Adapting Lyubashevsky's Signature Schemes to the Ring Signature Setting

Carlos Aguilar-Melchor[1], Slim Bettaieb[1], Xavier Boyen[2], Laurent Fousse[3], and Philippe Gaborit[1]

[1] XLIM-DMI, Université de Limoges, France
{carlos.aguilar,slim.bettaieb,philippe.gaborit}@xlim.fr
[2] Queensland University of Technology, Brisbane, Australia
xb@cs.stanford.edu
[3] Laboratoire Jean-Kuntzmann, Université de Grenoble, France
laurent.fousse@imag.fr

Abstract. Basing signature schemes on strong lattice problems has been a long standing open issue. Today, two families of lattice-based signature schemes are known: the ones based on the hash-and-sign construction of Gentry et al.; and Lyubashevsky's schemes, which are based on the Fiat-Shamir framework.
In this paper we show for the first time how to adapt the schemes of Lyubashevsky to the ring signature setting. In particular we transform the scheme of ASIACRYPT 2009 into a ring signature scheme that provides strong properties of security under the random oracle model. Anonymity is ensured in the sense that signatures of different users are within negligible statistical distance even under full key exposure. In fact, the scheme satisfies a notion which is stronger than the classical full key exposure setting as even if the keypair of the signing user is adversarially chosen, the statistical distance between signatures of different users remains negligible.
Considering unforgeability, the best lattice-based ring signature schemes provide either unforgeability against arbitrary chosen subring attacks or insider corruption in log-sized rings. In this paper we present two variants of our scheme. In the basic one, unforgeability is ensured in those two settings. Increasing signature and key sizes by a factor k (typically $80 - 100$), we provide a variant in which unforgeability is ensured against insider corruption attacks for arbitrary rings. The technique used is pretty general and can be adapted to other existing schemes.

Keywords: Ring signatures, lattices.

1 Introduction

In 2001 Rivest, Shamir and Tauman [21] introduced the concept of ring signature. In such a scheme, each user has a keypair; a secret signing key and a public verification key. Any of them can choose a subset of the public keys (including his own), the ring, and sign on behalf of the associated subset of users, without permission or assistance. This signature can be verified using the ring of public keys. Such a scheme must have the classic unforgeability property (it is not possible to sign on behalf of a ring without knowing one of the associated secret keys) as well as an anonymity property: it is not possible to know which secret key was used, just that it is associated to one of the public keys in the ring.

In 2006 [3], Bender *et al.* noted that the usual security setting with respect to anonymity and unforgeability did not take into account that some of the public keys in the ring may have been issued adversarially by an attacker. They therefore proposed adapted security definitions for such a situation as well as for the one in which all the secret keys including the one of the signers would be exposed. Of course if all the keys are exposed it is no longer possible to ensure unforgeability but anonymity of previously issued signatures may be preserved.

Such a strong anonymity property is a reassuring guarantee for a user hesitating to leak a secret, specially if the consequences of an identification are dire. It also seems reasonable, specially if anonymity must be preserved for a few decades (e.g. depending on the statute of limitations) not to rely on an estimation of

the computational complexity of a given problem and require unconditional anonymity. The definitions of Bender *et al.* can be easily translated to the unconditional setting and thus cover such a requirement.

A close but different setting, introduced by Chaum and van Heyst in 1991 [9], is the one of group signature. In this setting there is an anonymity revocation mechanism that allows a given group manager to reveal who was the signer of a given message. This property comes however at a cost as it requires a group setup procedure which is not needed in the ring signature setting.

Most of the existing ring signature schemes are based on number theory assumptions: large integer factorization [10, 21], discrete logarithm problem [1, 12] and bilinear pairing problems [22, 26, 5].

There are also a few ring signature schemes with security based on standard lattice problems. In [6], Brakerski and Kalai propose a ring signature scheme in the standard model based on the Small Integer Solution (SIS) problem using the hash-and-sign/bonsai-tree [11, 7] approach. Using again this approach Wang and Sun propose in [25] two other ring signature schemes, one under the random oracle model and one on the standard model. Both papers provide constructions in the standard model but, on the other hand, use Gentry-Peikert-Vaikuntanathan's (hereafter GPV) [11] strong trapdoors. These trapdoors are known to be very versatile and the ring signature constructions come naturally. However, when using these strong trapdoors a (hidden) structure is added to the underlying lattice which is, from a theoretical point of view, an important price to pay. In practice, it is possible to give public bases of these lattices which are closer to uniform, but some parameters, namely the dimension of the "perp" lattice (for a definition and details see [20]), are increased significantly in the process.

Two other schemes, one by Kawachi et al. [13], and one by Cayrel et al. [8], follow a very different approach. These schemes are based on weak trapdoors in which the underlying lattice is completely uniform except for some public syndromes given (which corresponds to a small vectors in the "perp" lattice). Even if we consider the lattice together with the syndromes, very little structure is added and in practice, it is close to uniform for smaller parameters than with GPV trapdoors. Such "weak trapdoors" can be used in not that many contexts (when compared to the strong GPV trapdoors) and [13, 8] prove that they are enough for ring signature. On the other hand, these schemes use pretty straightforwardly Stern's construction for signature schemes [24] which is somehow a bad property. Of course, it is clearly important to note that the code-based constructions can be translated to the lattice setting. In the case of the scheme by Cayrel et al. this has the added benefit that by adapting the code-based construction of Aguilar et al. [2] they obtain a threshold scheme, which is a pretty hard feature to obtain. However, we believe that finding ring signatures which follow the more recent and promising techniques of lattice-based cryptography, without the cumbersome zero-knowledge proofs used in code-based cryptography, is of independent interest.

Our Contributions. In this paper we present a lattice-based ring signature algorithm. As in the other lattice-based ring signature schemes, each signature and verification key is composed of a linear amount of sub-elements in the ring size.

Our main contribution is that our scheme is the first one to be based on Lyubashevsky's approach to lattice-based signature [16–18]. This is interesting from a theoretical point of view for two reasons. First, it is one of the major approaches to build standard signatures and no ring signature was until now based on it. Second, as Lyubashevsky's signatures, our scheme uses a weak trapdoor (a uniformly random lattice with a single syndrome) *without* Stern's zero-knowledge proofs (previous lattice-based schemes used either GPV strong trapdoors or Stern's proofs).

We describe our scheme as a modification of the scheme of ASIACRYPT 2009 [17]. The ideas and proofs can be re-used for the more recent schemes of [18], but the whole presentation gets trickier and the practical benefits are of limited interest (unlike in the standard signature setting).

As a second contribution we present a modification of our scheme, which can be applied to other lattice-based ring signature schemes to provide unforgeability in the insider corruption setting, even if the ring is of polynomial size in the security parameter. To the best of our knowledge this is the first time such a security property is obtained with lattice-based schemes.

Indeed, the schemes of Kawachi et al., Cayrel et al. and Brakerski and Kalai only provide a proof for the fixed ring or chosen subring attack settings. Wang and Sun provide a proof for insider corruption which only

works for log-sized rings. More precisely, the advantage of the SIS attacker in their proof is in $O(1/\binom{q_E}{q_E/2})$, q_E being an upper-bound on the ring size (see first line of the Setup step in the proof of Theorem 2 in [25]).

The third contribution we would like to mention is that our ring signature scheme provides unconditional anonymity even if the secret key of the signer and the public parameters have not been generated following the key generation process (in fact, even if they are adversarially chosen). In the light of recent results on the high percentage of unsure RSA keypairs [14], we find such a result a reassuring property for the users of a ring signature scheme.

Finally, we would like to note that the underlying scheme on which we are based seems to be more efficient than the schemes on which the alternative works are based. However, all ring signature lattice-based schemes, including ours, are still pretty unpractical and in any case far behind the best results of number theoretic schemes, such as [5], in size as well as in versatility. We therefore focus on the theoretical contributions and leave aside practical parameter comparisons.

2 Preliminaries

2.1 Notations

Polynomials and vectors of polynomials. Let $\mathbb{Z}_p$ denote the quotient ring $\mathbb{Z}/p\mathbb{Z}$. In this work we build our cryptographic construction upon the ring $\mathcal{D} = \mathbb{Z}_p[x]/\langle x^n + 1\rangle$; where $x^n + 1$ is irreducible, n is a power of two, and p is a prime such that $p = 3 \mod 8$. The elements of $\mathcal{D}$ will be represented by polynomials of degree $n-1$ having coefficients in $\{-(p-1)/2, \ldots, (p-1)/2\}$.

We will denote polynomials by roman letters $(a, b, \ldots)$, vectors of polynomials will be denoted by a roman letter with a hat $(\hat{a}, \hat{b}, \ldots)$. Let m some positive integer such that, $a_1, \ldots, a_m$ are polynomials in $\mathcal{D}$, then we can write $\hat{a} = (a_1, \ldots, a_m)$. For any polynomial a, the infinity norm ℓ_∞ is defined by $\|a\|_\infty = max_i|a^{(i)}|$, with $a^{(i)}$ the coefficients of the polynomial, and for a vector of polynomials by $\|\hat{a}\|_\infty = max_i\|a_i\|_\infty$.

Sets. For a positive integer i, $[i]$ denotes the set $\{1, \ldots, i\}$. For a given set S, the notation $x \leftarrow S$ represents a uniformly random sample from the set, and for a given randomized algorithm $x \leftarrow$ `RandomizedAlgorithm` represents a sampling from the possible outputs following the distribution given by the algorithm.

