the probability of abort is negligible. Therefore:

$$\exists 1 \leq i, j \leq n+1 : \mathsf{RS.Link}(\vec{pk}, m_i, \sigma_i, m_j, \sigma_j, \mathsf{prfx}) = 1$$

We have by construction, $\forall\ 1 \leq i \neq j \leq n : \mathsf{RS.Link}(\vec{pk}, m_i, \sigma_i, m_j, \sigma_j, \mathsf{prfx}) = 0$. Hence, there exists $1 \leq k \leq n$ such that $\mathsf{RS.Link}(\vec{pk}, m_k, \sigma_k, m_{n+1}, \sigma_{n+1}, \mathsf{prfx}) = 1$. Now to break non-slanderability, $\mathcal{B}_1$ produces first $(m_{n+1}, \mathsf{prfx}, \sigma_{n+1})$ which was output by $\mathcal{A}$ as a forgery, and then (m_k, σ_k) where σ_k is the result of signing query $(m_k, \mathsf{prfx}, pk_k)$ to $\mathsf{SO}_{\vec{pk}}$.