In our scheme, forgery attacks become easier as the ring size grows (this is also true for other schemes). We therefore suppose that there is a constant c such that acceptable ring sizes are bounded from above by k^c, k being the security parameter. As signature and verification key sizes have an amount of sub-elements proportional to the ring size the reader can replace c by 1 or 2 which will cover any reasonable use of these signatures. The table below defines different sets we use and the parameters associated to these sets.

n	power of 2 greater than the security parameter k
p	prime of order $\Theta(n^{4+c})$ such that $p \equiv 3 \mod 8$
m_u	$(3 + 2c/3)\log n$
m	$(3 + 2c/3)n^c \log n$
D_h	$\{g \in \mathcal{D} : \|g\|_\infty \leq mn^{1.5}\log n + \sqrt{n}\log n\}$
D_y	$\{g \in \mathcal{D} : \|g\|_\infty \leq mn^{1.5}\log n\}$
D_z	$\{g \in \mathcal{D} : \|g\|_\infty \leq mn^{1.5}\log n - \sqrt{n}\log n\}$
$D_{s,c}$	$\{g \in \mathcal{D} : \|g\|_\infty \leq 1\}$

Table 1. Sets and parameters.

Rings and random oracles. Each keypair of the ring signature scheme is in any protocol, game or experiment we may present always uniquely defined by an integer (its index). We define a ring R as a set of verification keys. We consider that there is a bijection between users and keypairs and sometimes we will

implicitly use this bijection saying that a user belongs to a ring. We also define $\#R$ as the size of the ring (i.e. the amount of verification keys it contains), and as $index(R)$ the set of integers corresponding to the indexes of the verification keys in R (each keypair being uniquely defined by its index).

$H : \{0,1\}^* \to D_{s,c}$ denotes a random oracle. We describe a ring $R = \{pk_{i_1}, \ldots, pk_{i_{\#R}}\}$, for example when using it as input to the random oracle, by $desc(pk_{i_1}) \| \ldots \| desc(pk_{i_{\#R}})$ where $desc(pk)$ is a binary description of a public key, $\|$ is the concatenation operator and $i_1 < \cdots < i_{\#R}$ (the representation is thus unique). When possible we will just skip $desc()$ in such notations, $f_1 \| f_2$ meaning the concatenation of the description of functions f_1 and f_2.

2.2 Collision-Resistant Hash Functions

In [15] Lyubashevsky and Micciancio introduced a family $\mathcal{H}$ of collision-resistant hash functions with security based on the worst-case hardness of standard lattice problems over ideal lattices.[4]

Definition 1. *For any integer m and $D_\times \subseteq \mathcal{D}$, let $\mathcal{H}(\mathcal{D}, D_\times, m) = \{h_{\hat{a}} : \hat{a} \in \mathcal{D}^{m_u}\}$ be the function family such that for any $\hat{z} \in D_\times^{m_u}$, $h_{\hat{a}}(\hat{z}) = \hat{a} \cdot \hat{z} = \sum a_i z_i$, where $\hat{a} = (a_1, ..., a_m)$ and $\hat{z} = (z_1, ..., z_m)$ and all the operations $a_i z_i$ are performed in the ring $\mathcal{D}$.*

Note that hash functions in $\mathcal{H}(\mathcal{D}, D_\times, m)$ satisfy the following two properties for any $\hat{y}, \hat{z} \in \mathcal{D}^{m_u}$ and $\boldsymbol{c} \in \mathcal{D}$:

$$h(\hat{y} + \hat{z}) = h(\hat{y}) + h(\hat{z}) \tag{1}$$

$$h(\hat{y}c) = h(\hat{y})c \tag{2}$$

Moreover, when the input domain is restricted to a strategically chosen set $D_\times^{m_u} \subset \mathcal{D}^{m_u}$, the function family is collision resistant. We first introduce the collision finding problem and then present the security reduction result for a well-chosen $D_\times$.

Definition 2 (Collision Problem). *Given an element $h \in \mathcal{H}(\mathcal{D}, D_\times, m)$, the collision problem $Col(h, D_\times)$ (where $D_\times \subset \mathcal{D}$) asks to find distinct elements $\hat{\boldsymbol{z}}_1, \hat{\boldsymbol{z}}_2 \in D_\times$ such that $h(\hat{z}_1) = h(\hat{\boldsymbol{z}}_2)$.*

It was shown in [15] that, when $D_\times$ is restricted to a set of small norm polynomials, solving $Col(h, D_\times)$ is as hard as solving $SVP_\gamma(\mathcal{L})$ in the worst case over lattices that correspond to ideals in $\mathcal{D}$.

Theorem 1 (Theorem 1 in [17], applied to our setting). *Let $\mathcal{D}$ be the ring $\mathbb{Z}_p[x] / \langle x^n + 1 \rangle$ for n a power of two. Define the set $D_\times = \{y \in \mathcal{D} \mid \|y\|_\infty \leq d\}$ for some integer d. Let $\mathcal{H}(\mathcal{D}, D_\times, m)$ be a hash function family as in Definition 1 such that $m > \frac{\log p}{\log 2d}$ and $p \geq 4dmn^{1.5} \log n$. If there is a polynomial-time algorithm that solves $Col(h, D_\times)$ for random $h \in \mathcal{H}(\mathcal{D}, D_\times, m)$ with some non-negligible probability, then there is a polynomial-time algorithm that can solve $SVP_\gamma(\mathcal{L})$ for every lattice corresponding to an ideal in $\mathcal{D}$, where $\gamma = 16dmn \log^2 n$.*

In this paper we will set $d = mn^{1.5} \log n + \sqrt{n} \log n$, $D_\times = D_h$, and n, p, m as suggested in Table 1. This ensures that the conditions required by the above theorem are verified and that finding collisions for $\mathcal{H}(\mathcal{D}, D_h, m)$ implies an algorithm for breaking SVP in the worst-case over ideal lattices for polynomial gaps (in n and therefore in k).

In the ring signature scheme we present, the manipulated hash functions will always belong to sets $\mathcal{H}(\mathcal{D}, D_h, m')$ with $m' \leq m$. It is important to note that if an attacker is able to solve the above problem for $m' \leq m$ he can also solve it for m. Indeed, when given a challenge $h \in \mathcal{H}(\mathcal{D}, D_h, m)$ the attacker can puncture the tuple of polynomials describing h, to obtain a tuple of m' polynomials, solve the collision problem for m' and the pad the obtained solution with zeros on the punctured coordinates to obtain a solution to the problem for h.

[4] In this work, by ideal lattices, we make reference to the discrete subgroups of $\mathbb{Z}_p^n$ that can be mapped from ideals in rings of the form $\mathbb{Z}_p[x] / \langle f \rangle$ for some irreducible polynomial of degree n. The mapping between ideals and ideal lattices is trivially derived from the canonical isomorphism between polynomials $v^{(0)} + v^{(1)}x + \ldots + v^{(n-1)}x^{n-1}$ in $\mathbb{Z}_p[x]/\langle f \rangle$ and vectors $v = (v^{(0)}, \ldots, v^{(n-1)})$ in $\mathbb{Z}_p^n$.

2.3 Statistical Distance

The statistical distance measures how different are two probability distributions. In this paper we will use this tool to prove that the ring signature scheme presented is anonymous.

Definition 3 (Statistical Distance). *Let X and X' be two random variables over a countable set S. We define by:*

$$\Delta(X, X') = \frac{1}{2} \sum_{x \in S} |Pr[X = x] - Pr[X' = x]|$$

the statistical distance between X and X'.

One important property of the statistical distance is that it cannot be increased by a randomized algorithm which is formalized by the following proposition.

Proposition 1 (Proposition 8.10 in [19]). *Let X, X' be two random variables over a common set A. For any (possibly randomized) function f with domain A, the statistical distance between $f(X)$ and $f(X')$ is at most*

$$\Delta(f(X), f(X')) \leq \Delta(X, X').$$

This proposition implies that if the statistical distance of two families of random variables (X_k) and (X'_k) is negligible,[5] an attacker given a sample will only obtain a negligible advantage over a wild guess when trying to distinguish between the distributions of (X_k) and the ones of (X'_k). Note that the proposition above does not make any assumption on the computational complexity of f and thus this is true whether the attacker is computationally bounded or unbounded.

Note that the statistical distance may grow if we consider multiple variables. It is easy to verify from definition 3 that if X, Y follow a distribution ϕ and X', Y' a distribution ϕ', we have

$$2\Delta(X, X') \geq \Delta((X, Y), (X', Y')) \geq \Delta(X, X'). \tag{3}$$

Thus, if an attacker is given many samples of the same distribution he may be able to distinguish better than with just one sample. More specifically, using (3) iteratively, if the attacker is given $\#s$ samples of the same distribution and the families of random variables have an upper-bound $\epsilon(k)$ on the statistical distance, the advantage over a wild guess for such an attacker will be bounded from above by $\#s * \epsilon(k)$.

In Section 4.1 we prove that, for our scheme, the signatures of two different users have a statistical distance which is exponentially small in k and thus, even a computationally unbounded attacker given an exponential amount of signatures of the same user will have a negligible advantage over a wild guess when trying to break anonymity.[6]

Attacker with additional information. An attacker trying to distinguish between the distributions of two random variables X_k, X'_k may have some extra information which we model as a third random variable Z_k. This information may for example be obtained during an indistinguishability game prior to obtaining a sample (e.g. two public keys). If X_k and X'_k are not dependent on Z_k, this extra information is of no use to the attacker as then, using proposition 8.8 from [19], $\Delta((X_k, Z_k), (X'_k, Z_k)) = \Delta(X_k, X'_k)$ and therefore we still have $\Delta(f(X_k, Z_k), f(X'_k, Z_k)) \leq \Delta(X_k, X'_k)$.

If X_k or X'_k depend on Z_k (as signatures depend on the public keys in our case) we cannot use the same argument, as proposition 8.8 from [19] only applies to independent variables. However, noting $X_{k,z}$ and $X'_{k,z}$ the random variables conditioned on $Z = z$, if we have an upper-bound

$$\Delta(X_{k,z}, X'_{k,z}) < \epsilon(k)$$

[5] i.e. asymptotically bounded from above by k^{-c} for any c, k being the security parameter.

[6] For example for $2^{k/2}$ samples, and $\epsilon(k) = 2^{-k}$ we have $\#s * \epsilon(k) = 2^{-k/2}$.

which is independent of z it is also an upper-bound for $\Delta(f(X_k, Z_k), f(X'_k, Z'_k))$. Indeed, we have

$$\begin{aligned}
\Delta(f(X_k, Z_k), f(X'_k, Z_k)) &\leq \Delta((X_k, Z_k), (X'_k, Z_k)) \\
&= \frac{1}{2} \sum_{x,z} |Pr[(X_k, Z_k) = (x, z)] - Pr[(X'_k, Z_k) = (x, z)]| \\
&= \frac{1}{2} \sum_{z} Pr[Z_k = z] \sum_{x} |Pr[X_k = x \mid Z_k = z] - Pr[X'_k = x \mid Z_k = z]| \\
&= \sum_{z} Pr[Z_k = z] \Delta(X_{k,z}, X'_{k,z}) \\
\Delta(f(X_k, Z_k), f(X'_k, Z_k)) &\leq \epsilon(k).
\end{aligned}$$

Note that the upper bound is valid independently of the distribution followed by Z_k, and thus we can include in this random variable parameters adversarially chosen by the attacker.

2.4 Ring Signature Schemes: Definitions and Properties

A ring signature schemes gives means to individual users to define an arbitrary set of public keys R (the ring), and issue a signature using a secret key associated to one of the public keys in the ring. Using the set R and a verification algorithm it is possible to verify that a signature has been issued by a member of the ring (i.e. by a user who knows a secret key associated to one of the public keys in R), but it is not possible to learn whom.

Ring Signature Scheme. We will describe a ring signature scheme by triple of algorithms ($\mathsf{Ring{-}gen}$, $\mathsf{Ring{-}sign}$, $\mathsf{Ring{-}verify}$):

- $\mathsf{Ring{-}gen}(1^k)$: A probabilistic polynomial time algorithm that takes as input a security parameter k, outputs a a public key pk and a secret key sk. For many schemes, the users in a ring must share in common some public information derived from k. We thus suppose that $\mathsf{Ring{-}gen}(1^k)$ has two sub-algorithms: $\mathsf{Ring{-}gen{-}params}(1^k)$ which generates a set of public parameters $\mathcal{P}$ which are used in all the algorithms; and $\mathsf{Ring{-}gen{-}keys}(\mathcal{P})$ which generates keypairs based on the public parameters. We suppose that the constant c is defined in the description of the scheme (we could also define it as an input parameter of $\mathsf{Ring{-}gen{-}params}$).
- $\mathsf{Ring{-}sign}(\mathcal{P}, sk, \mu, R)$: A probabilistic polynomial time algorithm that takes as input a set of parameters $\mathcal{P}$, a signing key sk, a message $\mu \in \mathcal{M}$ ($\mathcal{M}$ being the message space of the scheme) and a set of public keys R (the ring). It returns a signature σ for μ under sk, or `failed`.
- $\mathsf{Ring{-}verify}(\mathcal{P}, \sigma, \mu, R)$: A deterministic algorithm that takes as input a set of parameters $\mathcal{P}$, a ring signature σ on a message μ and a set of public keys R, and outputs 1 or 0 for accept or reject respectively.

We require the following correctness condition: for any k, any ℓ (bounded by a polynomial in k), any $\mathcal{P} \in \mathsf{Ring{-}gen{-}params}(1^k)$, any $\{(pk_i, sk_i)\}_{i\in[\ell]} \subset \mathsf{Ring{-}gen{-}keys}(\mathcal{P})$, any $i_0 \in [\ell]$, any message μ, and any $\sigma \in \mathsf{Ring{-}sign}(\mathcal{P}, sk_{i_0}, \mu, \{pk_i\}_{i\in[\ell]})$, $\sigma \neq \texttt{failed}$, we have $\mathsf{Ring{-}verify}(\mathcal{P}, \mu, \sigma, \{pk_i\}_{i\in[\ell]}) = 1$. Moreover, in order to be considered secure, a ring signature scheme must satisfy some anonymity and unforgeability properties. Our goal will be to obtain the following two properties:

Anonymity. In [3], Bender et al. define various levels of anonymity for a ring signature scheme. Among them the highest is *anonymity against full key exposure*. The authors of [3] use the same definition for two levels of security (attribution attacks and full-key exposure) which results in a definition which is slightly too complex for our needs. We use here a definition which is stronger and as a consequence, much simpler. Indeed, we do not need to define a signing or corruption oracle as the attacker knows all the secrets and can thus simulate both oracles effectively, and we don't have a first step in which the challenger generates the parameters and keys as these can be adversarially chosen. The definition is given by the following game:

Unconditional Anonymity Against Chosen Setting Attacks

1. $\mathcal{A}$ outputs a set of public parameters $\mathcal{P} = (k, n, m_u, p, S)$, a ring $R = \{pk_1, \ldots, pk_\ell\}$ for ℓ in $[k^c]$, two distinct indices $i_0, i_1 \in [k^c]$, two secret keys sk_{i_0}, sk_{i_1}, and a message μ.
2. Two signatures $\sigma_0 \leftarrow \mathsf{Ring{-}sign}(\mathcal{P}, sk_{i_0}, \mu, R)$, $\sigma_1 \leftarrow \mathsf{Ring{-}sign}(\mathcal{P}, sk_{i_1}, \mu, R)$ are generated and a random bit b is chosen. If $\sigma_0 \neq \texttt{failed}$ and $\sigma_1 \neq \texttt{failed}$, $\mathcal{A}$ is given σ_b, else the game is restarted.
3. $\mathcal{A}$ outputs a bit b' and succeeds if $b' = b$.

The ring signature scheme achieves unconditional anonymity against chosen setting attacks if any adversary $\mathcal{A}$ has an advantage with respect to a wild guess which is negligible in the security parameter k.

This definition can be easily generalized to the case in which the adversary is given sets of samples instead of a single sample. In order to simplify the definition we use just one sample, but the proofs of Section 4.1 show that the anonymity is ensured even if the attacker is given an exponential amount of samples.

Unforgeability. In [3] different notions of unforgeability are introduced. We present here all of them as going from the simplest to the most complex definition helps in the presentation.

For a ring signature scheme with ℓ members, the unforgeability against fixed-ring attacks is defined using the following experiment. The challenger firstly runs the algorithm $\mathsf{Ring{-}gen}$ to obtain compatible keypairs $(pk_1, sk_1), \ldots, (pk_\ell, sk_\ell)$ for the signature scheme and sends $R = \{pk_i\}_{i \in [\ell]}$ to the forger. The forger can then make polynomially many signing queries. A ring signing query is of the form $(i_{\text{signer}}, \mu, R)$ for varying $\mu \in \mathcal{M}$, $i_{\text{signer}} \in index(R)$. The challenger replies with $\sigma \leftarrow \mathsf{Ring{-}sign}(\mathcal{P}, sk_{i_{\text{signer}}}, \mu, R)$. Finally the forger outputs $(\sigma^\star, \mu^\star, R)$ and it wins if $\mathsf{Ring{-}verify}(\mathcal{P}, \sigma^\star, \mu^\star, R)$ outputs accept and $\mu^\star$ is not one of the messages for which a signature was queried.

An intermediate, stronger, definition is unforgeability against chosen subring attacks in which signing queries and the final forgery can be done with respect to any subring $S \subset R$. The strongest definition proposed is unforgeability with respect to insider corruption. In this setting, signing queries can be done with respect to *any* ring (i.e. the attacker can add to these rings adversarially generated public keys). The attacker is also given a corruption oracle that for any $i \in index(R)$ returns sk_i. Forgeries are valid if they do not correspond to a signing query, the subring of the forgery is a subset of R, and none of the secret keys of the subring have been obtained through the corruption oracle.

In this paper we will first show that the proof of [16] can be adapted to our scheme in order to prove unforgeability with respect to subring attacks. We will then prove that for an alternative version of this scheme it is also possible to ensure unforgeability with respect to insider corruption.

3 Our Scheme

3.1 Informal Description

In this section we first provide a high-level description of the tree-less signature scheme in [16], and then we show how to transform it into a ring signature scheme. In [16], the signer has as (secret) signing key $\hat{s}$ and a (public) verification key (h, S) such that $h(\hat{s}) = S$. The domains to which these keys belong will be made explicit later.

To make a signature of some message μ, the signer will prove that he can:

- Choose a random vector of polynomials $\hat{y}$ (which he does not reveal)
- Output a vector of polynomials whose difference with $\hat{y}$ is $\hat{s}$ (his secret key) times a small polynomial e that is not of his choice

In order to do this the signer will select some random $\hat{y}$, compute $e = H(h(\hat{y}), \mu)$ and output $(\hat{z}, e)$ with $\hat{z} = \hat{s}e + \hat{y}$. The verifier tests that $e = H(h(\hat{z}) - Se, \mu)$. This is true for a correct signature thanks to the linearity of h as $h(\hat{z}) - Se = h(\hat{s}e + \hat{y}) - Se = h(\hat{y}) + Se - Se = h(\hat{y})$.

To obtain a ring signature from this scheme we make two major modifications. The first one is to ensure that each user has in his public key a function h_i that satisfies $h_i(\hat{s}_i) = S$ where $\hat{s}_i$ is the secret key and S is a fixed standard polynomial (not null). Consider a ring $R = \{h_i\}_{i\in[\ell]}$. The second modification keeps the real signer anonymous when he signs a message. We do this by simply adding $\ell - 1$ random variables that will corresponds to the ring members except the real signer. For example, suppose that the real signer is indexed by $j \in [\ell]$, the signer sends the signature $(\hat{\boldsymbol{z}}_i; i \in [\ell], e)$ with $e = H(\sum_{i\in[\ell]} h_i(\hat{y}_i), \mu)$, $\hat{z}_j = \hat{s}_j e + \hat{y}_j$ and $\hat{z}_i = \hat{y}_i$ for $i \in [\ell] \setminus \{j\}$. Therefore, the final signature will contain $[\ell]$ elements one for each member in the ring. Now we go gack to the first modification and we will show its utility in the correctness of the scheme. In the verification step, the verifier checks if the hash value of $(\sum_{i\in[\ell]} h_i(\hat{z}_i) - Se, \mu)$ is equal to e. Note that this will be true only if $\sum_{i\in[\ell]} h_i(\hat{z}_i) - Se$ is equal to $\sum_{i\in[\ell]} h_i(\hat{y}_i)$. In fact using the linearity of h_j we have $h_j(\hat{z}_j) = h_j(\hat{s}_j e + \hat{y}_j) = h_j(\hat{y}_j) + Se$. Since all the ring members have key pairs $(h_i, \hat{s}_i)$ such that $h_i(\hat{s}_i) = S$, the verifier will always accept when one of them produces a ring signature.

In order to resist to chosen subring attacks against unforgeability we must modify the scheme to include in the random oracle call a description of the ring for which the signature is valid (if not, it is easy to reuse a signature to generate a forged signature on a larger ring). In order to resist to attacks on adversarially chosen parameters, the ring signature algorithm starts with an initial step in which the inputs are required to pass simple tests (bounds on the amount of coordinates, scalar sizes, etc.). The security proofs proposed by Lyubashevsky must also be modified to take into account the existence of multiple hash functions and signature elements. Moreover, using modified hash functions also requires that we introduce a few new lemmas and propositions to complete the proofs.

3.2 A More Formal Description

Ring Signature Scheme

Ring−gen−params(1^k):
Given an integer k define the common public parameters.

1. Set n as a power of two larger than k
2. Set $m_u = 3\log n$, and p as a prime larger than n^4 such that $p = 3 \mod 8$

— Note: these parameters define the sets $\mathcal{D}, D_h, D_z, D_y, D_{s,c}$ and the family $\mathcal{H}$.

3. Set $S \leftarrow \mathcal{D}$, $S \neq 0$
4. Output $\mathcal{P} = (k, n, m_u, p, S)$

Ring−gen−keys($\mathcal{P}$):
Generate a keypair.

1. Set $\hat{s} = (s_1, s_2, \ldots, s_{m_u}) \leftarrow D_{s,c}^{m_u}$
2. If none of the s_i is invertible, go to 1.
3. Let $i_0 \in \{1, \ldots, m\}$ such that s_{i_0} is invertible.
4. $(a_1, a_2, \ldots, a_{i_0-1}, a_{i_0+1}, \ldots, a_{m_u}) \leftarrow \mathcal{D}^{m_u-1}$.
5. Let $a_{i_0} = s_{i_0}^{-1}(S - \sum_{i\neq i_0} a_i s_i)$ and note $\hat{a} = (a_1, \ldots, a_{m_u})$
6. Output $(pk, sk) = (h, \hat{s})$, h being the hash function in $\mathcal{H}$ defined by $\hat{a}$

Ring−sign$(\mathcal{P}, sk, \mu, R)$:
Given a message $\mu \in \mathcal{M}$, a ring of ℓ members with public keys $R = \{h_i\}_{i\in[\ell]} \subset \mathcal{H}(\mathcal{D}, D_h, m_u)$, and a private key $sk = \hat{s}_j$ associated to one of the public keys h_j in R, generate a ring signature for the message.

0. Verify that: the public parameters respect the constraints of steps 1−3 in Ring−gen−params ; sk is in $D_{s,c}^{m_u}$; R is of size bounded by k^c ; one of the public keys in R is associated to sk. If the verification fails output `failed`.
1. For all $i \in [\ell]$; $i \neq j$;$\hat{y}_i \leftarrow D_z^{m_u}$
2. For $i = j$; $\hat{y}_j \leftarrow D_y^{m_u}$
3. Set $\boldsymbol{e} \leftarrow H(\sum_{i\in[\ell]} h_i(\hat{y}_i), R, \mu)$ (e is therefore in $D_{s,c}$)
4. For $i = j$, $\hat{z}_j \leftarrow \hat{s}_j \cdot e + \hat{y}_j$
5. If $\hat{\boldsymbol{z}}_j \notin D_z^{m_u}$ then go to Step 2
6. For $i \neq j$, $\hat{z}_i = \hat{y}_i$
7. Output $\sigma = (\hat{\boldsymbol{z}}_i; i \in [\ell], e)$

Ring−verify$(\mathcal{P}, \mu, R, \sigma)$:
Given a message μ, a ring $R = \{h_i\}_{i\in[\ell]}$ and a ring signature $\sigma = (\hat{\boldsymbol{z}}_i; i \in [\ell], e)$, the verifier accepts the signature only if both of the following conditions satisfied:

1. $\hat{z}_i \in D_z^{m_u}$ for all $i \in [\ell]$
2. $e = H(\sum_{i\in\{1,\ldots,\ell\}} h_i(\hat{z}_i) - S \cdot e, R, \mu)$

Otherwise, the verifier rejects.

3.3 Correctness and convergence of the algorithms

The correctness of the signing algorithm is pretty straightforward. Indeed, let $\sigma = (\hat{\boldsymbol{z}}_i; i \in [\ell], \{h_i\}_{i\in[\ell]}, e) \leftarrow$ Ring−sign$(\mathcal{P}, \hat{s}_j, \mu, \{h_i\}_{i\in[\ell]})$ be a signature with $j \in [\ell]$ and $(h_j, \hat{s}_j)$ a given keypair. The first test in Ring−verify is always passed by a valid signature as steps 2 and 5 of Ring−sign ensure that signatures only contain elements in $D_z^{m_u}$. With respect to the second test we have:

$$\begin{aligned}
\sum_{i\in[\ell]} h_i(\hat{z}_i) - S \cdot e &= h_j(\hat{z}_j) - S \cdot e + \sum_{i\in[\ell]\setminus\{j\}} h_i(\hat{z}_i) \\
&= h_j(\hat{s}_j e + \hat{y}_j) - S \cdot e + \sum_{i\in[\ell]\setminus\{j\}} h_i(\hat{y}_i) && \text{by replacing } \hat{z}_j \text{ by } \hat{s}_j \cdot e + \hat{y}_j \text{ and } \hat{z}_i \text{ by } \hat{y}_i, \\
&= h_j(\hat{s}_j) \cdot e + h_j(\hat{y}_j) - S \cdot e + \sum_{i\in[\ell]\setminus\{j\}} h_i(\hat{y}_i) && \text{using the homomorphic properties of } h_j \in \mathcal{H}, \\
&= \sum_{i\in[\ell]} h_i(\hat{y}_i) && \text{as } h_j(\hat{s}_j) = S.
\end{aligned}$$

As $e = H(\sum_{i\in[\ell]} h_i(\hat{y}_i), \{h_i\}_{i\in[\ell]}, m)$ the second test of Ring−verify is therefore always satisfied by a valid signature.

A correctly issued signature is therefore always verified. Let's consider now the expected running time of the different algorithms.

Proposition 2. *The expected running times of* Ring−gen−params, Ring−gen−keys, Ring−sign *and* Ring−verify *are polynomial in the security parameter.*

Proof. All the operations in the different algorithms can be executed in polynomial time in the security parameter. Thus, Ring−gen−params and Ring−verify run in polynomial time and the only possible issue would be the amount of iterations in the loops of Ring−gen−keys and Ring−sign.

Lemma 3, proved in the appendix, states that each of the polynomials chosen in step 1 of algorithm Ring−gen−keys is invertible with probability exponentially close to one and thus the expected amount of iterations in the associated loop is roughly one. Thus the expected running time of Ring−gen−keys is polynomial.

In Ring−sign the outer loop has a polynomial amount of iterations, on the other hand inner loop between steps 2 and 5 which will continue as long as $\hat{z_j} = \hat{s}_j \cdot e + \hat{y}_j \notin D_z^{m_u}$. Corollary 6.2 from [16] states that for any $\hat{s} \in D_s^{m_u}$,

$$Pr_{c \leftarrow D_{s,c}, \hat{y} \leftarrow D_y^{m_u}} [\hat{s}c + \hat{y} \in D_z^{m_u}] = \frac{1}{e} - o(1).$$

As e and $\hat{y}_j$ are drawn uniformly from $D_{s,c}$ and $D_y^{m_u}$ we can use this result. This implies that the expected amount of iterations in the inner loop of Ring−sign is less than 3, and therefore that Ring−sign also runs in expected polynomial time. □

4 Security of the proposed scheme

4.1 Anonymity

In the anonymity against chosen setting attacks game, the adversary receives a signature which depends on a random bit b as well as on a set of public parameters $\mathcal{P} = (k, n, m_u, p, S)$, two secret keys sk_{i_0}, sk_{i_1}, a message μ, and a ring of public keys R. All of these parameters have been chosen adversarially except the random bit b.

Let $X_{b,\mathcal{P},sk_{i_b},\mu,R}$ be the random variable that represents the signature received by the adversary for a given set of parameters. The following theorem states that for any choice of $\mathcal{P}, sk_{i_0}, sk_{i_1}, \mu, R$ which does not result in a game restart, the statistical distance between $X_{0,\mathcal{P},R,sk_{i_0},\mu}$ and $X_{1,\mathcal{P},R,sk_{i_1},\mu}$ is negligible in k.

Theorem 2 (Anonymity). *For $b \in \{0,1\}$, let $X_{b,\mathcal{P},sk_{i_b},\mu,R}$ be the random variable describing the output of* Ring−sign$(\mathcal{P}, sk_{i_b}, \mu, R)$ *with $\mathcal{P} = (k, n, m_u, p, S), sk_{i_b}, \mu, R$ a set of arbitrary inputs to the algorithm. If the domains of these variables are both different from* {failed} *we have*

$$\Delta(X_{0,\mathcal{P},sk_{i_0},\mu,R}, X_{1,\mathcal{P},sk_{i_1},\mu,R}) = n^{-\omega(1)}.$$

The proof of this theorem is available on the appendix. Using the properties of statistical distance presented in Section 3 this implies that our scheme ensures unconditional anonymity against chosen setting attacks.

4.2 Unforgeability against chosen subring attacks

In this section we will show that an adversary able to break the unforgeability in the chosen subring setting for our scheme is also able to break the unforgeability of Lyubashevsky's scheme. Given the results of [16], this implies an efficient algorithm to solve SVP_γ on ideal lattices (for γ defined as in Theorem 1). It is important to note that in Table 1 we change the parameters given by Lyubashevsky by increasing m and p, but the reader can easily check (using the sketch proof of the corollary below) that the proofs given in [16] are still valid and that the inequalities in Theorem 1 are verified.

Corollary 1 (of Theorems 6.5 and 6.6 in [16]). *If there is a polynomial time algorithm that can break the unforgeability of the signature scheme proposed in [16] for the parameters presented in Table 1, and more precisely for hash functions with $m = n^c * m_u$ columns, there is a polynomial time algorithm that can solve SVP_γ for $\gamma = O(n^{2.5+2c})$ for every lattice L corresponding to an ideal in D.*

Proof (Sketch.). The unforgeability proof of [16] is split in two. First it reduces signature unforgeability from collision finding and then collision finding from SVP (for a given gap). The first part of the reduction is given by Theorem 6.6 which is almost unchanged for our parameters. As we have a major modification on m, the amount of columns of the hash functions, the only point which could raise an issue is when Lemma 5.2 and Theorem 6.5 are used. These prove that for the parameters in [16] there is a second pre-image for a given output of the hash function with high probability and that signatures using two such pre-images are indistinguishable. As we increase the amount of columns in the hash functions used, the probability of a second pre-image and the uniformity of the output are increased and thus we can still use these results.

Therefore using Theorem 6.6 of [16] we can deduce that breaking unforgeability for our parameters implies finding collisions for the corresponding hash functions. As our hash functions have more columns than in [16] it is easier to find such collisions than with the original parameters. Using Theorem 1 on our parameters shows that collision finding can be reduced from SVP_γ for $\gamma = O(n^{2.5+2c})$ (instead of $O(n^{2.5})$ for the original scheme).

□

In our ring signature scheme, an unforgeability challenge is a set of verification hash functions defined by a set of ℓ tuples of m_u polynomials. In Lyubashevsky's signature scheme an unforgeability challenge is a single tuple of m' polynomials. The idea is thus, considering an attacker to our scheme, to set $m' = \ell \times m_u$, transform this challenge into a set of ℓ tuples and show that if we give this as a ring signature challenge to the attacker, we will obtain with non-negligible probability a valid forgery for Lyubashevsky's scheme.

The are two key issues. First, the polynomials in the m'-tuple are uniformly random but the polynomials in the m_u-tuples that the attacker expects to receive are not. We address this issue by proving, in Corollary 2, that the statistical distance between public keys in our ring signature scheme, and hash functions chosen uniformly at random is negligible. This implies that an attacker able to forge signatures for hash functions associated to the public keys is also able to forge them for hash functions chosen uniformly at random. Indeed, if we model the attacker by a randomized function, it cannot increase the statistical distance between the two families of functions and thus if it succeeds with non-negligible probability for a family he also will for the other. The second issue is that we do not know the private keys associated to the challenge given to the attacker but we must be able to answer to his signing queries. We solve this issue simply by programming the random oracle that the attacker uses.

In order to prove Corollary 2, we will use the following theorem.

Theorem 3 (Adapted from [23], Theorem 3.2). *For $g_1, \ldots, g_{m_u-1} \in \mathcal{D}$, we denote by $F(g_1, \ldots, g_{m_u-1})$ the random variable $\sum_{i\in[m_u-1]} s_i g_i \in D$ where $s_1, \ldots, s_{m_u-1}$ are chosen uniformly at random in $D_{s,c}$. Noting $U_1, \ldots, U_{m_u}$ independent uniform random variables in D, we have*

$$\Delta((U_1, \ldots, U_{m_u-1}, F(U_1, \ldots, U_{m_u-1})), (U_1, \ldots, U_{m_u})) \leq \frac{1}{2}\sqrt{\left(1 + \left(\frac{p}{3^{m_u-1}}\right)^{n/2}\right) - 1}$$

Proof. Just apply Theorem 3.2 from [23] to our setting, using the fact that by Lemma 2.3 from [23] our choice of parameters ensures that $x^n + 1 = f_1 f_2 \bmod p$ where each f_i is irreducible in $\mathbb{Z}_p[x]$ and can be written $f_i(x) = x^{n/2} + t_i x^{n/4} - 1$ with $t_i \in \mathbb{Z}_p$.

□

Corollary 2 (Uniformity of the public key). *Let $X_\mathcal{P}$ be a random variable describing the distribution of the hash functions resulting from the key generation algorithm* $\mathsf{Ring{-}gen{-}keys}(\mathcal{P})$ *and $U_1, \ldots, U_{m_u}$ denote independent uniform random variables in D. Then*

$$\Delta\left(X_\mathcal{P}, (U_1, \ldots, U_{m_u})\right) \leq n^{-\omega(1)}.$$

Proof. We describe a hash function $h_{\hat{a}}$ by the set of polynomials $\hat{a} = (a_1, \ldots, a_{m_u})$. We suppose, w.l.o.g. that $(a_1, \ldots, a_{m_u}) = (a_1, \ldots, a_{m_u-1}, s_{m_u}^{-1}(S - \sum_{i\in[m_u-1]} a_i s_i))$, where $\hat{s} = (s_1, \ldots, s_{m_u})$ is the secret key corresponding to $h_{\hat{a}}$.

We first note that the function on the right side of the inequality in Theorem 3 is negligible for our parameters as $p = \Theta(n^{4+c})$ and $3^{m_u-1} = 3^{(3+2c/3)\log_2 n}/3 = n^{(3+2c/3)\log_2 3}/3 > n^{4.5+c}/3$. Thus, using Theorem 3, we have

$$\Delta((U_1, \ldots, U_{m_u-1}, F(U_1, \ldots, U_{m_u-1})), (U_1, \ldots, U_{m_u})) \leq n^{-\omega(1)}.$$

Proposition 1 states that a function cannot increase the statistical distance. Noting $f(g_1, \ldots, g_{m_u})$ the function that leaves unchanged the $m_u - 1$ first coordinates and replaces g_{m_u} by $s_{m_u}^{-1}(S - g_{m_u})$ we get

$$\Delta(f(U_1, \ldots, U_{m_u-1}, F(U_1, \ldots, U_{m_u-1})), f(U_1, \ldots, U_{m_u})) \leq n^{-\omega(1)}.$$

To prove the corollary we just need to note that $f(U_1, \ldots, U_{m_u-1}, F(U_1, \ldots, U_{m_u-1}))$ has exactly the same distribution as $X_{\mathcal{P}}$, and $f(U_1, \ldots, U_{m_u})$ the same as $(U_1, \ldots, U_{m_u})$. The first assertion is trivially true given the $\mathsf{Ring-gen-keys}$ algorithm. The second one comes from the fact that adding an element (in our case S) or multiplying by an invertible element (in our case $s_{m_u}^{-1}$) in D just permutes the elements of the ring and thus the uniform distribution remains unchanged. We therefore have

$$\Delta(f(U_1, \ldots, U_{m_u-1}, F(U_1, \ldots, U_{m_u-1})), f(U_1, \ldots, U_{m_u})) = \Delta(X_{\mathcal{P}}, (U_1, \ldots, U_{m_u}))$$

and thus, $\Delta(X_{\mathcal{P}}, (U_1, \ldots, U_{m_u})) \leq n^{-\omega(1)}$.

□

We are now ready to prove our theorem on unforgeability against chosen subring attacks.

Theorem 4. *If there is a polynomial time algorithm that can break the unforgeability under chosen subring attacks of the ring signature scheme described in Section 3.2 for the parameters presented in Table 1, there is a polynomial time algorithm that can solve $SVP_\gamma(\mathcal{L})$ for $\gamma = \tilde{O}(n^{2.5+2c})$ for every lattice $\mathcal{L}$ corresponding to an ideal in $\mathcal{D}$.*

Proof (Sketch.).

Suppose that we have an adversary $\mathcal{A}$ that can output a forgery for the ring signature scheme with non-negligible advantage in the chosen subring setting. Given Corollary 1, it is enough to prove that using $\mathcal{A}$, we can construct a polynomial time adversary $\mathcal{B}$ that outputs a forgery for Lyubashevsky's scheme with non-negligible advantage for the parameters given in Table 1.

Setup: $\mathcal{B}$ is given as a challenge a description of a hash function (namely a tuple of $m' = \ell \times m_u$ polynomials $(a_1, \ldots, a_{\ell \times m_u})$ in D), an element S of $\mathcal{D}$, and access to the random oracle H_L of the signing algorithm. $\mathcal{B}$ splits the set of polynomials in ℓ sets of m_u polynomials $(a_{i,1}, \ldots, a_{i,m_u})$ for $i \in [\ell]$. Finally, $\mathcal{B}$ initializes $\mathcal{A}$ by giving it the set of tuples generated, the public parameters associated (among which S) and access to the *ring signature* random oracle H which it controls.

Query Phase: $\mathcal{B}$ answers the random oracle and signing queries of $\mathcal{A}$ as follows. For each random oracle query (x_y, x_h, x_m) it will test whether it has already replied to such a query. If so, it will reply consistently. If not, it will reply with $H_L(x_y, x_h \| x_m)$, $\|$ being the concatenation operator and store the result. For each signing query $(\{h_i\}_{i \in T}, i_0, \mu)$ for $i_0 \in T \subseteq [\ell]$, $\mathcal{B}$ programs H to produce a signature. In other words:

1. It follows the steps of the signature by generating a set of $\hat{y}_i \leftarrow D_y^{m_u}$ for $i \in T$
2. It generates at random $r \leftarrow D_{s,c}$
3. It checks whether H_L has been called with parameters $(\sum_i h_i(\hat{y}_i) - S \cdot r, \{h_i\}_{i \in T} \| \mu)$ (if so it aborts)
4. It programs the random oracle H so that $H(\sum_i h_i(\hat{y}_i) - S \cdot r, \{h_i\}_{i \in T}, \mu) = r$, and stores the result
5. Finally, it outputs $(\hat{y}_i; i \in T, r)$.

Forgery Phase: At a given point, $\mathcal{A}$ finishes running and outputs a forgery $((\hat{z}_i; i \in T, e), \mu, \{h_i\}_{i \in T})$, for $T \subseteq [\ell]$ with non-negligible probability. $\mathcal{B}$ just pads the remaining coordinates of the signature with null polynomials $\hat{z}_i = 0$ for $i \in [\ell] \backslash T$ and outputs $((\hat{z}_1 \| \cdots \| \hat{z}_\ell, e), \mu)$, $\hat{z}_1 \| \cdots \| \hat{z}_\ell$ being the vector of polynomials resulting from the concatenation of each of the vectors of polynomials $\hat{z}_i$.

Analysis: In this analysis we will detail completely the reasoning but just sketch the statistical arguments as they are pretty standard and easy to verify.

First of all, note that in step 3 of the protocol above there is the possibility of an abort. The probability of this event is negligible. Indeed, given the sizes of D_y and p^n, m_u is large enough to ensure by leftover hash lemma [4] that $\sum_i h_i(\hat{y}_i) - S \cdot r$ is within negligible distance from an uniformly random distribution. Thus, the probability that $\mathcal{A}$ has generated beforehand a query colliding with the one of the protocol is negligible.

In order to prove that $\mathcal{A}$ will output a ring signature forgery with non-negligible probability we must show that all the inputs it receives are within negligible statistical distance to the ones it would have received in a ring signature challenge. Once this is proved, we must show that the final forgery that $\mathcal{B}$ outputs is valid in the basic signature scheme.

First note that by Corollary 2 and using equation (3) of Section 3, we have that the challenge given to $\mathcal{A}$ is within negligible statistical distance to the one it has in a ring signature challenge. Another family of inputs to consider are the signatures generated by $\mathcal{B}$. These signatures are trivially valid for the given random oracle. Following the same ideas as those used in Theorem 2 we have that these signatures are withing negligible statistical distance from a signature generated using the signing algorithm (omitted). The last inputs to consider are the ones coming from the random oracle H which are pairs $(preimage, image)$ of the graph of H. All of the images of H are chosen uniformly at random over $D_{s,c}$. The issue is that we have programmed H and thus, in our protocol the first coordinate of the pre-image and the image are related. However, using the leftover hash lemma as in the first paragraph of this analysis (when considering aborts) we have that we add to this coordinate $\sum_i h_i(\hat{y}_i)$ which is close to uniform and independent of the image. This implies that (omitting the coordinates set by $\mathcal{A}$ with his query), the pairs $(preimage, image)$ are statistically close to uniform.

We therefore can ensure that $\mathcal{A}$ outputs a forgery for the ring signature scheme with non-negligible probability and the only issue left to prove is the validity of the forgery given by $\mathcal{B}$. The basic idea we will use in the rest of this proof is that in the forgery $((\hat{z}_i; i \in T, e), \mu, \{h_i\}_{i\in T})$ that $\mathcal{A}$ outputs, if e has been obtained during a direct call to the random oracle we can prove that the forgery is valid in the basic signature scheme. If not, we use the same ideas than in Theorem 6.6 of [16].

A wild guess of e can only happen with negligible probability. Indeed, as $D_{s,c}$ is exponentially large, if H has not been queried the probability that the output of $\mathcal{A}$ will give e when using H in the verification step is exponentially small. If we therefore suppose that e has been obtained through a call to H there are two possibilities: either e has been generated during a signing query, or it has been generated during a direct random oracle query. We leave the latter option for the end of the proof.

Suppose e has been generated during a signing query which resulted in an output $(\hat{z}'_i; i \in T', e)$ for a ring $\{h_i\}_{i\in T'}$ and a message μ'. In order to be valid, the forgery must be different from $((\hat{z}'_i; i \in T', e), \mu', \{h_i\}_{i\in T'})$ and thus we must have either $\{h_i\}_{i\in T'}\|\mu' \neq \{h_i\}_{i\in T}\|\mu$ or $(\hat{z}'_i; i \in T') \neq (\hat{z}_i; i \in T)$. The former case can only happen with negligible probability as it implies a collision for H. Indeed, $\{h_i\}_{i\in T'}\|\mu' \neq \{h_i\}_{i\in T}\|\mu$ implies $\{h_i\}_{i\in T'} \neq \{h_i\}_{i\in T}$ or $\mu' \neq \mu$ and in both cases we have a collision as $H(\sum_{i\in T'} h_i(\hat{z}'_i) - S \cdot e, \{h_i\}_{i\in T'}, \mu') = H(\sum_{i\in T} h_i(\hat{z}_i) - S \cdot e, \{h_i\}_{i\in T}, \mu) = r$ (we have twice the same image for two pre-images that are different either in the second or the third coordinate). In the latter case, using the same reasoning, the first coordinate of H is the same in the two cases with overwhelming probability and thus we have $\sum_{i\in T'} h_i(\hat{z}'_i) - S \cdot e = \sum_{i\in T} h_i(\hat{z}_i) - S \cdot e$ for $(\hat{z}'_i; i \in T') \neq (\hat{z}_i; i \in T)$. Setting $\hat{z}_i = 0$ for $i \in [\ell]\backslash T$ and $\hat{z}'_i = 0$ for $i \in [\ell]\backslash T'$ we obtain a collision for a random hash function of $\mathcal{H}(\mathcal{D}, D_h, \ell)$. If this event happens with non-negligible probability, using Theorem 1 we deduce that we can solve SVP_γ.

We conclude this proof by working on the case in which e has been generated during a direct random oracle query. We have the forgery $((\hat{z}_i; i \in T, e), \mu, \{h_i\}_{i\in T})$ and as the ring signature must be valid we have $H(\sum_{i\in T} h_i(\hat{z}_i) - S \cdot e, \{h_i\}_{i\in T}, \mu) = e$. Noting $x_y = \sum_{i\in T} h_i(\hat{z}_i) - S \cdot e$ and given the algorithm followed by $\mathcal{B}$, we also have $H_L(x_y, \{h_i\}_{i\in T}\|\mu) = e$. If we pad the $\hat{z}_i$ with $\hat{z}_i = 0$ for $i \in [\ell]\backslash T$ we still have $\sum_{i\in[\ell]} h_i(\hat{z}_i) - S \cdot e = x_y$ and thus $H_L(\sum_{i\in[\ell]} h_i(\hat{z}_i) - S \cdot e, \{h_i\}_{i\in T}\|\mu) = e$. We therefore have that $\sigma' = (\hat{z}_1\| \cdots \|\hat{z}_\ell, e)$ is a signature of $\mu' = \{h_i\}_{i\in T}\|\mu$ and, as we have not done any signing query in the basic signature scheme, $(\sigma', \mu', [\ell])$ is a valid forgery.

4.3 Unforgeability against insider corruption attacks

Suppose that we have an algorithm $\mathcal{A}$ able to break the unforgeability of our scheme with insider corruption attacks. We would like to prove again that there exists an algorithm $\mathcal{B}$ which using $\mathcal{A}$ can break the unforgeability of the underlying basic signature scheme. The main issue is that when $\mathcal{B}$ gets the challenge from the basic signature scheme he can split the tuple of polynomials but he does not know the signing keys associated to those split tuples and thus he cannot answer to the corresponding corruption queries.

One idea would be to pass to $\mathcal{A}$ modified versions of these tuples so that $\mathcal{B}$ knows the signing keys associated to them, but we have been unable to find a way to use the final forgery with such a strategy. A more standard approach to solve this issue is the one used by Wang and Sun in [25], which consists in giving to $\mathcal{A}$ more tuples than the ones obtained from splitting the initial challenge. These additional tuples are generated using the key generation algorithm and $\mathcal{B}$ knows the associated signing keys. If there are enough of them it is feasible to obtain a run from $\mathcal{A}$ in which all the corruption queries correspond to tuples for which $\mathcal{B}$ has the signing key.

Unfortunately, this creates a new issue in [25] as well as in our case. If we have a lot of tuples which do not correspond to the challenge then, for some strategies of $\mathcal{A}$ which are plausible, the final forgery is for a ring which contains tuples which were not in the challenge with overwhelming probability. In some number theory schemes this is not an issue, but in our case such forgeries are useless as $\mathcal{B}$ could have generated them (as the statistical distance between ring signatures by different members of the ring is negligible).

In the end, if the ring size for which $\mathcal{A}$ works is polynomial in the security parameter, for some strategies of $\mathcal{A}$ (in particular if he corrupts half of the ring and gives a ring signature for the other half), all the trade-offs fail. If we have a super logarithmic amount of new tuples there is an overwhelming probability to have a forgery for an inappropriate ring, and if not there is an overwhelming probability that $\mathcal{B}$ will be unable to answer to some of the corruption queries.

Our approach In order to solve these issues we modify the key generation process. Each user generates a set of k verification keys, k being the security parameter. Among these keys $k/2$ are generated through the original key generation algorithm $\mathsf{Ring{-}gen{-}keys}$, and the user stores the associated signing keys. The other $k/2$ verification keys are just uniformly chosen hash functions. The k verification keys are numbered, the order being chosen at random (mixing both types of keys).

$\mathsf{Ring{-}gen{-}keys{-}ic}(\mathcal{P})$**:**

1. Choose randomly a subset $T \subset [k]$ of size $k/2$.
2. For $i \in T$, set $pk_i \leftarrow \mathcal{H}(\mathcal{D}, D_h, m)$ and $sk_i = 0$.
3. For $i \in [k] \backslash T$, set $(pk_i, sk_i) \leftarrow \mathsf{Ring{-}gen{-}keys}(\mathcal{P})$.
4. Output $(pk, sk) = (\{pk_i\}_{i\in[k]}, \{sk_i\}_{i\in[k]})$.

For a set of users $S \subset \mathbb{Z}$ (we associate users to integers), we define $\texttt{fulldesc}(S)$ as a description of the full set of verification keys of the users of S.

When signing a message μ for a set of users S, the user calls a first random oracle with input $(\mu, \texttt{fulldesc}(S))$. The output of this random oracle is $\{T_{\sigma,i}\}_{i\in S}$, a set of subsets of $[k]$, each of them of size $k/2$. In order to sign, the user will create a ring of verification keys T which includes for each user i the subset of verification keys indexed by $T_{\sigma,i}$.

Ring−sign−ic$(\mathcal{P}, sk, \mu, R)$:

0. Verify that: the public parameters respect the constraints of steps $1-3$ in Ring−gen−params ; each $sk_i \in sk$ is in $D_{s,c}^{m_u}$; R is of size bounded by k^c ; one of the public keys in R is associated to sk. If the verification fails output `failed`.
1. Set $\{T_{\sigma,i}\}_{i\in R} \leftarrow H_\sigma(\mu, \text{keys}(R))$
2. Define $\text{keys}(T) = (pk_{i,j})_{i\in R, j\in T_{\sigma,i}}$
3. Let i_0 denote the index of the signer in R. Choose randomly $sk_i \in sk$ with $i \in T_{\sigma,i_0}$ such that $sk_i \neq 0$. If none exists, abort.
4. Output Ring−sign$(\mathcal{P}, sk_i, \mu, T)$.

Note that since the random oracle chooses $k/2$ verification keys of the signer at random, the probability that they all are uniformly chosen random hash functions is exponentially small and thus the probability of an abort in Step 3 is negligible.

The adaptation of the verification algorithm to this new setting is straightforward.

Sketch of proof in the insider corruption setting We just outline the ideas of the proof which can be obtained by small modifications in the proof of Theorem 4.

We use an insider corruption attacker to break a challenge of Lyubashevsky's signature. In order to do this we get as a challenge a tuple of $m' = k/2 \times \ell \times m_u$ polynomials. We use them in our ring signature algorithm instead of the uniformly chosen hash function obtained in the usual key generation process. The rest of the keys are generated according to Ring−gen−keys−ic (i.e. through the original algorithm Ring−gen−keys). Note that as half of the verification keys of each user are generated using this algorithm, we can answer to the corruption queries of the attacker.

In order to answer to the challenge we want the attacker to output a signature that only uses as verification keys the ones corresponding to the challenge and none of the ones generated by Ring−gen−keys. The main idea is to call the attacker with a controlled random oracle H_σ, pre-generate a polynomial number of the outputs of this oracle sampling uniformly its range and guess which one of these outputs will be used in the forgery. There is a polynomial loss in the reduction but using this approach we can then order the keys of the different users so that if the attacker uses the random oracle reply we expect, he will only use keys for which we do not have the associated signing key. When this happens (after a polynomial number of calls to the attacker), we obtain a forgery which can be used to answer the initial challenge.

References

1. M. Abe, M. Ohkubo, and K. Suzuki. 1-out-of-n signatures from a variety of keys. Advances in Cryptology-Asiacrypt 2002, pages 639–645, 2002.
2. Carlos Aguilar Melchor, Pierre-Louis Cayrel, Philippe Gaborit, and Fabien Laguillaumie. A new efficient threshold ring signature scheme based on coding theory. IEEE Transactions on Information Theory, 57(7):4833–4842, 2010.
3. Adam Bender, Jonathan Katz, and Ruggero Morselli. Ring signatures: Stronger definitions, and constructions without random oracles. In Proceedings of TCC 2006, volume 3876 of LNCS, pages 60–79. Springer-Verlag, 2006.
4. Charles H. Bennett, Gilles Brassard, and Jean-Marc Robert. Privacy amplification by public discussion. SIAM J. Comput., 17(2):210–229, 1988.
5. Xavier Boyen. Mesh signatures. In Advances in Cryptology - EUROCRYPT 2007, 26th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Barcelona, Spain, May 20-24, 2007, Proceedings, pages 210–227, 2007.
6. Zvika Brakerski and Yael Tauman Kalai. A framework for efficient signatures, ring signatures and identity based encryption in the standard model. Technical report, Cryptology ePrint Archive, Report 2010086, 2010.
7. David Cash, Dennis Hofheinz, Eike Kiltz, and Chris Peikert. Bonsai trees, or how to delegate a lattice basis. In Advances in Cryptology - EUROCRYPT 2010, 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, pages 523–552, 2010.

8. Pierre-Louis Cayrel, Richard Lindner, Markus Rückert, and Rosemberg Silva. A lattice-based threshold ring signature scheme. In Progress in Cryptology - LATINCRYPT 2010, First International Conference on Cryptology and Information Security in Latin America, Puebla, Mexico, August 8-11, 2010, Proceedings, volume 6212 of Lecture Notes in Computer Science, pages 255–272. Springer, 2010.
9. David Chaum and EugÃ¨ne van Heyst. Group signatures. In Advances in Cryptology - EUROCRYPT '91, Workshop on the Theory and Application of of Cryptographic Techniques, Brighton, UK, April 8-11, 1991, Proceedings, pages 257–265, 1991.
10. Y. Dodis, A. Kiayias, A. Nicolosi, and V. Shoup. Anonymous identification in ad hoc groups. In Advances in Cryptology-EUROCRYPT 2004, pages 609–626. Springer, 2004.
11. C. Gentry, C. Peikert, and V. Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In Proceedings of the 40th annual ACM symposium on Theory of computing, pages 197–206. ACM, 2008.
12. J. Herranz and G. Sáez. Forking lemmas for ring signature schemes. Progress in Cryptology-INDOCRYPT 2003, pages 266–279, 2003.
13. Akinori Kawachi, Keisuke Tanaka, and Keita Xagawa. Concurrently secure identification schemes based on the worst-case hardness of lattice problems. In ASIACRYPT '08: Proceedings of the 14th International Conference on the Theory and Application of Cryptology and Information Security, pages 372–389, 2008.
14. Arjen K. Lenstra, James P. Hughes, Maxime Augier, Joppe W. Bos, Thorsten Kleinjung, and Christophe Wachter. Ron was wrong, whit is right. Cryptology ePrint Archive, Report 2012/064, 2012. http://eprint.iacr.org/.
15. V. Lyubashevsky and D. Micciancio. Generalized compact knapsacks are collision resistant. Automata, Languages and Programming, pages 144–155, 2006.
16. Vadim Lyubashevsky. Towards practical lattice-based cryptography. PhD thesis, University of California, San Diego, 2008.
17. Vadim Lyubashevsky. Fiat-shamir with aborts: Applications to lattice and factoring-based signatures. In Advances in Cryptology - ASIACRYPT 2009, 15th International Conference on the Theory and Application of Cryptology and Information Security, Tokyo, Japan, December 6-10, 2009. Proceedings, volume 5912 of Lecture Notes in Computer Science, pages 598–616. Springer, 2009.
18. Vadim Lyubashevsky. Lattice signatures without trapdoors. In Advances in Cryptology - EUROCRYPT 2012 - 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15-19, 2012. Proceedings, volume 7237 of Lecture Notes in Computer Science, pages 738–755. Springer, 2012.
19. Daniele Micciancio and Shafi Goldwasser. Complexity of Lattice Problems: a cryptographic perspective, volume 671 of The Kluwer International Series in Engineering and Computer Science. Kluwer Academic Publishers, Boston, Massachusetts, March 2002.
20. Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In Advances in Cryptology - EUROCRYPT 2012, volume 7237 of Lecture Notes in Computer Science, pages 700–718. Springer, 2012.
21. Ronald L. Rivest, Adi Shamir, and Yael Tauman. How to leak a secret. In Advances in Cryptology - ASIACRYPT 2001, 7th International Conference on the Theory and Application of Cryptology and Information Security, Gold Coast, Australia, December 9-13, 2001, Proceedings, volume 2248 of Lecture Notes in Computer Science, pages 552–565. Springer, 2001.
22. H. Shacham and B. Waters. Efficient ring signatures without random oracles. Public Key Cryptography–PKC 2007, pages 166–180, 2007.
23. D. Stehlé, R. Steinfeld, K. Tanaka, and K. Xagawa. Efficient public key encryption based on ideal lattices. Advances in Cryptology–ASIACRYPT 2009, pages 617–635, 2009.
24. Jacques Stern. A new identification scheme based on syndrome decoding. In CRYPTO, volume 773 of Lecture Notes in Computer Science, pages 13–21. Springer, 1993.
25. J. Wang and B. Sun. Ring signature schemes from lattice basis delegation. Information and Communications Security, pages 15–28, 2011.
26. F. Zhang, R. Safavi-Naini, and W. Susilo. An efficient signature scheme from bilinear pairings and its applications. Public Key Cryptography–PKC 2004, pages 277–290, 2004.

A Proof of Theorem 2

The contents of this section are closely related to Theorem 6.5 in [16], and the associated proof, which deals with statistical distance between signatures with two related secret keys in the signature scheme we are

based on. The first major difference when dealing with statistical distance comes from the fact that we do not have a single random polynomial $y \leftarrow D_y^{m_u}$ (resp. a random hash function) but a set of ℓ polynomials (resp. a set of hash functions) on each signature. The second major difference is that we have to verify that the adversarial nature of some parameters does not result in an explosion of the statistical distance. In order to make that clear we first introduce a lemma that we will need in the main proof.

In order to get lighter notations we will denote the random variables of the theorem X_0 and X_1 dropping the other parameters. As the output of the ring signature algorithm is a vector of $\ell+1$ coordinates we will note $X_b^{(i)}$ for $i \in [\ell+1]$ and $b \in \{0,1\}$ the random variable associated to the i-th coordinate of X_b. Suppose that none of these variables has $\{\texttt{failed}\}$ as domain. We can then guarantee that the parameters given in the theorem verify the properties tested on step 0 of the algorithm $\mathsf{Ring-sign}$. We will say that these parameters have passed the *sanity check*.

As in [16], we therefore start by noting that the set

$$D_{s,c}(sk_{i_0}, sk_{i_1}) = \{c \in D_{s,c} : \|sk_{i_0}c\|_\infty, \|sk_{i_1}c\|_\infty \leq \sqrt{n}\log n\}$$

has a cardinality negligibly close (in a relative sense) to the one of $D_{s,c}$. As the secret keys have passed the sanity check, they belong to $D_{s,c}$ and therefore, even if they are chosen adversarially, Lemma 1 guarantees that

$$\frac{|D_{s,c}(sk_{i_0}, sk_{i_1})|}{|D_{s,c}|} = 1 - n^{-\omega(1)}. \tag{4}$$

Note that n is also an adversarially chosen parameter but, again, the sanity check ensures that $n \geq k$ and thus that $n^{-\omega(1)}$ is a negligible function. Splitting the statistical distance in two we get

$$\Delta(X_0, X_1) = \frac{1}{2} \sum_{\hat{\alpha}_i \in D_z^{m_u}; i \in [\ell], \beta \notin D_{s,c}(sk_{i_0}, sk_{i_1})} |Pr[X_0 = (\hat{\alpha}_i; i \in [\ell], \beta)] - Pr[X_1 = (\hat{\alpha}_i; i \in [\ell], \beta)]| \tag{5}$$

$$+ \frac{1}{2} \sum_{\hat{\alpha}_i \in D_z^{m_u}; i \in [\ell], \beta \in D_{s,c}(sk_{i_0}, sk_{i_1})} |Pr[X_0 = (\hat{\alpha}_i; i \in [\ell], \beta)] - Pr[X_1 = (\hat{\alpha}_i; i \in [\ell], \beta)]|. \tag{6}$$

In order to prove that the statistical distance is negligible, we will first prove that (5) is negligible and then that (6) is equal to zero. The first assertion is almost trivial, as the last coordinate in the signature comes from a random oracle and thus the probability it does not belong to $D_{s,c}(sk_{i_0}, sk_{i_1})$ is negligible. Noting that this is true for X_0 as for X_1 and that $\sum |Pr[X_0 = (\hat{\alpha}_i; i \in [\ell], \beta)] - Pr[X_1 = (\hat{\alpha}_i; i \in [\ell], \beta)]| \leq \sum Pr[X_0 = (\hat{\alpha}_i; i \in [\ell], \beta)] + \sum Pr[X_0 = (\hat{\alpha}_i; i \in [\ell], \beta)]$ we can prove that

$$(5) \leq 1 - \frac{|D_{s,c}(sk_{i_0}, sk_{i_1})|}{|D_{s,c}|} = n^{-\omega(1)}. \tag{7}$$

More formally, we have

$$(5) \leq \frac{1}{2} \sum_{\hat{\alpha}_i \in D_z^{m_u}; i \in [\ell], \beta \notin D_{s,c}(sk_{i_0}, sk_{i_1})} Pr[X_0 = (\hat{\alpha}_i; i \in [\ell], \beta)] + Pr[X_1 = (\hat{\alpha}_i; i \in [\ell], \beta)]$$

and for any $b \in \{0,1\}$, noting that $\sum_{\forall A} Pr[A \wedge B] = \sum_{\forall A} Pr[A|B]Pr[B] = Pr[B]$, we have

$$\sum_{\hat{\alpha}_i \in D_z^{m_u}; i \in [\ell], \beta \notin D_{s,c}(sk_{i_0}, sk_{i_1})} Pr[X_b = (\hat{\alpha}_i; i \in [\ell], \beta)] = \sum_{\beta \notin D_{s,c}(sk_{i_0}, sk_{i_1})} Pr\left[X_b^{(\ell+1)} = \beta\right].$$

Finally, noting that $X_b^{(\ell+1)}$ is obtained through a call to a random oracle $H(\sum_{i \in [\ell]} h_i(y_i), R, \mu)$, the probability it is equal to a given β is $1/|D_{s,c}|$. It is important to note that this is true, independently of the distribution of the input. Thus, even if the h_i in $H(\sum h_i(y_i), R, r\mu)$ are adversarially chosen (say, $h_i(x) = 0$

for all x and i), the probability given is still valid. Using this probability for all $\beta \notin D_{s,c}(sk_{i_0}, sk_{i_1})$, equation (7) follows immediately.

In order to prove that (6) is equal to zero we will show that each term in the sum is null. As the last coordinate of a signature is generated through a call to a random oracle we have for both random variables the same probability to obtain β. Therefore, we must prove that for each term in (6):

$$Pr\left[(X_0^{(i)}; i \in [\ell]) = (\hat{\alpha}_i; i \in [\ell]) | X_0^{(\ell+1)} = \beta\right] = Pr\left[(X_1^{(i)}; i \in [\ell]) = (\hat{\alpha}_i; i \in [\ell]) | X_1^{(\ell+1)} = \beta\right] \quad (8)$$

We will prove that for all $b \in \{0,1\}$, $Pr[X_b^{(i)} = \hat{\alpha}_i | X_b^{(\ell+1)} = \beta]$ is equal to $1/|D_z^{m_u}|$ if $i \in [\ell] \backslash i_b$ and to $1/|D_y^{m_u}|$ if $i = i_b$. This is enough to prove (8) as the first ℓ coordinates of the random variables are independently chosen in the signature algorithm.

We note $\hat{y}_{b,i}$ the variable $\hat{y}_i$ corresponding to an execution of the signature algorithm in which sk_{i_b} is used. For $i \neq i_b$, we have $X_b^{(i)} = \hat{\alpha}_i$ if $\hat{y}_{b,i} = \hat{\alpha}_i$. As $\hat{y}_{b,i}$ is drawn uniformly at random from $D_z^{m_u}$, and $\hat{\alpha}_i \in D_z^{m_u}$, the probability that both values are equal is $1/|D_z^{m_u}|$. For $i = i_b$, we have $X_b^{(i_b)} = \hat{\alpha}_{i_b}$ if $\hat{y}_{b,i_b} = \hat{\alpha}_{i_b} - sk_{i_b}\beta$. As $\hat{y}_{b,i}$ is drawn uniformly at random from $D_y^{m_u}$, the probability that it is equal to a given value is $1/|D_y^{m_u}|$ if this value is in $D_y^{m_u}$, and 0 if not. By the definition of $D_{s,c}(sk_{i_0}, sk_{i_1})$, to which β belongs, we have $sk_{i_b}\beta \leq \sqrt{n}\log n$ and thus $\hat{\alpha}_{i_b} - sk_{i_b}\beta$ belongs to $D_y^{m_u}$ which completes the proof.

□

B Lemmas

Lemma 1 (Lemma 2.11 in [16] restricted to our setting). *Let a be any polynomial in $D_{s,c}$ and b a polynomial uniformly chosen in $D_{s,c}$. Then*

$$Pr[\|ab\|_\infty \geq \sqrt{n}\log n] \leq 4ne^{\frac{-\log^2 n}{8}}.$$

This lemma will be used to show that for any two secret keys, an overwhelming fraction of the polynomials in $D_{s,c}$ result in a small polynomial when multiplied by any of the two keys. Note that the lemma is valid for *any* polynomial a in $D_{s,c}$ (i.e. even if it is adversarially chosen).

Before proving Lemma 3, which is used in Proposition 2, we recall the following lemma which is adapted from Lemma 3 in [23].

Lemma 2. *Let $f = x^n + 1$, $r \geq 2$ and $n = 2^r$ and p is a prime with $p \equiv 3 \bmod 8$, then there exist f_1, f_2 such that $f = f_1 f_2 \bmod p$ where each f_i is irreducible in $\mathbb{Z}_p[x]$ and can be written $f_i(x) = x^{n/2} + t_i x^{n/4} - 1$ with $t_i \in \mathbb{Z}_p$*

Lemma 3. *Let $D_{s,c}^{\times}$ denote the set of non-invertible polynomials of $D_{s,c}$. We have*

$$Pr_{f \leftarrow D_{s,c}}\left[f \in D_{s,c}^{\times}\right] \leq \frac{2}{3^{n/2}}.$$

Proof (Sketch.). We have $\mathcal{D} = \mathbb{Z}_p[x]/\langle x^n + 1\rangle$ and by Lemma 2 and Table 1 we have $x^n + 1 = f_1 f_2 \mod p$, both factors being of degree $n/2$ and irreducible over $\mathbb{Z}_p[x]$. As these factors are irreducible, we have that the non-invertible polynomials of $\mathcal{D}$ are such that $f = 0 \mod f_1$ or $f = 0 \mod f_2$.

As $D_{s,c} = \{g \in \mathbb{Z}_p[x]/\langle x^n + 1\rangle : \|g\|_\infty \leq 1\}$ we have

$$D_{s,c}^{\times} = \{f \in \mathbb{Z}_p[x]/\langle x^n + 1\rangle : \|f\|_\infty \leq 1 \text{ and } (f = 0 \bmod f_1 \text{ or } f = 0 \bmod f_2)\}.$$

By the union bound, we have

$$Pr_{f \leftarrow D_{s,c}}\left[f \in D_{s,c}^{\times}\right] \leq Pr_{f \leftarrow D_{s,c}}\left[f = 0 \bmod f_1\right] + Pr_{f \leftarrow D_{s,c}}\left[f = 0 \bmod f_1\right].$$

It is easy to see that for each choice of $n/2$ higher order terms of f there is as most one choice on the $n/2$ lower order terms that satisfies $f = 0 \bmod f_1$. The same can be said for $f = 0 \bmod f_2$. Thus the probability of each of the right-hand terms is bounded from above by $1/3^{n/2}$ which immediately proves the result.